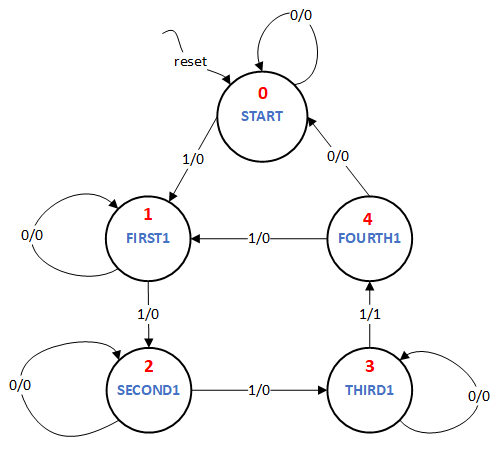
Introduction

In this exercise, you will create a test to check the functionality of a Finite State Machine (FSM) which detects the accumulation of number of 1 at the input. It has the following properties:

* one input (**in**) and one output (**out**)
* output is **1**when the number of input bit equal to 1 is accumulated to **4**.
* the i/o behaviour in the FSM is encoded as follows: **in/out (eg. 1/0 means in = 1'b1 and out = 1'b0).**

The following shows the input/output behavior for the FSM:

**in : 0 0 1 0 1 1 0 1 1 1 0 1 1 ....  
 out : 0 0 0 0 0 0 0 1 0 0 0 0 1 ...**



The diagram above illustrates the transitions of the FSM that should be implemented to achieve this. It shows a Mealy Machine where the output is defined both by the state as well as the input. The machine has five states. The state numbers are marked in red.

Assignment

Write a FSM testbench named **accu\_top\_tb**. It will drive the module named **accu\_top** which takes **input signals: clk, reset, in, next** and produces **output signals: out, state\_display**. Assume all input and output signals to be 1 bit wide except state\_display which is 3 bits wide and little endian.

**Note:**A transition will happen only if **next** is high.

You have write the testbench which tests the given finite state machine for the following input sequence:

**1 => 0  => 0  => 1  => 0 => 1  => 0 => 1  => 1  => 1 =>  1 => 1**

Print the following after each input stimuli:

$display("state %x, out %x", state\_display, out);

Use the following template for writing the testbench:

`timescale 1ns / 1ps  
module accu\_top\_tb();  
  
 <<DECLARE SIGNALS TO DRIVE MODULE accu\_top>>  
     
 accu\_top accu\_top\_inst(  
 .clk(clk),  
 .reset(reset),  
 .next(next),  
 .in(in),  
 .out(out),  
 .state\_display(state\_display));

<<DECLARE CLOCK>>  
     
             
   initial begin  
   $dumpfile("dut.vcd");  
      $dumpvars(0, accu\_top\_inst);

<<INITIALIZE INPUT SIGNALS>>

  <<RESET>>  
   #500  
   $display("state %x, out %x", state\_display, out);  
     
     
   <<STIMULI 1>>  
   #500  
   $display("state %x, out %x", state\_display, out);  
     
   <<STIMULI 2>>  
   #500  
   $display("state %x, out %x", state\_display, out);  
     
   .................. continue for the rest of the stimuli ..............................  
     
   #100  
      $finish;  
   end  
endmodule

**STRICT NOTE:** Just change the portions marked in red in the tesbench template.