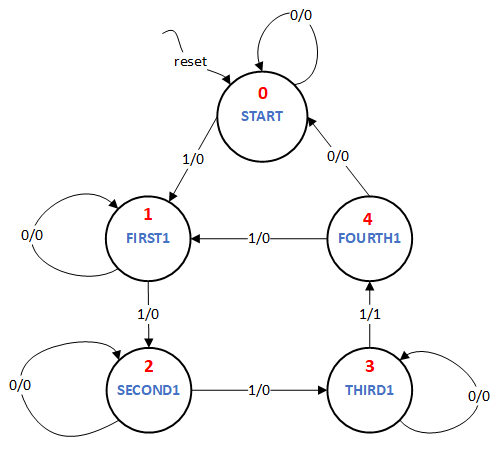
Introduction

In this exercise, you will create a verilog implementation 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 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. The state\_display must output the number of the current state using the same numbers as the states in the FSM diagram above (shown in red). The **reset**input always makes a transition to the **START** state. The **reset**output is a **synchronous** signal (i.e. it goes high only at posedge of clock).

**Note1:**A transition will happen only if **next** is high. If **next** is low, the **state does not change**. Make sure that you only make one transition when the **next** is set from 0 to 1. For doing this, check whether the current value is different from its previous value. Do not use: always @ (posedge **next**).

**Note2:**Make sure the output signal **out** only changes when you get the desired condition and the **next** signal is set from 0 to 1.