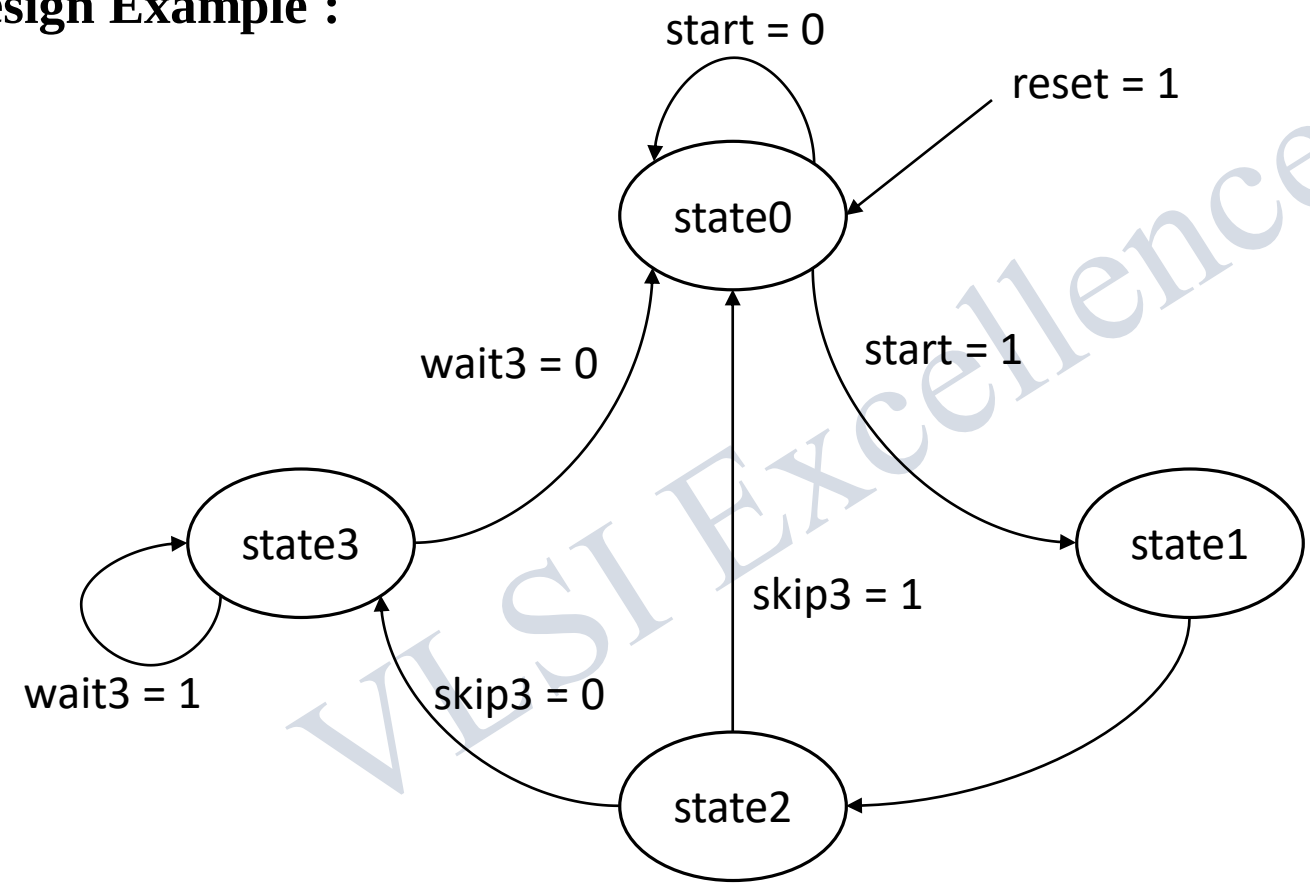
Module #17 : Verilog Test Bench Design

17.1: Verilog Test Bench Design:

- A Test bench supplies the signals and dumps the outputs to simulate a Verilog design (module(s)).
- It invokes the design under test, generates the simulation input vectors, and implements the system tasks to view/format the results of the simulation.
- It is never synthesized so it can use all Verilog commands

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.2: FSM Design Example :



state0: Z = 3'b000;
state1: Z = 3'b101;
state2: Z = 3'b111;
state3: Z = 3'b001;

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.3: FSM Design Example :

Example 17.1:

```
module my_fsm (clk, rst, start, skip3, wait3, Z);
input clk, rst, start, skip3, wait3;
output [2:0] Z; // Z is declared reg so that it can
reg [2:0] Z;   // be assigned in an always block.
parameter state0=0, state1=1, state2=2, state3=3;
reg [1:0] state, nxt_st;
always @ (state or start or skip3 or wait3)
begin : next_state_logic //Name of always procedure.
case (state)
state0: begin
if (start) nxt_st = state1;
else nxt_st = state0;
end
state1: begin
nxt_st = state2;
end
state2: begin
if (skip3) nxt_st = state0;
else nxt_st = state3;
end
state3: begin
if (wait3) nxt_st = state3;
else nxt_st = state0;
end
default: nxt_st = state0;
endcase // default is optional since all 4 cases are
end       // covered specifically. Good practice
          // says uses it
```

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.3: FSM Design Example :

```
always @(posedge clk or posedge rst)
begin : register_generation
if (rst) state = state0;
else state = nxt_st;
end

always @(state) begin : output_logic
case (state)
state0: Z = 3'b000;
state1: Z = 3'b101;
state2: Z = 3'b111;
state3: Z = 3'b001;
default: Z = 3'b000;  // default avoids latches
endcase
end
endmodule
```

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.4: Test Bench Design :

Example 17.2:

```
‘timescale 1 ns /100 ps // time unit = 1ns; precision = 1/10 ns;
 module my_fsm_tb; // Test Bench of FSM Design of Example 17.1
 /* ports of the design under test are variables in the test bench */
reg clk, rst, start, skip3, wait3;
wire Z;
 /***** DESIGN TO SIMULATE (my_fsm) INSTANTIATION *****/
my_fsm dut1 (clk, rst, start, skip3, wait3, Z);
/***** RESET AND CLOCK SECTION *****/
initial begin
clk = 0; rst=0;
 #1 rst = 1; // The delay gives rst a posedge for sure.
#200 rst = 0; // Deactivate reset after two clock cycles +1 ns*/
end
always #50 clk = ~clk; // 10 MHz clock (50*1 ns*2) with 50% duty-cycle
```

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.4: Test Bench Design :

Continued . . .

```
/**** SPECIFY THE INPUT WAVEFORMS skip3 & wait3 ****/
initial begin
skip3 = 0; wait3 = 0; // at time 0, wait3=0, skip3=0
#1; // Delay to keep inputs from changing on clock edge.
#600 skip3 = 1; // at time 601, wait3=0, skip3=1
#400 wait3 = 1; // at time 1001, wait3=1, skip3=0
 skip3= 0;
#400 skip3 = 1; // at time 1401, wait3=1, skip3=1
wait(Z) skip3 = 0; // Wait until Z=1, then make skip3 zero.
 wait3 = $random; //Generate a random number, transfer lsb into wait3
$finish; // stop simulation. Without this it will not stop.
end
endmodule
```

Module #17 : Verilog HDL Finite State Machines (FSMs)

17.5: Synchronous Test Bench :

- In synchronous designs, one changes the data during certain clock cycles.
- In the previous test bench one had to keep counting delays to be sure the data came in the right cycle.
- With a synchronous test bench the input data is stored in a vector or array and one part injected in each clock cycle.
- Synchronous test benches are essential for cycle based simulators, which do not use any delays smaller than a clock cycle

Module #17 : Verilog HDL Finite State Machines (FSMs)

Example 17.3:

```
// Synchronous test bench
module SynchTstBch:
reg [8:1] data;
reg x, clk;
wire y;
integer I;
topmod top1(clk, x, y); //DUT Instantiation
initial begin
 data[8:1]=8'b1010_1101; // Underscore spaces bits.
 I=1;
 x=0;
 clk=0;
 forever #5 clk= ~clk;
 // Any statements placed after forever will never be reached!
 end
```

Module #17 : Verilog HDL Finite State Machines (FSMs)

Example 17.3:

```
/*** Send in a new value of x every 3rd clock cycle***/
always
 begin: data_in_proc
 @(posedge clk)
 if (I= =9) $finish; // End of simulation
 @(posedge clk) // Wait here for the 2nd clock edge.
 @(posedge clk) // After the 3rd edge, execute begin ...
 begin
 #1; // Keeps data from changing on clock edge.
 x <= data[I];
// Wait for y to respond (change);
 @(negedge clk)
// Check the result; print a warning if it is not as expected.
 if (x != y) $display ("x != y, x=%b, y=%b," x , y);
 I<=I+1;
 end
 end // data_in_proc

endmodule
```