CSE 20221: Logic Design

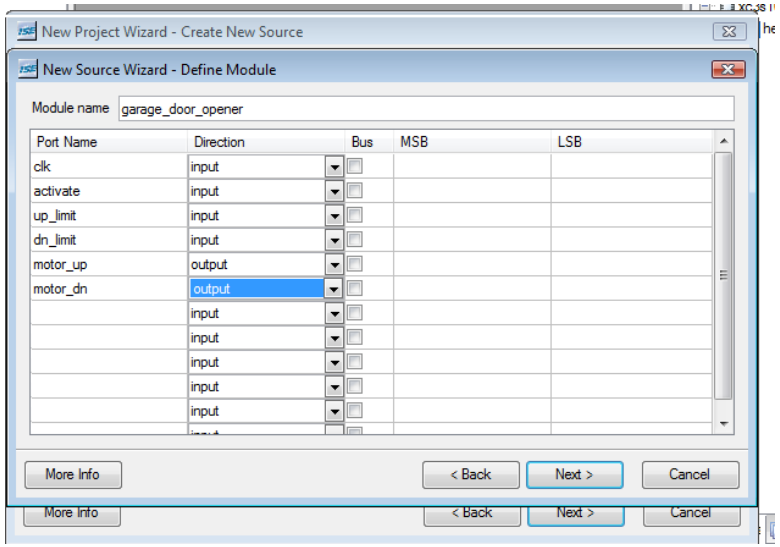
Verilog FSM Design Example

Automatic Garage Door Opener
&
Timers

Inputs / Outputs

NAME	TYPE	FUNCTION
activate (A)	input	starts the door to go up or down or stops the motion
Up_limit (UPL)	input	indicates maximum upward travel
Dn_limit (DNL)	input	indicates maximum downward travel
Motor_up (MU)	output	Causes motor to run in direction to raise the door
Motor_dn (MD)	output	Causes motor to run in direction to lower door

Define the module interface

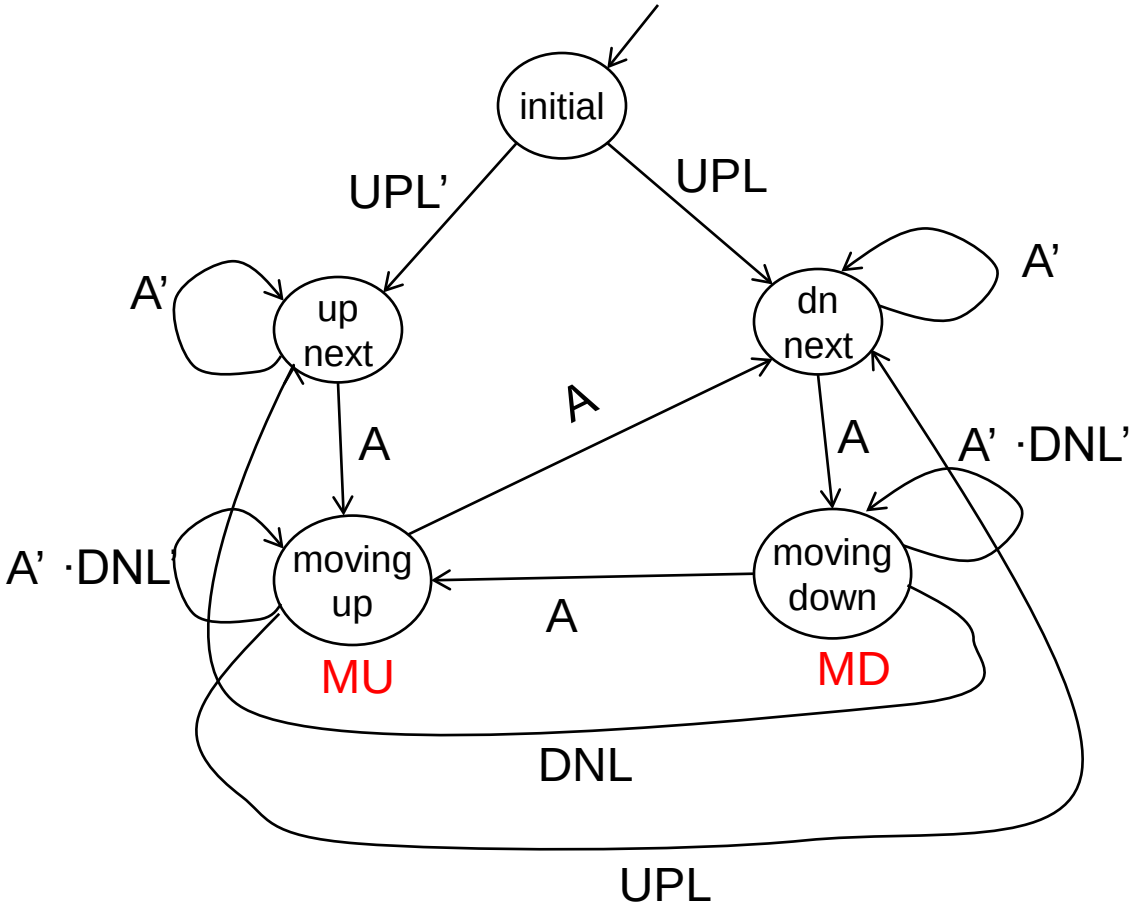


Inputs / Outputs

NAME	TYPE	FUNCTION
activate (A)	input	starts the door to go up or down or stops the motion
Up_limit (UPL)	input	indicates maximum upward travel
Dn_limit (DNL)	input	indicates maximum downward travel
Motor_up (MU)	output	Causes motor to run in direction to raise the door
Motor_dn (MD)	output	Causes motor to run in direction to lower door
reset	input	Force the controller to enter into the initial state

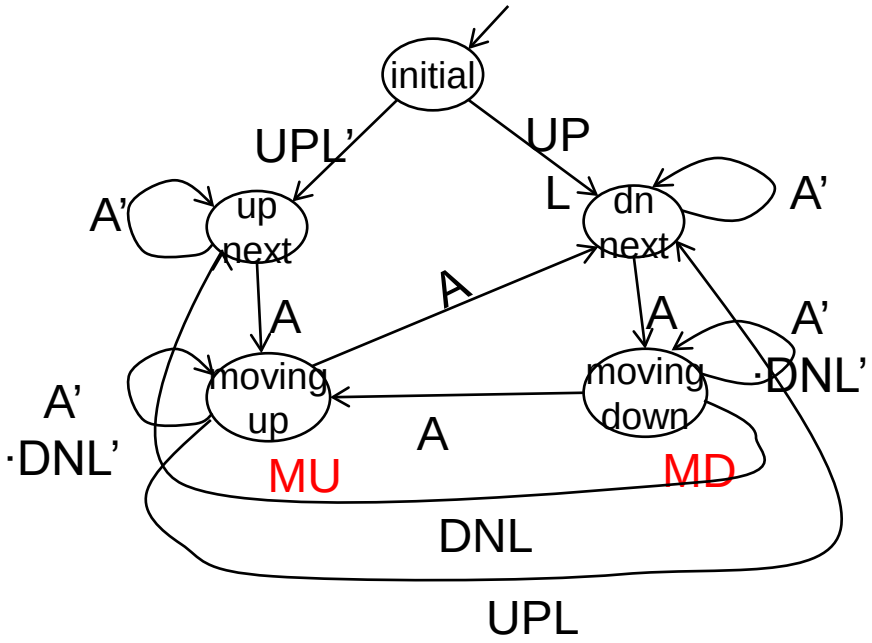
```
21  module DoorOpener(clk,activate,up_limit,dn_limit,reset,motor_up,motor_dn);
22      input clk,activate,up_limit,dn_limit,reset;
23      output motor_up,motor_dn;
24      reg motor_up,motor_dn;
25
```

State Diagram



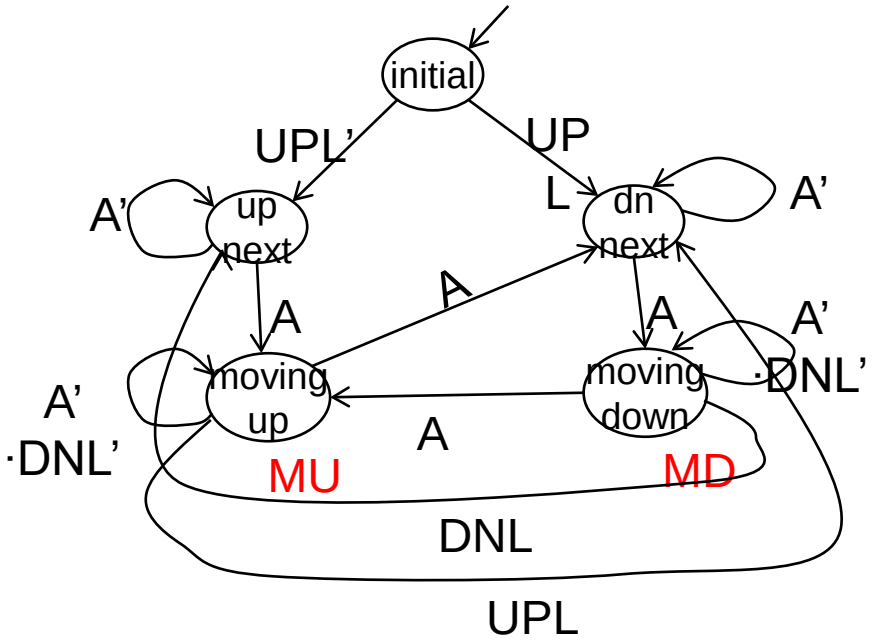
Make the State Assignments

```
25
26    reg [2:0] state, nextState;
27    parameter _initial = 3'b000, upNext = 3'b001, dnNext = 3'b010,
28              movingUp = 3'b011, movingDn = 3'b100;
29
```



Setup clear and state register

```
30    always @(posedge clk)
31     begin
32       if (reset) begin
33          state <= _initial;
34          motor_up = 0; motor_dn = 0;
35       end
36       else state <= nextState;
37     end
38
```



Describe the Behavior

```
39    always @(state,activate,up_limit,dn_limit)
40    begin
41          case (state)
42        _initial: begin
43           motor_up = 0; motor_dn = 0;
44           if (up_limit == 1) nextState <= dnNext;
45           else nextState <= upNext;
46          end
47        upNext: begin
48           motor_up = 0; motor_dn = 0;
49           if (activate)nextState <= movingUp;
50           else nextState <= upNext;
51          end
52        dnNext: begin
53           motor_up = 0; motor_dn = 0;
54           if (activate) nextState <= movingDn;
55           else nextState <= dnNext;
56          end
57        movingDn: begin
58           motor_up = 0; motor_dn = 1;
59           if (activate) nextState <= movingUp;
60            else if (dn_limit) nextState <= dnNext;
61            else nextState <= movingDn;
62          end
```

Behavior Continued

```
52          dnNext: begin
53             motor_up = 0; motor_dn = 0;
54             if (activate) nextState <= movingDn;
55             else nextState <= dnNext;
56           end
57          movingDn: begin
58             motor_up = 0; motor_dn = 1;
59             if (activate) nextState <= movingUp;
60              else if (dn_limit) nextState <= dnNext;
61              else nextState <= movingDn;
62           end
63          movingUp: begin
64             motor_up = 1; motor_dn = 0;
65             if (activate | up_limit) nextState <= dnNext;
66              else nextState <= movingUp;
67           end
68          default : begin
69              nextState <= _initial;
70           end
71        endcase
72     end
73  endmodule
```

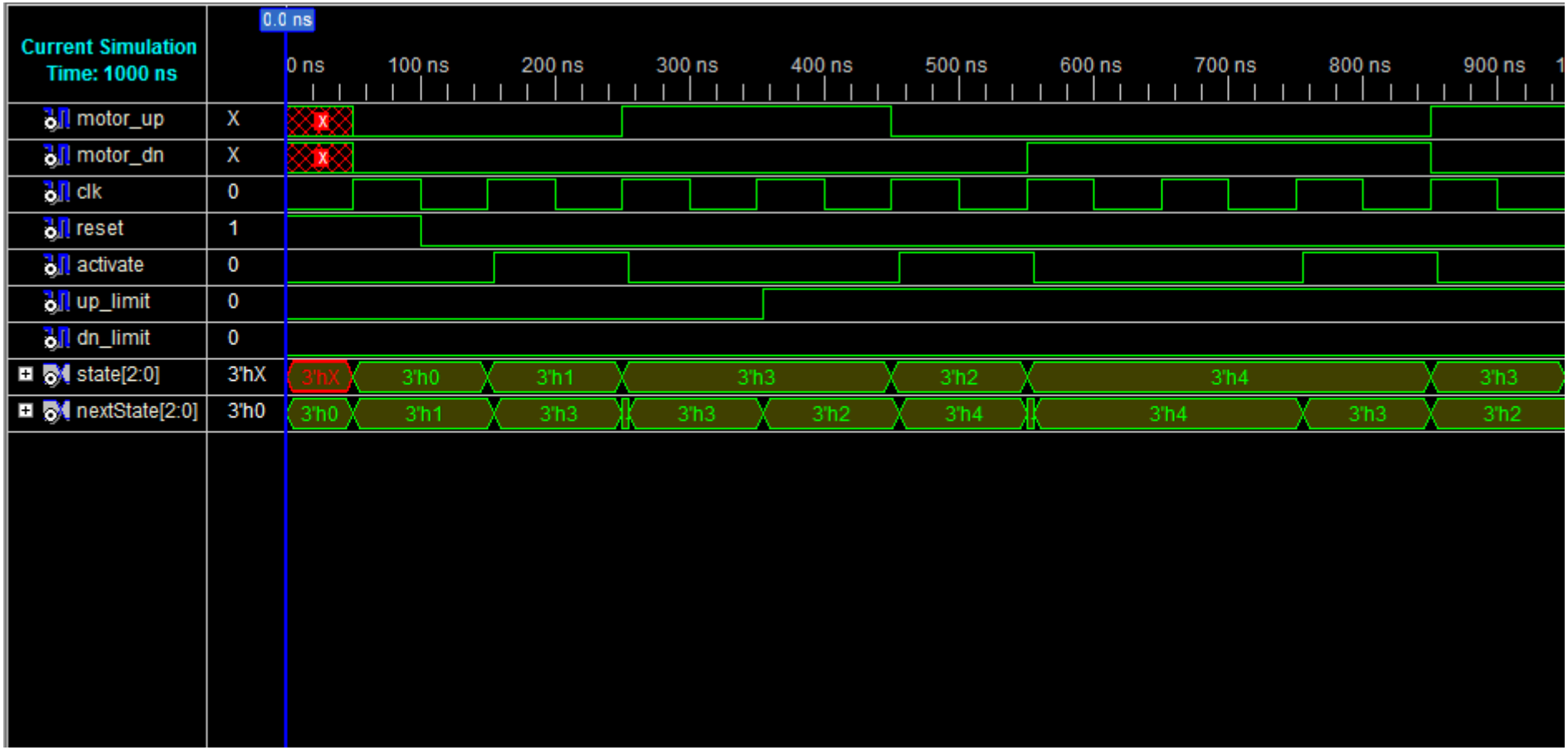
Xilinx Verilog Test Fixture

```
49      initial begin
50          // Initialize Inputs
51          clk = 0;
52          activate = 0;
53          up_limit = 0;
54          dn_limit = 0;
55          reset = 1 ;
56
57          // Wait 100 ns for global reset
58          #100;
59          #100;
60          reset = 0;
61          @(posedge clk);
62          #5 activate = 1;
63          @(posedge clk);
64          #5 activate = 0;
65          @(posedge clk);
66          #5 up_limit = 1;
67          @(posedge clk);
68          #5 activate = 1;
69          @(posedge clk);
70          #5 activate = 0;
71          @(posedge clk);
72          @(posedge clk);
73          #5 activate = 1;
74          @(posedge clk);
75          #5 activate = 0;
76          @(posedge clk);
77          #5 activate = 1;
78          @(posedge clk);
79          #5 activate = 0;
80          @(posedge clk);
81          #5 dn_limit = 1;
82          @(posedge clk);
83          @(posedge clk);
84          #5 activate = 1;
85          @(posedge clk);
86          #5 activate = 0;
87          @(posedge clk);
88          @(posedge clk);
89          #5 activate = 1;
90          @(posedge clk);
91          #5 activate = 0;
92          @(posedge clk);
93          #5 up_limit = 1;
94          @(posedge clk);
95          @(posedge clk);
96
97      end
```

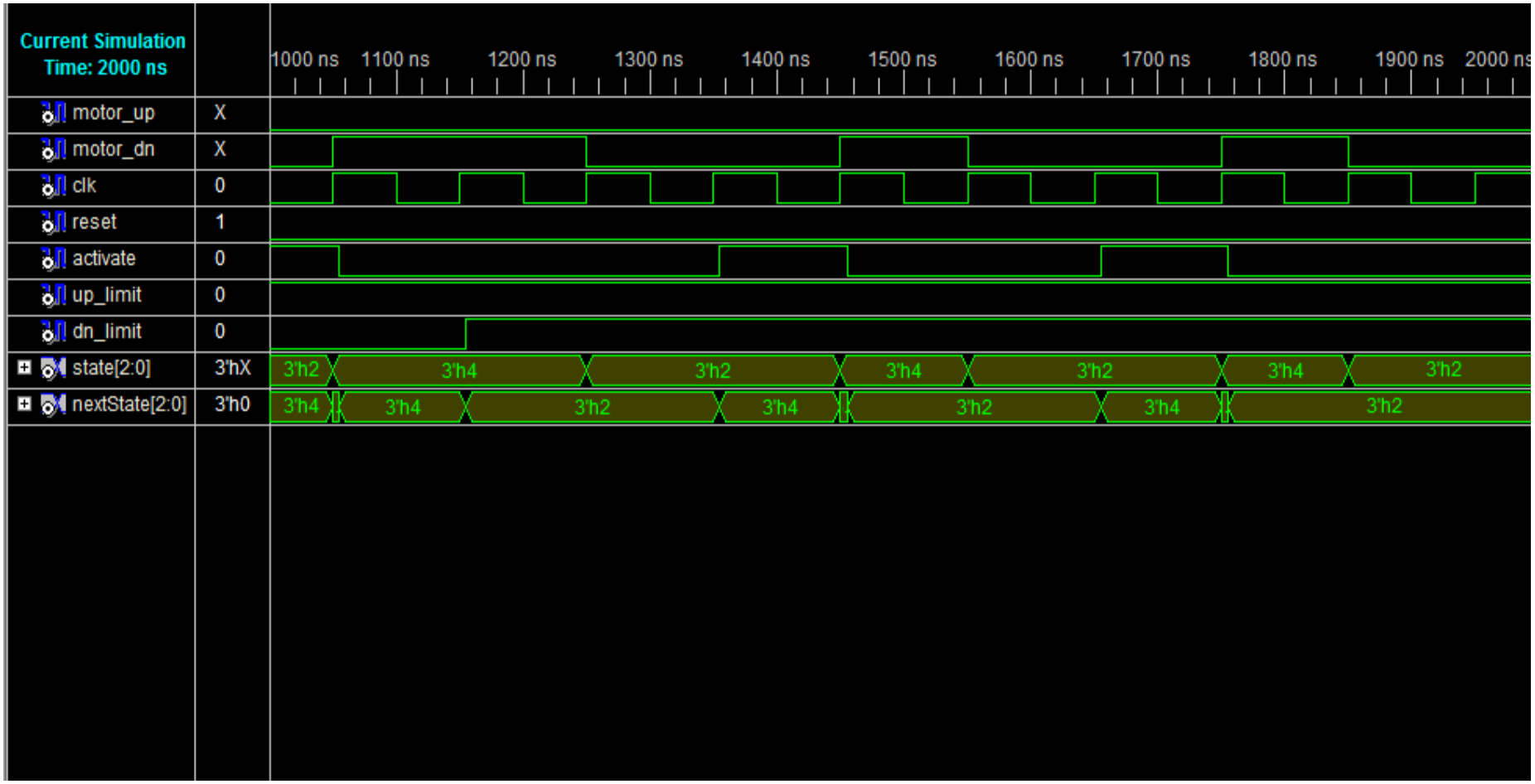
Test Bench for Clock

```
 98        always begin
 99        clk <= 0;
100        #50;
101          clk <= 1;
102        #50;
103        end
104  endmodule
```

Simulation Results



Simulation Results Continued



Xilinx Simulation Tips

- Provide a means (reset signal) to initialize all internal variables, otherwise don't care conditions occur throughout the simulation.

```
always @(posedge clk)
begin
if (reset) begin
    countValue = 0;
    clkDivOut <= 0;
```

- In the test bench code, first initialize the circuit under test.
- Select the *sim instance tab in the source window to bring up internal signals to be placed in the simulator waveform.*

CSE 20221: Logic Design

Timers, Frequency Divider Examples

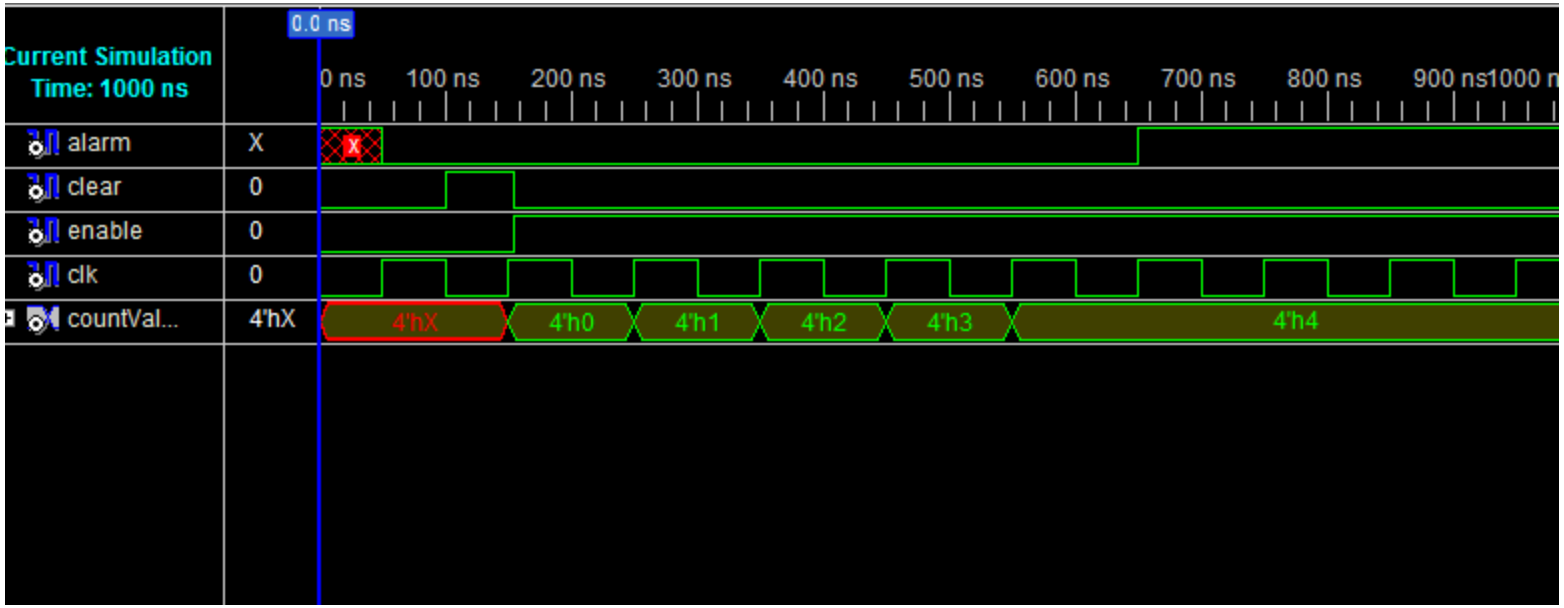
Timers

- Timer
  - time events
  - divide clock frequency
  - provide delay
- In each case the basic idea is to count clock pulses

Verilog Code for Timer

```
21  module timer(clear,enable,clk,alarm);
22      input clear,enable,clk;
23      output alarm;
24      reg alarm;
25      reg [3:0] countValue;
26      parameter terminalCount = 5;
27
28     always @(posedge clk)
29       begin
30        alarm <=0;
31        if (clear) countValue <= 0;
32        else begin
33           if (enable) begin
34              if (countValue == terminalCount -1) alarm <= 1;
35              else countValue <= countValue + 1;
36           end
37        end
38       end
39
40  endmodule
41
```

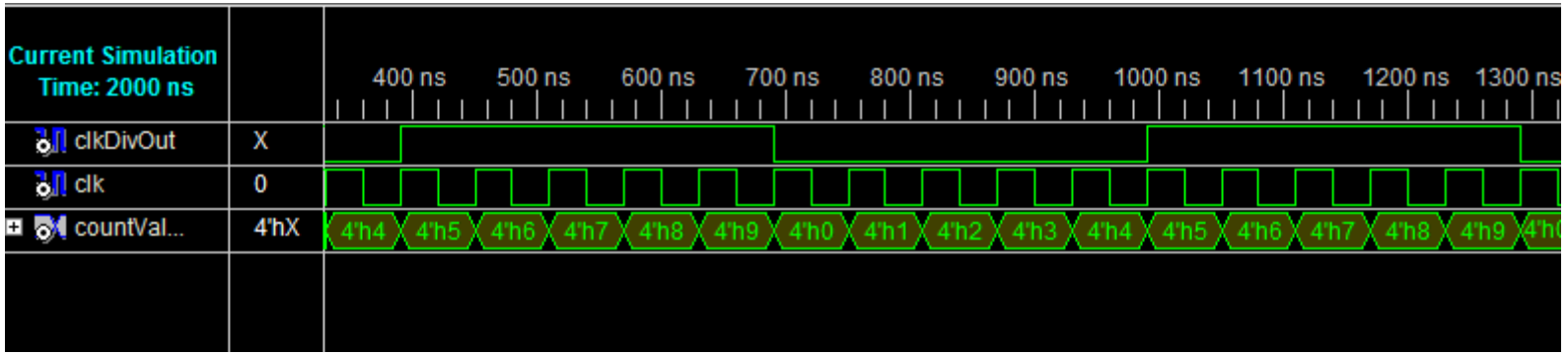
Timer Simulation



Verilog Code for Frequency Divider

```
21  module clockDivider(clk,reset,clkDivOut);
22      input clk,reset;
23      output clkDivOut;
24      reg clkDivOut;
25      parameter period = 10;
26      parameter halfPeriod = period / 2;
27      reg [3:0] countValue;
28
29      always @(posedge clk)
30      begin
31       if (reset) begin
32         countValue = 0;
33         clkDivOut <= 0;
34       end
35        else begin
36         if (countValue == period -1) begin
37           countValue = 0;
38           clkDivOut <= 0;
39           end
40         else  countValue = countValue + 1;
41        if (countValue == halfPeriod) clkDivOut <= 1;
42       end
43      end
44  endmodule
```

Frequency Divider Simulation



Design of a Derived Clock

- Design a 1 millisecond clock that is derived from a 50 MHz system clock.
- Design approach
  - Frequency divider
  - Divide by 50,000
- Determining size (N) of counter
  - given division factor, DF
  - N = roundUp(ln DF / ln 2) -1
  - Parameter [N:0] countValue;

Verilog Description

```
21  module millisecClk(clk,reset,clkOut);
22      input clk,reset;
23      output clkOut;
24      reg clkOut;
25      parameter period = 50000; //nanoseconds
26      parameter N = 15; // bits - 1 for countValue
27      parameter halfPeriod = period / 2;
28      reg [N:0] countValue;
29
30      always @(posedge clk)
31      begin
32       if (reset) begin
33         countValue = 0;
34         clkOut <= 0;
35       end
36        else begin
37         if (countValue == period -1) begin
38           countValue = 0;
39           clkOut <= 0;
40           end
41         else  countValue = countValue + 1;
42        if (countValue == halfPeriod) clkOut <= 1;
43       end
44      end
45  endmodule
```

Test Fixture

```
33
34      // Instantiate the Unit Under Test (UUT)
35      millisecClk uut (
36          .clk(clk),
37          .reset(reset),
38          .clkOut(clkOut)
39      );
40
41      initial begin
42          // Initialize Inputs
43          clk = 0;
44          reset = 1;
45
46          @ (posedge clk)
47          #5 reset = 0;
48      end
49      always begin
50        clk = 1;
51        #10;// period = 20 ns
52        clk = 0;
53        #10;
54      end
55
56  endmodule
```

Similation Results