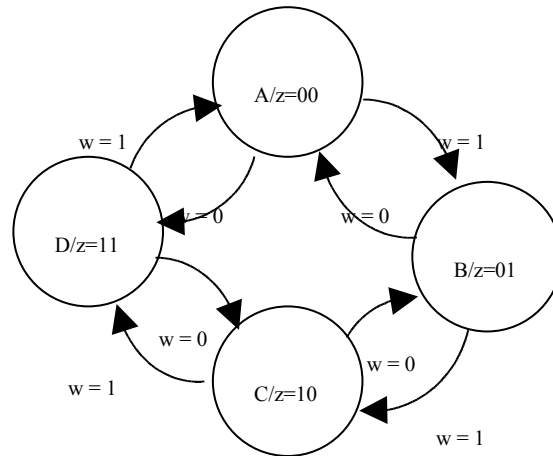


## EE 231L

### Using Verilog to Design State Machines

Finite state machine is another name for sequential circuits. A two-bit up-down counter can be described as a state machine with one input and two outputs:



The state table for the sequential circuit is

Present y	Next State Y		Output z
	w = 0	w = 1	
A	D	B	00
B	A	C	01
C	B	D	10
D	C	A	11

A state-assigned table for the circuit looks like this

Present y	Next State Y		Output z
	w = 0	w = 1	
00	11	01	00
01	00	10	01
10	01	11	10
11	10	00	11

There are a few ways to design state machines using Verilog. Here is one design for the two-bit up-down counter:

```

module Altera_State_Verilog1 (Clock, w, y);
  input Clock, w;
  output reg [2:1] y;
  reg [2:1] Y;
  parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

  always @(w, y)
    case (y)
      A: if (w) Y = B;
         else Y = D;
      B: if (w) Y = C;
         else Y = A;
      C: if (w) Y = D;
         else Y = B;
      D: if (w) Y = A;
         else Y = C;
      default: Y = 2'bxx;
    endcase

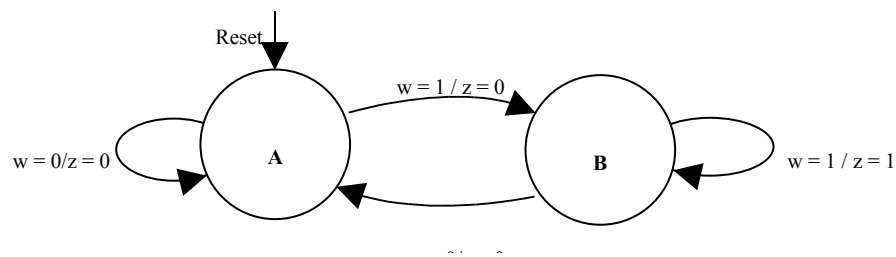
  always @(posedge Clock)
    y <= Y;

endmodule

```

The two-bit up-down counter is a Moore machine — i.e., the outputs of the machine depend only on the current state, and not on the current input. You can design a Moore machine by specifying a bit pattern associated with each state. The  $y[2:1]$  outputs are directly associated with bits of the state machine. This means that the  $y[2:1]$  ( $z$ ) outputs will be the outputs of flip-flops, and will not change value until the machine changes states.

You can use Verilog to design state machines with asynchronous outputs, also called Mealy machines. Here is an example from your textbook:



The state table for this sequential circuits is

Present	Next State	Output $z$
---------	------------	------------

State	w = 0	w = 1	w = 0	w = 1
A	A	B	0	0
B	A	B	0	1

And the state-assigned table is as follows

Present State	Next State		Output z	
	w = 0	w = 1	w = 0	w = 1
Y	Y	Y	z	z
A	A	B	0	0
B	A	B	0	1

Here is a Verilog file to implement the design. This example shows how to reset a state machine. When reset goes high, the machine will be reset to the first state in the state machine list; in this case, that will be state A. The reset is done using the asynchronous CLRN (active-low clear)/RESETN (active-low reset) and PRN (active-low preset) inputs to D flip-flops, so the reset is done as soon as reset goes high; it is not necessary to wait for a clock edge.

When in state B, the output will be 0 when the input is 0, and the output will be 1 when the input is 1. The output will change multiple times while in state B if the input changes multiple times. For a Moore machine, the output changes only when the machine switches from one state to another.

```
module Altera_State_Verilog2 (Clock, w, Resetn, z);
  input Clock, w, Resetn;
  output reg z;
  reg y, Y;
  parameter A = 1'b0, B = 1'b1;

  always @(w, y)
    case (y)
      A: if (w == 0)
          begin
            Y = A; z = 0;
          end
        else
          begin
            Y = B; z = 0;
          end
        end
      B: if (w == 0)
          begin
            Y = A; z = 0;
          end
        else
          begin
            Y = B; z = 1;
          end
        end
    endcase

  always @(posedge Clock , negedge Resetn)
    if (Resetn == 0) y <= A;
    else y <= Y;

endmodule
```