Lecture 11
Finite State Machine with Verilog

Outline

- Verilog for Finite State Machine (***)
- Textbook chapters: Chapter 9.6 and 12

Let's go deeper into a FSM Example

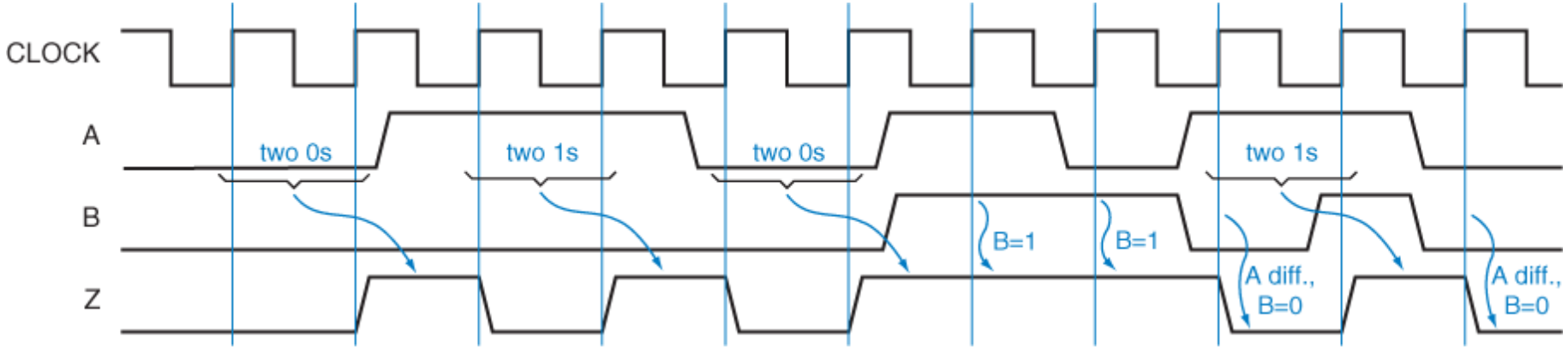
Design a state machine with two inputs, A and B, and a single output Z that is 1 if:

- A had the same value at each of the two previous clock ticks, *or*
- B has been 1 since the last time that the first condition was true.

Otherwise, the output should be 0.

- Based on Textbook example in 9.3

Step 1 Draw Timing Diagram



Design a state machine with two inputs, A and B, and a single output Z that is 1 if:

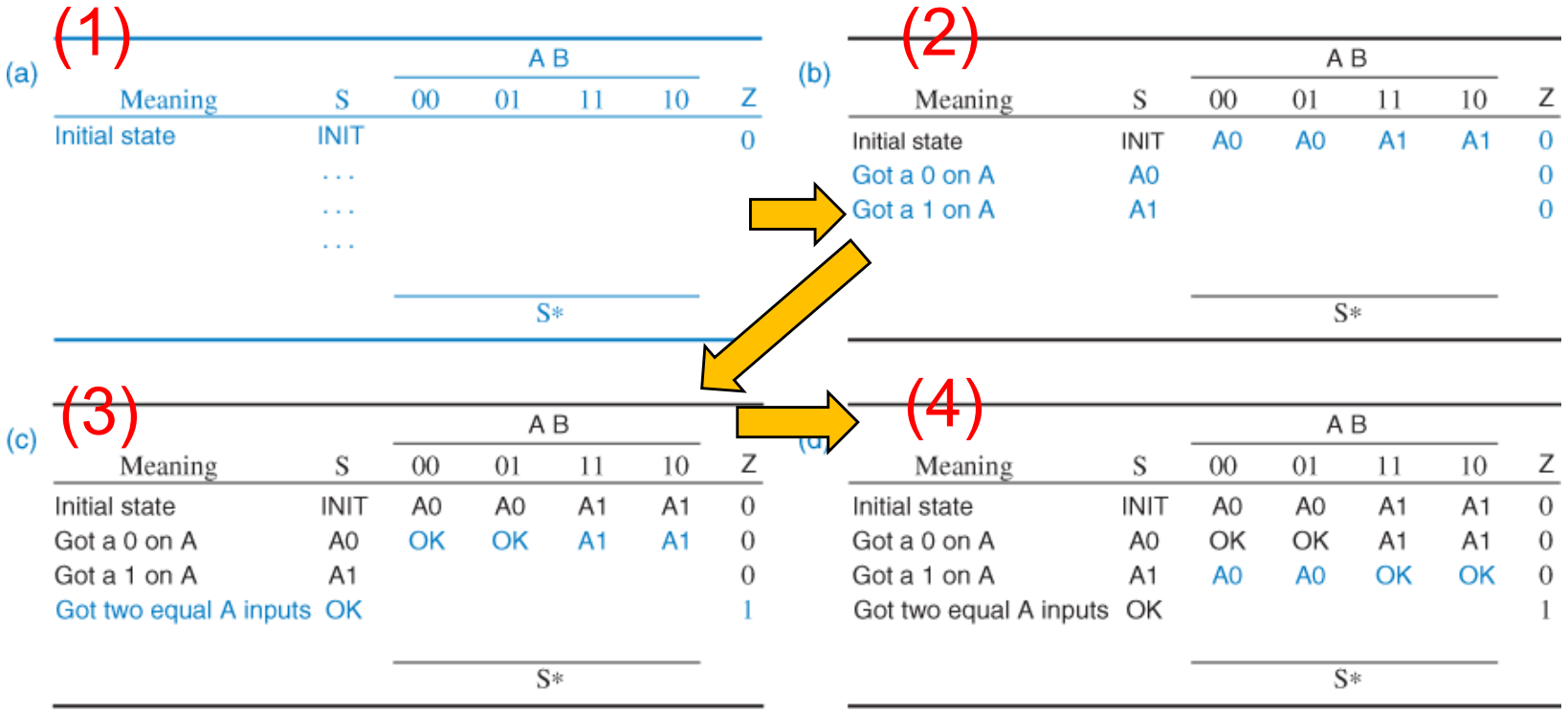
– A had the same value at each of the two previous clock ticks, *or*

– B has been 1 since the last time that the first condition was true.

Otherwise, the output should be 0.

- Timing diagram may not be logically complete; But it helps understand the conditions for state transitions

Step 2 Draw State Table



- This is designer's role. Computer cannot help you at this step.

Step 2 Draw State Table

(a) (1)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK	OK	A1	A1	0
Got a 1 on A	A1	A0	A0	OK	OK	0
Got two equal A inputs	OK	?	OK	OK	?	1
		S*				

(b) (2)

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0					1
Two equal, A=1 last	OK1					1
		S*				

(c) (3)

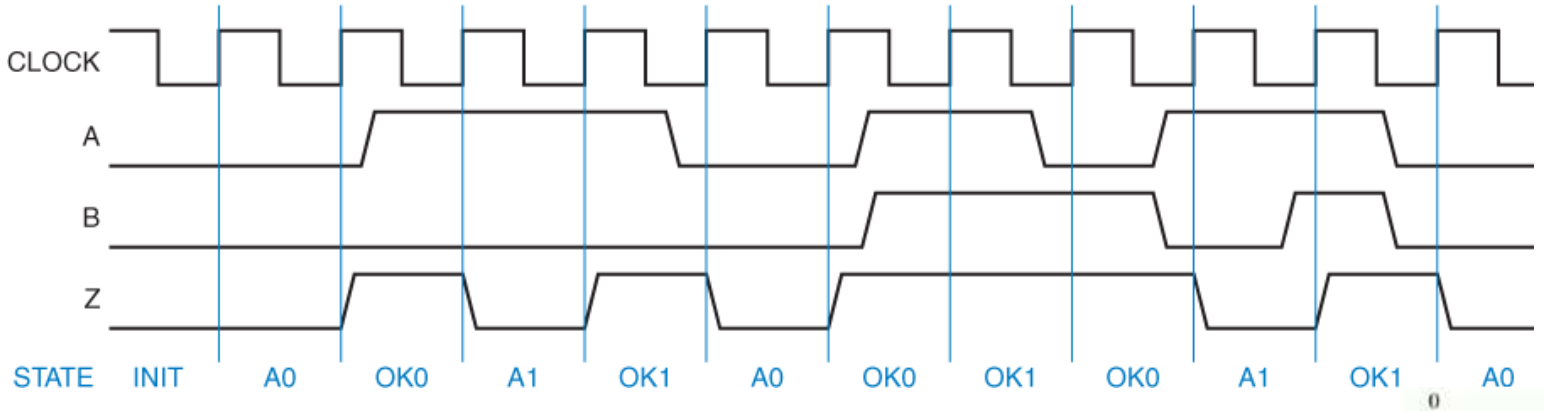
Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1					1
		S*				

(4)

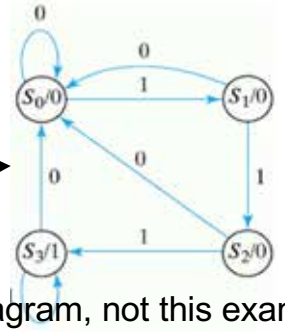
Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				

Step 2 Draw State Table

Meaning	S	A B = 00	A B = 01	A B = 11	A B = 10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				



- State minimization may happen at this step
- State diagram can be also created
  - State diagram is more friendly for human to read

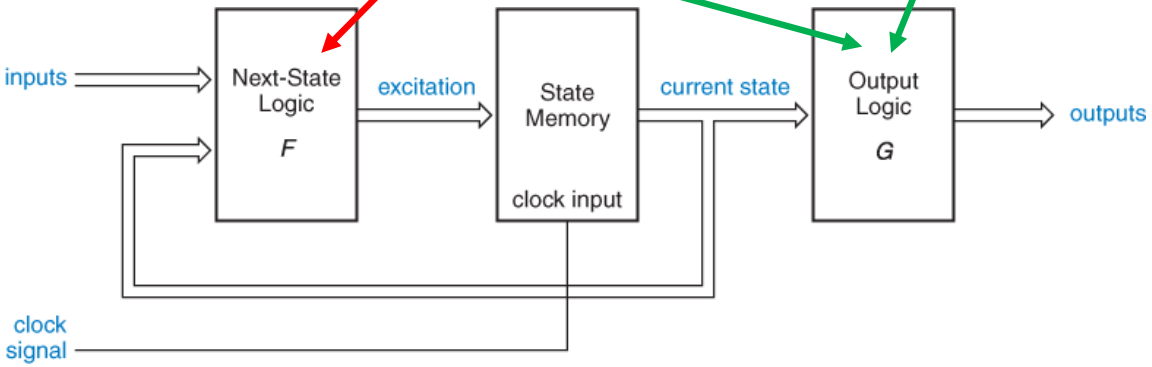


(Example of state diagram, not this example)

Step 3 Map State Table to FSM Structure

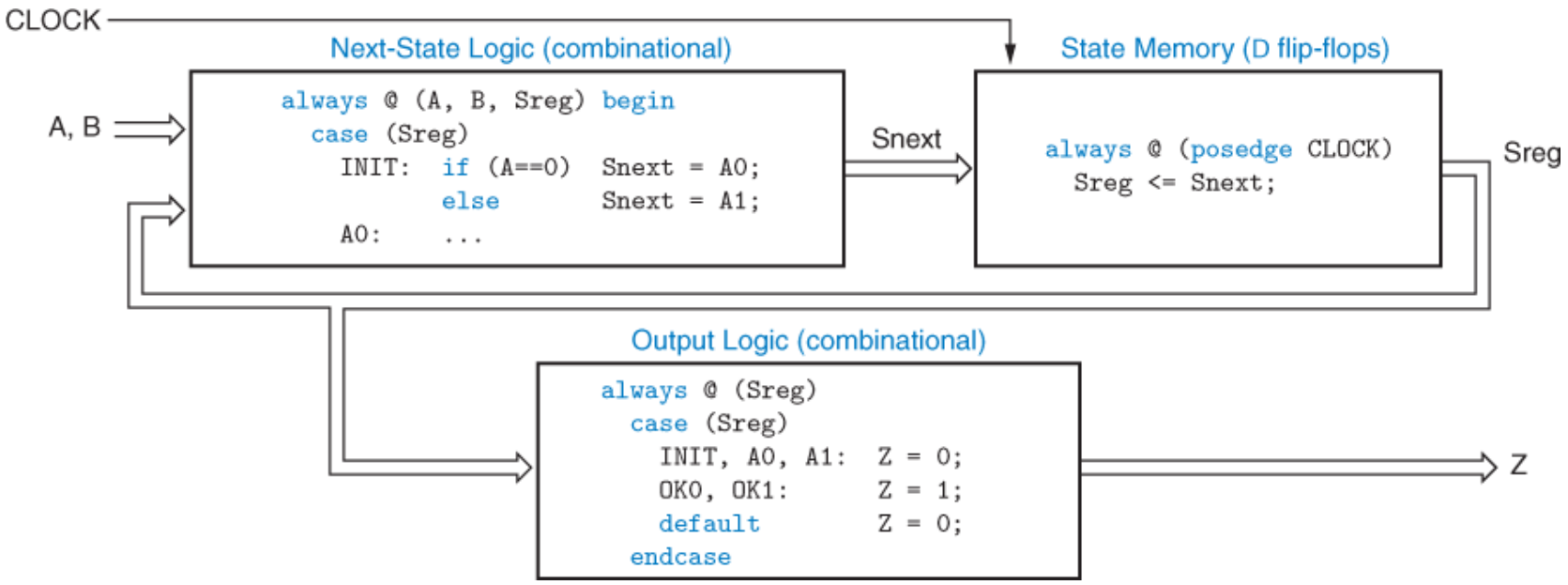
Moore State Machine is used in here

Meaning	S	A B 00	01	11	10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				



- In addition, assign states to state-variables
  E.g. "parameter [2:0] INIT = 3'b000;"

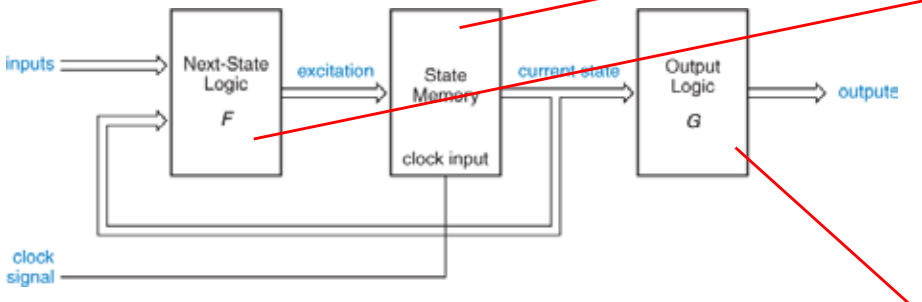
Step 4 Convert into RTL



- Only the "state memory" needs to be sequential circuits
- Can be simplified into single "always" loop

Step 4 Convert into RTL

Should have RSTB or INIT state defined in the Verilog (missing RSTB to start from INIT in this example)



Meaning	S	A B = 00	A B = 01	A B = 11	A B = 10	Z
Initial state	INIT	A0	A0	A1	A1	0
Got a 0 on A	A0	OK0	OK0	A1	A1	0
Got a 1 on A	A1	A0	A0	OK1	OK1	0
Two equal, A=0 last	OK0	OK0	OK0	OK1	A1	1
Two equal, A=1 last	OK1	A0	OK0	OK1	OK1	1
		S*				

```
module VrSMex( CLOCK, A, B, Z );
  input CLOCK, A, B;
  output reg Z;
  reg [2:0] Sreg, Snext;          // State register and next state
  parameter [2:0] INIT = 3'b000,  // Define the states
                  A0   = 3'b001,
                  A1   = 3'b010,
                  OK0  = 3'b011,
                  OK1  = 3'b100;
  always @ (posedge CLOCK)                // Create the state memory
    Sreg <= Snext;
  always @ (A, B, Sreg) begin  // Next-state logic
    case (Sreg)
      INIT:   if (A==0)  Snext = A0;
              else       Snext = A1;
      A0:     if (A==0)  Snext = OK0;
              else       Snext = A1;
      A1:     if (A==0)  Snext = A0;
              else       Snext = OK1;
      OK0:    if (A==0)  Snext = OK0;
              else if ((A==1) && (B==0)) Snext = A1;
              else                       Snext = OK1;
      OK1:    if ((A==0) && (B==0))      Snext = A0;
              else if ((A==0) && (B==1)) Snext = OK0;
              else                       Snext = OK1;
      default Snext = INIT;
    endcase
  end
  always @ (Sreg)       // Output logic
    case (Sreg)
      INIT, A0, A1: Z = 0;
      OK0, OK1:     Z = 1;
      default       Z = 0;
    endcase
endmodule
```

Simplifying the RTL

```
assign Z = (Sreg==OK0) || (Sreg==OK1);
always @ (posedge CLOCK) // State memory and next-state logic
        case (Sreg)
                INIT: if (A==0) Sreg <= A0;
                      else Sreg <= A1;
                A0: if (A==0) Sreg <= OK0;
                    else Sreg <= A1;
                A1: if (A==0) Sreg <= A0;
                    else Sreg <= OK1;
                OK0: if (A==0) Sreg <= OK0;
                      else if ((A==1) && (B==0))      Sreg <= A1;
                      else                            Sreg <= OK1;
                OK1: if ((A==0) && (B==0))            Sreg <= A0;
                           else if ((A==0) && (B==1)) Sreg <= OK0;
                           else                       Sreg <= OK1;
                default Sreg <= INIT;
        endcase
```

"Output Logic"

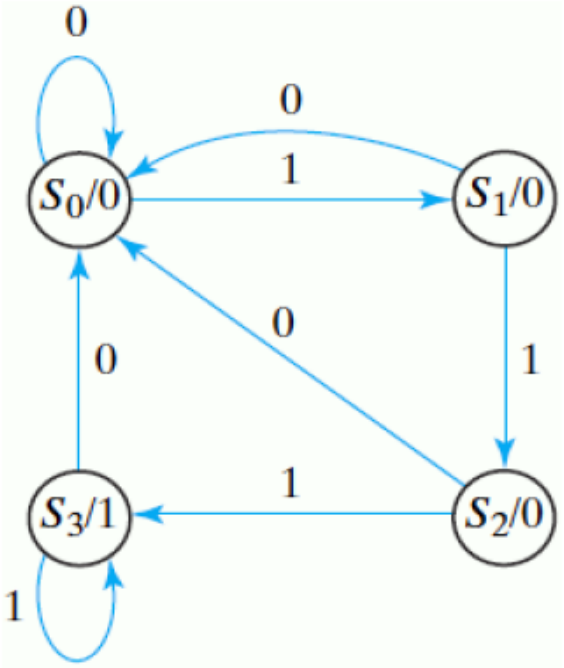
Combining "Next State Logic" and "State Memory"

- Can be simplified into a single "always" and "assign" statement

Another Simple Verilog Example

Detecting three-1s sequence

```
module seq3_detect_moore(x,clk, y);
// Moore machine for a three-1s sequence detection
     input x, clk;
     output y;
     reg [1:0] state;
     parameter S0=2'b00, S1=2'b01, S2=2'b10,
S3=2'b11;
// Define the sequential block
     always @(posedge clk)
          case (state)
                    S0: if (x) state <= S1;
                              else     state <= S0;
                    S1: if (x) state <= S2;
                              else     state <= S0;
                    S2: if (x) state <= S3;
                              else     state <= S0;
                    S3: if (x) state <= S3;
                              else     state <= S0;
          endcase
// Define output during S3
     assign y = (state == S3);
endmodule
```



Should have RSTB or INIT state defined in the verilog

Appendix