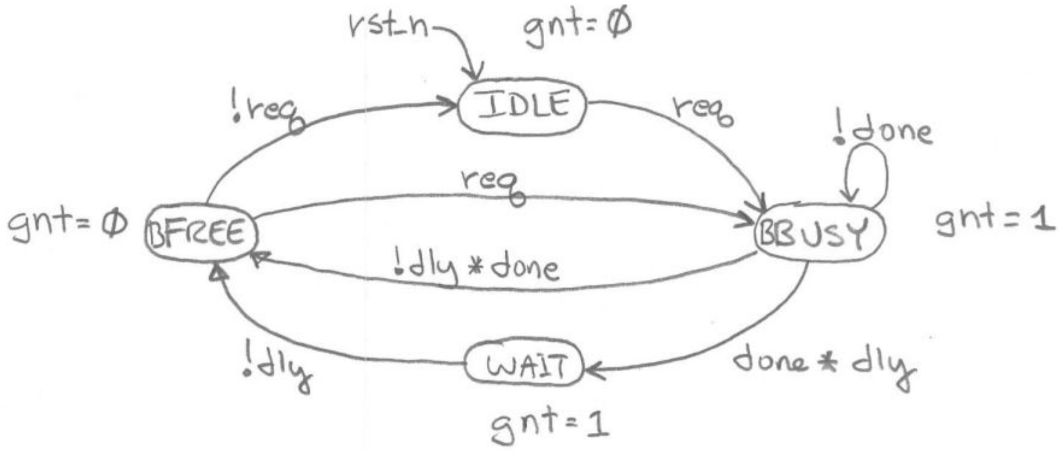
Finite State Machine

Dongjun Lee

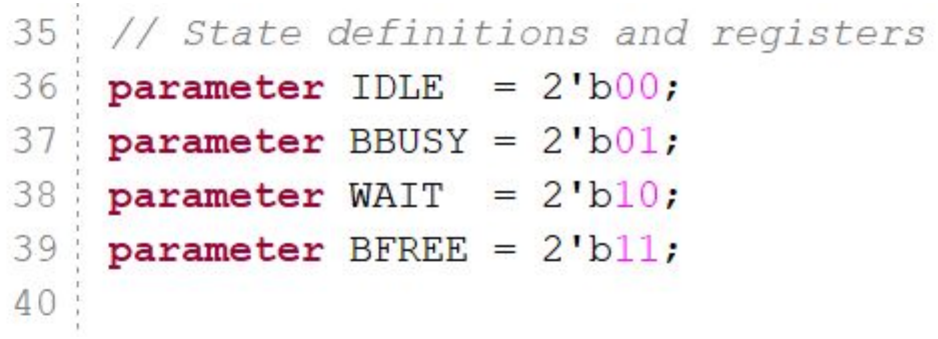
State Machines

- States are given names inside the bubbles
- Transitions between states are indicated by arcs
- Conditions for taking the arc given by a logic equation
- Outputs asserted in states are given by a list beside the state



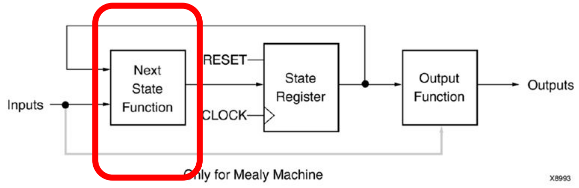
State Encoding

- State encoding done with parameters

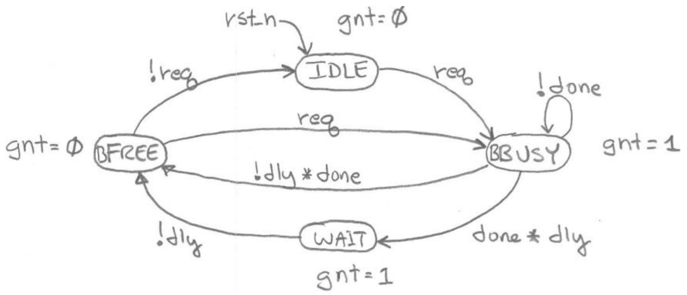


```
// State definitions and registers
parameter IDLE  = 2'b00;
parameter BBUSY = 2'b01;
parameter WAIT  = 2'b10;
parameter BFREE = 2'b11;

```

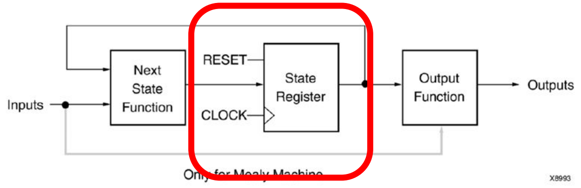
Next State Decoder - Combinational logic



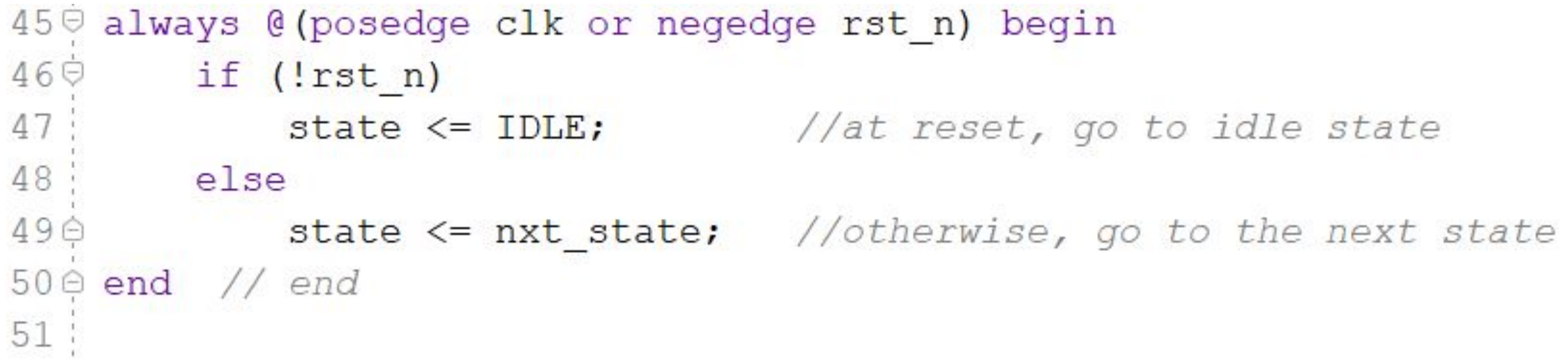
```
always @* begin
    case (state)
        IDLE: begin
            if (req == 1'b1)
                nxt_state = BBUSY;
            else
                nxt_state = IDLE;
        end // end IDLE

        BBUSY: begin
            if (done == 1'b0)
                nxt_state = BBUSY;
            else if (done == 1'b1 && dly == 1'b1)
                nxt_state = WAIT;
            else if (done == 1'b1 && dly == 1'b0)
                nxt_state = BFREE;
            else
                nxt_state = BBUSY;
        end // end BBUSY

        WAIT: begin
             if (dly == 1'b0)
                nxt_state = BFREE;
             else
                nxt_state = WAIT;
        end // end WAIT


        BFREE: begin
            if (req == 1'b1)
                nxt_state = BBUSY;
            else
                nxt_state = IDLE;
        end // end BFREE

        default: nxt_state = IDLE;

    endcase  // end case

    end  // end
```



State Register

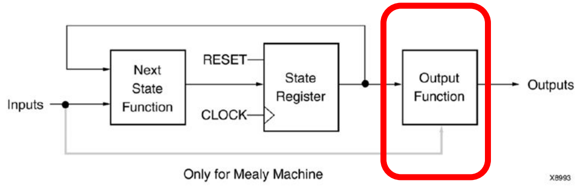


- State Register is implemented with a sequential logic
- Inferring rising edge triggered D-FFs to hold arbiter present state
- Asynchronous reset, active low
- Transitions between states occur only at the clock edge.

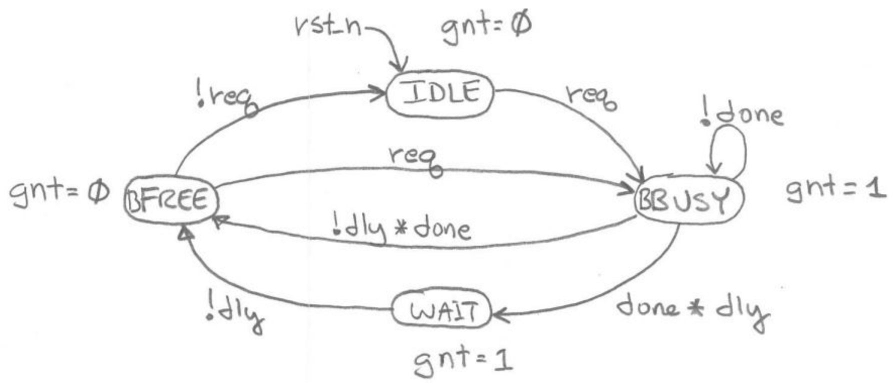


```verilog
always @(posedge clk or negedge rst_n) begin
    if (!rst_n)
        state <= IDLE;          //at reset, go to idle state
    else
        state <= nxt_state;     //otherwise, go to the next state
end  // end

```

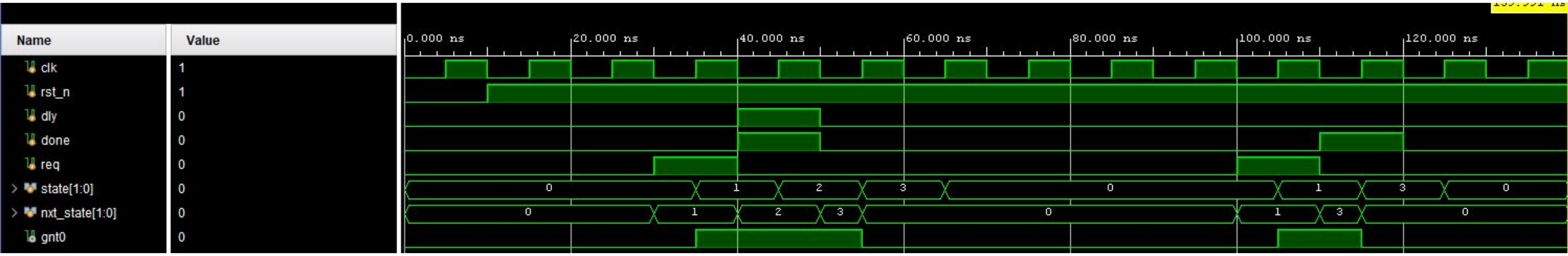
Output Logic - Combinational Logic



```
always @* begin
    case(state)
        IDLE: begin
            gnt = 1'b0;
        end // end IDLE

        BBUSY: begin
            gnt = 1'b1;
        end // end BBUSY

        WAIT: begin
           gnt = 1'b1;
        end // end WAIT

        BFREE: begin
            gnt = 1'b0;
        end // end BFREE

        default: gnt = 1'b0;

    endcase
end
```



Output



Name	Value
clk	1
rst_n	1
dly	0
done	0
req	0
state[1:0]	0
nxt_state[1:0]	0
gnt0	0