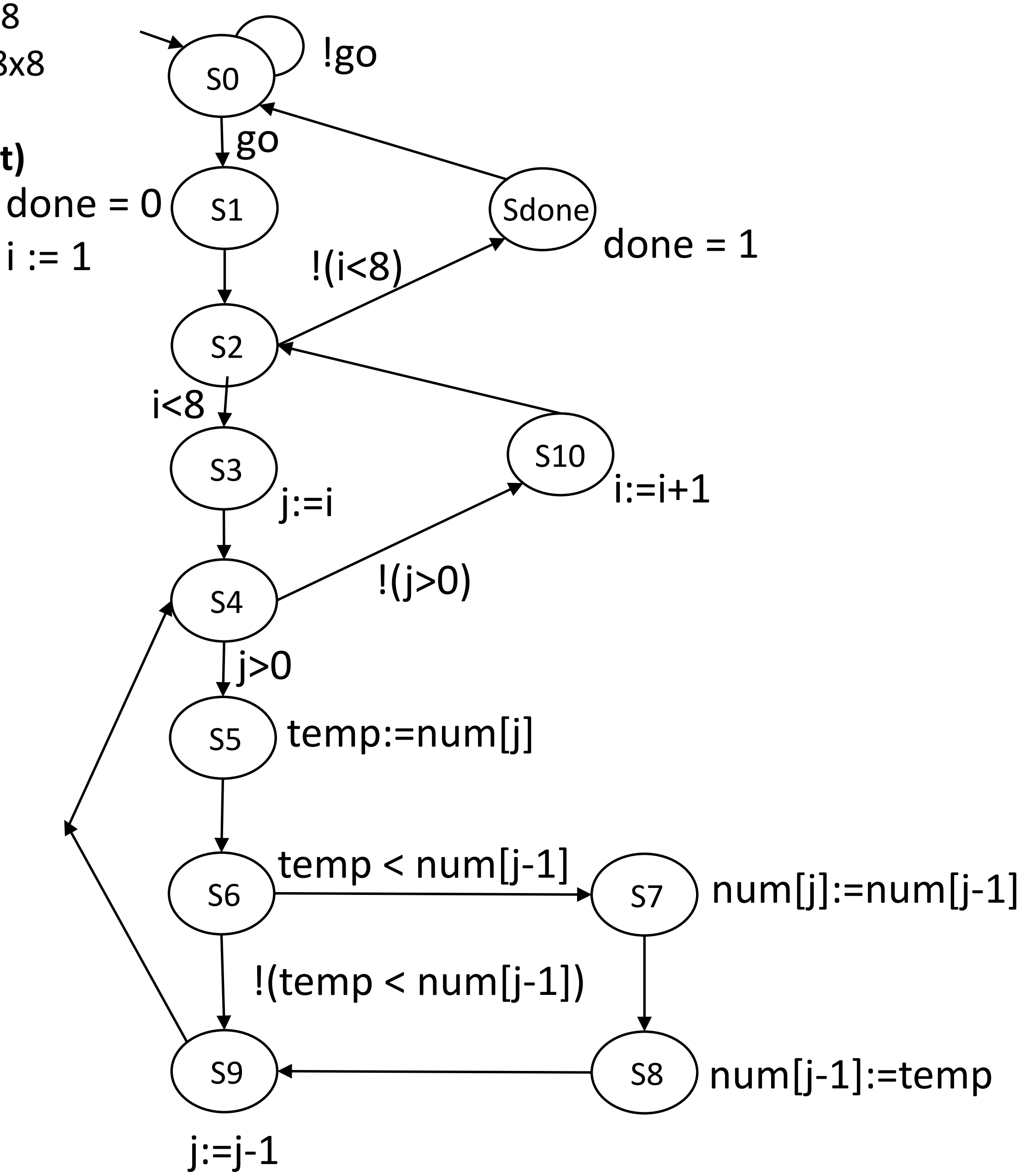
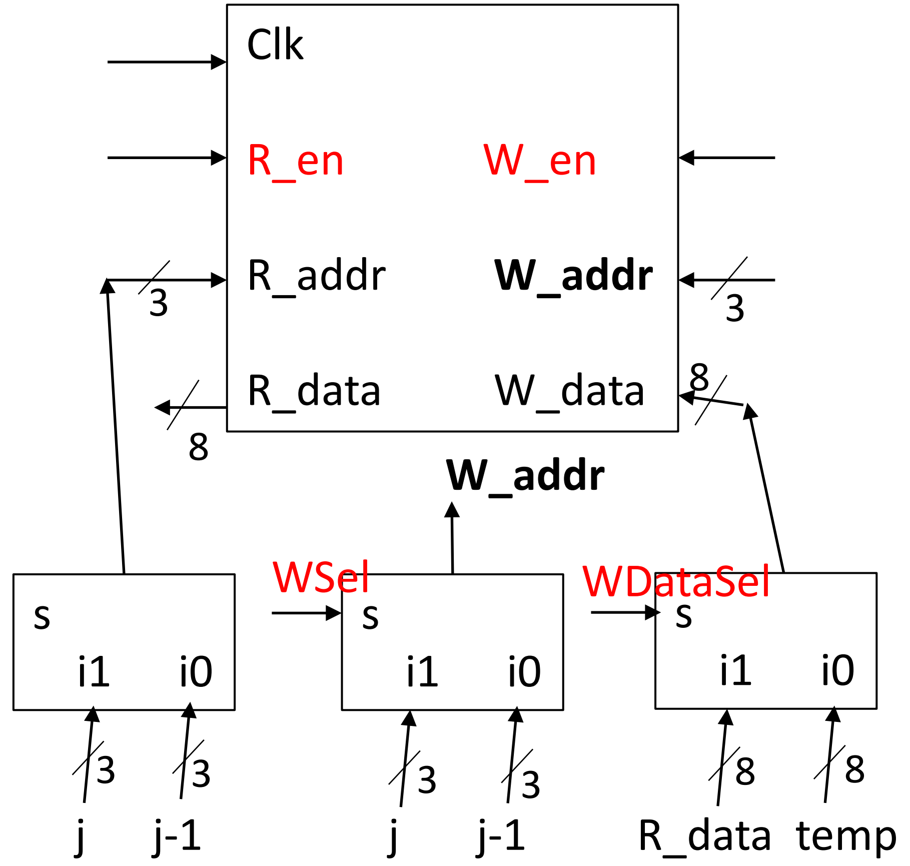
Given the following pseudo code, draw its HLSM and write Verilog code (behavioral)

input: go(1 bit), Regfile 8x8
output: go(1 bit), Regfile 8x8
Local storage/variable:
temp(8-bit), i(4-bit),j(3-bit)





```
`timescale 1ns / 1ps
module RTL_insertionsort(Clk, Rst, go, done);
  input Clk, Rst, go;
  output reg done;

  reg [3:0] state, nextstate;
  parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3;
  parameter s4 = 4, s5 = 5, s6 = 6, s7 = 7;
  parameter s8 = 8, s9 = 9, s10 = 10, sdone = 11;


  reg [3:0] i;
  reg [2:0] j;
  reg [7:0] temp;

  reg R_en, W_en;
  wire [7:0] R_Data, W_Data;
  wire [2:0] R_Addr, W_Addr;

  RegisterFile_8_8 Reg(                                             );
  always @(posedge Clk)begin
    if(Rst == 1)
      state <= s0;
    else
      state <= nextstate;
      case(state)








   endcase
  end
```

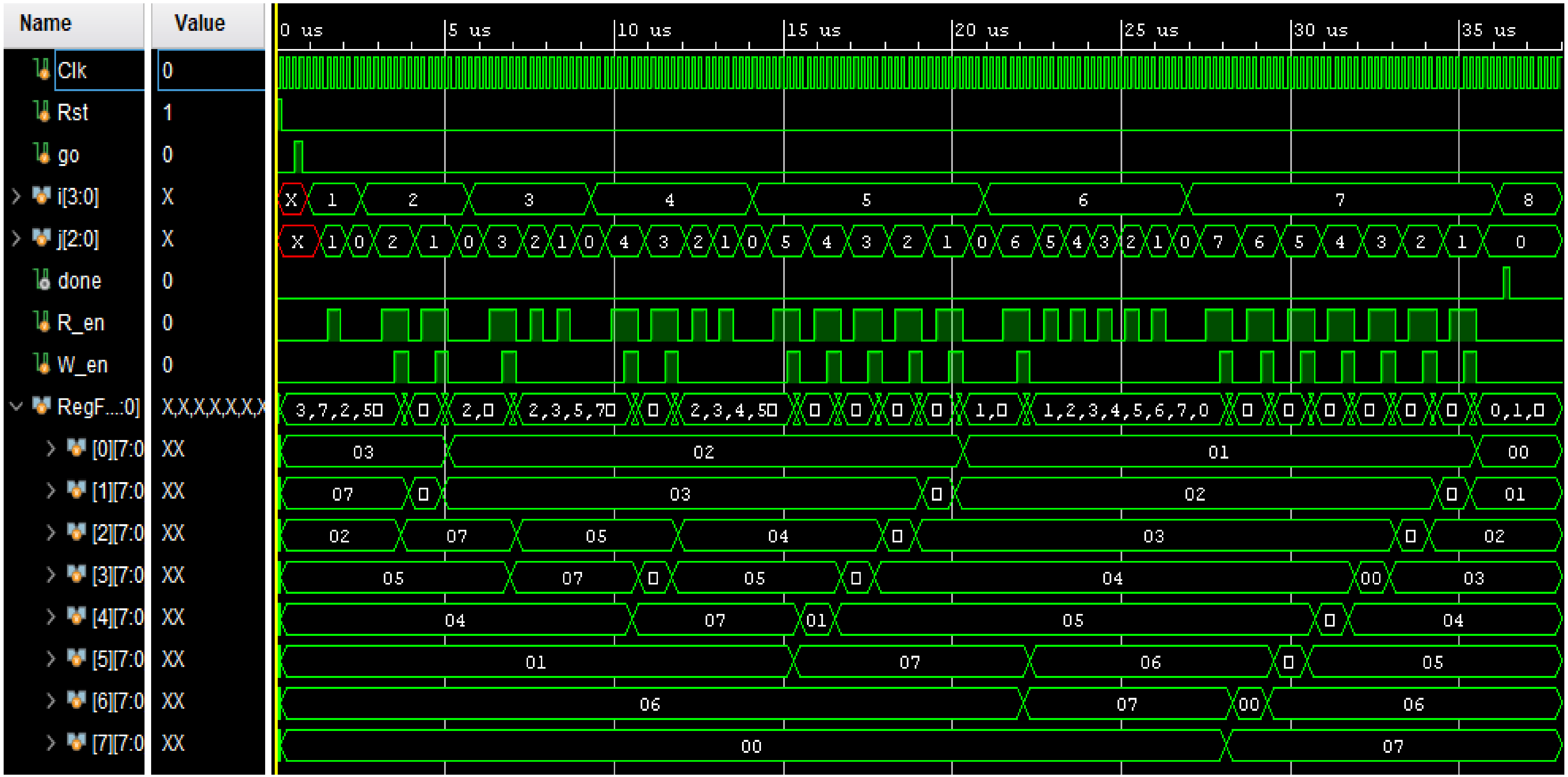
```
always @(*) begin
    //make all outputs to 0
    done <= 0;RSel <= 0; WSel <= 0; WDataSel <= 0;
    R_en <= 0;W_en <= 0;
    case(state)
      s0: begin
        if(go) nextstate <= s1;
        else nextstate <= s0;
      end
      s1: begin
        nextstate <= s2;
      end
      s2: begin





      end
      s3: begin
        nextstate <= s4;
      end
      s4: begin



       end
      s5: begin





     end
      s6: begin





    end
```

```
  s7: begin






  end
  s8: begin







  end
        s9: begin
          nextstate <= s4;
        end
        s10: begin
          nextstate <= s2;
        end
        sdone: begin
          done <= 1;
          nextstate <= s0;
        end
        default:nextstate <= s0;
      endcase
    end
  endmodule
```

```
`timescale 1ns / 1ps
module RegisterFile_8_8(R_Addr, W_Addr, R_en, W_en,R_Data, W_Data, Clk, Rst);
  input [2:0] R_Addr, W_Addr;
  input R_en, W_en;
  output reg [7:0] R_Data;
  input [7:0] W_Data;
  input Clk, Rst;

  reg [7:0] RegFile [0:7];
 // Write procedure
  always @(posedge Clk) begin
    if (Rst==1) begin
      RegFile[0] <= 8'd3;
      RegFile[1] <= 8'd7;
      RegFile[2] <= 8'd2;
      RegFile[3] <= 8'd5;
      RegFile[4] <= 8'd4;
      RegFile[5] <= 8'd1;
      RegFile[6] <= 8'd6;
      RegFile[7] <= 8'd0;
    end
    else if (W_en==1) begin
      RegFile[W_Addr] <= W_Data;
    end
  end

  // Read procedure
  always @* begin
    if (R_en==1)
      R_Data <= RegFile[R_Addr];
    else
      R_Data <= 8'hZZ;
  end
endmodule
```

```
//Testbench

`timescale 1ns / 1ps
module RTL_tb();
  reg Clk, Rst, go;
  wire done;
  RTL_insertionsort a1(Clk, Rst, go, done);

  always
  begin
    Clk <= 0;
    #100;
    Clk <= 1;
    #100;
  end
  initial
  begin
    Rst <= 1'b1; go <= 0;
    @ (posedge Clk);
    #50 Rst <= 1'b0;
    @ (posedge Clk);
    @ (posedge Clk);
    #50 go <= 1;
    @ (posedge Clk);
    #50 go <= 0;
  end
endmodule
```

Behavioral simulation

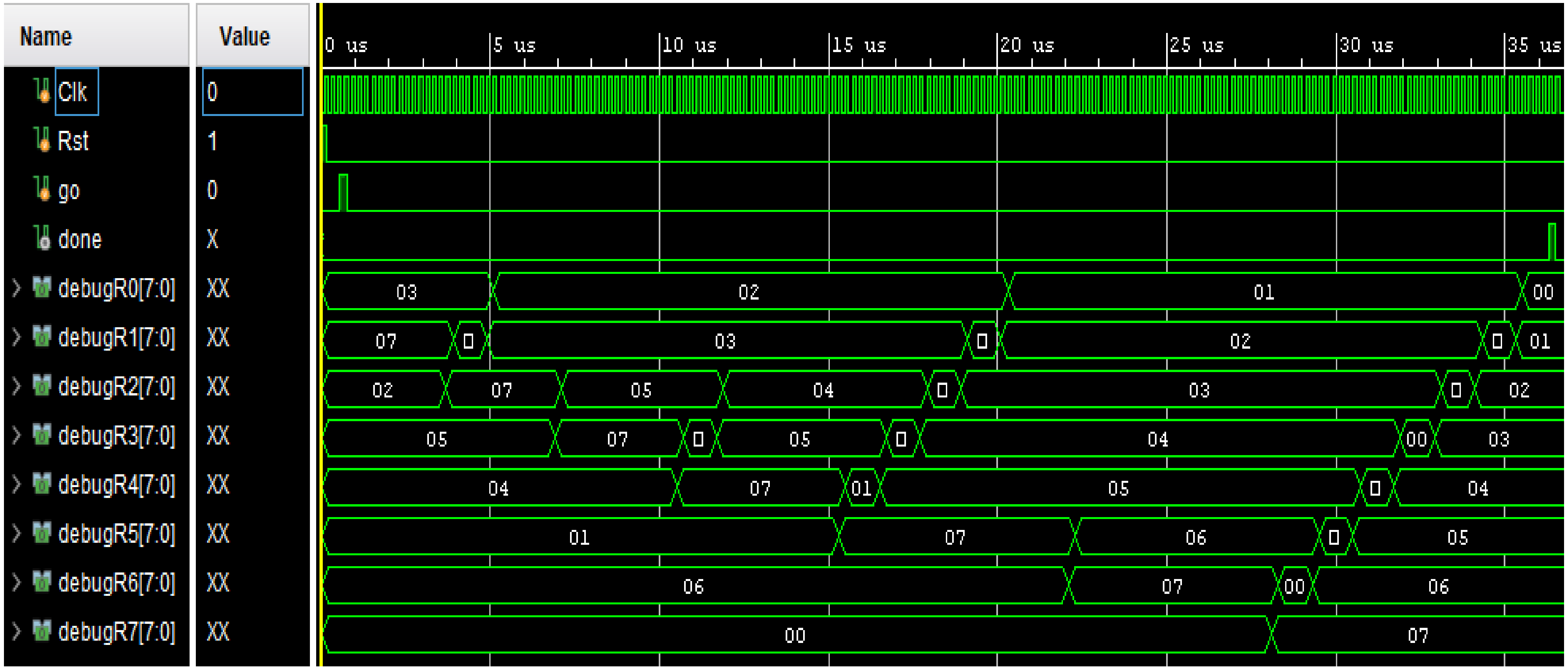


```
`timescale 1ns / 1ps
module RegisterFile_8_8(R_Addr, W_Addr, R_en, W_en,R_Data, W_Data, Clk, Rst, debugR0, debugR1,
debugR2, debugR3, debugR4, debugR5, debugR6, debugR7);
  output [7:0] debugR0, debugR1, debugR2, debugR3, debugR4, debugR5, debugR6, debugR7;
  input [2:0] R_Addr, W_Addr;
  input R_en, W_en;
  output reg [7:0] R_Data;
  input [7:0] W_Data;
  input Clk, Rst;
   (* mark_debug = "true" *) reg [7:0] RegFile [0:7];
  // Write procedure
  always @(posedge Clk) begin
    if (Rst==1) begin
      RegFile[0] <= 8'd3;RegFile[1] <= 8'd7;RegFile[2] <= 8'd2;RegFile[3] <= 8'd5;
      RegFile[4] <= 8'd4;RegFile[5] <= 8'd1;RegFile[6] <= 8'd6;RegFile[7] <= 8'd0;
    end
    else if (W_en==1) begin
      RegFile[W_Addr] <= W_Data;
    end
  end
  // Read procedure
  always @* begin
    if (R_en==1)
      R_Data <= RegFile[R_Addr];
    else
      R_Data <= 8'hZZ;
  end
    assign debugR0 = RegFile[0];
    assign debugR1 = RegFile[1];
    assign debugR2 = RegFile[2];
    assign debugR3 = RegFile[3];
    assign debugR4 = RegFile[4];
    assign debugR5 = RegFile[5];
    assign debugR6 = RegFile[6];
    assign debugR7 = RegFile[7];
endmodule
```

To see the register file content in post synthesis, retaining the signal is needed

Post-synthesis function simulation

debugRx (under Register file after simulation waveform shows up) are added to the waveform


Name
Value
Clk 0
Rst 1
go 0
done X
debugR0[7:0] XX
debugR1[7:0] XX
debugR2[7:0] XX
debugR3[7:0] XX
debugR4[7:0] XX
debugR5[7:0] XX
debugR6[7:0] XX
debugR7[7:0] XX
0 us
5 us
10 us
15 us
20 us
25 us
30 us
35 us
03 02 01 00
07 03 02 01
02 07 05 04 03 02
05 07 05 04 00 03
04 07 01 05 04
01 07 06 05
06 07 00 06
00 07