



Comparison of SystemVerilog Coding Style to Verilog-2001 One Always Block and Two Always Blocks Coding Styles

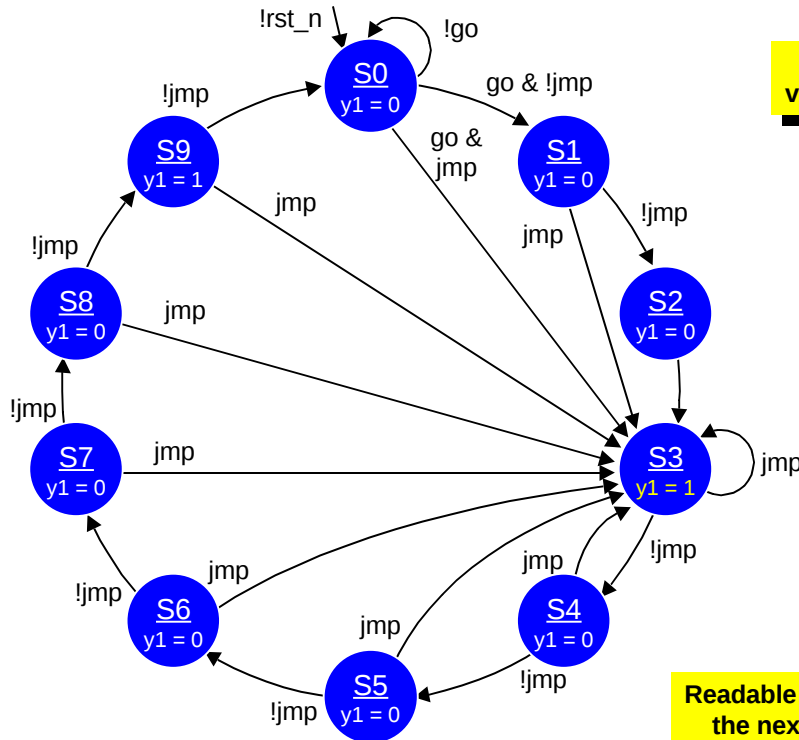
Advanced Verilog for Synthesis & Verification

by Sunburst Design, Beaverton, Oregon, © 1996-2001

7-109

FSM with Multiple Transition Arcs

State Diagram - One Always Block Coding Style



Very
verbose

Readable code on
the next slide

```
module fsm1 (output reg y1,
             input jmp, go, clk, rst_n);

    parameter S0 = 4'b0000,
               S1 = 4'b0001,
               S2 = 4'b0010,
               S3 = 4'b0011,
               S4 = 4'b0100,
               S5 = 4'b0101,
               S6 = 4'b0110,
               S7 = 4'b0111,
               S8 = 4'b1000,
               S9 = 4'b1001;

    reg [3:0] state;

    always @(posedge clk or negedge rst_n)
    if (!rst_n) begin
        state <= S0;
        y1 <= 1'b0;
    end
    else begin
        y1 <= 1'b0;
        state <= 4'bxx;
        case (state)
            S0 : if (!go) state <= S0;
                  else if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S1;
                  end
            S1 : if (!go) state <= S1;
                  else if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S2;
                  end
            S2 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S3 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
                  else state <= S4;
            S4 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S5 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S6 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S7 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S8 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            S9 : if (!jmp) begin
                      y1 <= 1'b1;
                      state <= S3;
                  end
            else state <= S0;
        endcase
    end
endmodule
```

Advanced Verilog for Synthesis & Verification

by Sunburst Design, Beaverton, Oregon, © 1996-2001

```
module fsm1 (output reg y1,
             input      jmp, go, clk, rst_n);

    parameter S0 = 4'b0000,
              S1 = 4'b0001,
              S2 = 4'b0010,
              S3 = 4'b0011,
              S4 = 4'b0100,
              S5 = 4'b0101,
              S6 = 4'b0110,
              S7 = 4'b0111,
              S8 = 4'b1000,
              S9 = 4'b1001;

    reg [3:0] state;

    always @(posedge clk or negedge rst_n)
        if (!rst_n) begin
            state <= IDLE;
            y1 <= 1'b0;
        end
        else begin
            y1 <= 1'b0;
            state <= 4'bx;
            case (state)
                S0 : if (!go)      state <= S0;
                     else if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S1;
                S1 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S2;
                S2 : begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                S3 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S4;
                S4 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S5;
                S5 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S6;
                S6 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S7;
                S7 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S8;
                S8 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S9;
                S9 : if (jmp) begin
                         y1 <= 1'b1;
                         state <= S3;
                     end
                     else          state <= S0;
            endcase
        end
    end
endmodule
```

by Sunburst Design, Beaverton, Oregon, © 1996-2001

FSM with Multiple Transition Arcs

Readable code on the next slide

Rev. 200112

Advanced Verilog for Synthesis & Verification

by Sunburst Design, Beaverton, Oregon, © 1996-2001

```
module fsm3 (output reg y1,
             input      jmp, go, clk, rst_n);

enum_state {S0 = 4'b0000,
            S1 = 4'b0001,
            S2 = 4'b0010,
            S3 = 4'b0011,
            S4 = 4'b0100,
            S5 = 4'b0101,
            S6 = 4'b0110,
            S7 = 4'b0111,
            S8 = 4'b1000,
            S9 = 4'b1001,
            XX = 4'bxxxx} state @(posedge clk);

always_comb begin
    y1 = 1'b0;
    if (!rst_n) -> state.S0;
    else begin
        -> state.XX;
        transition (state)
        S0 : if (!go)          -> S0;
              else if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S1;
        S1 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S2;
        S2 : begin
                  y1 = 1'b1;
                  -> S3;
              end
        S3 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S4;
        S4 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S5;
        S5 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S6;
        S6 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S7;
        S7 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S8;
        S8 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S9;
        S9 : if (jmp) begin
                  y1 = 1'b1;
                  -> S3;
              end
              else          -> S0;
        endtransition
    end
endmodule
```

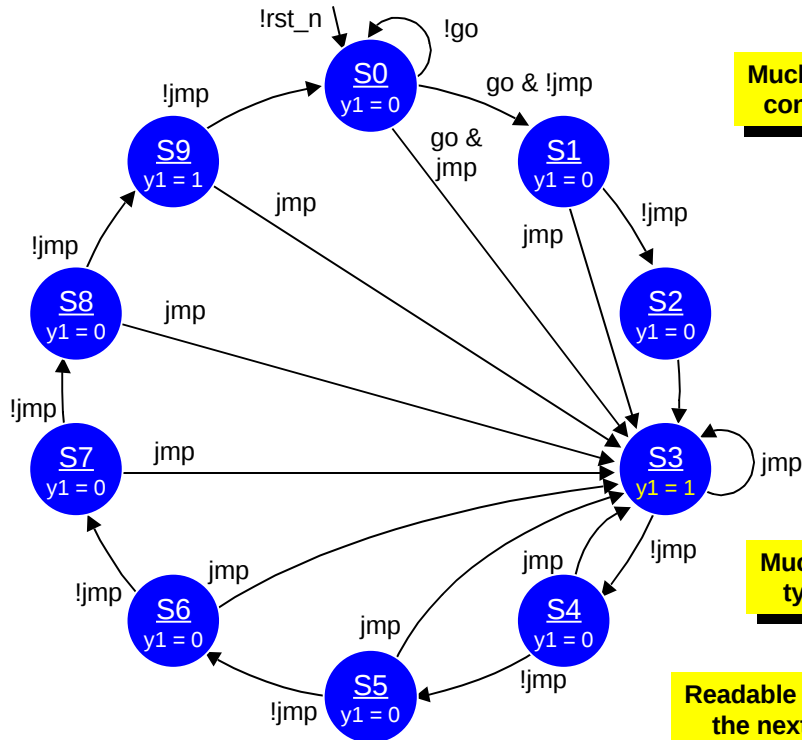
Advanced Verilog for Synthesis & Verification

by Sunburst Design, Beaverton, Oregon, © 1996-2001

7-113

FSM with Multiple Transition Arcs

State Diagram & Two Always Block Coding Style



Much more
concise!

Much less
typing

Readable code on
the next slide

```
module fsm2 (output reg y1,
             input jmp, go, clk, rst_n);

    parameter S0 = 4'b0000,
               S1 = 4'b0001,
               S2 = 4'b0010,
               S3 = 4'b0011,
               S4 = 4'b0100,
               S5 = 4'b0101,
               S6 = 4'b0110,
               S7 = 4'b0111,
               S8 = 4'b1000,
               S9 = 4'b1001;

    reg [3:0] state, next;

    always @(posedge clk or negedge rst_n)
        if (!rst_n) state <= IDLE;
        else state <= next;

    always @* begin
        state = 4'bxx;
        y1 = 1'b0;
        case (state)
            S0 : if (!go) state = S0;
                  else if (!jmp) state = S3;
                  else state = S1;
            S1 : if (!go) state = S2;
                  else if (!jmp) state = S3;
                  else state = S3;
            S2 : if (!jmp) state = S3;
                  else state = S3;
            S3 : begin y1 = 1'b1;
                    if (!jmp) state = S3;
                    else state = S4;
                end
            S4 : if (!jmp) state = S3;
                  else state = S3;
            S5 : if (!jmp) state = S3;
                  else state = S3;
            S6 : if (!jmp) state = S3;
                  else state = S3;
            S7 : if (!jmp) state = S3;
                  else state = S3;
            S8 : if (!jmp) state = S3;
                  else state = S3;
            S9 : if (!jmp) state = S3;
                  else state = S0;
        endcase
    end
endmodule
```

Advanced Verilog for Synthesis & Verification

by Sunburst Design, Beaverton, Oregon, © 1996-2001

```
module fsm2 (output reg y1,
             input  jmp, go, clk, rst_n);

    parameter S0 = 4'b0000,
               S1 = 4'b0001,
               S2 = 4'b0010,
               S3 = 4'b0011,
               S4 = 4'b0100,
               S5 = 4'b0101,
               S6 = 4'b0110,
               S7 = 4'b0111,
               S8 = 4'b1000,
               S9 = 4'b1001;

    reg [3:0] state, next;

    always @(posedge clk or negedge rst_n)
        if (!rst_n) state <= IDLE;
        else        state <= next;

    always @* begin
        state = 4'bx;
        y1 = 1'b0;
        case (state)
            S0 : if (!go)      state = S0;
                  else if (jmp) state = S3;
                  else        state = S1;
            S1 : if (jmp)      state = S3;
                  else        state = S2;
            S2 :              state = S3;
            S3 : begin y1 = 1'b1;
                  if (jmp)    state = S3;
                  else        state = S4;
                  end
            S4 : if (jmp)      state = S3;
                  else        state = S5;
            S5 : if (jmp)      state = S3;
                  else        state = S6;
            S6 : if (jmp)      state = S3;
                  else        state = S7;
            S7 : if (jmp)      state = S3;
                  else        state = S8;
            S8 : if (jmp)      state = S3;
                  else        state = S9;
            S9 : if (jmp)      state = S3;
                  else        state = S0;
        endcase
    end
endmodule
```