RVM Machine

RVM Machine has the following file with following hierarchy.

* RVM\_wrapper
  + RVM
    - Clk\_div
    - FSM
    - Bottle\_counter
    - Can\_counter
    - display

**RVM \_wrapper:**

|  |
| --- |
| module RVM\_wrapper  (AF,  BF,  LED,  SF,  anodes,  cathodes,  clk,  reset,  sensor);  output AF;  output BF;  output LED;  output SF;  output [3:0]anodes;  output [6:0]cathodes;  input clk;  input reset;  input [3:0]sensor;  wire AF;  wire BF;  wire LED;  wire SF;  wire [3:0]anodes;  wire [6:0]cathodes;  wire clk;  wire reset;  wire [3:0]sensor;  RVM RVM\_i  (.AF(AF),  .BF(BF),  .LED(LED),  .SF(SF),  .anodes(anodes),  .cathodes(cathodes),  .clk(clk),  .reset(reset),  .sensor(sensor));  endmodule |

**RVM:**

|  |
| --- |
| module RVM  (AF,  BF,  LED,  SF,  anodes,  cathodes,  clk,  reset,  sensor);  output AF;  output BF;  output LED;  output SF;  output [3:0]anodes;  output [6:0]cathodes;  input clk;  input reset;  input [3:0]sensor;  wire FSM\_0\_AF;  wire FSM\_0\_BC;  wire FSM\_0\_BF;  wire FSM\_0\_CC;  wire FSM\_0\_LED;  wire FSM\_0\_SF;  wire [2:0]FSM\_0\_data\_out;  wire [3:0]bottle\_counter\_0\_bot\_counter;  wire [3:0]can\_counter\_0\_can\_counter;  wire clk\_1;  wire clk\_div\_0\_clk\_div;  wire [3:0]display\_0\_anodes;  wire [6:0]display\_0\_cathodes;  wire reset\_1;  wire [3:0]sensor\_1;  assign AF = FSM\_0\_AF;  assign BF = FSM\_0\_BF;  assign LED = FSM\_0\_LED;  assign SF = FSM\_0\_SF;  assign anodes[3:0] = display\_0\_anodes;  assign cathodes[6:0] = display\_0\_cathodes;  assign clk\_1 = clk;  assign reset\_1 = reset;  assign sensor\_1 = sensor[3:0];  FSM FSM\_0  (.AF(FSM\_0\_AF),  .BC(FSM\_0\_BC),  .BF(FSM\_0\_BF),  .CC(FSM\_0\_CC),  .LED(FSM\_0\_LED),  .SF(FSM\_0\_SF),  .clk(clk\_div\_0\_clk\_div),  .data\_out(FSM\_0\_data\_out),  .reset(reset\_1),  .sensor(sensor\_1));  bottle\_counter bottle\_counter\_0  (.BC(FSM\_0\_BC),  .bot\_counter(bottle\_counter\_0\_bot\_counter),  .clk(clk\_div\_0\_clk\_div),  .reset(reset\_1));  can\_counter can\_counter\_0  (.CC(FSM\_0\_CC),  .can\_counter(can\_counter\_0\_can\_counter),  .clk(clk\_div\_0\_clk\_div),  .reset(reset\_1));  clk\_div clk\_div\_0  (.clk(clk\_1),  .clk\_div(clk\_div\_0\_clk\_div),  .clk\_reset(reset\_1));  display display\_0  (.anodes(display\_0\_anodes),  .bottle\_counter(bottle\_counter\_0\_bot\_counter),  .can\_counter(can\_counter\_0\_can\_counter),  .cathodes(display\_0\_cathodes),  .clk(clk\_1),  .clk\_div(clk\_div\_0\_clk\_div),  .data\_out(FSM\_0\_data\_out),  .reset(reset\_1));  endmodule |

**Clk\_Div:**

|  |
| --- |
| module clk\_div#  (  parameter REQUIRED\_CLK = 250000 //2 Hz, Supposing base clock as 50Mhz  )  (  input clk,  input clk\_reset,  output wire clk\_div  // output reg done\_clk  );  reg [31:0]divider\_clk\_counter = 32'd0;  reg [3:0] done\_counter = 4'd0;  reg clk\_reset\_det = 0;  reg clk\_inv = 0;    always@(posedge clk)  begin  clk\_reset\_det <= clk\_reset;  if (clk\_reset)  begin  done\_counter <= 0;  divider\_clk\_counter <= 0;  clk\_inv <= 0;  end  else  begin  if (divider\_clk\_counter < REQUIRED\_CLK -1)  begin  divider\_clk\_counter <= divider\_clk\_counter + 1;  end  else  begin  if(done\_counter > 5)  begin  // done\_clk <= 1;  done\_counter <= 0;  end  else  begin  divider\_clk\_counter <= 0;  done\_counter <= done\_counter + 1;  clk\_inv = ~clk\_inv;  // done\_clk <= 0;  end  end  end  end  assign clk\_div = clk\_inv;  endmodule |

**FSM:**

|  |
| --- |
| module FSM  (  input clk,  input reset,  input [3:0]sensor,  output reg AF,  output reg SF,  output reg BF,  output reg [2:0]data\_out,  output wire BC,  output wire CC,  output wire LED  // output reg clk\_div\_in  );  reg [3:0]state = 4'd0;  reg LED\_ON = 0;  reg [9:0]state\_counter = 0;  reg [1:0] data\_counter = 2'd0;  reg [2:0] can\_counter = 3'd0;  reg [2:0] bottle\_counter = 3'd0;  reg [2:0] Error\_counter = 3'd0;  reg [3:0] LED\_Counter = 3'd0;  reg BCC = 1'b0;  reg CCC = 1'b0;  reg LEDD = 1'b0;  always@(posedge clk)  begin  if (reset)  begin  state <= 4'd0;  end  else  begin  case(state)  4'b0000: //IDLE  begin  BF <= 1'b1;  SF <= 1'b1;  AF <= 1'b0;  data\_out <= 3'd0;  if (sensor == 4'd1)  begin  state <= 4'b0001;  end  else  begin  state <= 4'd0;  end  end  4'b0001: // DETECTED  begin  SF <= 1'b1;  AF <= 1'b1;  BF <= 1'b1;  data\_out <= 3'd1;  if (data\_counter > 1)  begin  data\_counter <= 0;  if (sensor == 4'b0011)  begin  state <= 4'b0011;  end  else  if (sensor == 4'b0101)  begin  state <= 4'b0101;  end  else  begin  if (sensor == 4'b1001)  begin  state <= 4'b1001;  end  else  begin  state <= 4'b0001;  end  end  end  else  begin  data\_counter <= data\_counter + 1;  end  end  4'b0011: // CAN DETECTED  begin  SF <= 1'b0;  AF <= 1'b1;  BF <= 1'b1;  data\_out <= 3'd2;  if (can\_counter > 3)  begin  can\_counter <= 0;  CCC <= 1;  state <= 4'b1010;  end  else  begin  can\_counter <= can\_counter + 1;  end  end  4'b0101: // BOTTLE DETECTED  begin  SF <= 1'b0;  AF <= 1'b1;  BF <= 1'b0;  data\_out <= 3'd3;  if (bottle\_counter > 3)  begin  bottle\_counter <= 0;  state <= 4'b1010;  BCC <= 1;  end  else  begin  bottle\_counter <= bottle\_counter + 1;  end  end  4'b1001: // Error Detection  begin  SF <= 1'b1;  AF <= 1'b0;  data\_out <= 3'd4;  if (Error\_counter > 3)  begin  Error\_counter <= 0;  state <= 4'b1010;  end  else  begin  Error\_counter <= Error\_counter + 1;  end  end  4'b1010: // END  begin  data\_out <= 3'd5;  BCC <= 0;  CCC <= 0;  LED\_ON <= 1;  // clk\_div\_in <= 1;  if (LED\_Counter > 5)  begin  state <= 4'b0000;  LED\_Counter <= 0;  LEDD <= 0;  end  else  begin  LED\_Counter <= LED\_Counter + 1;  LEDD = ~LEDD;  end  end  default:  state <= 4'd0;  endcase  end  end  assign BC = BCC;  assign CC = CCC;  assign LED = LEDD;  endmodule |

**Can\_Counter:**

|  |
| --- |
| module can\_counter(  input clk,  input reset,  input CC,  output reg [3:0]can\_counter = 8'd0  );    reg CC\_det;    always@(posedge clk)  begin  if(reset)  begin  can\_counter <= 0;  end  else  begin  CC\_det <= CC;  if (CC)  begin  if (can\_counter > 9)  begin  can\_counter <= 0;  end  else  begin  can\_counter <= can\_counter + 1;  end  end  else  can\_counter <= can\_counter;  end  end  endmodule |

**Bottle\_Counter:**

|  |
| --- |
| module bottle\_counter(  input clk,  input reset,  input BC,  output reg [3:0]bot\_counter = 8'd0  );    reg BC\_det;    always@(posedge clk)  begin  if(reset)  begin  bot\_counter <= 0;  end  else  begin  BC\_det <= BC;  if (BC)  begin  if (bot\_counter > 9)  begin  bot\_counter <= 0;  end  else  begin  bot\_counter <= bot\_counter + 1;  end  end  else  bot\_counter <= bot\_counter;  end  end  endmodule |

**Display:**

|  |
| --- |
| module display(  input clk,  input clk\_div,  input reset,  input [2:0]data\_out,  input [3:0]can\_counter,  input [3:0]bottle\_counter,  output reg [6:0] cathodes,  output reg [3:0] anodes  );  reg [1:0]switch = 2'd0;  reg [19:0]clock\_counter = 20'd0;  reg [3:0]clk\_counter\_1 = 4'd0;  reg [3:0]clk\_counter\_2 = 4'd0;  reg [3:0]clk\_counter\_3 = 4'd0;  reg [1:0] switcheroni = 2'd0;  always@(posedge clk)  begin  if(reset)  begin  anodes <= 4'b1111;  end  else  begin  case(switch)  2'd0:  begin  anodes <= 4'b1110;  end  2'd1:  begin  anodes <= 4'b1101;  end  2'd2:  begin  anodes <= 4'b1011;  end  2'd3:  begin  anodes <= 4'b1111;  end  default:  begin  anodes <= 4'b1111;  end  endcase  end  end  always@(posedge clk)  begin  if(reset)  begin    end  else  begin  if (clock\_counter < 100000-1)  begin  clock\_counter <= clock\_counter + 1;  end  else  begin  clock\_counter <= 0;  if (switch > 2)  begin  switch <= 0;  end  else  begin  switch <= switch + 1;  end  end  case (switch)  2'd0:  begin  case(switcheroni)  2'd0:  begin  case (data\_out)  4'd0:  begin  cathodes <= 7'b1110001;  end  4'd1:  begin  cathodes <= 7'b1110000;  end  4'd2:  begin  cathodes <= 7'b0001001;  end  4'd3:  begin  cathodes <= 7'b1110000;  end  4'd4:  begin  cathodes <= 7'b1111010;  end  4'd5:  begin  cathodes <= 7'b1000010;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd1:  begin  case (can\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd2:  begin  case (bottle\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd1:  begin  case(switcheroni)  2'd0:  begin  case (data\_out)  4'd0:  begin  cathodes <= 7'b1000010;  end  4'd1:  begin  cathodes <= 7'b0110000;  end  4'd2:  begin  cathodes <= 7'b0001000;  end  4'd3:  begin  cathodes <= 7'b0000001;  end  4'd4:  begin  cathodes <= 7'b1111010;  end  4'd5:  begin  cathodes <= 7'b1101010;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd1:  begin  case (can\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd2:  begin  case (bottle\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd2:  begin  case(switcheroni)  2'd0:  begin  case (data\_out)  4'd0:  begin  cathodes <= 7'b1001111;  end  4'd1:  begin  cathodes <= 7'b1000010;  end  4'd2:  begin  cathodes <= 7'b0110001;  end  4'd3:  begin  cathodes <= 7'b1100000;  end  4'd4:  begin  cathodes <= 7'b0110000;  end  4'd5:  begin  cathodes <= 7'b0110000;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd1:  begin  case (can\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  2'd2:  begin  case (bottle\_counter)  4'd0:  begin  cathodes <= 7'b0000001;  end  4'd1:  begin  cathodes <= 7'b1001111;  end  4'd2:  begin  cathodes <= 7'b0010010;  end  4'd3:  begin  cathodes <= 7'b0000110;  end  4'd4:  begin  cathodes <= 7'b1001100;  end  4'd5:  begin  cathodes <= 7'b0100100;  end  4'd6:  begin  cathodes <= 7'b0100000;  end  4'd7:  begin  cathodes <= 7'b0001111;  end  4'd8:  begin  cathodes <= 7'b0000000;  end  4'd9:  begin  cathodes <= 7'b0000100;  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end  default:  begin  cathodes <= 7'b1111111;  end  endcase  end    default:  begin  switch <= 2'd0;  end  endcase  end  end  always@(posedge clk\_div)  begin  if (clk\_counter\_1 > 9)  begin  if(clk\_counter\_2 > 1)  begin  if(clk\_counter\_3 > 1)  begin  clk\_counter\_1 <= 0;  clk\_counter\_2 <= 0;  clk\_counter\_3 <= 0;  switcheroni <= 2'd0;  end  else  begin  switcheroni <= 2'd2;  clk\_counter\_3 <= clk\_counter\_3 + 1;  end  end  else  begin  switcheroni <= 2'd1;  clk\_counter\_2 <= clk\_counter\_2 + 1;  end  end  else  begin  clk\_counter\_1 <= clk\_counter\_1 + 1;  end  end  endmodule |

**Simulation Results:**

