Problem 1.

(a) Diagram with states A0, A1, and A2, transitions based on inputs A[0] and A[1], and outputs state and q.

(b) State transition diagram with states 0, 1, and 2, transitions based on inputs A[0] and A[1].

(c) Timing diagram with inputs clk, rst, A[0], A[1], state, and output q.
Problem 2

a)

Diagram of a seven-segment display with states 0 to 6 and reset and up transitions.

Note: load causes branch to count = data [2:0]

b)

Diagram of a counter with reset, load, and data inputs, and a clock signal.

Output: count [2:0]
module hw2_2FSM_v1(
    input clk, input reset,
    input up, input load, input [2:0] data,
    output [2:0] count
);
reg [2:0] state;
always @(posedge clk)
begin
    if(reset)
        state <= 0;
    else if(load)
        state <= data;
    else if(up)
        state <= state + 3;
    else
        state <= state - 3;
end
assign count = state;
endmodule

module hw2_2FSM_v2(
    input clk, input reset,
    input up, input load, input [2:0] data,
    output [2:0] count
);
reg [2:0] state, nextstate;
always @(posedge clk)
begin
    if(reset) state <= 0;
    else state <= nextstate
end
always @(*)
begin
    nextstate <= state
    if(load)
        nextstate <= data;
    else if(up)
        nextstate <= state + 3;
    else
        nextstate <= state - 3;
end
assign count = state;
endmodule
Problem 3

module summer(
    input clk, input rst,
    output [7:0] total, output done
);

reg [3:0] addr_counter;
reg [7:0] accumulator;
wire [3:0] mem_out;

always @(posedge clk)
begin
    if(rst) addr_counter <= 0;
    else addr_counter <= addr_counter + 1;
end

RAM16x4S #(  
    // These constants can be anything but this part should be here.
    .INIT_00(16'hffff),   // INIT for bit 0 of RAM
    .INIT_01(16'h0000),   // INIT for bit 1 of RAM
    .INIT_02(16'h0000),   // INIT for bit 2 of RAM
    .INIT_03(16'h0000)    // INIT for bit 3 of RAM
) RAM16x4S_inst (  
    .O0(mem_out[0]),
    .O1(mem_out[1]),
    .O2(mem_out[2]),
    .O3(mem_out[3]),
    .A0(addr_counter[0]),
    .A1(addr_counter[1]),
    .A2(addr_counter[2]),
    .A3(addr_counter[3]),
    .D0(0),   // can be whatever
    .D1(0),   // can be whatever
    .D2(0),   // can be whatever
    .D3(0),   // can be whatever
    .WCLK(0), // can be whatever
    .WE(0)   // THIS MUST BE HERE
);

always @(posedge clk)
begin
    if(rst) accumulator <= 0;
    else accumulator <= accumulator + mem_out;
end

assign total = accumulator;
assign done = addr_counter == 4'b111;

endmodule
\[ 106 \text{ MHz} \Rightarrow 10 \text{ ns} \]

\[ \text{clk} \]

\[ P_0 \rightarrow 1.5 \text{ ns} = T_{cko} \]

\[ Q_0 \]

\[ T_{cko} + T_{delayy} + T_{inv} \]

\[ T_{setup} \]

\[ P_1 \rightarrow T_{setup} \]

\[ Q_1 \]

\[ b, 2.8 \text{ ns} + 1.0 \text{ ns} = 3.8 \text{ ns} \text{ minimum clock period} \]

\[ b \text{ min period} = \max \left\{ T_{cko} + T_{delayy} + T_{setup}, T_{cko} + T_{delayy}^2 + T_{inv} + T_{setup} \right\} \]

\[ = \max \left\{ 3.5, 3.8 \text{ ns} \right\} \]