`define INITIAL 3'b000

`define DEPOSIT 3'b001

`define BUY 3'b010

`define CHANGE 3'b011

`define RETURN 3'b100

module lab05(clk,rst,money\_5,money\_10,cancel,drink\_A,drink\_B,/\*drop\_money,\*/enough\_A,enough\_B,DIGIT,DISPLAY);

    input clk;

    input rst;

    input money\_5;

    input money\_10;

    input cancel;

    input drink\_A;

    input drink\_B;

    //output reg [9:0] drop\_money;\*/

    output reg enough\_A;

    output reg enough\_B;

    output [3:0]DIGIT;

    output [6:0]DISPLAY;

    wire clk16;

    wire clk26;

    wire clk13;

    wire clk\_;

    wire money\_5\_de;

    wire money\_5\_pul;

    wire money\_10\_de;

    wire money\_10\_pul;

    wire drink\_A\_de;

    wire drink\_A\_pul;

    wire drink\_B\_de;

    wire drink\_B\_pul;

    wire cancel\_de;

    wire cancel\_pul;

    reg [3:0]BCD0;

    reg [3:0]BCD1;

    reg [3:0]BCD2;

    reg [3:0]BCD3;

    reg [3:0]nextBCD0;

    reg [3:0]nextBCD1;

    reg [3:0]nextBCD2;

    reg [3:0]nextBCD3;

    reg [6:0]display;

    reg [3:0]digit;

    reg [3:0]val;

    reg [1:0]state;

    reg [1:0]nextstate;

    reg [3:0]moneydigit;

    reg [3:0]moneyten;

    reg [3:0]nextmoneydigit;

    reg [3:0]nextmoneyten;

    reg countA;

    reg nextcountA;

    reg countB;

    reg nextcountB;

    reg DetoBuy;

    reg cofe;

    reg nextcofe;

    reg coke;

    reg nextcoke;

    reg iscancel;

    reg nextiscancel;

    clk\_divider #(16)clk1 (.clk(clk),.clk\_div(clk16));

    clk\_divider #(26)clk2 (.clk(clk),.clk\_div(clk26));

    clk\_divider #(13)clk3 (.clk(clk),.clk\_div(clk13));

    debounce de1(.pb\_debounced(money\_5\_de),.pb(money\_5),.clk(clk16));

    onepulse on1(.clk(clk16),.pb\_debounced(money\_5\_de),.pb\_1pulse(money\_5\_pul));

    debounce de2(.pb\_debounced(money\_10\_de),.pb(money\_10),.clk(clk16));

    onepulse on2(.clk(clk16),.pb\_debounced(money\_10\_de),.pb\_1pulse(money\_10\_pul));

    debounce de3(.pb\_debounced(drink\_A\_de),.pb(drink\_A),.clk(clk16));

    onepulse on3(.clk(clk16),.pb\_debounced(drink\_A\_de),.pb\_1pulse(drink\_A\_pul));

    debounce de4(.pb\_debounced(drink\_B\_de),.pb(drink\_B),.clk(clk16));

    onepulse on4(.clk(clk16),.pb\_debounced(drink\_B\_de),.pb\_1pulse(drink\_B\_pul));

    debounce de5(.pb\_debounced(cancel\_de),.pb(cancel),.clk(clk16));

    onepulse on5(.clk(clk16),.pb\_debounced(cancel\_de),.pb\_1pulse(cancel\_pul));

    always@(\*)

        begin

            if(rst==1'b1) begin nextBCD0=4'b0000; nextBCD1=4'b0000; nextBCD2=4'b0000; nextBCD3=4'b0000; nextstate=`INITIAL;

            enough\_A=1'b0;enough\_B=1'b0;DetoBuy=1'b0;nextcountA=1'b0;nextcountB=1'b0; nextcofe=1'b0; nextcoke=1'b0; nextiscancel=1'b0;end

            case(state)

                `INITIAL:begin

                    nextstate=`DEPOSIT;

                    enough\_A=1'b0;

                    enough\_B=1'b0;

                    DetoBuy=1'b0;

                end

                `DEPOSIT:begin

                    if(BCD1>=4'b0010) begin enough\_A=1'b1; end

                    else begin enough\_A=1'b0; end

                    if((BCD1==4'b0010 && BCD0==4'b0101)||(BCD1>4'b0010)) begin enough\_B=1'b1; end

                    else begin enough\_B=1'b0; end

                    if(money\_5\_pul==1'b1)

                        begin

                            if(BCD1==4'b1001 && BCD0==4'b0101) begin nextBCD1=BCD1; nextBCD0=BCD0; end

                            else if(BCD1!=4'b1001 && BCD0==4'b0101) begin nextBCD1=BCD1+4'b0001; nextBCD0=4'b0000; end

                            else begin nextBCD1=BCD1; nextBCD0=BCD0+4'b0101; end

                            nextBCD2=BCD2;

                            nextBCD3=BCD3;

                            nextcountA=countA;

                            nextcountB=countB;

                        end

                    else if(money\_10\_pul==1'b1)

                        begin

                            if((BCD1==4'b1001 && BCD0==4'b0101)||(BCD1==4'b1001 && BCD0==4'b0000))

                                begin

                                    nextBCD1=BCD1; nextBCD0=BCD0;

                                end

                            else begin nextBCD1=BCD1+4'b0001; nextBCD0=BCD0; end

                            nextBCD3=BCD3;nextBCD2=BCD2;

                            nextcountA=countA;

                            nextcountB=countB;

                        end

                    else if(drink\_A\_pul==1'b1)

                        begin

                            nextcountA=countA+1'b1;

                            nextcountB=4'b0000;

                            if(countA==1'b1&&enough\_A==1'b1) begin DetoBuy=1'b1; nextcoke=1'b1; nextcofe=1'b0;end

                            else begin DetoBuy=1'b0;  nextBCD3=4'b0010; nextBCD2=4'b0000; nextBCD1=BCD1; nextBCD0=BCD0; end

                            nextBCD3=4'b0010; nextBCD2=4'b0000; nextBCD1=BCD1; nextBCD0=BCD0;

                        end

                    else if(drink\_B\_pul==1'b1)

                        begin

                            nextcountB=countB+1'b1;

                            nextcountA=4'b0000;

                            if(countB==1'b1&&enough\_B==1'b1) begin DetoBuy=1'b1; nextcoke=1'b0; nextcofe=1'b1; end

                            else begin DetoBuy=1'b0; nextBCD3=4'b0010; nextBCD2=4'b0101; nextBCD1=BCD1; nextBCD0=BCD0;end

                            nextBCD3=4'b0010; nextBCD2=4'b0101; nextBCD1=BCD1; nextBCD0=BCD0;

                        end

                   else

                        begin

                            nextcountA<=countA;

                            nextcountB<=countB;

                            nextBCD0<=BCD0;

                            nextBCD1<=BCD1;

                            nextBCD2<=BCD2;

                            nextBCD3<=BCD3;

                            nextcofe<=1'b0;

                            nextcoke<=1'b0;

                        end

                    nextstate=(DetoBuy==1'b1)?`BUY:`DEPOSIT;

                end

                `BUY:begin

                    nextstate=`CHANGE;

                    //begin nextBCD3=4'b1010; nextBCD2=4'b1011; nextBCD1=4'b1100; nextBCD0=4'b1101; end

                    if(coke==1'b1)begin nextBCD3=4'b1010; nextBCD2=4'b1011; nextBCD1=4'b1110; nextBCD0=4'b1101; end

                    else if(cofe==1'b1) begin nextBCD3=4'b1010; nextBCD2=4'b1011; nextBCD1=4'b1100; nextBCD0=4'b1101; end

                    /\*if(cofe==1'b1) begin nextBCD3=4'b1010; nextBCD2=4'b1011; nextBCD1=4'b1100; nextBCD0=4'b1101; end

                    else if(coke==1'b1) begin nextBCD3=4'b1010; nextBCD2=4'b1011; nextBCD1=4'b1110; nextBCD0=4'b1101; end\*/

                end

                `CHANGE:begin

                    nextstate=`CHANGE;

                    nextBCD3=4'b1001; nextBCD2=4'b0100; nextBCD1=4'b0010; nextBCD0=4'b0000;

                    /\*if(BCD1==4'b0000) begin nextBCD3=4'b0000; nextBCD2=4'b0000; nextBCD1=BCD1; nextBCD0=BCD0-4'b0101; end

                    else begin nextBCD3=4'b0000; nextBCD2=4'b0000; nextBCD1=BCD1-4'b0001; nextBCD0=BCD0; end\*/

                end

            endcase

            case(val)

                4'b0000:begin display=7'b0000001; end

                4'b0001:begin display=7'b1001111; end

                4'b0010:begin display=7'b0010010; end

                4'b0011:begin display=7'b0000110; end

                4'b0100:begin display=7'b1001100; end

                4'b0101:begin display=7'b0100100; end

                4'b0110:begin display=7'b0100000; end

                4'b0111:begin display=7'b0001111; end

                4'b1000:begin display=7'b0000000; end

                4'b1001:begin display=7'b0000100; end

                4'b1010:begin display=7'b1110010; end //c=10

                4'b1011:begin display=7'b1100010; end //o=11

                4'b1100:begin display=7'b0111000; end //f=12

                4'b1101:begin display=7'b0110000; end //e=13

                4'b1110:begin display=7'b0101000; end //k=14

                default: begin display=7'b0000001; end

            endcase

        end

    always@(posedge clk\_,posedge rst)

        begin

            if(rst==1'b1)

                begin

                    BCD0<=4'b0000;

                    BCD1<=4'b0000;

                    BCD2<=4'b0000;

                    BCD3<=4'b0000;

                    state<=`INITIAL;

                    countA<=1'b0;

                    countB<=1'b0;

                end

            else

                begin

                    BCD0<=nextBCD0;

                    BCD1<=nextBCD1;

                    BCD2<=nextBCD2;

                    BCD3<=nextBCD3;

                    countA<=nextcountA;

                    countB<=nextcountB;

                    cofe<=nextcofe;

                    coke<=nextcoke;

                    state<=nextstate;

                end

        end

    /\*always@(posedge clk13) //update FSM

            begin

                lastpushA<=pushA;

                lastpushB<=pushB;

            end      \*/

    always@(posedge clk13)

            begin

                case(digit)

                    4'b1110: begin val=BCD1; digit=4'b1101; end

                    4'b1101: begin val=BCD2; digit=4'b1011; end

                    4'b1011: begin val=BCD3; digit=4'b0111; end

                    4'b0111: begin val=BCD0; digit=4'b1110; end

                    default: begin val=BCD0; digit=4'b1110; end

                endcase

            end

    assign DISPLAY=display;

    assign DIGIT=digit;

    assign clk\_=(state==`INITIAL)?clk16:(state==`DEPOSIT)?clk16:(state==`BUY)?clk26:clk26;

endmodule

module clk\_divider(clk,clk\_div);

    parameter n=4;

    input clk;

    output clk\_div;

    reg [n-1:0]num;

    wire [n-1:0]nextnum;

    always@(posedge clk)

        begin

            num=nextnum;

        end

    assign nextnum=num+1;

    assign clk\_div=num[n-1];

endmodule

module debounce(pb\_debounced,pb,clk);

    output pb\_debounced;

    input pb;

    input clk;

    reg[3:0] shift\_reg;

    always@(posedge clk)

        begin

            shift\_reg[3:1]<=shift\_reg[2:0];

            shift\_reg[0]<=pb;

        end

    assign pb\_debounced=((shift\_reg==4'b1111)?1'b1:1'b0);

endmodule

module onepulse(pb\_debounced,clk,pb\_1pulse);

    input pb\_debounced;

    input clk;

    output pb\_1pulse;

    reg pb\_1pulse;

    reg pb\_debounced\_delay;

    always@(posedge clk)

        begin

            if(pb\_debounced==1'b1 & pb\_debounced\_delay==1'b0)

                begin

                    pb\_1pulse<=1'b1;

                end

            else

                begin

                    pb\_1pulse<=1'b0;

                end

            pb\_debounced\_delay<=pb\_debounced;

        end

endmodule