3η Εργαστηριακή Άσκηση Σχεδιασμός Συστημάτων VLSI

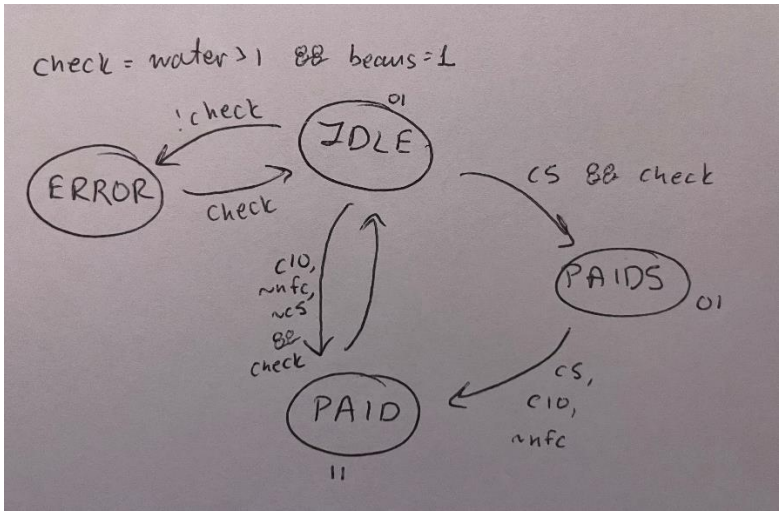
Κόλλιας Ιωάννης 1084578

Ποδηματά Θεοδώρα Παναγιώτα 1090031

4ο Έτος 2023-2024

Ερώτημα 1

Παρακάτω σας παραθέτουμε το διάγραμμα καταστάσεων.



Έχουμε 4 καταστάσεις, η αρχική κατάσταση IDLE, αν εισάγουμε 5 cents πάμε στην PAID5, αν εισάγουμε 10 cents ή ~nfc πάμε στην PAID η οποία ξανά επιστρέφει πάντα πίσω στην αρχική και αν υπάρχει κάποιο λάθος, έλλειψη νερού ή καφέ πάμε στην ERROR.

Σημειώνουμε ότι βάζουμε σε μια μεταβλητή check τον έλεγχο για επαρκή νερό και καφέ, καθώς αν κάτι από τα δύο λείπει κατευθείαν πρέπει να πάμε στην κατάσταση λάθους.

```
module vmcoffee(input c5, input c10, input nfc, input [5:0] water, input beans, input rstn, input clk, output reg coffee, output reg  error);


parameter [1:0]

IDLE=2'b00, PAID=2'b01, PAID5=2'b10,  ERROR=2'b11;

reg [1:0] state, next;

wire check;


always @(posedge clk or negedge rstn)

        if(!rstn) state <= IDLE;

        else state<=next;



assign check=(water>1&&beans)?1:0;
```

```
always @(state or c5 or c10 or nfc or water or beans)begin
        next=2'bx;
    //coffee=1'b0; error=1'b1;

        case(state)
            IDLE: if((!nfc||c10)&&check) next=PAID;
                else if(c5&&nfc&&!c10&&check) next=PAID5;
                else if(!check) next=ERROR;
                else next=IDLE;
            PAID5: if(!nfc||c10||c5) next=PAID;
                else next=PAID5;
            PAID:  next=IDLE;
            ERROR: if(check) next=IDLE;
                else next=ERROR;
        endcase
end


always @(posedge clk or negedge rstn)begin
        if(!rstn) begin
            coffee<=1'b0;
            error<=1'b1;
        end
        else begin
            coffee<=1'b0;
            error<=1'b1;
            case(next)
                IDLE,PAID5: ;
                PAID: coffee<=1'b1;
                ERROR: error<=1'b0;
            endcase
        end
end
```

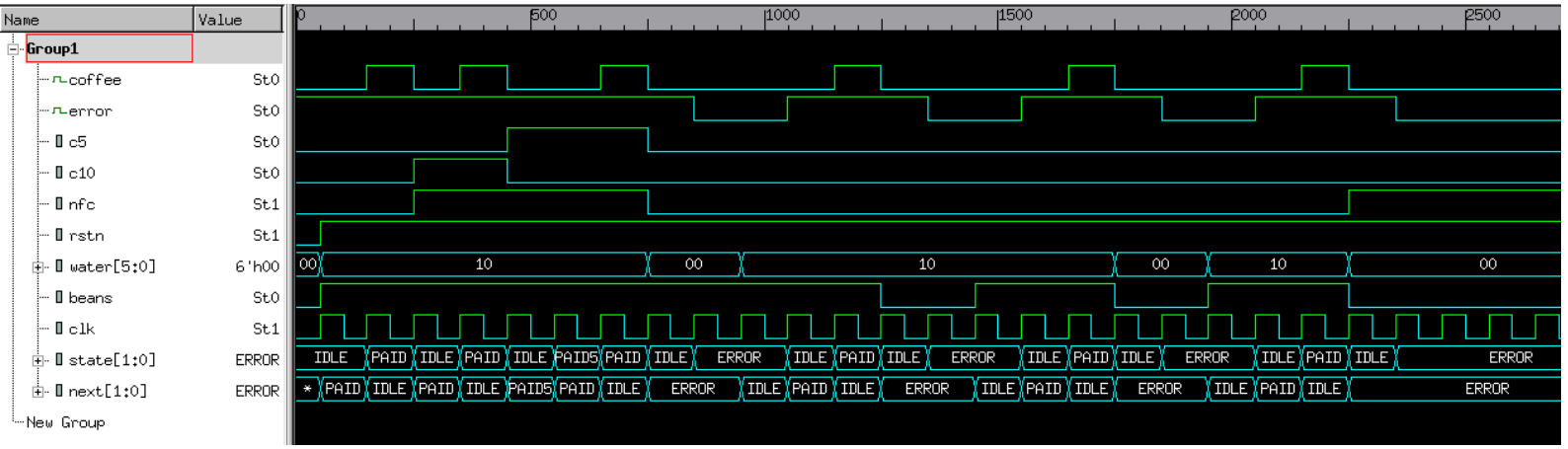
```
endmodule
```

testbench

```
module vmcoffeetb;

// Inputs
reg c5, c10, nfc, rstn;
reg [5:0] water;
reg beans;
reg clk = 0;

// Outputs
wire coffee, error;



vmcoffee dut (
    .c5(c5),
    .c10(c10),
    .nfc(nfc),
    .water(water),
    .beans(beans),
    .rstn(rstn),
    .clk(clk),
    .coffee(coffee),
    .error(error)
);

always #5 clk = ~clk;

initial begin
    rstn = 0;
    c5 = 0;
    c10 = 0;
    nfc = 0;
    water = 0;
    beans = 0;
```

```
#5;
//NFC payment, enough water, enough beans
rstn <= 1;
water <= 6'h10;
beans <= 1;
nfc <= 0;

#20;
nfc <= 1;
c10 <= 1;

#20;
c10 <= 0;
c5 <= 1;

//no water 750
#30;
c5 <= 0;
nfc <= 0;
water <= 6'h00;

#20;
water <= 6'h10;

//no beans 1250
#30;
beans <= 0;

#20;
beans <= 1;

//No water, no beans 1750
#30;
beans <= 0;
```

```
    water <= 6'h00;


    #20;
    beans <= 1;
    water <= 6'h10;


    //No money inserted, no water, no beans
    #30;
    beans <= 0;
    water <= 6'h00;
    nfc <= 1;


end


endmodule
```



Ερώτημα 2.1

Το παρακάτω κύκλωμα σύγκρισης (lockcomp), συγκρίνει έναν 16-bit αριθμό εισόδου με έναν αποθηκευμένο αριθμό 16-bit. Προσοχή, το preset και reset να γίνουν 1 πριν το ρολόι, για να μπορέσει πρώτα να διαβάσει και να αποθηκεύσει (όταν clk=1) την νέα τιμή που του δίνεται.

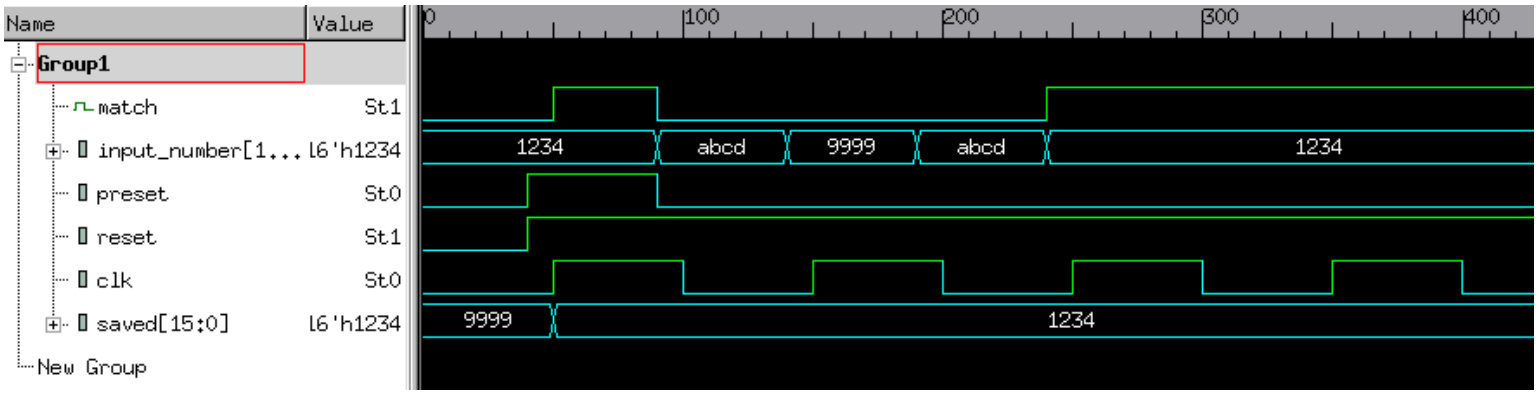
```
module lockcomp (
    input [15:0] input_number,
    input preset,
    input reset,
    input clk,
    output match
);
```

```
reg[15:0] saved;

always @(posedge clk or negedge reset) begin
   if (!reset) begin
      saved<=16'h9999;
   end
            else if (preset) begin
      saved<=input_number;
   end
            else
      saved<=saved;

end

assign match=(saved==input_number)?1:0;

endmodule
```

testbench

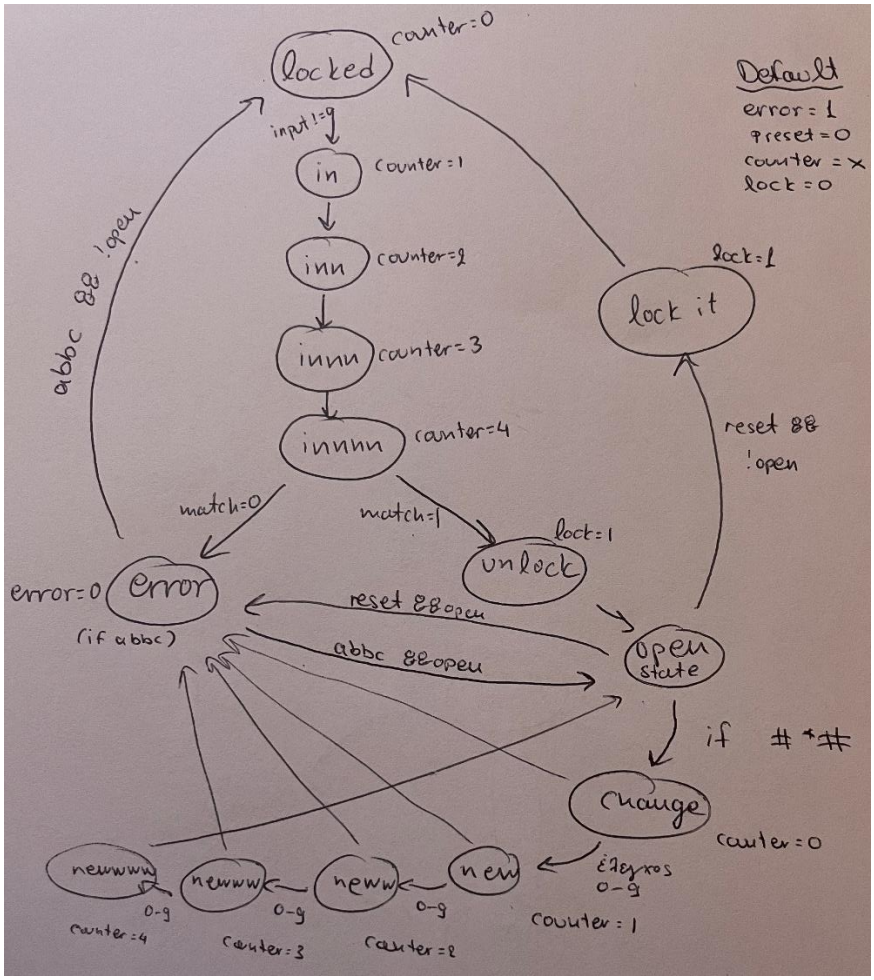
```
module lockcomp_tb;

reg [15:0] input_number;
reg preset, reset, clk;
wire match;

lockcomp dut (
   .input_number(input_number),
   .preset(preset),
   .reset(reset),
   .clk(clk),
   .match(match)
);

always #5 clk = ~clk;
```

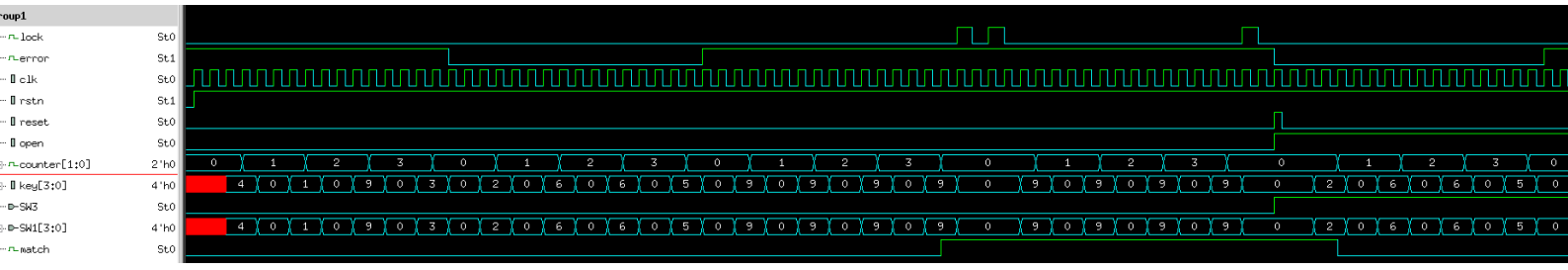
```
initial begin
   preset <= 0;
   reset <= 0;
   clk<=0;
   input_number = 16'h1234;

   #4;
   preset <= 1;
   reset <= 1;

   #5 input_number = 16'hABCD;
   preset = 0;

   #5  input_number = 16'h9999;

#5 input_number=16'hABCD;
#5 input_number=16'h1234;
end

endmodule
```

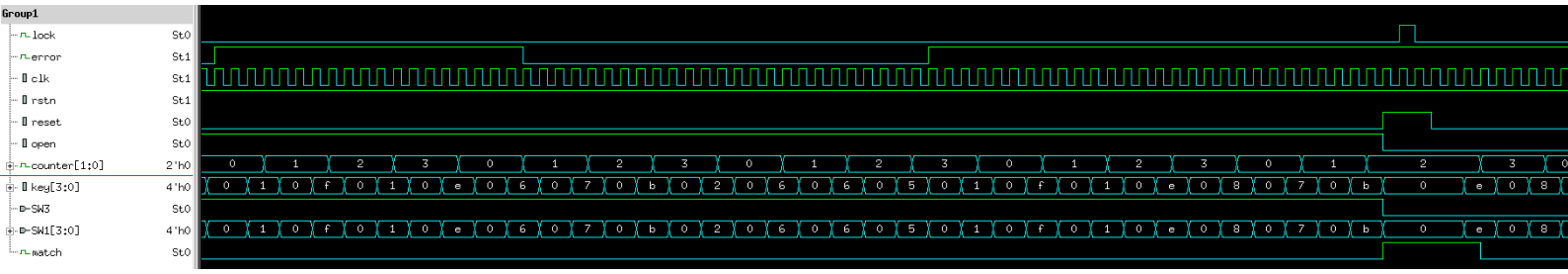


Ερώτημα 2.2

Παρακάτω σας παραθέτουμε το διάγραμμα καταστάσεων του κυκλώματος dlock. Αποτελείται από 14 καταστάσεις και έχουμε υλοποιήσει «ένα ξεχωριστό» κύκλωμα για την εισαγωγή εισόδου από το πληκτρολόγιο. Αναλυτικότερα, έπειτα από κάθε πάτημα ενός κουμπιού από το πληκτρολόγιο εισάγουμε και έναν παλμό (0) σαν καθυστέρηση.







Κώδικας:

module dlock(input reset, input clk, input rstn, input SW3, input [3:0] SW1, output reg lock, output reg error, output [1:0] counter);

```
wire [3:0] inputdata;

assign inputdata=SW1;

assign open=SW3;


reg [15:0] pass;


parameter [5:0] locked=6'd0,
in=6'd1,inn=6'd2,innn=6'd3,innnn=4,errorstate=5,unlock=9,openstate=10,lockit=11,cc=12,change=13,new=14,ne
ww=6,newww=7,newwww=8;

parameter [1:0] idle=0, numin=1, waitz=2;

wire match;

reg preset;

reg pulse;

wire [3:0] inputt;

reg[4:0] kstate, knext;

reg[5:0] state, next;

reg [1:0] counter_internal;


assign counter = counter_internal;


lockcomp compare(.input_number(pass),.preset(preset),.reset(rstn),.clk(clk),.match(match));


always @(posedge clk or negedge rstn)begin
        if(!rstn)begin
                kstate<=0;
                state<=0;
                counter_internal<=0;
                pass<=0;
        end
        else begin
        kstate<=knext;
         state <= next;
end
end


always @(kstate or inputdata)begin
```

```
        knext=2'bx;
        case(kstate)
                idle: if(inputdata) knext=numin;
                        else knext=idle;
                numin: knext=waitz;
                waitz: if(!inputdata) knext=idle;
                        else knext=waitz;
        endcase
end

assign inputt= (pulse)?inputdata:0;

always @(posedge clk)begin
        pulse<=0;
        if(knext==numin) pulse<=1;
end

always @(posedge clk)begin
        if(pulse) begin
        pass<={pass[11:0],inputdata};
        counter_internal=counter_internal+1;
        end
        else pass<=pass;
end

always @(state or reset or inputt or preset or open)begin
        next=6'bx;
        case(state)
                locked:if(inputt) next=in;
                        else next=locked;
                in:if(inputt) next=inn;
            else next=in;
                inn:if(inputt) next=innn;
            else next=inn;
                innn:if(inputt) next=innnn;
```

```
            else next=innn;
                innnn:if(match) next=unlock;
            else if(!match) next=errorstate;
                            else next=innnn;
                errorstate: if(pass==16'h2665&&!open) next=locked;
                                    else if(pass==16'h2665&&open) next=openstate;
                                    else next=errorstate;
                unlock: next=openstate;
                openstate: if(!open&&reset) next=lockit;
                                    else if(open&&reset) next=errorstate;
                                    else if(pass[11:0]==12'h1f1) next=change;
                                    else next=openstate;
                lockit: next=locked;
                change: if(inputt==1||inputt==2||inputt==5||inputt==6||inputt==15) next=errorstate;
                            else if(!inputt) next=change;
                            else next=new;
                new:if(inputt==1||inputt==2||inputt==5||inputt==6||inputt==15) next=errorstate;
            else if(!inputt) next=new;
            else next=neww;
                neww:if(inputt==1||inputt==2||inputt==5||inputt==6||inputt==15) next=errorstate;
            else if(!inputt) next=neww;
            else next=newww;
                newww:if(inputt==1||inputt==2||inputt==5||inputt==6||inputt==15) next=errorstate;
            else if(!inputt) next=newww;
            else next=newwww;
                newwww:next=openstate;
        endcase
end


always @(posedge clk or negedge rstn)begin
        if(!rstn)begin
                error<=1;
                lock<=0;
                preset<=0;
```

```
        end
        else begin
                error<=1;
                lock<=0;
                preset<=0;
                case(next)
                        errorstate:begin
                         error<=0;
                        end
                        lockit,unlock: lock<=1;
                        newwww:preset<=1;
                endcase



end
end
endmodule
```

testbench (run it ./locktest or make dlock)

```
module testdlock;


reg clk, rstn,reset,open;
reg [3:0] key;
wire [1:0] counter;
wire lock,error;



dlock thelock(.reset(reset),.clk(clk),.rstn(rstn),.SW3(open),.SW1(key),.lock(lock),.error(error),.counter(counter));


always #5 clk=~clk;


initial begin
clk=0;
rstn=0;
open=0;
reset=0;
```

```
#5 rstn=1;

repeat(4)begin
        #20 key=$random;
        #20 key=0;
end

@(negedge error);

$display("pressing wrong password");

#20 key=2;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=5;
#20 key=0;



@(posedge error);

$display("try again");
repeat(4)begin
        #20 key=4'h9;
        #20 key=4'h0;
end

@(posedge lock);

$display("unlocked");
```

```
#20 reset=1;
@(posedge lock);

$display("locked again");

reset=0;



repeat(4)begin
      #20 key=4'h9;
      #20 key=4'h0;
end

@(posedge lock);

$display("unlocked again");

#20 open=1;
reset=1;

@(negedge error);
$display("tried to lock with door open");

#5 reset=0;
#20 key=2;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=5;
#20 key=0;

@(posedge error);
```

```
$display("error resolved");

#20 key=1;
#20 key=0;
#20 key=4'hf;
#20 key=0;
#20 key=1;
#20 key=0;

$display(" new  password");

#20 key=14;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=7;
#20 key=0;
#20 key=11;
#20 key=0;

if(error==0)
$display("bad new password");

#20 key=2;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=6;
#20 key=0;
#20 key=5;
#20 key=0;
@(posedge error);
$display("error resolved");
```

```
#20 key=1;
#20 key=0;
#20 key=4'hf;
#20 key=0;
#20 key=1;
#20 key=0;

$display("new attempt of new password");

#20 key=14;
#20 key=0;
#20 key=8;
#20 key=0;
#20 key=7;
#20 key=0;
#20 key=11;
#20 key=0;

open=0;
reset=1;
@(posedge lock);
#20 reset=0;

#20 key=14;
#20 key=0;
#20 key=8;
#20 key=0;
#20 key=7;
#20 key=0;
#20 key=11;
#20 key=0;



@(posedge lock);
$finish;
```

```
end

endmodule
```