CS223 – Digital Design

Ahmet Faruk Ulutaş

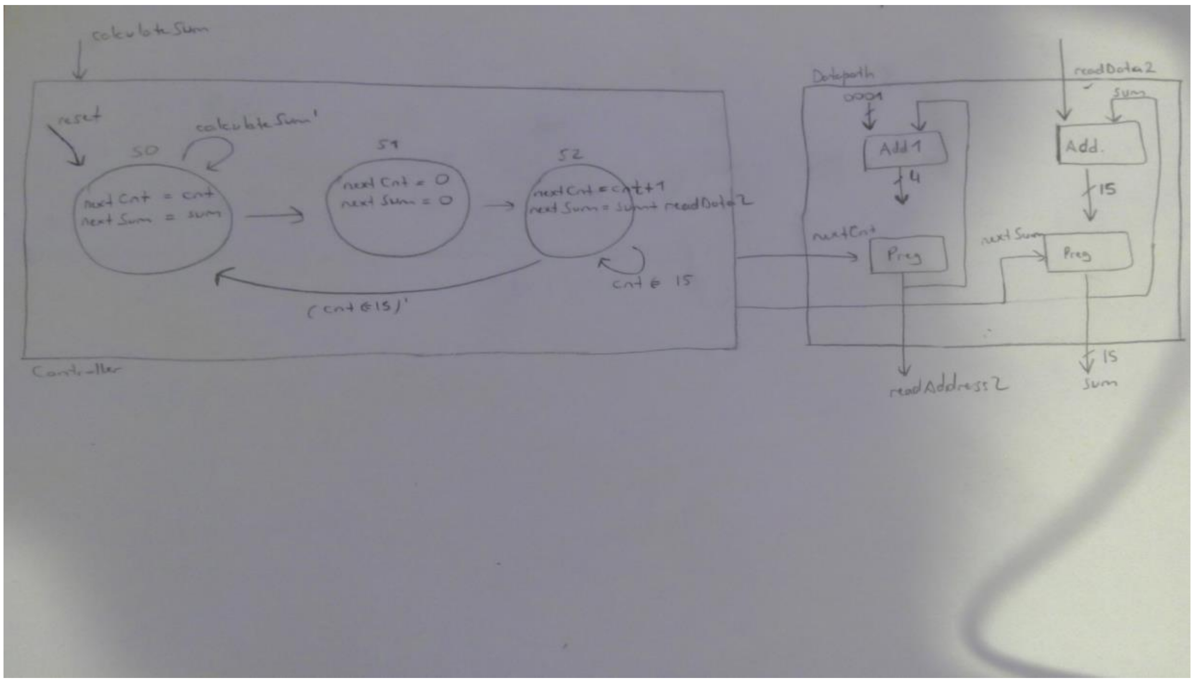
21803717

Section 1

Lab 5

Preliminary Work

A) Reduce HSLM



B) You will be given a debouncer module for your pushbuttons. Research and explain briefly why there is a need for such a circuit.

We use it to prevent the Debounce effect. This effect is that when we press the button, it creates vibration and the effect of multiple presses occurs even though we press it once.

C) Memory module System Verilog

```
module memory(input logic clk, input logic [3:0] writeAddress, input logic [7:0] writeData,
input logic writeEnable, input logic [3:0] readAddress1, input logic [3:0] readAddress2,
output logic [7:0] readData1, output logic [7:0] readData2);


   logic [7:0] mem[15:0];



   always_ff @ (posedge clk)
      if (writeEnable)
         mem[writeAddress] <= writeData;



   assign readData1 = mem[readAddress1];
   assign readData2 = mem[readAddress2];
```

```
endmodule
```

Testbench of memory module

```
module testbench_memory();

    logic clk;

    logic [3:0] writeAddress;

    logic [7:0] writeData;

    logic writeEnable;

    logic [3:0] readAddress1;

    logic [3:0] readAddress2;

    logic [7:0] readData1;

    logic [7:0] readData2;

    memory dut(clk, writeAddress, writeData, writeEnable, readAddress1, readAddress2,
readData1, readData2);

    always

    begin

    clk = 1; #5; clk = 0; #5;

    end
```

```
initial begin

   writeEnable = 1;

   writeAddress = 4'b0000;

   writeData = 8'b00010001;

   readAddress1 = 4'b0000;

   readAddress2 = 4'b0000;

   #10;

   writeAddress = 4'b0001;

   writeData = 8'b00010011;

   readAddress1 = 4'b0000;

   readAddress2 = 4'b0001;

   #10;

   writeAddress = 4'b0010;

   writeData = 8'b00010111;
```

```
readAddress1 = 4'b0001;

readAddress2 = 4'b0010;

#10;

writeAddress = 4'b0011;

writeData = 8'b01110001;

readAddress1 = 4'b0001;

readAddress2 = 4'b0011;

#10;

writeAddress = 4'b0100;

writeData = 8'b00011101;

readAddress1 = 4'b0010;

readAddress2 = 4'b0100;

#10;

writeAddress = 4'b0101;

writeData = 8'b11110001;

readAddress1 = 4'b0011;
```

```
readAddress2 = 4'b0101;

#10;

writeAddress = 4'b0110;

writeData = 8'b00010011;

readAddress1 = 4'b0101;

readAddress2 = 4'b0110;

#10;

writeAddress = 4'b0111;

writeData = 8'b01110001;

readAddress1 = 4'b0110;

readAddress2 = 4'b0111;

#10;

writeAddress = 4'b1000;

writeData = 8'b00000001;

readAddress1 = 4'b0111;
```

```
readAddress2 = 4'b1000;

#10;

writeAddress = 4'b1001;

writeData = 8'b00000011;

readAddress1 = 4'b1000;

readAddress2 = 4'b1001;

#10;

writeAddress = 4'b1010;

writeData = 8'b10010001;

readAddress1 = 4'b1001;

readAddress2 = 4'b1010;

#10;

writeAddress = 4'b1011;

writeData = 8'b11010001;

readAddress1 = 4'b1010;

readAddress2 = 4'b1011;
```

```
#10;

writeAddress = 4'b1100;

writeData = 8'b00011111;

readAddress1 = 4'b1011;

readAddress2 = 4'b1100;

#10;

writeAddress = 4'b1101;

writeData = 8'b00011001;

readAddress1 = 4'b1011;

readAddress2 = 4'b1101;

#10;

writeAddress = 4'b1110;

writeData = 8'b00110001;

readAddress1 = 4'b1101;

readAddress2 = 4'b1110;
```

```
        #10;

        writeAddress = 4'b1111;

        writeData = 8'b00110011;

        readAddress1 = 4'b1111;

        readAddress2 = 4'b1111;

        #10;

    end

endmodule
```

D)

ReduceSum Module System Verilog

```
module reduceSum(input logic clk, input logic reset, input logic calculateSum, input logic
[7:0] readData2, output logic [3:0] readAddress2, output logic [15:0] sum);

    typedef enum logic [1:0] {S0, S1, S2} statetype;

    logic [4:0] cnt;
    logic [4:0] nextCnt;
    logic [15:0] nextSum;
    logic [15:0] curSum;

    statetype [1:0] state, nextstate;

    always_ff @ (posedge clk)
```

```
if (reset)
begin
    cnt <= 0;
    curSum <= 0;
    state <= S0;



end


else
begin
    cnt <= nextCnt;


    state<= nextstate;


    curSum <= nextSum;


end


always_comb


    case(state)


        S2:


        begin


            if (cnt <= 15)


            begin
                nextstate = S2;
                nextSum = curSum + readData2;
            end

            else
                nextCnt = cnt + 1;
                nextstate = S0;
            end

        S1:
        begin

            nextCnt = 0;

            nextSum = 0;

            nextstate = S2;

        end

        S0:
        begin

            if (calculateSum)

                nextstate = S1;

            else

                nextstate = S0;
```

```
                nextCnt = cnt;

                nextSum = curSum;

            end
                default: nextstate = S0;
        endcase

    assign readAddress2 = cnt;
    assign sum = curSum;
endmodule
```

Testbench for ReduceSum Module

```
module testbench_reduceSum();
logic clk;
logic [3:0] writeAddress;
logic [7:0] writeData;
logic writeEnable;
logic calculateSum;
logic [3:0] readAddress1, readAddress2;
logic [7:0] readData1,readData2;
logic [15:0] sum;
logic reset;

memory hl(clk, writeAddress, writeData, writeEnable, readAddress1, readAddress2, readData1,
readData2);
reduceSum rs(clk, reset, calculateSum, readData2, readAddress2, sum);

always
   begin
      clk <= 1; #5;
      clk <= 0; #5;
   end
```

```
initial begin
    writeAddress=4'b0000;
    writeData=8'b00010001;
    writeEnable=1;
    calculateSum=0;
    reset=0;
    readAddress1=3'b0000;
    #10;
    writeAddress=4'b0001;
    writeData=8'b00010011;
    writeEnable=1;
    calculateSum=0;
    readAddress1=3'b0001;
    #10;
    writeAddress=4'b0010;
    writeData=8'b00011111;
    writeEnable=1;
    calculateSum=0;
    readAddress1=3'b0010;
    #10;
    writeAddress=4'b0000;
    writeData=8'b00000000;
    writeEnable=0;
    calculateSum=1;
    readAddress1=3'b0000;
    #50;
    writeAddress=4'b0000;
    writeData=8'b00010000;
    writeEnable=0;
    calculateSum=0;
    readAddress1=3'b0000;
    #10;
    writeAddress=4'b0000;
```

```
writeData=8'b00010001;
writeEnable=0;
calculateSum=0;
readAddress1=3'b0000;
#10;writeAddress=4'b0011;
writeData=8'b11010001;
writeEnable=1;
calculateSum=0;
readAddress1=3'b0011;
#10;
writeAddress=4'b0100;
writeData=8'b00011101;
writeEnable=1;calculateSum=0;
readAddress1=3'b0100;
#10;
writeAddress=4'b0000;
writeData=8'b00000000;
writeEnable=0;
calculateSum=1;
readAddress1=3'b0000;
#90;
writeAddress=4'b0000;
writeData=8'b00000000;
writeEnable=0;
calculateSum=0;
readAddress1=3'b0000;
reset = 1;
#10;
writeAddress=4'b0000;
writeData=8'b00000000;
writeEnable=0;
calculateSum=0;
readAddress1=3'b0000;
reset = 0;
```

```
    #10;
    writeAddress=4'b0101;
    writeData=8'b00110001;
    writeEnable=1;
    calculateSum=0;
    readAddress1=3'b0000;
    #10;
    writeAddress=4'b0000;
    writeData=8'b00000000;
    writeEnable=0;
    calculateSum=1;
    readAddress1=3'b0000;
    #10;
    end
endmodule
```

E)

TopDesign System Verilog

```
module reduceSumTopDesign(input logic clk, input logic reset, input logic [3:0] writeAddress, input logic
[7:0] writeData, input logic writeEnable, input logic calculateSum, input logic displayNext, input logic
displayPrev, output logic [15:0] totalSum, output logic [6:0] seg, output logic dp, output logic [3:0] an);
logic [7:0] readData1, readData2;
logic [3:0] readAddress1, readAddress2;
logic [15:0] sum;
logic pulse1, pulse2, pulse3, pulse4, pulse5;
initial begin
    readData1 = 0;
    readAddress1 = 0;
    readData2 = 0;
    readAddress2 = 0;
    end

    debounce de1(clk, writeEnable, pulse1);
    debounce de2(clk, calculateSum, pulse2);
    debounce de3(clk, displayNext, pulse3);
    debounce de4(clk, displayPrev, pulse4);
    debounce de5(clk, reset, pulse5);

    always_ff @ (posedge clk)

        if (pulse4)
        begin
            if (readAddress1 > 0)
                readAddress1 <= readAddress1 -1;
            else
                readAddress1 <= 15;
        end

        else if (pulse3)
        begin
            if (readAddress1 < 16)
                readAddress1 <= readAddress1 + 1;
            else
                readAddress1 <= 4'b0000;
        end
        else
            readAddress1 <= readAddress1;

   memory dut(clk,writeAddress,writeData,pulse1,readAddress1,readAddress2,readData1,readData2);
   reduceSum rs(clk,pulse5,pulse2,readData1,readAddress2, sum);
   SevSeg_4digit se(clk, readAddress1, 17, readData1[7:4], readData1[3:0], seg, dp, an);

   assign totalSum = sum;
endmodule
```