Course Code: CS223

Course Name: Digital Design **Section:** 4

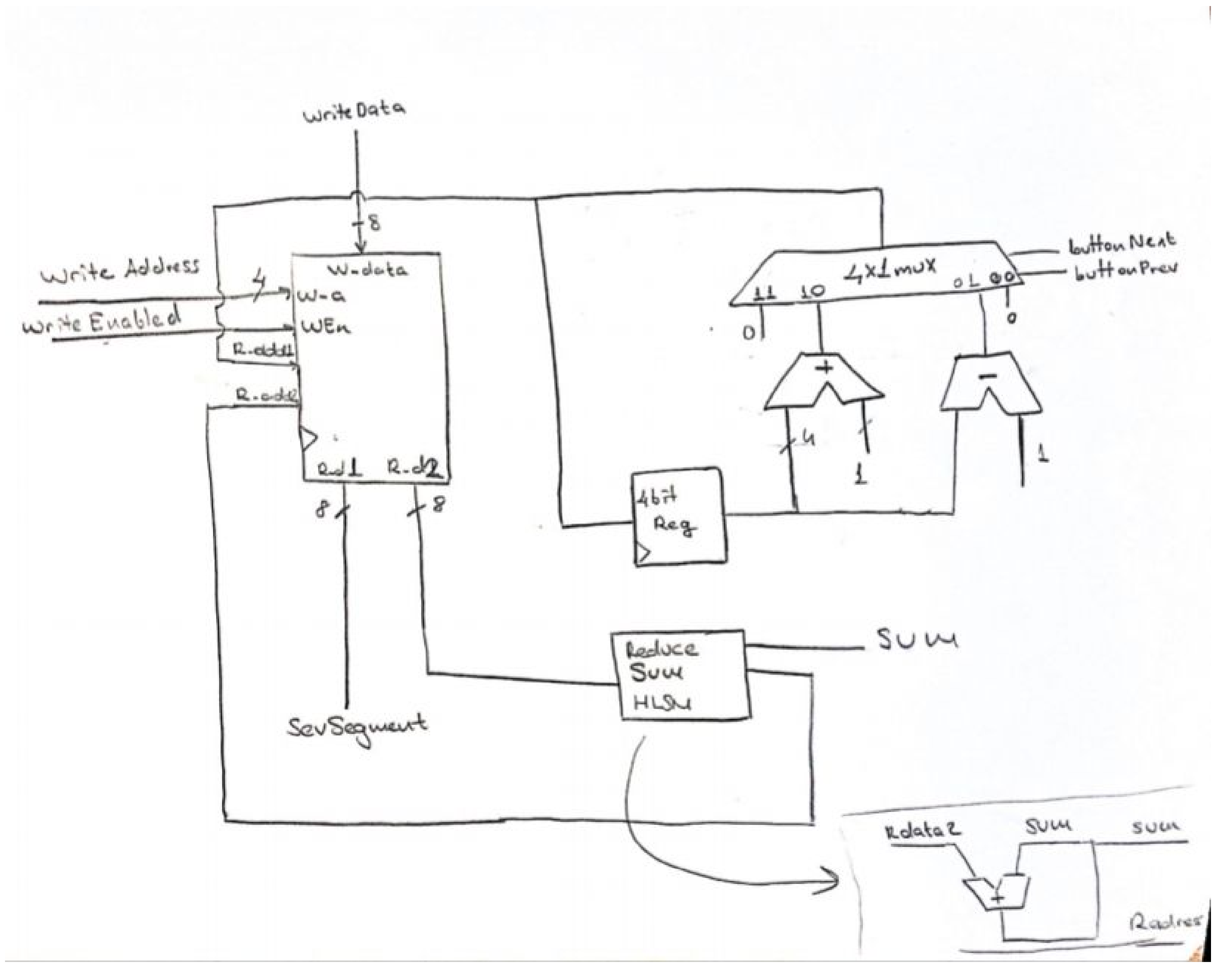
Lab 5

Name: Berk Saltuk Yılmaz **ID:** 21903419

Date: 04.12.2020

Part (a):

Controller/datapath diagram for HLSM:

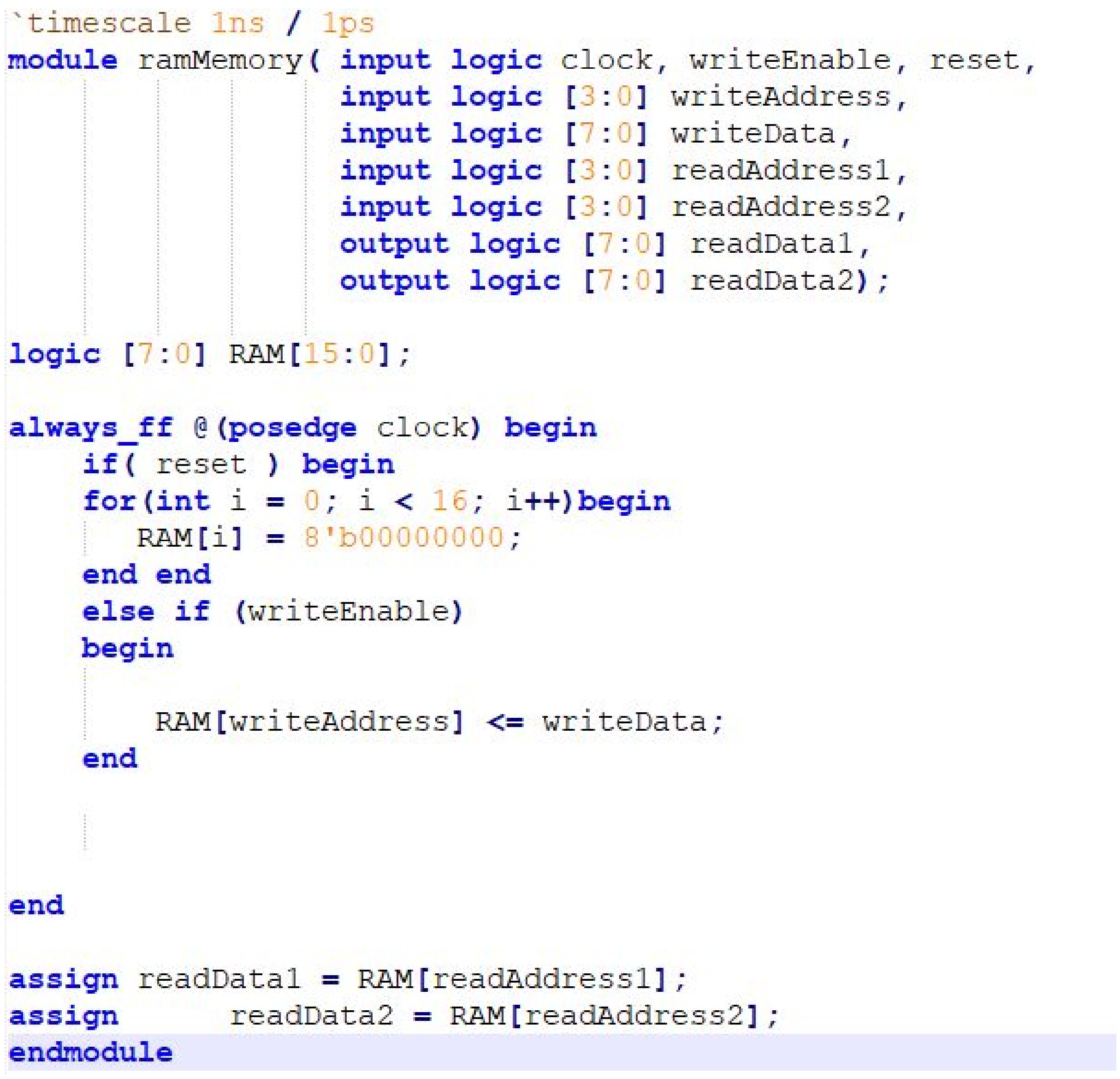


Part (b):

The debouncer module aims to provide a smooth transition between the buttons pushed and the output to be displayed. When the button is pressed in the first place an electrical break might occur. The debouncer module removes these breaks and everything keeps working properly.

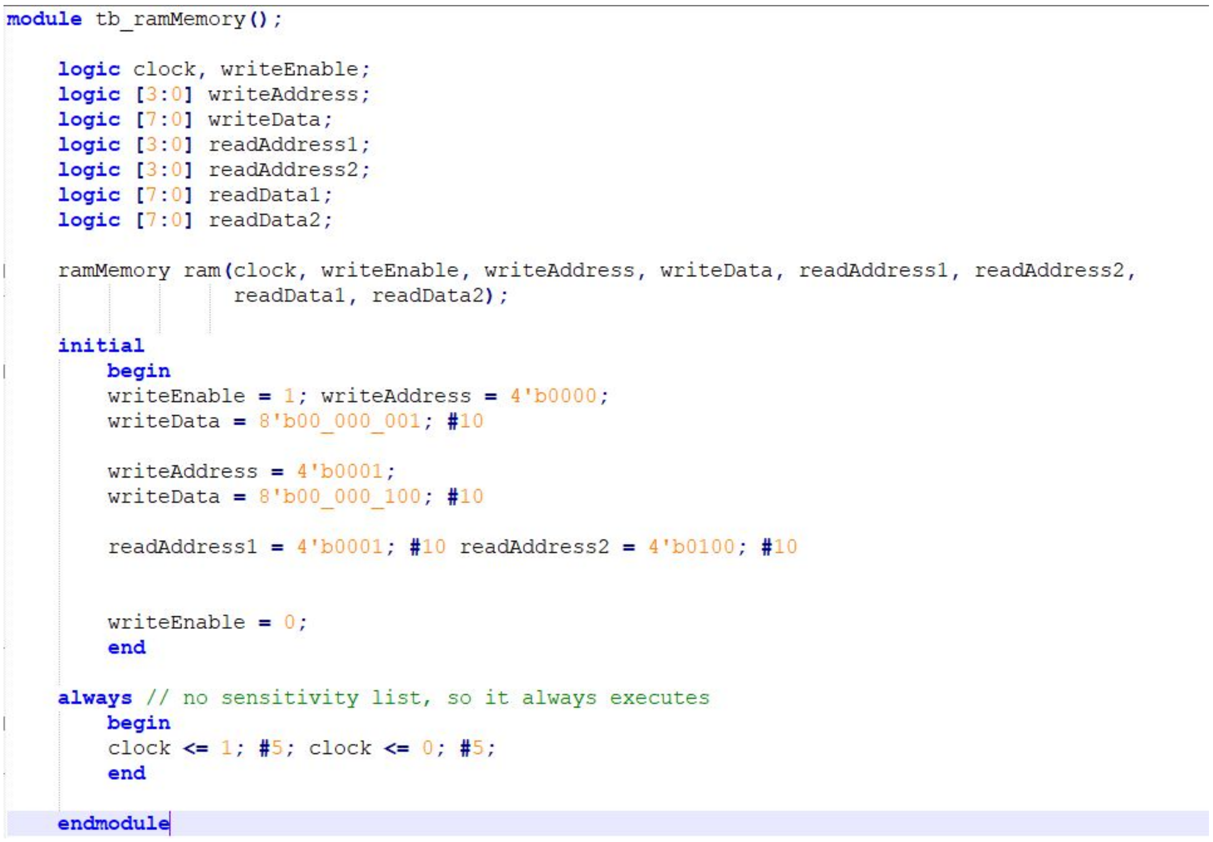
Part (c):

SystemVerilog Code for the memory module:



```
`timescale 1ns / 1ps
module ramMemory( input logic clock, writeEnable, reset,
                  input logic [3:0] writeAddress,
                  input logic [7:0] writeData,
                  input logic [3:0] readAddress1,
                  input logic [3:0] readAddress2,
                  output logic [7:0] readData1,
                  output logic [7:0] readData2);

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

always_ff @(posedge clock) begin
    if( reset ) begin
    for(int i = 0; i < 16; i++)begin
       RAM[i] = 8'b00000000;
    end end
    else if (writeEnable)
    begin

        RAM[writeAddress] <= writeData;
    end


end

assign readData1 = RAM[readAddress1];
assign      readData2 = RAM[readAddress2];
endmodule
```

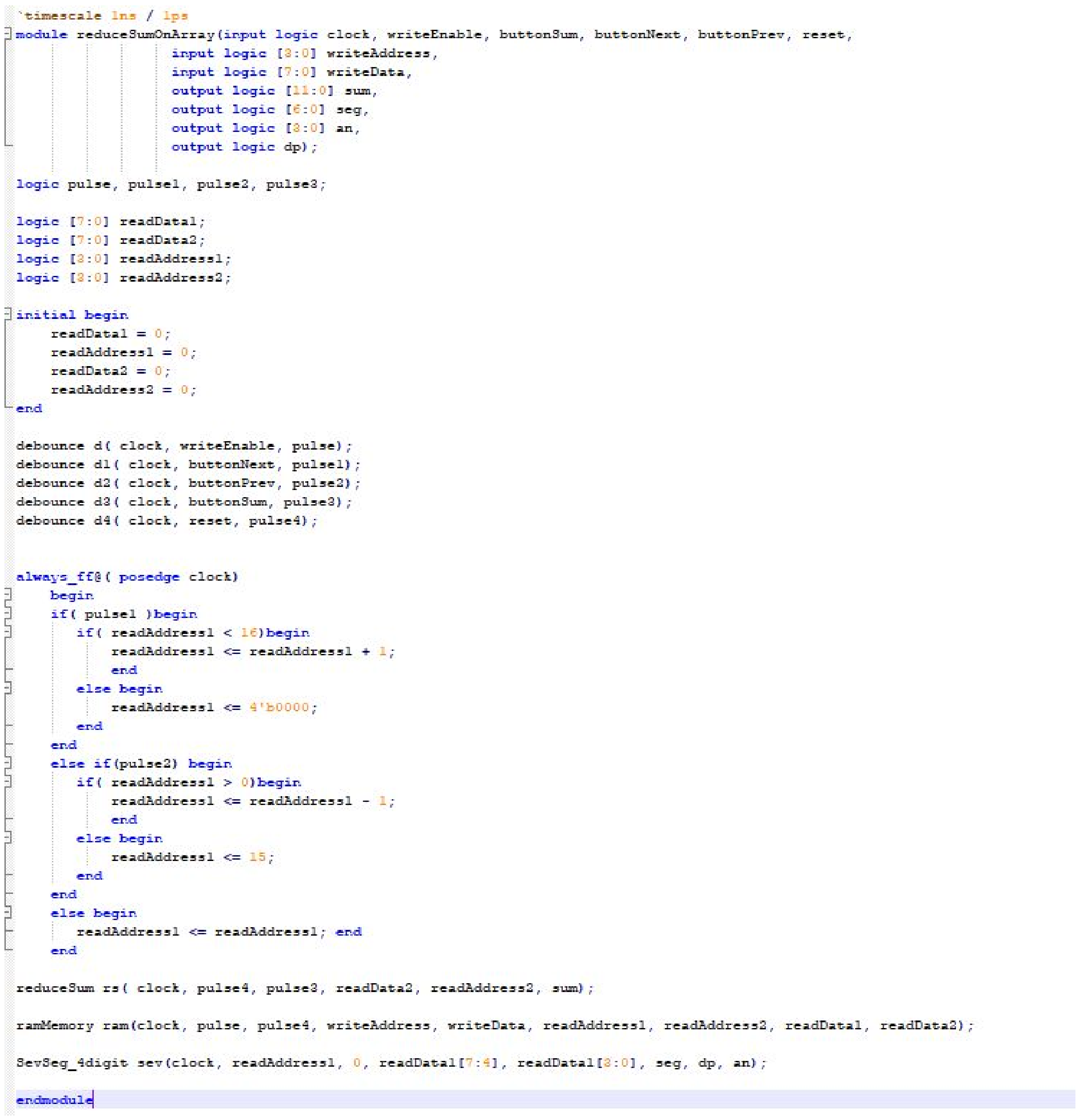
Testbench for it;



```
module tb_ramMemory();

    logic clock, writeEnable;
    logic [3:0] writeAddress;
    logic [7:0] writeData;
    logic [3:0] readAddress1;
    logic [3:0] readAddress2;
    logic [7:0] readData1;
    logic [7:0] readData2;

    ramMemory ram(clock, writeEnable, writeAddress, writeData, readAddress1, readAddress2,
                  readData1, readData2);

    initial
        begin
        writeEnable = 1; writeAddress = 4'b0000;
        writeData = 8'b00_000_001; #10

        writeAddress = 4'b0001;
        writeData = 8'b00_000_100; #10

        readAddress1 = 4'b0001; #10 readAddress2 = 4'b0100; #10


        writeEnable = 0;
        end

    always // no sensitivity list, so it always executes
        begin
        clock <= 1; #5; clock <= 0; #5;
        end

    endmodule
```

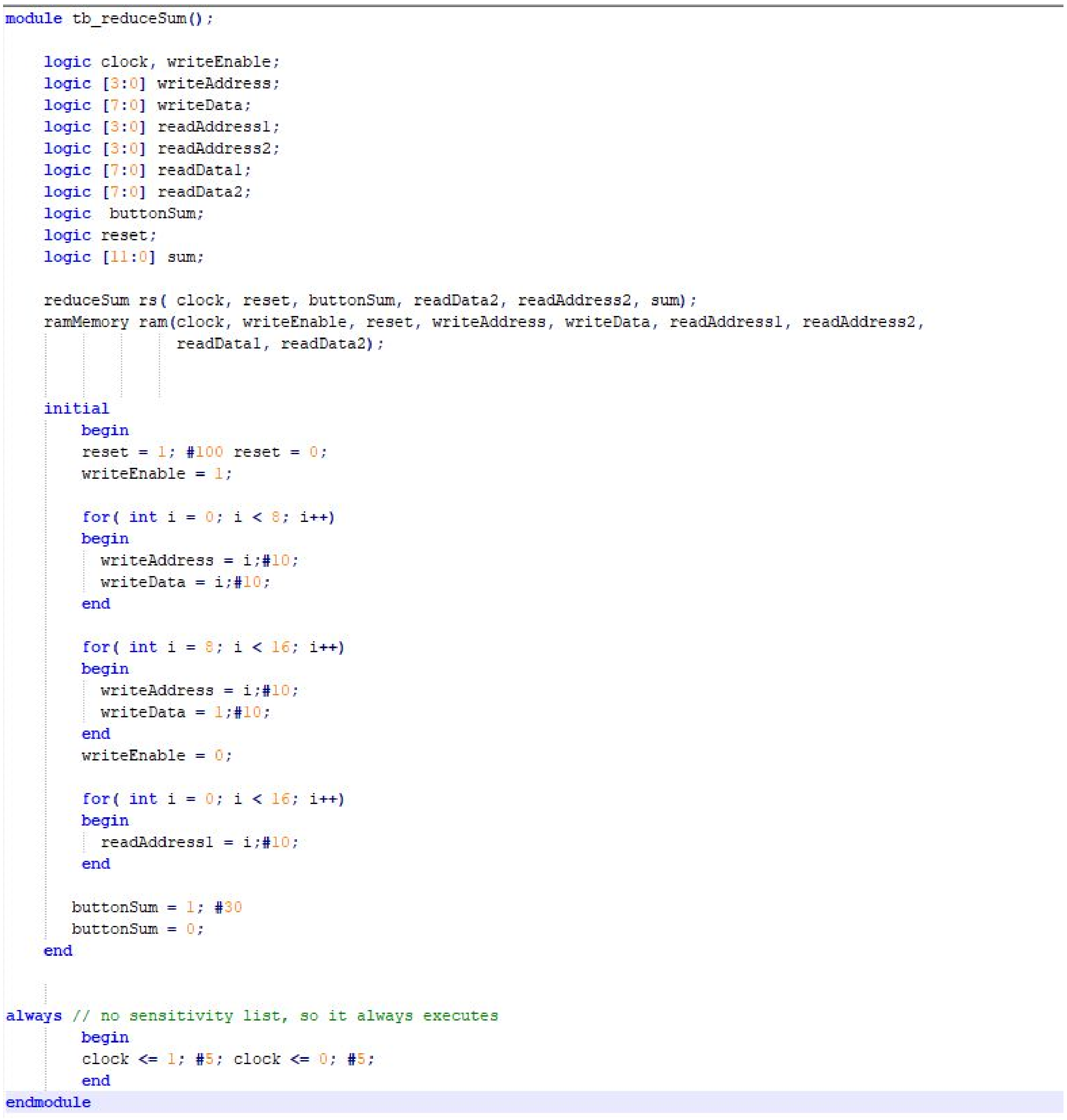
Part (d):

SystemVerilog Code for the ReduceSum Module:



```
`timescale 1ns / 1ps
module reduceSumOnArray(input logic clock, writeEnable, buttonSum, buttonNext, buttonPrev, reset,
                        input logic [3:0] writeAddress,
                        input logic [7:0] writeData,
                        output logic [11:0] sum,
                        output logic [6:0] seg,
                        output logic [3:0] an,
                        output logic dp);

logic pulse, pulse1, pulse2, pulse3;

logic [7:0] readData1;
logic [7:0] readData2;
logic [3:0] readAddress1;
logic [3:0] readAddress2;

initial begin
    readData1 = 0;
    readAddress1 = 0;
    readData2 = 0;
    readAddress2 = 0;
end

debounce d( clock, writeEnable, pulse);
debounce d1( clock, buttonNext, pulse1);
debounce d2( clock, buttonPrev, pulse2);
debounce d3( clock, buttonSum, pulse3);
debounce d4( clock, reset, pulse4);


always_ff@( posedge clock)
    begin
    if( pulse1 )begin
       if( readAddress1 < 16)begin
           readAddress1 <= readAddress1 + 1;
           end
       else begin
           readAddress1 <= 4'b0000;
       end
    end
    else if(pulse2) begin
       if( readAddress1 > 0)begin
           readAddress1 <= readAddress1 - 1;
           end
       else begin
           readAddress1 <= 15;
       end
    end
    else begin
       readAddress1 <= readAddress1; end
    end

reduceSum rs( clock, pulse4, pulse3, readData2, readAddress2, sum);

ramMemory ram(clock, pulse, pulse4, writeAddress, writeData, readAddress1, readAddress2, readData1, readData2);

SevSeg_4digit sev(clock, readAddress1, 0, readData1[7:4], readData1[3:0], seg, dp, an);

endmodule
```

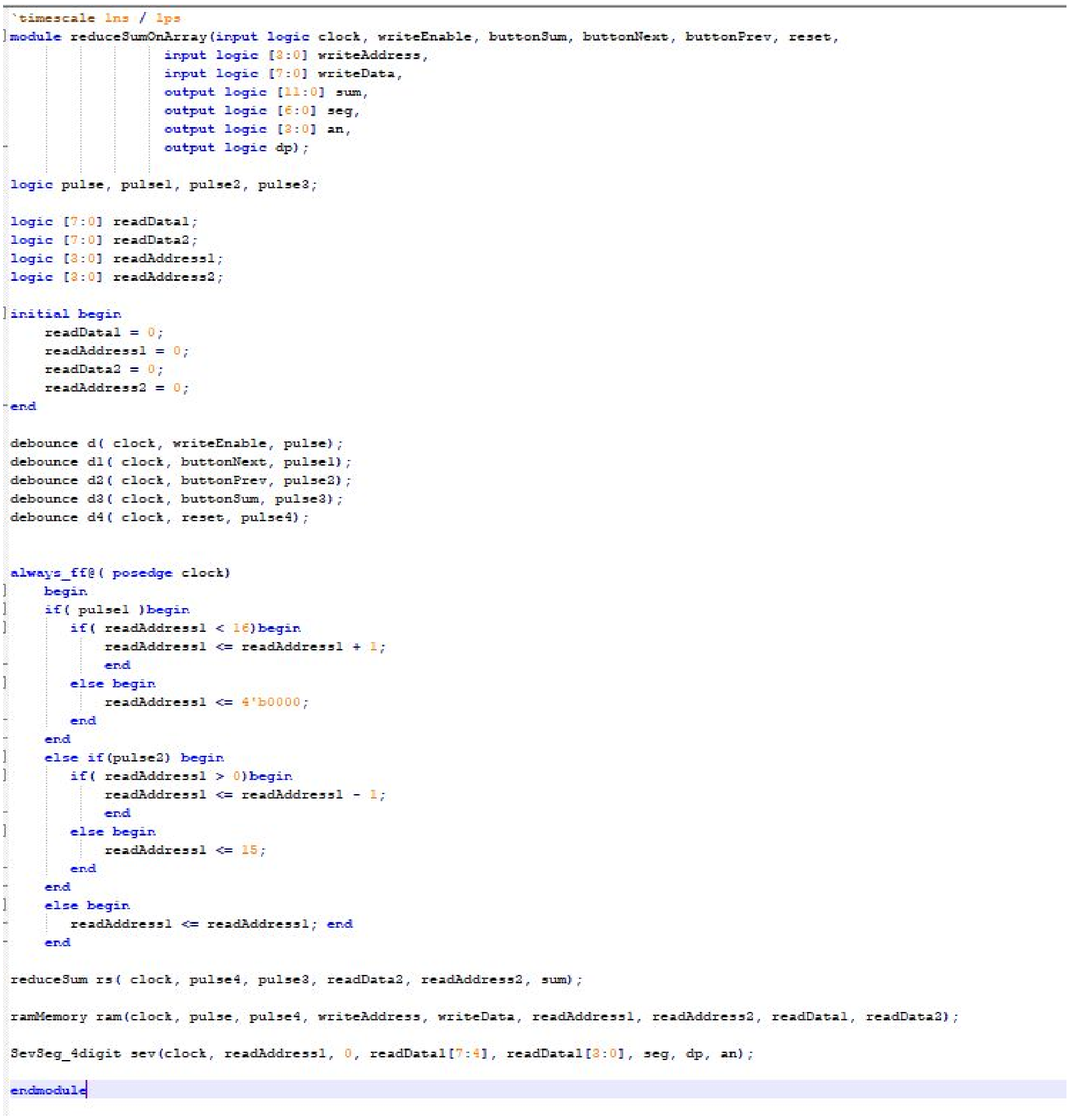
Testbench for it:



```
module tb_reduceSum();

    logic clock, writeEnable;
    logic [3:0] writeAddress;
    logic [7:0] writeData;
    logic [3:0] readAddress1;
    logic [3:0] readAddress2;
    logic [7:0] readData1;
    logic [7:0] readData2;
    logic  buttonSum;
    logic reset;
    logic [11:0] sum;

    reduceSum rs( clock, reset, buttonSum, readData2, readAddress2, sum);
    ramMemory ram(clock, writeEnable, reset, writeAddress, writeData, readAddress1, readAddress2,
                  readData1, readData2);


    initial
        begin
        reset = 1; #100 reset = 0;
        writeEnable = 1;

        for( int i = 0; i < 8; i++)
        begin
          writeAddress = i;#10;
          writeData = i;#10;
        end

        for( int i = 8; i < 16; i++)
        begin
          writeAddress = i;#10;
          writeData = 1;#10;
        end
        writeEnable = 0;

        for( int i = 0; i < 16; i++)
        begin
          readAddress1 = i;#10;
        end

       buttonSum = 1; #30
       buttonSum = 0;
    end


always // no sensitivity list, so it always executes
        begin
        clock <= 1; #5; clock <= 0; #5;
        end
endmodule
```

Part (e):

SystemVerilog Code for the top design which also includes the Seven Segment Display and debouncer modules:



```
`timescale 1ns / 1ps
module reduceSumOnArray(input logic clock, writeEnable, buttonSum, buttonNext, buttonPrev, reset,
                        input logic [3:0] writeAddress,
                        input logic [7:0] writeData,
                        output logic [11:0] sum,
                        output logic [6:0] seg,
                        output logic [3:0] an,
                        output logic dp);

logic pulse, pulse1, pulse2, pulse3;

logic [7:0] readData1;
logic [7:0] readData2;
logic [3:0] readAddress1;
logic [3:0] readAddress2;

initial begin
    readData1 = 0;
    readAddress1 = 0;
    readData2 = 0;
    readAddress2 = 0;
end

debounce d( clock, writeEnable, pulse);
debounce d1( clock, buttonNext, pulse1);
debounce d2( clock, buttonPrev, pulse2);
debounce d3( clock, buttonSum, pulse3);
debounce d4( clock, reset, pulse4);


always_ff@( posedge clock)
    begin
    if( pulse1 )begin
        if( readAddress1 < 16)begin
            readAddress1 <= readAddress1 + 1;
            end
        else begin
            readAddress1 <= 4'b0000;
        end
    end
    else if(pulse2) begin
        if( readAddress1 > 0)begin
            readAddress1 <= readAddress1 - 1;
            end
        else begin
            readAddress1 <= 15;
        end
    end
    else begin
        readAddress1 <= readAddress1; end
    end

reduceSum rs( clock, pulse4, pulse3, readData2, readAddress2, sum);

ramMemory ram(clock, pulse, pulse4, writeAddress, writeData, readAddress1, readAddress2, readData1, readData2);

SevSeg_4digit sev(clock, readAddress1, 0, readData1[7:4], readData1[3:0], seg, dp, an);

endmodule
```