به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

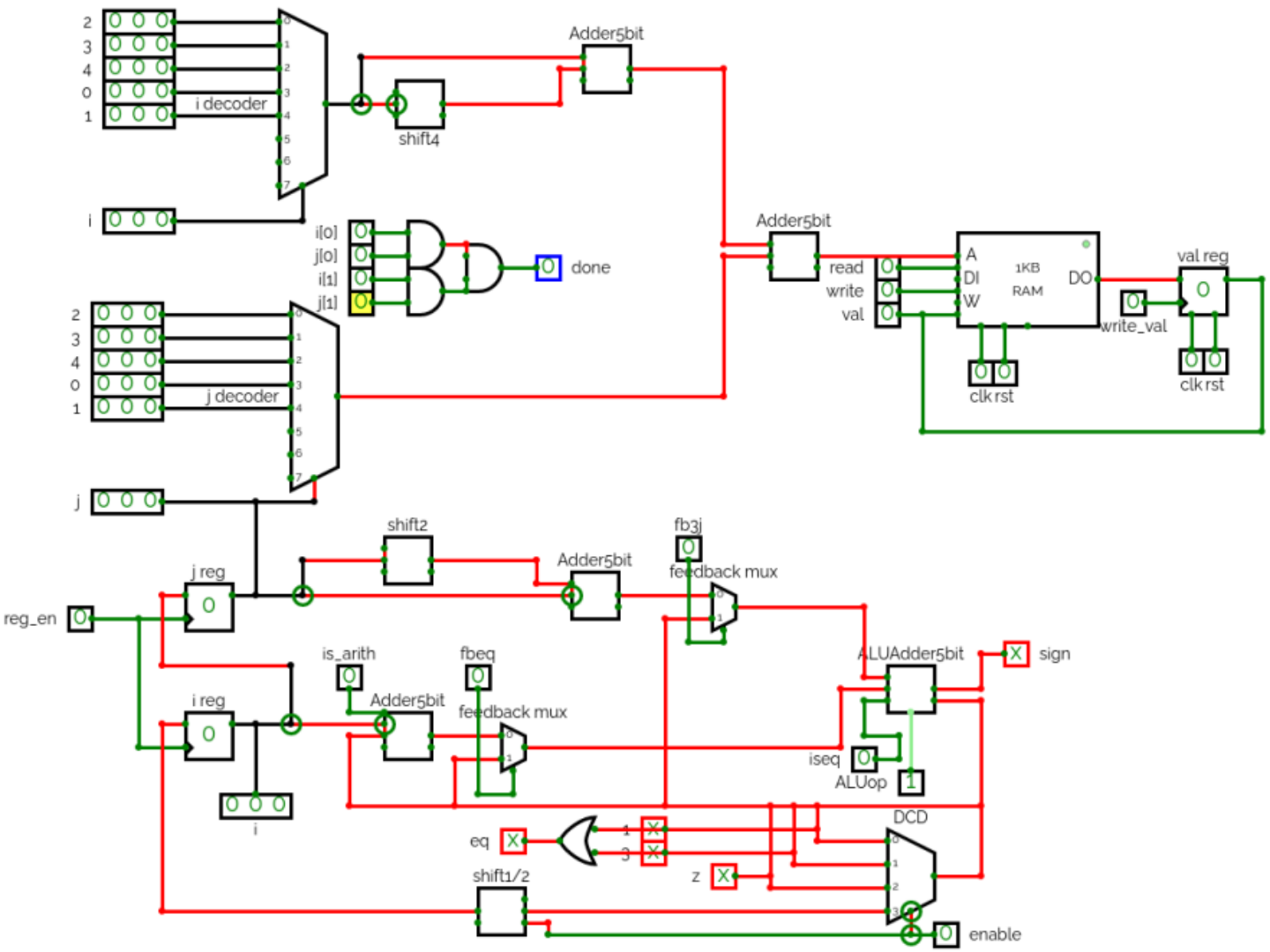
طراحی کامپیوتری سیستم دیجیتال

تمرین دستی۳

نام و نام خانوادگی	آرین سلطانی-ریحانه احمدپور
شماره دانشجویی	810198494-810198558
تاریخ ارسال گزارش	

فهرست گزارش سؤالات

دیتاپث:



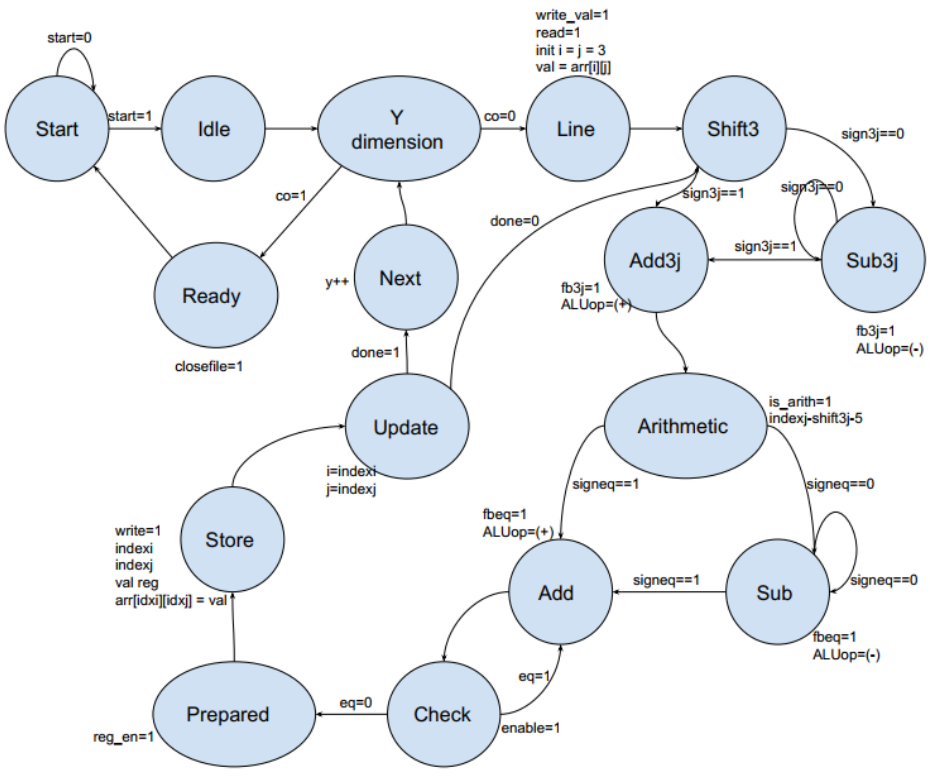

```
`timescale 1ps/1ps
module FDatapath ( // fpga implement of datapath
    clk,
    rst,
    IJen,
    ALUop,
    read,
    write,
    initLine,
    line,
    writeVal,
    IJregen,
    fb3j,
    fbeq,
    isArith,
```

```
    enable,
    waitCalNexti,
    writeMemReg,
    ldTillPositive,
    update,//enable for updating i,j after being checked and update "i"

    sign3j,
    signeq,
    done,
    sign,
    eq,
    mem,
    firstread,
    ok
);
    parameter size = 5;
    parameter memsize = 25;

    parameter initValIJ = 3;

    input clk, rst, firstread;
    input IJen, ALUop, read, write, initLine, writeMemReg;
    input writeVal, IJregen, fbeq, fb3j, isArith, enable, update, waitCalNexti,
ldTillPositive, ok;
    input [memsize-1:0]line;
    output [24:0]mem;

    output sign3j, signeq, done, sign, eq;

    wire [2:0]i;
    wire [2:0]j;
    wire [4:0]iMult4;
    wire [2:0]iReg;
    wire [2:0]iRegSaved;
    wire [2:0]jReg;
    wire [2:0]jRegSaved;
    wire [4:0]iMult5;
    wire [4:0]memIdx;
    wire [4:0]iMult2;
    wire [4:0]iMult3;
    wire [4:0]lastIndex;
    wire [4:0]lastIndexSaved;
    wire [4:0]iNextMult2;
    wire [4:0]iNextPos;
    wire [4:0]iNextPosSaved;
    wire [4:0]iNextPosAdd5;
    wire [2:0]iAtLast;
    wire [2:0]convertedI;
```

```
    wire [2:0]convertedJ;
    wire [4:0]memIdxOut;
    wire [4:0]memIdxOutSaved;
    wire [4:0]memInp;


    wire newVal;
    wire regVal;
    wire regValSaved;

    C2 #(3) newI(.D0({1'b0, 1'b1, jReg[0]}),.D1({3'b0}),.D2({~jReg[0], 1'b0,
1'b0}),.D3({3'b001}),.A1(jReg[2]),.B1(jReg[1]),.A0(~jReg[0]),.B0(jReg[2]),.out(convertedI));
    C2 #(3) newJ(.D0({1'b0, 1'b1, iReg[0]}),.D1({3'b0}),.D2({~iReg[0], 1'b0,
1'b0}),.D3({3'b001}),.A1(iReg[2]),.B1(iReg[1]),.A0(~iReg[0]),.B0(iReg[2]),.out(convertedJ));

    C2Adder #(5) multiplyI5(.i1({2'b00, convertedI}), .i2({convertedI, 2'b00}), .o(iMult5));
    C2Adder #(5) indexAdder(.i1(iMult5), .i2({2'b00, convertedJ}), .o(memIdx));

    assign memIdxOutSaved = memIdxOut;
    S2 #(5)
indexMemReg(.D0(5'b0),.D1(5'b0),.D2(memIdxOutSaved),.D3(memIdx),.A1(1'b1),.B1(1'b1),.A0(write
MemReg),.B0(writeMemReg),.CLR(rst),.clk(clk),.out(memIdxOut));

    assign memInp = write ? memIdxOut: memIdx;

    MemoryBlock #(5,25) MB(.clk(clk), .rst(rst), .init(initLine), .line(line),
        .index(memInp), .val(regVal), .write(write), .read(read), .out(newVal), .mem(mem),
.firstread(firstread), .ok(ok));

    assign regValSaved = regVal;
    S2 #(1)
valRegister(.D0(5'b0),.D1(5'b0),.D2(regValSaved),.D3(newVal),.A1(1'b1),.B1(1'b1),.A0(writeVal
),.B0(writeVal),.CLR(rst),.clk(clk),.out(regVal));

    assign jRegSaved = jReg;
    S2 #(5)
JRegister(.D0(5'b0),.D1(5'b0),.D2(jRegSaved),.D3(j),.A1(1'b1),.B1(1'b1),.A0(IJregen),.B0(IJre
gen),.CLR(rst),.clk(clk),.out(jReg));

    C2Adder #(5) multiplyI3(.i1({1'b0, iReg, 1'b0}), .i2({2'b00, iReg}), .o(iMult3));

    assign iNextPosSaved = iNextPos;
    S2 #(5)
regTillPositive(.D0(5'd0),.D1(5'd0),.D2(iNextPosSaved),.D3(iNextPosAdd5),.A1(1'b1),.B1(1'b1),
.A0(ldTillPositive),.B0(ldTillPositive),.CLR(rst),.clk(clk),.out(iNextPos));

    assign sign = iNextPosAdd5[4];
```

```
    assign iNextPosAdd5 = (waitCalNexti) ? (iNextPos + 5'b00101): iNextMult2;

    assign lastIndexSaved = lastIndex;
    S2 #(5)
registerLastIndex(.D0(5'b0),.D1(5'b0),.D2(lastIndexSaved),.D3(memIdx),.A1(1'b1),.B1(1'b1),.A0
(ld_index),.B0(ld_index),.CLR(rst),.clk(clk),.out(lastIndex));

    assign iRegSaved = iReg;
    S2 #(3)
IRegister(.D0(3'b0),.D1(3'b0),.D2(iRegSaved),.D3(i),.A1(1'b1),.B1(1'b1),.A0(IJregen),.B0(IJre
gen),.CLR(rst),.clk(clk),.out(iReg));

    assign iAtLast = (iNextPos == 5'b00000) ? 3'b000: (iNextPos == 5'b00001) ?
        3'b011: (iNextPos == 5'b00010) ? 3'b001: (iNextPos == 5'b00011) ? 3'b100:
        3'b010;

    C2Adder #(5) twiceNextI(.i1({2'b00, jReg}), .i2(~iMult3 + 1), .o(iNextMult2));

    assign i = IJen ? 3'b011: update ? iAtLast : i;
    assign j = IJen ? 3'b011: update ? iReg : j;

    assign done = iReg[0] & iReg[1] & jReg[0] & jReg[1];

endmodule
```

کنترلر:



```
`timescale 1ps/1ps
module Controller (
    clk,
    rst,
    start,

    sign3j,
    signeq,
    done,
    sign,
    eq,

    waitCalNexti,
    writeMemReg,

    IJen,
    ALUop,
    read,
    write,
    initLine,
    line,
    writeVal,
    IJregen,
    fb3j,
    fbeq,
    isArith,
    enable,
    update,
    readLine,
    ldTillPositive,
    count,
    firstread,
    ok
);
    parameter size = 5;
    parameter memsize = 25;

    input clk, rst;
    input start, sign3j, signeq, done, sign, eq;
    input [5:0]count;
    input [memsize-1:0]line;

    output reg ldTillPositive;
    output reg waitCalNexti;
    output reg fb3j;
    output readLine;
    output ok, firstread, writeMemReg, IJen, ALUop, read, write, initLine;
    output update, enable, isArith, fbeq, IJregen, writeVal;
```

```
    parameter [3:0]
        Start = 4'd0,       //0000
        Idle = 4'd1,        //0001
        Ydimension = 4'd2,  //0010
        Line = 4'd3,        //0011
        Shift3 = 4'd4,      //0100
        Sub3j = 4'd5,       //0101
        Add3j = 4'd6,       //0110
        Arithmetic = 4'd7,  //0111
        Sub = 4'd8,         //1000
        Add = 4'd9,         //1001
        Check = 4'd10,      //1010
        Prepared = 4'd11,   //1011
        Store = 4'd12,      //1100
        Updater = 4'd13,    //1101
        Next = 4'd14,       //1110
        Ready = 4'd15;      //1111

    reg first = 0;
    reg [5:0]loadInit = 0;
    wire [5:0]prevCounter, currCounter;
    wire [5:0]count2, newCount2;
    wire coutCount, coutCount2;
    wire valBitxx01, valBitxx00, valBitxx10, xorBit0and1, andBit0and1;
    wire enCount, loadCount;
    wire [5:0] sig;
    wire tmp;

    wire [3:0] ps, ns;

    S2 #(6)
update_counter_s2(.D0(prevCounter),.D1(currCounter),.D2(loadInit),.D3(loadInit),.A1(loadCount
),.B1(loadCount),.A0(enCount),.B0(enCount),.CLR(rst),.clk(clk),.out(prevCounter));
    C2Adder #(6) increase_counter_c2(.i1(prevCounter), .i2(6'd1), .o({coutCount,
currCounter}));

    S2 #(6)
update_counte2_s2(.D0(count2),.D1(newCount2),.D2(loadInit),.D3(loadInit),.A1(rst),.B1(rst),.A
0(ps[0]&ps[1]&~ps[2]&~ps[3]),.B0(ps[0]&ps[1]&~ps[2]&~ps[3]),.CLR(rst),.clk(clk),.out(count2))
; //A0(ps[0]&ps[1]&~ps[2]&~ps[3]) means it is line state and count2 needs to update
    C2Adder #(6) increase_counter2_c2(.i1(count2), .i2(6'd1), .o({coutCount2, newCount2}));

    assign valBitxx10 = ps[1]&(~ps[0]);
    assign valBitxx01 = (~ps[1])&ps[0];
    assign valBitxx00 = (~ps[1])&(~ps[0]);
    assign xorBit0and1 = ps[0]^ps[1];
    assign andBit0and1 = ps[0]&ps[1];
```

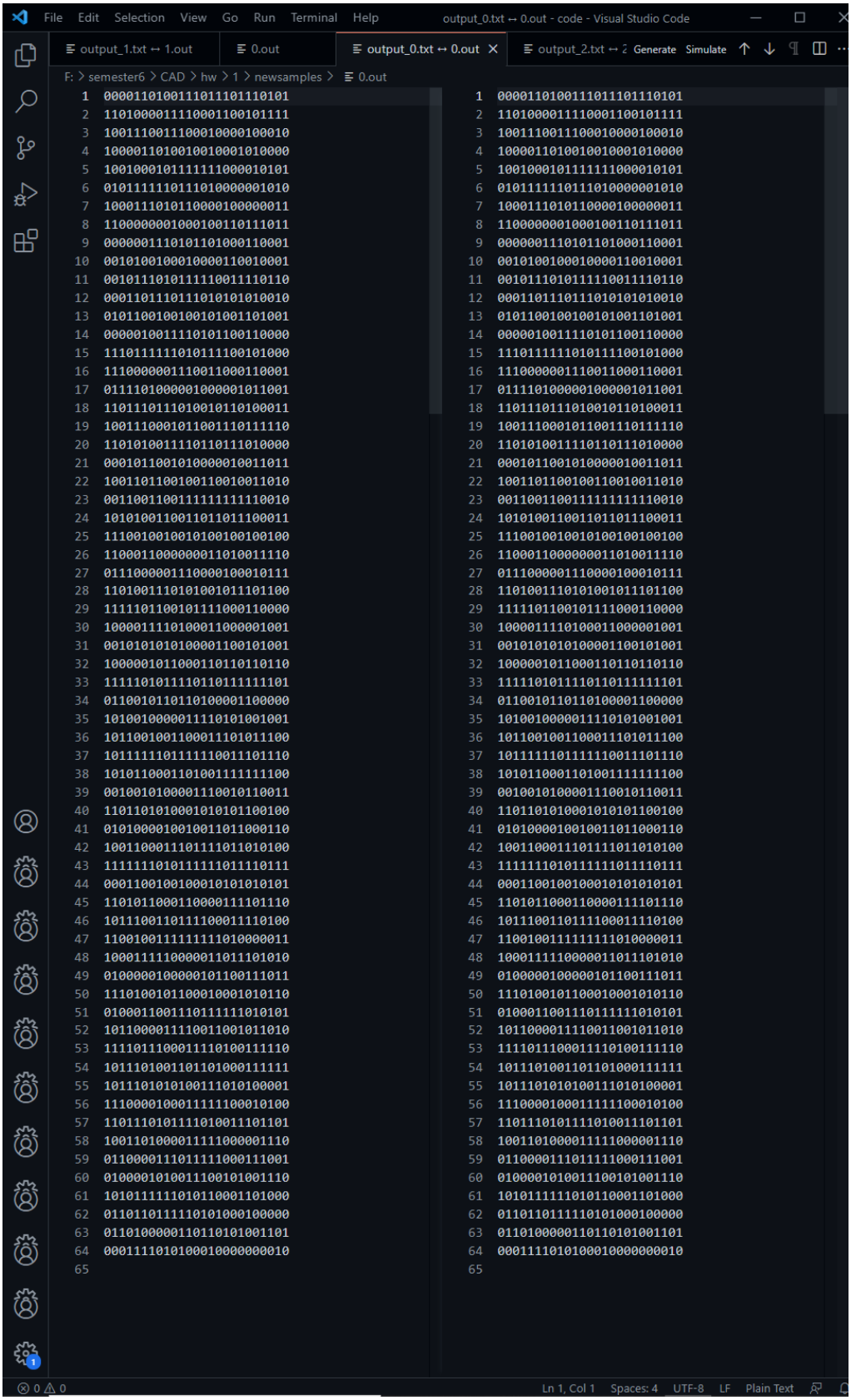
```
    S2 #(1)
ns0_s2(.D0(~ps[0]),.D1(done&valBitxx01),.D2(((~ps[0])&(~valBitxx00))|(start&(valBitxx00))),.D
3((~ps[0])|(sign&valBitxx01)),.A1(~ps[3]),.B1(1'b0),.A0(ps[2]),.B0(ps[2]),.CLR(rst),.clk(clk)
,.out(ps[0])); //
    S2 #(1)
ns1_s2(.D0(ps[0]),.D1(((xorBit0and1)&~(valBitxx01))|(done&(valBitxx01))),.D2(xorBit0and1),.D3
(((xorBit0and1)&~(valBitxx01))|((~sign)&(valBitxx01))),.A1(~ps[3]),.B1(1'b0),.A0(ps[2]),.B0(p
s[2]),.CLR(rst),.clk(clk),.out(ps[1])); //
    S2 #(1)
ns2_s2(.D0(ps[1]),.D1(((~done)&valBitxx01)|((~tmp)&valBitxx10)|(valBitxx00)),.D2((ps[1]&~(val
Bitxx10))|(coutCount&valBitxx10)),.D3(~andBit0and1),.A1(~ps[3]),.B1(1'b0),.A0(ps[2]),.B0(ps[2
]),.CLR(rst),.clk(clk),.out(ps[2])); //
    S2 #(1)
ns3_s2(.D0(1'b1),.D1((done&valBitxx01)|((~tmp)&valBitxx10)),.D2((ps[1]&~(valBitxx10))|(coutCo
unt&valBitxx10)),.D3(andBit0and1),.A1(~ps[3]),.B1(1'b0),.A0(ps[2]),.B0(ps[2]),.CLR(rst),.clk(
clk),.out(ps[3])); //

    assign sig = ~(count + ~count2 + 6'b000001);
    assign tmp = sig[5] & sig[4] & sig[3] & sig[2] & sig[1] & sig[0];

    C2 #(1)
ok_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&((~ps[2])|(ps[2]&ps
[1]&(~ps[0])))),.B0(1'b1),.out(ok)); //A0(ps[3]&((~ps[2])|(ps[2]&ps[1]&(~ps[0])))) means it
is sub|add|check|prepared|next state and ok needs to update
    C2 #(1)
firstread_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&ps[2]&(~ps[1
])&(~ps[0])),.B0(1'b1),.out(firstread)); //A0(ps[3]&ps[2]&~ps[1]&~ps[0]) means it is store
state and firstread needs to update
    // C2 #(1)
ldTillPositive_c2(.D0(1'b1),.D1(sign),.D2(1'b0),.D3(1'b0),.A1(~ps[2]|ps[3]|ps[1]),.B1(rst),.A
0(ps[0]),.B0(1'b1),.out(ldTillPositive)); //A0(ps[0]) if it is one means it is sub3j
otherwise it is shift3 if it isnt these states we have zero and ldTillPositive needs to
update
    C2 #(1)
writeMemReg_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0((~ps[3])&ps[2]&(
~ps[1])&(~ps[0])),.B0(1'b1),.out(writeMemReg)); //A0((~ps[3])&ps[2]&(~ps[1])&(~ps[0])) means
it is shift3 state and writeMemReg needs to update
    // C2 #(1)
waitCalNexti_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0((~ps[3])&ps[2]&
(~ps[1])&ps[0]),.B0(1'b1),.out(waitCalNexti)); //A0((~ps[3])&ps[2]&(~ps[1])&ps[0]) means it
is sub3j state and waitCalNexti needs to update
    C2 #(1)
IJen_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[1]&ps[0]&(~ps[3])&(~
ps[2])),.B0(1'b1),.out(IJen)); //A0(ps[1]&ps[0]&~ps[3]&~ps[2]) means it is line state and
IJen needs to update
    C2 #(1)
ALUop_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&ps[0]&(~ps[2])&(
```

```
~ps[1])),.B0(1'b1),.out(ALUop)); //A0(ps[3]&ps[0]&~ps[2]&~ps[1]) means it is add state and
ALUop needs to update
    C2 #(1)
read_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&(~ps[0])&(~ps[2])
&(~ps[1])),.B0(1'b1),.out(read)); //A0(ps[3]&(~ps[0])&(~ps[2])&(~ps[1])) means it is sub
state and read needs to update
    C2 #(1)
write_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&ps[0]&(~ps[2])&(
~ps[1])),.B0(1'b1),.out(write)); //A0(ps[3]&ps[0]&~ps[2]&~ps[1]) means it is add state and
write needs to update
    C2 #(1)
initLine_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[1]&ps[0]&(~ps[3]
)&(~ps[2])),.B0(1'b1),.out(initLine)); //A0(ps[1]&ps[0]&~ps[3]&~ps[2]) means it is line state
and initLine needs to update
    C2 #(1)
writeVal_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&ps[2]&(~ps[1]
)&(~ps[0])),.B0(1'b1),.out(writeVal)); //A0(ps[3]&ps[2]&~ps[1]&~ps[0]) means it is store
state and writeVal needs to update
    C2 #(1)
IJregen_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[1]&(~ps[3])),.B0(
1'b1),.out(IJregen)); //A0(ps[1]&(~ps[3])) means it is line|add3j state and IJregen needs to
update
    C2 #(1)
fbeq_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&(~ps[2])&(~ps[1])
),.B0(1'b1),.out(fbeq)); //A0(ps[3]&(~ps[2])&(~ps[1])) means it is add|sub state and fbeq
needs to update
    C2 #(1)
isArith_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[0]&ps[1]&ps[2]&(~
ps[3])),.B0(1'b1),.out(isArith)); //A0(ps[0]&ps[1]&ps[2]&(~ps[3])) means it is arithmetic
state and isArith needs to update
    C2 #(1)
enable_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[3]&ps[1]&(~ps[2])&
(~ps[0])),.B0(1'b1),.out(enable)); //A0(ps[3]&ps[1]&(~ps[2])&(~ps[0])) means it is check
state and enable needs to update
    C2 #(1)
update_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[2]&ps[0]&(~ps[3])&
(~ps[1])),.B0(1'b1),.out(update)); //A0(ps[2]&ps[0]&(~ps[3])&(~ps[1])) means it is sub3j
state and update needs to update
    C2 #(1)
readLine_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[1]&(~ps[3])&(~ps
[2])&(~ps[0])),.B0(1'b1),.out(readLine)); //A0(ps[1]&(~ps[3])&(~ps[2])&(~ps[0])) means it is
ydimension state and readLine needs to update
    C2 #(1)
loadCount_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0(ps[0]&(~ps[3])&(~p
s[2])&(~ps[1])),.B0(1'b1),.out(loadCount)); //A0(ps[0]&(~ps[3])&(~ps[2])&(~ps[1])) means it
is idle state and loadCount needs to update
    C2 #(1)
enCount_c2(.D0(1'b0),.D1(1'b1),.D2(1'b0),.D3(1'b0),.A1(rst),.B1(rst),.A0((~ps[2])&ps[3]&ps[1]
```

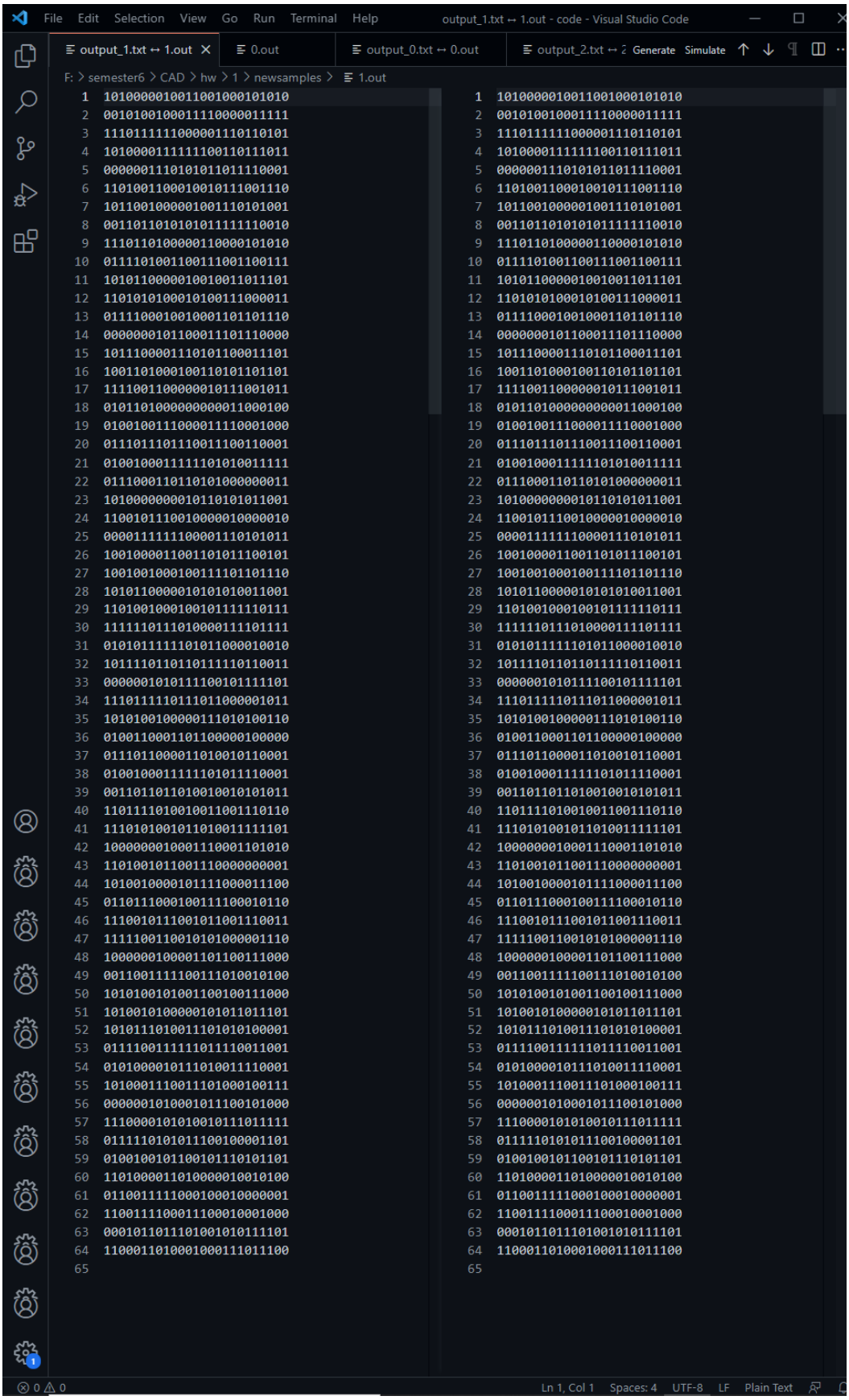
```
&ps[0]),.B0(1'b1),.out(enCount)); //A0(ps[0]&(~ps[3])&(~ps[2])&(~ps[1])) means it is idle
state and enCount needs to update

    always @(ps) begin
        {ldTillPositive, waitCalNexti} = 0;
        case (ps)
            Start:      begin
            end
            Idle:       begin
            end
            Ydimension: begin
            end
            Line:       begin
            end
            Store:     begin
            end
            Shift3:     begin
                ldTillPositive =1 'b1;
            end
            Sub3j:      begin
                waitCalNexti = 1'b1;
                ldTillPositive = sign;
            end
            Add3j:      begin
            end
            Arithmetic: begin
            end
            Sub:        begin
            end
            Add:        begin
            end
            Check:      begin
            end
            Prepared:   begin
            end
            Store:      begin
            end
            Updater:    begin
            end
            Next:       begin
            end
            Ready:     begin
            end
        endcase
    end
endmodule
```

نتایج خروجی:

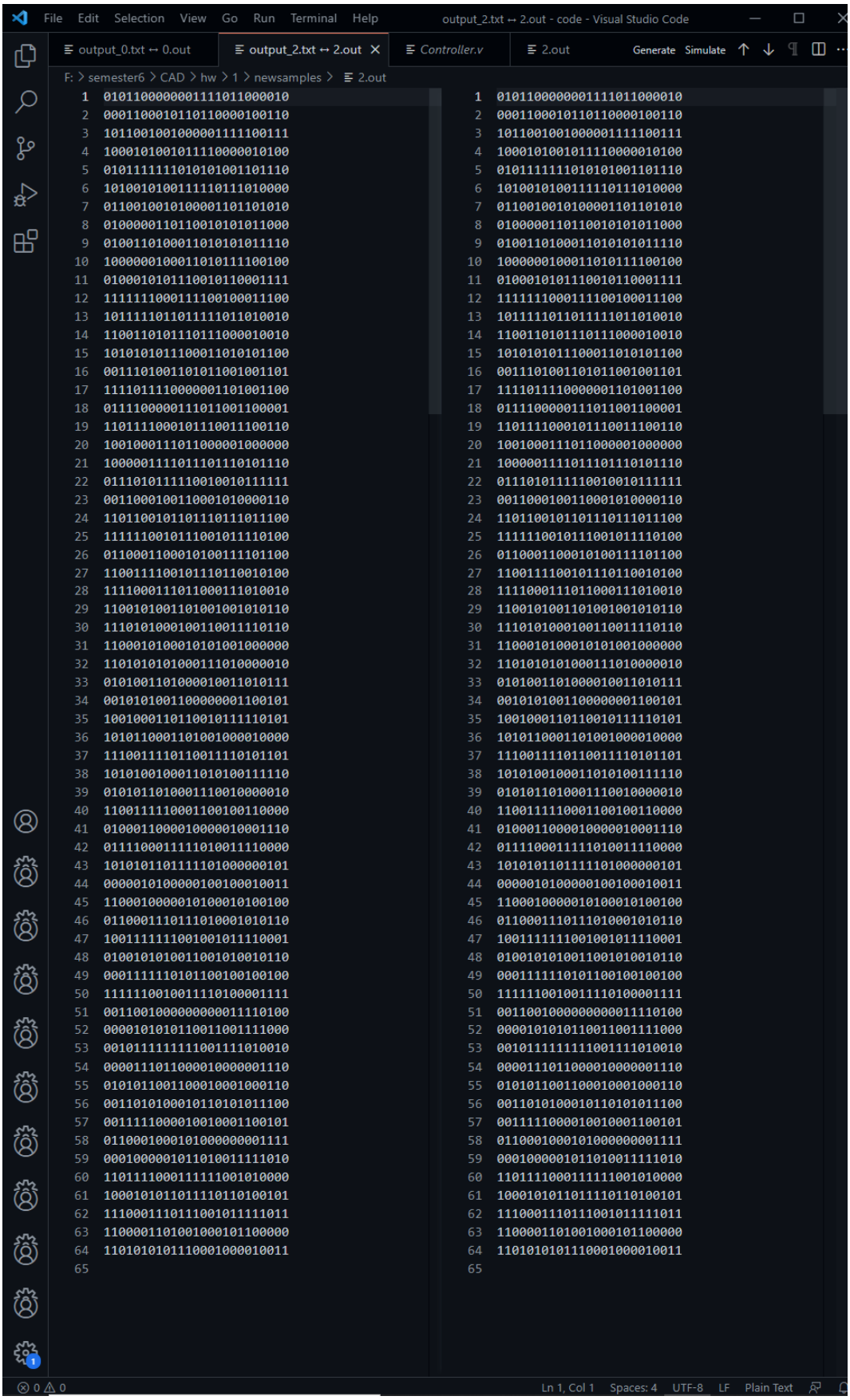
تست صفر

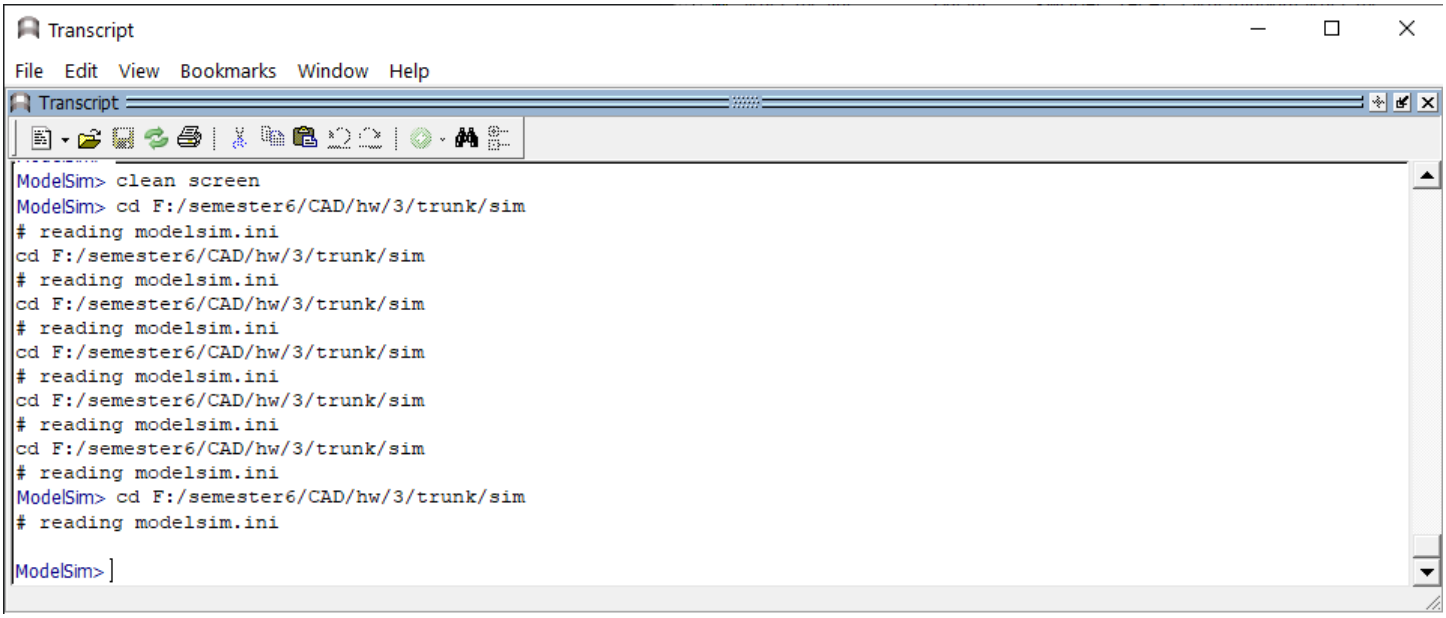


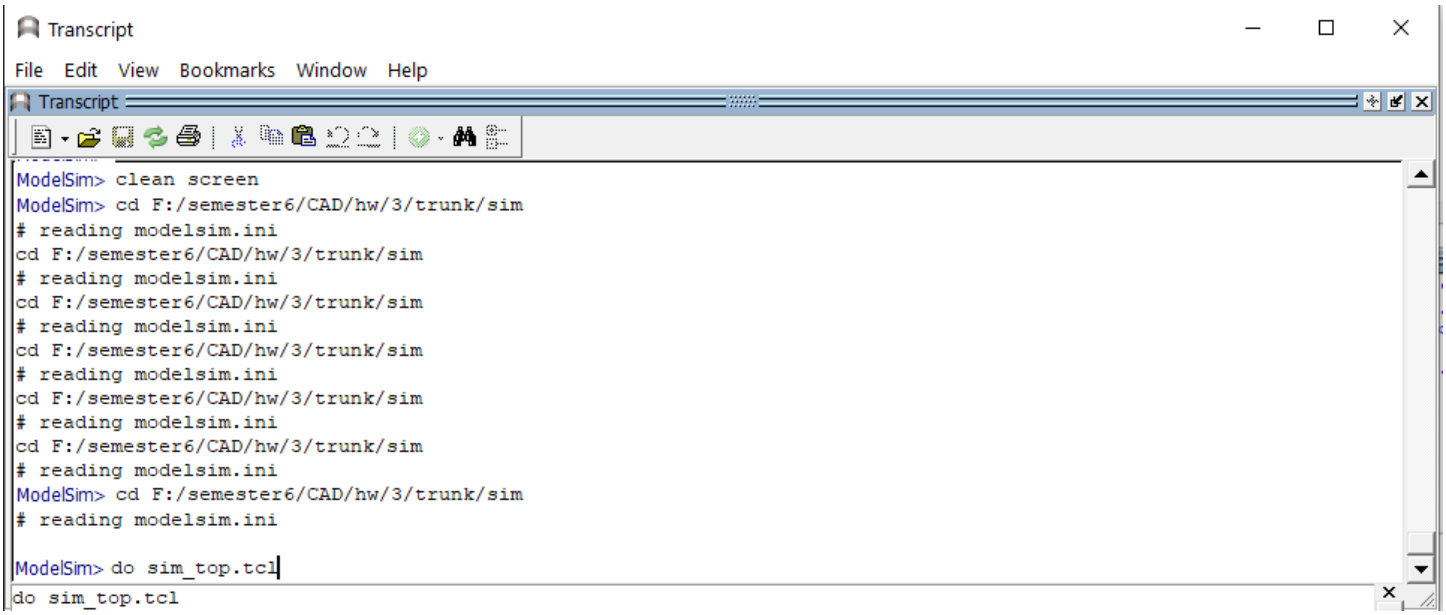
تست یک:

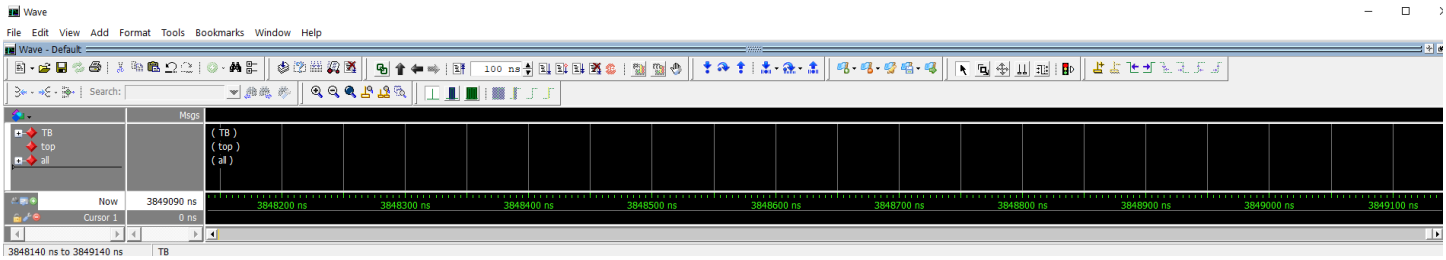

File Edit Selection View Go Run Terminal Help
output_1.txt ↔ 1.out - code - Visual Studio Code
output_1.txt ↔ 1.out
0.out
output_0.txt ↔ 0.out
output_2.txt ↔ 2
Generate Simulate
F: > semester6 > CAD > hw > 1 > newsamples > 1.out
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text

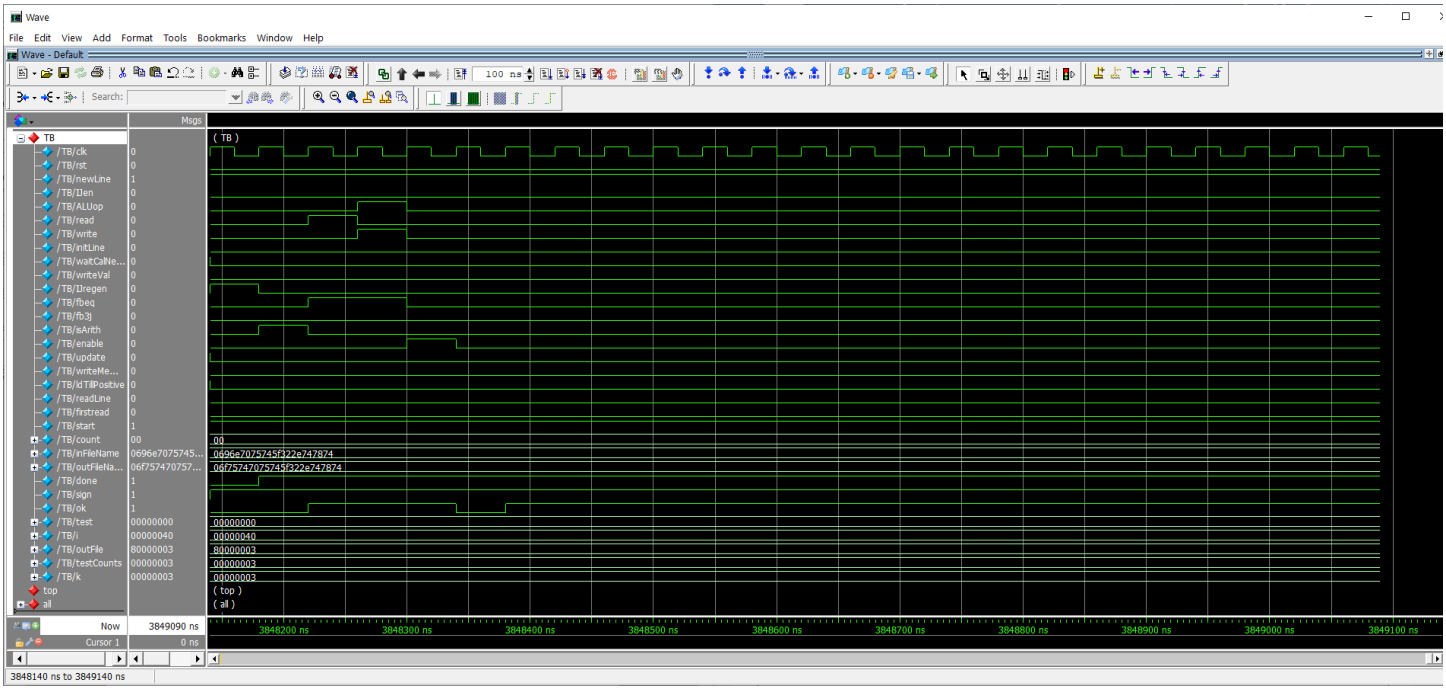
تست دو:

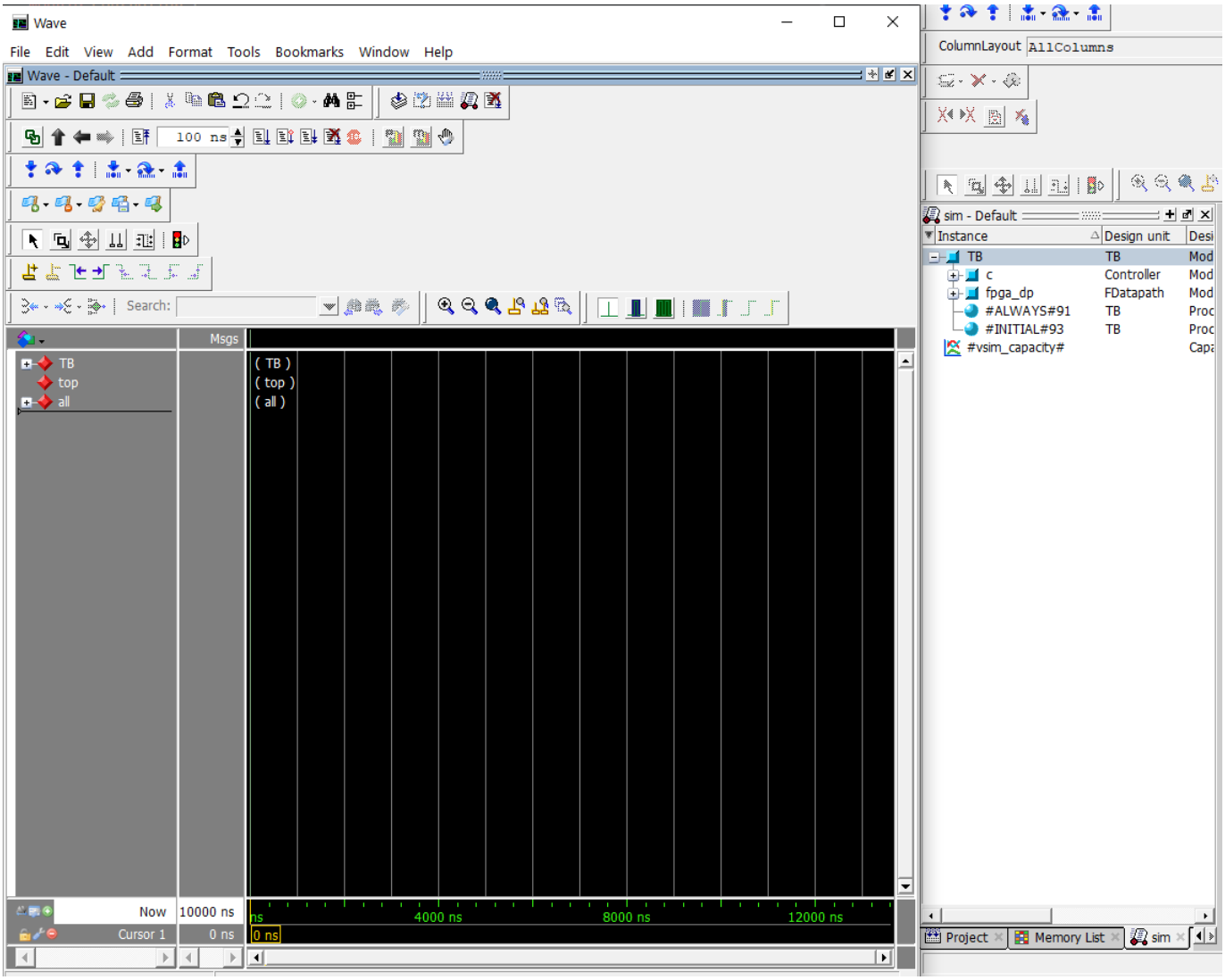

File Edit Selection View Go Run Terminal Help
output_2.txt ↔ 2.out - code - Visual Studio Code
output_0.txt ↔ 0.out
output_2.txt ↔ 2.out
Controller.v
2.out
Generate Simulate
F: > semester6 > CAD > hw > 1 > newsamples > 2.out
Ln 1, Col 1 Spaces: 4 UTF-8 LF Plain Text


Transcript
File Edit View Bookmarks Window Help
Transcript
ModelSim> clean screen
ModelSim> cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
ModelSim> cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
ModelSim>

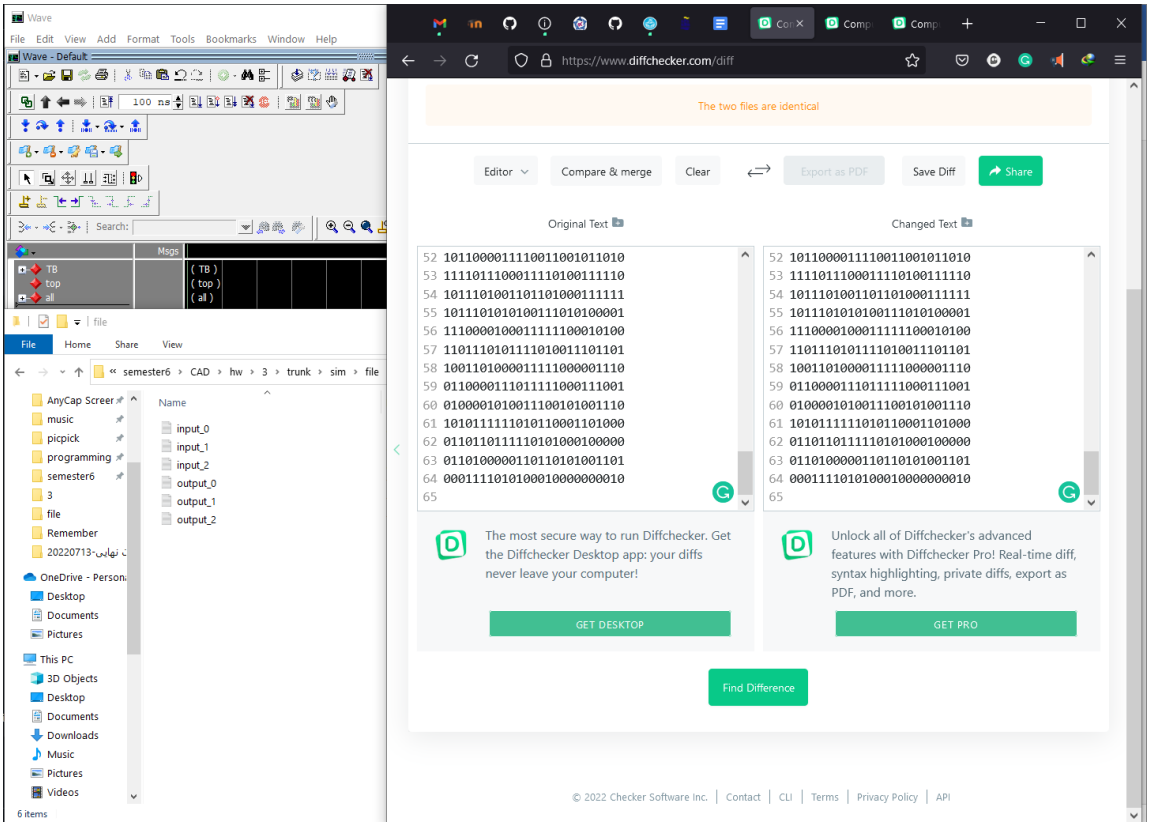

Transcript
File Edit View Bookmarks Window Help
Transcript
ModelSim> clean screen
ModelSim> cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
ModelSim> cd F:/semester6/CAD/hw/3/trunk/sim
reading modelsim.ini
ModelSim> do sim_top.tcl
do sim_top.tcl


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
TB
top
all
(TB)
(top)
(all)
Now
3849090 ns
Cursor 1
0 ns
3848200 ns
3848300 ns
3848400 ns
3848500 ns
3848600 ns
3848700 ns
3848800 ns
3848900 ns
3849000 ns
3849100 ns
3848140 ns to 3849140 ns
TB

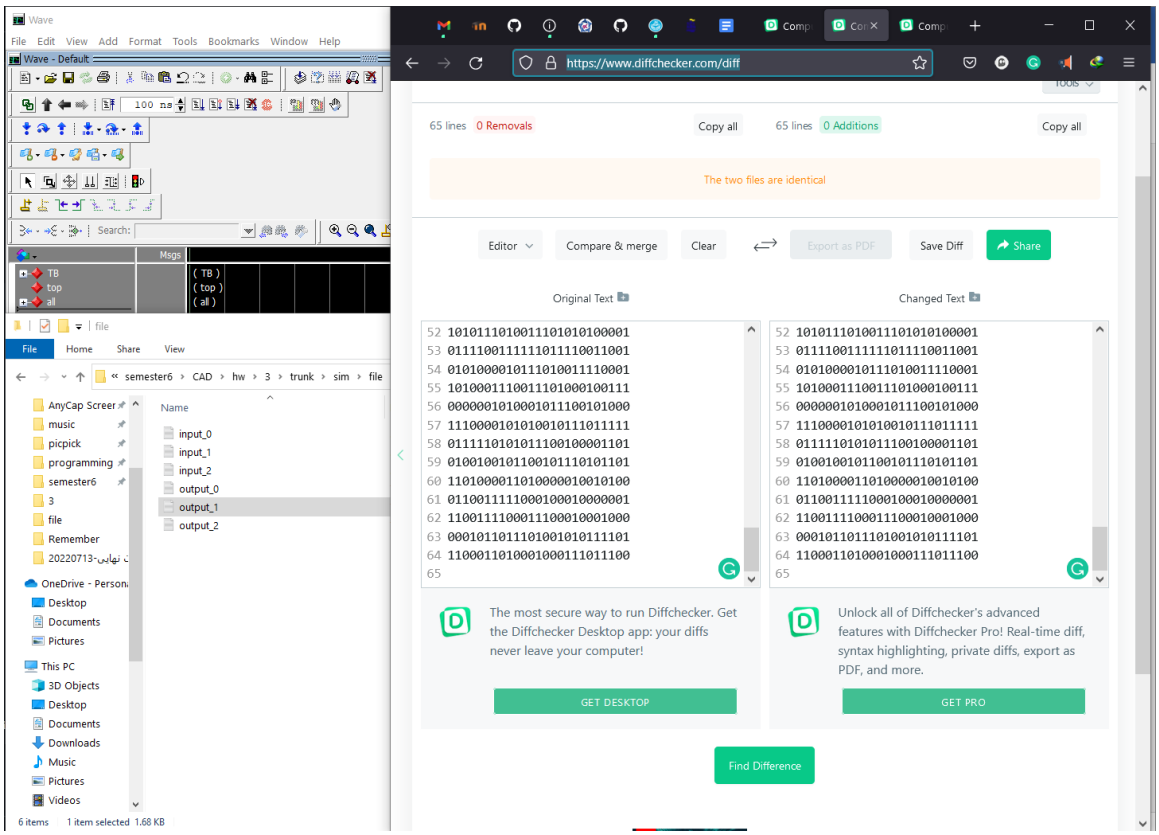

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
TB
/TB/clk
/TB/rst
/TB/newLine
/TB/IDen
/TB/ALUop
/TB/read
/TB/write
/TB/initLine
/TB/waitCalNe...
/TB/writeVal
/TB/IDregen
/TB/fbeq
/TB/fb3j
/TB/isArith
/TB/enable
/TB/update
/TB/writeMe...
/TB/IdTillPositive
/TB/readLine
/TB/firstread
/TB/start
/TB/count
/TB/inFileName
/TB/outFileNa...
/TB/done
/TB/sign
/TB/ok
/TB/test
/TB/i
/TB/outFile
/TB/testCounts
/TB/k
top
all
(TB)
(top)
(all)
Now
Cursor 1
3848140 ns to 3849140 ns


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
100 ns
Search:
Msgs
TB
top
all
(TB)
(top)
(all)
Now
10000 ns
Cursor 1
0 ns
4000 ns
8000 ns
12000 ns
ColumnLayout
AllColumns
sim - Default
Instance
Design unit
TB
c
Controller
fpga_dp
FDatapath
#ALWAYS#91
#INITIAL#93
#vsim_capacity#
Project
Memory List
sim

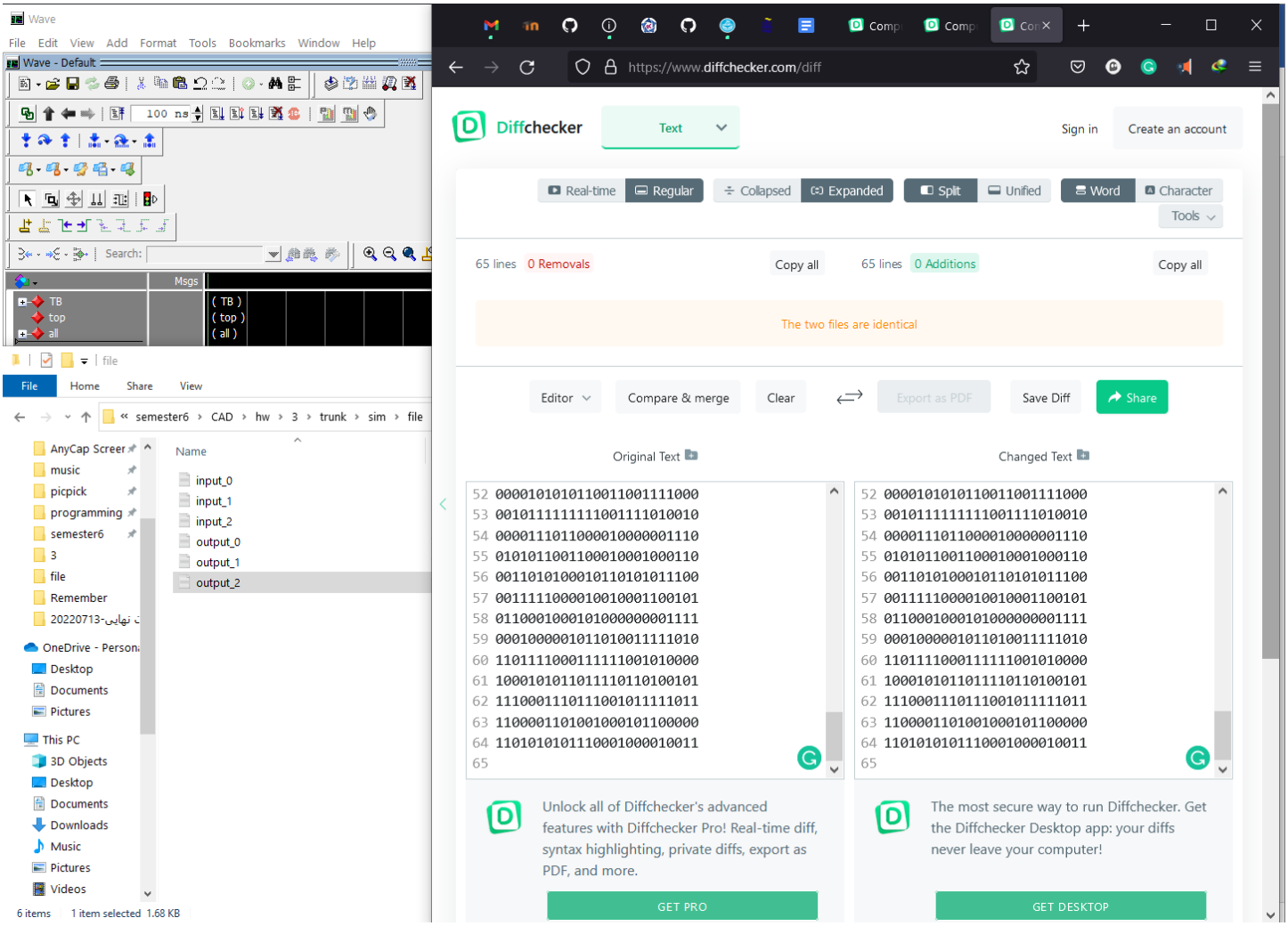
تست0:

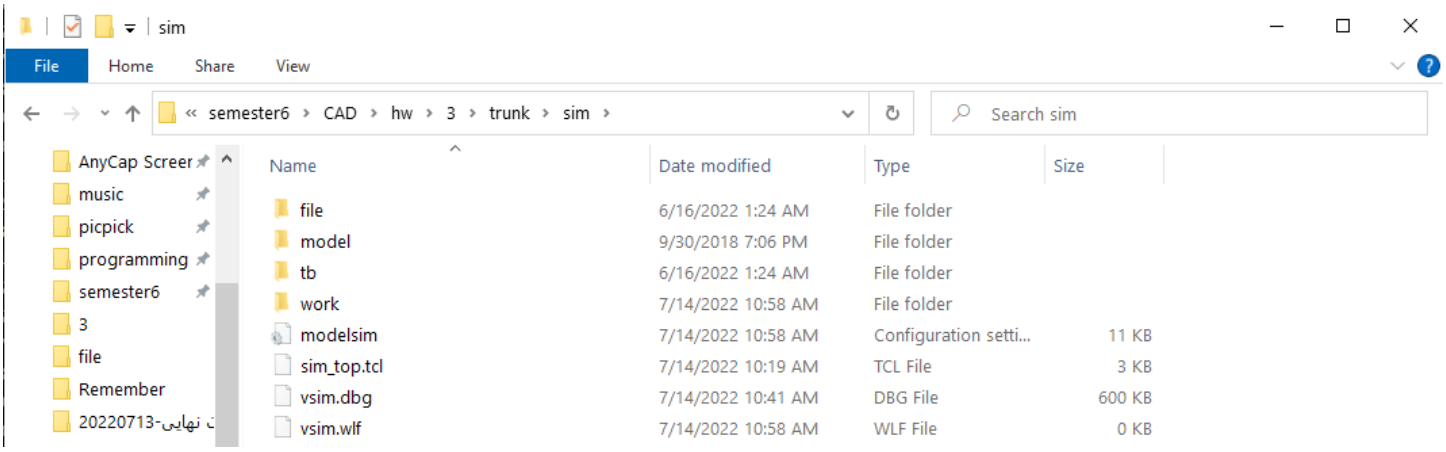

https://www.diffchecker.com/diff
The two files are identical
Editor
Compare & merge
Clear
Export as PDF
Save Diff
Share
Original Text
Changed Text
The most secure way to run Diffchecker. Get the Diffchecker Desktop app: your diffs never leave your computer!
GET DESKTOP
Unlock all of Diffchecker's advanced features with Diffchecker Pro! Real-time diff, syntax highlighting, private diffs, export as PDF, and more.
GET PRO
Find Difference
© 2022 Checker Software Inc. | Contact | CLI | Terms | Privacy Policy | API

تست1:


https://www.diffchecker.com/diff
65 lines 0 Removals
Copy all
65 lines 0 Additions
Copy all
The two files are identical
Editor
Compare & merge
Clear
Export as PDF
Save Diff
Share
Original Text
Changed Text
The most secure way to run Diffchecker. Get the Diffchecker Desktop app: your diffs never leave your computer!
GET DESKTOP
Unlock all of Diffchecker's advanced features with Diffchecker Pro! Real-time diff, syntax highlighting, private diffs, export as PDF, and more.
GET PRO
Find Difference

تست2:





توضیحات مربوط به fpga:

(C1)

$S0=S1=\phi \longrightarrow SA.A1+\overline{SA}.A0$

else $\longrightarrow SB.B1+\overline{SB}.B0$

(C2)

$A1=B1=\phi \longrightarrow A0=B0=1 \longrightarrow D01$

$\longrightarrow$ else $\longrightarrow D00$

else $\rightarrow A0=B0=1 \longrightarrow D11$

else $\rightarrow D10$

[4]Mux

1. 9bp $\longrightarrow \phi 1\phi$
2. 991 $\longrightarrow \phi 11$
3. 9tϕ $\longrightarrow 1\phi\phi$
4. 911 $\longrightarrow \phi\phi\phi$
5. 199 $\longrightarrow \phi\phi 1$

(C2) دیکودر با خروجی فعال صفر

$\{A1, B1\} = \{iReg[2], iReg[1]\}$

$\{A0, B0\} = \{iReg[\phi], iReg[2]\}$

$O00 = \{\phi, 1, iReg[4]\}$

$D01 = \{\}$ → مقاومت بالاکشنده

$D11 = \{\phi, \phi, iReg[2]\}$ $D10 = \{1, \phi\phi\}$

$\{A1, B1\} = \phi\phi$ → 1. Cin

$\phi\phi\phi$ $\phi 1\phi$

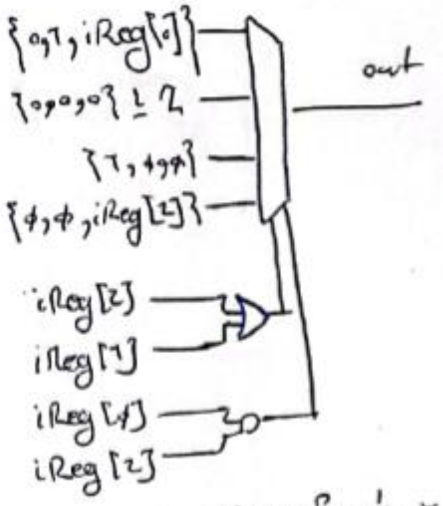
$\phi\phi 1$ $\phi 11$ → 2. "

else → $\{A0, B0\} = 11$ → 4. "

$0 11$ $\phi\phi\phi$

$1\phi\phi$ $\phi\phi 1$ → 3. "

else → $\phi 1\phi$ 1ϕ → 3. Cin



* تعداد ورودی Reg ها برابر با تعداد حالات ممکن برای انتخاب آن‌ها است.

[4]Adder

16bitAdder (with carry)

(C=0)

$\phi, \phi \longrightarrow \phi\phi$

$\phi, 1$ $1, \phi$ $\longrightarrow \phi 1$

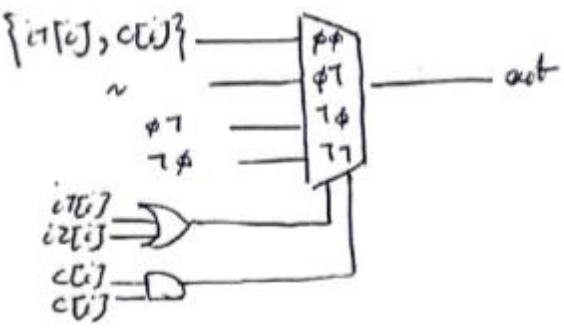
$1, 1 \longrightarrow 1\phi$

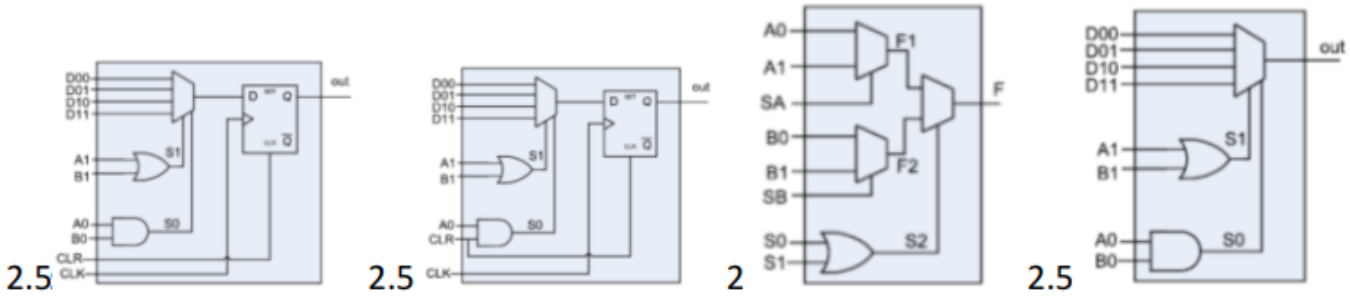
(C=1)

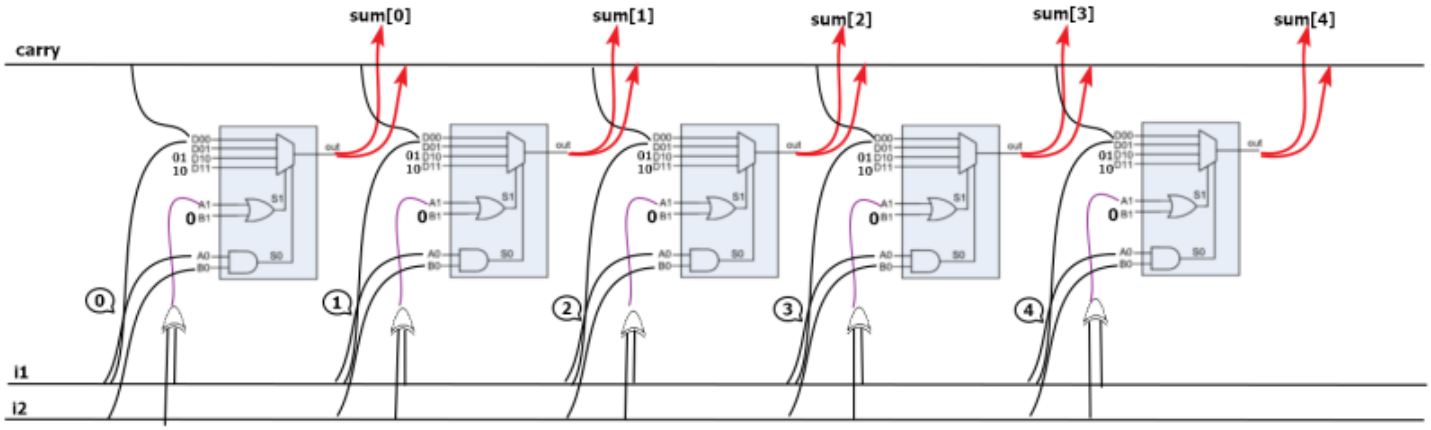
$\phi, \phi \longrightarrow \phi 1$

$\phi, 1$ $1\phi\phi$ $\longrightarrow 1\phi$

$1, 1 \longrightarrow 11$







دیلی برای ۵ بیت اددر می‌شود ۱۲/۵.

۲/۵ هم برای رجیسترها و ماژول‌های مشابه.

طولانی ترین مسیر دیتاپث هم به ۲۲/۵ می‌رسد.