



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

طراحی کامپیوتری سیستم دیجیتال

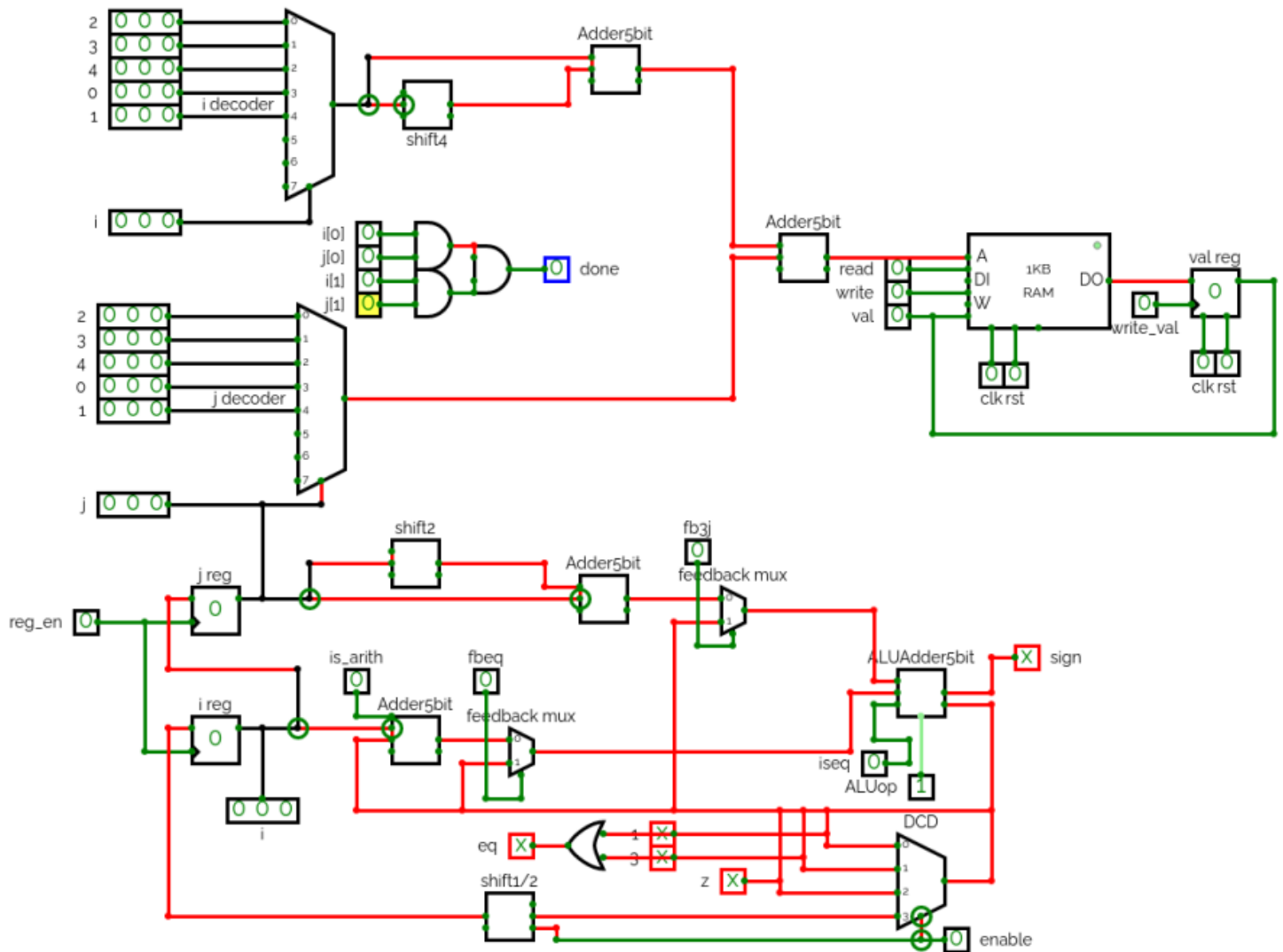
تمرین دستی ۳

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

فهرست گزارش سوالات

- سوال 1 - دیتابث 2
- سوال 2 - کنترلر 5
- سوال 3 - نتایج خروجی 11

دیتا پاتھ:



```
`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));

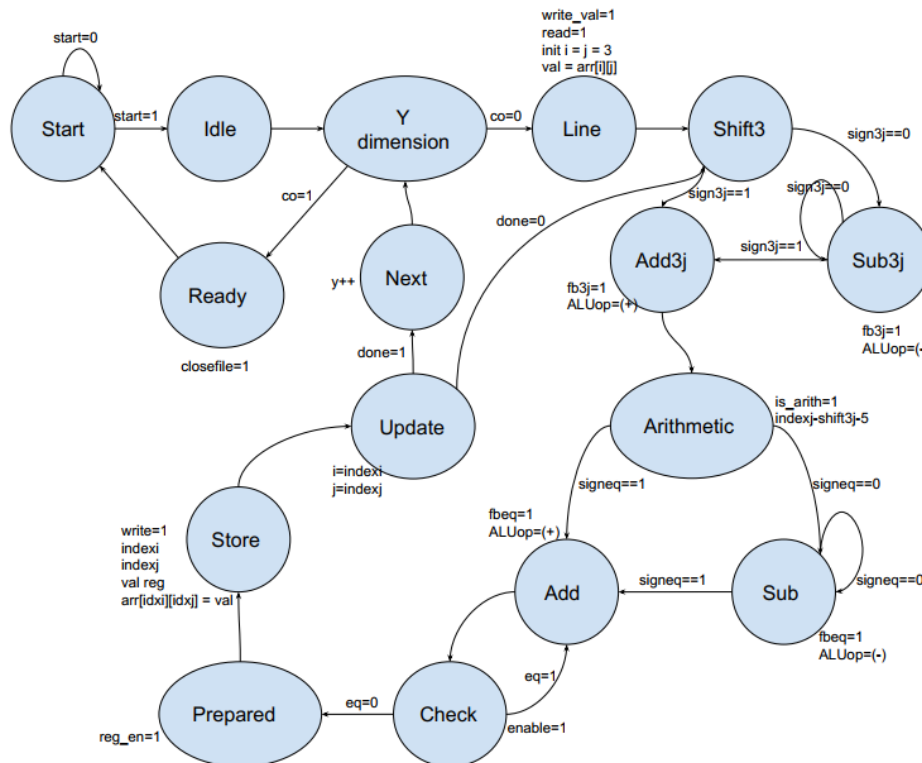
assign i = IJen ? 3'b011: update ? iAtLast : i;
assign j = IJen ? 3'b011: update ? iReg : j;

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

endmodule

```

کنترلر:



```

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

    sign3j,
    signeql,
    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, signeql, 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))),.D3(
((~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
```

```

File Edit Selection View Go Run Terminal Help
output_1.txt ↔ 1.out 0.out output_0.txt ↔ 0.out X output_2.txt ↔ 2 Generate Simulate ↑ ↓
F: > semester6 > CAD > hw > 1 > newsamples > 0.out

1 0000110100111011101110101
2 1101000011110001100101111
3 1001110011100010000100010
4 1000011010010010001010000
5 100100010111111000010101
6 010111110111010000001010
7 1000111010110000100000011
8 110000001000100110111011
9 0000001110101101000110001
10 0010100100010000110010001
11 0010111010111110011110110
12 0001101110111010101010010
13 0101100100100101001101001
14 000001001110101100110000
15 111011111010111100101000
16 111000001110011000110001
17 0111101000001000001011001
18 1101110111010010110100011
19 1001110001011001110111110
20 1101010011110110111010000
21 0001011001010000010011011
22 1001101100100110010011010
23 0011001100111111111110010
24 1010100110011011011100011
25 1110010010010100100100100
26 1100011000000011010011110
27 011100000110000100010111
28 1101001110101001011101100
29 1111101100101111000110000
30 1000011110100011000001001
31 0010101010100001100101001
32 1000001011000110110110110
33 111110101110110111111101
34 0110010110110100001100000
35 1010010000011110101001001
36 1011001001100011101011100
37 1011111011111100111011110
38 1010110001101001111111100
39 0010010100001110010110011
40 11011010100010101100100
41 0101000010010011011000110
42 1001100011101111011010100
43 111111101011111011110111
44 0001100100100010101010101
45 1101011000110000111011110
46 1011100110111100011110100
47 1100100111111111010000011
48 1000111110000011011101010
49 0100000100000101100111011
50 1110100101100010001010110
51 0100011001110111111010101
52 1011000011110011001011010
53 1111011100011101001111110
54 1011101001101101000111111
55 101110101000111010100001
56 1110000100011111100010100
57 11011101011110100111101101
58 1001101000011111000001110
59 0110000111011111000111001
60 0100001010011100101001110
61 1010111111010110001101000
62 0110110111110101000100000
63 0110100000110110101001101
64 0001111010100010000000010
65

1 0000110100111011101110101
2 1101000011110001100101111
3 1001110011100010000100010
4 1000011010010010001010000
5 100100010111111000010101
6 010111110111010000001010
7 1000111010110000100000011
8 110000001000100110111011
9 0000001110101101000110001
10 0010100100010000110010001
11 0010111010111110011110110
12 0001101110111010101010010
13 0101100100100101001101001
14 000001001110101100110000
15 111011111010111100101000
16 111000001110011000110001
17 0111101000001000001011001
18 1101110111010010110100011
19 1001110001011001110111110
20 1101010011110110111010000
21 0001011001010000010011011
22 1001101100100110010011010
23 0011001100111111111110010
24 1010100110011011011100011
25 1110010010010100100100100
26 1100011000000011010011110
27 011100000110000100010111
28 1101001110101001011101100
29 1111101100101111000110000
30 1000011110100011000001001
31 0010101010100001100101001
32 1000001011000110110110110
33 111110101110110111111101
34 0110010110110100001100000
35 1010010000011110101001001
36 1011001001100011101011100
37 1011111011111100111011110
38 1010110001101001111111100
39 0010010100001110010110011
40 11011010100010101100100
41 0101000010010011011000110
42 1001100011101111011010100
43 111111101011111011110111
44 0001100100100010101010101
45 1101011000110000111011110
46 1011100110111100011110100
47 1100100111111111010000011
48 1000111110000011011101010
49 0100000100000101100111011
50 1110100101100010001010110
51 0100011001110111111010101
52 1011000011110011001011010
53 1111011100011101001111110
54 1011101001101101000111111
55 101110101000111010100001
56 1110000100011111100010100
57 11011101011110100111101101
58 1001101000011111000001110
59 0110000111011111000111001
60 0100001010011100101001110
61 1010111111010110001101000
62 0110110111110101000100000
63 0110100000110110101001101
64 0001111010100010000000010
65

```

```

1 1010000010011001000101010
2 0010100100011110000011111
3 111011111100000110110101
4 101000011111100110111011
5 00000011101010111110001
6 1101001100010010111001110
7 1011001000001001110101001
8 00110110101011111110010
9 111011010000110000101010
10 0111101001100111001100111
11 1010110000010010011011101
12 1101010100010100111000011
13 0111100010010001101101110
14 0000000101100011101110000
15 1011100001110101100011101
16 1001101000100110101101101
17 1111001100000010111001011
18 010110100000000011000100
19 0100100111000011110001000
20 0111011101110011100110001
21 0100100011111101010011111
22 011100011011010100000011
23 1010000000010110101011001
24 1100101110010000010000010
25 0000111111100001110101011
26 1001000011001101011100101
27 1001001000100111101101110
28 1010110000010101010011001
29 1101001000100101111101111
30 1111110111010000111101111
31 0101011111101011000010010
32 1011110110110111110110011
33 0000001010111100101111101
34 111011110111011000001011
35 1010100100000111010100110
36 0100110001101100000100000
37 0111011000011010010110001
38 0100100011111101011110001
39 0011011011010010010101011
40 1101111010010011001110110
41 1110101001011010011111101
42 1000000010001110001101010
43 1101001011001110000000001
44 1010010000101111000011100
45 0110111000100111100010110
46 1110010111001011001110011
47 1111100110010101000001110
48 1000000100001101100111000
49 0011001111100111010010100
50 1010100101001100100111000
51 1010010100000101011011101
52 1010111010011101010100001
53 011110011111011110011001
54 0101000010111010011110001
55 1010001110011101000100111
56 0000001010001011100101000
57 1110000101001011101111111
58 0111110101011100100001101
59 0100100101100101110101101
60 1101000011010000010010100
61 0110011111000100010000001
62 1100111100011100010001000
63 0001011011101001010111101
64 1100011010001000111011100
65

```

The image shows a Visual Studio Code editor window with two files open: `output_0.txt` and `output_2.txt`. Both files contain 65 lines of binary code (0s and 1s). The editor interface includes a menu bar (File, Edit, Selection, View, Go, Run, Terminal, Help), a toolbar with icons for Explorer, Search, Source Control, Run and Debug, and Extensions, and a status bar at the bottom showing 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and 'Plain Text'.

```

1 0101100000001111011000010
2 0001100010110110000100110
3 1011001001000001111100111
4 1000101001011110000010100
5 0101111111010101001101110
6 1010010100111110111010000
7 0110010010100001101101010
8 0100000110110010101011000
9 0100110100011010101011110
10 1000000100011010111100100
11 0100010101110010110001111
12 1111110001111001000111000
13 101111011011111011010010
14 1100110101110111000010010
15 101010101110001101011100
16 0011101001101011001001101
17 1111011110000001101001100
18 0111100000111011001100001
19 1101110001011100111001110
20 1001000111011000001000000
21 1000001111011101110101110
22 0111010111110010010111111
23 0011000100110001010000110
24 1101100101101110111011100
25 1111110010111001011110100
26 0110001100010100111101100
27 1100111100101110110010100
28 1111000111011000111010010
29 1100101001101001001010110
30 1110101000100110011110110
31 1100010100010101001000000
32 1101010101000111010000010
33 0101001101000010011010111
34 0010101001100000001100101
35 1001000110110010111110101
36 1010110001101001000010000
37 1110011110110011110101101
38 1010100100011010100111110
39 0101011010001110010000010
40 1100111110001100100110000
41 0100011000010000010001110
42 0111100011111010011110000
43 101010110111110100000101
44 0000010100000100100010011
45 1100010000010100010100100
46 0110001110111010001010110
47 1001111111001001011110001
48 01001010011001010010110
49 0001111110101100100100100
50 1111110010011110100001111
51 0011001000000000011110100
52 0000101010110011001111000
53 001011111111001111010010
54 0000111011000010000001110
55 0101011001100010001000110
56 0011010100010110101011100
57 0011111000010010001100101
58 0110001000101000000001111
59 0001000001011010011111010
60 1101111000111111001010000
61 1000101011011110110100101
62 1110001110111001011111011
63 1100001101001000101100000
64 11010101110001000010011
65

```

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> ]
```

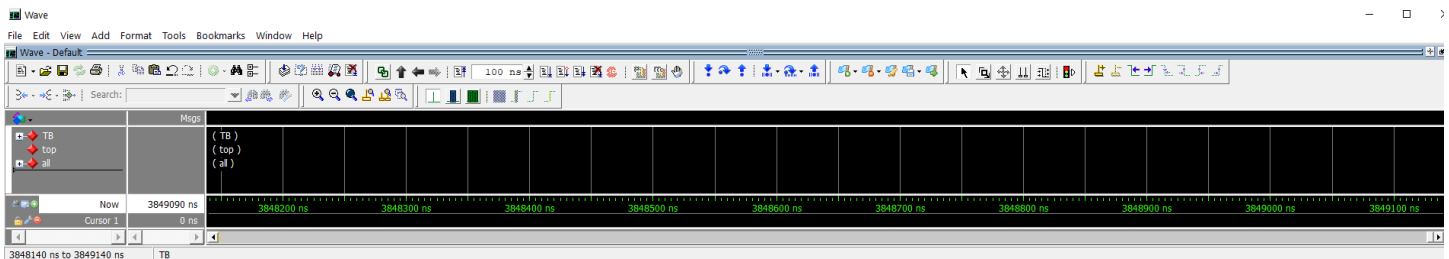
Transcript

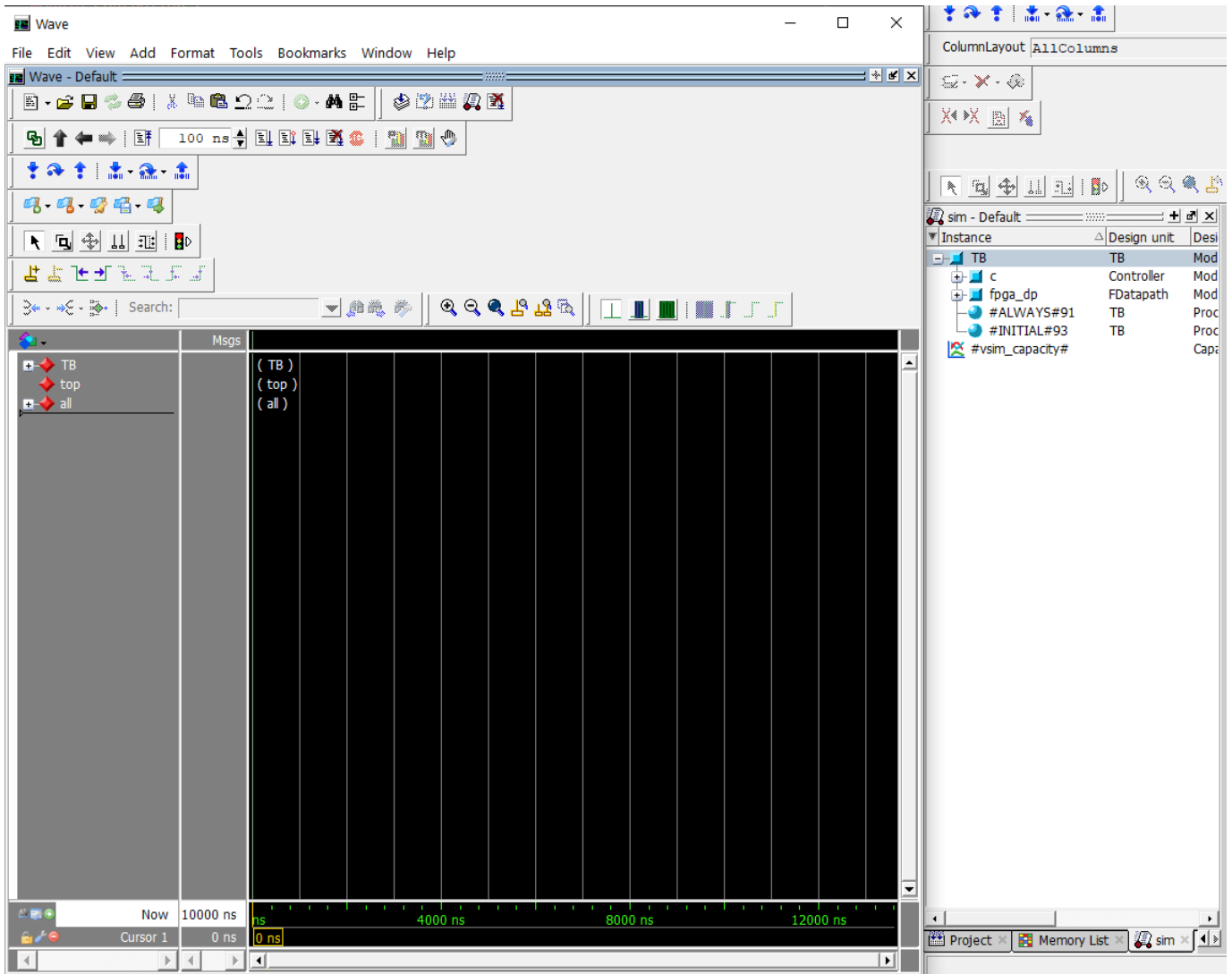
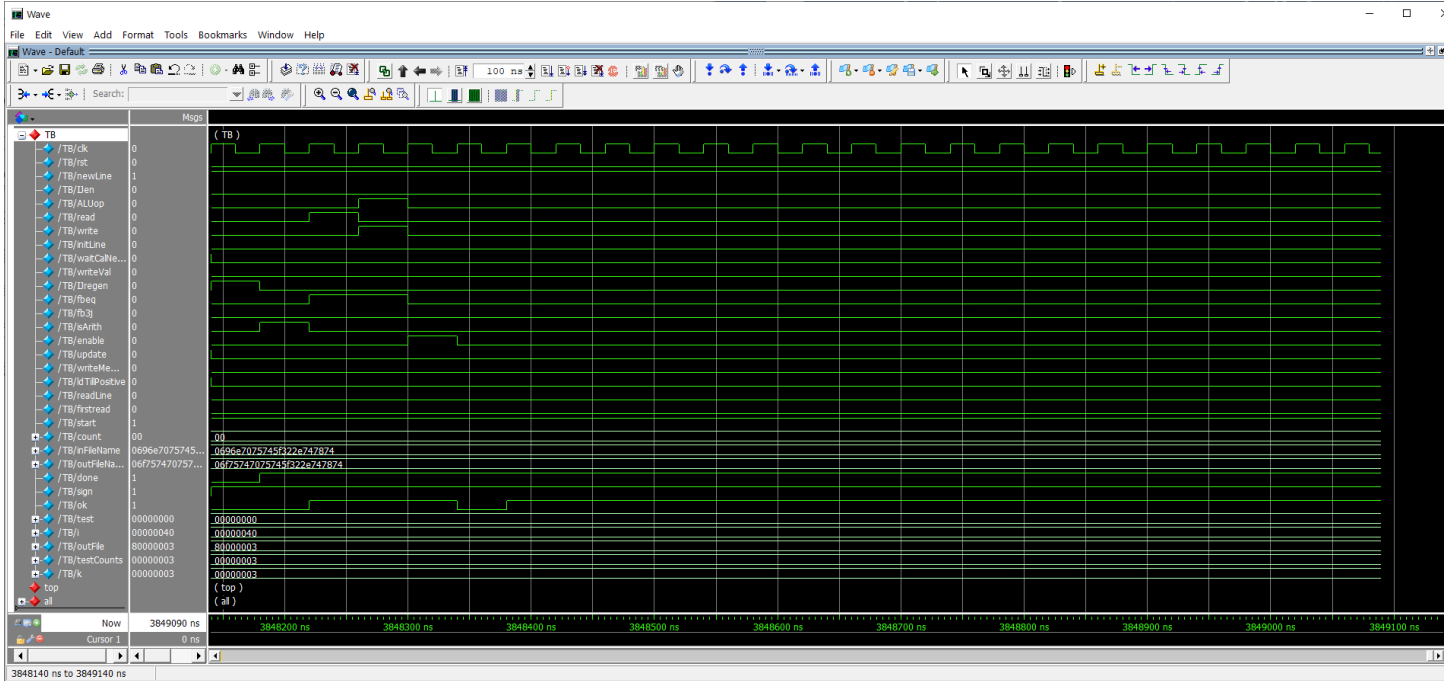
File Edit View Bookmarks Window Help

Transcript

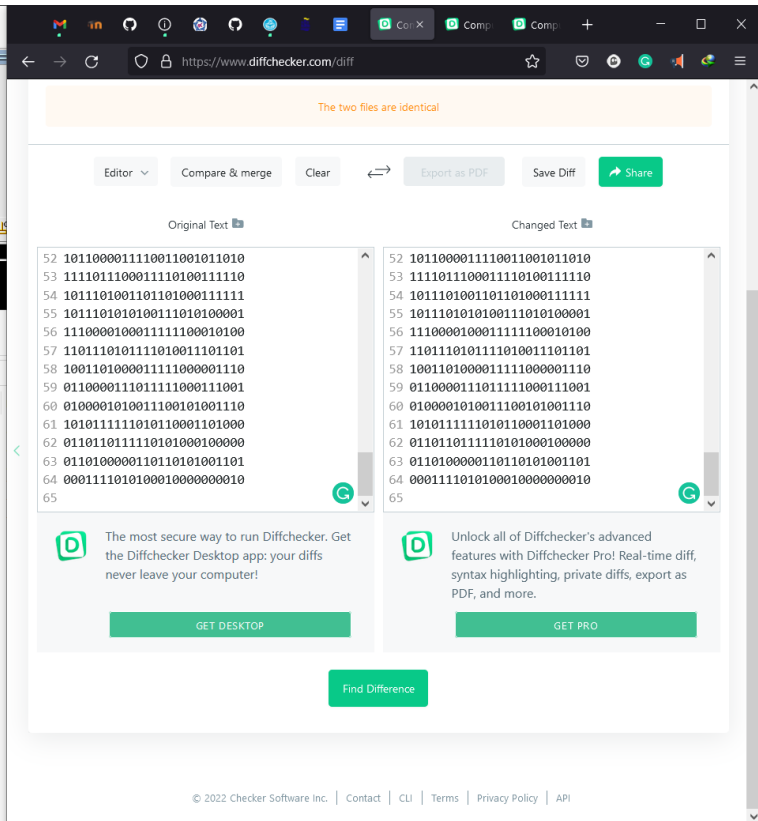
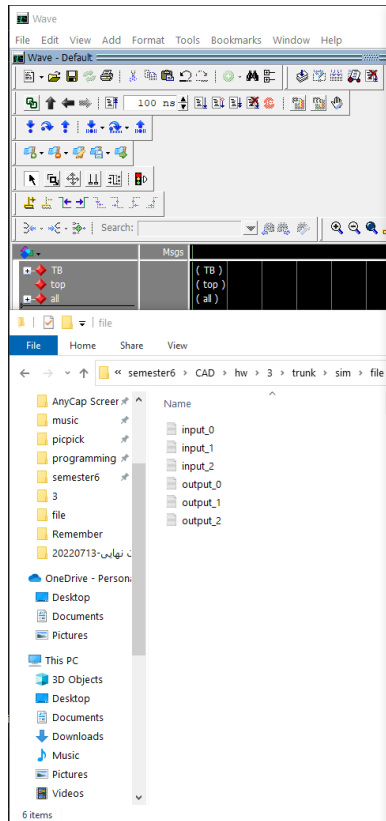
```
ModelSim> clean screen
ModelSim> cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
cd F:/semester6/CAD/hw/3/trunk/sim
# reading modelsim.ini
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
```

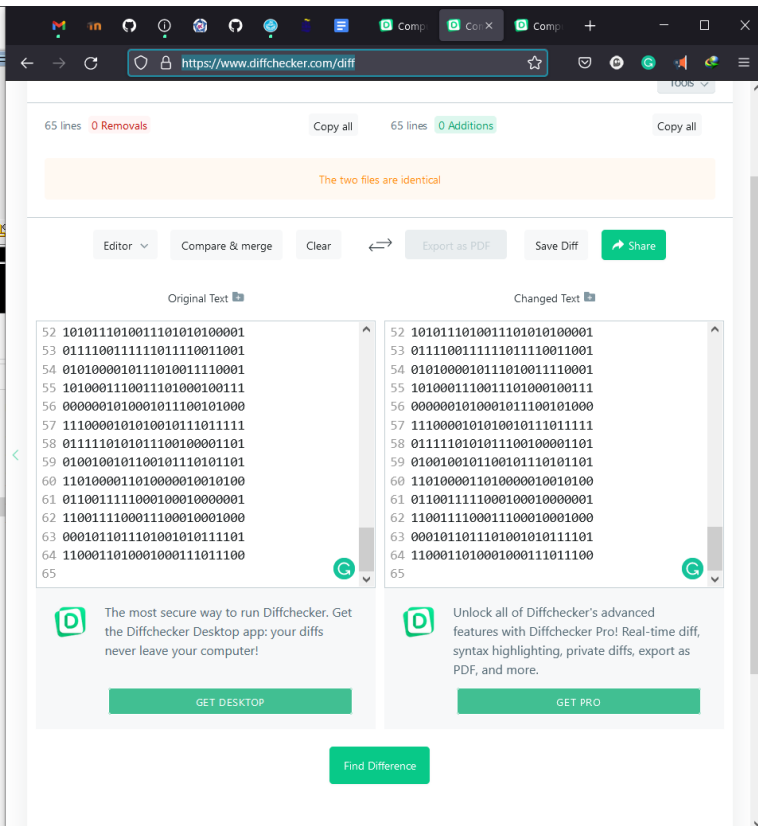
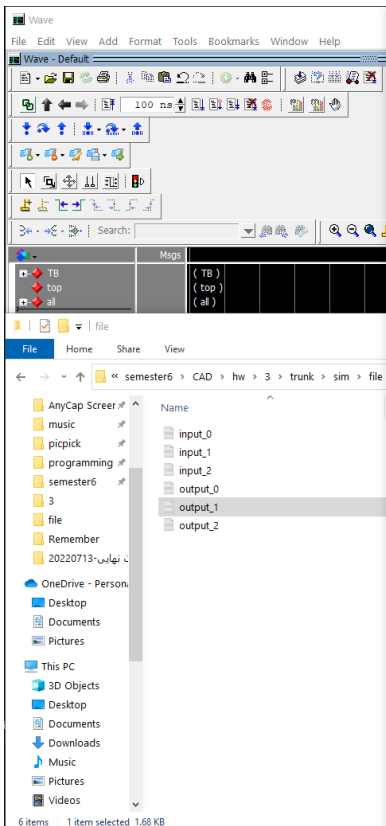


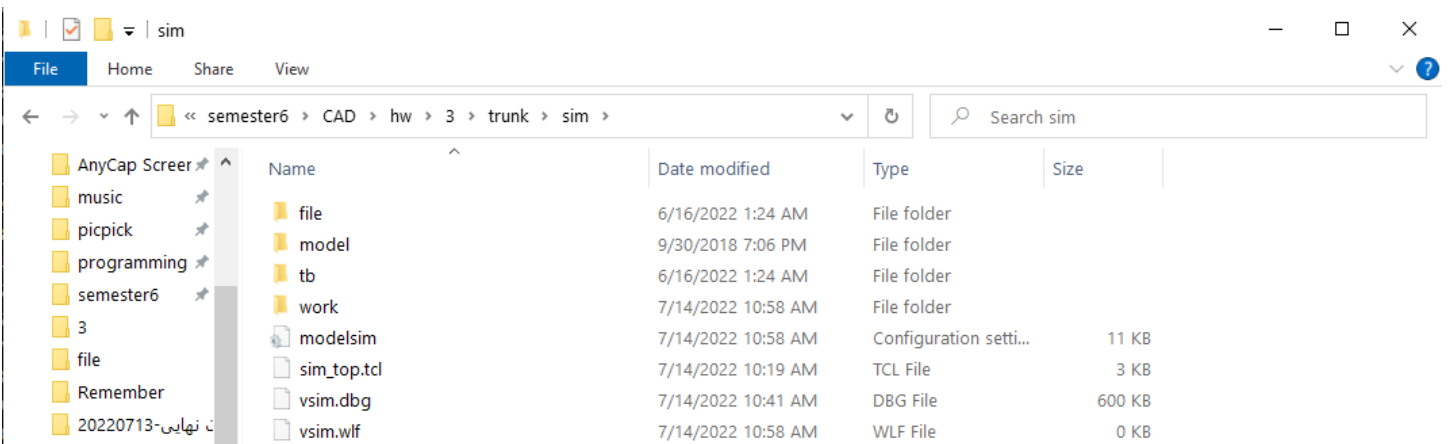
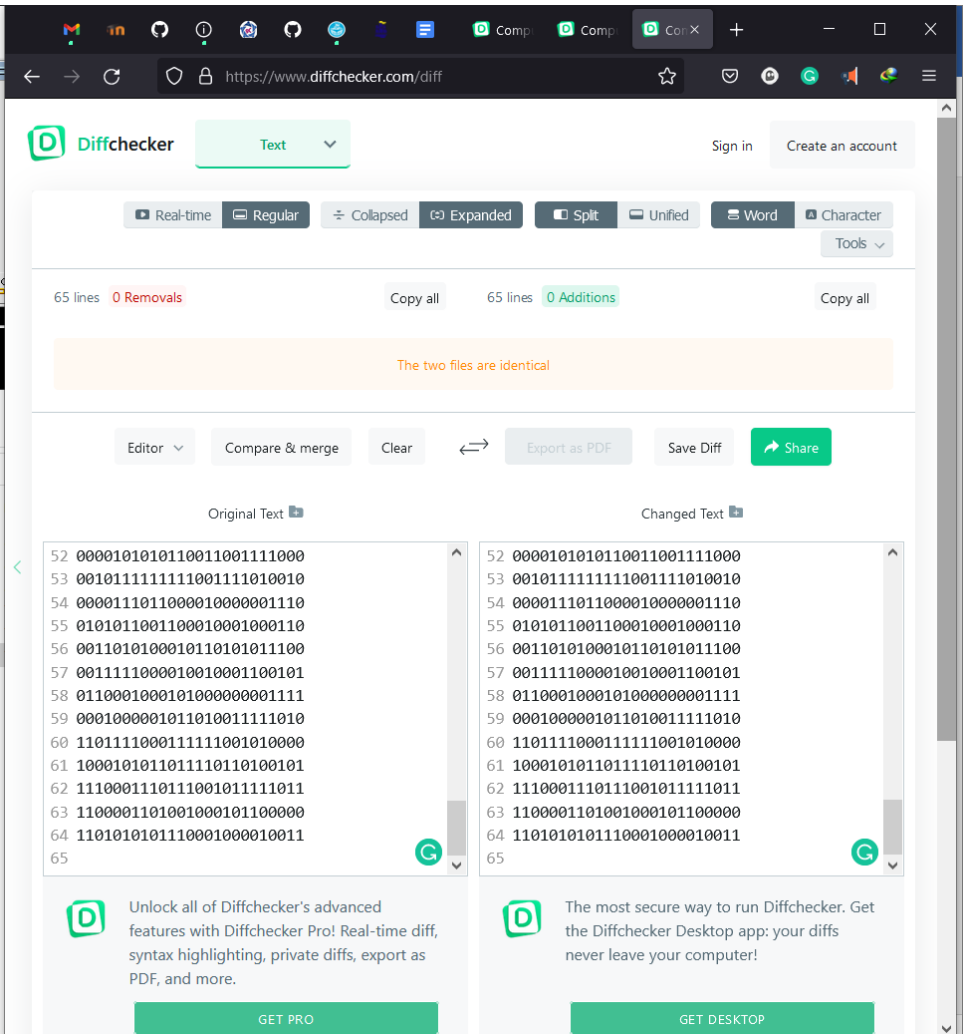
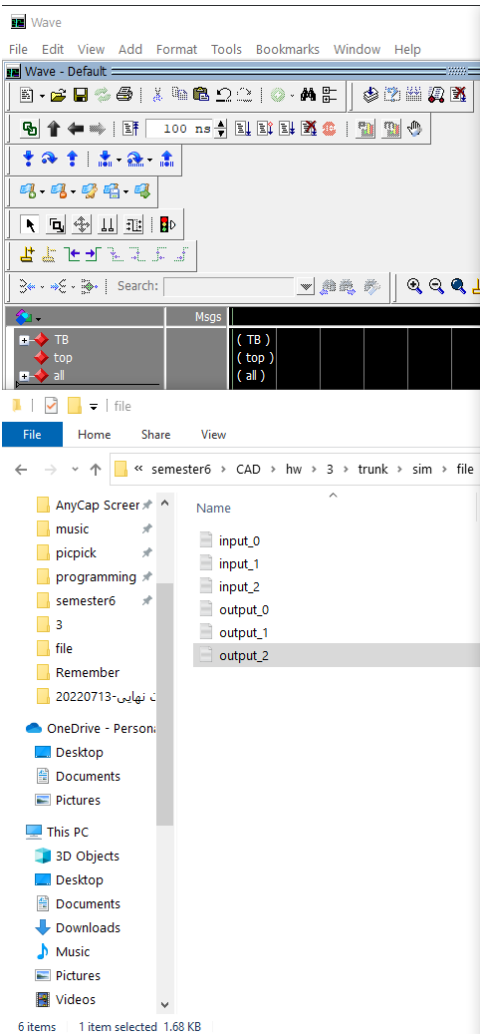


تست 0:



تست 1:





توضیحات مربوط به fpga:

(C1)

$$S0 = S1 = \phi \rightarrow SA \cdot A1 + \overline{SA} \cdot A0$$

$$\text{else} \rightarrow SB \cdot B1 + \overline{SB} \cdot B0$$

(C2)

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

$$\text{else} \rightarrow D00$$

$$\text{else} \rightarrow A0 = B0 = 1 \rightarrow D11$$

$$\text{else} \rightarrow D10$$

[IJMux]

1. $\phi\phi \rightarrow \phi\phi$
2. $\phi\phi 1 \rightarrow \phi 11$
3. $\phi 1\phi \rightarrow 1\phi\phi$
4. $\phi 11 \rightarrow \phi\phi\phi$
5. $1\phi\phi \rightarrow \phi\phi 1$

(C2) $\phi\phi, \phi\phi 1, \phi 1\phi, \phi 11, 1\phi\phi$

$$\{A1, B1\} = \{iReg[1], iReg[0]\}$$

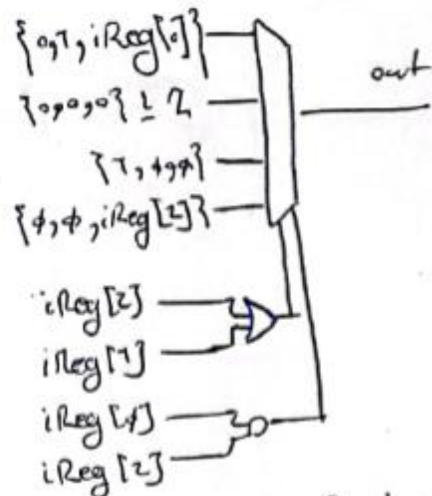
$$\{A0, B0\} = \{iReg[2], iReg[3]\}$$

$$D00 = \{\phi, 1, iReg[4]\}$$

$$D01 = \{1, \phi, iReg[4]\}$$

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

$$D10 = \{1, 1, \phi\}$$



$\{A1, B1\} = \phi\phi$ $\rightarrow \phi\phi\phi$ $\rightarrow 1. \text{Cra}$

$\phi\phi 1$ $\rightarrow 2. \text{Cra}$

$\phi 1\phi$ $\rightarrow 4. \text{Cra}$

$\phi 11$ $\rightarrow 3. \text{Cra}$

$1\phi\phi$ $\rightarrow 3. \text{Cra}$

[Adder]

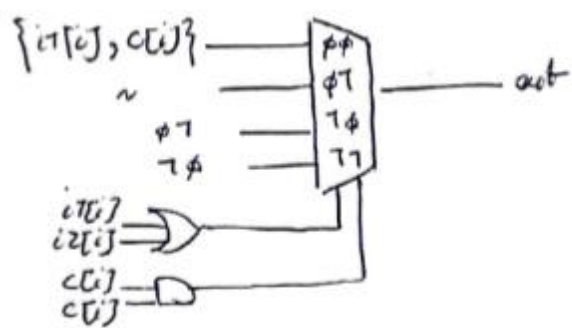
1bit Adder (with carry)

(C50)

- $\phi\phi \rightarrow \phi\phi$
- $\phi\phi 1 \rightarrow \phi 1$
- $1\phi\phi \rightarrow 1\phi$
- $1, 1 \rightarrow 11$

(C51)

- $\phi\phi \rightarrow \phi 1$
- $\phi\phi 1 \rightarrow 1\phi$
- $1\phi\phi \rightarrow 11$



برای تست کردن ما می‌توانیم از این تست‌ها استفاده کنیم. به عنوان مثال برای تست H و H' می‌توانیم از تست $C2$ استفاده کنیم. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

در هفتم این تست $C2$ را داریم. انتخاب تست‌ها بسیار مهم است. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

* جهت تست کردن ما می‌توانیم از این تست‌ها استفاده کنیم. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

$$\begin{aligned} A_1 &= \sim ps[3] & A_0 &= ps[2] & A_1 &= B_1 = \phi & \checkmark & \rightarrow & A_0 &= B_0 = \phi & \checkmark & \rightarrow & D_{01} &\rightarrow psbit \ 11xx \\ B_1 &= \phi & B_0 &= ps[2] & & & & & & & & & \checkmark & \rightarrow & D_{00} &\rightarrow psbit \ 10xx \\ & & & & & & & & & & & & \checkmark & \rightarrow & D_{10} &\rightarrow psbit \ 01xx \\ & & & & & & & & & & & & \checkmark & \rightarrow & D_{11} &\rightarrow psbit \ 00xx \end{aligned}$$

نمایش جدول truth table برای تست $C2$ را می‌توانیم از این تست‌ها استفاده کنیم. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

تست $C2$

$$\begin{aligned} D_{01} &= done \& (\sim ps[3]) \& ps[2] \\ D_{00} &= \sim ps[2] \\ D_{11} &= \sim ps[2] \mid sign \& (\sim ps[3]) \& ps[2] \\ D_{10} &= (\sim ps[2]) \& (ps[3] \mid ps[2]) \mid start \& (\sim ps[3]) \& (\sim ps[2]) \\ D_{01} &= (ps[2] \mid ps[3]) \& (\sim ps[3]) \& (\sim ps[2]) \mid done \& (\sim ps[3]) \& ps[2] \\ D_{00} &= ps[2] \\ D_{11} &= ((ps[2] \mid ps[3]) \& (\sim ps[3]) \& (\sim ps[2]) \mid \sim sign \& (\sim ps[3]) \& ps[2]) \\ D_{10} &= ps[2] \mid ps[3] \\ D_{01} &= (\sim ps[3]) \& ps[2] \& mbnc \mid (ps[3] \& \sim ps[2] \& \sim mbnc) \mid \sim (ps[3] \mid ps[2]) \\ D_{00} &= ps[2] \\ D_{11} &= \sim (ps[3] \& ps[2]) \\ D_{10} &= ps[3] \& \sim (ps[3] \& \sim ps[2]) \mid count \& (ps[3] \& \sim ps[2]) \end{aligned}$$

تست $C2$

تست $C2$

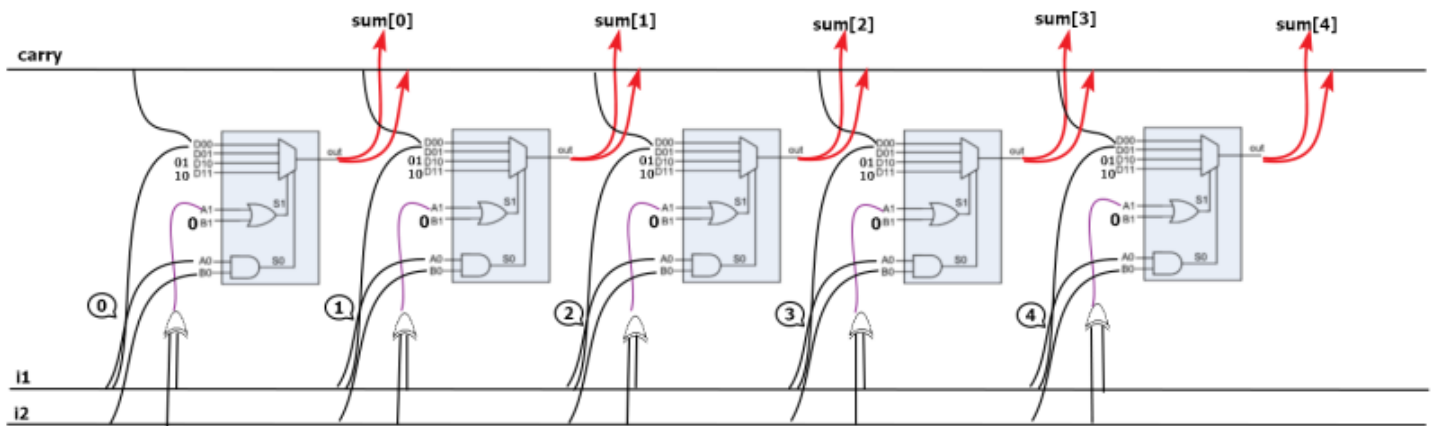
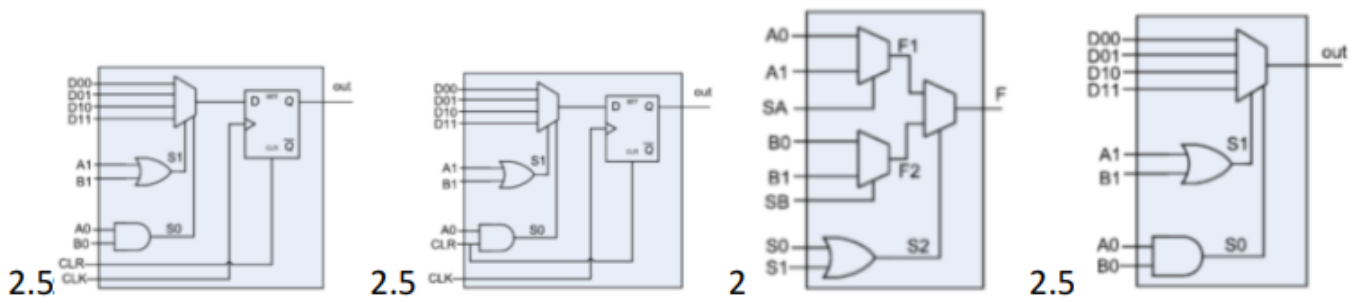
تست $C2$

$$\begin{aligned} D_{01} &= (\sim ps[3] \& ps[2] \& done) \mid (ps[3] \& \sim ps[2] \& \sim mbnc) \\ D_{00} &= 1 \\ D_{11} &= (ps[3] \& ps[2]) \\ D_{10} &= ps[3] \& \sim ps[2] \& \sim mbnc \mid count \& (ps[3] \& \sim ps[2]) \end{aligned}$$

نمایش جدول truth table برای تست $C2$ را می‌توانیم از این تست‌ها استفاده کنیم. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

نمایش جدول truth table برای تست $C2$ را می‌توانیم از این تست‌ها استفاده کنیم. به عنوان مثال برای تست $C2$ می‌توانیم از تست $C2$ استفاده کنیم.

گزارش دیلی‌ها:



دیلی برای ۵ بیت ادر می شود ۱۲/۵.

۲/۵ هم برای رجیسترها و ماژول های مشابه.

طولانی ترین مسیر دیتا پت هم به ۲۲/۵ می رسد.