module fuckthisdamn(clk,opp);

//DECLARATIONS

input clk;

output reg opp;

reg [15:0]array1[0:127];

reg [15:0]array2[0:49];

reg [15:0]indata;

reg [15:0]insign;

reg [15:0]insign1;

reg [31:0]index[0:49];

reg [31:0]identifier;

reg a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16;

reg b1,b2,b3,b4,b5,b6,b7,b8,b9,b10,b11,b12,b13,b14,b15,b16;

reg c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16;

reg d1,d2,d3,d4,d5,d6,d7,d8;

reg e1,e2,e3,e4,e5,e6,e7,opp1;

integer i=0,j,k=0,l,hh,c,reset=0,flag,flag1=0,z,rst;

wire m,n,o,p,d,e;

wire check;

reg check1;

// INITIALIZING ARRAY TO ZERO

/\*initial

begin

for(l=0;l<50;l=l+1)

index[l]=32'b0;

end\*/

//READING AN INCOMING DATA FILE FROM ARRAY1

initial

begin

$readmemb("indataex.txt",array1);

for(z=0;z<128;z=z+1)

$display("array1[%0d]=%b",z,array1[z]);

end

//INPUT POSEEDGE OF CLOCK

always@(posedge clk)

begin

if(check1==1)

opp <= opp1;

else

opp<= ~opp;

end

//READING A SIGNATURE DATA FILE FROM ARRAY2

initial

begin

$readmemb("insignex.txt",array2);

for(z=0;z<50;z=z+1)

$display("array2[%0d]=%b",z,array2[z]);

end

//COMPARING MSB BITS OR PREFIX MATCHING

assign check=1'b0;

initial

begin

for(j=0;j<50;j=j+1)

begin

insign=array2[j];

if(indata[15:12] == insign[15:12])

begin

if(indata[3:0]==insign1[3:0])

begin

if(indata[9:6]==insign1[9:6])

begin

a1 = insign1[0];

a2 = insign1[1];

a3 = insign1[2];

a4 = insign1[3];

a5 = insign1[4];

a6 = insign1[5];

a7 = insign1[6];

a8 = insign1[7];

a9 = insign1[8];

a10 = insign1[9];

a11 = insign1[10];

a12 = insign1[11];

a13 = insign1[12];

a14 = insign1[13];

a15 = insign1[14];

a16 = insign1[15];

//16 BIT MATCHED SIGNATURE (PREFIX ,SUFFIX AND CENTRE BITS)

b1 = indata[0];

b2 = indata[1];

b3 = indata[2];

b4 = indata[3];

b5 = indata[4];

b6 = indata[5];

b7 = indata[6];

b8 = indata[7];

b9 = indata[8];

b10 = indata[9];

b11 = indata[10];

b12 = indata[11];

b13 = indata[12];

b14 = indata[13];

b15 = indata[14];

b16 = indata[15];

//FILTER USING XNOR GATE

c1= (~a1 & ~b1)|(a1 & b1);

c2= (~a2 & ~b2)|(a2 & b2);

c3= (~a3 & ~b3)|(a3 & b3);

c4= (~a4 & ~b4)|(a4 & b4);

c5= (~a5 & ~b5)|(a5 & b5);

c6= (~a6 & ~b6)|(a6 & b6);

c7= (~a7 & ~b7)|(a7 & b7);

c8= (~a8 & ~b8)|(a8 & b8);

c9= (~a9 & ~b9)|(a9 & b9);

c10= (~a10 & ~b10)|(a10 & b10);

c11= (~a11 & ~b11)|(a11 & b11);

c12= (~a12 & ~b12)|(a12 & b12);

c13= (~a13 & ~b13)|(a13 & b13);

c14= (~a14 & ~b14)|(a14 & b14);

c15= (~a15 & ~b15)|(a15 & b15);

c16= (~a16 & ~b16)|(a16 & b16);

// COMBINING OUTPUT FROM XNOR GATE TO AND GATE

d1 = c1 & c2;

d2 = c3 & c4;

d3 = c5 & c6;

d4 = c7 & c8;

d5 = c9 & c10;

d6 = c11 & c12;

d7 = c13 & c14;

d8 = c15 & c16;

e1 = d1 & d2;

e2 = d3 & d4;

e3 = d5 & d6;

e4 = d7 & d8;

e5 = e1 & e2;

e6 = e3 & e4;

opp1 = e5 & e6;

if(opp1)

check1=1;

//check1=check;

end

end

end

end

end

//assign check=

//assign check = (opp) ? opp:1'b0;

/\*always@(opp)

begin

if(opp==1)

rst=0;

else

rst=1;

end\*/

/\*index[k]=j;

//$display("index [%0d]=%b",k,index[k]);

k=k+1;

//flag1=1;

end

end

end

identifier=k;

end

if(identifier>0)

begin

if (indata != 16'b0)

begin

for(c=0;c<50;c=c+1)

begin

if(c<identifier)

begin

hh=index[c];

insign1=array2[hh];

//$display("the prefix matched: array value is array2[%0d] =%b",hh,insign1);

//SUFFIX MATCHING OR LSB BITS

if(indata[3:0]==insign1[3:0])

begin

//$display("the suffix matched: array value is array2[%0d] =%b",hh,insign1);

//COMPARING CENTRE BITS

if(indata[9:6]==insign1[9:6])

begin

//$display("the centre bits matched: array value is array2[%0d] =%b",hh,insign1);

//16 BIT INCOMING DATA

a1 = insign1[0];

a2 = insign1[1];

a3 = insign1[2];

a4 = insign1[3];

a5 = insign1[4];

a6 = insign1[5];

a7 = insign1[6];

a8 = insign1[7];

a9 = insign1[8];

a10 = insign1[9];

a11 = insign1[10];

a12 = insign1[11];

a13 = insign1[12];

a14 = insign1[13];

a15 = insign1[14];

a16 = insign1[15];

//16 BIT MATCHED SIGNATURE (PREFIX ,SUFFIX AND CENTRE BITS)

b1 = indata[0];

b2 = indata[1];

b3 = indata[2];

b4 = indata[3];

b5 = indata[4];

b6 = indata[5];

b7 = indata[6];

b8 = indata[7];

b9 = indata[8];

b10 = indata[9];

b11 = indata[10];

b12 = indata[11];

b13 = indata[12];

b14 = indata[13];

b15 = indata[14];

b16 = indata[15];

//FILTER USING XNOR GATE

c1= (~a1 & ~b1)|(a1 & b1);

c2= (~a2 & ~b2)|(a2 & b2);

c3= (~a3 & ~b3)|(a3 & b3);

c4= (~a4 & ~b4)|(a4 & b4);

c5= (~a5 & ~b5)|(a5 & b5);

c6= (~a6 & ~b6)|(a6 & b6);

c7= (~a7 & ~b7)|(a7 & b7);

c8= (~a8 & ~b8)|(a8 & b8);

c9= (~a9 & ~b9)|(a9 & b9);

c10= (~a10 & ~b10)|(a10 & b10);

c11= (~a11 & ~b11)|(a11 & b11);

c12= (~a12 & ~b12)|(a12 & b12);

c13= (~a13 & ~b13)|(a13 & b13);

c14= (~a14 & ~b14)|(a14 & b14);

c15= (~a15 & ~b15)|(a15 & b15);

c16= (~a16 & ~b16)|(a16 & b16);

// COMBINING OUTPUT FROM XNOR GATE TO AND GATE

d1 = c1 & c2;

d2 = c3 & c4;

d3 = c5 & c6;

d4 = c7 & c8;

d5 = c9 & c10;

d6 = c11 & c12;

d7 = c13 & c14;

d8 = c15 & c16;

e1 = d1 & d2;

e2 = d3 & d4;

e3 = d5 & d6;

e4 = d7 & d8;

e5 = e1 & e2;

e6 = e3 & e4;

opp = e5 & e6;

end

else

opp=0;

end

else

opp=0;

end

end

end

if(c==identifier)

identifier=32'b0;

end

end\*/

//KNOWING WHICH INDEX HAS MATCHED

endmodule