`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 04:20:37 06/27/2021

// Design Name:

// Module Name: fir2d16x16

// Project Name:

// Target Devices:

// Tool versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module fir2d16x16(in,sc,clk,load,yn,a00,a01,a02,a03,a04,a05,a06,a07,a08,a09,a010,a011,a012,a013,a014,a015);

input [7:0]in;

input clk,sc,load;

input [7:0]a00,a01,a02,a03,a04,a05,a06,a07,a08,a09,a010,a011,a012,a013,a014,a015;

wire [7:0]io1,io2,io4,io8,yo1,yo2,yo4,yo8;//L//

wire [8:0]io3,io5,io6,io9,io10,io12,yo3,yo5,yo6,yo9,yo10,yo12;//L+1//

wire [9:0]io7,io11,io13,io14,io15,yo7,yo11,yo13,yo14,yo15;//L+2//

wire [11:0]w5,w6,w7,w8;

wire [8:0]w9,w10,w11;

wire [7:0]o1,o2,o3;

wire [10:0]w1,w2,w3,w4;

wire [10:0]s1,c1,s2,c2,s3,c3,s4,c4;

wire [3:0]sel1,sel2,sel3,sel4,sel5,sel6,sel7,sel8;

wire en;

output [7:0]yn;

wire [9:0]y1,y2,y3,y4,yn1,yn2,yn3,yn4;

enable en1(clk,en);

coeff coe1(a00,a01,a02,a03,clk,en,sel1,load);

coeff coe2(a04,a05,a06,a07,clk,en,sel2,load);

coeff coe3(a08,a09,a010,a011,clk,en,sel3,load);

coeff coe4(a012,a013,a014,a015,clk,en,sel4,load);

DA\_8 daf1(in,clk,en,io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13,io14,io15);

DAA psf1(io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13,io14,io15,sel1,y1),

psf2(io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13,io14,io15,sel2,y2),

psf3(io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13,io14,io15,sel3,y3),

psf4(io1,io2,io3,io4,io5,io6,io7,io8,io9,io10,io11,io12,io13,io14,io15,sel4,y4);

assign w1=y1;

assign w2=y2;

assign w3=y3;

assign w4=y4;

CSA11 cs1(w1,sc,clk,en,s1,c1),

cs2(w2,sc,clk,en,s2,c2),

cs3(w3,sc,clk,en,s3,c3),

cs4(w4,sc,clk,en,s4,c4);

assign w5=c1+(s1>>1);

assign w6=c2+(s2>>1);

assign w7=c3+(s3>>1);

assign w8=c4+(s4>>1);

shft sh1(w5[11:4],clk,o1,en);

assign w9=w6[11:4]+o1;

shft sh2(w9[8:1],clk,o2,en);

assign w10=w7[11:4]+o2;

shft sh3(w10[8:1],clk,o3,en);

assign w11=w8[11:4]+o3;

dff1 df1(w11[8:1],clk,en,yn);

endmodule

module DAA(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,sl,y);

input [3:0]sl;

output [9:0]y;

wire [9:0]X0;

input [7:0]X1,X2,X4,X8;//L//

input [8:0]X3,X5,X6,X9,X10,X12;//L+1//

input [9:0]X7,X11,X13,X14,X15;//L+2//

wire [9:0]XN1,XN2,XN4,XN8,XN3,XN5,XN6,XN9,XN10,XN12;

assign X0=10'd0;

assign XN1={2'b0,X1};

assign XN2={2'b0,X2};

assign XN4={2'b0,X4};

assign XN8={2'b0,X8};

assign XN3={1'b0,X3};

assign XN5={1'b0,X5};

assign XN6={1'b0,X6};

assign XN9={1'b0,X9};

assign XN10={1'b0,X10};

assign XN12={1'b0,X12};

mux\_16 m1(

.In0(X0),

.In1(XN1),

.In2(XN2),

.In3(XN3),

.In4(XN4),

.In5(XN5),

.In6(XN6),

.In7(X7),

.In8(XN8),

.In9(XN9),

.In10(XN10),

.In11(X11),

.In12(XN12),

.In13(X13),

.In14(X14),

.In15(X15),

.sel(sl),

.Y(y)

);

endmodule

module DA\_8(In,Clk,Reset,O1,O2,O3,O4,O5,O6,O7,O8,O9,O10,O11,O12,O13,O14,O15);

input [7:0]In;

output [7:0]O1,O2,O4,O8;//L//

input Clk,Reset;

output [8:0]O3,O5,O6,O9,O10,O12;//L+1//

output [9:0]O7,O11,O13,O14,O15;//L+2//

wire [8:0]A,B,D;

wire [9:0]C,E,F,G;

DFF8 M1(

.In(In),

.Clk(Clk),

.Reset(Reset),

.Out(O1)

);

DFF8 M2(

.In(O1),

.Clk(Clk),

.Reset(Reset),

.Out(O2)

);

assign A=In+O1;

DFF9 M3(

.In(A),

.Clk(Clk),

.Reset(Reset),

.Out(O3)

);

DFF8 M4(

.In(O2),

.Clk(Clk),

.Reset(Reset),

.Out(O4)

);

assign B=In+O2;

DFF9 M5(

.In(B),

.Clk(Clk),

.Reset(Reset),

.Out(O5)

);

DFF9 M6(

.In(O3),

.Clk(Clk),

.Reset(Reset),

.Out(O6)

);

assign C=In+O3;

DFF10 M7(

.In(C),

.Clk(Clk),

.Reset(Reset),

.Out(O7)

);

DFF8 M8(

.In(O4),

.Clk(Clk),

.Reset(Reset),

.Out(O8)

);

assign D=In+O4;

DFF9 M9(

.In(D),

.Clk(Clk),

.Reset(Reset),

.Out(O9)

);

DFF9 M10(

.In(O5),

.Clk(Clk),

.Reset(Reset),

.Out(O10)

);

assign E=In+O5;

DFF10 M11(

.In(E),

.Clk(Clk),

.Reset(Reset),

.Out(O11)

);

DFF9 M12(

.In(O6),

.Clk(Clk),

.Reset(Reset),

.Out(O12)

);

assign F=In+O6;

DFF10 M13(

.In(F),

.Clk(Clk),

.Reset(Reset),

.Out(O13)

);

DFF10 M14(

.In(O7),

.Clk(Clk),

.Reset(Reset),

.Out(O14)

);

assign G=In+O7;

DFF10 M15(

.In(G),

.Clk(Clk),

.Reset(Reset),

.Out(O15)

);

endmodule

module mux\_16(In0,In1,In2,In3,In4,In5,In6,In7,In8,In9,In10,In11,In12,In13,In14,In15,sel,Y);

input [9:0]In0,In1,In2,In3,In4,In5,In6,In7,In8,In9,In10,In11,In12,In13,In14,In15;

input [3:0]sel;

output [9:0]Y;

reg [9:0] Y;

always @(sel or In0 or In1 or In2 or In3 or In4 or In5 or In6 or In7 or In8 or In9 or In10 or In11 or In12 or In13 or In14 or In15 or Y)

begin

if (sel == 4'd0)

Y = In0;

else if(sel == 4'd1)

Y = In1;

else if(sel == 4'd2)

Y = In2;

else if(sel == 4'd3)

Y = In3;

else if(sel == 4'd4)

Y = In4;

else if(sel == 4'd5)

Y = In5;

else if(sel == 4'd5)

Y = In5;

else if(sel == 4'd6)

Y = In6;

else if(sel == 4'd7)

Y = In7;

else if(sel == 4'd8)

Y = In8;

else if(sel == 4'd9)

Y = In9;

else if(sel == 4'd10)

Y = In10;

else if(sel == 4'd11)

Y = In11;

else if(sel == 4'd12)

Y = In12;

else if(sel == 4'd13)

Y = In13;

else if(sel == 4'd14)

Y = In14;

else if(sel == 4'd15)

Y = In15;

end

endmodule

module CSA11(Y,Sc,Clk,Reset,S,C);

input Sc,Clk,Reset;

input [10:0]Y;

output [10:0]S,C;

wire E1,E2,E3,E4,E5,E6,E7,E8,E9,E10,E11,a0,a1,a2,a3,a4,a5,a7,a8,a9,b0,b1,b2,b3,b4,b5,b6,b7,b9,a10,b10;

xor M1(E1,Sc,Y[0]);

FullAdder M2(

.A(E1),

.B(S[1]),

.Cin(C[0]),

.Sum(a0),

.Cout(b0)

);

DFF M3(

.In(a0),

.Clk(Clk),

.Reset(Reset),

.Out(S[0])

);

DFF M4(

.In(b0),

.Clk(Clk),

.Reset(Reset),

.Out(C[0])

);

xor M5(E2,Sc,Y[1]);

FullAdder M6(

.A(E2),

.B(S[2]),

.Cin(C[1]),

.Sum(a1),

.Cout(b1)

);

DFF M7(

.In(a1),

.Clk(Clk),

.Reset(Reset),

.Out(S[1])

);

DFF M8(

.In(b1),

.Clk(Clk),

.Reset(Reset),

.Out(C[1])

);

xor M9(E3,Sc,Y[2]);

FullAdder M10(

.A(E3),

.B(S[3]),

.Cin(C[2]),

.Sum(a2),

.Cout(b2)

);

DFF M11(

.In(a2),

.Clk(Clk),

.Reset(Reset),

.Out(S[2])

);

DFF M12(

.In(b2),

.Clk(Clk),

.Reset(Reset),

.Out(C[2])

);

xor M13(E4,Sc,Y[3]);

FullAdder M14(

.A(E4),

.B(S[4]),

.Cin(C[3]),

.Sum(a3),

.Cout(b3)

);

DFF M15(

.In(a3),

.Clk(Clk),

.Reset(Reset),

.Out(S[3])

);

DFF M16(

.In(b3),

.Clk(Clk),

.Reset(Reset),

.Out(C[3])

);

xor M17(E5,Sc,Y[4]);

FullAdder M18(

.A(E5),

.B(S[5]),

.Cin(C[4]),

.Sum(a4),

.Cout(b4)

);

DFF M19(

.In(a4),

.Clk(Clk),

.Reset(Reset),

.Out(S[4])

);

DFF M20(

.In(b4),

.Clk(Clk),

.Reset(Reset),

.Out(C[4])

);

xor M21(E6,Sc,Y[5]);

FullAdder M22(

.A(E6),

.B(S[6]),

.Cin(C[5]),

.Sum(a5),

.Cout(b5)

);

DFF M23(

.In(a5),

.Clk(Clk),

.Reset(Reset),

.Out(S[5])

);

DFF M24(

.In(b5),

.Clk(Clk),

.Reset(Reset),

.Out(C[5])

);

xor M25(E7,Sc,Y[6]);

FullAdder M26(

.A(E7),

.B(S[7]),

.Cin(C[6]),

.Sum(a6),

.Cout(b6)

);

DFF M27(

.In(a6),

.Clk(Clk),

.Reset(Reset),

.Out(S[6])

);

DFF M28(

.In(b6),

.Clk(Clk),

.Reset(Reset),

.Out(C[6])

);

xor M29(E8,Sc,Y[7]);

FullAdder M30(

.A(E8),

.B(S[8]),

.Cin(C[7]),

.Sum(a7),

.Cout(b7)

);

DFF M31(

.In(a7),

.Clk(Clk),

.Reset(Reset),

.Out(S[7])

);

DFF M32(

.In(b7),

.Clk(Clk),

.Reset(Reset),

.Out(C[7])

);

xor M33(E9,Sc,Y[8]);

FullAdder M34(

.A(E9),

.B(S[9]),

.Cin(C[8]),

.Sum(a8),

.Cout(b8)

);

DFF M35(

.In(a8),

.Clk(Clk),

.Reset(Reset),

.Out(S[8])

);

DFF M36(

.In(b8),

.Clk(Clk),

.Reset(Reset),

.Out(C[8])

);

xor M37(E10,Sc,Y[9]);

FullAdder M38(

.A(E10),

.B(S[10]),

.Cin(C[9]),

.Sum(a9),

.Cout(b9)

);

DFF M39(

.In(a9),

.Clk(Clk),

.Reset(Reset),

.Out(S[9])

);

DFF M40(

.In(b9),

.Clk(Clk),

.Reset(Reset),

.Out(C[9])

);

xor M41(E11,Sc,Y[10]);

FullAdder M42(

.A(E11),

.B(S[10]),

.Cin(C[10]),

.Sum(a10),

.Cout(b10)

);

DFF M43(

.In(a10),

.Clk(Clk),

.Reset(Reset),

.Out(S[10])

);

DFF M44(

.In(b10),

.Clk(Clk),

.Reset(Reset),

.Out(C[10])

);

endmodule

module shft(a,clk,out,en);// byte shifter

input clk,en;

input [7:0]a;

output [7:0]out;

reg [7:0]r[15:0];

initial

begin r[0]=0;r[1]=0;r[2]=0;r[3]=0;r[4]=0;r[5]=0;r[6]=0;r[7]=0;r[8]=0;r[9]=0;r[10]=0;r[11]=0;r[12]=0;r[13]=0;r[14]=0;r[15]=0; end

assign out= r[15];

always @ (posedge clk)

begin

if(en)

begin

r[0]<=a;

r[1]<=r[0];

r[2]<=r[1];

r[3]<=r[2];

r[4]<=r[3];

r[5]<=r[4];

r[6]<=r[5];

r[7]<=r[6];

r[8]<=r[7];

r[9]<=r[8];

r[10]<=r[9];

r[11]<=r[10];

r[12]<=r[11];

r[13]<=r[12];

r[14]<=r[13];

r[15]<=r[14];

end

end

endmodule

module DFF(In,Clk,Reset,Out);

input Clk,Reset;

input In;

output Out;

reg Out;

initial

Out=0;

always @ (posedge Clk)

begin

if (Reset==1)

Out<=1'b0;

else

Out<=In;

end

endmodule

module dff1(In,Clk,Reset,Out);

input Clk,Reset;

input [7:0]In;

output [7:0]Out;

reg Out;

initial

Out=0;

always @ (posedge Clk)

begin

if (Reset==1)

Out<=In;

end

endmodule

module DFF9(In,Clk,Reset,Out);

input Clk,Reset;

input [8:0]In;

output [8:0]Out;

reg [8:0]Out=0;

always @ (posedge Clk)

begin

if (Reset==1)

Out<=In;

end

endmodule

module DFF8(In,Clk,Reset,Out);

input Clk,Reset;

input [7:0]In;

output [7:0]Out;

reg [7:0]Out=0;

always @ (posedge Clk)

begin

if (Reset)

Out<=In;

end

endmodule

module DFF10(In,Clk,Reset,Out);

input Clk,Reset;

input [9:0]In;

output [9:0]Out;

reg [9:0]Out=0;

always @ (posedge Clk)

begin

if (Reset)

Out<=In;

end

endmodule

module enable(clk,out);

input clk;

output out;

reg[8:0]in=9'b000000001;

assign out=in[0];

always@(posedge clk)

begin

in[0]<=in[1];

in[1]<=in[2];

in[2]<=in[3];

in[3]<=in[4];

in[4]<=in[5];

in[5]<=in[6];

in[6]<=in[7];

in[7]<=in[8];

in[8]<=in[0];

end

endmodule

module coeff(a,b,c,d,clk,en,sel,load);

input [7:0]a,b,c,d;

input clk,load,en;

output [3:0]sel;

shifter shft1(a,clk,load,sel[0],en),

shft2(b,clk,load,sel[1],en),

shft3(c,clk,load,sel[2],en),

shft4(d,clk,load,sel[3],en);

endmodule

module shifter(i,clk,load,out,en);//for getting bits of coefficient

input clk,load,en;

output out;

input [7:0]i;

reg[7:0]in=0;

assign out=in[0];

always@(posedge clk)

begin

if(load)

in<=i;

else if(!en)

begin

in[0]<=in[1];

in[1]<=in[2];

in[2]<=in[3];

in[3]<=in[4];

in[4]<=in[5];

in[5]<=in[6];

in[6]<=in[7];

in[7]<=in[0];

end

end

endmodule

module S\_ha(A,B,S,C);

input A,B;

output S,C;

assign S=A^B;

assign C=A&B;

endmodule

module FullAdder(A,B,Cin,Sum,Cout);

input A,B,Cin;

output Sum,Cout;

wire c,d,e;

S\_ha S1(

.A(A),

.B(B),

.S(c),

.C(d)

);

S\_ha S2(

.A(c),

.B(Cin),

.S(Sum),

.C(e)

);

or S3(Cout,e,d);

endmodule