**Adaptive filter**

**module delay (in,clk,out,en);**

**input clk,en;**

input [7:0]in;

output [7:0]out;

reg [7:0]out;

initial

begin

out=8'd0;

end

always @ (posedge clk)

begin

if(!en)

out=in;

end

endmodule

**module and1(Dout,y,z);**

**input [7:0]y;**

output [7:0]z;

input Dout;

and w1(z[0],Dout,y[0]);

and w2(z[1],Dout,y[1]);

and w3(z[2],Dout,y[2]);

and w4(z[3],Dout,y[3]);

and w5(z[4],Dout,y[4]);

and w6(z[5],Dout,y[5]);

and w7(z[6],Dout,y[6]);

and w8(z[7],Dout,y[7]);

endmodule

module **LUT(x0**,x1,x2,x3,v1,v2,v3,v4,z,clk,en,sign);

input [7:0]x0,x1,x2,x3;

wire [9:0]k;

output [7:0]z;

wire [7:0]t0,t1,t2,t3;

wire [8:0]t4,t5;

input clk,en,sign;

input v1,v2,v3,v4;

and1 b0(v1,x0,t0);

and1 b1(v2,x1,t1);

and1 b2(v3,x2,t2);

and1 b3(v4,x3,t3);

assign t4=t0+t1;

assign t5=t2+t3;

assign k=t4+t5;

//enable k1(clk,en);

acc q1(k,en,sign,z,clk);

endmodule

module **acc(**in,en,sign,z,clk);

input [9:0] in;

input clk;

input en,sign;

reg [9:0]k=0;

output reg [7:0]z=0;

//reg [5:0] y=0;

always@(posedge clk)

begin

if(en==1)

begin

if(sign)

begin

k<=in+(k>>1);

end

else if(!sign)

k<=in-(k>>1);

end

else if(!en)

begin

z<=k[9:2];

k<=0;

end

end

endmodule

**module filter**(x0,x1,x2,x3,w0,w1,w2,w3,z0,z1,z2,z3,d0,d1,d2,d3,e0,e1,e2,e3,a,b,c,d,u0,u1,u2,u3,clk);

input [7:0]x0,x1,x2,x3;

input clk;

output [7:0]z0,z1,z2,z3,e0,e1,e2,e3;

wire [7:0]x4,x5,x6;

wire en,sign;

input [7:0]d0,d1,d2,d3;

output a,b,c,d;

output [7:0]u0,u1,u2,u3;

output reg [7:0]w0,w1,w2,w3;

initial

begin

w0=0;

w1=0;

w2=0;

w3=0;

end

wire v1,v2,v3,v4;

delay s1(x0,clk,x4,en);

delay s2(x1,clk,x5,en);

delay s3(x2,clk,x6,en);

enable h1(clk,en,sign);

piso j1(w0,clk,v1,en);

piso j2(w1,clk,v2,en);

piso j3(w2,clk,v3,en);

piso j4(w3,clk,v4,en);

LUT l1(x0,x1,x2,x3,v1,v2,v3,v4,z0,clk,en,sign);

LUT l2(x1,x2,x3,x4,v1,v2,v3,v4,z1,clk,en,sign);

LUT l3(x2,x3,x4,x5,v1,v2,v3,v4,z2,clk,en,sign);

LUT l4(x3,x4,x5,x6,v1,v2,v3,v4,z3,clk,en,sign);

error\_comp n1(z0,z1,z2,z3,d0,d1,d2,d3,e0,e1,e2,e3,a,b,c,d,clk,en);

error n2(x0,x1,x2,x3,a,b,c,d,u0,clk,en,sign);

error n3(x1,x2,x3,x4,a,b,c,d,u1,clk,en,sign);

error n4(x2,x3,x4,x5,a,b,c,d,u2,clk,en,sign);

error n5(x3,x4,x5,x6,a,b,c,d,u3,clk,en,sign);

always @(posedge clk)

begin

w0=w0+(u0>>2);

w1=w1+(u1>>2);

w2=w2+(u2>>2);

w3=w3+(u3>>2);

end

endmodule

module error\_comp(y0,y1,y2,y3,d0,d1,d2,d3,e0,e1,e2,e3,a,b,c,d,clock,en);

input [7:0]y0,y1,y2,y3,d0,d1,d2,d3;

output [7:0]e0,e1,e2,e3;

output a,b,c,d;

input clock;

input en;

assign e0=d0-y0;

assign e1=d1-y1;

assign e2=d2-y2;

assign e3=d3-y3;

//enable s1(clock,en);

piso p1( e0,clock,a,en);

piso p2( e1,clock,b,en);

piso p3( e2,clock,c,en);

piso p4( e3,clock,d,en);

endmodule

module error(x0,x1,x2,x3,v1,v2,v3,v4,z,clk,en,sign);

input [7:0]x0,x1,x2,x3;

output [7:0]z;

wire [7:0]t0,t1,t2,t3;

wire [8:0]t4,t5;

input clk;

wire [9:0]k;

input v1,v2,v3,v4;

input en,sign;

and1 b0(v1,x0,t0);

and1 b1(v2,x1,t1);

and1 b2(v3,x2,t2);

and1 b3(v4,x3,t3);

assign t4=t0+t1;

assign t5=t2+t3;

assign k=t4+t5;

//enable k1(clk,en);

acc q1(k,en,sign,z,clk);

endmodule

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

input clk,en;

output out;

input [7:0]i;

reg[7:0]in=0;

assign out=in[0];

always@(posedge clk)

begin

if(!en)

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 enable(clk,out,sign);

input clk;

output out,sign;

reg[8:0]in1=9'b111111110;

reg[7:0]in2=8'b11111110;

assign out=in1[0];

assign sign=in2[0];

always@(posedge clk)

begin

in1[0]<=in1[1];

in1[1]<=in1[2];

in1[2]<=in1[3];

in1[3]<=in1[4];

in1[4]<=in1[5];

in1[5]<=in1[6];

in1[6]<=in1[7];

in1[7]<=in1[8];

in1[8]<=in1[0];

in2[0]<=in2[1];

in2[1]<=in2[2];

in2[2]<=in2[3];

in2[3]<=in2[4];

in2[4]<=in2[5];

in2[5]<=in2[6];

in2[6]<=in2[7];

in2[7]<=in2[0];

end

endmodule