```
          else if(en)
          beginmodule mac16ved(ac_val,cout,a,b,clk,rst,en);
          output reg[15:0]ac_val;
          output cout;
          input [7:0]a,b;
   input clk,rst,en;
          wire [15:0]p,s;
   reg [15:0] pipo;
   reg [15:0] product;

          vedic8bit v1(p,a,b);
          rca16bit s1(s,cout,pipo,product,1'b0);
          always@(posedge clk)
            if(rst)
            ac_val= 16'h0000;
            else if(en)
           begin
           ac_val =s;
          end

          always@(negedge clk)
           if(rst)
           product = 16'h0000;

            product = p;
           end

            always@(posedge clk)
           if(rst)
            pipo = 16'h0000;
           else if(en)
            pipo = s;


endmodule

module vedic8bit(t,a,b);
input [7:0]a,b;
output [15:0]t;
wire [7:0]w,w1,w2,w3,w4,w5,w6;
wire ca1,ca2,c,ca3;

vedic4bit x1(w[7:0],a[3:0],b[3:0]);
vedic4bit x2(w1[7:0],a[3:0],b[7:4]);
```

```
vedic4bit x3(w2[7:0],a[7:4],b[3:0]);
vedic4bit x4(w3[7:0],a[7:4],b[7:4]);

assign t[3:0]=w[3:0];
rca8bit n1(w4[7:0],ca1,w1[7:0],w2[7:0],1'b0);

rca8bit n2(w5[7:0],ca2,{4'b0,w[7:4]},w4[7:0],1'b0);
assign t[7:4]=w5[3:0];
//or qwe(c,ca1,ca2);

rca8bit n3(w6[7:0],ca3,w3[7:0],{3'b0,ca1,w5[7:4]},1'b0);

assign t[15:8]=w6[7:0];
//assign ca3;
endmodule

module vedic4bit(p,a,b);
input [3:0]a,b;
output [7:0]p;
wire [32:1]w;

andg z1(p[0],a[0],b[0]);// and block

andg z2(w[1],a[1],b[0]);
andg z3(w[2],a[0],b[1]);
hag z4(p[1],w[3],w[1],w[2]);  //1-1st ha adder

andg z5(w[4],a[2],b[0]);
andg z6(w[5],a[1],b[1]);
andg z7(w[6],a[0],b[2]);
fag z8(w[7],w[8],w[4],w[5],w[6]);  //1-1 full adder
hag z9(p[2],w[9],w[3],w[7]); //3-1 half addr

andg z10(w[10],a[3],b[0]);
andg z11(w[11],a[2],b[1]);
andg z12(w[12],a[1],b[2]);
fag z13(w[13],w[14],w[10],w[11],w[12]); //1-2 full addr

andg z14(w[15],a[0],b[3]);
hag z15(w[16],w[17],w[13],w[15]); //2-1 ha

fag z16(p[3],w[18],w[16],w[8],w[9]);//3-1 fa

andg z17(w[19],a[3],b[1]);
andg z19(w[20],a[2],b[2]);
andg z20(w[21],a[1],b[3]);
fag z21(w[22],w[23],w[20],w[21],w[19]);//1-3 full addr
hag z22(w[24],w[25],w[22],w[14]); //2-2 full addr
fag z23(p[4],w[26],w[18],w[17],w[24]);//3-2 full addr

andg z24(w[27],a[2],b[3]);
andg z25(w[28],a[3],b[2]);
fag z26(w[29],w[30],w[23],w[27],w[28]);  // 1-3 full adder
fag z27(p[5],w[31],w[26],w[25],w[29]); //3-3 full adder

andg z28(w[32],a[3],b[3]);
fag z29(p[6],p[7],w[31],w[30],w[32]);
endmodule

module hag(s,c,a,b);
input a,b;
output s,c;
xor a1(s,a,b);
and a2(c,a,b);
endmodule

module andg(output y,input a,b);

and a1(y,a,b);

endmodule

module fag(s,c,a,b,cin);
input a,b,cin;
output s,c;
wire [3:1]w;
xor x1(w[1],a,b);
and x2(w[2],a,b);
xor x3(s,w[1],cin);
and x4(w[3],cin,w[1]);
or x5(c,w[2],w[3]);
endmodule


module rca16bit(s,cout,a,b,cin);
input [15:0]a,b;
output [15:0]s;
input cin;
```

```
output cout;
wire cw;

rca8bit r0(s[7:0],cw,a[7:0],b[7:0],1'b0);
rca8bit r1(s[15:8],cout,a[15:8],b[15:8],cw);

endmodule

module rca8bit(s,cout,a,b,cin);
input [7:0]a,b;input cin;
output [7:0]s;
output cout;
wire [7:1]w;

//hag x1(s[0],w[1],a[0],b[0]);
fag x1(s[0],w[1],a[0],b[0],cin);
fag x2(s[1],w[2],a[1],b[1],w[1]);
fag x3(s[2],w[3],a[2],b[2],w[2]);
fag x4(s[3],w[4],a[3],b[3],w[3]);
fag x5(s[4],w[5],a[4],b[4],w[4]);
fag x6(s[5],w[6],a[5],b[5],w[5]);
fag x7(s[6],w[7],a[6],b[6],w[6]);
fag x8(s[7],cout,a[7],b[7],w[7]);
endmodule
```