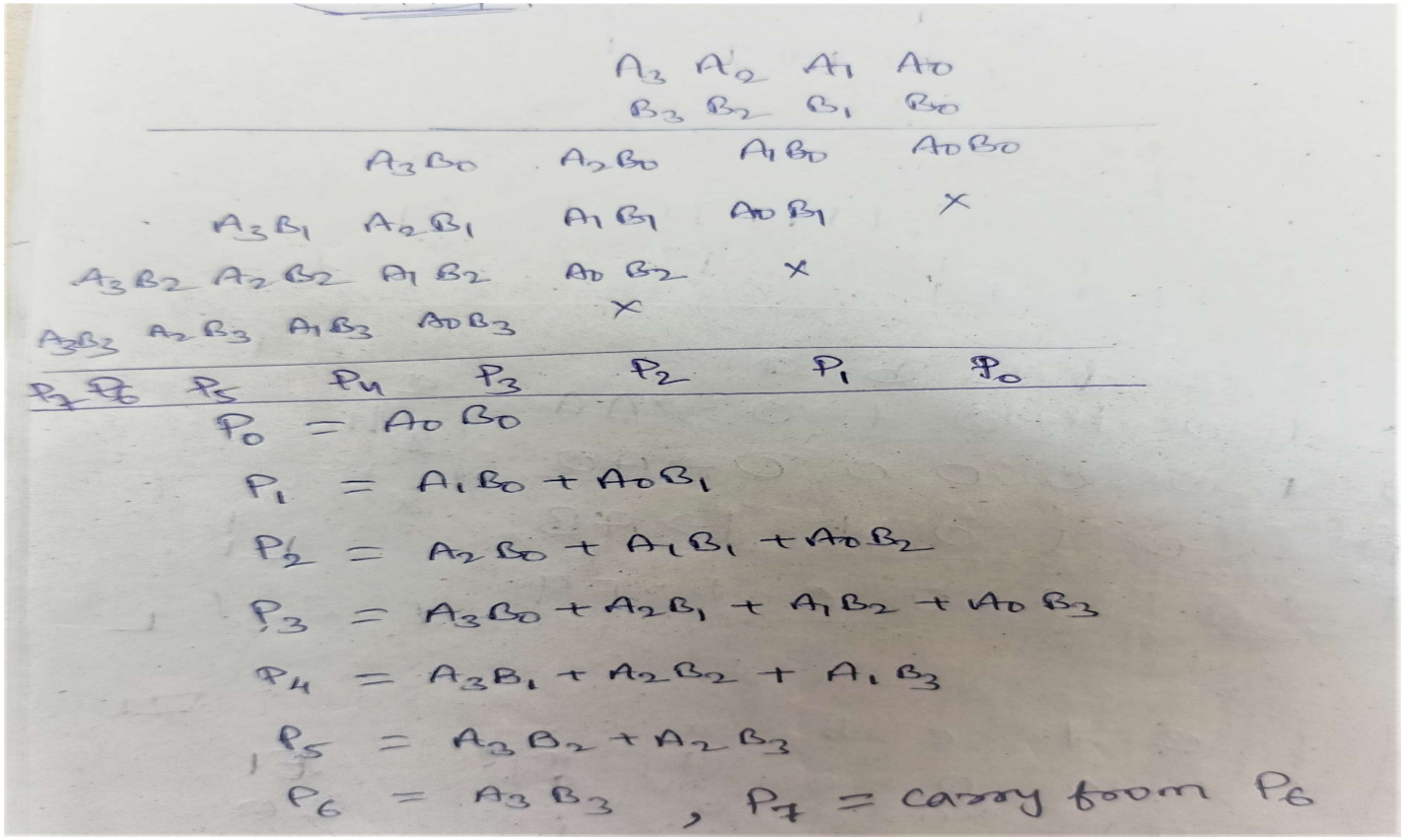
SDSD LAB PROJECT

Topic:-Vedic Multiplier(4 and 8 Bit)

PRESENTED By:-

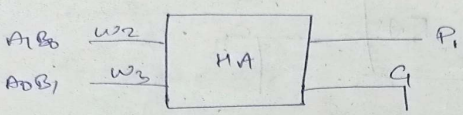
Narendra Kumar Nehra

4 Bit Vedic Multiplier

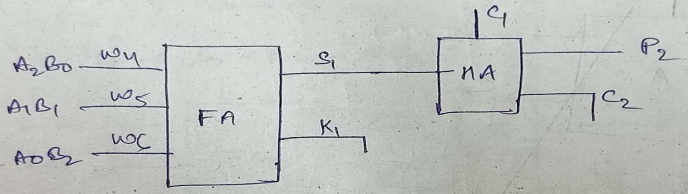


A_0B_0 ——— P_0

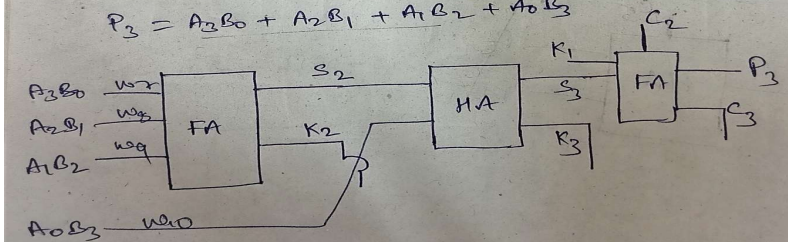
$P_1 = A_1B_0 + A_0B_1$



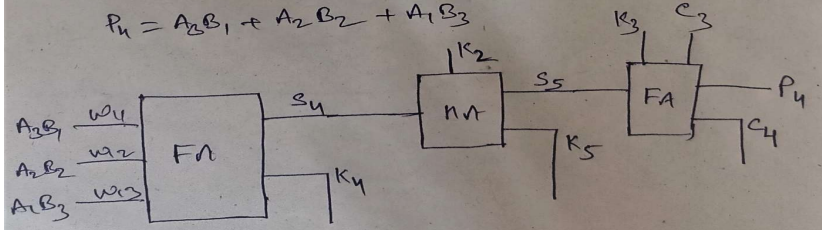
$P_2 = A_2B_0 + A_1B_1 + A_0B_2$



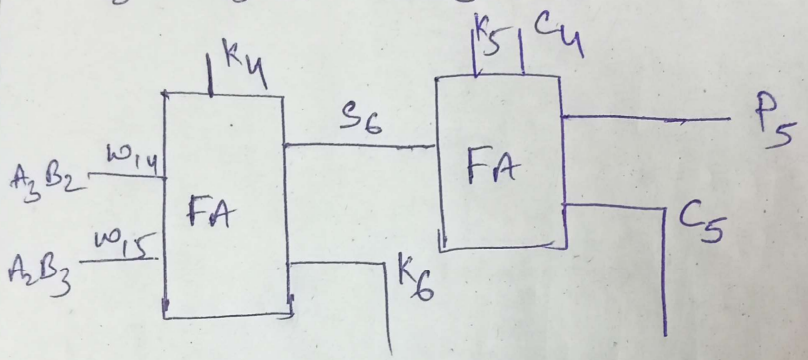
$P_3 = A_2B_0 + A_2B_1 + A_1B_2 + A_0B_3$



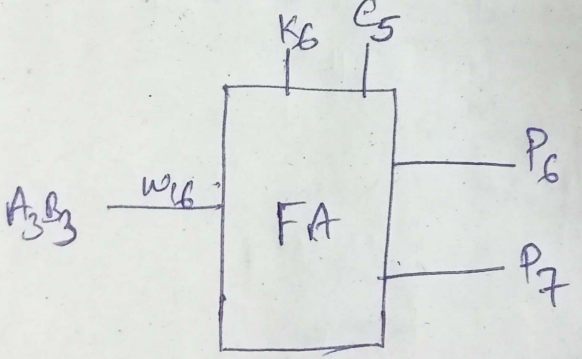
$P_4 = A_2B_1 + A_2B_2 + A_1B_3$



$P_5 = A_3B_2 + A_2B_3$



$P_6 = A_3B_3$

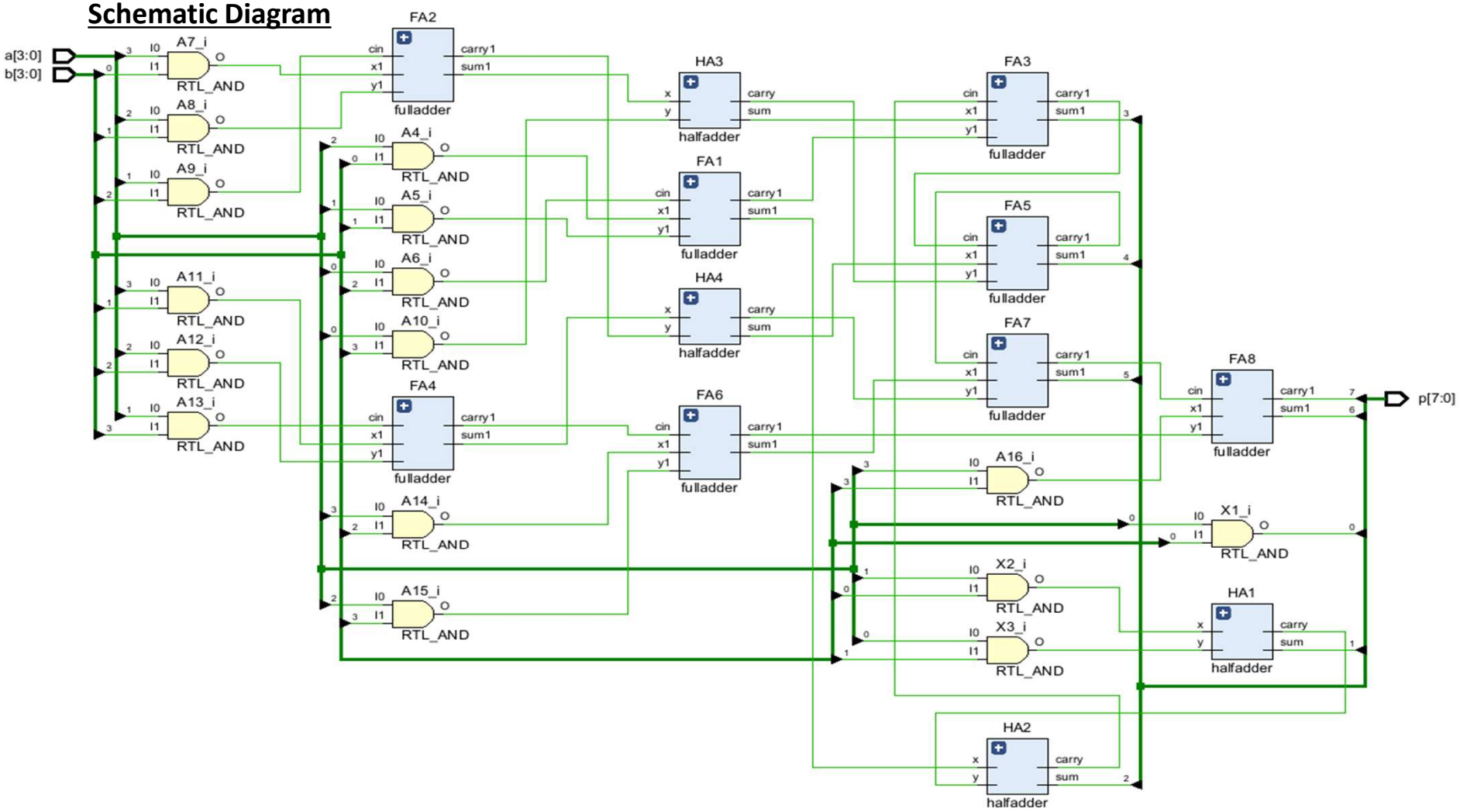


Code for 4 Bit

```
12
13      module fourbitmultiplier(p,a,b);
14      input [3:0]a;
15      input [3:0]b;
16      output [7:0]p;
17      wire [7:0]p;
18      wire c1,c2,c3,c4,c5,k1,k2,k3,k4,k5,k6,s1,s2,s3,s4,s5,s6;
19      wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16;
20      and X1(p[0],a[0],b[0]);
21
22      and X2(w2,a[1],b[0]);
23      and X3(w3,a[0],b[1]);
24      halfadder HA1(p[1],c1,w2,w3);
25
26      and A4(w4,a[2],b[0]);
27      and A5(w5,a[1],b[1]);
28      and A6(w6,a[0],b[2]);
29      fulladder FA1(s1,k1,w4,w5,w6);
30      halfadder HA2(p[2],c2,s1,c1);
31
32      and A7(w7,a[3],b[0]);
33      and A8(w8,a[2],b[1]);
34      and A9(w9,a[1],b[2]);
35      fulladder FA2(s2,k2,w7,w8,w9);
36      and A10(w10,a[0],b[3]);
37      halfadder HA3(s3,k3,s2,w10);
38      fulladder FA3(p[3],c3,s3,k1,c2);
39
40      and A11(w11,a[3],b[1]);
41      and A12(w12,a[2],b[2]);
42      and A13(w13,a[1],b[3]);
43      fulladder FA4(s4,k4,w11,w12,w13);
44      halfadder HA4(s5,k5,s4,k2);
45      fulladder FA5(p[4],c4,s5,k3,c3);
46
47      and A14(w14,a[3],b[2]);
48      and A15(w15,a[2],b[3]);
49      fulladder FA6(s6,k6,w14,w15,k4);
50      fulladder FA7(p[5],c5,s6,k5,c4);
51
52      and A16(w16,a[3],b[3]);
53      fulladder FA8(p[6],p[7],w16,k6,c5);
54      endmodule
55
```

```
51
52      and A16(w16,a[3],b[3]);
53      fulladder FA8(p[6],p[7],w16,k6,c5);
54      endmodule
55
56
57      module halfadder(sum,carry,x,y);
58      input x,y;
59      output sum,carry;
60      wire sum,carry;
61      xor X1(sum,x,y);
62      and Y1(carry,x,y);
63      endmodule
64
65      module fulladder(sum1,carry1,x1,y1,cin);
66      input x1,y1,cin;
67      output sum1,carry1;
68      wire k1,k2,k3;
69      xor X1(k1,x1,y1);
70      xor X2(sum1,cin,k1);
71      and X3(k2,cin,k1);
72      and X4(k3,x1,y1);
73      or X5(carry1,k3,k2);
74      endmodule
```

Schematic Diagram



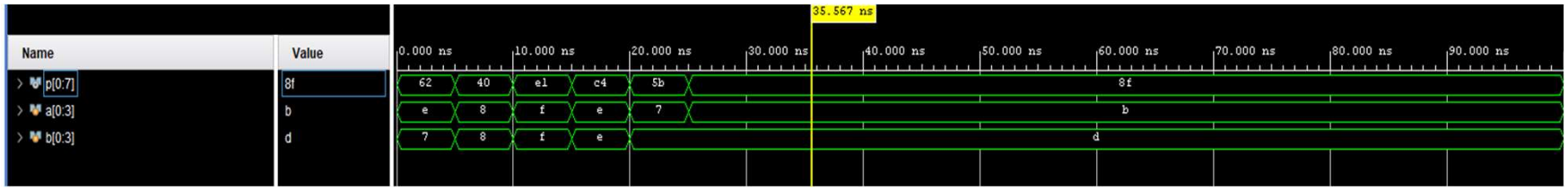
Test Bench

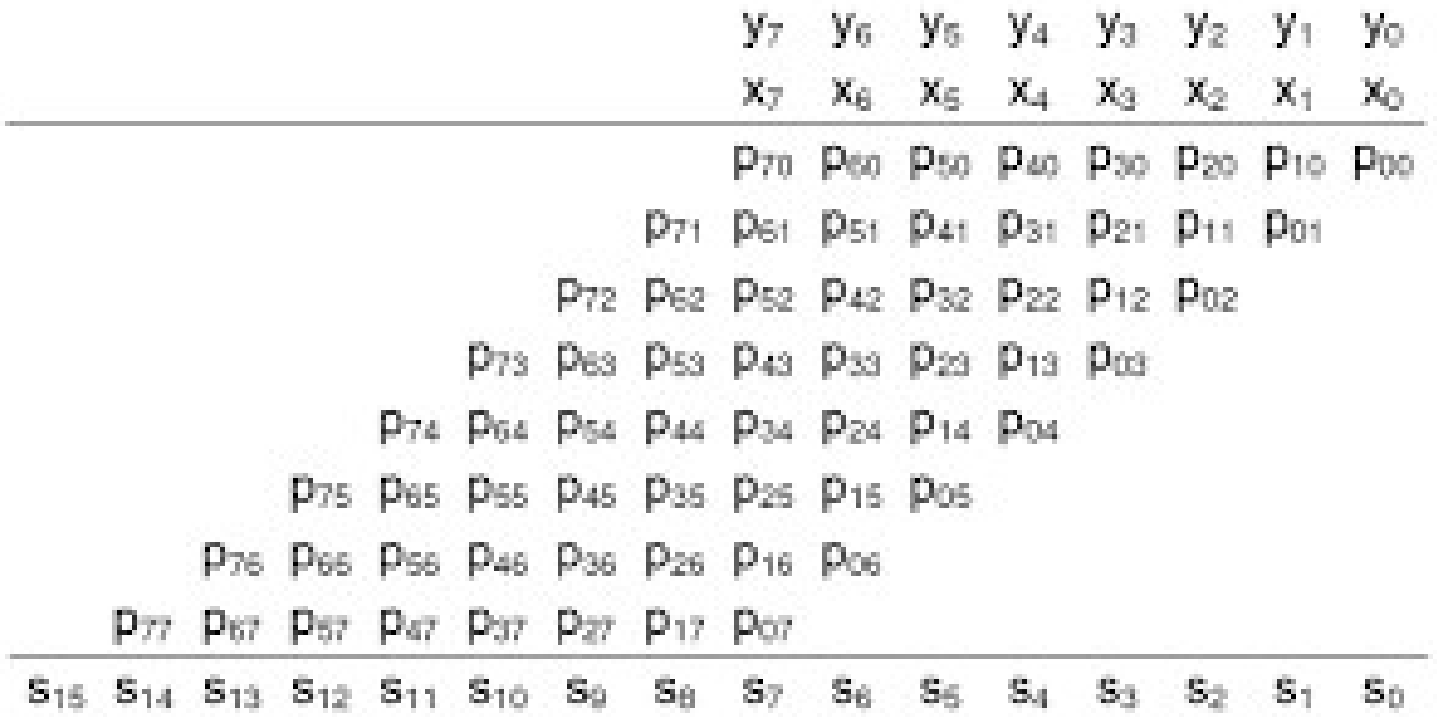
```
module fourbitmultiplier_tb();
wire [0:7]p;
reg [0:3]a,b;
fourbitmultiplier Multiiplier4_bit(p,a,b);
initial
begin
a=4'b1110; b=4'b0111;
#5 a=4'b1000; b=4'b1000;
#5 a=4'b1111; b=4'b1111;
#5 a=4'b1110; b=4'b1110;
#5 a=4'b0111; b=4'b1101;
#5 a=4'b1011; b=4'b1101;
end
initial $monitor($time,"a= %b,b= %b,p= %d", a, b, p);
initial #100 $stop;
endmodule
```

Result

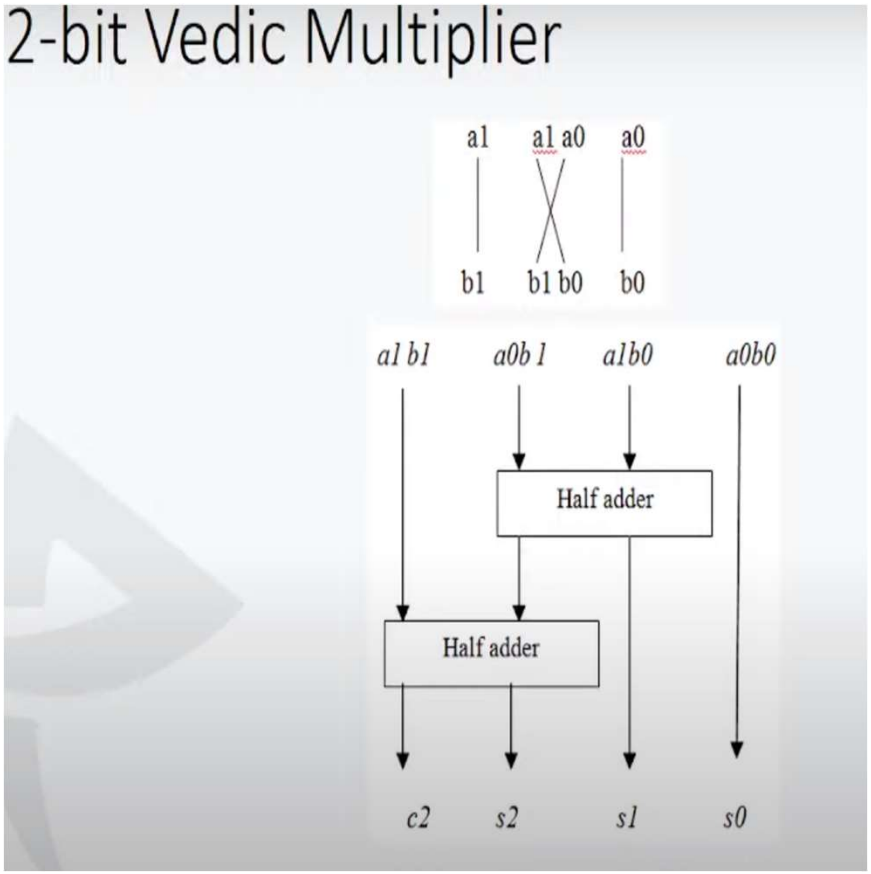
```
# run 1000ns
 0a= 1110,b= 0111,p=  98
 5a= 1000,b= 1000,p=  64
10a= 1111,b= 1111,p= 225
15a= 1110,b= 1110,p= 196
20a= 0111,b= 1101,p=  91
25a= 1011,b= 1101,p= 143
```

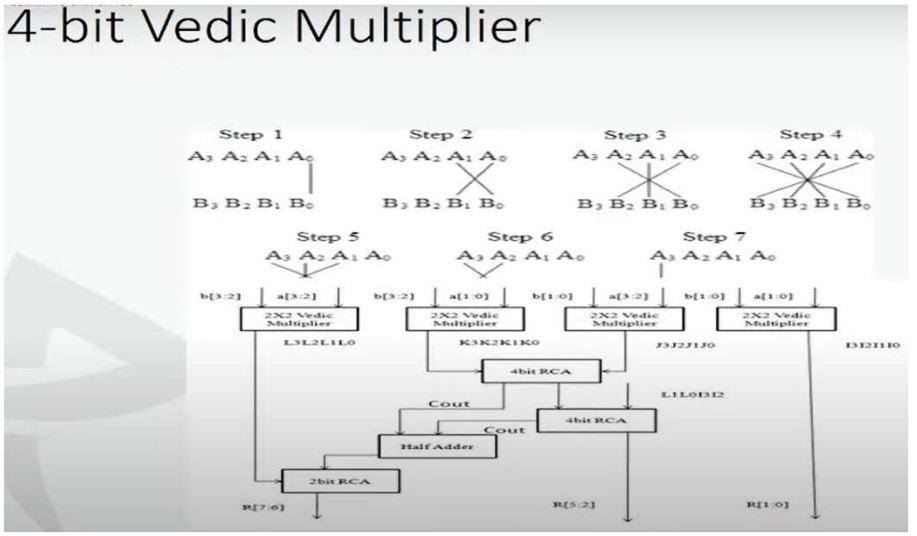
Waveform

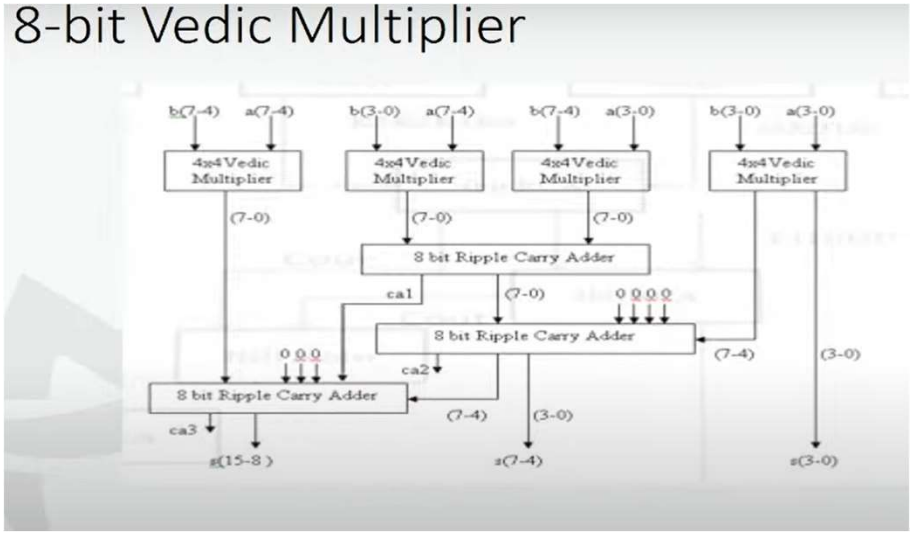



y7 y6 y5 y4 y3 y2 y1 y0
x7 x6 x5 x4 x3 x2 x1 x0
p70 p60 p50 p40 p30 p20 p10 p00
p71 p61 p51 p41 p31 p21 p11 p01
p72 p62 p52 p42 p32 p22 p12 p02
p73 p63 p53 p43 p33 p23 p13 p03
p74 p64 p54 p44 p34 p24 p14 p04
p75 p65 p55 p45 p35 p25 p15 p05
p76 p66 p56 p46 p36 p26 p16 p06
p77 p67 p57 p47 p37 p27 p17 p07
s15 s14 s13 s12 s11 s10 s9 s8 s7 s6 s5 s4 s3 s2 s1 s0

8 Bit Vedic Multiplier







Advantages

1. consumes 66% less area.
2. 76.1% less power.
3. 60% less delay

Code for 8-bit

```
module vedic8bit(output [15:0] out,input [7:0] in1,in2);
  wire[7:0] w1,w2,w3,w4,s1,s2;
  wire c1,c2,c3;
  wire [7:0] W1,W2;

 VM4bit VM4BIT_1(w1,in1[3:0],in2[3:0]);
 VM4bit VM4BIT_2(w2,in1[3:0],in2[7:4]);
 VM4bit VM4BIT_3(w3,in1[7:4],in2[3:0]);
 VM4bit VM4BIT_4(w4,in1[7:4],in2[7:4]);

  buf(s1[0],w1[4]);
  buf(s1[1],w1[5]);
  buf(s1[2],w1[6]);
  buf(s1[3],w1[7]);
  buf(s1[4],0);
  buf(s1[5],0);
  buf(s1[6],0);
  buf(s1[7],0);


   ADDER8bit A8_1( c1,W1, w2,w3);
   ADDER8bit A8_2( c2,W2,W1,s1);

   or o_1(c3,c1,c2);

  buf(s2[0],W2[4]);
  buf(s2[1],W2[5]);
  buf(s2[2],W2[6]);
  buf(s2[3],W2[7]);
  buf(s2[4],c3);
  buf(s2[5],0);
  buf(s2[6],0);
  buf(s2[7],0);



   assign out[0] = w1[0];
   assign out[1] = w1[1];
   assign out[2] = w1[2];
   assign out[3] = w1[3];
   assign out[4] = W2[0];
   assign out[5] = W2[1];
   assign out[6] = W2[2];
   assign out[7] = W2[3];


   ADDER8bit A8_3(extra,out [15:8], w4,s2);


endmodule


module VM4bit(output [7:0] out,input [3:0] in1,in2);
  wire[3:0] w1,w2,w3,w4,s1,s2;
  wire c1,c2,c3;
  wire [3:0] W1,W2;

 VM2bit VM2BIT_1(w1,in1[1:0],in2[1:0]);
 VM2bit VM2BIT_2(w2,in1[1:0],in2[3:2]);
 VM2bit VM2BIT_3(w3,in1[3:2],in2[1:0]);
 VM2bit VM2BIT_4(w4,in1[3:2],in2[3:2]);

  buf(s1[0],w1[2]);
  buf(s1[1],w1[3]);
  buf(s1[2],0);
  buf(s1[3],0);


   ADDER4bit A4_1( c1,W1, w2,w3);
   ADDER4bit A4_2( c2,W2,W1,s1);

   or o_1(c3,c1,c2);

  buf(s2[0],W2[2]);
  buf(s2[1],W2[3]);
  buf(s2[2],c3);
  buf(s2[3],0);



   assign out[0] = w1[0];
   assign out[1] = w1[1];
```

```verilog
   assign out[0] = w1[0];
   assign out[1] = w1[1];
   assign out[2] = W2[0];
   assign out[3] = W2[1];

   ADDER4bit A4_3(extra,out [7:4], w4,s2);


 endmodule

module VM2bit(output [3:0] out, input [1:0] A,B);

wire [3:0] s;

and a1(s[0],A[0],B[0]);
and a2(s[1],A[0],B[1]);
and a3(s[2],A[1],B[0]);
and a4(s[3],A[1],B[1]);

assign out[0] = s[0];
halfadder H_1(out[1],C,s[1],s[2]);
halfadder H_2(out[2],out[3],s[3],C);

endmodule

module ADDER8bit(C,out,A,B );
output [7:0] out;
output C;



input [7:0] A,B;
  halfadder H_1(out[0],c1,A[0],B[0]);
  fulladder F_1(out[1],c2,A[1],B[1],c1);
  fulladder F_2(out[2],c3,A[2],B[2],c2);
  fulladder F_3(out[3],c4,A[3],B[3],c3);
  fulladder F_4(out[4],c5,A[4],B[4],c4);
  fulladder F_5(out[5],c6,A[5],B[5],c5);
  fulladder F_6(out[6],c7,A[6],B[6],c6);
  fulladder F_7(out[7],C,A[7],B[7],c7);

endmodule
```

```verilog fulladder : module ```

```verilog
endmodule


module ADDER4bit(C,out,A,B );
output [3:0] out;
output C;


input [3:0] A,B;
  halfadder H_1(out[0],c1,A[0],B[0]);
  fulladder F_1(out[1],c2,A[1],B[1],c1);
  fulladder F_2(out[2],c3,A[2],B[2],c2);
  fulladder F_3(out[3],C,A[3],B[3],c3);
endmodule


module halfadder(sum, carry ,A,B);
output  sum,carry;
input A,B;

 assign sum = A^B;
 assign carry = A&B;
endmodule

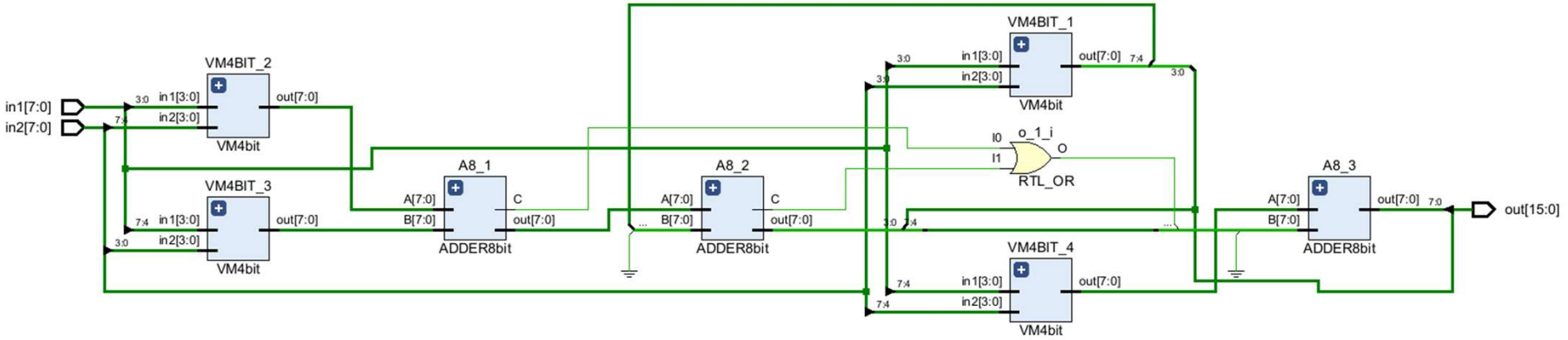
module fulladder(sum, carry ,A,B, cin );
output  sum,carry;
input A,B,cin;


  assign sum  = A^B^cin;
  assign carry = A&B|B&cin|A&cin;

endmodule

```
```

Schematic Diagram



Test Bench

```
13
14  module vedic8bit_tb();
15
16  wire [0:15]out;
17  reg [0:7]in1,in2;
18  vedic8bit Multiiplier4_bit(out,in1,in2);
19  initial
20  begin
21  in1=8'b00001110; in2=8'b00001110;
22  #5 in1=8'b10001000; in2=8'b11000000;
23  #5 in1=8'b10001110; in2=8'b11000011;
24  end
25  initial $monitor($time,"in1= %d,in2= %d,out= %d", in1, in2, out);
26  initial #100 $stop;
27  endmodule
```

Result

```
# run 1000ns
0in1=  14,in2=  14,out=   196
5in1= 136,in2= 192,out= 26112
10in1= 142,in2= 195,out= 27690
```

THANK YOU