DIGITAL ASSIGNMENT

Report submitted in partial completion of the course EEE/INSTR F313

ANALOG AND DIGITAL VLSI DESIGN

For

Dr. Anu Gupta

By

Vidhi Shah	2016A3PS0169P
Ardra Ayyappath	2016A3PS0261P
Aditya Sharma	2016A8PS0453P

Birla Institute of Technology and Science, Pilani – Pilani Campus

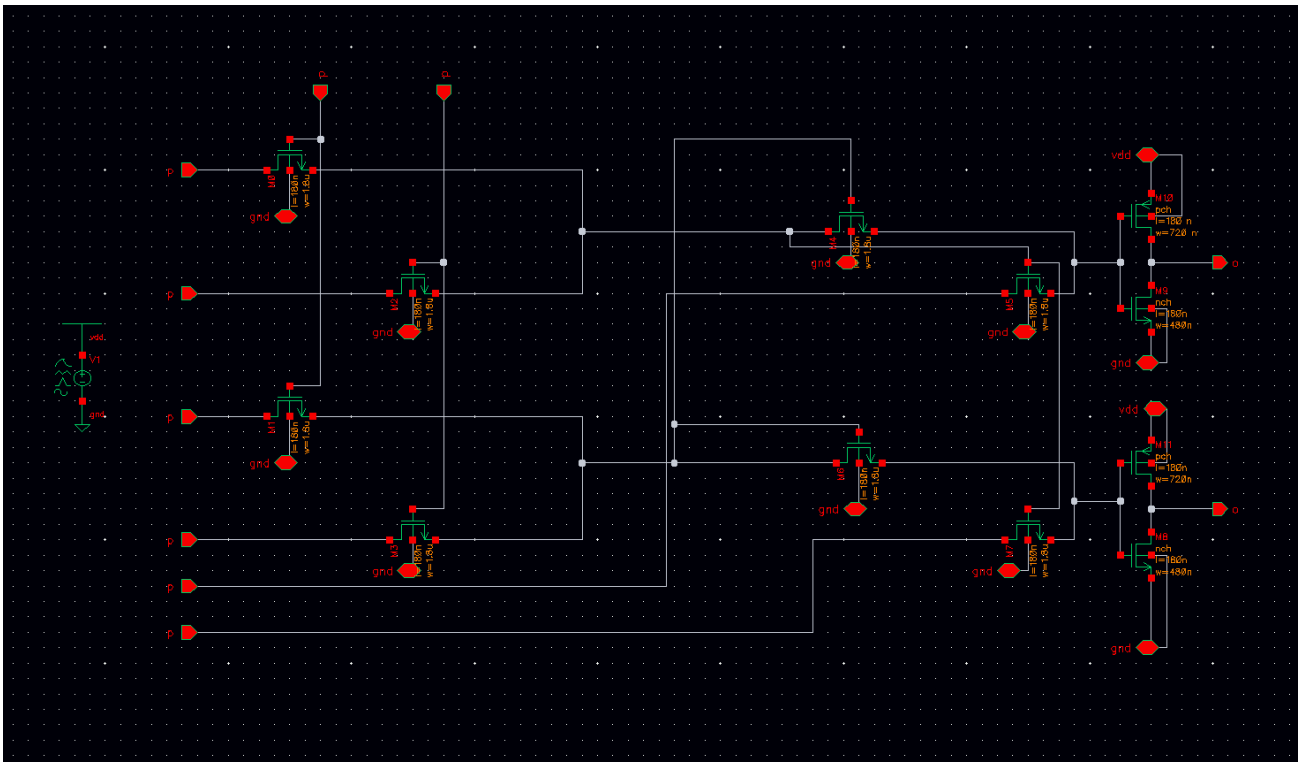
Problem 31

(a) OR/NOR USING CPL
Design of a 3-input OR/NOR logic gate using CPL logic style at 1 GHz with load capacitance of 500fF.

(b) Design an eight bit array multiplier using gate level modelling only using two input NAND gates.

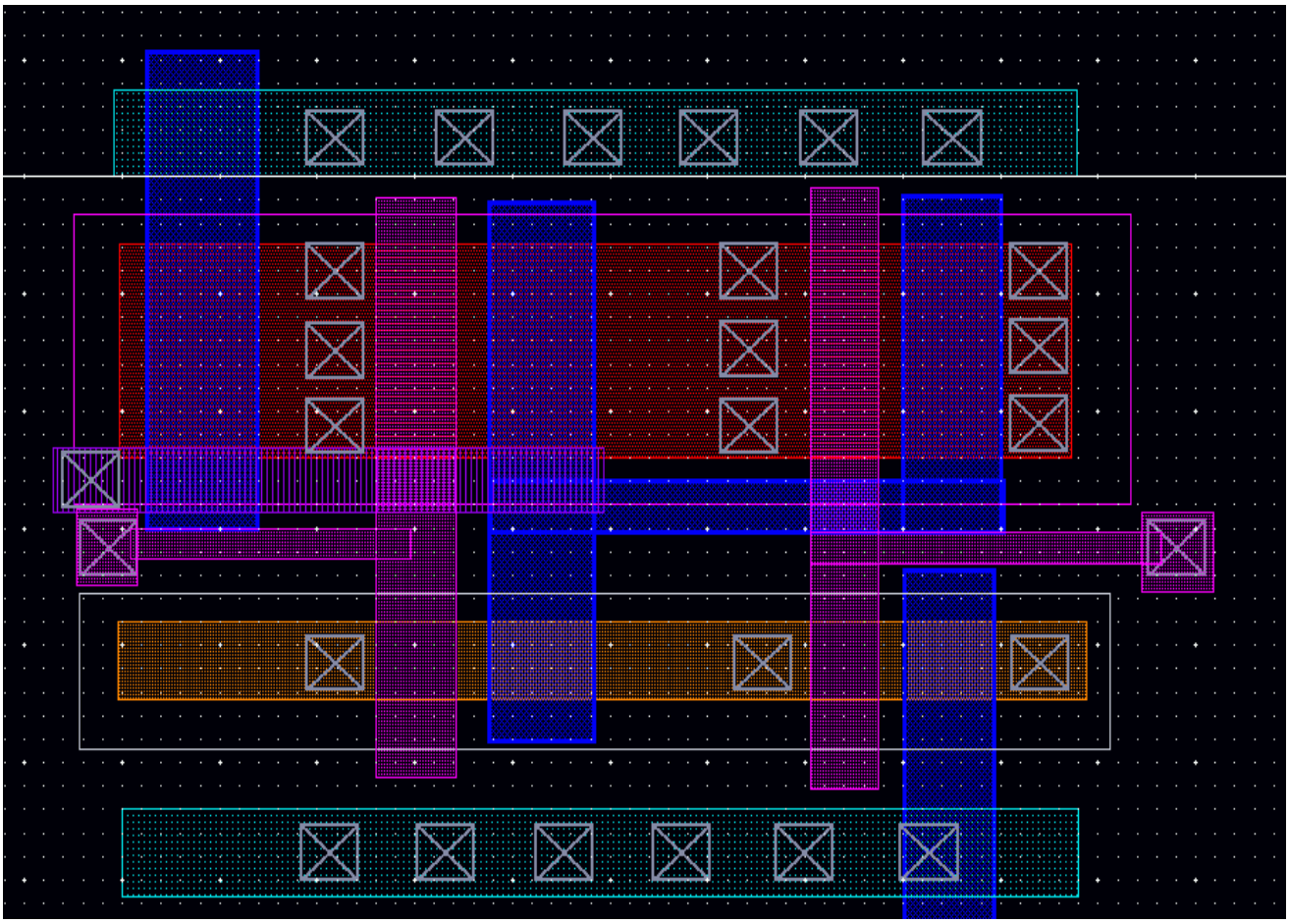
PART A

Schematic



Software: Cadence Virtuoso

Circuit Layout



Software: Cadence Virtuoso

Components:

1. 2 CMOS inverters
2. 8 NMOS
3. 500fF load capacitance

Power Calculation

Power = $\alpha f(V_{DD})^2 C_L$

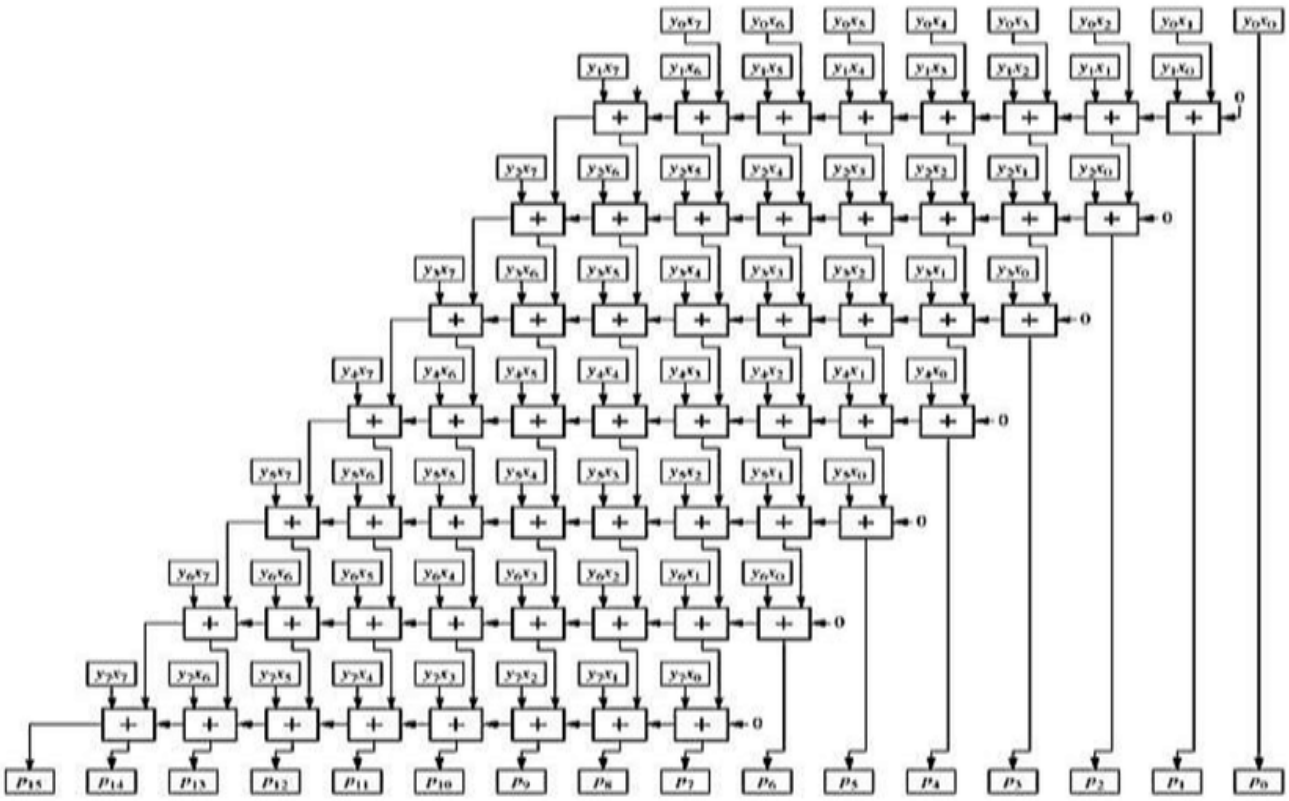
Theoretically, as per the question, C_L is given to be 500fF and frequency is taken as 1 GHz. We take V_{DD} as 1.8V.

CMOS parasitic capacitance at output = 34 fF

Therefore, the power consumption by above formula becomes **110.16 µW**.

PART B

Eight-bit Array Multiplier



Verilog Code

```
module array8x8(a,b,p);
//inputs
input [7:0]a,b;
//outputs
output [15:0]p;
//wires
wire [175:0]w;


//andgate instantiations
andf a1(w[0],a[0],b[0]);
andf a2(w[1],a[1],b[0]);
andf a3(w[2],a[2],b[0]);
```

```
andf a4(w[3],a[3],b[0]);
andf a5(w[4],a[4],b[0]);
andf a6(w[5],a[5],b[0]);
andf a7(w[6],a[6],b[0]);
andf a8(w[7],a[7],b[0]);

andf a9(w[8],a[0],b[1]);
andf a10(w[9],a[1],b[1]);
andf a11(w[10],a[2],b[1]);
andf a12(w[11],a[3],b[1]);
andf a13(w[12],a[4],b[1]);
andf a14(w[13],a[5],b[1]);
andf a15(w[14],a[6],b[1]);
andf a16(w[15],a[7],b[1]);

andf a17(w[16],a[0],b[2]);
andf a18(w[17],a[1],b[2]);
andf a19(w[18],a[2],b[2]);
andf a20(w[19],a[3],b[2]);
andf a21(w[20],a[4],b[2]);
andf a22(w[21],a[5],b[2]);
andf a23(w[22],a[6],b[2]);
andf a24(w[23],a[7],b[2]);

andf a25(w[24],a[0],b[3]);
andf a26(w[25],a[1],b[3]);
andf a27(w[26],a[2],b[3]);
andf a28(w[27],a[3],b[3]);
andf a29(w[28],a[4],b[3]);
andf a30(w[29],a[5],b[3]);
andf a31(w[30],a[6],b[3]);
andf a32(w[31],a[7],b[3]);

andf a33(w[32],a[0],b[4]);
andf a34(w[33],a[1],b[4]);
andf a35(w[34],a[2],b[4]);
andf a36(w[35],a[3],b[4]);
andf a37(w[36],a[4],b[4]);
andf a38(w[37],a[5],b[4]);
andf a39(w[38],a[6],b[4]);
andf a40(w[39],a[7],b[4]);
```

```
andf a41(w[40],a[0],b[5]);
andf a42(w[41],a[1],b[5]);
andf a43(w[42],a[2],b[5]);
andf a44(w[43],a[3],b[5]);
andf a45(w[44],a[4],b[5]);
andf a46(w[45],a[5],b[5]);
andf a47(w[46],a[6],b[5]);
andf a48(w[47],a[7],b[5]);


andf a49(w[48],a[0],b[6]);
andf a50(w[49],a[1],b[6]);
andf a51(w[50],a[2],b[6]);
andf a52(w[51],a[3],b[6]);
andf a53(w[52],a[4],b[6]);
andf a54(w[53],a[5],b[6]);
andf a55(w[54],a[6],b[6]);
andf a56(w[55],a[7],b[6]);


andf a57(w[56],a[0],b[7]);
andf a58(w[57],a[1],b[7]);
andf a59(w[58],a[2],b[7]);
andf a60(w[59],a[3],b[7]);
andf a61(w[60],a[4],b[7]);
andf a62(w[61],a[5],b[7]);
andf a63(w[62],a[6],b[7]);
andf a64(w[63],a[7],b[7]);


assign p[0]=w[0];
//full adders instatiations
fulladder a65(1'b0,w[1],w[8],w[64],w[65]);


fulladder a66(w[65],w[9],w[16],w[66],w[67]);
fulladder a67(1'b0,w[66],w[2],w[68],w[69]);


fulladder a68(w[67],w[24],w[17],w[70],w[71]);
fulladder a69(w[70],w[10],w[69],w[72],w[73]);
fulladder a70(1'b0,w[3],w[72],w[74],w[75]);


fulladder a71(w[71],w[32],w[25],w[76],w[77]);
fulladder a72(w[76],w[18],w[73],w[78],w[79]);
fulladder a73(w[11],w[78],w[75],w[80],w[81]);
```

```
fulladder a74(1'b0,w[80],w[4],w[82],w[83]);


fulladder a75(w[77],w[40],w[33],w[84],w[85]);
fulladder a76(w[84],w[26],w[79],w[86],w[87]);
fulladder a77(w[86],w[19],w[81],w[88],w[89]);
fulladder a78(w[12],w[88],w[83],w[90],w[91]);
fulladder a79(1'b0,w[90],w[5],w[92],w[93]);


fulladder a80(w[41],w[48],w[85],w[94],w[95]);
fulladder a81(w[94],w[34],w[87],w[96],w[97]);
fulladder a82(w[96],w[27],w[89],w[98],w[99]);
fulladder a83(w[98],w[20],w[91],w[100],w[101]);
fulladder a84(w[100],w[13],w[93],w[102],w[103]);
fulladder a85(1'b0,w[102],w[6],w[104],w[105]);


fulladder a86(w[56],w[49],w[95],w[106],w[107]);
fulladder a87(w[105],w[42],w[97],w[108],w[109]);
fulladder a88(w[107],w[35],w[99],w[110],w[111]);
fulladder a89(w[109],w[28],w[101],w[112],w[113]);
fulladder a90(w[111],w[21],w[103],w[114],w[115]);
fulladder a91(w[113],w[14],w[105],w[116],w[117]);
fulladder a92(1'b0,w[115],w[7],w[118],w[119]);


fulladder a93(1'b0,w[57],w[107],w[120],w[121]);
fulladder a94(w[120],w[50],w[109],w[122],w[123]);
fulladder a95(w[122],w[43],w[111],w[124],w[125]);
fulladder a96(w[124],w[36],w[113],w[126],w[127]);
fulladder a97(w[126],w[29],w[115],w[128],w[129]);
fulladder a98(w[128],w[22],w[117],w[130],w[131]);
fulladder a99(w[130],w[15],w[119],w[132],w[133]);


fulladder a100(w[121],w[58],w[123],w[134],w[135]);
fulladder a101(w[134],w[51],w[125],w[136],w[137]);
fulladder a102(w[136],w[44],w[127],w[138],w[139]);
fulladder a103(w[138],w[37],w[129],w[140],w[141]);
fulladder a104(w[140],w[30],w[131],w[142],w[143]);
fulladder a105(w[142],w[23],w[133],w[144],w[145]);


fulladder a106(w[59],w[135],w[137],w[146],w[147]);
fulladder a107(w[146],w[52],w[139],w[148],w[149]);
fulladder a108(w[148],w[45],w[141],w[150],w[151]);
```

```
fulladder a109(w[150],w[38],w[143],w[152],w[153]);
fulladder a110(w[152],w[31],w[145],w[154],w[155]);


fulladder a111(w[147],w[60],w[149],w[156],w[157]);
fulladder a112(w[156],w[53],w[151],w[158],w[159]);
fulladder a113(w[158],w[46],w[153],w[160],w[161]);
fulladder a114(w[160],w[39],w[155],w[162],w[163]);


fulladder a115(w[157],w[61],w[159],w[164],w[165]);
fulladder a116(w[164],w[54],w[161],w[166],w[167]);
fulladder a117(w[166],w[47],w[163],w[168],w[169]);


fulladder a118(w[165],w[62],w[167],w[170],w[171]);
fulladder a119(w[170],w[55],w[169],w[172],w[173]);

fulladder a120(w[171],w[63],w[173],w[174],w[175]);


//output assignments
assign p[1]=w[64];
assign p[2]=w[68];
assign p[3]=w[74];
assign p[4]=w[82];
assign p[5]=w[92];
assign p[6]=w[104];
assign p[7]=w[118];
assign p[8]=w[132];
assign p[9]=w[144];
assign p[10]=w[154];
assign p[11]=w[162];
assign p[12]=w[168];
assign p[13]=w[172];
assign p[14]=w[174];
assign p[15]=w[175];

endmodule


//FULL ADDER
module fulladder(a,b,c,s,ca);
wire w1,w2,w3,w4,w5,w6,w7;
nand n1(w1,a,b);
nand n2(w3,a,w1);
nand n3(w2,w1,b);
```

```
nand n4(w4,w3,w2);
nand n5(w5,w4,c);
nand n6(w6,w4,w5);
nand n7(w7,w5,c);
nand n8(ca,w5,w1);
nand n9(s,w6,w7);
endmodule

//AND FROM NAND
module andf(f,x,y);
reg x,y;
wire f,w;
nand m1(w,x,y);
nand m2(f,w,w);
endmodule
```

Testbench Module

```
module tb_multiplier;
reg [7:0]a;
reg [7:0]b;
wire [15:0]p;
array8x8 ex(.a(a),.b(b),.p(p));


initial begin
a = 8'b00010010;
b = 8'b00000010;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b01000000;
b=8'b00000010;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b11001000;
b=8'b01011010;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b11110000;
```

```
b=8'b00000001;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b10001010;
b=8'b01110110;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b00110011;
b=8'b00000011;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b10000001;
b=8'b10000000;
$display("Input is %b and %b, Output is %b",a,b,p);

#10
a=8'b00001111;
b=8'b00011000;
$display("Input is %b and %b, Output is %b",a,b,p);

$stop;
end
endmodule
```

```
Input is 00010010 and 00000010, Output is 0000000000100100
Input is 01000000 and 00000010, Output is 0000000010000000
Input is 11001000 and 01011010, Output is 110011001010000
Input is 11110000 and 00000001, Output is 0000000011110000
Input is 10001010 and 01110110, Output is 0011111110011100
Input is 00110011 and 00000011, Output is 00000000010011001
Input is 10000001 and 10000000, Output is 1000000100000000
Input is 00001111 and 00011000, Output is 0000000101101000
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 80 ticks.
>
```