A report on

Design and Synthesis of 8 Bit Array Multiplier Using 2x1 Multiplexers via Gate Level Modelling in Verilog

By

Akshit Goel **2016A3PS0317P**
Saksham Consul **2016A8PS0424P**
Nishad Sahu **2016A3PS0215P**

Submitted in partial fulfillment of the course

EEE F313/ INSTRF313 – Analog & Digital VLSI Design

Under the Guidance of

Prof. Anu Gupta
Instructor In-charge *EEE F313/ INSTR F313*
EEE Department, BITS Pilani, Pilani Campus

Birla Institute of Technology and Science, Pilani

August- September 2018

Acknowledgement

We would like to thank Dr. Anu Gupta, Dr Nitin Chaturvedi and the entire ADVD team for their constant guidance and support in the project work.

We are also thankful to the TAs; Siddhant Gangwal, Raveesh Garg and Mohit Garg for their help with software related issues and understanding of the various protocols.

Contents

1a. Problem Statement:

Design an eight bit array multiplier using gate level modeling only using 2X1 multiplexers.

1b. Problem Approach:

We first designed a simple 2x1 multiplexer using AND, NOT and OR gates. Using the mux, we created a half adder and a full adder. We then created a 8 bit AND gate and a 8 bit adder. We then cascaded the 8 bit adders and AND gates to creat the 8 bit multiplier.

2. Code:

```
//Model a 2x1 Mux using gate level modelling
module mux (X,A,B,S);
input A, B,S;
output X;
wire sno, C, D;
and g1(C, S, A);
not g2(sno, S);
and g3(D, sno, B);
or g4(X, C, D);
endmodule

//Model a Half Adder using 2x1 Mux
module HA (S,Cout,A,B);
input A, B;
output S,Cout;
wire B_not;
mux m1 (.X(Cout), .A(B), .B(1'b0), .S(A));
mux m2 (.X(B_not), .A(1'b0), .B(1'b1), .S(B));
mux m3 (.X(S), .A(B_not), .B(B), .S(A));
```

```
endmodule

// Creating an OR gate from 2x1 MUX
module mux_or(X,A,B);
output X;
input A, B;
mux mor (.X(X), .A(1'b1), .B(B), .S(A));
endmodule

//Model a Full Adder using 2x1 Mux
module FA (S,Cout,A,B,Cin);
input A, B, Cin;
output S, Cout;
wire x1,x2,x3;
HA ha1(.S(x1), .Cout(x2), .A(A), .B(B));
HA ha2(.S(S), .Cout(x3), .A(x1), .B(Cin));
mux_or mor1(.X(Cout), .A(x3), .B(x2));
endmodule

//8 Bit adder
```

```
module add (S,Cout,A,C,B);
input [7:0]A;
input [6:0] B;
input C;
output [7:0]S;
output Cout;
wire c1, c2, c3, c4, c5, c6, c7;
HA h1(S[0], c1, A[0], B[0]);
FA f1(S[1], c2, A[1], B[1], c1);
FA f2(S[2], c3, A[2], B[2], c2);
FA f3(S[3], c4, A[3], B[3], c3);
FA f4(S[4], c5, A[4], B[4], c4);
FA f5(S[5], c6, A[5], B[5], c5);
FA f6(S[6], c7, A[6], B[6], c6);
FA f7(S[7], Cout, A[7], C, c7);
endmodule

// Creating an AND gate from 2x1 MUX
module mux_and(X,A,B);
output X;
```

```
input A, B;
mux mand (.X(X), .A(B), .B(1'b0), .S(A));
endmodule

// 1x8 AND
module AND18 (X,B,A);
output [7:0]X;
input [7:0] B;
input A;
mux_and ma0 (X[0], A,  B[0]);
mux_and ma1 (X[1], A,  B[1]);
mux_and ma2  (X[2], A,  B[2]);
mux_and ma3  (X[3], A,  B[3]);
mux_and ma4  (X[4], A,  B[4]);
mux_and ma5 (X[5], A,  B[5]);
mux_and ma6  (X[6], A,  B[6]);
mux_and ma7  (X[7], A,  B[7]);
endmodule

// 8 Bit Array Multiplication
```

```
module multiply(output[15:0]M , input[7:0] A,B);
wire [7:0]y0,y1,y2,y3,y4,y5,y6,y7,
        s0,s1,s2,s3,s4,s5,s6;
wire c0,c1,c2,c3,c4,c5,c6;
AND18 an0(y0,B,A[0]);
AND18 an1(y1,B,A[1]);
AND18 an2(y2,B,A[2]);
AND18 an3(y3,B,A[3]);
AND18 an4(y4,B,A[4]);
AND18 an5(y5,B,A[5]);
AND18 an6(y6,B,A[6]);
AND18 an7(y7,B,A[7]);

add ad1(s0,c0,y1,1'b0,y0[7:1]);
add ad2(s1,c1,y2,c0,s0[7:1]);
add ad3(s2,c2,y3,c1,s1[7:1]);
add ad4(s3,c3,y4,c2,s2[7:1]);
add ad5(s4,c4,y5,c3,s3[7:1]);
add ad6(s5,c5,y6,c4,s4[7:1]);
add ad7(s6,c6,y7,c5,s5[7:1]);
```

```
mux_and ma8  (M[0], 1'b1,  y0[0]);
mux_and ma9  (M[1], 1'b1,  s0[0]);
mux_and ma10 (M[2], 1'b1,  s1[0]);
mux_and ma11 (M[3], 1'b1,  s2[0]);
mux_and ma12 (M[4], 1'b1,  s3[0]);
mux_and ma13 (M[5], 1'b1,  s4[0]);
mux_and ma14 (M[6], 1'b1,  s5[0]);
mux_and ma15 (M[7], 1'b1,  s6[0]);
mux_and ma16 (M[8], 1'b1,  s6[1]);
mux_and ma17 (M[9], 1'b1,  s6[2]);
mux_and ma18 (M[10], 1'b1,  s6[3]);
mux_and ma19 (M[11], 1'b1,  s6[4]);
mux_and ma20 (M[12], 1'b1,  s6[5]);
mux_and ma21 (M[13], 1'b1,  s6[6]);
mux_and ma22 (M[14], 1'b1,  s6[7]);
mux_and ma23 (M[15], 1'b1,  c6);
endmodule

//Testbench for Multiplier
```

```
module mult_tb();
reg [7:0]A,B;
reg [15:0]M;
multiply(.M(M), .A(A), .B(B));
initial begin A=0; B=0;
A = 8'b00000000;
B = 8'b00000000;
#10
A = 8'b11111111;
B = 8'b11111111;
#10
A = 8'b00001111;
B = 8'b00001111;
#10
A = 8'b00001111;
B = 8'b11110000;
$finish;
end
endmodule
```

3. Simulation Results:

Case 1:

A = 0000 0000

B = 1111 1111

M = 0000 0000 0000 0000

Case 2:

A = 1111 1111

B = 1111 1111

M = 1111 1110 0000 0001

Case 3:

A = 1111 0000

B = 0000 1111

M = 0000 1110 0001 0000

Case 4:

A = 1111 0000

B = 1100 1100

M = 1011 1111 0100 0000

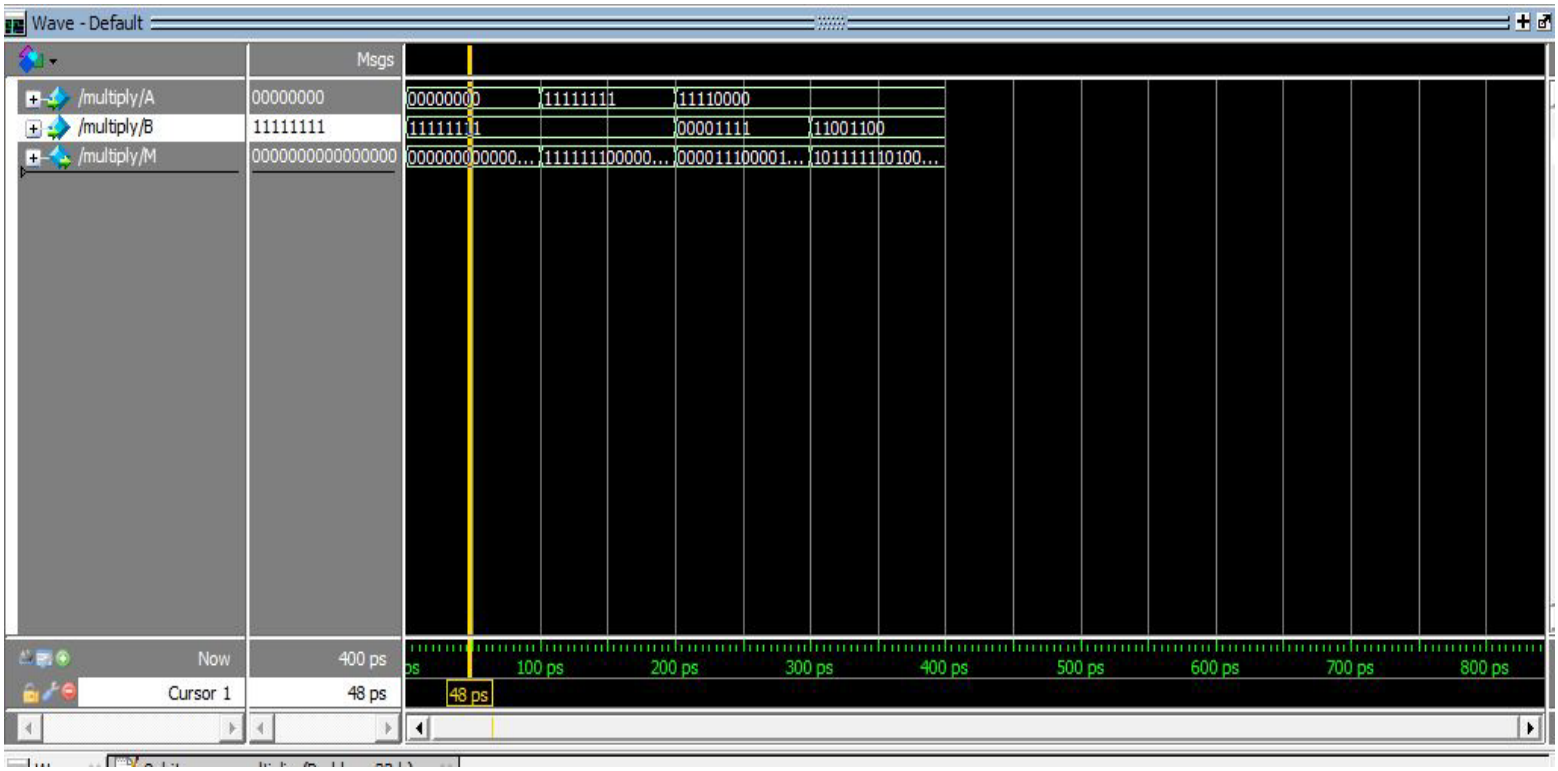


Figure 1: Simulation Results

4. RTL Synthesis

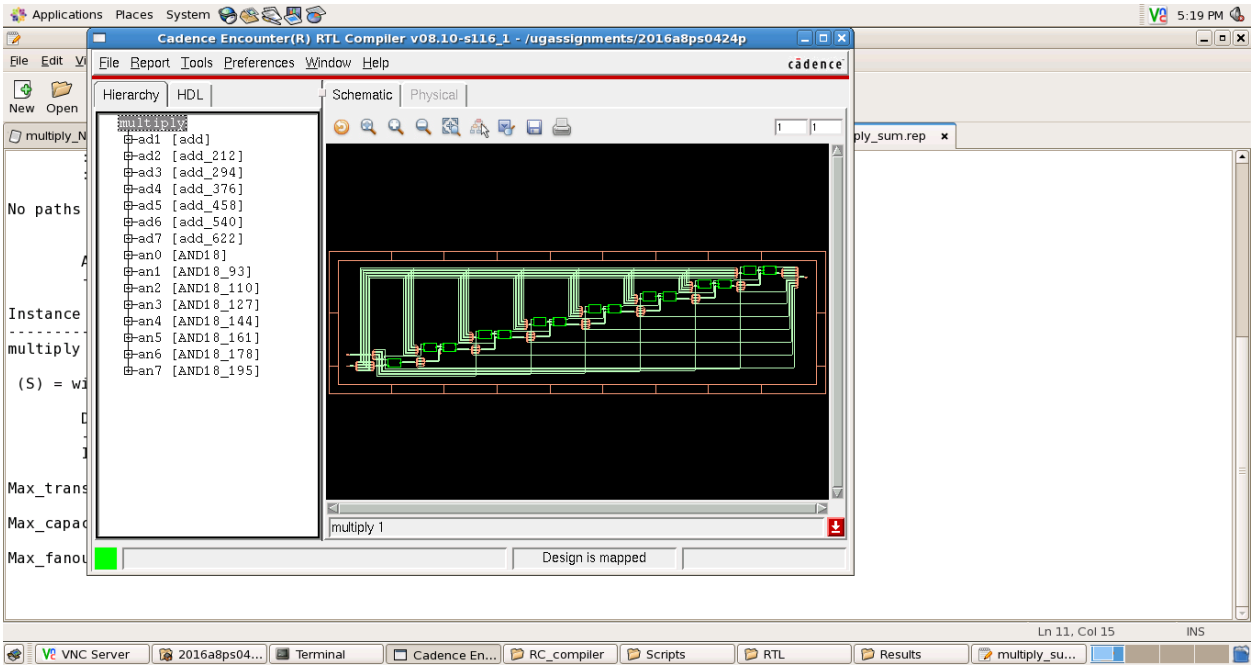


Figure 2: 8 bit mulitplier

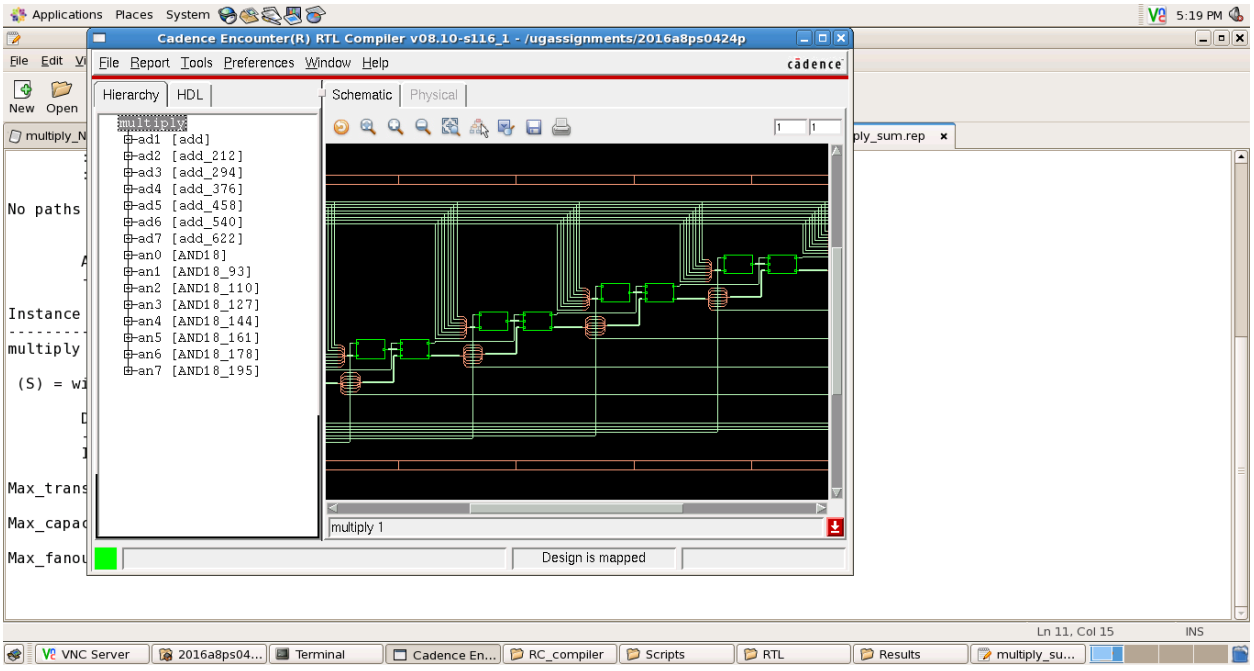


Figure 3: 8 bit multiplier magnified

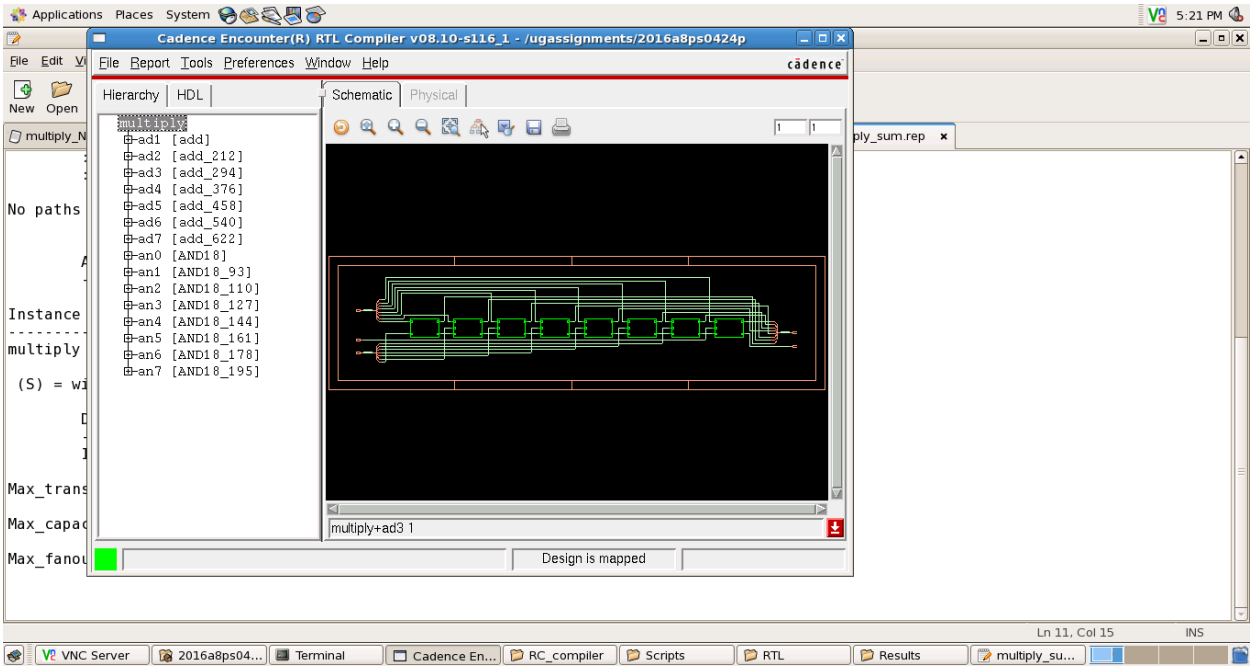


Figure 4: 8 bit adder

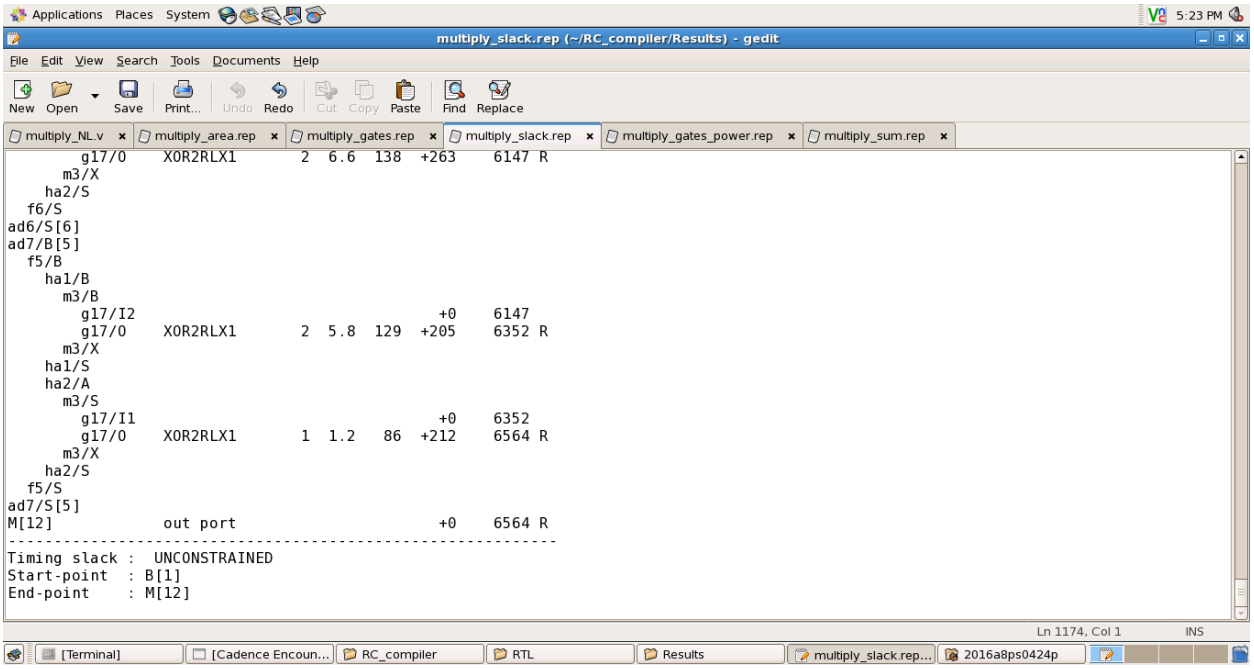


Figure 5: Slack Report

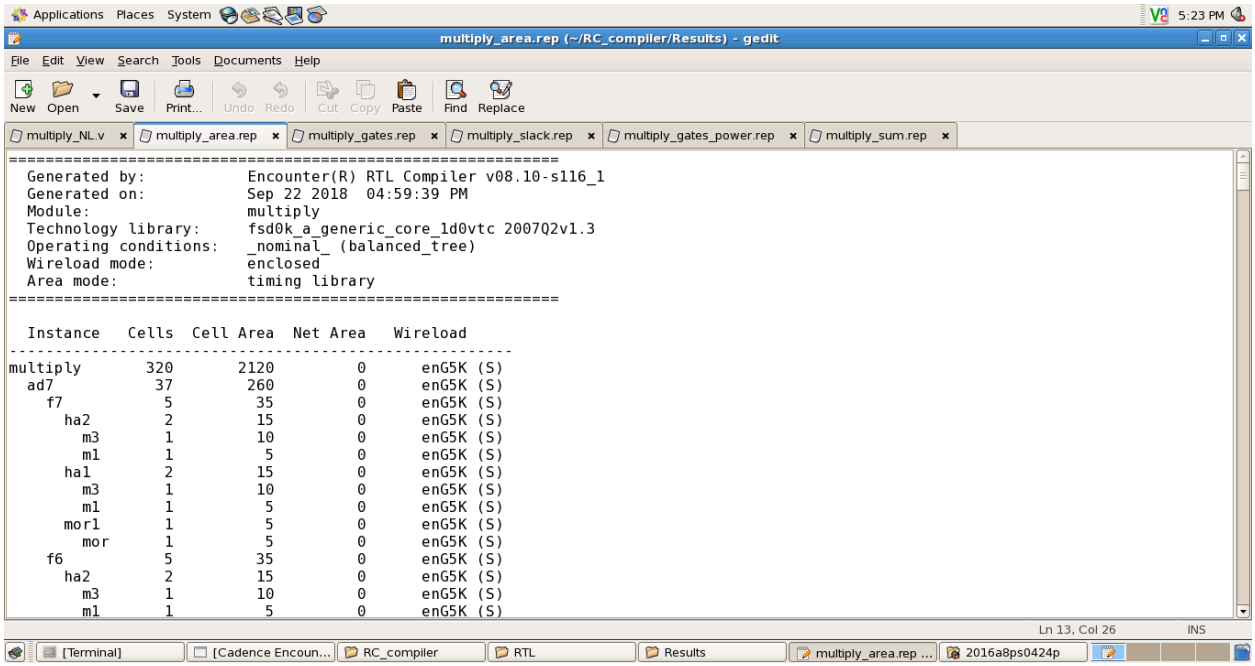


Figure 6: Area Report

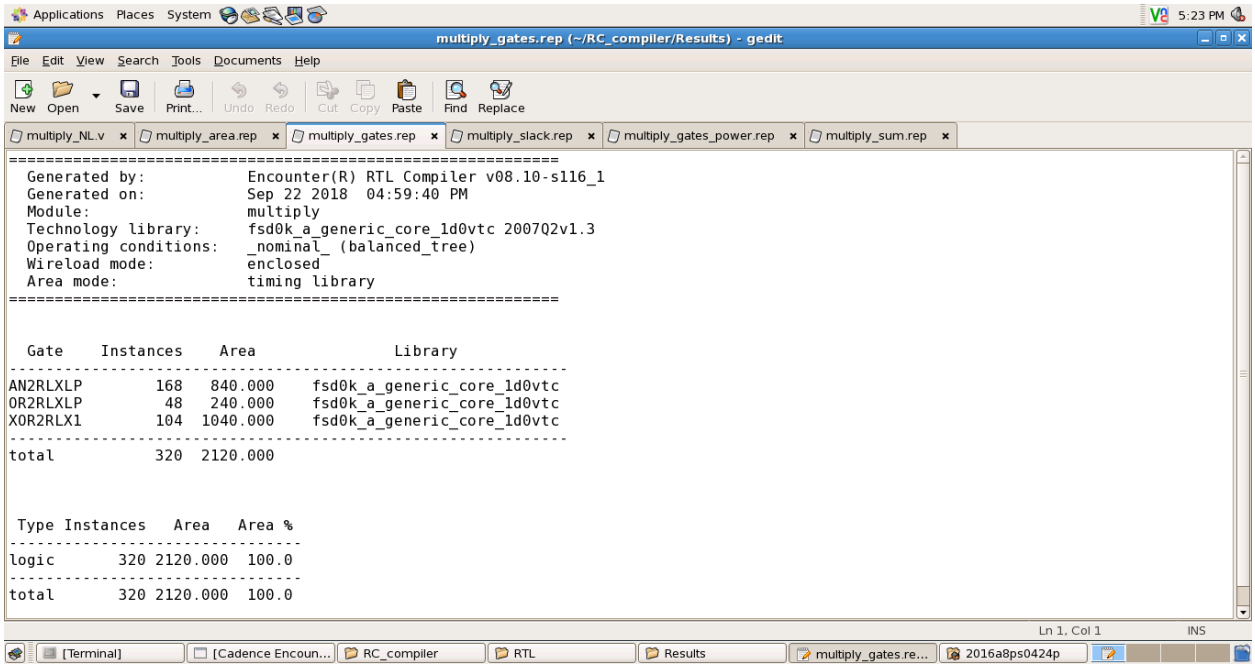
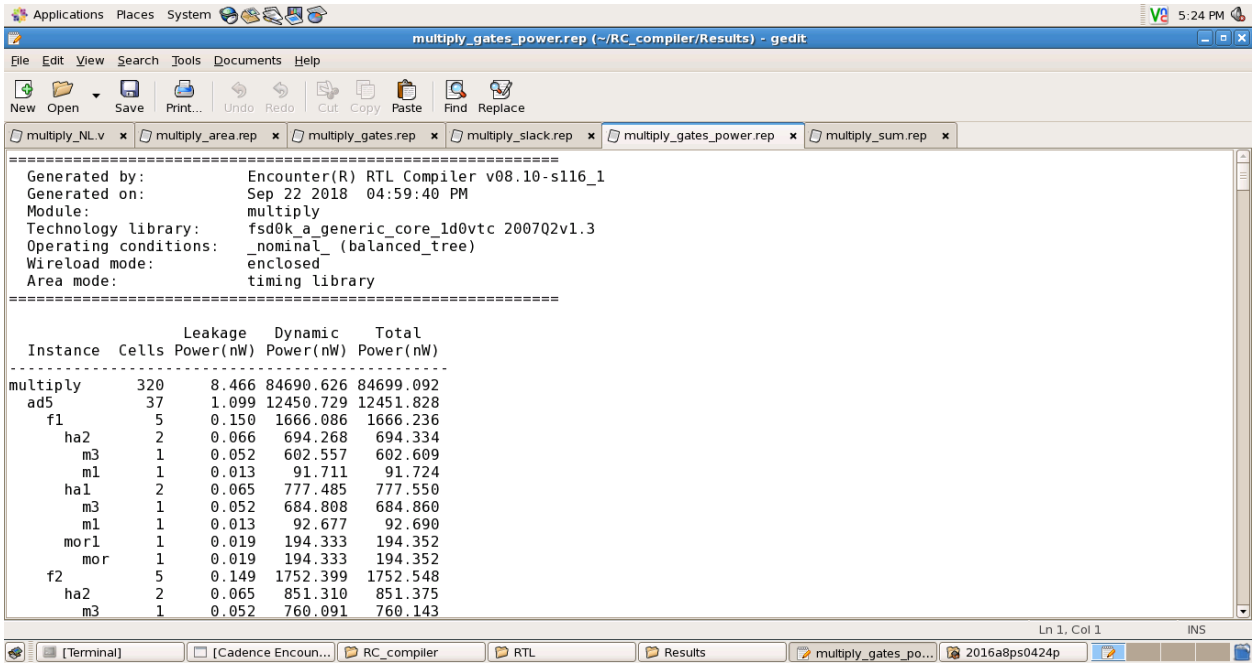


Figure 7: Gate Report




```
============================================================
  Generated by:           Encounter(R) RTL Compiler v08.10-s116_1
  Generated on:           Sep 22 2018  04:59:40 PM
  Module:                 multiply
  Technology library:     fsd0k_a_generic_core_1d0vtc 2007Q2v1.3
  Operating conditions:   _nominal_ (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                  Leakage   Dynamic    Total
  Instance  Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------------
multiply      320     8.466 84690.626 84699.092
  ad5          37     1.099 12450.729 12451.828
    f1          5     0.150  1666.086  1666.236
      ha2       2     0.066   694.268   694.334
        m3      1     0.052   602.557   602.609
        m1      1     0.013    91.711    91.724
      ha1       2     0.065   777.485   777.550
        m3      1     0.052   684.808   684.860
        m1      1     0.013    92.677    92.690
      mor1      1     0.019   194.333   194.352
        mor     1     0.019   194.333   194.352
    f2          5     0.149  1752.399  1752.548
      ha2       2     0.065   851.310   851.375
        m3      1     0.052   760.091   760.143
```

Figure 8: Power Report