f

Advanced VLSI Design

EE 7325

Comparison of a 32-bit Behavioral Multiplier and Booth-2 Algorithm Multiplier

Mark Ripley Sears (mrs171030)

Navya Sri Sreeram (nxs161131)

Pin Pitch	0.26um
Design #1: Behavioral Multiplier	7,658 Cells
Design #2: Booth-2 Algorithm	23,598 Cells

08/06/2018

32-bit Multiplier Design

In this project, two multiplier designs were synthesized and the tradeoffs are compared.

1) **Design #1: Behavioral Multiplier (X*Y)**

   A default synthesis tool inferred 32x32 bit multiplier design was synthesized and simulated.

2) **Design #2: Booth-2 Algorithm Multiplier with CLA adder**

   A 32-bit multiplier design that uses the Booth-2 algorithm to reduce the number of partial products was synthesized and simulated. The verilog implementation consists of three stages:

   1. Compute 16 partial products (using Booth-2 reduces partial products by half)
   2. Compression of partial products into two 64-bit rows.
   3. Final fast 64-bit carry look-ahead adder to compute the final product.

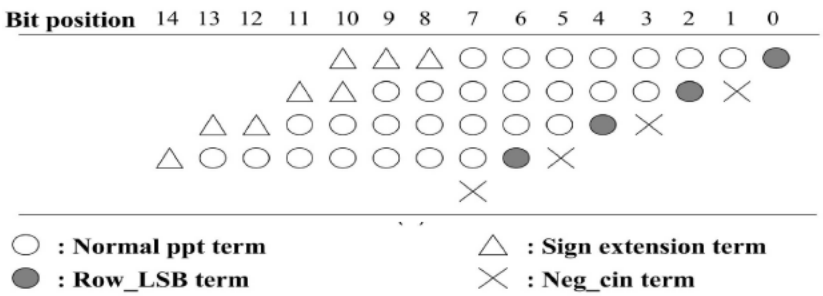
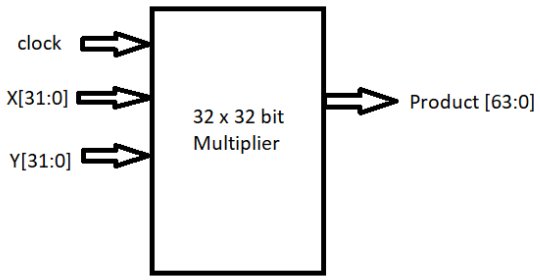


Figure 1: Partial Products for 8-bit Multiplier using booth algorithm

Multiplier Block Diagram

Both the designs have 65 input pins and 64 output pins.



Verilog Code Description

The full Verilog code used for both designs can be found in the Appendix. For Design #1, the verilog is simple and entirely behavioral, letting the synthesizer do the hard work of multiplication. Design #2 uses a hierarchical approach to specify and generate the logic. Modules are written to generate the partial products, compressors, and CLA adder. The top level multiplier module instantiates many copies of these modules. A parameter "WIDTH" was used to specify the number of bits in the multiplier, and generate-for loops were used to instantiate many modules.

Partial Products:

A 32-bit Booth-2 multiplier will have 16 partial products (PP). To compute each PP a module "booth2PP_32b" was created. It uses the multiplicand and 3 of the multiplier bits to correctly compute a single partial product. Signals "neg", "single", and "double" are determined by the 3 multiplier bits based on the Booth-2 method. Then, the partial product is modified based on these signals:

- If neither the "single" or "double" is true, then the PP is all 0's.
- If the "double" signal is true, then the PP is shifted left.
- If the "neg" signal is true, then the PP is inverted and a 1 is added.

Each partial product is 33 bits. The extra bit accounts for shifting left. Since there are 16 partial products for a 32-bit multiplier, our design instantiates 16 of the Booth-2 modules. Using Booth-2 is beneficial because it halves the number of PPs and reduces the size of the compressors.

Compressors:

A "compressor" module was created to compress each PP column into only 2 rows of bits. This is simply a large tree of full adders. For our design, we used a 17 bit compressor for each column of PPs. The 17th bit is required for certain cases of the multiplier. For n = 17, each module contains 15 full adders, and has 14 pass-carries in addition to the outputs "Sum" and "C_{drop}". Since the product will have 64 bits, our design instantiates 64 of these compressors. The sign extend bits are a function of the MSB of a PP and are added as inputs to certain compressors. The extra "add 1" for computing the 2's complement of a PP is also added as an input to certain compressors. A carefully crafted generate-for loop is used to instantiate all the compressors.

Admittedly, using 17-bit compressors for all columns is inefficient because many columns will always have 0's as their inputs, so some of the full adders will always be adding 0. This is actually a trade off between design complexity and area/power consumption. To get the minimum area, more effort would need to be put into organizing these compressors and avoid adding all the 0's.

Carry-Lookahead Adder

The final part of the multiplier is the fast adder. A carry-lookahead tree module is implemented to precompute all the carries to speed this up. For a 64-bit adder, the lookahead trees get quite large and are only tractable in verilog through careful use of generate loops. In an attempt to minimize overall delay at the expense of area and power, 16 lookahead trees were used. The leftmost bit of each tree being the 16 MSBs. For 64-bits, each tree consisted of 6 layers ($2^6 = 64$).

Each lookahead tree consists of nodes which represent group "Generates" and "Propagates". A verilog module was created to instantiate each tree, which computes these Generates and Propagates.

Logical Formulas for computing Generates and Propagates

	Generate	**Propagate**
Single bit	$G^0_0 = A \& B$	$P^0_0 = A + B$
Group	$G^1_0 = G^0_1 + (P^0_1 \& G^0_0)$	$P^1_0 = P^0_1 \& P^0_0$

The verilog module for the carry-lookahead adder instantiated 16 trees and uses them to compute each carry without the need for a ripple carry. For example, the 16 MSB carries can be found using the top level group generate of each tree:

$$\text{Carry \#63} = G^6_0$$

(since there is no carry-in for this group, the propagate term is not needed).

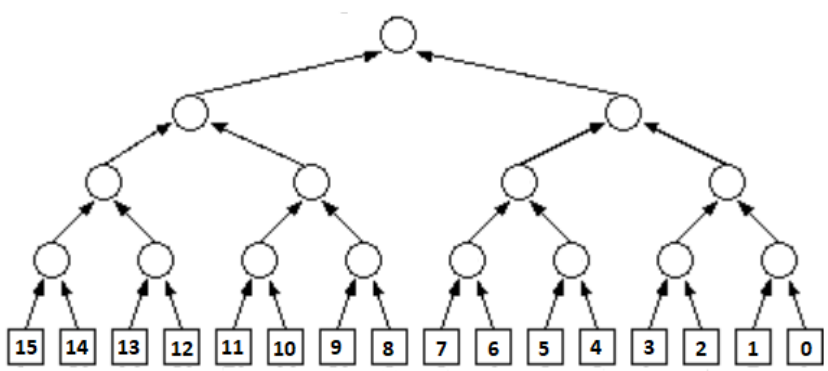


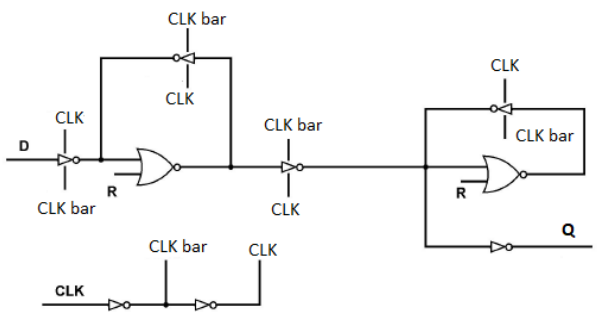
Figure 2: Example of 16-bit Carry Lookahead tree

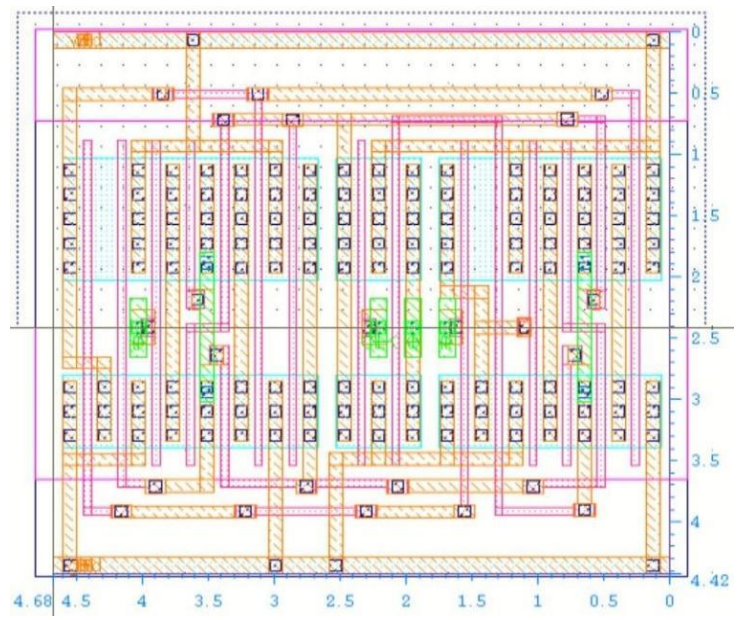
The precomputed carries feed into 64 full adders. The output of this 64-bit adder is the final product, which gets latched into a register and completes the multiplier.

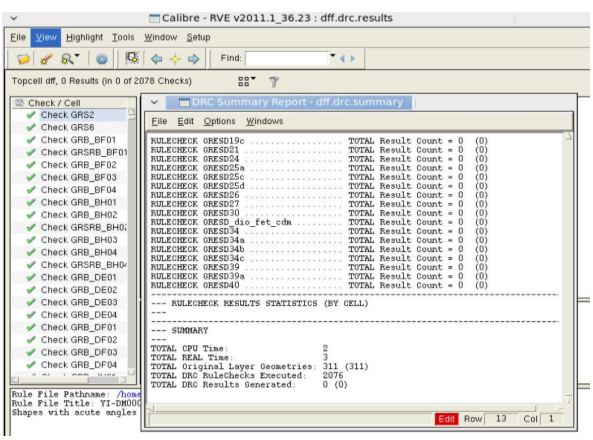
Cell Library : Layout, DRC and LVS Reports

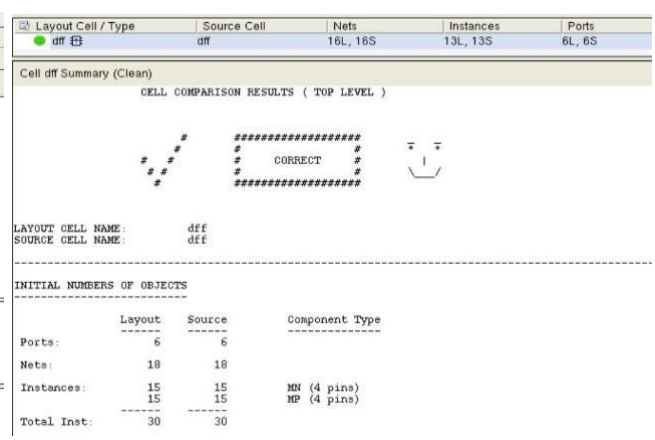
The standard cell library was created using the 65nm process. Each of our cells have a pin pitch of 0.26um and a height of 4.42um. Schematic and layout views were generated for each cell, and DRC, LVS, and PEX were run.

DFF Design (Positive edged FF with Reset Pin)









DFF Measured Timing characteristics

	Transition to 0	Transition to 1
T_{su_dd}	20ps	56ps
T_{hold}	-55ps	-19ps
$T_{clk \rightarrow Q}$	251ps	276ps

Figure 3: D-Flip Flop Measurement for Tsu_dd(0)

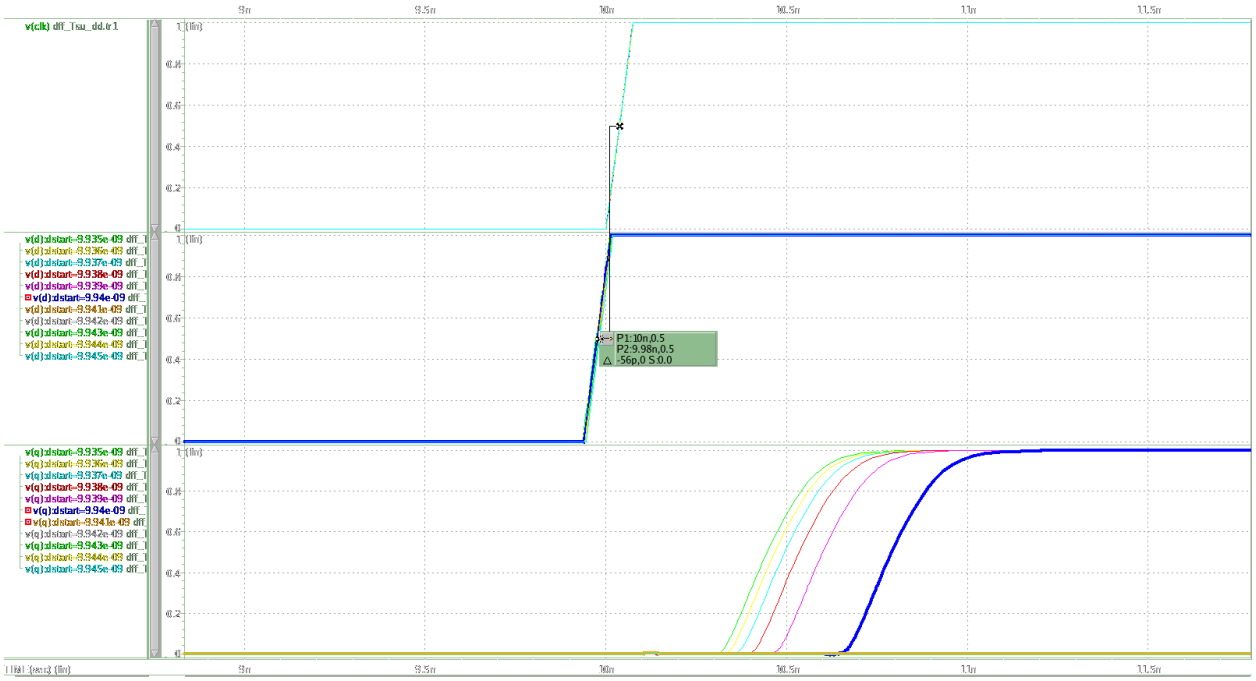
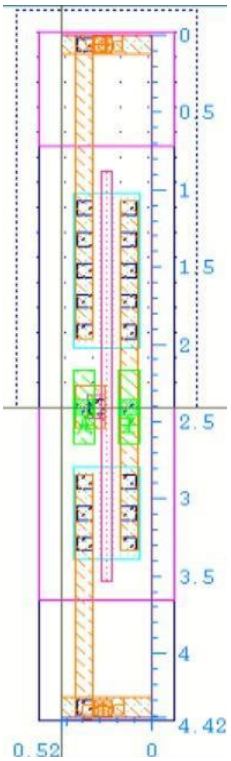
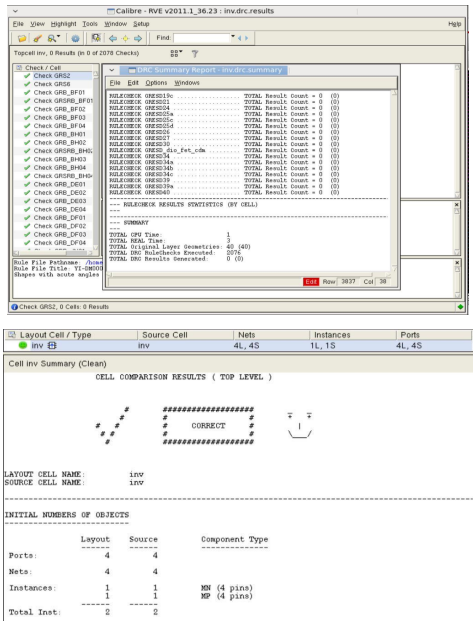


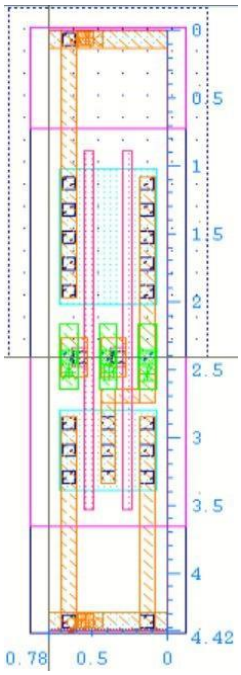
Figure 4: D-Flip Flop Measurement for Tsu_dd(1)

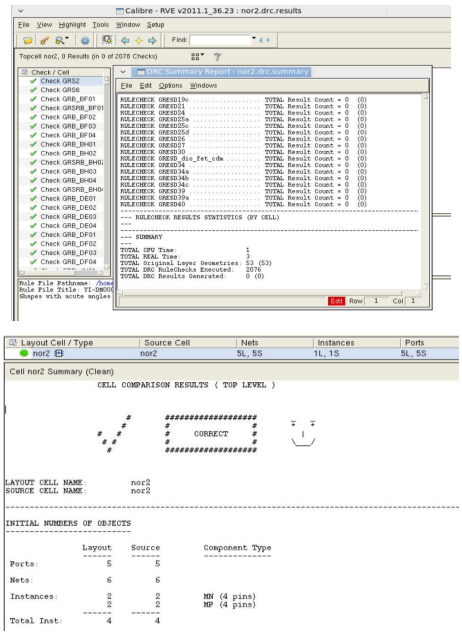
inverter



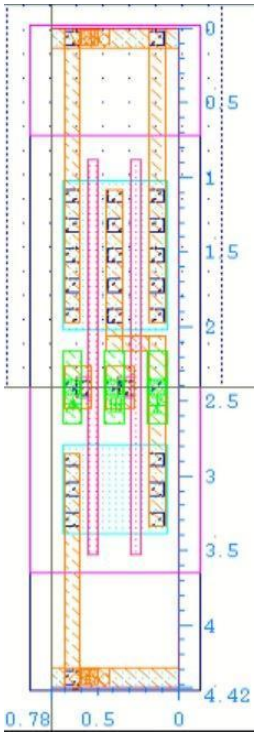


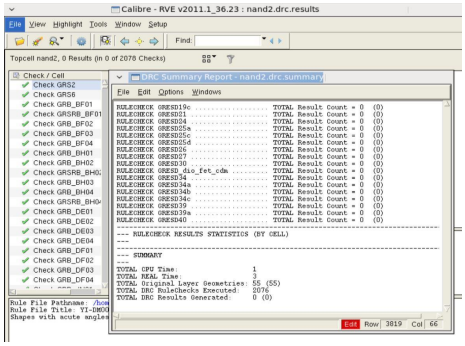
nor2

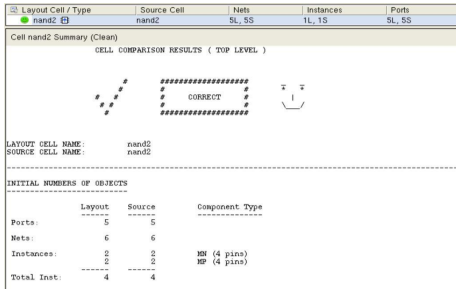




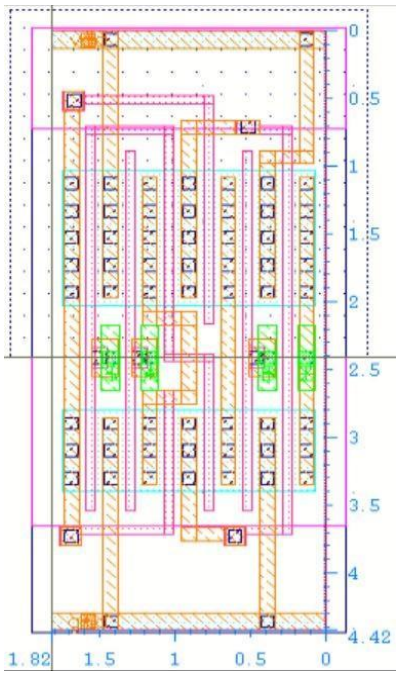
nand2

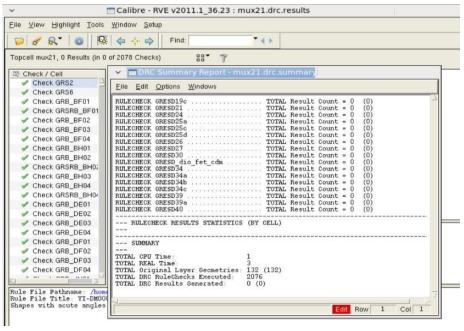

0
0.5
1
1.5
2
2.5
3
3.5
4
4.42
0.78
0.5
0

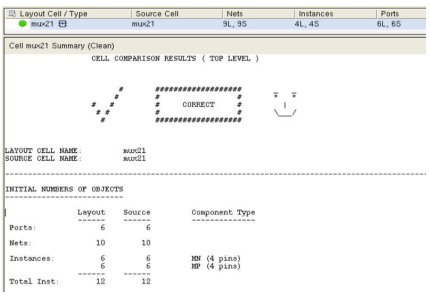

Calibre - RVE v2011.1_36.23 : nand2.drc.results
DRC Summary Report - nand2.drc.summary
TOTAL CPU Time: 1
TOTAL REAL Time: 3
TOTAL Original Layer Geometries: 55 (55)
TOTAL DRC RuleChecks Executed: 2076
TOTAL DRC Results Generated: 0 (0)


Layout Cell / Type
Source Cell
Nets
Instances
Ports
nand2
nand2
5L, 5S
1L, 1S
5L, 5S
Cell nand2 Summary (Clean)
CELL COMPARISON RESULTS (TOP LEVEL)
CORRECT
LAYOUT CELL NAME: nand2
SOURCE CELL NAME: nand2
INITIAL NUMBERS OF OBJECTS
Layout
Source
Component Type
Ports: 5 5
Nets: 6 6
Instances: 2 2 MN (4 pins)
2 2 MP (4 pins)
Total Inst: 4 4

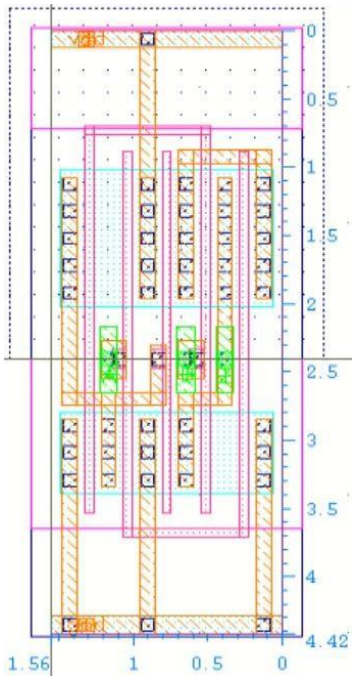
mux21

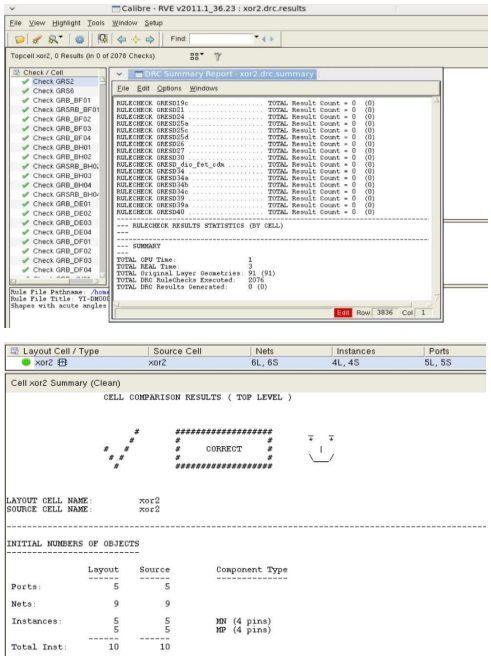

0
0.5
1
1.5
2
2.5
3
3.5
4
4.42
1.82
1.5
1
0.5
0


Calibre - RVE v2011.1_36.23 : mux21.drc.results
DRC Summary Report - mux21.drc.summary
TOTAL CPU Time: 1
TOTAL REAL Time: 3
TOTAL Original Layer Geometries: 132 (132)
TOTAL DRC RuleChecks Executed: 2076
TOTAL DRC Results Generated: 0 (0)

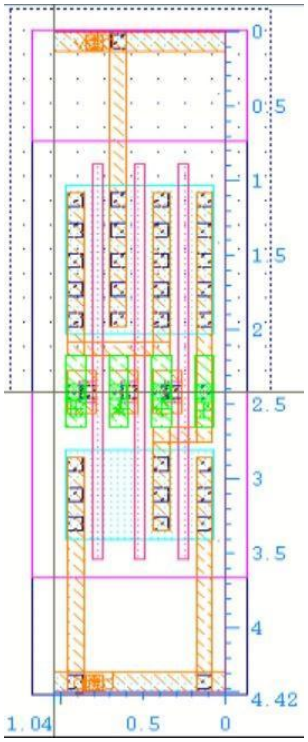

Layout Cell / Type
Source Cell
Nets
Instances
Ports
mux21
mux21
9L, 9S
4L, 4S
6L, 6S
Cell mux21 Summary (Clean)
CELL COMPARISON RESULTS (TOP LEVEL)
CORRECT
LAYOUT CELL NAME: mux21
SOURCE CELL NAME: mux21
INITIAL NUMBERS OF OBJECTS
Layout
Source
Component Type
Ports: 6 6
Nets: 10 10
Instances: 6 6 MN (4 pins)
6 6 MP (4 pins)
Total Inst: 12 12

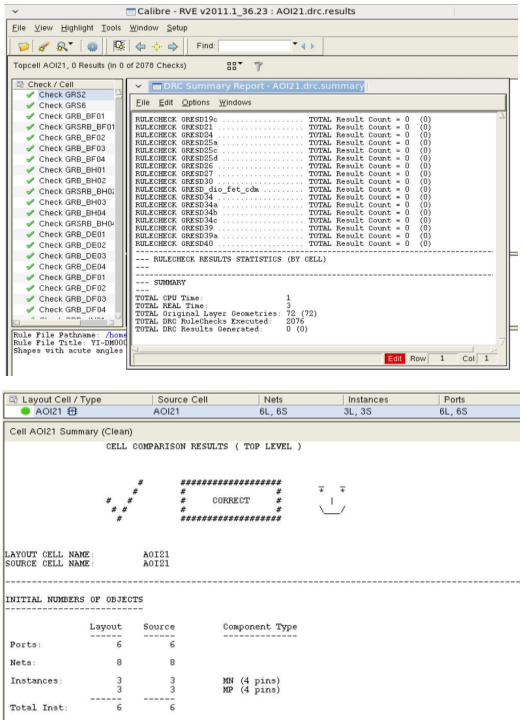
Xor


0
0.5
1
1.5
2
2.5
3
3.5
4
4.42
1.56
1
0.5
0

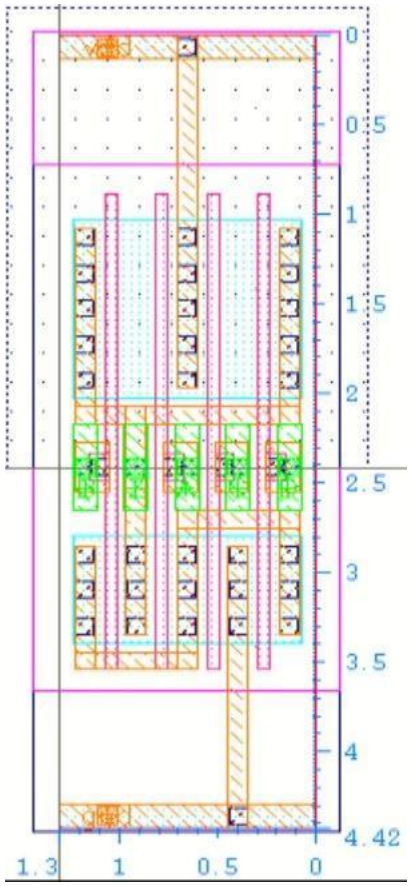

Calibre - RVE v2011.1_36.23 : xor2.drc.results
DRC Summary Report - xor2.drc.summary
Layout Cell / Type | Source Cell | Nets | Instances | Ports
xor2 | xor2 | 6L, 6S | 4L, 4S | 5L, 5S
Cell xor2 Summary (Clean)
CELL COMPARISON RESULTS (TOP LEVEL)
CORRECT
LAYOUT CELL NAME: xor2
SOURCE CELL NAME: xor2
INITIAL NUMBERS OF OBJECTS
Layout Source Component Type
Ports: 5 5
Nets: 9 9
Instances: 5 5 MN (4 pins)
5 5 MP (4 pins)
Total Inst: 10 10

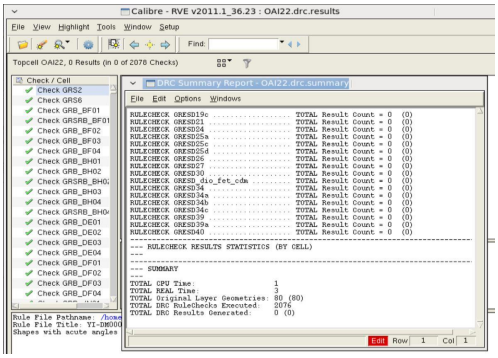
AOI21

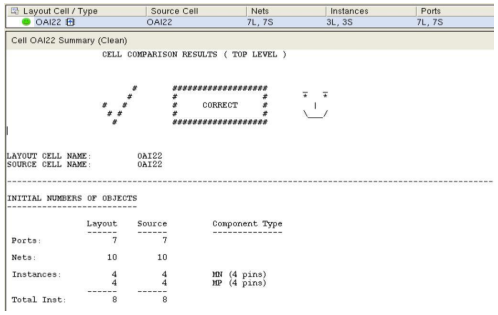

0
0.5
1
1.5
2
2.5
3
3.5
4
4.42
1.04
0.5
0


Calibre - RVE v2011.1_36.23 : AOI21.drc.results
DRC Summary Report - AOI21.drc.summary
Layout Cell / Type | Source Cell | Nets | Instances | Ports
AOI21 | AOI21 | 6L, 6S | 3L, 3S | 6L, 6S
Cell AOI21 Summary (Clean)
CELL COMPARISON RESULTS (TOP LEVEL)
CORRECT
LAYOUT CELL NAME: AOI21
SOURCE CELL NAME: AOI21
INITIAL NUMBERS OF OBJECTS
Layout Source Component Type
Ports: 6 6
Nets: 8 8
Instances: 3 3 MN (4 pins)
3 3 MP (4 pins)
Total Inst: 6 6

OAI22


0
0.5
1
1.5
2
2.5
3
3.5
4
4.42
1.3
1
0.5
0


Calibre - RVE v2011.1_36.23 : OAI22.drc.results
File View Highlight Tools Window Setup
Find:
Topcell OAI22, 0 Results (in 0 of 2076 Checks)
Check / Cell
Check GRS2
Check GRS6
Check GRB_BF01
Check GRSRB_BF01
Check GRB_BF02
Check GRB_BF03
Check GRB_BF04
Check GRB_BH01
Check GRB_BH02
Check GRSRB_BH0
Check GRB_BH03
Check GRB_BH04
Check GRSRB_BH0
Check GRB_DE01
Check GRB_DE02
Check GRB_DE03
Check GRB_DE04
Check GRB_DF01
Check GRB_DF02
Check GRB_DF03
Check GRB_DF04
Rule File Pathname: /home
Rule File Title: YI-DM000
Shapes with acute angles
DRC Summary Report - OAI22.drc.summary
File Edit Options Windows
RULECHECK GRESD19c ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD21 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD24 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD25a ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD25c ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD25d ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD26 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD27 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD30 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD_dio_fet_cdm ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD34 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD34a ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD34b ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD34c ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD39 ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD39a ..... TOTAL Result Count = 0 (0)
RULECHECK GRESD40 ..... TOTAL Result Count = 0 (0)
--- RULECHECK RESULTS STATISTICS (BY CELL)
--- SUMMARY
TOTAL CPU Time: 1
TOTAL REAL Time: 3
TOTAL Original Layer Geometries: 80 (80)
TOTAL DRC RuleChecks Executed: 2076
TOTAL DRC Results Generated: 0 (0)
Edit Row 1 Col 1


Layout Cell / Type
Source Cell
Nets
Instances
Ports
OAI22
OAI22
7L, 7S
3L, 3S
7L, 7S
Cell OAI22 Summary (Clean)
CELL COMPARISON RESULTS (TOP LEVEL)
CORRECT
LAYOUT CELL NAME: OAI22
SOURCE CELL NAME: OAI22
INITIAL NUMBERS OF OBJECTS
Layout Source Component Type
Ports: 7 7
Nets: 10 10
Instances: 4 4 MN (4 pins)
4 4 MP (4 pins)
Total Inst: 8 8

Library Characterization

After the standard cell library is completed and layouts are extracted, the library must be characterized with the Siliconsmart tool. Doing this generates a library with timing and power information about all the cells. The script runs simulations on each cell for various input slopes on all input pins and creates an output profile array from the results. After characterization, the library can be used by other tools, like Primetime, to accurately determine cell timings and power consumption with given input slopes.

This characterization process was a bit difficult in the 65nm process, since scripts had to be modified in order to work properly. The modified “siliconsmart.pl” script is included in the appendix. Here are some things we learned during the process:

1. The input, output and gnd pins are set as inout pins in the inst files by default. The siliconsmart.pl had to be modified for assigning the inputs as input pins, outputs as output pins and power pins (VDD and GND) as supply pins. The script had to be changed for dff to assign input D, clock and reset pins as input pins. Output Q as output pin and power pins (VDD and GND) as supply pins.

2. The technology file sourced in the configure.tcl script should be gf65. The name of configure.tcl has to remain the same. The Siliconsmart ACE tool does not recognize the configure script if the name changes.

3. The pin names are converted to uppercase by the Siliconsmart ACE tool. It is important to change the pin labels in the layout to uppercase so that they match with the standard library file.

Design Synthesis

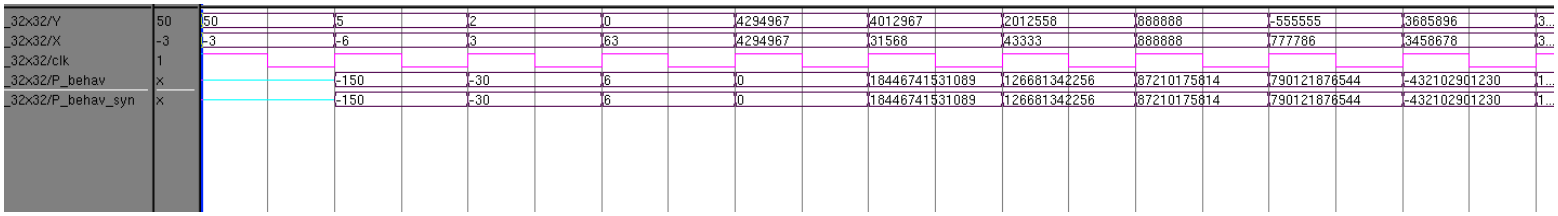
With our standard cell library complete, the verilog code is synthesized using the Design Vision tool into a structural netlist of individual logic gates from our cell library. Doing this allows the place and route tool to realize the logic using only standard gates.

Design #1: Behavioral Multiplier	7658 Cells
Design #2: Booth-2 Algorithm	23,598 Cells

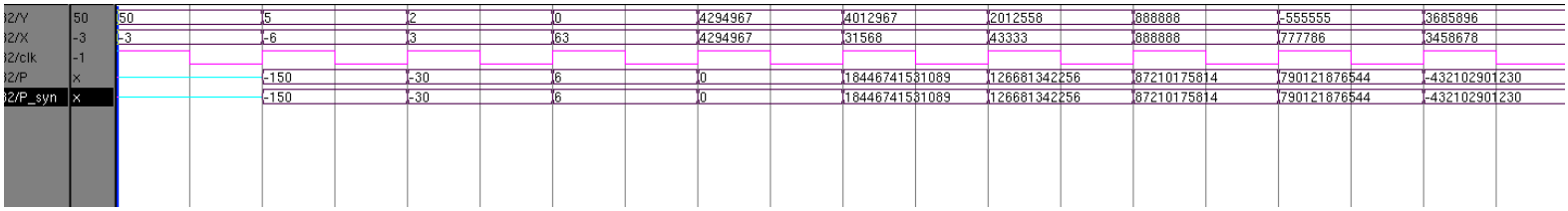
Verilog Waveform Verification of Synthesized Designs

After synthesis, the structural netlist functionality is checked using the same verilog testbench. Below are waveforms showing that the synthesized designs function correctly and give the same outputs.

1) **Behavioral Multiplier (X*Y)**



2) **Booth-2 Algorithm Multiplier**



Timing and Power Analysis

Primetime was used for timing and power analysis of both synthesized designs. Primetime uses the information from a standard cell library along with the synthesized structural netlist to simulate critical path timings and power consumption for the design. In order for the script to work properly, Primetime needs a path between two clocked DFFs or else the paths would be unconstrained. In the verilog code, we had to ensure there were registers on both the input and output bits.

Below are the results of the primetime script. The data arrival time reported is the worst case path for any two flip flops, so after this time all the output bits should be correct.

```
clock clk (rise edge)                                  0.00      20.00      20.00
clock network delay (ideal)                                       0.00      20.00
clock reconvergence pessimism                                     0.00      20.00
P_reg[61]/CLK (dff)                                                         20.00 r
library setup time                                               -0.09      19.91
data required time                                                          19.91
-------------------------------------------------------------------------------
data required time                                                          19.91
data arrival time                                                          -13.61
-------------------------------------------------------------------------------
slack (MET)                                                                  6.30


                         Internal  Switching  Leakage    Total
Power Group              Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network            1.623e-04    0.0000    0.0000 1.623e-04 ( 9.14%)  i
register                 -9.797e-05
                                   1.366e-03 2.604e-07 1.268e-03 (71.44%)
combinational            1.635e-04 1.804e-04 7.165e-07 3.446e-04 (19.42%)
sequential                  0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                      0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                      0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                   0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 1.546e-03   (87.11%)
  Cell Internal Power  = 2.278e-04   (12.83%)
  Cell Leakage Power   = 9.768e-07   ( 0.06%)
                         ---------
Total Power            = 1.775e-03  (100.00%)
```

```
clock clk (rise edge)                                  0.00      20.00      20.00
clock network delay (ideal)                                       0.00      20.00
clock reconvergence pessimism                                     0.00      20.00
Prod_reg[33]/CLK (dff)                                                      20.00 r
library setup time                                               -0.07      19.93
data required time                                                          19.93
-------------------------------------------------------------------------------
data required time                                                          19.93
data arrival time                                                          -10.39
-------------------------------------------------------------------------------
slack (MET)                                                                  9.53


                         Internal  Switching  Leakage    Total
Power Group              Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network            1.623e-04    0.0000    0.0000 1.623e-04 ( 6.72%)  i
register                 -9.805e-05
                                   1.373e-03 2.604e-07 1.275e-03 (52.78%)
combinational            4.094e-04 5.686e-04 7.684e-07 9.788e-04 (40.50%)
sequential                  0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                      0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                      0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                   0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 1.942e-03   (80.36%)
  Cell Internal Power  = 4.736e-04   (19.60%)
  Cell Leakage Power   = 1.029e-06   ( 0.04%)
                         ---------
Total Power            = 2.416e-03  (100.00%)
```

Figure 5: PrimeTime Results for Behavioral Design #1 (Left) and Booth-2 Algorithm Design #2 (Right)

	Critical Path data arrival time	**Total Power Consumption**
Design #1	13.61ns	1.775mW
Design #2	10.39ns	2.416mW

As expected, Design #2 with Booth-2 and CLA adder is faster than the purely behavioral Design #1. Design #2 is about 24% faster, but uses 36% more total power. This seems like a reasonable tradeoff. It is interesting to compare where the power is used. Both designs use the same amount of power for the clock network and registers, the difference is only in the combinational logic. Design #2 uses 2.84 times more power in the combinational logic, which is mostly due to the large carry lookahead trees.

Placement and Routing

The Library Exchange Format file (LEF) should be extracted from virtuoso and a header file should be added to this. The encounter can read only the LEF file to understand the cell library created in virtuoso. The LEF header file that was added includes design rules like minimum width, minimum area and pitch size of metal layers and the LEF file includes abstract information of the cells in the cell library.

- Edited the via rules and pitch size in the LEF to use 65nm sizes.
- Edited pitch size and cell height in the header LEF.
- Load the LEF file and synthesized verilog file in Encounter.
- Specified the floor plan with Aspect Ratio as 1, core utilization as 0.7 and core margin as 10.
- Placed the standard cells (medium congestion effort) and filler cells in rows with a row spacing of 1.04um.
- Connected Tie-hi nets to global vdd and Tie-lo nets to global ground.
- Placed the Power Rings (width of ring: 0.78 and spacing between the rings: 1.3) and Power Stripes for every row.
- Route->Nanoroute for the routing.
- pincover.tcl was run in the terminal.
- The file was saved as Design Exchange Format file (DEF)
- Because Encounter does not export the via definitions to the DEF, a perl (addvias.pl) script was used for adding the vias as per the editing done in the LEF file.
- The DEF file is imported into virtuoso.
- A cadence SKILL script called “pins.il” was written and run from the CIW terminal to add the input and output pin labels to the layout view. This was needed so that the layout and schematic had the same number of ports in order to pass LVS. The Calibre tool performs the LVS test based on the pin labels rather than the pins placed, so having pin labels for all the input and output pins is important.

Layout, DRC and LVS Reports of the Final Design

- In order to pass DRC, had to update via object sizes to 65nm rules in the .lef and .def (by modifying addvias.pl)
- To pass LVS on Design #2, had to replace some wire names in the synthesized netlist because hspice is case sensitive, but Calibre is not.

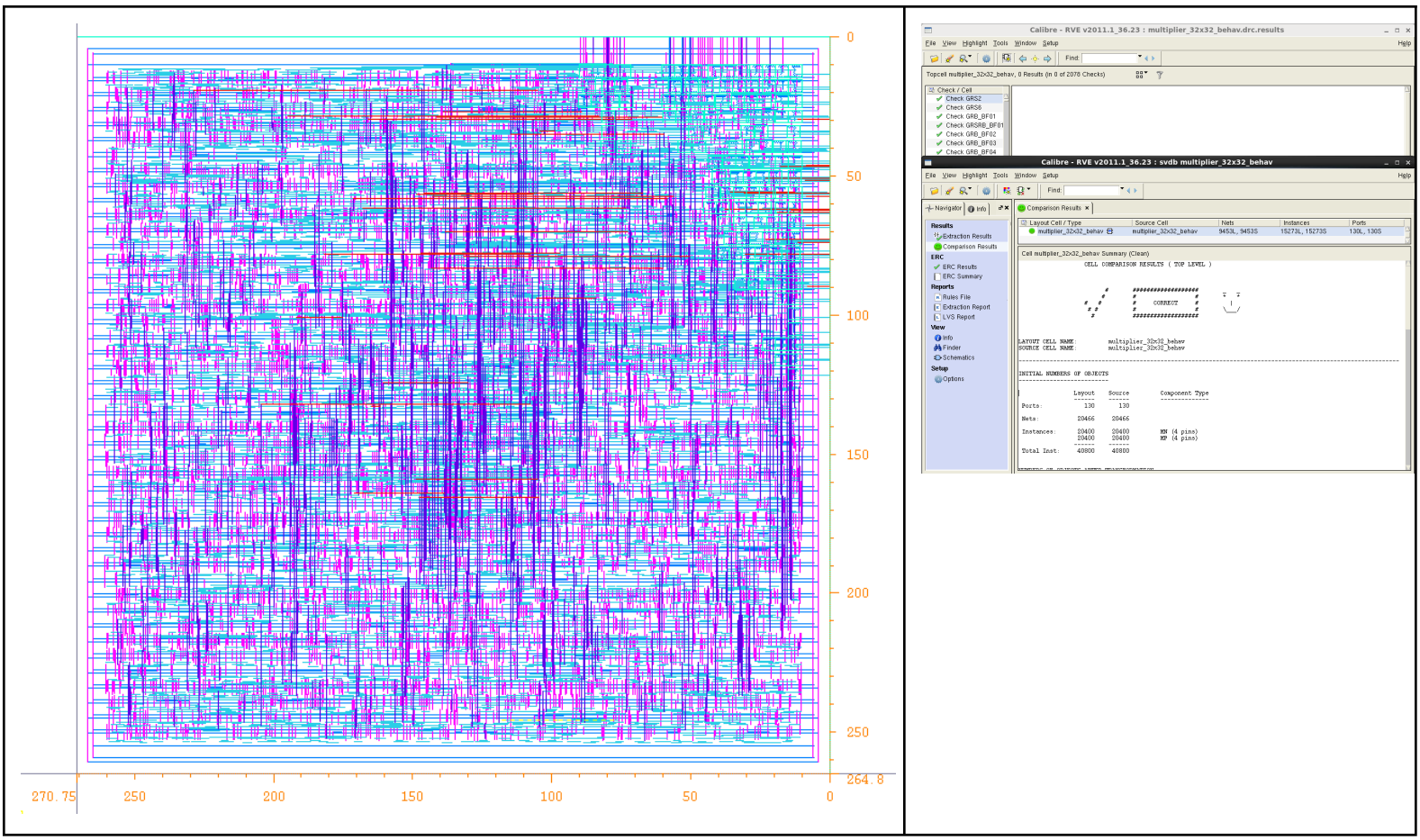


Figure 6: Final Layout, DRC report and LVS report for Design #1 Behavioral Multiplier

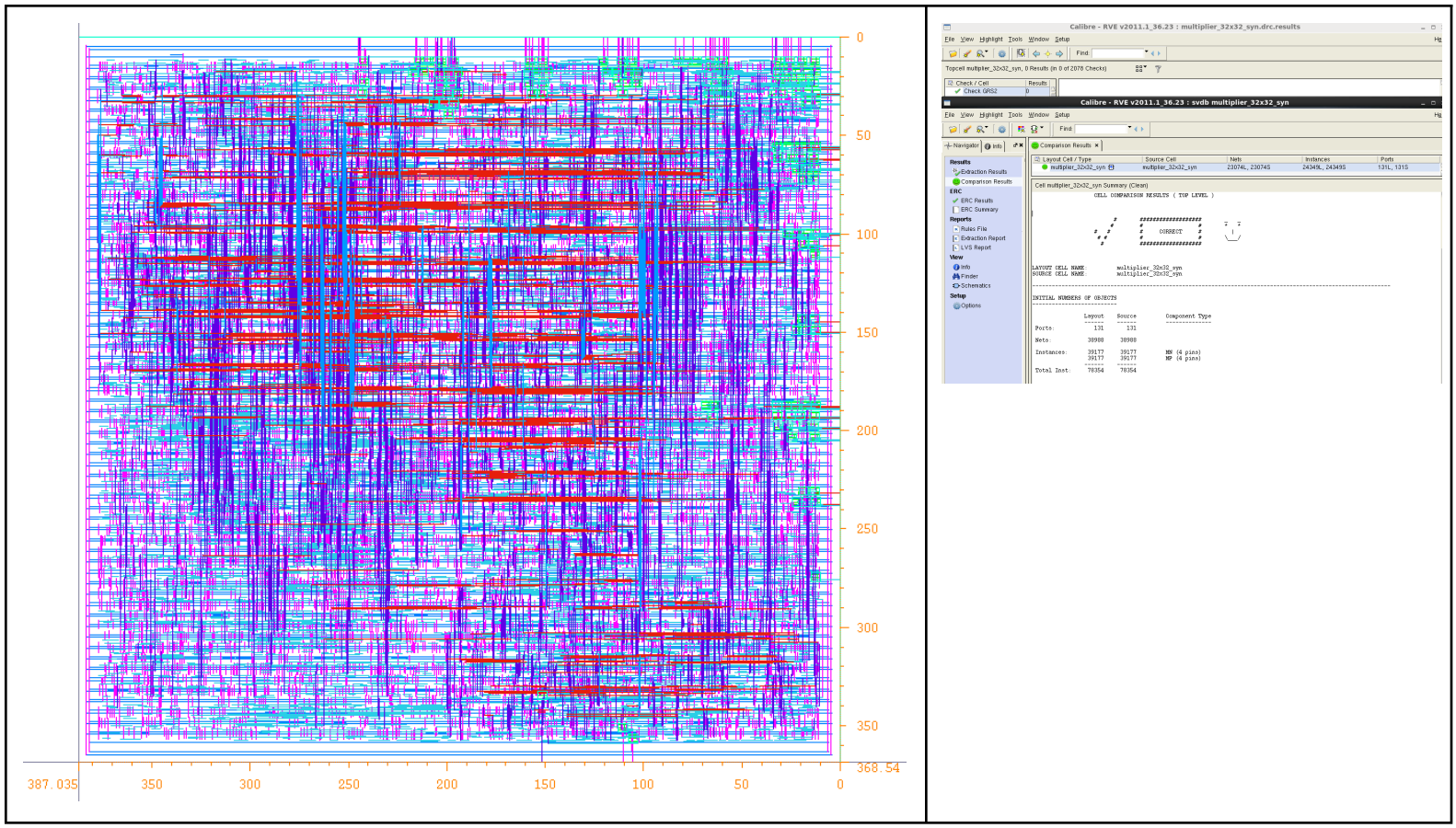
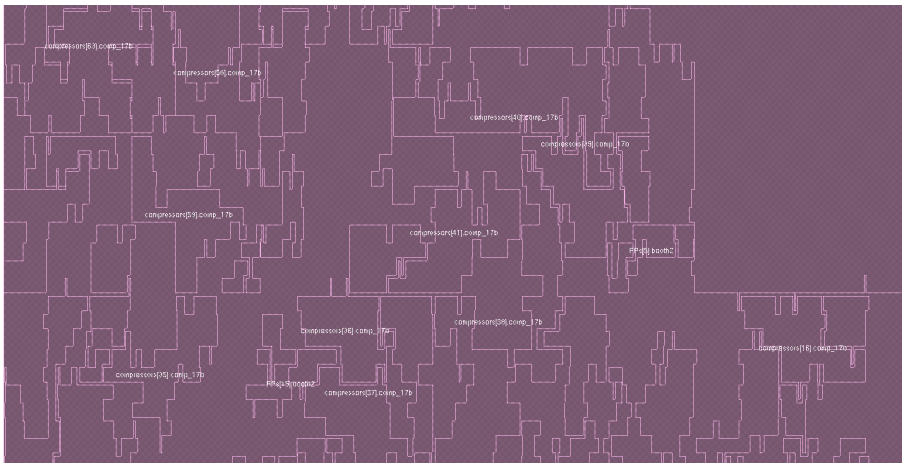


Figure 7: Final Layout, DRC report and LVS report for Design #2 Booth-2 Algorithm Multiplier

“Amoeba” view of Design #2, showing how each module was placed into the layout. The final adder with lookahead trees takes up nearly ¼ of the total area.



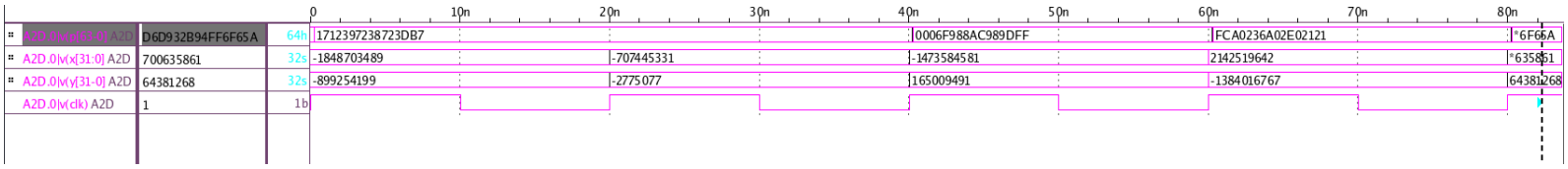
“Amoeba” view detail, showing numerous compressors and Booth-2 PP modules

Design Verification in Hspice

After the chip layout is complete, the layout is extracted to a spice netlist and the final functionality of the chip is verified using Hspice. The Hspice test file used is included in the Appendix: “multiplier_verification.sp”. Output pins were loaded with 25fF. Each input pin was controlled by using the PWL hspice function.

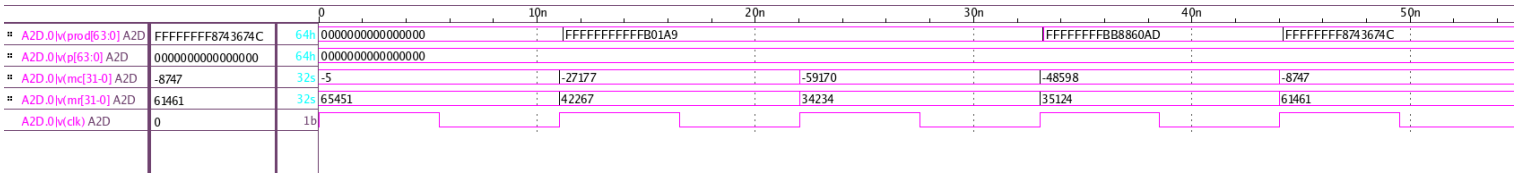
Below are waveforms showing the correct functionality of both multiplier designs. For the 32-bit inputs, large numbers were chosen to exercise most of the bits. The product is reported in hexadecimal because WaveView does not display 64-bit signed decimal numbers correctly. On the first clock rising edge, the inputs are latched into the input registers. On the next clock rising edge, the outputs are latched into the output register, at the same time new inputs are latched into the input register. In this way, there is 1 clock cycle of delay before the output product becomes correct.

Behavioral Multiplier (X*Y)



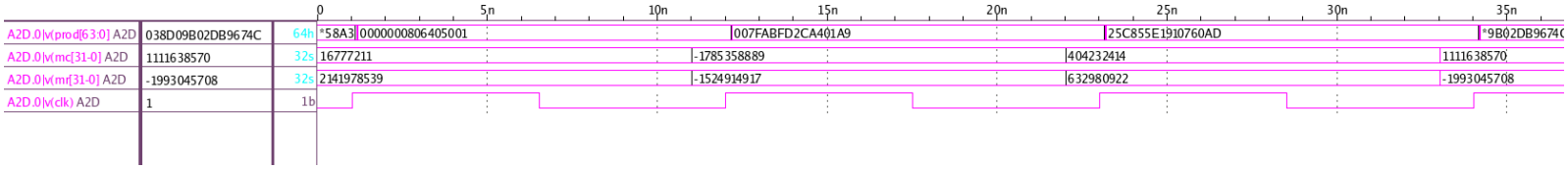
Clock Edge #	Time Index (ns)	Input Y (multiplicand)	Input X (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	20	-899,254,199	-1,848,703,489	1712 3972 3872 3DB7	1,662,454,375,189,200,311	Y
2	40	-2,775,077	-707,445,331	0006 F988 AC98 9DFF	1,963,215, 266,815	Y
3	60	165,009,491	-1,473,584,581	FCA0 236A 02E0 2121	-243,155,441,656,258,271	Y
4	80	-1,384,016,767	2,142,519,642	D6D9 32B9 4FF6 F65A	-2,965,283,108,154,837,414	Y

Booth-2 Algorithm Multiplier



Clock Edge #	Time Index (ns)	Input MC (multiplicand)	Input MR (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	25	-5	65,451	FFFF FFFF FFFB 01A9	-327,255	Y
2	45	-59,170	34,234	FFFF FFFF 8743 674C	-2,025,625,780	Y

Design #2 was also simulated using a clock of 11ns to show functionality at this increased speed.



Clock Edge #	Time Index (ns)	Input MC (multiplicand)	Input MR (multiplier)	Output Product (Hexadecimal)	Output Product (Decimal)	Correct Output?
1	12	16,777,211	2,141,978,539	007F ABFD 2CA4 01A9	35,936,425,906,274,729	Y
2	23	-1,785,358,889	-1,524,914,917	25C8 55E1 9107 60AD	2,722,520,402,034,647,213	Y
3	34	404,232,414	632,980,922	038D 09B0 2DB9 674C	255,871,406,116,005,708	Y

32-bit Multiplier Design Trade-Offs

The Booth-2 Algorithm Multiplier which uses a Carry Lookahead Adder for the final 64-bit addition is faster as compared to the Behavioral Multiplier Design. The additional hardware increases the speed by 24%. However, it increases the Total Power Consumption by 36%. The area for Design #2 is twice that of Design #1. This meets our expectation that increasing speed by using a CLA adder is very costly in terms of power and area, however it is likely still a good trade off for many applications with modern technology processes.

The Booth-2 Algorithm Multiplier uses 15 Full Adders for each column in the "compressor" stage to reduce 16 Partial Products to two rows. The 64 columns use a total of 960 Full Adders. Many of these Full Adders add zeroes. The design can further be improved by using fewer adders in the process. This would reduce the area and the power consumption, but would require significant effort to fine tune the design.

Below is a table summarizing the two designs. It is easy to see that the increased speed of Design #2 is paid for with area and power.

	Number of Cells	**Total Area (sq. um)**	**Total Power Consumption (mW)**	**Data Arrival Time (ns)**
Design #1 (Behavioral)	7,658	71,694.6	1.775	13.61
Design #2 (Booth-2 Algorithm)	23,598	142,637.9	2.416	10.39