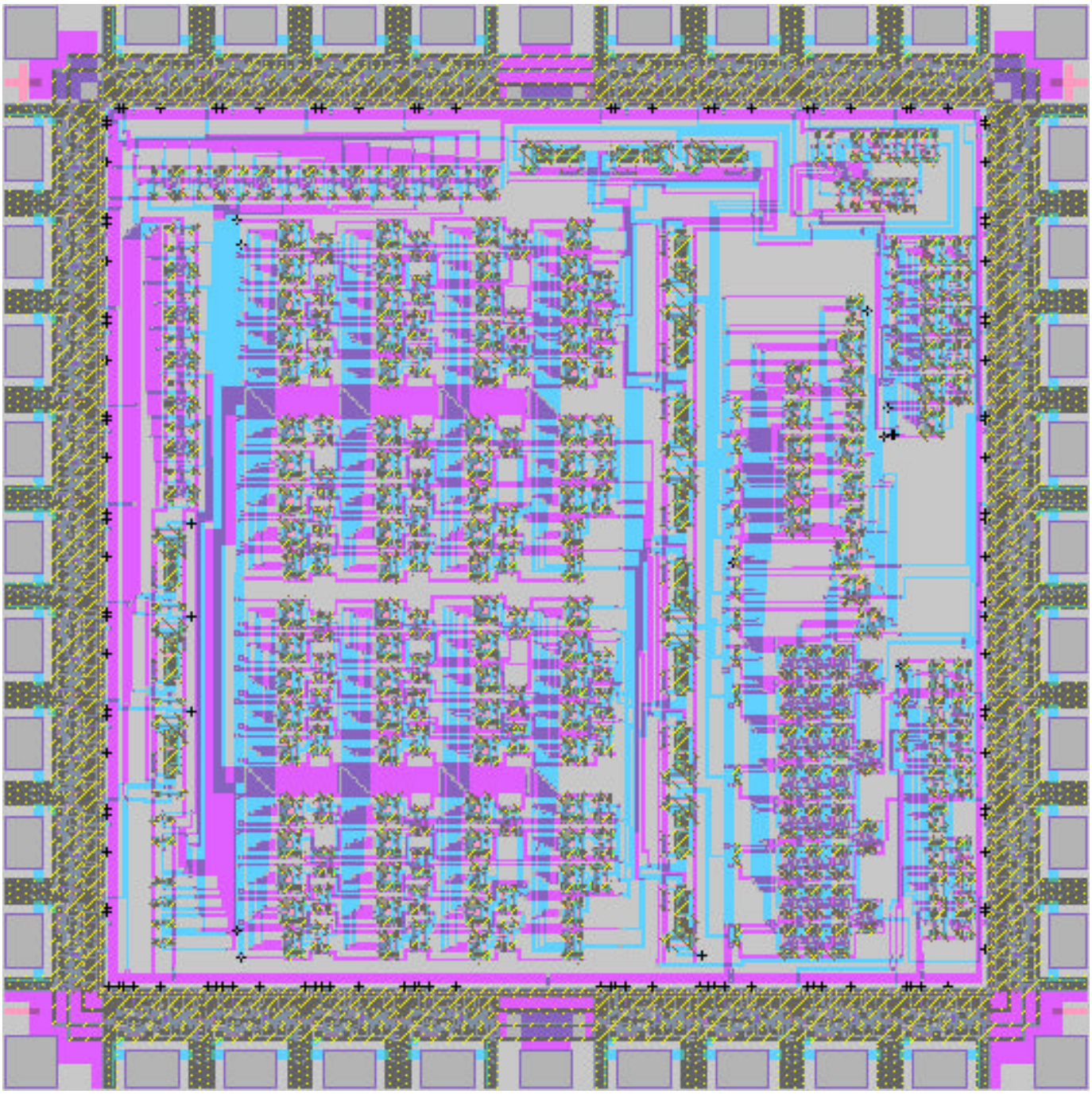
E158 Final Report

Ryan Jackson & Lance Feagan

IEEE 754 Floating Point Addition Unit

Color Chip Plot



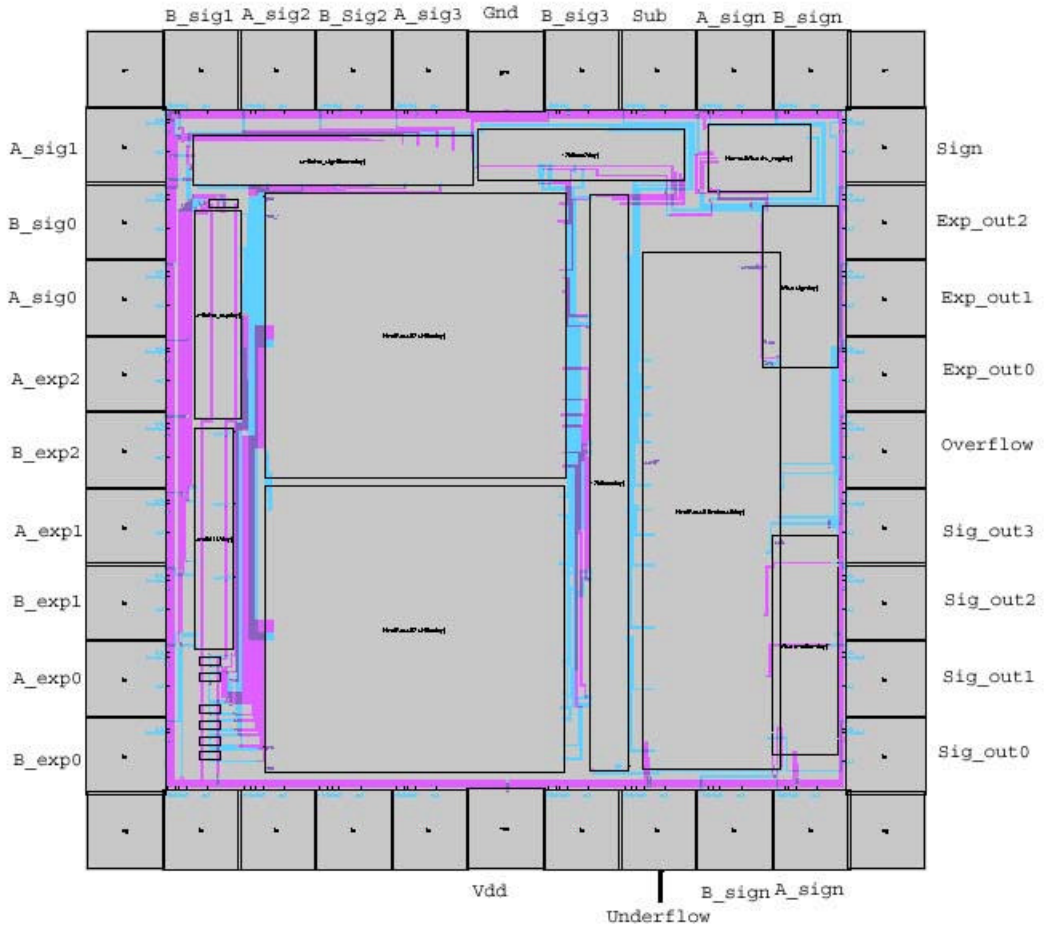
Functional Overview

We implemented a subset of the IEEE 754 Floating Point Standard. Our Floating Point Unit (FPU) chip handles addition and subtraction. It does not handle standard single or double precision numbers as the complexity of the needed circuitry exceeds our project's time and space resources. Instead we devised an 8-bit variation for our chip. It should be noted that although the single or double precision number standard is not implemented, this 8-bit subset and could easily be scaled, requiring more time and space, and thus represents a proof of concept for a larger design. In our variation the first bit represents the sign of the number. This is in the standard manner of (-1)^(sign). The next 3 bits represent the exponent of the number. This is in the manner of (2)^(exp). For versatility an offset of 3 is utilized. This gives a smallest exponent of -3 and a largest exponent of +4. The significand comprises the remaining 4 bits. This allows the representation of fractions as small as 1/8 (0.125) with increments of 1/128 (0.0078125). This allows the representation of positive numbers as large as 31 and as small as -31.

There are overflow and underflow exception outputs. The number output during an overflow or underflow are not meaningful. We did not implement the NaN (Not a Number), infinity, or denormalized number special events included in the standard. Additionally, numbers are rounded using truncation.

Chip Pinout

There are 40 available pins on our chip, including the power and ground, but we are using only 33.



Inputs: A_exp0,1,2; B_exp0,1,2; A_sig0,1,2,3; B_sig0,1,2,3; A_sign; B_sign; Sub

Outputs: Exp_out0,1,2; Sig_out0,1,2,3; Overflow; Underflow; Sign;

Inputs

1 Pins Vdd
1 Pins Gnd

1 Add/Sub Selector- Determines whether operation is addition or subtraction

2 x 8-bit Arguments- The two numbers to be operated on, must be in above format

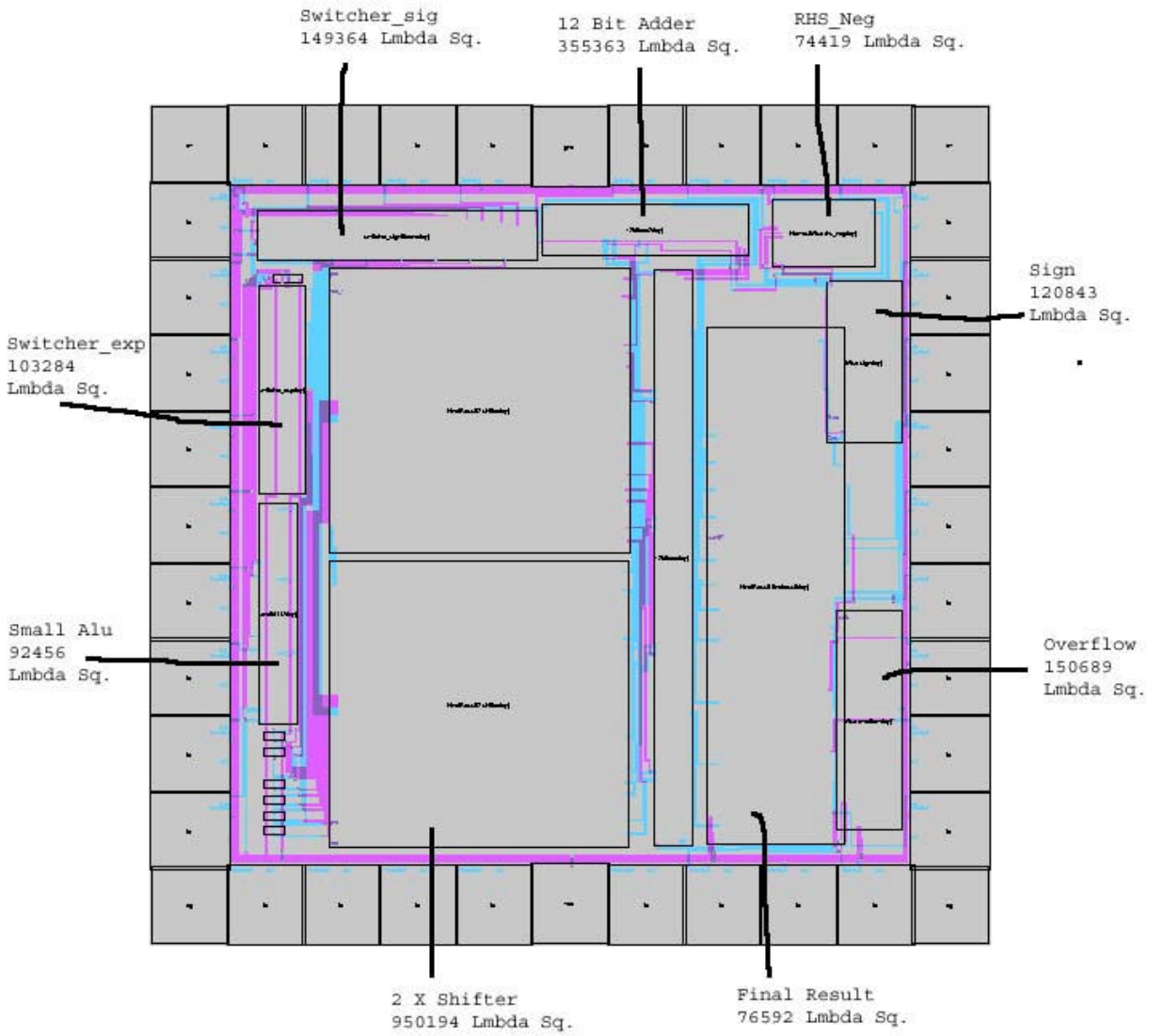
Outputs

8-bit Result- The result of the FPUs computation in above format

1 Overflow- To detect if FPU yields an answer greater than can be represented in above format

1 Underflow- Same as overflow, except for numbers which are too small

Chip Floorplan



Area and Design Time Data

The time reported is in man-hours. The facets below that have NA are not applicable because they were integrated into another facet for convenience.

Schematic

	Logic Creation Time (hrs.)	**Draw Time (hrs.)**
8-bit Mux	**1**	**1**
adder 12-bit	1	2
final result	4	6
overflow	3	1.5
rhs_neg	3	0.5
shifter	10.5	8
sign	2	1
small ALU	0.5	1
switcher_exp	2	1.5
switcher_sig	2	0.5
underflow	2	0
top level	1	8
total:	30	30

Layout	**Area (lambda squared)**	**Design Time (hrs.)**
8-bit Mux	59148	1
adder 12-bit	355363	2
final result	766592	12
overflow	150689	1
rhs_neg	74419	1
shifter	950194	18
sign	120843	2
small ALU	92456	1
switcher_exp	103284	2
switcher_sig	149364	2
underflow	NA	NA
top level	7584516	19
total:	7584516	60

Simulation Results:

Verification Results:

Cell Name	Complexity	Layout	Compelete	DRC	ERC	NCC
12-bit Adder	3	X		X	X	X
And2	1	X		X	X	X

And3	1X	X	X	X
And4	1X	X	X	
Binvert	2X	X	X	X
Complete Design	5X	X	X	
Final Result	4X	X	X	
FullAdder	2X	X	X	X
Inv	0X	X	X	X
Mux2	1X	X	X	X
Mux4	1X	X	X	X
Mux8	2X	X	X	
Or2	1X	X	X	X
Or3	1X	X	X	X
Or4	1X	X	X	
Overflow	3X	X	X	X
RHS_neg	2X	X	X	
Shifter	5X	X	X	
Sign	3X	X	X	
Small ALU	2X	X	X	X
Switcher_exponent	2X	X	X	
Switcher_significand	2X	X	X	
Xor2	1X	X	X	X

Many areas of our design did not past NCC. However, they did simulate correctly. The layout and schematics have identical logic, however, often times the nets were very different in their structure. As a result of NCC not passing, we took copious efforts to test all parts to verify the layout functioning properly. All parts did successfully pass our simulation tests.

Postfabrication Test Plan

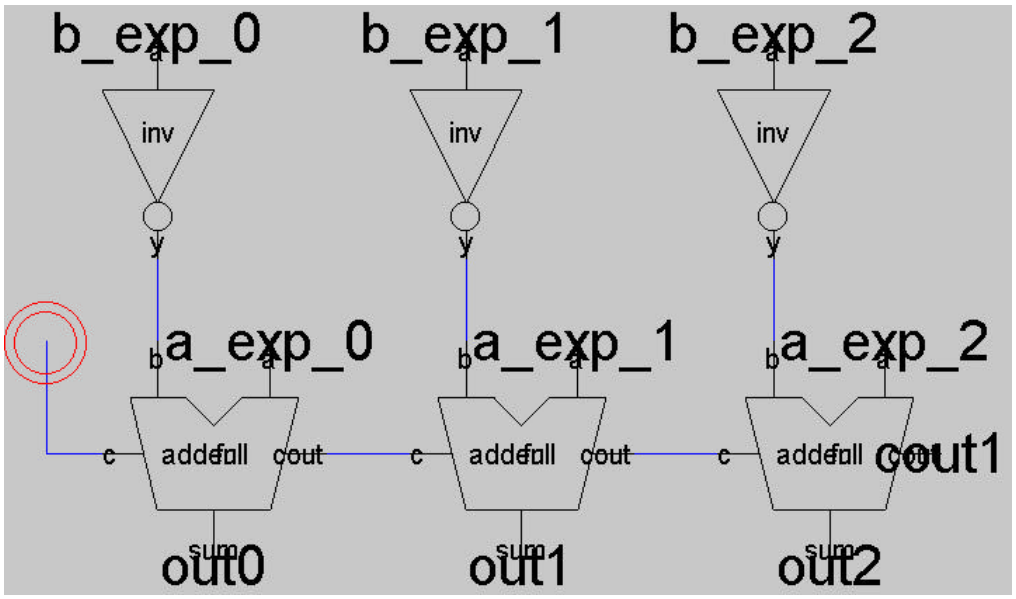
A good way to start with testing this chip would be to test the corner cases upon receiving it back. The following chart gives a set of test vectors (corner cases) that should ensure functionality. Reading from right to left are the test#, input A, sign, input B, output, and notes. Zeros and ones for each of the input and output bits correspond to high and low voltage respectively. For each of the inputs (A, B), the eight bits are given, read from left to right, as sign, exp, and significand. Underneath each of these inputs is the decimal representation of the test input. The bit to be entered is in the next column to the right of the inputs. The output is represented in its eight-bit form in the same manner as the inputs. Beneath the bit representation is the binary representation of the output. Given in this manner, it is hoped that the tester will be able to more easily correlate the output bits with the calculation performed. Finally, in the column to the far right are some notes the user may find useful.

test#	input A	sign	input B	output	notes:
1	0 101 0000	0	0 011 0000	0 101 0100	
	4	+	1	1.01*2^2	

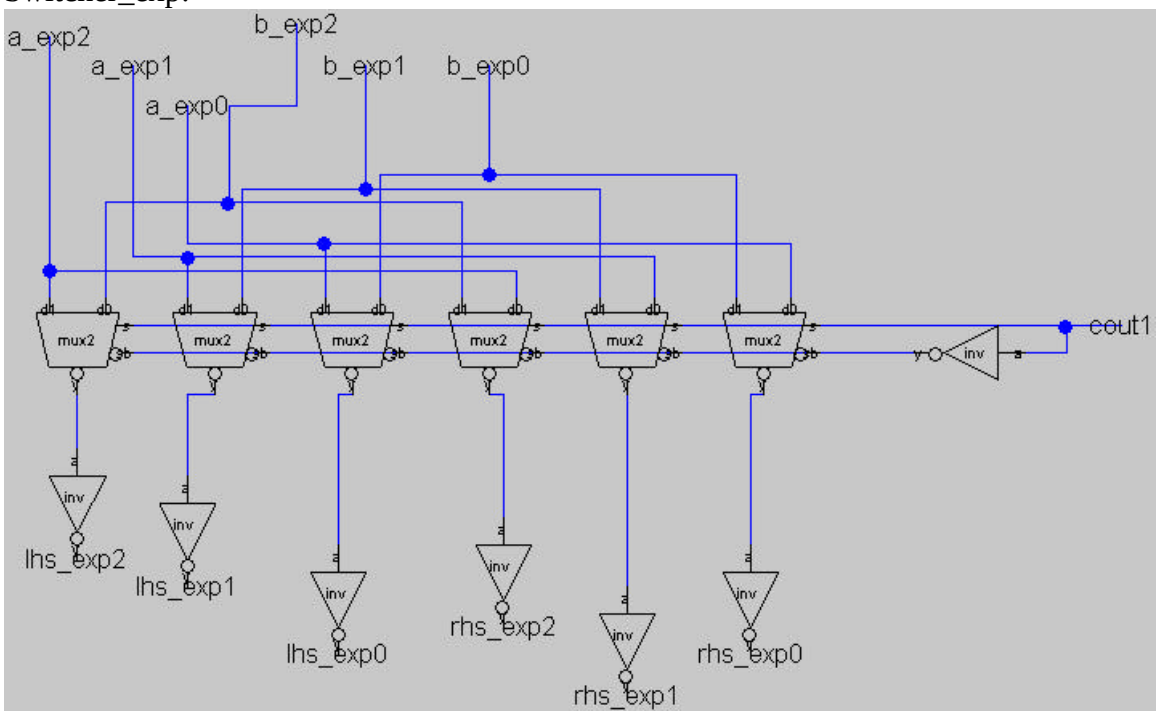
2	0 000 0000	0	0 011 0000	0 011 0010	
	0.125	+	1	1.001*2^0	
3	0 101 0000	0	0 000 0000	0 101 0000	
	4	+	0.125	1.00001*2^2	0.125 does not appear in result
4	0 101 0000	1	0 011 0000	0 100 10000	
	4	-	1	1.1*2^1	
5	0 011 0000	1	0 101 0000	1 100 1000	
	1	-	4	-2.2	
6	1 000 0000	0	0 011 0000	0 010 1100	
	-0.125	+	1	1.11*2^-1	
7	1 000 0000	1	0 011 0000	1 011 0010	
	-0.125	-	1	-1.001	
8	1 000 0000	1	1 011 0000	0 010 1100	
	-0.125	-	-1	1.11*2^-1	
9	0 101 0000	0	1 000 0000	0 100 1111	
	4	+	-0.125	1.1111*2^1	
10	0 110 0000	0	1 000 0000	0 101 1111	
	8	+	-0.125	1.11111*2^2	truncation of last bit
11	0 111 0000	0	0 111 0000	overflow	
	16	+	16	1*2^5	
12	0 011 0001	1	0 011 0000	underflow	
	1.0625	-	1	1*2^-4	
13	0 000 0000	0	0 000 0000	0 001 0000	
	0.125	+	0.125	1*2^-2	default with zero inputs
14	0 110 0101	0	0 101 0101	0 110 1111	
	10.5	+	5.25	1.11111*2^3	truncation of last bit
15	0 110 0101	1	0 110 0101	underflow	
	10.5	-	10.5	0	

Schematics

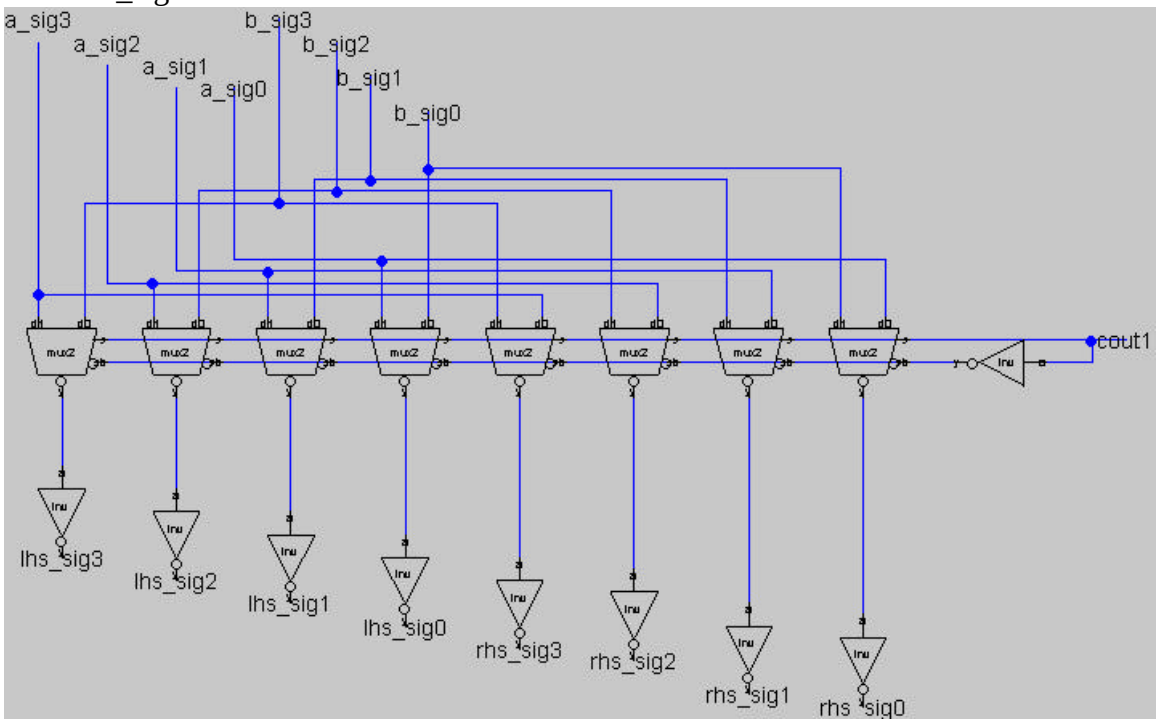
Small ALU:



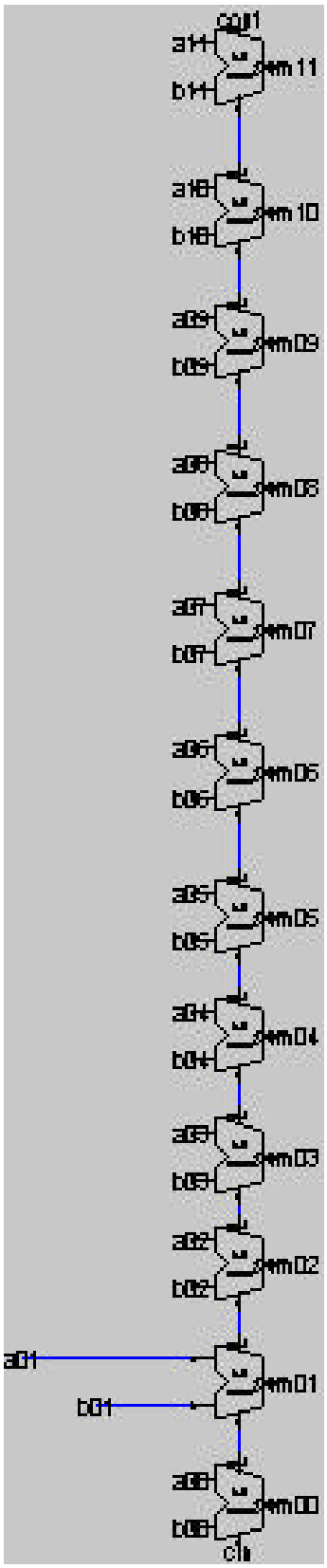
Switcher_exp:



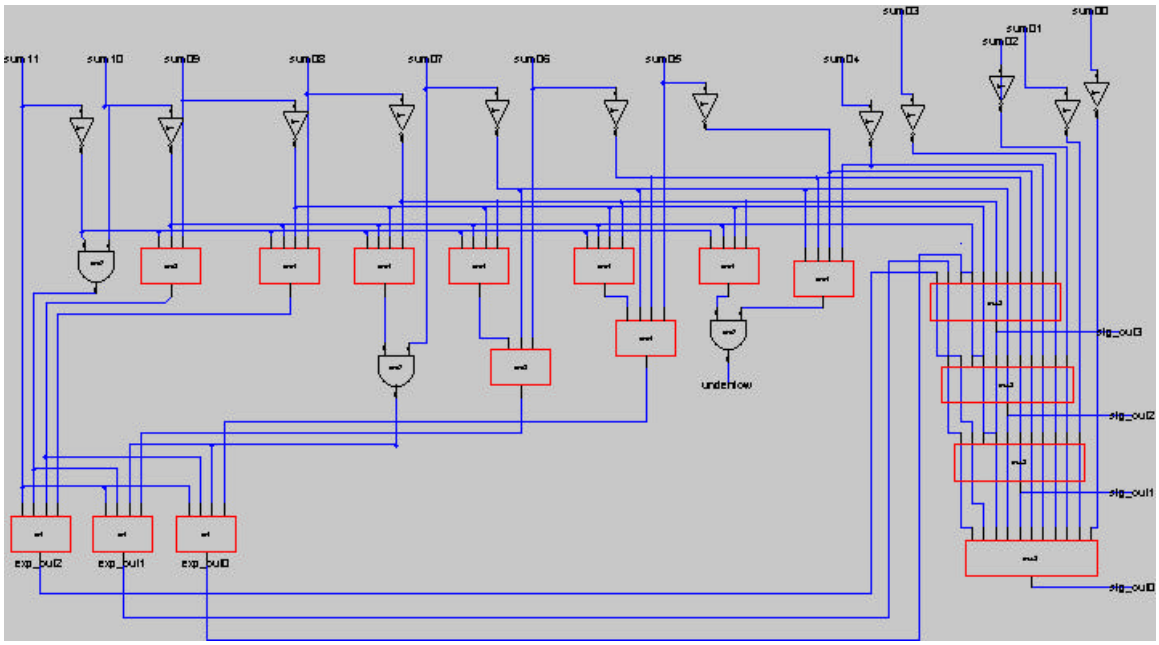
Switcher_sig:



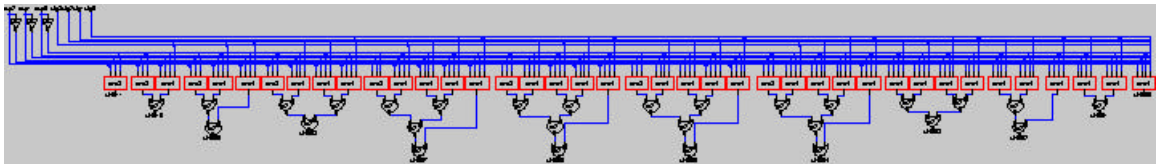
12 Bit Adder:


a11
b11
sum11
a10
b10
sum10
a09
b09
sum09
a08
b08
sum08
a07
b07
sum07
a06
b06
sum06
a05
b05
sum05
a04
b04
sum04
a03
b03
sum03
a02
b02
sum02
a01
b01
sum01
a00
b00
sum00

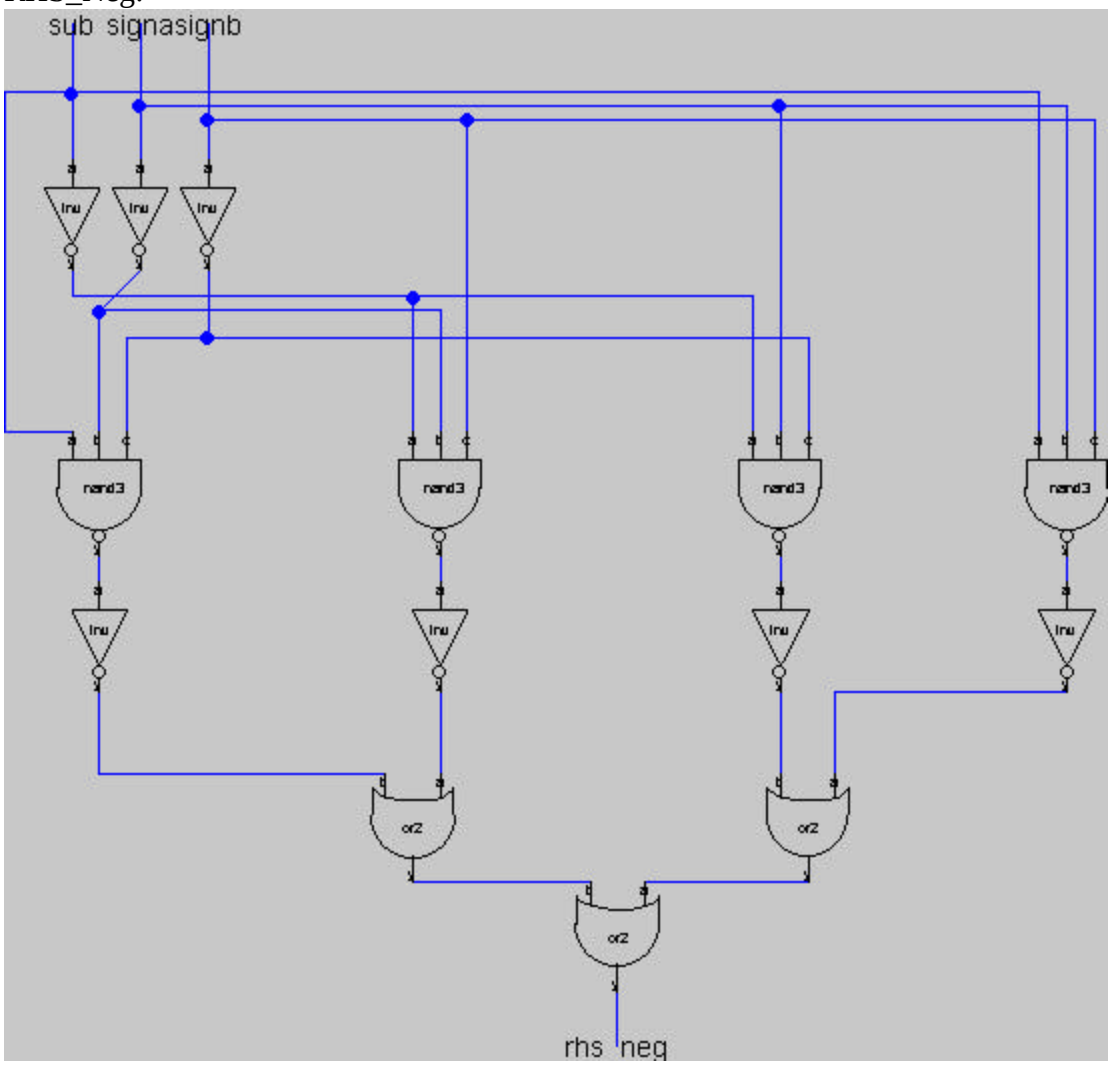
Final Result:



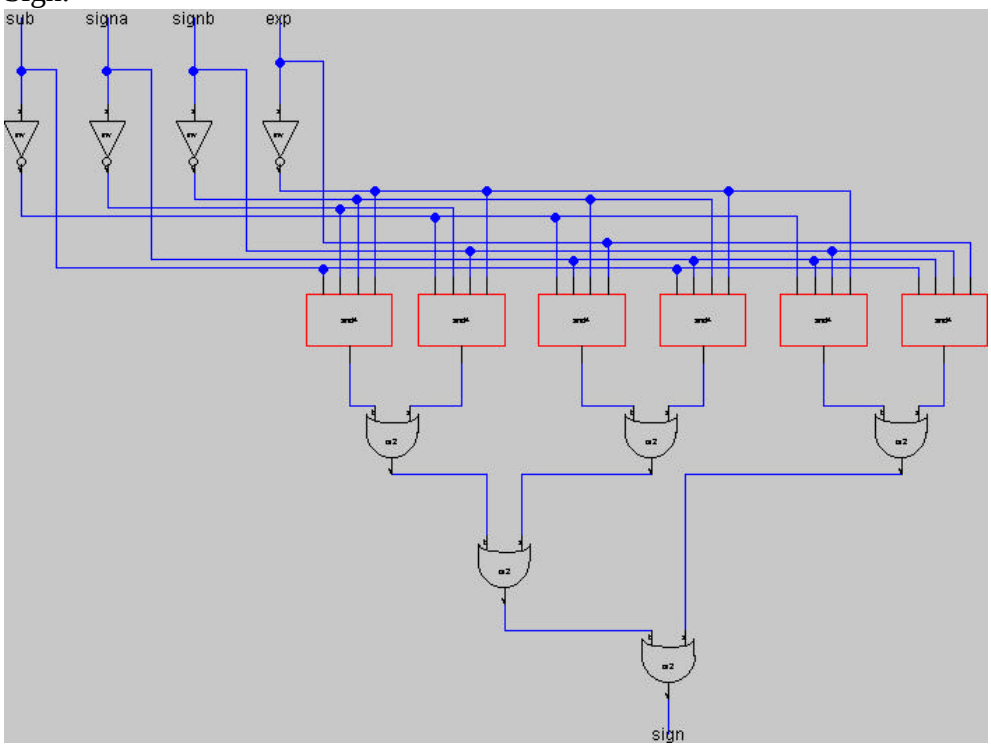
Shifter:



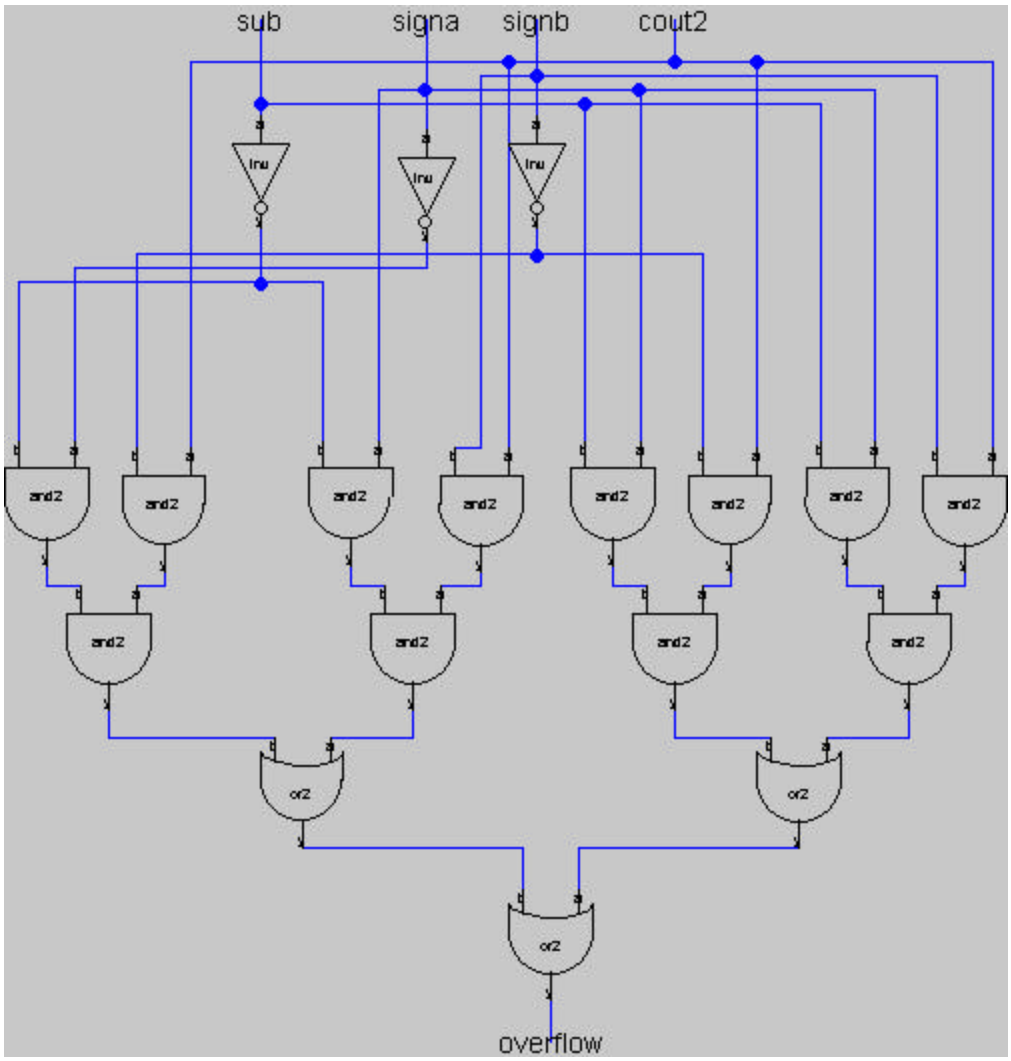
RHS_Neg:


sub signasignb
inu
inu
inu
nand3
nand3
nand3
nand3
inu
inu
inu
inu
or2
or2
or2
rhs neg

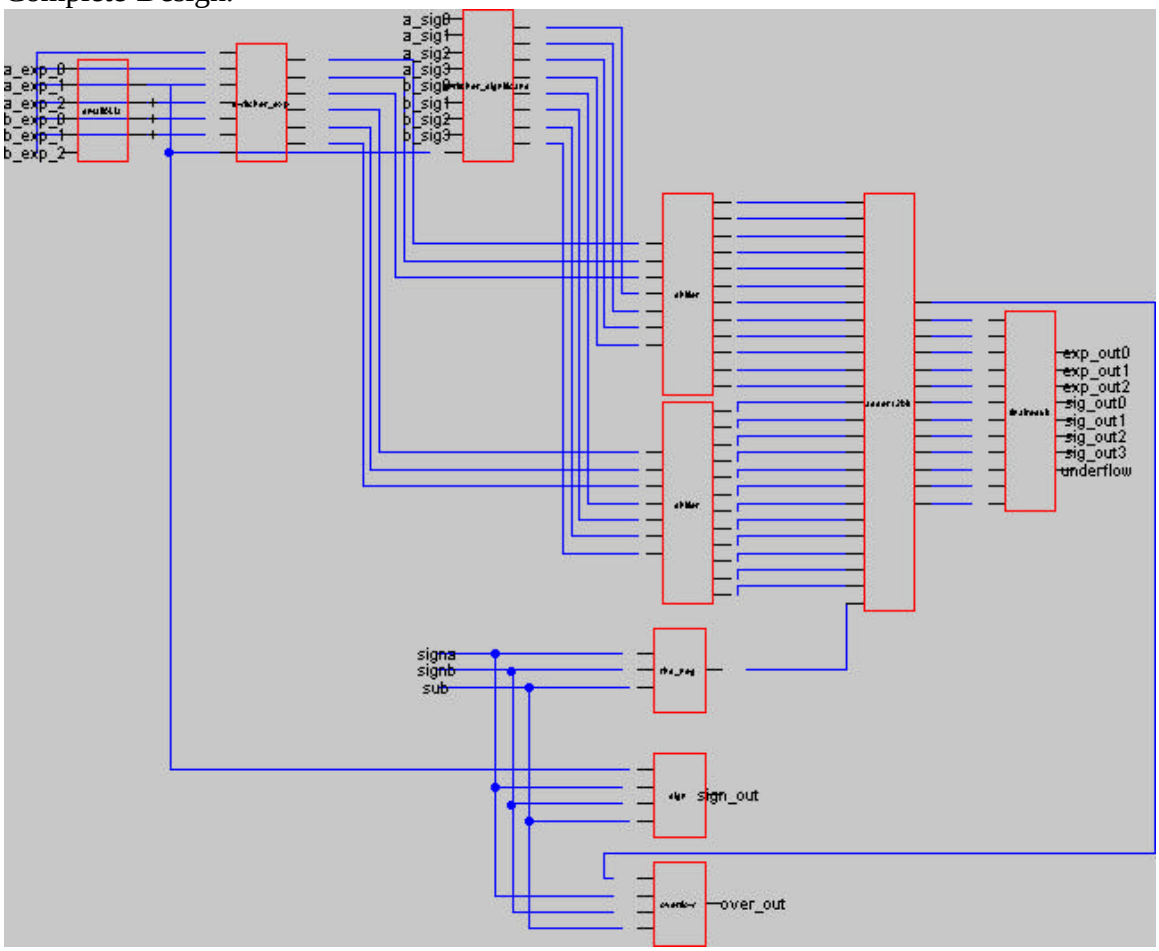
Sign:



Overflow:

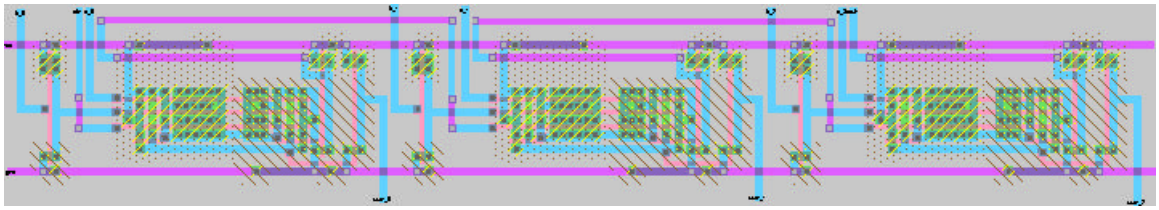

sub
signa
signb
cout2
inv
inv
inv
and2
and2
and2
and2
and2
and2
and2
and2
and2
and2
and2
and2
or2
or2
or2
overflow

Complete Design:

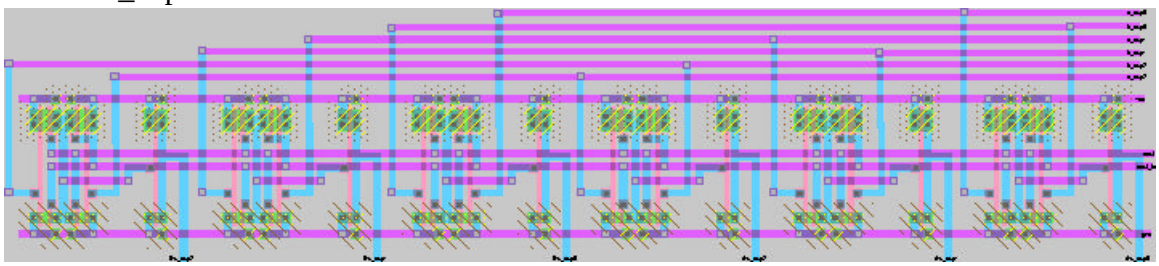


Layout:

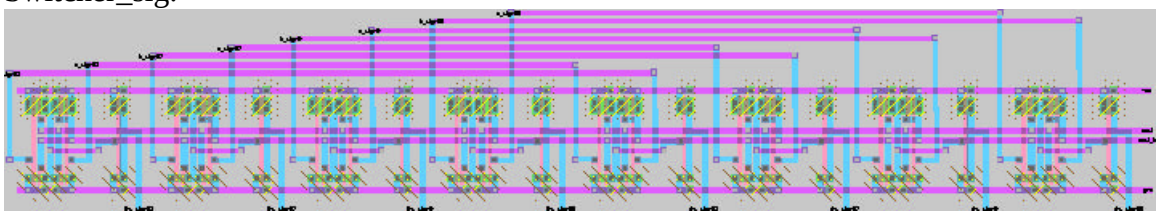
Small ALU:



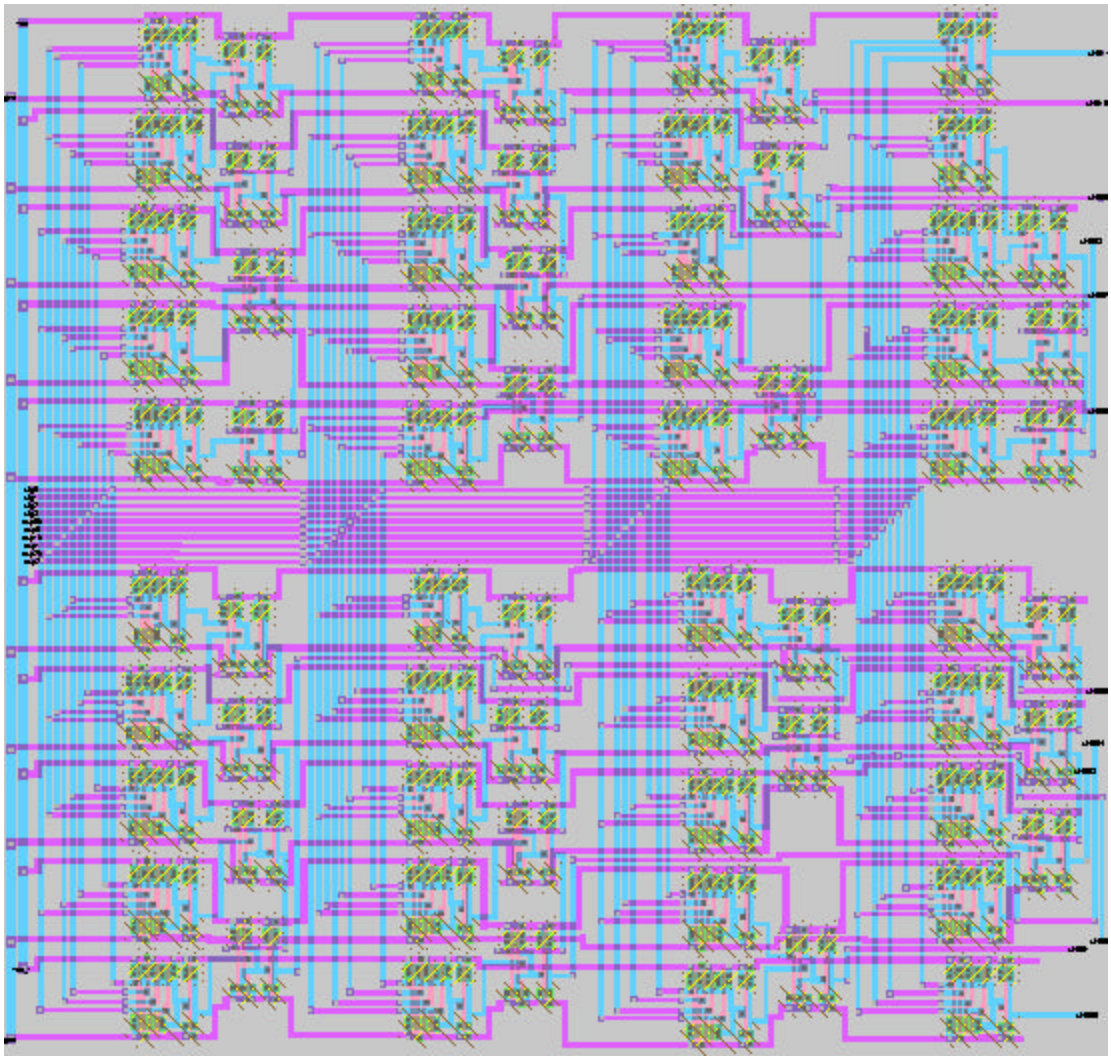
Switcher_exp:



Switcher_sig:

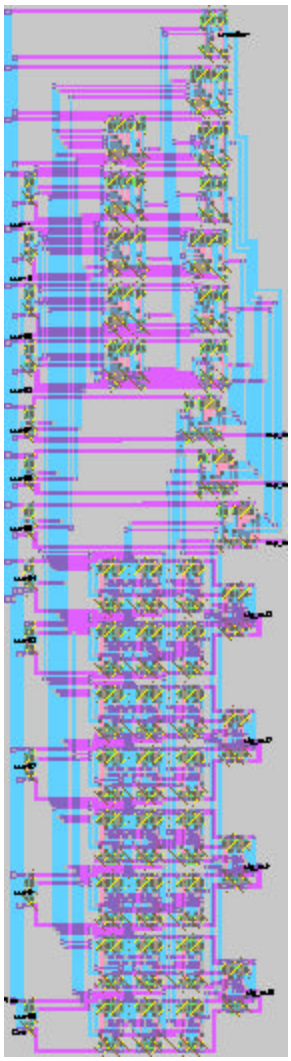


Shifter:

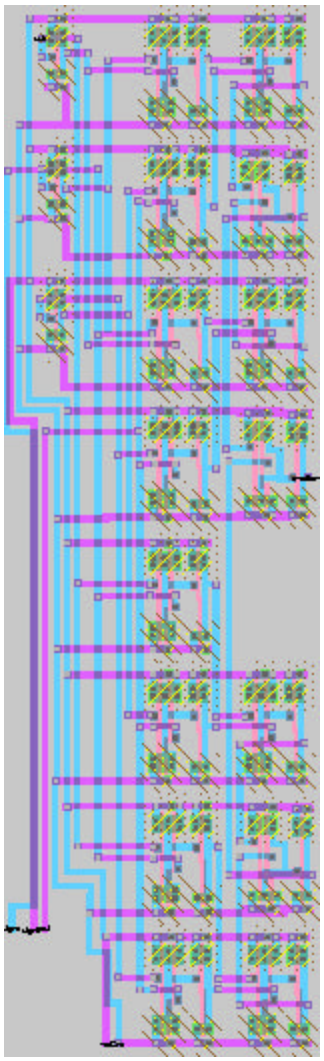


12 Bit Add: (has two parts because of a length issue)

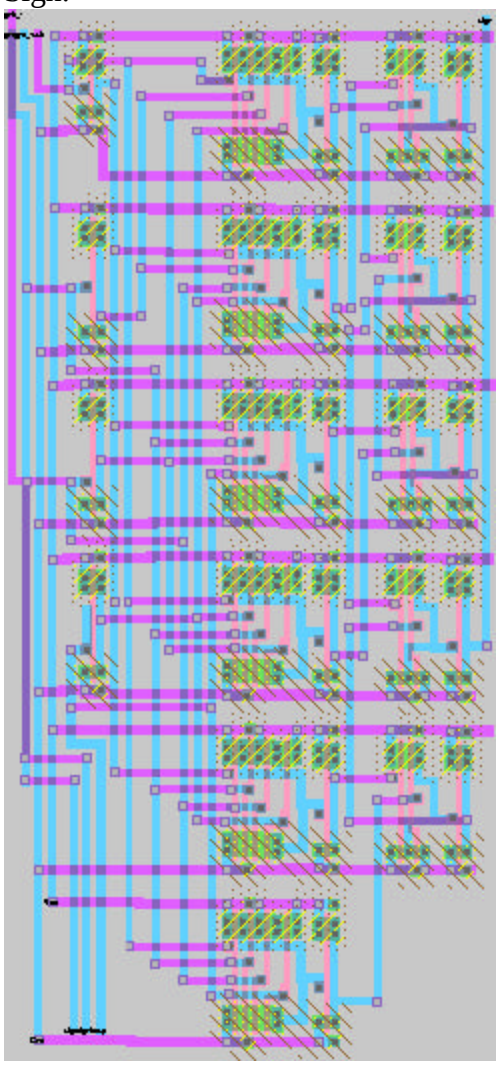
Final Result:



Overflow:



Sign:



RHS_Neg: