Introduction

Multiplication is a heavily used arithmetic operation that is prominently used in signal processing and scientific applications. In order to speed up the process of multiplication (multiple number of shifting and adding), Booth's algorithm is used which reduces the number of required partial products in half in turn reducing the hardware and delay required to sum the partial products.

Project Description:

- 21b X 21b multiplier design with emphasis on speed.
- The schematic is designed using the IBM 130 nm process technology
- Input operands are assumed to be positive
- Design is verified using Hspice
- Siliconsmart ACE is used to characterize the cells
- Power and delay found from Primetime
- The design uses Booth-2 algorithm, the partial products are compressed and a combination of carry look ahead adder and carry select adder is used.

Booth-2 Algorithm:

In this algorithm, adjacent pairs of bits of the *N*-bit multiplier are examined including an implicit bit below the least significant bit ($Yi_{-1} = 0$). These pairs are used to generate the partial products from the multiplicand by either multiplying it by 1 (i.e. no change), multiplying it by 2 (shift left by one bit), multiplying it by -1 (2's complement) or multiplying it by -2(2's complement and shift left by one bit). The recoded digits are shown in Table 1. These partial products are shifted by two bits for each partial product after the first. The product is equal to the sum of these terms. This algorithm reduces the number of partial products from n to n/2 there by improve the speed of the multiplication.

Yi_{+1}	Yi	Yi_{-1}	Recoded Digit
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

Table 1

Architecture:

A high speed 21 bit x 21bit multiplier is designed using Booth 2 algorithm. The block diagram of the complete design is as shown in Figure 1.

The 21bit multiplier is divided into 2 bits of 11 groupings. Each of these groupings is passed into a Recoder, whose output bits corresponding to the operations described in Table 1.Each group of these selection bits are sent to a PPMux block which outputs the appropriate partial product bits based on the selected operation. These partial products are then sign extended. Then the rows of partial products are compressed. The output compressed bits are then added using a high speed, combination of carry look ahead adder and carry select adder to output the final product.

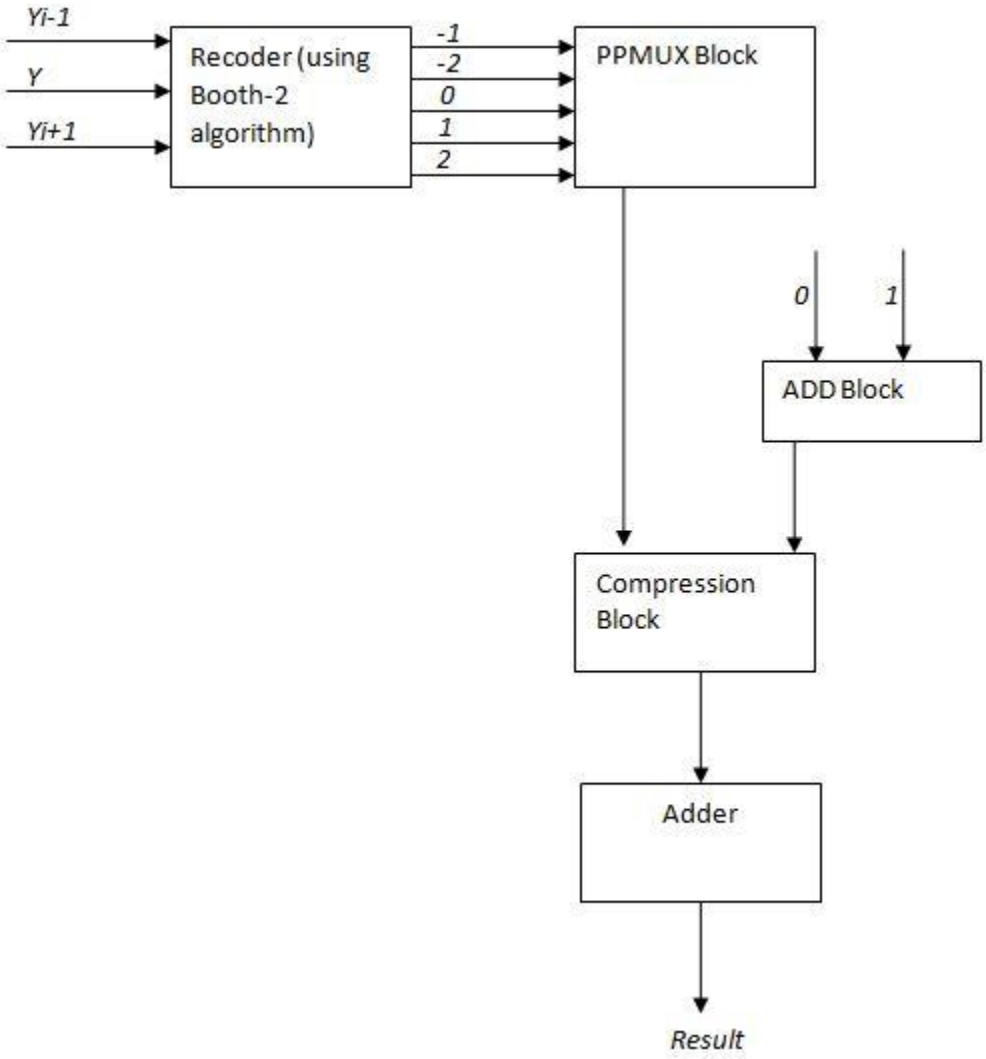


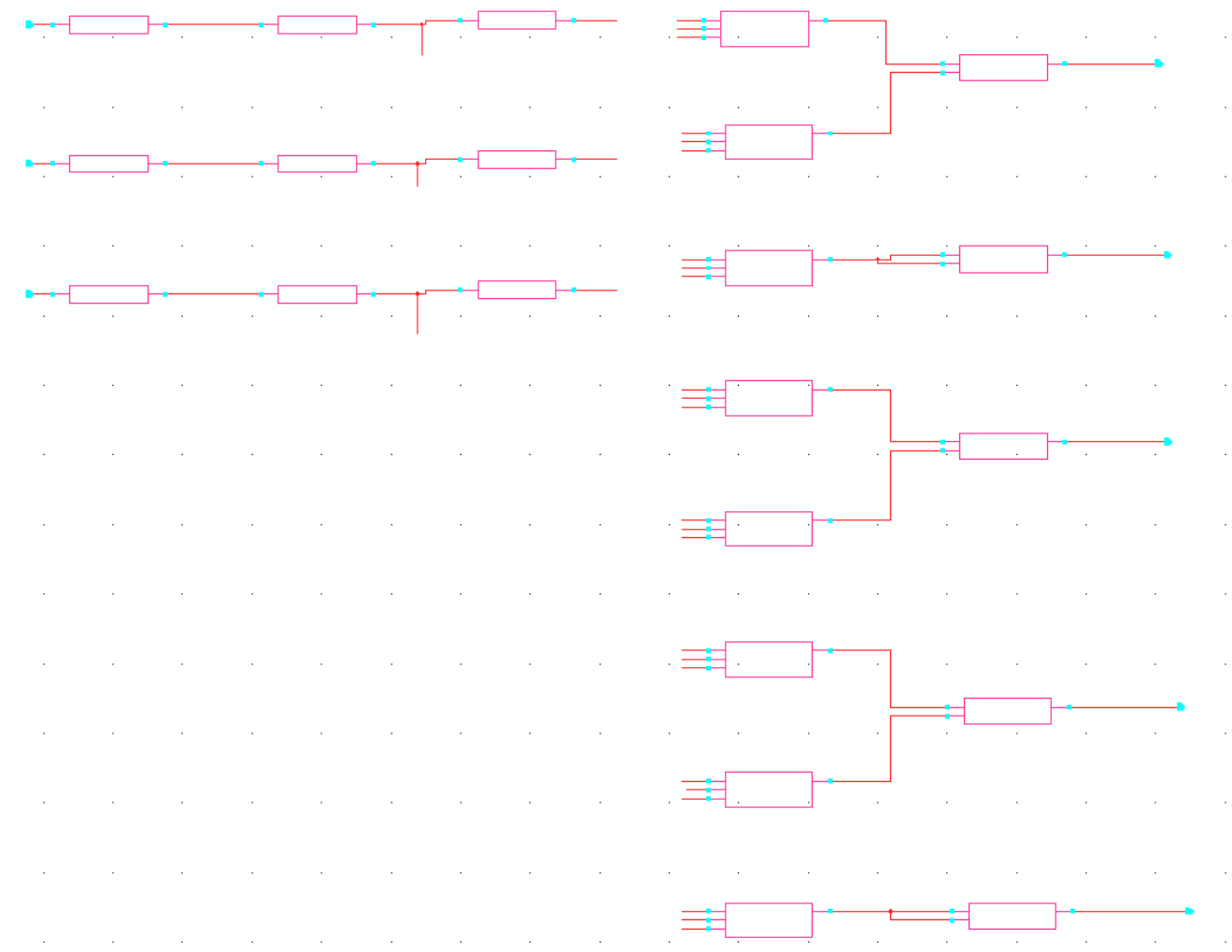
Figure 1

Design of each block:

1) Recoder [Library name: Newone, Cell name: booth]

Recoder block is designed using Table 1. The recoded bits are generated using the corresponding input logic. The complete schematic of the Recoder is as shown below.

The schematic of Recoder:

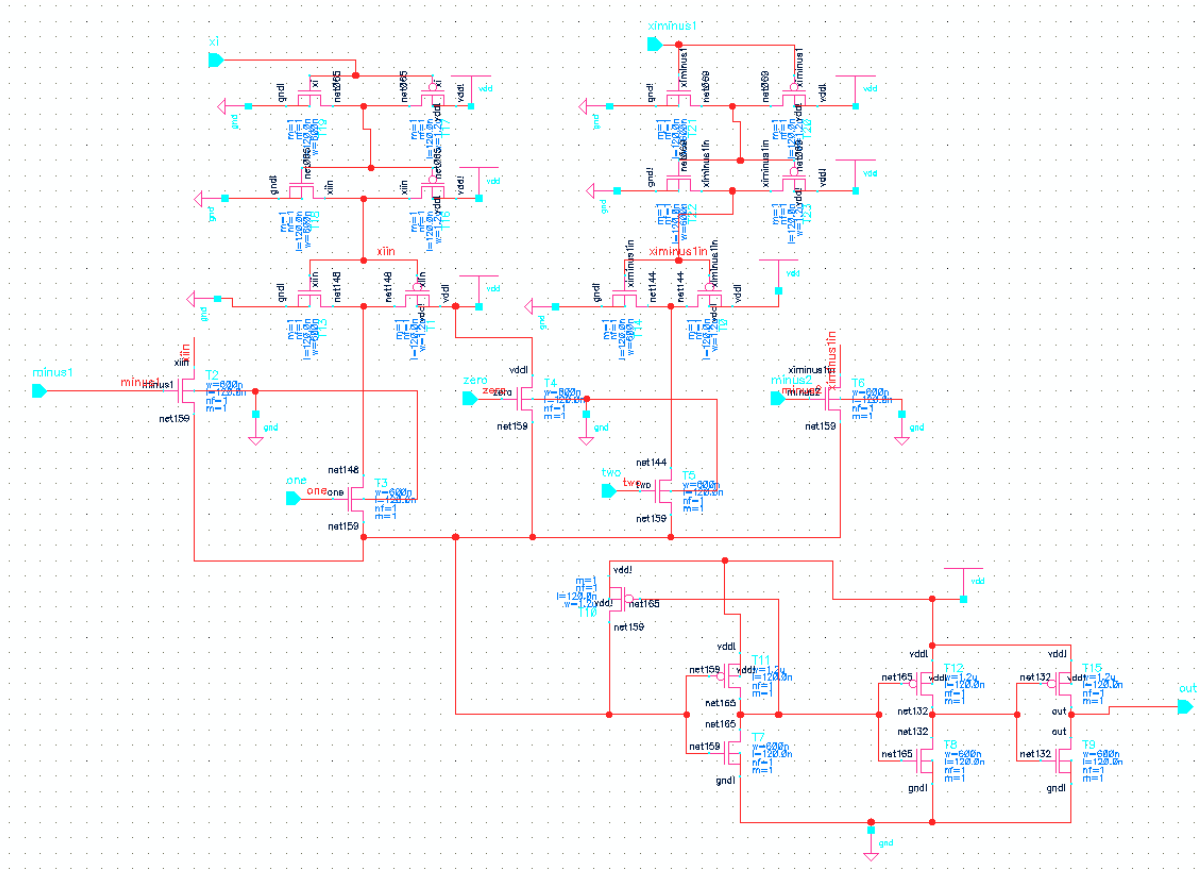


The symbol view of Recoder:

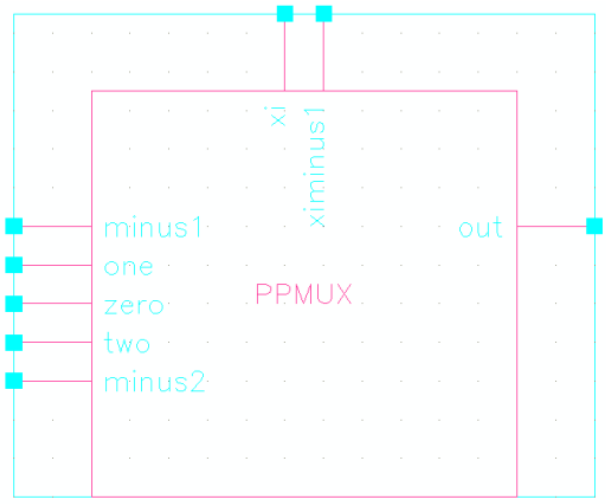
2) Partial Product Multiplexer (PPMUX) [Library name: Newone, Cell name: PPMUX]

Pass transistor logic is used to form the PPMUX. The recoded bits decide which bits of the multiplicand should be manipulated and then output the corresponding output bits for the partial product. The schematic of the PPMUX is as shown in the figure.

The schematic of the PPMUX:



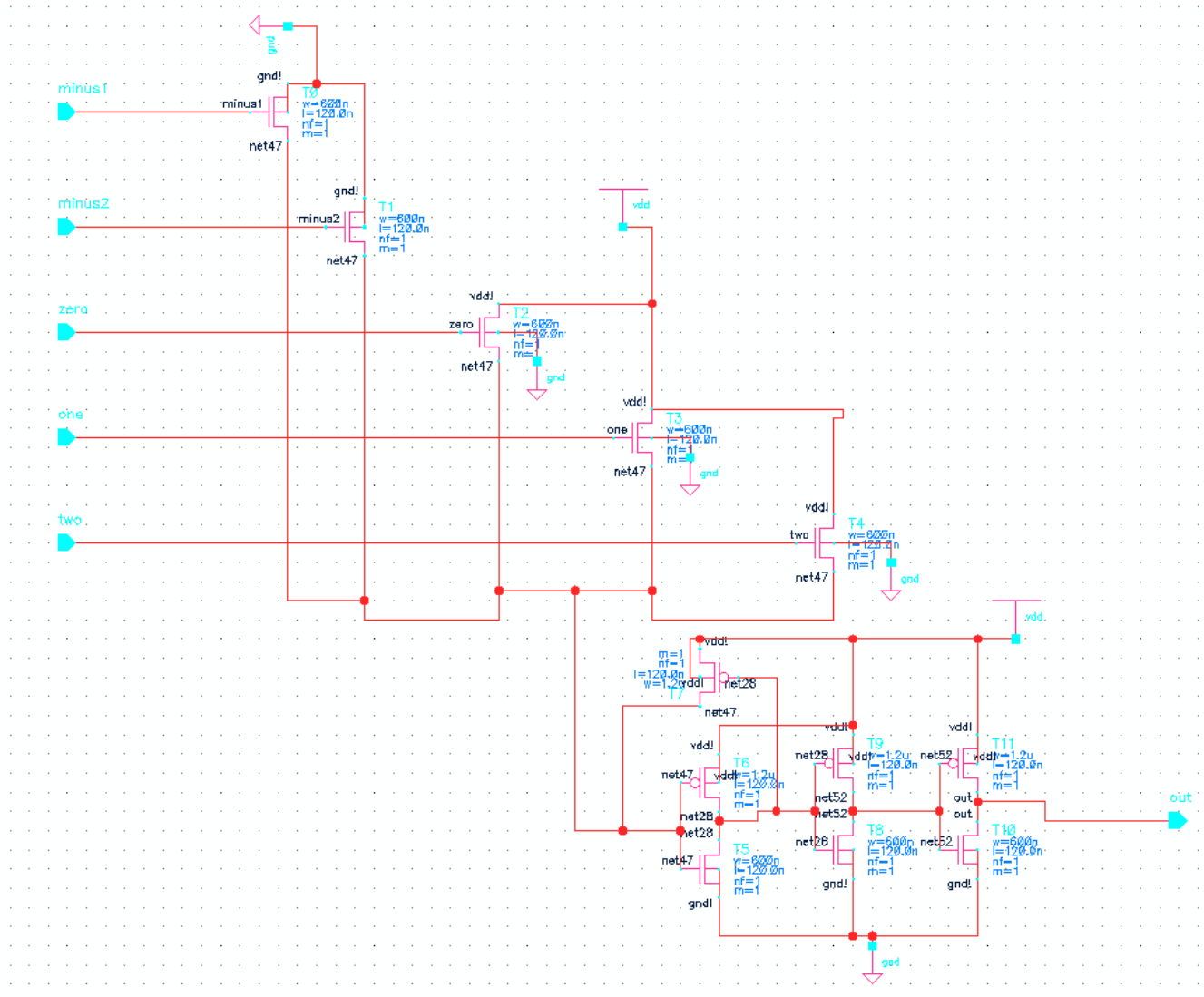
The symbol view of the PPMUX:



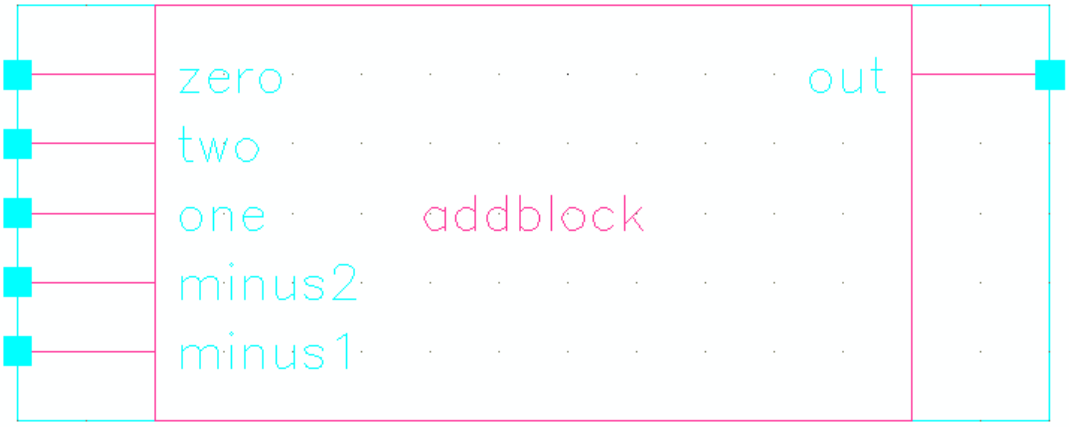
3) ADD block [Library name: Newone, Cell name: addblock]

If the recoded digit is negative, then the 2's complement of the multiplicand should be taken. 2’s complement is done by complementing all the bits and adding one to the LSB. The complement operation is done by the PPMUX. An ADD block is designed in order to add a one. The schematic of the ADD block is as shown below.

The schematic of the ADD block:



The symbol view of the ADD block:

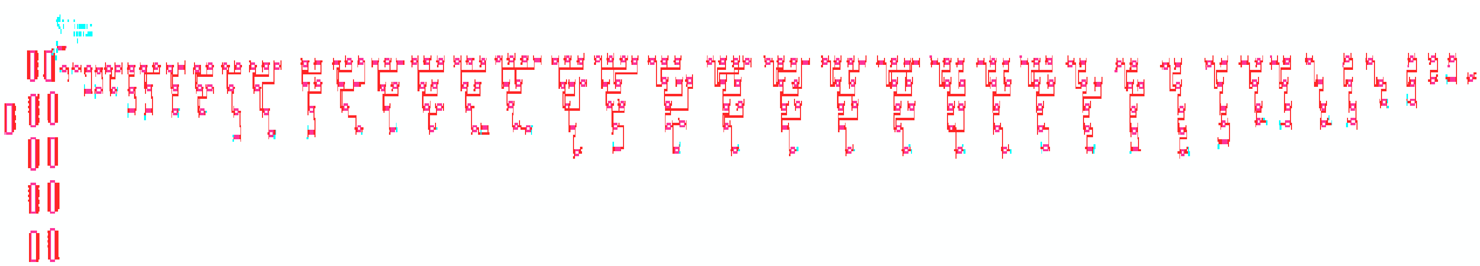


4) Compression Block [Library name: Newone, Cell name: compressor]

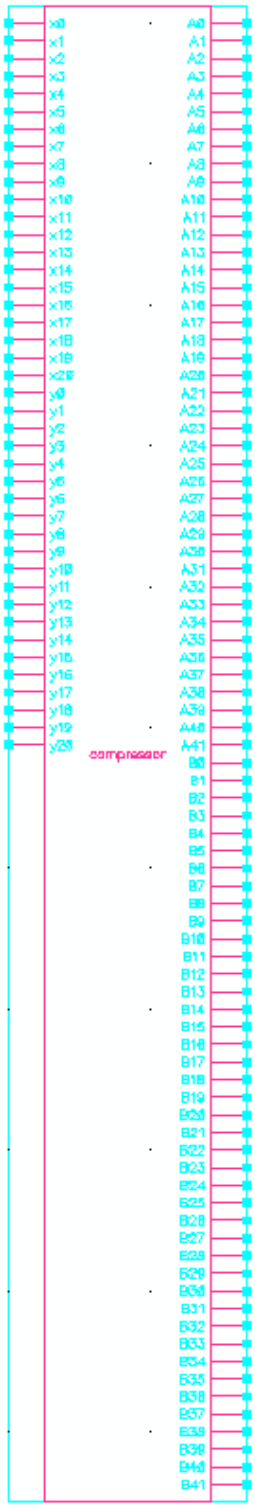
The generated partial products are then compressed using compressors at every column. All the columns are compressed into 2 columns as shown below in the figure.

Multiplicand																					0	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0
Multiplier																					0	y20	y19	y18	y17	y16	y15	y14	y13	y12	y11	y10	y9	y8	y7	y6	y5	y4	y3	y2	y1	y0
1st Partial Product																	x21bar	x21	x21	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0
																	x21bar	x21	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0
																																										1
2nd Partial Product																1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0		
																1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0		
																																								1		
3rd Partial Product														1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0				
														1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0				
																																						1				
4th Partial Product												1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0						
												1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0						
																																				1						
5th Partial Product										1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0								
										1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0								
																																			1								
6th Partial Product								1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0										
								1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0										
																																	1										
7th Partial Product						1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0												
						1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0												
																															1												
8th Partial Product				1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0														
				1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0														
																													1														
9th Partial Product		1	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0																
		1	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0																
																											1																
10th Partial Product	x21bar	x21	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0																		
	x21bar	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0																		
																								1																		
11th Partial Product	x21	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0																				
	x20	x19	x18	x17	x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1	x0	0																				
																					1																					
	A41	A40	A39	A38	A37	A36	A35	A34	A33	A32	A31	A30	A29	A28	A27	A26	A25	A24	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	B0
	B41	B40	B39	B38	B37	B36	B35	B34	B33	B32	B31	B30	B29	B28	B27	B26	B25	B24	B23	B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	A0
Output of Compressor block																																										

The schematic of the compressor block (along with the PPMUX):



The symbol view of the compressor block:



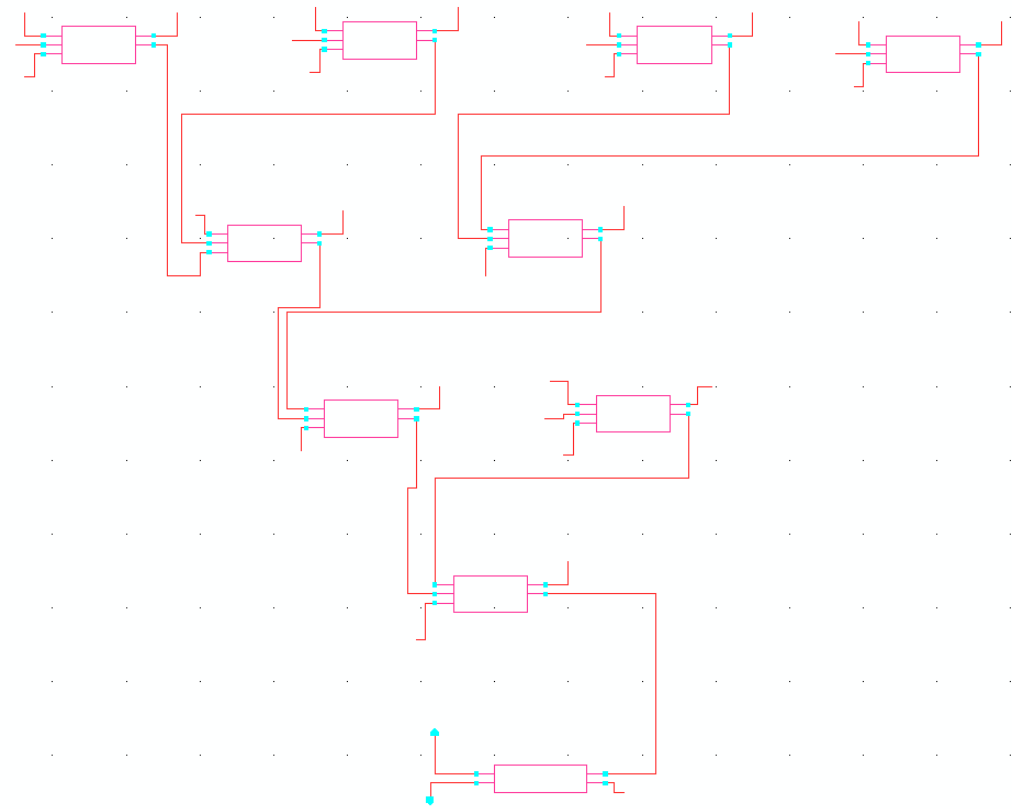
The 21st column has the maximum number of inputs which are 12. So the approximate number of full adders required is given by,

$$N = X_{in} + C_{in} - D$$

Where,
X_{in} = Number of inputs to be compressed = 12
C_{in} = Carries passed from previous column= n-3 =9
D = Number of drops = 1

So $N = X_{in} + C_{in} - D = 20$
Number of full adders = $\frac{N}{2}$ = 10 Full adders

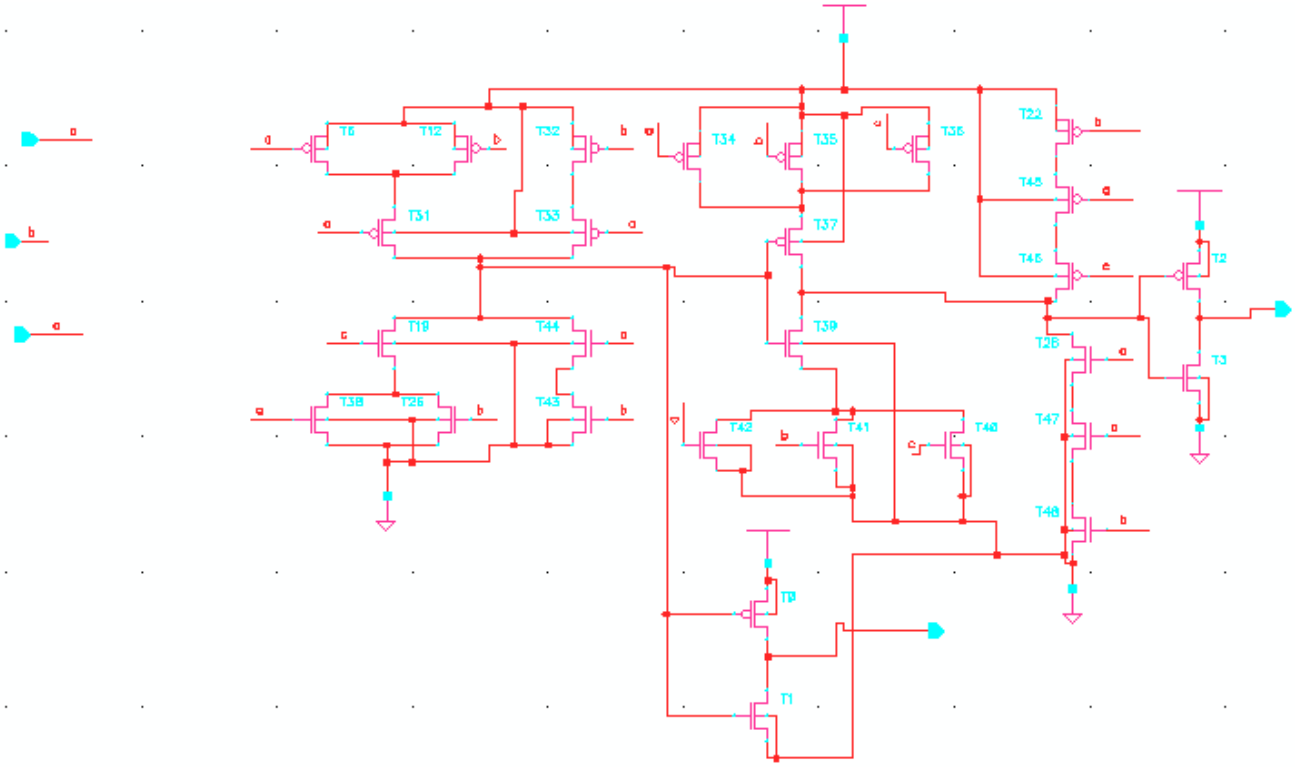
Zoom-in view of the compressor at 21st bit (12:2 compressor):



Full adder (Mirror adder) [Library name: pro-2 (pro#2d2), Cell name: mir-add (mir#2dadd)]

The full adder is designed using Mirror carry (MC) and Mirror Sum(MS)

The schematic of the full adder:

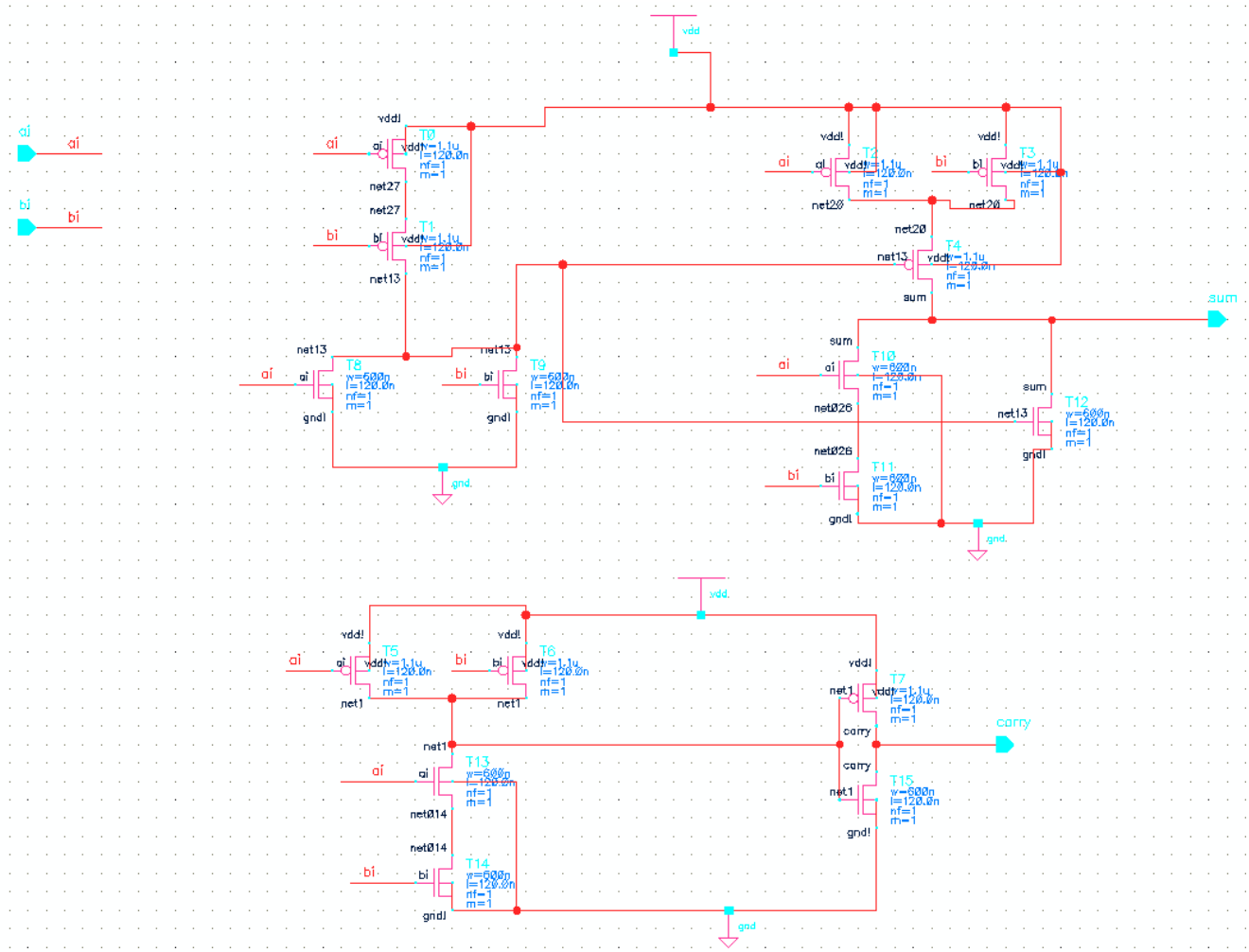


The symbol of the full adder (mirror adder):

Half adder [Library name: pro-2 (pro#2d2), Cell name: half_adder]

The half adder sum is an XOR of the two bits and the carry is an AND operation. The XOR is implemented using NOR2 + AOI21 combination and the AND is implemented as NAND2+INV.

The schematic of the half adder:

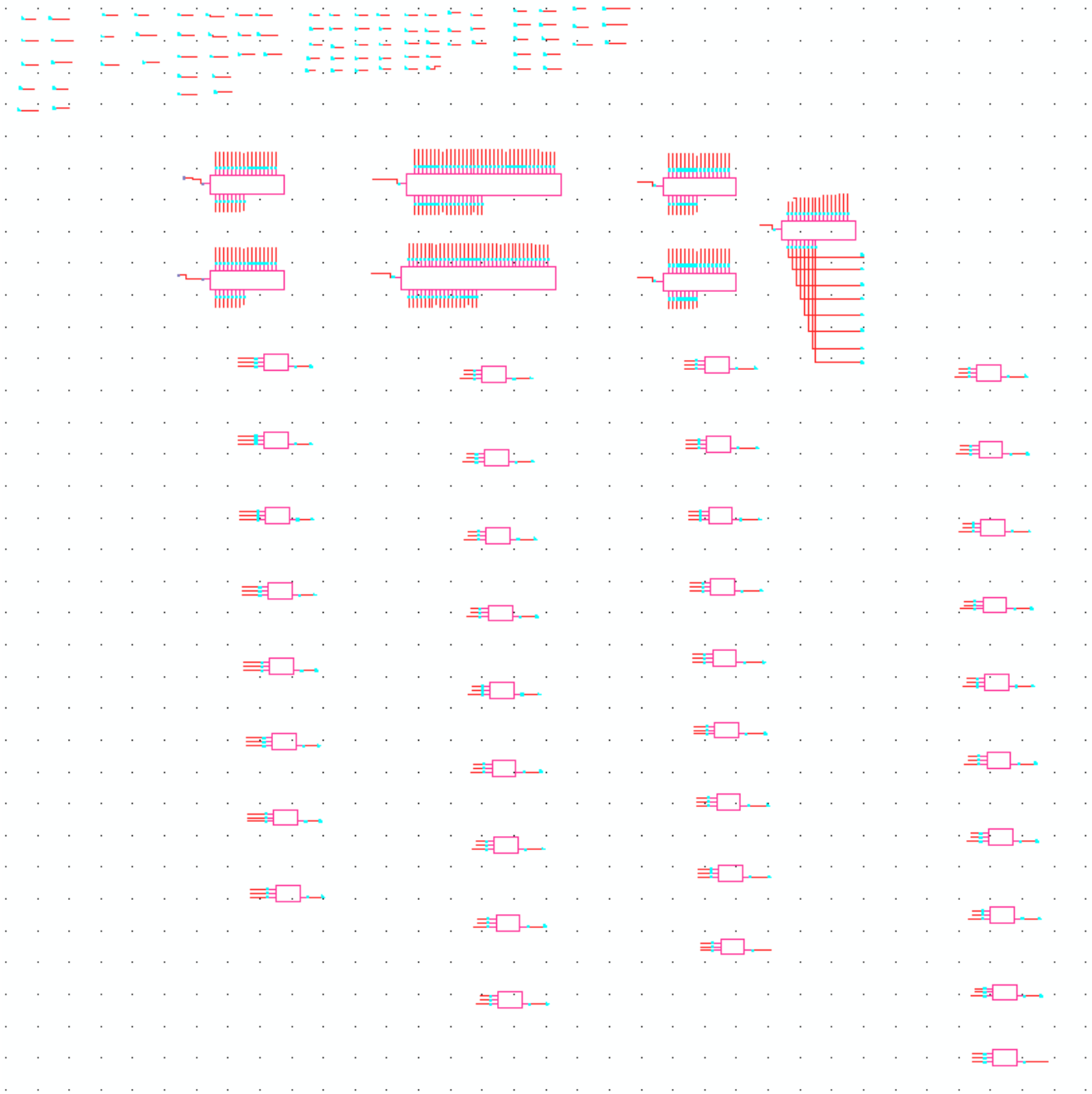


The symbol of the half adder:

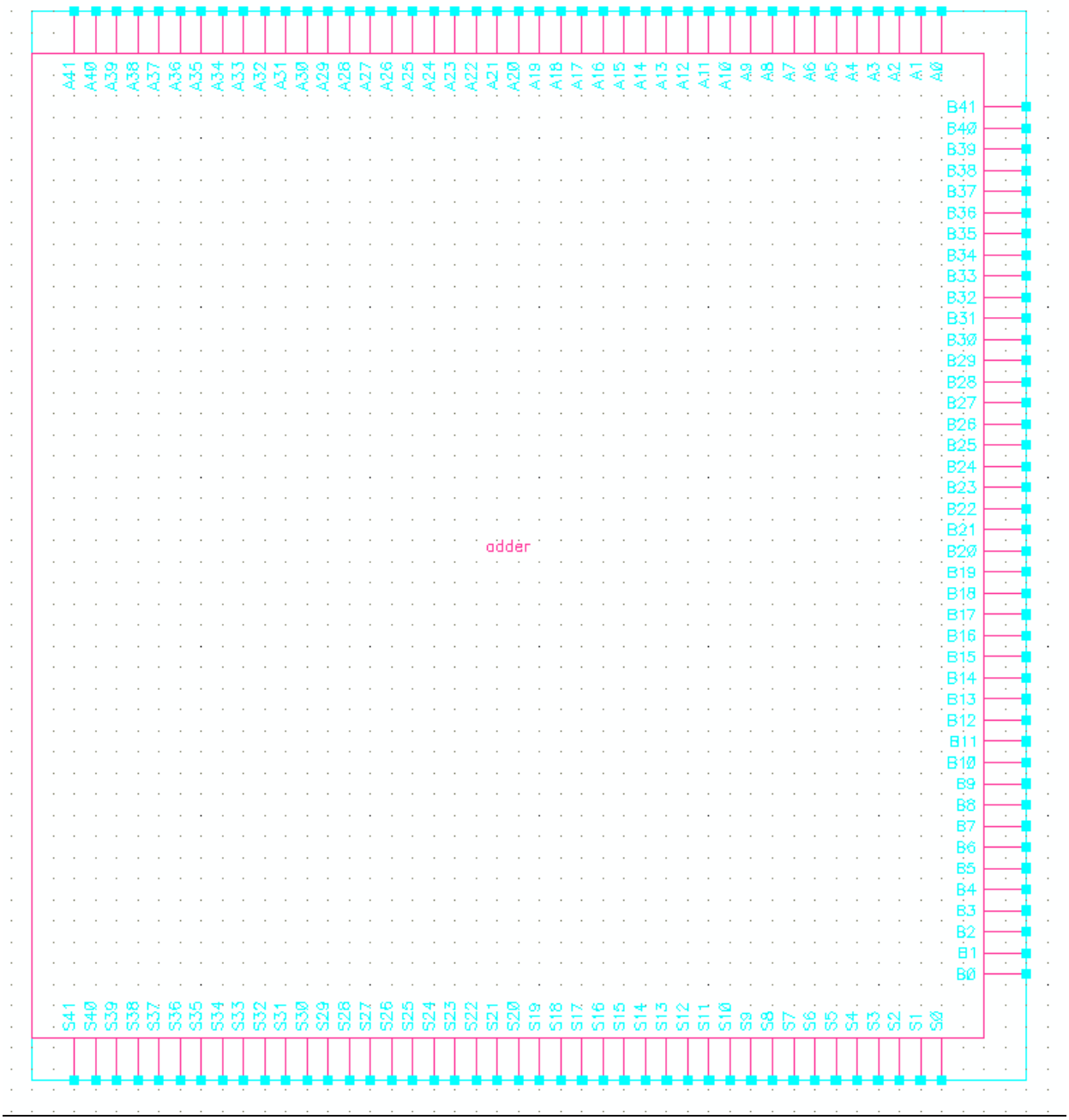
5) ADDER (42 bit) [Library name: Adders, Cell name: final_ad]

After Compressing into two rows, a combination of carry look ahead adder and carry select adder is used to obtain the sum of the two generated rows giving the final output.

The schematic of the 42 bit combined carry look ahead adder and carry select adder:



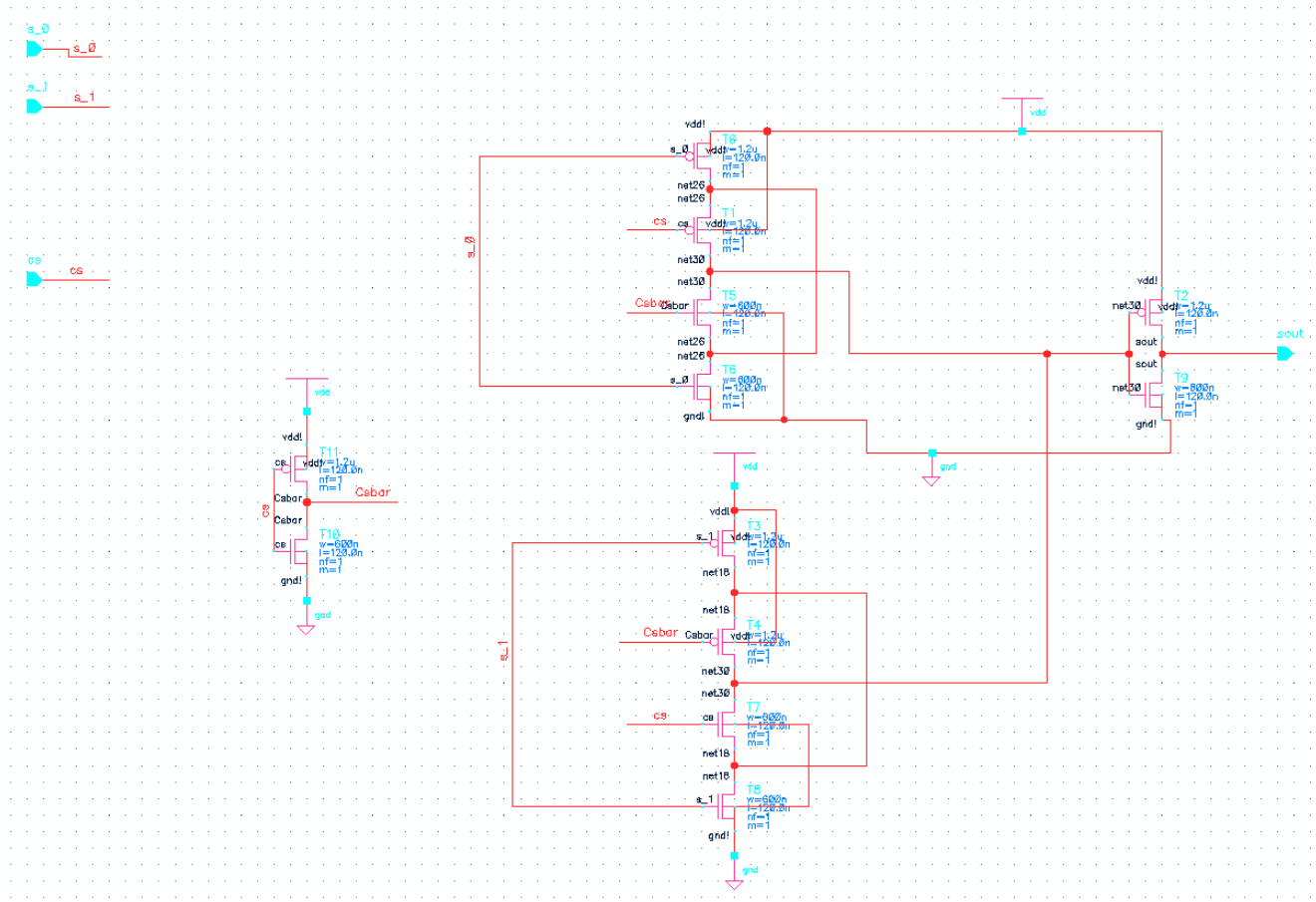
The symbol view of the adder:



Full (2:1) MUX [Library name: Adders, Cell name: full_2:1Mux (full_2#3a1Mux)]

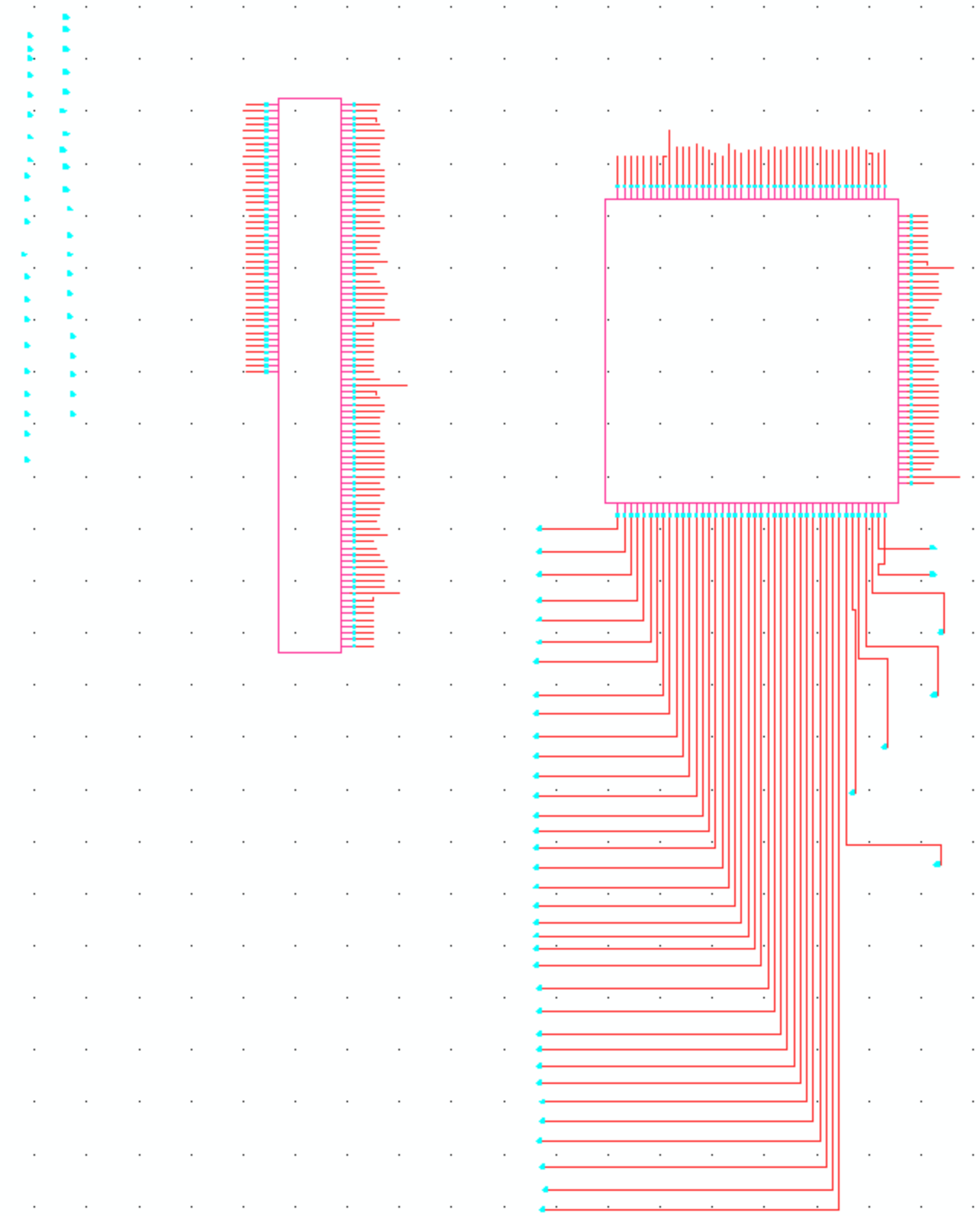
Full 2:1 MUX is used to select the sum and carry bits generated in the carry select adder.

The schematic of the full 2:1 MUX:



The symbol of the full 2:1 MUX:

Final representation of the 21 bit Multiplier [Library name: Newone, Cell name: multiplier]



Circuit Functionality:

The circuit functionality is verified for different input vectors and the output product is verified.

Example: I nput vector1:

X: 1 1111 1111 1111 1111 1111= 2097151(d)

Y: 1 1111 1111 1111 1111 1111=2097151(d)

Output:111111111111111111110000000000000000000001= 4398042316801(d)

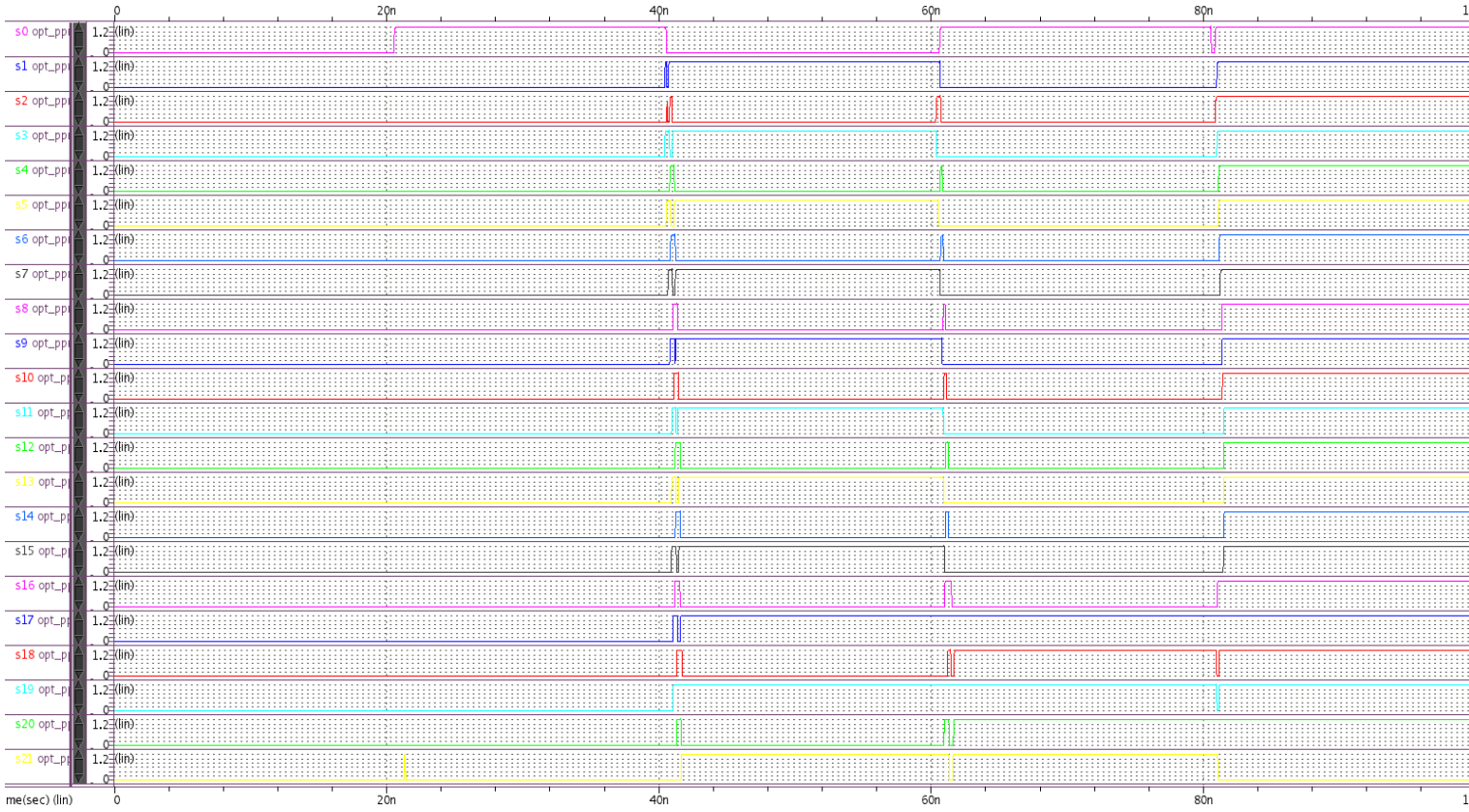
Input vectors2:

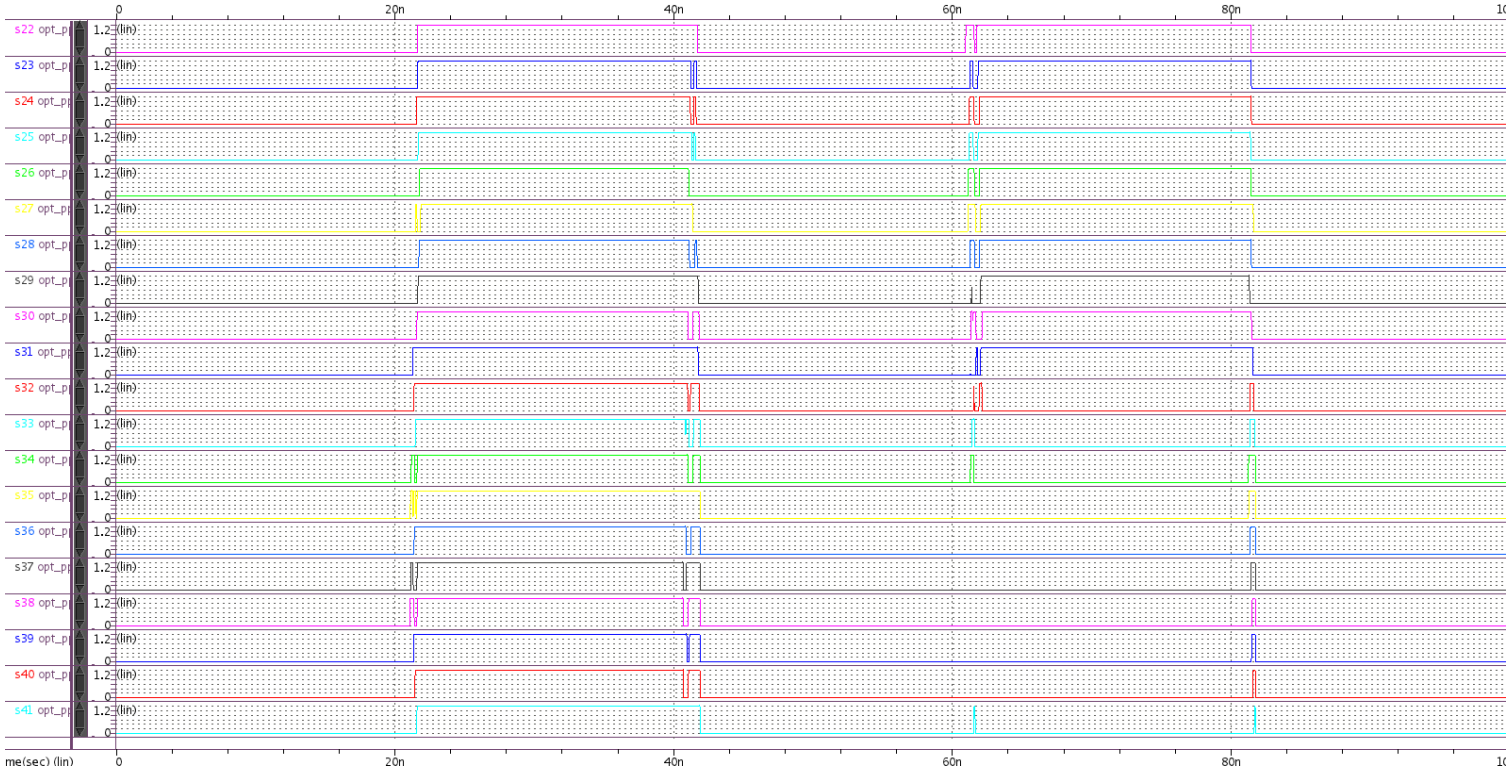
X:1 0101 0101 0101 0101 0101= 1398101(d)

Y: 0 0000 0000 0000 0000 0010= 2(d)

Output: 10101010101010101010=2796202(d)

The simulation waveform for different input vectors:




0
20n
40n
60n
80n
s22 opt_p
s23 opt_p
s24 opt_p
s25 opt_p
s26 opt_p
s27 opt_p
s28 opt_p
s29 opt_p
s30 opt_p
s31 opt_p
s32 opt_p
s33 opt_p
s34 opt_p
s35 opt_p
s36 opt_p
s37 opt_p
s38 opt_p
s39 opt_p
s40 opt_p
s41 opt_p
1.2 (lin)
0
me(sec) (lin)

Power and delay found from Primetime:

Delay: 2.15 ns

Power: 5.815 mW

Conclusion:

Successfully designed and implemented a 21b X 21b Multiplier using Cadence Virtuoso in 130nm IBM process technology. This gave me a very good understanding of hardware perspective of a multiplier. Verified its functionality for different input vectors in HSpice and calculated the delay and power of the design using Primetime.