Optimal 500MHz Arithmetic Logic Unit

An arithmetic logic unit (ALU) is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary numbers.

The goal of this project is to design a simple 4bits ALU with the minimum energy under the worst-case delay of 2 ns.The ALU should take a 10-bit instruction as input and perform addition, subtraction, multiplication or comparation then output the result in 8-bit.

Two input numbers are given in one instruction, A and B. Both are represented in 4-bit 2s complement.

1 4-bit ALU

1.1 Addition,Substraction,Comparation

Firstly, We use ripple carry mathod to realize 4bits addition.Substraction is realized by the adder.The difference betwween addition and substraction is that the second input of subtractor is the 2'complement of the second input of adder.Thus we can use adder to realize addition. Then we use substrator to realize comparation. We use first input to minus second input. When the MSB of output of the subtractor is 1 and the two input are equal , the output should be 0. When the MSB of output of the subtractor is 0 and the two input are ot equal, the output should be 1.

The main component of the three functions of 4bits adder. 4bits adder puls XOR, multiplexer and so on realize the three functions.

$$
\begin{array}{rcccccccc}
 & A3 & A3 & A3 & A3 & A3 & A2 & A1 & A0 \\
+ & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
+ & B3 & B3 & B3 & B3 & B3 & B2 & B1 & B0 \\
+ & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
- & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\hline
 & 0 & 0 & 0 & 0 & \bar{A}3 & A2 & A1 & A0 \\
+ & 0 & 0 & 0 & 0 & \bar{B}3 & B2 & B1 & B0 \\
+ & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
\hline
= & \bar{Y}4 & \bar{Y}4 & \bar{Y}4 & \bar{Y}4 & Y3 & Y2 & Y1 & Y0
\end{array}
$$

We use 4 fulladders and invertion property to realize the addition. The input of 4 fulladder are $\bar{A}3$ $\bar{B}3$ $A2$ $B2$ $A1$ $B1$ $A0$ $B0$. The output of the 4 fulladders are $Y4, Y3, Y2, Y1, Y0$. Then we use 4 inverters to get the 8 bits output $\bar{Y}4$ $\bar{Y}4$ $\bar{Y}4$ $\bar{Y}4$ $Y3$ $Y2$ $Y1$ $Y0$.

We use the fulladder in the following figure

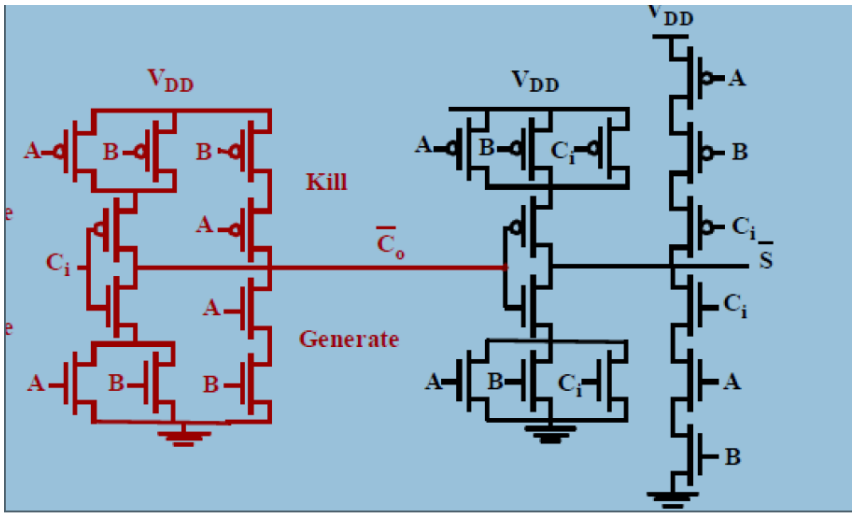

Figure 1: Fulladder

The 4bits adder is shown below.

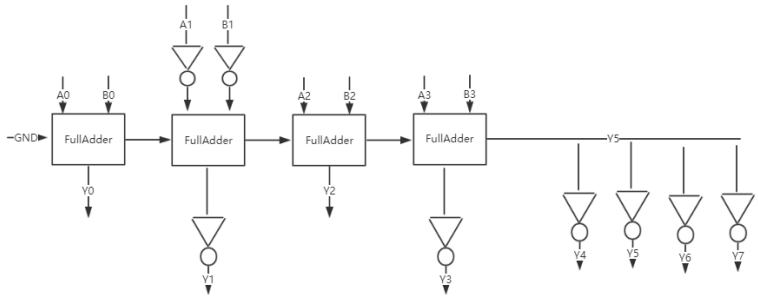

Figure 2: 4bits Adder

Then we use 4bits adder to realize substraction.We add XOR before the input and C_{in}. When S0=0, the input of 4bits adder is the original input and $C_{in} = 0$.The output of the adder is the addition result . When S0=1,the input of 4bits adder is the invertion of original input and $C_{in} = 1$.The output of the adder is the substraction result .

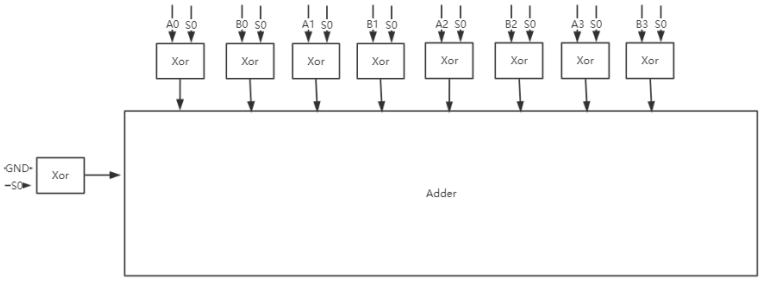

Figure 3: Substractor

However, the structure can not realize -1000(+8). There are two ways to solve the problems. One is using 5 bits adder, the other is dealing with this situation separately. We choose the latter because the

latter does not increase critical path delay. When the second input of substractor is 1000. The result will be $\bar{A}3A2A1A0$. We can use multuplexer to realize that.

When the MSB of output of the subtractor is 1 and the two input are equal , the output should be 0 and 1 when others. The special case is the two input are equal.The MSB of output of the subtractor is 0. We can judge whether the output are all zero or the two input are equal. We choose the latter becauese it can be determined when the input reach and does not increase critical path delay.

1.2 Multiplication

$$
\begin{array}{rcccccc}
 & & & A_3 & A_2 & A_1 & A_0 \\
 & & \times & B_3 & B_2 & B_1 & B_0 \\
\hline
 & & & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 \\
 & & A_3B_1 & A_2B_1 & A_1B_1 & A_0B_1 & \\
 & A_3B_2 & A_2B_2 & A_1B_2 & A_0B_2 & & \\
+ & A_3B_3 & A_2B_3 & A_1B_3 & A_0B_3 & & \\
\hline
\end{array}
$$

Figure 4: Multiplication

The input of multiplier are positive so we only need 3 bits multiplier.

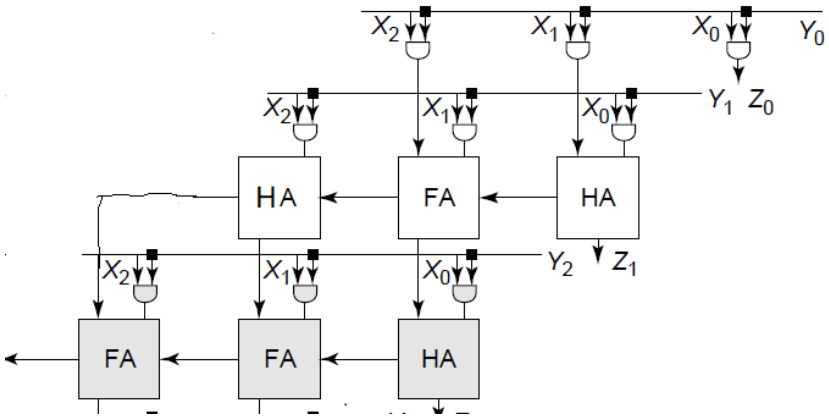

Figure 5: Multiplier

1.3 Result

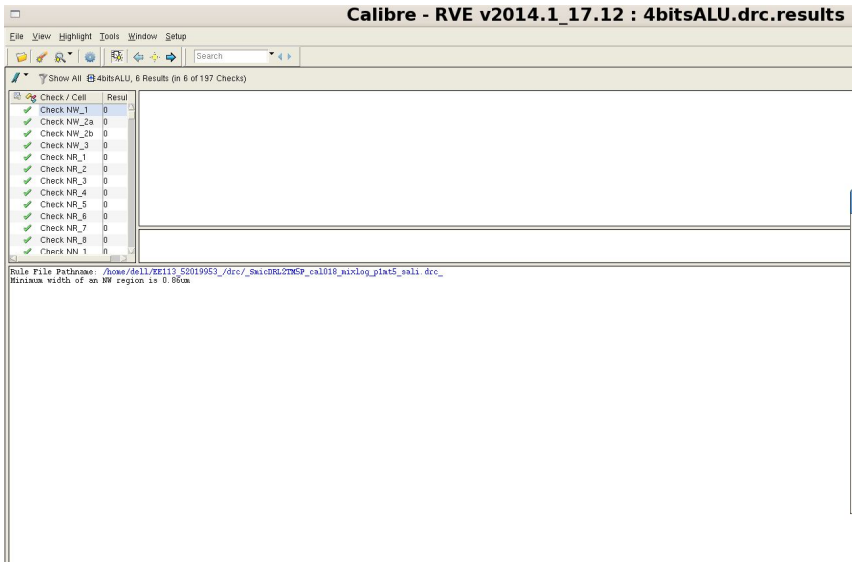

Figure 6: DRC

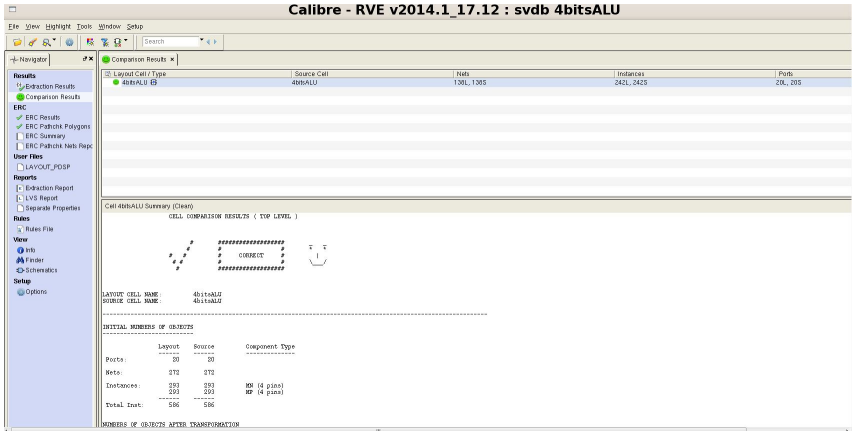

Figure 7: LVS

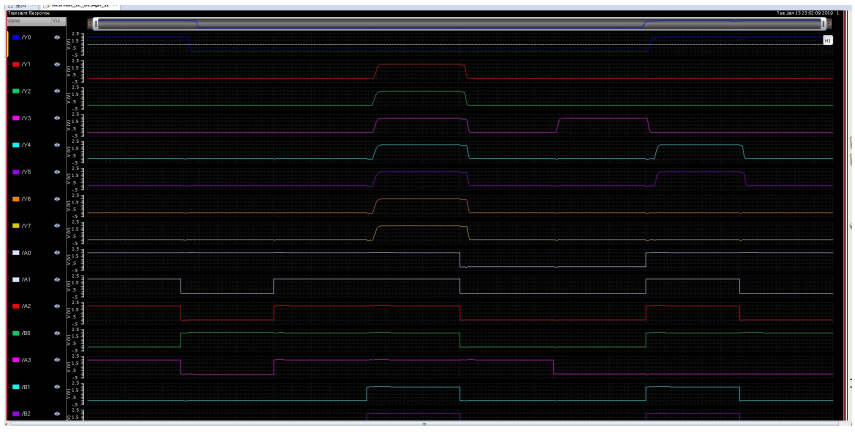

Figure 8: post-simulation

2 8-bit ALU

Below I will introduce the design method of the 8-bit ALU. There are three ways to design an 8-bit ALU, which we will cover separately and compare the advantages and disadvantages of each. The first two are common unoptimized methods. Finally, we propose a structural optimization method for the ALU, which can significantly reduce the area and power consumption without significantly reducing the delay.

2.1 Method 1

The first method is a very straightforward method. We design four basic modules, namely the addition module, the subtraction module, the comparison module, and the absolute value comparison module. When there is an input signal input, all the inputs are simultaneously transmitted to the four modules, and the four modules are simultaneously operated, and finally the final result is output through the 4-1MUX.

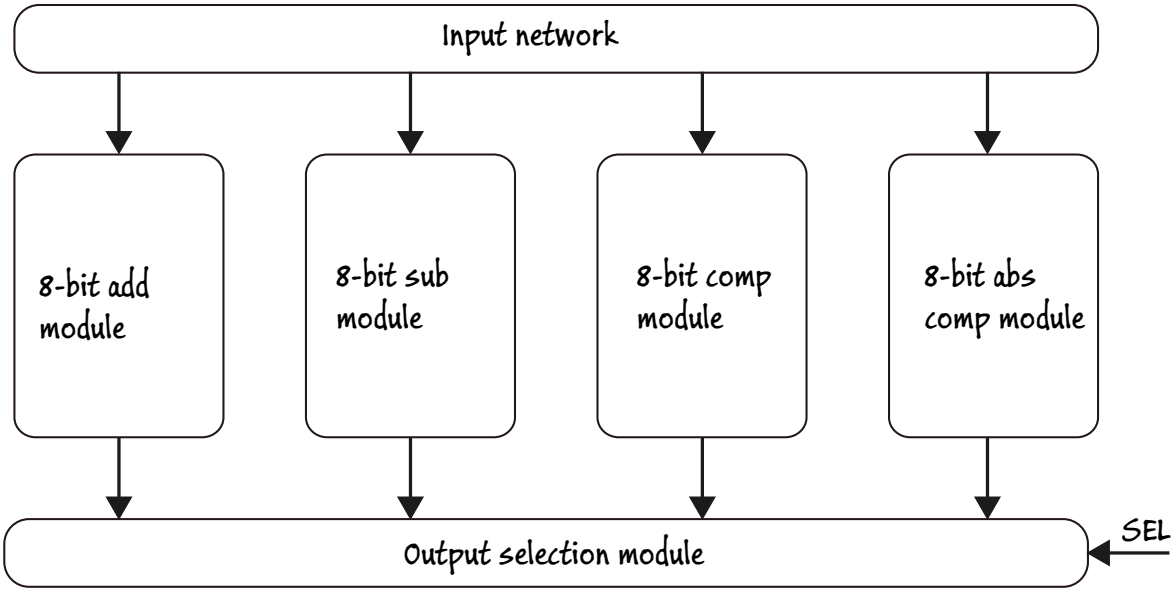

Figure 9: ALU diagram of method 1(straitforward method)

The advantage of this method is that it is fast and the design method is simple, and different modules can be optimized separately. However, this design method also has great shortcomings. For the upper level of the ALU, the input signal needs to be transmitted to the four calculation modules at the same time, so the fanout of the upper level will become very large, although we have not in this project consider the impact of the ALU on the upper level, but this design is obviously not good in practical use. The second drawback of this design is that the required circuit area and power consumption are too large, every time we only want a result of four functions, but we have to run four modules at the same time, which is obviously an incorrect design for ALUs that often perform computational functions.

2.2 Method 2

For method two, we still use four calculation modules, and method two is the same as method one on the area. However, Method 2 adds a PMOS to the VDD of each module to control the switching of the module. Therefore, in terms of energy, Method 2 has a significant improvement over Method 1.

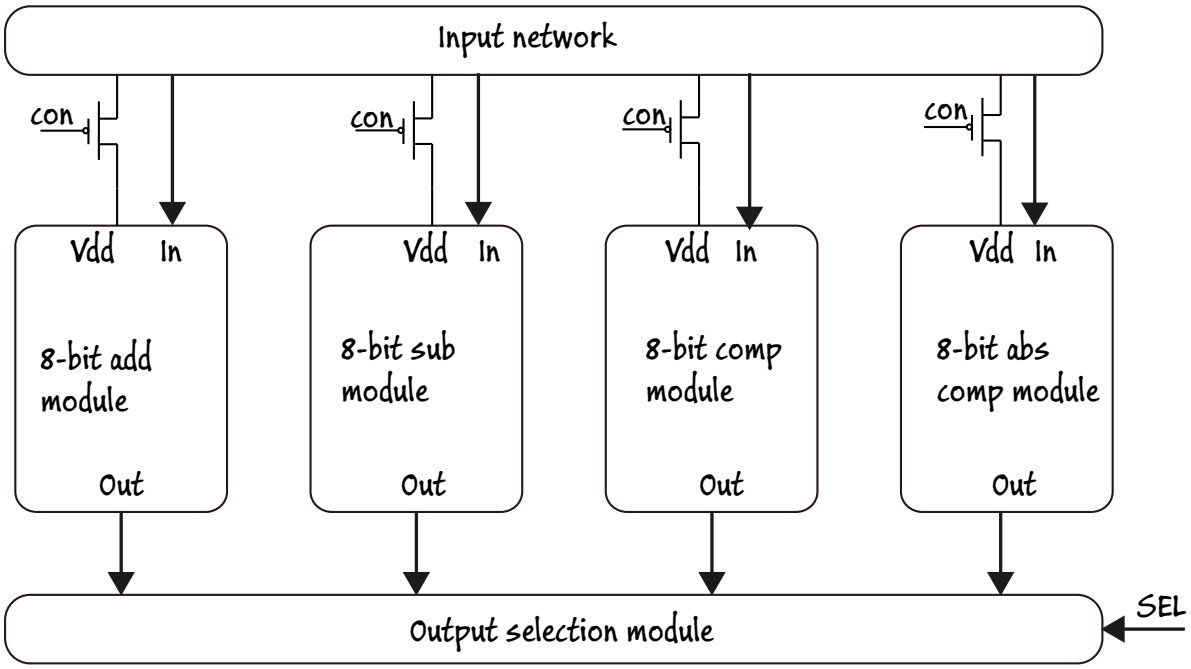

Figure 10: ALU diagram of method 2(straitforward method with energy optimization)

2.3 Method 3

The third method is the method we used. We redesigned the structure of the ALU to find a trade-off between speed and power consumption. We first combine the addition and subtraction module, this is very simple.By observing, we first find that the subtraction and addition modules can share an adder, but we need to perform the inverse plus one when doing the subtraction. This is done by the XOR array and the control module. Adding 0/1 directly to the carry of the 8-bit adder realizes the operation of adding one.

Then we figured out how to redesign the comparator and the absolute comparator. We found that for the comparison operation, we can still use the subtraction method. For the two operands A, B, we can use A-B and observe the sign bit. For absolute comparisons, we still use the subtraction operation.First we need to change the value of input A,B according the sign of each operators.Then subtract the operands after the transformation, and observe the carry information to get the final result. Here we need to judge whether the result of the subtraction is 0 to correct the result.

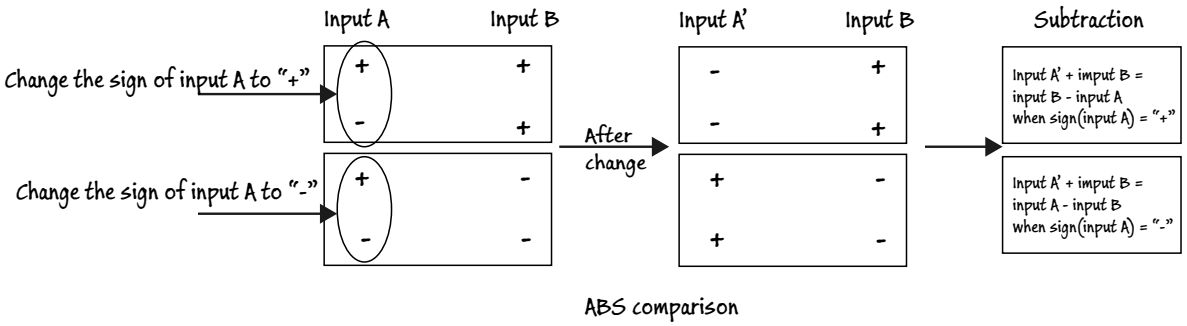

Figure 11: ABS comparison principle

In the end, we turn the comparison operation and the absolute value comparison operation into subtraction operations. We can combine the operations of subtraction, addition, comparison and absolute value comparison with some control signals.

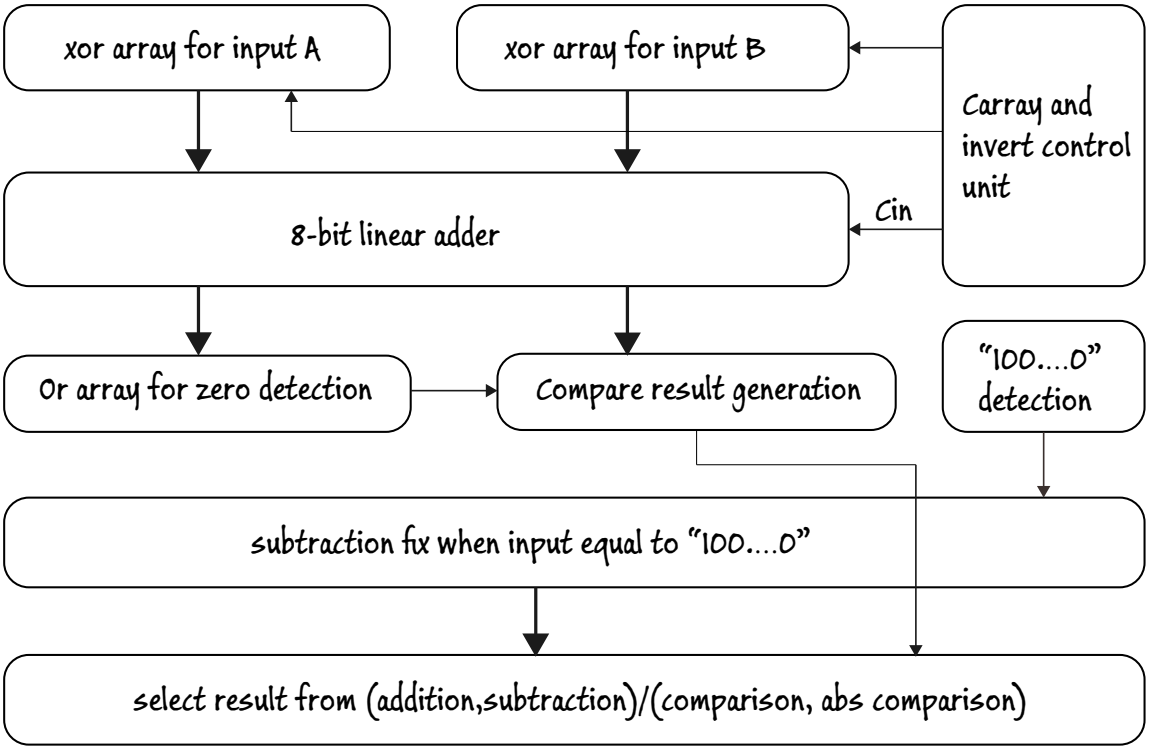

Figure 12: ALU diagram of method 2(Area, power and delay optimization

In this way, we only use an 8-bit adder, and with some control unit, 0 detector, we realize, add, subtract, compare, and compare absolute values.Here we also optimized the 8-bit adder using the Linear-ripple structure. The delay of the final 8-bit adder is only about 4 full-adder delays.Through this structure optimization and change our area is almost 1/4 of the above two methods, and the speed of our ALU is not significantly reduced, our method of power consumption is higher than method two, but compared with method one Our power consumption has dropped significantly.

	Area	Delay	Power	Overall performance
Method 1	5	1	5	11
Method 2	5	1	1	7
Method 3	1.5	1.5	1.5	4.5
Best	Method 3	Method 1/2	Method 2	Method 3

Table 1: Comparison of three method, The smaller the number, the higher the performance.

3 8-bit Sign Multiplier(Baugh-wooley + wallace tree method)

3.1 Baugh-wooley algorithm

Multiplication of twos complement numbers at first might seem more difficult because some partial products are negative and must be subtracted. Recall that the most significant bit of a twos complement number has a negative weight. Hence, the product is:

$$\begin{aligned} P &= \left(-y_{M-1}2^{M-1} + \sum_{j=0}^{M-2} y_j 2^j\right)\left(-x_{n-1}2^{N-1} + \sum_{i=0}^{N-2} x_i 2^i\right) \\ &= \sum_{i=0}^{N-2}\sum_{j=0}^{M-2} x_i y_j 2^{i+j} + x_{N-1}y_{M-1}2^{M+N-2} - \left(\sum_{i=0}^{N-2} x_i y_{M-1} 2^{i+M-1} + \sum_{j=0}^{M-2} x_{N-1} y_j 2^{j+N-1}\right) \end{aligned} \tag{1}$$

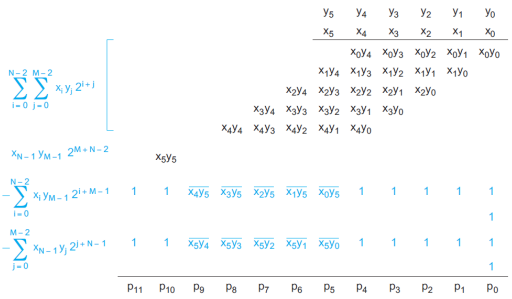

Figure 13: Principle of 4 bit baugh-wooley

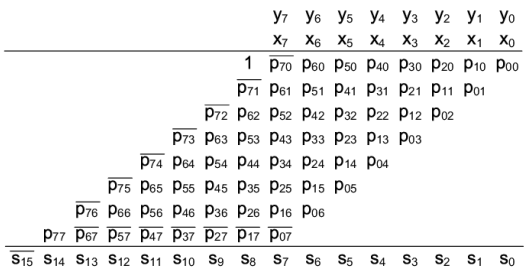

Figure 14: Result of 8 bit baugh-wooley

3.2 Wallace method

A Wallace treeis a very useful hardware function that is used in digital circuit for multiplies two integers. In Wallace method the multiplication of two numbers is done by reducing the partial product matrix into a two-row matrix by a half adder, full adder, carry-save adder & these two rows are added utilizing a fast carry propagate adder to produce the output product. In this Wallace tree method we used half adder for summation of 2 bit and used full adder for summation of 3 bit. For multiplicands of higher than 8-bits this advantage is more beneficial, because the addition of partial products is low in Wallace tree and hence increases speed. Here each bit of each partial product in every column is added together by a set of counters used in parallel so that no carry is propagated further. Then this matrix is reduced by another set of counters until a two-row matrix generates.

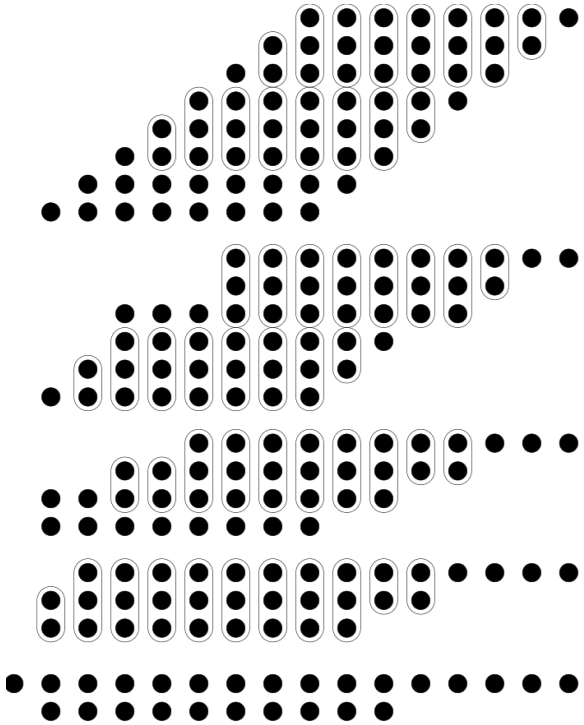

Figure 15: Wallace tree method

4 Overall block diagram

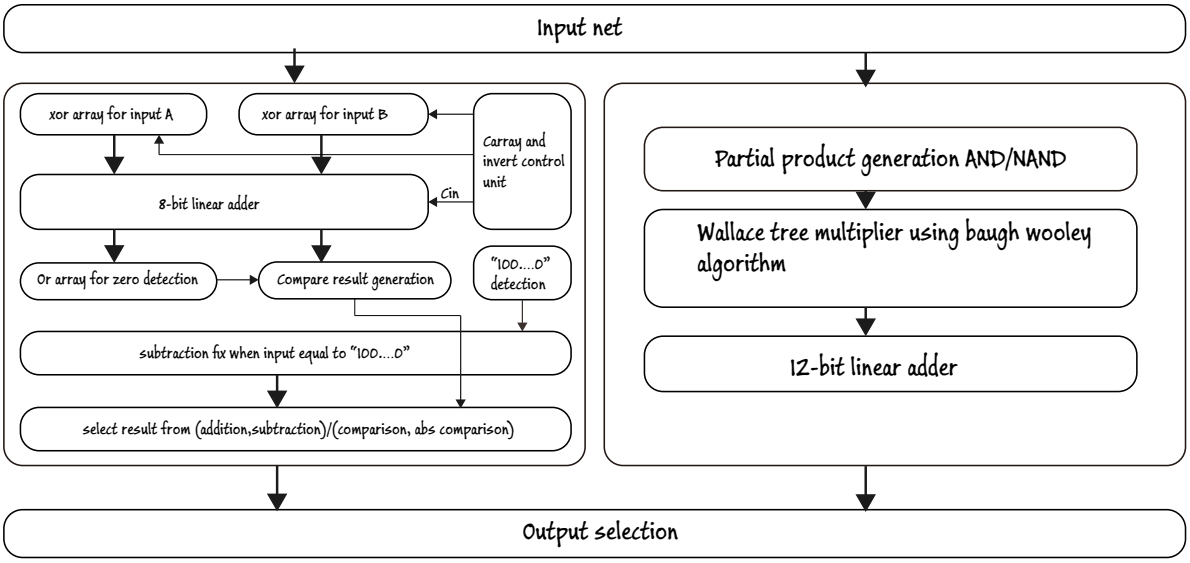

Figure 16: Overall block diagram

5 Layout and simulation

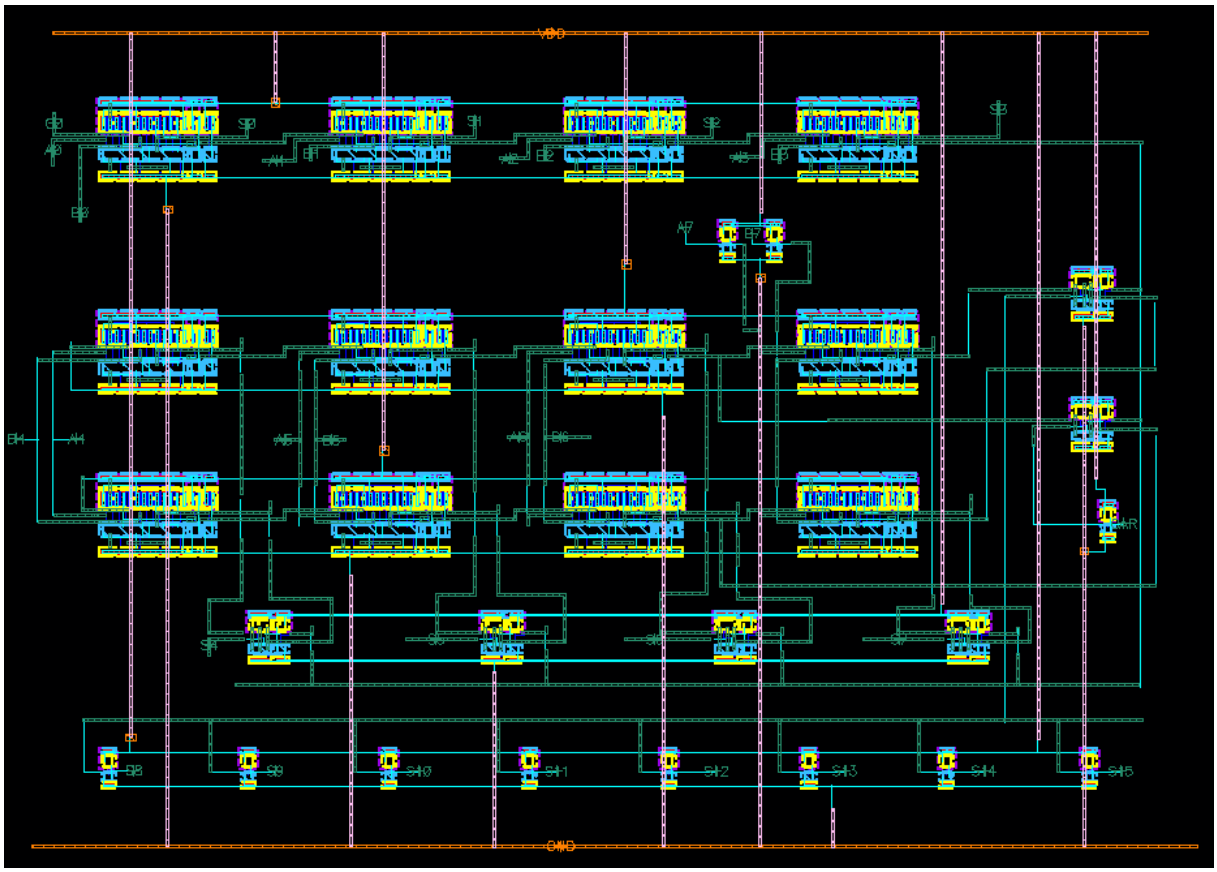

Figure 17: Linear adder (8-bit) layout

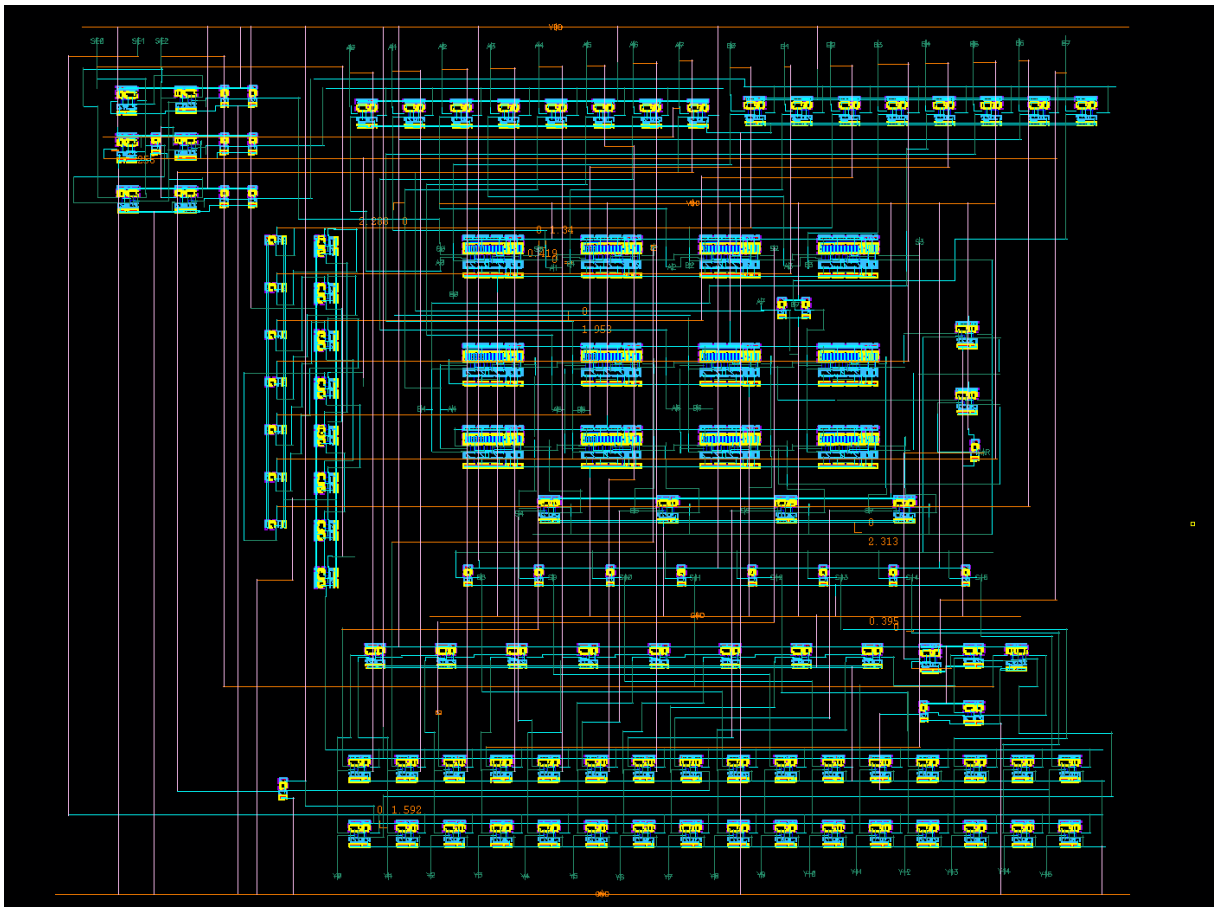

Figure 18: 8-bit ALU add,sub,comp,abs-comp, layout

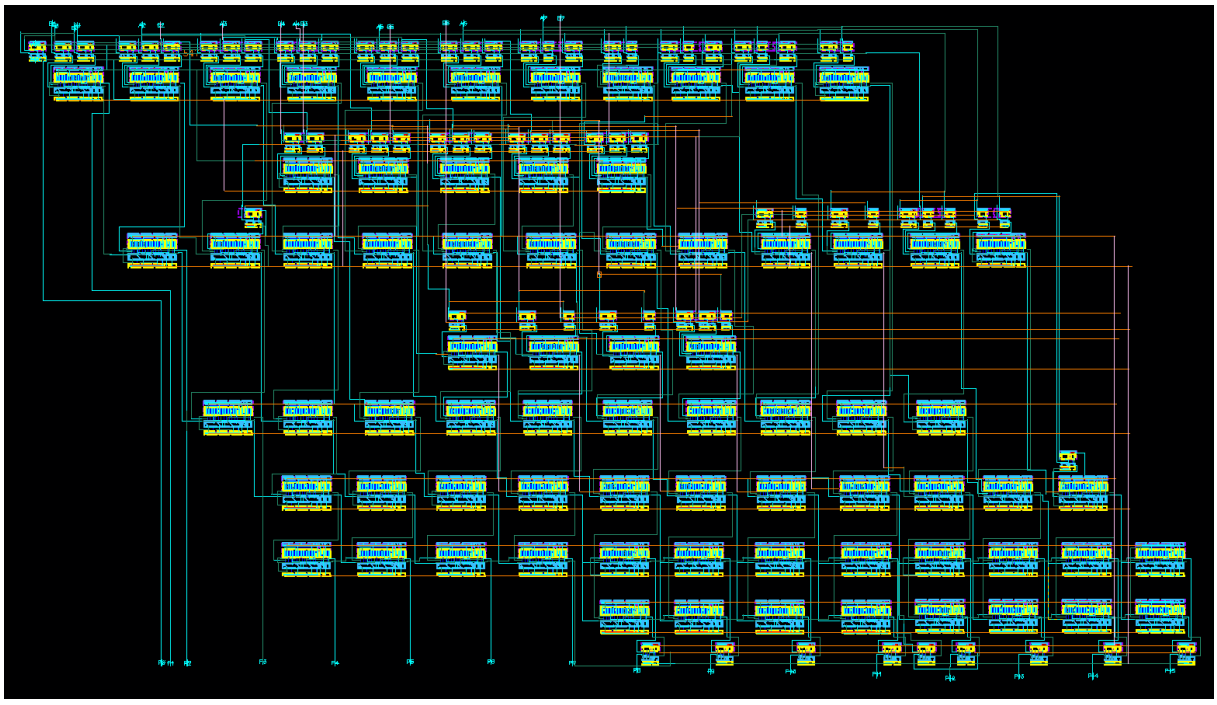

Figure 19: Baugh-wooley + wallace tree sign multiplier

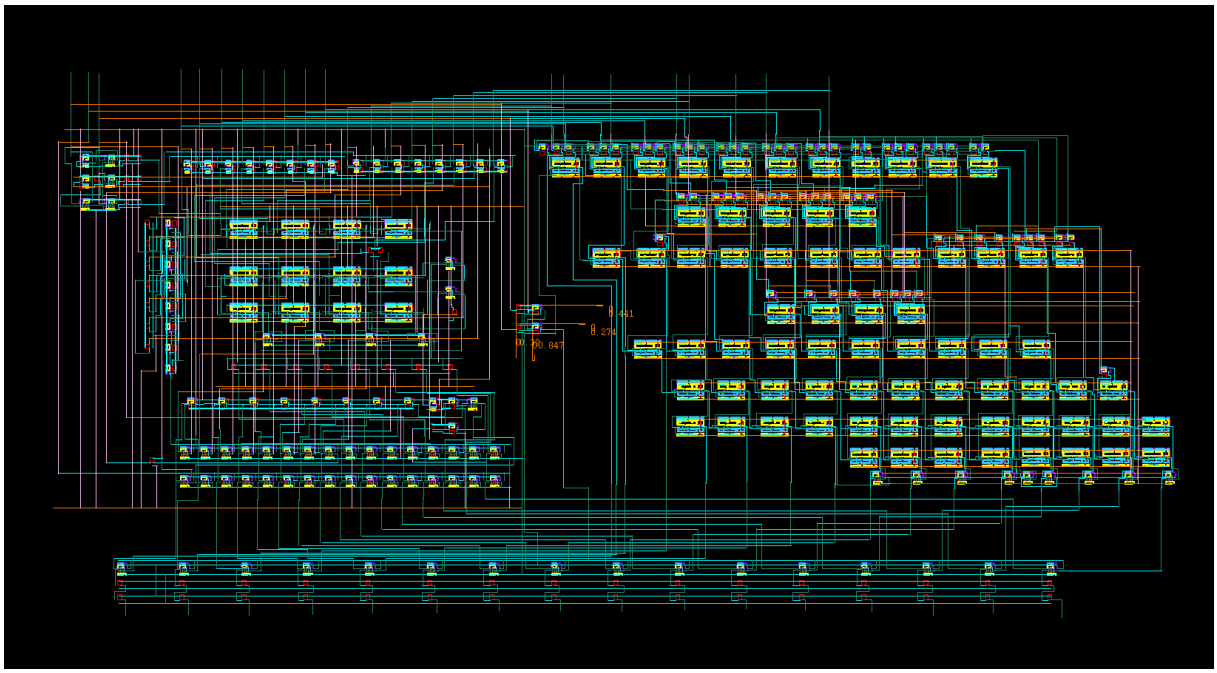

Figure 20: Overall layout

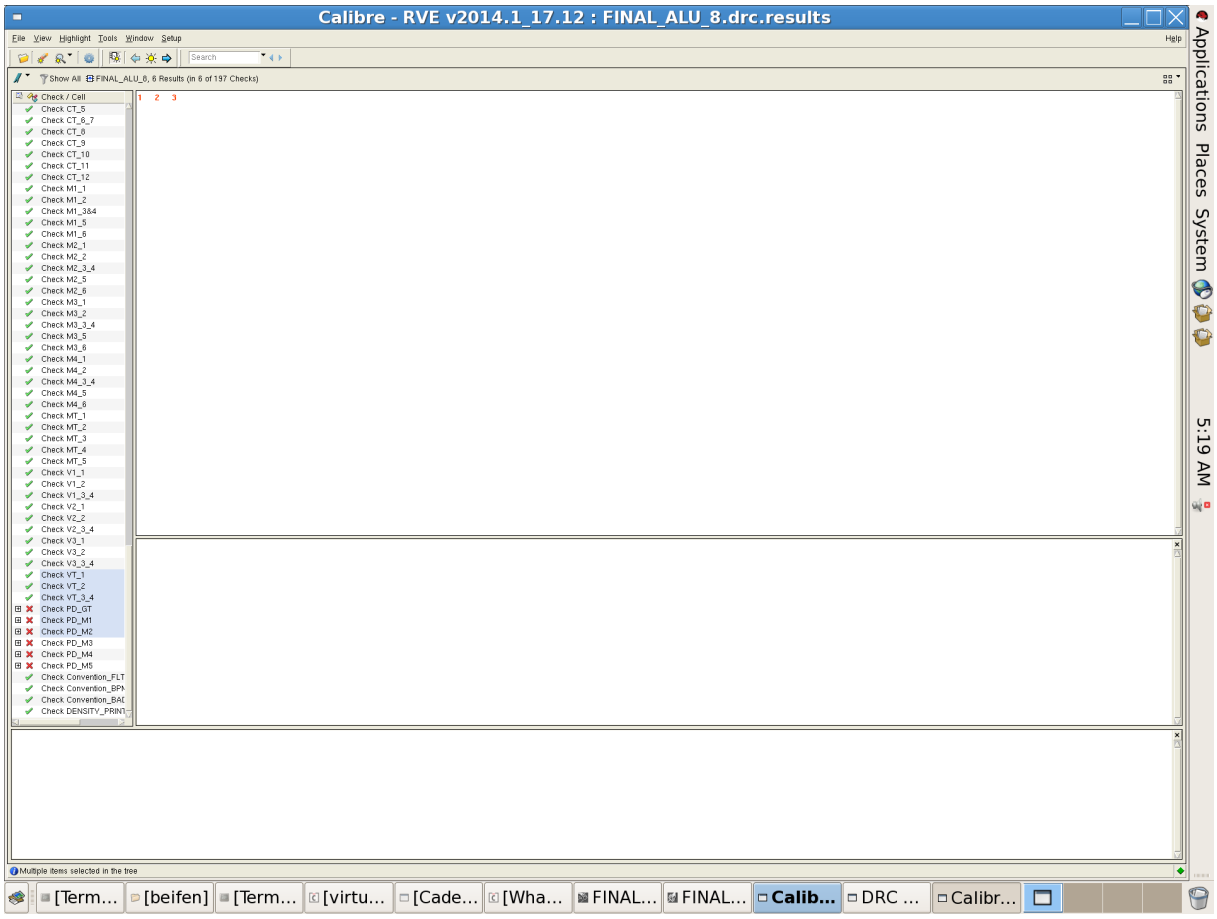

Figure 21: DRC pass

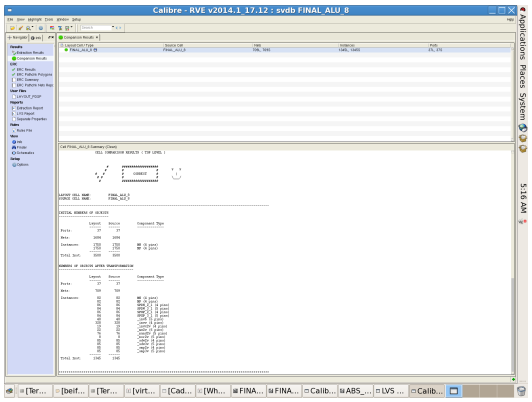

Figure 22: LVS pass

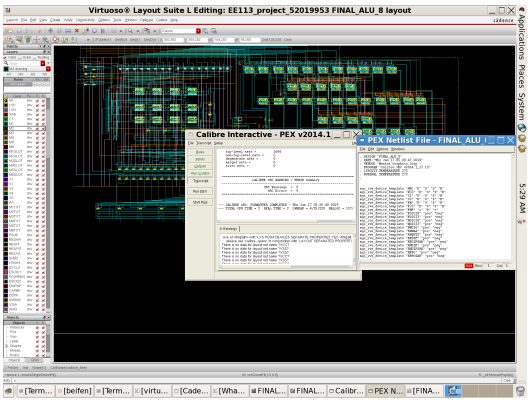

Figure 23: Pex pass

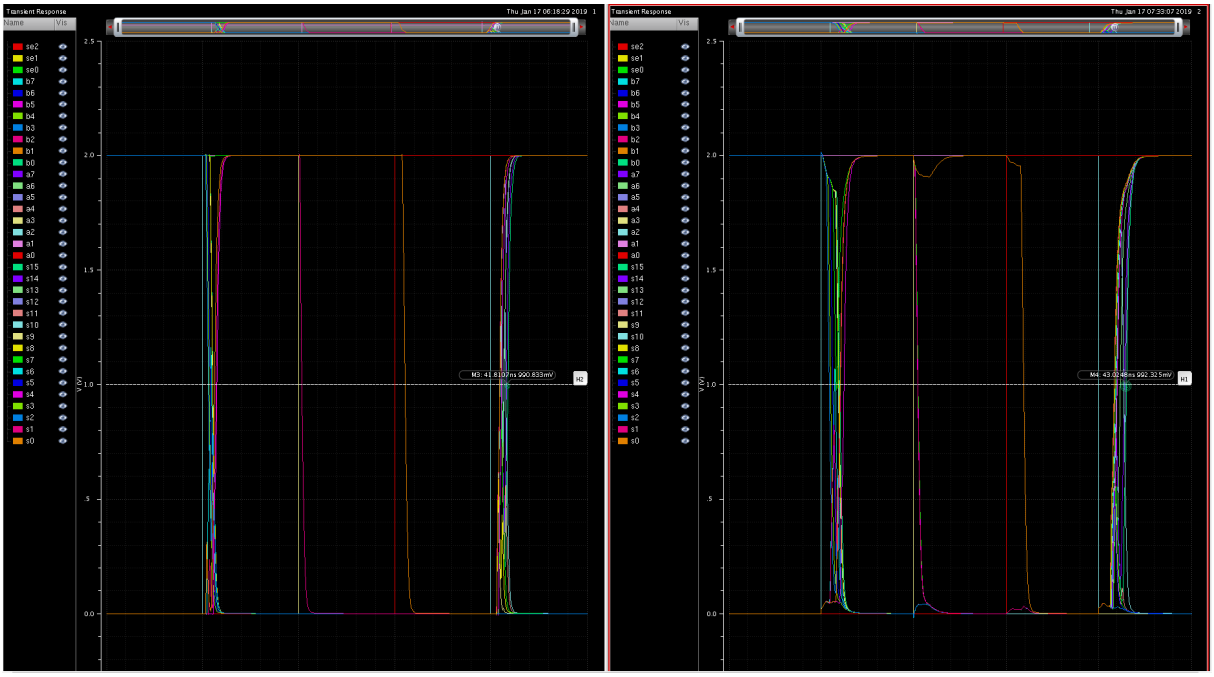

Figure 24: 8-bit ALU delay