ELEC4410 Group Project Report

1) Group Information

Group Representative: Ching Wai Kin 20546660

Group Members: Chiu Siu Ki 20519253

Hau, Kai Hong 20428090

Ip, Ming Chiu 20520537

2) Standard Cell Library

Inverters are widely used in our 16-bits carry-skip adder, especially in full adders. A NAND gate and an XOR gate are used to build a half adder, which is the first input stage of our carry-skip adder. NOR gates with different numbers of inputs are used for the implementation of the carry-skip function.

The delay depending on the extrinsic load capacitance in the transistor is found. There is a directly proportional relationship between the delay and the external load capacitance. The result shown in table 1 reveals that a CMOS inverter has a great delay which increases with the extrinsic load capacitance. Therefore, instead of an inverter, more complex transistors with more than one input are fabricated in the design. For instance, the gates used for the skip-carry function are initially more than one input OR-AND logic gates. However, this kind of circuit schematic requires many inverters. Instead, NOR-NOR gates are used to avoid the appearance of inverters. It completely reduces the propagation delay due to the inverter.

	W/L (NMOS) (in um)	W/L (PMOS) (in um)	t_{PHL} (15fF)	t_{PLH} (15fF)	t_{PHL} (120fF)	t_{PLH} (120fF)	t_{PHL} (960fF)	t_{PLH} (960fF)
Inverter	0.48/0.24	1.2/0.24	0.1355ns	0.13911ns	0.5702ns	0.5806ns	4ns	4.024ns
2 input nand	0.96/ 0.24	1.2/ 0.24	0.14721 ns	0.19292ns	0.52873ns	0.6339ns	3.589ns	4.1029n
2 input xor	0.48/0.24	1.2/0.24	0.326ns	0.1892ns	1.033ns	0.612ns	5.373ns	4.0675ns
2 input nor	0.48/0.24	4.8/0.24	0.183ns	0.133ns	0.6257ns	0.351ns	4.0731ns	2.058ns
3 input nor	0.48/0.24	4.8/0.24	0.202ns	0.19ns	0.644ns	0.51ns	4.095ns	3.07ns
4 input nor	0.48/0.24	4.8/0.24	0.219ns	0.169ns	0.66ns	0.593ns	4.116ns	3.965ns
5 input nor	0.48/0.24	6/0.24	0.2659ns	0.172ns	0.689ns	0.59ns	4.14ns	4.15n

Table 1. The width and length, the output low-to-high propagation and the output high-to-low propagation of each basic cell

3) Full Adder

The full adder is designed based on tri-state inverters, referring to [1].

At the first half of the full adder, as depicted in figure 3.2, two tri-state inverters are connected in parallel, which represents a logic OR, such that $\underline{C}$can be obtained at the output node of the second tri-state inverter. An extra inverter is connected to retrieve C as the carry output. According to the truth table, table 3.1, the equation of C_{out} can be derived as $C_{out} = A \cdot B + C_{in} \cdot (A + B)$. The first tri-state inverter only performs the logic operation of $\underline{C_{in} \cdot (A + B)}$. The $\underline{A \cdot B}$logic operation is executed in the second tri-state inverter. In the first tri-state inverter, when C_{in} is 1 and one of the A and B or both are also 1, the output node will be shorted to ground, resulting in a logic 0 as output. Similarly, when C_{in} is 0 and one of the A and B or both are also 0, the output will be connected to vdd, resulting in a logic 1 as output. In the second tri-state inverter, when both A and B are 1, the output node will be grounded, as logic 0; when both A and B are 0, the output node will be connected to vdd, as logic 1.

At the second half of the full adder, as depicted in figure 3.3, $\underline{S}$ can be obtained by the two parallel tri-state inverters, with using the result of $\underline{C}$.

The equation for S, $S = A \cdot \underline{B} \cdot \underline{C} + \underline{A} \cdot B \cdot \underline{C} + \underline{A} \cdot \underline{B} \cdot C + A \cdot B \cdot C$

$= (A + B + C) \cdot (\underline{B} \cdot \underline{C} + \underline{A} \cdot \underline{C} + \underline{A} \cdot \underline{B}) + A \cdot B \cdot C$

$= (A + B + C) \cdot \underline{C} + A \cdot B \cdot C$ where $\underline{C} = \underline{B} \cdot \underline{C} + \underline{A} \cdot \underline{C} + \underline{A} \cdot \underline{B}$

The first tri-state inverter in figure 3.3 executes $\underline{(A + B + C) \cdot \underline{C}}$, while the second one executes $\underline{A \cdot B \cdot C}$. The output of the tri-state inverters will be inverted to obtain S as the sum.

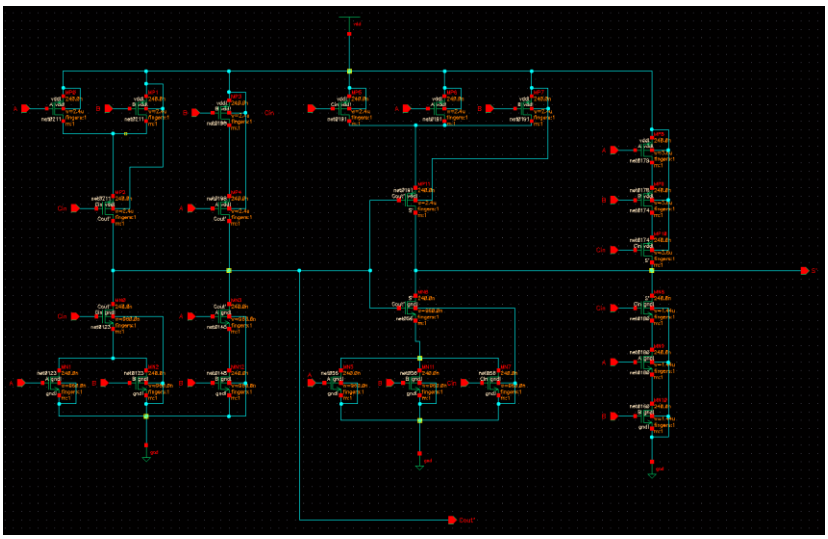


Figure 3.1) Full Adder Schematic (draw with cadence)

A	B	Cin	$\underline{C_{out}}$	C_{out}	$\underline{S}$	S
0	0	0	1	0	1	0
0	0	1	1	0	0	1
0	1	0	1	0	0	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	0	1	0	1

Table 3.1)Truth table of full adder

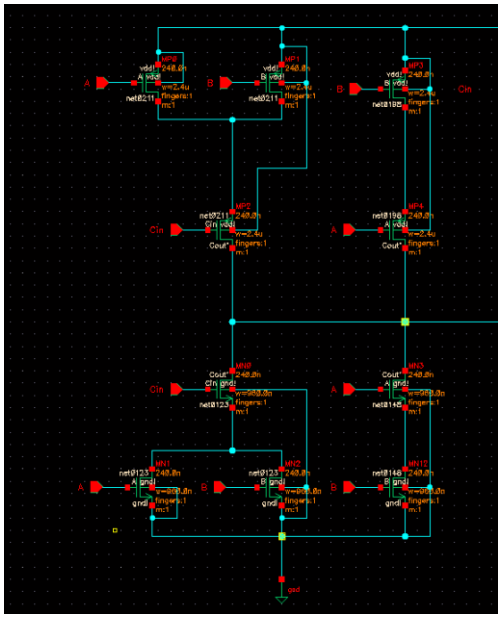


Figure 3.2) First half of the full adder (draw with cadence)

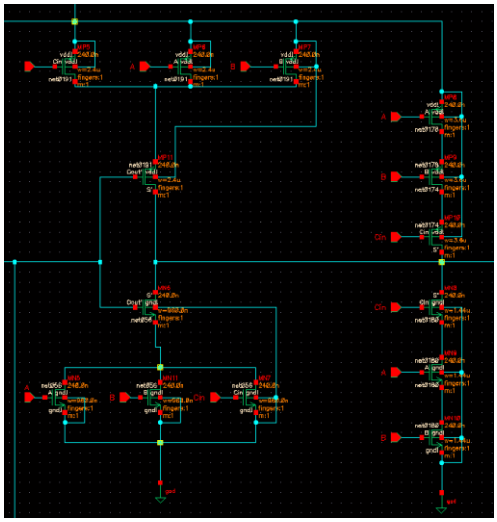


Figure 3.3) Second half of the full adder (draw with cadence)

	MN0	MN1	MN2	MN3	MN5	MN6	MN7
W/L	2.88u/0.24u	2.88u/0.24u	2.88u/0.24u	0.96u/0.24u	0.96u/0.24u	0.96u/0.24u	0.96u/0.24u

MN8	MN9	MN10	MN11	MN12	MN13	MN14
1.44u/0.24u	1.44u/0.24u	1.44u/0.24u	0.96u/0.24u	0.96u/0.24u	1.44u/0.24u	0.48u/0.24u

MP0	MP1	MP2	MP3	MP4	MP5	MP6
7.2u/0.24u	7.2u/0.24u	7.2u/0.24u	2.4u/0.24u	2.4u/0.24u	2.4u/0.24u	2.4u/0.24u

MP7	MP8	MP9	MP10	MP11	MP12	MP13
2.4u/0.24u	3.6u/0.24u	3.6u/0.24u	3.6u/0.24u	2.4u/0.24u	1.2u/0.24u	3.6u/0.24u

4) 16-bit Carry-Skip Adder

The 16-bit Carry-Skip Adder is designed using the Variable Size Block-Carry-Skip Adders Structure.

The adder block will arrange in the following order, 1bit (LSB) -> 2bit -> 3bit -> 4bit -> 3bit -> 2bit ->1bit (MSB).

As an inverting multiplexer structure is applied on Full Adder, the Skip logic needs to be modified. Consider a full adder, the carry bit can be passed when one of the input bits(a_i , b_i) is high, so We added a condition T_i and $T_i = a_i + b_i$. When all T in the n-bit block are true, the carry bit can be passed with equation:

$$C_{out} = C_{in} \cdot T_i \cdot T_{i+1} \cdot T_{i+2} \cdot \ldots \cdot T_{i+n}$$

To improve the performance we modify the equation a little bit, it become:

$$C_{out} = \underline{C_{in}} + \underline{T_i} + \underline{T_{i+1}} + \underline{T_{i+2}} + \ldots + \underline{T_{i+n}}$$

with this equation, we are using NOR gate only which has a better performance compared to using AND gate and OR gate. Finally, use a OR gate to merge with the result from the adder path.

The following graph will show the schematic of the 16-bit adder

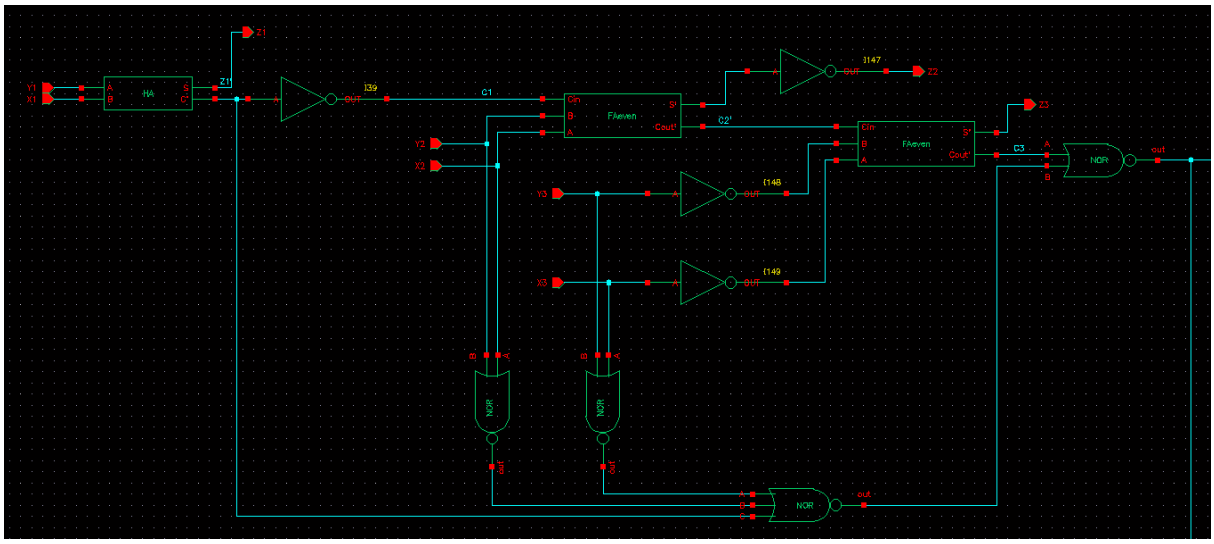


Fig 4.1) The first two block of 16 bit adder (1bit -> 2bit) (draw with cadence)

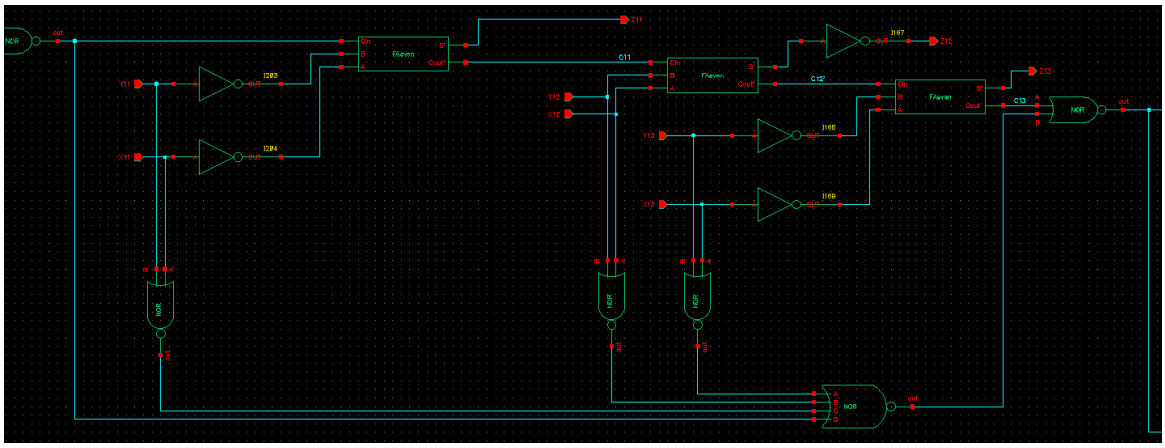


Fig 4.2) The third block of 16 bit adder (3bit)(draw with cadence)

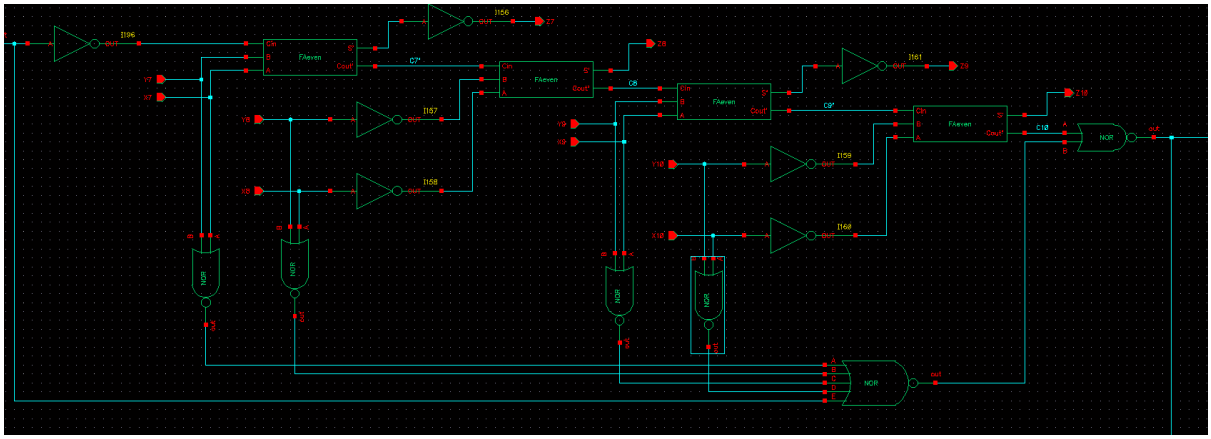


Fig 4.3) The fourth block of 16 bit adder (4bit)(draw with cadence)

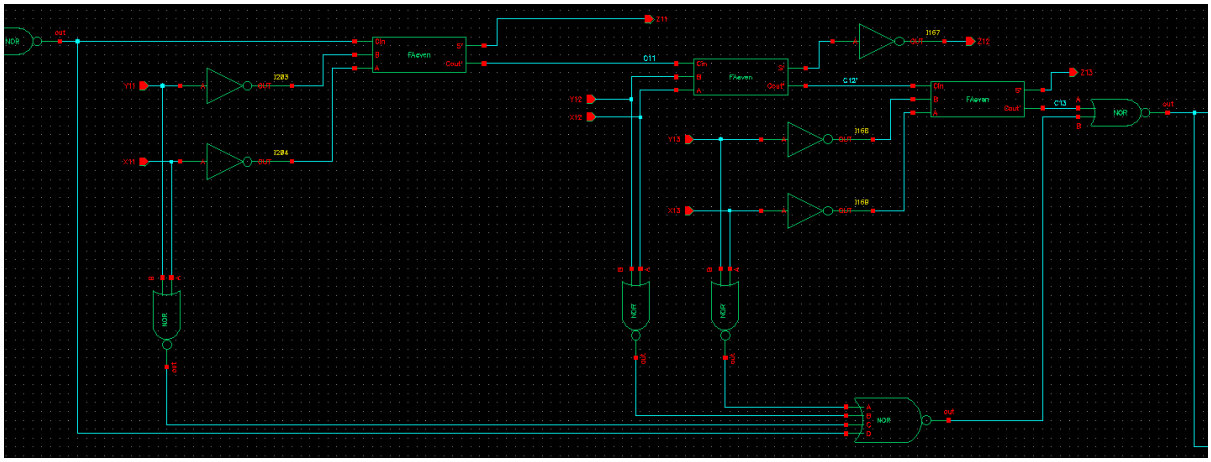


Fig 4.4) The fifth block of 16 bit adder (3bit)(draw with cadence)

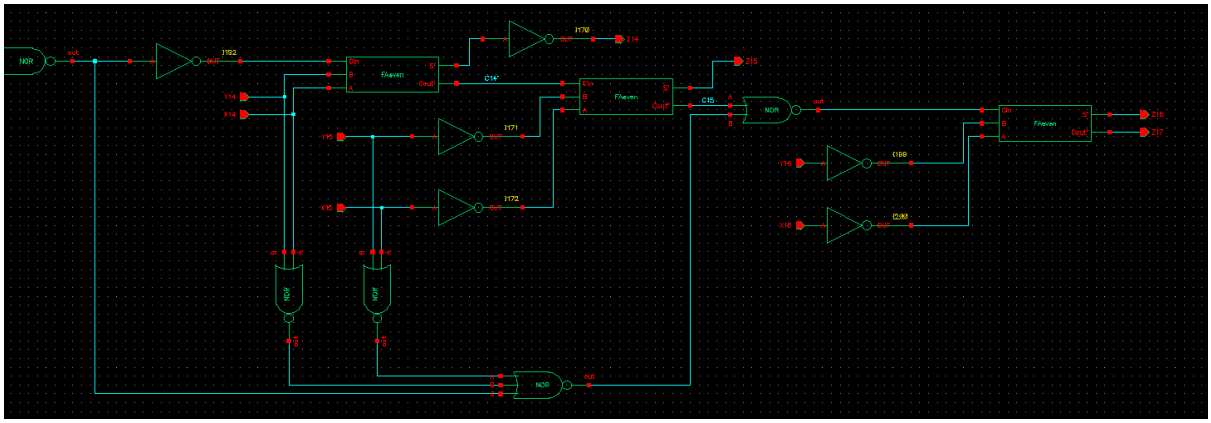


Fig 4.5) The last two block of 16 bit adder (2bit->1bit)(draw with cadence)

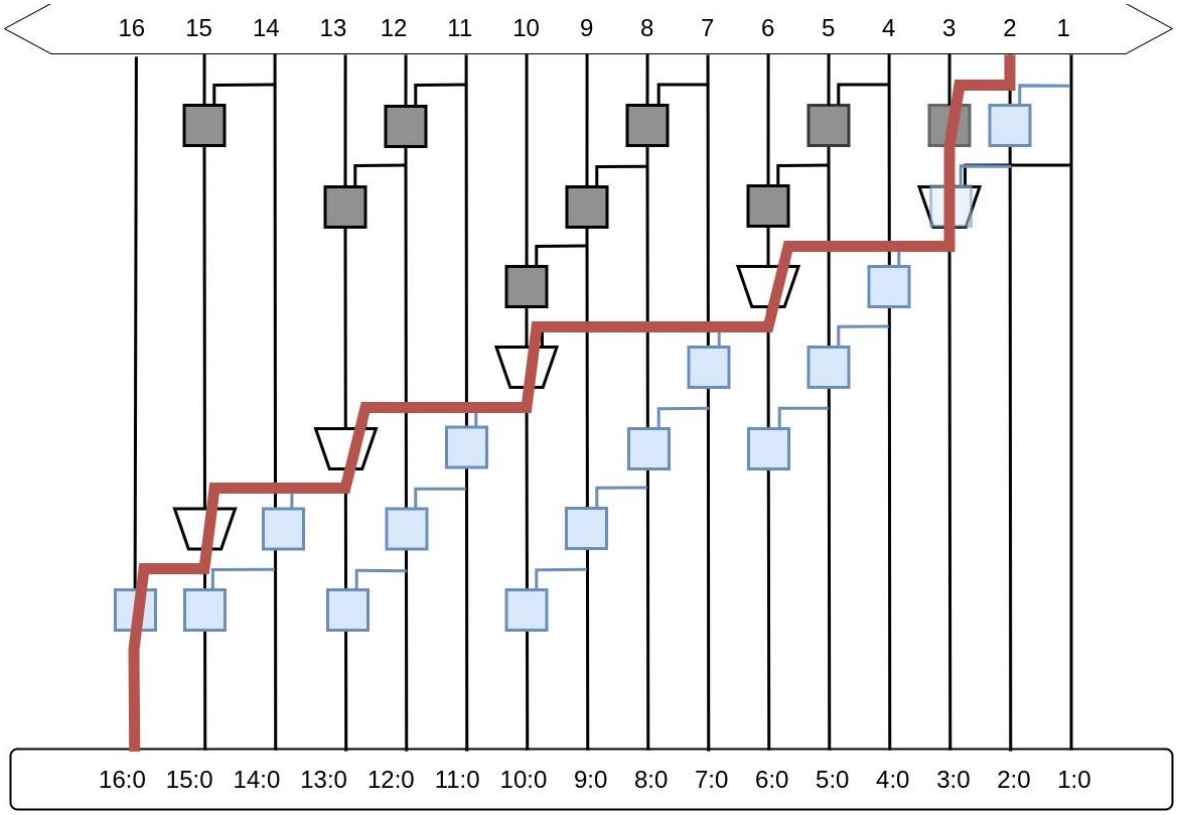


Fig 4.6) Network of the adder (Draw at https://www.diagrameditor.com/)

The red line is one of the critical delay paths on the network diagram. In order to measure the critical delay, there are three requirements:
Firstly, the $Z16$output needs to wait for the result of $Z17$due to the alternative design of the full adder. Therefore, $(X16, Y16)$need to be $(1,0)$or $(0,1)$ and wait for $C_{in}$to achieve max. delay.
Secondly, the initial carry bit needs to be generated in the adder of $X2, Y2$, but not LSM. Therefore, it needs to fulfill 3 things. LSM must not generate a carry bit. $(X2, Y2)$ need to be $(1,1)$. $X3, Y3$need to be $(1,0)$ or$(0,1)$.
Thirdly, other bits need to be $(1,0)$ or$(0,1)$.

The number of possible critical delay path

$$= \prod_{i=1}^{16} \quad possible\ combination\ of\ (Xi, Yi)$$

$$= 2 * 3 * 1 * 2 * 2^{(16-4)}$$

$$= 49152$$

One of them is $(X, Y) = (1111\ 1111\ 1111\ 1110, 0000\ 0000\ 0000\ 0011)$

Propagation delay with schematic level simulation
(input file in appendix)

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
t_{PLH}	0.14ns	0.91ns	0.91ns	1.53ns	2.01ns	2.07ns	2.9ns	3.09ns
t_{PHL}	0.15ns	0.8ns	0.87ns	1.58ns	2.02ns	2.07ns	2.16ns	2.33ns

Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17
3.56ns	3.63ns	4.25ns	4.72ns	4.79ns	5.59ns	5.67ns	6.36ns	2.57ns
2.78ns	2.83ns	2.28ns	2.73ns	2.78ns	2.84ns	2.89ns	3.04ns	5.91ns

5) 16-bit Carry-Skip Adder Layout Area and Delay

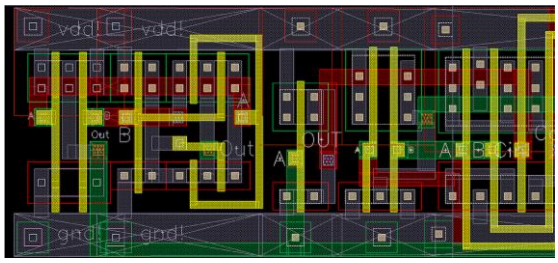


Figure 5.1.1) Layout of the design

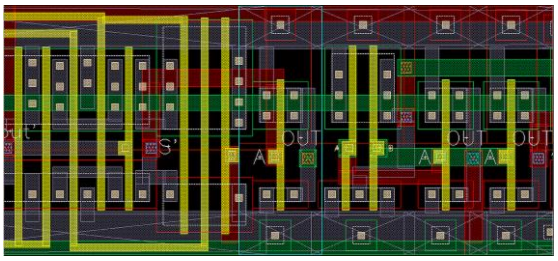


Figure 5.1.2) Layout of the design

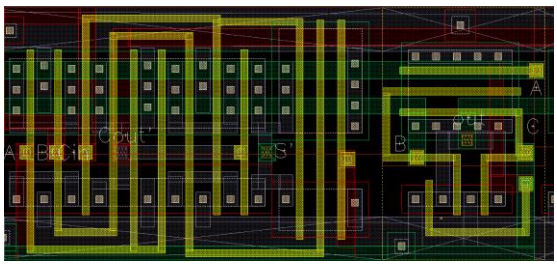


Figure 5.1.3) Layout of the design

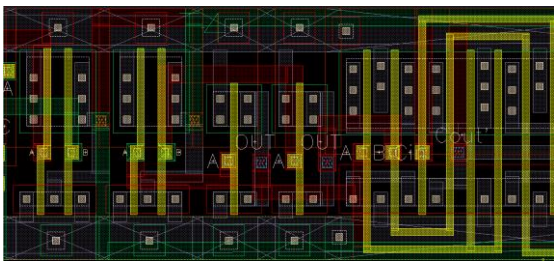


Figure 5.1.4) Layout of the design

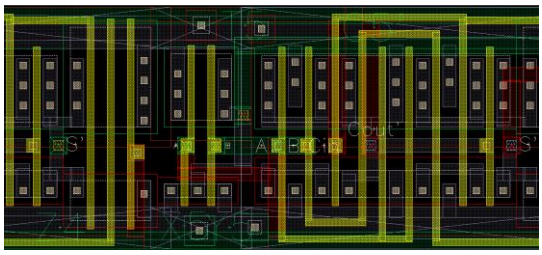


Figure 5.1.5) Layout of the design

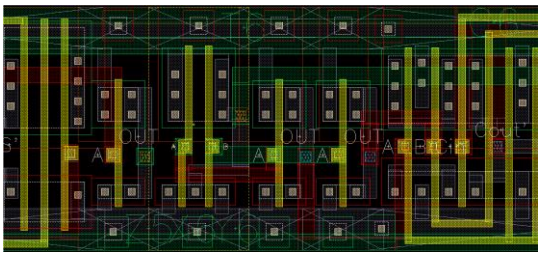


Figure 5.1.6) Layout of the design

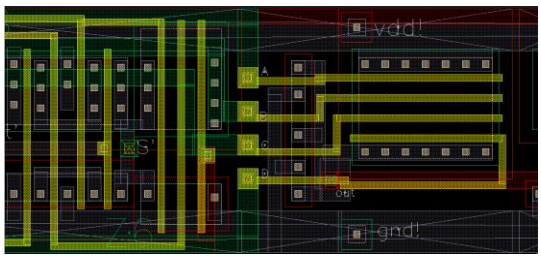


Figure 5.1.7) Lathe desiyout of gn

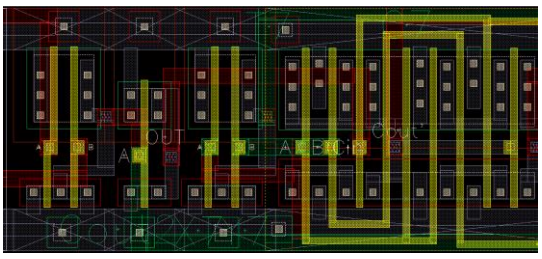


Figure 5.1.8) Layout of the design

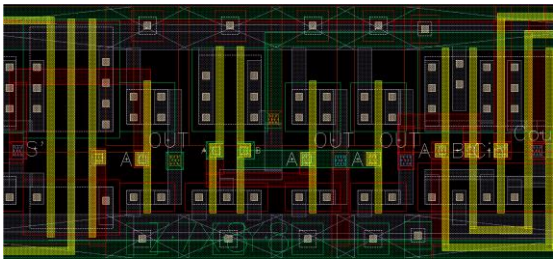


Figure 5.1.9) Layout of the design

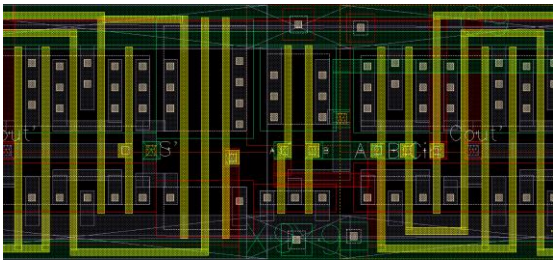
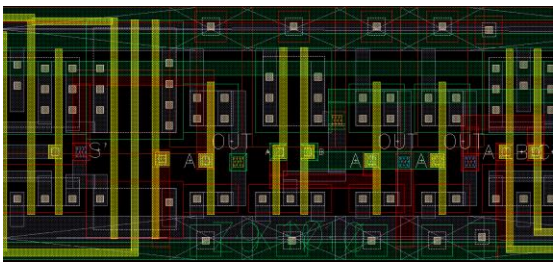
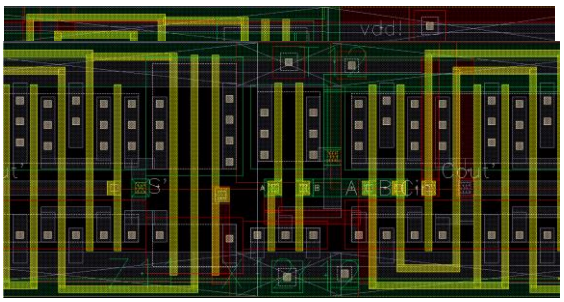


Figure 5.1.10) Layout of the design

Figure 5.1.11) Layout of the design

Figure 5.1.12) Layout of the design





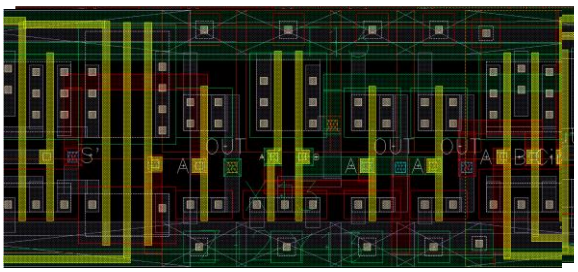


Figure 5.1.13) Layout of the design

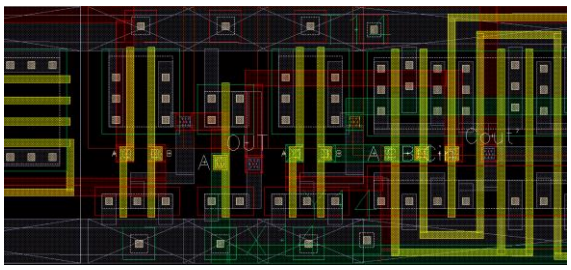


Figure 5.1.14) Layout of the design

Figure 5.1.15) Layout of the design

Figure 5.1.16) Layout of the design

Figure 5.1.17) Layout of the design

Figure 5.1.18) Layout of the design

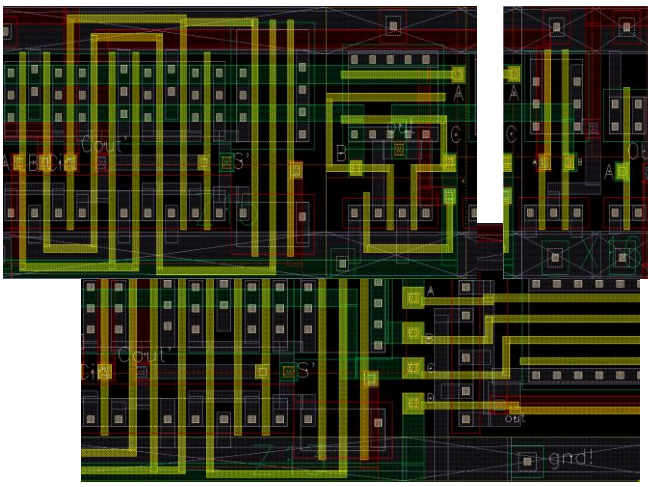


Figure 5.1.19) Layout of the design

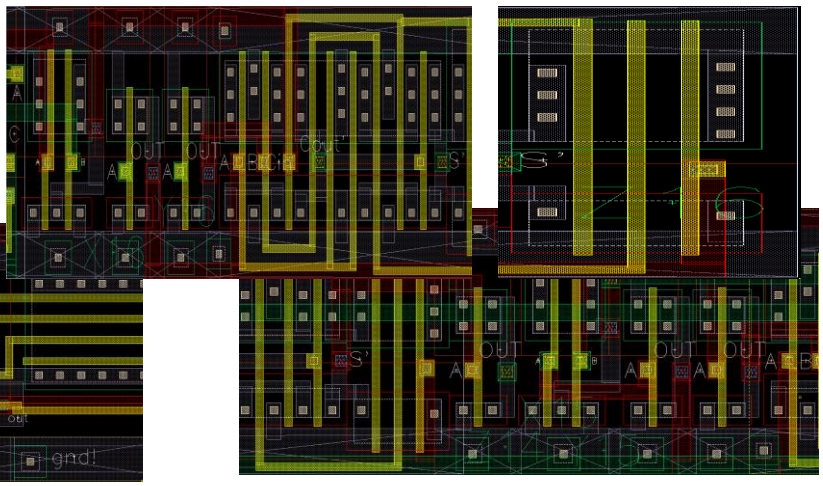


Figure 5.1.21) Layout of the design

Figure 5.1.20) Layout of the design

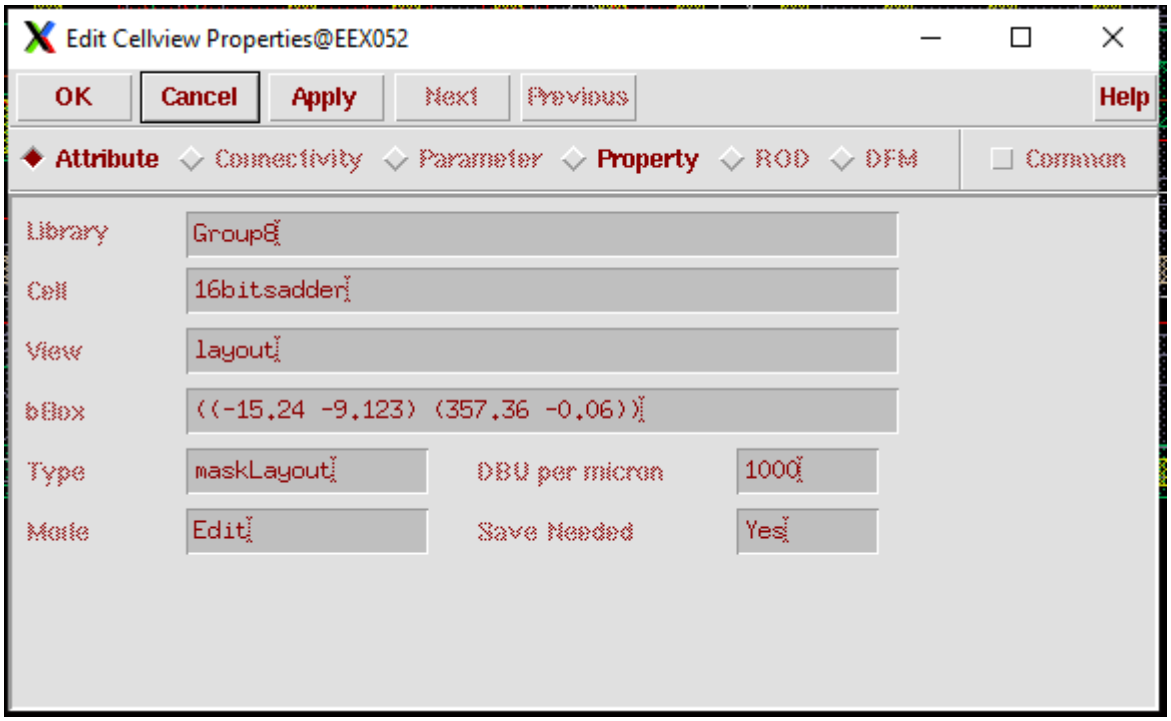


Fig 5.2) Area of Layout of 16-bit Carry Skip Adder

The area of the design = (357.36 - (-15.24))(-0.06 - (-9.123)) = 3376.8738 $(\mu m)^2$

Propagation delay with schematic level simulation
(input file in appendix)

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
t_{PLH}	0.14ns	0.91ns	0.91ns	1.53ns	2.01ns	2.07ns	2.9ns	3.09ns
t_{PHL}	0.15ns	0.8ns	0.87ns	1.58ns	2.02ns	2.07ns	2.16ns	2.33ns

Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17
3.56ns	3.63ns	4.25ns	4.72ns	4.79ns	5.59ns	5.67ns	6.36ns	2.57ns
2.78ns	2.83ns	2.28ns	2.73ns	2.78ns	2.84ns	2.89ns	3.04ns	5.91ns

Propagation delay with C only extraction

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
t_{PLH}	0.16ns	0.947ns	1.019ns	1.744ns	2.283ns	2.364ns	3.396ns	3.615ns
t_{PHL}	0.185ns	0.91ns	1.02ns	1.9ns	2.45ns	2.57ns	2.71ns	2.95ns

Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17
3.165ns	4.158ns	4.983ns	5.527ns	5.61ns	6.588ns	6.668ns	7.396ns	3.4ns
3.51ns	3.62ns	2.92ns	3.48ns	3.6ns	3.69ns	3.8ns	3.85ns	6.891ns

Propagation delay with R only extraction

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
t_{PLH}	0.1566ns	0.947ns	1.02ns	1.743ns	2.282ns	2.365ns	3.394ns	3.613ns
t_{PHL}	0.158ns	0.913ns	1.026ns	1.903ns	2.463ns	2.587ns	2.715ns	2.957ns

Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17
4.154ns	4.237ns	4.979ns	5.525ns	5.608ns	6.585ns	6.668ns	7.397ns	3.447ns
3.514ns	3.629ns	2.937ns	3.496ns	3.612ns	3.7308ns	3.815ns	3.862ns	6.89ns

Propagation delay with RC only extraction

	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8
t_{PLH}	0.16	0.95ns	1.02ns	1.75ns	2.29ns	2.37ns	3.4ns	3.62ns
t_{PHL}	0.16ns	0.91ns	1.03ns	1.9ns	2.46ns	2.58ns	2.72ns	2.96ns

Z9	Z10	Z11	Z12	Z13	Z14	Z15	Z16	Z17
4.16ns	4.24ns	4.99ns	5.53ns	5.62ns	6.59ns	6.68ns	7.41ns	3.42ns
3.52ns	3.63ns	2.93ns	3.5ns	3.61ns	3.71ns	3.81ns	3.88ns	6.9ns

Quality = 7.41ns* 3376.8738 $(\mu m)^2$

Discussion

By comparing the longest delay times among the post-layout simulation and schematic level simulation, as highlighted in the tables above, the delay times in post-layout are longer than the one in schematic level. We believe that this is because the post-layout simulation has more accurate parasitic capacitance and resistance calculation, which are based on the layout have drawn. The schematic level circuit can only simulate the delay with minimum parasitic capacitance and resistance, which may not match the of the layout, and lead to under-estimation of the actual delay time. Accurate resistance and capacitance extraction for correct delay estimation with CADENCE tools is important for the pre-fabrication process, for which we can simulate the performance of the device to be fabricated, such that we can make suitable modification, if needed, before the fabrication.

Appendix:

Input file:

```
simulator lang=spectre
v0 (vdd! 0) vsource type=dc dc=2.5
v1 (X1 0) vsource type=pwl wave=[0n 0]
v2 (X2 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v3 (X3 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v4 (X4 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v5 (X5 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v6 (X6 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v7 (X7 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v8 (X8 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v9 (X9 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v10 (X10 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v11 (X11 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v12 (X12 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v13 (X13 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v14 (X14 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v15 (X15 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v16 (X16 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5]
v17 (Y1 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5 10n 2.5 10.2n 0 15n 0 15.2n 2.5]
v18 (Y2 0) vsource type=pwl wave=[0n 0 5n 0 5.2n 2.5 10n 2.5 10.2n 0 15n 0 15.2n 2.5]
v19 (Y3 0) vsource type=pwl wave=[0n 0]
v20 (Y4 0) vsource type=pwl wave=[0n 0]
v21 (Y5 0) vsource type=pwl wave=[0n 0]
v22 (Y6 0) vsource type=pwl wave=[0n 0]
v23 (Y7 0) vsource type=pwl wave=[0n 0]
v24 (Y8 0) vsource type=pwl wave=[0n 0]
v25 (Y9 0) vsource type=pwl wave=[0n 0]
v26 (Y10 0) vsource type=pwl wave=[0n 0]
v27 (Y11 0) vsource type=pwl wave=[0n 0]
v28 (Y12 0) vsource type=pwl wave=[0n 0]
v29 (Y13 0) vsource type=pwl wave=[0n 0]
v30 (Y14 0) vsource type=pwl wave=[0n 0]
v31 (Y15 0) vsource type=pwl wave=[0n 0]
v32 (Y16 0) vsource type=pwl wave=[0n 0]
ca1 (Z1 0) capacitor c=50f
ca2 (Z2 0) capacitor c=50f
ca3 (Z3 0) capacitor c=50f
ca4 (Z4 0) capacitor c=50f
```

```
ca5 (Z5 0) capacitor c=50f
ca6 (Z6 0) capacitor c=50f
ca7 (Z7 0) capacitor c=50f
ca8 (Z8 0) capacitor c=50f
ca9 (Z9 0) capacitor c=50f
ca10 (Z10 0) capacitor c=50f
ca11 (Z11 0) capacitor c=50f
ca12 (Z12 0) capacitor c=50f
ca13 (Z13 0) capacitor c=50f
ca14 (Z14 0) capacitor c=50f
ca15 (Z15 0) capacitor c=50f
ca16 (Z16 0) capacitor c=50f
ca17 (Z17 0) capacitor c=50f
save :pwr
```

Schematic design	Basic Cell	All
	Structure for 16-bit adder	Hau Kai Hong
Layout design	Basic Cell	All
	16-bit adder	Hau Kai Hong
Report	Part 1	Ching Wai Kin
	Part 2	Ching Wai Kin, Chiu Siu Ki
	Part 3	Ching Wai Kin
	Part 4	IP Ming Chiu
	Part 5	Chiu Siu Ki
	Part 5 - Case Testing	Ching Wai Kin Chiu Siu Ki Ip Ming Chiu

Ching Wai Kin
Chiu Siu Ki
Hau Kai Hong
Ip Ming Chiu