8-bit Kogge Stone Adder

Thet Paing Zaw A16576828; Pengxin Meng A15432376; Yuxiang Chen A16552742

Since we chose the Kogge Stone adder, there would be many more challenges we will meet as its structure is very different and complicated. Moreover, before that, we chose the Ripple Carry adder since we want to start from a simple and easy way. However, after the testbench, we realized that we got what we paid for: the ripple-carry adder could not work fine in the high-frequency situation and its output is not adorable. Therefore, we decided to abandon plan A and move on to the Kogge Stone Adder. Obviously, we spent more time analyzing the structure and building up components since we have Black Cells and Gray cells and some other propagation elements that help us look further. Nevertheless, they are still quite important since those extra but inevitable cells could promote the adder's performance. Another challenge is the optimization of the W/L ratio. Not directly dragging the components from the library, we designed the basic components such as NAND, and XOR from the very beginning and figured out the best ratio for different cells.

For our previous design architecture, Ripple Carry Adder, its simple structure decides that it is destined to be not optimal for speed. We could compare Kogge Stone Adder and Carry Ripple Adder by their critical path. For Ripple Carry Adder, it must calculate everything from the first bit to the last bit; But for Kogge Stone Adder, once it notices that the carry out is one, it would automatically jump to the next bit so that the Kogge Stone Adder will save a lot of time on calculating the following bit. Therefore, that is why Kogge Stone Adder can reach optimal speed compared to the Ripple Carry Adder.

Just like I mentioned earlier, Kogge Stone Adder can have better performance due to its carry look-ahead functionality. However, it is still different from the original Carry Look Ahead adder that the Kogge Stone adder uses the parallel algorithm that, for example, the first (least-significant) sum bit is calculated by XORing the propagate in the farthest-right box (a "1") with the carry-in (a "0"), producing a "1". The second bit is calculated by XORing the propagate in the second box from the right (a "0") with C0 (a "0"), producing a "0" [1].

Although the Kogge Stone Adder might be one of the fastest Adder in the world, it still has some potential shortcomings which can be further improved. First of all, although we have built up the adder based on its structure, modifying the size of PDN and PUN would make the adder behave better than usual. Second, better parallelism is another innovation we brought to our Kogge Stone Adder. We reduced the number of required stages so that it will save much more time and energy. (need implement later)

During our tests of the custom design adder, we tried different frequencies to see which one is the maximum. Starting from the 2.3 GHz, which is the maximum frequency of the adder which the lab material provided us, we finally reached around 3.7GHz. After that the output no longer followed the exact calculation results (00000001 + 01111111 = 1000000) but showed some disorder outcomes. We thought we have reached the maximum frequency. However, after we tried around 7 GHz, it worked well again and showed correct output after the flip-flop except a big ramp. In the very beginning, we thought that the adder may have better performance in higher frequencies but after taking a deeper look into it, the ramp that occurred abnormally told us that this may caused by the flip-flop that the raw output was not correct, but after the procession of flip flop, the output accidentally matched the correct output.

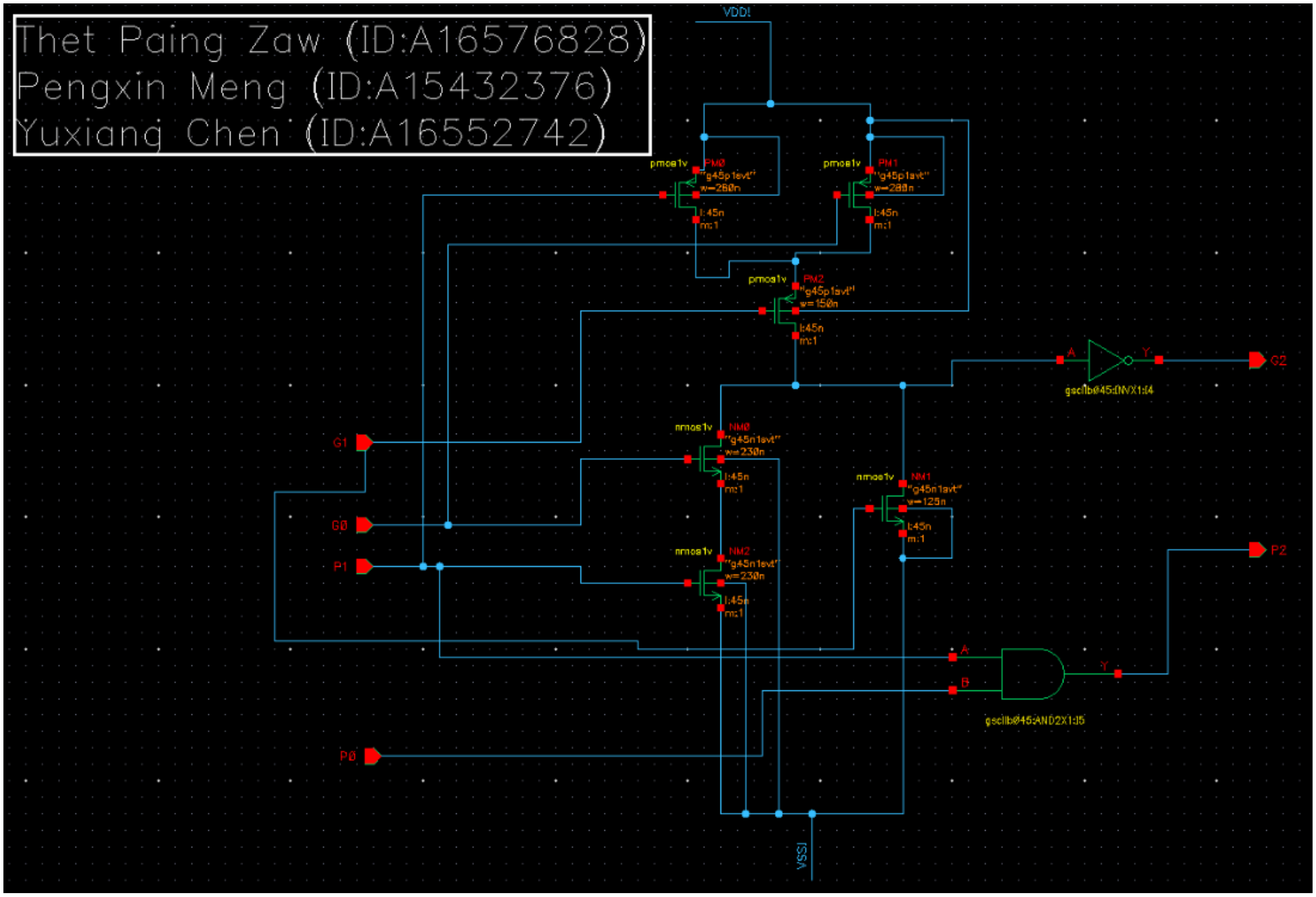
For the future design iteration, I think we could mainly focus on the wiring since although the Kogge Stone adder has better performance, it sacrifices the area, routing congestion, and power efficiency. Probably multi-layer technology will be helpful for designing high-performance adders in limited areas. However, it may generate more heat and noise and thus set up higher requirements for the current science. Moreover, since we didn't try the layout design, some merges and fusion of parasitic capacitors that can improve the efficiency of the circuit were not tested. Therefore, we could probably set up the layout of the adder and thus, find a method to improve its performance by inspecting the routing and wiring. Furthermore, the sizing of all components, such as Gray and Black cells, may have a better measurement to reduce the propagation delay by finding the critical path and setting those unimportant paths as small as possible. Besides that, we found out that there is an enhanced model Kogge Stone Adder called sparse Kogge Stone Adder [2] that can reduce the total need for computation and routing congestion. Its theory is that it will measure how many carry bits are generated and organize them to be used in certain carry-in inputs which will have the much shorter ripple carry adders or some other carry design.And thus, generate the final sum bit. Additionally. There are still many adders we haven't tried, for example, Brent-Kung adder, which saves more area than Kogge Stone; Ripple save adder, Ripple Select adder, etc.

One of our team members, Yuxiang, needs to be absent from the campus for one week due to personal issues. Thet created the schematic and Pengxin wrote the final report. Then, all team members discussed the further possible improvements and Yuxiang tested the Adder to make sure it could work properly and recorded the parameters such as maximum frequency and power consumption under certain conditions.

Acknowledgments:
First and foremost, to our greatest gratitude, thank Professor Patrick Mercier for his excellent guidance and patient instructions that help us have a fundamental recognition of Very Large Scale Integrated Circuits Design. Although the class is intense, he still shows great professionalism. Second, we would like to appreciate the help from TAs that we could not get timely help and inspirational hints without their assistance. Their professional ethics will be deeply embedded in the project and shall be remembered thoroughly. Last but not least, thank you very much to all team members for their great effort and indefatigability. They are the foundation for the success of this project.

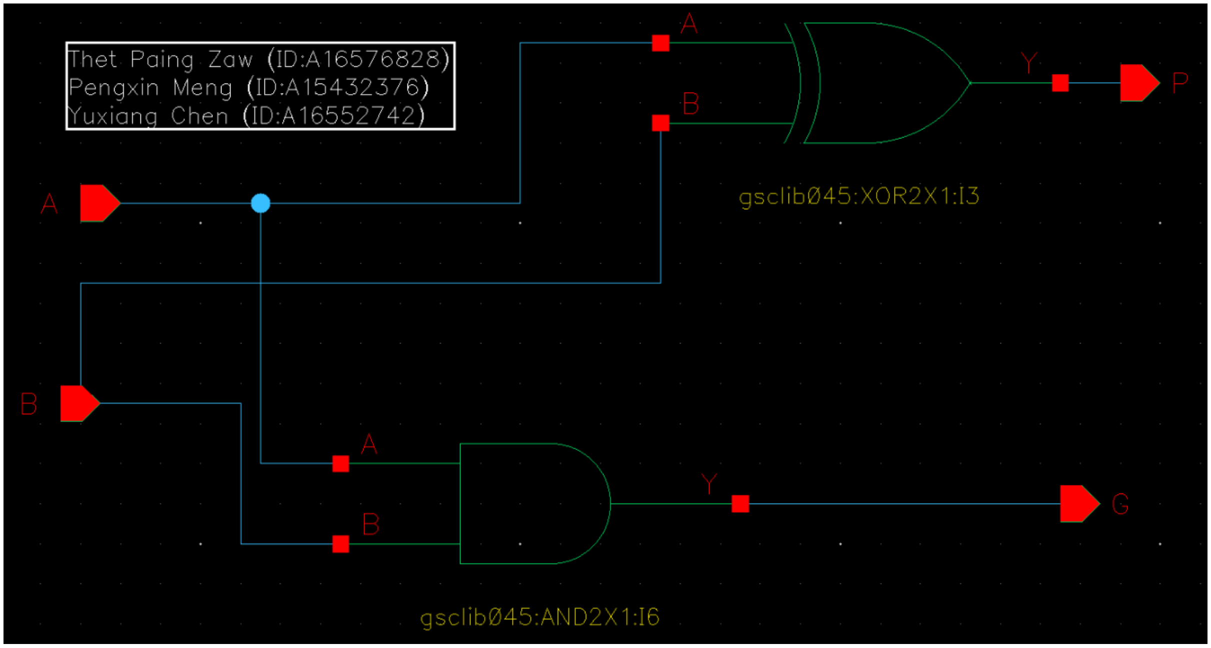
References:
[1] https://en.wikipedia.org/wiki/Kogge%E2%80%93Stone_adder
[2] Enhancement/
https://en.wikipedia.org/wiki/Kogge%E2%80%93Stone_adder

1. Largest Cell (Black Cell)

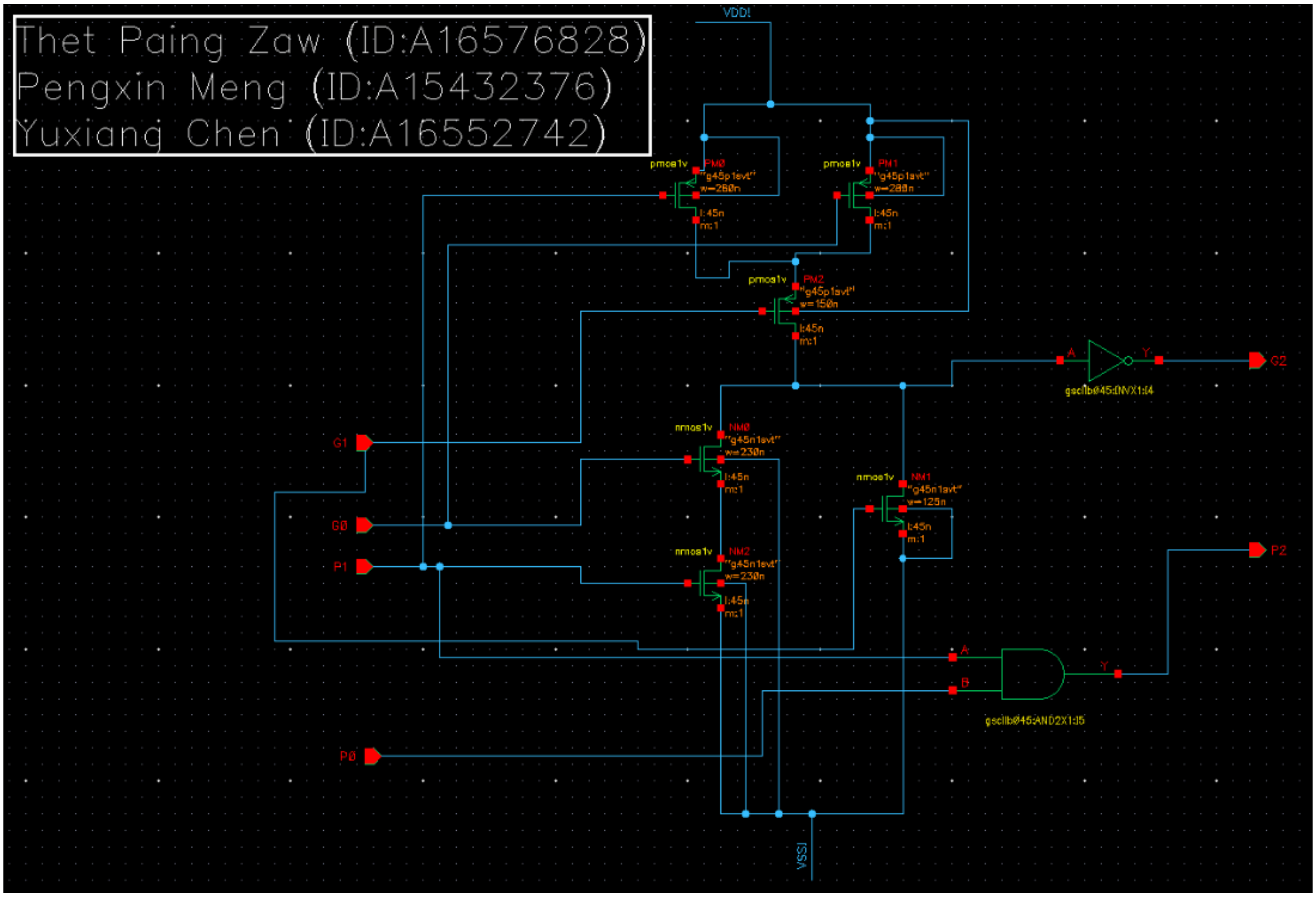


2. Important Circuit

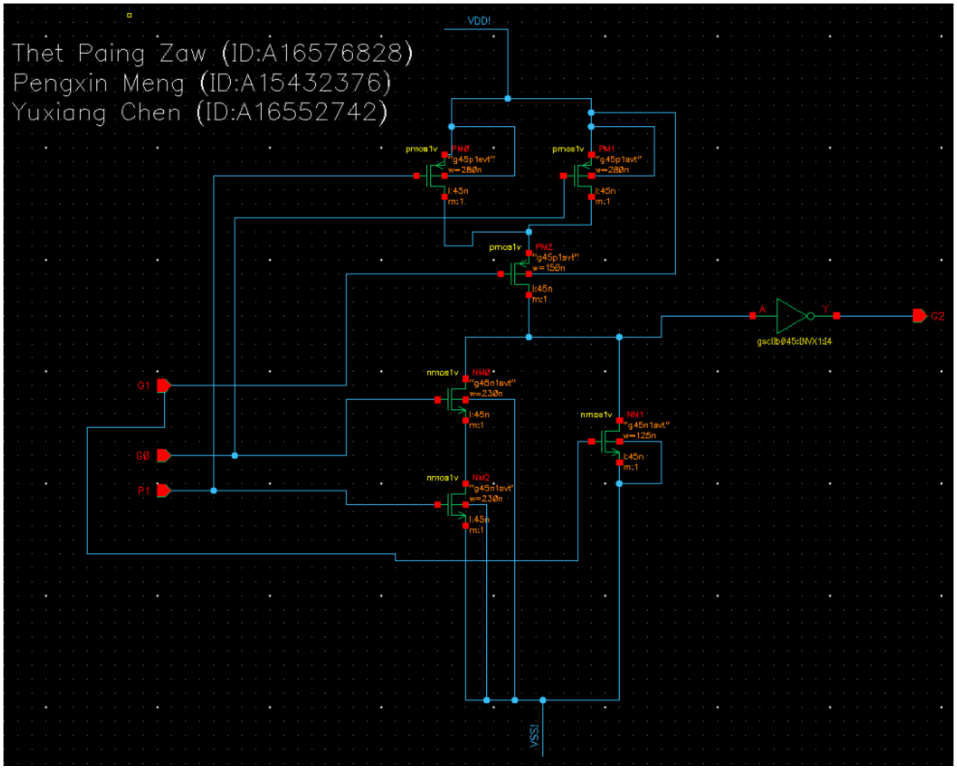
2.1. Preprocess



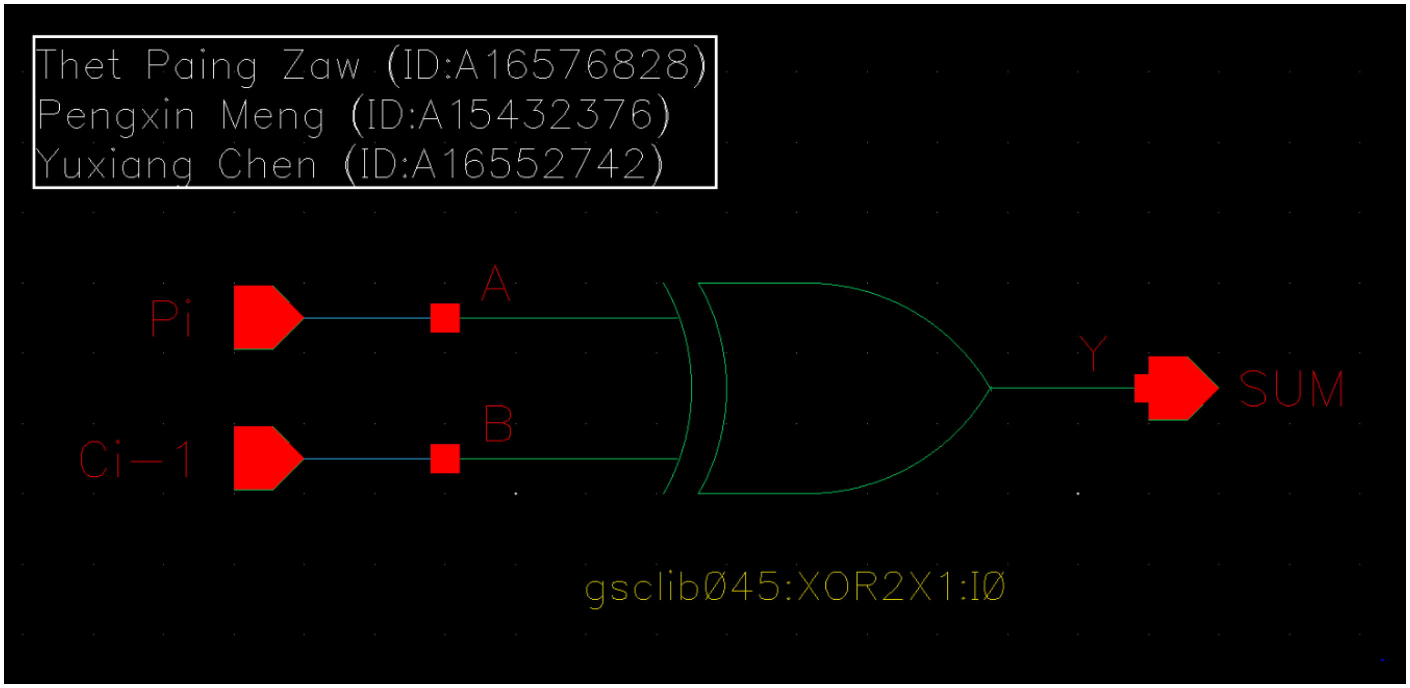
2.2. Black Cell



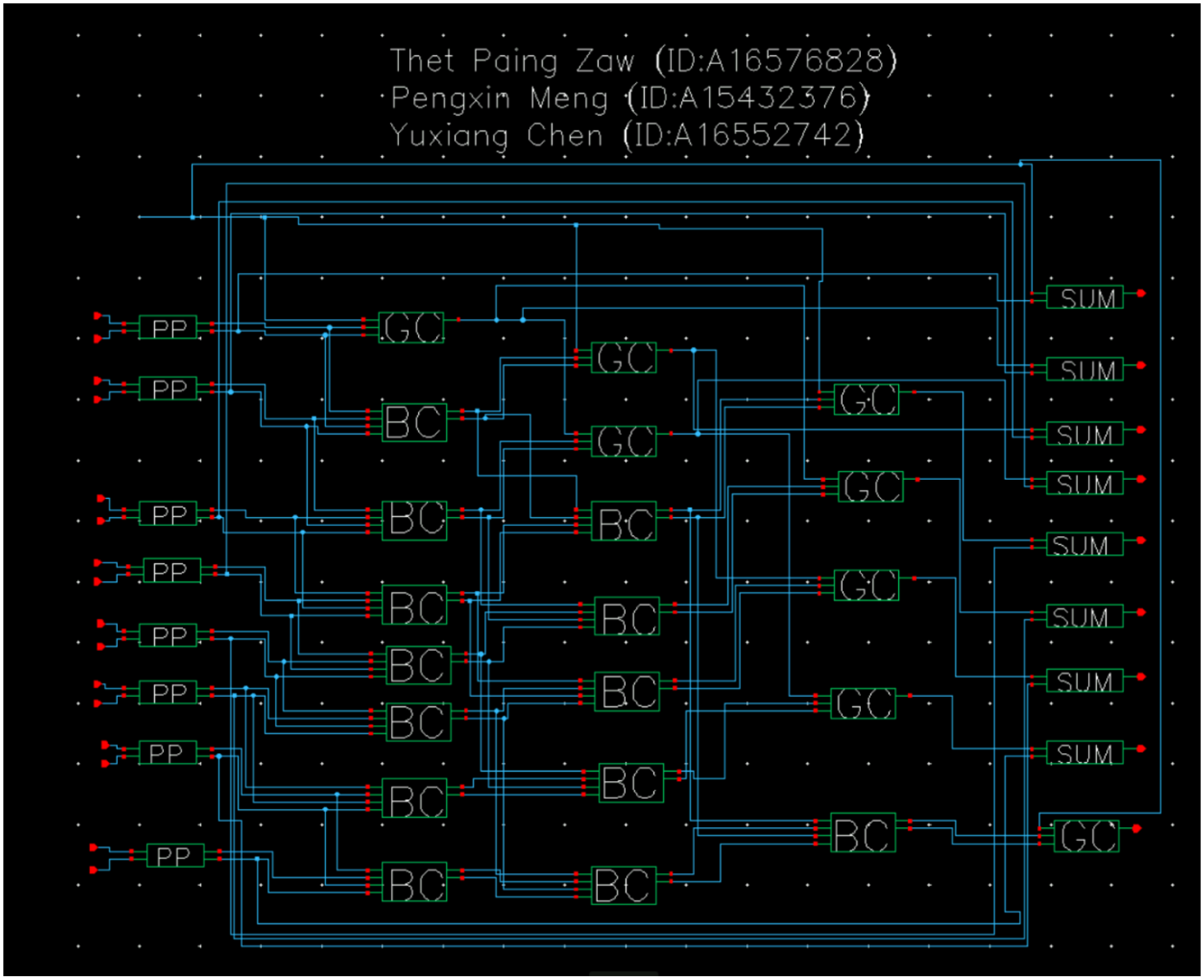
2.3. Gray Cell



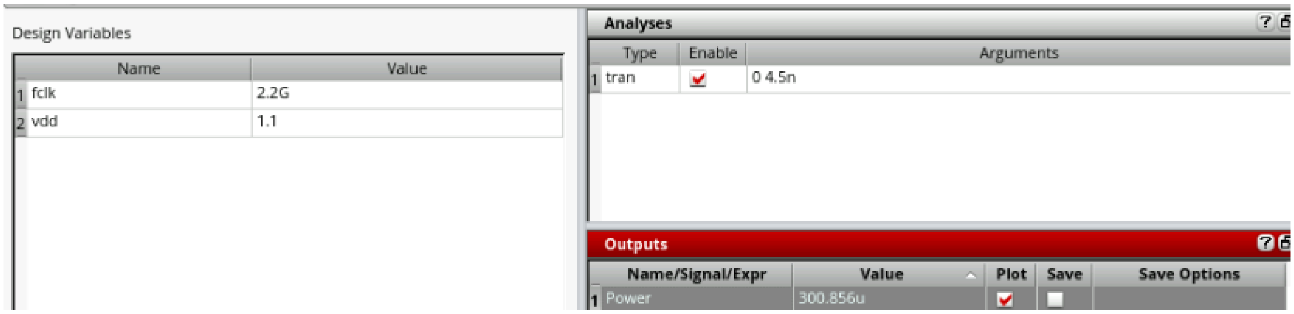
2.4. Sum



3. 8-bit Kogge Stone Adder



4. Transient Simulation result



Design Variables

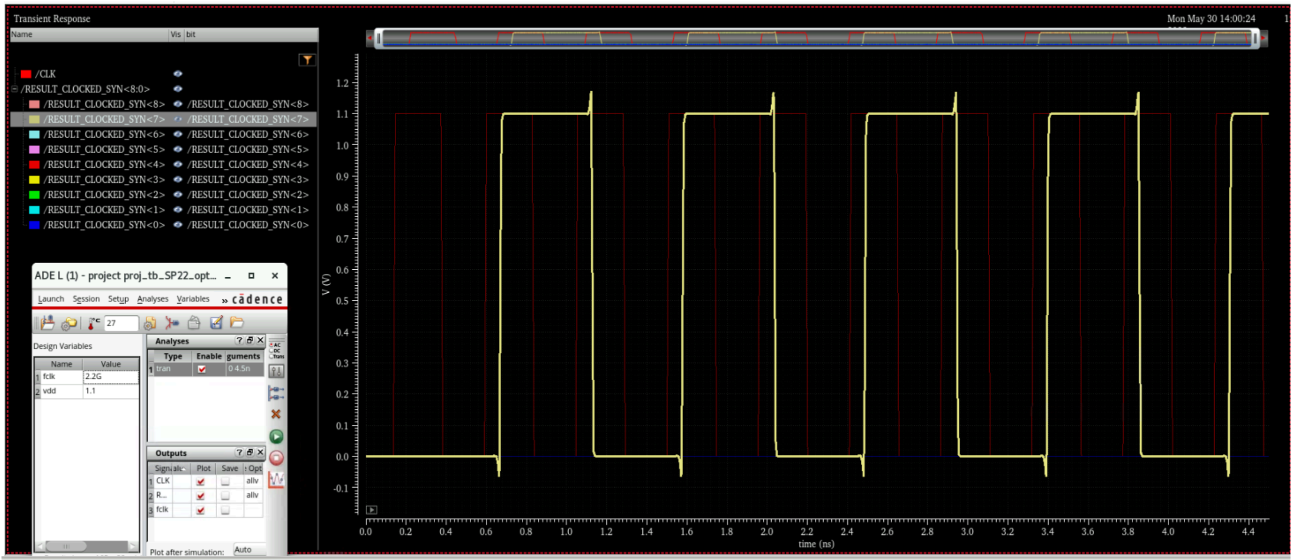
	Name	Value
1	fclk	2.2G
2	vdd	1.1

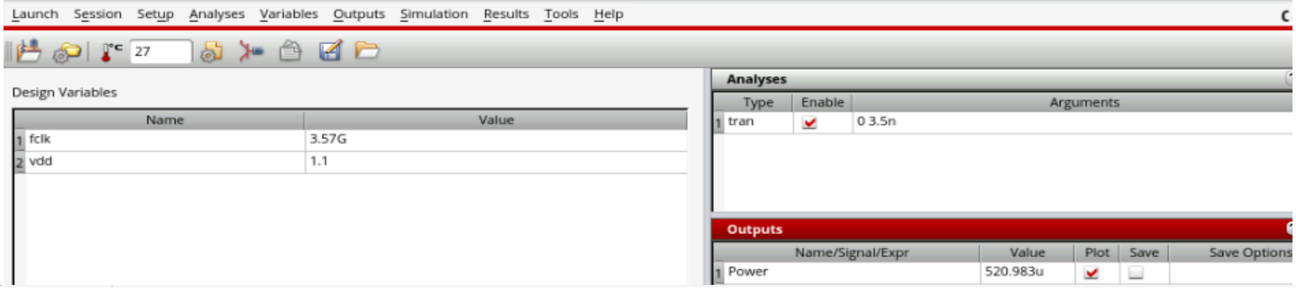
Analyses

	Type	Enable	Arguments
1	tran	☑	0 4.5n

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	Power	300.856u	☑	☐	





Design Variables

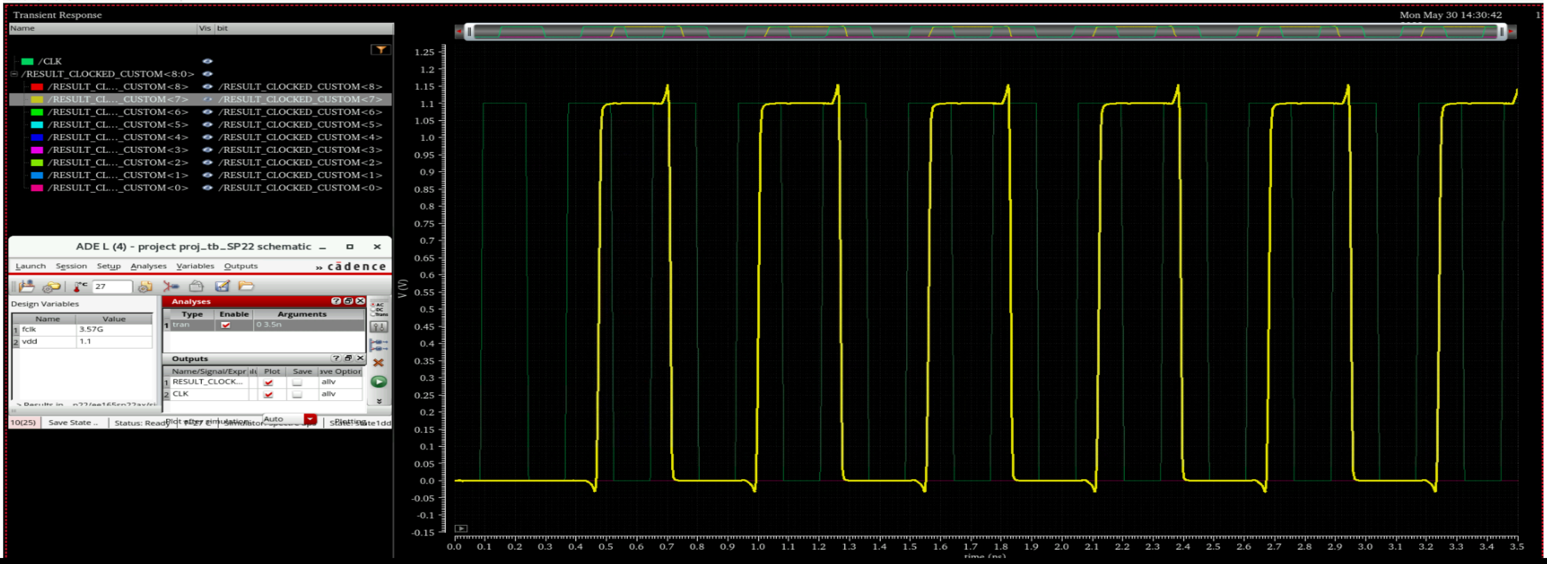
	Name	Value
1	fclk	3.57G
2	vdd	1.1

Analyses

	Type	Enable	Arguments
1	tran	☑	0 3.5n

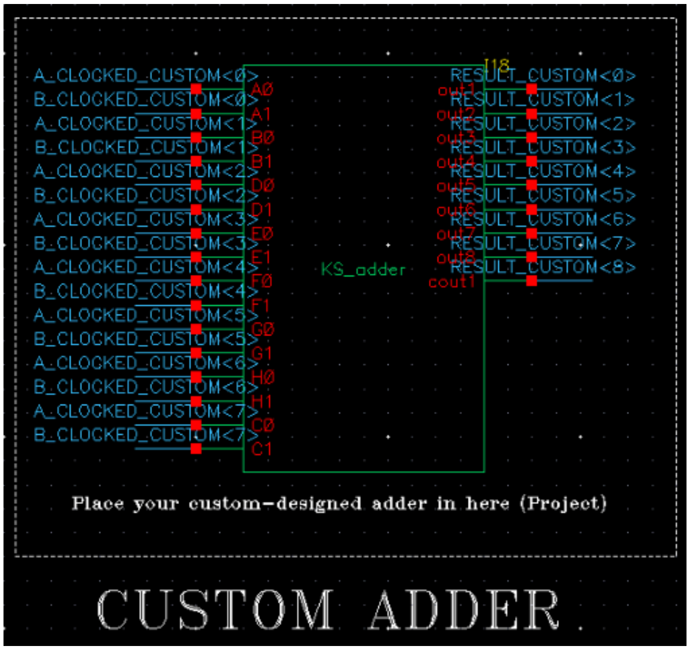
Outputs

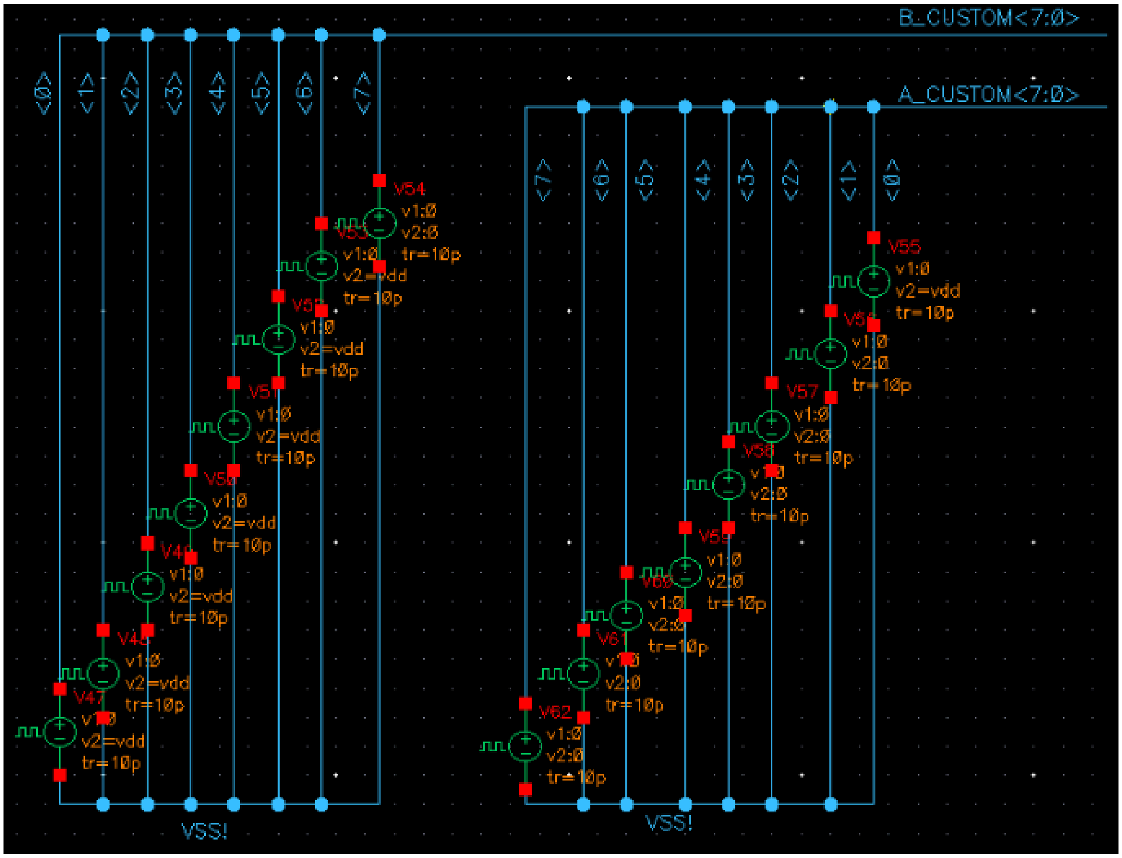
	Name/Signal/Expr	Value	Plot	Save	Save Options
1	Power	520.983u	☑	☐	

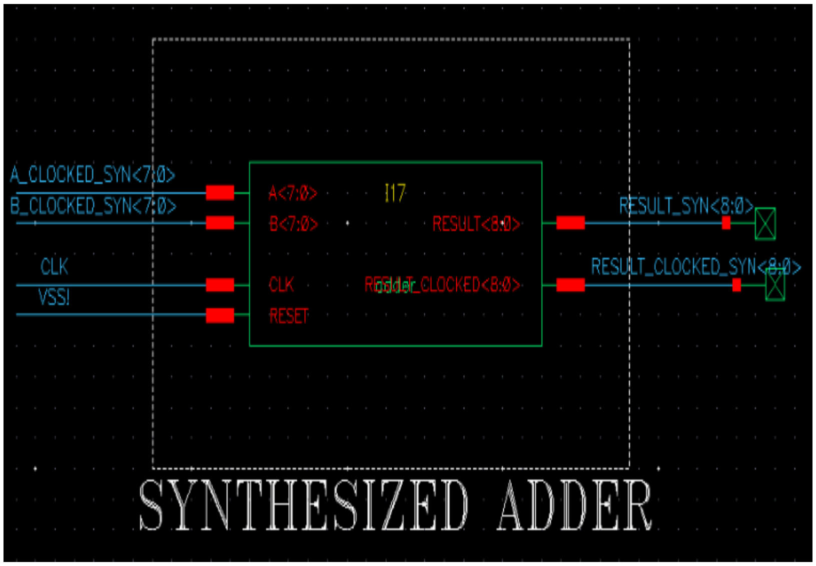


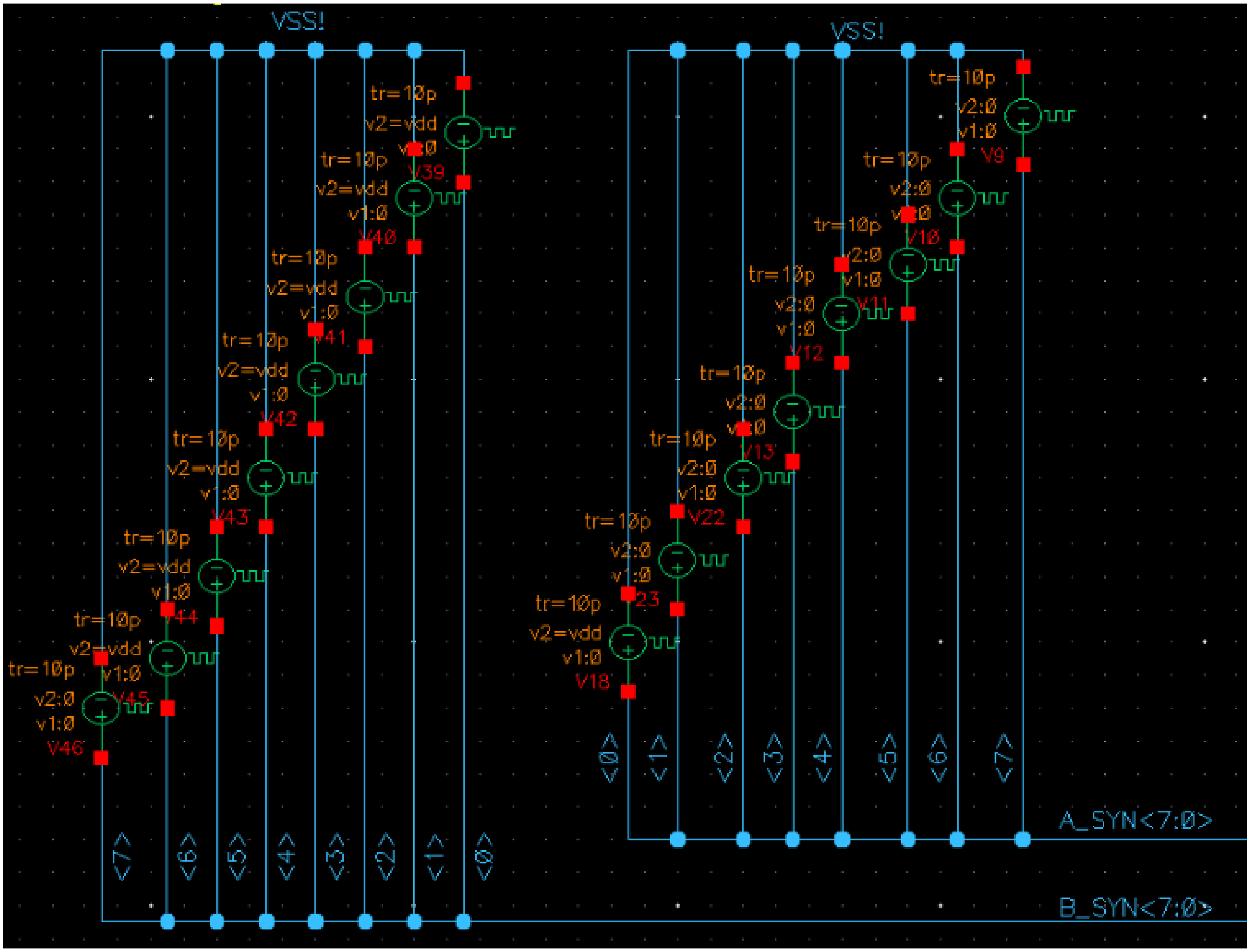
5. Table of Results

	Place+route schematic	Custom design Schematic	Place+route (optional)	Custom design extracted (optional)
Performance for VDD = 1.1V				
fmax	**2.2 GHz**	**3.57 GHz**	-	-
Power consumption @ fmax	**300.856u W**	**520.983u W**	-	-
Energy per operation @ fmax	**137 fJ**	**145.9 fJ**	-	-
Performance for VDD = 1.1V, fCLK = 1 GHz				
Power consumption @ 1 GHz	**137.1u W**	**141.996u W**	-	-
Energy per operation @ 1 GHz	**137.1 fJ**	**141.996 fJ**	-	-
Other important parameters				
Adder architecture	Ripple-carry	Kogge-Stone	Ripple-carry	Kogge-Stone
Core area	-	-	-	-
Critical input pair (i.e., A=?, B=?)	A = 00000001 B = 01111111	A = 00000001 B = 01111111	-	-
Transistor types used (e.g., VTL, VTG)	nmos1v, pmos1v	nmos1v, pmos1v	-	-




B_CUSTOM<7:0>
A_CUSTOM<7:0>
<0>
<1>
<2>
<3>
<4>
<5>
<6>
<7>
V47
V48
V49
V50
V51
V52
V53
V54
V55
V56
V57
V58
V59
V60
V61
V62
v1:0
v2=vdd
v2:0
tr=10p
VSS!
VSS!


A_CLOCKED_SYN<7:0>
B_CLOCKED_SYN<7:0>
CLK
VSS!
A<7:0>
B<7:0>
CLK
RESET
I17
adder
RESULT<8:0>
RESULT_CLOCKED<8:0>
RESULT_SYN<8:0>
RESULT_CLOCKED_SYN<8:0>
SYNTHESIZED ADDER


VSS!
VSS!
tr=10p
v2=vdd
v1:0
V39
V40
V41
V42
V43
V44
V45
V46
v2:0
V9
V10
V11
V12
V13
V22
23
V18
<7>
<6>
<5>
<4>
<3>
<2>
<1>
<0>
A_SYN<7:0>
B_SYN<7:0>