Construction of a 4x4 Systolic Array

Kyle E. Berger, Abhishek Damle, and Jack Retcher

Abstract - The systolic array was first designed in the 1970s. However, it did not see its full potential until the rise of neural networks and matrix-focused calculations. Systolic arrays provide large performance gains for matrix multiplication and other linear algebra operations used by neural networks. A 4x4 systolic array with 4 bits per value was constructed. The findings and results are outlined below.

I. INTRODUCTION

Systolic Arrays are arrangements of interconnected data processing units (DPUs) that process and store data independent of each other. Generally, each DPU's inputs and outputs are connected to multiple other DPUs and data gets processed as it flows synchronously between them. These DPUs allow Systolic Arrays to solve large problems by subdividing it into portions that can be computed concurrently. This parallelism results in systolic arrays' high throughput and allows them to perform large computations very quickly.

Matrix multiplication is one area where systolic arrays can be applied to greatly reduce computation time. The multiplication of N x N matrices through conventional means takes $O(N^3)$ time, while the same computation can be performed in O(N) time through utilizing a systolic array. The systolic array implementation of a 4x4 matrix multiplier is shown in figure 1.

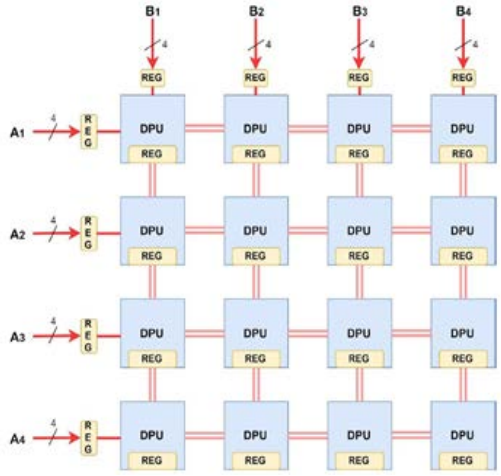


Figure 1: Systolic Array implementation of a 4x4 matrix multiplier[1]

Each DPU in figure 1 multiplies its inputs and store the results in an accumulator. In order to allow data to propagate synchronously, each DPU also stores its input in registers which are then used as inputs for other DPUs. Staggering the inputs as shown in figure 2 allows for the correct propagation of data and enables the systolic array from figure 1 to multiply two 4x4 matrices together.

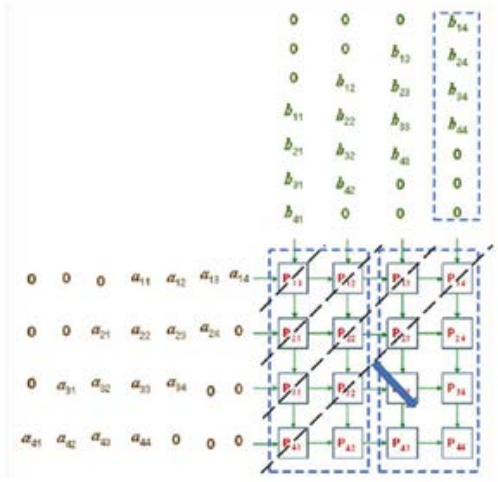


Figure 2: Data propagation through systolic array[1]

II. APPROACH

This project was very large, and therefore needed to be completed in sections. The first layout that was completed was the multiplier. There were two 4-bit inputs that produced an 8 bit result. The multiplier used is known as a Carry Save Adder array. It consists of an array of 2 submodules: a typical ripple carry full adder, and a modified ripple carry full adder that will be described as a CSA unit.

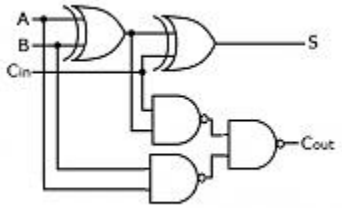


Figure 3. The gates used in the full adder.

Figure 3 displays the gates used in the full adder unit. NAND gates were used instead of AND and NOR to reduce size. The XOR gates were made using transmission gates, also to reduce size (the circuit diagram can be found in the addendum below the XOR layout). The design for the CSA unit, shown in Figure 4, simply added a NAND gate and an inverter to one of the addend inputs. The Virtuoso schematics and layout for both items can be found in the addendum.

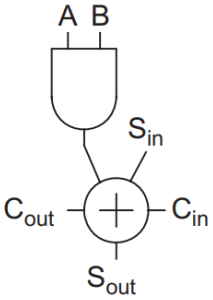


Figure 4. CSA unit circuit.[2]

These submodules were laid out in a cascading array consisting of 16 CSA units and 4 ripple carry full adders. Each bit of the 4 bit inputs were connected to 4 of the CSA units, and each carry out was connected to the next carry in. The initial carry inputs were grounded. The last 4 CSA units were connected to a 4 bit ripple carry adder, which computed the last 4 bits of the result. The 8 bit result was read from each sum output of the bottom CSA units and full adders. See figure 6 for a diagram.

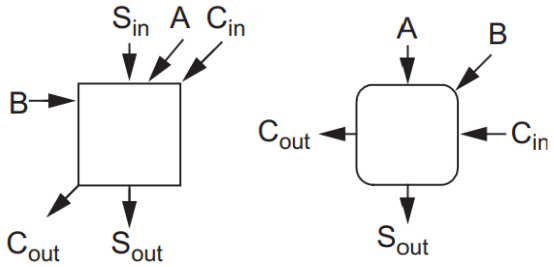


Figure 5: CSA and Full Adder Units.[2]

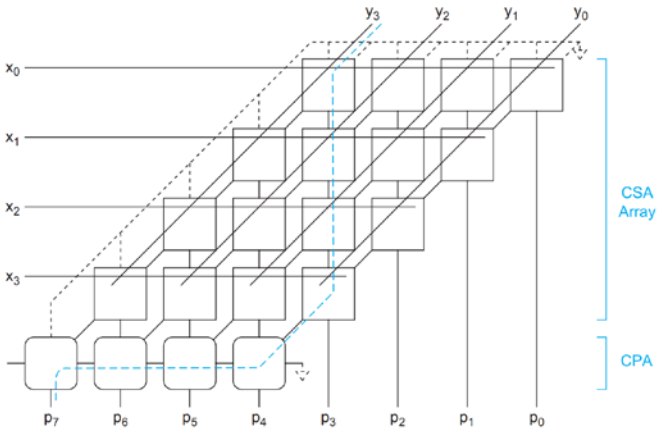


Figure 6: Multiplier design.[2]

The final design was laid out using Virtuoso. The layouts can be found in the addendum. In order to make the designs as small as possible, the layouts were modeled after the units shown in figure 5. This made it trivial to connect the submodule layouts in the final multiplier design. The layouts were also kept within a rectangular shape so that there would be no issues with metal layers being too close.

Once the multiplier layout and schematic were completed, both were tested. Two separate testbenches were created where each applied a different load to the multiplier. The first testbench fed each output into an inverter while the second tested the multiplier driving the twelve bit adder.

When testing the design using a fanout of 1, the multiplier was successful. After testing

under a 12 bit adder load though, it became clear that the XOR transmission gates in the multiplier weakened the signal too much. To solve this, a buffer was added at every multiplier output in the testbench. After more tests, it was clear that the buffers fixed the problem.

After the multiplier was completed, design and layout of the DPU began. Both designs can be found in the addendum.

The DPU was constructed from a 12 bit register, 12 bit adder, multiplier, and two 4 bit registers. After running prelayout simulations, it was clear that the registers were not able to drive large loads, so a buffer was attached to the output of each of the D flip flops in the registers. The spacing of the D flip flops in the registers correlated with the placements of the inputs and output ports in the adder and simplified the connection between the register and the adder. While this allowed the 12 bit adder and 12 bit register to occupy a minimal amount of space, their dimensions were mismatched with those of the multiplier (see layout of DPU in addendum). This led to a large amount of wasted space in the DPU layout and significantly increased the area of the systolic array. The DPU was shaped like an L, so theoretically, the area used by the final design could be reduced if each DPU is oriented differently.

With the schematic/layout of the DPU completed, tests were performed on it to ensure proper operation. The testbench tested the outputs against a fanout of 1 inverter and were initially incorrect. However, after initializing the registers to zero, the results came out as expected. Two waveforms can be seen in the Addendum that show 1) the register storing the correct results of the multiplier and 2) the register contents being added to the results of the multiplier.

Once the DPU was verified, construction of the final design began. Again, both the schematic and layout can be found in the addendum. The systolic array was relatively simple. It was composed of 16 DPU instances, so the only work that needed to be done was connecting each instance and placing the I/O pins. Connecting each pin was very simple, as the outputs of each DPU aligned well with the inputs of the next DPU. The biggest challenge was placing each pin. There were 224 pins that needed to be placed manually, which took a significant amount of time. The final area was about 22,500 μm^2. The rest of the areas can be found in the table below.

Systolic Array	22,254 μm^2
DPU	1354.6 μm^2
12 bit register	69.2 μm^2
12 bit adder	78.4 μm^2
CSA unit/Full Adder	8.2 μm^2

III. EXPERIMENTS

A similar testing methodology was applied to the final design using a test bench. A screenshot of the design used to simulate the final circuit, along with the benches used to simulate the submodules, is in the addendum. A fanout of 1 inverter was used on every output, with each inverter connected to a 100fF capacitor. A fanout of only one was used instead of a higher fanout for a few reasons. First, it reduced the complexity of testing. In addition, it also was difficult to imagine a situation in which the output would be more than one logic gate. This is dedicated hardware, so the output would likely only be a single set of registers.

IV. RESULTS

Once the testbench was created, an initial simulation of multiplying a 4x4 matrix of zero by a 4x4 matrix of zero was conducted. After viewing the simulation graph, featured in the

addendum, the systolic array was deemed working for that set of inputs. This test took approximately 45 minutes to complete. When actual values were used, the simulation still had not completed after an hour. As a result, additional tests were not conducted. The test results from the individual DPUs can be found in the addendum. While normally, it would be wrong to assume the final design would work based only on the subcircuits working; given that every output is buffered and the difficulty in testing the full systolic array, we were forced to assume the full array was working.

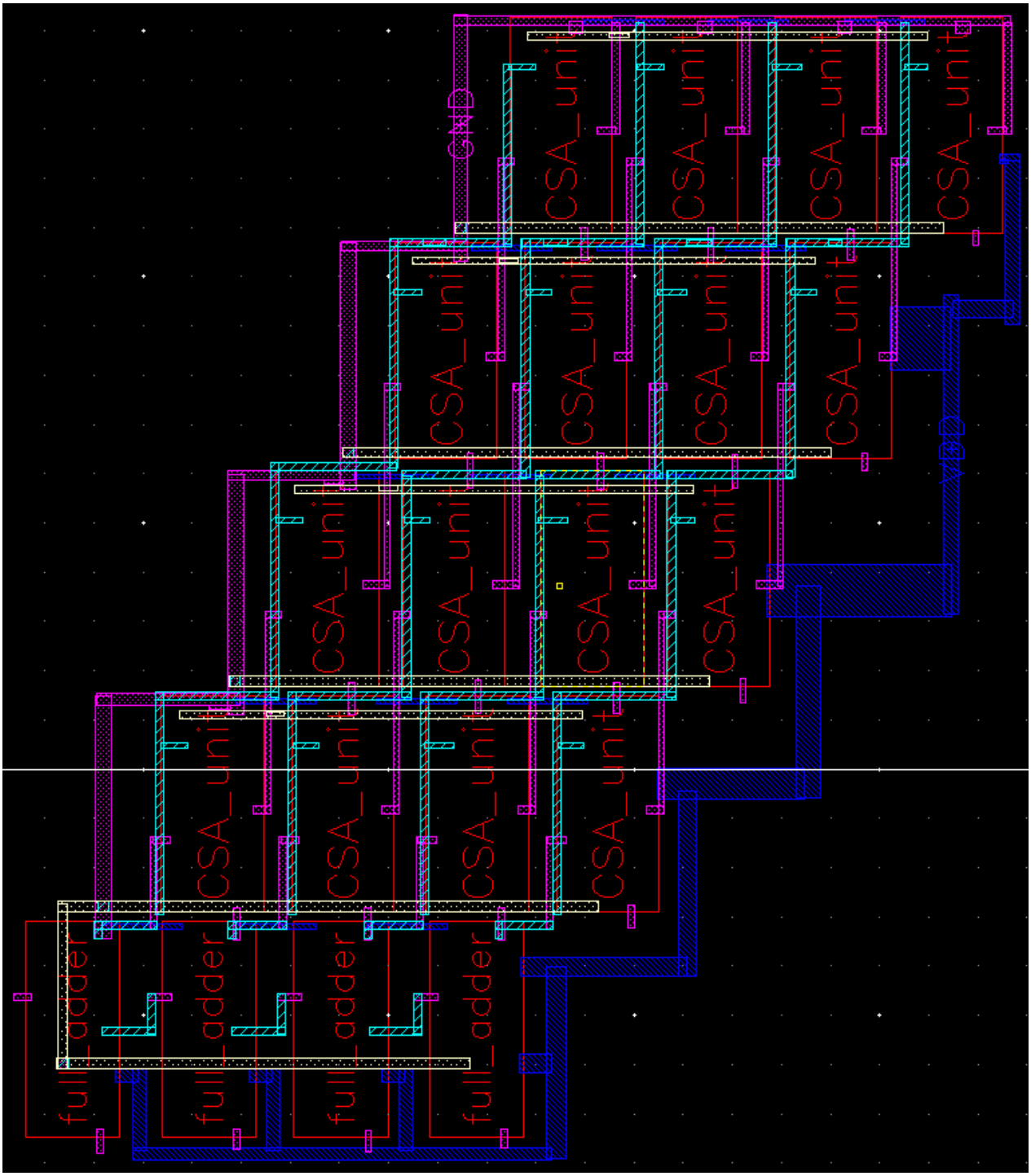
V. SUMMARY & CONCLUSIONS

Overall, this was an excellent learning experience. It is very clear that VLSI is challenging yet exciting work. This project was difficult, but it was made easier by splitting into submodules and testing every stage. This project also emphasized the importance of not only testing every module, but also testing each module under an appropriate load. If tests with full loads had not been conducted, the debugging process would have been significantly longer.

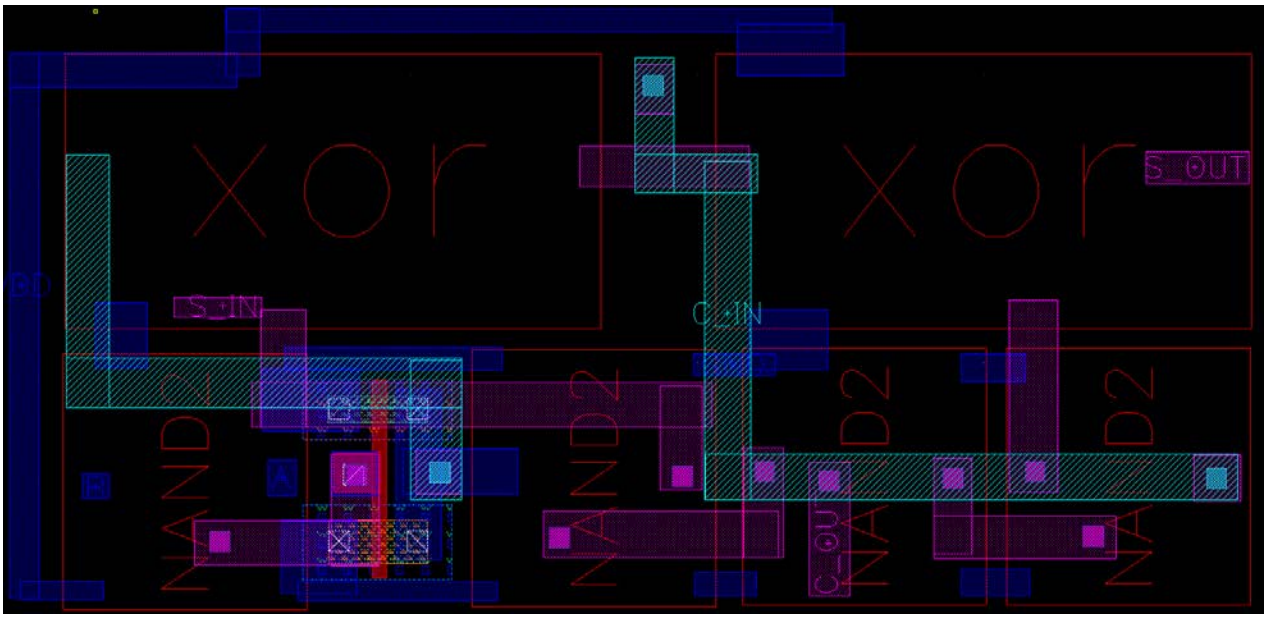
REFERENCES

[1] V. Gan, *Final Project: Systolic Array Matrix Multiplier*.
[2] N. H. E. Weste and D. M. Harris, *CMOS VLSI design: a circuits and systems perspective*, 4th ed. Boston, MA: Addison-Wesley, 2011.

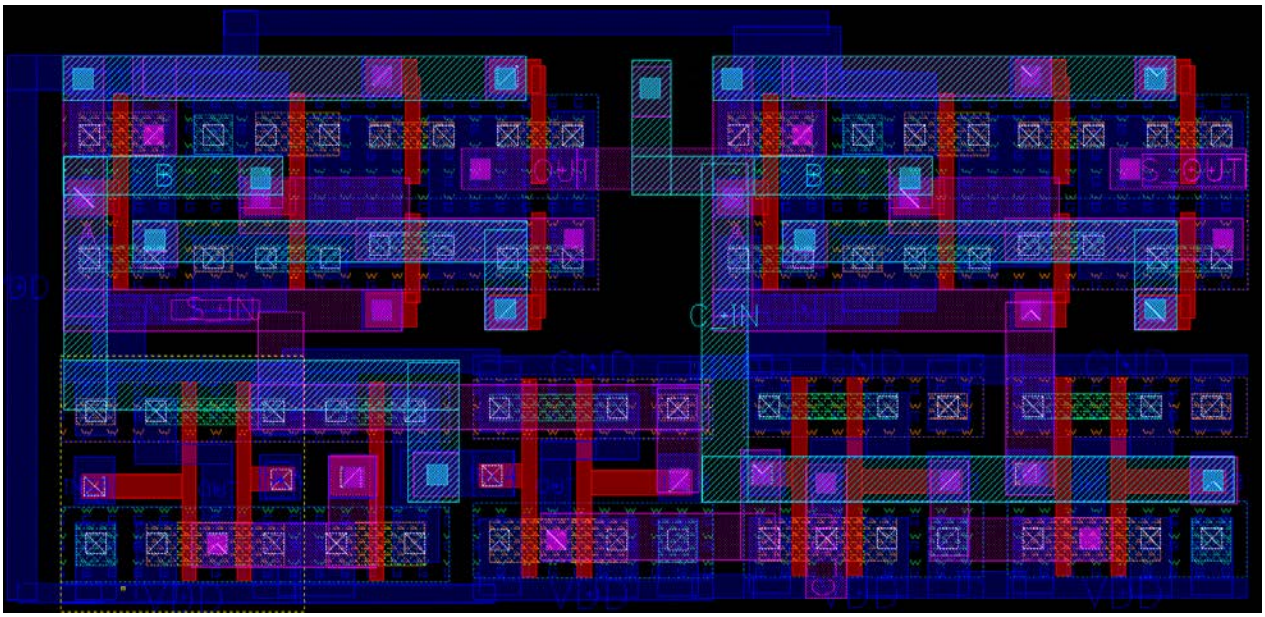
Addendum



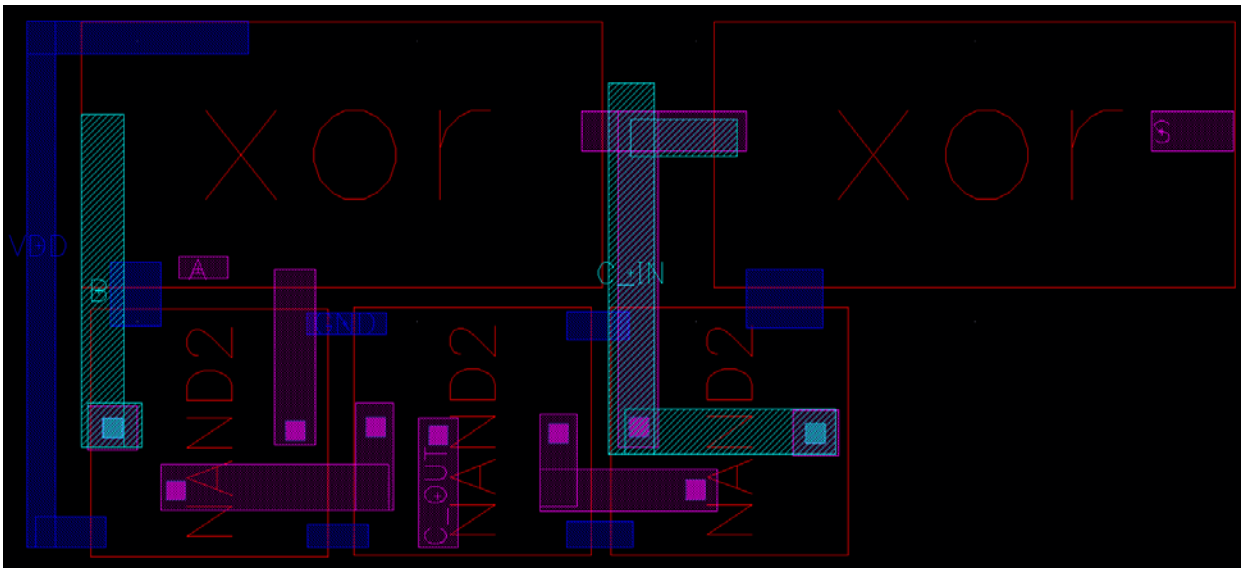
Multiplier Layout



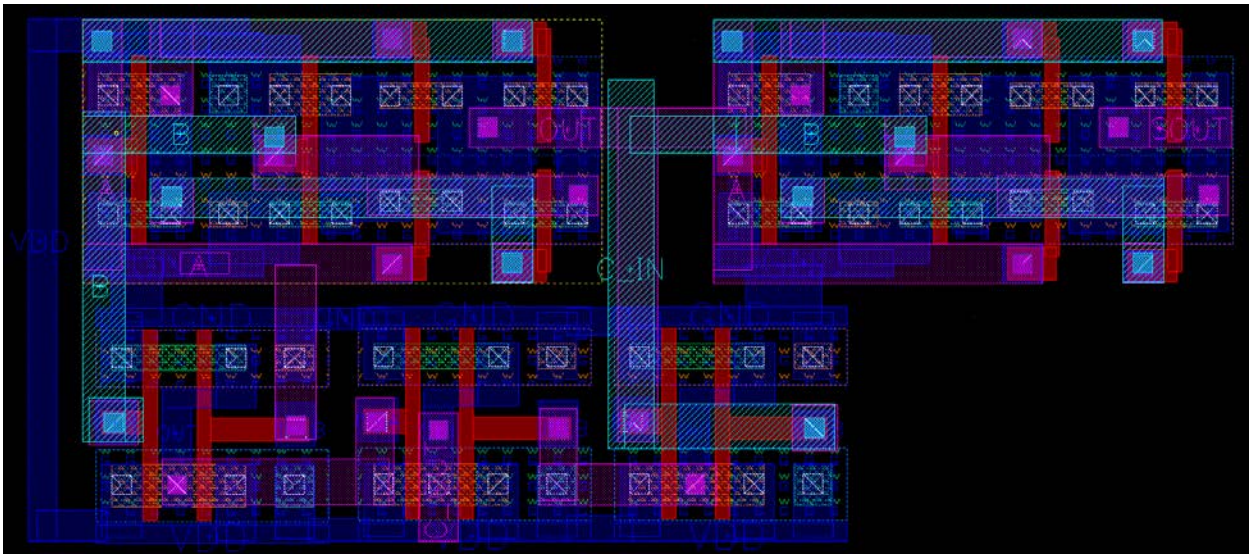
CSA Unit Layout



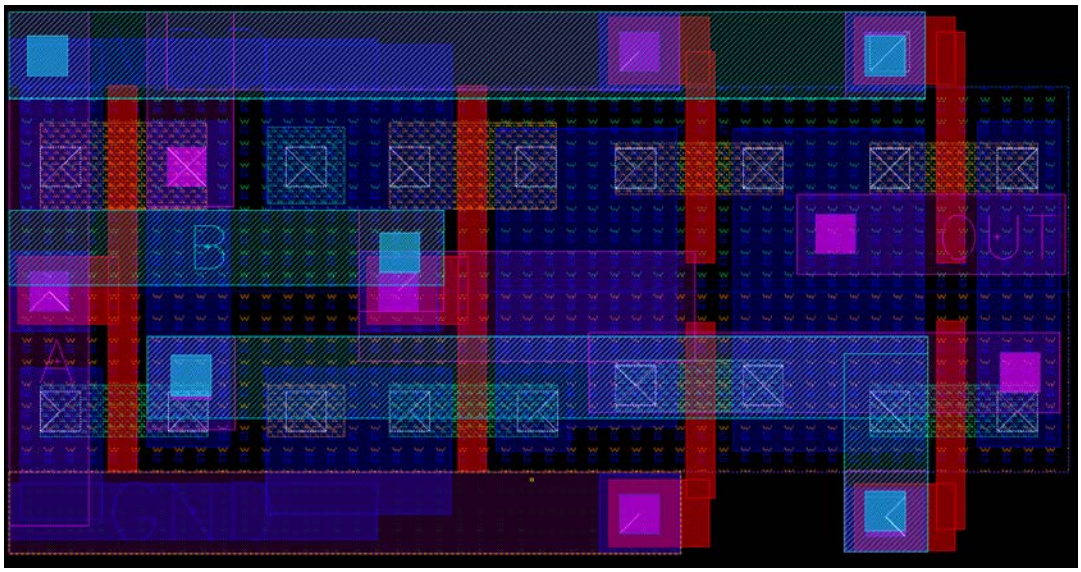
CSA Unit Layout (X Ray)

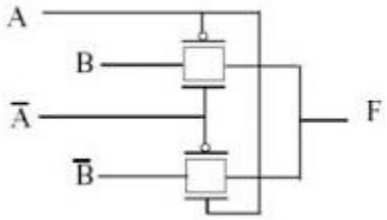


Full Adder Layout

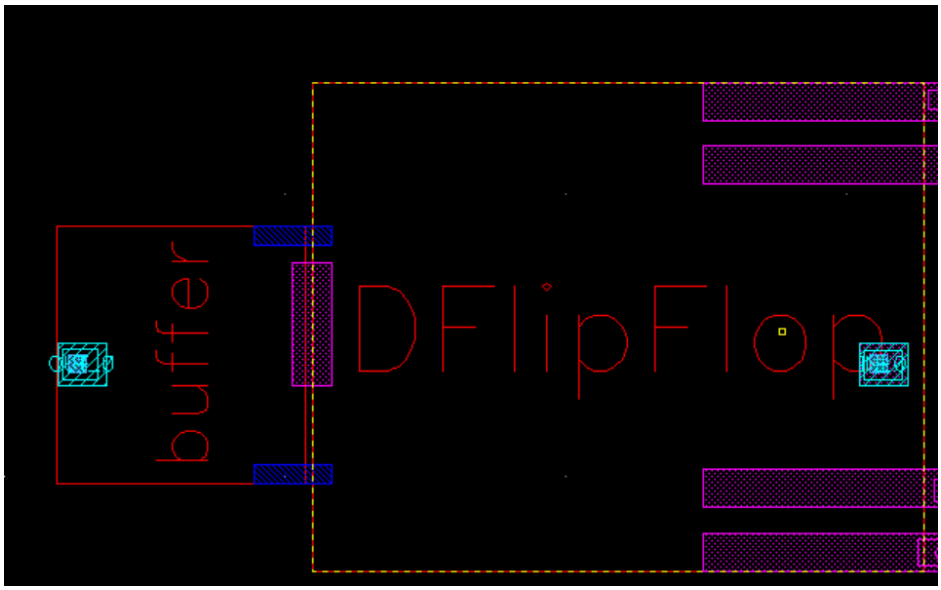


Full Adder Layout (X Ray)

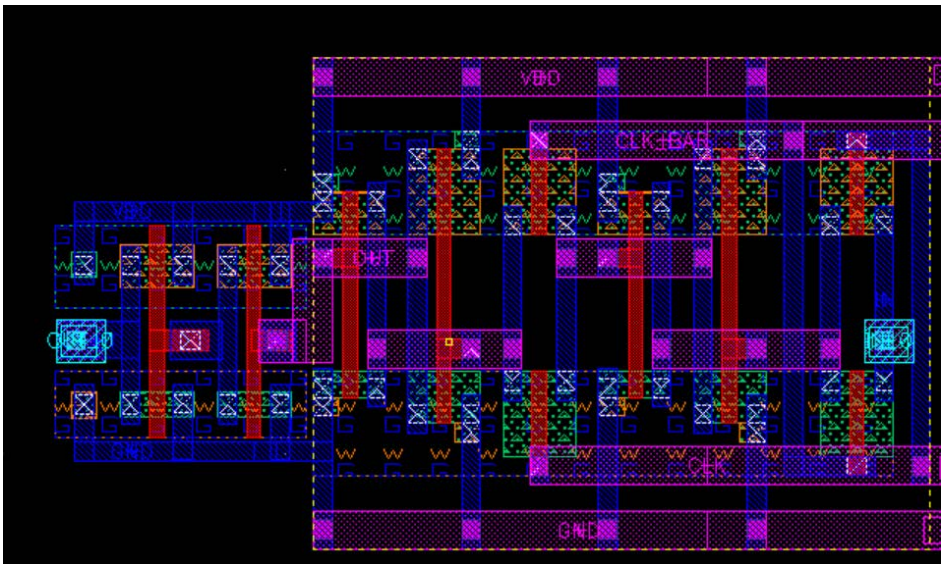




XOR Layout, and the circuit diagram it is based on

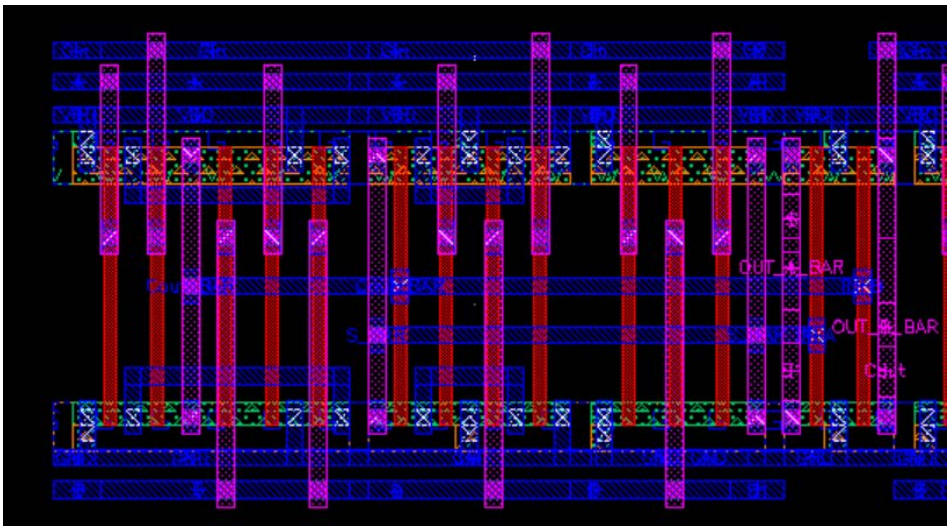


Register Layout(1 bit shown)

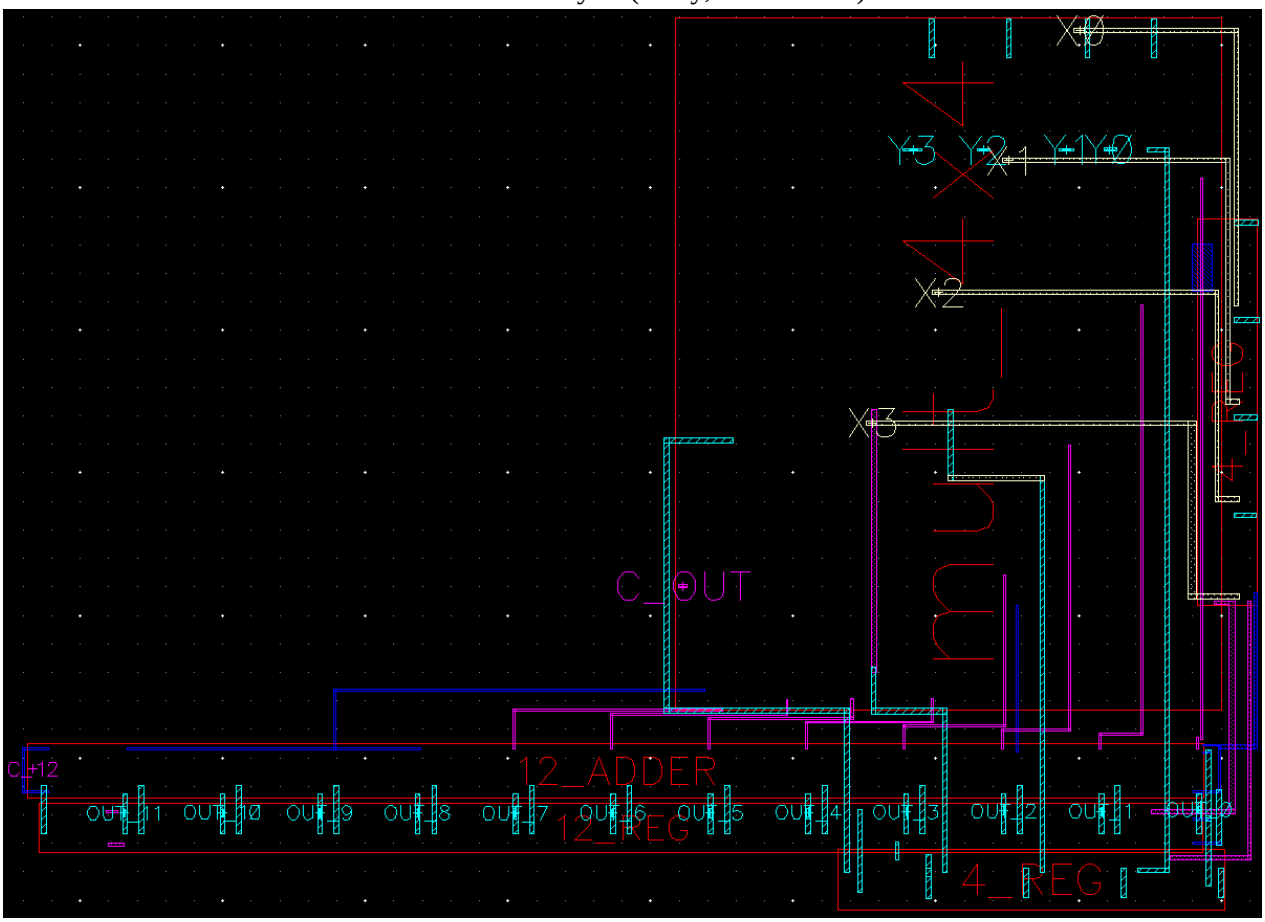


Register Layout(X ray, 1 bit shown)

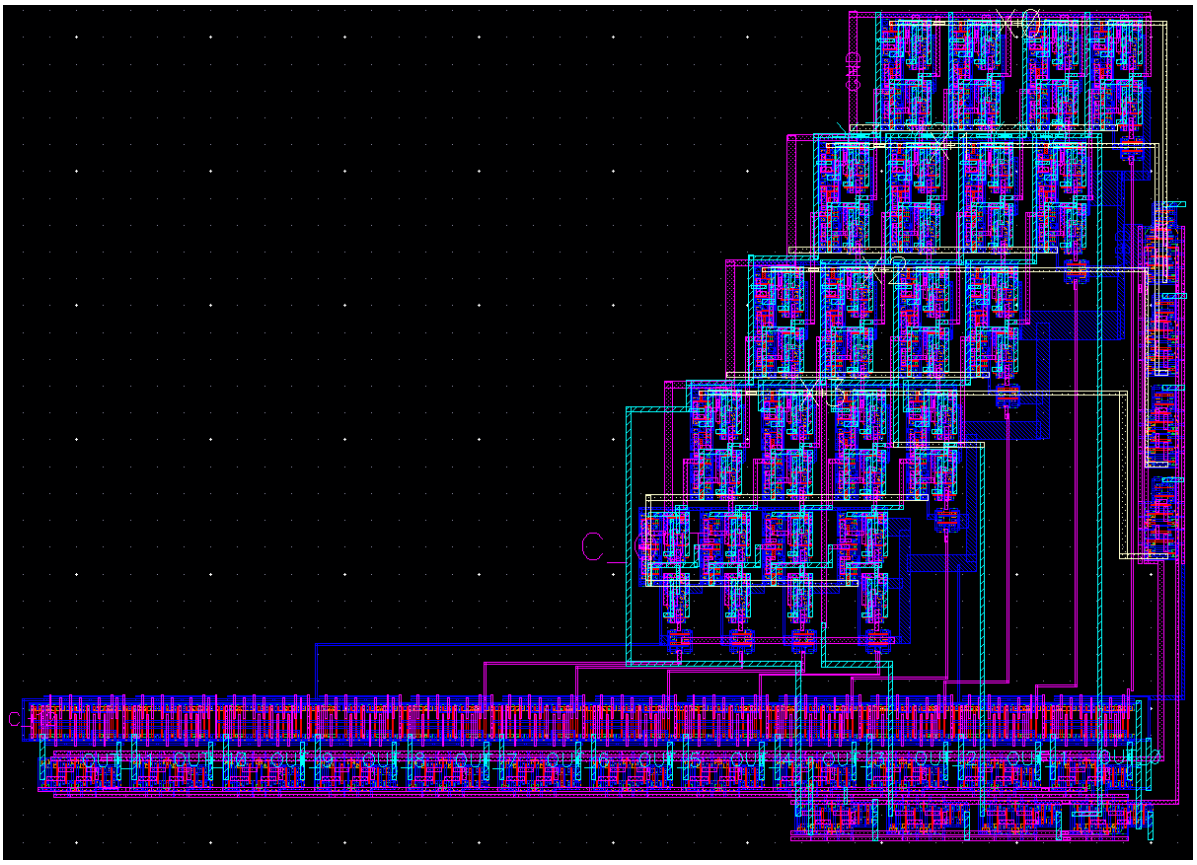
12-bit Adder Layout(1 bit shown)



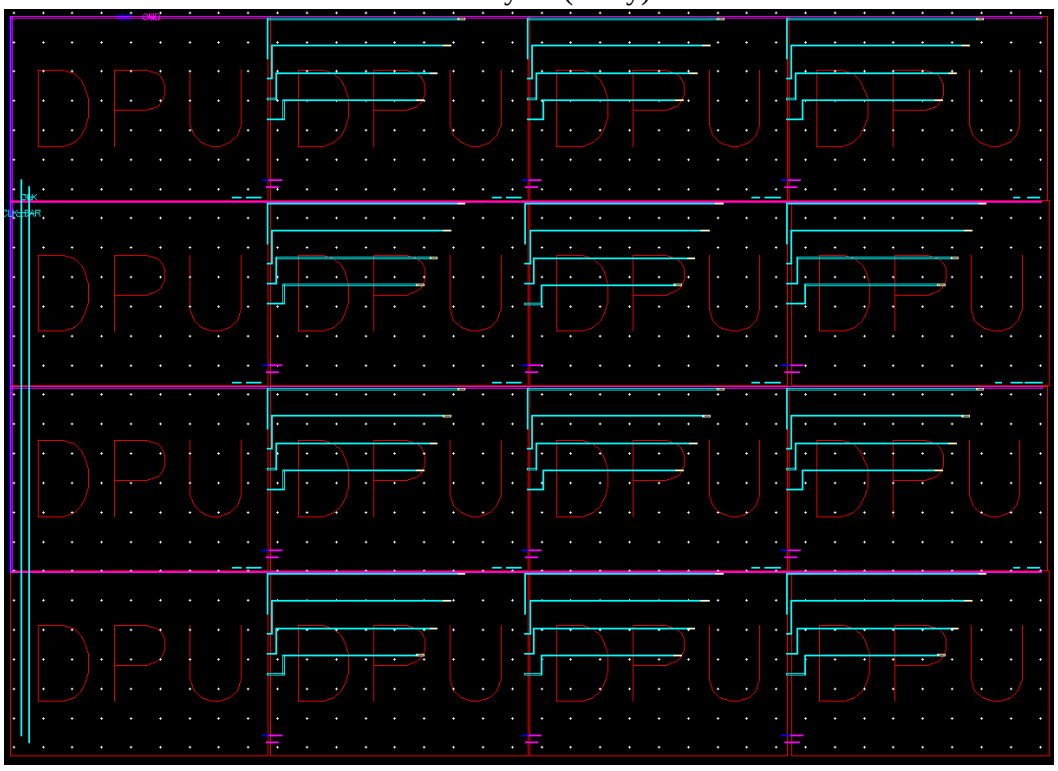
12-bit Adder Layout(X ray, 1 bit shown)



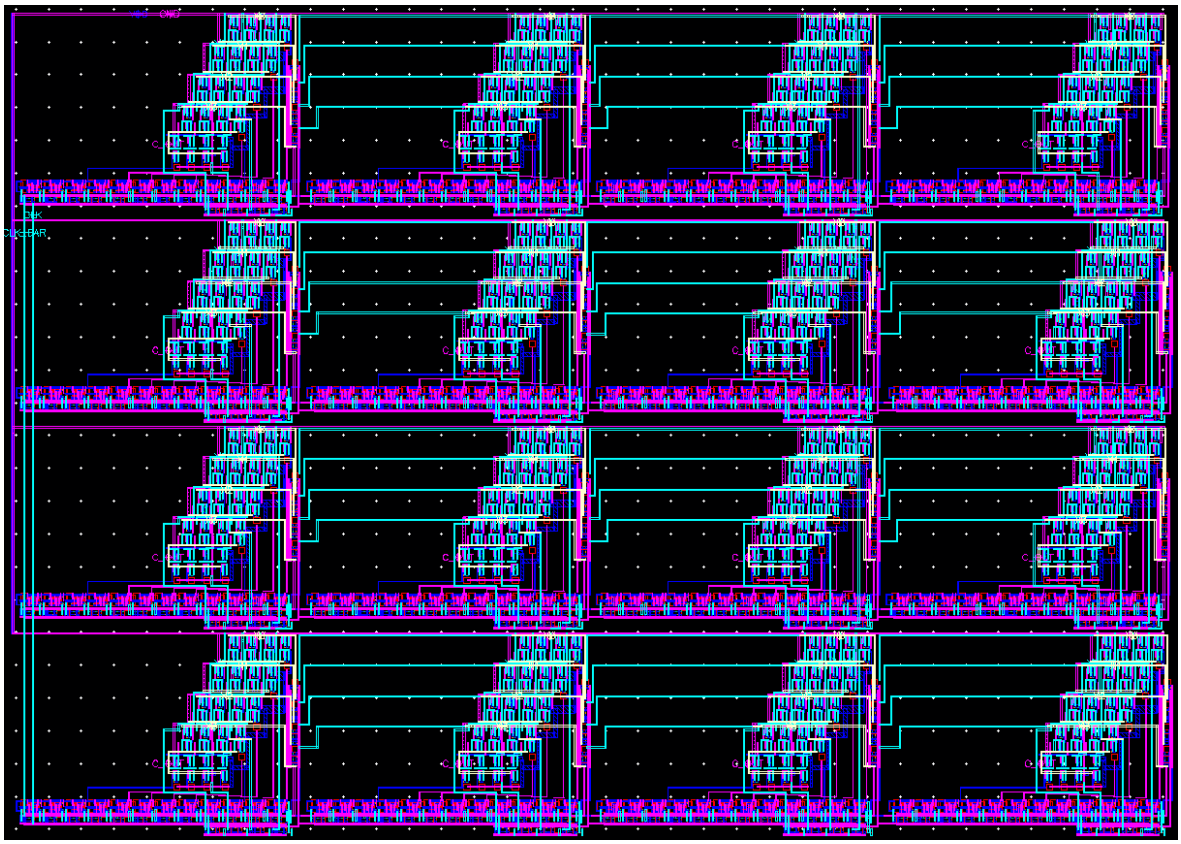
DPU Layout



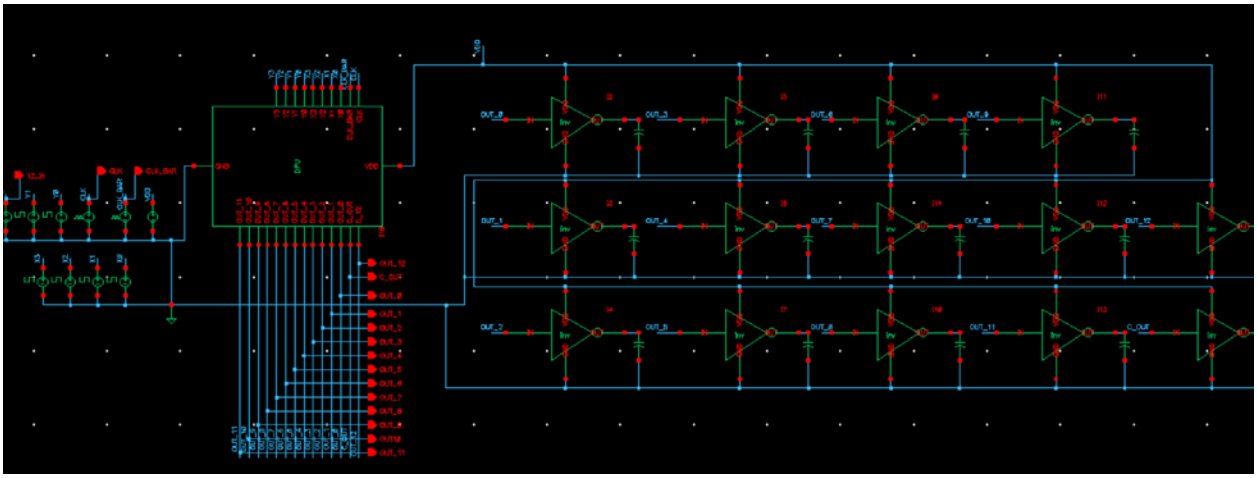
DPU Layout (X-ray)



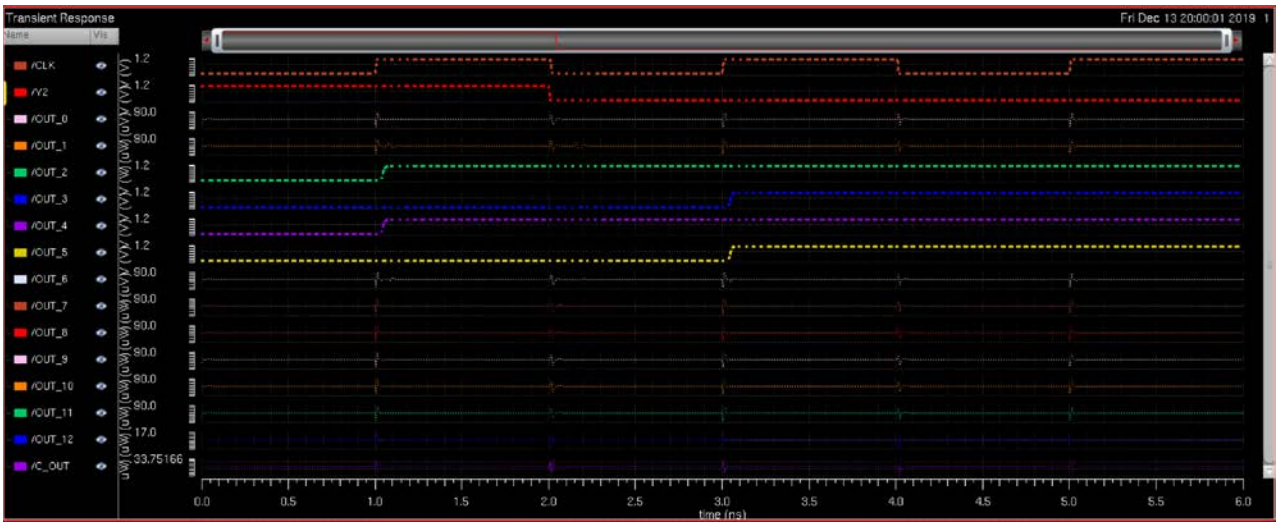
Systolic Array Layout



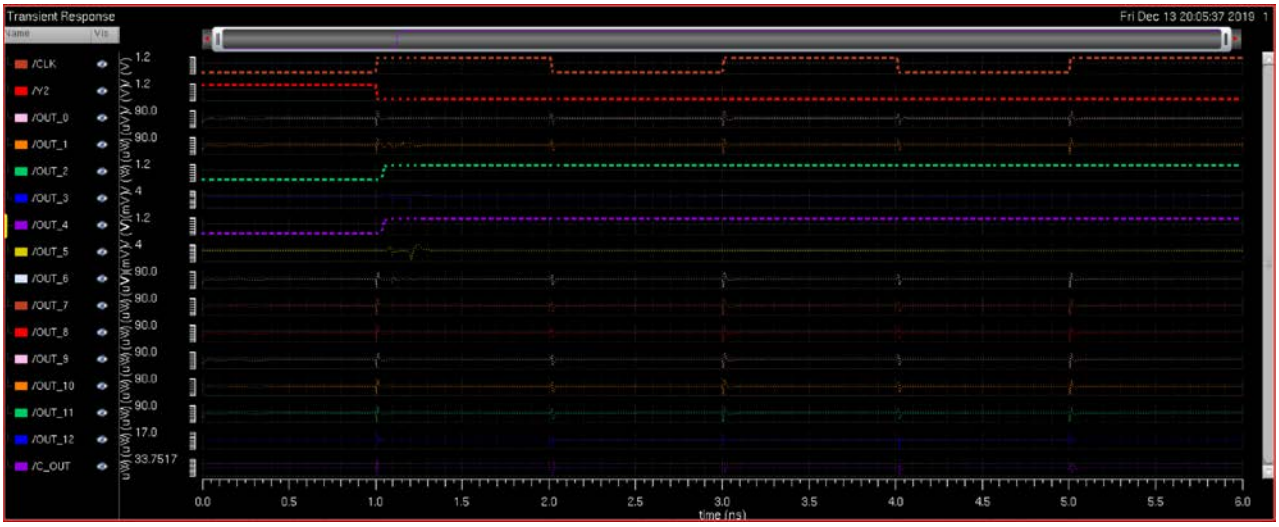
Systolic Array Layout (X-ray)



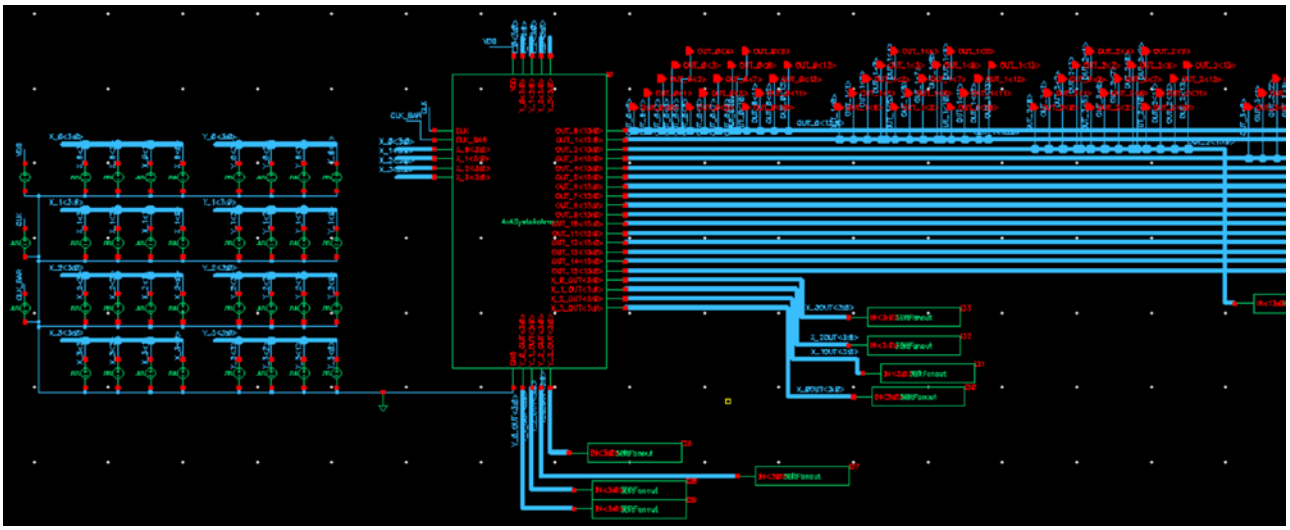
DPU Test Bench



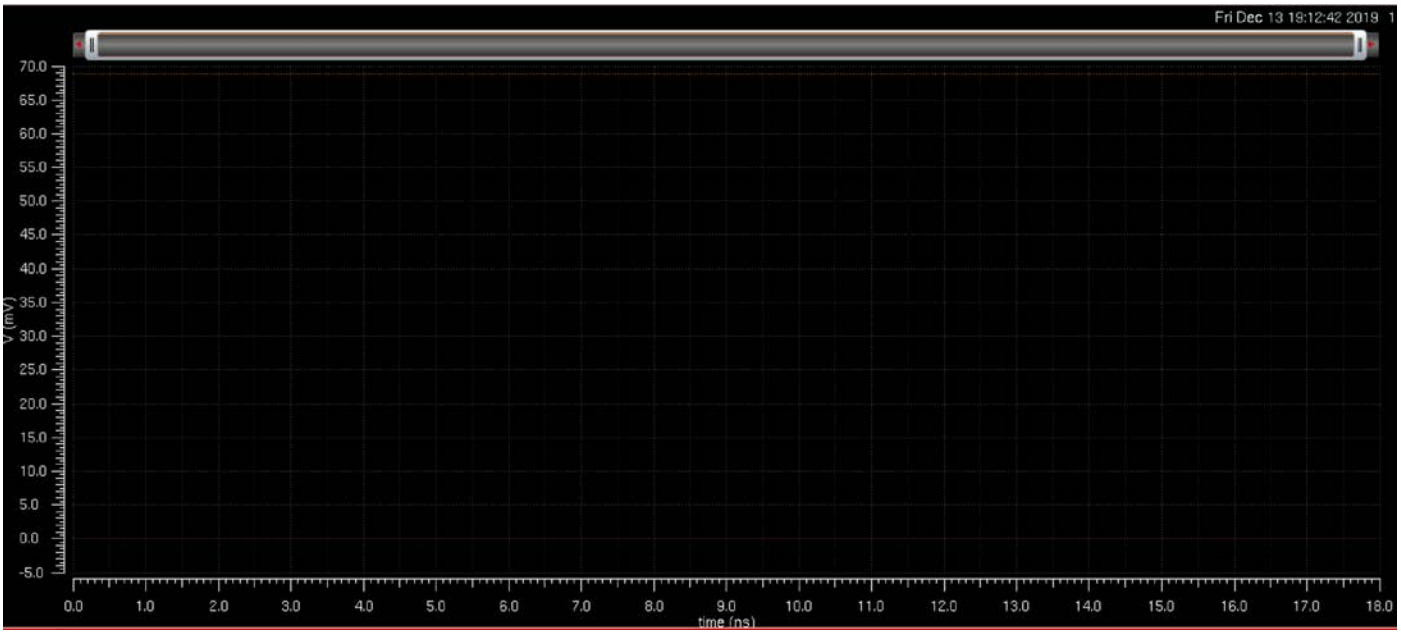
DPU Waveform with 0100 and 0101 as inputs over two clock cycles



DPU Waveform with 0100 and 0101 as inputs over one clock cycle



Systolic Array Test Bench (part of it)



Multiplying a 4x4 matrix of zero by a 4x4 matrix of zero in the final systolic array