

## Computer-Aided VLSI System Design

### Final Project Report

**Team ID:** team51

**Student ID:** R12K41025、R12943137


**Student Name:** 杜冠廷、陳泓瑋

### Algorithm

In general, there are three well-known algorithms for performing QR decomposition, namely Householder, Gram-Schmidt, and Givens Rotation. Among them, Givens Rotation is considered more suitable, as it can be implemented using Cordic. This makes it relatively preferable compared to the other two algorithms. The implementation steps of the algorithm are as follows:


#### Step1: Rotate Imaginary Part to Real Part

Due to the complex nature of the input matrix  $H$ , as opposed to real matrices, it is necessary to convert the imaginary elements of each row to their real components before rotating the row to align its length with the main diagonal. Simultaneously, the other elements in the same column should be rotated by the same angle to maintain consistency in the transformation process.



$$H_{1j} = a + jb$$

Figure 1 Rotate complex part to real part

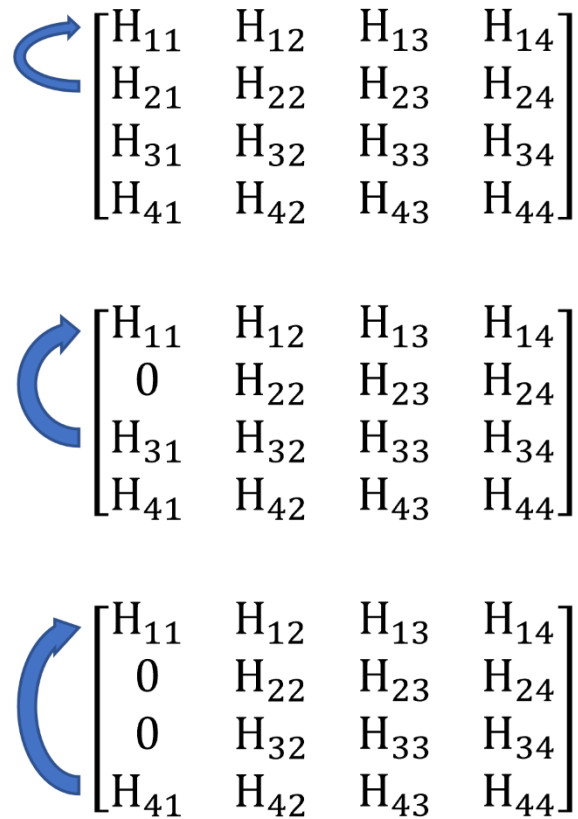


$$\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix}$$

Figure 2 Rotate all complex part to real part in column

**Step2: Rotate to Main diagonal**

Rotate all elements below the main diagonal to align with the main diagonal, and ensure that the remaining elements in the same column as the rotated elements undergo the same rotation.



$$\begin{aligned}
 & \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix} \\
 & \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ 0 & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix} \\
 & \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ 0 & H_{22} & H_{23} & H_{24} \\ 0 & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix}
 \end{aligned}$$

Figure 3 Rotate to Main diagonal

**Step3: Repeat Step1 and Step2**

Repeat steps 1 and 2 iteratively until the matrix transforms into the configuration illustrated in Figure 4.

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & H'_{14} \\ 0 & R_{22} & R_{23} & H'_{24} \\ 0 & 0 & R_{33} & H'_{34} \\ 0 & 0 & 0 & H'_{44} \end{bmatrix}$$

Figure 4 Finish Column 2 and 3

**Step4: Rotate  $H_{44}$  to Real Value**

Imitate step 1 to transform  $H_{44}$  into real numbers.

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ 0 & R_{22} & R_{23} & R_{24} \\ 0 & 0 & R_{33} & R_{34} \\ 0 & 0 & 0 & R_{44} \end{bmatrix}$$

Figure 5 Finish Rotate

As this assignment requires obtaining both the upper triangular matrix and the corresponding y matrix, reshape the input as shown in Figure 6, and the output as shown in Figure 7. It is important to note that after step 4, the output can be obtained, and there is no need to set the imaginary part of y to 0.

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} & Y_{11} \\ H_{21} & H_{22} & H_{23} & H_{24} & Y_{12} \\ H_{31} & H_{32} & H_{33} & H_{34} & Y_{13} \\ H_{41} & H_{42} & H_{43} & H_{44} & Y_{14} \end{bmatrix}$$

Figure 6 The transformed input after the changes

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} & \widehat{Y_{11}} \\ 0 & R_{22} & R_{23} & R_{24} & \widehat{Y_{12}} \\ 0 & 0 & R_{33} & R_{34} & \widehat{Y_{13}} \\ 0 & 0 & 0 & R_{44} & \widehat{Y_{14}} \end{bmatrix}$$

Figure 7 The transformed output after the changes

## Design Flow

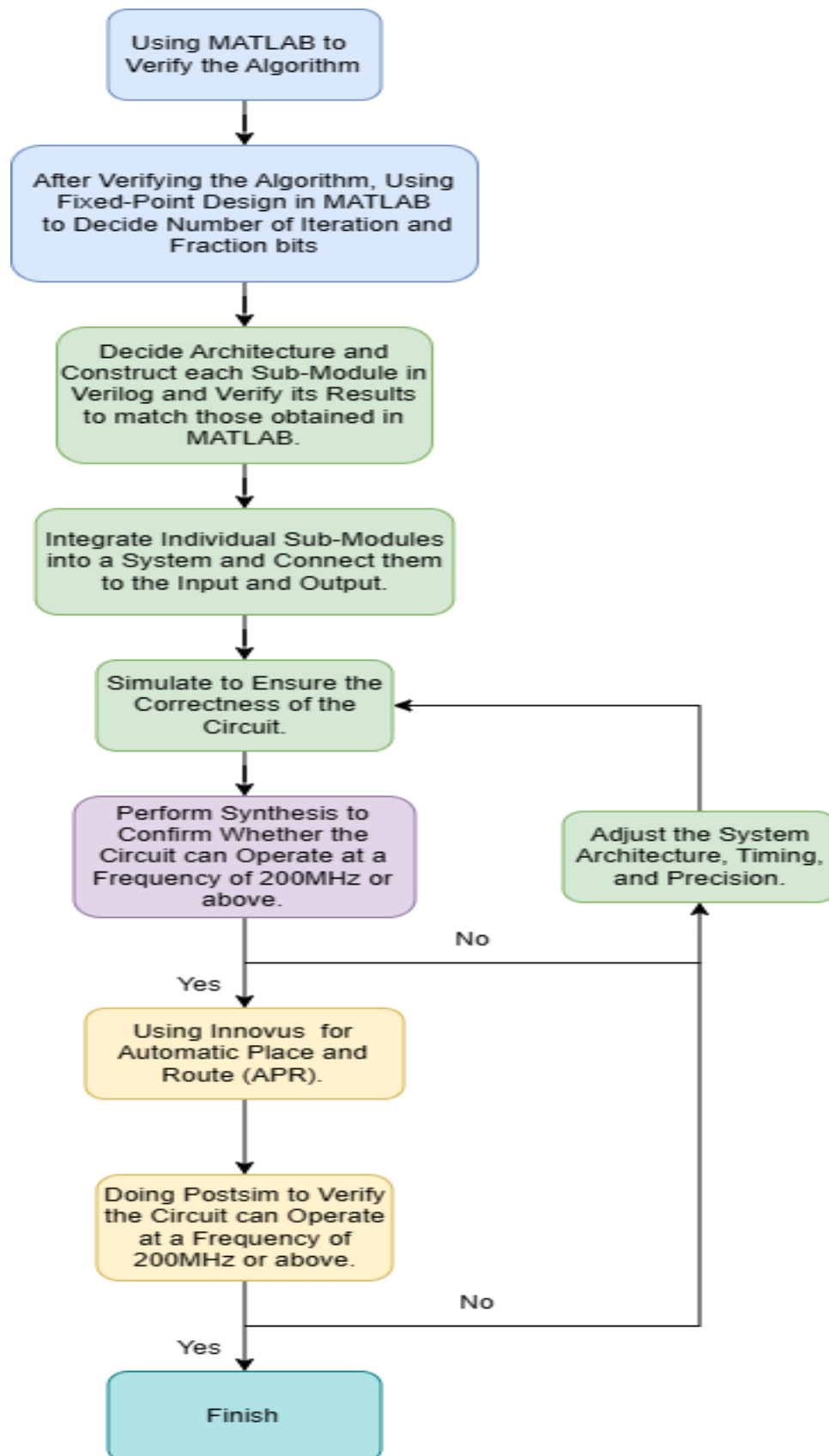


Figure 8 Design Flow

## Hardware Architecture

To achieve low-power design, our hardware architecture will focus on sharing arithmetic units and storage units, with the goal of minimizing the frequency of data reads and writes from SRAM.

### 1. Shared Arithmetic Units:

To reduce the required hardware resources, we need to share arithmetic units that perform the same operations. By performing two projections based on the Dependence Graph shown in Figure 9 to 12, the result after the first projection along the  $j$ -direction is depicted in Figure 13, and the result after the second projection along the  $w$ -direction is shown in Figure 14. Additionally, it is essential to note that the step of rotating the imaginary part to the real part is performed only when  $K=4$ .

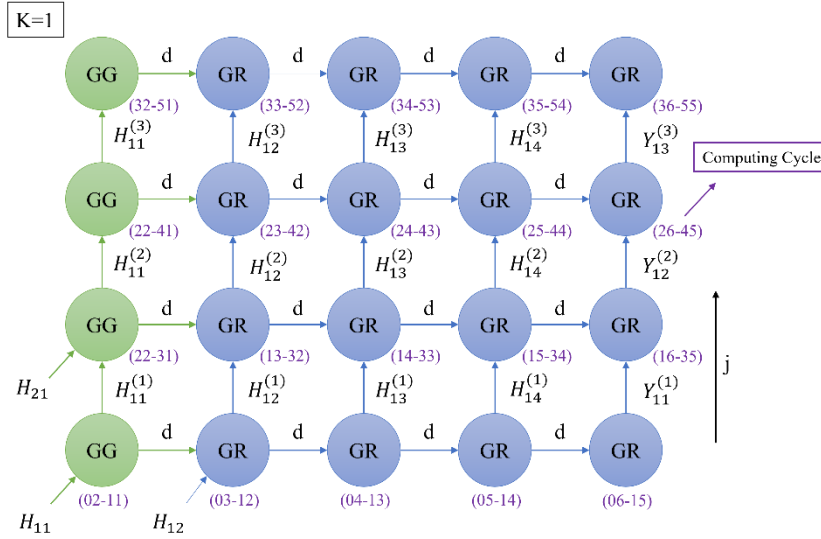


Figure 9 Dependence Graph ( $K=1$ )

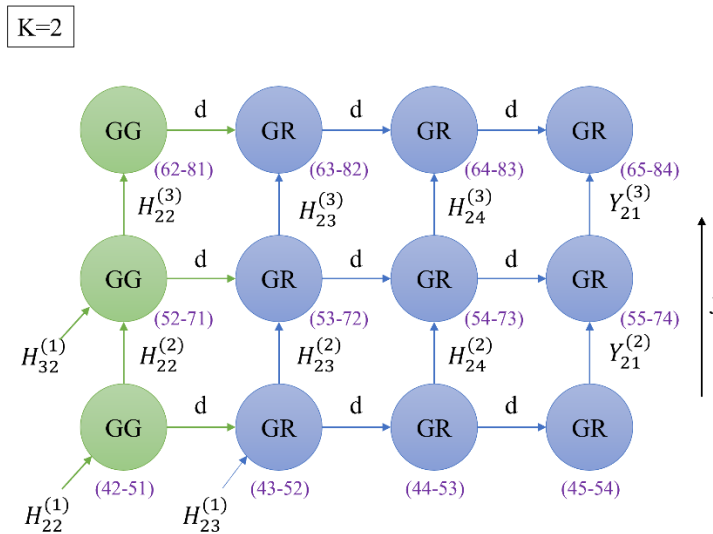


Figure 10 Dependence Graph ( $K=2$ )

K=3

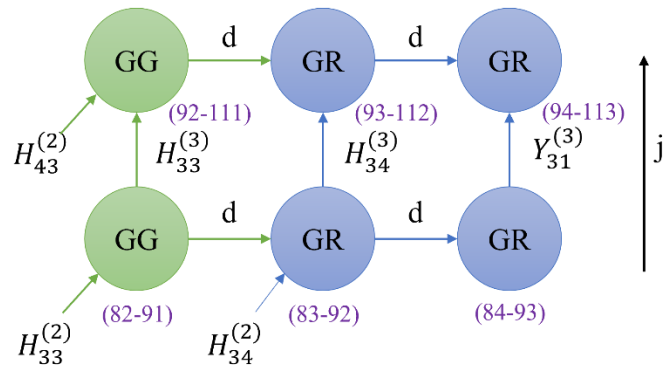


Figure 11 Dependence Graph (K=3)

K=4

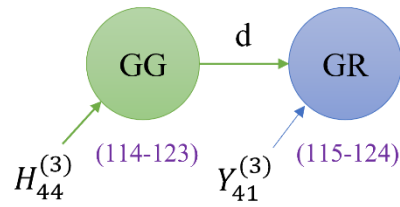


Figure 12 Dependence Graph (K=4)

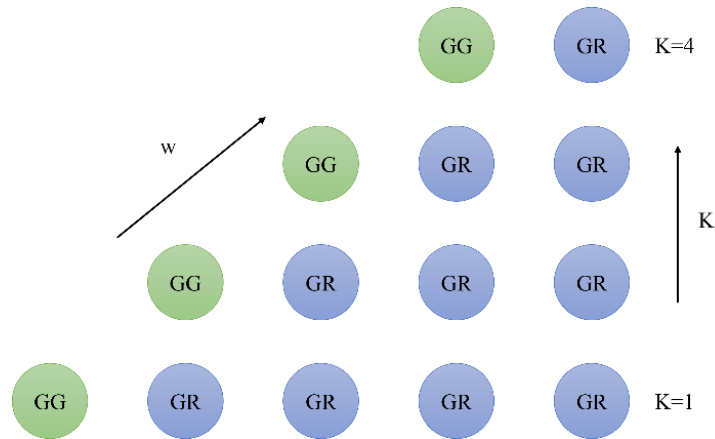


Figure 13 After First Projection



Figure 14 After Second Projection

## 2. Sub-Module Architecture

There are two sub-modules, namely GG (Givens Generator) and GR (Givens Rotation). GG consists of two vectoring mode Cordic units and a set of registers, as shown in Figure 15. GR, on the other hand, comprises three Rotation Mode Cordic units and two sets of registers, as depicted in Figure 16. The left-side arithmetic units of the registers in both sub-modules rotate the real and imaginary parts of the same element, while the right-side units rotate the real and imaginary parts of elements in different columns.

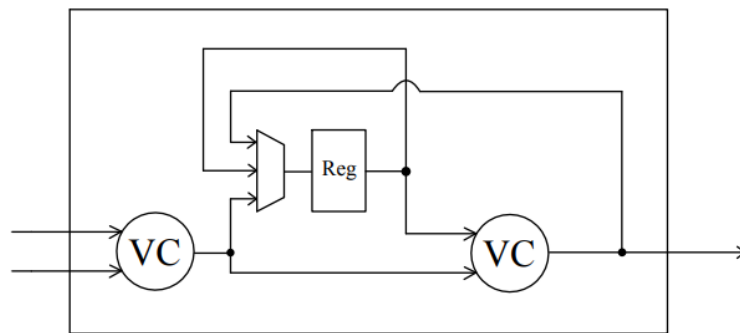


Figure 15 Givens Generator Arithmetic

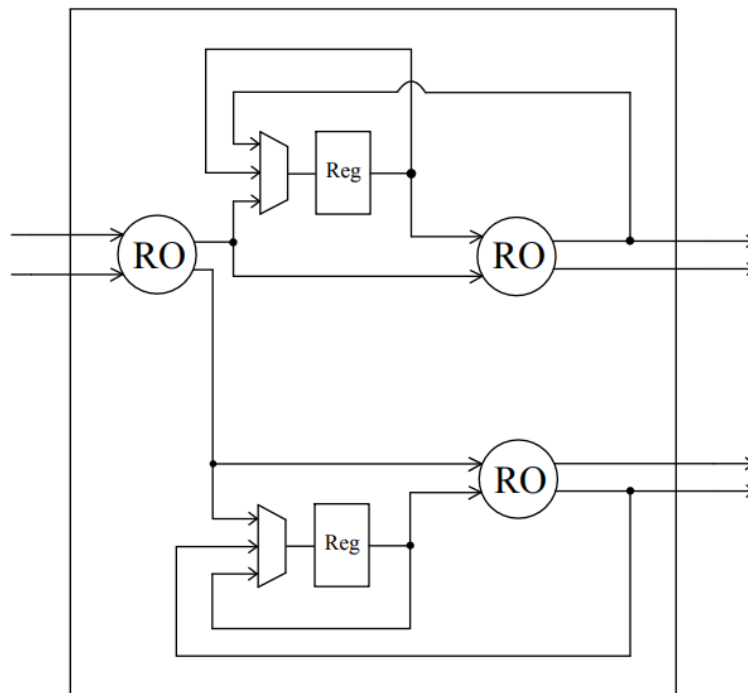


Figure 16 Givens Rotation Arithmetic

### 3. System Architecture

In our design, there are a total of 4 sets of 256x8 SRAM for storing input data, one GG (Givens Generator), and four GRs (Givens Rotation) for computation, along with three sets of main diagonal registers for temporary storage.

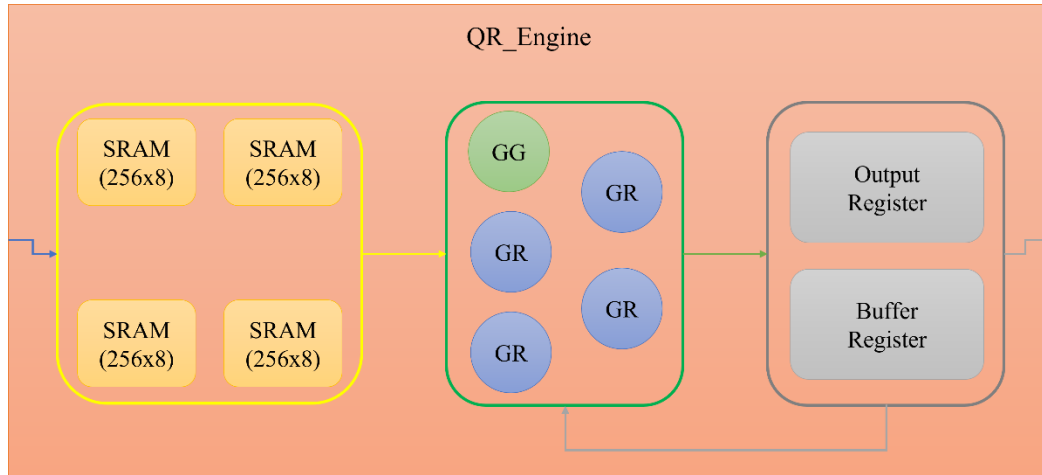


Figure 17 System Architecture

### 4. Data Flow

When data needs to flow from a dimension with a smaller  $k$  to a dimension with a larger  $k$ , it is necessary to temporarily store the data to avoid data conflicts. Therefore, we utilize the output registers along the main diagonal and additionally construct temporary storage units for the imaginary parts of columns 2 to 4.

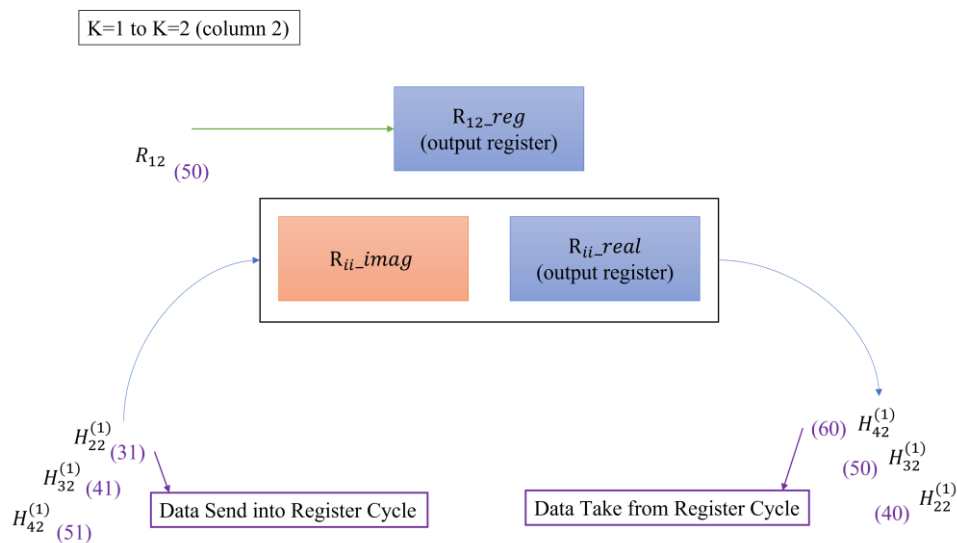


Figure 18 Construct Main Diagonal's Imaginary Part



## 5. Hardware Resource

Table 1 Hardware Resource

Signal	Signed	Bit	Fraction Bits
Cordic Opetator	True	17	12
Cordic Output	True	16	12
K (Cordic)	True	7	6
Iteration Number (Cordic Rotation Times)		9	
Rounding Method		Floor	
Hardware Resource		Number	
SRAM(256×8)		4	
GG (VC mode Cordic * 2)		1	
GR (RO mode Cordic * 3)		3	

## 6. Error Rate

## a. SNR = 15dB

Table 2 Error Rate (SNR = 15dB)

P1	P2	P3	E1	E2	E3
0.71401%	0.98774%	0.80884%	0.82933%	0.67621%	1.02240%
E4	E5	E6	E7	E8	E9
0.66256%	0.85748%	0.97011%	0.60913%	0.92497%	0.91258%
E10	-	-	-	-	-
0.93483%	-	-	-	-	-

## b. SNR = 10dB

Table 3 Error Rate (SNR = 15dB)

P6	P7	P8	E1	E2	E3
0.43330%	0.51316%	0.65443%	0.40262%	0.64865%	0.39244%
E4	E5	E6	E7	E8	E9
0.44056%	0.54164%	0.32117%	0.54500%	0.45286%	0.53670%
E10	-	-	-	-	-
0.60552%	-	-	-	-	-

## Result

### 1. PPA(Power 、Performance 、Area)

Table 4 PPA Table

	Power(W)	Latency(ps)	Area ( $\mu m^2$ )	Performance Gain	Performance
Gate Level	0.0135	694981501	378290.17	1.76	2.0166E+12
Transistor Level	0.0192	694981501	389269.83	1.76	2.95129E+12

### 2. Improve Performance

- Efficient algorithm
- Well-designed architecture and data flow
- Use clock gating to reduce power consumption
- Increase Core Utilization during APR to reduce area
- Add more power stripes to mitigate power impact on circuit operation

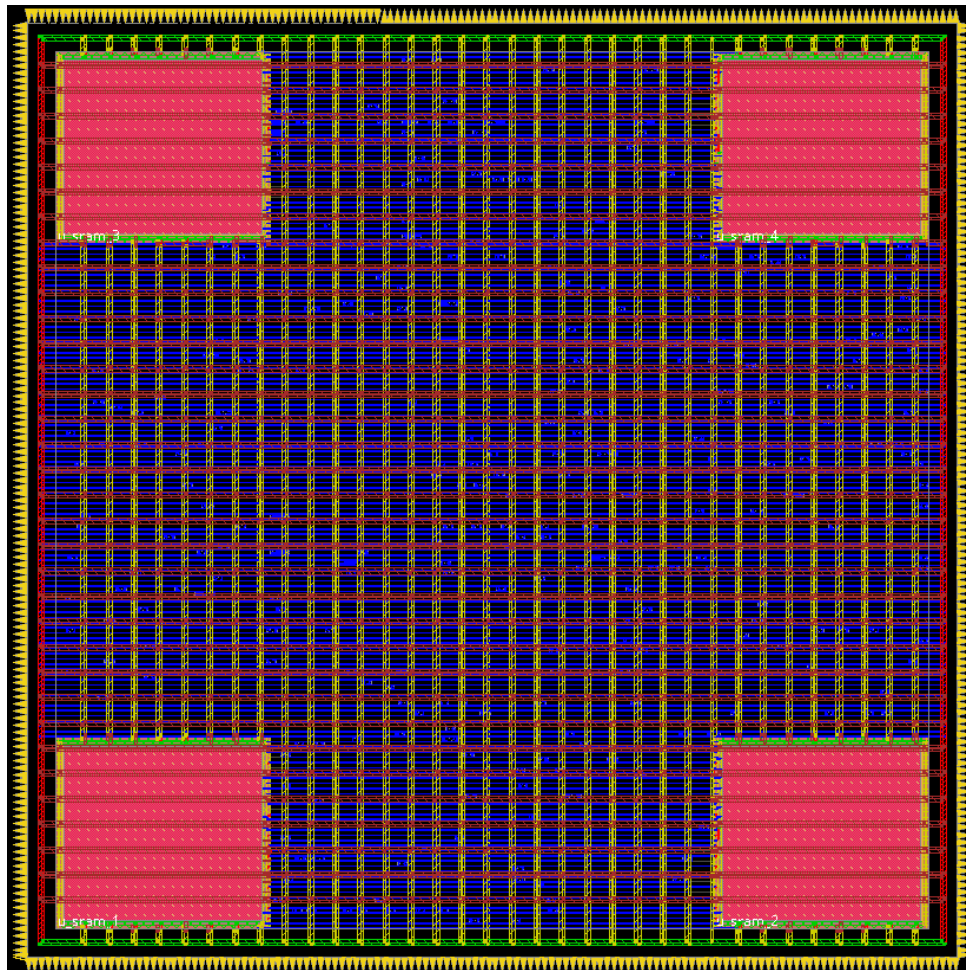


Figure 19 Add More Power Stripes

## 3. Result Table

Table 5 Result Table

Design Stage	Description	Value
P&R	Die Area (um <sup>2</sup> )	441221.21
	Core Area (um <sup>2</sup> )	389269.83
	Core Density (Counting Std Cells and MACROs)	96.851%
Post-layout Simulation	Clock Period for Post-layout Simulation (ex. 10ns)	5.0

## 4. Result Figure

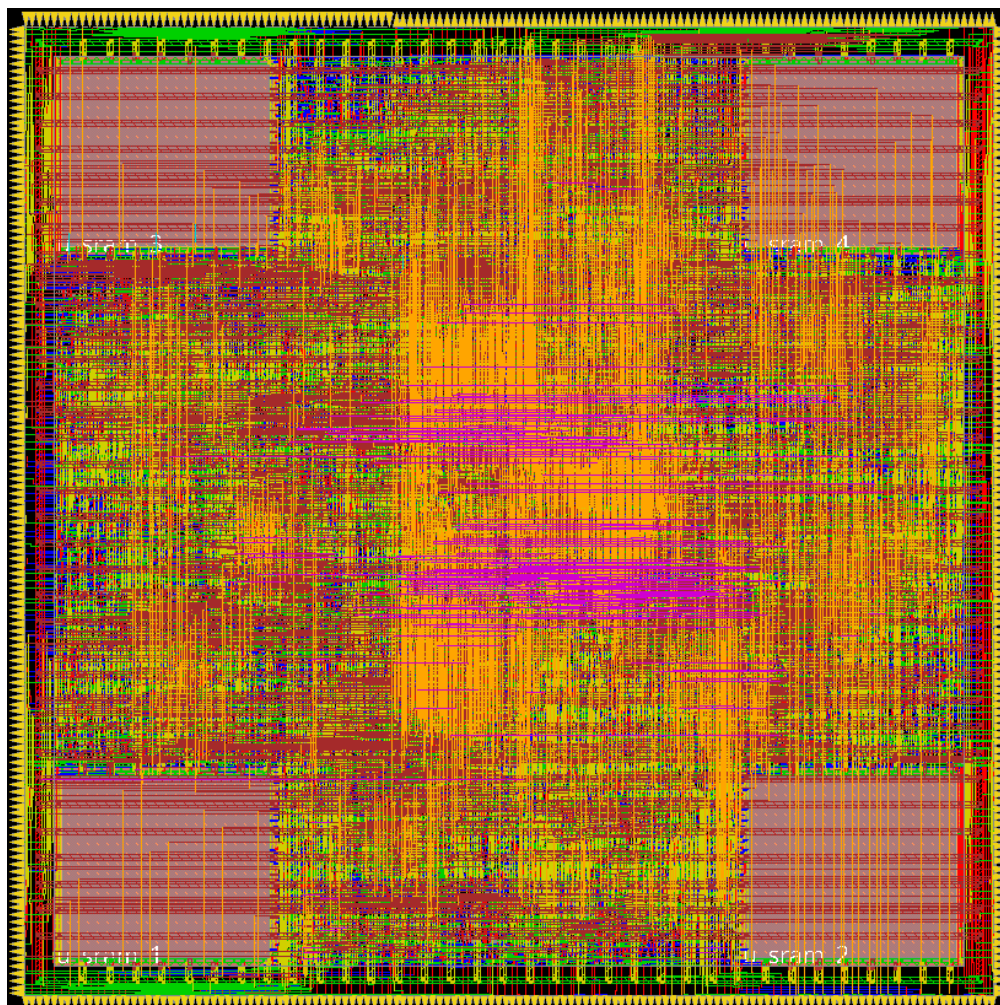


Figure 20 Layout

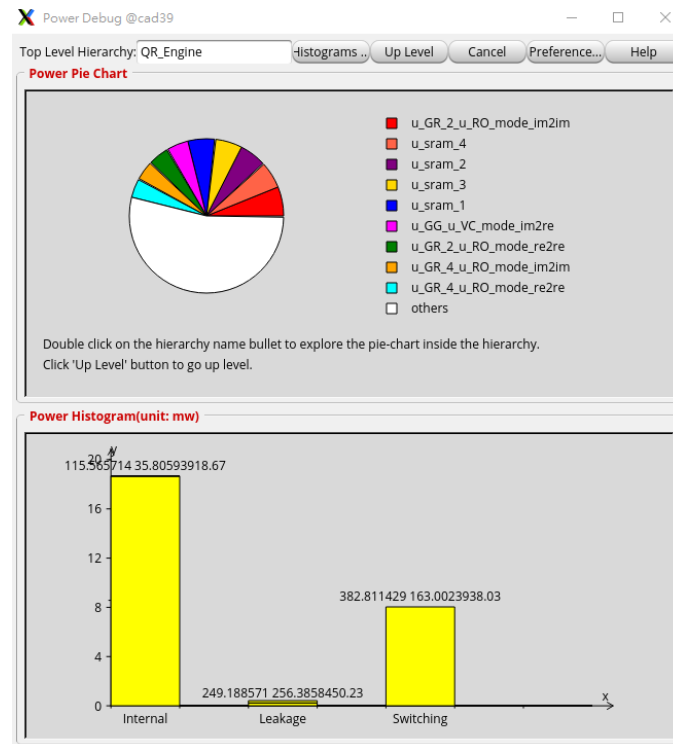


Figure 21 Power Pie Chart

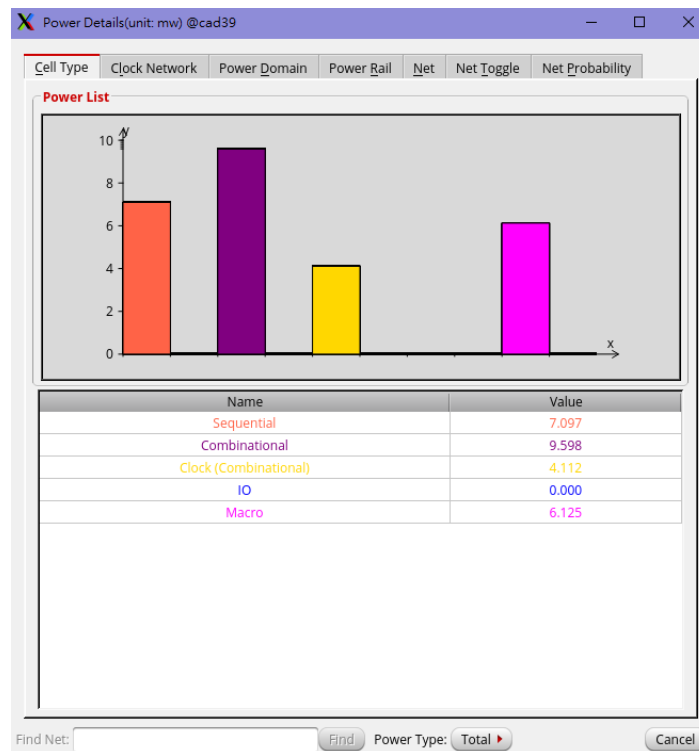


Figure 22 Power Cell Type

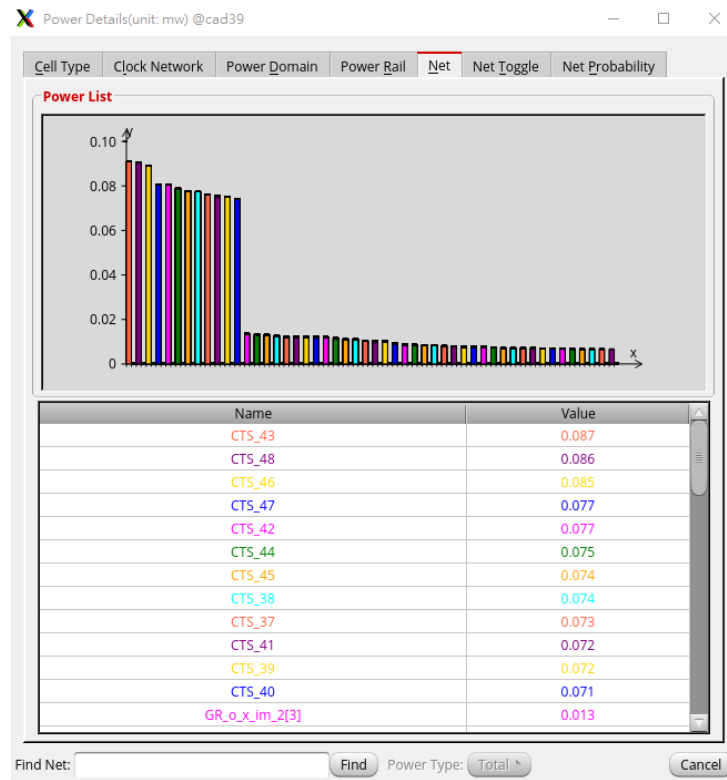


Figure 23 Power Net

## Reference

- 
- [1] Z. -Y. Huang and P. -Y. Tsai, "Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 10, pp. 2531-2542, Oct. 2011, doi: 10.1109/TCSI.2011.2123770.
  - [2] S. S. Omran and A. K. Abdul-abbas, "Fast QR Decomposition Based on FPGA," 2018 International Conference on Advanced Science and Engineering (ICOASE), Duhok, Iraq, 2018, pp. 189-193, doi: 10.1109/ICOASE.2018.8548895.
  - [3] D. Chen and M. Sima, "Fixed-Point CORDIC-Based QR Decomposition by Givens Rotations on FPGA," 2011 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 2011, pp. 327-332, doi: 10.1109/ReConFig.2011.38.
  - [4] A. Maltsev, V. Pestretsov, R. Maslennikov and A. Khoryaev, "Triangular systolic array with reduced latency for QR-decomposition of complex matrices," 2006 IEEE International Symposium on Circuits and Systems (ISCAS), Kos, Greece, 2006, pp. 4 pp.-, doi: 10.1109/ISCAS.2006.1692603.
  - [5] D. Patel, M. Shabany and P. G. Gulak, "A low-complexity high-speed QR decomposition implementation for MIMO receivers," 2009 IEEE International Symposium on Circuits and Systems, Taipei, Taiwan, 2009, pp. 33-36, doi: 10.1109/ISCAS.2009.5117678.
  - [6] Yin-Tsung Hwang and Wei-Da Chen, "A low complexity complex QR factorization design for signal detection in MIMO OFDM systems," 2008 IEEE International Symposium on Circuits and Systems (ISCAS), Seattle, WA, USA, 2008, pp. 932-935, doi: 10.1109/ISCAS.2008.4541572.
  - [7] R. C. -H. Chang, C. -H. Lin, K. -H. Lin, C. -L. Huang and F. -C. Chen, "Iterative \$QR\$ Decomposition Architecture Using the Modified Gram-Schmidt Algorithm for MIMO Systems," in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 5, pp. 1095-1102, May 2010, doi: 10.1109/TCSI.2010.2047744.
  - [8] Simple, Fast and Practicable Algorithms for Cholesky, LU and QR Decomposition Using Fast Rectangular Matrix Multiplication