Computer-Aided VLSI System Design

Final Project Report

Team Number: 21

Student Name: R12943013 聶家任、R12943126 鄭亦洹

Outline

1.Algorithm

Generally, there are three algorithms can solve QR decomposition. [1] The first one is Householder Reflection Algorithm; the second one is Givens Rotation Algorithm, and the last one is Modified Gram-Schmidt Process Algorithm. After we realize all these three algorithms, we choose two among them to implement. The Householder Reflection Algorithm looks pretty umbalenced in hardware and cannot take the benefit of hardware sharing to optimize area, so we choose the last two algorithms to implement and hopefully we can compare these two methods in performance at last. In the Modified Gram-Schmidt Process Algorithm, it is necessary to perform square root and division calculations. To simplify hardware complexity, we employ the Newton interpolation method to approximate the result of the reciprocal square root.

Even with this approach, a significant number of multipliers are still required in the computation process of Modified Gram-Schmidt Process Algorithm. In comparison to the Givens Rotation Algorithm, which relies primarily on adders and is more advantageous in terms of hardware complexity, Modified Gram-Schmidt Process Algorithm may not exhibit a clear advantage.

After conducting experiments, it was observed that the hardware area required for the Modified Gram-Schmidt Process Algorithm is approximately three to four times that of the Givens Rotation Algorithm. However, the latency is roughly the same. Consequently, **we have chosen Givens Rotation Algorithm as the algorithm for our final project.**

Since Modified Gram-Schmidt Process Algorithm is introduced detailedly in class, we introduce only Givens Rotation Algorithm below.

The operation of Givens rotation is to find $\cos(\theta)$ and $sin(\theta)$ such that [2]

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{1}$$

Where $r = \sqrt{a^2 + b^2}$. So, the basic idea of Givens rotation is to find the θ so that the vector $[a\ b]^T$ is rotated to the real axis $[r\ 0]^T$. The solution of this equation (1) can be easily derived by

$$\cos(\theta) = \frac{a}{\sqrt{a^2 + b^2}} \tag{2}$$

$$\sin(\theta) = \frac{-b}{\sqrt{a^2 + b^2}} \tag{3}$$

However, since we are dealing with complex numbers in this project. The equation (1) should be re-written by [3]

$$\begin{bmatrix} \cos(\theta)\, e^{-j\theta_a} & -\sin(\theta)e^{-j\theta_b} \\ \sin(\theta)e^{-j\theta_a} & \cos(\theta)e^{-j\theta_b} \end{bmatrix} \begin{bmatrix} |a|e^{j\theta_a} \\ |b|e^{j\theta_b} \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix} \tag{4}$$

Where $r = \sqrt{|a|^2 + |b|^2}$. And

$$\cos(\theta) = \frac{|a|}{\sqrt{|a|^2 + |b|^2}} \tag{5}$$

$$\sin(\theta) = \frac{-|b|}{\sqrt{|a|^2 + |b|^2}} \tag{6}$$

By applying this rotation matrix on a complex vector, the element in the lower row can be zeroed, with the element in the upper row real.

Denote h_{ij} as the entry of a 4x4 channel coefficient matrix $\boldsymbol{H}$ in the *i*-th row and *j*-th column. We can apply Givens rotation on $(h_{11}, h_{21}), (h_{11}, h_{31}), (h_{11}, h_{41}), (h_{22}, h_{32}), (h_{22}, h_{42}), (h_{33}, h_{43})$ and a upper-triangular matrix $\boldsymbol{R}$ are made. However, since the diagonal entries of the output $\boldsymbol{R}$ matrix should be real. One ordinary rotation is used to rotate the complex r_{44} to real. [3]

In equation (4), the rotation operation can be done using two sequential Cordic operation. [4] The first Cordic is used to rotate complex *a* and *b* to real, and the second Cordic is to rotate $(|a|, |b|)$ to $(r, 0)$. That is, the first Cordic is operated in vectoring mode to rotate *a* by $-\theta_a$, *b* by $-\theta_b$ and the second Cordic is also operated in vectoring mode to rotate $(|a|, |b|)$ by $-\theta$. By memorizing the rotating angles in vectoring mode, we could use them to rotate the entries in the same rows but on the right hand side of the diagonal and in this time the Cordics are opetating in rotation mode.

2.Architecture

2.1 FXP Setting

There are two main fixed point bit length parameters we should determine. The first one is the bit length of the adders in Cordics, which is mainly related to the computation resource we should have and is a little related to the flipflops we should create in Cordics and QR Engine output buffers. The second one is the bit length of input data, which is mainly related to the number of srams we need to use. The thorough fixed point bit length decisions are discussed in Section 2.4.

2.2 Scheduling

As data being fed into QR Engine, say, i_trig signal is high, we store the four rows of $\boldsymbol{H}$ and y of the first RE into four FIFOs and save the data of last nine REs into sram macros. If we use full length of S1.22 from input data, in every clock cycle, we would need to store 24 bits in the real data and 24 bits in the imaginary data, so $(24+24)/8 = 6$ 8-bit sram macros are needed. Because of the data count of the last nine REs are $20*9 = 180 < 256$, the depth of our sram macros are 256. Therefore, six sram_256x8s should be used in our design.

QR Decomposition could be implemented by a systolic array. [3] As shown in Fig. 1, where PE, DU and RU are discussed in Section 2.3, QR Decomposition using Givens Rotation Algorithm can be benefited from this systolic architecture. And when RE data in a FIFO is fully consumed, we could load new data of the next RE from sram macros into this FIFO. Therefore, a high throughput can be achieved and no time overhead when switching REs.

When storing last 9 REs into sram macros, we can’t take RE1 from these sram macros at the same time even we finish the computation of RE0, so we need not to calculate RE0 too early. Instead, we make the first trigger of this systolic array at the time the first row of RE1 could be taken out right after the i_trig signal. So the time overhead of storing last nine REs is reduced.

Besides, when informed from TAs that we can raise o_last_data earlier than we finish the computation of the last RE, we start to load next 10RE earlier.

Consequently, the systolic array is working during the beginning and during the end of the data loading of 10RE. Hence the time overhead of switching to next 10RE is minimized.

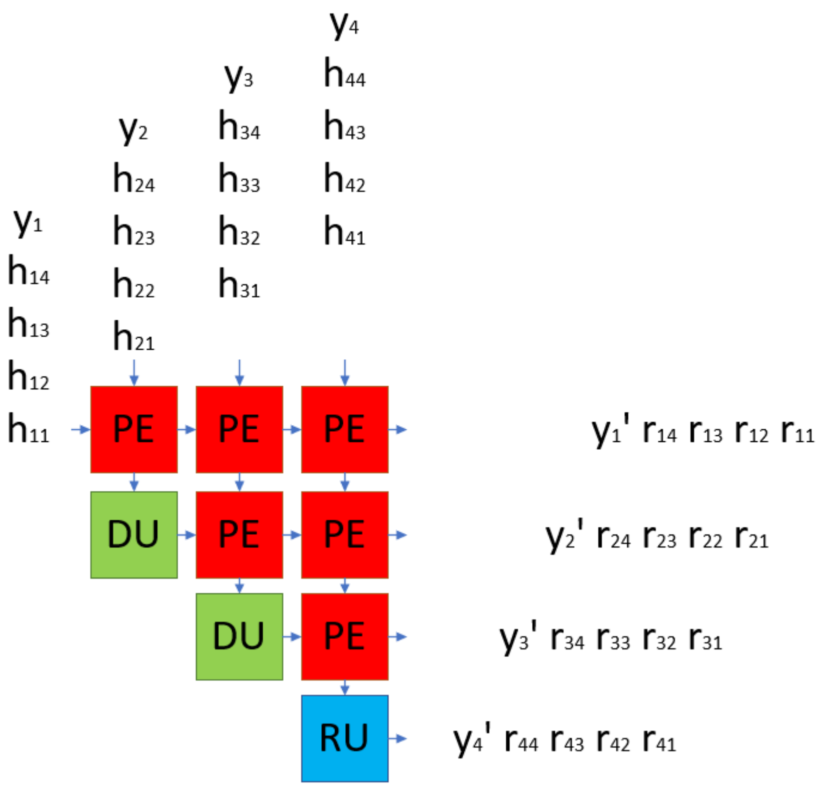


FIGURE 1 **Systolic Array (Note that r_{21}, r_{31}, r_{32}, r_{41}, r_{42} and r_{43} are zero)**

2.3 Block Diagram

In PE block, as shown in Fig. 2, when h_{ii}, $i \in \{1,2,3\}$, gets into PE from the left, the Cordics are operating in vectoring mode, and the θ_a, θ_b and θ are memorized in this PE such that when h_{ij}, $i \neq j$ entering this PE from the left, the same rotation can be reproduced.

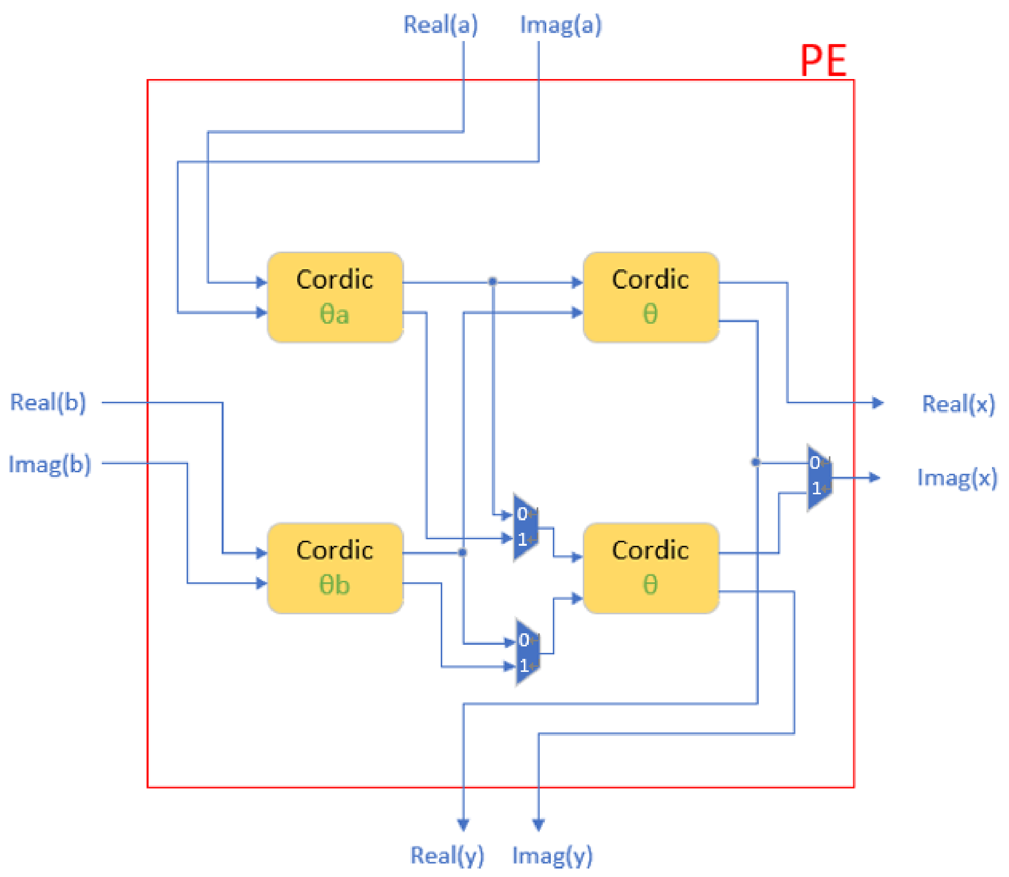


FIGURE 2 PE Block

To describe this PE clearly, when Cordics in this PE operate in vectoring mode, as shown in Fig. 3, the angles are needed to be recorded. After a passes the Cordic in the up left, $(Real(a), Imag(a))$ is rotated to $(|a|, 0)$, and so is b. Afterwards, $|a|$ and $|b|$ are fed into the the up right and down right Cordics to rotate $(|a|, |b|)$ to $(Real(x), Imag(x)) = (\sqrt{|a|^2 + |b|^2}, 0)$ and record the angle. And the datapath in the multiplexers is from 0.

Note that what go to $Real(y)$ and $Imag(y)$ do not matter because every Cordic is needed to first operate in vectoring mode, say, h_{ii}, $i \in \{1,2,3\}$, enters from the left, to record the angle, and if not, the output in this systolic array would not be recorded and these values are corresponded to those zeros in $\boldsymbol{R}$.

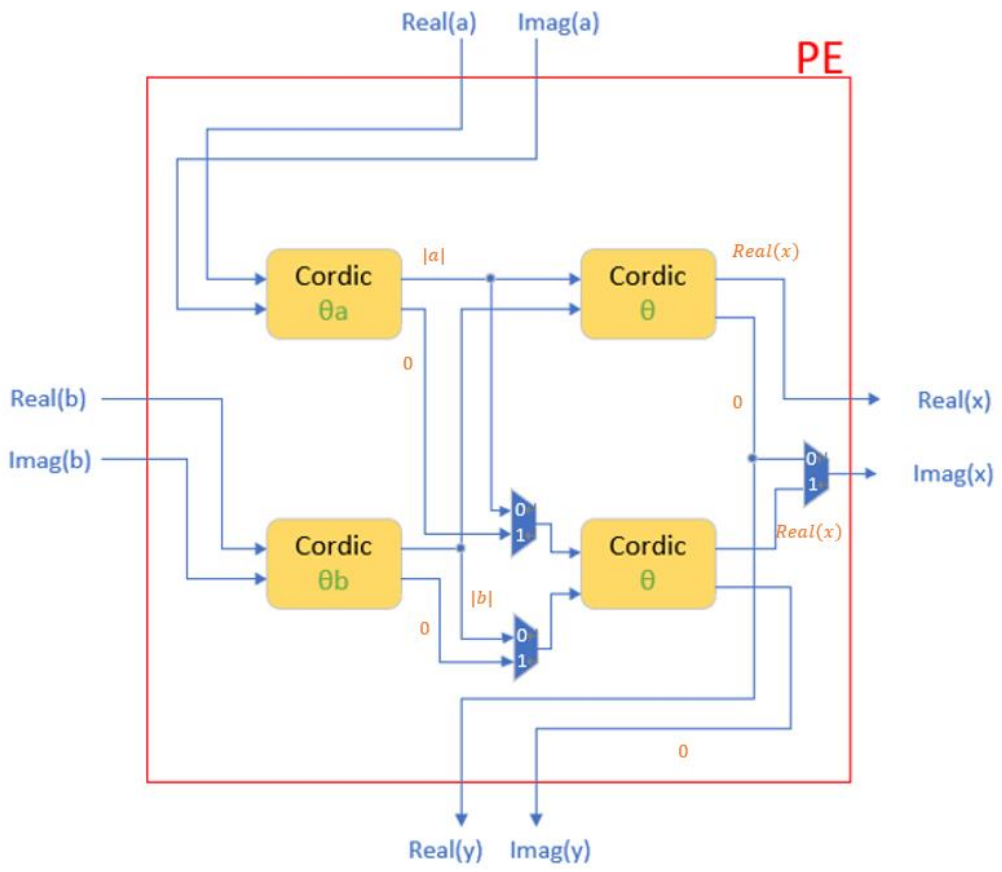


FIGURE 3 **PE Block Operates in Vectoring Mode (Multiplexer Selects 0)**

While Cordics in this PE operate in rotation mode, as shown in Fig. 4, we may denote the variables entering this PE by a' and b' to replace a and b in Fig. 2. a' and b' first undergo rotations by $-\theta_a$ and $-\theta_b$ to $a'e^{-j\theta_a}$ and $b'e^{-j\theta_b}$, and the real parts of $a'e^{-j\theta_a}$ and $b'e^{-j\theta_b}$ experience another rotation by $-\theta$ in the up right Cordic from $\left(Real(a'e^{-j\theta_a}),\ Real(b'e^{-j\theta_b})\right)$ to $\left(Real(x), Real(y)\right) = \left(Real((Real(a'e^{-j\theta_a}) + jReal(b'e^{-j\theta_b}))e^{-j\theta}), Imag((Real(a'e^{-j\theta_a}) + jReal(b'e^{-j\theta_b}))e^{-j\theta})\right)$. And so do the imaginary parts in the down right Cordic. And the datapath in the multiplexers is from 1.

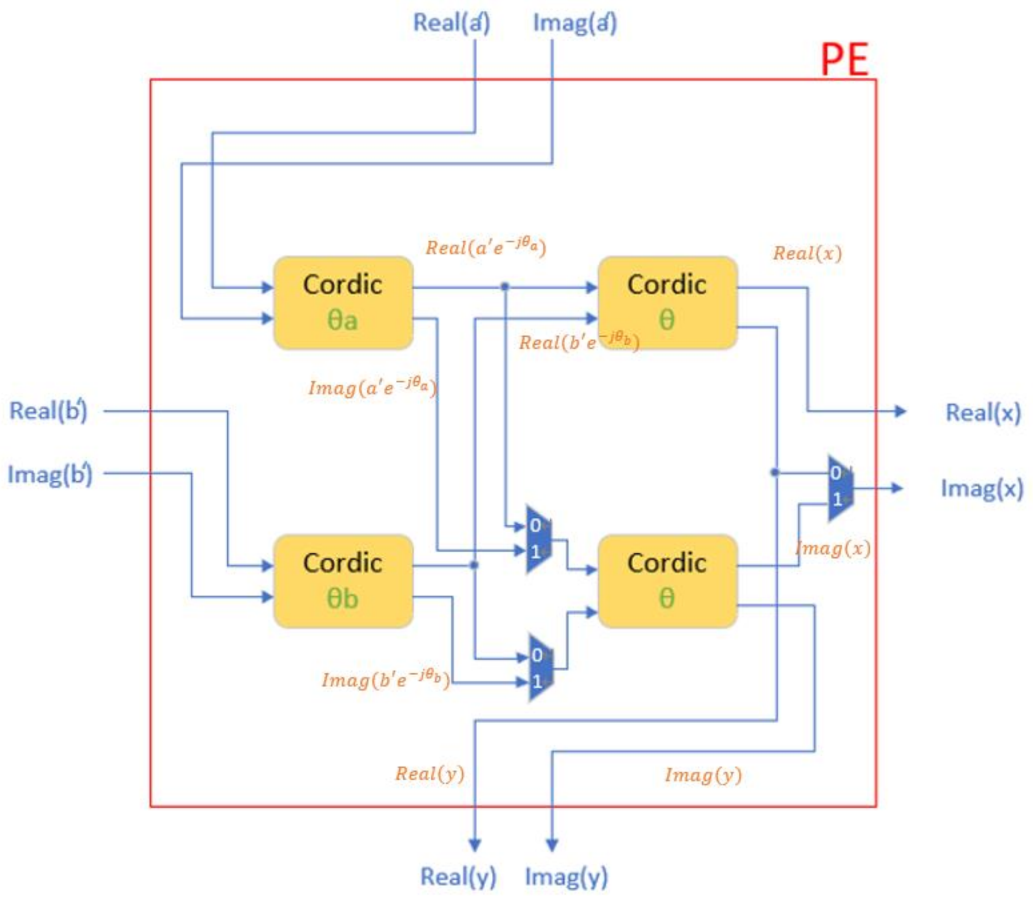


FIGURE 4 PE Block Operates in Rotation Mode (Multiplexer Selects 1)

In DU block, a simple delay is achieved to have correct entries enter its right PE block. In RU block, a unitary rotation is made to make the complex r_{44} real. In this case, one Cordic operates in vectoring mode can be used when h_{44} enters RU from upside, and the rotation angle is memorized in this Cordic to rotate the following entry.

2.4 Hardware Improvement Process

Since we implement this design with systolic array, REs in every 10RE could be sent into this systolic array without any time overhead, and a high throughput can be achieved. Hence, our hardware improvement process focuses mainly on the area. In our experiences, power is roughly proportional to area, so we list only the area changes and the techniques we use in Table 1.

TABLE 1 Hardware Improvement Process

Version	Area(μm^2)	Technique
0	1184193.18	Initial function correct design with 6 sram_256x8s, use full width S3.22 in adders of Cordics and full width of S1.22 from input
1	798973.35	Left and right Cordics sharing in Fig. 2
2	568646.35	Cordic coefficient use 0.101 to approximate 0.10011011011111011011, that is, use (x >>> 1) + (x >>> 3) to approximate (x >>> 1) + (x >>> 4) + (x >>> 5) + (x >>> 7) + (x >>> 8) + (x >>> 10) + (x >>> 11) + (x >>> 12) + (x >>> 13) + (x >>> 15) + (x >>> 16) + (x >>> 18) + (x >>> 19)
3	402923.95	Use S3.16 in adders of Cordics rather than the full width S3.22, and use S1.16 from input rather than the full width S1.22
4	335233.92	Use S3.16 in adders of Cordics and S1.14 from input such that 2 sram_256x8 can be reduced (Now 4 sram_256x8s)
5	281017.71	Use S3.16 in adders of Cordics and S1.6 from input such that 2 sram_256x8 can be reduced (Now 2 sram_256x8s)
6	275536.19	Force positive and negative rotation in Cordics use the same adders
7	188266.68	Use S3.8 in adders of Cordics

After our hardware improvements, the hardware area can reduce 6x from our initial design and the performance evaluation before APR can have 10^{11} in magnitude.

3.APR Results

3.1 Floorplan

TABLE 2 **Floorplan Parameter Setting**

Aspect Ratio	Core Utilization	Core to Left(㎛)	Core to Right(㎛)	Core to Top(㎛)	Core to Bottom(㎛)
1	0.97	6	6	6	6

TABLE 3 **Halo Parameter Setting**

Top(㎛)	Bottom(㎛)	Left(㎛)	Right(㎛)
6	6	6	6

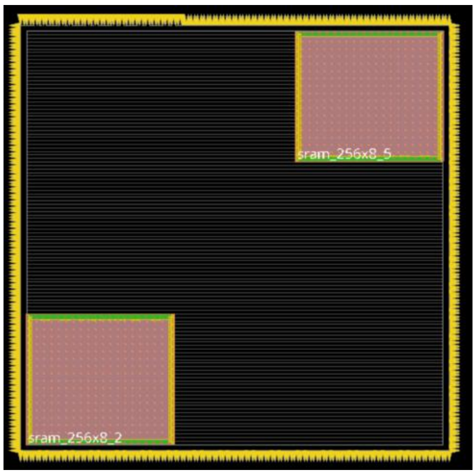


FIGURE 5 **Floorplan Result**

3.2 Powerplan

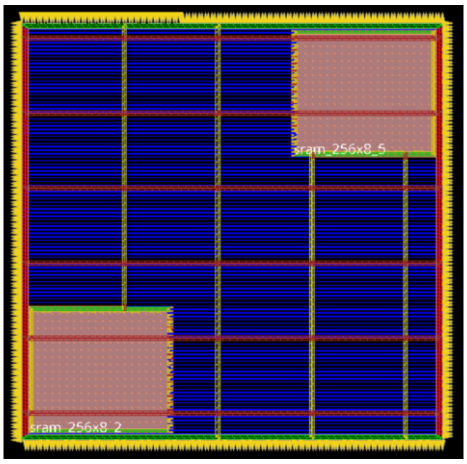


FIGURE 6 **Powerplan Result**

3.3 Final Result

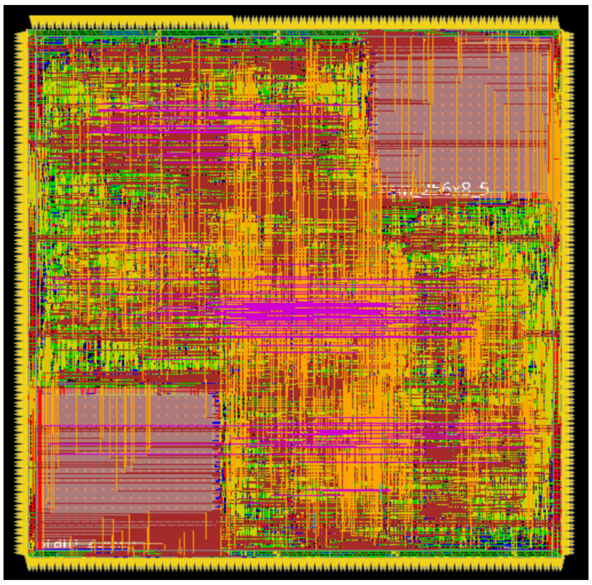
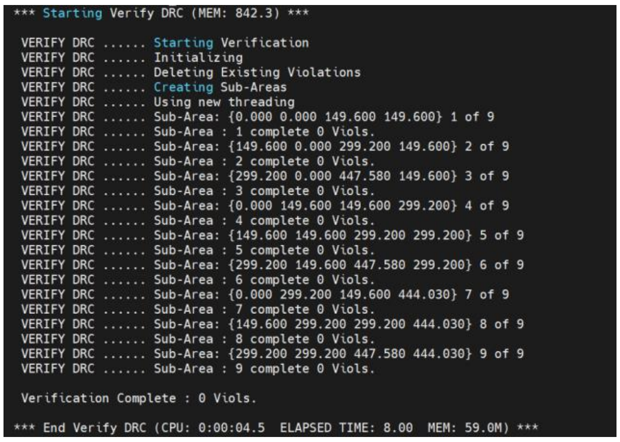
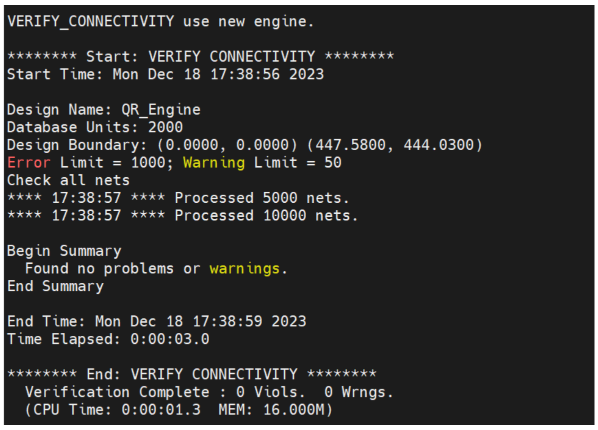


FIGURE 7 APR Final Result



```
*** Starting Verify DRC (MEM: 842.3) ***

 VERIFY DRC ...... Starting Verification
 VERIFY DRC ...... Initializing
 VERIFY DRC ...... Deleting Existing Violations
 VERIFY DRC ...... Creating Sub-Areas
 VERIFY DRC ...... Using new threading
 VERIFY DRC ...... Sub-Area: {0.000 0.000 149.600 149.600} 1 of 9
 VERIFY DRC ...... Sub-Area : 1 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {149.600 0.000 299.200 149.600} 2 of 9
 VERIFY DRC ...... Sub-Area : 2 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {299.200 0.000 447.580 149.600} 3 of 9
 VERIFY DRC ...... Sub-Area : 3 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {0.000 149.600 149.600 299.200} 4 of 9
 VERIFY DRC ...... Sub-Area : 4 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {149.600 149.600 299.200 299.200} 5 of 9
 VERIFY DRC ...... Sub-Area : 5 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {299.200 149.600 447.580 299.200} 6 of 9
 VERIFY DRC ...... Sub-Area : 6 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {0.000 299.200 149.600 444.030} 7 of 9
 VERIFY DRC ...... Sub-Area : 7 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {149.600 299.200 299.200 444.030} 8 of 9
 VERIFY DRC ...... Sub-Area : 8 complete 0 Viols.
 VERIFY DRC ...... Sub-Area: {299.200 299.200 447.580 444.030} 9 of 9
 VERIFY DRC ...... Sub-Area : 9 complete 0 Viols.

 Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:04.5  ELAPSED TIME: 8.00  MEM: 59.0M) ***
```

FIGURE 8 DRC Result

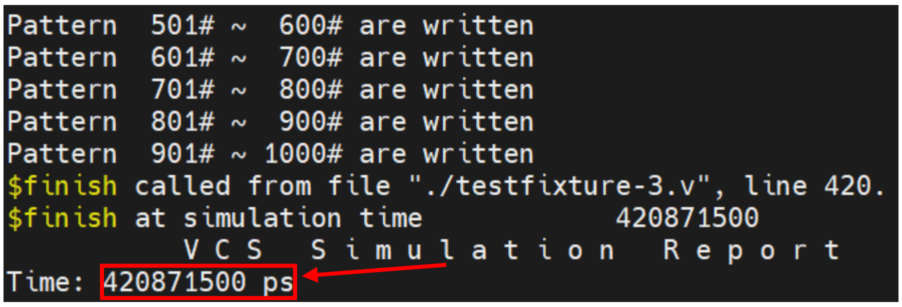


```
VERIFY_CONNECTIVITY use new engine.

******** Start: VERIFY CONNECTIVITY ********
Start Time: Mon Dec 18 17:38:56 2023

Design Name: QR_Engine
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (447.5800, 444.0300)
Error Limit = 1000; Warning Limit = 50
Check all nets
**** 17:38:57 **** Processed 5000 nets.
**** 17:38:57 **** Processed 10000 nets.

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Mon Dec 18 17:38:59 2023
Time Elapsed: 0:00:03.0

******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
  (CPU Time: 0:00:01.3  MEM: 16.000M)
```

FIGURE 9 LVS Result

4.Performance

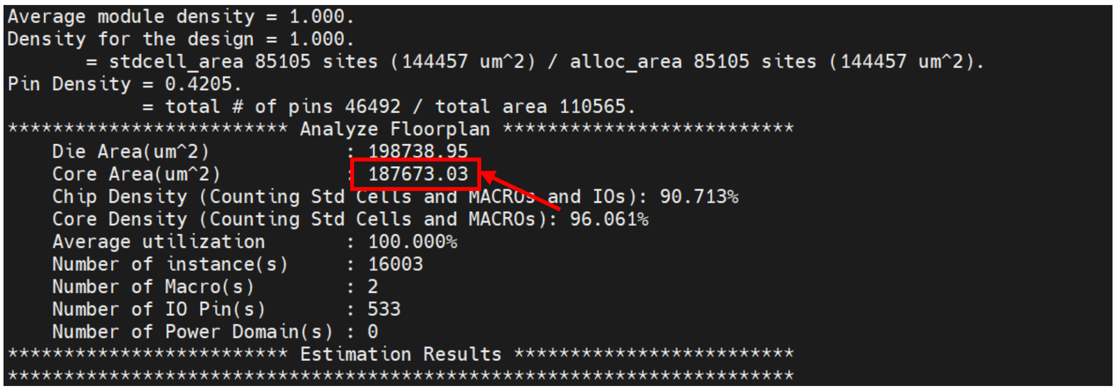
4.1 Timing Result



```
Pattern  501# ~  600# are written
Pattern  601# ~  700# are written
Pattern  701# ~  800# are written
Pattern  801# ~  900# are written
Pattern  901# ~ 1000# are written
$finish called from file "./testfixture-3.v", line 420.
$finish at simulation time           420871500
         V C S   S i m u l a t i o n   R e p o r t
Time: 420871500 ps
```

FIGURE 10 Timing Report

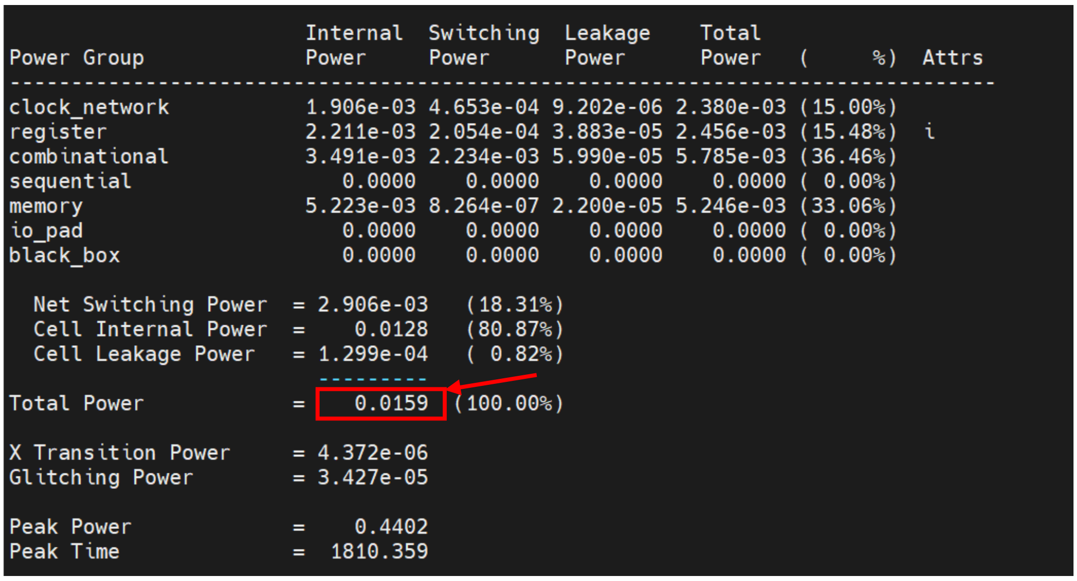
4.2 Area Result



```
Average module density = 1.000.
Density for the design = 1.000.
       = stdcell_area 85105 sites (144457 um^2) / alloc_area 85105 sites (144457 um^2).
Pin Density = 0.4205.
            = total # of pins 46492 / total area 110565.
************************* Analyze Floorplan *************************
    Die Area(um^2)              : 198738.95
    Core Area(um^2)             : 187673.03
    Chip Density (Counting Std Cells and MACROs and IOs): 90.713%
    Core Density (Counting Std Cells and MACROs): 96.061%
    Average utilization         : 100.000%
    Number of instance(s)       : 16003
    Number of Macro(s)          : 2
    Number of IO Pin(s)         : 533
    Number of Power Domain(s) : 0
************************* Estimation Results *************************
**********************************************************************
```

FIGURE 11 Area Report

4.3 Primetime Power Report



```
                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network           1.906e-03 4.653e-04 9.202e-06 2.380e-03 (15.00%)
register                2.211e-03 2.054e-04 3.883e-05 2.456e-03 (15.48%)  i
combinational           3.491e-03 2.234e-03 5.990e-05 5.785e-03 (36.46%)
sequential                 0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                  5.223e-03 8.264e-07 2.200e-05 5.246e-03 (33.06%)
io_pad                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                  0.0000    0.0000    0.0000    0.0000 ( 0.00%)

  Net Switching Power  = 2.906e-03   (18.31%)
  Cell Internal Power  =    0.0128   (80.87%)
  Cell Leakage Power   = 1.299e-04   ( 0.82%)
                         ---------
Total Power            =    0.0159   (100.00%)

X Transition Power     = 4.372e-06
Glitching Power        = 3.427e-05

Peak Power             =    0.4402
Peak Time              =  1810.359
```

FIGURE 12 Power Result

4.4 Error Rate Result

TABLE 4 **Error Rate Result**

Pattern	Soft Error Rate
P1	1.3069%
P2	1.8538%
P3	1.0958%
P4	0.2389%
P5	0.2362%
P6	0.9915%

4.5 Area x Time x Power / (Performance Gain) Performance

$$420872(\mathrm{ns}) \times 187673.03(\mu\mathrm{m}^2) \times 15.9(\mathrm{mW})/1.76 = \mathbf{7.136 \times 10^{11}}$$

5.Reference

1. https://en.wikipedia.org/wiki/QR_decomposition
2. https://en.wikipedia.org/wiki/Givens_rotation
3. Maltsev, Alexander, et al. "Triangular systolic array with reduced latency for QR-decomposition of complex matrices." 2006 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2006.
4. https://en.wikipedia.org/wiki/CORDIC