

VLSI DPS HW#4

電機四 4109064119 陳柏淳

1. Calculate the Frobenius Distance

```
Matrix A :
-95 -125 -53 -108
113 -117 62 109
116 -85 -80 70
19 38 47 -4
-113 59 -82 -17
-68 37 -34 -14
-38 -13 32 -50
82 12 71 2
```

● Generate a random matrix A

```
Matrix Q_float :
-0.3841 0.4569 0.4690 0.0768 -0.4569 -0.2749 -0.1536 0.3315
-0.7889 -0.4293 -0.2600 0.2247 0.1308 0.0864 -0.1251 0.1873
0.0765 0.3000 -0.7361 0.2164 -0.3511 -0.1076 0.3423 0.2520
-0.3412 0.6795 -0.2002 -0.1280 0.3468 0.2077 -0.1314 -0.4297
0.3284 0.1393 -0.1124 0.4462 0.2219 0.1268 -0.6965 0.3325
0 0.1891 0.2463 0.1124 0.5477 0.1913 0.5293 0.5236
0 0 0.2393 0.7462 -0.2118 0.3381 0.2361 -0.4135
0 0 0 0.3344 0.3695 -0.8338 0.0720 -0.2264
```

```
Matrix Q_fix :
-0.3828 0.4561 0.4697 0.0752 -0.4609 -0.2783 -0.1582 0.3291
-0.7939 -0.4307 -0.2637 0.2217 0.1270 0.0830 -0.1270 0.1777
0.0703 0.3018 -0.7363 0.2188 -0.3545 -0.1074 0.3516 0.2422
-0.3506 -0.0820 0.0967 -0.1367 0.3350 0.1953 -0.0889 -0.4463
0.3164 -0.6748 0.1904 0.4541 0.2139 0.1182 -0.6777 0.3076
0 -0.1846 -0.2178 0.0420 0.5234 0.1914 0.4834 0.5400
0 0 0.2695 0.7607 -0.1572 0.3066 0.2988 -0.3828
0 0 0 0.2969 0.3965 -0.8467 0.0889 -0.1816
```

```
Matrix R_float :
247.3297 -73.5415 80.2087 143.7676
-0.0000 194.2618 42.1888 22.5000
0.0000 0.0000 144.8919 -37.0943
0.0000 -0.0000 -0.0000 94.3326
0.0000 0.0000 -0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 -0.0000
```

```
Matrix R_fix :
248.0000 -74.7500 79.5000 143.0000
0 193.7500 41.0000 21.5000
0 0 146.0000 -37.2500
0 0 0 93.7500
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

- Q_float and R_float are generated from $Q^* [A | I] = [R | Q]$
- Q_fix and R_fix are generated from Cordic given's rotation
- The Frobenius Distance (Euclidean Norm) is $\text{sqrt}(\text{trace}((\text{float-fix})^*(\text{float-fix}')))$

```
----- Hardware implementation loss -----
Q Fix Point Loss :
1.3433

R Fix Point Loss :
2.6959
```

Q_loss : 1.3433
R_loss : 2.6959

- Fix-point precision

Data	Sign bit	Integer part	Fraction part	Length
A (input)	1	7	0	8
Q (output)	1	1	10	12
R (output)	1	9	2	12
K (parameter)	1	0	9	10

2. Timing diagram

- N : No Operation (Idle)
 - MK : Multiplied by K
 - | | | | | |
|----|----|----|----|---|
| 78 | 78 | 78 | 78 | N |
|----|----|----|----|---|

 : Perform rotation on 7th and 8th row
 - | | | | | | | | |
|----|----|----|----|----|----|----|----|
| 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
|----|----|----|----|----|----|----|----|

: The matrix index of data write back to tb
 - MK1_Q_former : process Q11~Q14
 - MK1_Q_later : process Q15~Q18

3. Clock cycles needed to complete one QR factorization

```

----- S U M M A R Y -----
*****
** Congratulations!! R matrix data have been generated successfully! **
** Congratulations!! Q matrix data have been generated successfully! **
*****
** The simulation results are all Pass!! **
** Get finish at cycle: 57 **
*****

```

57 cycles (As the timing diagram)

4. The initiation interval of two successive QR factorizations

1 CYCLE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	
2 RD ADDR	81	82	83	84	71	72	73	74	61	62	63	64	51	52	53	54	41	42	43	44	31	32	33	34	21	22	23	24	11	12	13	14					
3 RD DATA		81	82	83	84	71	72	73	74	61	62	63	64	51	52	53	54	41	42	43	44	31	32	33	34	21	22	23	24	11	12	13	14				
4 GG1		N	N	N	N	78	78	78	N	67	67	67	N	56	56	56	N	45	45	45	N	34	34	N	23	23	N	12	12	12	N						
5 GR11			N	N	N	N	78	78	78	N	67	67	67	N	56	56	56	N	45	45	N	34	34	N	23	23	N	12	12	12	N						
6 GR12				N	N	N	N	78	78	78	N	67	67	67	N	56	56	56	N	45	45	N	34	34	34	N	23	23	N	12	12	12	N				
7 GR13					N	N	N	N	78	78	78	N	67	67	67	N	56	56	56	N	45	45	45	N	34	34	34	N	23	23	N	12	12	12	N		
8 MK1									K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K			

When GG1 performs all vectoring over, the next input is valid.

The initiation interval is **33 cycles**.

● Matlab result

Matrix A :
-95 -125 -53 -108
113 -117 62 109
116 -85 -80 70
19 38 47 -4
-113 59 -82 -17
-68 37 -34 -14
-38 -13 32 -50
82 12 71 2

● Random matrix A

● Hardware loss

-----	Hardware implementation loss	-----
Q Fix Point Loss :	1.3477	
R Fix Point Loss :	2.3847	

Actual values

Original scale

Matrix Q_float :

-0.3841 0.4569 0.4690 0.0768 -0.4569 -0.2749 -0.1536 0.3315
-0.7889 -0.4293 -0.2600 0.2247 0.1308 0.0864 -0.1251 0.1873
0.0765 0.3000 -0.7361 0.2164 -0.3511 -0.1076 0.3423 0.2520
-0.3412 0.6795 -0.2002 -0.1280 0.3468 0.2077 -0.1314 -0.4297
0.3284 0.1393 -0.1124 0.4462 0.2219 0.1268 -0.6965 0.3325
0 0.1891 0.2463 0.1124 0.5477 0.1913 0.5293 0.5236
0 0 0.2393 0.7462 -0.2118 0.3381 0.2361 -0.4135
0 0 0 0.3344 0.3695 -0.8338 0.0720 -0.2264

Shift to integer to simulate hardware performance

INT scale

Matrix Q_float :

-393.3211 467.8451 480.2657 78.6642 -467.8451 -281.5351 -157.3284 339.4982
-807.8038 -439.6231 -266.2414 230.0868 133.8913 88.4553 -128.0858 191.7783
78.3759 307.1939 -753.7280 221.6229 -359.5194 -110.1942 350.5433 258.0019
-349.4265 695.8533 -204.9684 -131.0403 355.1718 212.6709 -134.5897 -439.9903
336.2825 142.6095 -115.1396 456.8971 227.2753 129.8618 -713.2458 340.4449
0 193.6781 252.2366 115.1114 560.8470 195.9207 542.0406 536.1423
0 0 245.0265 764.1410 -216.8794 346.2087 241.7700 -423.4107
0 0 0 342.4072 378.3236 -853.8199 73.7731 -231.8482

Matrix Q_fix :

-0.3828 0.4561 0.4678 0.0752 -0.4590 -0.2783 -0.1592 0.3320
-0.7881 -0.4268 -0.2578 0.2256 0.1328 0.0947 -0.1260 0.1914
0.0762 0.3018 -0.7344 0.2168 -0.3545 -0.1094 0.3457 0.2441
-0.3477 -0.0918 0.0977 -0.1357 0.3340 0.1914 -0.0967 -0.4404
0.3223 -0.6748 0.1914 0.4375 0.2178 0.1328 -0.6846 0.3213
0 -0.1846 -0.2324 0.0781 0.5205 0.2012 0.5020 0.5146
0 0 0.2500 0.7598 -0.1924 0.3154 0.2549 -0.4063
0 0 0 0.3193 0.3926 -0.8369 0.0762 -0.1963

Matrix Q_fix :

-392 467 479 77 -470 -285 -163 340
-807 -437 -264 231 136 97 -129 196
78 309 -752 222 -363 -112 354 250
-356 -94 100 -139 342 196 -99 -451
330 -691 196 448 223 136 -701 329
0 -189 -238 80 533 206 514 527
0 0 256 778 -197 323 261 -416
0 0 0 327 402 -857 78 -201

Matrix R_float :

247.3297 -73.5415 80.2087 143.7676
-0.0000 194.2618 42.1888 22.5000
0.0000 0.0000 144.8919 -37.0943
0.0000 -0.0000 -0.0000 94.3326
0.0000 0.0000 -0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 -0.0000

DataTypeMode: Fixed-point: binary point scaling

Signedness: Signed

WordLength: 12

FractionLength: 0

Matrix R_fix :

248.0000 -74.2500 79.7500 143.0000
0 195.2500 42.0000 22.7500
0 0 146.5000 -37.5000
0 0 0 94.0000
0 0 0 0
0 0 0 0
0 0 0 0

Matrix R_float :

989.3190 -294.1660 320.8349 575.0704
-0.0000 777.0472 168.7551 89.9999
0.0000 0.0000 579.5677 -148.3771
0.0000 -0.0000 -0.0000 377.3306
0.0000 0.0000 -0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 0.0000
-0.0000 0.0000 0.0000 -0.0000

Matrix R_fix :

992 -297 319 572
0 781 168 91
0 0 586 -150
0 0 0 376
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

DataTypeMode: Fixed-point: binary point scaling

Signedness: Signed

WordLength: 12

FractionLength: 0

● Matlab result

Record the values during iteration, which can be used for debugging in hardware implementation.

k = 1 row78:

```
d = 1 1 -1 1
GG1 Iteration 4 times: input: X = 152; Y = -328 output: X = 594; Y = -5
GR11 Iteration 4 times: input: X = 52; Y = -48 output: X = 108; Y = 43
GR12 Iteration 4 times: input: X = -128; Y = -284 output: X = 331; Y = -390
GR13 Iteration 4 times: input: X = 200; Y = -8 output: X = 153; Y = 291
Q11 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q12 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q13 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q14 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q15 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q16 Iteration 4 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q17 Iteration 4 times: input: X = -1024; Y = 0 output: X = -720; Y = -1520
Q18 Iteration 4 times: input: X = 0; Y = -1024 output: X = 1520; Y = -720
```

d = 1 -1 -1 -1

```
GG1 Iteration 8 times: input: X = 594; Y = -5 output: X = 596; Y = 1
GR11 Iteration 8 times: input: X = 108; Y = 43 output: X = 107; Y = 45
GR12 Iteration 8 times: input: X = 331; Y = -390 output: X = 334; Y = -388
GR13 Iteration 8 times: input: X = 153; Y = 291 output: X = 150; Y = 293
Q11 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q12 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q13 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q14 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q15 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q16 Iteration 8 times: input: X = 0; Y = 0 output: X = 0; Y = 0
Q17 Iteration 8 times: input: X = -720; Y = -1520 output: X = -711; Y = -1528
Q18 Iteration 8 times: input: X = 1520; Y = -720 output: X = 1528; Y = -708
```

d = -1 0 0

```
GG1 Iteration 12 times: input: X = 832; Y = 3 output: X = 832; Y = 0
MK: X = 505; Y = 0
GR11 Iteration 12 times: input: X = -711; Y = -384 output: X = -713; Y = -381
MK: X = 492; Y = -344
GR12 Iteration 12 times: input: X = 33; Y = 758 output: X = 35; Y = 758
MK: X = 428; Y = -460
GR13 Iteration 12 times: input: X = 499; Y = 355 output: X = 500; Y = 354
MK: X = 492; Y = -204
Q11 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q12 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q13 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q14 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q15 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q16 Iteration 12 times: input: X = 0; Y = 0 output: X = 0; Y = 0
MK: X = 0; Y = 0
Q17 Iteration 12 times: input: X = -247; Y = 1669 output: X = -241; Y = 1670
MK: X = -147; Y = 1014
Q18 Iteration 12 times: input: X = -1669; Y = -246 output: X = -1670; Y = -239
MK: X = -1015; Y = -146
```

k = 4 row45:

```
d = 1 -1 1 1
GG4 Iteration 4 times: input: X = 346; Y = -331 output: X = 784; Y = -57
Q41 Iteration 4 times: input: X = 759; Y = 0 output: X = 960; Y = 794
Q42 Iteration 4 times: input: X = 97; Y = 321 output: X = -212; Y = 507
Q43 Iteration 4 times: input: X = -201; Y = 53 output: X = -309; Y = -144
Q44 Iteration 4 times: input: X = 186; Y = -615 output: X = 880; Y = -584
Q45 Iteration 4 times: input: X = 235; Y = 127 output: X = 165; Y = 406
Q46 Iteration 4 times: input: X = 86; Y = -710 output: X = 853; Y = -809
Q47 Iteration 4 times: input: X = -134; Y = 157 output: X = -334; Y = 58
Q48 Iteration 4 times: input: X = -57; Y = 143 output: X = -221; Y = 120
```

d = 1 1 -1 -1

```
GG4 Iteration 8 times: input: X = 784; Y = -57 output: X = 789; Y = -2
Q41 Iteration 8 times: input: X = 960; Y = 794 output: X = 904; Y = 862
Q42 Iteration 8 times: input: X = -212; Y = 507 output: X = -248; Y = 492
Q43 Iteration 8 times: input: X = -309; Y = -144 output: X = -299; Y = -166
Q44 Iteration 8 times: input: X = 880; Y = -584 output: X = 921; Y = -522
Q45 Iteration 8 times: input: X = 165; Y = 406 output: X = 136; Y = 418
Q46 Iteration 8 times: input: X = 853; Y = -809 output: X = 910; Y = -749
Q47 Iteration 8 times: input: X = -334; Y = 58 output: X = -338; Y = 35
Q48 Iteration 8 times: input: X = -221; Y = 120 output: X = -230; Y = 104
```

d = 1 -1 0 0

```
GG4 Iteration 12 times: input: X = 789; Y = -2 output: X = 790; Y = 0
MK: X = 479; Y = 0
Q41 Iteration 12 times: input: X = 904; Y = 862 output: X = 902; Y = 864
MK: X = 547; Y = 524
Q42 Iteration 12 times: input: X = -248; Y = 492 output: X = -249; Y = 492
MK: X = -152; Y = 298
Q43 Iteration 12 times: input: X = -299; Y = -166 output: X = -299; Y = -167
MK: X = -182; Y = -102
Q44 Iteration 12 times: input: X = 921; Y = -522 output: X = 922; Y = -520
MK: X = 560; Y = -316
Q45 Iteration 12 times: input: X = 136; Y = 418 output: X = 135; Y = 418
MK: X = 82; Y = 253
Q46 Iteration 12 times: input: X = 910; Y = -749 output: X = 911; Y = -747
MK: X = 553; Y = -454
Q47 Iteration 12 times: input: X = -338; Y = 35 output: X = -338; Y = 34
MK: X = -206; Y = 20
Q48 Iteration 12 times: input: X = -230; Y = 104 output: X = -230; Y = 104
MK: X = -140; Y = 63
```

Matlab code (only core part):

● Floating-point ($Q^*[A|I] = [R|Q]$)

```
1 % Floating point QR factorization using  $Q^*[A|I] = [R|Q]$ 
2 function R_Q = float_QR(row_R, col_R, row_Q, Q, A)
3     %  $Q^*[A|I] = [R|Q]$ 
4     R_Q = [A Q];
5     % Perform Givens rotations
6     for p_float = 1 : col_R
7         for q_float = (row_R-1) : (-1) : p_float
8             Q = eye(row_Q);
9             theta = atan2(R_Q(q_float+1, p_float), R_Q(q_float, p_float));
10            % Givens Q
11            Q(q_float , q_float ) = cos(theta);
12            Q(q_float , q_float+1) = sin(theta);
13            Q(q_float+1, q_float ) = -sin(theta);
14            Q(q_float+1, q_float+1) = cos(theta);
15            %  $Q^*[A|I] = [R|Q]$ 
16            R_Q = Q * R_Q;
17        end
18    end
19 end
```

- Fixed-point (Cordic Given's rotation)

```

1 % Fixed point QR factorization with the CORDIC
2 function [Q_cordic, R_cordic] = Cordic_QR(K_cordic, Q_scaled, R_scaled, row_R, col_R, col_Q, iter_num,
3 R_sign, R_len, R_frac, Q_sign, Q_len, Q_frac)
4     F = fimath('RoundingMethod','Floor');
5     Q_cordic = Q_scaled;
6     R_cordic = R_scaled;
7
8     % Eliminate A(q+1,p) by A(q,p)
9     for p_fix = 1 : col_R
10        for q_fix = (row_R-1) : (-1) : p_fix
11            % Column q and column q+1 are rotated 180 degrees
12            if R_cordic(q_fix,p_fix) < 0
13                for reverse_R = p_fix : col_R
14                    R_cordic(q_fix ,reverse_R) = -R_cordic(q_fix ,reverse_R);
15                    R_cordic(q_fix+1,reverse_R) = -R_cordic(q_fix+1,reverse_R);
16                end
17                for reverse_Q = p_fix : col_Q
18                    Q_cordic(q_fix ,reverse_Q) = -Q_cordic(q_fix ,reverse_Q);
19                    Q_cordic(q_fix+1,reverse_Q) = -Q_cordic(q_fix+1,reverse_Q);
20                end
21            end
22            x_pre = zeros(col_Q, 1);
23            y_pre = zeros(col_Q, 1);
24            for iter = 0 : iter_num-1
25                % vectoring mode
26                x_vect = R_cordic(q_fix , p_fix);
27                y_vect = R_cordic(q_fix+1, p_fix);
28                [X_vect, Y_vect, d] = GG(x_vect, y_vect, R_len, R_frac, iter);
29
30                if iter == iter_num-1
31                    R_cordic(q_fix, p_fix) = fi((X_vect*K_cordic),R_sign,R_len,R_frac,F);
32                    R_cordic(q_fix+1,p_fix)= fi((Y_vect*K_cordic),R_sign,R_len,R_frac,F);
33                else
34                    R_cordic(q_fix , p_fix) = X_vect;
35                    R_cordic(q_fix+1, p_fix) = Y_vect;
36                end
37
38                % rotation mode
39                for rot_R = 1 : (col_R-p_fix)
40                    x_rot_R = R_cordic(q_fix , p_fix+rot_R);
41                    y_rot_R = R_cordic(q_fix+1, p_fix+rot_R);
42                    [X_rot_R, Y_rot_R] = GR(x_rot_R, y_rot_R, d, R_len, R_frac, iter);
43
44

```

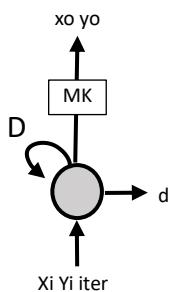
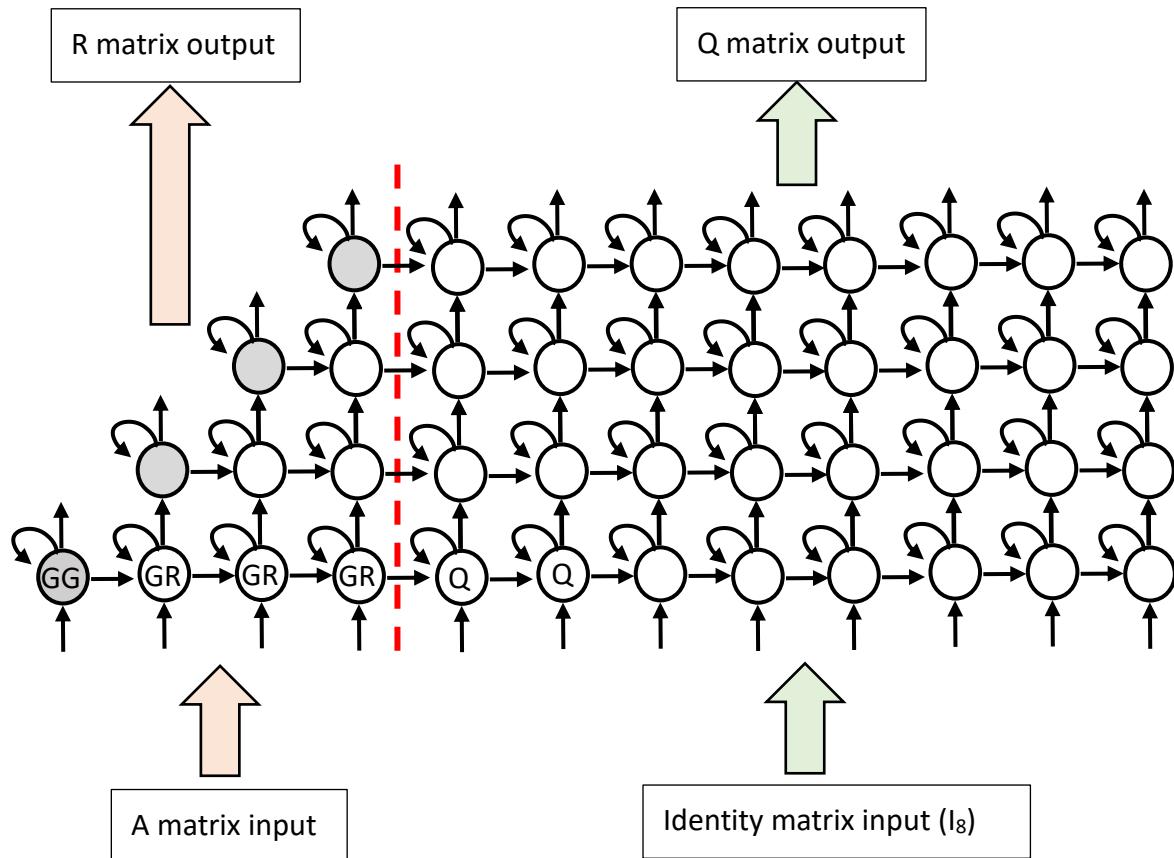
```

45    if iter == iter_num-1
46        R_cordic(q_fix,p_fix+rot_R)=fi((X_rot_R*K_cordic),R_sign,R_len,R_frac,F);
47        R_cordic(q_fix+1,p_fix+rot_R)=fi((Y_rot_R*K_cordic),R_sign,R_len,R_frac,F);
48    else
49        R_cordic(q_fix , p_fix+rot_R) = X_rot_R;
50        R_cordic(q_fix+1, p_fix+rot_R) = Y_rot_R;
51    end
52    end
53    % compute Q (As the processing of R)
54    for rot_Q = 1 : col_Q
55        x_rot_Q = Q_cordic(q_fix , rot_Q);
56        y_rot_Q = Q_cordic(q_fix+1, rot_Q);
57        [X_rot_Q, Y_rot_Q] = GR(x_rot_Q, y_rot_Q, d, Q_len, Q_frac, iter);
58
59    if iter == iter_num-1
60        Q_cordic(q_fix,rot_Q)=fi((X_rot_Q*K_cordic),Q_sign,Q_len,Q_frac,F);
61        Q_cordic(q_fix+1,rot_Q)=fi((Y_rot_Q*K_cordic),Q_sign,Q_len,Q_frac,F);
62    else
63        Q_cordic(q_fix , rot_Q) = X_rot_Q;
64        Q_cordic(q_fix+1, rot_Q) = Y_rot_Q;
65    end
66    end
67    end
68 end
69 end
70end
71
72% GG : vectoring mode
73function [X, Y, d] = GG(x, y, len, frac, iter)
74    d = -sign(x * y);
75    X = x - d * bitsra(y, iter);
76    Y = y + d * bitsra(x, iter);
77
78    F = fimath('RoundingMethod','Floor');
79    X = fi(X, 1, len, frac, F);
80    Y = fi(Y, 1, len, frac, F);
81end
82
83% GR : rotation mode
84function [X, Y] = GR(x, y, d, len, frac, iter)
85    X = x - d * bitsra(y, iter);
86    Y = y + d * bitsra(x, iter);
87
88    F = fimath('RoundingMethod','Floor');
89    X = fi(X, 1, len, frac, F);
90    Y = fi(Y, 1, len, frac, F);
91end

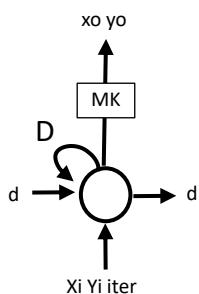
```

● Hardware implementation

1. Hardware architecture ($s=[1 \ 1 \ 0]^t$, $d=[1 \ 0 \ 0]^t$)



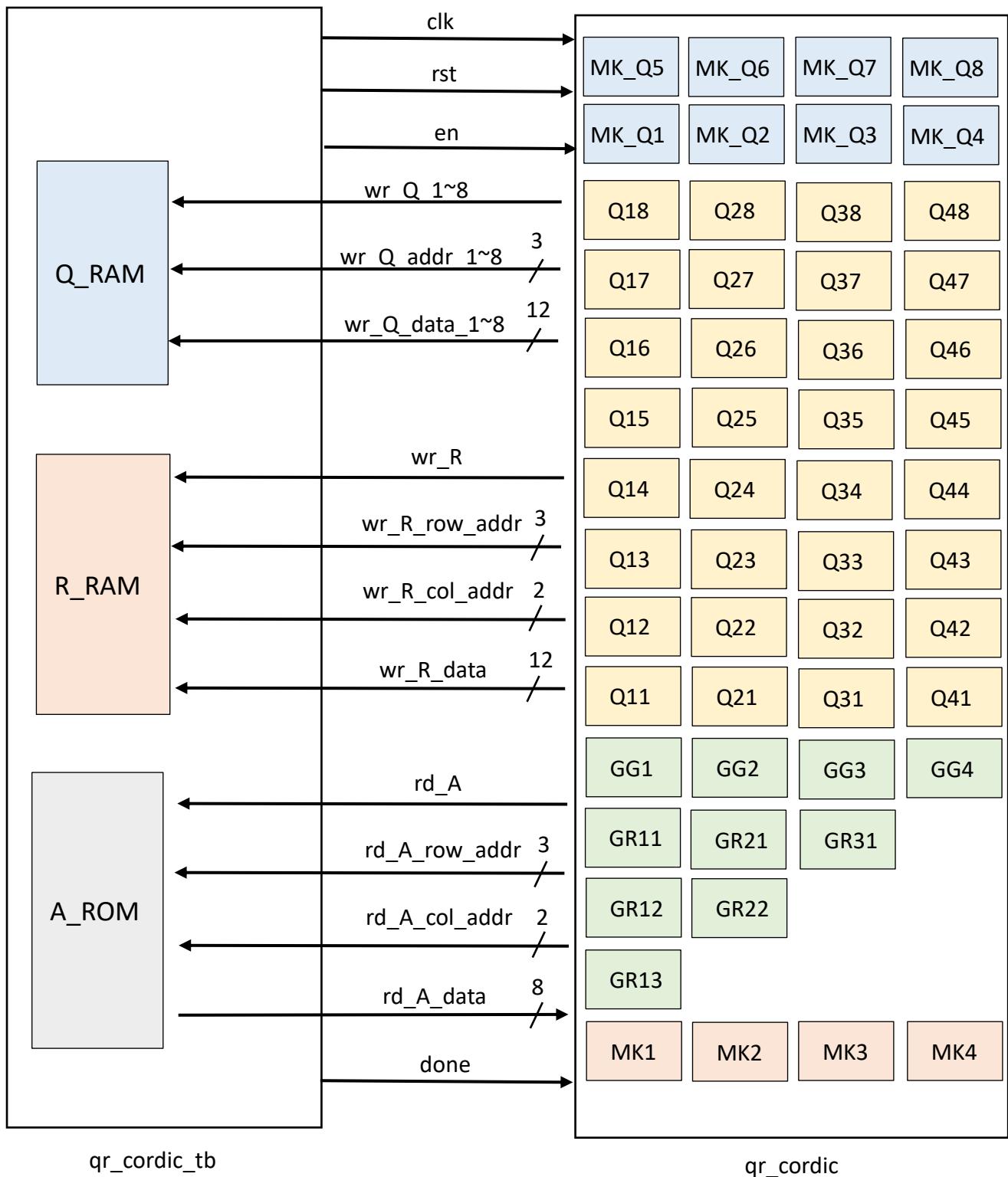
GG(vectoring mode) : One GG contains 4 micro-rotation modules, when iteration over, it outputs to above and multiplied by K(MK module), the rest of data will be transmitted to the right GR and Q modules.



GR(rotation mode) : GR is almost the same module as GG, the only difference is rotation direction d , d will input from the left module(GG or GR).

- There are 4 GG (vectoring mode) and (6+32) GR (rotation mode)
- GG output the direction of vector rotation and transmits it to GR. In each iteration, the rotation angle is halved to approach the target angle.

2. System diagram



- I/O signals description

Signal Name	I/O	Width	Simple Description
clk	I	1	Clock Signal (positive edge trigger)
rst	I	1	Synchronous reset signal (active high)
en	I	1	Enable signal for controlling the module
rd_A	O	1	Read enable signal for A matrix
rd_A_row_addr	O	3	Row address for matrix A
rd_A_col_addr	O	2	Column address for accessing matrix A
rd_A_data	O	8	Data input for matrix A
wr_R	O	1	Write enable signal for R matrix
wr_R_row_addr	O	3	Row address for writing to matrix R
wr_R_col_addr	O	2	Column address for writing to matrix R
wr_R_data	O	12	Data output from matrix R
wr_Q_1~8	O	1	Write enable signals for rows 1 to 8 of matrix Q
wr_Q_addr_1~8	O	3	Column addresses for rows 1 to 8 of matrix Q
wr_Q_data_1~8	O	12	Data outputs for rows 1 to 8 of matrix Q
done	O	1	Signal goes high for one cycle after a QR factorization is complete

3. Verilog code (only core part):

a. GG module

```
1 module GG_one_iter #(
2     parameter R_LEN  = 12,
3     parameter R_FRAC = 2
4 )((
5     input      signed [R_LEN-1:0]    xi,
6     input      signed [R_LEN-1:0]    yi,
7     input          [3:0]           iter,
8     output         [1:0]           d,
9     output reg   signed [R_LEN-1:0]    xo,
10    output reg   signed [R_LEN-1:0]    yo
11 );
12
13 assign d = (yi == 'd0) ? 'd2 : xi[R_LEN-1] ^ yi[R_LEN-1];
14
15 always @(*) begin
16     if(d == 'd2) begin
17         xo = xi;
18         yo = yi;
19     end
20     else if(d == 'd1) begin
21         xo = xi - (yi >>> iter);
22         yo = yi + (xi >>> iter);
23     end
24     else begin
25         xo = xi + (yi >>> iter);
26         yo = yi - (xi >>> iter);
27     end
28 end
29
30 endmodule
```

b. GR module

```
1 module GR_one_iter #(
2     parameter R_LEN  = 12,
3     parameter R_FRAC = 2
4 )
5 (
6     input      signed [R_LEN-1:0]    xi,
7     input      signed [R_LEN-1:0]    yi,
8     input      [3:0]                iter,
9     input      [1:0]                d,
10    output reg   signed [R_LEN-1:0]    xo,
11    output reg   signed [R_LEN-1:0]    yo
12 );
13
14 always @(*) begin
15     if(d == 'd2) begin
16         xo = xi;
17         yo = yi;
18     end
19     else if(d == 'd1) begin
20         xo = xi - (yi >> iter);
21         yo = yi + (xi >> iter);
22     end
23     else begin
24         xo = xi + (yi >> iter);
25         yo = yi - (xi >> iter);
26     end
27 end
28 endmodule
```

c. Q module is the same module as GR

d. MK (multiplied by K)

```
1 module MK #(
2     parameter R_LEN  = 12,
3     parameter R_FRAC = 2,
4     parameter K_LEN  = 10,
5     parameter K_FRAC = 9
6 )(
7     input      signed [R_LEN-1:0]      xi,
8     input      signed [R_LEN-1:0]      yi,
9     output     signed [R_LEN-1:0]      xo,
10    output     signed [R_LEN-1:0]      yo
11 );
12
13 localparam      signed   K = 10'b0_100110111; // K = 0.607421875
14
15 wire      signed [R_LEN+K_LEN-1:0]  xo_0;
16 wire      signed [R_LEN+K_LEN-1:0]  yo_0;
17
18
19 assign xo_0 = xi * K;
20 assign yo_0 = yi * K;
21
22 // truncate to R_LEN bits
23 assign xo = xo_0[R_LEN+K_FRAC-1:K_FRAC];
24 assign yo = yo_0[R_LEN+K_FRAC-1:K_FRAC];
25
26 endmodule
27
```

e. GG iteration control

```
1 // iteration times
2 always @(posedge clk or posedge rst) begin
3     if (rst) begin
4         iter_gg1 <= 'd0;
5     end
6     else if(ROT_wire) begin
7         if(nop_gg1) begin
8             iter_gg1 <= 'd0;
9         end
10        else if(iter_last_gg1) begin
11            iter_gg1 <= iter_gg1 + 'd1;
12        end
13        else begin
14            iter_gg1 <= iter_gg1 + ITER_ONE_CYCLE;
15        end
16    end
17    else begin
18        iter_gg1 <= 'd0;
19    end
20 end
21
22 // GG1 input data xi, yi
23 always @(posedge clk or posedge rst) begin
24     if (rst) begin
25         xi_gg1 <= 'd0;
26         yi_gg1 <= 'd0;
27     end
28     else if(OP_wire) begin
29         case(iter_gg1)
30             0: begin
31                 if(start_gg1) begin
32                     xi_gg1 <= 'd0;
33                     yi_gg1 <= rd_A_data_ext;
34                 end
35                 else if(nop_gg1 && !finish_gg1) begin
36                     xi_gg1 <= rd_A_data_ext;
37                     yi_gg1 <= yo_gg1;
38                 end
39                 else begin
40                     xi_gg1 <= xo_gg1;
41                     yi_gg1 <= yo_gg1;
42                 end
43             end
44             ITER_K: begin
45                 if(finish_gg1) begin
```

```
46          xi_gg1 <= xo_gg1;
47          yi_gg1 <= yo_gg1;
48      end
49      else begin
50          xi_gg1 <= rd_A_data_ext;
51          yi_gg1 <= xo_mk1;
52      end
53  end
54  default: begin
55      xi_gg1 <= xo_gg1;
56      yi_gg1 <= yo_gg1;
57  end
58 endcase
59 end
60 else begin
61     xi_gg1 <= 'd0;
62     yi_gg1 <= 'd0;
63 end
64end
65
66// GG1 mk_count
67always @(posedge clk or posedge rst) begin
68    if (rst) begin
69        mk_count_gg1 <= 'd0;
70    end
71    else if(multk_gg1) begin
72        mk_count_gg1 <= mk_count_gg1 + 'd1;
73    end
74end
```

f. GR iteration control

```
1 // data propagated from left to right
2 always @(posedge clk or posedge rst) begin
3     if (rst) begin
4         iter_gr11    <= 'd0;
5         nop_gr11    <= 'd0;
6         d1_gr11    <= 'd0;
7         d2_gr11    <= 'd0;
8         d3_gr11    <= 'd0;
9         d4_gr11    <= 'd0;
10        neg_gr11   <= 'd0;
11        mk_count_gr11 <= 'd0;
12    end
13    else begin
14        iter_gr11    <= iter_gg1;
15        nop_gr11    <= nop_gg1;
16        d1_gr11    <= d1_gg1;
17        d2_gr11    <= d2_gg1;
18        d3_gr11    <= d3_gg1;
19        d4_gr11    <= d4_gg1;
20        neg_gr11   <= neg_gg1;
21        mk_count_gr11 <= mk_count_gg1;
22    end
23 end
24
25 // GR11 input data xi, yi
26 always @(posedge clk or posedge rst) begin
27     if (rst) begin
28         xi_gr11 <= 'd0;
29         yi_gr11 <= 'd0;
30     end
31     else if(OP_wire) begin
32         case(iter_gr11)
33             0: begin
34                 if(start_gr11_reg) begin
35                     xi_gr11 <= 'd0;
36                     yi_gr11 <= rd_A_data_ext;
37                 end
38                 else if(nop_gr11 && !finish_gr11) begin
39                     xi_gr11 <= rd_A_data_ext;
40                     yi_gr11 <= yo_gr11;
41                 end
42                 else begin
43                     xi_gr11 <= xo_gr11;
44                     yi_gr11 <= yo_gr11;
45                 end

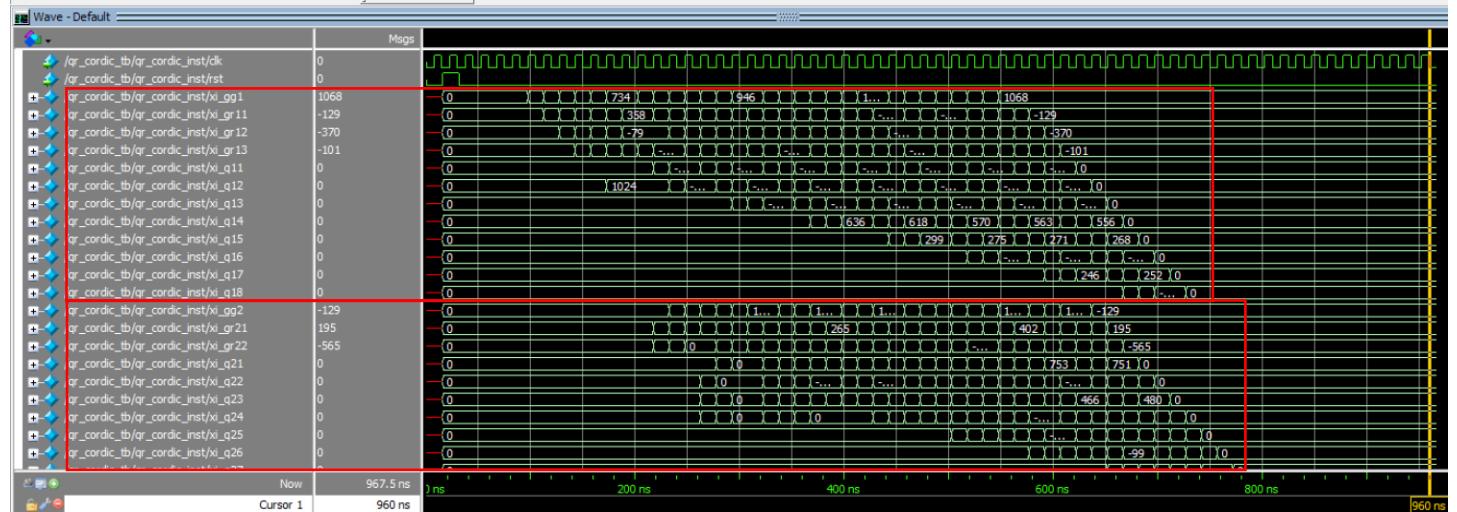
```

```
46      end
47      ITER_K: begin
48          if(finish_gr11) begin
49              xi_gr11 <= xo_mk1;
50              yi_gr11 <= yo_mk1;
51          end
52          else begin
53              xi_gr11 <= rd_A_data_ext;
54              yi_gr11 <= xo_mk1;
55          end
56      end
57      default: begin
58          xi_gr11 <= xo_gr11;
59          yi_gr11 <= yo_gr11;
60      end
61  endcase
62 end
63 else begin
64     xi_gr11 <= 'd0;
65     xi_gr11 <= 'd0;
66 end
67 end
```

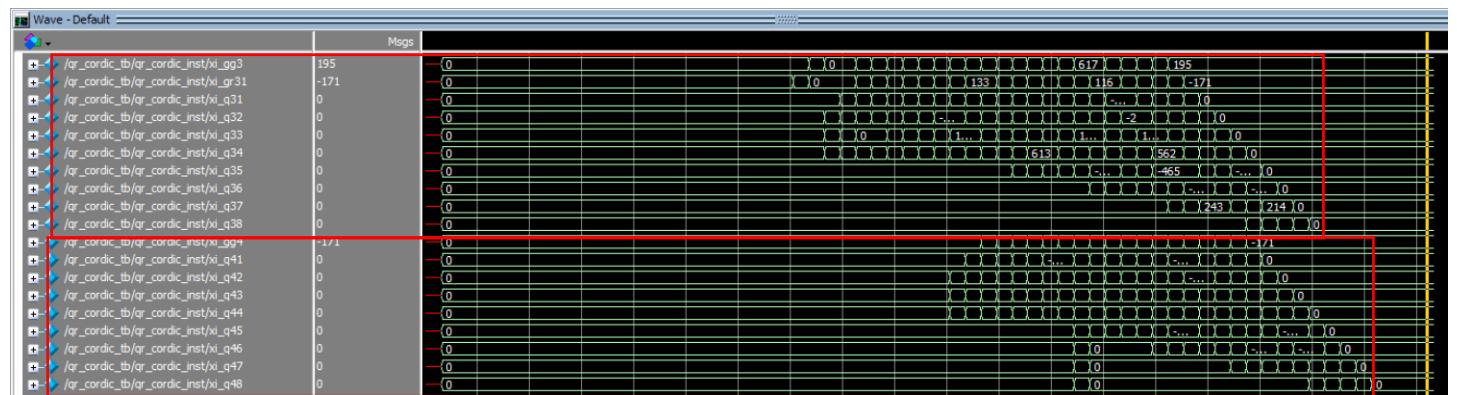
4. RTL Simulation

a. Simulation waveform (Data propagate form left to right PE (GG, GR, Q))

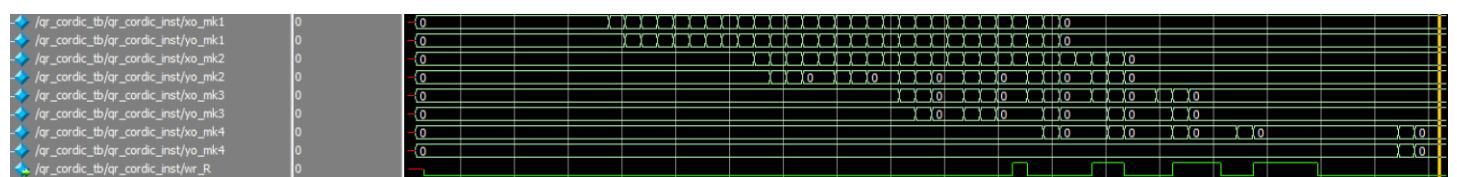
- Data propagate : GG1 → Q18, GG2 → Q28



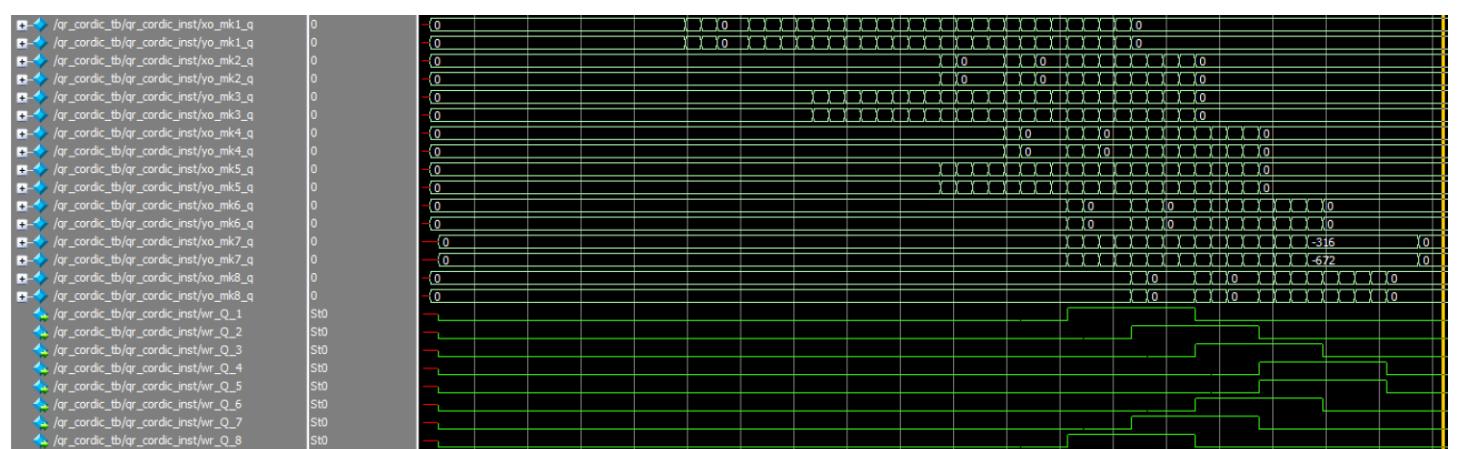
- Data propagate : GG3 → Q38, GG4 → Q48



- Output (MK output R)



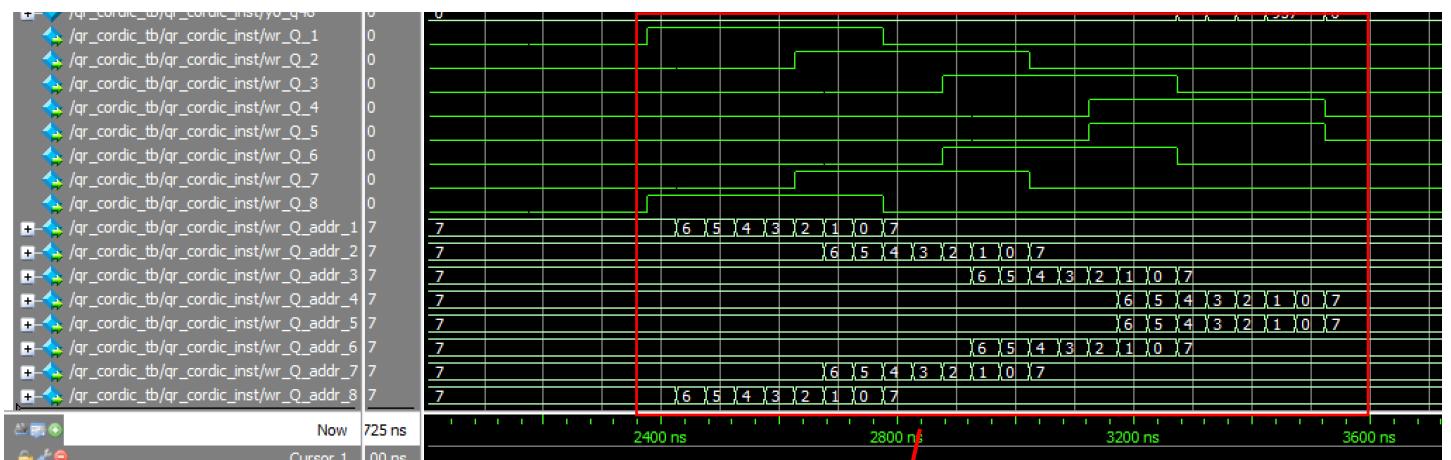
- Output (MK output Q)



- Data flow & data scheduling
GG1 → GR11 → GR12 → GR13 → Q11 → Q12 → Q13 → → Q18
----- GG2 → GR21 → GR22 → Q21 → Q22 → Q23 → → Q28
----- GG3 → GR31 → Q31 → Q32 → Q33 → → Q38
----- GG4 → Q41 → Q42 → Q43 → → Q48

- Corresponding Timing diagram

- Write Q matrix data back



Each row has an independent write enable signal, since multiple data outputs from Q occur simultaneously in the systolic array.

- Corresponding Timing diagram

b. Simulation transcript (4 patterns)

Pattern 1:

```
START!!! Simulation Start .....
```

Input A matrix:

75	63	127	94
-49	-13	-108	-107
7	-107	-15	-26
-86	70	-101	-62
16	105	118	76
-61	-89	-127	-18
39	83	70	105
48	9	81	-82

Output R matrix golden pattern:

620	586	1061	535
0	637	238	318
0	0	304	89
0	0	0	623
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

R matrix calculated result:

620	586	1061	535
0	637	238	318
0	0	304	89
0	0	0	623
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Output Q matrix golden pattern:

495	-325	44	-572	176	-411	263	320
-48	215	-735	77	520	-195	300	-242
12	500	216	577	580	-130	-212	166
-213	-553	660	-42	-2	353	347	-720
-869	-54	-93	330	-84	136	-218	-23
0	-580	-92	-70	443	-62	255	-154
0	0	-113	351	-423	-787	168	-300
9	0	0	-381	181	-251	-768	-472

Q matrix calculated result:

495	-325	44	-572	176	-411	263	320
-48	215	-735	77	520	-195	300	-242
12	500	216	577	580	-130	-212	166
-213	-553	660	-42	-2	353	347	-720
-869	-54	-93	330	-84	136	-218	-23
0	-580	-92	-70	443	-62	255	-154
0	0	-113	351	-423	-787	168	-300
9	0	0	-381	181	-251	-768	-472

SUMMARY

```
*****
** Congratulations!! R matrix data have been generated successfully! **
** Congratulations!! Q matrix data have been generated successfully! **
*****
** The simulation results are all Pass!! **
** Get finish at cycle: 57 **
*****
```

Congratulations !!
Simulation PASS!!

```
START!!! Simulation Start .....
```

Input A matrix:

-95	-125	-53	-108
113	-117	62	109
116	-85	-80	70
19	38	47	-4
-113	59	-52	17
68	37	-34	14
-38	-13	32	-50
82	12	71	2

Output R matrix golden pattern:

992	-299	318	572
0	775	164	86
0	0	584	-149
0	0	0	375
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

R matrix calculated result:

992	-299	318	572
0	775	164	86
0	0	584	-149
0	0	0	375
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Output Q matrix golden pattern:

-39	461	491	77	-472	-285	-162	337
-813	-441	-270	227	130	85	-130	182
72	309	-754	224	-363	-110	360	248
-359	-84	99	-140	343	200	-91	-457
324	-691	195	465	219	121	-694	315
0	-189	-223	43	536	196	495	553
0	0	276	779	-161	314	306	-392
0	0	0	0	304	406	-867	91
0	0	0	0	0	0	-186	0

Q matrix calculated result:

-39	461	491	77	-472	-285	-162	337
-813	-441	-270	227	130	85	-130	182
72	309	-754	224	-363	-110	360	248
-359	-84	99	-140	343	200	-91	-457
324	-691	195	465	219	121	-694	315
0	-189	-223	43	536	196	495	553
0	0	276	779	-161	314	306	-392
0	0	0	0	304	406	-867	91
0	0	0	0	0	0	-186	0

SUMMARY

```
*****
** Congratulations!! R matrix data have been generated successfully! **
** Congratulations!! Q matrix data have been generated successfully! **
*****
** The simulation results are all Pass!! **
** Get finish at cycle: 57 **
*****
```

Congratulations !!
Simulation PASS!!

```
$finish called from file "qr_cordic_tb.v", line 239.
$finish at simulation time 96750
          VCS Simulation Report
Time: 967500 ps
CPU Time: 0.510 seconds;      Data structure size: 0.1MB
Sat May 25 23:13:26 2024
CPU time: 1.381 seconds to compile + .577 seconds to elab + .458 seconds to link + .557 seconds in simulation
```

```
$finish called from file "qr_cordic_tb.v", line 239.
$finish at simulation time 96750
          VCS Simulation Report
Time: 967500 ps
CPU Time: 0.500 seconds;      Data structure size: 0.1MB
Sat May 25 23:15:04 2024
CPU time: 1.368 seconds to compile + .552 seconds to elab + .452 seconds to link + .556 seconds in simulation
```

Pattern 3:

```

START!!! Simulation Start .....

Input A matrix:
-24   -47    58   -69
-24   -20    -3   -3
-61     2    28   31
26   -107   55    45
54   -61   -11   -27
-72     77   118   -34
-98  -121    11   124
-53   109     5  -119

Output R matrix golden pattern:
 648  -129  -370  -101
 0    891   195  -565
 0     0   443  -171
 0     0     0   521
 0     0     0     0
 0     0     0     0
 0     0     0     0
 0     0     0     0

R matrix calculated result:
 648  -129  -370  -101
 0    891   195  -565
 0     0   443  -171
 0     0     0   521
 0     0     0     0
 0     0     0     0
 0     0     0     0

Output Q matrix golden pattern:
 -152   153  -388   162   337  -461  -630  -342
 -237   -72   -47  -474  -241   291  -647  -456
 522   129  -115  -286  -281   603  -134  -448
 -653   -34   -86  -224  -316  -155   109  -664
 531  -147  -113  -147  -672  -395  -194  -260
 0    991    51   -51  -220  -44    41   84
 0     0  -933    48  -279  -76  -306  -100
 0     0     0  -803   343  -458   282   44

Q matrix calculated result:
 -152   153  -388   162   337  -461  -630  -342
 -237   -72   -47  -474  -241   291  -647  -456
 522   129  -115  -286  -281   603  -134  -448
 -653   -34   -86  -224  -316  -155   109  -664
 531  -147  -113  -147  -672  -395  -194  -260
 0    991    51   -51  -220  -44    41   84
 0     0  -933    48  -279  -76  -306  -100
 0     0     0  -803   343  -458   282   44

----- SUMMARY -----
***** Congratulations!! R matrix data have been generated successfully!
** Congratulations!! Q matrix data have been generated successfully! **
***** The simulation results are all Pass!! **
** Get finish at cycle: 57 **
***** Simulation PASS!! *****
----- / 0.0 -----
----- \m_n\_w -----
```

\$finish called from file "qr_cordic_tb.v", line 239.
\$finish at simulation time 96750
VCS Simulation Report
Time: 967500 ps
CPU Time: 0.510 seconds; Data structure size: 0.1Mb
Sat May 25 23:15:48 2024
CPU time: 1.372 seconds to compile + .552 seconds to elab + .456 seconds to link + .562 seconds in simulation

Pattern4:

```

START!!! Simulation Start .....

Input A matrix:
 -34    -19    -54
 127   -11     -5
 -85   123   -98
 -120   -88    22
 15     91   -71
 97     37   -30
 43   -32    21
 -80   -80   -64

Output R matrix golden pattern:
 1020   185   137   121
 0    792  -368  -40
 0     0   429  -234
 0     0     0   698
 0     0     0     0
 0     0     0     0
 0     0     0     0

R matrix calculated result:
 1020   185   137   121
 0    792  -368  -40
 0     0   429  -234
 0     0     0   698
 0     0     0     0
 0     0     0     0
 0     0     0     0

Output Q matrix golden pattern:
 -386   509   344  -487
 84    178    71   346
 125   -315  -213   64
 298    81  -330   575
 -892  -225  -225  -41
 0    -774   304   317
 0     0   352   505
 0     0     0   24
 0     0     0   456

Q matrix calculated result:
 -386   509   344  -487
 84    178    71   346
 125   -315  -213   64
 298    81  -330   575
 -892  -225  -225  -41
 0    -774   304   317
 0     0   352   505
 0     0     0   24
 0     0     0   456

----- SUMMARY -----
***** Congratulations!! R matrix data have been generated successfully!
** Congratulations!! Q matrix data have been generated successfully! **
***** The simulation results are all Pass!! **
** Get finish at cycle: 57 **
***** Simulation PASS!! *****
----- / 0.0 -----
----- \m_n\_w -----
```

\$finish called from file "qr_cordic_tb.v", line 239.
\$finish at simulation time 96750
VCS Simulation Report
Time: 967500 ps
CPU Time: 0.510 seconds; Data structure size: 0.1Mb
Sat May 25 23:13:16 2024
CPU time: 1.571 seconds to compile + .579 seconds to elab + .452 seconds to link + .563 seconds in simulation

- Both R and Q are all pass in RTL simulation
- Clock rate : 15ns
- Cycles : 57
- Run time : 967.5ns

RTL simulation Commands (VCS):

```
vcs -full64 -R -sverilog qr_cordic_tb.v qr_cordic.v +access+r +vcs+fsdbon +fsdb+mda
+fsdbfile+qr_cordic.fsdb
```

5. Synthesis

a. Synopsys Design Constraints (SDC)

```
1 # operating conditions and boundary conditions #
2
3 set_cycle 15.0
4 create_clock -name clk -period $cycle [get_ports clk]
5
6 set_dont_touch_network [all_clocks]
7 set_fix_hold [all_clocks]
8 set_clock_uncertainty 0.1 [all_clocks]
9 set_clock_latency 0.5 [all_clocks]
10 set_ideal_network [get_ports clk]
11
12
13 #Don't touch the basic env setting as below
14 set_input_delay 1 -clock clk [remove_from_collection [all_inputs] [get_ports clk]]
15 set_output_delay 1 -clock clk [all_outputs]
16
17 set_load 1 [all_outputs]
18 set_drive 0.1 [all_inputs]
19
20 set_operating_conditions -max_library slow -max slow
21 set_wire_load_model -name tsmc13_wl10 -library slow
22 set_max_fanout 20 [all_inputs]
23
```

- Clock rate = 15ns
- I/O delay = 1ns

b. Design compiler synthesis .tcl

```
1 #Read All Files
2 read_file -format verilog qr_cordic.v
3 current_design qr_cordic
4 link
5
6 #Setting Clock Constraints
7 source -echo -verbose qr_cordic.sdc
8 check_design
9 set high_fanout_net_threshold 0
10 uniquify
11 set_fix_multiple_port_nets -all -buffer_constants [get_designs *]
12 #set_max_area 0
13 #Synthesis all design
14 #compile -map_effort high -area_effort high
15 #compile -map_effort high -area_effort high -inc
16 #compile_ultra
17 #compile_ultra -area
18 compile
19
20 write -format ddc -hierarchy -output "qr_cordic_syn.ddc"
21 write_sdf -version 1.0 qr_cordic_syn.sdf
22 write -format verilog -hierarchy -output qr_cordic_syn.v
23 report_area > area.log
24 report_timing > timing.log
25 report_qor > qr_cordic_syn.qor
26
```

- Use Standard synthesis run without specifying additional optimization efforts for mapping or area.
- Synthesis on the CBDK_IC_Contest_v2.5 and TSMC 130nm process

c. Timing report

```

1 ****
2 Report : timing
3   -path full
4   -delay max
5   -max_paths 1
6 Design : qr_cordic
7 Version: U-2022.12
8 Date  : Sat May 25 21:21:51 2024
9 ****
10 Operating Conditions: slow Library: slow
11 Wire Load Model Mode: top
12 Path Group: clk
13 Path Type: max
14 Startpoint: iter_q28_reg[0]
15   (rising edge-triggered flip-flop clocked by clk)
16 Endpoint: yi_mk4_q_reg[8]
17   (rising edge-triggered flip-flop clocked by clk)
18 Path Group: clk
19 Path Type: max
20 Des/Clust/Port      Wire Load Model      Library
21 -----
22 qr_cordic          tsmc13_wl10        slow
23 -----
24 Point               Incr      Path
25 -----
26 clock clk (rise edge)    0.00    0.00
27 clock network delay (ideal) 0.50    0.50
28 iter_q28_reg[0]/CK (DFFRX4) 0.00    0.50 r
29 iter_q28_reg[0]/Q (DFFRX4)  0.61    1.11 f
30 Q28_inst7/iter[0] (Q_16)    0.00    1.11 f
31 Q28_inst/Q_one_iter_inst_1/iter[0] (Q_one_iter_67) 0.00    1.11 f
32 Q28_inst/Q_one_iter_inst_1/U162/Y (INVX4) 0.17    1.27 r
33 Q28_inst/Q_one_iter_inst_1/U122/Y (NAND2X8) 0.20    1.48 f
34 Q28_inst/Q_one_iter_inst_1/U45/Y (INVX4) 0.30    1.78 r
35 Q28_inst/Q_one_iter_inst_1/U11/Y (MX2X4) 0.33    2.11 f
36 Q28_inst/Q_one_iter_inst_1/U61/Y (AOI221X4) 0.48    2.59 r
37 Q28_inst/Q_one_iter_inst_1/sub_100/B[6] (Q_one_iter_67_DW01_sub_3) 0.00    2.59 r
38 Q28_inst/Q_one_iter_inst_1/sub_100/U126/Y (CLKINVX1) 0.24    2.83 f
39 Q28_inst/Q_one_iter_inst_1/sub_100/U128/Y (NOR2X2) 0.43    3.26 r
40 Q28_inst/Q_one_iter_inst_1/sub_100/U163/Y (NOR2X4) 0.19    3.46 f
41 Q28_inst/Q_one_iter_inst_1/sub_100/U165/Y (AOI21X4) 0.18    3.63 r
42 Q28_inst/Q_one_iter_inst_1/sub_100/U185/Y (AOI21X1) 0.15    3.78 f
43 Q28_inst/Q_one_iter_inst_1/sub_100/U185/Y (XOR2X2) 0.27    4.06 r
44 Q28_inst/Q_one_iter_inst_1/sub_100/DIFF[10] (Q_one_iter_67_DW01_sub_3) 0.21    4.27 r
45 Q28_inst/Q_one_iter_inst_1/xo[10] (Q_one_iter_67) 0.00    4.27 r
46 Q28_inst/Q_one_iter_inst_1/U144/Y (AO22X2) 0.26    4.53 r
47 Q28_inst/Q_one_iter_inst_1/U180/Y (AO21X4) 0.28    4.81 r
48 Q28_inst/Q_one_iter_inst_1/xo[10] (Q_one_iter_66) 0.00    4.81 r
49 Q28_inst/Q_one_iter_inst_2/U117/Y (INVX8) 0.11    4.92 f
50 Q28_inst/Q_one_iter_inst_2/U28/Y (AOI222X4) 0.55    5.47 r
51 Q28_inst/Q_one_iter_inst_2/U27/Y (INVX4) 0.08    5.55 f
52 Q28_inst/Q_one_iter_inst_2/U168/Y (AOI2BB1X2) 0.22    5.77 f
53 Q28_inst/Q_one_iter_inst_2/U21/Y (AOI221X4) 0.34    6.11 r
54 Q28_inst/Q_one_iter_inst_2/sub_105/B[1] (Q_one_iter_66_DW01_sub_2) 0.00    6.11 r
55 Q28_inst/Q_one_iter_inst_2/sub_105/U128/Y (INVX3) 0.10    6.21 f
56 Q28_inst/Q_one_iter_inst_2/sub_105/U185/Y (NAND2X1) 0.26    6.46 r
57 Q28_inst/Q_one_iter_inst_2/sub_105/U132/Y (AO21X2) 0.36    6.83 r
58 Q28_inst/Q_one_iter_inst_2/sub_105/U154/Y (AO21X4) 0.23    7.06 r
59 Q28_inst/Q_one_iter_inst_2/sub_105/U143/Y (AOI21X4) 0.15    7.20 f
60 Q28_inst/Q_one_iter_inst_2/sub_105/U131/Y (AOI21X1) 0.28    7.48 r
61 Q28_inst/Q_one_iter_inst_2/sub_105/U166/Y (XOR2X2) 0.21    7.69 r
62 Q28_inst/Q_one_iter_inst_2/sub_105/DIFF[10] (Q_one_iter_66_DW01_sub_2) 0.00    7.69 r
63 Q28_inst/Q_one_iter_inst_2/U130/Y (AO22X2) 0.22    7.92 r
64 Q28_inst/Q_one_iter_inst_2/U177/Y (AO21X4) 0.28    8.20 r
65 Q28_inst/Q_one_iter_inst_2/yo[10] (Q_one_iter_66) 0.00    8.20 r
66 Q28_inst/Q_one_iter_inst_3/yi[10] (Q_one_iter_65) 0.00    8.20 r
67 Q28_inst/Q_one_iter_inst_3/U19/Y (CLKINVX8) 0.16    8.35 f
68 Q28_inst/Q_one_iter_inst_3/U16/Y (OR2X1) 0.36    8.71 f
69 Q28_inst/Q_one_iter_inst_3/U18/Y (NAND3X4) 0.15    8.86 r
70 Q28_inst/Q_one_iter_inst_3/U22/Y (INVX3) 0.10    8.96 f
71 Q28_inst/Q_one_iter_inst_3/U62/Y (AOI221X4) 0.47    9.44 r
72 Q28_inst/Q_one_iter_inst_3/sub_100/B[0] (Q_one_iter_65_DW01_sub_3) 0.00    9.44 r
73 Q28_inst/Q_one_iter_inst_3/sub_100/U116/Y (INVX4) 0.09    9.52 f
74 Q28_inst/Q_one_iter_inst_3/sub_100/U132/Y (NOR2X2) 0.21    9.73 r
75 Q28_inst/Q_one_iter_inst_3/sub_100/U130/Y (AO21X4) 0.25    9.98 r
76 Q28_inst/Q_one_iter_inst_3/sub_100/U170/Y (AOA21X4) 0.21    10.20 r

```

86	Q28_inst/Q_one_iter_inst_3/sub_100/U155/Y (INVX6)	0.10	10.30 f
87	Q28_inst/Q_one_iter_inst_3/sub_100/U121/Y (AOI21X2)	0.19	10.49 r
88	Q28_inst/Q_one_iter_inst_3/sub_100/U133/Y (XOR2X4)	0.17	10.66 f
89	Q28_inst/Q_one_iter_inst_3/sub_100/DIFF[5] (Q_one_iter_65_DW01_sub_3)	0.00	10.66 f
90			
91	Q28_inst/Q_one_iter_inst_3/U8/Y (AO22X4)	0.25	10.91 f
92	Q28_inst/Q_one_iter_inst_3/U183/Y (AO21X4)	0.31	11.23 f
93	Q28_inst/Q_one_iter_inst_3/xo[5] (Q_one_iter_64)	0.00	11.23 f
94	Q28_inst/Q_one_iter_inst_4/xi[5] (Q_one_iter_64)	0.00	11.23 f
95	Q28_inst/Q_one_iter_inst_4/U66/Y (INVX1)	0.30	11.52 r
96	Q28_inst/Q_one_iter_inst_4/U151/Y (AO22X1)	0.33	11.85 r
97	Q28_inst/Q_one_iter_inst_4/U45/Y (AOI221X4)	0.32	12.17 f
98	Q28_inst/Q_one_iter_inst_4/U44/Y (INVX4)	0.07	12.25 r
99	Q28_inst/Q_one_iter_inst_4/U108/Y (AOI221X4)	0.36	12.60 f
100	Q28_inst/Q_one_iter_inst_4/sub_105/B[1] (Q_one_iter_64_DW01_sub_2)	0.00	12.60 f
101			
102	Q28_inst/Q_one_iter_inst_4/sub_105/U136/Y (INVX3)	0.13	12.73 r
103	Q28_inst/Q_one_iter_inst_4/sub_105/U162/Y (NOR2X4)	0.10	12.84 f
104	Q28_inst/Q_one_iter_inst_4/sub_105/U137/Y (AOI21X2)	0.35	13.18 r
105	Q28_inst/Q_one_iter_inst_4/sub_105/U165/Y (AOI21X4)	0.21	13.39 f
106	Q28_inst/Q_one_iter_inst_4/sub_105/U164/Y (AOI21X4)	0.27	13.66 r
107	Q28_inst/Q_one_iter_inst_4/sub_105/U113/Y (XOR2X2)	0.24	13.98 f
108	Q28_inst/Q_one_iter_inst_4/sub_105/DIFF[8] (Q_one_iter_64_DW01_sub_2)	0.00	13.90 f
109			
110	Q28_inst/Q_one_iter_inst_4/U78/Y (AO22X4)	0.21	14.11 f
111	Q28_inst/Q_one_iter_inst_4/U171/Y (AO21X1)	0.30	14.42 f
112	Q28_inst/Q_one_iter_inst_4/yo[8] (Q_one_iter_64)	0.00	14.42 f
113	Q28_inst/U117/Y (AO22X1)	0.42	14.84 f
114	Q28_inst/yo[8] (Q_16)	0.00	14.84 f
115	U5086/Y (CLKINVX3)	0.14	14.98 r
116	U7140/Y (AOI221X2)	0.16	15.13 f
117	yi_mk4_q_reg[8]/D (DFFSRHQX1)	0.00	15.13 f
118	data arrival time		15.13
119			
120	clock clk (rise edge)	15.00	15.00
121	clock network delay (ideal)	0.50	15.50
122	clock uncertainty	-0.10	15.40
123	yi_mk4_q_reg[8]/CK (DFFSRHQX1)	0.00	15.40 r
124	library setup time	-0.27	15.13
125	data required time		15.13
126			
127	data required time		15.13
128	data arrival time		-15.13
129			
130	slack (MET)	0.00	
131			
132			
133	1		
134			

- Clock rate = 15ns
- Slack \geq 0ns

d. Area report

```
1
2 ****
3 Report : area
4 Design : qr_cordic
5 Version: U-2022.12
6 Date  : Sat May 25 21:21:51 2024
7 ****
8
9 Library(s) Used:
10
11    slow (File: /home/cell_library/CBDK_IC_Contest_v2.5/SynopsysDC/db/slow.db)
12
13 Number of ports:          39008
14 Number of nets:           145079
15 Number of cells:          107729
16 Number of combinational cells: 103742
17 Number of sequential cells: 2096
18 Number of macros/black boxes: 0
19 Number of buf/inv:         25537
20 Number of references:     191
21
22 Combinational area:      1187614.757948
23 Buf/Inv area:            176923.394568
24 Noncombinational area:   79423.041847
25 Macro/Black Box area:    0.000000
26 Net Interconnect area:  11491491.069855
27
28 Total cell area:        1267037.799795
29 Total area:              12758528.869650
30
31
```

- Area: 12567037.8 μm^2 (clock rate:15ns, standard compile)

e. Power

```
20 Operating Conditions: slow  Library: slow
21 Wire Load Model Mode: top
22
23 Design      Wire Load Model      Library
24 -----
25 qr_cordic    tsmc13_wl10      slow
26
27
28 Global Operating Voltage = 1.08
29 Power-specific unit information :
30   Voltage Units = 1V
31   Capacitance Units = 1.000000pf
32   Time Units = 1ns
33   Dynamic Power Units = 1mW    (derived from V,C,T units)
34   Leakage Power Units = 1pW
35
36
37 Attributes
38 -----
39 i - Including register clock pin internal power
40
41
42 Cell Internal Power = 4.4245 mW  (63%)
43 Net Switching Power = 2.5960 mW  (37%)
44
45 Total Dynamic Power = 7.0205 mW (100%)
46
47 Cell Leakage Power = 1.0694 mW
48
49
50      Internal      Switching      Leakage      Total
51 Power Group    Power       Power       Power      Power  ( % ) Atts
52 -----
53 io_pad        0.0000      0.0000      0.0000      0.0000 ( 0.00% )
54 memory        0.0000      0.0000      0.0000      0.0000 ( 0.00% )
55 black_box     0.0000      0.0000      0.0000      0.0000 ( 0.00% )
56 clock_network 3.5668      0.0000      0.0000      3.5668 ( 44.09% ) i
57 register      6.8604e-02  8.5020e-02  7.0488e-07  0.2241 ( 2.77% )
58 sequential     0.0000      0.0000      0.0000      0.0000 ( 0.00% )
59 combinational  0.7891      2.5108      9.9895e+08  4.2990 ( 53.14% )
60
61 Total          4.4245 mW    2.5959 mW    1.0694e+09 pW    8.0899 mW
62 1
```

- Total Power : 8.0899mW

f. Gate-level simulation (Run time : 967.5ns)

Pattern 1:

```

START!!! Simulation Start .....

Input A matrix:
 -95   -125   -53   -108
 113   -117    62   109
 116    -85   -80    70
 19     38    47    -4
 -113    59   -82   -17
 -68    37   -34   -14
 -38    13    32   -50
 82     12    71     2

Output R matrix golden pattern:
 992   -299    318   572
 0     775   164    86
 0     0     584  -149
 0     0     0    375
 0     0     0     0
 0     0     0     0
 0     0     0     0
 0     0     0     0

R matrix calculated result:
 992   -299    318   572
 0     775   164    86
 0     0     584  -149
 0     0     0    375
 0     0     0     0
 0     0     0     0
 0     0     0     0

Output Q matrix golden pattern:
 -392    467    481    77   -472   -285   -162    337
 -813   -441   -270   227   130    85   -130   182
 -72    -309   -754   224   -363   -110   360   248
 -359   -84    99  -140   343   200   -91   -457
 324   -691   195   465   219   121   -694   315
 0     -189   -223   43    536   196   495   553
 0     0     276   779  -161   314   306  -392
 0     0     0     0     0     0     0    -186
 0     0     0     0     304   406  -867   91

Q matrix calculated result:
 -392    467    481    77   -472   -285   -162    337
 -813   -441   -270   227   130    85   -130   182
 -72    -309   -754   224   -363   -110   360   248
 -359   -84    99  -140   343   200   -91   -457
 324   -691   195   465   219   121   -694   315
 0     -189   -223   43    536   196   495   553
 0     0     276   779  -161   314   306  -392
 0     0     0     0     304   406  -867   91

***** S U M M A R Y *****
*****
** Congratulations!! R matrix data have been generated successfully! **
** Congratulations!! O matrix data have been generated successfully! **
*****
** The simulation results are all Pass! **
** Get finish at cycle: 57 **
*****



.....
-- Congratulations !!
-- / 0.0
-- Simulation PASS!!
-- ( \^ ^\ ) |w|
-- \m _ m_ |_

```

```

$finish called from file "qr_cordic_tb.v", line 239.
$finish at simulation time 967500
VCS Simulation Report
Time: 967500 Ps
CPU Time: 4.160 seconds; Data structure size: 18.0MB
Sat May 25 23:17:49 2024
CPU time: 16.593 seconds to compile + 3.690 seconds to elab + 1.084 seconds to link + 4.225 seconds in simulation

```

Pattern 2:

```

START!!! Simulation Start .....

Input A matrix:
 75    -68    127    94
 -49    -13   -108  -107
 7     -107   -15    -26
 -86    -70   -101    -62
 26    105    118    76
 -61    -89   -127    -18
 39     83     70   105
 48      9     81  -82

Output R matrix golden pattern:
 620    586   1061   535
 0     637    238   318
 0     0     304    89
 0     0     0   623
 0     0     0     0
 0     0     0     0
 0     0     0     0
 0     0     0     0

R matrix calculated result:
 620    586   1061   535
 0     637    238   318
 0     0     304    89
 0     0     0   623
 0     0     0     0
 0     0     0     0
 0     0     0     0

Output Q matrix golden pattern:
 495   -325    44   -572    176   -411   263   320
 -48    215   -735    77    520   -195   300  -242
 12     500    216   577    580   -130   -212   166
 -213   -535    660   -42    -72   -353   347  -720
 -869   -54    93   330   -84    136   -218   -23
 0     -580    -92   -70    443   -62   -255  -154
 0     0     -113   351  -423   -787   168  -300
 0     0     0     0     -381   181  -251  -768  -472

Q matrix calculated result:
 495   -325    44   -572    176   -411   263   320
 -48    215   -735    77    520   -195   300  -242
 12     500    216   577    580   -130   -212   166
 -213   -535    660   -42    -72   -353   347  -720
 -869   -54    93   330   -84    136   -218   -23
 0     -580    -92   -70    443   -62   -255  -154
 0     0     -113   351  -423   -787   168  -300
 0     0     0     0     -381   181  -251  -768  -472

***** S U M M A R Y *****
*****
** Congratulations!! R matrix data have been generated successfully! **
** Congratulations!! O matrix data have been generated successfully! **
*****
** The simulation results are all Pass! **
** Get finish at cycle: 57 **
*****



.....
-- Congratulations !!
-- / 0.0
-- Simulation PASS!!
-- ( \^ ^\ ) |w|
-- \m _ m_ |_

```

```

$finish called from file "qr_cordic_tb.v", line 239.
$finish at simulation time 967500
VCS Simulation Report
Time: 967500 Ps
CPU Time: 4.370 seconds; Data structure size: 18.0MB
Sat May 25 23:17:10 2024
CPU time: 18.198 seconds to compile + 3.690 seconds to elab + 1.044 seconds to link + 4.444 seconds in simulation

```

Pattern 3:

```
START!!! Simulation Start .....
```

```
Input A matrix:
 -24   -47    58   -69
 24    -20    -3     -3
 -61      2    20    31
 26   -107   -68   -45
 54    -61   -11   -27
 -72    77   118   -34
 -98   -121   11   124
 -53   109     5  -119
```

```
Output R matrix golden pattern:
```

```
 648   -129   -370   -101
  0    891   195   -565
  0      0   443   -171
  0      0      0   521
  0      0      0     0
  0      0      0     0
  0      0      0     0
  0      0      0     0
```

```
R matrix calculated result:
```

```
 648   -129   -370   -101
  0    891   195   -565
  0      0   443   -171
  0      0      0   521
  0      0      0     0
  0      0      0     0
  0      0      0     0
  0      0      0     0
```

```
Output Q matrix golden pattern:
```

```
 -152   153   -388   162   337   -461   -630   -342
 -237   -72   -47   -474   -241   291   -647   456
 522   129   -115   -286   288   603   -134   -448
 -653   -34   -86   -224   -316   155   109   -664
 531   147   -113   -147   -72   -359   -194   -260
  0    991   -51   -51   -220   -44   41   84
  0      0   -933   48   279   -76   -306   -100
  0      0   -803   343   -458   282   44
```

```
Q matrix calculated result:
```

```
 -152   153   -388   162   337   -461   -630   -342
 -237   -72   -47   -474   -241   291   -647   456
 522   129   -115   -286   288   603   -134   -448
 -653   -34   -86   -224   -316   155   109   -664
 531   147   -113   -147   -72   -359   -194   -260
  0    991   -51   -51   -220   -44   41   84
  0      0   -933   48   279   -76   -306   -100
  0      0   -803   343   -458   282   44
```

```
----- SUMMARY -----
```

```
** Congratulations!! R matrix data have been generated successfully!
** Congratulations!! Q matrix data have been generated successfully!
```

```
** The simulation results are all Pass!!
** Get finish at cycle: 57
```

```
-----
```

```
-- Congratulations !!
-- Simulation PASS!!
```



```
$finish called from file "qr_cordic_tb.v", line 239.
```

```
$finish at simulation time 967500
```

```
          VCS Simulation Report
```

```
Time: 967500 ps
```

```
CPU Time: 4.220 seconds; Data structure size: 18.0MB
```

```
Sat May 25 23:19:36 2024
```

```
CPU time: 17.759 seconds to compile + 3.739 seconds to elab + 1.098 seconds to link + 4.296 seconds in simulation
```

Pattern4:

```
START!!! Simulation Start .....
```

```
Input A matrix:
 -96   -34   -19   -54
 127   -11    -5    29
 -85   123   -98   -61
 -120   -86   -22    83
 15    91    71   123
 97    37   -30    58
 43    -32    21   -40
 -80    -80   -64    21
```

```
Output R matrix golden pattern:
```

```
 1020   185   137   121
  0    792   -368   -40
  0      0   429   -234
  0      0      0   698
  0      0      0     0
  0      0      0     0
  0      0      0     0
```

```
R matrix calculated result:
```

```
 1020   185   137   121
  0    792   -368   -40
  0      0   429   -234
  0      0      0   698
  0      0      0     0
  0      0      0     0
  0      0      0     0
```

```
Output Q matrix golden pattern:
```

```
 -386   509   -344   -487   56   385   167   -325
  -84   -178   717   -346   456   94   -211   -341
 125   -315   -213   64   -315   -339   -40   -800
 298    81   -330   575   628   155   -297   -100
 -892   -225   -225   -41   234   180   -51   -319
  0    -774   304   317   -83   -416   -101   -193
  0      0   352   505   -265   597   437   -230
  0      0      0     24   456   -428   812   -36
```

```
Q matrix calculated result:
```

```
 -386   509   -344   -487   56   385   167   -325
  -84   -178   717   -346   456   94   -211   -341
 125   -315   -213   64   -315   -339   -40   -800
 298    81   -330   575   628   155   -297   -100
 -892   -225   -225   -41   234   180   -51   -319
  0    -774   304   317   -83   -416   -101   -193
  0      0   352   505   -265   597   437   -230
  0      0      0     24   456   -428   812   -36
```

```
----- SUMMARY -----
```

```
** Congratulations!! R matrix data have been generated successfully!
** Congratulations!! Q matrix data have been generated successfully!
```

```
** The simulation results are all Pass!!
** Get finish at cycle: 57
```

```
-----
```

```
-- Congratulations !!
-- Simulation PASS!!
```



```
$finish called from file "qr_cordic_tb.v", line 239.
```

```
$finish at simulation time 967500
```

```
          VCS Simulation Report
```

```
Time: 967500 ps
```

```
CPU Time: 4.180 seconds; Data structure size: 18.0MB
```

```
Sat May 25 23:19:28 2024
```

```
CPU time: 17.053 seconds to compile + 3.551 seconds to elab + 1.078 seconds to link + 4.249 seconds in simulation
```

- Both R and Q are all pass in gate-level simulation

- Clock rate : 15ns

- Cycles : 57

- Run time : 967.5ns

Gate-level simulation Commands (VCS):

```
vcs -full64 -R -sverilog qr_cordic_tb.v qr_cordic_syn.v +define+SDF+access+r+neg_tchk
```

```
+vcs+fsdbon+fsdb+mda+fsdbfile+qr_cordic.fsdb -v
```

```
/home/cell_library/CBDK_IC_Contest_v2.5/Verilog/tsmc13_neg.v +maxdelays
```

- The complete code and related data are placed on [GitHub](#)