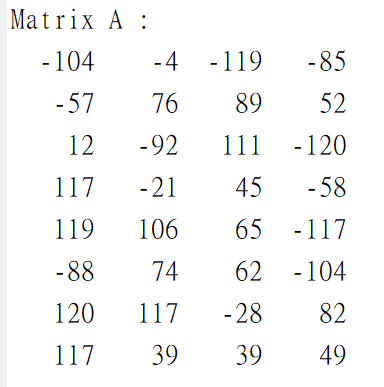
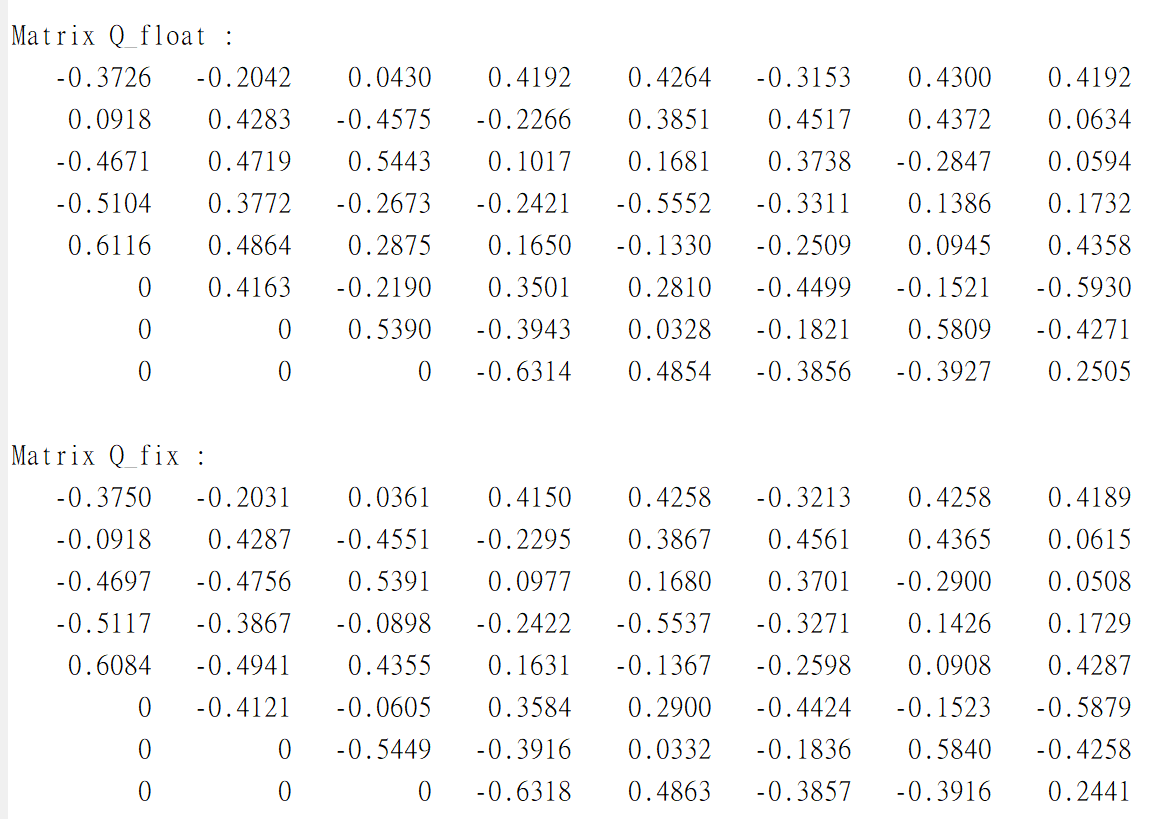
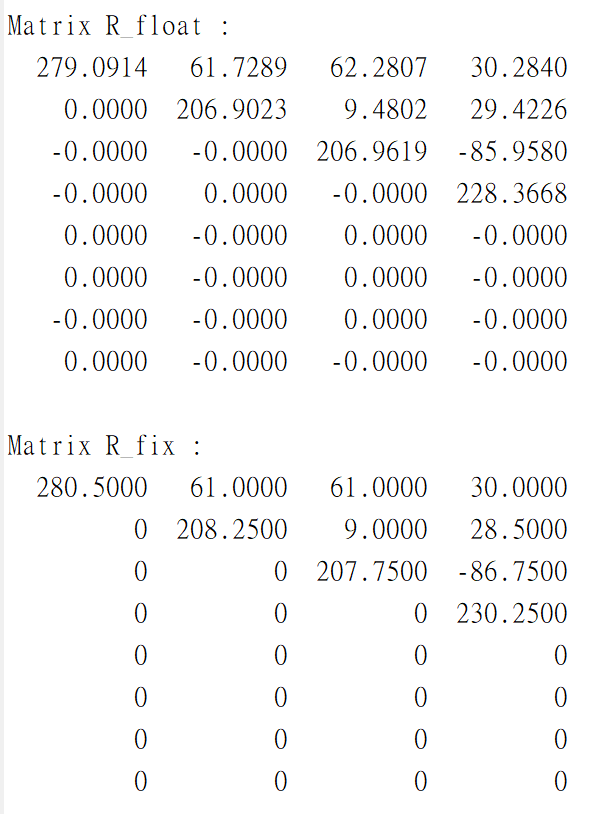
**VLSI DPS HW#4**

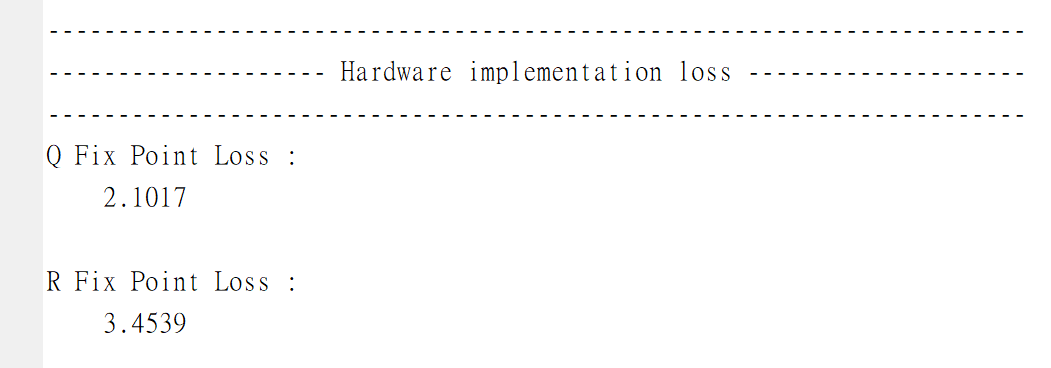
**電機四 4109064119 陳柏淳**

**1. Calculate the Frobenius Distance**

* Generate a random matrix A

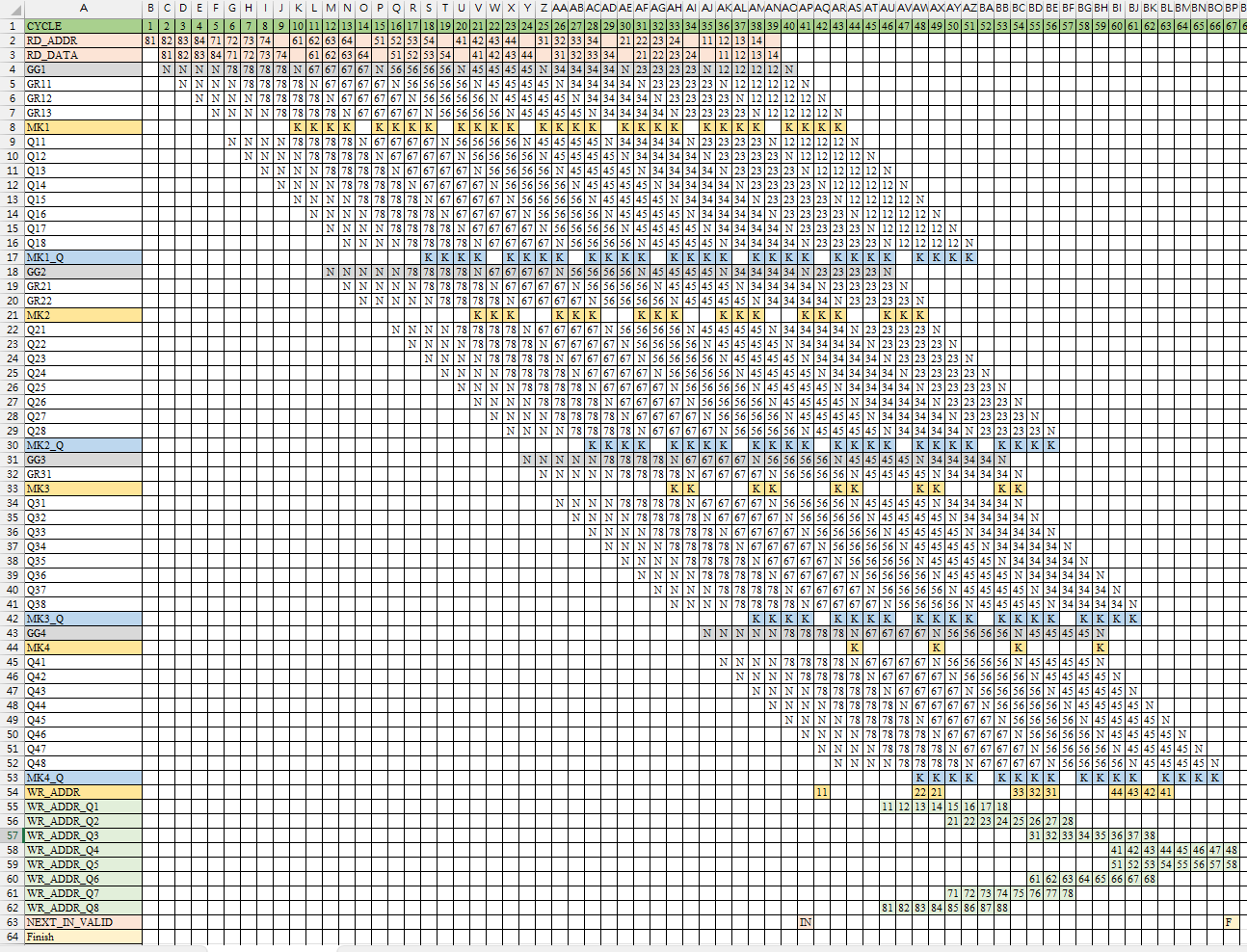


* Q\_float and R\_float are generated from Q\*[A|I] =[R|Q]
* Q\_fix and R\_fix are generated from Cordic given’s rotation
* The Frobenius Distance (Euclidean Norm) is sqrt(trace((float-fix)\*(float-fix)'))



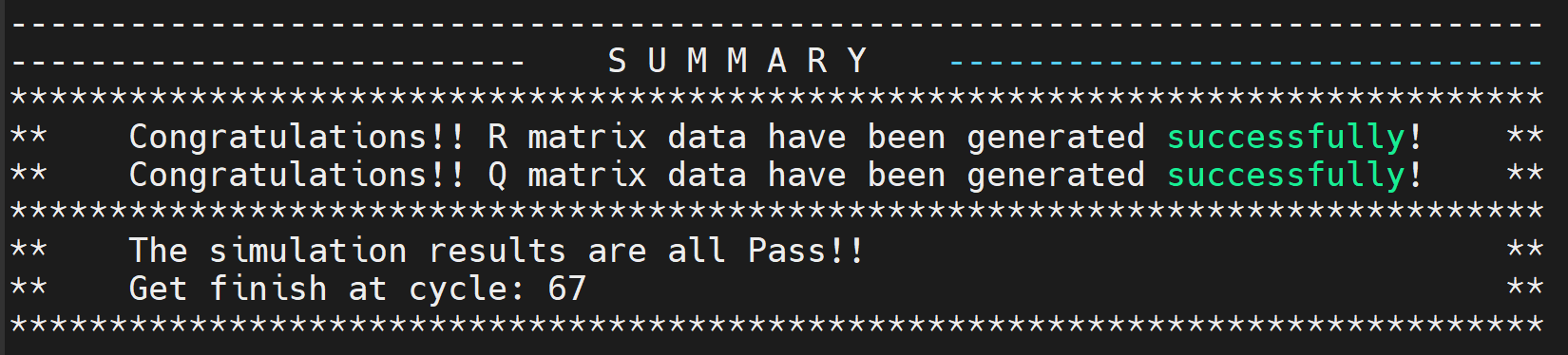
* Fix-point precision

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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**

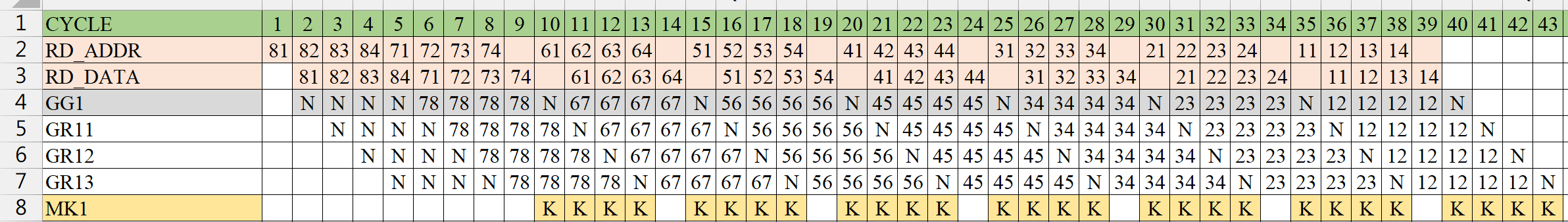
* N : No Operation(idle)
* K : Multiplied by K
*  : Perform rotation on 7th and 8th row
* : The matrix index of data write back to tb

**3. Clock cycles needed to complete one QR factorization**

****

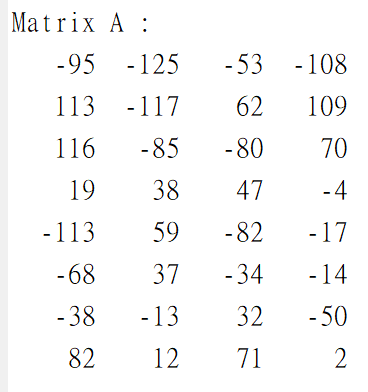
67 cycles

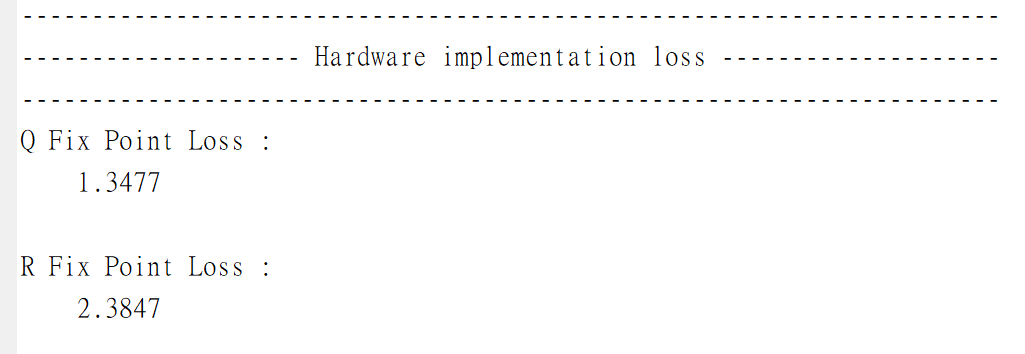
**4. The initiation interval of two successive QR factorizations**

****

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

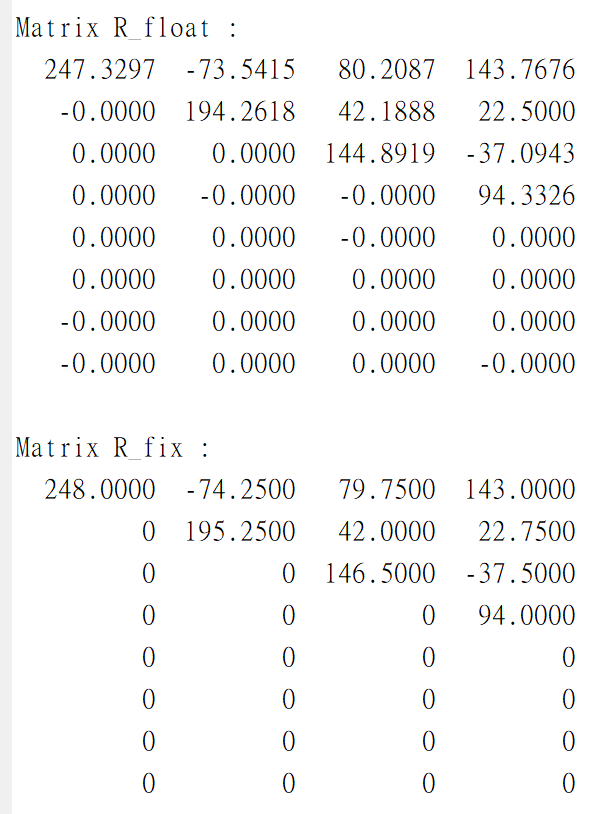
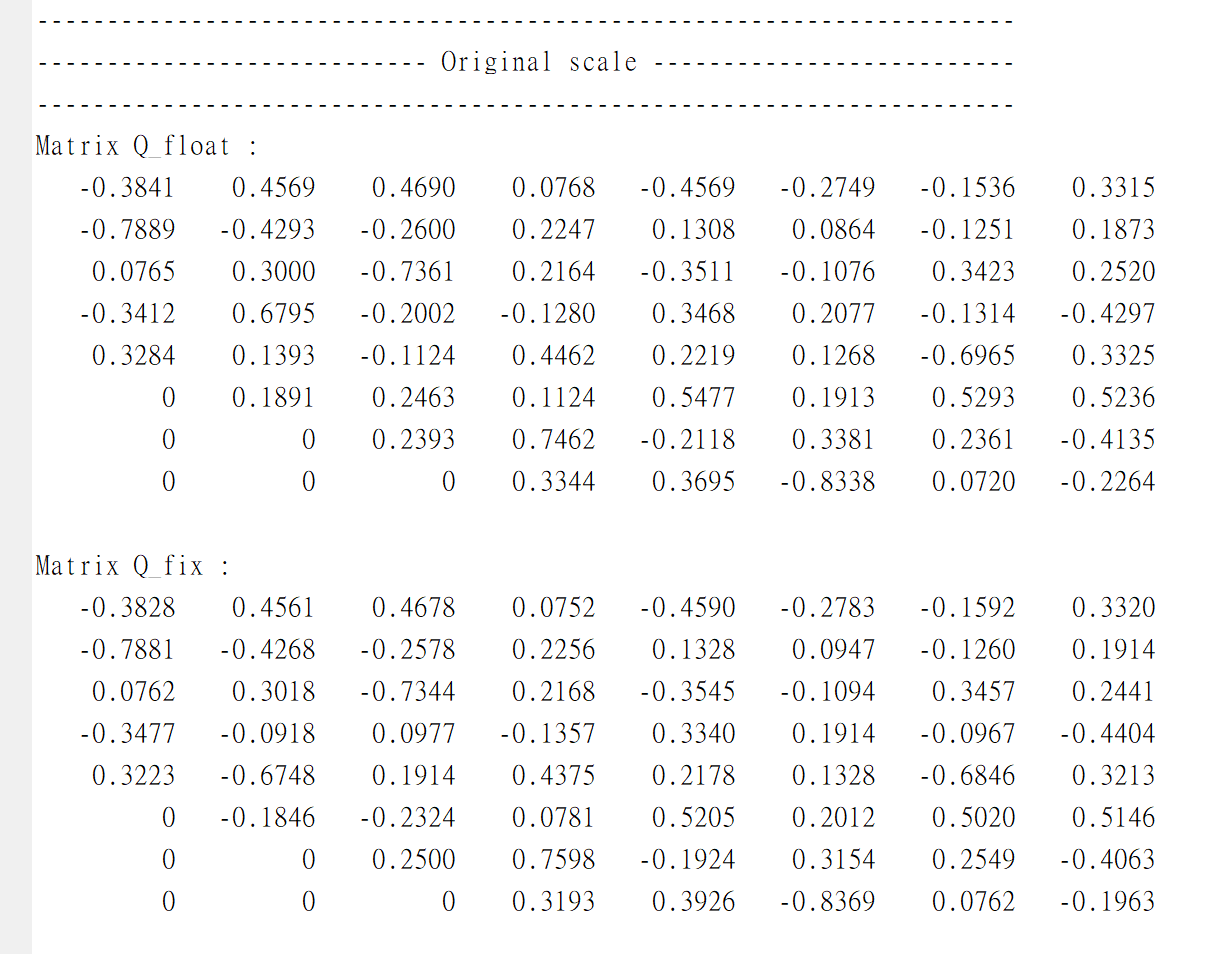
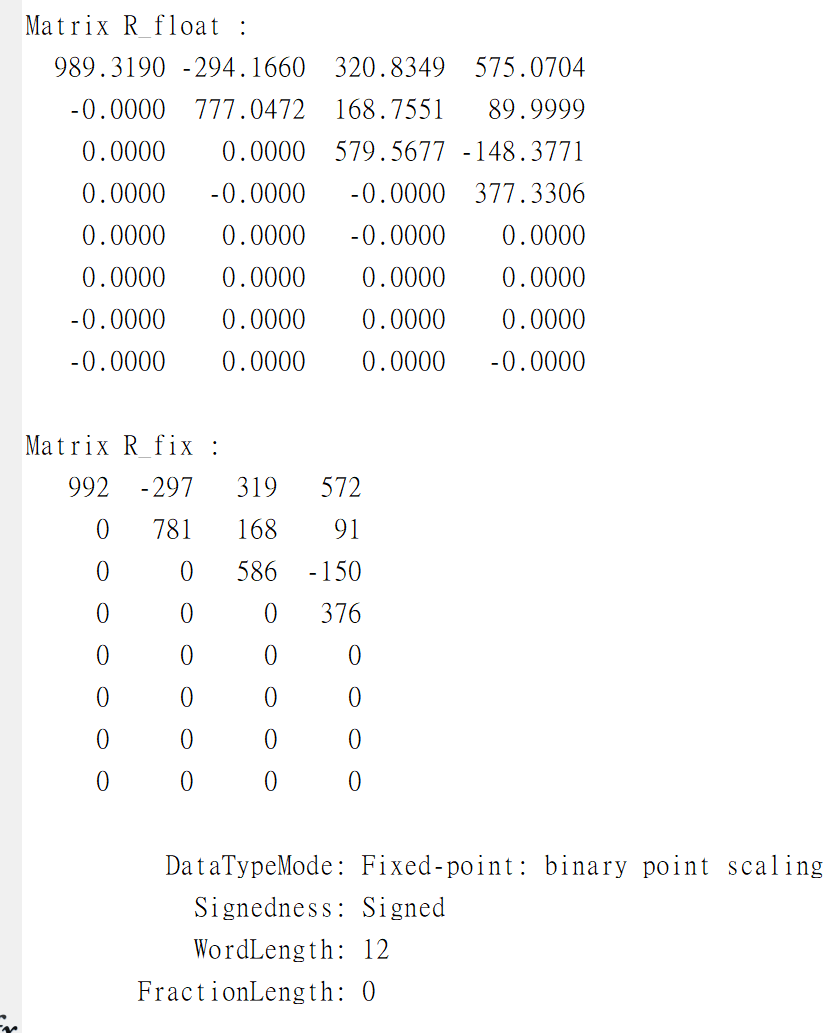
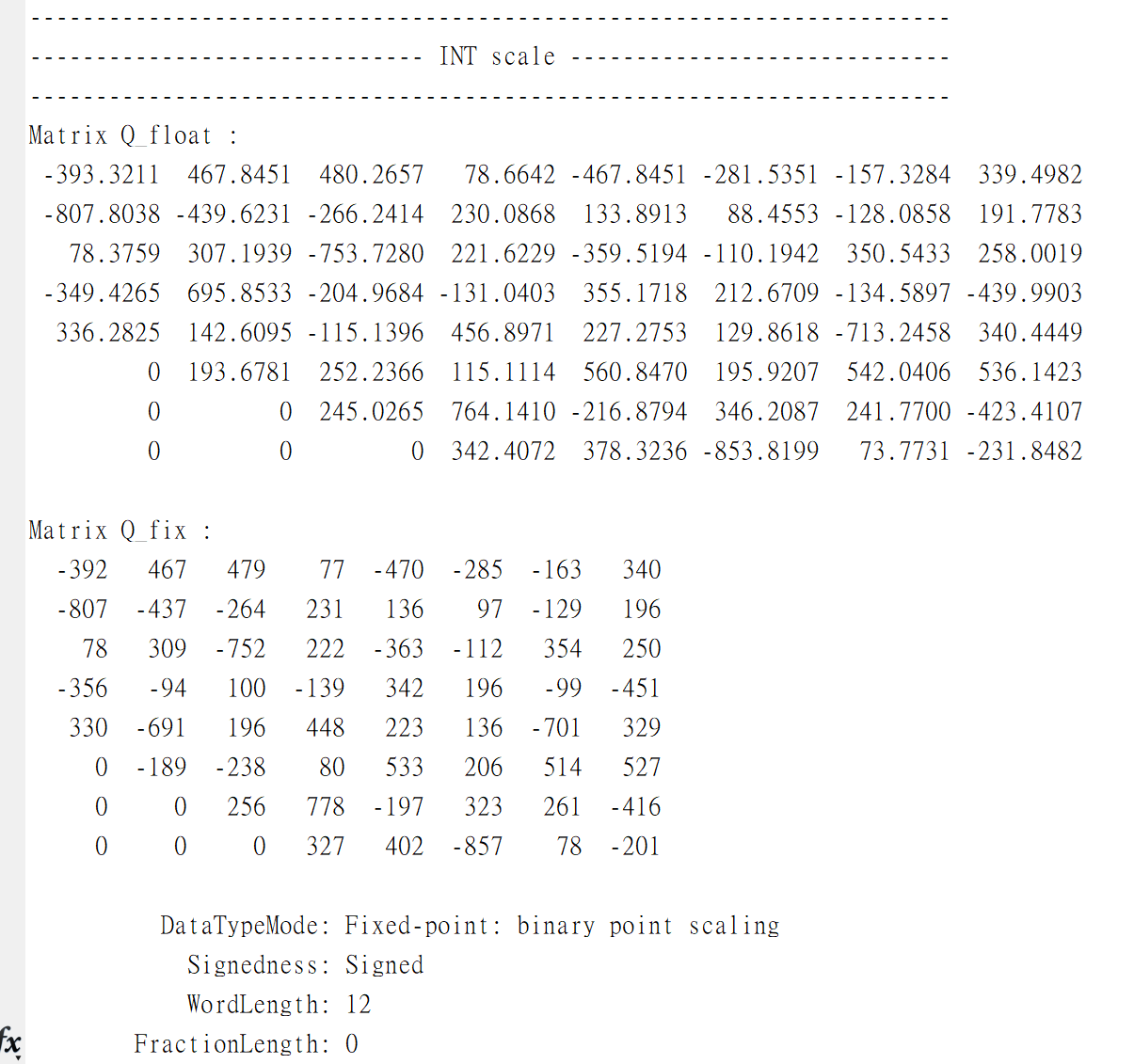
The initiation interval is 41 cycles.

* **Matlab result**
* Random matrix A
* Hardware loss



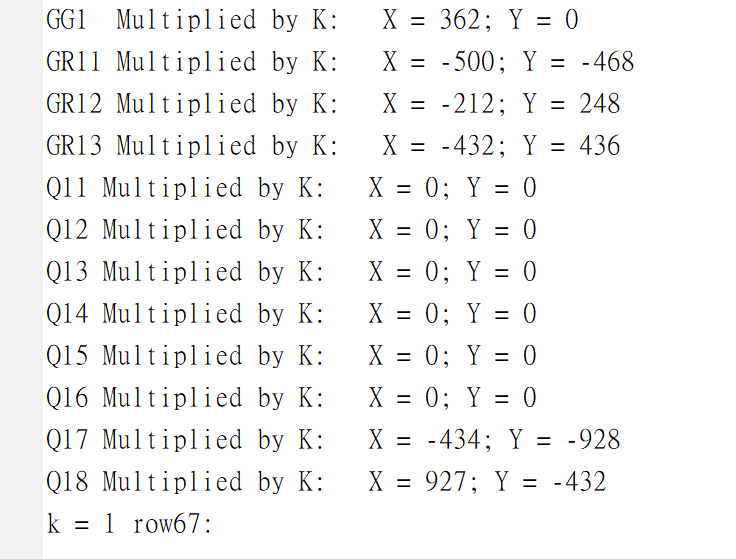
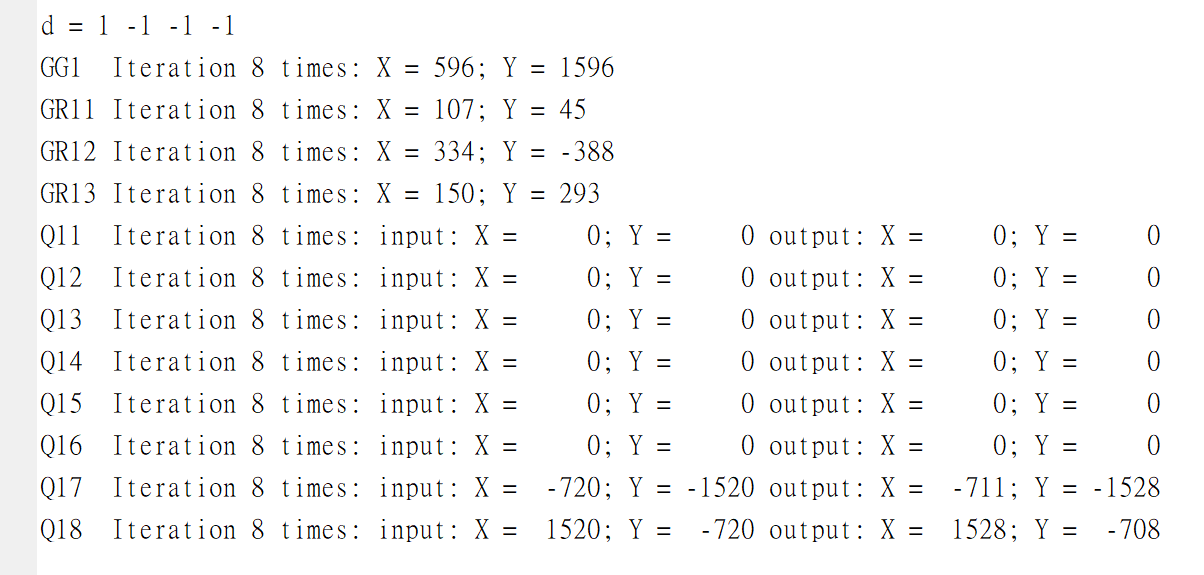
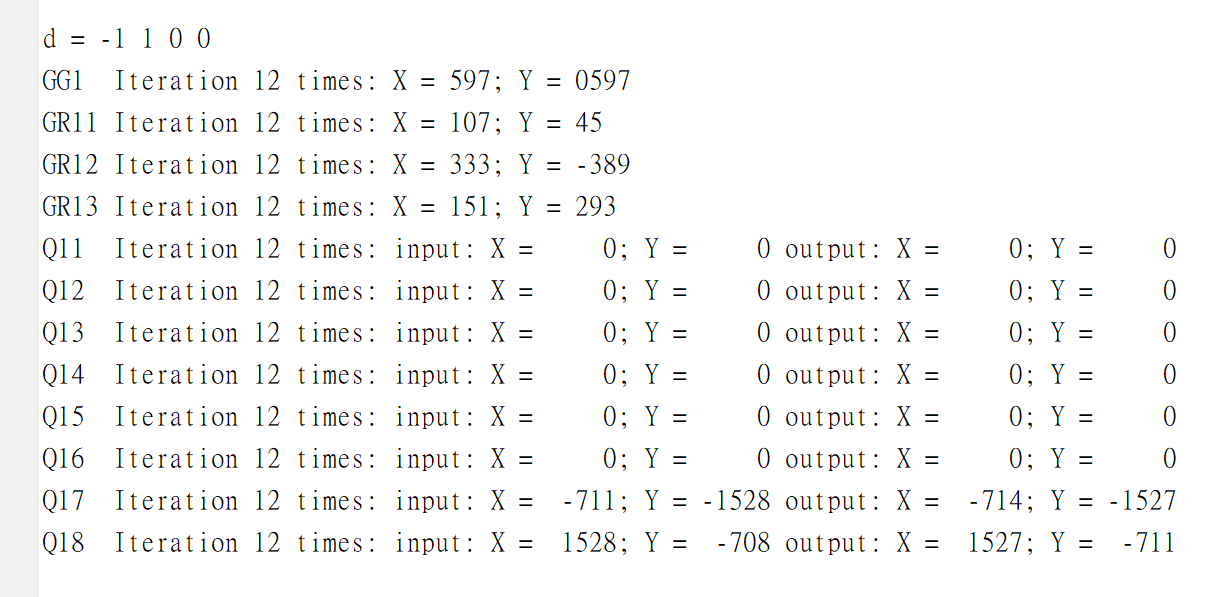
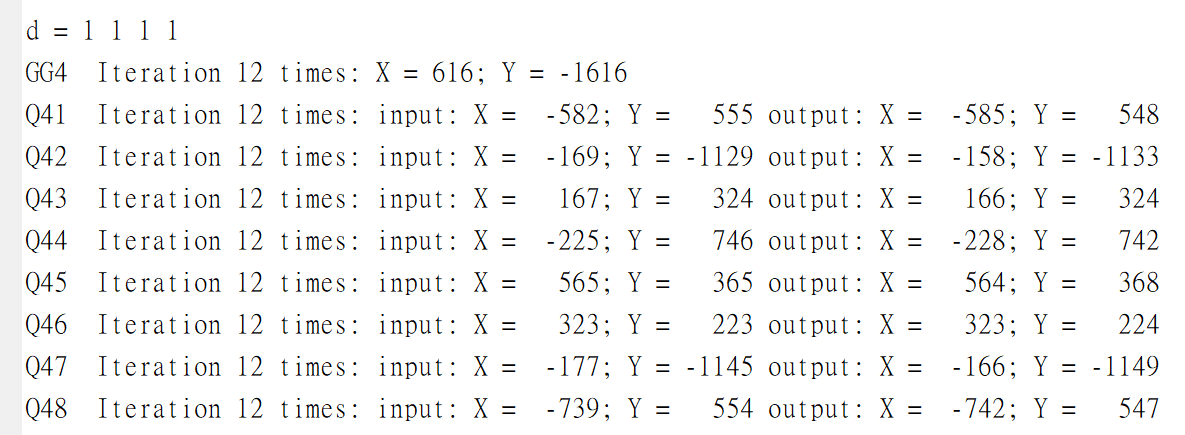
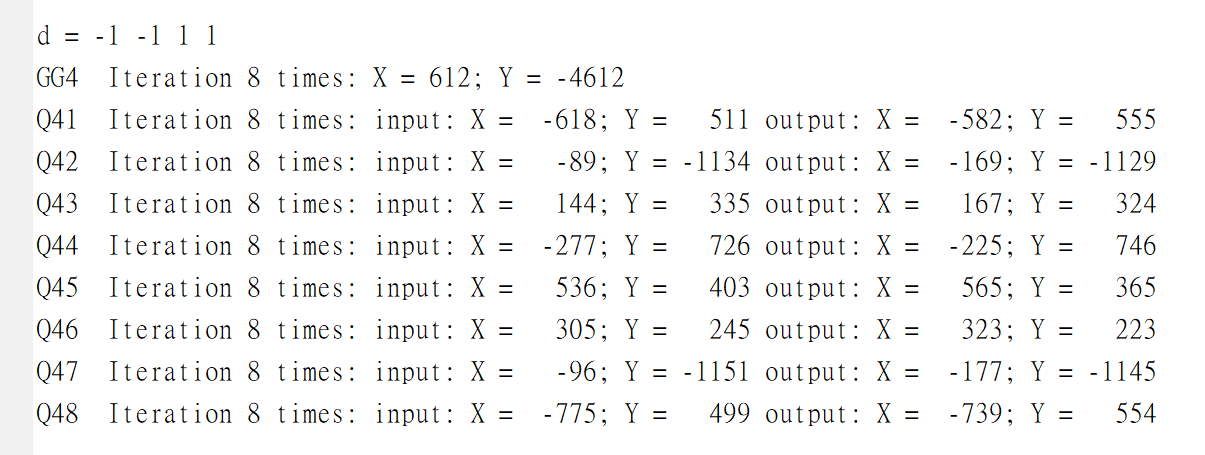
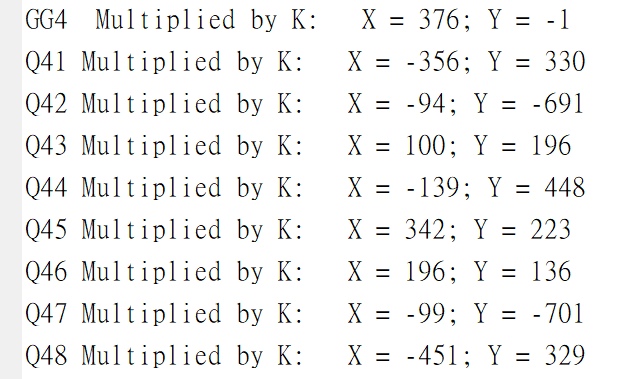
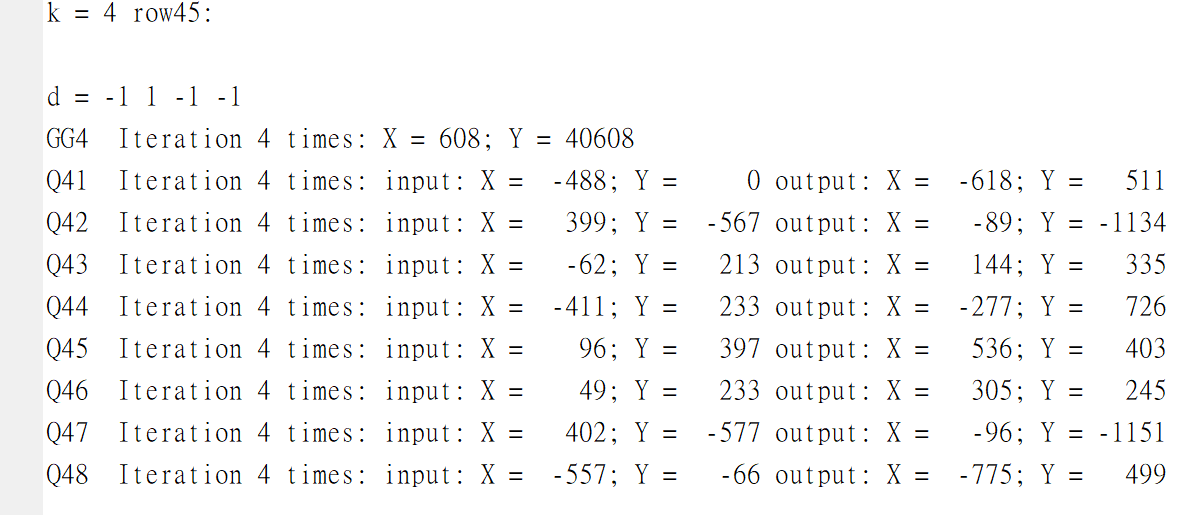
Shift to integer to simulate hardware performance

Actual values



* **Matlab result**

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



* **Hardware implementation result**

1. Hardware architecture

A matrix input

Identity matrix input(I8)

R matrix output

Q matrix output

GG

GR

GR

GR

Q

Q

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).

D

MK

d

Xi Yi iter

xo yo

d

D

MK

d

Xi Yi iter

xo yo

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.

2. Verilog code (only core part):

|  |  |
| --- | --- |
| 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 | **module** GG\_one\_iter **#(**  **parameter** R\_LEN **=** 12**,**  **parameter** R\_FRAC **=** 2  **)(**  **input** **signed** **[**R\_LEN**-**1**:**0**]** xi**,**  **input** **signed** **[**R\_LEN**-**1**:**0**]** yi**,**  **input** **[**3**:**0**]** iter**,**  **output** **[**1**:**0**]** d**,**  **output** **reg** **signed** **[**R\_LEN**-**1**:**0**]** xo**,**  **output** **reg** **signed** **[**R\_LEN**-**1**:**0**]** yo  **);**  **assign** d **=** **(**yi **==** 0**)** **?** 2 **:** xi**[**R\_LEN**-**1**]** **^** yi**[**R\_LEN**-**1**];**  **always** **@(\*)** **begin**  **if(**d **==** 2**)** **begin**  xo **=** xi**;**  yo **=** yi**;**  **end**  **else** **if(**d **==** 1**)** **begin**  xo **=** xi **-** **(**yi **>>>** iter**);**  yo **=** yi **+** **(**xi **>>>** iter**);**  **end**  **else** **begin**  xo **=** xi **+** **(**yi **>>>** iter**);**  yo **=** yi **-** **(**xi **>>>** iter**);**  **end**  **end**  **endmodule** |

a. GG module

b. GR module

|  |  |
| --- | --- |
| 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 | **module** GR\_one\_iter **#(**  **parameter** R\_LEN **=** 12**,**  **parameter** R\_FRAC **=** 2  **)**  **(** **input** **signed** **[**R\_LEN**-**1**:**0**]** xi**,**  **input** **signed** **[**R\_LEN**-**1**:**0**]** yi**,**  **input** **[**3**:**0**]** iter**,**  **input** **[**1**:**0**]** d**,**  **output** **reg** **signed** **[**R\_LEN**-**1**:**0**]** xo**,**  **output** **reg** **signed** **[**R\_LEN**-**1**:**0**]** yo  **);**  **always** **@(\*)** **begin**  **if(**d **==** 2**)** **begin**  xo **=** xi**;**  yo **=** yi**;**  **end**  **else** **if(**d **==** 1**)** **begin**  xo **=** xi **-** **(**yi **>>>** iter**);**  yo **=** yi **+** **(**xi **>>>** iter**);**  **end**  **else** **begin**  xo **=** xi **+** **(**yi **>>>** iter**);**  yo **=** yi **-** **(**xi **>>>** iter**);**  **end**  **end**  **endmodule** |

c. Q module is the same as GR

d. MK (multiplied by K)

|  |  |
| --- | --- |
| 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 | **module** MK **#(**  **parameter** R\_LEN **=** 12**,**  **parameter** R\_FRAC **=** 2**,**  **parameter** K\_LEN **=** 10**,**  **parameter** K\_FRAC **=** 9  **)(**  **input** **signed** **[**R\_LEN**-**1**:**0**]** xi**,**  **input** **signed** **[**R\_LEN**-**1**:**0**]** yi**,**  **output** **signed** **[**R\_LEN**-**1**:**0**]** xo**,**  **output** **signed** **[**R\_LEN**-**1**:**0**]** yo  **);**  **localparam** **signed** K **=** 10'b0\_100110111**;** // K = 0.607421875  **wire** **signed** **[**R\_LEN**+**K\_LEN**-**1**:**0**]** xo\_0**;**  **wire** **signed** **[**R\_LEN**+**K\_LEN**-**1**:**0**]** yo\_0**;**  **assign** xo\_0 **=** xi **\*** K**;**  **assign** yo\_0 **=** yi **\*** K**;**  //truncate to R\_LEN bits  **assign** xo **=** **{**xo\_0**[**R\_LEN**+**K\_LEN**-**1**],** xo\_0**[**R\_LEN**+**K\_FRAC**-**2**:**K\_FRAC**]};**  **assign** yo **=** **{**yo\_0**[**R\_LEN**+**K\_LEN**-**1**],** yo\_0**[**R\_LEN**+**K\_FRAC**-**2**:**K\_FRAC**]};**  **endmodule** |

e. GG iteration control

|  |  |
| --- | --- |
| 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  2728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273 | **always** **@(posedge** clk **or** **posedge** rst**)** **begin**  **if** **(**rst**)** **begin**  iter\_gg1 **<=** 'd0**;**  **end**  **else** **if(**ROT\_wire**)** **begin**  **if(**nop\_gg1**)** **begin**  iter\_gg1 **<=** 'd0**;**  **end**  **else** **if(**iter\_last\_gg1**)** **begin**  iter\_gg1 **<=** iter\_gg1 **+** 'd1**;**  **end**  **else** **begin**  iter\_gg1 **<=** iter\_gg1 **+** ITER\_ONE\_CYCLE**;**  **end**  **end**  **else** **begin**  iter\_gg1 **<=** 'd0**;**  **end**  **end**  // GG1 input data xi, yi  **always** **@(posedge** clk **or** **posedge** rst**)** **begin**  **if** **(**rst**)** **begin**  xi\_gg1 **<=** 'd0**;**  yi\_gg1 **<=** 'd0**;**  **end**  **else** **if(**OP\_wire**)** **begin**  **case(**iter\_gg1**)**  0**:** **begin**  **if(**start\_gg1**)** **begin**  xi\_gg1 **<=** 'd0**;**  yi\_gg1 **<=** rd\_A\_data\_ext**;**  **end**  **else** **if(**nop\_gg1 **&&** **!**finish\_gg1**)** **begin**  xi\_gg1 **<=** rd\_A\_data\_ext**;**  yi\_gg1 **<=** yo\_gg1**;**  **end**  **else** **begin**  xi\_gg1 **<=** xo\_gg1**;**  yi\_gg1 **<=** yo\_gg1**;**  **end**  **end**  ITER\_K**:** **begin**  **if(**finish\_gg1**)** **begin**  xi\_gg1 **<=** xo\_gg1**;**  yi\_gg1 **<=** yo\_gg1**;**  **end**  **else** **begin**  xi\_gg1 **<=** rd\_A\_data\_ext**;**  yi\_gg1 **<=** xo\_mk1**;**  **end**  **end**  **default:** **begin**  xi\_gg1 **<=** xo\_gg1**;**  yi\_gg1 **<=** yo\_gg1**;**  **end**  **endcase**  **end**  **else** **begin**  xi\_gg1 **<=** 'd0**;**  yi\_gg1 **<=** 'd0**;**  **end**  **end**  // GG1 mk\_count  **always** **@(posedge** clk **or** **posedge** rst**)** **begin**  **if** **(**rst**)** **begin**  mk\_count\_gg1 **<=** 'd0**;**  **end**  **else** **if(**multk\_gg1**)** **begin**  mk\_count\_gg1 **<=** mk\_count\_gg1 **+** 'd1**;**  **end**  **end** |

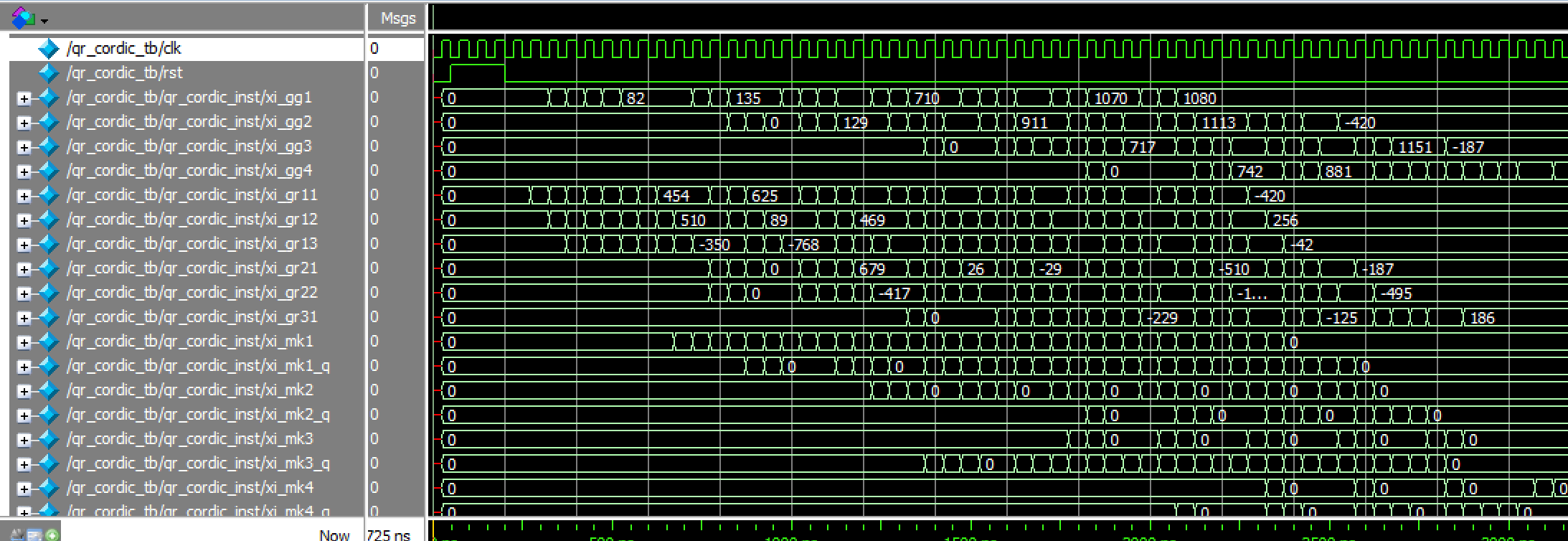
f. GR iteration control

|  |  |
| --- | --- |
| 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  2728293031323334353637383940414243444546474849505152535455565758596061626364656667 | // GR11 current iteration number  **always** **@(posedge** clk **or** **posedge** rst**)** **begin**  **if** **(**rst**)** **begin**  iter\_gr11 **<=** 'd0**;**  nop\_gr11 **<=** 'd0**;**  d1\_gr11 **<=** 'd0**;**  d2\_gr11 **<=** 'd0**;**  d3\_gr11 **<=** 'd0**;**  d4\_gr11 **<=** 'd0**;**  neg\_gr11 **<=** 'd0**;**  mk\_count\_gr11 **<=** 'd0**;**  **end**  **else** **begin**  iter\_gr11 **<=** iter\_gg1**;**  nop\_gr11 **<=** nop\_gg1**;**  d1\_gr11 **<=** d1\_gg1**;**  d2\_gr11 **<=** d2\_gg1**;**  d3\_gr11 **<=** d3\_gg1**;**  d4\_gr11 **<=** d4\_gg1**;**  neg\_gr11 **<=** neg\_gg1**;**  mk\_count\_gr11 **<=** mk\_count\_gg1**;**  **end**  **end**  // GR11 input data xi, yi  **always** **@(posedge** clk **or** **posedge** rst**)** **begin**  **if** **(**rst**)** **begin**  xi\_gr11 **<=** 'd0**;**  yi\_gr11 **<=** 'd0**;**  **end**  **else** **if(**OP\_wire**)** **begin**  **case(**iter\_gr11**)**  0**:** **begin**  **if(**start\_gr11\_reg**)** **begin**  xi\_gr11 **<=** 'd0**;**  yi\_gr11 **<=** rd\_A\_data\_ext**;**  **end**  **else** **if(**nop\_gr11 **&&** **!**finish\_gr11**)** **begin**  xi\_gr11 **<=** rd\_A\_data\_ext**;**  yi\_gr11 **<=** yo\_gr11**;**  **end**  **else** **begin**  xi\_gr11 **<=** xo\_gr11**;**  yi\_gr11 **<=** yo\_gr11**;**  **end**  **end**  ITER\_K**:** **begin**  **if(**finish\_gr11**)** **begin**  xi\_gr11 **<=** xo\_mk1**;**  yi\_gr11 **<=** yo\_mk1**;**  **end**  **else** **begin**  xi\_gr11 **<=** rd\_A\_data\_ext**;**  yi\_gr11 **<=** xo\_mk1**;**  **end**  **end**  **default:** **begin**  xi\_gr11 **<=** xo\_gr11**;**  yi\_gr11 **<=** yo\_gr11**;**  **end**  **endcase**  **end**  **else** **begin**  xi\_gr11 **<=** 'd0**;**  xi\_gr11 **<=** 'd0**;**  **end**  **end** |

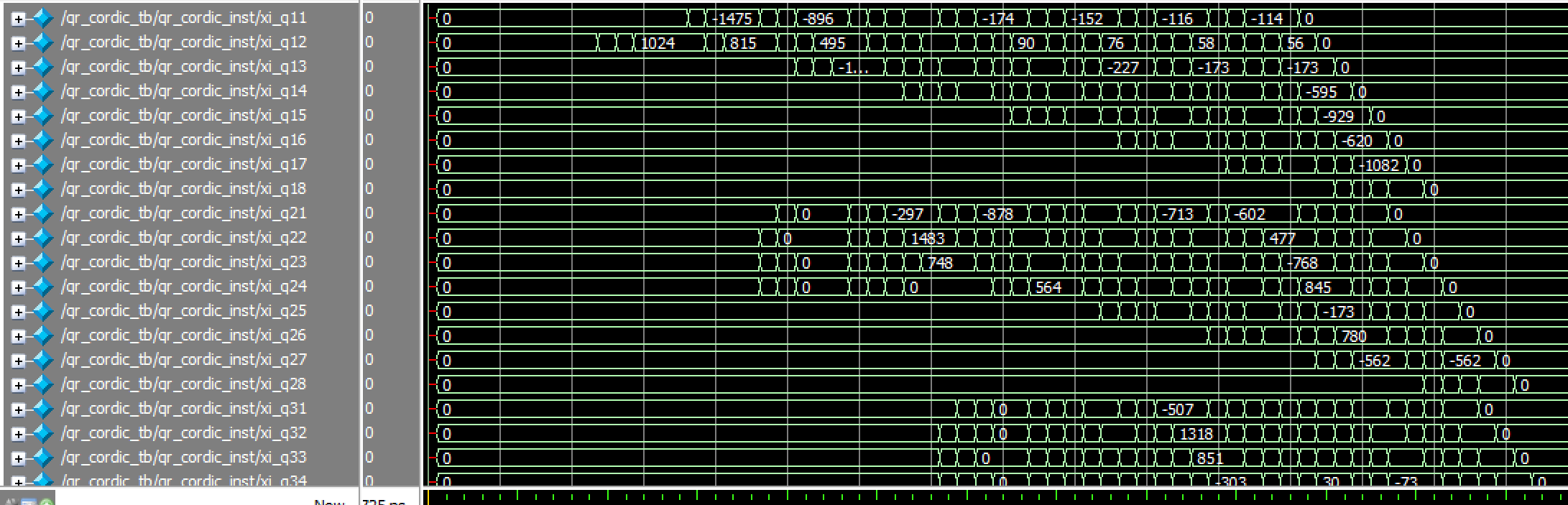
3. Simulation result

a. simulation waveform

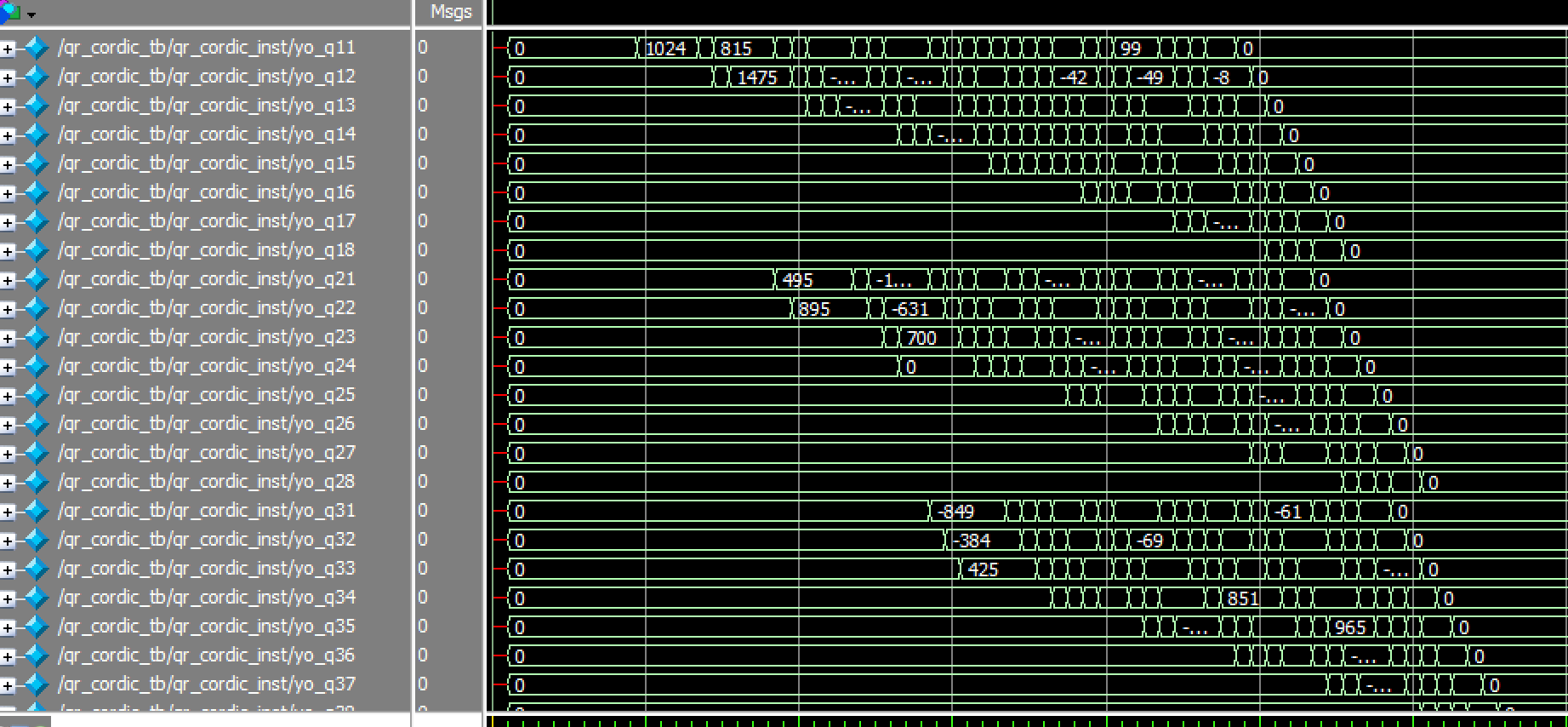
* Data propagate form left to right PE(GG, GR, Q)



* Process matrix R



* Process matrix Q



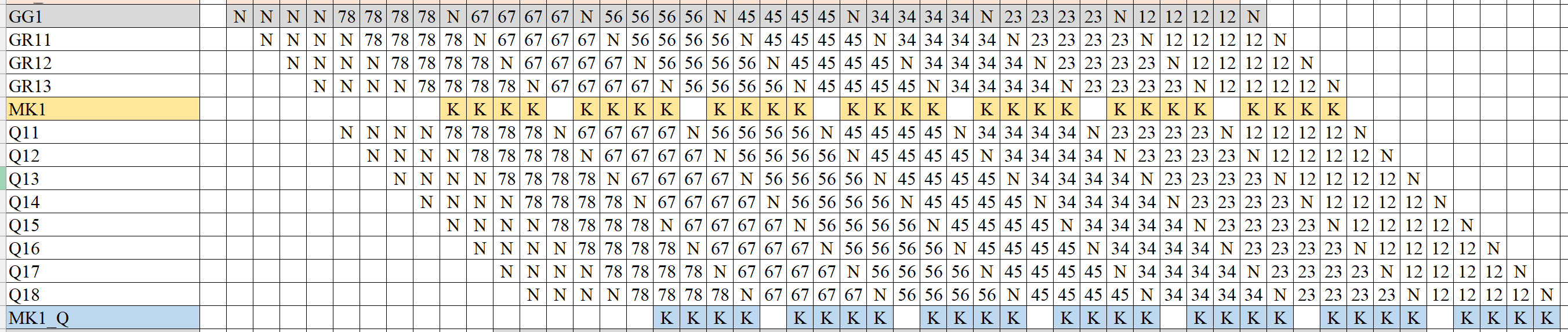
* Data flow & data scheduling

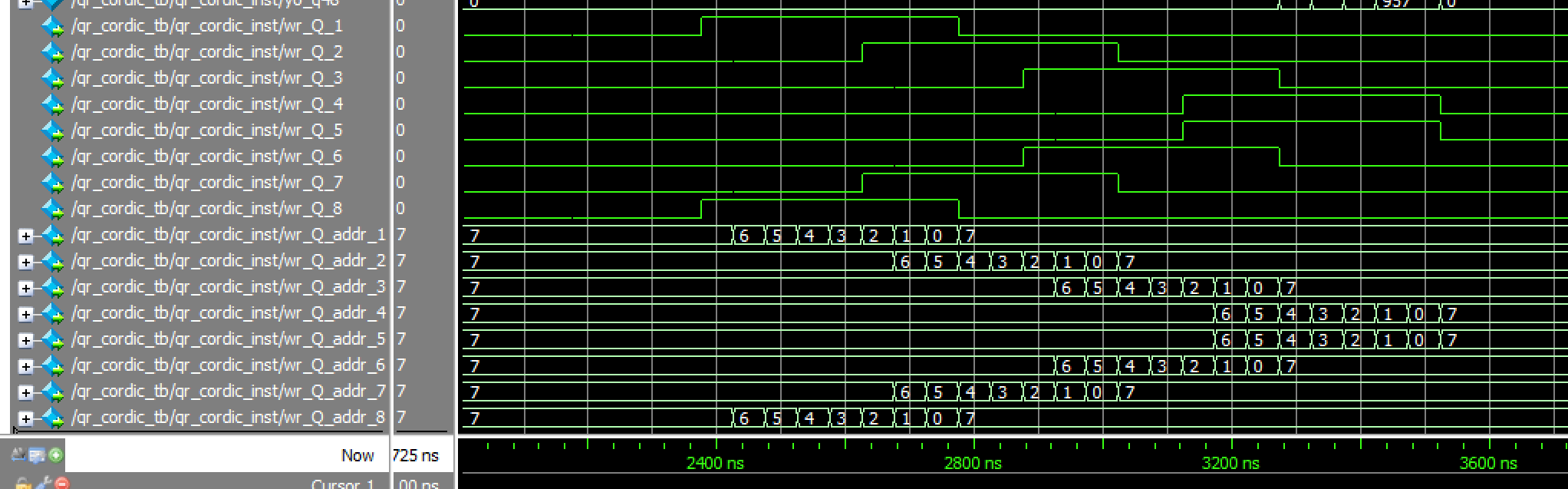
GG1→GR11→GR12→GR13→Q11→Q12→Q13→…..→Q18

-------- GG2→GR21→GR22→Q21→Q22→Q23→…..→Q28

---------------- GG3→GR31→Q31→Q32→Q33→…..→Q38

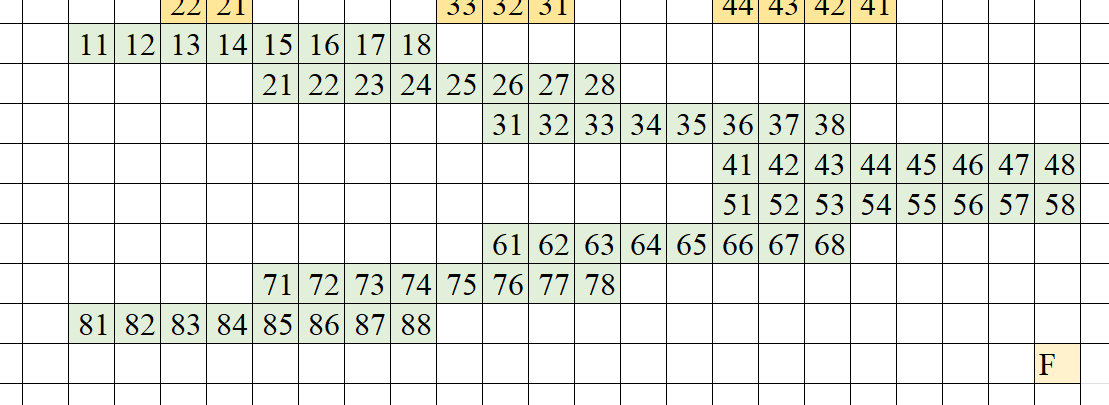
-------------------------- GG4→Q41→Q42→Q43→…..→Q48

* timing diagram
* Write Q matrix data back

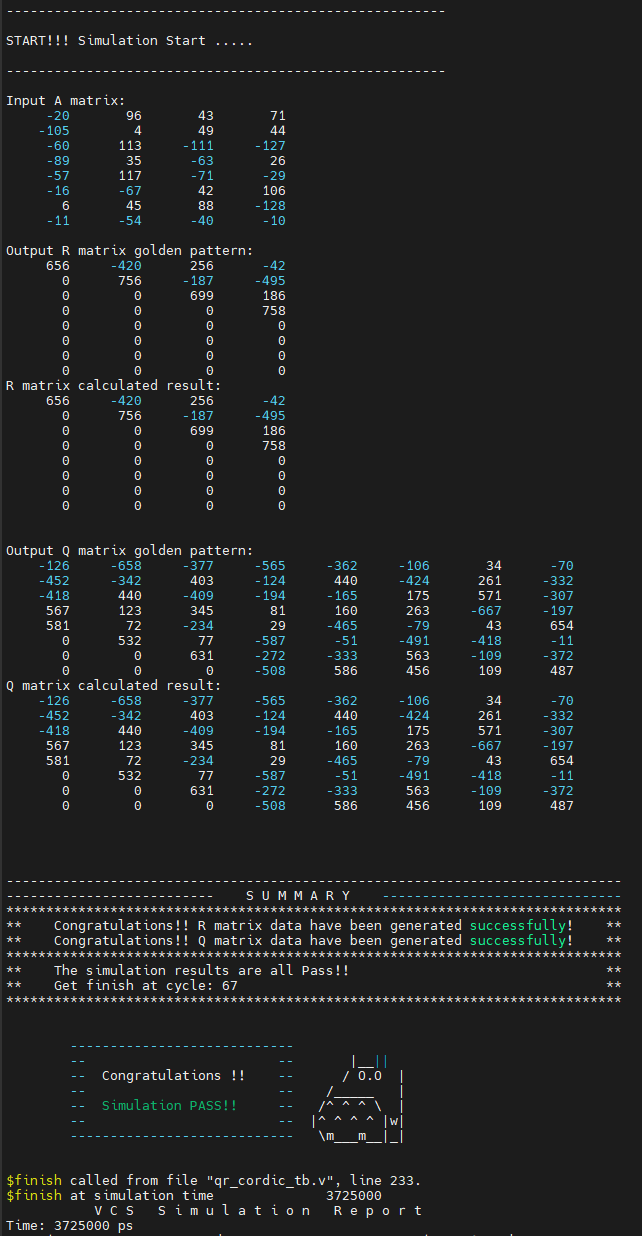
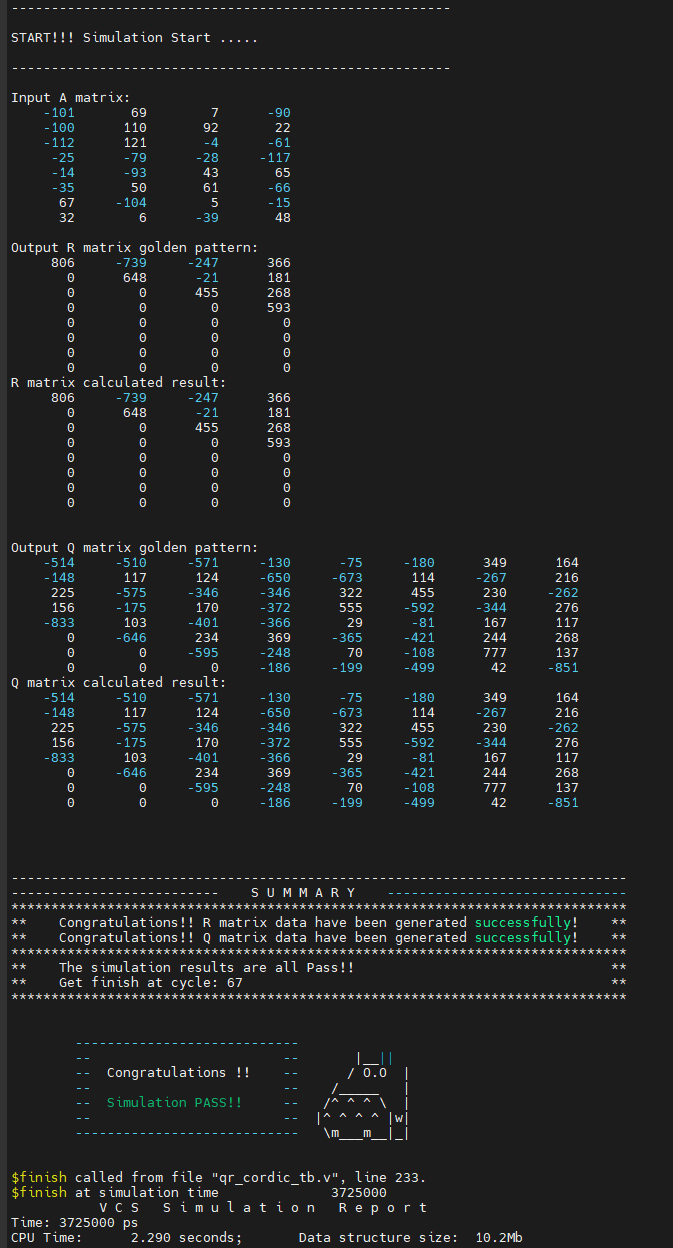


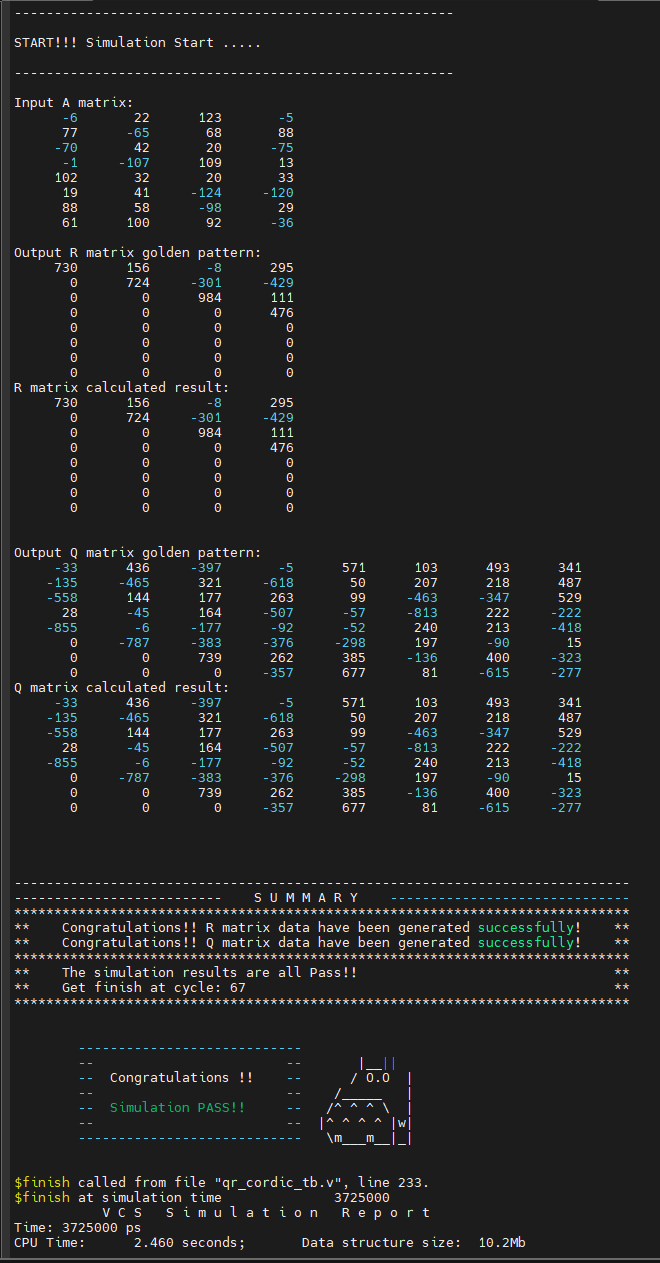
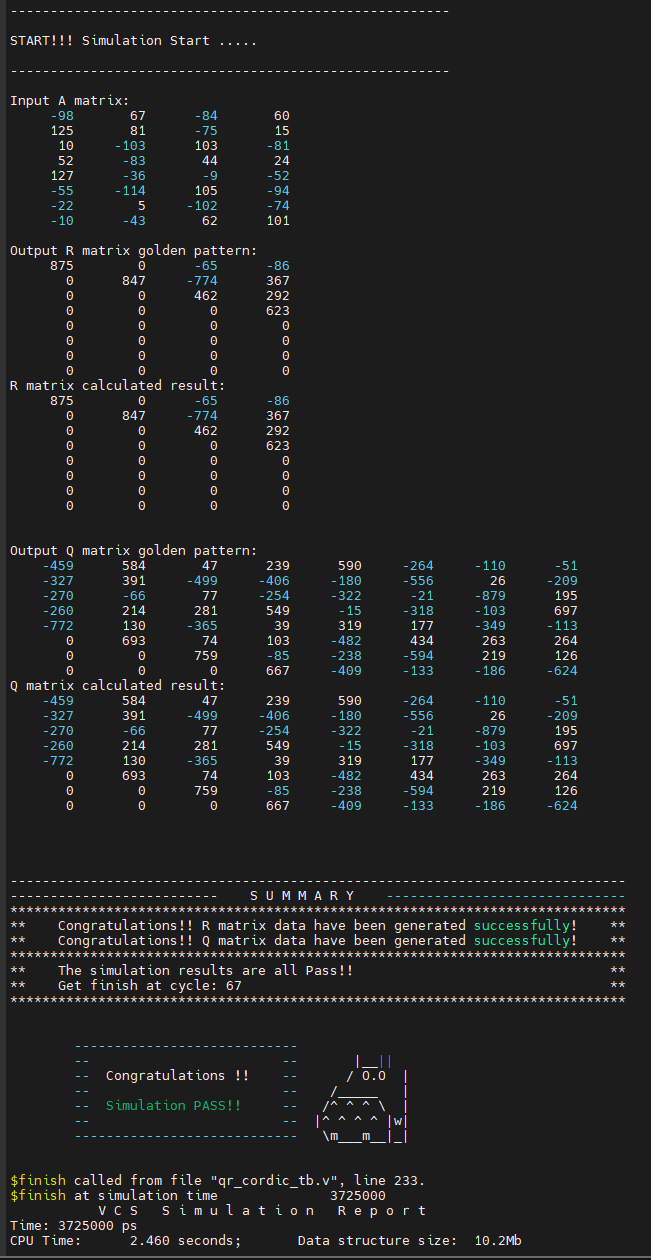
Each row has an independent write enable signal, since multiple data outputs from Q occur simultaneously in the systolic array.

* Timing diagram



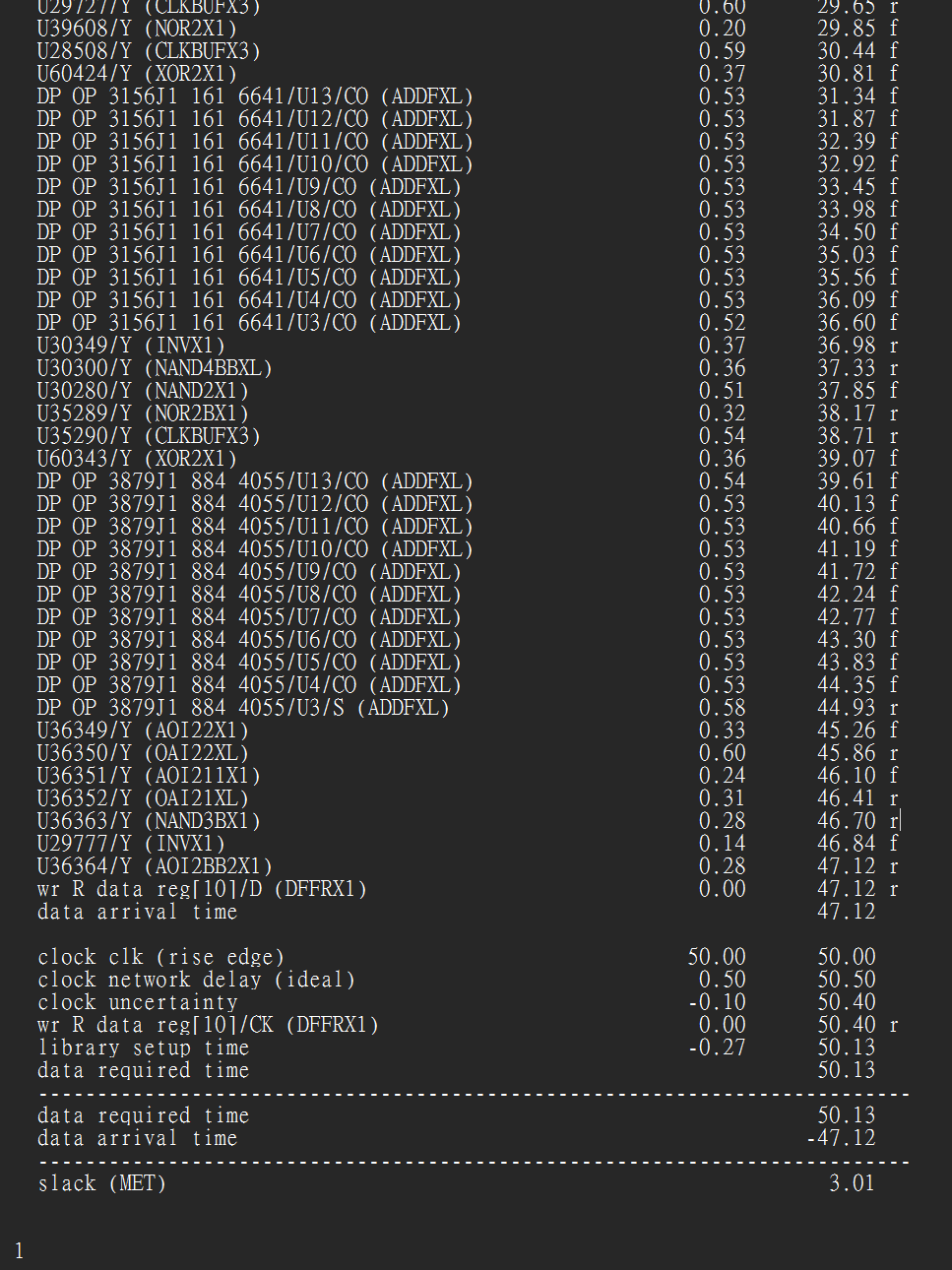
b. RTL simulation

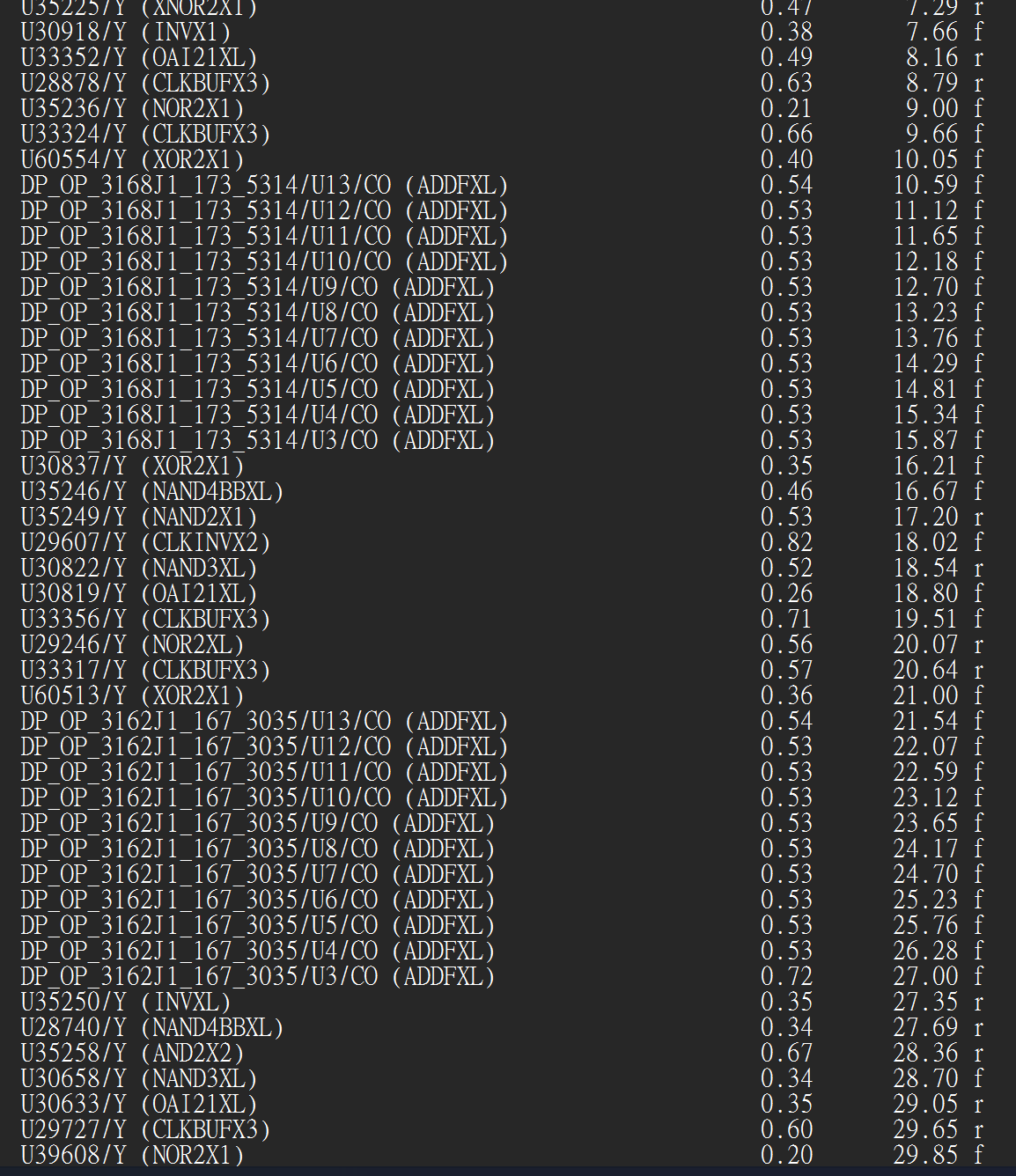
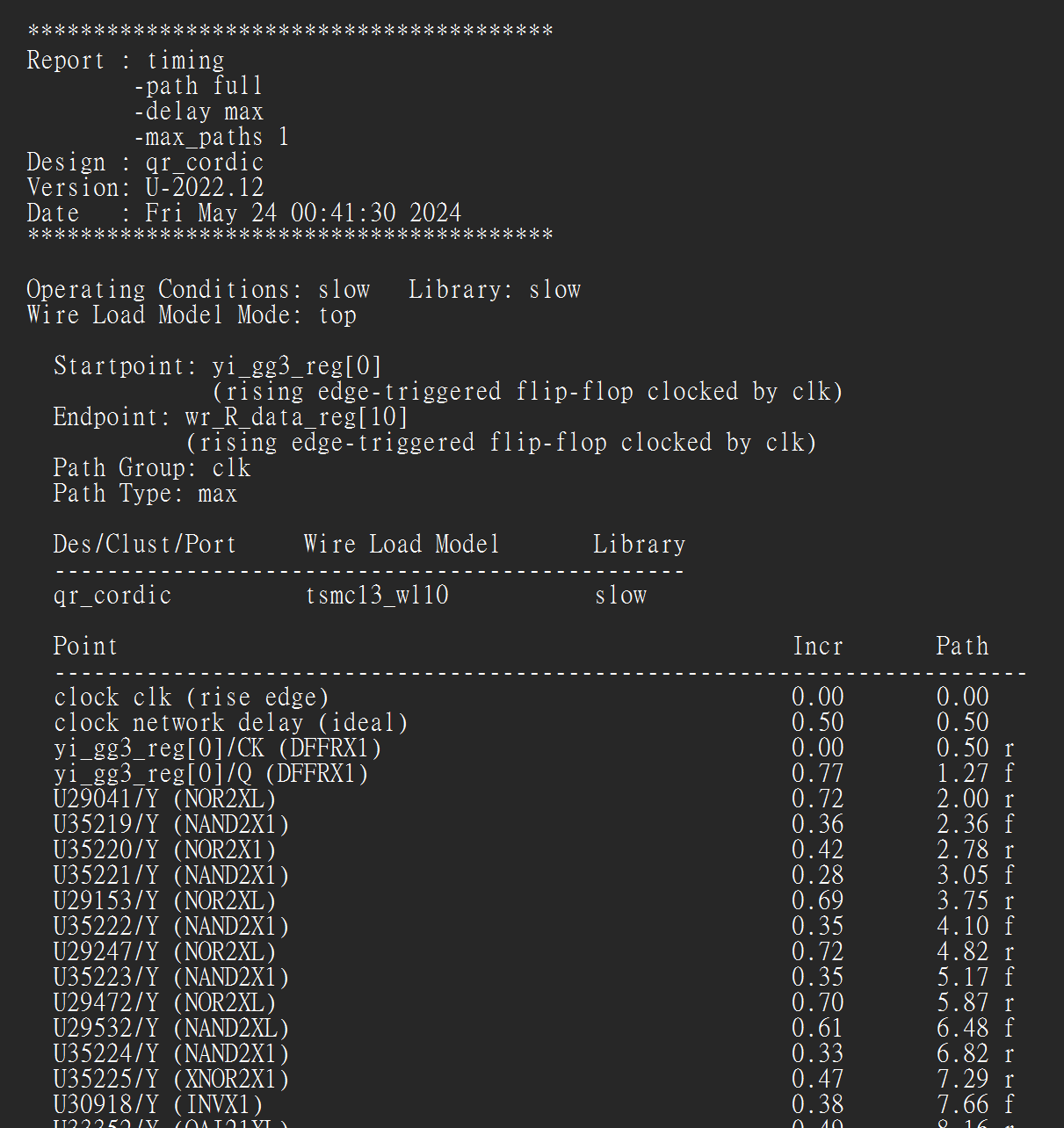
Pattern 1: Pattern 2:

Pattern 3: Pattern4:

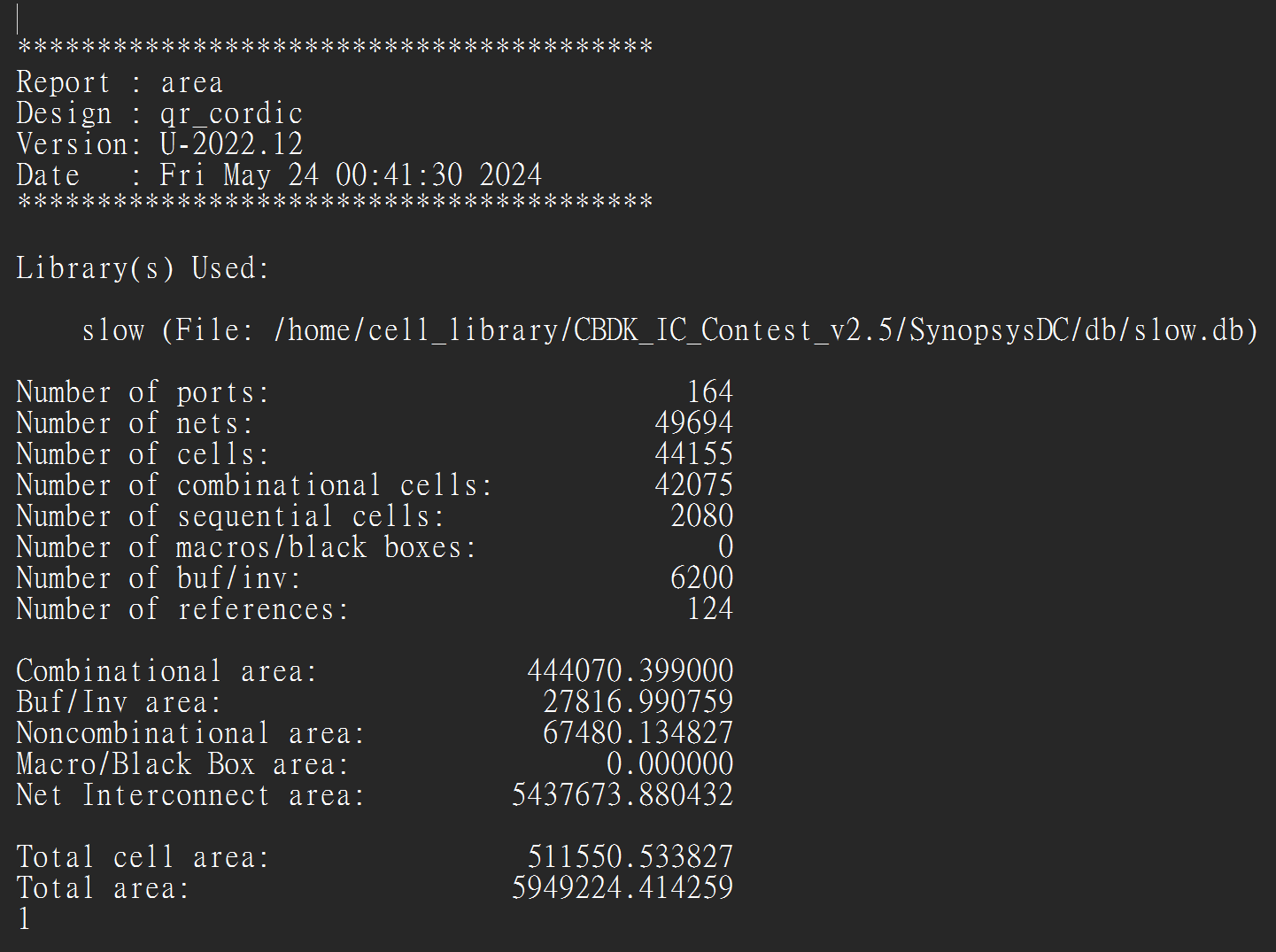
c. synthesis report (clock rate = 50ns)

i. timing report

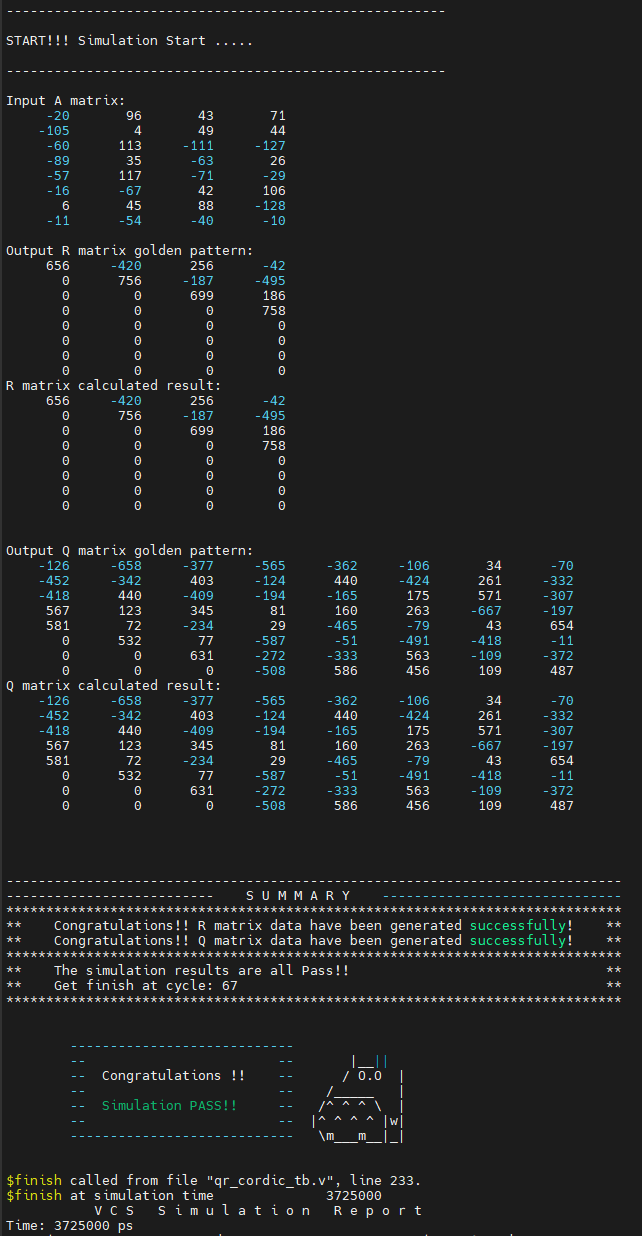
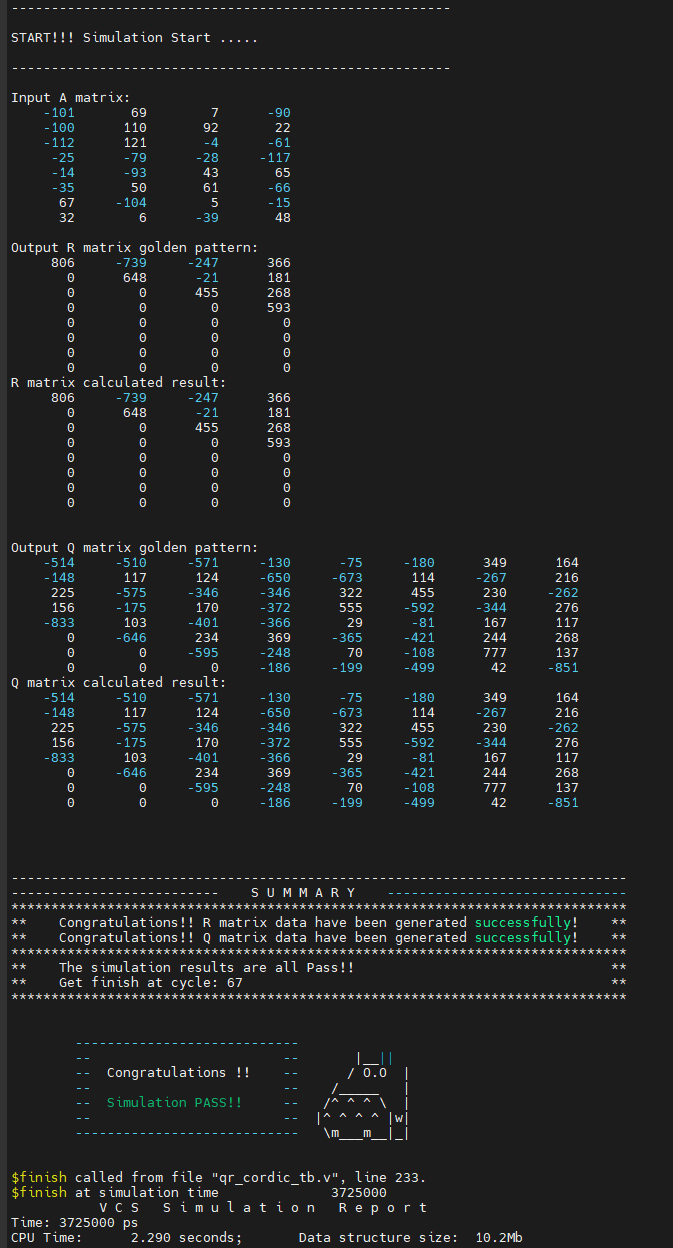


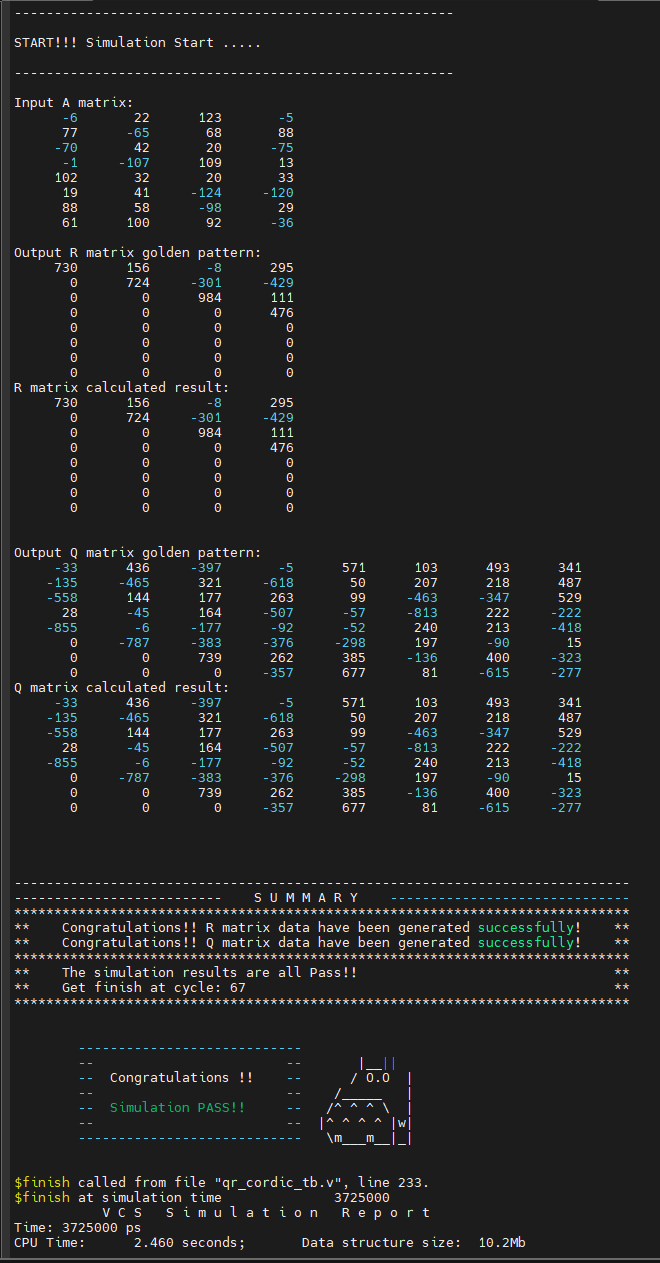
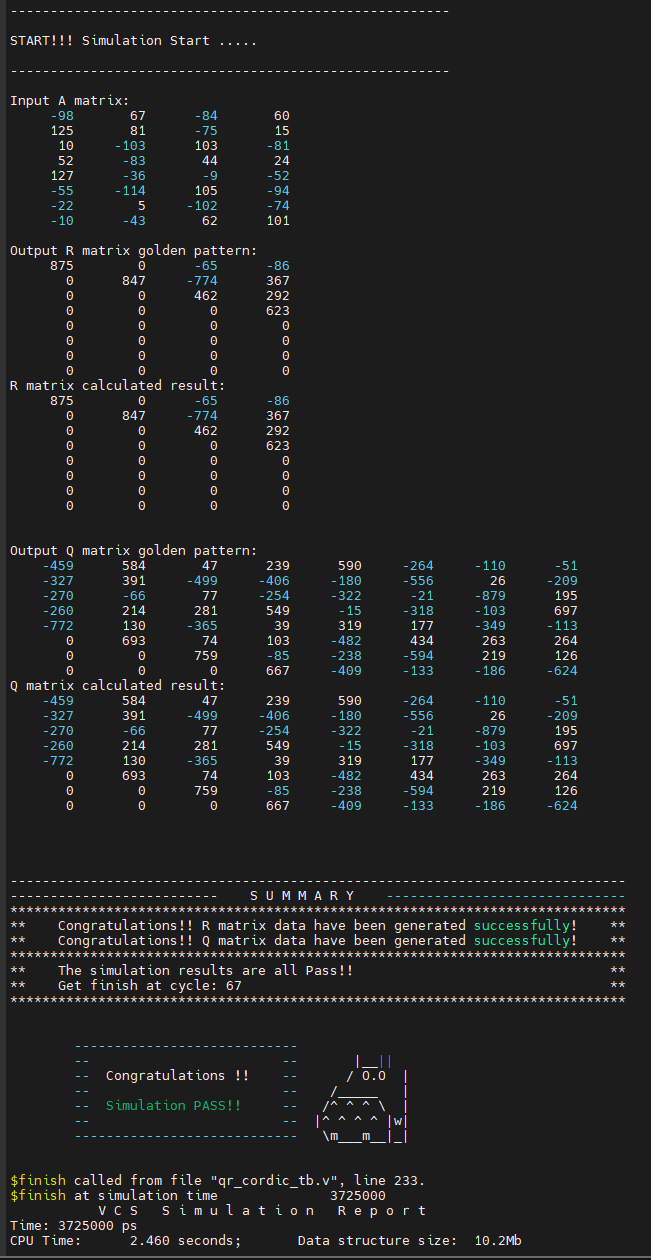


ii. Area report



d. Gate-level simulation

Pattern 1: Pattern 2:

Pattern 3: Pattern4: