# 2022 Spring VLSI DSP Homework Assignment #2

**Due date: 2022/5/3**

## 1. Introduction

For the QR factorization problem given in homework assignment 1, please design its systolic array design. Assume matrix $\mathbf{A}_{8\times4}$, you may apply a sequence of Givens rotation to convert it into an upper triangular matrix $\mathbf{R}_{4\times4}$. As shown in Fig. 1, a total of 22 nullifications is required to obtain a 4×4 triangular matrix. The nullifications proceed from the first column to the last column; and each in column, the nullifications proceed from bottom to top. The basic nullification operation is achieved by using a Givens rotation, which operates on the two neighboring rows.

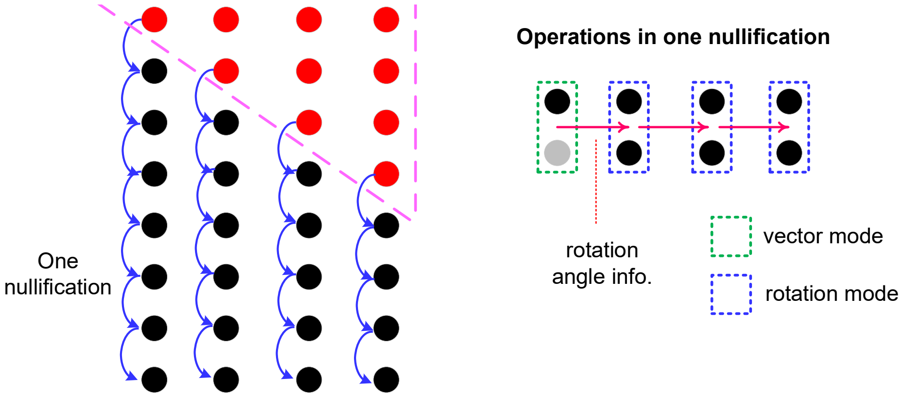

Fig. 1 QR factorization scheme achieved by using a sequence of Givens rotations

## 2. Dependence Graph derivation and mapping

For the factorization scheme described in Fig. 1, please derive its dependence graph first. The row, column, and iteration indexes are *i*, *j*, and *k*, respectively. Then apply space time transform scheme in chapter 5 to derive its systolic array design as shown in Fig. 2. The projection direction is $[1\ 0\ 0]^t$ to obtain this design.

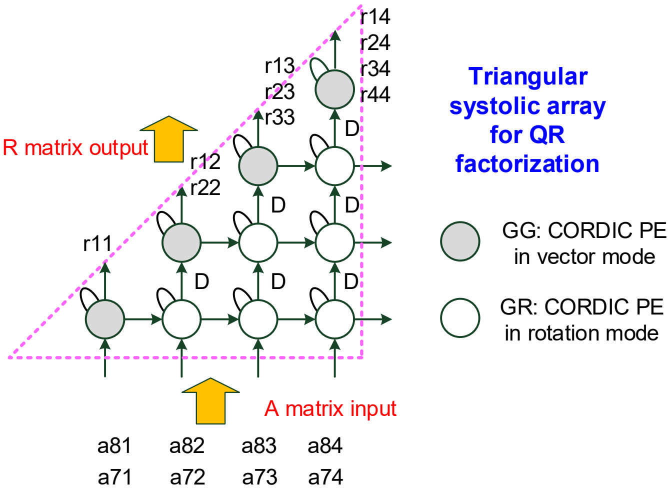

Fig. 2. Triangular systolic array design for QR factorization

Input A matrix is fed to the bottom of the systolic array starting from the last row. The upper triangular matrix is obtained from the PEs along the hypotenuse of the triangular array. Each processing element

(PE) is a CORDIC processor. Those PEs along the hypotenuse operate in the vector mode and also pass rotation angle information to the downstream PEs. The rest of the PEs operate in the rotation mode and receive the rotation angle information from the left PE.

### 3. Scheduling and PE design

- **Scheduling**

  Refer to Fig. 2, There is a delay element D associated with the vertical data link. D indicates the delay needed to complete a Givens rotation. Since the rotation is accomplished by using an iterative CORDIC scheme, it takes multiple clock cycles to complete the rotation. Assume *p* iterations are need in CORDIC iteration, it takes *p*+1 clock cycles. The additional one clock cycle is for the multiplication of scaling factor K. There is no delay element associated with the horizontal data link in the systolic array. This means the leading CORDIC PE needs to determine the micro-rotation direction and then passes the information to all CORDIC PEs in the same row in the same clock cycle. This may lengthen the critical path of your design. Another approach is passing the micro-rotation direction to the downstream PEs one clock cycle per PE. This will shorten the critical path delay, increase the computing latency, but not the initiation interval.

- **CORDIC PE design**

  As described in the previous paragraph, if only one micro-rotation is performed in each clock cycle. It takes *p*+1 clock cycles to complete one Givens rotation. Although the hardware design complexity would be minimized, this will lead to a very long computing latency of the QR factorization. An improved CORDIC architecture, as shown in Fig. 3, performing two micro-rotations in one clock cycle is suggested. The same hardware is used repetitively until all micro rotations are done. At the end of the iterations, a scaling factor is multiplied to obtain the final result. A constant multiplier can be used for this. This can reduce the computing latency from *p*+1 clock cycles to *p*/2+1 clock cycles. Note that in this design, two micro-rotation directions must be passed the downstream CORDIC PEs at a time.

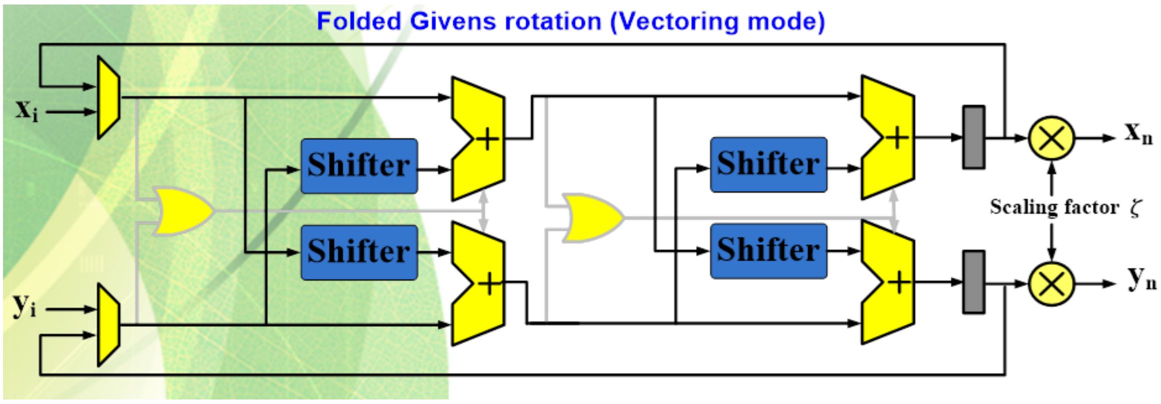

Figure 3. improved CORDIC processor design

### 4. Hand in requirements

- A report highlights your architecture design (system block diagram), PE designs, memory configuration, system timing diagram and the Verilog simulation waveforms.
- The fixed point simulation results including the word length, iteration number and the

quantization error value $\delta$.

- Calculate the number of clock cycles needed to complete one QR factorization
- Calculate your hardware complexity, in terms of adders, shifters and constant multipliers.