2022 Spring VLSI DSP Homework Assignment #1

Due date: 2022/3/22

1. Introduction

In this assignment, you are asked to develop a QR factorization module. Given an 8×4 matrix, the QR factorization converts it to the product of a unitary matrix **Q** and an upper triangular matrix **R**. QR factorization is useful in many digital communication applications and usually demands a high computing throughput. This calls for a dedicated hardware design to meet the demands.

2. QR factorization scheme

Given a 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 adjacent rows.
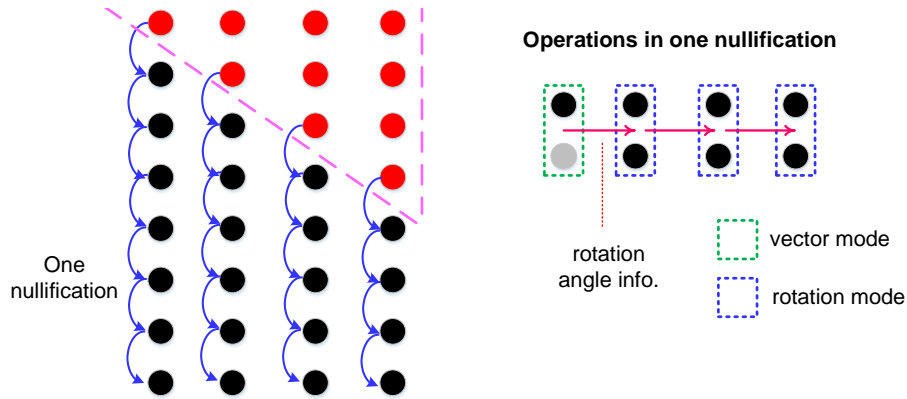

Fig. 1 QR factorization scheme achieved by using a sequence of Givens rotations

3. Basics of CORDIC computing scheme

A CORDIC computing scheme is an iterative algorithm to compute various arithmetic functions. There are three types of CORDIC computations, i.e., circular, linear and hyperbolic. Type 1 (circular) is specific to trigonometric functions. In each iteration, it needs only shift and add operations and thus hardware economical in implementation. Circular type CORDIC is further divided into vectoring mode and rotation mode. In Fig. 1, the green dashed box indicates a vectoring mode computation (determining the rotation angle while nullifying the y "component") while all blue-dashed boxes correspond to rotation mode computations (updating "x" and "y" components subject to the determined rotation angle). Table 1 summarizes the operations of these two modes. "x" and "y" are the two entries of the Givens rotation. Since the rotation angle is not calculated explicitly, the "z" input can be omitted. The rotation angle θ is expressed as a sequence of ±1 indicating the directions of micro-rotations. In the vector mode, a multiplication with the scaling factor K (a constant depending on the number of iterations performed) is needed when all micro-rotations are completed. Similarly, in the rotation mode, input "x" can be set to 1 (instead of 1/K) and an additional final stage scaling is

performed.

Table 1. operation summary of circular type CORDIC scheme

	Rotation mode $d_i = \text{sign}(z_i)$, $z_i \to 0$	Vectoring mode $d_i = -\text{sign}(x_i\, y_i)$, $y_i \to 0$
$\mu = 1$ Circular $e(i) =$ $\tan^{-1} 2^{-i}$	$x \to$ CORDIC $\to K(x \cos z - y \sin z)$ $y \to$ CORDIC $\to K(y \cos z + x \sin z)$ $z \to$ CORDIC $\to 0$ For cos & sin, set x=1/K, y=0 $\tan z = \sin z / \cos z$	$x \to$ CORDIC $\to K (x^2 + y^2)^{1/2}$ $y \to$ CORDIC $\to 0$ $z \to$ CORDIC $\to z + \tan^{-1}(y/x)$ For $\tan^{-1}$, set $x = 1$, $z = 0$ $\cos^{-1} w = \tan^{-1}[(1-w^2)^{1/2}/w]$ $\sin^{-1} w = \tan^{-1}[w/(1-w^2)^{1/2}]$

4. Fixed point word length and CORDIC iteration number simulations

Two hardware design parameters must be determined via simulations. The first one is the variable word length. The second one is the iteration number in the CORDIC scheme. Assume all elements of the input A matrix are 8-bit wide, signed, and the magnitudes are confined to a range of (1~1/4). Please use Matlab to obtain the floating point version of matrix **R** as the reference. The precision requirement for the fixed point version result $\hat{\mathbf{R}}$ is as follows

$$\delta = \frac{\sqrt{\sum (r_{ij} - \hat{r}_{ij})^2}}{\sqrt{\sum r_{ij}^2}} < 0.01$$

You are advised to conduct the simulations in two phases. In phase 1, determine the required word length of matrix **R** elements first by using the trigonometric functions available in Matlab. In phase 2, replace the trigonometric functions with the CORDIC scheme and determine the required iteration number and the accumulation word length in the shift-and-add operations. Subject to the iteration number you derived, you also need to determine the word length of the scaling factor K.

For testbench, please use MatLab to generate at least 10 random matrices of size 8×4 and with a full column rank (4). All magnitudes of all matrix elements should be confined to a range of (1~1/4) and can be either positive or negative. The precision requirement should be met in all cases.

5. Hand in requirements

- Your Matlab or C code with comments to explain the code
- Show the 10 random matrices you used as the testbench
- Show the floating point version results of QR factorization, verify that the Q matrix obtained in indeed unitary
- Show the fixed point simulation results including the word length, iteration number and the quantization error value δ.