Lesson 104: COordinate Rotation DIgital Computer (CORDIC)

Motivation

- Assume vector $[x, y]^T$ is being rotated with an angle θ
- Assuming the rotated vector is $[x', y']^T$
- The equation set that describes this rotation is:

$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = y\cos\theta - x\sin\theta \end{cases}$$

- Direct evaluation is computationally demanding
  - Evaluation of trigonometric functions translates to a sequence of multiplications, additions, and memory look-up operations if the common Taylor series expansion is employed.

COordinate Rotation DIgital Computer (CORDIC)

- **CORDIC** is an iterative method performing vector rotations by arbitrary angles using only shifts and additions
  - **Cheap**: only shifts and additions are needed
  - **Sequential**: it is an iterative method

- The iterative method: the rotation angle θ is splitted into a sequence of subrotations of elementary angles $\theta[i]$, where the rotation for iteration i is

$$\begin{cases} x[i+1] = x[i]\cos\theta[i] + y[i]\sin\theta[i] \\ y[i+1] = y[i]\cos\theta[i] - x[i]\sin\theta[i] \end{cases}$$

- The elementary angles $\theta[i]$ are predefined

CORDIC

- Only shifts and additions: rotation angles are restricted so that $\tan\theta[i] = \pm 2^{-i}$
  - Multiplication by the tangent factor is reduced to a shift operation:

$$\begin{cases} x[i+1] = x[i]\cos\theta[i] + y[i]\sin\theta[i] = \cos\theta[i](x[i] + y[i]\tan\theta[i]) \\ y[i+1] = y[i]\cos\theta[i] - x[i]\sin\theta[i] = \cos\theta[i](y[i] - x[i]\tan\theta[i]) \end{cases}$$

$$\begin{cases} x[i+1] = \cos\theta[i](x[i] + 2^{-i}y[i]) \\ y[i+1] = \cos\theta[i](y[i] + 2^{-i}x[i]) \end{cases}$$

- Arbitrary rotation angles can be obtained by performing a series of successively smaller elementary rotations

- The decision at each iteration is which direction to rotate
  - The factor $\cos\theta[i]$ is a constant for the current iteration
  - The product of all these cosine values is a constant = system processing gain

CORDIC

- The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations
- That sequence can be represented by a decision vector
- The set of all possible decision vectors is an angular measurement system based on binary arctangents
- Conversions between this angular system and any other can be accomplished using an additional adder-subtractor that accumulates the elementary rotation angles at each iteration

$$z[i+1] = z[i] - \sigma[i] \arctan\left(2^{-i}\right)$$

- The elementary angles are supplied by a small look-up table (one entry per iteration), or are hardwired, depending on the implementation

The arctangent table in degrees

$\theta(i) = \arctan(2^{-i})$

Iteration i	Elementary angle degrees
0	45.00
1	26.56
2	14.04
3	7.13
4	3.58
5	1.79
6	0.89
7	0.45

Example: $30.00 \approx 45.00 - 26.56 + 14.04 - 7.13 + 3.58 + 1.79 - 0.89 + 0.45$

CORDIC rotation mode

- The angle accumulator is initialized with the desired rotation angle
- The rotation decision at each iteration is made to diminish the magnitude of the residual angle in the angle accumulator

$$\begin{cases} x[i+1] = x[i] - \sigma[i]2^{-i}y[i] \\ y[i+1] = y[i] + \sigma[i]2^{-i}x[i] \\ z[i+1] = z[i] - \sigma[i]\arctan\left(2^{-i}\right) \end{cases} \quad \text{where} \quad \sigma[i] = \begin{cases} -1 & \text{if } z[i] < 0, \\ +1 & \text{otherwise.} \end{cases}$$

- After n iterations, the result is:

$$\begin{cases} x[n] = A[n](x[0]\cos z[0] - y[0]\sin z[0]) \\ y[n] = A[n](y[0]\cos z[0] - x[0]\sin z[0]) \\ z[n] = 0 \end{cases} \quad \text{where} \quad A[n] = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

CORDIC vectoring mode

- The input vector is rotated through whatever angle is necessary to align the result vector with the x axis
- The result of the vectoring operation is a rotation angle and the scaled magnitude of the original vector (the x component of the result)

$$\begin{cases} x[i+1] = x[i] - \sigma[i]2^{-i}y[i] \\ y[i+1] = y[i] + \sigma[i]2^{-i}x[i] \\ z[i+1] = z[i] - \sigma[i]\arctan\left(2^{-i}\right) \end{cases} \quad \text{where} \quad \sigma[i] = \begin{cases} -1 & \text{if } y[i] \geq 0, \\ +1 & \text{otherwise.} \end{cases}$$

- After n iterations, the result is:

$$\begin{cases} x[n] = A[n]\sqrt{x_0^2 + y_0^2} \\ y[n] = 0 \\ z[n] = z[0] + \arctan(y[0]/x[0]) \end{cases} \quad \text{where} \quad A[n] = \prod_{i=0}^{n} \sqrt{1 + 2^{-2i}}$$

CORDIC numerical properties

- The CORDIC algorithm produces one bit of accuracy for each iteration
  - Accuracy can be adjusted dynamically by adding or removing iterations
- To preserve N bits of significance in a fixed-point implementation, $\log_2 N$ additional low-order bits are necessary for intermediate values.
  - $N + \log_2 N$-bit word length is needed for N-bit CORDIC precision
  - Example: 12-bit CORDIC precision is guaranteed with a 16-bit wordlength
- Domain of convergence is $-\pi/2 \cdots + \pi/2$
  - Rotation mode: $-\pi/2 \leq \theta \leq +\pi/2$
  - Vectoring mode: $x \geq 0$
- Vectoring mode with a zero input vector: the result is undefined.

How to calculate transcendental functions using CORDIC

- $\mathbf{arctan}(y/x)$
  - Build a vector $[x, y]^T$ with the x and y
  - Initialize $z = 0$ and run CORDIC in vectoring mode
  - After n iterations, $z = \arctan(y/x)$ with n-bit precision

- $\mathbf{arctan}(x)$
  - Build a vector $[1, x]^T$ with the x and y
  - Initialize $z = 0$ and run CORDIC in vectoring mode
  - After n iterations, $z = \arctan(x)$ with n-bit precision

- $\mathbf{cos}\,\boldsymbol{\theta}$ and $\mathbf{sin}\,\boldsymbol{\theta}$
  - Build a vector $[1, 0]^T$
  - Initialize $z = \theta$ and run CORDIC in rotation mode
  - After n iterations, $x = \cos\theta$, and $y = \sin\theta$ with n-bit precision

CORDIC – project requirements

- Build the testbench:
  - Values for x and y to calculate $\arctan$
  - values for θ to calculate $\cos$ and $\sin$
- Implement the CORDIC algorithm using integer arithmetic
  - software (write C routines)
  - horizontal firmware with two issue slots
  - custom hardware (write VHDL/Verilog)
- Define a new instruction that will return the trigonometric function
  - You must comply with the ARM architecture (you can have at most two arguments and one result per instruction call)

CORDIC – project requirements

- Rewrite the high-level code and instantiate the new instruction
  - Use assembly inlining
- Estimate
  - the performance improvement of hardware-based solution versus software-based solution
  - the performance improvement of a 2-issue slot firmware-based solution versus software-based solution
- Estimate the penalty in terms of number of gates for the hardware solution

Questions, feedbacks