Coordinate Rotation DIgital Rotation(CORDIC)

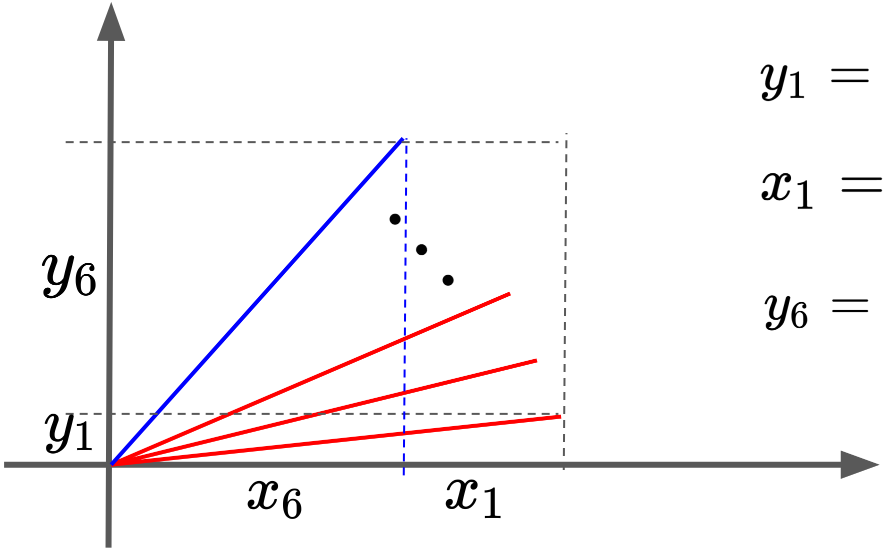
Speaker: Jia-Ming Lin

Outline

- How to calculate sine and cosine?
- Background of CORDIC
- Calculating sine and cosine efficiently
- Number Representation
- Labs
    - Download
    - Baseline
    - Using CORDIC

How to calculate Sine and Cosine?

- If we know $\sin(10^\circ) = 0.1736$
- How to calculate $\sin(60^\circ)$
- Using rotation, doing five times



$$y_1 = \sin(10^\circ)$$

$$x_1 = \sqrt{1 - y_1^2}$$

$$y_6 = \sin(60^\circ)$$

How to calculate Sine and Cosine?

- Rotation matrix,

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

- To perform one rotation

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \begin{array}{l} \text{cosine} \\ \\ \text{sine} \end{array}$$

How to calculate Sine and Cosine?

- Back to the example
- If we know $\sin(10^\circ) = 0.1736$
- Calculate $\sin(60^\circ)$ by using rotation five times

$\cos(0^\circ)$

$$\underbrace{\begin{bmatrix} \cos(10^\circ) & -\sin(10^\circ) \\ \sin(10^\circ) & \cos(10^\circ) \end{bmatrix} \cdots \begin{bmatrix} \cos(10^\circ) & -\sin(10^\circ) \\ \sin(10^\circ) & \cos(10^\circ) \end{bmatrix}}_{\text{Six matrix multiplication}} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$\sin(0^\circ)$

How to calculate Sine and Cosine?

- How many “multiplication units” we need?
    - At least 4
- How many iterations(rotations) we need?
    - Target degree / 10
    - E.g. 6 times in our case
- Can we do better?
    - In terms of less multiplication units and consistent latency
    - Idea of CORDIC is an efficient way to perform a series of rotations
        - No multiplication units and constant iterations

Background of CORDIC

CORDIC: Background(1/2)

- Consider the rotation matrix

$$R_i(\theta) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix}$$

- Using the following trigonometric identities

$$\cos(\theta_i) = \frac{1}{\sqrt{1+\tan^2(\theta_i)}} \qquad \sin(\theta_i) = \frac{\tan(\theta_i)}{\sqrt{1+\tan^2(\theta_i)}}$$

- Rewrite the rotation matrix

$$R_i = \frac{1}{\sqrt{1+\tan^2(\theta_i)}} \begin{bmatrix} 1 & -\tan(\theta_i) \\ \tan(\theta_i) & 1 \end{bmatrix}$$

CORDIC: Background(2/2)

- Rotation matrix

$$R_i = \frac{1}{\sqrt{1+\tan^2(\theta_i)}} \begin{bmatrix} 1 & -\tan(\theta_i) \\ \tan(\theta_i) & 1 \end{bmatrix}$$

- If $\tan(\theta_i) = 2^{-i}$, the rotation can be performed using “shift” and “additions”

$$v_i = K_i \begin{bmatrix} 1 & -2^{-i} \\ 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} \qquad K_i = \frac{1}{\sqrt{1+2^{-2i}}}$$

- Rotate by $\pm\theta$.

$$v_i = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

$\sigma_i = 1$ Positive rotation

$\sigma_i = -1$ Negative rotation

CORDIC: Example $\sin(60^\circ)$

$$v_i = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

$$K_i = \frac{1}{\sqrt{1+2^{-2i}}} \qquad x_{-1} = 1, y_{-1} = 0$$

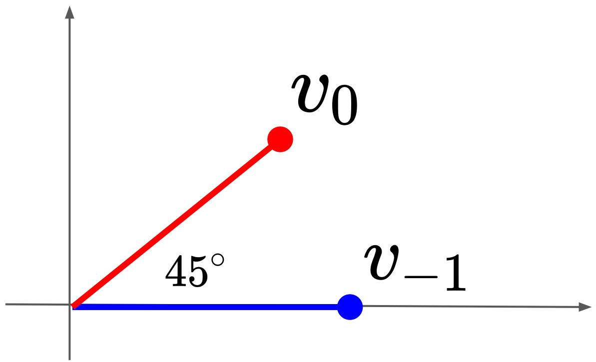
Lookup table: cordic_phase

i	2^{-i}	Rotating Angle	Scaling Factor	CORDIC Gain
0	1.0	45.000°	1.41421	1.41421
1	0.5	26.565°	1.11803	1.58114
2	0.25	14.036°	1.03078	1.62980
3	0.125	7.125°	1.00778	1.64248
4	0.0625	3.576°	1.00195	1.64569
5	0.03125	1.790°	1.00049	1.64649
6	0.015625	0.895°	1.00012	1.64669

$v_{-1} \longrightarrow v_0$

σ_0= (0<60)?1:-1

Current deg < target deg?



CORDIC: Example $\sin(60^\circ)$

$$v_i = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

$$K_i = \frac{1}{\sqrt{1+2^{-2i}}} \qquad x_{-1} = 1, y_{-1} = 0$$

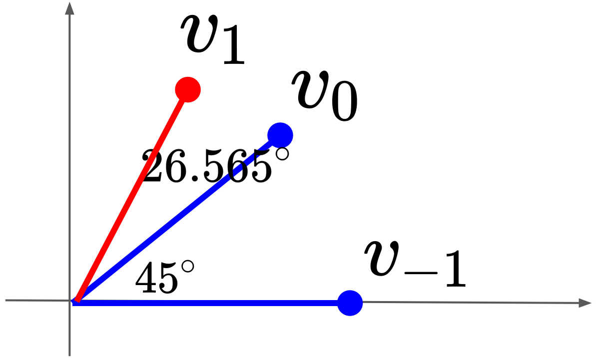
Lookup table: cordic_phase

i	2^{-i}	Rotating Angle	Scaling Factor	CORDIC Gain
0	1.0	45.000°	1.41421	1.41421
1	0.5	26.565°	1.11803	1.58114
2	0.25	14.036°	1.03078	1.62980
3	0.125	7.125°	1.00778	1.64248
4	0.0625	3.576°	1.00195	1.64569
5	0.03125	1.790°	1.00049	1.64649
6	0.015625	0.895°	1.00012	1.64669

$$v_0 \longrightarrow v_1$$

σ_1 = (45<60)?1:-1

Current deg < target deg?



CORDIC: Example $\sin(60^\circ)$

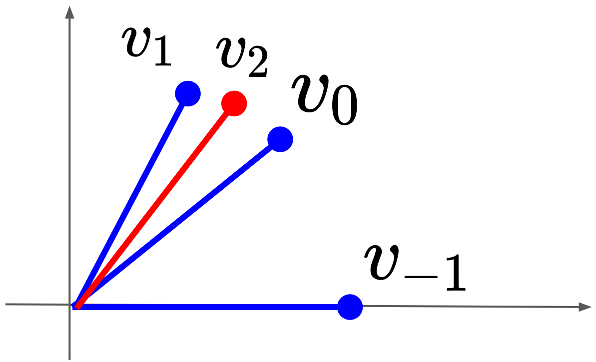
$$v_i = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix}$$

$$K_i = \frac{1}{\sqrt{1 + 2^{-2i}}} \qquad x_{-1} = 1, y_{-1} = 0$$

Lookup table: cordic_phase

i	2^{-i}	Rotating Angle	Scaling Factor	CORDIC Gain
0	1.0	45.000°	1.41421	1.41421
1	0.5	26.565°	1.11803	1.58114
2	0.25	14.036°	1.03078	1.62980
3	0.125	7.125°	1.00778	1.64248
4	0.0625	3.576°	1.00195	1.64569
5	0.03125	1.790°	1.00049	1.64649
6	0.015625	0.895°	1.00012	1.64669

$$v_1 \longrightarrow v_2$$



σ_2= (45+26.565<60)?1:-1

Current deg < target deg?

Calculate Sine and Cosine efficiently: Procedure

1. Initialize
    - Starting from zero degree, theta = 0,
    - Initial vector = [current_cos = 1,current_sin = 0]
    - Lookup table: cordic_phase
    - Max Iterations
2. j = 0...max_iteration
    - Positive rotate or negative rotate? sigma = (theta < target degree)?
    - Multiply 2^(-j)
        - cos_shift = (current_cos >> j)*sigma
        - sin_shift = (current_sin >> j)*sigma

      Right shift in HLS
    - Rotation
        - current_cos = current_cos - sin_shift
        - current_sin = current_sin + cos_shift
    - Current degree, theta = theta + cordic_phase[j]
3. Output [“current_cos”, “current_sin”] * 0.60725

Calculate Sine and Cosine efficiently: Normalization

- In the final output result, we scaled the vector by a factor 1.64676
  - The cumulative scaling factor

$$K(\text{max\_iter}) = \prod_{j=0}^{\text{max\_iter}} K_j$$

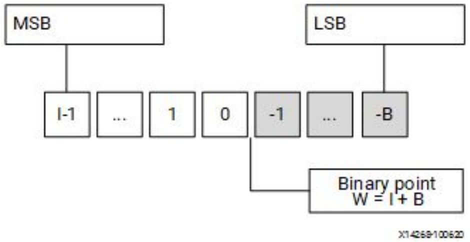
  - When max_iter → infinity,

$$K \sim 0.60725$$

- To avoid the final normalization, initialize the starting vector as [0.60725, 0]

Number Representation

- Datatype of variables store sin and cos values
    - **Floating point**:
        - **Advantage**: accurate in most cases,
            - Smallest positive number: 1.175494e-38, Max number: 3.402823466e+38
        - **Disadvantage**: resource cost
    - **Fixed-Point**:
        - #include<ap_fixed.h>;
          ap_fixed<W, I, Q, O>
        - **W**: word length, **I**: bits length of integer value, **Q**: quantization mode, **O**: Overflow mode
            - Q is default to “Truncation to minus infinity”,
            - O is default to “Wrap around”.



Number Representation

- **Example**: ap_fixed<8,4>
    - W = 8, I = 4
    - 8 bits variable, 4 bits representing the integer, 4 bits representing fractional number.

-2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	two's complement
1	0	1	1	0	1	0	0	$= -4.75$

    - (-1*2**3) + 2**1 + 2**0 + 2**(-2)

Lab 1: Baseline implementation

- Given $\sin(1^\circ) = 0.01745$
- Using rotation to calculate
  - $\sin(\theta), \theta \in [1:89]$
- Implementation in HLS and generate the report
- Compare the solution with results from standard library
- Python code for your reference

```python
t = 1
init = np.array([[1],[0]])

def sincos(theta):
    temp = init
    for i in range(theta):
        temp = rotate(t, temp)

    return temp
sincos(69)
```

Lab 2: Implement CORDIC

- Implement a function to calculate “sin” and “cos” in HLS using CORDIC
  - Calculate $\sin(\theta), \theta \in [1:89]$
- Generate Report, and compare with Lab 1
  - Resource consumption, Latency, Accuracy
- Change “data_t” to floating point or fixed point
  - Comparing the accuracy and resource consumption