THE CORDIC ALGORITHM AND CORDIC ARCHITECTURES

Implementation of Digital Signal Processing

Sabih H. Gerez
University of Twente

OUTLINE

• CORDIC algorithm:
  – Rotation and vectoring modes
• CORDIC architectures
• Applications of CORDIC

REFERENCES


WHAT IS CORDIC?

• CORDIC: abbreviation of *coordinate rotation digital computer*.
• First publication by Volder, 1959.
• A method from the field of *computer arithmetic* allowing for the efficient implementation of a wide range of computations.
VECTOR ROTATIONS (1)

- Consider a sequence of rotations of a vector \( (x^{(i)}, y^{(i)})^T \) rotated by \( a_i \) to give vector \( (x^{(i+1)}, y^{(i+1)})^T \).
- So:
  \[
  \begin{bmatrix}
  x^{(i+1)} \\
  y^{(i+1)} 
  \end{bmatrix} =
  \begin{bmatrix}
  \cos(a_i) & -\sin(a_i) \\
  \sin(a_i) & \cos(a_i) 
  \end{bmatrix}
  \begin{bmatrix}
  x^{(i)} \\
  y^{(i)} 
  \end{bmatrix}
  
\]
- After rewrite:
  \[
  \begin{bmatrix}
  x^{(i+1)} \\
  y^{(i+1)} 
  \end{bmatrix} = \cos(a_i) \begin{bmatrix}
  1 & -\tan(a_i) \\
  \tan(a_i) & 1 
  \end{bmatrix} \begin{bmatrix}
  x^{(i)} \\
  y^{(i)} 
  \end{bmatrix}
  
\]
- If \( \tan(a_i) \) is chosen such that \( \tan(a_i) = d_i 2^{-i} \), with \( d_i = \pm 1 \), then the rotations can be executed without multiplications except for initial factor \( \cos(a_i) = \frac{1}{\sqrt{1+2^{-2i}}} \).

VECTOR ROTATIONS (2)

- If \( \tan(a_i) = d_i 2^{-i} \), this means: \( a_i = d_i \arctan(2^{-i}) \).
- For an arbitrary angle \( \alpha \), the angle can then be decomposed as:
  \[
  \alpha = \sum_{i=0}^{n} d_i \arctan(2^{-i})
  \]
- Angles involved:
  \[
  \begin{array}{cccccccccc}
  \alpha & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  2^{-i} & 1 & 1/2 & 1/4 & 1/8 & 1/16 & 1/32 & 1/64 & 1/128 & 1/256 \\
  \arctan(2^{-i})[\text{deg}] & 45.0 & 26.6 & 14.0 & 7.1 & 3.6 & 1.8 & 0.9 & 0.4 & 0.2 \\
  \end{array}
  \]

VECTOR ROTATION EXAMPLE

- The 8 subsequent rotations for a rotation of 15 degrees are:
  \[
  \begin{array}{cccccccc}
  i & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
  2^{-i} & 1 & 1/2 & 1/4 & 1/8 & 1/16 & 1/32 & 1/64 & 1/128 & 1/256 \\
  \arctan(2^{-i})[\text{deg}] & 45.0 & 26.6 & 14.0 & 7.1 & 3.6 & 1.8 & 0.9 & 0.4 & 0.2 \\
  d_i & 1 & -1 & -1 & 1 & 1 & -1 & 1 & 1 & 1 \\
  \Sigma a_i & 45.0 & 18.4 & 4.4 & 11.5 & 15.1 & 13.3 & 14.2 & 14.7 & 14.9 \\
  \end{array}
  \]

- The arctangent values can be precomputed and stored in a look-up table (LUT), say \( L(i) \).
- The \( d_i \) depend on the required rotation angle.

ANGLE ACCUMULATION

- Keep track of total rotation angle in an angle accumulator:
  \[
  z^{(i+1)} = z^{(i)} - d_i L(i)
  \]
- The angle accumulator can be used to determine \( d_i \):
  - Initialize \( z^{(0)} = \alpha \).
  - Factor \( d_{i+1} \) becomes 1 when \( z^{(i)} \geq 0 \) and -1 otherwise.
CORDIC EQUATIONS SUMMARY

- Original equations were:

\[
\begin{bmatrix}
    x(i+1) \\ y(i+1)
\end{bmatrix} =
\begin{bmatrix}
    1 & -\tan(\alpha_i) \\ \tan(\alpha_i) & 1
\end{bmatrix}
\begin{bmatrix}
    x(i) \\ y(i)
\end{bmatrix}
\]

- Making use of the special values for the tangent, leaving out the multiplication by the cosine and combining with angle accumulation, one gets:

\[
\begin{aligned}
x(i+1) &= x(i) - d_i 2^{-i} y(i) \\
y(i+1) &= d_i 2^{-i} x(i) + y(i) \\
z(i+1) &= z(i) - d_i L(i)
\end{aligned}
\]

ROTATION-MODE CORDIC

- Goal is to rotate vector by angle \( \alpha \).

- Initialization:

\[
\begin{aligned}
x(0) &= x \\
y(0) &= y \\
z(0) &= \alpha
\end{aligned}
\]

- Final result:

\[
\begin{aligned}
x(n) &= K (x \cos(\alpha) - y \sin(\alpha)) \\
y(n) &= K (x \sin(\alpha) - y \cos(\alpha)) \\
z(n) &= 0
\end{aligned}
\]

- Where:

\[
K = \prod_{i=1}^{n} \sqrt{1 + 2^{-2i}}
\]

- \( K \) converges to 1.647.

- Conclusion: the result vector is rotated but scaled version of original vector.

VECTORIZING-MODE CORDIC

- Determine \( d_i \) by an alternative rule:

\[
d_i = \begin{cases} 
-1 & \text{when } y(i) > 0 \\
+1 & \text{when } y(i) \leq 0
\end{cases}
\]

- Initialization:

\[
\begin{aligned}
x(0) &= x \\
y(0) &= y \\
z(0) &= 0
\end{aligned}
\]

- Final result:

\[
\begin{aligned}
x(n) &= K \sqrt{x^2 + y^2} \\
y(n) &= 0 \\
z(n) &= \arctan \left( \frac{y}{x} \right)
\end{aligned}
\]

- This means that the initial vector has been rotated (and scaled) onto the X-axis, while the angle with the X-axis has been computed as well.

BASIC APPLICATIONS OF CORDIC

- **Arctangent, vector-magnitude calculation** and **rectangular-to-polar conversion**: direct result of vectoring-mode CORDIC.

- **Polar-to-rectangular conversion**, i.e. from \((r, \theta)\) to \((x, y)\):

  - Set \( x(0) = r, \ y(0) = 0 \), and \( z(0) = \theta \) in rotation mode.

  - Result will be \( x = x(n) = Kr \cos(\theta), \ y = y(n) = Kr \sin(\theta) \).

  - Correction for scaling by \( K \) may be necessary (does not require a full-fledged multiplier as \( K \) is constant).

- **Sine or cosine** calculation:

  - See above, set \( x(0) = 1/K \). Then \( x(n) = \cos(\theta) \) and \( y(n) = \sin(\theta) \).
Controller should take care of initializations, add/subtract decisions, number of iterations, etc.

The iterative architecture requires one clock cycle per iteration.
- It requires a barrel shifter to shift operand over a variable number of positions.

One can also unroll the architecture to perform all operations in a single clock cycle:
- Amounts to instantiate new hardware for each iteration.
- Possibly adding pipelining if the critical path becomes too long.
- The barrel shifter is no longer necessary: each stage in the hardware has a fixed shift which costs just wires.
- One could also unroll the architecture partially.

What is GFSK?
- Gaussian frequency shift keying
- Method for digital transmission based on frequency modulation (FM).
- To transmit a 1 carrier frequency is slightly increased and to transmit a 0 the frequency is slightly decreased (or vice versa).
- The transition steps are smoothed by a Gaussian filter.
- Found in many standards: Bluetooth, DECT, Wavenis, ...
- Proposed version uses parameters not related to any standard.

Model entire system: transmitter, receiver, and a channel adding noise (AWGN).
- Leave out analog circuitry for upconversion to RF and downconversion back to IF.
- Use IT++ to set up testbench.
- The testbench computes bit error rates (BERs) for different signal-to-noise ratios (SNRs).
- Goal is to preserve BER performance when designing hardware.
IMPLEMENTATION ASPECTS

• Projects focus on designing in Arx.
• Testbenches for generated C++ and VHDL will be provided.
• As C++ and VHDL behave exactly the same, most simulations will be done in C++ (simulation speed in e.g. BER simulations is important).
• C++ testbenches make use of IT++, an open-source library for telecom/signal processing:
  – It provides Matlab-style programming in C++, so vectors, matrices, etc. and lots of powerful functions to manipulate them.

GFSK: MODULATION IN FORMULAE

• The modulated signal: \( s(t) = A \cos(\omega_{IF} t + \phi(t)) \)
• where:
  – \( A \) is the constant amplitude
  – \( \omega_{IF} \) is the intermediate frequency (acts as carrier frequency)
  – \( \phi(t) \) is the phase deviation, derived from the bit stream
• The phase deviation:
  \[ \phi(t) = h \pi \int_{-\infty}^{t} \sum_{i} a_i g(\tau - iT) d\tau \]
• where:
  – \( h \) is the modulation index
  – \( g(t) \) is a Gaussian-filtered square wave
  – \( a_i \) is 1 for a transmitted 1 and -1 for a transmitted 0.
Digital downconversion is a common operation in digital radio receivers. It is used to shift the carrier frequency of a radio signal (e.g. from IF to baseband) or correct for frequency offset.

This is done by multiplying an input signal by a sine and cosine of some frequency. Think of the GFSK demodulator.

CORDIC FOR DOWNCONVERSION (2)

\( I(k) = s_{BP}(k) \cos \left( 2 \pi \frac{f_c}{f_s} \right) \)

\( Q(k) = -s_{BP}(k) \sin \left( 2 \pi \frac{f_c}{f_s} \right) \)

This solution requires look-up table (LUT) and 2 multipliers.

CORDIC FOR DOWNCONVERSION (3)

This solution requires just one CORDIC.
CORDIC FOR DOWNCONVERSION (4)

IMAGE REJECTION MIXER: “complex” input with in-phase (I) and quadrature (Q) component. CORDIC replaces 4 multipliers!

Phase accumulator

Loehning, et al. Figure 4