

CORDIC Fixed Point Simulation

Copyright (c) 2012 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Based on the following site:

drdobbs.com

“Implementing CORDIC Algorithms”, P. Jarvis, Dr Dobb's

ANSI-C version of the above by P. Knoppers.

Circular

```
void Circular (long x, long y, long z)
{
  int i;
  X = x;
  Y = y;
  Z = z;
  for (i = 0; i <= fractionBits; ++i)
  {
    x = X >> i;
    y = Y >> i;
    z = atan[i];
    X -= Delta (y, Z);
    Y += Delta (x, Z);
    Z -= Delta (z, Z);
  }
}
```

$$x \Rightarrow x'_{i+1} = (x'_i - y'_i \sigma_i 2^{-i})$$

$$y \Rightarrow y'_{i+1} = (x'_i \sigma_i 2^{-i} + y'_i)$$

$$z \Rightarrow \alpha_{i+1} = \alpha_i - \tan(\sigma_i 2^{-i})$$

$$x = X \gg i; \quad \Rightarrow X \cdot 2^{-i}$$

$$y = Y \gg i; \quad \Rightarrow Y \cdot 2^{-i}$$

z = atan[i];

$$X -= \text{Delta}(y, Z); \quad \Rightarrow X = (X - Y \sigma_i 2^{-i})$$

$$Y += \text{Delta}(x, Z); \quad \Rightarrow y'_{i+1} = (x'_i \sigma_i 2^{-i} + y'_i)$$

Z -= Delta (z, Z);

Delta

```
#define Delta(n, Z) (Z >= 0) ? (n) : -(n)
```

-2π $-\pi$ 0 π 2π

References

- [1] <http://en.wikipedia.org/>
- [2] "Implementing CORDIC Algorithms", P. Jarvis, Dr Dobb's
- [3] ANSI-C version of [2] by P. Knoppers.