CORDIC IN CHISEL

BY MATTHEW MOSHER

https://github.com/Mohound22/CHISEL-CORDIC

CO ordinate R otation DI gital C omputer

1. Set initial values

2. Rotate by precomputed constants

3. Repeat

4. Profit???

WHY THOUGH?

- → FPGAs and ASICs are often hardware and space limited
- → The CORDIC algorithms can calculate complicated functions like sin/cos, arctan, ln(x), e^x, multiplication
- → The CORDIC algorithms uses additions and rotations instead of multiplications to do calculations
- → Faster implementation than using a non hardware multiplier
- → Can be parallelized and pipelined for even faster calculations

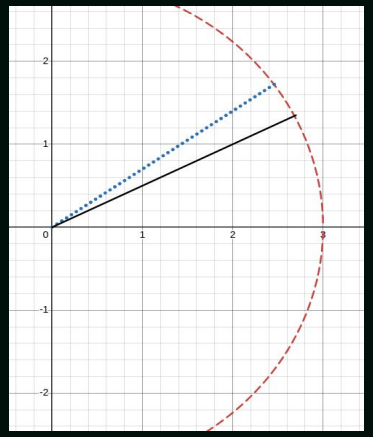
WHAT IT DOES

CIRCULAR
- Sin/Cos
- ArcTan/Magnitude

HYPERBOLIC
- Sinh/Cosh
- e^x/e^-x
- ArcTanh/Magnitude
- ln(x)

LINEAR
- Multiplication
- Division

CIRCULAR



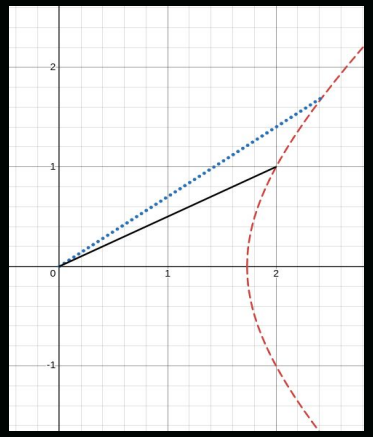
$$\begin{cases} x_{i+1} = x_i - \sigma_i * y_i * 2^{-i} \\ y_{i+1} = y_i + \sigma_i * x_i * 2^{-i} \\ z_{i+1} = z_i - \sigma_i * arctan(2^{-i}) \end{cases}$$

Constraints: pi/2 > z > -pi/2

Gain: 1.64676

Precision: ~1 bit / cycle

HYPERBOLIC



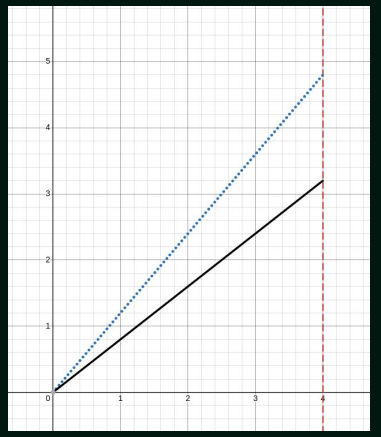
$$\begin{cases} x_{i+1} = x_i + \sigma_i * y_i * 2^{-i} \\ y_{i+1} = y_i + \sigma_i * x_i * 2^{-i} \\ z_{i+1} = z_i - \sigma_i * arctanh(2^{-i}) \end{cases}$$

Constraints: 1.118 > z > -1.118

Gain: 0.82815

Precision: ~0.95 bits / cycle

LINEAR



$$\begin{cases} x_{i+1} = x_i \\ y_{i+1} = y_i + \sigma_i * x_i * 2^{-i} \\ z_{i+1} = z_i - \sigma_i * 2^{-i} \end{cases}$$

Constraints: 2 > z > -2

Gain: No Gain

Precision: ~1 bit / cycle

THE INTERFACE

- → MODE (DECOUPLED)
- → INPUT X
- → INPUT Y
- → INPUT THETA

THE OUTERFACE

- → OUTPUT 1 (DECOUPLED)
- → OUTPUT 2 (DECOUPLED)

MEET THE PARAMETERS

WIDTH
- Sets internal and external wire widths

INTEGER BITS
- Sets location of the fixed point for inputs and calculations

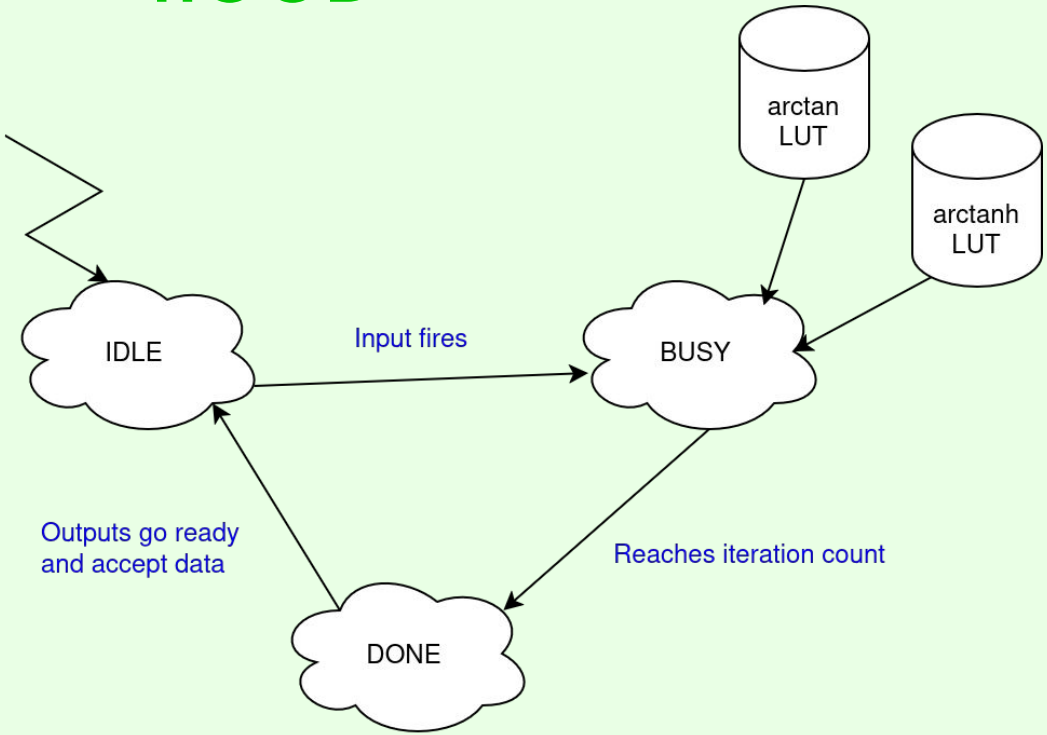
CYCLE COUNT
- Sets the number of cycles to complete a calculation
- Affects precision and latency

GAIN CORRECTION
- Turns gain correction on or off which affects accuracy

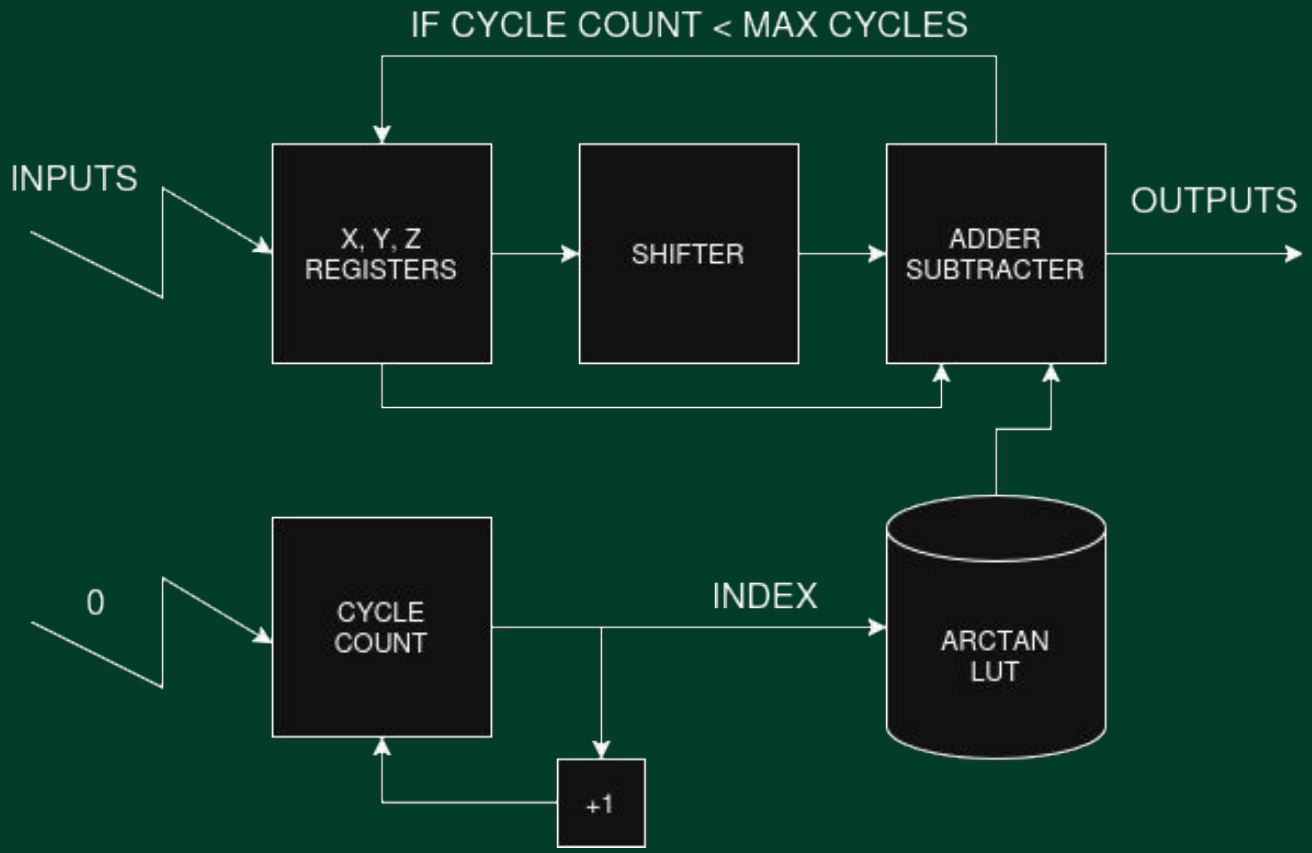
INCLUDE HYPERBOLIC
- Adds or removes the hyperbolic CORDIC operations at compile time

INCLUDE LINEAR
- Adds or removes the linear CORDIC operations at compile time

UNDER THE HOOD



SIMPLIFIED DATAPATH



STARTED FROM THE BOTTOM NOW WE'RE HERE

:|

:<

:]

:()

Sin/Cos Scala Model

- Started just learning the CORDIC algorithm in scala
- Created model and test bench to ensure functionality

CORDIC in CHISEL

- Took the learning from the scala model and implemented it in CHISEL
- Used the scala model to confirm correct operation of the hardware

All Modules Complete

- Created new files for each geometry and repeated until done
- All CHISEL operations were incrementally tested against the ground truth and scala models until bug free

FULL INTEGRATION

- Created one master CORDIC core
- Used the perfected logic from the disconnected models to create a unified version
- Squashed new bugs and implemented parameters as well as ready/valid

WORK TO BE DONE

DOCUMENTATION

- README needs to be updated and expanded for new capabilities
- Code needs to be refined for readability
- Comments need to be added to the code for future users

- **GREEN = COMPLETE**
- **RED = INCOMPLETE**

TESTING

- A more thorough test bench has to be written for the main CORDIC core
- Performance evaluations should be done and a closer look at exactly how each operation could be optimized
- A more streamlined test bench should be written as well

MORE OPERATIONS

- There are CORDIC operations that are not implemented yet that could add to the usefulness of the core
- Most operations are defined for small domains so the algorithms need to be mathematically extended

PIPELINING AND PARALLELISM

- Since the CORDIC algorithms are low area, high efficiency, implementing some sort of parameterizable parallelism would be idea for actual use
- Pipelining would also make the core much more useful since it has such a latency penalty

THINGS I LEARNED AND ADVICE

- I did not use much inheritance or functional programming but if I had to refactor my code they I definitely would
- A waveform viewer is your friend, I should have used one much more often
- Planning ahead is always better than just jumping straight in head first, I got burned a couple times by this
- A partner for the project would have made the process much more enjoyable

THANK YOU!, ENJOY THIS PHOTO OF TREES