Implementation of Hyperbolic CORDIC with AXI Full Interface

Project submitted by: Sagar Vaidya
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mail: sagarvaidya@oakland.edu

CORDIC

- CORDIC – COORDINATE ROTATION DIGITAL COMPUTER

- Basic CORDIC algorithms –
  - Circular CORDIC
  - Linear CORDIC
  - Hyperbolic CORDIC

- Hyperbolic CORDIC – Used to compute hyperbolic functions in efficient and fast way

Introduction

- Basic hyperbolic CORDIC algorithm is a fixed-point architecture with an expanded range of convergence and a scale-free fixed-point hardware

- It can be used in powering architecture generally that takes high cost of computation where FPGAs can be a solution if designed efficiently.

- This project mainly focuses on design of a hyperbolic CORDIC design and verification.

Methodology

- This is extension to the original CORDIC equations that allows for the computation of hyperbolic functions, where i is the index of the iteration (i = 1, 2, 3, …)
- The following iterations must be repeated to guarantee convergence: i = 4, 13, 40, … , k, $3k$ + 1
- Below are the equations for hyperbolic CORDIC –

$$x_{i+1} = x_i - \delta_i x_i 2^{-i}$$
$$y_{i+1} = y_i - \delta_i x_i 2^{-i}$$
$$z_{i+1} = z_i + \delta_i \theta_i, \theta_i = tanh^{-1}(2^{-i})$$

$$Rotation: \delta_i = +1 \; if \; z_i < 0; \; -1, otherwise$$
$$Vectoring: \delta_i = +1 \; if \; x_i y_i \geq 0; \; -1, otherwise$$

Methodology contnd.

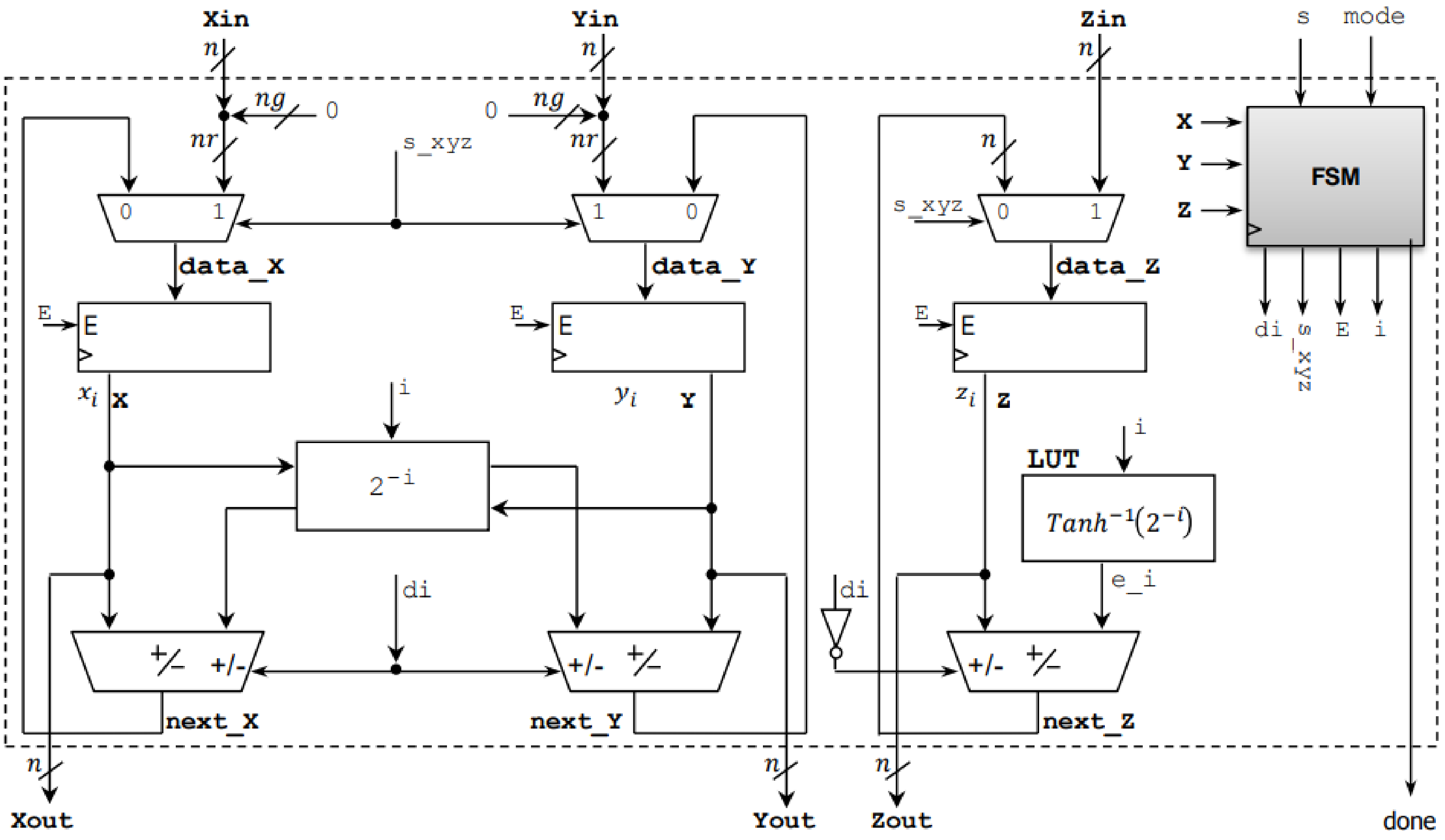
- Depending on the mode of operation, the quantities X, Y and Z converge to the following values, for sufficiently large N –

Rotation Mode	Vectoring Mode
$x_n = A_n(x_1 coshz_1 + y_1 sinhz_1)$ $y_n = A_n(y_1 coshz_1 + x_1 sinhz_1)$ $z_n = 0$	$x_n = A_n \sqrt{x_1^2 - y_1^2}$ $y_n = 0$ $z_n = z_1 + tanh^{-1}(y_1/x_1)$

$A_n \leftarrow \prod_{i=1}^{N} \sqrt{1 - 2^{-2i}}$ (this includes the repeated iterations $i = 4, 13, 40, \ldots$). For $N \rightarrow \propto$, $A_n \cong 0.8$

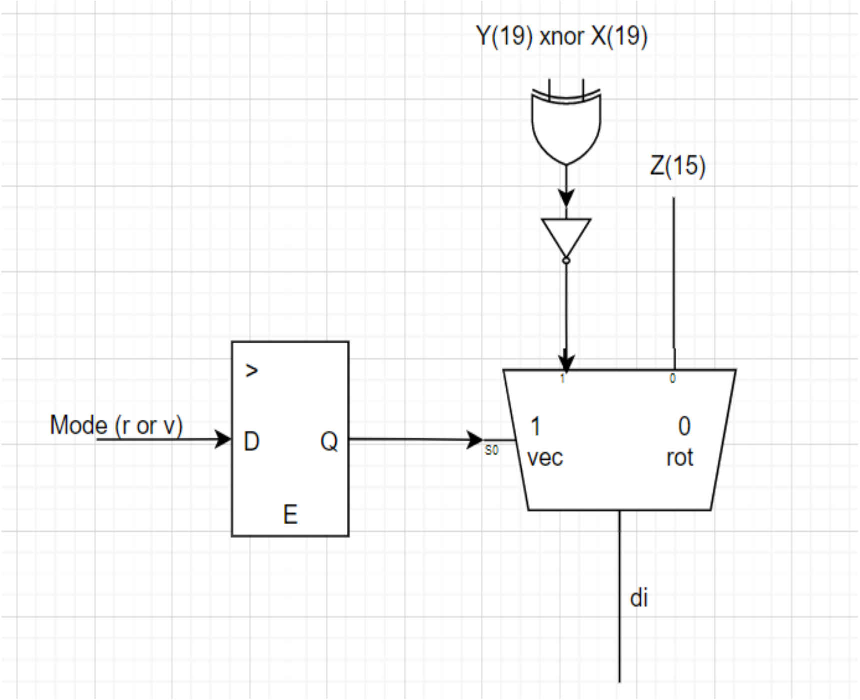
- With a proper choice of the initial values $x1$, $y1$, $z1$ and the operation mode, the above functions can be directly computed.

HARDWARE BLOCK DIAGRAM



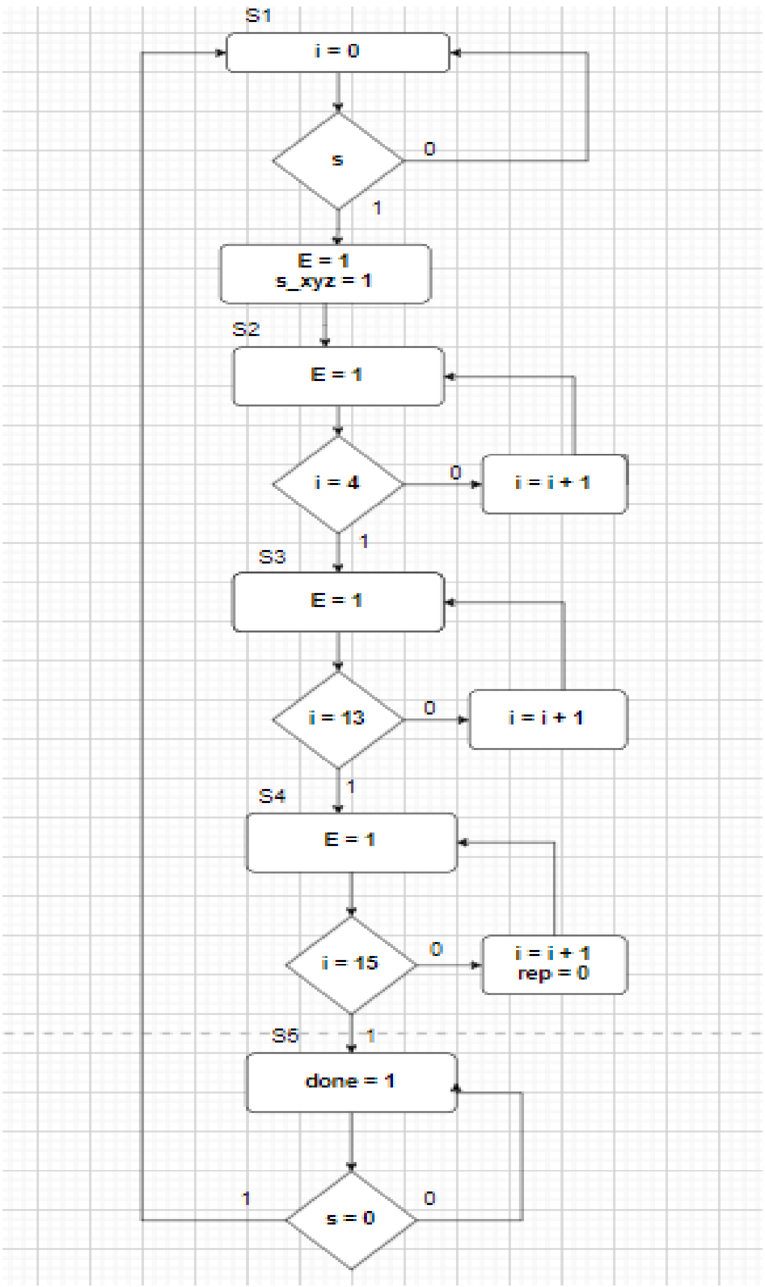
CONTROL DIAGRAM

- This control diagram depicts mode selection of the hyperbolic CORDIC based on assignment of input data in the word
- Vectoring and Rotating modes are selected based on MUX inputs



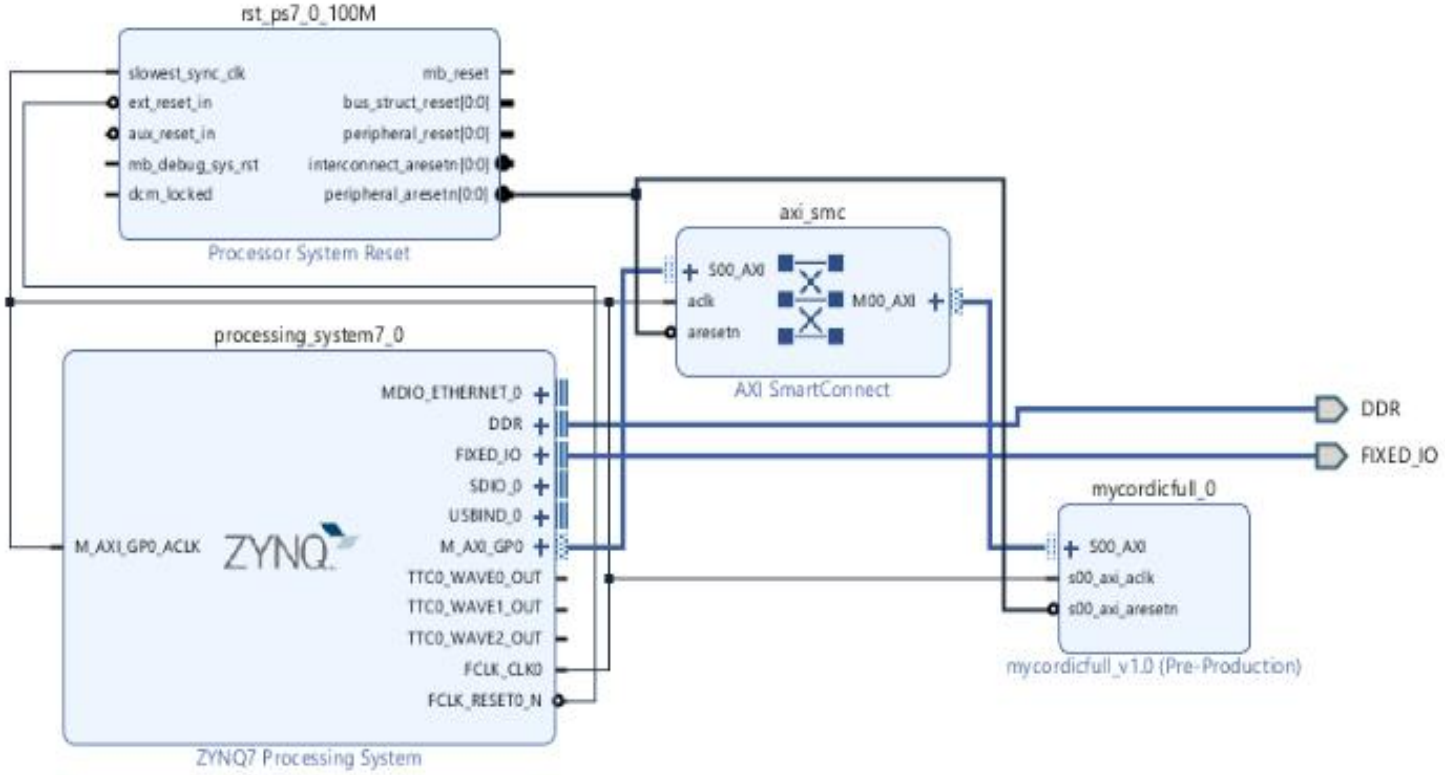
FSM

- FSM implements 16 iterations, though starting point is the 1st iteration
- Iterations 4 and 13 are repeated
- When iteration is #4 or #13, we go back to the same state
- When iteration is #15, we go back to the first state
- This can be extended to implemented n number of operations, in that case, it needs to be repeating for 4, 13, 40, …, k, 3k + 1



BLOCK DIAGRAM

This is the block design that is generated in Vivado project by importing AXI interface IP generated (mycordicfull) and Zybo board



SOFTWARE ROUTINE

Software routine is implemented in SDK to verify the functionality of the design on Zybo board

Steps involved are as below –

- Initialize base address based on AXI base address
- Write first set of input data using AXI interface write API defined in the design
- This includes writing 2 words sequentially
- Add a small delay to process the data
- Read 2 words using AXI interface read API defined in the design
- Repeat these steps for additional sets of inputs and outputs
- Build the SDK, connect the board and run on the hardware to see the output in serial console

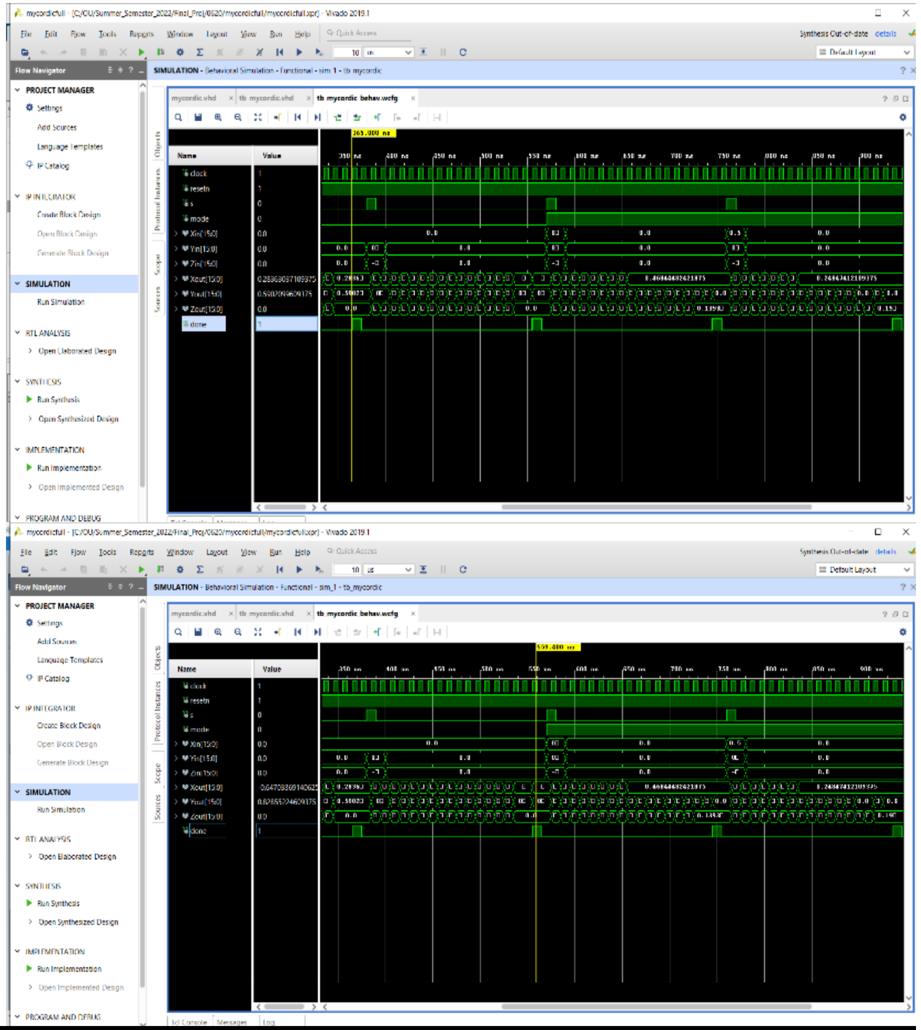
SIMULATION OF AXI FULL INTERFACE

- Generated AXI interface is then simulated in Vivado by adding the testbench for simulation that uses AXI read/write APIs on the input/output dataset

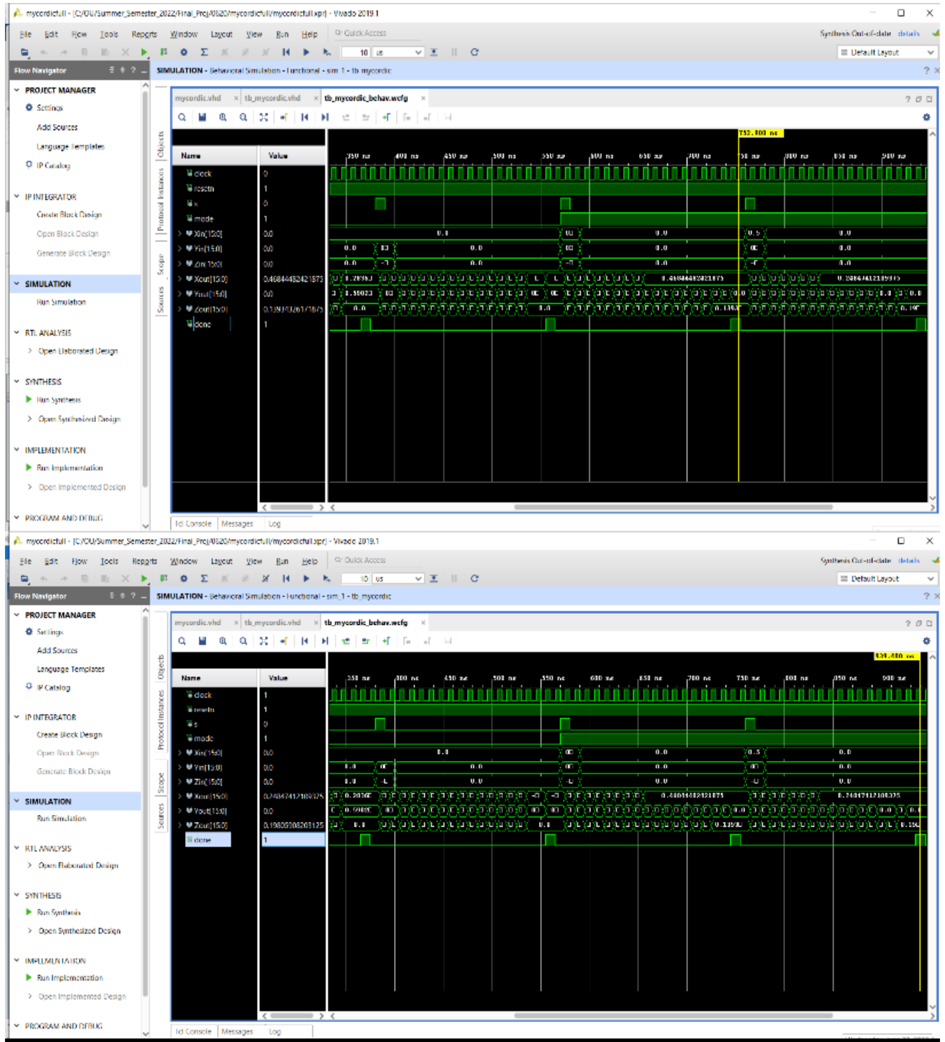
- Dataset

		Input Data			Output Data	
An=0.8	**x0**	**y0**	**z0**	**xN**	**yN**	**zN**
M=0	0	(1/2An)2800	(pi/6)2182	(0.273927)1188	(0.570119)247C	0
	0	(1/2An)2800	(-pi/3)BCFA	(-0.624684)D806	(0.800143)3335	0
M=1	(0.9)3999	(0.7)2CCC	(-0.9)(C667)	(0.452548)1CF6	0	(0.139721)08F1
	(0.5)2000	(0.4)1999	s	(0.24)0F5C	0	(0.198612)0CB6

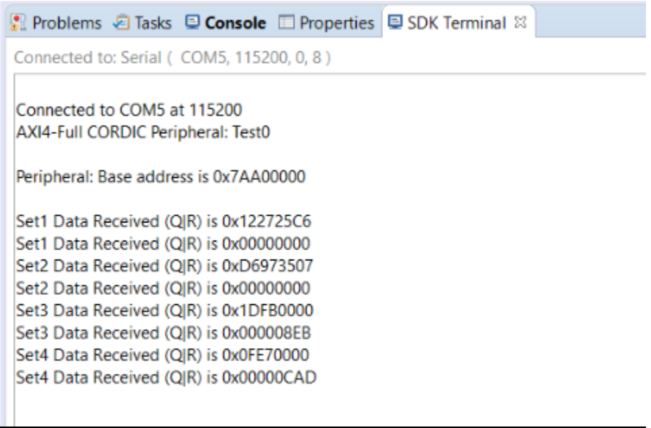
SIMULATION RESULTS



Rotating Mode



Vectoring Mode



SDK Program Output on HW

This verifies that the output of the hardware hyperbolic CORDIC design and the calculated output in the above dataset match with very little deviation

CONCLUSION

- Designing a hyperbolic CORDIC hardware software interface needs good mathematics and electronics knowledge

- These mathematical operators and its VHDL implementation is useful in different applications, such as computing (x^y) powering formulae, sinh, cosh, tanh of the given angles, logarithmic computations and many more

- Components implemented in VHDL are generic and can be adapted easily to any application listed but not limited to the above ones.

REFERENCES

[1] Digital Library - Arithmetic Cores (oakland.edu)

[2] https://moodle.oakland.edu/pluginfile.php/7667929/mod_resource/content/5/Notes%20-%20Unit%203.pdf

[3] Circular CORDIC implementation with AXI Full interface from Lab 3 assignment