Dual Fixed-Point CORDIC Processor: Architecture and FPGA Implementation

Andres Jacoby and Daniel Llamocca
Electrical and Computer Engineering Department
Oakland University
Rochester, MI, USA
ajacobykrateil@oakland.edu, llamocca@oakland.edu

Abstract

We introduce Dual Fixed Point CORDIC, that provides a compromise between Fixed Point and Floating Point CORDIC hardware implementations. A fully parameterized hardware is presented that allows for extensive exploration of the resources-accuracy design space, from which we generate optimal (in the multi-objective sense) realizations. We compare Fixed Point, Dual Fixed Point, and Floating Point CORDIC units in terms of resources and accuracy. Results show the effectiveness of Dual Fixed Point for CORDIC implementation where the increase in resources is largely offset by the high accuracy improvements.

Key Contributions

- **Parameterized architecture validated on a FPGA**
- **Design Space Exploration**
- **Comparisons among DFX, FX, and FP architectures**

Methodology and Architectures

We used the expanded CORDIC algorithms to implement the DFX Hyperbolic and Circular CORDICs. The expanded hyperbolic CORDIC is described mathematically by:

$$i \le 0: \begin{cases} x_{i+1} = x_i - \delta_i y_i\left(1-2^{i-2}\right) \\ y_{i+1} = y_i - \delta_i x_i\left(1-2^{i-2}\right) \\ z_{i+1} = z_i + \delta_i\theta_i, \theta_i = Tanh^{-1}\left(1-2^{i-2}\right) \end{cases}$$

$$i > 0: \begin{cases} x_{i+1} = x_i - \delta_i y_i 2^{-i}, \quad y_{i+1} = y_i - \delta_i x_i 2^{-i} \\ z_{i+1} = z_i + \delta_i\theta_i, \theta_i = Tanh^{-1}\left(2^{-i}\right) \end{cases}$$

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

$$Vectoring: \delta_i = +1 \; if \; x_i y_i \ge 0; \; -1, otherwise$$

$$Rotation: \begin{cases} x_n = A_n(x_{in} cosh z_{in} + y_{in} sinh z_{in}) \\ y_n = A_n(y_{in} cosh z_{in} + x_{in} sinh z_{in}) \\ z_n = 0 \end{cases}$$

$$Vectoring: \begin{cases} x_n = A_n\sqrt{x_{in}^2 - y_{in}^2}, \quad y_n = 0 \\ z_n = z_{in} + tanh^{-1}(y_{in}/x_{in}) \end{cases}$$

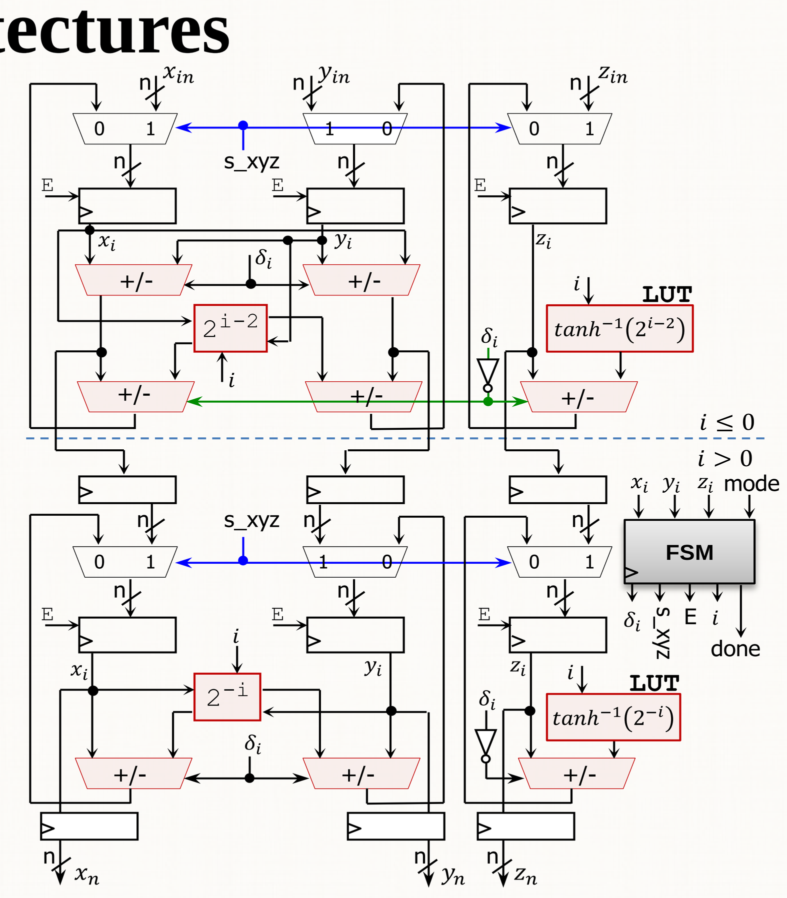

Figure 1 : Expanded DFX Hyperbolic CORDIC Architecture.

An n-bit Dual Fixed-Point (DFX) number is composed of a (n-1)-bit signed significand (X) and an exponent bit (E). The exponent determines the scaling for the significand:

$$D = \begin{cases} num0: \; X.2^{-p_0}, & if \; E = 0 \\ num1: \; X.2^{-p_1}, & if \; E = 1 \end{cases}, \quad p_0 > p_1$$

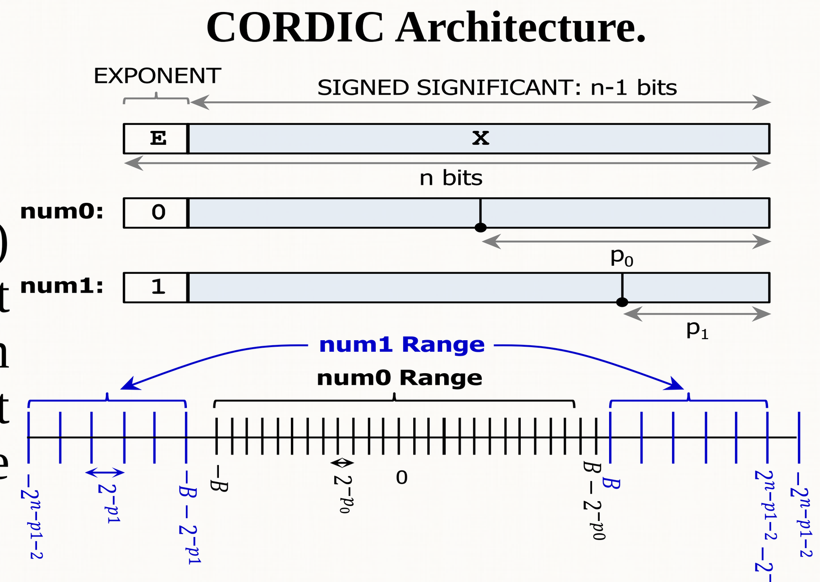

Figure 2 :DFX number and range of values.

DFX adder/subtractor architecture includes a pre-scaler, an FX adder/subtractor, and a post-scaler. The pre-scaler aligns the DFX input operands so they can be added in FX arithmetic. In the post-scaler, the range detector determines whether the result is a $num0$ or $num1$; we then select the proper result and set the exponent bit.

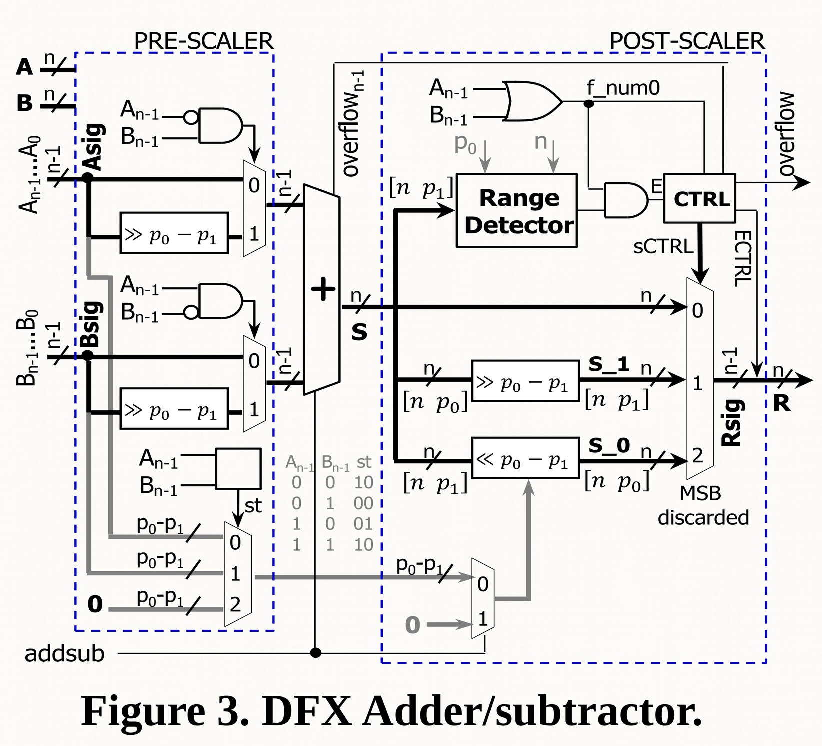

Figure 3. DFX Adder/subtractor.

The function $x^y = e^{y \ln x}$ is computed in two steps.

1. We first get $z_n = (\ln x)/2$, followed by $z_n \times 2y = y \ln x$.
2. Then, we use $x_{in} = y_{in} = 1/A_n$, $z_{in} = y \ln x$, $mode$=$rotation$ to get $x_n = e^{y \ln x} = x^y$.

The argument bounds of x^y ((x, y) values for which x^y converges) are given by $|y \ln x| \le \theta_{max}(M)$.

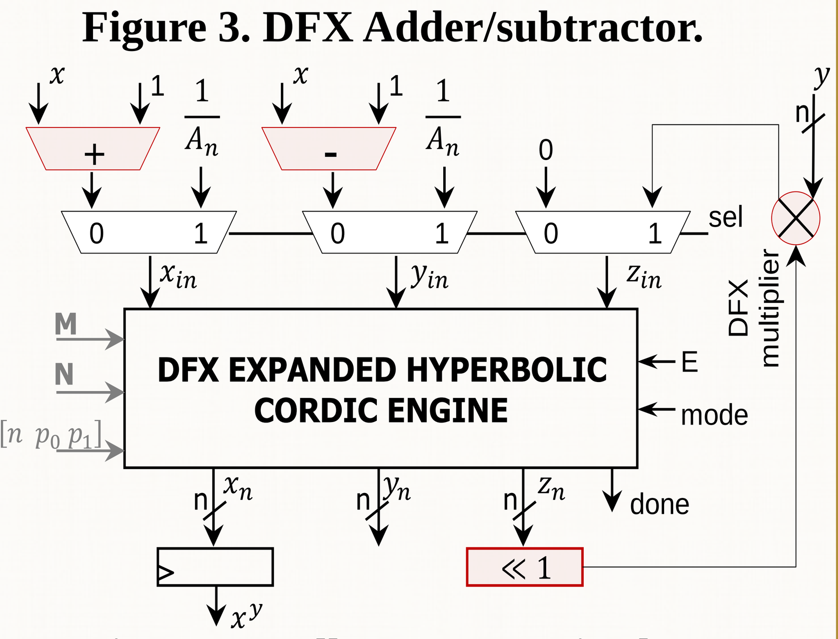

Figure 4. Fully parameterized DFX Powering.

Setup and Design Space Exploration

	DFX formats used					DFX formats for x^y
	sin, cos	atan	sinh, cosh	e^x, $\sqrt{x}$	atanh, ln	
[16 11 6]	✓				✓	[24 8 7]
[16 12 7]	✓				✓	[24 10 8]
[16 13 8]	✓				✓	[24 11 10]
[16 14 9]	✓				✓	[24 12 11]
[16 14 10]	✓				✓	[24 13 12]
[24 17 10]	✓	✓	✓	✓	✓	[32 11 10]
[24 18 11]	✓	✓	✓	✓	✓	[32 13 11]
[24 19 12]	✓	✓	✓	✓	✓	[32 14 13]
[24 20 13]	✓	✓	✓	✓	✓	[32 16 14]
[24 22 14]	✓	✓			✓	[32 18 16]
[24 22 16]	✓				✓	[48 17 14]
[32 22 13]	✓	✓	✓	✓	✓	[48 19 17]
[32 24 14]	✓	✓	✓	✓	✓	[48 22 19]
[32 26 16]	✓	✓	✓	✓	✓	[48 24 22]
[32 27 18]	✓	✓	✓	✓	✓	[48 26 24]
[32 29 19]	✓	✓	✓	✓	✓	[48 34 31]
[32 29 21]	✓	✓	✓	✓	✓	[64 22 19]
[48 34 19]	✓	✓	✓	✓	✓	[64 26 22]
[48 36 22]	✓	✓	✓	✓	✓	[64 29 26]
[48 38 24]	✓	✓	✓	✓	✓	[64 32 29]
[48 41 26]	✓	✓	✓	✓	✓	[64 35 32]
[48 43 29]	✓	✓	✓	✓	✓	[64 45 42]
[48 43 31]	✓	✓	✓	✓	✓	
[64 45 26]	✓	✓	✓	✓	✓	
[64 48 29]	✓	✓	✓	✓	✓	
[64 51 26]	✓	✓	✓	✓	✓	
[64 54 35]	✓	✓	✓	✓	✓	
[64 58 38]	✓	✓	✓	✓	✓	
[64 58 42]	✓	✓	✓	✓	✓	

Table 1 DFX Formats used in the experimental setup.

By varying n, p_0, p_1 (the DFX format), we create a design space of hardware configurations for every function to be tested. This also requires careful selection of the domain of the inputs. Some functions were only explored for a subset of the design space; this is due to intrinsic limitations such as convergence or CORDIC algorithm, scaling factor representation, input/output numerical representation. We completed 249 individual tests.

Function	Input domain for testing	CORDIC module	M
$\sin(x)$, $\cos(x)$	$-\pi \le x \le \pi$	Circular: Rotation. $z_{-M+1} = x$ $x_{-M+1} = 1/A_n, y_{-M+1} = 0$	2
$\text{atan}(x)$	$0 \le x \le 20$	Circular: Vectoring. $y_{-M+1} = x$ $x_{-M+1} = 1, z_{-M+1} = 0$	2
$\sinh(x)$, $\cosh(x)$	$0 \le x \le 4$	Hyperbolic: Rotation. $z_{-M} = x$, $x_{-M} = 1/A_n, y_{-M} = 0$	4
e^x	$-2 \le x \le 2$	Hyperbolic: Rotation. $z_{-M} = x$, $y_{-M} = x_{-M} = 1/A_n$	4
$\text{atanh}(x)$	$\lvert x \rvert \le 0.9995$	Hyperbolic: Vectoring. $y_{-M} = x$ $x_{-M} = 1, z_{-M} = 0$	5
$\sqrt{x}$	$0 \le x \le 36$	Hyperbolic: Vectoring. $z_{-M} = 0$, $x_{-M} = x + 1/(4A_n^2)$, $y_{-M} = x - 1/(4A_n^2)$	3
$\ln(x)$	$0.0005 \le x < 15$	Hyperbolic: Vectoring. $z_{-M} = 0$, $x_{-M} = x + 1$, $y_{-M} = x - 1$	5
x^y	$0.135 \le x \le 7.39$ $-2 \le y \le 2$	Hyperbolic: Vectoring and Rotation	4

Table 2 : Testing domain for the CORDIC-based functions.

For our accuracy metric we used:

$$MSE = \frac{\sum(HW\ value - ideal\ value)^2}{number\ of\ samples}$$

$$Relative\ Error = \left|\frac{HW\ value - ideal\ value}{ideal\ value}\right|$$

Fn.	FX		DFX	avg resource accuracy inc. (DFX/FX)	FP EW:24 FW: 16
x^y	[24 15]	[24 9]	[24 15 9]		
	343 115.78 dB	326 100.42dB	518 46.65 dB	55% 61.45 dB	769 7.61 dB
lnx	[24 10]	[24 20]	[24 20 10]		
	198 -34.70dB	200 28.49 dB	439 -104.61dB	120% 101.5 dB	718 -135.2dB
sinh	[24 15]	[24 10]	[24 15 10]		
	201 71.62 dB	197 -11.17 dB	399 -35.29 dB	100% 65.52 dB	605 -37.92dB

Table 3 : DFX vs FX. Resources and accuracy.

Results

The Pareto-optimal realizations for $atan/lnx$ allows us to only consider optimal hardware realizations while simultaneously satisfying accuracy and resource constraints. For lnx, if we want highest accuracy and fewer than 1k slices, we would pick DFX[48 43 29]. Table 3 depicts how DFX compares to FX in terms of resources and accuracy. For x^y on average, resources increased by 55% while accuracy improved 61.45dB. Fig 6 shows the relative error of DFX and two FX realizations each with the same $p0$ or $p1$. Table 4 lists resources an accuracy values of FP and DFX units for e^x, x^y. The resource increase and accuracy improvement of the FP units over the DFX units. For x^y, a 53% resource increase yields a 108.48dB gain in accuracy.

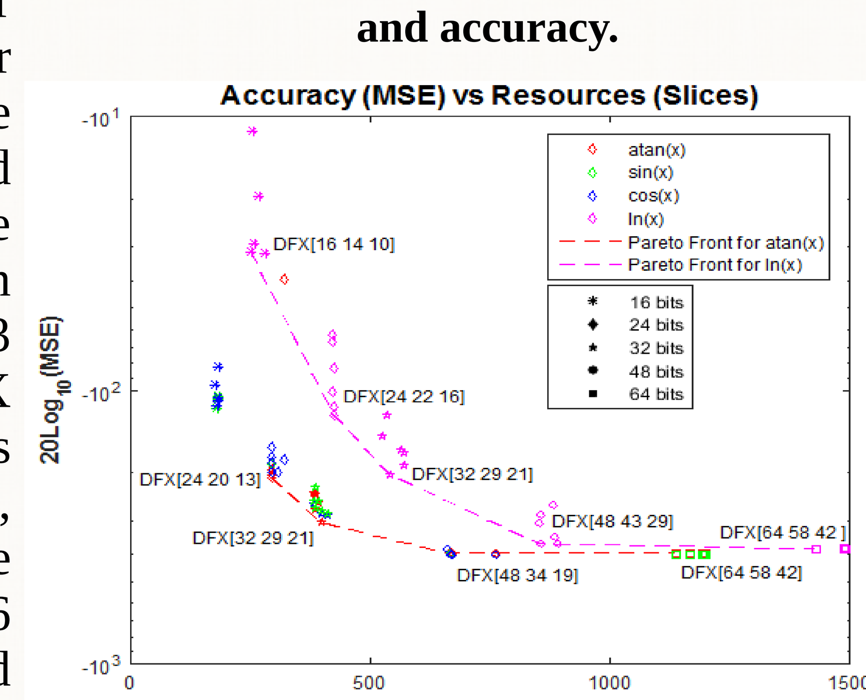

Figure 5 : Accuracy-Resources design space for atan/sin/cos/lnx.

Function	FP	DFX	Increase in resources and accuracy (FP/DFX)
x^y	EW: 24, FW: 16	[24 12 5]	
	776 / 76.12 dB	506 / 184.6dB	53% / 108.48dB
e^x	Single Precision	[32 27 12]	
	782 / 29.1 dB	559 / 71.85dB	40% /42.75dB

Table 4 : DFX vs FP. Resources and accuracy.

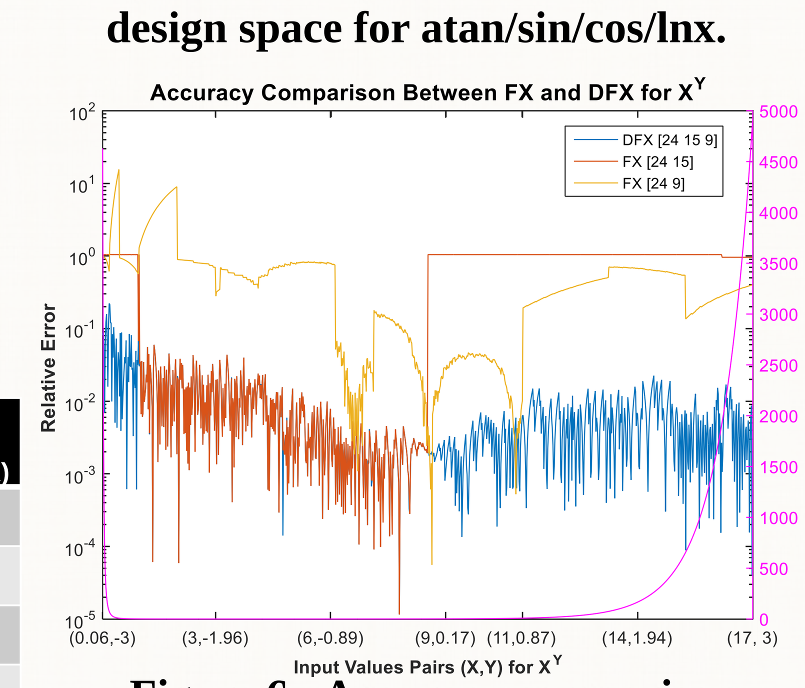

Figure 6 : Accuracy comparison between 24-bit FX and DFX.

Conclusion

We presented and validated fully customized DFX CORDIC and lnx,√x,x^y units. We extensively explored the accuracy-resources design space and extracted the Pareto front. Comparisons between DFX, FX, and FP CORDIC architectures demonstrate that DFX is an efficient alternative for CORDIC implementation: DFX accuracy improvements more than make up for the resource increase with respect to FX. Further work will focus on the implementation of scale-free CORDIC that requires fewer iterations for the same range of convergence, and by leveraging Partial Reconfiguration technology to implement Dynamic Dual Fixed Point for even larger dynamic ranges.