High Performance Architecture for Real-Time Cylinder Pressure Estimation

Joshua Mack, Sam Bellestri, and Nia Simmonds
Interdisciplinary Research Experience in Electrical and Computer Engineering
IREECE 2015
School of Engineering and Computer Science, Oakland University
Faculty Mentor: Dr. Daniel Llamocca | llamocca@oakland.edu |

Introduction

A common method to calculate x^y is a combination of the natural logarithm and exponential function, i.e.: $x^y = e^{ylnx}$. To implement x^y in hardware, we use the expanded hyperbolic CORDIC algorithm to compute e^x and $\ln x$, for which we implemented floating-point and fixed-point versions. Our architectures are fully customizable and allow us to explore a large space of hardware profiles. We can then determine the optimum bit-width and number of iterations for a given accuracy requirement. We use this efficient powering hardware in a real-time Hardware-in-the-loop application: Cylinder Pressure Estimation. Here, pressure is updated iteratively based on the instantaneous heat transfer coefficient (h), and our proposed hardware assists by computing h, whose equation is given by:

$$h = k * B^{-0.2} * p^{0.8} * {T_g}^{-0.53} * {v_c}^{0.8}, \text{ where}$$

$h: instantaneous\ heat\ transfer\ coefficient, k: constant,$
$B: cylinder\ bore\ diameter, p:\ cylinder\ pressure,$
$T_g: gas\ temperature, v_c: characteristic\ velocity.$

Methodology and Architectures

Hyperbolic CORDIC provides two modes of operation (rotation and vectoring) that allow for the direct computation of $\cosh x$, $\sinh x$, $\tanh^{-1} x$, e^x. Using mathematical properties, we can calculate $\ln x$ and thus x^y. The expanded hyperbolic CORDIC algorithm is given by:

- $For\ i \leq 0$:

$$X_{i+1} = X_i + \delta_i Y_i\left(1 - 2^{i-2}\right)$$
$$Y_{i+1} = Y_i + \delta_i X_i\left(1 - 2^{i-2}\right)$$
$$Z_{i+1} = Z_i - \delta_i tanh^{-1}\left(1 - 2^{i-2}\right)$$

- $For\ i > 0$:

$$X_{i+1} = X_i + \delta_i Y_i 2^{-i}$$
$$Y_{i+1} = Y_i + \delta_i X_i 2^{-i}$$
$$Z_{i+1} = Z_i - \delta_i tanh^{-1}(2^{-i})$$

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

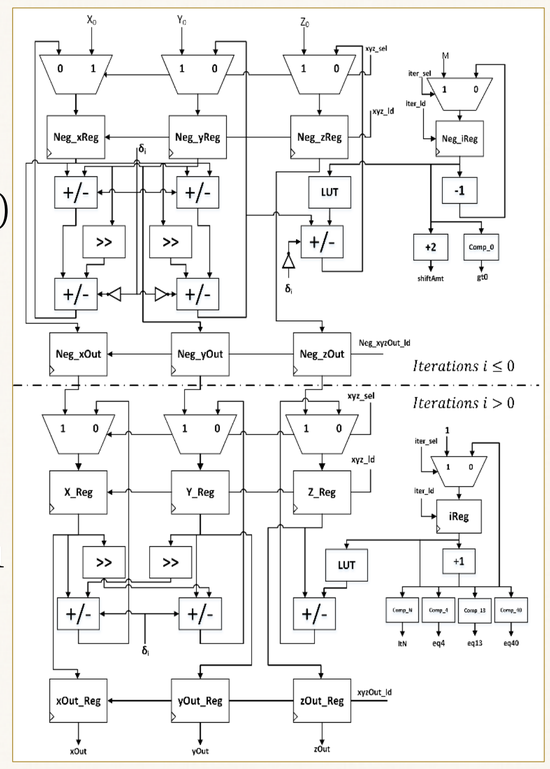

Figure 1 : Expanded Hyperbolic CORDIC Architecture.

Using CORDIC, we implemented a fully customizable x^y engine in VHDL. CORDIC has specific advantages in hardware due to its shift and add nature. The x^y architecture executes two consecutive operations.

1. Load $x_0 = x + 1, y_0 = x - 1, z_0 = 0$ onto the CORDIC engine in vectoring mode, so that $z_n = 0.5 \ln x$. A floating-point shifter generates $\ln x$ and a floating-point multiplier computes y $\ln x$ which is fed back to the CORDIC engine for the second operation.
2. Load $x_0 = y_0 = 1/A_n$ and $z_0 = y \ln x$ onto the CORDIC engine in the rotation mode so that $y_n = e^{y \ln x} = x^y$.

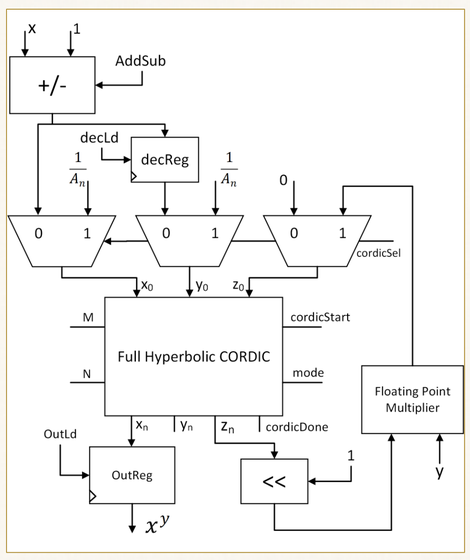

Figure 2 : Full Architecture of x^y Implementation.

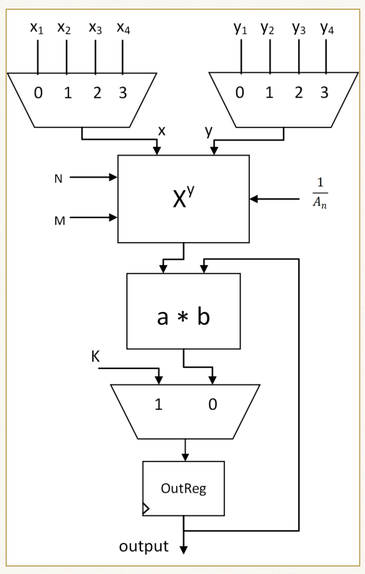

Figure 3 : Instantaneous Heat Transfer Coefficient Equation Architecture.

Design Space Exploration

The goal was computing the heat transfer coefficient in less than 10us at the start of the project. A complete design space exploration (generation of a large family of hardware profiles) was performed by varying parameters such as number representation and number of iterations. Since the most resource-consuming implementation of x^y was within our constraints of hardware resources and execution time, we present results from our floating point architecture (we are interested in the highest accuracy).

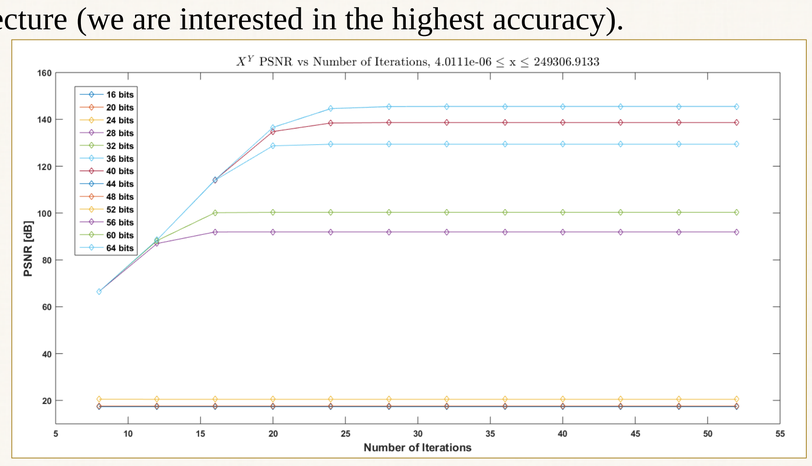

Figure 4 : x^y Architecture - Peak Signal-to-Noise Ratio (PSNR) vs. Number of iterations.

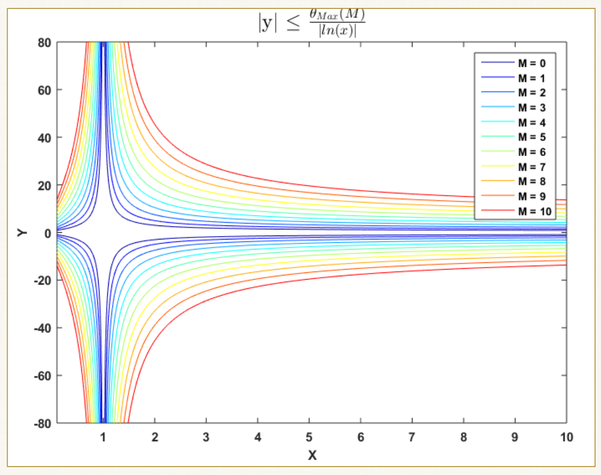

Figure 5: Range of Convergence Plot for x^y.

CORDIC does not converge for all values of x and y.

The parameter M represents the number of additional iterations in expanded CORDIC for $i \leq 0$.

If chosen values of x and y are between curves of the same color, x^y converges to the correct value.

Table 1: Execution Time (μs) x^y Architecture.

Function	Number of Iterations (N)						
	8	12	16	20	24	32	52
e^x	0.136	0.168	0.208	0.240	0.272	0.336	0.504
$\ln x$	0.144	0.176	0.216	0.248	0.280	0.344	0.512
x^y	0.288	0.352	0.432	0.496	0.560	0.688	1.024

Table 2: Execution Time (μs) Applied Instantaneous Heat Transfer Coefficient Equation Architecture

Function	Number of Iterations (N)						
	8	12	16	20	24	32	52
e^x	0.584	0.712	0.872	1.000	1.128	1.384	2.056
$\ln x$	0.616	0.744	0.904	1.032	1.160	1.416	2.088
x^y	1.192	1.448	1.768	2.024	2.280	2.792	4.136

The execution times in Tables 1 and 2 and the number of slices in Figure 6 were for a Xilinx® Zynq-7000 XC7Z010-1CLG400 SoC that runs at 125 MHz.

The Pareto front shows the optimal hardware profiles for our x^y architecture that is used for the heat transfer equation.

We note that 44 bits with 32 iterations provides the highest accuracy at the expense of a large resource usage. The case of 28 bits with 16 iterations provides the smallest hardware at the expense of lower accuracy.

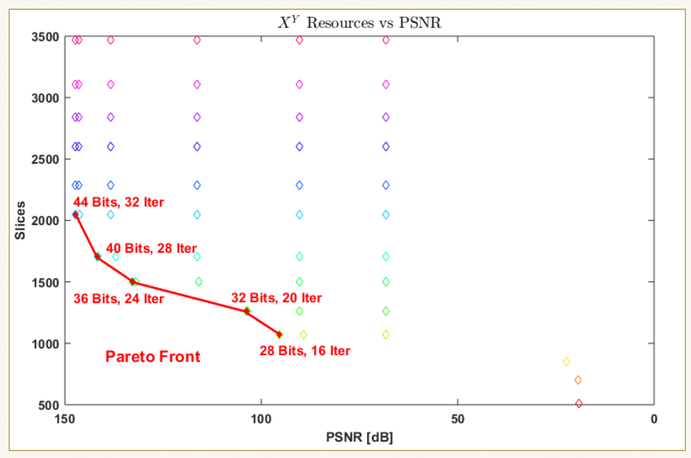

Figure 6: x^y Architecture Resources vs. PSNR with Pareto Front

Conclusion

The results of the heat transfer coefficient implementation prove that our x^y and expanded hyperbolic CORDIC architectures are suitable for a wide range of applications. For each application, the designer can select the optimal bit width, number of iterations, and number representation for a desired accuracy and resource usage. This scalability is useful in hardware-in-the-loop testing that is prevalent in the automotive industry.

This work was sponsored in part by the National Science Foundation (NSF) under award number EEC-1263133. We would like to acknowledge Dr. Osamah Rawashdeh for providing an application for powering: Cylinder Pressure Estimation.