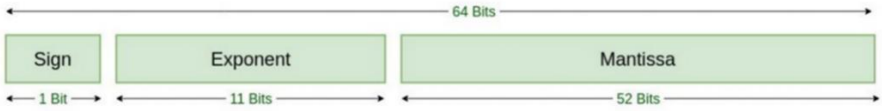
National Tsing Hua University
Department of Electrical Engineering
EE4292 IC Design Laboratory
Fall 2021

Double Precision Floating Point Arithmetic Logic Unit

Team Member:蔡忠浩(107061240)、陳麒任(107012013)

1. Introduction & Motivation

In the present technology, precision plays an important role in many applications like digital signal processing. As a result, floating-point is used to measure a more precise quantity value. In this project, the operations are performed on double precision (64-bit) floating point numbers.



Through this project, we can further understand how ordinary operations are implemented in hardware. Besides, we can also learn more efficient algorithms to approximate special functions, such as sine/cosine, square root, and natural logarithm. And analyze errors between the simulation results and actual answers.

2. CORDIC algorithm

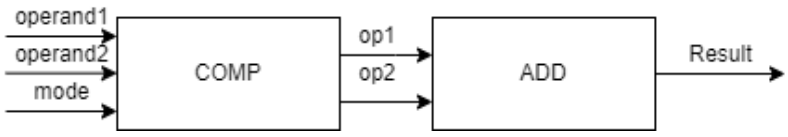
CORDIC (Coordinate Rotation Digital Computer) is a simple and efficient algorithm to calculate trigonometric functions, square roots, logarithms, and so on. CORDIC is commonly used when no hardware multiplier is available, as the only operations it requires are additions, subtractions, bit-shift, and lookup tables. As a result, they all belong to the class of shift-and-add algorithms.

3. Booth algorithm

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. With this algorithm, the multiplication can actually be replaced by the string of ones in the original number by simpler operations, adding the multiplier, shifting the partial product thus formed by appropriate places, and then finally subtracting the multiplier.

4. Architecture

A. Addition/ Subtraction：



In the COMP block, the main goal is to adjust the operand1 and operand2 to the same exponent, which allows the computer to calculate the result correctly in ADD block.

In the ADD block, add the op1 and op2 derived from the COMP block together, check

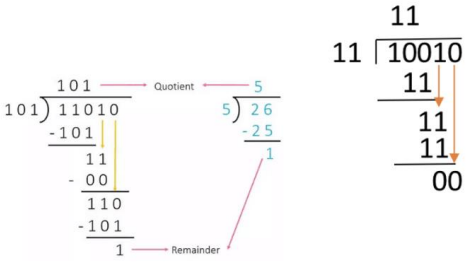
whether there is a need to carry, and obtain the final result.

B. Multiplication：

With the booth algorithm, we can complete floating point multiplier by multiplying two mantissa bits, xor sign bit, and adding exponent bits from operands.

C. Division：

In this part, we use the concept of long division (shift & subtract) to handle the division of mantissa bits. Otherwise, it is similar to Multiplication. We can complete floating point divider by dividing two mantissa bits, xor sign bit, and subtracting exponent bits from operands.



D. Sine/ Cosine：

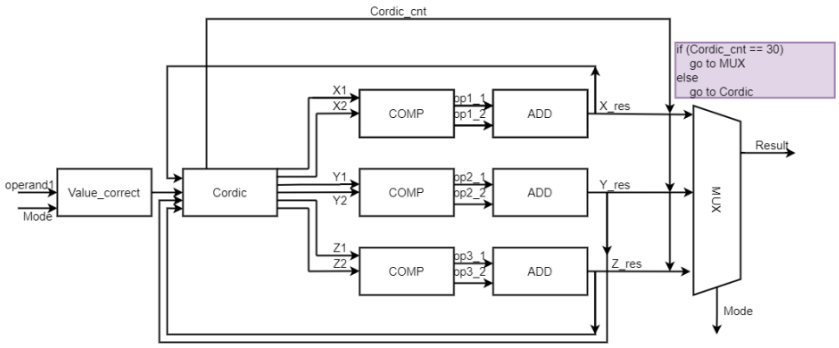
From the CORDIC algorithm, derive the answer using the following equations：

$$\begin{cases} X_{n+1} = X_n - S_n 2^{-n} Y_n \\ Y_{n+1} = Y_n + S_n 2^{-n} X_n \\ Z_{n+1} = Z_n - S_n \theta_n \end{cases}$$

Where $S_n = \begin{cases} -1 \; if \; Z_n < 0 \\ +1 \; if \; Z_n \geq 0 \end{cases}$, $\theta_n = arctan\frac{1}{2^n}$

(There will be a look-up table for $acrtan\frac{1}{2^n}$ in the circuit)

According to the above equations, construct the following block diagram：



The block diagram is composed of three parallel addition blocks, the CORDIC block, and the Value_correct block. The Value_correct block is to control the value of *operand1* within 0° to 90°. In addition, the CORDIC block is to determine the S_n of the next iteration. After 30 times iterations, we can use *Mode* to choose the desired answer.

With $X_0 = 1$, $Y_0 = 1$, and $Z_0 = operand1(\ -90^{\circ} \sim +90^{\circ})$, this mode provides the following result as N approaches $+\infty$. (In this project, we let N equal to 30)

$\boldsymbol{X_N \approx \cos(operand1)/K}$

$\boldsymbol{Y_N \approx \sin(operand1)/K}$

$Z_N \approx 0$

where $K = \prod_{n=0}^{N} \cos(\arctan(2^{-n})) \approx 0.607253$

E. Square root：

From the CORDIC algorithm, derive the answer using the following equations：

$$\begin{cases} X_{n+1} = X_n + S_n 2^{-n} Y_n \\ Y_{n+1} = Y_n + S_n 2^{-n} X_n \end{cases} \qquad \text{Where } S_n = \begin{cases} -1 \; if \; Y_n \geq 0 \\ +1 \; if \; Y_n < 0 \end{cases}$$

According to the above equations, we can derive the result without a look-up table in the square root function. So, there is only two parallel ADD & COMP block in the above block diagram. Besides, since the square root function using the CORDIC algorithm is a hyperbolic CORDIC-based algorithm, certain iteration ($i = 4,13,40, \dots, 3k+1, \dots$) are repeated to achieve result convergence.

Note that the acceptable range of input is $0.5 \leq v < 2$, so we should first process the input using the Value_correct block.

$operand1 = v * 2^n$, for some $0.5 \leq v < 2$ and some even integer *n*

$\sqrt{operand1} = \sqrt{v} * 2^{n/2}$

With $X_0 = v + 0.25$ and $Y_0 = v - 0.25$, this mode provides the following result as N approaches $+\infty$. (In this project, we let N equal to 30)

$$\boldsymbol{X_N \approx A_N \sqrt{(v+0.25)^2 - (v-0.25)^2} \approx A_N \sqrt{v}}$$

$Y_N \approx 0$ where $A_N = \prod_{n=1}^{N} \sqrt{1 - 2^{-2i}} \approx 0.8298$

We can get the final answer by multiplying X_N by $2^{n/2}$ derived from the Value_correct block.

F. Natural logarithm：

From the CORDIC algorithm, derive the answer using the following equations：

$$For \; n \leq 0 \begin{cases} X_{n+1} = X_n + S_n(1 - 2^{n-2}) Y_n \\ Y_{n+1} = Y_n + S_n(1 - 2^{n-2}) X_n \\ Z_{n+1} = Z_n - S_n Tanh^{-1}(1 - 2^{n-2}) \end{cases} \qquad \text{Where } S_n = \begin{cases} -1 \; if \; X_n Y_n \geq 0 \\ +1 \; if \; X_n Y_n < 0 \end{cases}$$

$$For \; n > 0 \begin{cases} X_{n+1} = X_n + S_n 2^{-n} Y_n \\ Y_{n+1} = Y_n + S_n 2^{-n} X_n \\ Z_{n+1} = Z_n - S_n Tanh^{-1}(2^{-n}) \end{cases}$$ (There will be a look-up table for $Tanh^{-1}(1 - 2^{n-2})$ and $Tanh^{-1}(2^{-n})$)

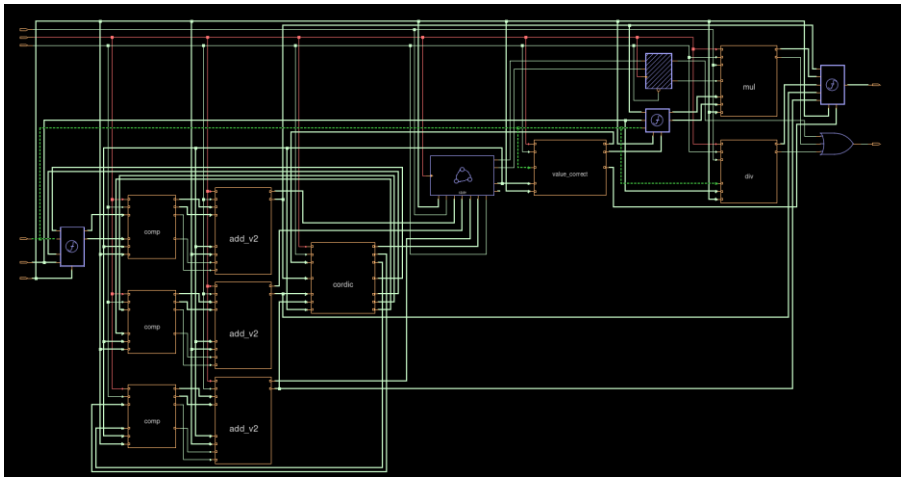
Since the range of input (α) is limited by the $\left| Tanh^{-1}\left(\frac{\alpha-1}{\alpha+1}\right) \right| \leq 1.1182$ when $n > 0$. To satisfy normal applications of $\ln(\alpha)$. Hu [2] has modified the basic hyperbolic CORDIC algorithm by including additional iterations $(M+1)$ for negative indexes $i: (i = 0, -1, \dots - M)$. Using Hu's [2] method we can increase the range of input from $0.106843 \leq \alpha \leq 9.35947$ to $1.16e^{-11} \leq \alpha \leq 6.21e^{10}$ by setting M = 5.

With $X_0 = \alpha + 1$, $Y_0 = \alpha - 1$, and $Z_0 = 0$, this mode provides the following result as N approaches $+\infty$. (In this project, we let N equal to 18)

$X_N \approx A_N \sqrt{{X_0}^2 - {Y_0}^2}$ $Y_N \approx 0$ Where $A_n \approx \prod_{i=-M}^{0} \sqrt{1 - (1 - 2^{i-2})^2} \prod_{i=1}^{N} \sqrt{1 - 2^{-2i}}$

$$\boldsymbol{Z_N = \frac{ln(\alpha)}{2}}$$

5. Overall block diagram



6. Verification tools

For addition, subtraction, multiplication, and division, use the website http://weitz.de/ieee/ to verify the result.

For sine, cosine, square root, and natural logarithm, since the CORDIC algorithm uses an approximation method, there will be errors between simulation results and actual answers. In addition to calculating errors between them, MatLab code is used to verify whether the result is in line with expectations.

7. Simulation result

For all functions, there are ten test set of input data and their corresponding answers generated by verification tools to verify the correctness of implementations of functions.

For addition, subtraction, multiplication, and division (mode 0 ~ 3), there is no error between simulation results and golden answers.

In contrast, for sine/cosine, square root, natural logarithm, there are errors between simulation results and golden answers. The following table represents the errors.

sine/cosine	simulation	matlab code	golden answer	error(%)
sin(90)/cos(0)	1.00000E+00	1.00000E+00	1.00000E+00	1.07025E-05
sin(75)/cos(15)	9.65926E-01	9.65926E-01	9.65926E-01	1.07488E-05
sin(60)/cos(30)	8.66025E-01	8.66025E-01	8.66025E-01	1.07324E-05
sin(45)/cos(45)	7.07107E-01	7.07107E-01	7.07107E-01	1.06656E-05
sin(30)/cos(60)	5.00000E-01	5.00000E-01	5.00000E-01	1.06126E-05
sin(15)/cos(75)	2.58819E-01	2.58819E-01	2.58819E-01	1.00571E-05
sin(0)/cos(90)	1.52656E-09	1.52656E-09	0.00000E+00	#DIV/0!
square root	sim	matlab code	golden	error(%)
0.5	7.05724E-01	7.05724E-01	7.07107E-01	1.95504E-01
20.2	4.48565E+00	4.48565E+00	4.49444E+00	1.95504E-01
580.69	2.40504E+01	2.40504E+01	2.40975E+01	1.95504E-01
0.00125	3.52862E-02	3.52862E-02	3.53553E-02	1.95504E-01
789456	8.86776E+02	8.86776E+02	8.88513E+02	1.95504E-01
8000.5689	8.92710E+01	8.92710E+01	8.94459E+01	1.95505E-01
natural log	sim	matlab code	golden	error(%)
3	1.09861E+00	1.09861E+00	1.09861E+00	1.69338E-04
58741.235	1.09809E+01	1.09809E+01	1.09809E+01	6.27763E-05
0.5895	-5.28481E-01	-5.28481E-01	-5.28481E-01	1.92728E-05
123.5	4.81624E+00	4.81624E+00	4.81624E+00	4.01410E-05
6.20E+10	2.47275E+01	2.47275E+01	2.48504E+01	4.94705E-01
1.16E-11	-2.47275E+01	-2.47275E+01	-2.51800E+01	1.79727E+00

8. Specification

Function	CYCLE Count	Acceptable Range
Addition	5	$-\infty\sim\infty$
Subtraction	5	$-\infty\sim\infty$
Multiplication	110	$-\infty\sim\infty$
Division	114	$-\infty\sim\infty$
Sine/ Cosine	181	$-90^{\circ}\sim+90^{\circ}$
Square root	293	$0\sim\infty$
Natural logarithm	205	$1.16e^{-11}\sim6.21e^{10}$

	Goal we set	Synthesis	After APR
Area(um^2)	200000	65146	76742
Power(mW)	10	1.78	1.97
Timing(ns)	10	9.1	10

	Primetime(post-sim)	core area(ICC)	die area(ICC)
Total power(W)	1.69E-03	57542.27794	68045.27514
Cell leakage power(W)	8.29E-04	cell area(DC)	core utilization rate
Total dynamic power(W)	4.57E-04	52135	0.906029477

(P.S :floorplan utilization ≥ 90% and P&R timing ≤ 1.1*synthesis timing =>achieve bonus goal)

9. Contribution

107061240 蔡忠浩：addition/ subtraction, sine/ cosine, square root, natural logarithm, presentation ppt (interim, final)

107012013 陳麒任：multiplication, division, APR, post-sim, power analyze, presentation (interim, final)

(functions include RTL codes, testbench, contents of the final report, and all RTL codes are designed by ourselves with the paper reference.)

10. Reference

[1] Nisha Singh, and R Dhanabal. "Design of Single Precision F

[2] loating Point Arithmetic Logic Unit." IEEE. 2018.

[3] X. Hu, R. Huber, S. Bass, "Expanding the Range of Convergence of the CORDIC Algorithm", IEEE.1991.

[4] Daniel Llamocca, and Carla Agurto. "A fixed-point implementation of the natural logarithm based on an expended hyperbolic CORDIC algorithm." XII Workshop IBERCHIP, 2006.

[5] Puli Anli Kumar. "FPGA Implementation of the Trigonometric Functions Using the CORDIC Algorithm." IEEE. 2019.

[6] Barun Biswas, and Bidyut Baran Chaudhuri. "Generalization of Booth's Algorithm for Efficient Multiplication." Procedia Technology. 2013.