Hardware Implementation of High Precision Floating Point Transcend Function

Jun Zhang [a], Mingjiang Wang [b] and Nanxin Hou [c]
School of Harbin Institute of Technology, Shenzhen 518000, China.
[a]zhjun612@126.com, [b]mjwang@hit.edu.cn, [c]1059369773@qq.com

Abstract. In this paper, the combination of CORDIC algorithm and mathematical method is used to complete the implementation of 128-bit floating-point exponential function and logarithmic function. The implementation of exponential functions and logarithmic functions is involved in speech recognition, image processing, and radar detection, but the computational domain of [-6, 6] is completed in current hardware implementations. It is too narrow in practice. Here we use the CORDIC algorithm combined with the mathematical method to expand. In this design, the calculation domain of the logarithm function is 128 bits of all floating-point numbers and the calculation domain of the exponential function is [-8192, 8192]. The practicality of hardware implementation transcendental functions is improved by extending the computational domain. In the overall hardware architecture, based on the Xilinx ISE14.7 development environment, the Virtex7 development board, the combined maximum frequency can reach 353.189MHz.

Keywords: Floating point, transcendental function, CORDIC algorithm, pipeline architecture.

1. Introduction

In This topic comes from using the CORDIC algorithm to implement various transcendental functions. In this design, the realization of the four-precision floating-point exponential function and the logarithmic function are mainly realized. Compared with the existing single-precision floating-point exponential function implementation, the use of four-precision floating-point input can improve the accuracy of the calculated data.

In 1959, JD volder proposed the CORDIC algorithm to calculate the exponential function and the logarithm function, the calculation domain is [-1.1182, 1.1182]. The exponential function has a convergence domain extended to [-6,6] [1]. In 2014 Ateeq Ur Rahman Shark, Rakesh Gangarajaiah, Erik Hertz using the series convergence to calculate exponential function, but this set of architecture can only be used to calculate the exponential function so that the reusability is low [3]. In this design we use the same set of architecture to complete the hardware construction of the exponential function and the logarithmic function.

2. The Basic Hardware Implementation

2.1 Overall Hardware Framework

2.1.1Implementation of the Exponential Function

The biggest difficulty in implementing an exponential function is to store the corresponding exponential results. The result implemented in the FPGA is fixed-point storage. However, the result of the exponential function calculation is very large. It is easy to overflow if the intermediate result is directly stored using the fixed point. This can result in a narrow range of calculations in hardware implementations. For this reason, In this design we directly use the mathematical method to get the mantissa part and the exponent part of the exponential function result. The corresponding exponential part and mantissa part are implemented by hardware respectively, which solves the shortcoming of fixed-point storage overflow. The specific derivation is as follows:

$$e^{M\times 2^E} = X\times 2^Y \rightarrow 1 \le X = \frac{e^{M\times 2^E}}{2^Y} < 2$$

$$2^Y \le e^{M\times 2E} < 2^{Y+1} \rightarrow Y \le \frac{M\times 2^E}{\ln 2} < Y+1$$

$$\rightarrow Y = \left[\frac{M\times 2^E}{\ln 2}\right], X = e^{M\times 2^E - Y\ln 2}$$

In this way, Y can be obtained by the divider and shift and the final result E can be obtained by adding the offset b. X is obtained by the rotation mode in the hyperbolic model of the CORDIC algorithm. Remove the highest bit of X to get the final result M.

2.1.2 Logarithmic Function Implementation

The logarithmic function uses $\ln M\times 2^{E-b}$ =lnM+(E-b) ln2, lnM is implemented by the vector mode in the CORDIC algorithm hyperbolic model and Eln2 is implemented by multiplier design. The result of the two modules is then converted to a standard floating-point number and sent to the floating-point adder (using the sign bit to determine whether it is an addition or subtraction operation) to get the final result.

The specific framework is shown below:

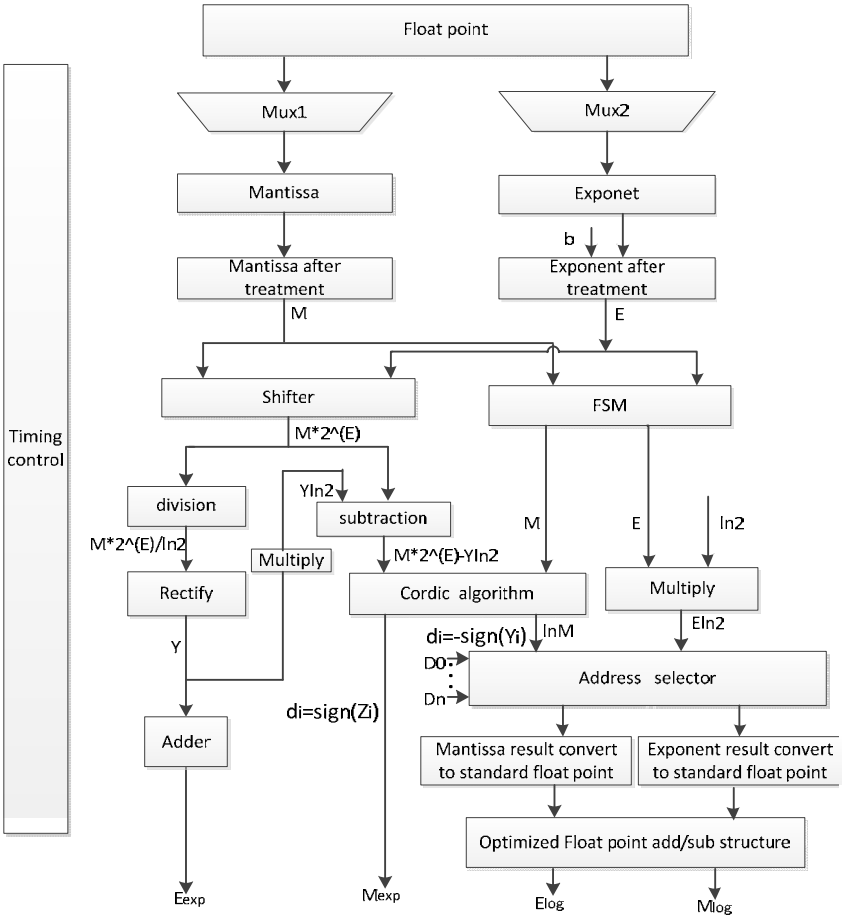

Figure 1. Overall hardware framework.

2.2 Floating Point

The representation of floating-point numbers is $N=M\times 2^{E-b}$, M is called the mantissa, R is the base number, and E-b is the exponent. In the early days of computers, companies used different representations to represent floating-point numbers. This led to poor portability of the corresponding

floating-point arithmetic program. For this reason, IEEE International Standards Organization was in 1985. Standards for floating point and floating-point arithmetic were developed and revised in 2001 and 2008. The basic floating-point data formats specified in the IEEE754-2008 standard are single-precision, double-precision, extended double-precision, and quad-precision. In this design, we use a four-precision floating-point number (128-bit).

2.3 CORDIC Algorithm

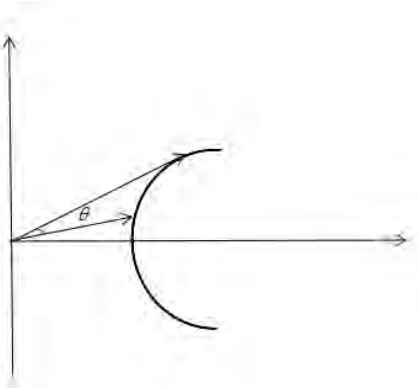

Figure 2. Hyperbolic model of CORDIC algorithm.

The mathematical relationship used in the hyperbolic model is:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cosh\theta & \sinh\theta \\ \sinh\theta & \cosh\theta \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} = \frac{1}{\sqrt{1+\tan^2\theta}} \begin{bmatrix} 1 & \tanh\theta \\ \tanh\theta & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

Through the definition of hyperbolic function , when $X_0=1, Y_0=0$,We can get $e^{\theta} = X_{n+1} + Y_{n+1}$ .But this time will involve a series of multiplication and tangent function calculations, it is very troublesome to implement with hardware. In order to facilitate the hardware implementation, we will use the method of decomposing the angle value to perform clockwise or inverse at a fixed angle $\theta_i = \arctan h2^{-i}$. The hour hand rotates (by judging the direction of rotation) to keep approaching. The relationship corresponding to each rotation is:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \frac{1}{\sqrt{1-2^{-2i}}} \begin{bmatrix} 1 & d_n 2^{-n} \\ d_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \left(Z_{n+1} = Z_n - d_n\theta_n\right) \begin{cases} Z_{n+1} > 0, d_n = -1 \text{（clockwise rotation）} \\ Z_{n+1} < 0, d_n = 1 \text{（Anticlockwise rotation）} \end{cases}$$

Let K = $\prod_{N=0}^{n} \frac{1}{\sqrt{1-2^{-2i}}}$, so that after the last n iterations, simply multiply K by the result to ensure the same result as originally requested. To ensure that the K value converges, iterative iterations are performed at i=4, 13, 40, 4K+1..., so that the K value converges to 1.20750 when n>=13. In the implementation of the high-precision exponential function, the number of iterations is 67 (64 iterations plus repeated iterations at i=4, 13, 40), then the K value is a constant, after n iterations you will get:

$$\begin{cases} X_{n+1} = K^{-1}(X_0\cosh\theta + Y_0\sinh\theta) \\ Y_{n+1} = K^{-1}(Y_0\cosh\theta + X_0\sinh\theta) \\ Z_{n+1} = 0 \end{cases}$$

The calculation result of the hyperbolic sine and cosine functions can be obtained by reasonably assigning the initial value. At the same time, in order to realize more calculations of transcendental functions, we introduce a vector model:

$$\begin{cases} X_{n+1} = K\sqrt{X^2 + Y^2} \\ Y_{n+1} = 0 \\ Z_{n+1} = Z_0 + \arctan h\theta \end{cases}$$

which is introduced in order to realize the exponential, logarithmic and hyperbolic functions simultaneously with the same set of structures after n times of generation. `The general relationship of each iteration obtained by these definitions is:

$$\begin{cases} x_{i+1} = x_i + \sigma_i 2^{-i} y_i \\ y_{i+1} = y_i + \sigma_i 2^{-i} x_i \\ z_{i+1} = z_i - \sigma_i \theta_i \end{cases} \quad \sigma_i = \begin{cases} sign(z_i) rotation \bmod el \\ -sign(y_i) vector \bmod el \end{cases}$$

In order to speed up the calculation run, the CORDIC iterative algorithm is implemented using a pipeline structure. Each level of the circuit structure mainly includes two shifters and three adder (subtractor), which are directly connected between the stages and do not require additional registers. The corresponding structure diagram is as follows:

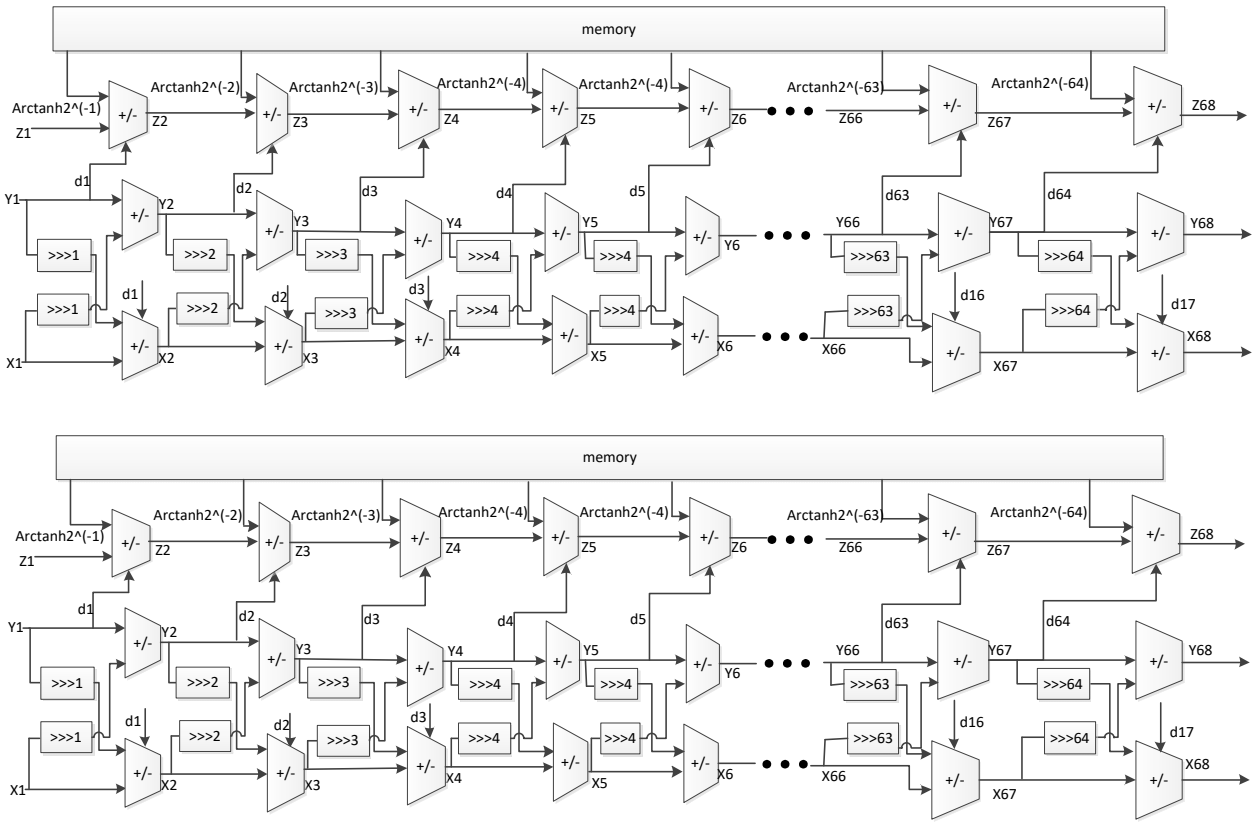

Figure 3. CORDIC algorithm pipeline architecture (rotation model and vector model).

3. Simulation Results

Because the graphic display is limited, we select a representative value for logarithmic calculation and convert it to a decimal number to compare with the value calculated by the calculator.

F	C	Calculator calculation result
3.4761658651976052035388621247844*10^{37}	178.54498208406676185412042829919	178.54498208406676185428261933052
2.5286213820173450211556965602732*10^{2543}	5856.4015657297858537517715676188	5856.4015657297875852653085872435
2.64224026132222578606141967127 43*10^{3776}	8695.5329382866604477836313651349	8695.5329382866651995801211242074
6.8721436229630888947490111346875*10^{-1773}	-4080.55589471180845776836999004661	-4080.5558937942467435983013385097
3.0444802651351243086172649525 37*10^{-4855}	-11177.986139051924546116156156 82419	-11177.937296284182912399329078302

Figure 4. Partial logarithmic results display and error analysis.

```
Minimum period: 2.831ns (Maximum Frequency: 353.189MHz)
Minimum input arrival time before clock: 17.858ns
Maximum output required time after clock: 33.081ns
Maximum combinational path delay: 2.303ns
```

Figure 5. Comprehensive results based on Vitex7.

4. Conclusion

Through the above simulation results and the data shown, we can see that whether X is a large value or a small value, we can complete the logarithmic function. Can guarantee the corresponding data behind the decimal point. We can also find that the hardware implementation results are larger than the calculator calculation results when performing data operations. The main reasons here are: 1. The corresponding calculator can only input 32-bit data, when corresponding to a large value, directly rounds off the following values in decimal. Corresponding to the result calculated by the calculator, there will be data loss. 2.floating point numbers in the representation of real numbers, because the use of binary representation and the exact real value will have some errors.

References

[1]. Hu X B, Harber R G, Bass S C. Expanding the range of convergence of the CORDIC algorithm[J]. IEEE Transactions on computers.

[2]. Shi Xiongwei, Wang Cheng, Zhang Chunlei, Chen Naikui. Research and Implementation of Floating-Point Exponential Function Algorithm Based on FPGA[J]. Computer Measurement & Control,2017,25(10):221-223+231.

[3]. Ateeq Ur Rahman Shark, Rakesh Gangarajaiah, Erik Hertz, Hardware Implementation of the Exponential Function Using Taylor Series. IEEE, 2014: 1783-1786.

[4]. Pan Hongliang. Research on Algorithm and Hardware Implementation of Floating Point Index Class Transcendental Function [D]. Northwestern Polytechnical University, 2006.

[5]. Chen Kean Tack, Siti Noormaya Bilmas. Hardware Implementation of Math Module based on CORDIC Algorithm using FPGA. IEEE, 2013: 1521-9097.