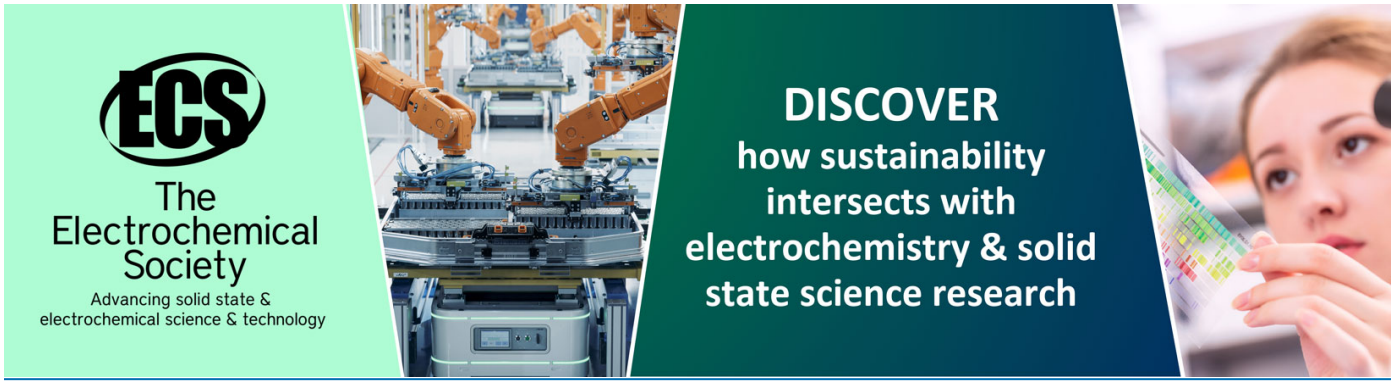
Research and design of activation function hardware implementation methods



Research and design of activation function hardware implementation methods

Chen Zhengbo[1,a], Tong Lei[1], Chen Zuoning[2]

[1]Information Engineering University, Zhengzhou, China

[2]Chinese Academy of Engineering, Beijing, China

[a]chenzb12@mails.tsinghua.edu.cn

[*]chenzb1996@163.com

Abstract—Activation function is an important part of neural network. Many artificial intelligence accelerators specially design hardware component supporting for activation function. In this article, we analyze normal activation functions, study kinds of activation function hardware implementation methods, and propose a parallel look-up table based piecewise linear fitting method. In many-core AI processor, we determine the table precision by memory size, and propose the basic algorithm and detailed structural design scheme. After the design is completed, we verify its correctness and evaluate its performance. The result shows that proposed method meets the need of correctness, and can finish the parallel computation of sixteen activation function fitting results within thirteen cycles. It greatly improves the efficiency of activation function hardware design.

1. INTRODUCTION

Recently, Artificial Intelligence technology develops rapidly, Deep Neural Network is going to be state-of-the-art in many applications such as computer vision [1], nature language processing [2]. Nowadays main stream neural network includes Convolutional Neural Network [1], Recurrent Neural Network [3], Long Short Term Memory [4], Generative Adversarial Nets [5]. As a nonlinear function, activation function plays a very important role in lots of neural networks. Activation function which commonly used in neural network includes Sigmoid, Tanh, Softmax, Relu. No matter training or inference, we have to use activation function in forward and backward propagation. Activation function is of great significance to the convergence of neural networks.

Activation function is nonlinear, making it hard to compute using the floating point multiply-add component. We need an efficient hardware implementation method.

Currently mainstream artificial intelligence chips specially design hardware component supporting for activation function, such as TPU [6], DianNao [7]. The overall architecture of DianNao is shown in Figure 1. In DianNao, the layers in CNN are divided into three levels. The first level is Multiply. The second level is Add/Max. The third level is Transfer Function (i.e., nonlinear function like Sigmoid). Convolutional layers and fully connected layers are composed of three levels, and pooling layers are composed of the first two levels. In the Transfer Function level, DianNao uses look-up table based linear fitting method to achieve hardware supporting for activation function.

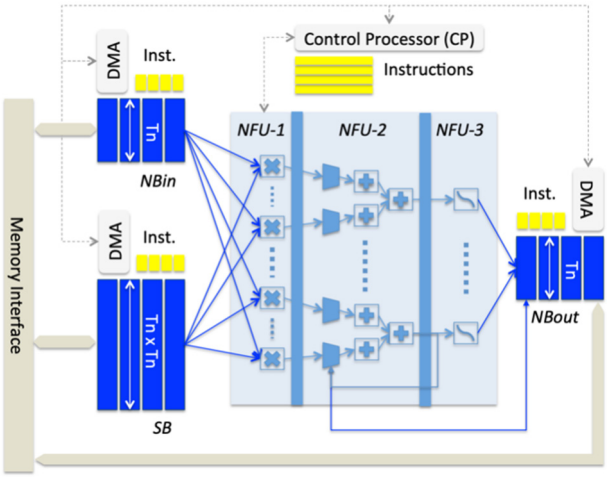


Figure 1. Overall architecture of DianNao.

Many-core AI chip is composed of many-core processor architecture, including Management Processing Element and Computing Processing Element. Computing Processing Element contains general-purpose computing component and AI acceleration core. Convolution is finished in AI acceleration core, and the results are sent to general-purpose computing component to finish activation function computation. The activation function component is in general-purpose computing component, realized by parallel look-up table based piecewise linear fitting method.

In this article, we analyze normal activation functions, study kinds of activation function hardware implementation methods. In many-core AI processor, we propose a parallel look-up table based piecewise linear fitting method, design the detailed architecture, and do some experimental verification. The result shows this architecture can accomplish activation function.

2. Analysis of Activation Function

Normal activation function includes Sigmoid, Tanh, Relu [8], Softmax, etc. Sigmoid function expression is in (1). Tanh function expression is in (2). Relu function sets the negative value to 0, and expression is in (3). Softmax function expression is in (4).

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$Tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2}$$

$$Relu(x) = \begin{cases} x, if\ x > 0 \\ 0, if\ x \leq 0 \end{cases} \tag{3}$$

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_1^C e^{x_i}} \tag{4}$$

The hardware implementation of Relu function is simple, and Softmax function is composed of exponential function and logarithmic function. So, in this article, we only consider Sigmoid, Tanh, Log(x) and Exp(x).

In the process of neural network training, we have to use activation function and derivation of activation function in forward and backward propagation. During the forward propagation, Sigmoid function is $\mathrm{Sigmoid}(x) = \sigma(x)$; During the backward propagation, derivation of Sigmoid function is $\sigma'(\mathrm{x}) = \sigma(\mathrm{x})(1 - \sigma(\mathrm{x}))$. So we can use function result and additional floating point multiply-add operation get the final result. Similarly, the forward propagation of Tanh function is $\mathrm{Tanh(x)} =$

$2\sigma(2x) - 1$, and the backward propagation of Tanh function is $\mathrm{Tanh}'(x) = 4\sigma'(2x)$. We can also use sigmoid function result and additional floating point multiply-add operation get the final result.

The hardware implementation of activation function includes look-up table method, Taylor expansion method, piecewise linear function method [9], etc.

Look-up table method is most intuitive and universal. Within the allowable range of error, it stores sampling value of nonlinear function. For any input number, this method directly accesses corresponding result. Look-up table method match input and output, getting the result in one memory access, but it costs too much memory.

Taylor expansion method uses Taylor expansion formula to simulate nonlinear function. Normal activation function is infinitely derivative, and Taylor expansion formula can approach the real value. But the computing units are finite, and high order operations are too complex. Taylor expansion method is rarely used. Its expression is in (5).

$$f(x) = f(x_0) + f^{(1)}(x_0)(x - x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{5}$$

Piecewise linear fitting method divides the nonlinear function, and uses multiple sections of linear function to fit the nonlinear function within the allowable range of error. The simplest piecewise linear fitting method is equal division. It divides the independent variable interval equally, and stores the slope and bias value per section. We access the corresponding slope and bias value by independent variable, and use floating point operation to get the fitting result. Its expression is in (6).

$$\tilde{y} = \begin{cases} a_0 x + b_0, x \in [x_0, x_0 + \Delta x) \\ a_1 x + b_1, x \in [x_0 + \Delta x, x_0 + 2\Delta x) \\ \cdots \\ a_k x + b_k, x \in [x_0 + k\Delta x, x_0 + (k+1)\Delta x) \\ a_{k+1} x + b_{k+1}, x \in [x_0 + k\Delta x, x_0 + (k+2)\Delta x) \\ a_n x + b_n, x \in [x_0 + (n-1)\Delta x, x_n] \end{cases} \tag{6}$$

3. MANY-CORE AI CHIP

Many-core AI chip is composed of many-core processor architecture, including Management Processing Element and Computing Processing Element. Computing Processing Element contains general-purpose computing component, AI acceleration core and Local Data Memory. The architecture of Computing Processing Element is shown in Figure 2. Proposed hardware implementation of activation function is completed in general-purpose computing component.

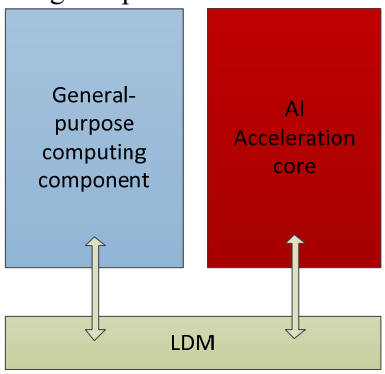


Figure 2. Architecture of Computing Processing Element

The memory of CPE consists of 256KB Local Data Memory. According to analysis of error and LDM size, we choose 64 as number of sections. This component can parallel finish 16 look-up table operations, and all slope and bias values store as single precision 32-bit floating point number. The overall memory is 64*32bit*2*16=64Kbit=8Kbyte, only costing 1/32 of LDM size.

4. DESIGN OF ACTIVATION FUNCTION ARCHITECTURE

4.1. Basic algorithm and overall architecture

Nonlinear function such as Sigmoid and Tanh is hard to directly solve, so piecewise linear function fitting method can reduce implementation cost. Linear fitting coefficients store in the table, and we can transfer the input single precision 32-bit number to get the signed offset address in the table.

The basic algorithm of look-up table based piecewise linear fitting method is shown as follow. Firstly, according to the exponent of input data, we align the 24-bit mantissa. Then we cut out the result based on instruction flag, combining the sign of input number, and get 8-bit signed number. Finally we expand 8-bit number with 6 zeros to 14-bit result, as the final table offset address.

Now we can determine the input and output of activation function architecture. Input 32-bit single precision number *i_src[31:0]*, 6-bit instruction flag *i_type[5:0]*(*i_type[5:3]* represent integer intercept bits, and *i_type[2:0]* represent fraction intercept bits). Output 14-bit signed offset address *o_disp[13:0]*, instruction exception flag *o_type_excep* and input data exceeding linear section exception flag *o_line_overflow*.

The architecture of parallel look-up table based activation function implementation is shown in Figure 3, including Alignment Module, Post-process Module and Exception Detection Module.

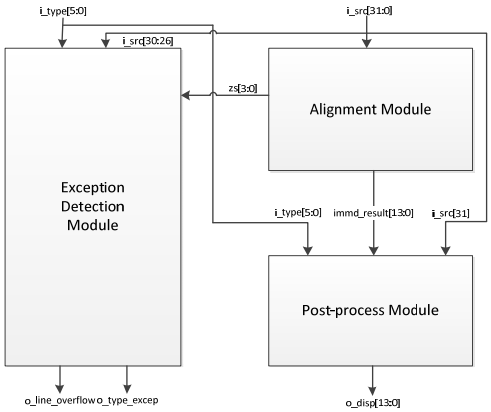


Figure 3. Architecture of parallel look-up table based activation function

4.2. Alignment Module

Alignment Module aligns the mantissa part of input data according to the exponent part of input data, getting 14-bit immediate result (7-bit integer and 7-bit fraction). The logic diagram of Alignment Module is shown in Figure 4.

Firstly, we transpose mantissa using the sign of input data, getting 24-bit number which need to be aligned. When the sign is zero, 24-bit number combines 1-bit 1 and 23-bit mantissa; When the sign is one, 24-bit number combines 1-bit 0 and 23-bit inverted mantissa.

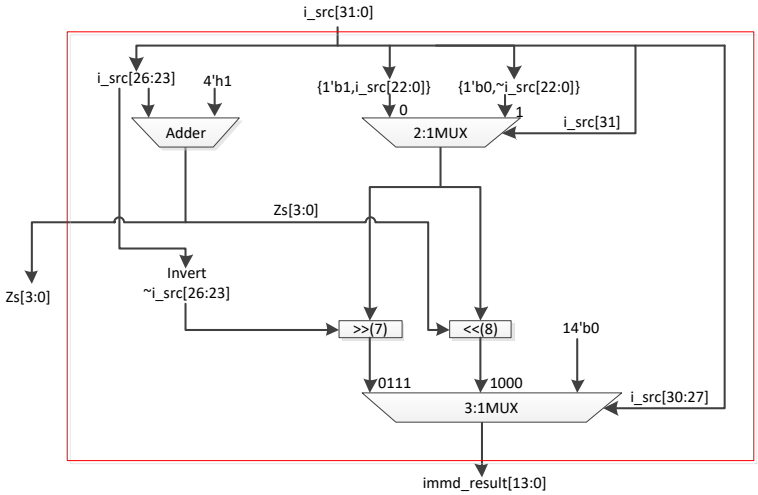


Figure 4. Alignment Module

Then, we align 24-bit number using the exponent of input data. After alignment, data need to be truncated, so the maximum align number is related to truncation. According to 7-bit integer, 7-bit fraction and position of decimal point, the maximum left align number is 8, and the maximum right align number is 7.

The maximum align number is 8, so we invert *i_src[26:23]* or plus *i_src[26:23]* and one, getting the right align number and left align number. The corresponding relationship of align number and exponent is shown in Table I. After alignment, we select correct 14-bit immediate result using *i_src[30:27]*. Meanwhile, left align number *zs[3:0]* is outputted to Exception Detection Module.

TABLE I. ALIGN NUMBER AND EXPONENT

Real Exp	Biased Exp	Binary Biased Exp	Alignment	Align Number
-127	0	00000000	7-bit right	111
…	…	…	…	…
-8	119	01110111	7-bit right	111
-7	120	01111000	7-bit right	111
…	…	…	…	…
-2	125	01111101	2-bit right	10
-1	126	01111110	1-bit right	1
0	127	01111111	none	0
1	128	10000000	1-bit left	1
2	129	10000001	2-bit left	10
…	…	…	…	…
8	135	10000111	8-bit left	1000
9	136	10001000	8-bit left	1000
…	…	…	…	…
128	255	11111111	8-bit left	1000

4.3. Post-process Module

Post-process Module calculates 14-bit table biased address using 14-bit immediate result and instruction flag. The logic diagram of Post-process Module is shown in Figure 5.

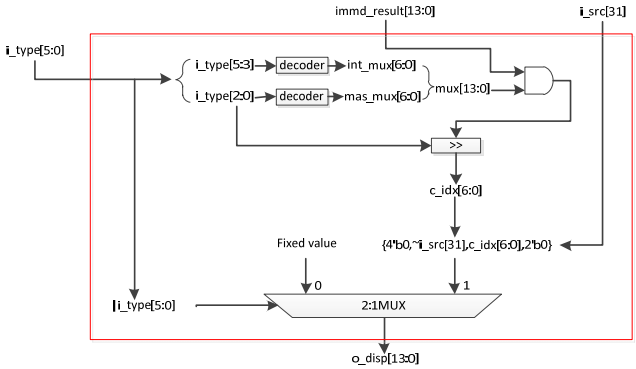


Figure 5. Post-process Module

We split instruction flag, high three bits *i_type[5:3]* representing integer, low three bits *i_type[2:0]* representing fraction. Then we decode split instruction flag using decoder and get 14-bit data. 14-bit data "AND" with 14-bit immediate result, and align using fraction flag, truncating 7-bit data, combining inverted input data sign *~i_src[31]* and 6 zeros, getting 14-bit table biased address. Normal look-up table operations output fixed value.

4.4. Exception Detection Module

Exception Detection Module makes exception judgment about instruction exception flag and input data exceeding linear section exception flag, these two exception are only effective during expanded parallel look-up table operations. The logic diagram of Exception Detection Module is shown in Figure 6.

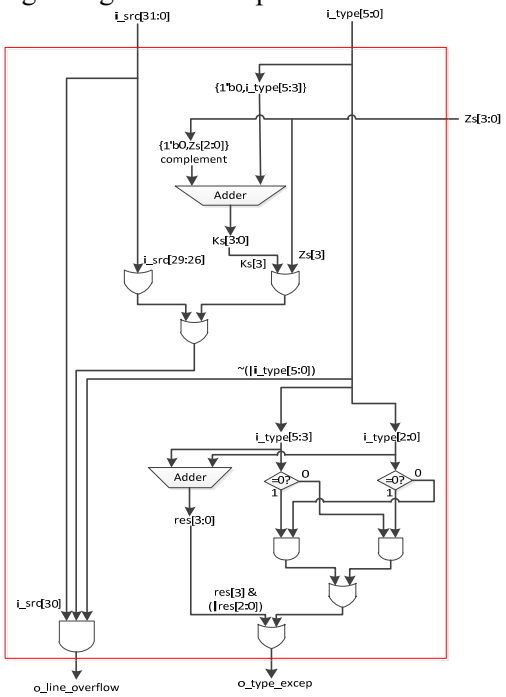


Figure 6. Exception Detection Module

Instruction exception flag. High three bits of instruction flag represent integer and low three bits of instruction flag represent fraction. If these two number are both zero or sum of them is less than or equal to 8, instruction has no exception. Or else instruction exception flag is 1.

Input data exceeding linear section exception flag. Input data exceeding linear section means input single precision data exceeds hardware representable largest linear section range, then input data exceeding linear section exception flag is 1. There are some conditions. Condition 1 is left alignment.

Condition 2 is left align number is more than 8. Condition 3 is left align number is more than largest representable bits. Input data exceeding linear section exception flag equals Condition 1 & (Condition 2 | Condition 3).

4.5. Experimental Verification

We implement parallel look-up table based activation function hardware in many-core AI processor, and verify its correctness and evaluate its performance.

After finishing the design of architecture and the code, we verify the correctness of this component using PSL language, including module level test, function points test and random number test. The results show that this component meets the need of correctness.

Parallel look-up table based piecewise linear fitting method implements activation function. Look-up table instruction takes 3 cycles, 1cycle for address calculation, 1cycle for memory access, 1 cycle for table look-up. Combing 6 cycles for two look-up table instructions and 7 cycles for floating point multiply-add instruction, 16 activation function fitting results are finished within 13 cycles totally. Comparing to 3 cycles for direct look-up table method, our method is far more efficient.

5. SUMMARY

In this article, we propose a parallel look-up table piecewise based linear fitting method for activation function hardware implementation, and use it in many-core AI processor.

We analyze normal activation functions and kinds of activation function hardware implementation methods, and propose our method. Then we determine the table precision by many-core AI processor memory size, and propose the basic algorithm and detailed structural design scheme. Finally, we do some experimental verification.

The results show that proposed method meets the need of correctness, and can finish the parallel computation of 16 activation function fitting results within 13 cycles, greatly improving the efficiency of activation function hardware design.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton G. “ImageNet classification with deep convolutional neural networks,” International Conference on Neural Information Processing Systems. 2012, pp. 1097-1105.

[2] L. Dong, F. Wei, M. Zhou, and K. Xu. “Adaptive multi compositionality for recursive neural models with applications to sentiment analysis,” AAAI Conference on Artificial Intelligence. Québec, Canada: AAAI Press, 2014: 1537-1543.

[3] K. Cho, B. V. Merrienboer, C. Gulcehre, et al. “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Computer Science, 2014, pp. 1724-1734.

[4] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. "LSTM: A Search Space Odyssey," IEEE Transactions on Neural Networks & Learning Systems, 2016, pp. 2222-2239.

[5] I. J. Goodfellow, J. P. Abadie, M. Mirza, et al. “Generative adversarial nets,” Cambridge: MIT Press, 2014, pp. 2672-2680.

[6] N. P. Jouppi, C. Young, N. Patil, et al. “In-Datacenter Performance Analysis of a Tensor Processing Unit,” Paper presented at the meeting of the ISCA, 2017.

[7] T. Chen, Z. Du, N. Sun, et al. “DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning,” Acm Sigplan Notices, 2014, pp. 269-284.

[8] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” Paper presented at the meeting of the ICML, 2010.

[9] S. Ngah, R. A. Bakar, A. Embong, and S. Razali. “Two-steps implementation of sigmoid function for artificial neural network in field programmable gate array,” ARPN Journal of Engineering and Applied Sciences, 2016, pp. 4882-4888.