Design of Neural Network Architecture using Systolic Array Implemented in Verilog Code

Trio Adiono[a,b,*], Grasia Meliolla[b], Erwin Setiawan[a], Suksmandhira Harimurti[a]

[a]University Center of Excellence on Microelectronics Institut Teknologi Bandung
4th Floor, PAU Building, Jl. Tamansari No. 126 Bandung 40132, West Java, Indonesia

[b]School of Electrical Engineering and Informatics Institut Teknologi Bandung
Achmad Bakrie Building, Jl. Ganesha No. 10 Bandung 40132, West Java, Indonesia

*Corresponding author: tadiono@stei.itb.ac.id

***Abstract*—In this paper, an implementation of a neural network model using systolic arrays, programmed in Verilog Code, is presented. The neural network model is mapped in a three-layer perceptron in forward mode. Dependency graph is also provided to illustrate the operations in each phases of the neural network model. Afterwards, the operations in a linear directional of systolic array is realized using a recursive iterative algorithm. The modelling in Verilog code is later confirmed with the MATLAB code for 9-input-output structure. The result proofs that the neural network architecture based on systolic array is successfully implemented in Verilog code.**

Keywords—artificial neural network; systolic array; Verilog; dependency graph; recursive iterative algorithm

I. INTRODUCTION

Artificial neural networks (ANN) have emerged as a promising method/system for solving a complex real world problems, such as speech or pattern recognition and signal tracking since the implementation of ANN is able to significantly enhance the execution/computing time of an existing system [1]-[3]. Basically, neural network is a life being's neurons imitating system constructed in mathematical model. ANN system principally can be categorized in three layers i.e., (1) input layer, (2) hidden layer, (3) output layer [4]. This construction is called three-layer structure. The relationship between these layers, is shown in Fig. 1. In the most complex structure, hidden layer can possibly be more than one layer. Neural network modifies several input signals by adding bias and weight to the signals. At the end of the output layer, there is a supervisor which has a given ideal output beforehand.

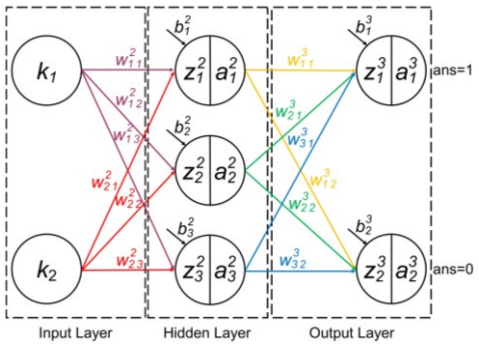

Fig. 1. Three-layer model of neural networks with parameter

The definition for each parameter shown in Fig. 1 is as follow:

k_i = input signal
w_{ij}^2 = weight from input layer to hidden layer
b_i^2 = bias for hidden layer
z_i^2 = hidden layer input
a_i^2 = hidden layer output
w_{ij}^3 = weight from hidden layer to output layer
b_i^3 = bias for output layer
z_i^3 = output layer input
a_i^3 = output layer output

There are several methods in implementing neural network, e.g. bit-slice, single instruction multiple data (SIMD), and systolic array [5-9]. Systolic array is the best parallel architectures amongst all for implementing neural network system since it can overcome the possible communication problem faced in a high degree of interconnections among neurons. Several systolic array algorithms can be represented with a matrix vector multiplication. This matrix vector is actually the basic computation of operations in a neural network. Here in this work, we propose an implementation of the neural network using systolic array. A matrix vector multiplication is used to do the computation of the neural network. The systolic array is then modelled and implemented in Verilog as well as MATLAB code. The simulation on both implementations are conducted to verify the validity of the model design.

II. SYSTEM DESIGN

In this work, the computations in the neural network are represented in a matrix vector multiplication as outlined in equation (1):

$$\begin{pmatrix} z_0^2 \\ z_1^2 \\ \vdots \\ z_8^2 \end{pmatrix} = \begin{pmatrix} w_{00}^2 & \dots & w_{08}^2 & b_0^2 \\ w_{10}^2 & \dots & w_{18}^2 & b_1^2 \\ \vdots & \vdots & \vdots & \vdots \\ w_{80}^2 & \dots & w_{88}^2 & b_8^2 \end{pmatrix} \bullet \begin{pmatrix} k_0 \\ \vdots \\ k_8 \\ 1 \end{pmatrix} \quad (1)$$

In equation (1), z, w, b and k vectors represents the outputs, weights, biases, and inputs values, respectively. A dependence graph (DG) for computing the z, b, and k values of a neuron layer from their values at the preceding layer is shown in Fig. 2. Every node presents multiplication and addition operation of processor element (PE). In this DG, there is no information about time or t = 0. Therefore, this DG cannot be mapped to processor array.

978-1-5386-6670-8/18/$31.00 ©2018 IEEE

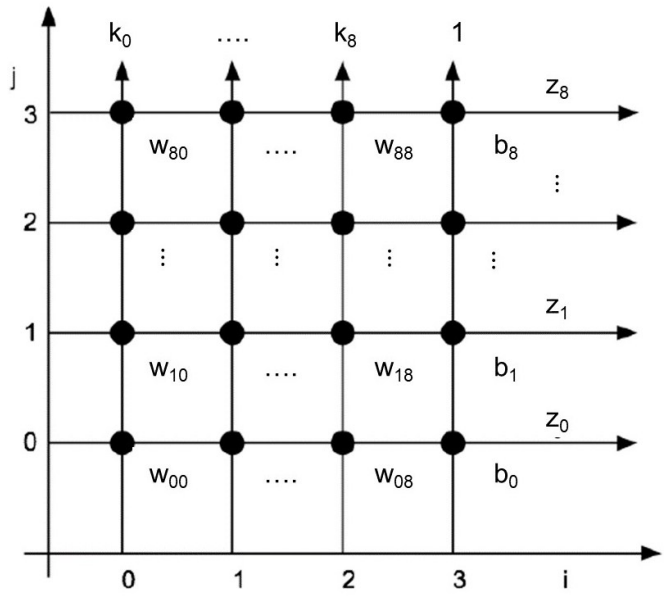

Fig. 2. Dependent graph of the neural network

From DG, the recursive iterative algorithm (RIA) can be written as in (2). The weight (w) position element is constant, the input (k) is propagated in j-axis, and the output (z) is the result from the addition process conducted in i-axis.

$$\begin{aligned} w(i,j) &= w(i,j) \\ k(i,j) &= k(i,j-1) \\ z(i,j) &= z(i-1,j) + w(i,j) * k(i,j) \end{aligned} \tag{2}$$

From the RIA equations, the reduced dependence graph (RDG) model is then constructed, as shown in Fig. 3.

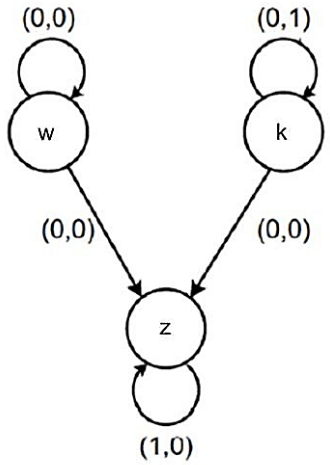

Fig. 3. The reduced dependence graph of the neural network

After applying the inequalities at every edge of the RDG, the inequalities, as in (3), are determined in order to obtain scheduling vector, projection vector and processor vector.

$$\begin{aligned} w \to w: e &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \gamma_w - \gamma_w \geq 0 \\ k \to k: e &= \begin{bmatrix} 0 \\ 1 \end{bmatrix}, s_2 \geq 0 \\ z \to z: e &= \begin{bmatrix} 1 \\ 0 \end{bmatrix}, s_1 \geq 1 \\ w \to z: e &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \gamma_z - \gamma_w \geq 0 \\ k \to z: e &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \gamma_z - \gamma_k \geq 0 \end{aligned} \tag{3}$$

Accordingly, we obtain s1 = 1, $s2 = 0$ and respectively the possible solutions are provided in (4).

$$\begin{aligned} d^T &= [1 \quad 0] \\ p^T &= [0 \quad 1] \\ s^T &= [1 \quad 0] \end{aligned} \tag{4}$$

where

d^T = projection vector

p^T = processor vector

s^T = scheduling vector

Based on these vectors, the calculation is conducted to obtain the systolic array architecture.

Firstly, the processor mapping is determined, as provided in (5).

$$j' = p^T l = [0 \quad 1] \begin{bmatrix} i \\ j \end{bmatrix} = j \tag{5}$$

and then time vector when the processor is executed, as provided in (6).

$$t' = s^T l = [1 \quad 0] \begin{bmatrix} i \\ j \end{bmatrix} = i \tag{6}$$

Afterwards, the DG is remapped in two directions, which are j for processor direction and i for scheduling (time), as shown in Fig. 4.

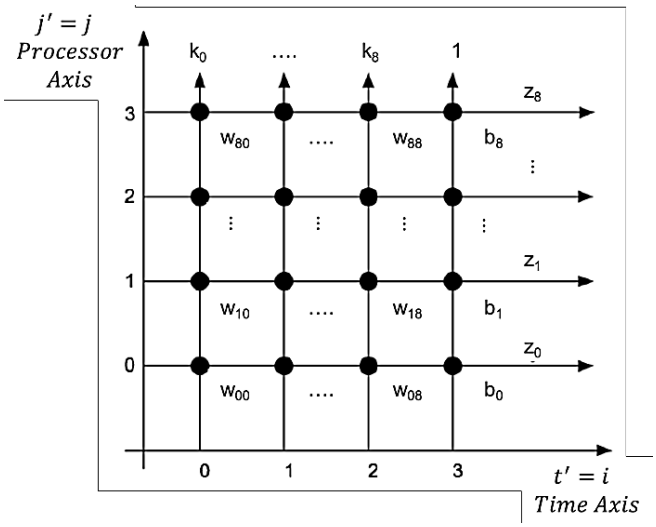

Fig. 4. Resulted dependent graph of the neural network

The Hardware Utilization Efficiency (HUE) of the systolic array is also calculated using (7).

$$HUE = \frac{1}{s^T l} = \frac{1}{[1 \quad 0]\begin{bmatrix}1\\0\end{bmatrix}} = 1 \tag{7}$$

From equation (7), the calculated HUE is 1. This results means that every processor works all the time and has been optimized. Furthermore, the edge mapping is also conducted, as shown in Table I.

TABLE I. EDGE MAPPING

Edge=e	**Input = $p^T e$ (processor direction)**	**Result = $s^T e$ (# Delay element)**	**Description**
Weight: $w = [0 \quad 0]$	0	0	Constant weight, 0 delay element
Input: $k = [0 \quad 1]$	1	0	Input is in the same direction with processor, 0 delay element
Result: $z = [1 \quad 0]$	0	1	Constant output, 1 delay element

Using this edge mapping, the data flow graph (DFG) of systolic array for the matrix multiplication of neural network system is then given in Fig. 5.

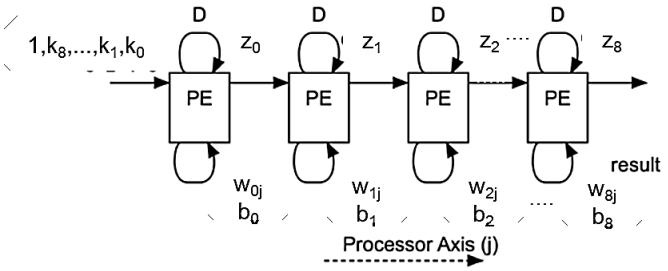

Fig. 5. Data flow graph of the systolic array of the neural network system

Accordingly, the block diagram of the architecture is shown in Fig. 6. The diagram shows that broadcast input and output has one delay element.

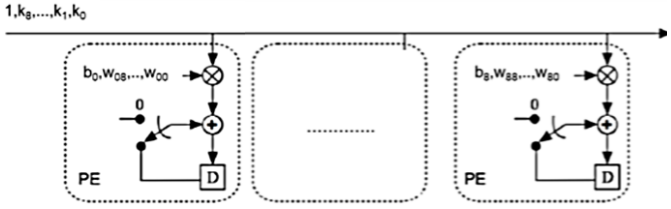

Fig. 6. Block diagram of the systolic array

III. RESULTS AND DISCUSSION

The systolic array model is implemented in Verilog code and has been verified using the 9-layer MATLAB model for the similar system without systolic calculation. The modelling has been conducted in forward mode. The results show that both methods of calculation has only small differences in output values and thus confirm the validity of our systolic array modelling.

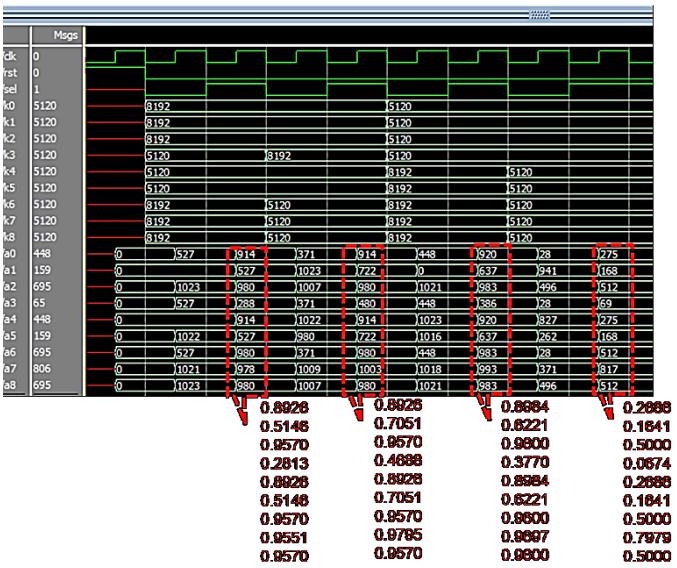

Fig. 7. Calculation result of the neural network output (using systolic array) in Verilog

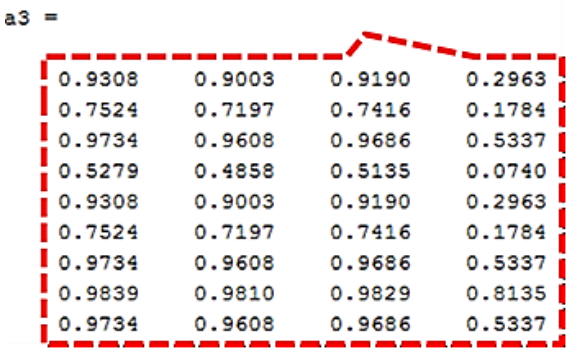

Fig. 8. Calculation result of the neural network output (without systolic array) in MATLAB

IV. CONCLUSION

An implementation of neural network architecture using systolic array has been designed. The computation method for the neural network is conducted by using matrix representation, while the systolic array itself is implemented in Verilog code. To validate the implemented model, the simulation in Verilog code as well as in MATLAB are conducted. The result shows a very small difference which proofing the validity of the neural network modelling using systolic array in Verilog code.

REFERENCES

[1] J. Li, S. L. Phung, F. H. C. Tivive and A. Bouzerdoum, "Automatic classification of human motions using Doppler radar," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Brisbane, QLD, 2012, pp. 1-6.

[2] W. Liu *et al*, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11-26, Apr. 2017.

[3] F. Agostinelli, M. R. Anderson, and H. Lee, "Adaptive multi-column deep neural networks with application to robust image denoising," in *the 26th International Conference on Neural Information Processing Systems*, Stateline, NV, 2013, pp. 1493-1501.

[4] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy Layer-wise Training of Deep Networks," in *the 19th International Conference on Neural Information Processing Systems*, Canada, 2006, pp. 153-160.

[5] M. Glesner, W. Poechmueller, *Neurocomputers: An Overview of Neural Networks in VLSI*. London, UK: Chapman and Hall, 1994.

[6] P. Ienne, "Digital hardware architectures for neural networks," *Speedup Journal*, vol. 9, no. 1, pp. 18-25, June 1995.

[7] H. Amin, K.M. Curtis, B.R. Hayes-Gill, "Two-ring systolic array network for artificial neural networks," *IEE Proceedings of Circuits, Devices, and Systems*, vol. 146, no.5, pp. 225-230, Oct. 1999.

[8] D.L. Hung, J. Wang, "Digital hardware realization of a recurrent neural network for solving the assignment problem," *Neurocomputing*, vol. 51, pp. 447-461, Apr. 2003.

[9] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1-3, pp. 239-255, Dec. 2010.