Implementation of artifiical neural networks by using synchronous dataflow models

Biao Lu

Department of Electrical and Computer Engineering

University of Texas, Austin, TX 78712-1084

E-mail: blu@ece.utexas.edu

January 26, 2002

Abstract

Artificial neural networks(ANNs) can be modeled by dataflow graphs because both of them use directed graphs with nodes and edges. Similarities between data flow graphs and neural networks prove that the dataflow models can be used for the parallel implementation of neural networks. Hopfield network, backpropagation network, and a biological network are modeled by the dataflow graphs and mapped to homogeneous synchronous dataflow. Describing neural networks as data flow models eases the embedding of neural networks into heterogeneous systems. (80 words)

1 Introduction

The principle of dataflow and the applications to the development of computer architectures were first proposed by Dennis [3]. A major objective of a dataflow concept is to facilitate the exploitation of data-driven parallel computation. The term "data-driven" means that the activation of an operation is determined by the availability of its input data. Because of their data-driven execution principle, dataflow graphs can be used for many parallel applications. One of the applications is to implement artificial neural networks by using dataflow models.

An ANN is a computational model of biological neural networks. All the neurons are interconnected and arranged in the different layers such as input, hidden, and output layers. A biological neuron is shown in figure 1(a). The dendrites of a neuron carry the signals in from the other neurons. There is a chemical process at the synaptic site to scale the signals. Once the signals are greater than a threshold, the neuron fires and broadcasts the output signal to the other neurons. A mathematical model of a neuron is shown in figure 1(b). x_1, x_2 and x_3 are input signals. w_1, w_2 and w_3 are the scaling factors at the synaptic site, which are called weights. The neuron will sum the weighted inputs and invoke the activation function. The output y will be passed to the other neurons.

Dataflow models are directed graphs. Their nodes represent operations, and the edges connecting these nodes represent the operands of these operations. An operation can be executed, that is, the node is fired, if all the tokens on incoming edges are available. *Synchronous* dataflow(SDF) is a restricted type of dataflow models (figure 1(c)). The number of tokens produced by a node firing on each output edge is a fixed number that is known at compile time. There is a non-negative integer delay on each edge in an SDF graph. The delay corresponds to the number of initial tokens on the edge. The usage of delays will be seen in the section of implementation.

Describing neural networks as data flow models eases the embedding of neural networks into heterogeneous systems. Some researchers have finished some previous work. Yuceturk, *et al* [9] gave a formal definition of mapping neural networks onto data flow graphs. Kim, *et al* [5] and Wang, *et al* [8] proposed the neural-dataflow graph transformation. Achyutahn, *et*

(a) A biological neuron (b) A mathematical model (c) A SDF graph

Figure 1: Models of a neuron

al [2] and Mutlaq, *et al* [7] addressed the hardware design of ANNs by the data flow models. Liu, *et al* [6] used a dataflow specification methodology to specify a neural network design and the system dynamics. Hopfield networks and backpropagation networks are simulated on the neural dataflow-based processor, the dataflow-based multiprocessor system, and the MIT tagged token dataflow software simulator by Mutlaq, *et al* [7], Wang, *et al* [8], and Kim, *et al* [5], respectively. A spike-processing biological neural network is simulated by Jahnke, *et al* [4]. In this work, the author presents the models and the implementation of ANNs by using the *synchronous* dataflow models, especially the *homogeneous* SDF models. Several neural networks are implemented in Ptolemy [10].

The remainder of this paper is organized as follows. Section 2 discusses the dataflow models of neurons and networks. The implementation of the different neural network is described in section 3. Section 4 concludes the work and points out the future directions of this work.

2 Dataflow models of neural networks

Figures 1 shows the similarity between the biological neuron and the SDF graph. Therefore, the dataflow graphs can be used to model neurons and neural networks. The mathematical model in figure 1(b) shows that the neuron sums the weighted inputs and invokes the activation functions. In this work, a neuron is divided into two parts. The neuron we called in this work only performs the weighted sum. The activation functions are generated separately because one neuron can invoke the different kinds of activation functions, but it always sums the weights inputs.

Two different kinds of neurons are defined. The first kind is the data-passing neuron such as input neurons and threshold neurons. All input neurons receive their inputs only from the external world and send them to the hidden layer. A threshold neuron will output -1 to neurons in the next layer. Both input and threshold neurons have no operational function. The second is called the data-processing neurons such as the hidden neurons in a hidden layer and the output neurons in an output layer. Data-processing neurons calculate the weighted sum and output the summation to an activation function. There are different types of activation functions based on the applications. The commonly-used activation functions are binary function like the unit step function, binary sigmoid function ($f(x) = \frac{1}{1+e^{-x}}$) with the output from 0 to 1, and bipolar sigmoid function ($g(x) = 2f(x)-1$) with the output from -1 to 1. A tanh function $tanh(t) = 2f(2x) - 1$ is usually used to model the bipolar sigmoid function.

There is a special neuron called McCulloch-Pitts neuron [1]. A McCulloch-Pitts neuron, the earliest artificial neuron, can sum the weighted input and invoke the binary function to produce the binary output. McCulloch and Pitts attempted to establish a link between neurology and computational theory. In particular, they focused on how primitive logical operations might be carried out by single neurons, and how complex logical operations might be carried out by complex networks of neurons. A McCulloch-Pitts neuron is used as a logic function is shown in figure 2(b).

Neural networks consists of data-passing and data-processing neurons and at least one of activation functions. Neurons and activation functions will form one layer. Therefore, there are single layer and multilayer neural networks, which will be implemented in next section.

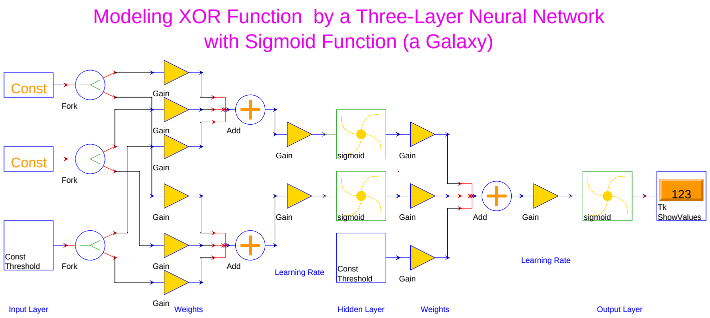
3 Implementation of neural networks

3.1 Model the simple neural networks

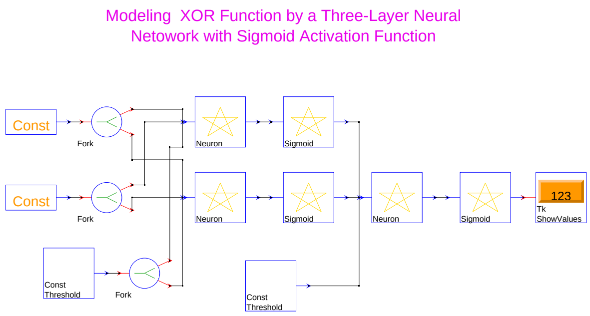
The simple AND function 2(a) is implemented by the existing stars in SDF domain in Ptolemy based on the SDF model of a neuron, which is a data-processing neuron plus an activation function. "Gain" stars here act as the weights. A "Sgn" star, a sign function, is used as a bipolar activation function. A more complicated XOR function with three neurons is shown in figure 3(a). The sigmoid function in figure 3(a) consists of the different existing stars, which is called a galaxy in Ptolemy. Since there are not enough stars to simulate the simple neural networks, the new stars are generated in Ptolemy. Figures 2(b) and 3(b) show the implementation of AND and XOR logic functions by using the new stars. Comparisons of implementation times are shown in tables 1 and 2 between these two methods: the implementation with the existing stars and the implementation with the new generated stars. The difference between these two methods is that the first method uses more blocks than the second method. Using more blocks causes more implementation times. Therefore, it is worth to generate the new stars to combine some calculations.

(a) simulation with the existing stars (b) simulation with the new star

Figure 2: An AND function



(a)simulation with the existing stars



(b) simulation with the new star

Figure 3: An XOR function

Input 1	Input 2	output(M-P)	time(s)	output(star)	time(s)
0	0	0	0.014471	-1	0.01783
0	1	0	0.010954	-1	0.015137
1	0	0	0.050023	-1	0.054113
1	1	1	0.01225	1	0.015562

Table 1: **Implementation times of AND functions**

Input 1	Input 2	output	time(new)	time(stars)
0	0	0.11026	0.214433	0.478633
0	1	0.861419	0.464358	0.969289
1	0	0.924015	0.21618	0.699778
1	1	0.101868	0.2803	0.491132

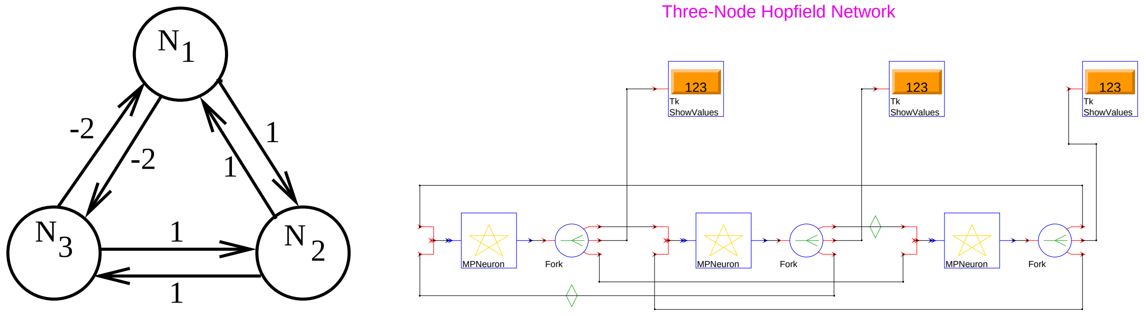
Table 2: **Implementation times of an XOR function**

3.2 Model the single-layer network

Hopfield network [7] [8] is a kind of single-layer ANN. The net is a fully interconnected neural network - each unit is connected to every other unit. The network has symmetric weights with no self-connections, i.e., $w_{ij} = w_{ji}$, and $w_{ii} = 0$. Figure 4(a) is a 3-neuron Hopfield network, and figure 4(b) is its implementation in Ptolemy. The two diamonds in figure 4(b) are the initial delays used in SDF models. The Hopfield network is a recurrent and feedback network without any input and output. The initial values must be set in order to avoid the deadlock. No matter where the delays are initialized, the network will reach the stable state since Hopfield network is a memory association network and the weights are associated with the stable state. The stable state for the network with the weights in figure 4(a) is 011.

3.3 Model a physiological phenomenon

The previous sections show that the SDF graphs can model the neural networks. However, they are the *artificial* neural networks. In this section, McCulloch-Pitts neurons are used to model the perception of heat and cold. This is a well-known and interesting physiological phenomenon. It was originally presented by McCulloch and Pitts [1] in 1943. If a cold stimulus is applied to a person's skin for a very short period of time, the person



(a) 3-neuron Hopfield network (b) Implementation of Hopfield network

Figure 4: Single layer Hopfield network

will perceive heat. However, if the same stimulus is applied for a longer period, the person will perceive cold. This physiological phenomenon can be modeled by SDF models and implemented in Ptolemy (figure 5). The impulse star is used to simulate the short-period stimulus. When the schematic is running, the heat perceptor will output 1.

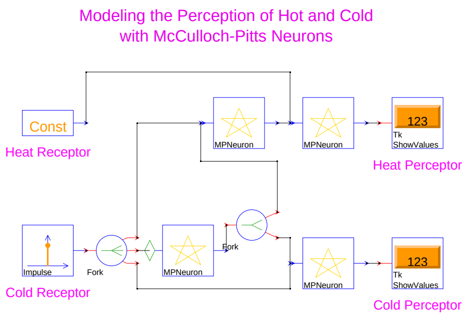


Figure 5: Simulation of the perception of heat and cold

4 Conclusions and future work

The similarities between ANNs and SDF graphs make it possible to implement ANNs by using SDF models. The development of massively parallel systems such as ANNs is a complicated process. SDF graphs are suitable models for the neural computing. The modelings of neurons and networks are described in this paper. A single-layer network such as Hopfield network, multilayer networks such as the backpropagation, and a biological simulation of the perception of heat and cold are implemented in SDF domain in Ptolemy. In other words, supervised network (backpropagation) and unsupervised network (Hopfield network) can be modeled by SDF models, especially *homogeneous* SDF models, because the number of tokens per arc is restricted to one during the entire computation.

All the weights in the networks are precalculated and transferred to the SDF models. Training ANN has not been implemented in Ptolemy. Although the author implements the training process by the interface to Matlab, this training is completed in Matlab instead of the SDF model. Standard backpropagation star can be generated to train the neural network. Since there are different kinds of backpropagation algorithms, a lot of issues need to be studied in order to generate the general backpropagation algorithms. Boolean dataflow models would be suggested to train the different topologies of ANNs inside Ptolemy.

All the simulations in this work are based on the small networks with a small amount of neurons. Higher order functions would be suggested to design large networks. Although there are the similarities between ANNs and SDF models, the author still doubts whether there exists a neural network which cannot be modeled by SDF models. This work proves the possibilities of designing the neural networks at the system level. It also raises a lot of open issues in embedding neural networks in heterogeneous systems.

Acknowledgments

I want to express my thanks to Dr. Brian Evans for his encouragement and the suggestions during the whole semester. He is a good example of embedding the hard-working system into himself.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943

[2] A. Achyuthan and M. I. Elmasry, "Mixed analog/digital hardware synthesis of artificial neural networks" *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems,* vol. 13, No. 9, pp. 1073-1087, September 1994.

[3] S. S. Battacharyya, P. K. Murthy, and E.A. Lee, "Software synthesis from dataflow graphs", *Kluwer Academic Publisher*, ISBN 0-7923-9722-3.

[4] A. Jahnke, U. Roth, and H. Klar, "A SIMD/dataflow architecture for a neurocomputer for spike-processing neural networks (NESPINN)', *Proc. of the Fifth Int. Conf. on Microelectronics for Neural Networks and Fuzzy Systems,* Lausanne, Switzerland, pp. 232-237, February 12-14, 1993.

[5] S. T. Kim, K. Suwunboriruksa, S. Herath, A. Jayasumana, and J. Herath, "Algorithmic transformations for neural computing and performance of supervised learning on a dataflow machine," *IEEE Transactions on Software Engineering,* vol. 18, No. 7, pp. 613-623, July 1993.

[6] Peter L. M. Liu, W. K. Kan and W. Wang, "A specification approach to artificial neural network design", *IEEE Region 10 Conf. on Computer and Communication System*, Hong Kong, pp. 40-44, September 24-27, 1990.

[7] M. Abu Mutlaq and R. Braham, "A dataflow processing element for neural network simulation," *Proc. of the IEEE Int. Conf. on Neural Networks,* Perth, Australia, vol. 1, pp. 398-402, 1995.

[8] C. C. Wang and B. Shirazi, "Neural networks implementation on a parallel machine," *Proc. SPIE: Applications of Optical Engineering: Proceedings of Optical Engineering , Rosemont, IL, USA, 27-28 Sept. 1990*, vol. 1396, pp. 209-213, 1991

[9] A. C. Yuceturk, B. Klauer, S. Zickenheiner, R. Moore, and K. Waldschmidt, "Mapping of neural networks onto data flow graphs" *Proc. of the 22nd EUROMICRO conf. Beyond 2000: Hardware and Software Design Strategies*, Prague, Czech Republic, pp. 51-57, September 2-5, 1996.

[10] Ptolemy Project, version 0.6, 1997, *http://ptolemy.eecs.berkeley.edu/.*