Simulation and Synthesis of Artificial Neural Networks Using Dataflow Models in Ptolemy

Biao Lu and Brian L. Evans
Laboratory for Vision Systems
Dept. of Electrical and Computer Eng.
The University of Texas at Austin
Austin, Texas 78712-1084
E-mail: *blu@ece.utexas.edu*
bevans@ece.utexas.edu

Dejan V. Tošić
School of Electrical Engineering
University of Belgrade
Bulevar Revolucije 73
11000 Belgrade, Yugoslavia
E-mail: *etosicde@ubbg.etf.bg.ac.yu*

Abstract— **Artificial neural networks (ANNs) are often mixed with digital signal processing (DSP) to form understanding and interpretation systems. These systems are *heterogeneous* in that they contain a variety of algorithms which may be synthesized onto a variety of software and hardware technologies. Because ANN and DSP subsystems are generally data-driven, their computation and control structure can be unified under a dataflow *model of computation*. In this paper, we demonstrate that ANN and DSP subsystems can be modeled using dataflow models that yield *static* implementations on either sequential or parallel machines. We map Hopfield, backpropagation, and biological networks to Homogeneous Synchronous Dataflow (SDF) models. We combine Boolean Dataflow (BDF) and SDF models to model Cellular Neural Networks (CNNs). By modeling DSP operations in SDF, we are free to mix ANNs and DSP subsystems and still retain efficient simulated and synthesized systems due to the static scheduling. We give several examples of simulating and synthesizing mixed ANN/DSP systems using the Ptolemy software environment.**

I. Introduction

The principle of dataflow and its application in computer architectures were first proposed by Dennis [1]. 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 to specify many parallel applications for simulation and synthesis, such as artificial neural networks (ANNs).

An ANN is a computational model of biological neural networks. In ANNs, all of the neurons are connected to at least one other neuron and arranged in 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 from the other neurons. A chemical process occurs 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.

ANNs are generalizations of mathematical models of human cognition or neural biology, which are based on the following assumptions [2]:

1. information processing occurs at many simple elements called neurons or nodes;
2. signals are passed between neurons over connection links;
3. each connection link has an associated weight; and
4. each neuron applies an activation function to its net input to determine its output signal.

Figure 1(b) shows a mathematical model that fits these assumptions, where x_1, x_2 and x_3 are input signals, and w_1, w_2 and w_3 are the scaling factors (weights) at the synaptic site. The artificial neuron sums the weighted inputs, applies an activation function to the weighted sum, and passes the result to the other neurons on output y.

Using nodes and links to specify an artificial neuron makes an ANN similar to a dataflow graph. A dataflow graph is a network of basic components such as nodes, arcs and tokens. Their nodes represent operations, and the arcs 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. Dataflow models have many characteristics in common with ANNs:

1. they can be represented by directed graphs with nodes and links (arcs);
2. each node functions as a simple processing element and operates asynchronously with other nodes; and
3. both models execute value-passing computations.

Therefore, dataflow models of computation offer an effective way to achieve efficient parallel computation and lead to efficient implementations of neural networks [3].

The execution of neurons obey *Synchronous* dataflow (SDF) semantics. SDF is a restricted type of dataflow (see Figure 1(c)) first proposed by Lee in 1986 [1]. In an SDF graph, nodes represent operations called actors. Arcs represent first-in first-out (FIFO) communication channels. with blocking reads and non-blocking writes. If an actor attempts to read more data than is available in the FIFO queue, then execution of the actor blocks (suspends) until enough data is available. The FIFO queues passed tokens which can represent an atomic data type (e.g. integer or real) or a data structure (e.g. a matrix or an image).

The research was supported in part by grants from the Shell Oil Company Foundation, the National Science Foundation CAREER Award under Grant MIP-9702707, and The University of Texas at Austin Summer Research Assignment Grant.

In this paper, we model ANNs using SDF, except for Cellular Neural Networks (CNNs). Section II describes SDF in more detail and summarizes previous work. Section III discusses dataflow models neural networks. Section IV presents the modeling and simulation of different neural networks. Section V discusses the synthesis of neural networks in software. Section VI concludes the paper.

II. Background

SDF graphs obey the following semantics:

1. An actor is enabled for execution when enough tokens are available at all of the inputs.
2. When an actor executes, it always produces and consumes the same fixed amount of tokens.
3. The flow of data through the graph may not depend on values of the data.

A non-negative integer delay may be associated with an arc (as will be seen in Section IV). When the SDF graph starts executing, the delay corresponds to the number of initial tokens on the arc. In SDF, all computation and data communication can always be scheduled statically. So, algorithms expressed as valid SDF graphs can always be converted into an implementation that is guaranteed to take finite-time to complete all tasks and use finite memory. Thus, an SDF graph can be executed over and over again in a periodic fashion without requiring additional resources as it runs. This type of operation is well-suited to digital signal processing and communications systems which often process an endless supply of data. Neural networks have been successful in a wide range of *heterogeneous* systems, e.g. in pattern recognition, vision, and control systems, in which neural networks are combined with other styles of algorithms. Dataflow models, which have a rigorous mathematical foundation, provide a formal way to specify, analyze, simulate, and synthesize heterogeneous systems. Many implementation properties such as boundedness of memory usage and execution time can be checked in some dataflow models *without* simulation. Because of their complexity, heterogeneous systems are often simulated using discrete-event techniques. Compiled simulation using dataflow models, however, can run up to ten thousand times faster.

Many other researchers have discussed dataflow modeling of neural networks. Yuceturk *et al.* [4] gave a formal definition of mapping neural networks onto dataflow graphs. Kim *et al.* [5] and Wang *et al.* [3] proposed the neural-dataflow graph transformation. Achyuthan *et al.* [6] and Mutlaq *et al.* [7] addressed the hardware design of ANNs by dataflow models. Liu *et al.* [8] 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 [7], the dataflow-based multiprocessor system, [3] and the MIT tagged token dataflow software simulator [5]. A spike-processing biological neural network is simulated by Jahnke *et al.* [9].

III. Dataflow models of neural networks

Figure 1 shows the similarity among the biological neuron, and the mathematical and computational models. Therefore, the dataflow graphs can be used to model neurons and neural networks. The mathematical model in Figure 1(b) shows that a neuron sums the weighted inputs and invokes the activation functions. In this paper, a neuron is divided into two parts. The first part of a neuron only performs the weighted sum. The second part of a neuron is an activation function which is generated separately because one neuron can invoke different kinds of activation functions, but it always sums the weighted inputs.

There are two different kinds of neurons defined in the first part of a neuron. 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 a 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 kind is called the data-processing neurons such as hidden neurons in a hidden layer and 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 restricted on the interval from -1 to 1. A tanh function $\tanh(t) = 2f(2x)-1$ is usually used to model the bipolar sigmoid function.

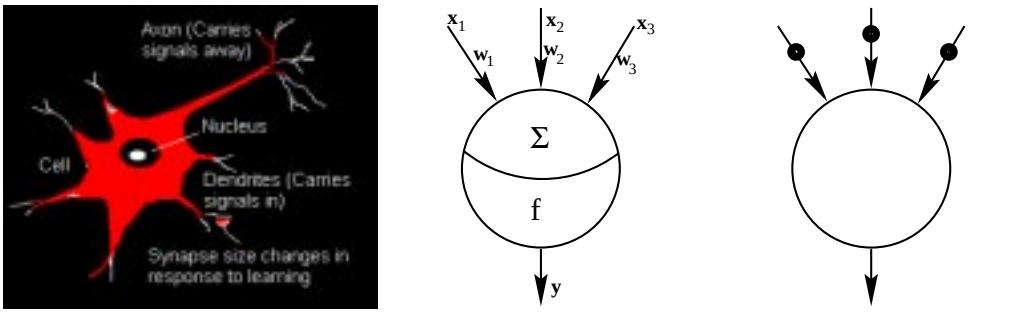
The McCulloch-Pitts neuron [11], the earliest reported artificial neuron, sums the weighted inputs and invokes the binary activation function to produce the binary output. McCulloch and Pitts attempted to establish a link between neurology and computational theory. In particular, they focused on how 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 shown in Figures 2(b) and 5.

Neural networks consist 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.

IV. Modeling and simulation of neural networks in SDF domain

A. Model of simple neural networks

The SDF model of a neuron consists of a data-processing neuron plus an activation function. Figure 2(a) implements a simple AND function using existing blocks (called stars) in the SDF domain in Ptolemy 0.6. The "Gain" stars act as the weights. The "Sgn" (signum) stars implement the bipolar activation function.



(a) Model of a biological neuron (b) Mathematical model (c) Computational model

Fig. 1. Models of a biological neuron and artificial neurons.

In Figure 3(a), which shows a more complicated XOR function with three neurons, the sigmoid function is implemented as a subsystem (called a galaxy) of existing Ptolemy 0.6 stars. We add several new stars to Ptolemy 0.7 to realize activation functions directly to improve the execution time. In simulation, execution time for a given system is directly proportional to the number of blocks in the specification. Figures 2(b) and 3(b) show the implementation of AND and XOR logic functions using the new stars. Tables I and II compare the execution time of the same systems using existing stars and the new stars. The new stars decrease the simulation time by a factor 2. In synthesize code, on the other hand, the dramatic improvement in writing custom stars is less in execution time and more in data memory usage because increasing the number of stars would increase the amount of buffer memory to implement the FIFO channels on the arcs.

Inputs		Outputs		Time(s)	
1	2	M-P	Star	M-P	Star
0	0	0	-1	0.014471	0.01783
0	1	0	-1	0.010954	0.015137
1	0	0	-1	0.050023	0.054113
1	1	1	1	0.01225	0.015562

TABLE I
IMPLEMENTATION TIMES OF AND FUNCTIONS.

Inputs		Output	Time(s)	
1	2	value	new	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 II
IMPLEMENTATION TIMES OF AN XOR FUNCTION.

B. Model of a single-layer network

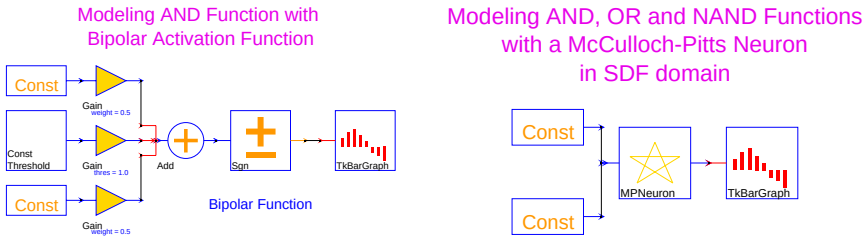
A Hopfield network [3], [7] is a single-layer ANN that is fully connected— each unit is connected to every other unit. The network has symmetric weights with no self-connections; i.e., $w_{ij} = w_{ji}$, and $w_{ii} = 0$. Figure 4(a) is a three-neuron Hopfield network, and Figure 4(b) is its simulation in Ptolemy. The two diamonds in Figure 4(b) are the initial delays used in SDF models. The flow of data in Hopfield network is not in the same direction, and it is possible for information to flow from one node back to itself through other nodes. Therefore, the Hopfield network is known as a feedback or recurrent network. The state of the network at any time is given by the node outputs (n_1, n_2, n_3) in Figure 4(a). The weights store a certain pattern, which is called a stable state of the network. The system evaluates in time from any arbitrary starting state to the stable states. In Figure 4(b), no matter where we put the initial delays as a starting state, the network will eventually reach one of the stable states, $(0, 1, 1)$ or $(1, 1, 0)$, which are the "pattern" stored by this network.

C. Model of physiological phenomena

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 [11] in 1943. If a cold stimulus is applied to a person's skin for a very short period of time, the person 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 using SDF 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.

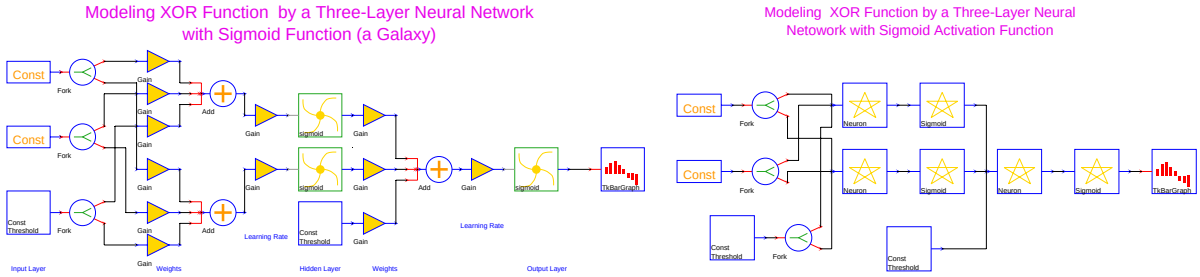
D. Model of a cellular neural network

Cellular neural networks (CNNs) were proposed by Chua and Yang in 1988 [12]. They are a new class of information-processing systems, which are widely used in image processing and pattern recognition. A CNN is an N-dimensional array of elements called cells. A two-dimensional CNN is shown in Figure 6(a). Each cell has multiple inputs and one output and is characterized by an internal state, which



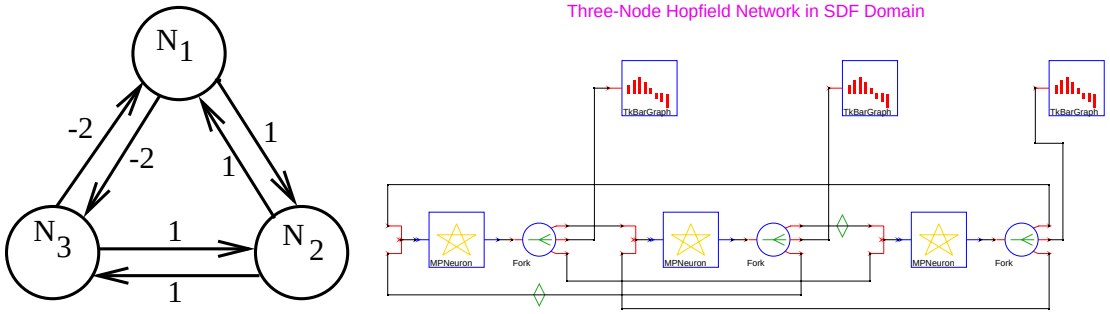
(a) simulation with the existing stars (b) simulation with the new star

Fig. 2. An AND function.



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

Fig. 3. An XOR function.



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

Fig. 4. Single-layer Hopfield network.

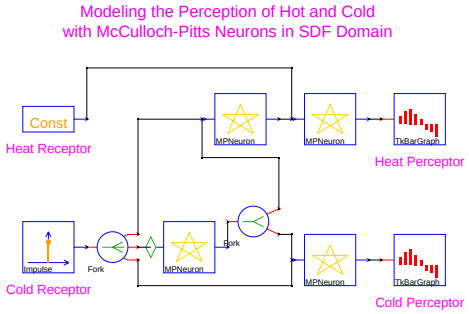


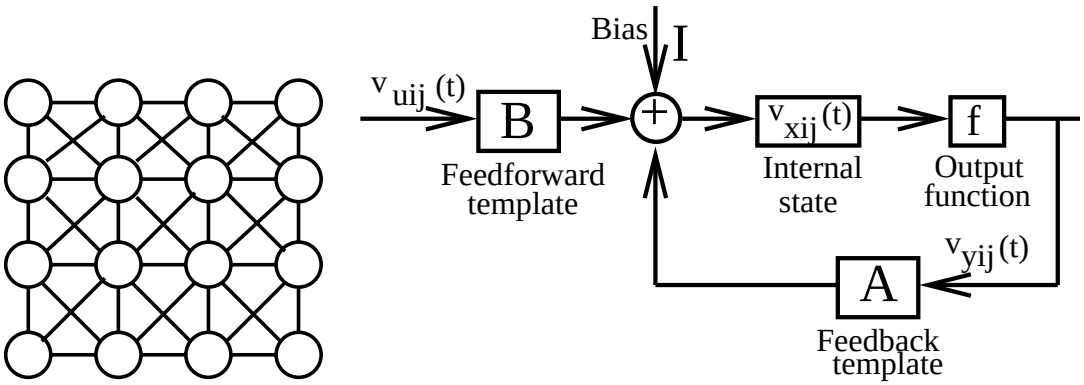
Fig. 5. Simulation of the perception of heat and cold.

is defined by:

$$\dot{v}_{xij}(t) = -v_{xij}(t) + \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl} + I_{ij} + \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl}$$

$$v_{xij}(t+1) = \dot{v}_{xij}(t) + v_{xij}(t)$$

where $v_{uij}(t)$, $v_{xij}(t)$ and $v_{yij}(t)$ are the input, state and output signals of cell $C(i,j)$, respectively. Each cell $C(i,j)$ interacts with the neighbors $C(k,l)$ within a finite neighborhood $N_r(i,j)$. Any arbitrary size of a CNN array is defined by only a few parameters such as A, B and I. A, B and I, which are arranged in matrices, correspond to a feedback template, a feedforward template, and a bias term, respectively. Together, they constitute a cloning template. Figure 6(b) shows a block diagram of a CNN and illustrates that CNNs are recurrent networks. After a finite number of iterations, the output of CNNs will converge to -1 and 1 [12].



(a)Structure of a 2-d CNN (b)Block diagram of a CNN

Fig. 6. Structure and block diagram of a cellular neural network.

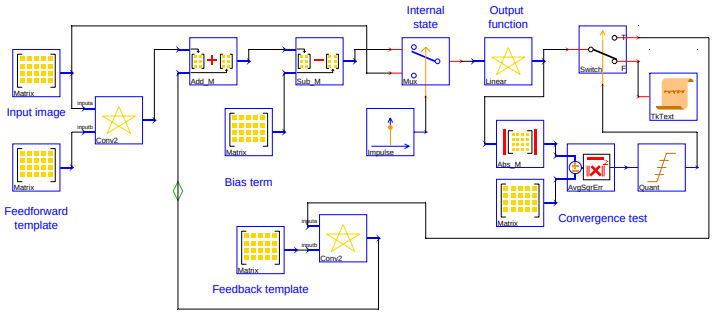
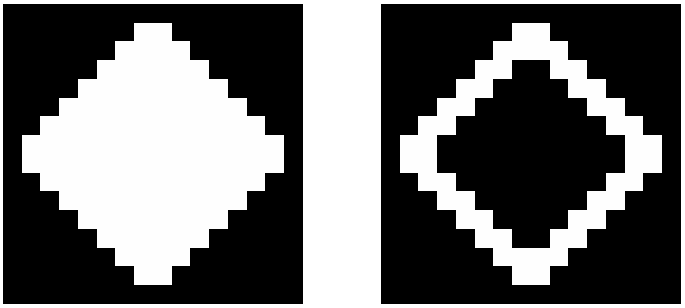


Fig. 7. A CNN for edge detection.



(a) input image to CNN (b) output edge image from CNN

Fig. 8. Input and output images for CNN edge detection.

The execution of CNNs is different from the execution of other ANNs. Although the weights in ANNs and the cloning templates in CNNs are precalculated, CNNs (unlike other ANNs) require data-dependent iteration. Thus, the flow of data in CNN graphs is dependent on values of the data. Therefore, the SDF model cannot model CNNs. An example of a CNN for edge detection [12] is shown in Figure 7, which is modeled by mixing SDF and Boolean Dataflow (BDF) models. BDF models are SDF models plus "Switch" and "Select" actors to implement if/then/else statements and while loops, i.e., to allow the flow of data to depend on the values of the data. A binary diamond image in Figure 8(a) is the input to the CNN, and the edge map with -1 and 1 in Figure 8(b) is the output from the CNN. We can change cloning templates to simulate different CNNs.

V. Synthesis of neural networks in CGC domain

Ptolemy 0.6 and 0.7 realize models of computation as simulation domains, and code generators for a subset of the dataflow models as code generation domains. Both domains execute a system by invoking the setup methods for each block, schedule the resulting system, invoke the begin method for each block, run the system according to the schedule, and then call a wrapup method for each block. In a simulation domain, the run method of the blocks implement the functionality of the block using C++ code. In a code generation domain, the run method (which is called only once) generates code to implement the functionality of the block. Ptolemy 0.7 supports code generation in C, C++, assembly (Motorola 56000 and Texas Instruments

TMS320C50), and VHDL. Next, we describe synthesizing SDF graphs for neural networks in C code.

We generate the stars with the same names and the same parameters in CGC domain as in SDF domain so that it is easy to move from one domain to another in Ptolemy. We retarget all the examples in Figures 2, 3, 4 and 5 from SDF domain to CGC domain. Therefore, the schematics in CGC domain are exactly the same as those in SDF domain.

VI. Conclusions and future work

The similarities between ANNs and SDF graphs make it possible to model ANNs using SDF models. The development of massively parallel implementations of ANNs is a complicated process. Since SDF captures only the data dependencies, parallel scheduling algorithms can be applied.

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, in which the number of tokens produced (or consumed) is one per arc. A novel class of neural networks, CNNs, is simulated by SDF and BDF models. All of the weights in ANNs and the cloning templates in CNNs are precalculated and transferred to the models. Training ANN has not been implemented in Ptolemy, but there are several methods to train neural networks in the system level. One way is to use the Ptolemy interface to Matlab to complete the training in Matlab (Figure 9). The second method is to generate the stars for the different training methods. The third approach is to train the different topologies of ANNs by using SDF and BDF in Ptolemy.

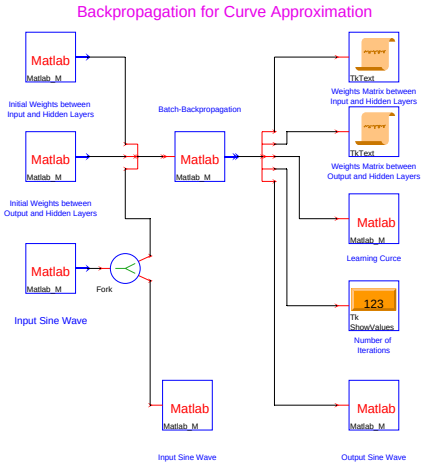


Fig. 9. Training of a backpropagation network by combining Ptolemy and Matlab.

This paper demonstrates system-level design of heterogeneous systems using neural networks. By using dataflow models, we can combine neural networks with signal processing, communications, and control algorithms in a formal, predictable way. Although the example systems in this paper are relatively small, the result scale to larger networks by using either higher-order functions as a graphical syntax to describe regularity and multidimensional SDF semantics to describe interconnected graphs. We show that ANNs can generally be modeled using SDF, except for Turing equivalent CNNs which require BDF.

VII. Acknowlegement

We would like to thank the Ptolemy team at the University of California at Berkeley for releasing the new neural stars in the June, 1997, release of Ptolemy 0.7. More information about the Ptolemy project can be obtained from

`http://ptolemy.eecs.berkeley.edu/`

Some of the demonstrations mentioned in this paper are available in Ptolemy 0.7. We are planning to submit the other demonstrations for the next release of Ptolemy.

References

[1] S. S. Battacharyya, P. K. Murthy, and E.A. Lee, *Software Synthesis from Dataflow Graphs*, Kluwer Academic Publishers, Norwell, MA, ISBN 0-7923-9722-3, 1996.

[2] L. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ 07632, ISBN 0-13-334186-0, 1994.

[3] C. C. Wang and B. Shirazi, "Neural networks implementation on a parallel machine," *Proc. SPIE Applications of Optical Engineering*, vol. 1396, pp. 209-213, Sep. 27–28, 1990, Rosemont, IL.

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

[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 Trans. on Software Engineering*, vol. 18, no. 7, pp. 613-623, Jul. 1993.

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

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

[8] P. L. M. Liu, W. K. Kan, and W. Wang, "A specification approach to artificial neural network design," *IEEE Region 10 Conf. on Comp. and Comm. System*, pp. 40-44, Sep. 24-27, 1990, Hong Kong.

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

[10] The Ptolemy Team, *Ptolemy 0.7 User's Manual*, vol I. of *The Almagest*, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720-1770. http://ptolemy.eecs.berkeley.edu/papers/almagest/.

[11] 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.

[12] L. O. Chua and L. Yang, "Cellular neural networks: theory and applications", *IEEE Trans. on Circuits and Systems*, vol. 35, no. 10, pp. 1257-1290, Oct. 1988.