Hybrid Pipeline Structure for Self-Organizing Learning Array

Janusz A. Starzyk, Mingwei Ding, Yinyin Liu

School of Electrical Engineering & Computer Science
Ohio University, Athens, OH 45701
starzyk@bobcat.ent.ohiou.edu

Abstract. In recent years, many efforts have been put in applying the concept of reconfigurable computing to neural networks. In our previous pursuits, an innovative self-organizing learning array (SOLAR) was developed. However, traditional multiplexer method to achieve reconfigurable connection has its limit for larger networks. In this paper, we propose a novel pipeline structure, which offers flexible, possibly large number of dynamically configurable connections and which utilizes each node's computing ability. The hardware re-sources demand of the proposed structure is a linear function of the network size, which is especially useful for building a large network that can handle complicated real-world applications.

1 Introduction

Reconfigurable computing has become an attractive research topic during past decade due to its good tradeoff between performance and flexibility [1][2][3]. At the same time, a significant effort has been made to introduce reconfigurable computing in hardware implementation of neural networks [4][5][6]. A novel data-driven self-organizing learning array (SOLAR) was proposed in [7] with the aim to develop hardware structures for machine intelligence. Our ultimate goal is to build a modular 3D SOLAR system consisting of hundreds or thousands of neurons.

In our previous work [8][9], a dynamic reconfigurable hardware implementation of the SOLAR algorithm was constructed based on Xilinx picoBlaze core [10]. This re-quires significant amount of silicon dedicated to wiring. Since the number of possible connections among n nodes grows at the level of $O(n^2)$ and the average wire length increases at a level of $O(n^{0.5})$ [11], the total design area occupied by wires grows at a level of $O(n^{2.5})$. In addition, the growing network size requires increasing number of wires to configure. To solve this problem, a new wiring structure is needed to achieve the same reconfigurability with less global wiring.

In this paper, we propose a novel hybrid pipeline structure focused on dynamically changing connectivity. The basic idea here is to utilize the computing ability of each node in the network to perform "soft" connections. The advantage of this structure is that the connections between nodes are fully configurable inside the corresponding nodes, thus saving global wiring, and avoiding complicated routing algorithm.

The rest of this paper is organized as follows: section II gives detailed description of proposed structure and pipeline dataflow, including the node structure that implements the “soft” connections; section III presents the simulation results; section IV concludes the paper with a summary of the proposed structure and future works.

2 Hybrid Pipeline with Sequential Channel

2.1 SOLAR Overview

SOLAR is a regular, 2D/3D array of identical processing neurons, connected to programmable routing channels. Each neuron chooses its input signals from the adjacent routing channels and chooses its arithmetic function to perform and sends its out-put signals to the routing channels. The optional arithmetic functions that neurons per-form include half, identity, logarithm, exponent, sigmoid, addition and subtraction in current design. The functions and connections are reconfigurable as a result of learning. The SOLAR implementation presented in this paper employs a feed-forward structure with all the neurons arranged in a 2D array as depicted in Fig.1.

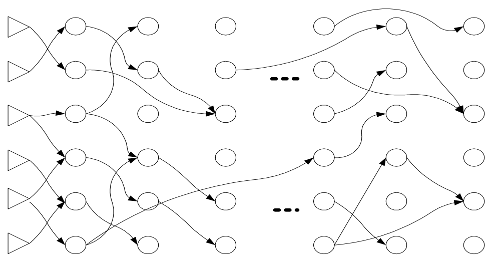

Fig. 1. SOLAR Structure Overview

A SOLAR structure in many ways resembles the organization of a CNN [12]. Like CNN, its architecture is defined by an array of identical cells, which adapt their behavior to the input data. Its neurons are cellular automata, which can be programmed to perform different computational tasks based on data received from its neighbors. Neurons can be either static or dynamic, as defined in [13], depending on the type of equation solved. However, unlike in a CNN, its connectivity structure is not fixed. In a CNN, the interconnect structure is defined by templates which limits its learning ability, while in a SOLAR the interconnect structure is an element of learning and can by dynamically changed even during the network’s operation. Thus a CNN can be considered as a special case of SOLAR structure.

An efficient and flexible routing scheme plays a key role in implementing SOLAR. The following sections detail a new hybrid pipeline structure to address this issue.

2.2 Hybrid Pipeline Structure Overview

The whole network is constructed in a rectangular 2D array. Each column of the array consists of a long shift register and several processing nodes (neurons) attached to the shift register to perform read/write operations. Each processing node has four different working modes: idle mode, reading mode, processing mode and writing mode. The long shift register implements routing channel through which all neurons are connected to primary input nodes or other neurons. Each input data item is first repeated several times, and then fed to the network sequentially. After the first column finishes data processing, the results are written back to the shift registers and then the data in shift register are shifted to the next column through routing channel. The next column will work on the data from the routing channel while the previous column starts processing new input data. Therefore, we call this organization a hybrid pipeline structure illustrated in Fig.2.

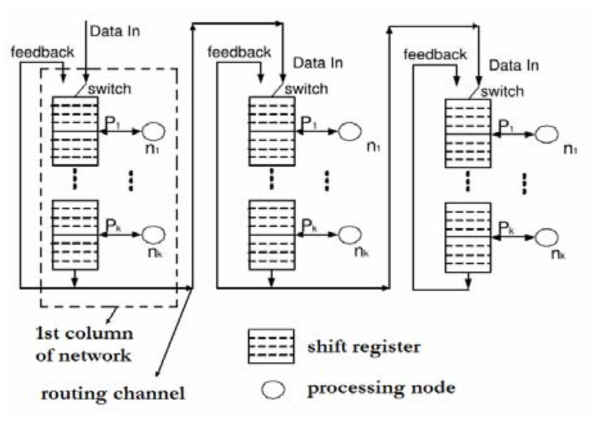

Fig. 2. Pipeline structure overview

Before we continue the detailed description of the structure and its data operations, let's first introduce some terminology to describe the network parameters:

c – input data copy ratio, defines how many times a data item is repeated before the next data is fed. The copy ratio only applies to the 1st column of the network.

k – the number of processing nodes in each column,

N – the number of input data items,

L – the total length of one column of shift register, $L = c \times N$,

R – the input range for a node (designed to be the same for all the nodes). It specifies the maximum range of nodes from the previous layer a node can reach.

$\{P_1, P_2, \ldots, P_k\}$ – A vector of length *k*, the i^{th} element of this vector specifies the read/write position for i^{th} node in the current column.

In the existing design of SOLAR, each neuron can possibly read the data from any other neuron in the previous layers of the network through the routing channel and write the processed results back to the same slots of the shift register. Repeating each data item c times in the 1st column statistically provides neurons the opportunity to read the data across several layers. On the other hand, if the SOLAR structure is strictly hierarchical, there is no need to connect across several layers and then c is set to 1. The copy ratio *c* should increase with the average number of neuron inputs. Also, it affects the effective size of the routing channel. *k* is usually related to *N* in 2D design, and *R* determines the neighborhood size of each neuron. For locally connected neurons, *R* is small. The optimum choices of design parameters *c*, *k*, *R*, etc. are application specific and will not be discussed in this paper. In the next section, a more detailed description of the dataflow in the implemented interconnect scheme is given.

2.3 Data flow Description

First of all, we base the timing circuitry on the clock that drives the shift registers. All the operations are synchronized to the rising edge of this clock. The operation of this structure is pipelined from column to column.

All the nodes are in idle mode before P_k cycle, and then the nodes begin reading data from certain slots. All the input data will be shifted to the 1st column after *L* cycles. At cycle *L*, the switch at the top of the column switches to feedback position and the data begin to circulate in the 1st column. All the nodes enter the processing mode after reading the input data, and should finish their computing tasks no later than at $L + P_k$ cycle (since if the longest combined reading time and computing time of all the nodes exceed *L* cycles, additional *L* cycles need to be added to complete these two operations), when all the nodes begin to write their processed result back to the specified slots. By the end of cycle $2L + P_k$, the nodes should finish writing and enter the idle mode.

At the arrival of *3L* cycle, the switch switches back to data and the next *N* input data begin feeding into the 1st column, and the content of the 1st column is copied to the 2nd column. From above description, we can conclude that the pipeline delay between two columns is *3L* cycles.

This computing scheme in which pipeline data is transported sequentially will yield performance slightly lower than full parallel hardware implementation, but significantly higher than sequential operation on a single processor. In general, if a node requires *p* cycles to process the input data, then fully parallel implementation requires *p+r* cycles, where *r* is the average number of neuron's inputs. The proposed pipeline scheme requires *2L+[P/L]L* cycles, while sequential implementation requires *Np+r* cycles to complete. For large *p*, the performance of the proposed pipeline structure is similar to the performance of fully parallel implementation, while for small *p*, its performance is a function of the channel size. Therefore, it is a good compromise between the hardware cost and performance.

2.4 Node Operations

In our design, each node is implemented with Xilinx picoBlaze core processor. To correctly operate, the node must be running at a higher speed than the shift register, because during one shift register clock period, the node needs to read timing information and to read data at that time if necessary. For a clear description of the operation, the period of that higher speed clock is denoted as node-cycle.

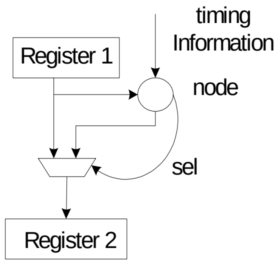

Fig. 3. Single Node Read/Write Structure

After a node finishes reading all the slots of required data, it begins working in the processing mode. Based on previous description, the node has $L-c\cdot R$ cycles to perform its computing task. Fig. 3 illustrates how each processing node is attached to the shift registers to perform read/write operations. Register 1 and register 2 are parts of the routing channel of the hybrid pipeline structure. The *sel* signal is always switched to let data flow from register 1 to register2 except when the node decides to output its processed result.

As stated above, the picoBlaze core is running at a frequency higher than the shift register clock and the ratio between the two clocks can be denoted as *m*. Based on previous analysis, node's computation time must be a multiple of *m* in order for the whole pipeline to work correctly. Thus, optimization of the computing time is necessary, especially for complicated operations, like exponent, logarithm, sigmoid, etc.

3 Simulation Results

In this section, we will first give the simulation results of a single node performing read/write and an *add* operations. Then a 4-row array is built to read and process Iris database. At last, four different sizes arrays are constructed and their design areas are compared.

The respective read/write waveform for a single node is illustrated in Fig. 4. The node is configured to read data from slot 4 and 5, and perform a modified *add* operation, and then write the results back to slots 4 and 5. As we can see, the node read the data value 47 and 57 at slot 4 and 5. According to the modified add function designed for our network, the processed result should be (47/2+57/2), which equals to 51. Then the processed result 51 is output by the node when it "sees" slot 4 and 5 again.

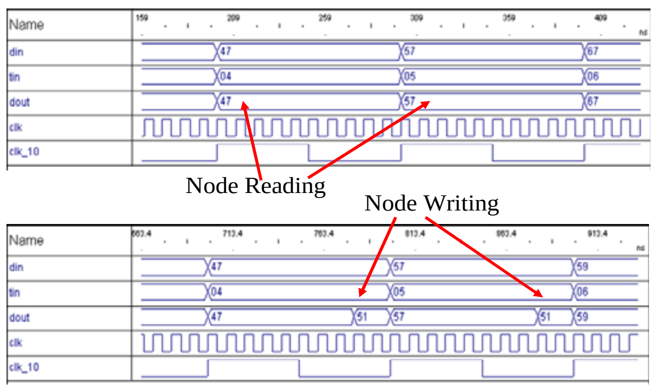

Fig. 4. Single Node Read/Write Waveform

As an example, a 4x3 SOLAR array with pseudo-random connections was built to process data from Iris database [14]. The training set consists of 75 samples with 4 features each. The connections and arithmetic functions chosen by neurons after learning are shown in Fig. 5a).

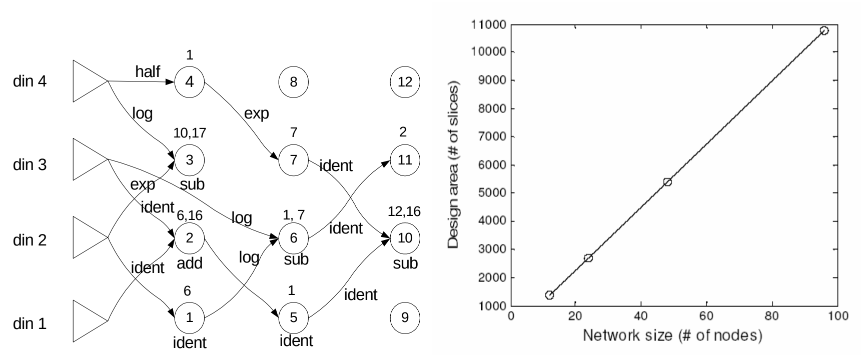

Fig. 5.a) 4x3 Array processing 4-feature data, b) Design area vs. network size

In Fig.5a), the numbers on the top of individual nodes are the neurons' processed results. The names on the arcs specify which arithmetic functions this node performs. The final output is read from the last column and is verified with Matlab simulation results. It is follows that using the proposed hybrid pipeline structure, the network implemented in FPGA can correctly perform all calculations.

To demonstrate the proposed pipeline approach advantages for larger-sized network, additional arrays of 4x6, 4x12 and 4x24 were constructed and synthesized targeting Xilinx Virtex II chip. The design area (number of slices) is observed to be a linear function of the network size (number of nodes), as shown in Fig.5b), while the maximum system clock is consistently kept at 81.1 MHz.

4 Conclusion and Future Works

In this paper, a novel hybrid pipeline scheme is presented and its flexibility in dynamic connectivity configurations is demonstrated by checking the hardware response with the simulation results. This structure is characterized by a linear increase of the hardware resources with respect to the network size. In addition, the proposed structure reduces global wiring to a minimum and shows good modularity, for easy placement and routing task in larger networks. The shift register cost can be mitigated by using Xilinx FPGA technology which possesses abundant resources of registers that can be efficiently implemented by a chain of look-up tables.

The proposed structure has been successfully applied to implement the SOLAR algorithm and has been tested in hardware for Iris database. With its excellent scaling property, it shows potential for implementing larger SOLAR networks in hardware targeting applications like data clustering, image processing, pattern recognition, etc.

At last we would like to point out that some variants of the proposed structure can achieve higher pipeline speed at the cost of limiting the range of input signals that each node may receive.

References

1. Tessier, R. Burleson, W., Reconfigurable Computing and Digital Signal Processing: A Survey, Journal of VLSI Signal Processing, Vol. 28 (2001) 7-27
2. Hartenstein, R., A Decade of Reconfigurable Computing: A Visionary Retrospective, Proceedings of Int'l Conf. on Design, Automation and Test in Europe (DATE'01), Munich, Germany (2001) 642-649
3. Singh, H., Lee, M., Lu, G., Kurdahi, F.J., Bagherzadeh, N., MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation–Intensive Applications, IEEE Transactions on Computers, Vol. 49, No. 5, (2000) 465-481
4. Lindsey, C., Lindblad, T., Neural networks in hardware-A survey, Proc. Conf. Application Science of ANN, Vol.2, (1995) 1194-1205
5. Tempesti, G., Mange, D., et al, The BioWall: an Electronic Tissue for Prototyping Bio-inspired Systems, Proceedings of the NASA/DoD Conference on Evolvable Hardware, Los Alamitos, CA (EH'2002), 221-230
6. Spaanenburg, L., Alberts, R., et al, Natural learning of neural networks by reconfiguration, Proceedings of SPIE, Vol. 5119, Spain (2003) 273-284
7. Starzyk, J.A., Zhu, Z., Liu, T., Self-Organizing Learning Array, IEEE Trans. on Neural Networks, Vol. 16, No. 2 (2005) 355-363
8. Starzyk, J.A., Guo, Y., Dynamically Self-Reconfigurable Machine Learning Structure for FPGA Implementation, Proc. Int. Conf. on Engineering of Reconfigurable Systems and Algorithms, Las Vegas, Nevada (2003) 296-299
9. Starzyk, J.A., Guo, Y., Zhu, Z., SOLAR and its hardware development, Proc. Computational Intelligence and Natural Computing (CINC'03), (2003)
10. Xilinx PicoBlaze soft processor, http://www.xilinx.com.
11. Donath, W.E., Wire length distribution for placement of computer logic, IBM Journal of Research and Development (1981) 152-155
12. Chua, L.O., Yang, L., Cellular neural networks: theory, IEEE Transactions on Circuits and Systems, Vol.35, No.10, (1988) 1257-1272
13. Gupta, M.M., Jin, L., Homma, N., Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, John Wiley & Sons (2003)
14. Iris Database, http://www.ics.uci.edu/~mlearn/MLRepository.html