

The Structure of Boolean Neuron for the Optimal Mapping to FPGAs

Roman Kohut, Bernd Steinbach

Abstract - In this paper, we present a new type of neuron, called Boolean neuron that may be mapped directly to configurable logic blocks (CLBs) of field-programmable gate arrays (FPGAs). The structure and logic of Boolean neuron allow a direct representation of the Boolean neural network architecture to FPGAs. Our approach solves digital design problems especially with respect of the performance and gate count. The additional advantages of Boolean neural networks consist in the reduction of memory space and computation time in comparison to usual neural networks. In the example, we describe the expansion of Boolean output neuron in order to the cascade Boolean neurons with a restricted number of inputs and also the mapping to lookup tables (LUTs).

Keywords - Boolean neuron, hardware neural network, data representation, FPGAs implementation.

I. INTRODUCTION

In the last twenty years the development of artificial neural networks (ANNs) has been greatly influenced by the requirements of different fields of science and technology. High speed, portability, low cost and reliability are required for the realization of neural networks in practical applications.

The majority of ANN applications in commercial use are implemented in software, and run on a conventional single processor general purpose computer. This fact is mainly due to the flexibility of software. However, specialised hardware offers appreciable advantages in several situations [19].

The most common reasons for using specialised ANN hardware are higher speed or lower cost. Parallelism, modularity and dynamic adaptation are three computational characteristics typically associated with neural networks [18]. Hardware implementations of ANNs use these features successfully. Therefore hardware neural networks can offer a considerable potential for speed improvements and reduction of the system costs. For similar reasons to cost reduction, a hardware implementation offers a greater number of components working in parallel. If an application has constraints in physical size or weight, the hardware implementation may be essential.

All hardware ANN implementations can be divided into three categories: digital, analogue and hybrid [19]. There is a large number of technologies that implement of digital neural network. Widely spread examples of them are hardware ANN implementations on FPGAs, VLSI and WSI [13].

Paul Morgan analyses in [13] these three types of digital hardware implementations of RAM-based neural network. In the following we consider FPGAs, because modern FPGAs have a several advantages in comparison to VLSI or WSI circuits which are expensive in the design and production. Hardware ANN implementations using FPGAs are more practicable for the research tasks because the development time is very short and FPGA circuits are cheaper than custom VLSI circuits.

FPGAs have become a viable alternative to custom gate arrays or full custom ASICs. They have increased gate count, functionality and speed. On-board memory is supported by some architecture. Development time and component cost can be dramatically less than a full custom VLSI system. Xilinx SRAM programming technology allows a quick reprogramming of the FPGA which reduces the product delivery times.

II. FIELD-PROGRAMMABLE GATE ARRAYS

FPGAs were originally created to serve as a hybrid device between PALs and Mask-Programmable Gate Arrays (MPGAs). Like PALs, FPGAs are fully electrically programmable. Like MPGAs, FPGAs can implement very complex computations on a single chip, with million gate devices currently in production [6]. FPGAs are becoming increasingly popular for prototyping and the design of complex hardware systems and are often primarily viewed as glue-logic replacement and rapid-prototyping vehicles.

The flexibility, capacity, and performance of FPGA devices have opened up completely new avenues in high-performance computation, forming the basis of reconfigurable computing. Run-time reconfigurable computer systems (RTR) are a promising approach in the designs of new computer systems that can satisfy the high demands of computational applications. Run-time reconfigurable computer systems typically combine a classical microprocessor with programmable logic resources, like field programmable gate arrays [2].

The first FPGA was introduced in 1985 by Xilinx. The structure of a FPGA can be described as an "array of blocks" connected together via programmable interconnections. The amount of logic that each logic block can implement depends on the FPGA family. The main advantage of FPGAs is their flexibility. An engineer can change and refine a design by exploiting the reprogrammability of the device [3].

Xilinx refers to its logic blocks as configurable logic blocks (CLBs) and the FPGA devices themselves as Logic Cell Arrays (LCAs). The typical CLB contains two identical

Roman Kohut, Bernd Steinbach - Institute of Computer Science, Freiberg University of Mining and Technology, Bernhard-von-Cotta-Straße 2, D-09596 Freiberg, GERMANY
E-mail: {kohut, steinb} @informatik.tu-freiberg.de

logic blocks (slices) with lookup tables (LUT), an optional D-flip-flop (DFF) and a additional control logic as shown in Figure 1.

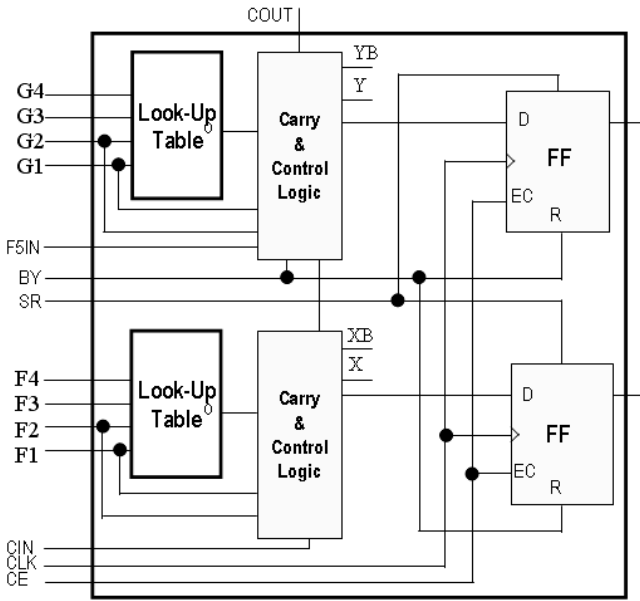


Figure 1. CLB Slice structure

The LUTs can implement any Boolean function of four variables. The D-flip-flops allow the user to produce efficient synchronous designs. It can be used for pipelining, registers, finite state machines, or any other situation where clocking is required. The number of CLBs occupied by a design gives a measure of the magnitude of that design [3 and 6].

III. NEURAL NETWORKS ON FPGAS.

FPGA-based reconfigurable computing architectures are well suited to implement ANNs completely. It can exploit both the fast reconfiguration for a new ANN and the concurrency in run time. FPGAs are cheap, flexible, and may be used for embedded applications (unlike neuroprocessors, which rapidly become outdated) [4].

There has been a lot of recent interest in the FPGA realization of neural networks [5, 8, 9, 14 and 18]. The first successful FPGA implementation [4] of artificial neural networks (ANNs) was published in 1992. Since this time FPGAs was improved to a powerful technical basis for ANN implementation. FPGA 2D-topology does not allow handling the connection complexity of standard neural network models. FPGA realization of ANNs with a large number of neurons is still a challenging task because ANN algorithms require many multiplications and it is relatively expensive to implement multipliers on fine-grained FPGAs [18]. Moreover, FPGAs still implement a limited number of logic gates, whereas neural computations require area-consuming operators as multipliers [4]. Usual solutions handle sequential computations, where the FPGA is used as a small neuroprocessor, or they implement very small low-precision neural networks without on-chip learning. Connectivity problems are not solved even by use reconfigurable FPGAs

with a bit-serial arithmetic [7 and 8], or by the use of small-area stochastic bit-stream operators [1].

Our research focuses on the development of neuron structure for the optimal mapping of neural network on the base of this neuron into FPGA implementation. For the realization of one usual neuron tens or hundreds of CLBs are required. Disadvantages of existing FPGA approaches were described in several papers. Well-known implementation of a simple three layer feed forward network GANGLION [4] needs 640 to 784 CLBs per neuron. Michael Gschwind suggests in [10] a bit-stream approach that requires only 22 CLBs to implement a neuron. In report [16] are used 51 CLBs for implementing a single neuron. A FPGA implementation of a Hopfield neural network working with a 64 neuron configuration was presented in [9 and 17]. Each neuron requires 26 CLBs on an XC4000 series Xilinx FPGA. The whole neural net requires 1664 CLBs.

In order to decrease of CLBs number for modelling of neural networks, we suggest functional changes of the usual neural element and offer this new type of neuron. This neuron should be implemented directly into a single CLB.

IV. BOOLEAN NEURON

In our previous papers [11, 12 and 15] we considered the representation of Boolean functions with neural networks. We stated the disadvantage that input signals and weighting coefficients of the usual neuron are real numbers, less often – integers. The output y is determined by an activation function f , and can be a real, integer or Boolean, but in any case the activation (transfer) function f operates with no Boolean data:

$$y = f(\mathbf{x}, \mathbf{w}), \quad (1)$$

where y – output signal of the neuron,
 $\mathbf{x} = \{x_1, x_2, \dots, x_{N_x}\}$ – input vector of signals,
 $\mathbf{w} = \{w_1, w_2, \dots, w_{N_x}\}$ – vector of synaptic weights,
 N_x – number of inputs,
 f – transfer function.

Using Boolean data for values $\mathbf{x}, \mathbf{w}, y$ is inefficient because of unnecessary memory expenses. The same inefficiency is valid for the mapping of such neurons to FPGAs because elements of FPGAs operate with Boolean signals.

In order to overcome these problems we propose Boolean neural networks based on Boolean neural elements. Both the suggested Boolean neuron and whole Boolean neural network (BNN) operates only with Boolean signals and uses only Boolean operations. The output signal of the Boolean neuron is defined as:

$$y = f_B(\mathbf{x}_B, \mathbf{w}_B), \quad (2)$$

where the index B indicates Boolean values and

$$\mathbf{x}_B = \{x_1, x_2, \dots, x_{N_x}\}, \quad x_i \in \{0, 1\},$$

$$\mathbf{w}_B = \{w_1, w_2, \dots, w_{N_x}\}, \quad w_i \in \{0, 1\},$$

f_B - Boolean dependence such as a Boolean transfer function,
 $f_B, y \in \{0,1\}$.

The change from a classical neuron to Boolean neuron with Boolean transfer function f_B reduces time for converting input signal vector into output signal significantly. An additional advantage of the Boolean neuron consists in the reduction of necessary memory size.

Remember, the basic elements of FPGAs are CLBs that contains two LUTs. Each LUT has 4 inputs and 1 output and can realize any Boolean function depending on the 4 Boolean variables. The general structure of lookup table is shown in Figure 2.

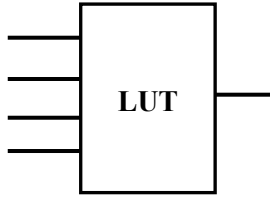


Figure 2. General LUT structure

On the other hand the structure of the Boolean neuron (see Figure 3) with 4 inputs has same structure as lookup table shown in Figure 2.

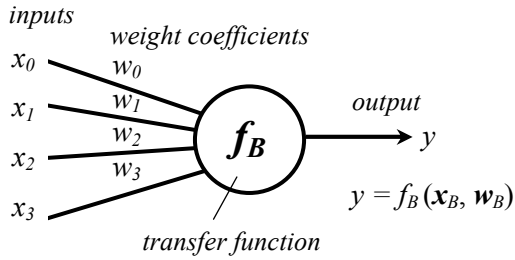


Figure 3. General structure of Boolean neuron

Consequently the limitation of a Boolean neuron to 4- or 5-inputs in the Boolean neural network allows a direct representation of neural network to FPGAs. Any Boolean function that depends on 4 variables, can be mapped to a single LUT and any Boolean function of 5 variables, can be mapped to a single CLB. Note, the behavioural complexity of a Boolean neuron, having such a restricted number of inputs, does not influence both the structural expenditure and the working speed.

Assume, if there is Boolean function f depending on n variables, then a restriction to m -inputs of Boolean neuron (where $m < n$) increases the required number of such neurons for the representation of the function f . In this case the Boolean neural network contains more than one Boolean neuron. For that reason the number of layers of BNN may be increased.

The advantage of Boolean neurons is that only one LUT is required for the hardware implementation of Boolean function f of $m = 4$ inputs. A single CLB can realize even a Boolean function f of $m = 5$ inputs. This is huge win compared to

existing FPGAs implementation of the classical neural networks [4, 9, 10, 16 and 17].

V. BOOLEAN NEURAL NETWORKS

In this section we do not stress the synthesis methodology of neural network structure on the base of Boolean neuron because it is described in our previous papers [11, 12]. Instead we explain details of a very promising approach of FPGA implementation of Boolean neural networks.

The earlier suggested learning algorithms for BNNs [12] allow the restriction of the number of inputs for Boolean neurons on the hidden layer of the BNN. The input number of BN on the output layer of BNN can be restricted to the number of inputs of the LUT of the FPGA by a transformation of output neuron into a cascade of Boolean neurons. For that the laws of superposition are used.

The equation (3) shows an example of the realization of Boolean function y that is decomposed to basic Boolean functions $k_0 \dots k_{10}$.

$$y = k_0 \oplus k_1 \oplus k_2 \oplus k_3 \oplus k_4 \oplus k_5 \oplus k_6 \oplus k_7 \oplus k_8 \oplus k_9 \oplus k_{10} \\ = f_0 \oplus f_1 \oplus f_2 \quad (3)$$

where \oplus is the Boolean operation EXOR and

$$f_0 = k_0 \oplus k_1 \oplus k_2 \oplus k_3, \\ f_1 = k_4 \oplus k_5 \oplus k_6 \oplus k_7, \\ f_2 = k_8 \oplus k_9 \oplus k_{10}.$$

Since the number of the Boolean subfunctions ($k_0 \dots k_{10}$) is larger as 4 (number of inputs in lookup table), we split Boolean functions k_i into three groups, where each of them includes at most 4 functions. The connection of functions by Boolean operation EXOR in a single group creates a new function f_j which can be realized by a single LUT. The connection of these f_j - functions by Boolean operation EXOR creates initial Boolean function y that can be realized again by a single LUT, because the number of f_j - functions is less then 4.

In this example the EXOR-decomposition was determined, but in general the decomposition operation can be OR, AND or "equivalence". Type of decomposition must be selected at the start of training algorithm of BNN [11].

The presentation of the Boolean function y by Boolean neuron on the output layer of BNN is shown in the top of Figure 4. According to (3) Boolean neuron is expanded to two-layer architecture as shown in bottom of Figure 4. In fact the expand boundary of the architecture is unlimited. Using the laws of superposition a multi-layered architecture of BNN will be created.

The most important result is a possibility to use the Boolean neuron for the synthesis of different structures of Boolean neural networks. As shown in our previous papers [11 and 12], the using of the Boolean neural networks for representation of Boolean function allows to determine a

special decomposition of Boolean functions sets into a small number of unique basic functions. It must be emphasized that the learning procedure of a BNN solves the very difficult decomposition task for a set of Boolean functions by adapting the FTFS approach of classical neural networks.

The implicit representation of Boolean function sets extends the possible approaches in circuit design significantly.

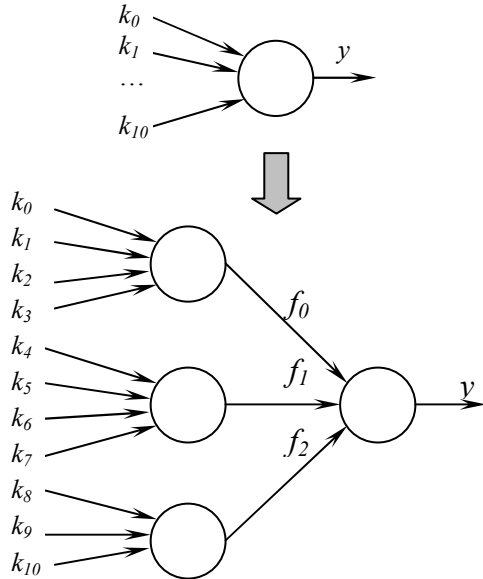


Figure 4. Representation of a Boolean function by a single Boolean neuron with 10 inputs or an equivalent two layer architecture of Boolean neurons with 4 inputs

The Figure 5 illustrates the mapping of the cascade of Boolean neurons with 4 inputs to the structure of LUTs of a FPGA. This structure of Boolean neurons realizes the Boolean function y described in (3).

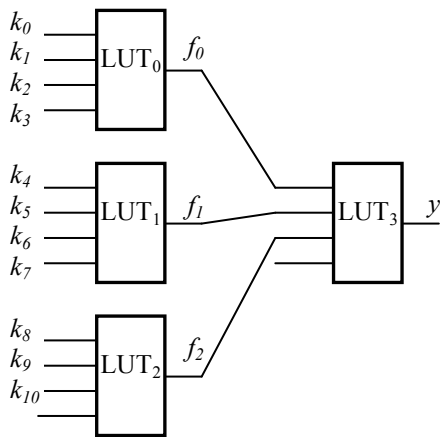


Figure 5. Mapping of Boolean neurons with 4 inputs to LUTs of a FPGA.

VI. CONCLUSION AND FUTURE WORK

The most common artificial neural network (ANN) applications in commercial use are implemented in software. But hardware implementations of ANN offer advantages in terms of high speed, portability, low price and reliability. FPGA-based reconfigurable architectures are well suited to implement artificial neural networks. However the direct mapping of classical neural networks into FPGAs structures needs a large number of logic gates for each neuron.

In this paper we suggested a new type of neural elements, called Boolean neuron, for modelling of neural networks in order to decrease the required number of configurable logic blocks (CLBs). We described the structure and mathematic definition of the Boolean neuron in detail. Boolean neurons allow a optimal mapping of neural network to FPGAs. The complete logic of a Boolean neuron is mapped directly into a lookup tables (LUT). A configurable logic block (CLB) of an FPGA includes two LTUs. Thus, a CLB can realize two Boolean neurons which are controlled by 4 inputs each.

The Boolean neuron can be used for the synthesis of various structures of neural network. We cite the Boolean neural networks (BNNs) as one of possible neural networks types that are based on Boolean neuron. The proposed Boolean neural networks can be used for compact presentation and fast calculation of Boolean functions. Using such Boolean neurons we could decrease the calculation time for both training and using the BNN significantly. An additional advantage of the Boolean neuron consists in the reduction of necessary memory size.

We have shown an approach to restrict the inputs number of Boolean neurons on the output layer by a using of superposition laws. Corresponding illustrations of the output neuron transformation to a cascade of Boolean neurons and further to LUTs structure are given.

Due to the limitation of a 4 inputs structure of Boolean neurons in the Boolean neural networks, efficient FPGA implementations of BNNs, in terms of performance and gate count are proposed. In comparison to classical neural networks the total number of neurons in the Boolean neural network can be increased, but because only one LUT is required for Boolean neuron the common effort can be decreased. That is huge benefit compared to existing approaches of neural network implementations on FPGAs. The suggested Boolean neuron extends the possible approaches of neural network implementation in circuit design significantly.

In the future, we continue our research for optimal Boolean functions presentation and calculation using hardware neural networks. Based on the results described in this paper we will design and develop a mapping methodology for any structures of Boolean neural networks both without on-chip learning and with on-chip learning. For that try use an UML-based approach of hardware-software co-design of run-time reconfigurable architectures.

REFERENCES

- [1] Bade S.L. and Hutchings R.L.: FPGA-based stochastic neural networks-implementation. In Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, pp. 189-198, 1994.
- [2] Beierlein, T; Fröhlich, D.; Steinbach, B.: UML-Based Co-Design for Run-Time Reconfigurable Architectures. in: Proceedings of the FORUM on Specification and Design Languages, September 23 - 26, 2003, Frankfurt, Germany, pp 285 - 296.
- [3] Blake J.J., Maguire L.P., Mc Ginnity T.M., Roche B., and Mc Daid L.J., "The Implementation of Fuzzy Systems, Neural Networks and Fuzzy Neural Networks using FPGAs", Information Sciences: An International Journal, Elsevier, 112 (1998), pp151-168.
- [4] Charles E.Cox and W.Ekkehard Blanz, GANGLION – a fast field-programmable gate array implementation of a connectionist classifier. IEEE Journal of Solid-State Circuits, 27(3):288-299, March 1992.
- [5] Cloutier J., Cosatto E., Pigeon S., Boyer F. and Simard P.: VIP:an FPGA-based processor for image processing and neural networks. In Proc. MicroNeuro, pp. 330-336, 1996.
- [6] Compton K. and Hauck S.: An Introduction to Reconfigurable Computing, Invited Paper, IEEE Computer, April, 2000
- [7] Eldredge J. G. and Hutchings B. L.: RRANN: a hardware implementation of the backpropagation algorithm using reconfigurable FPGAs. In *Proceedings of the IEEE World Conference on Computational Intelligence*, 1994.
- [8] Girau B. and LORIA, Dependencies of composite connections in Field Programmable Neural Arrays, NeuroCOLT2 Technical Report Series NC-TR-99-001, June, 1999
- [9] Gschwind M., Salapura V. and Maischberger O.: A Generic Building Block for Hopfield Neural Networks with On-Chip Learning. IEEE International Symposium on Circuits and Systems, Atlanta, GA, May 1996.
- [10] Gschwind M., Salapura V. and Maischberger O.: Space efficient neural net implementation. In Proc. of the Second International ACM/SIGDA workshop on Field-Programmable Gate Arrays, Berkeley, CA, February 1994.ACM.
- [11] Kohut, R.; Steinbach, B.: Decomposition of Boolean Function Sets for Boolean Neural Networks. in: Steinbach, B. (Hrsg.): Boolean Problems, Proceedings of the 5th International Workshops on Boolean Problems, 23. - 24. September 2004, Freiberg University of Mining and Technology, Freiberg, 2002, pp. 191 – 206.
- [12] Kohut, R.; Steinbach, B.: Boolean Neural Networks. Transactions on Systems, Issue 2, Volume 3, April 2004, pp. 420 – 425.
- [13] Morgan P., Ferguson A. and Bolouri H.: Cost-performance analysis of FPGA, VLSI and WSI implementations of a RAM-based neural network. Proc. 4th IEEE Int. Conf. on Microelectronics for Neural Networks and Fuzzy Systems (MicroNeuro'94), 235-243, 1994
- [14] Salapura V., Gschwind M. and Maischberger O.: A Fast FPGA Implementation of a General Purpose Neuron, in: R. Hartenstein, M. Servit(Eds.), "Field-Programmable Logic: Architectures, Synthesis and Applications -- 4th International Workshop on Field Programmable Logic and Applications", Lecture Notes in Computer Science 849, Springer Verlag, Berlin, 1994.
- [15] Steinbach, B. Kohut, R.: Neural Networks – A Model of Boolean Functions. Proceedings of 5th International Workshop on Boolean Problems, Freiberg, Germany, September 19-20, 2002;
- [16] Xilinx.. XACT Reference Guide, Xilinx, San Jose, CA, 1992.
- [17] Xilinx. The Programmable Logic Data Book. Xilinx, Inc., San Jose, CA, 1993.
- [18] Zhu J. and Sutton P.: FPGA Implementations of Neural Networks – a Survey of a Decade of Progress. In *Proceedings of 13th International Conference on Field Programmable Logic and Applications (FPL 2003)*, Lisbon, Sep 2003.
- [19] The Italian Managing Node of NeuroNet, <http://www.dibe.unige.it/neuronet/>