Computer Science and Engineering 2012, 2(7): 133-137
DOI: 10.5923/j.computer.20120207.02

A Multi-Layer Perceptron SoC for Smart Devices

Keuntak Yang[1], Bongkyu Lee[2,*]

[1]Dept. of computer science and Statistics, Jeju National University
[2]Dept. of computer science and Statistics, Jeju National University, corresponding author

Abstract This paper presents a programmable System-On-a-chip for various embedded applications that need Neural Network computations. The system is fully implemented into Field-Programmable Gate Array (FPGA) based prototyping platform. The SoC consists of an embedded processor core and a reconfigurable hardware accelerator for neural computations. The performance of the SoC is evaluated using real image processing applications, such as optical character recognition (OCR) system.

Keywords SoC, FPGA, MLP, Hardware Accelerator

1. Introduction

The demand for 'smart' devices in consumer electronics is increasing. This is motivated by the wide spread use of low-cost embedded electronics in various environments[1]. Also, it is desirable that electronic devices are capable of sensing and understanding their surroundings and adapting their services according to the context. Artificial Neural Networks (ANN) have been spot-lighted for this purpose, primary due to their wide range of applicability[2]. The Multilayer Perceptron (MLP) is the most frequently used ANN due to its ability to model non-linear systems and establish non-linear decision boundaries in classification problems such as optical character recognition (OCR), data mining and image processing/recognition[2].

However, since MLP requires extremely high throughput, this computational complexity is highly undesirable from real time operations for embedded devices which have constraints in their processing capabilities. An attractive solution to this is to design a dedicated hardware for MLP acceleration[3].

Hardware implementation of MLP has been a hot topic for many years, mainly due to accuracy, required space, and processing speed. Various hardware implementations for MLP were successful using entire hardware implementation such as ASIC design method[3,4,5]. However, full hardware implementation is not effective in terms of cost and implementation complexity. Recently, the reconfigurable computing paradigm is a topic of active research. Utilizing the capability of reconfigurable devices, the implementation of MLP structures in FPGA has been wide spreaded[6].

* Corresponding author:
bklee@jejunu.ac.kr (Bongkyu Lee)
Published online at http://journal.sapub.org/computer

Although a few hardware implementations using FPGA have been proposed thus far[7,8,9], a hardware implementation of an MLP still remains to be a challenging problem for embedded applications. Since different pre-processing and post-processing techniques can be combined with an MLP in real applications, the system should be reconfigured according to applications. Also, there have been strong needs for hardware design which can accommodate variations in network structure without hardware redesign[10]. These problems can be overcome by software/hardware co-design method. This method is carried out by analyzing the timing of the different portions of the algorithm and implementing the time extensive parts on hardware[11]. A SoC that has a microprocessor and related configurable hardware accelerators can deliver large speedups, while keeping the flexibility of software models.

In this paper, we implement a novel MLP-SoC architecture for smart applications into embedded devices. For testing and debugging the target architecture in the register transfer level (RTL) efficiently, an FPGA based prototyping platform is designed and implemented. The implemented SoC can accommodate variations in network structures and applications without hardware modification. To evaluate the SoC, an OCR system is built on the prototyping platform where the SoC is implemented. The experimental result proves the effectiveness of the SoC in terms of both speed and recognition rate.

2. Issues in the Implementation of MLP-SoC

Our goal is to implement a MLP-SoC which can be used for embedding processes. During the MLP-SoC implementation, a prototyping platform, data representation and precision and hardware components play important roles in design decisions.

2.1. The Prototyping Platform

To design and verification of SoC, FPGA-based prototyping platform has become popular in co-verification and rapid prototyping[12]. Mapping the entire design of the target SoC into an FPGA gives an accurate and fast representation. Some basic components, including CPU, bus system and associated interconnection blocks, are selected for designing the platform. LEON 2[13] is selected for the programmable processor and implemented into the FPGA. For the communication between internal components, on-chip AHB/APB AMBA bus system is also implemented into FPGA.

In addition to the FPGA chip, the platform offers the SDRAM-based memory (128 Mbytes) and the flash-based storage (8 Mbytes). SDRAM-based memory unit is used for storing external data, while flash-based storage is used for storing software blocks. Fig. 1 shows the implemented prototyping platform having compact size of 112x129mm.

Figure 1. The prototyping platform

2.2. MLP for Image Processing/Recognition

A MLP for image processing/recognition application consists of processing elements arranged in layers. Typically, it requires three or more layers of processing nodes: an input layer, one or more hidden layers, and an output layer. Every processing node in one particular layer is fully or partially connected to every node in the layers above and below it. The weighted connections define the behavior of the network and are adjusted during training through a supervised training algorithm called back-propagation[2].

In the recognition, an input vector is presented to the input layer. For successive layers, the input to each node is the sum of the scalar products of the incoming vector components with their respective weighted connections:

$$\text{sum}_i = \sum_j w_{ij}\ \text{out}_j \qquad (1)$$

where w_{ij} is the weight connecting node j to node i and out_j is the output from node j.

The output of a node i is $out_i = f(sum_i)$, which is then sent to all nodes in the next layer. This continues through all the layers of the network until the output layer is reached and the output vector is computed, where f denotes the activation function of each node. A sigmoid or a hyperbolic tangent function is frequently used. Table 1 shows the hardware constraints for the target SoC.

Table 1. Target hardware constraints

Parameters	Values
Maximum # of input nodes	1,000
Maximum # of hidden nodes and layers	128/2
Weight precision	12 bits (signed)
Activation function output precision	9 bits (signed)
Input data range	0 ~ 255
Output range	-255 ~ 255

Since a floating-point representation of data (weights, inputs, outputs) in a neural network may still be impractical for embedded hardware, we use fixed point representations for weights, inputs and outputs. Unsigned 8 bits are used for represent input values, while signed 9 bits are used for output precisions since some activation functions produce negative outputs. Weights are stored in the weight table using signed 12 bits fixed-point representations. The direct implementation of a specific activation function as hardware does not appropriate to our work since the target hardware should be reconfigurable. Thus, we use a lookup table storing output values for define activation functions. By using this method, a few different activation functions can be implemented with fixed hardware.

3. The MLP-SoC

Fig. 2 shows the top level block description of the MLP-SoC. It comprises the LEON 2 core (main processor), MLP co-processor (hardware accelerator), memory controller, camera interface and bus system. All of these components are integrated into FPGA of the prototyping platform.

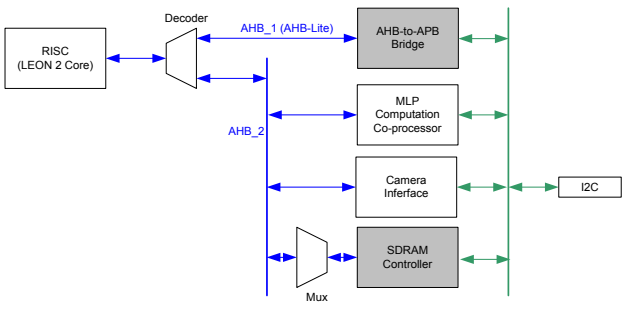


Figure 2. Architectural overview of the MLP-SoC

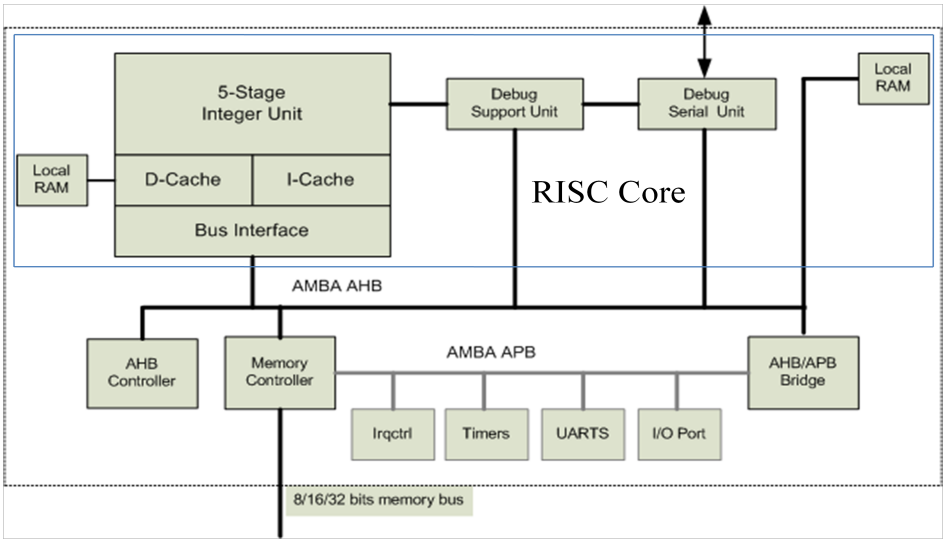


Figure 3. Block diagram of the standard LEON 2 processor

LEON 2 is a 32-bit RISC processor compliant with the SPARC V8 architecture. It is highly configurable and thus very suitable for SoC. Also, software written in C language can be directly executed under the LEON 2 core using cross-tool chain[13]. We implement LEON 2 core (shown in Fig. 3, dotted box) using open VHDL source into FPGA of the prototyping platform. The camera interface controller and the I^2C circuit are capable of handling a few image sensors with their fixed logics.

3.1. MLP Computation Co-processor

Fig. 4 shows the architecture of the implemented MLP computation co-processor that is dedicated to the basic computations of neurons. As seen in the figure, the MLP computation co-processor consists of two major parts - Host interface block for memory accesses and bus interface and MLP block for neural computing.

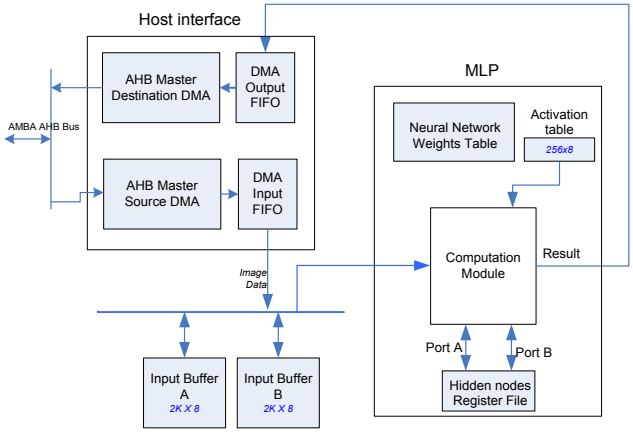


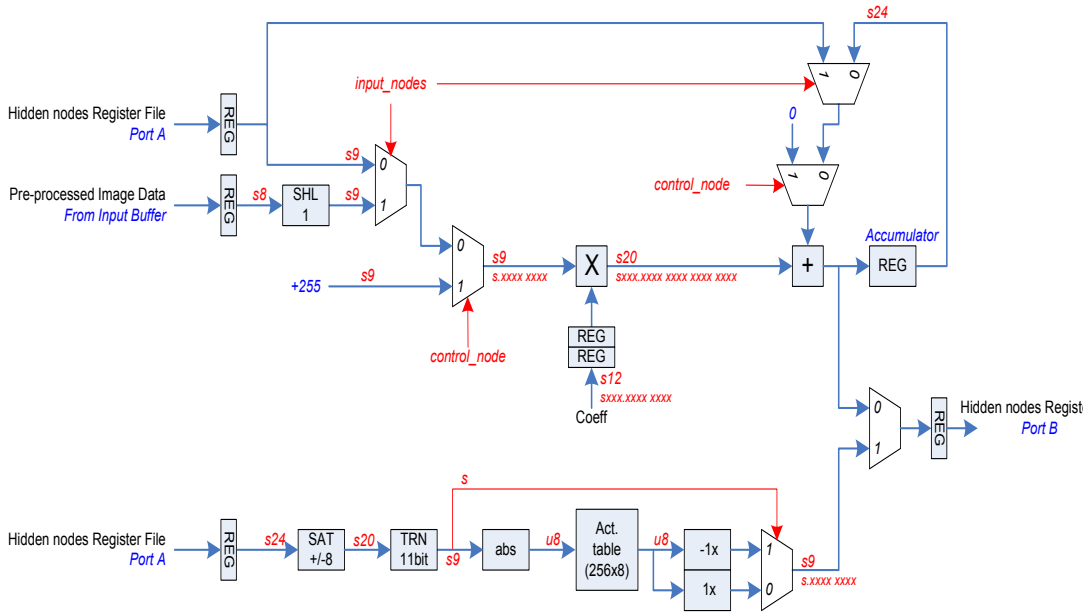
Figure 4. Architectural overview of the MLP computation co-processor

Host interface block is responsible for bus interfaces between MLP computation co-processor and other controllers. It consists of two direct memory access (DMA) units, source DMA and destination DMA. The source DMA block retrieves an input from the external memory and stores it into input buffers (2K X8 bits). Two buffers are prepared in order to be able to process one input, while another is being buffered for the next computation. The block sends the signal to the MLP block in order to start the computation task for the current input. When the computation task completes, the destination DMA block stores the generated output from the MLP block into the external memory. This data stream is useful when the size and the number of inputs are large.

In order to accommodate the constraints described in Table 1, the MLP block consists of storages and computation module. There are three different static memories: the function table, the hidden node register file and the weight table. An activation function, such as sigmoid or hyper-tangent function, can be implemented in the activation table without modifying hardware. The weight table consists of 128K * 19 bits, 12 bits are used for saving weight values and 7 bits are used for saving hidden node index. Hidden nodes register file consists of 128 * 24 bits for storing transient results of hidden/output nodes.

Fig. 5 shows the RTL diagram of the computation module. The computation module obtains inputs from an input buffer and computes activation values of all nodes of successive layers until the values of output nodes are computed. Then, it sends the output values to host interface block for saving them into SRAM-based memory. Fig. 5 also shows the precisions of the implemented logics.

The implemented MLP-SoC is fully synthesized by VHDL model and transferred into the FPGA (XILINX X2CV8000) of the prototyping platform. This MLP-SoC architecture provides fast processing of neural connections and transfer functions, and is well suited for MLP-type neural models. The operational clock rate of the FPGA is 30MHz. This clock source is fed into all components that are implemented on hardware. In the next section, we will verify the effectiveness of our reconfigurable architecture using a real application, building an OCR system onto the implemented architecture.



SAT: Saturation, TRN: Truncation, Act. Table : Activation function table

Figure 5. The RTL diagram of the computation module

4. Application Example: OCR System

OCR is the process by which a computer maps a digitized character image to text. This system is the base for many different types of embedded applications, such as portable translators, electronic dictionaries and personal data assistants[15]. The algorithm of the target OCR system consists of three main stages[16] as shown in the Fig. 6. First, an image is acquired by the MICRON MT9V112 image sensor[17] connected to camera interface. Second, preprocessing step is performed in order to segment the image into individual characters using histogram-based method. Extracted characters are converted into binary-valued images (0 or 255). Then the normalization for skew and size variations is performed to obtain 30X24 (pixels) sized actual input images of the MLP.

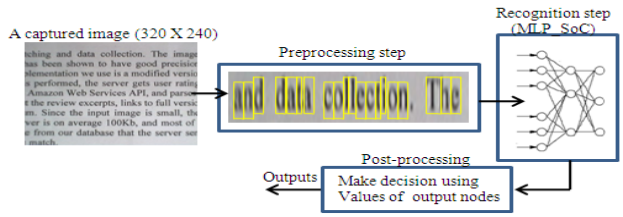


Figure 6. The processing flow of the implemented OCR system

Since a structure of a neural network, such as a number of nodes, an activation function, can be varied for a specific application for the better performance, a SoC for MLPs should accept these variations. To show the reconfigurable property of the MLP_SoC, we try to build two MLPs onto the same architecture. Table 2 shows configurations of the implemented MLPs. The number of hidden nodes of each MLP is randomly selected (3 ~ 5 % of the number of input nodes) and fixed while training and recognition phases.

We successfully implement each MLP into the FPGA of the prototyping platform independently without any modifications of the existing hardware. The training for each MLP is conducted onto the separated desktop computer with the same learning data set (English alphabets **a-z** and **A – Z,** Times New Roman). Using the trained weights, recognition experiments are performed with three 320X 240 (pixels) sized document images which have 730 characters total. The OCR systems recognize 686 (version_1) and 719 (version_2) characters correctly, thus the recognition rates are 94% (version_1) and 98% (version_2).

Table 2. Configurations of MLPs

Parameters	Version_1 MLP	Version_2 MLP
Input/hidden/output	720/24/26	720/32/26
Activation function	Hyperbolic tangent	Sigmoid
Connection type	Fully	Partially
Learning rule	Back-propagation	Back-propagation
Recognition rate	94%	98%

The other important issue of the evaluation for the MLP-SoC is the recognition speed. We check the required time of each module of the OCR system with version_2 MLP for recognizing one 320X240 (pixels) document which contains 260 characters. Table 3 shows the required times of all modules for the task. The OCR system can process nearly 43 characters per second. The neural computation module requires 3.9, while the software implementation of the target MLP requires 869 seconds under LEON2. This result is mainly due to the MLP computation co-processor that speed up the neural computing 223 times compared to the software implementation. Since almost commercial software OCR

systems are implemented on servers or desktop computers which have higher hardware capabilities such as powerful CPUs, they do not require a large amount of processing times. However wearable/mobile devices have constraints in their processing capabilities because of costs and power consumptions. Thus the hardware acceleration is the best solution to implement intelligence tasks into hardware constrained devices. From the experiments, we conclude that the implemented MLP_SoC can be used to build smart embedded devices capable of various image processing applications.

Table 3. Speed of each processing module for Reconfigurable OCR system

Algorithm stage	Processing mode	Required time
Segmentation	Software on LEON2	690 *m*sec
Binary and Normalization	Software on LEON2	1,240 *m*sec
Neural network	Hardware	3,980 *m*sec
Decision and save	Software on LEON2	120 *m*sec
Total time		6,030 *m*sec

5. Conclusions

In this paper, we design and implement the architecture of a MLP-SoC suitable for small-sized smart devices. The implemented SoC is tested and verified in the RTL using the FPGA-based prototyping platform. Without modifying the existing hardware, we can build two application systems on the designed architecture successfully by reconfiguring the SoC. The example shows that the MLP-SoC can be effectively used for various mobile/wearable devices which need intelligence capability. We are in the process of the chip fabrication for the implemented architecture.

REFERENCES

[1] Pentland and T. Choudhury, "Face recognition for smart environments," IEEE computer, vol. 33, no. 2, pp. 50-55, Feb., 2000.

[2] J. M. Zurada, *Introduction to Artificial Neural Systems*, PWS publishing company, 1992.

[3] T. Schoenauer, A. Jahnke, U. Roth and H. Klar, "Digital Neurohardware : Principals and Perspectives," Neural Networks in Applications (NN'98), Magdeburg, pp. 101 – 106, 1998.

[4] K. Mathia, J. Clark, B. Colbert and R. Saeks, "Benchmarking and MIMD Neural Network Processor," WCNN'96, San Diego, California, Sep., 1996.

[5] Theocharides, G. Link, N. Vijaykrishnan, M. J. Irwin and W. Wolf, "Embedded Hardware Face Detection", Proceedings of th 17th International Conference on VLSI Design (VLSID'04), Jan., 2004.

[6] M. Brogatti, F. Lertora, B. Foret and L. Cali, "A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O", IEEE J. Solid-State Circuits, vol. 38, pp. 521-529, Mar. 2003.

[7] E. M. Oritigosa, A. Canas, E. Ros, P. M. Ortigosa, S. Mota and J. Diaz, "Hardware description of multi-layer perceptrons with different abstraction levels," Microprocessors and Microsystems, vol. 30, pp. 435 – 444, 2006.

[8] S. Vitabile, V. Conti, F. Gennaro and F. Sorbello, "Efficient MLP Digital Implementation on FPGA," Proceedings of the 8th Euromicro conference on DSD, 2005.

[9] A. Rosado-Munoz, E. Soria-Olivas, L. Gomez-Chova and J. V. Frances, "An IP Core and GUI Implementing Multilayer Perceptron with a Fuzzy Activation Function on Configurable Logic Devices," Journal of Universal Computer Science, vol. 14, no. 10, pp. 1678 – 1694, 2008.

[10] M. Pormann, M. Franzmeier, H. Kalte, U. Witkowski and U. Ruckert, "A Reconfigurable SOM Hardware Accelerator," ESANN'2002 proceedings, pp. 337 – 342, 2002.

[11] M. Shabiul, M. S. Beg, M. S. Bhuyan and M. Othman, "Design and Implementation of Discrete Cosine Transform Chip for Digital Comsumer Products," IEEE Transaction on Consumer Electronics, vol. 52, no. 3, pp. 998 – 1003, 2006.

[12] P. G. D. Valle, D. Atienza, G. Paci and F. Poletti, "Application of FPGA Emulation to SoC Floorplan and Packaging Exploration," XXII Conference on Design of Circuits and Integrated System, pp. 236 – 240, 2007.

[13] LEON2 processor user's manual, Gaisler Research, http://www.gaisler.com

[14] S. M. Shon, S. H. Yang, S. W. Kim, K. H. Baek and W. H. Paik, "Soc Design of An Auto Focus Driving Image Signal Processor for Mobile Camera Applications", IEEE Transactions on Consumer Electronics, vol. 52, no. 1, pp. 10-16, Feb., 2006.

[15] H. Nakajima, Y. Matsuo, M. Nagata and K. Saito, "Portable Translator capable of Recognizing Characters on Signboard and Menu Captured by built-in camera," Proc. of the ACL Interactive Poster and Demonstration Sessions, pp. 61 – 64, June, 2005.

[16] B. Lee, Y. Cho and S. Cho, "Translation, scale and rotation invariant pattern recognition using PCA and reduced second order neural network ", Neural, Parallel & Scientific Computation, vol. 3 no.3, pp.417-429, 1995.

[17] MT9V112 manual, Micron Technology Inc., http://www. micron.com