MICRO NEURAL-CONTROLLER FOR OPTICAL CHARACTER RECOGNITION

Ker-Chin Lim and Cesar Ortega-Sanchez

Electrical and Computer Engineering Department
Curtin University of Technology
GPO Box U1987
Perth, Western Australia 6102
c.ortega@curtin.edu.au

ABSTRACT

This paper presents a Micro-Neural Controller (MNC) for Optical Character Recognition (OCR). The controller consists of both a multilayered feedforward Artificial Neural Network (ANN) and a Von Neumann-type microcontroller. The ANN is supervised and trained using back-propagation. The role of the ANN is to perform character recognition while the microcontroller coordinates the data transfer between the network and the user. The results show that the network recognizes the patterns effectively and the microcontroller is able to execute a demonstration program.

1. INTRODUCTION

OCR was first introduced by Intelligent Machine Corporation in 1959 [1]. OCR involves extracting characters from hard-copy paper and translating them into a document format that can be manipulated by a computer. Usually, the first step is to create a digital picture of the text. In this format, the image is simply an array of black and white dots. A special pattern-recognition algorithm is then applied to translate image into text. The effectiveness of OCR largely depends on the software used. However, good OCR software can be very expensive.

At present, the role of the scanner is to provide a clear scanned image and deliver it to the computer equipped with OCR software; it contributes very little to the entire OCR process. If the scanner were equipped with character recognition capabilities, software complexity would be diminished and throughput could be improved.

This paper presents the preliminary design and implementation of an embedded system to perform OCR in hardware.

2. SYSTEM DESCRIPTION

The embedded system consists of a custom microcontroller, memory, an ANN and I/O logic to interface with external hardware. All these components are implemented in a Field-Programmable Gate Array (FPGA). Figure 1 shows the basic blocks of the system.

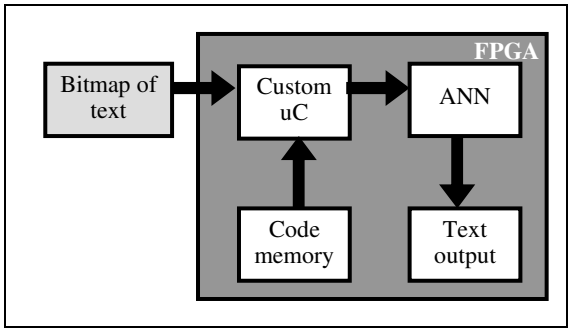

Figure 1. Block diagram of OCR system

2.1 Custom Microcontroller

A 16-bit custom microcontroller (uC) was designed for the OCR system. The uC is based on the traditional Von Neumann architecture, i.e. code and data are stored in the same memory, and the CPU continuously performs a fetch-execute-store cycle. In its current implementation, the uC can execute 16 different instructions; hence, one 16-bit micro instruction is sufficient to specify operation, operands and destination.

There are typically four stages in a machine cycle:

A. "Fetch the instruction" from memory. This step brings the instruction into the instruction register, so that it can be decoded and executed.
B. "Decode" the instruction. The instruction is used to generate a set of control signals that get the Arithmetic and Logic Unit (ALU) ready to perform the operation.
C. "Execute" the instruction. The individual steps of the instruction are executed. Several steps may be

required to perform an instruction, such as reading data from memory, then instructing the ALU to perform an addition.

D. "Store" the result of the instruction in memory, if needed.

Figure 2 shows the basic components of the microcontroller.

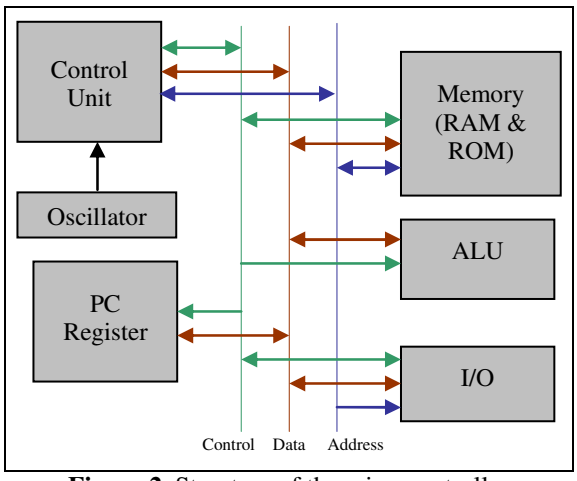

Figure 2. Structure of the microcontroller

In Figure 2, the control unit runs the program stored in memory. At the present stage, the program is part of the VHDL code describing the memory; however future versions will be able to receive new programs from a PC. The Control Unit (CU) reads instructions from the data bus (fetch). The CU then decides what to do depending on the value of the 4-bit, operation field loaded from the data bus (decode). The outputs from the control unit are set according to the current instruction and the ALU's status (execute).

The MCU has four main operations. Depending on the data bus value, it can perform read, write, arithmetic or logic, and jump operations.

- READ- The value read by the CU is stored in either register A or B.
- WRITE- The CU puts a value on the data bus.
- ARITHMETIC OR LOGIC- These operations are performed on values that have already been place in the registers
- JUMP- The jump operation sets the PC Register to a specific address so that the flow of execution is altered.

The ALU performs operations such as AND, OR, ADD and SUBTRACT on its inputs. The ALU gets its command line from the Control Unit where an instruction has been fetched and decoded, and is now part of the execution cycle. The value calculated by the ALU is put on the data bus and control signals are sent to the CU to indicate that data are available on the data bus.

The PC Register points to the next instruction to be executed and it is incremented after the MCU executes that instruction. Only the jump instruction can modify the default increment behaviour of the PC.

The RAM can store up to 128 bytes. At reset, the RAM's content is initialised to zero. The RAM can either perform read or write operations; depending on the signals issued by the Control Unit.

The ROM can store up to 128 16-bit words. Every word contains one instruction used by the Control Unit to perform operations. The eight lower bits of the instruction contain the input value to the microcontroller, and the four highest bits contain the command sent to the MCU. This field is used by the Control Unit to decide what operation to perform, and which output lines to drive high or low. The middle 4 bits code the destination where the result from the current instruction will be stored.

The I/O system interfaces the microcontroller with the I/O lines of the FPGA. Since some of the onboard I/O lines require a logic '0' to be active, NOT gates had to be interfaced in between to make the active signal as '1' before latching the results. This would make designing the microcontroller easier by standardising all active logic as '1'.

2.2 Artificial Neural Network (ANN)

The ANN implemented in this project is the multilayer feedforward perceptron trained with the back-propagation algorithm. This type of neural network is used in OCR applications because of its good pattern-recognition properties [2]. Figure 3 illustrates the use of ANN in OCR [3].

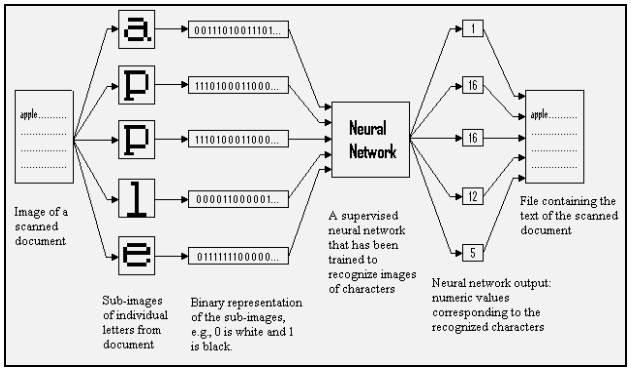

Figure 3. ANN used in OCR.

For testing purposes, the input to the system is set to a 3x5 pixel array that is used to represent 10 decimal digits. Figure 4 shows the 10 patterns used to train the ANN.

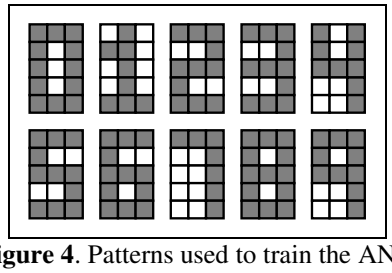

Figure 4. Patterns used to train the ANN

Regarding the topology of the ANN, the number of input neurons is determined by the dimension of the input vectors to be classified. Usually, the size of the input vector corresponds with the number of distinct features of the input pattern. Since the number of pixels within a single class is 15, therefore the number of input neurons was set to 15.

The number of output neurons is equal to the number of output classes. In this case, the output neurons is 10, corresponding to the 10 digits.

The size of the hidden layer is a very important consideration. However, this problem is still under intensive study with no conclusive results [4]. Using too few hidden neurons will starve the network. Using too many will increase the training time. One rule of thumb to choose the number of hidden neurons is the geometric pyramid rule [5]. Let's denote m as number of output neurons and n as number of input neurons, the number of hidden neurons is given by,

$$hidden = \sqrt{m.n} \qquad (1)$$

Hence, the number of hidden neurons was set to sqrt(15x10)≈12.

The choice and shape of the activation function strongly affects the speed of the network learning process. Considering that the final implementation would be in an FPGA, a positive ramping function was selected. Figure 5 shows a simplified representation of the ANN topology.

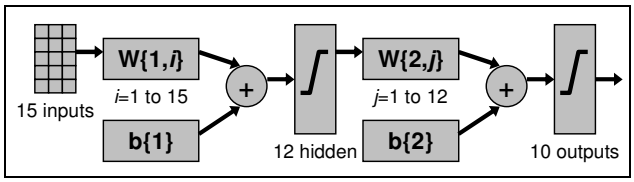

Figure 5. Topology of ANN

In Figure 4, W{1,*i*} and W {2,*j*} are the weights associated to each neuron in the network. b{1} and b{2} are biases used to accelerate the convergence of the back-propagation learning algorithm. They help to supplement the current weight adjustment with a fraction of the most recent weight adjustment. Momentum terms can be used for both input and hidden layers of the network during training.

The effectiveness and convergence of the error back-propagation learning algorithm depends significantly on the value of a learning constant η. The optimum value of η depends on the problem being solved. Typical values are between 0.001 and 10. Small learning constants would require longer time to reach convergence while large learning constants increase the learning speed but will not stabilize the network.

To determine the best value for the parameters of the ANN, various combinations were evaluated using MatLab's Neural Networks toolbox. It was found that a learning constant η of 0.1 and momentum of 0.9 yielded the best performance.

3 IMPLEMENTATION

The system was implemented using Xilinx's ISE 8.1 software and targeted for a Spartan 3MB FPGA. Table 1 shows a summary of the resources required by the OCR embedded system, including all the blocks shown in Figure 1. It is clear that there are sufficient FPGA resources for future improvements.

Table 1. FPGA Utilisation Summary

Logic Utilization	Used	Available	Utilisation
Number of Slice Flip Flops	1,236	26,624	4%
Number of 4-input LUTs	3,061	26,624	11%
Logic Distribution			
Number of occupied Slices	2,171	13,312	16%
Total Number of 4-input LUTs	3,512	26,624	13%
Number of bonded IOBs	75	487	15%
Number of Block RAMs	1	32	3%
Number of MULT18X18s	4	32	12%
Number of GCLKs	1	8	12%
Total equivalent gate count	129,740		

To simulate inputs, two 3x5 arrays of push-buttons were assembled. Every array can represent one decimal digit. Additional push buttons were included to provide auxiliary functions.

To test the system a 1-digit calculator was implemented. The additional buttons were used to perform addition, subtraction and multiplication of the two digits represented by the push-button matrices. Figure 6 shows a photograph of the FPGA development board and the input device.

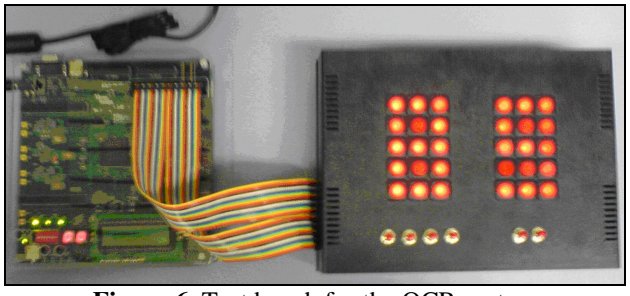

Figure 6. Test bench for the OCR system

To find a suitable ANN, the network described in section 2.2 was trained with different percentage of noise in the trained sets. Overall, the best network managed to recognize 97.5% of noisy characters when 2 or 3 of the pixels were corrupted. Thus, the weights and biases obtained for that particular ANN were used for implementation into the FPGA design.

4 CONCLUSIONS AND FUTURE WORK

An embedded OCR system based on a custom micro-neural controller was successfully developed and tested. Functionality can be tailored by varying the training examples or increasing the resolution of pattern coding. The required computational effort is rather low since the network and microcontroller are compact enough to fit into an FPGA. This structure is highly recommended for OCR applications.

The system presented in this paper is a work in progress. Future versions will incorporate a bigger set of characters, better resolution, a interface to read the output of a scanner directly and a microcontroller with a bigger instruction set.

Having a customised microcontroller in the system allows the efficient use of FPGA resources by synthesising only the functions that will be needed. Further research in this area is being contemplated.

REFERENCES

[1] R. Wisneski, "Digital Image Processing: Optical Character Recognition (OCR)", October 2004. Retrieved: May 1, 2006 from http://www.personal.kent.edu/~rwisnesk/jstor/jstor.htm.

[2] William R. Wiley Environmental Molecular Sciences Laboratory, "Neural Network Commercial Applications", 2006. Retrieved: April 12, 2006 from http://www.emsl.pnl.gov:2080/proj/neuron/neural/products/

[3] Neuro Dimension Inc., Retrieved November 2005, http://www.nd.com/neurosolutions/products/ns/whatisNN.html

[4] Kemao Peng, Shuzhi S. GE, Chuanyuan Wen, " An Algorithm To Determine Neural Network Hidden Layer Size And Weight Coefficients", IEEE Transactions, 17-19 July 2000. Retrieved: April 12, 2006 from http://vlab.ee.nus.edu.sg/~sge/conference/ISIC00-2.pdf

[5] Mohamed A. Shahin, Mark B. Jaksa, Holger R. Maier, "Applications Of ANN In Foundation Engineering", 2004. Retrieved: May 2, 2006 from http://www.ecms.adelaide.edu.au/civeng/staff/mjaksa01/pdf/e-Conf_2004.pdf