PCNN NEURON IMPLEMENTATION WITH DMA TRANSFER

Peter ŠEVČÍK

University of Žilina, Faculty of Management Science and Informatics, Slovak Republic
e-mail: peter.sevcik@fri.uniza.sk

Abstract

This paper presents implementation of one pulse coupled neural network (PCNN) neuron into Virtex 4 FPGA device on ML 403 development board. The implementation consists of three main units: Computing unit (CU), convolution computing unit (CCU) and MicroBlaze soft microprocessor. Neuron works with fixed point arithmetic, and is designed with the accent to effective utilization of the FPGA. The data transfer is effectively made by using DMA transfer unit.

Keywords: FPGA device implementation, DMA transfer, neural network, convolution computing, fixed point arithmetic, error rate

1 INTRODUCTION

The pulse coupled neural network, is a biological model inspired of mammalian visual cortex, proposed by Eckhorn et al. (1990). The PCNN is advisable to solve tasks as the feature generation for image and pattern recognition [1] and image segmentation [2]. The PCNN is easily implemented in software [3]. However, due to the parallel nature of the PCNN, it should be more advantageous to implement it in hardware. The modern FPGA chips contain large bitwise multipliers and adders, block rams and more than 1 million gates. These acquisitions are significant for successful implementation of high parallel embedded systems [4].

Several models of PCNN have been developed. The most used PCNN models are:

− Standard PCNN model,
− PCNN model with modified feeding input,
− Fast linking PCNN model,
− Feedback PCNN model.

PCNN model with modified feeding input is described as iteration of the following equations:
\[ F_{ij}(n) = S_{ij} \]  
\[ L_{ij}(n) = L_{ij}(n-1) \cdot e^{-\alpha L} + V_L \cdot (W \ast Y(n-1))_{ij} \]  
\[ U_{ij}(n) = F_{ij}(n) \cdot (1 + \beta \cdot L_{ij}(n)) \]  
\[ \Theta_{ij}(n) = \Theta_{ij}(n-1) \cdot e^{-\alpha \Theta} + V_{\Theta} \cdot Y_{ij}(n-1) \]  
\[ Y_{ij}(n) = \begin{cases} 1 & \text{if } U_{ij} > \Theta_{ij} \\ 0 & \text{otherwise} \end{cases} \]

Where \( F_{ij} \) is the feeding input, \( L_{ij} \) is the linking input, \( n \) is the iteration step, \( S_{ij} \) is the intensity of pixel in the input matrix, \( W \) is the weight matrix, \( \ast \) is the convolution operator, \( Y_{ij} \) is the output of the neuron, \( V_L \) and \( V_{\Theta} \) are potentials, \( \alpha L \) is decayed constant and \( i, j \) are indices.

### 1.1 Structure of PCNN

The structure of PCNN comes out from structure of input image, which will be processed. It means that PCNN is single layered, two dimensional, laterally connected neural network of pulse coded neurons, which are connected with image pixels each other. Each image pixel is associated with a pulse coupled neuron of specific structure (Figure 1). PCNN neuron consists of the input part, linking part and the pulse generator. The neuron receives the input signals from feeding and linking inputs. Feeding inputs is the primary input from the neuron’s receptive area. The neuron receptive area consists of the neighboring pixels of corresponding pixel in the input image. Linking input is the secondary input of lateral connections with neighboring neurons.

![Figure 1](image_url) The block diagram of the PCNN neuron with modified feeding input
1.2 Analysis of system specification

The original PCNN neuron algorithm contains 6 operations of multiplication, 3 operations of addition and one operation of the convolution. Some changes are made to the original algorithm and the result is that we save 2 operations of the multiplication. First change consists in modification of the linking equation. After making the substitution (6), we get new equations (7) and (8). Now we have two constants \( V_L \) and \( \beta \) joined together.

\[
L_{ij}' = \frac{L_{ij}}{V_L} \tag{6}
\]

\[
L_{ij}'(n) = L_{ij}'(n-1) \cdot e^{-\alpha L} + (W * Y(n-1))_{ij} \tag{7}
\]

\[
U_{ij}(n) = F_{ij}(n) \cdot (1 + \beta \cdot V_L \cdot L_{ij}'(n)) \tag{8}
\]

The second modification consists in solution of the multiplication \( V_{\Theta} \) and \( Y_{ij}(n-1) \) in (4). Because the values of the matrix \( Y_{ij} \) are only 0 or 1, we can use simple multiplexer (VT MUX) instead of the multiplier. The hardware implementation follows the equations (1), (4) (5), (7) and (8).

2 IMPLEMENTATION OF THE NEURON

2.1 Computing Unit (CU)

Computing unit of the PCNN neuron does all operations of multiplication and addition besides convolution.

![Figure 2](image-url) The block diagram of the PCNN neuron computing unit
It contains 4 hardware multiplication units (M1 ÷ M4), 3 hardware adders (A1 ÷ A3), one multiplexer and one comparator. CU, depicted at figure 2, is designed as a pipelined structure and works with fixed point arithmetic. All multipliers and adders were designed with the help of Core Generator, which is part of the ISE Foundation design environment.

2.2 Convolution computing unit (CCU)

The convolution computing for the claim of PCNN neuron algorithm is defined by (9). The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window itself. The window with its weights is called the convolution kernel.

$$W \ast Y(n-1)_{ij} = \sum_{k=i-r_0}^{k=i+r_0} \sum_{l=j-r_0}^{l=j+r_0} W_{(k-i) + r_0 + j-l} \cdot Y(n-1)_{kl}$$

$$Y(n-1)_{ij}$$ is the neuron’s output matrix in the step before, n is the iteration step, k and l are indices, $$r_0$$ is the radius of the convolution computing and W is the example of the convolution kernel matrix ($$r_0 = \sqrt{2}$$).

The architecture of the CCU unit (Figure 3) is designed for 128 x 128 Yi pattern computations. The base components of the data path block are three shifts registers SHR0 ÷ SHR2 128 x 1 bit. They are implemented as cascaded 8 x SRLC16E elements. Individual bits Yi of the pattern are shifted in successive step by CLK into shift registers. The computation of the convolution starts immediately after SHR0 and SHR1 are full; it means that 2 first rows of the Yi pattern are in appropriate shift registers. Shift registers Q output bits are clocked in D flip-flops. Bits Q1 - Q6 input in to the converters. These converters create binary sum of input bits (chain of 0 and 1). Resultant value of the convolution is written in to the FIFO 16384 x 4 bits. FIFO stack is also implemented as cascaded 4096 x SRLC16E elements. Signals P1 and P2 modify the computation at the beginning and in the end of every row. The current values of the convolution are used in the next cycle of the neuron algorithm computation. The whole cycle of the convolution computation lasts 16642 clock periods.

2.3 The MicroBlaze soft processor

The MicroBlaze is a soft processor core designed for Xilinx FPGAs from Xilinx. As a soft-core processor, MicroBlaze is implemented entirely in the general-purpose memory and logic fabric of Xilinx FPGA’s. MicroBlaze is designed for using in Spartan 3 devices or higher families. Processor runs at 100 MHz frequency in this
implementation. The MicroBlaze soft processor insures these functions in the PCNN neuron implementation:

- Sending and receiving results gained in the process of the computation and communication with PC via UART (57 600 bits/sec),
- Communication with local memory DDR SDRAM 64 MB,
- Activation and control of central DMA transfer controller,
- Control the process of the convolution computing realized by the CCU.

![CCU Architecture](image)

**Figure 3** The CCU architecture

The PCNN neuron design, which composed of the computation unit and convolution computing unit, is connected via user registers as slave to the MicroBlaze processor local bus (Figure 6). The bus provides bi-directional interface between microprocessor and PCNN neuron. It is 32 bits wide and works at 100 MHz frequency.

### 2.4 Central DMA unit

The central DMA controller provides simple direct memory access (DMA) services to peripherals and memory devices on the processor local bus. The controller transfers a programmable quantity of data from a source address to a destination address without processor intervention. For the most effective transfer we need to order input data. Figure 4 is shown the input data order used in this implementation. Input data used in particular iteration step are ordered one after another. DMA controller unit transfers data from 5 particular addresses (e.g. \( S_{0,0} \); \( L_{0,0} \); \( \Theta_{0,0} \); \( Y_{0,0} \) and \( Conv_{0,0} \)) to appropriate neuron computing unit registers in one command. After the computation is done, DMA central controller moves computed data back from neuron computing unit.
to appropriate memory addresses also in one command. Those commands are repeated for 16,384 times in one iteration step.

### 3 THE COMPLETE SYSTEM IMPLEMENTATION

The implemented PCNN neuron communicates via UART with PC. At the beginning MicroBlaze initializes all the peripherals which are connected to the MB PLB bus. After initialization, processor is waiting for the input data. Input matrix is represented picture with 128x128 pixels dimensions in 8 bit grayscale. After receiving input data, they are stored in particular memory addresses in the local DDR SDRAM. Afterwards the main computing cycle is started. DMA controller moves data between DDR SDRAM memory and neural computing unit without wasting useful processor time. After the computation is done, the results are sent to PC via RS 232C port. The results can be also processed and displayed by Matlab environment.

![Figure 4](image-url) The data storing positions in the DDR SDRAM

![Figure 5](image-url) The interconnection between MicroBlaze microprocessor and its peripheries
4 EXPERIMENTS AND RESULTS

The PCNN neuron was implemented also in the Matlab environment for testing and comparison purposes. The standard testing image is used as the input image. This grayscale input image (Figure 6), which is represented by two-dimensional matrix, is through PCNN transformed into a sequence of temporary binary images.

![Figure 6 Input testing image (128x128 pixels)](image)

Each of these binary images is a matrix with the same dimension as input matrix and it is generated by group of pixels with similar intensity. The sum of all activities in specific iteration step gives one value, which represents one feature for the classification. If we have \(n\) iteration steps, we obtain \(n\) features. We use this function (11) and (12) generated in every iteration step, for comparison both implemented versions of the PCNN neuron.

\[
G(n) = \sum_{i=0}^{127} \sum_{j=0}^{127} Y_{ij}(n)
\]  

(11)

\[
\text{error rate (n)} = \left( \frac{G(n)_{\text{MV}} \times 100}{G(n)_{\text{FV}}} \right) - 100
\]  

(12)

\(G(n)_{\text{FV}}\) is function which represents features computed in FPGA implemented CU and \(G(n)_{\text{MV}}\) function which represents features computed by Matlab. The work of the PCNN neuron interact constants \(K_L\) \((e^{-aL})\), \(A_\phi\) \((e^{-a\phi})\), \(V_\phi\), and \(\beta VL\). Values of these constants in decimal and binary coded fixed point arithmetic indicate table 1.

<table>
<thead>
<tr>
<th>Table 1 Setting up PCNN neuron constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>(K_L)</td>
</tr>
<tr>
<td>(A_\phi)</td>
</tr>
<tr>
<td>(V_\phi)</td>
</tr>
<tr>
<td>(\beta VL)</td>
</tr>
</tbody>
</table>
As we can see from figures 7a and 7b, there are some differences in generated features. These differences are caused by using the fixed point arithmetic in the FPGA implementation and iterative character of the computation. It means that error rate is cumulating together in every iteration step as is shown at figure 8.

![Figure 7](image-url) Features of the input image obtained from PCNN neuron implemented a) in the FPGA device b) in the Matlab environment c) error rate

5 CONCLUSION

This paper presents complete embedded system implementation of PCNN neuron into Virtex 4 FPGA device. The PCNN neuron implemented on this platform works with fixed point arithmetic and is controlled by the MicroBlaze microprocessor, which is also implemented in the same FPGA device. MicroBlaze uses DMA controller to move large amount of data between DDR SDRAM and neuron computing unit registers. PCNN neuron architecture is optimized for the image features generation from input grayscale images with dimensions 128 x 128 pixels. Despite the fact, that there are differences, the features generated by the embedded neuron are useful and have all the properties such as invariance to rotation, dilatation or translation.

REFERENCES


