Hardware and Software modifications on the Neocognitron and its binary implementations on FPGA

Basil Sh. Mahmood[1]
Computer Department, College of Electroni Engineering, University of Mosul

Shefa A. Dawwd[2]
Computer Department, College of Engineering, University of Mosul

[1] basilsh@mosuluniversity.org
[2] shefadawwd@yahoo.com

Abstracat: Convolutional neural network (CNN) is a well-known robust image recognition model. It is a multi-layer architecture where the successive layers are designed to learn progressively higher-level features, until the last layer which produces categories. In order to apply this model to robot vision or various intelligent systems, its VLSI implementation with high performance is required. In this paper, a hardware implementation of one of the most complex and powerful image recognition convolutional neural networks(the neocognitron) has been considered. The original neocognitron has been modified to reduce its complexity whether it is hardwarely or softwarely implemented while its principle working ideas has been kept. As a result to these modifications and simplifications, a relatively small FPGA hardware model of 200,000 gates has been used to implement a relatively complex design. The system is tested by using the Optical Recognition Library(ORL) face database for face recognition problem and its performance is compared with the results obtained using advanced software system designed specifically for face recognition. The recognition rate achieved from both software and hardware versions were equal to 93%. A speed up of (88) is achieved for the parallel architecture implemented in an FPGA as compared with the computer software. Also a performance of 1GCPS is achieved and seems reasonable when it is compared to the today available neuro-hardwares[1].

Keywords: Convolutional neural network implementation, neocognitron, neural hardware implementation.

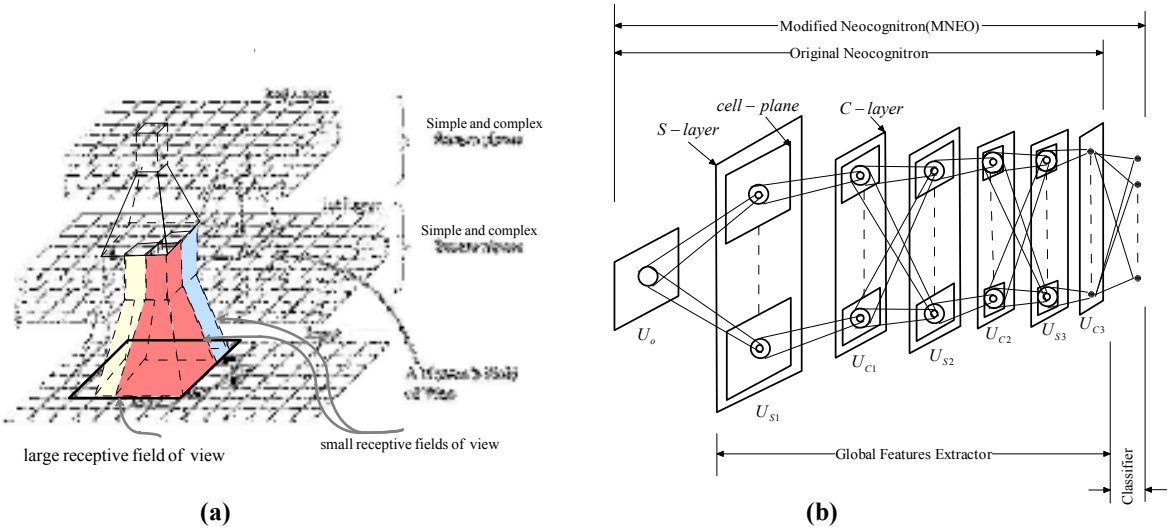
1 Introduction

Convolutional neural networks(CNN) with local weight sharing topology gained considerable intrest both in the field of speech and image analysis. Their topology is more similar to biological networks based on receptive fields and improves tolerance to local distortions. Additionally, the model complexity and the number of the weights are efficiently reduced by weight sharing. This is an advantage when images with high-dimensional input vectors are to be presented directly to the network instead of explicit feature extraction that results in reduction which is usually applied before classification. Weight sharing can also be considered as alternative to weight elimination in order to reduce the number of the weights. Moreover, networks with local topology can more effectively be migrated to a locally connected parallel computer than fully connected feedforwared network [2].

In this paper, the neocognitron convolutional network structure which will be used as a target model and will be hardwarely implemented.

2 Neocognitron

Fukushima [3] was amongst the first to experiment with convolutional neural networks and obtained good results for character recognition by applying convolutional neural networks within an image pyramid architecture: processing layers alternate between convolution and sub-sampling. This architecture is called Neocognitron. This multi-scale architecture has been now widely adopted and appears to provide a robust representation in many object recognition problems. A practical architecture of the Neocognitron is shown in Fig. 1(a).



(a) **(b)**

Figure 1: (a)Neocognitron main layers and receptive fields (b) Schematic diagram illustrating the structure of the Neocognitron.

Each layer extracts certain shape-features, as for example edge orientation, from a localized region of the preceding layer and projects the extracted information to the next higher layer. The complexity and abstractness of the detected features grow with the layer height, until complicated objects can be recognized. A layer consists of a number of feature planes, each of which is assigned to recognize one specific image feature.

Neurons belonging to the same plane are identical in the sense that they share the same synaptic weights. This architecture, showing a high degree of self-similarity, seems particularly dedicated to be implemented on a parallel hardware platform.

For simplicity, another illustration of the Neocognitron when the feature planes are arranged serially and the receptive fields are represented as circles is shown in Fig.1(b). Note that the modified neocognitron(MNEO) is proposed and implemented in this paper.

The neocognitron consists of a cascade connection of a number of modular structures preceded by an input layer *Uo*. Each of the modular structures is composed of two sub-layers of cells, namely a sub-layer *Us* consisting of S-cells, and a sub-layer *Uc* consisting of C-cell (S-cells and C-cells are named after simple cells and complex cells in physiological terms, respectively). Regard to CNN cells and layers names, S-cells refer to cells in convolution layers whereas C-cells refer to cells in down-sampling layers. In the Neocognitron, only the input interconnections to S-cells are variable and modifiable and in contrast to the down-sampling layers in CNN, the input interconnections to C-cells are fixed and unmodifiable.

2.1 Cells Employed In the Neocognitron

All the cells employed in the neocognitron are of analogue type: i.e, the input and output signals of the cells have non-negative analoge values. Each cell has characteristics analogous to a biological neuron. In the Neocognitron, it is used four different kinds of cells, i.e., S-cells, C-cells, Vs-cells and Vc-cells.

An S-cell has a lot of input terminals, either excitatory or inhibitory. If the cell receives signals from excitatory input terminals, the output of the cell will increase. On the other hand, a signal from an inhibitory input terminal will suppress the output. Each input terminal has its own interconnecting coefficient whose value is positive. Although the cell has only one output terminal, it can send signals to a number of input terminals of other cells.

An S-cell has an inhibitory input which causes a shunting effect. Let *u(1), u(2),……u(N)* be the excitatory inputs and *v* be the inhibitory input.

The output *w* of this S-cell is defined by[3]:

$$w = \varphi\left[\frac{1+e}{1+h} - 1\right] = \varphi\left[\frac{e-h}{1+h}\right] \qquad (1)$$

where :

$$e = \sum_{v=1}^{N} a(v).u(v)$$

$$h = b.v$$

$$\varphi[x] = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where $a(v)$ and b represent the excitatory and inhibitory interconnecting coefficients, respectively.

The cells other than S-cells also have characteristics similar to those of S-cells. The input-to-output characteristics of a C-cell are obtained from the last equation if we replace $\varphi[\ ]$ by $\psi[\ ]$, where $\psi[\ ]$ is a saturation function defined by:

$$\psi[x] = \begin{cases} \dfrac{x}{\alpha + x} & x > 0 \\ 0 & x \leq 0 \end{cases} \qquad (2)$$

The parameter α is a positive constant which determines the degree of saturation of the output. In the computer simulation and in hardware implementation discussed in section 5, the parameter α is chosen to be equal to zero.

S-cells and C-cells are excitatory cells, i.e., the output terminals of these cells are connected only to excitatory input terminals of other cells. On the other hand, Vs-cells and Vc-cells are inhibitory cells, whose output terminals are connected only to inhibitory input terminals of the other cells. A Vs-cell has only excitatory input terminals and the output of the cell is proportional to the sum of all the inputs weighted with the interconnecting coefficients. That is a Vs-cell yields an output proportional to the (weighted) arithmetic mean of its inputs.

A Vc-cell also has only excitatory input terminals, but its output is proportional to the (weighted) root-mean-square of its input. Let $u(1), u(2), \ldots\ldots u(N)$ be the inputs to a Vc-cell and $c(1), c(2), \ldots\ldots c(N)$ be the interconnecting coefficients of its input terminals. The output w of this Vc-cell is defined by:

$$w = \sqrt{\sum_{v=1}^{N} c(v).u^2(v)} \qquad (3)$$

2.2 Formulae Governing the network

S-cells have inhibitory inputs with a shunting mechanism. The output of an S-cell of the k_l-*th* S-plane in the l-th module is given by [3]

$$u_{sl}(k_l, n) = r_{ol} \cdot \varphi\left[\frac{1 + \sum_{k_{l-1}=1}^{K_{l-1}} \sum_{v \in S_l} a_l(k_{l-1}, v, k_l).u_{cl-1}(k_{l-1}, n+v)}{1 + \frac{r_{ol}}{1+r_{ol}}.b_l(k_l).v_{cl-1}(n)} - 1\right] \qquad (4)$$

where :

$$v_{cl-1}(n) = \sqrt{\sum_{k_{l-1}=1}^{K_{l-1}} \sum_{v \in S_l} c_{l-1}(v).u^2_{cl-1}(k_{l-1}, n+v)}$$

$\varphi[\ ]$	: a function defined by equation (1)
$v_{cl-1}(n)$	: Vc-cell,layer l-1, position n
$a_l(\ ), b_l(\ )$	: modifiable weights
$c_{l-1}(\ )$	:positive fixed weights
r_{ol}	:selectivity parameter
S_l	: receptive field

The selectivity parameter r_{ol} in the above equation controls the intensity of the inhibition. The larger the value of r_{ol} is, the more selective becomes the cell's response to its specific feature. r_{ol} is believed that it is a key factor to control the ability of the neocognitron to recognize deformed patterns. If the selectivity is too high, the neocognitron loses the ability to generalize and cannot recognize deformed patterns robustly. If the selectivity is too low, the neocognitron loses the ability to differentiate between similar patterns of different categories. The values of fixed interconnections $c_{l-1}(\ )$ are determined so as to decrease monotonically with respect to $|v|$. The size of the connecting area S_l of these cells is set to be small in the first module and to increase with respect to depth l.

The interconnections from S-cell to C-cell are fixed and unmodifiable as mentioned. Each C-cell has input interconnection leading from a group of S-cells in the S-plane preceding it (i.e., in the S-plane with the same k_l–number as that of the C-cell). This means that all of the S-cells in the C-cell's connecting area extract the same stimulus features but from slightly different positions on the input layer. The values of the interconnections are determined in such a way that the C-cell will be activated whenever at least one of these S-cells is active, hence, even if a stimulus pattern which has elicited a large response from the C-cell is shifted a little in position, the C-cell will still keep responding as before, because another neighboring S-cell in its connecting area will become active instead of the first. In other words, a C-cell responds to the same stimulus feature as the S-cell

preceding it, but is less sensitive to a shift in position of the stimulus feature.

Quantitatively, the output of a C-cell of the k_l-th C-plane in the l-th module is given by:

$$u_{cl}(k_l,n)=\psi\left[\frac{1+\sum_{v\in D_l} d_l(v)u_{sl}(k_l,n+v)}{1+v_{sl}(n)}-1\right] \quad (5)$$

where:

$$v_{sl}(n)=\frac{1}{K_l}\sum_{k_l=1}^{K_l}\sum_{v\in D_l} d_l(v)u_{sl}(k_l,n+v)$$

$\psi[\ ]$: a function defined by equation (2)
$v_{sl}(n)$: Vs-cell,layer l, position n
$d_l(v)$: fixed interconnection which determined so as to decrease monotonically with respect to $|v|$
D_l : receptive field, the size of D_l is set to be small in the first module and to increase with the depth of l

3 The proposed Image Recognition Neural Network System

The image recognition system described in this paper consists of a hierarchy of several layers of artificial neurons, arranged in planes to form layers. The system consists of two parts: feature extractor and classifier. The feature extractor operates on an input image, which are then processed by the classifier (see Fig. 1(b)). The neocognitron is used as a feature extractor. An image is divided by the feature extractor into subimages. The extraction of local features is based on the similarity among subimages. The feature extractor is usually trained by unsupervised training algorithm. The training is achieved sequentially layer by layer, and the output of each layer will be considered as the input of the next layer.

The main role of the classifier is to relate the global features generated by the feature extractor (neocognitron) to the desired recognition code. The classifier is usually feedforward and fully connected. The classifier is usually trained by supervised training algorithm. If two images belonging to the same category of a traing set have different global features that result from the output of heighest U_{cl} sub-layer of the neocognitron, then, the classifier will associate these two different global features to the same recognition code. This is considered as the advantage of the classifier.

It can be said that the Convolutional Image recognition system used in this paper is based on the original neocognitron but with some modifications and additional parts. This new structure is called MNEO to differentiate it from the original neocognitron (see Fig. 1(b)).

4 Modification of the Neocognitron(MNEO)

Since the layers of the neocognitron in this work are independently trained, therefor, there are several possibilities for combining different kinds of neurons and learning rules. One method is proposed in this work which uses Mc Culloch-Pitts [2] neurons in S-sublayers instead of using complicated neurons based on the original neocognitron. Also kohonen's topology preserving mapping algorithm is used for parameter adaptation[4]. In order to reduce the training time, only one representation map is trained and then copy its representations to create the layer's planes. While the classifier discussed above is considered as an additional part for the original neocognitron.

4.1 Simple Model of Neurons

In contrast to the neocognitron, the network designed in this paper, the MNEO uses S-neuron based on the Mc Culloch-Pitts model. Inhibitory cells are not used and consequently S-sublayers can be easily trained by any training algorithm.

Therefore, the output of an S-cell of the k_l-*th* S-plane in the l-th module will be

$$u_{sl}(n,k_l)=\phi\left[\sum_{k_{l-1}}^{K_{l-1}}\sum_{v\in S_l} a_l(k_{l-1},v,k_l)u_{cl-1}(k_{l-1},n+v)\right] \quad (6)$$

where:

$$\phi(x)=1/(1+\exp(-x))$$

4.2 Learning algorithm of the MNEO

As mentioned earlier, since the neocognitron is used for feature extraction, the unsupervised self-organizing learning algorithm (SOM) [4] can be used to develop the representations in the S-sublayer(s). The SOM algorithm requires initializing the map size before starting of the training. Learning occurs only in S-sublayers. Essentially the algorithm modifies an unsupervised learning rule to cope with competition in a weight shared layer as follows:

First after an input has been presented to the network, the most active node (i.e. the winning neuron) is determined, second, the S-neuron connections are updated by using kohonen's rule. After learning has been completed, weight sharing is performed along the spatial to create the S-planes that represent the S-sublayer.

After learning has been completed and S-planes have been created, the input image is projected through S-sublayer. For each overlapped spatial window (each subimage), the input vector is projected to each neuron in each S-plane at the same spatial coordinate, then the most active neuron among these planes is selected .The later operation determines which feature is included in that subimage. Then the other neurons are set to zeros.

4.3 Complex Model of Neurons

For the designed network we do not care about the values and type of connections of C-cells to the input vector represented by the receptive field of the corresponding S-plane. As mentioned earlier to simplify the implementation of U_{cl} sub-layers, α is chosen to be zero. Since the inhibitory cells in the complex sublayer of the modified S-layer are not of use, therefore, the output of a C-cell of the k_l-th C-plane in the l-th module will be:

$$u_{cl}(k_l, n) = \psi\left[\sum_{v \in D_l} d_l(v).u_{Sl}(k_l, n+v)\right] \quad (7)$$

and $\psi[\ ]$ is defined as:

$$\psi[x]=\begin{cases}1 & x>0\\ 0 & x\le 0\end{cases} \quad (8)$$

5 CNN Hardware Implementation

CNN benefit, greatly from the specialized implementation compared to the implementation for general purpose processor (GPP) . Parallel, distributed processing, seen in CNN architectures, are an attractive model for computation. This model is particularly suitable for hardware design for the following reasons:

1. Inherent Parallel Processing: the final output of a network is a result of partial calculations performed by each node.
2. Simple Processing Elements: each node of the network needs only be capable of solving part of a particular problem and therefore are relatively simple.
3. Modular architecture: nodes are usually homogeneous across the network leading to simple large-scale designs.
4. Compact memory resources: the property of weights sharing enormously reduces the size for storage elements**.**

In this paper parallel digital hardware implementation of the MNEO in FPGA platform is presented in details.

5.1 Network parameters

The size and parameters of the MNEO neural network is dependent on the application. Therefore, a specified application should be determined beforehand. In this paper, the most challenging problem in image recognition is considered, the face recognition, since face is a complex character and has a great deal of information.

5.1.1 Image database

A resolution of 32x32 pixels is sufficient for the task of face recognition since a face is primarily characterized by existence of eyes, nose and mouth together with their geometrical relationship all of which can be recognized at low spatial resolution [3]. The Optical Recognition Library(ORL) database [5], which has 10 different images of 40 distinct subjects is used in this work. The images are grayscaled with a resolution of 92x112, but the resolution is reduced to 32x32. The influence of the resolution reduction in hardware field is to reduce the probability of FPGA overfitting, and to lower the information content that has to be learned by the networks and consequently reduce the required hardware resources. The designed network can classify 12 out of the 40 subjects. The experiments were performed on five training images and five testing images per person. A total of 60 training images and 60 testing images are used to adjust the parameters which presented in the next section. The training is wholly implemented in software.

5.1.2 Parameters Setting of the MNEO

Parameters setting of the MNEO such as the choice of the number of layers, neurons, cell planes, and so on, is a complex process. In fact, this process requires a lot of 'fine-tuning' effort and can be obtained by multiple simulation run of networks with different parameters. Then the most precise network is selected and its parameters are adapted. This selection is based on the evaluation of the network with respect to the recognition rate. The network which to be implemented in this paper is depicted in Fig. 2. The network structure consists of five layers, first hidden layer is a simple layer of four convolutional planes. The second hidden layer is a complex layer of also four convolutional planes. The third and fourth hidden layers are simple and complex layers respectively, each is of 16 convolutional planes. There are 12 output neurons in the last layer (fully connected feedforward layer), according to 12 different subjects (faces). Receptive fields sizes are chosen as 5x5, with 4 overlapped pixels(each field is

overlapped over another by four pixels in both horizontal and vertical directions). The features in the hidden layers are organized as (4x4)x4, with 2 overlapped pixels,(4x4)x4, with 3 overlapped pixels,(4x4)x16, with 2 overlapped pixels, and (4x4)x16.

5.1.2.1 Hardware implementation of the S-cell

In return to equation(6), the response of S-cell is simply a function of input vector $\hat{x}$ (receptive field) and weight vector $\hat{w}$ which can be written as[2]:

$$u = \phi(\sum_{i=0}^{N-1} x_i.w_i) = \phi(\hat{x} \bullet \hat{w}) = \phi\left(\frac{|\hat{x}|^2 + |\hat{w}|^2 - d^2}{2}\right) \quad (9)$$

where: $d^2 = |\hat{x} - \hat{w}|^2 = |\hat{x}|^2 + |\hat{w}|^2 - 2\hat{x} \bullet \hat{w}.$

It can be seen from the above equation that the neuron response depends on the distance between $\hat{x}$ and $\hat{w}$ .Thus the smaller the distance between them, the greater the response of the neuron. Now considering that of one specific feature each receptive field has to be detected by a number of S-cells distributed over different plans. This means, the sigmoid activation functions defined in equation(6) for those S-cells (spatially localized cells) can be replaced by a competitive function. This function sets the cell that has a minimum distance for that receptive field and resets the other cells.

The above calculation for the S-cell responses is softwarely achieved during the MNEO training phase. For the MNEO propagation phase, the same approach is used but achieved hardwarely. This ensures that the cell itself that is stimulated during the learning phase will also be stimulated during the propagation phase when the same input is applied.

In this approach, since the value of the output cell is either '0' or '1', then only one storage element is required which simplifies the successive operations and their hardware. This is because there is no need to deal with real numbers that are usually produced from the sigmoid function. This simplification also plays a positive role when implementing the downsampling operation done by complex layers.

The selection of the similarity measure is another factor that influences on the haradware implementation of the S-cell. Manhattan distance is used as a measure of similarity between $\hat{x}$ and $\hat{w}$. It measures the features that are detected by the S-cell either in training or in propagation phase. In particular, the Manhattan distance is used to avoid multiplications that are required in the calculation of Euclidean distance (the most critical operation in hardware). Also dot product between $\hat{x}$ and $\hat{w}$ is avoided when using Manhattan distance. Manhattan distance is defined [6] as:

$$d = |\hat{x} - \hat{w}| = \sum_{i=o}^{N-1} |x_i - w_i| \quad (10)$$

It can be seen from the above equation that the implementation of Manhattan distance in hardware requires absoluter, subtractor, and accumulator.

5.1.2.2 Hardware implementation of the C-cell

As it is done in simplifying the hardware implementation of the S-cell by representing its output by only one bit, the same is done with the C-cell. From equation(2) it can be seen that if the parameter α is chosen equal to 0, the output of the C-cell will be either one or zero as shown in (8). Here, also reduction of storage elements and simplification of the hardware connected to the output of the C-cell are achieved.

Depending on the representation of the S-cell and the C-cell activation functions, the C-cell can be built by only using OR function as shown in Fig. 3.

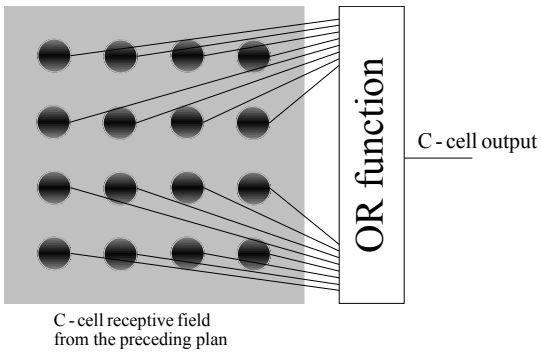


Figure 3: C-cell representation for hardware implementation

The benefits from these representations not only influence on the implementation of simple and complex layers, but also they play an important and essential role for implementing the final layer of the MNEO(the fully connected feedforwared layer).

5.1.2.3 Hardware implementation of the Feedforward-cell

In feedforward layer, the dot product calculations among the weight vectors and the receptive field vectors of the last complex layer in

the MNEO require multiplications. But the mentioned modification of the MNEO network does not need this multiplication operation. Feedforward layer only needs accumulation operation and the number of required accumulators equals to the number of the feedforward cells. Each accumulator accumulates the scalar weight of a cell if its input is '1'.
The equation for feedforward cell is:

$$P = \theta(\sum_{i=0}^{N-1} x_i.w_i) \quad (11)$$

θ is the sigmoid transfer function, P is the cell output , x_i and w_i are the input and weight vectors respectively. For example assume the feedforward cell receives 3 weighted inputs as $\hat{x}$=(1,0,1) and its trained weights are $\hat{W}$=(0.98,0.13,0.22), then the accumulator produces (0.98+0.22) which equal to 1.2. Along with the accumulator, conditioning circuit(mainly AND gates) is used to select which value of $\hat{W}$ vector is to be accumulated.

To produce cell's output, activation function (θ) (sigmoid usually used in feedforward) should be implemented and as will be shown below, its implementation will require only one multiplication operation. Some transfer functions like sigmoid function (frequently used in the MLP model) need some modifications to simplify the hardware of the function. In this case, the sigmoid function has been substituted by a piecewise linear function like satlin function [7]. The substitution is based on the selection of a linear satlin equation that has a minimum least square error with the original sigmoid function (see Fig. 4). This is achieved in software, where the sigmoid equation that is used during the learning phase is approximated with a linear satlin function that best fit it, then the last function parameters are adapted. The algorithm used to implement the selected approximated function is described as following:

```
if    x<= - th
   out=0;
elseif x>= +th
   out=1;
else
   out=0.5*(1+(x*1/th));
end;
```

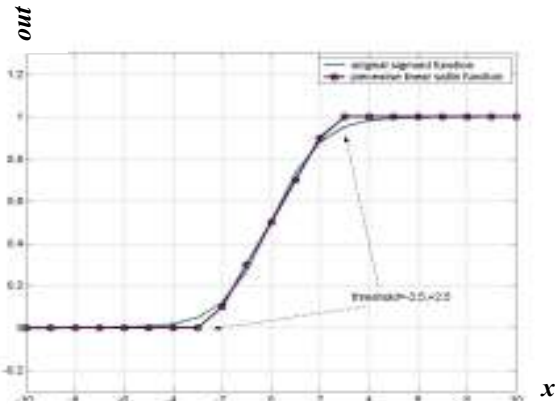


Figure 4: sigmoid and piecewise linear satlin function with minimum least square error.

Now, to implement the above algorithm, we need two multipliers and one adder. But if we simplify the linear equation to *out=0.5+(x*0.5/th)* then only one multiplier and one adder are required.

Using one multiplier and one adder for each feedforward node may be seemed a critical problem if the number of output nodes exceeds the number of embedded multipliers in the FPGA chip. To solve this problem, only one satlin unit is built and made common to all feedforward nodes, such that the output nodes use this unit sequentially in a pipelined manner.

5.2 Architecture Implementation

A parallel FPGA VLSI architecture is proposed and used to implement the hierarchical MNEO network. The main design strategy of the network and the details of the architecture prototype are explained in [8][9].

5.2.1 The building Blocks of the MNEO system

The MNEO architecture is a processor array with SIMD control. It consists of four main parts as shown in Fig. 5.

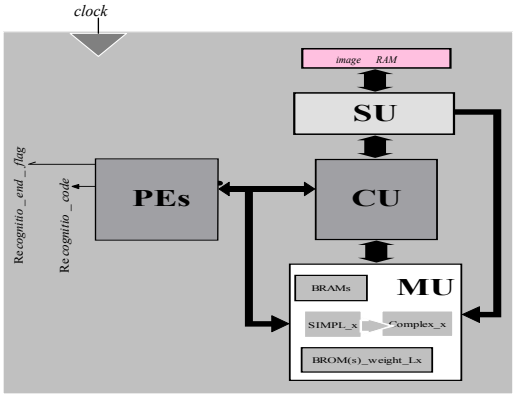


Figure 5: MNEO main building blocks

This is a fairly SIMD array configuration with one control unit (CU), one memory unit (MU), one segmentation unit (SU) and many processing element units (PEs).

SU and PEs are the most important units that determine the entire speedup of the network that is considered as the major factor of switching the implementation of ANNs from software to hardware.

6 Results and Discussion

Xilinx Spartan-3 FPGAs are used for implementations because these devices come with large internal SRAM blocks so that each block can be used for internal weight storage and for buffering the segmented image data vectors. A relatively small FPGA of type XC3S200 of 200,000 gates, 12 BRAMs and 12 (18×18bits) integrated multipliers is used to build the designed system.

6.1 MNEO performance Evaluation

In the MNEO_SIMD_FPGA system, since the system does not support learning, the Connection Per Second (CPS) and Connection-bytes-per-second (CBS)[1] are considered to evaluate the system speed performance. The speed performance of the FPGA based system among other FPGA systems depends on the operation frequency of the FPGA model used. The operation frequency of the XC3S200 FPGA model is 50 MHz, the number of input vectors that processed in parallel are five, in each clock cycle, 5 input connections of (9bits≈1byte) are evaluated by 4 weight connections of the same bit precision, then the maximum CPS and CBS achieved from the designed system are:

$$CBS=1\times1\times CPS \quad =1GCBS$$

The above performance seems reasonable and comparable with the available neural network hardware[1].

6.2 MNEO System and Face recognition

Face recognition has the benefit of being passive, non-intrusive system for verifying personal identity. The techniques used in best face recognition systems may depend on the application of the system[10].

The goal of the MNEO system is to identify particular person in real-time (e.g. in a security monitoring system, location tracking system, etc.), or to allow access to a group of people and deny access to all others (e.g. access to a building, computer, etc.). Multiple images per person are often available for training and real-time recognition is required.

The MNEO system is tested in the application of face recognition. ORL_face database were considered. The MNEO hardware system can recognize face's image with the same recognition accuracy that achieved when using the software version. This is due to the use of efficient model, its parameters setting, functions approximations and the hardware implementation such as S-neuron that is based on the realization of a competition unit. The system is trained to recognize 12 different classes. The recognition rate achieved from both software and hardware versions were equal to 93% when 60(12x5) training image and 60(12x5) testing images were used. Further recognition rate improvements can be obtained by performing more fine tuning to the MNEO parameters during learning which is implemented in software. Some techniques for fine tuning improvements can be found in [4].

7 Conclusions

In this paper we have succeeded in mapping one of the most complex neural networks (the neocognitron) on an FPGA SIMD architecture. The modifications of the neocognitron to reduce its complexity give it the possibility of realizing and processing in real-time.

The simplifications and modifications that are implemented on the original model of the neocognitron include: using low precisions fixed point for weights and inputs, and the piecewise approximation of the sigmoid transfer function that is used in the fully connected feedforward layer. In spite of these approximations, the hardware model always produces correct recognition codes that usually result from the GPP software where no precision limitations and no approximation in the actual sigmoid function.

Using the binary representation of cells outputs in all the network layers highly reduced the hardware resources required to implement the network. Consequently a relatively small FPGA model of 200,000 gate, 12 integrated BRAMs, 12 integrated 18×18 multipliers can implement the complex design of the MNEO system.

Using Manhattan distance as a competition strategy for computing the S-neuron response instead of Euclidean distance led to avoid the most FPGA bottlenecks: (the multiplications). Also using this winning take all competition strategy led to avoid calculations of sigmoid transfer function. Therefore, the Mc Culloch-Pitts S-neuron model is not used for the simple layers, although its sigmoid function can be approximated

to piece-wise linear function, since at least one multiplication is required. Therefore the bit serial arithmetic computation, stochastic computing and power of two numeric representation methods which are usually used to manipulate the multiplication problems in many neural hardware were avoided in this paper.

Replacing the computing strategy of Mc Culloch-Pitts of S-neuron by a competition based neuron can not be applied for the last feedforward layer since it is learned by delta rule. In this layer, the inner product between the input and weight vector when calculated, involves multiplications. But since the input vectors of this layer has binary values, then multiplication is replaced by weight accumulation for the input value equals to '1'.

The speedup of the 50 MHz FPGA platform over the GPP software operating on 2.4 GHz Pentium 4 computer is 88. This is due to the efficient parallel SIMD architecture of the modified MNEO. This parallelism is achieved on both data transference and data processing, while GPP uses sequential SISD architecture. The designed MNEO system is considered as a RISC processor specifics for one target problem.

For future work there are several research problems which can be
suggested and in what follows some of these are summarized:

1. To implement a larger network with fewer numbers of gates, a run time dynamic reconfigurable CNN system can be designed.
2. The plane feature can be extracted by using methods other than neural network learning such as Gabor or Wavelet techniques. Then a hybrid system can be implemented on an FPGA platform.

References:

[1]Beiu, Valeriu, "*Digital integrated circuit implementations*", Handbook of Neural Computation, release 97/1, © 1997 IOP Publishing Ltd and Oxford University Press.

[2] Neubauer, C., "*Evaluation of Convolutional Neural Networks for Visual Recognition*", IEEE Transactions on Neural Networks. vol. 9, no. 4, July 1998, pp. 685-696.

[3]Fukushima, K.; Miyake, S., "*Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position*", pattern recognition, vol. 15, no. 6, 1982, pp.455-469.

[4]Dawwd, Sh. A., "*Face Recognition using Neocognitron Neural Network*" M.Sc thesis, Univ of Mosul, Mosul, Iraq, July 2000.

[5] http://www.cam-orl.co.uk/facedatabase.html.

www.csse.monash.edu.au/courseware/cse5301/04/Lnts/L09.pdf

[6] "*NNets — L*", September 15, 2004, www.csse.monash.edu.au/courseware/cse5301/04/Lnts/L09.pdf

[7]Tommiska, M.T., "*Efficient digital implementation of the sigmoid function for reprogrammable logic*", IEE Proceedings – Computers and Digital Techniques 150, number 6, 2003, pp. 403-411.

[8]Shefa. A. Dawwd,Basil. Sh. Mahmood," FPGA Design Implementation of a Reconfigurable Architecture for Convolutional Neural Network",The 6th International Philadelphia Engineering Conference(IPEC 2006) Amman-Jordan ,19-21 Sep 2006.

[9] Sh. A. Dawwd," Software and Hardware design of Image Neurorecognizer and its FPGA Implementation", Ph.D thesis, Univ. of Mosul, Nov. 2006.

[10]Lawrence , Steve; Lee Giles, C.; Chung Tsoi, Ah ; Back, A. D., "*Face Recognition: A Hybrid Neural Network Approach*", Technical Report, UMIACS-TR-96-16 and CS-TR-3608, Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, April 1996 (Revised August 1996).