The Multi 2D Systolic Design and Implementation of Convolutional Neural Networks

Shefa A. Dawwd
Computer Engg. Dept/ Univ. of Mosul
Mosul-Iraq
shefadawwd@yahoo.com

***Abstract*- A novel Systolic Convolutional Neural Network (SCoNN) architecture suitable to be implemented in a single hardware die is proposed in this paper. A large size of memory is required to store the input and intermediate data resulted among convolutional layers in the CoNN. However, to parallelize the computations, multiple memory accesses should be achieved simultaneously. Consequently, memory bandwidth should be increased by the use of either fast expensive components or complicated managed interleaved scheme. Our proposed system can address this challenge efficiently. This is achieved due to the use of smaller memory to store the input pattern and to re-use it as an intermediate memory for inter-processing patterns that produced between two convolutional layers. Using systolic technique which can efficiently exploit its input leads to achieve this goal. A high parallelism level can be achieved with a minimal number of memory accesses. Also, the proposed computational units is designed to be suitable for processing an image of scaled resolutions. The reason behind that is attributed to using a sliding window architecture to implement the convolution process. Now apart from the size of the input pattern or window and instead of feeding the window across the image, the image is fed through the window due to using FIFO caching scheme. According to these advantages, an easily, efficient, and scalable SCoNN architecture can be implemented using small FPGA resources.**

I. INTRODUCTION

CoNN is trainable, multi-layer system that can operate at the pixel level and learn local and global features in an integrated manner. The main advantage of the CoNN is that it can extract these features with high level of invariance against scale, translation, and rotations which are the most challenging issue in image detection and recognition algorithms.

An efficient computational engine is required to implement CoNN in hardware. A VLSI system of high performance and low power is achieved in [1] by using merged/mixed analog-digital architecture. In [2], digital multiple FPGA-VLSI mixed models are combined to implemented the Neocognitron CoNN. While our previous architecture of CoNN [3] is efficiently implemented on an FPGA of reduced cost and power consumption.

In all these works a window or multiple windows of pixels of the input receptive field in a specific layer should be simultaneously accessed to be processed in parallel approach. For a single I/O port memory, this becomes critical. The alternative is to use an expensive memory(s).

Another approach can deal with window based parallel image processing algorithms. In [4], a sliding window architecture can efficiently simplify reading pixels serially and processes them in parallel.

Another issue for an efficient hardware implementation of CoNN, is to build an efficient mechanism to exchange the intermediate results among different processing layers.

The contribution of our current paper, is to build a simple and an efficient memory managed implementation of CoNN with a highly parallel computational level by employing the systolic idea. The SCoNN can be used as either a system on chip of small amount of an internal required memory(to reduce the power consumption), or a system of external memory with a minimal required accesses and bandwidth(to overcome a bus bottleneck).

II. CONVOLUTIONAL NEURAL NETWORKS

A. CoNN structure and theory

Fig.1 shows the structure of a CoNN [5]. Every layer of CoNN are spatially connected to a particular regions of the preceding layer except the final layer which are fully connected with its precedent layer. A typical CoNN consists of a set of layers, each one containing one or more planes(filters). The input image I is first approximately centered and normalized before entering at the input layer. Thus, each unit in a plane receives input from a small neighborhood in the planes of the previous layer. The CoNN is based on using two types of layers. The first type is the overlapping convolution layer which extracts C_{kl}^{i} feature. The second type is the non-overlapping down sampling layer which extract D_{kl}^{i} features. These two types of extracted feature are defined as:

$$C_{kl}^{i} = g(I_{kl}^{i} \otimes W_{kl} + B_{kl}) \quad (1)$$

$$D_{kl}^{i} = g(I \downarrow_{kl}^{i} w_{kl} + \Theta b_{kl}) \quad (2)$$

where :-

g: transfer function ; B: biases of C-layer; b: biases of D-layer;
W: weights of C-layer ; w: weights of D-layer;

I_{kl}^{i} : i[th] input of the neuron of group k of layer l;

$I \downarrow_{kl}^{i}$:down sampled i[th] input of the neuron of group k of layer l

$\otimes$:two dimension convolution; Θ :matrix composed by 1

B. Parameters Adaptation

Weights and biases of all C-layers (W and B) are trainable and can be achieved using supervised training algorithms. While weights and biases of all of D-layers (w and b)are fixed. Training the CoNN layers is not a simple

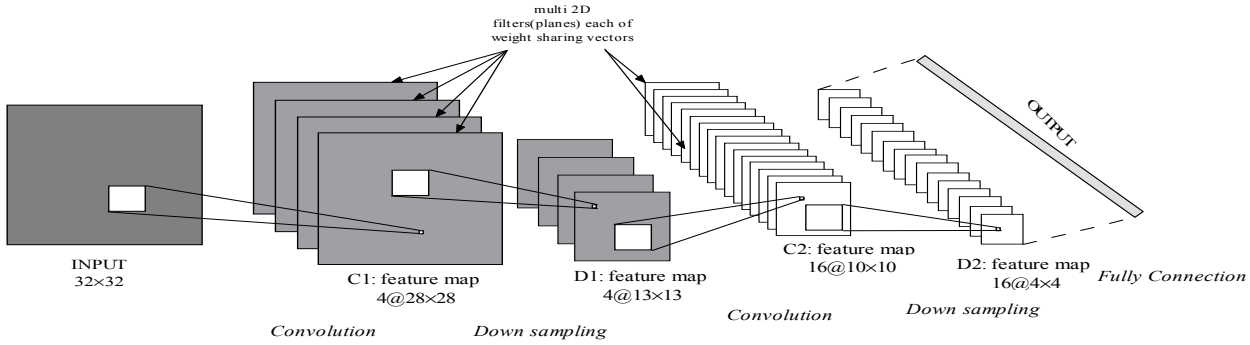

Fig.1. The CoNN structure (the receptive field for all layers are set to 5×5).

task. This is due to using different types of neurons, filters, and multiple layers.

As mentioned above, CoNN is trained usually in supervised fashion. Using the traditional back-propagation(BP) training algorithm with different neurons and layers types is a difficult task. In CoNN , a local features are extracted in the *C*-layers that located close to the input while global features are extracted in the layer that proceeded the final fully connected layer. In this paper, BP training is used only to train the fully connected layer of CoNN. While the C-layers weights are adapted by using unsupervised training approaches. 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 unsupervised competitive learning algorithm (CL) [6] can be used to develop the representations in the *C*-layer(s).

After learning has been completed, weight sharing is performed along the spatial to create the convolution filters. To keep weights precision high, training is preferred to be performed using floating point calculations which in turn are difficult to be implemented using hardware platform. Thus, all training operations for CoNN are performed in software paradigm, then the resulted weights are downloaded to the hardware storage elements.

III. Hardware Design and Implementation

A. 2D Systolic Implementation with Sliding Window and FIFOs

The systolic array processor has some distinguished features [9] such as: multiple use of each input data item; extensive concurrency usually by pipelining; a few types of simple cells; simple and regular data and control flow; high performance that can speed up a compute-bound computation in a relatively simple and inexpensive manner; and modular and expandable. The systolic structure in computer world is illustrated in Fig.2

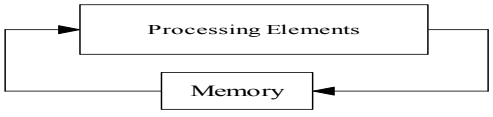

Fig.2. The basic structure of systolic processor.

In the figure shown above, one can see that the main memory has still accessed by using only one input and one output port. But the input stream to processing elements is used extensively. Processing elements may take different forms such as: linear, rectangular, and hexagonal, depending on interconnected communications among them. The above systolic feature leads to maximize processing per memory access. For image processing widowing operations, the straightforward approach is to store the entire input image of size equals to $N \times N$ into a frame buffer of at least N^2 locations. Accessing the neighborhood pixels and applying the function as needed to produce the output image. If window size equals to $w \times w$, then, $w \times w$ pixel values are required to perform the calculations each time the window is moved, and then each pixel in the image is read up to $w \times w$ times. Memory bandwidth constraints make obtaining all these pixels each clock cycle impossible unless some form of local caching is performed.

To deal with this constrain, researchers used different ways. For example, by caching the last windows in internal registers to use its pixel's data in the next windows calculations [7], or by using a high density DRAM integrated with computing logics on a single die to store the image data in wider width [8]. Or parallelizing the pixel buffering to different memory banks [3] and then another copy of neighboring pixel can be accessed from another memory bank.

In our systolic architecture, input data from the previous N rows can be cached using a w-1 numbers of FIFO buffers when the window is scanned along subsequent raster lines. This leads to the block diagram shown in Fig. 3(assume w=5). One 2D convolution process of 2D input pattern can be achieved using this architecture of systolic processor. The size of the FIFO buffer is given as N-w. To access all the values of the window for every clock cycle the w-1 FIFO buffers must be full. For every clock cycle, a pixel is read from the RAM and placed into the upper left corner location of the window.

A $w \times w$ processing elements is required to perform a function on the sliding window. Each processing element (PE) is responsible for computing the point cross correlation between a filter parameter and its corresponding pixel value. The processing operator can perform any simple operation like subtraction, or multiplication. Two storage elements are required for each PE, one to hold the filter weight element (among feature weights vector) and another to store pixel value (among the input stream).

In Fig.3, we divide the traditional sliding window architecture into two separate array. One for window pixels and another for processing elements and weights. Note that, instead of sliding the window across the image, the above implementation now feeds the image through the window. This sliding window based systolic array architecture can

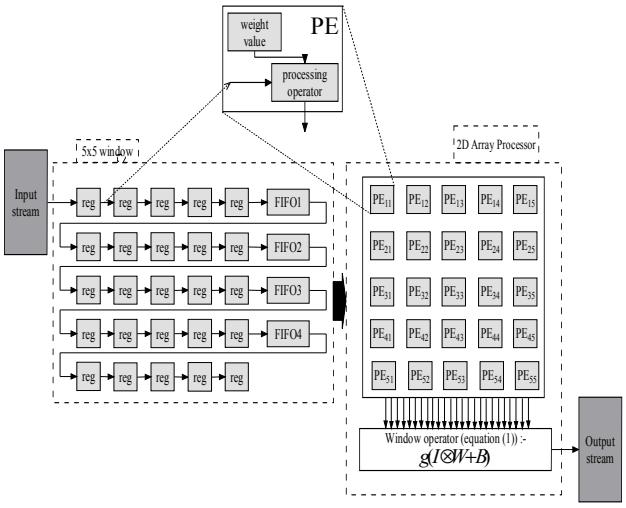

Fig.3. 2D Systolic processor with Sliding Window and FIFOs.

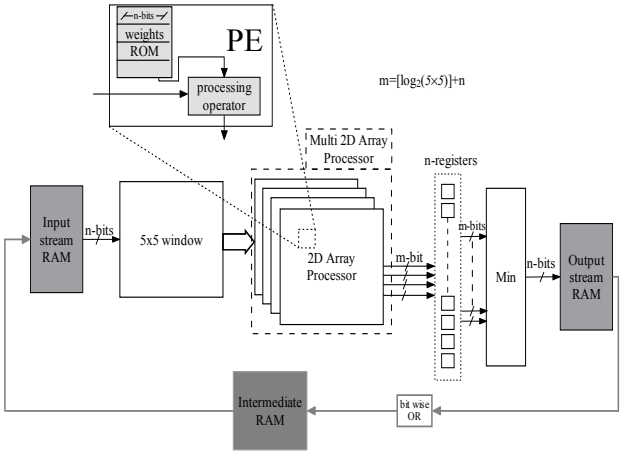

Fig.4.: A Multi 2D Systolic implementation of CoNN (SCoNN) processor with one Sliding Window and FIFOs(the architecture preserves the basic structure of systolic processor).

be used to complete the response of one CoNN convolutional filter(plane) in a specific *C*-layer. The basic processes among PE's weights and the window pixels depend on the similarity measure between input and weights vector.

Now, the function of the window operator is to sum the processing dot product values (Sum Of Products: SOP) with a previously stored value of *B* and then map the adder output value to transfer function g(.).

All the filter(plane) responses are achieved by scanning the input pattern over the 2D array architecture.

All the above operations should be repeated for other filters(k=2,3,….,n) to complete all the *C*-layer map. But, this leads to decrease the benefit properties of systolic technique. Hence, the input and output stream may duplicated for each filter map and for each layer map. Systolic algorithms have been suggested for solving compute-bound problems rather than input-output bound problems.

To keep the systolic proposal efficient, and to reduce the cost of a convolution, a new approach is used as will be presented in the following section.

B. Multi 2D Systolic Implementation of Sliding Window

The first modification of the last architecture is that the new multi 2D systolic architecture of the proposed convolution process is composited of multiple 2D array processors and only one sliding window with FIFOs (Fig.4). At this point, the input stream is kept constant, then it is exploited efficiently from every cell in the 3D architecture.

Actually, since there is no inter-processor communication along the 3rd dimension, the systolic class has still be considered as 2D. Each array processor is mapped to each filter parameters (single weight vector).

Thus, as the input slide is applied to the Multi 2D array processors, the convolution processes are achieved in parallel.

The second modification is in the computation strategy of the window feature. We simulate what was implemented in the training phase to reduce the cost of SOP and transfer function implementation. As an alternative approach, we implement the Manhattan similarity measure instead of SOP. Now each PE find difference (subtraction) between a weight and a pixel element. Hence, the bottleneck of implementing multiplication in hardware is avoided. Each window operator that related to each 2D array processor is composited of a pipelined adder tree. All the multi adder tree outputs are fed to temporary registers as an interface storage elements with a minimum unit which its output is n-bits. Each bit is associated to one convolution filter. Hence, biases(*B*) are ignored, so there is no need for expensive implementation of transfer function g(.), since that all filter outputs are obtained simultaneously and applied to minimum unit which in turn simulate the function of g(.). The output n-bits word of minimum unit is composited of one '1' and the rest are zeros. The '1' represents that a specified feature are detected and most dominated. Each output of the minimum unit is stored in an output stream RAM of n-bits width as a pixel value in raster scan. The memory representation of *C*-layer is shown in Fig.5.

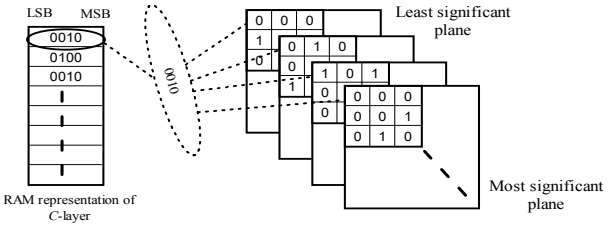

Fig.5. Memory representation of *C*-layer.

C. Hardware Implementation Cycle of SCoNN

As the calculations of C-layer has completed, the down sampling process initializes. To simplify this process, biases (*b* in equation (2)) are ignored and fixed weights(w) are set to ones for all filters associated with *D*-layers. According to the above simplification process, and since the down sampling weights are set to ones, OR function can be applied to each window in preceding *C*-layer filter. This can be performed directly by bitwise OR operation among associated addresses contents of output stream RAM. Then each result is stored in a compacted intermediate RAM. The intermediate RAM contents then can be re-applied as

inputs to the multi 2D systolic architecture for the next layer(s).

The width: n, of all memories should be configured so that it can accommodate the responses of the minimum unit which depends on the maximum number of convolutional filters associated with each of the *C*-layer. Refer to Fig.1, memory width should be configured to16-bits. If the number of the current processed layer filters less than n, then zero padding is involved.

The minimum computing unit requires the outputs of all the 2D Array Processor's tree adders to be ready. Then the minimum computation is performed also in a pipelined tree approach as in the adder of window operator.

Another factor that should be taken into accounts, is to reduce the hardware constrain of using 2D array processor for every convolution filter, and as it can be noticed in Fig. 4, four 2D processors are used in parallel to implement the convolution process for every four filters. Then if the *C*-layer consists of more than four filters, the available four processor can be used sequentially. For example, the 2nd *C*-layer consists of 16-filters, then 16/4=4 sequential sessions are required to complete its computation. Note that the weight storage element of PE is replaced on weights ROM. This ROM is configured so that for each of the sequential session, it can provide new set of weights for every 2D processor. The synchronization clock and signals that is required to manage the above data flow cycle, are provided from a central control unit.

The output stream RAM contents of the final SCoNN *D*-layer are considered as inputs to the last fully connected feed forward layer. The hardware implementation approach of this layer can be found in our previous work [10].

IV Benchmarks and Results

The design architecture is modeled by using VHDL language and implemented on Spartan3-E of 50MHz FPGA platform. The selection of FPGA is based on some properties: flexibility, re- programmability, and low design time in comparison with ASIC. Also, FPGA is provided with group of easy re-configured on-chip RAM blocks.

Hardware approaches are very different in neural networks, it is almost impossible to use the same benchmark on all system. For neural hardware architecture which do not support learning, the number of synaptic multiplication per second or *CPS* (Connections Per Second) is considered as a benchmark to measure the speed (*CPS*=*input* × *weight* × *frequency*). If the data bit precessions of both input and weight are to be taken into account, then Connection Bytes per Second (*CBS*) benchmark is considered (*CBS*= *input (bytes)* × *weight (bytes)* × *frequency*).

A CoNN with parameters shown in Fig.1 is implemented using the architecture shown in Fig.4. For approximately the same utilized silicon area ≈ 1500 slices, in comparison with our previous work presented in [3], for buffering the input vector, a reduction of four RAM units is achieved. Also, table I compares between the performance of our SCoNN with [3]. One can see that the speed of the SCoNN exceeds that is presented in [3]. In table II, the number of the memory accesses required to process input pattern of $N\times N$ pixels using window of $w\times w$ pixels for SCoNN in comparison to other methods that is mentioned in the literature is shown. In this table, one can notice that in SCoNN, the number of memory accesses is independent of window size. So, large reduction of memory accesses are achieved in comparison with other approaches. This in turn reduced the required bandwidth.

TABLE I. A Performance comparison between SCoNN and [3]

	f=50MHz		f_{max}=132 MHz	f_{max}=123 MHz
	SCoNN	[3]	SCoNN	[3]
CPS	5×10^9	1×10^9	13.2×10^9	2.46×10^9
CBS	20×10^9	4×10^9	52.8×10^9	9.84×10^9

TABLE II. A comparison of memory performance

	SCoNN	[3]	[7]
# memory accesses	$N\times N$	$N\times N\times w\times w$	$N\times N\times w$

V Conclusions

An efficient architecture is proposed to implement CoNN in hardware. The systolic parallel processing technique is used to overcome the memory constrain that is imposed to deal with input and intermediate data among different CoNN layers. The SCoNN can limit the memory size, accesses and bandwidth required to exploit the input from the computational processing elements. Also, small amount of utilized silicon area is required to implement a highly parallelism level architecture of multiple 2D array of processing elements. The architecture can be easily scaled up to fit for bigger task, and scaled down to fit for a lower available hardware.

References

[1] K. Korekado and T. Morie and O. Nomura, "A convolutional Neural Network VLSI for image Recognition Using Merged/Mixed Analoge-Digital Architecture", Knowledge-Based Intelligent Information & Engineering Systems, 2003.

[2] J. Fieres, A. Grubl, S. Philipp, K. Meier, J. Schemmel and F. Schurmann, "A Platform for Parallel Operation of VLSI Neural Networks", BICS 2004 Aug29 - Sept 1 2004.

[3] Sh. Dawwd and B. Mahmood, "A reconfigurable interconnected filter for face recognition based on convolution neural network",4th IEEE international Workshop for Design and Test, Riyadh, Saudi Arabia, 2009.

[4] T. Morie and T. Nakano,"A face/object recognition system using coarse region segmentation and dynamic-link matching", International Congress Series, vol. 1269 August, pp. 177-180, 2004.

[5] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech, and time series", The Handbook of Brain Theory and Neural Networks, MIT Press, pp. 255–258, Cambridge MA, 1995.

[6] T. Kohonen, "The self-organizing map", Proceedings of the IEEE, vol. 78, No. 9, pp. 1464-1480, 1990.

[7] A. Bruce and J. Bevwridge, "Accelerated Image Processing in FPGA", IEEE Transactions on Image Processing vol. 12, No. 12, pp. 1543-1551, 2003.

[8] D. Landis, P. Hulina, S. Deno, L. Roth and L. Coraor, "Evaluation of Computing in Memory Architectures for Digital Image Processing Applications", iccd, 1999 IEEE International Conference on Computer Design (ICCD'99), pp. 146, 1999.

[9] H. Kung, "Why Systolic Architectures?," IEEE Computer, vol. 15, No. 1, pp. 37-46, January 1982.

[10] Sh. Dawwd and B. Mahmood , "Video Based Face Recognition using Convolutional Neural Network", book chapter-in: New approaches to Characterization and Recognition of faces, InTeck Open Access Publisher, Croatia, pp. 131-152, 2011.