A-CS247-Scalable Deep Learning Accelerator Unit on FPGA

Yuvraj Singh Kanwar, Computer Science Department, San Jose State University, San Jose, CA 95112
818-625-2322, yuvraj.kanwar@sjsu.edu

ABSTRACT: Deep Learning is an emerging field of machine learning and it shows amazing ability to solve complex learning problems. Although, as the size of networks keeps increasing day-by-day, due to demands of more practical implementations, it becomes a significant challenge to create a high-performance application of deep learning neural networks. In this paper, I present the design and architecture of deep learning scalable accelerator unit (DLAU), which helps improve the performance as well as maintain low power cost by implementing large-scale deep learning networks using field-programmable gate array (FPGA) as a hardware prototype. It employs three pipelined processing units which helps improve throughput. DLAU utilizes tile techniques to explore locality for deep learning applications. The results of this research show that the DLAU accelerator achieves 36.1X speedup on Xilinx FPGA board (state-of-the-art) when compared with Intel Core2 processors, with the power consumption of 234 mW.

Index terms- deep learning, field-programmable gate array (FPGA), hardware accelerator and neural network.

I. INTRODUCTION

Machine learning is becoming pervasive in all commercial applications and research fields in the past few years, and it is able to accomplish various satisfactory products. The innovative concept of deep learning accelerated the growth of artificial intelligence and machine learning. This led to deep learning becoming a research hotspot for research organizations [1]. It uses multilayer neural network model to extract features of high-level which are generally combination of abstractions of low-level. Using deep learning we are able to search the distributed data features to solve machine learning problems. The most famously used neural models of deep learning currently being used are convolution neural networks (CNNs) [3] and deep neural networks (DNNs) [2], which have time-to-time proven to exhibit amazing capability to solve voice recognition, picture recognition, and other complex problems in machine learning.

In recent times however, the size of neural networks has become increasingly become huge in scale with demand for increased accuracy and complex requirements for the practical applications. Some examples include Google cat-recognizing system with 1 billion neuronal connections and the Baidu Brain with 100 billion neuronal connections. The great volume in the data centers make them quite power consuming. Consumption of electricity of the data centers in US, in particular, are projected to increase to roughly 140 billion kW-hr annually by 2020 [4]. This poses a serious problem to apply high performance deep learning networks with less power cost for huge deep neural network models. Today, the best means for speeding up deep learning algorithms are FPGA (field-programmable gate array), GPU (graphic processing unit) and ASIC (application specific integrated circuit). Hardware accelerators using ASIC and FPGA can accomplish at least moderate performance with lesser consumption of power compared to GPU acceleration. However, both ASIC and FPGA have I/O bandwidth, memory and computing resources in relatively limited amount, therefore it becomes harder to create massive and complex deep neural networks using hardware accelerators. There is a longer development cycle and lesser flexibility for ASIC. Chen et al. [6] in their research demonstrated DianNao, a ubiquitous machine learning hardware accelerator, which started the advent of deep learning processor. It starts a new focused neural networks approach to machine learning hardware accelerators But DianNao was not developed using reconfigurable hardware like field-programmable gate array, so it could not accommodate to different application demands. Presently, around field-programmable gate array acceleration researches, Ly and Chow [5] created field-programmable gate array-based approach to speed up the restricted Boltzmann machine (RBM). They designed exclusive hardware processing cores that were revamped for the RBM algorithm. Also, Kim et al. [7] created a field-programmable gate array-based accelerator for the RBM. They utilized several RBM processing modules in parallel, such that each module was responsible for a comparably fewer nodes. Other similar researches also demonstrate field-programmable gate array-based neural network accelerators [9]. Yu et al. [8] designed a field-programmable gate array-based accelerator, that could not achieve efficiency with changing network topologies and size. Finally, all these researches main focus was developing a particular deep learning algorithm accurately and efficiently, but accommodating increasing size of the neural networks with flexible and scalable hardware architecture is still an area that can be explored more.

To accommodate these scalability concerns, Wang et. Al. [10] designed a scalable deep learning accelerator unit which they named DLAU and was aimed to accelerate the kernel computational areas of deep learning algorithms. Particularly, they utilized pipelines, FIFO buffers, and tile techniques to reduce transfers to memory operations, and reutilize computing units to process huge sized neural networks. This approach was different from previous researches with following improvements:

1) They employed tile techniques to partition input data in order to examine locality of large-scale DL application. DLAU architecture can be built to operate different tile data sizes to leverage the tradeoffs between hardware and speedup costs. Therefore, the field-programmable gate array-based accelerator becomes more scalable to adjust various machine learning applications.
2) DLAU accelerator is designed with three fully pipelined processing units, including activation function acceleration unit (AFAU), part sum accumulation unit (PSAU) and tiled matrix multiplication unit (TMMU). DNN, CNN or any emerging neural network topology can be designed from these basic modules. Thereby, enabling higher scalability of FPGA-based accelerator as compared to ASIC-based accelerator.

II. TILE TECHNIQUES AND HOT SPOT PROFILING

RBMs have been extensively used to effectively train each deep network layer. Generally, a Deep Neural Network is composed of several hidden layers, one input layer and one classifier layer. Units in adjoining layers are all-to-all weighted and connected. Training process involves pretraining which locally adjust connection weights between the units in adjacent layers and global training which globally adjust connection weights with back propagation (BP) process. Feedforward computation is done in prediction process from input neurons to the output neurons with the current network configurations.

The large-scale Deep Neural Nets involve iterative computations that have less conditional branch operations, so, they are perfect for optimization of hardware in parallel. With this research, I first explore the hot spot using the profiler. Outcomes in Table. 1 show the percentage of running time which includes activation, vector operations and matrix multiplication (MM). For model three key operations: feed forward, RBM; and BP, MM demonstrates a significant role in overall execution. Particularly, activation function only takes 1.40%, 1.48%, and 0.42% of the three operations in comparison to MM which takes 98.6%, 98.2%, and 99.1% of the feed forward, RBM, and BP operations. Experimental conclusions on profiling show that the implementation and design of MM accelerators will provide better overall speedup of the system.

TABLE I
PROFILING OF HOT SPOTS OF DNN

Algorithms	Matrix Multiplication	Activation	Vector
Feedforward	98.60%	1.40%	
RBM	98.20%	1.48%	0.30%
BP	99.10%	0.42%	0.48%

However, Parallel processing requires considerable computing resources and memory bandwidth, which poses big challenge to FPGA implementations as compared to CPU and GPU optimization measures. Wang et al. [10] employed tile techniques to partition massive input data set to tiled subsets to tackle this problem. Tiled subset of data is buffered for processing by each designed hardware accelerator. Architecture of the accelerator is reused to support large-scale neural networks. Moreover, Computation of hardware accelerators and data access for each subset can execute in parallel. In particular, output neurons are reutilized for each partition as input neurons for next run. To create output neurons for each execution, we have to multiply input neurons to every column in weight matrix.

Algorithm 1 Pseudocode Code of the Tiled Inputs

```
Require:
  Ni: the number of the input neurons
  No: the number of the output neurons
  Tile_Size: the tile size of the input data
  batchsize: the batch size of the input data
  for $n = 0; n < batchsize; n++$ do
    for $k = 0; k < Ni; k+= Tile\_Size$ do
      for $j = 0; j < No; j++$ do
        $y[n][j] = 0;$
        for $i = k; i < k + Tile\_Size \&\& i < Ni; i++$ do
          $y[n][j] += w[i][j] * x[n][i]$
          if $i == Ni - 1$ then
            $y[n][j] = f(y[n][j]);$
          end if
        end for
      end for
    end for
  end for
```

As demonstrated by Algorithm 1, input data are partitioned to tiles and then multiplied by the corresponding weights. Then the evaluated part sum is accrued to get the outcome. We also divided the weight matrix, besides the I/O neurons, into tiles corresponding to tile size. Consequently, Hardware cost only depends on tile size thereby saving significantly on hardware resources. Thus, we are able to solve problem utilizing minimal hardware by implementing large networks with tile techniques. Also, the pipelined hardware employment is a big advantage of FPGA when compared to GPU architecture, that utilizes huge parallel SIMD architectures to better the overall throughput and performance. As per the profiling outcomes shown in Table I, during training process and prediction process in deep learning algorithms, the crucial computational parts are activation functions and MM.

III. DLAU ARCHITECTURE AND EXECUTION MODEL

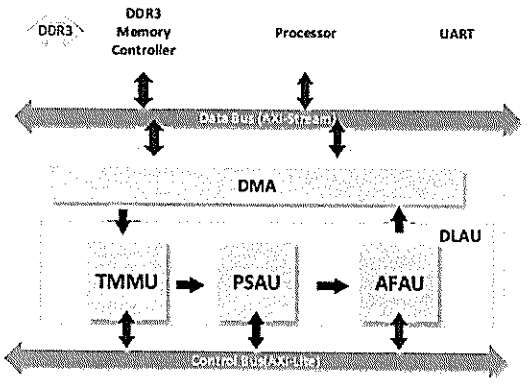

Fig. 1. DLAU accelerator architecture.

Figure 1 demonstrates DLAU system architecture which consists of a DDR3 memory controller, the DLAU accelerator, an embedded processor, and a DMA module. The DLAU is combined as a standalone unit that is adaptive and flexible to adjust different applications with configurations. Embedded processor is required for providing programming interface to the users and communicating with DLAU via JTAG-UART. Basically, it transfers the weight matrix and the input data to internal BRAM blocks, activates the DLAU accelerator, and sends back the outcomes to the user after processing. The DLAU contains three processing units organized in a pipeline manner:

1) TMMU
2) PSAU
3) AFAU

For processing, DLAU reads from memory, the tiled data, by DMA. It then calculates utilizing all the three processing units one-by-one, and then returns the outcomes back to memory. In particular, the DLAU accelerator architecture has the following key features.

1) FIFO Buffer: Each processing unit has an input buffer and an output buffer to receive or send the data in FIFO. These buffers are employed by DLAU to avoid data loss caused by conflicting or inconsistent throughput between every processing unit.
2) Tiled Techniques: Various ML applications may need particular neural network sizes. Tile technique is implemented to divide huge amount of data to small tiles that can be cached on a chip, enabling the accelerator to be applied to several neural network size. As a consequence, FPGA-based accelerator becomes scalable to adjust to different ML applications.
3) Pipeline Accelerator: Utilizing stream-like data sharing process (e.g., AXI-Stream for demonstration) to transmit data between adjoining processing units, so that TMMU, PSAU, and AFAU can calculate in stream-like way. TMMU is the primary computational unit of these three processing units and it reads the tiled nodes data and net weight through DMA, performs the computations, and then transmits the evaluated transitional part sum results to PSAU. PSAU accumulates part sums. When the accumulation is done, results are passed on to AFAU. AFAU executes the activation function by utilizing piecewise linear interpolation methods. Now, I will elaborate on the implementation of the 3 processing units, respectively.

A. TMMU Architecture

TMMU is responsible for accumulation and multiplication operations. It is specially developed to utilize the data locality of the weights and is in charge for evaluating part sums. It uses input FIFO buffer that collects the data transmitted from DMA and output FIFO buffer to dispatch part sums to PSAU. Figure 2 demonstrates the TMMU schematic diagram, in which tile size of 32 is set as an example. It first interprets the weight matrix data from input buffer into different BRAMs in 32 by the row number of the weight matrix (n = i%32 where n refers to the number of BRAM, and i is the row number of weight matrix). Then, it starts to buffer tiled node data. In the first run, TMMU interprets the 32 tiled values to register Reg_a and begins processing. TMMU reads, in parallel to the calculation in every cycle, the next node from input buffer and saves to the register Reg_b. As a consequence, the registers Reg_a and Reg_b can be used alternately.

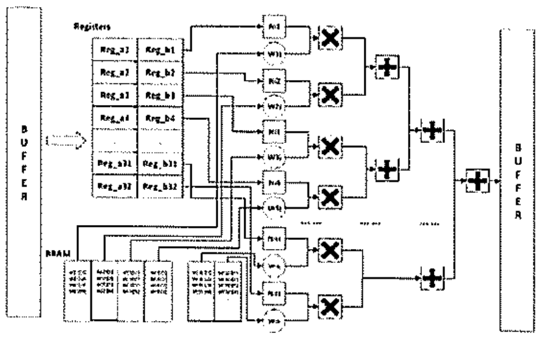

Fig. 2. TMMU schematic.

For the computations, we can use pipelined binary adder tree structure which optimizes the efficiency.

As shown in Figure 2, the node data and the weight data are maintained in registers and BRAMS. Time-sharing the coarse-grained accelerators is extensively utilized by the pipeline. Consequenntly, this design allows TMMU unit to generate a part sum outcome in each clock cycle.

B. PSAU Architecture

PSAU is in charge for accumulation operation. Figure 3 demonstrates the PSAU architecture, that accumulates the part sum generated by TMMU. If part sum is the eventual outcome, PSAU will write the result to output buffer and transmits outcome to AFAU in a pipeline manner. PSAU can accrue one-part sum in each clock cycle, consequently, the throughput of PSAU accumulation matches the generation of the part sum in TMMU.

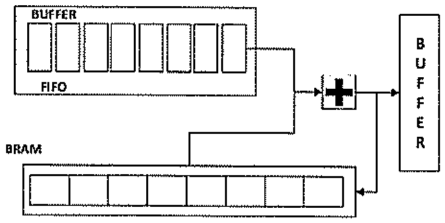

Fig. 3. PSAU schematic.

C. AFAU Architecture

The last processing unit, AFAU develops the activation function utilizing the piecewise linear interpolation (y = ai ∗ x + bi, x ∈ [x1, xi+1]). This linear interpolation technique is the most commonly applied to realize activation functions with almost insignificant efficiency loss when interval between xi and xi+1 is very small. Equation (1) shows the implementation of sigmoid function. For x > 8 and x ≤ −8, the outcomes are adequately close to the bounds of 1 and 0, respectively. For the cases in −8 < x ≤ 0 and 0 < x ≤ 8, different functions are composed.

$$f(x) = \begin{cases} 0 & \text{if } x \le -8 \\ 1 + a\left[\left\lfloor \frac{-x}{k} \right\rfloor\right]x - b\left[\left\lfloor \frac{-x}{k} \right\rfloor\right] & \text{if } -8 < x \le 0 \\ a\left[\left\lfloor \frac{x}{k} \right\rfloor\right]x + \left[\left\lfloor \frac{x}{k} \right\rfloor\right] & \text{if } 0 < x \le 8 \\ 1 & \text{if } x > 8. \end{cases} \quad (1)$$

Same like PSAU, AFAU also consists of both input buffer and output buffer which maintains throughput with other processing units. Particularly, it uses 2 different BRAMs to contain the values of a and b. The calculation of AFAU is pipelined to execute sigmoid function in each clock cycle. Consequently, the 3 processing units are fully pipelined to ensure maximum throughput.

IV. EXPERIMENTS AND DATA ANALYSIS

To measure the cost and performance of the DLAU accelerator, Wang et al. [10] have designed hardware prototype on the Xilinx Zynq Zedboard development board, which comes equipped with ARM Cortex-A9 processors clocking at 667 MHz and programmable fabrics. They used the Mnist data set to develop benchmarks by training the 784×M×N×10 DNNs in MATLAB, and use M×N layers' nodes and weights value for the input data. The results were then compared with Intel Core2 processor clocked at 2.3 GHz as the baseline.

The experiments were conducted with 32 tile size seeing the integrated hardware resources on the Zedboard development board.

1) calculated 32 hardware neurons
2) 32 weights in each cycle
3) 200 MHz clock of DLAU (one cycle takes 5 ns).
4) 3 sizes of network—64×64, 128×128, and 256×256 are tested.

A. Speedup Analysis

In this section I demonstrate speedup of DLAU and similar desgns of the deep learning algorithms in Table II. Experimental results show that the DLAU was able to accumulate performance of up to 36.1× at 256×256 network size. Kim et al. [7] and Ly and Chow [5] in comparison demonstrated the implementation only on RBM algorithms, proving that DLAU is more flexible and scalable. DianNao [6] achieved up to 117.87× performance speedup because of its high working frequency of 0.98 GHz. But it is hardwired on an FPGA platform, so, it cannot accurately adjust to various neural network sizes. Figure 4 demonstrates the performance speedup that DLAU achieved for different network sizes of 64×64, 128×128, and 256×256, respectively.

TABLE II
COMPARISONS BETWEEN SIMILAR APPROACHES

Work	Network	Clock	Speedup	Baseline
Ly&Chow [5]	256×256	100MHz	32×	2.8GHz P4
Kim et.al [7]	256×256	200MHz	25×	2.4GHz Core2
DianNao [6]	General	0.98GHz	117.87×	2GHz SIMD
Zhang et.al [3]	256×256	100MHz	17.42×	2.2GHz Xeon
DLAU	256×256	200MHz	36.1×	2.3GHz Core2

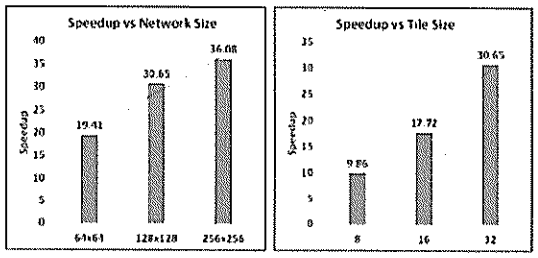

Fig. 4. Speedup at different network sizes and tile sizes.

Experimental results show a desirable ascent in performance speedup when neural networks size grows. Particularly, the acceleration increases from 19.2× in 64×64 network size to 36.1× at the 256×256 network size. The right diagram of Figure 4 shows that the tile size causes change in the performance of DLAU. It can be observed that larger tile size would mean a greater number of neurons to be calculated concurrently. At the network size of 128×128, the speedup is 9.2× when the tile size is 8. When we increase the tile size to 32, the performance speedup of 30.5× is achieved. Experimental results show that DLAU framework is scalable and configurable with different tile sizes. To achieve satisfying tradeoffs, speedup can be leveraged with hardware cost.

B. Resource Utilization and Power

Table III lists the utilization of resource by DLAU in 32×32 tile size including the DSPs, BRAM resources, LUTs and FFs.

TABLE III
RESOURCE UTILIZATION OF DLAU AT 32×32 TILE SIZE

Component	BRAMs	DSPs	FFs	LUTs
TMMU	32	158	25356	32461
PSAU	1	2	754	632
AFAU	2	7	2216	3291
Total	35	167	28326	36384
Available	280	220	106400	53200
Utilization	12.5%	75.9%	26.6%	68.4%

TMMU is very complex as compared to rest two processing units which causes it to consume most hardware resources. The overall utilization by taking limited number of hardware logic resources is reasonable. Due to use of Floating-point multiplication operations and Floating-point addition operations, DLAU utilizes 167 DSP blocks

Table IV showcases a comparison of the utilization of resources by DLAU with other two FPGA-based implementations.

TABLE IV
RESOURCE COMPARISONS BETWEEN SIMILAR APPROACHES

Implementation	FPGA	BRAMs	DSPs	FFs	LUTs
Ly&Chow [5]	XC2VP70	257	N/A	30403	29885
Kim et.al [7]	N/A	589824	18	11790	7662
DLAU	XC7Z020	35	167	28326	36384

Experimental results show that DLAU accelerator employs similar number of FFs and LUTs to Ly and Chow's work [5], however it consumes 35/257 = 13.6% on the BRAMs only. When compared with Kim et al.'s work [7], the BRAM utilization of DLAU is negligible. This happens because of utilization of tile techniques which enable large scale neural networks to be divided into small tiles, improving the scalability and flexibility of the architecture significantly. Xilinx Vivado tool set was used to evaluate the power consumption of accelerator to get power cost of each processing unit in DLAU and the DMA module. The outcomes are demonstrated in Table V which show that the net power of DLAU is only 234 mW, which is very less compared to DianNao (485 mW).

TABLE V
POWER CONSUMPTION OF THE UNITS

Component	Power	Component	Power
Accelerator-TMMU	189mW	Processor	1307mW
Accelerator-PSAU	5mW	DDR Controller	177mW
Accelerator-AFAU	25mW	Peripherals	26mW
Accelerator-DMA	15mW	Clocks	70mW
Accelerator-Total	234mW	System Total	1814mW

The conclusions we can draw from this is that DLAU is energy efficient as well as hugely scalable compared to other accelerating techniques. Next step is to compare power and energy between GPU-based accelerators and FPGA-based accelerator. This was experimented on a prototype utilizing state-of-the-art NVIDIA Tesla K40c as the baseline. K40c has 2880 stream cores that can reach peak frequency of 875 MHz, and its max memory bandwidth is 288 (GB/s). In comparison, DLAU was employed on the FPGA board working at frequency of 100 MHz. Three benchmarks, including Caltech101, Cifar-10, and MNIST, respectively were modelled using DNN to evaluate the performance of accelerators in a real DL application. Figure 5 demonstrates comparison between FPGA-based GPU+cuBLAS implementations.

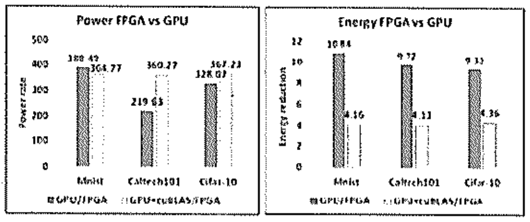

Fig. 5. Power and energy comparison between FPGA and GPU.

FPGA-based accelerator is 10× more energy efficient than GPU in terms of net energy consumption, and 4.2× better than GPU+cuBLAS optimizations. Finally, Figure 6 shows the floorplan of the FPGA chip. The left corner shows the ARM processor which is hardwired in the FPGA chip.

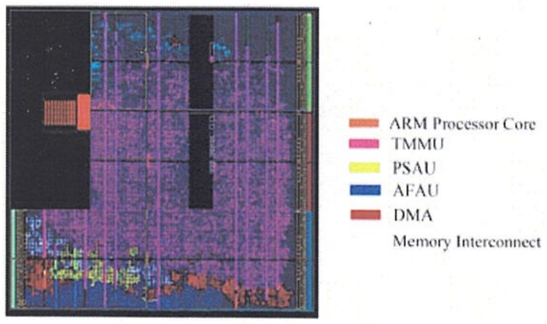

Fig. 6. Floorplan of the FPGA chip.

V. CONCLUSION

With this paper, I have demonstrated DLAU, a flexible and scalable deep learning accelerator based on FPGA. DLAU consists of 3 pipelined processing modules, that can be reutilized in large scale neural networks. It utilizes tile techniques to partition input node data into smaller tiled subsets and calculate again and again using time-sharing arithmetic logic. Experiments conducted on Xilinx FPGA prototype conclude DLAU can gather 36.1× performance acceleration with low power utilization and reasonable hardware cost.

VI. REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015

[2] C. Zhang et al., “Optimizing FPGA-based accelerator design for deep convolutional neural networks,” in Proc. FPGA, Monterey, CA, USA, 2015, pp. 161–170.

[3] J. Hauswald et al., “DjiNN and Tonic: DNN as a service and its implications for future warehouse scale computers,” in Proc. ISCA, Portland, OR, USA, 2015, pp. 27–40.

[4] P. Thibodeau. Data Centers are the New Polluters. Accessed on Apr. 4, 2016. [Online]. Available: http://www.computerworld.com/article/2598562/data-center/data-centers-are-the-new-polluters.html

[5] D. L. Ly and P. Chow, “A high-performance FPGA architecture for restricted Boltzmann machines,” in Proc. FPGA, Monterey, CA, USA, 2009, pp. 73–82.

[6] T. Chen et al., “DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning,” in Proc. ASPLOS, Salt Lake City, UT, USA, 2014, pp. 269–284.

[7] S. K. Kim, L. C. McAfee, P. L. McMahon, and K. Olukotun, “A highly scalable restricted Boltzmann machine FPGA implementation,” in Proc. FPL, Prague, Czech Republic, 2009, pp. 367–372.

[8] Q. Yu, C. Wang, X. Ma, X. Li, and X. Zhou, “A deep learning prediction process accelerator-based FPGA,” in Proc. CCGRID, Shenzhen, China, 2015, pp. 1159–1162.

[9] J. Qiu et al., “Going deeper with embedded FPGA platform for convolutional neural network,” in Proc. FPGA, Monterey, CA, USA, 2016, pp. 26–35.

[10] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie and X. Zhou, "DLAU: A Scalable Deep Learning Accelerator Unit on FPGA," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 513-517, March 2017. doi: 10.1109/TCAD.2016.2587683