DESIGN AND IMPLEMENTATION OF A SCALABLE DEEP LEARNING ACCELERATOR UNIT

[1]B. Aparna , [2]Dr. J. V. Rao
[1] Vignan Institute Of Technology & Science , [2] Vignan Institute Of Technology & Science,
[1] VLSI System Design (M.Tech), [2] Professor,Department of ECE

ABSTRACT

As the emerging field of machine learning, deep learning shows excellent ability in solving complex learning problems. However, the size of the networks becomes increasingly large scale due to the demands of the practical applications, which poses significant challenge to construct a high performance implementations of deep learning neural networks. In order to improve the performance as well to maintain the low power cost, in this paper we design DLAU, which is a scalable accelerator architecture for large-scale deep learning networks using FPGA as the hardware prototype. The DLAU accelerator employs three pipelined processing units to improve the throughput and utilizes tile techniques to explore locality for deep learning applications. Experimental results on the state-of-the-art Xilinx FPGA board demonstrate that the DLAU accelerator is able to achieve up to 36.1x speedup comparing to the Intel Core2 processors, with the power consumption at 234mW.

1. INTRODUCTION

As transistor density keeps on developing exponentially, the constrained power spending plan permits just a little division of dynamic transistors, which is alluded to as dark silicon. Dark silicon drives us to exchange silicon territory for vitality. Specific equipment speeding up has developed as a compelling strategy to moderate the dull silicon, as it conveys up to a few requests of size preferred vitality effectiveness over universally useful processors. Heading towards the enormous information period, a key test in the outline of equipment quickening agents is the manner by which to proficiently exchange information between the memory chain of importance and the quickening agents, primarily while focusing on risin.

Be that as it may, with the expanding precision necessities and many-sided quality for the down to earth applications, the extent of the neural systems turns out to be dangerously substantial scale, for example, the Baidu Brain with 100 Billion neuronal associations, and the Google feline perceiving framework with 1 Billion neuronal associations. The dangerous volume of information makes the server farms very power devouring. In this way, it postures huge difficulties to execute elite profound learning systems with low power cost, particularly for large scale profound learning neural system models. Up until this point, the stateof-the-workmanship implies for quickening profound learning calculations are Field-Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), and Graphic Processing Unit (GPU).

Contrasted and GPU quickening, equipment quickening agents like FPGA and ASIC can accomplish at any rate direct execution with bring down power

utilization. Be that as it may, both FPGA and ASIC have moderately restricted registering assets, memory, and I/O data transfer capacities, in this way it is trying to create mind boggling and huge profound neural systems utilizing equipment quickening agents. For ASIC, it has a more drawn out advancement cycle and the adaptability isn't fulfilling. Chen et al exhibits a pervasive machine-learning equipment quickening agent called DianNao, which opens another worldview to machine learning equipment quickening agents concentrating on neural systems. Be that as it may, DianNao isn't executed utilizing reconfigurable equipment like FPGA, in this way it can't adjust to various application requests. As of now around FPGA quickening looks into, Ly and Chow planned FPGA based answers for quicken the Restricted Boltzmann Machine (RBM). They made committed equipment preparing centers which are improved for the RBM calculation. Likewise Kim et al additionally built up a FPGA based quickening agent for the limited Boltzmann machine. They utilize different RBM handling modules in parallel, with every module in charge of a moderately modest number of hubs. Other comparative works likewise show FPGA based neural system quickening agents .

To handle these issues, we introduce an adaptable deep learning accelerator unit named DLAU to accelerate the portion computational parts of deep learning algorithms. Specifically, we use the tile systems, FIFO buffers, and pipelines to limit memory exchange tasks, and reuse the registering units to actualize the vast size neural networks

1.With a specific end goal to investigate the region of the profound learning application, we utilize tile systems to segment the substantial scale input information. The DLAU engineering can be designed to work diverse sizes of tile information to use the exchange offs amongst speedup and equipment costs. Subsequently the FPGA based quickening agent is more adaptable to suit distinctive machine learning algorithms.

2.The DLAU accelerator is made out of three completely pipelined handling units, including TMMU, PSAU, and AFAU. Distinctive system topologies, for example, CNN, DNN, or notwithstanding rising neural systems can be made from these essential modules. Therefore the adaptability of FPGA based quickening agent is higher than ASIC based quickening agent.

2. RESEARCH WORK

INTRODUCTION TO DLAU

In the previous couple of years, machine learning has turned out to be unavoidable in different research fields and business applications, and accomplished acceptable items. The rise of profound learning speeded up the improvement of machine learning and computerized reasoning. Subsequently, profound learning has turned into an examination problem area in inquire about associations. When all is said in done, profound learning utilizes a multi-layer neural system model to remove abnormal state highlights which are a mix of low level reflections to locate the dispersed information highlights, with a specific end goal to tackle complex issues in machine learning. As of now the most generally utilized neural models of profound learning are Deep Neural Networks (DNNs) and Convolution Neural Networks (CNNs) , which have been demonstrated to have phenomenal ability in explaining picture acknowledgment, voice acknowledgment and other complex machine learning undertakings. Be that as it may, with the expanding exactness prerequisites and many-sided quality for the functional applications, the measure of the neural systems turns out to be violently extensive

scale, for example, the Baidu Brain with 100 Billion neuronal associations, and the Google feline perceiving framework with 1 Billion neuronal associations. The unstable volume of information makes the server farms very power expending. Specifically, the power utilization of server farms in U.S. are anticipated to increment to about 140 C. In this way, it postures huge difficulties to execute superior profound learning systems with low power cost, particularly for large scale profound learning neural system models. Up until now, the state of-the-craftsmanship implies for quickening profound learning calculations are Field-Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), and Graphic Processing Unit (GPU). Contrasted and GPU quickening, equipment quickening agents like FPGA and ASIC can accomplish at any rate direct execution with bring down power utilization. In any case, both FPGA and ASIC have generally restricted figuring assets, memory, and I/O transmission capacities, in this manner it is trying to create mind boggling and gigantic profound neural systems utilizing equipment quickening agents. For ASIC, it has a more drawn out improvement cycle and the adaptability isn't fulfilling. Chen et al displays a pervasive machine-learning equipment quickening agent called DianNao, which opens another worldview to machine learning equipment quickening agents concentrating on neural systems. Yet, DianNao isn't actualized utilizing reconfigurable equipment like FPGA, hence it can't adjust to various application requests. At present around FPGA speeding up explores, Ly and Chow composed FPGA based answers for quicken the Restricted Boltzmann Machine (RBM). They made devoted equipment handling centers which are improved for the RBM calculation. Thus Kim et al additionally built up a FPGA based quickening agent for the limited Boltzmann machine. They utilize different RBM preparing modules in parallel, with every module in charge of a generally modest number of hubs. Other comparative works additionally introduce FPGA based neural system quickening agents Qi et al. show a FPGA based quickening agent yet it can't suit changing system size and system topologies. To aggregate up, these examinations center around executing a specific profound learning calculation proficiently, yet how to expand the span of the neural systems with versatile and adaptable equipment engineering has not been appropriately unraveled. To handle these issues, we introduce an adaptable profound learning quickening agent unit named DLAU to accelerate the piece computational parts of profound learning calculations. Specifically, we use the tile strategies, FIFO cushions, and pipelines to limit memory exchange tasks, and reuse the registering units to actualize the vast size neural systems. This approach separates itself from past written works with following commitments:

1. Keeping in mind the end goal to investigate the territory of the profound learning application, we utilize tile methods to parcel the expansive scale input information. The DLAU engineering can be arranged to work diverse sizes of tile information to use the exchange offs amongst speedup and equipment costs. Subsequently the FPGA based quickening agent is more versatile to suit diverse machine learning applications.
2. The DLAU quickening agent is made out of three completely pipelined handling units, including TMMU, PSAU, and AFAU.

Distinctive system topologies, for example, CNN, DNN, or notwithstanding developing neural systems can be created from these essential modules. Subsequently the adaptability of FPGA based quickening agent is higher than ASIC based quickening agent.

2.1Tile Techniques and Hot Spot Profiling

Confined Boltzmann Machines (RBMs) have been broadly used to proficiently prepare each layer of a profound system. Ordinarily a profound neural system is made out of one info layer, a few shrouded layers and one classifier layer. The units in nearby layers are all-to-all weighted associated. The forecast procedure contains feed forward calculation from given info neurons to the yield neurons with the present system setups. Preparing process incorporates pre-preparing which locally tune the association weights between the units in adjoining layers, and worldwide preparing which all inclusive tune the association weights with Back Propagation process. The extensive scale profound neural systems incorporate iterative calculations which have couple of restrictive branch activities, in this way they are reasonable for parallel improvement in equipment. In this paper we initially investigate the problem area utilizing the profiler. Results in Fig. I delineates the level of running time including Matrix Multiplication (MM), Activation, and Vector tasks. For the agent three key activities: feed forward, Restricted Boltzmann Machine (RBM), and back proliferation (BP), grid duplication assumes a noteworthy part of the general execution. Specifically, it takes 98.6%, 98.2%, and 99.1% of the feed forward, RBM, and BP activities. In examination, the enactment work just takes 1.40%, 1.48%, and 0.42% of the three tasks. Exploratory outcomes on profiling show that the plan and execution of MM quickening agents can enhance the general speedup of the framework altogether. Be that as it may, impressive memory data transfer capacity and registering assets are expected to help the parallel preparing, thusly it represents a huge test to FPGA usage contrasted and GPU and CPU streamlining measures. With a specific end goal to handle the issue, in this paper we utilize tile procedures to segment the huge information informational index into tiled subsets. Each outlined equipment quickening agent can support the tiled subset of information for preparing. So as to help the expansive scale neural systems, the quickening agent engineering are reused. In addition, the information access for each tiled subset can keep running in parallel to the calculation of the equipment quickening agents. Calculation 1 Pseudo code of the Tiled Inputs Require: Ni: the quantity of the information neurons No: the quantity of the yield neurons Tile Size: the tile size of the information batch size: the cluster size of the info information for n = 0; n <batch size; n + improve the situation k = 0; k < N I; k+ = Tile Size improve the situation j = 0; j < No; j + do y[n][j] = 0; for I = k;i< k + T ile Size&&i< N i;i + do y[n][j]+ = w[i][j] □ x[n][i] in the event that I == N I − 1 then y[n][j] = f(y[n][j]); end if end for end for end for specifically, for every emphasis, yield neurons are reused as the info neurons in next cycle. To produce the yield neurons for every emphasis, we have to duplicate the info neurons by every segment in weights grid. As showed in Algorithm 1, the info information are parceled into tiles and afterward increased by the relating weights. From that point the figured part whole are amassed to get the outcome. Other than the information/yield neurons, we additionally isolated the weight framework into tiles

comparing to the tile measure. As an outcome, the equipment cost of the quickening agent just relies upon the tile estimate, which spares huge number of equipment assets. The tiled method can take care of the issue by actualizing extensive systems with constrained equipment. In addition, the pipelined equipment usage is another favorable position of FPGA innovation contrasted with GPU engineering, which utilizes huge parallel SIMD models to enhance the general execution and throughput. As per the profiling comes about portrayed in Table I, amid the forecast procedure and the preparation procedure in profound learning calculations, the normal however critical computational parts are framework duplication and actuation capacities, thus in this paper we execute the specific quickening agent to accelerate the network augmentation and initiation capacities.

2.2 DLAU Architecture and Execution Model

Fig. 1 depicts the DLAU framework engineering which contains an implanted processor, a DDR3 memory controller, a DMA module, and the DLAU quickening agent. The installed processor is in charge of giving programming interface to the clients and speaking with DLAU through JTAG-UART.

2.2.1 TMMU Architecture

Specifically it exchanges the info information and the weight network to inner BRAM squares, actuates the DLAU quickening agent, and returns the outcomes to the client after execution. The DLAU is coordinated as an independent unit which is adaptable and versatile to suit diverse applications with setups. The DLAU comprises of 3 handling units sorted out in a pipeline way: Tiled Matrix Multiplication Unit (TMMU), Part Sum Accumulation Unit (PSAU), and Activation Function Acceleration Unit (AFAU). For execution, DLAU peruses the tiled information from the memory by DMA, registers with all the three preparing units thus, and after that composes the outcomes back to the memory. Specifically, the DLAU quickening agent design has following key highlights: FIFO Buffer: Each handling unit in DLAU has an info support and a yield cushion to get or send the information in FIFO. These supports are utilized to keep the information misfortune caused by the conflicting throughput between each.

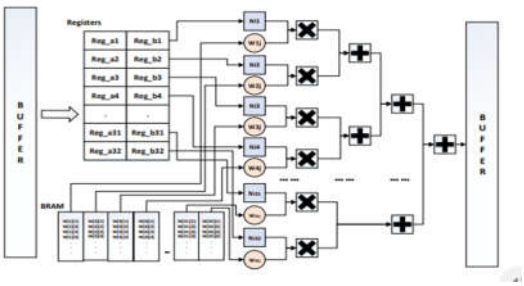


Figure 2.1 TMMU Architecture

At that point, TMMU starts to buffer the tiled hub information. In the first run through, TMMU peruses the tiled 32 esteems to registers Regan and begins execution. In parallel to the calculation at each cycle, TMMU peruses the following hub from input cushion and spares to the registers Reg b. Subsequently the registers Regan and Reg b can be utilized on the other hand. For the count, we utilize pipelined parallel snake tree structure to enhance the execution. As portrayed in Fig. 2, the weight information and the hub information are spared in BRAMs and registers. The pipeline exploits time-sharing the coarse-grained quickening agents. As an outcome, this usage empowers the TMMU unit to create a Part Sum result each clock cycle. red from DMA and a yield FIFO support to send Part Sums to PSAU.

2.2.2 PSAU architecture

Part Sum Accumulation Unit (PSAU) is responsible for the accumulation operation.

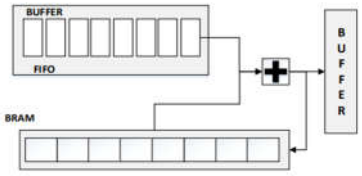


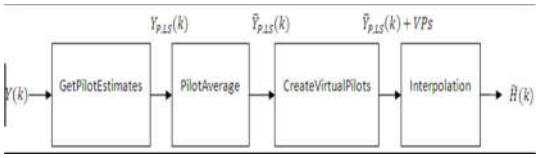
Figure 2.2 PSAU Architecture

Fig. Presents the PSAU architecture, which accumulates the part sum produced by TMMU.

On the off chance that the Part Sum is the last outcome, PSAU will compose the incentive to yield cushion and send results to AFAU in a pipeline way. PSAU can gather one Part Sum each clock cycle, subsequently the throughput of PSAU aggregation coordinates the age of the Part Sum in TMMU.

2.2.3 AFAU engineering

At long last, Activation Function Acceleration Unit (AFAU) executes the initiation work utilizing piecewise straight interjection (y=ai*x+bi, x□[x1,xi+1)). This technique has been generally connected to execute actuation capacities with immaterial precision misfortune when the interim amongst xi and xi+1 is unimportant. Eq. (1) demonstrates the usage of sigmoid capacity. For x>8 and x≤-8, the outcomes are adequately near the limits of 1 and 0, separately. For the cases in - 8



Like PSAU, AFAU likewise has both info cradle and yield cushion to keep up the throughput with other preparing units. Specifically, we utilize two separate BRAMs to store the estimations of “a, b”. The calculation of AFAU is pipelined to work sigmoid capacity each clock cycle. As a result, all the three handling units are completely pipelined to guarantee the pinnacle throughput of the DLAU quickening agent engineering.

3. IMPLEMENTATION

3.1 UART BASED DLAU

Our design specification models the concept of UART which would provide an estimation of how the DLAU operates while TX and RX is communicated via UART. Considering the fact, we provide and optimum solution for our designed application which could progress for better performance depending upon the designer specification.

In our modeled design we have consider the UART application which is accelerated faster depending upon the design parametric condition produced from the DLAU architecture. Here, the DLAU works in three different stages namely:

1. TMMU
2. PSAU
3. AFAU

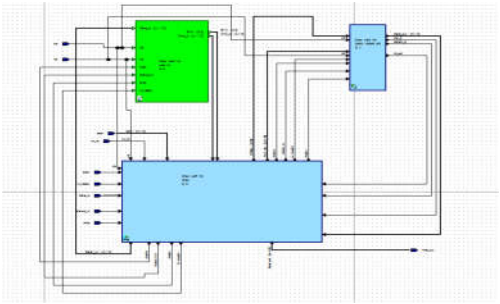


Figure 3.1 Existing model for DLAU

As form the above concept we have seen the existing design, is modeled only on DLAU architectures for different no of bits for different application, hence we are unable to provide an accurate solution for it. To provide such complex features in one application we need to know how each module is operated based on the application chosen.

Our concept have provided modules namely: TMMU, PSAU, AFAU for which need to estimate and analyze its behavior, characteristics and reliability for which the application chosen will be liable for the design considerations.

3.2 TMMU Structure design

In view of Existing design we have seen the architecture where each TMMU is utilized for FIFO design and register modeling by which each tiled operated data is processed accordingly in each registers and stored in FIFO.

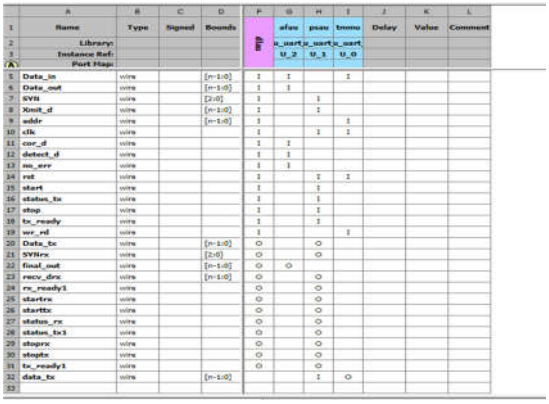
As per our design we have utilized TMMU as a static RAM with synchronous design where each data and its address are being programmed based on the designer's choice. Now our design modeling for RAM is based on the interface design which is shown in figure below:

The modeling in done via HDL designer series which would provide different ways of design based on the user choices. As we know that we have ASM, FSM, CSM etc… charts, we utilize the design using one of the charts for the synchronous RAM to provide this module as an interface based design with the utilization of spread sheet.

From the figure shown for IBD design view we could depict the response of how each signal from the input is varying and subsequently we could provide the output for other modules. The representation of IDB view very specific for the given design because it could provide a proper critical path for the design consideration.

Each variation of the input would suggest a particular data and its representation in the spread sheet for the current and next modules which are interfaced with TMMU. According to our concept we need to utilize the data from the incrementers or counters where these can be modified as sequential data or random data for the UART (TX) inputs. The control operation ad generation of the each inputs from the TX and RX is controlled by PSAU which would be explained in the next section.

3.3 IBD VISUALIZATION for TMMU



	Name	Type	Signed	Bounds	afau	psau	tmmu	Delay	Value	Comment
	Library:									
	Instance Ref:				u_uart_0	u_uart_1	u_uart_2			
	Port Map:									
5	Data_in	wire		[n-1:0]	I	I	I			
6	Data_out	wire		[n-1:0]	I	I				
7	SYN	wire		[2:0]	I		I			
8	Xmit_d	wire		[n-1:0]	I		I			
9	addr	wire		[n-1:0]	I		I			
10	clk	wire			I		I			
11	cor_d	wire			I	I				
12	detect_d	wire			I	I				
13	no_err	wire			I	I				
14	rst	wire			I		I			
15	start	wire			I		I			
16	status_tx	wire			I		I			
17	stop	wire			I		I			
18	tx_ready	wire			I		I			
19	wr_rd	wire			I		I			
20	Data_tx	wire		[n-1:0]	O		O			
21	SYNrx	wire		[2:0]	O		O			
22	final_out	wire		[n-1:0]	O	O				
23	recv_drx	wire		[n-1:0]	O		O			
24	rx_ready1	wire			O		O			
25	startrx	wire			O		O			
26	starttx	wire			O		O			
27	status_rx	wire			O		O			
28	status_tx1	wire			O		O			
29	stoprx	wire			O		O			
30	stoptx	wire			O		O			
31	tx_ready1	wire			O		O			
32	data_tx	wire		[n-1:0]			I	O		
33										

3.4 PSAU

From the design point of view we have considered the PSAU module as Data accelerated Controller. The term data acceleration refers to provide specific data parametric consideration or criteria which provide a specific set of stream of data it might results in serial with secure transmission or parallel i.e. burst mode with unsecure data transmission. The flow diagram for the PSAU is shown below which represents the different set of states for data transmission that could work on any given data specified.

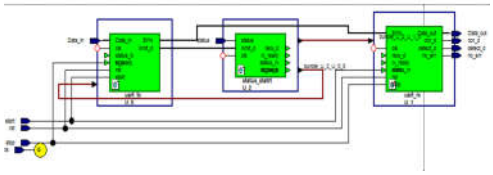
As per the design consideration we analyze the important feature for the designing of the FSM and CSM for the shown diagram below which consists of two different sections of design levels for each transmission and reception.

Now we propose the concept with the understanding that, we analyze the corresponding flow diagram as TX phase and RX phase. N Tx phase we consider the outputs from the TMMU, where theses will be the inputs to the PSAU specifically. Now the mention inputs would be processed and controlled in PSAU by which in generate

specific design inputs to the TX and RX phases simultaneously.

3.5 UART Application block diagram

From the above section we have described about how we utilize the DLAU for UART design, considering all the above facts we here by state the concept of proposed UART which is operated by status registers. Here, each status register would be interfacing the TX and RX circuit simultaneously, resulting in parallel modeling of data transfer.



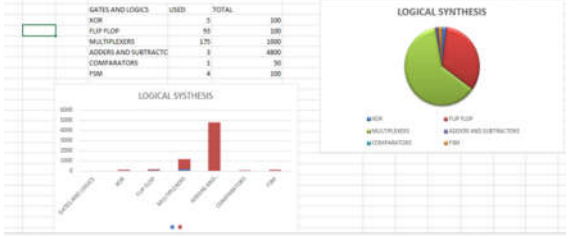
Now, we analyze the interface in such a way that each output generated would assign corresponding interrupt resulting in operation TX and RX simultaneously as shown below:

4. RESULTS

4.1 Synthesis Reports

4.1.1 Power analysis

4.1.2 Area analysis



5. CONCLUSION

As per the proposed design we have estimated and calculated the proposed design consideration with an application point of view where the DLAU is being utilized and verified with UART based modeling. We compare the results accordingly based and tabulate it. So as per design consideration we have verified our design model based on the existing design which is DLAU and with an application point of view where UART with DLAU is being considered.

Now according to the results and its implementation cycle we have shown the comparisons for the existing and proposed design model.

Hence from the result and implementation point of view we have proven our proposed method is more reliable and more effective with different kinds of application where power and area are critical.

6. REFERENCES

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

[2] 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.

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

[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.