Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

Implementation of Deep Learning Accelerator Unit

Surapaneni Aparna[1]
[1]M.Tech, Embedded Systems,
Gudlavalleru Engineering College,
Gudlavalleru, India

M. Kamaraju[3]
[3]Professor, ECE Dept,,
Gudlavalleru Engineering College,
Gudlavalleru, India

T. Venkata Lakshmi [2]
[2] Associate Professor, ECE Dept,
Gudlavalleru Engineering College,
Gudlavalleru, India

Y. Sri Chakrapani [4]
[4]Associate Professor, ECE Dept,
Gudlavalleru Engineering College,
Gudlavalleru, India

***Abstract*: Machine learning is usually used in cloud administration and apps. Also, as AI's increasing field, deep learning shows brilliant ability to address complicated learning problems. Superior executions of deep learning apps appear to be important in order to offer customers better participation. As a typical manner of speeding up calculations, FPGA uses superior, low energy, Little size and various characteristics. The DLAU Quickening Agent utilizes three performance-enhancing pipeline training devices and utilizes tile procedures to discover territory for machine learning applications. Test results on the best Xilinx FPGA panel indicate that DLAU can reach speeds of up to 36.1x relative to Intel Core2 computers and energy.**

Index Terms— Deep learning, prediction process, accelerator, neural network.

1. INTRODUCTION

Deep Learning Accelerator (DLA) is a free, open architecture that encourages with its modular architecture a conventional way of designing deep learning inference accelerator. Machine learning has recently been commonly Used in cloud services and applications such as image search, face identification, speech recognition, etc. A subset of artificial neural networks has emerged since 2006 compared to traditional state-of - the-art algorithms to obtain higher accuracy and good results across a broad spectrum of machine learning applications. Deep learning is an effective multi-layer neural network computing and intensive memory. However, with Examples include the Google cat recognition scheme (1 trillion neuronal links) and the Baidu Brain scheme (100 trillion neuronal links) to increase the precision demands and the complexity of practical apps. The high-performance application of large-scale deep-learning neural networks is therefore particularly important..

We are introducing a scalable deep learning accelerator machine called DLAU to speed up deep learning algorithm component computing in the kernel to tackle these problems. In particular, FIFO buffers and pipelines, we use tile techniques to minimize memory transfer operations and reuse software equipment to perform large-size neural networks. The previous observations characterize this technique prior Literatures:

1. We use tile methods to partition large-scale input information

2 investigate the location of the deep learning implementation. The DLAU accelerator consists of three completely pipeline handling systems, TMMU, PSAU and AFAU included

2. LITERATURE SURVEY

Deep learning accelerator system with elevated effectiveness on FPGA

Due to the requirements of practical apps, the deep learning network the range of the networks becomes increasingly big. The DLAU architecture can be designed to function distinct tile information dimensions to exploit trade-Offs between cost of speed and cost of hardware. As a result, the accelerator centered on FPGA is more scalable to fit various devices. The DLAU consists of three pipeline handling systems that can be exchanged for neural networks of big scale**.** High efficiency application of deep learning neural network is maintaining small energy costs and less delay. In this job we do the network signals samples that achieve information optimization, upgrading velocity.

An Adaptable Deep Learning Accelerator Unit (DLAU) for FPGA As the evolving machine learning sector, deep learning demonstrates great capacity to solve complicated learning issues. However, the magnitude of the networks is becoming increasingly big owing to the requirements of practical apps, which presents a important difficulty in building deep learning neural high-performance implementations. In this document we design Deep Learning Accelerator Unit (DLAU), a scalable accelerator architecture for large-scale deep learning networks using field-programmable gate array (FPGA) as a hardware prototype, in order to enhance efficiency and keep small energy costs. The DLAU accelerator uses three pipeline handling devices to enhance the performance and uses tile methods to investigate locations for deep learning apps. Experimental findings on the state-of - the-art Xilinx FPGA panel show that the DLAU accelerator can accelerate to a maximum speed of 36.1

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

compared to the Intel Core2 processors, with a power consumption of 234 mW.

3. EXISTING SYSTEM

Machine learning has become omnipresent in multiple areas of research and commercial apps over the previous few years and has produced adequate products. The advent of deep learning has accelerated the growth of artificial intelligence and machine learning. As a result, deep learning in study organizations has become a hot spot for studies. Deep learning generally utilizes a multi-layer neural high-level Features that combine low-level abstractions to identify distributed information functions to address complex machine learning issues. Deep Neural Networks (DNNs) and Convolution Neural Networks (CNNs) are currently the most commonly used neural profound learning models., Excellent ability to solve image recognition, voice recognition and other complex machine learning assignments has been demonstrated

4. PROPOSED SYSTEM

We are presenting a scalable deep learning accelerator tool called DLAU to speed up the parts of the kernel's deep learning algorithms. We specifically use tile techniques, FIFO buffers and pipelines to minimize memory transmission operations and reuse software equipment to run big neural networks.

DLAU Architecture and Execution Model

The DLAU structure design that includes an embedded processor, DDR3 memory controller, DMA module, and DLAU accelerator: it is the responsibility of the embedded processor to provide programming interfaces for users and to communicate with DLAU through JTAG-UART. In particular, it transfers input and weight matrix data to internal BRAM blocks, activates the DLAU accelerator and returns the data

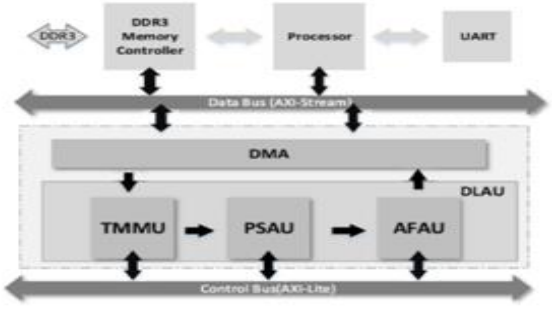


Fig 1: DLAU Architecture Topology

The DLAU is incorporated as a stand-alone device that is versatile and adaptable with settings to suit various apps.

1. Memory controller: Memory controller is a logical unit that conducts memory-based reads / writes

2. Processor: A processor or processing unit is an electronic circuit which performs operations on some external data source, usually memory or some other data stream. The term is frequently used to refer to the central processor (central processing unit) in a system

3.UART: UART is the universal transmitter asynchronous receiver. It's not a communication protocol like SPI and I2C, but a microcontroller's physical module, or a stand-alone IC. The primary aim of a UART is to send and collect serial information. One of the greatest stuff about UART is that it transmits information between appliances using only two cables.

4. DMA: Without DMA, when programmed input output is used by the CPU, it is typically fully occupied for the entire duration of the read or write operation and is therefore unavailable to perform other work. With DMA, the CPU initiates the transfer first and then performs other activities while the transfer is ongoing and lastly gets an interruption from the DMA controller when the procedure is completed.

The DLAU consists of three pipeline-based processing units, TMMU, PSAU and AFAU. DLAU reads DMA's memory tiled data for execution, then calculates the results with all three processing units and then writes them back to memory.

FIFO Buffer: For each processing unit in DLAU, there is an entry buffer and an output buffer for receiving or sending data in FIFO. These buffers are used to prevent information loss due to inaccurate results between each handling unit.

Tiled Techniques: Different apps for machine learning may involve particular dimensions of the neural network. The tile technique is used to divide the large quantity of data into tiny sheets that can be cached on a board; therefore, the accelerator can be tailored to distinct neural network sizes. Therefore, the FPGA-based accelerator is more scalable to fit various computer training apps.

Pipeline Accelerator: To migrate data between adjacent handling devices, we use stream-like data exchanging scheme (e.g. AXI-Stream display) so that TMMU, PSAU, and AFAU can calculate in streaming-like fashion. TMMU is the primary computing unit of these three computing modules, which reads through DMA the total weights and tiled node data. Calculations and additional calculations are carried out part sum outcomes are then transferred to PSAU. PSAU gathers and accumulates part sums. The findings will be carried on to AFAU once the accumulation is finished. AFAU uses linear interpolation techniques to perform the activation function. We will outline the execution of these three handling units in the remainder of this chapter, respectively.

TMMU (Tiled Matrix Multiplication Unit) Architecture

We are now able to define multiplication of matrixes. The result of these two matrices is the matrix displaying the amount of longitudinal routes between each vertical couple.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

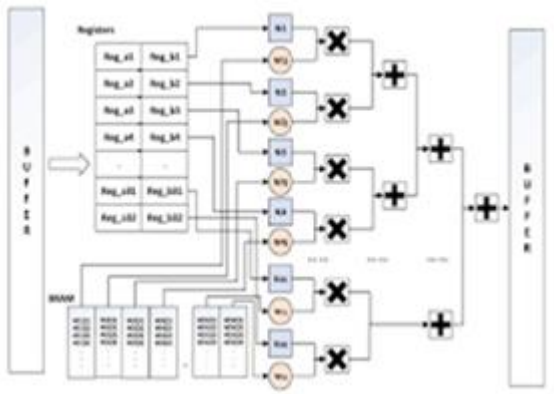


Fig 2: TMMU Architecture

Discover the matrix's row and column. Before moving on, it's a good idea to make sure you understand those two lines if the statement in the next line ends the thread if its row or column places it outside the product matrix boundaries. This will only occur in those buildings overhanging either the matrix's correct or bottom corner. We are caching the weight coefficients between two neighboring layers using the BRAM funds. The accelerator selects the matrix of weight coefficients information from the entry buffer and loops to save by the row amount of the weight matrix to distinct BRAMs in 32 (n= I percent 32 n relates to the amount of BRAMs and I relates to the amount of weight matrix row amount). Thus, the accelerator can print in conjunction 32 weight values. To decrease the effect of data access time on efficiency, we design two registers set to read the information needed for next iteration computation or alternatively use iteration in present iteration. The time to cache 32 initial points in our experiment is much lower than the calculation moment of 32 attributes. Except for the first iteration, the iteration computing will begin without waiting.

PSAU (Part Sum Accumulation Unit)

The accumulation procedure is the responsibility of PSAU. Presents the architecture of the PSAU, the sections of accumulation are generated by the TMMU. The total amount is then taken from the value s and added using PSAU is responsible for the accumulation process. Presents the PSAU architecture, the TMMU generates the accumulation parts. The complete sum from the item s is then drawn and matched using PSAU

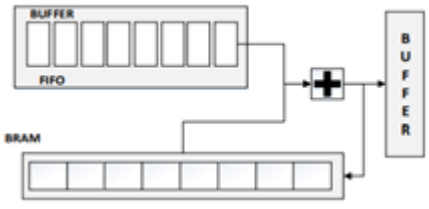


Fig 3: PSAU Architecture

PSAU accumulation is in line with the Part Sum generation in TMMU.

AFAU (Function Acceleration Unit for Activation)

Activation Function Acceleration Unit (AFAU) performs elevated accuracy computation of DNNs ; linear interpolation can perform better than other techniques, such as binomial expansion.

The linear interpolation of parts (y = ai*x+bi, x2[x1, xi+1)). Can enforce any activation function with negligible loss of precision if the interval k is tiny enough between xi and xi+1.

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

This application requires benefit of symmetry and limited scope to implement the sigmoid function. For x>8 and x -8, the results are close enough respectively to the 1 and 0 boundaries. Several functions are configured for the cases in-8<x0 and 0<x 8. In total we divide the sigmoid function into four segments.

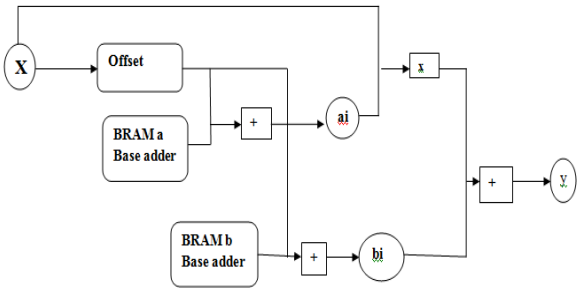


Fig 4: AFAU Architecture

Use BRAMs to store a number and b values. The variables a, b and k are corrected. After multiplication and division, locate the respective value and b value according to the value x, and get y. The computing method is pipelined and every clock cycle we can get a value.

Sampling Technique

How many samples are needed to ensure that the information in the signal is preserved? If the signal includes parts of high frequency, we will have to sample at

A greater speed to prevent the loss of message data. In particular, it is essential to sample at twice the highest signal frequency to maintain the complete data in the signal. The Sampling Theorem says that when tested at a frequency F, a signal can be accurately recorded, Where F is more than twice the signal's highest frequency. Sample the 4bit down samples in this scheme so that the process interval is reduced. and the accelerator velocity is improved.

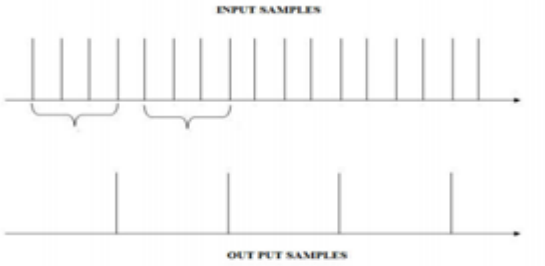


Fig 5: Sampling technique in DLAU

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

In this scheme, the image will be separated in the shape of tests, then the entry model will be limited by supplying specimens to the prototype according to this method. We apply all the entry model in this location we only apply the samples, the entry model will be coated in these samples.

Brentkung Adder

The suggested Brent-kung adder is versatile to accelerate the binary addition, and the arrangement feels like a tree structure for elevated arithmetic operations efficiency. Field programmable gate arrays[FPGA's] have been used mostly in latest years because they enhance the pace of microprocessor-based apps such as mobile communication, DSP and telecommunications. Research on the basics and motive of binary operation provides device growth. There are two phases in the building of an effective Brent-kung adder. They are phase and phase of development of pre-processing.

Pre-Processing Stage: Generate and propagate are from each couple of outputs at the pre-processing phase. The phenomenon

Propagate provides "XOR" input bits procedure and produces "AND" input bits operation[7]. The propagate (Pi) and produce (Gi) is shown in equations 1 and 2

Pi = Ai XOR Bi ——- (1)

Gi = Ai AND Bi ——- (2) below.

Generation Stage: Carry is produced at this point for each bit called hold create (Cg) and carry propagates for each bit called carry create (Cp). The carry propagates and generates for further procedure, the ultimate cell current in each procedure of the bits provides carry

The last bit to perform will assist to concurrently add the next bit up to the last bit. In the following equations3 & 4, the hold produces and carries propagate.

Cp=P1 AND P0 ---------- (3)

Cg=G1 OR (P1 AND G0) ------ (4)

The first input parts are undergoing pre-processing and will generate and propagate. These propagates and generates when the phase of generation produces carriage generates and carries propagates and then provides the ultimate amount. The effective Brent-kung adder step-by-step method is shown in Fig.6.

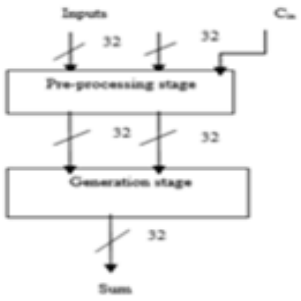


Fig 6: Block Diagram of Brent-kung Adder

The effective Brent-kung adder system for elevated quality arithmetic operations looks like a tree structure and it is the high-speed adder that relies on gate level logic. It models with the amount of doors being reduced. The delay and memory used in this architecture is therefore reduced

Extension
PASTA (Parallel Self-Timed Adder)

For each bit, the adder accepts two input operands first to perform half additions. It then iterates using previously produced carriage and amounts to execute half-additions constantly until all carriage parts are eaten and resolved at zero point.

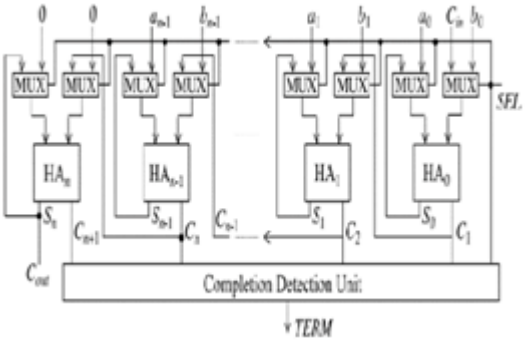


Fig 7: PASTA Architecture

Fig.7 shows the adder's overall architecture. The two-input multiplexer choice sample refers to the Req handshake signal and will be a single 0-1 SEL transformation. Using SEL= 1 it will originally pick the real operands and move to feedback / carry routes for subsequent iterations.. The HAs feedback route allows various iterations to proceed until closure when zero values are assumed by all carry signals.

State Diagrams:

For the original stage and the iterative stage of the suggested architecture, two state diagrams are traced in Fig.8. Each state is displayed by (Ci+1 Si) couple in which Ci+1, Si, respectively, depicts performance and total numbers from the ith bit adder block. The circuit during the first stage Works simply as a fundamental-mode combination HA. It is evident that state can not occur due

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

to the use of HAs instead of FAs. The feedback route through the multiplexer block is triggered during the iterative stage (SEL = 1). It is permitted to perform transformations (Ci) as many times as The recursion required to be completed. The present design can not be regarded as a fundamental mode circuit from the concept of fundamental mode circuits as the input-outputs will go Before the ultimate production is generated through several stages. It is not a Muller scheme that either operates externally outside of the fundamental phase; as shown in the state diagram, several modifications will occur. This is comparable to cyclic sequential circuits where gate delays are used to differentiate between different nations

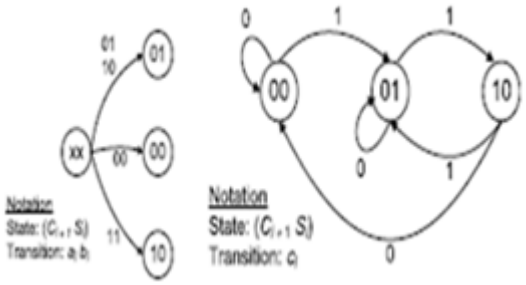
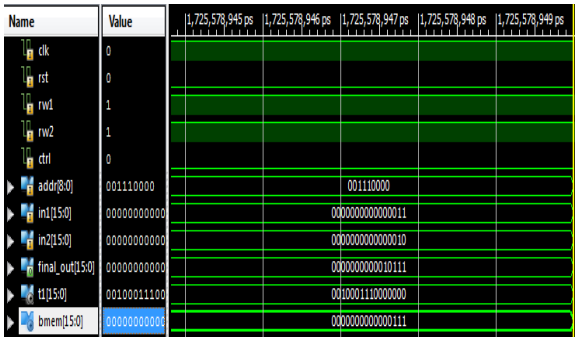


Fig 5: State Diagram of PASTA: (a) Initial Phase and (b) Iterative Phase

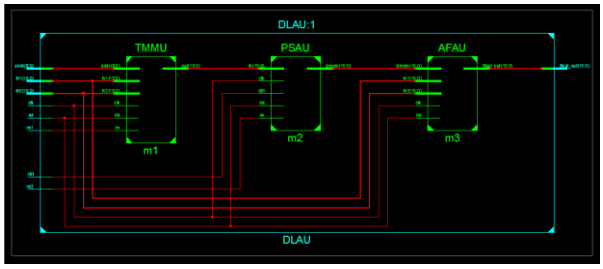
5. RESULTS

Various parts of the suggested scheme are stored in VERILOG HDL, displayed in I simulator and Xilinx ISE is the FPGA synthesis software tool.

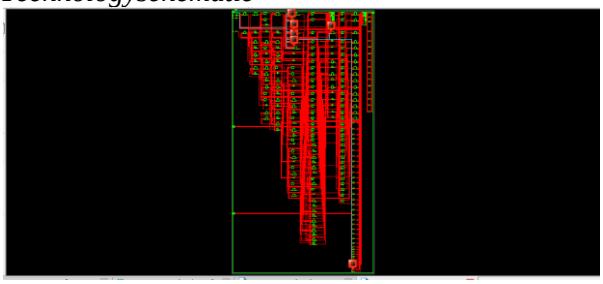
Simulation



RTLSchematic



Technologyschematic



DESIGNSUMMARY

Table: 16 bit DLAU by Using Tile Technique

Logic utilization	Used	Available	Utilization
Number of slice registers	67	4800	1%
Number of slice LUTs	63	2400	2%
Number of fully used LUT-FF pairs	31	99	31%
Number of bonded IOBs	52	102	50%
Number of BUFG/BUFGCTRL/BUFHCEs	2	16	12%
Number of DSP 48A Is	2	8	25%

TIMINGREPORT

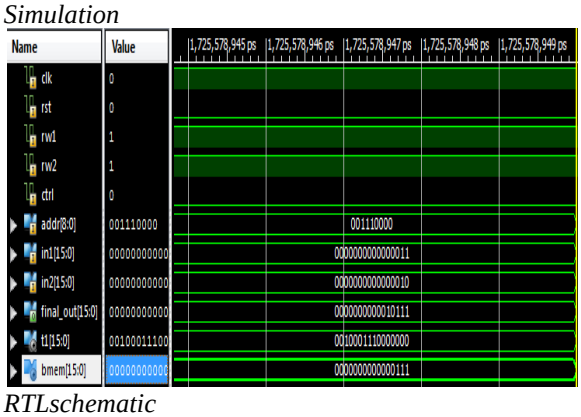
```
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 16 / 16
--------------------------------------------------------------------------------
Offset:              4.350ns (Levels of Logic = 1)
  Source:            m3/Maddsub_n0007 (DSP)
  Destination:       final_out<15> (PAD)
  Source Clock:      clk rising

  Data Path: m3/Maddsub_n0007 to final_out<15>
                                Gate     Net
    Cell:in->out      fanout   Delay    Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     DSP48A1:CLK->P15      1   1.200    0.579  m3/Maddsub_n0007 (final_out_15_OBUF)
     OBUF:I->O                 2.571           final_out_15_OBUF (final_out<15>)
    ----------------------------------------
    Total                     4.350ns (3.771ns logic, 0.579ns route)
                                      (86.7% logic, 13.3% route)
```

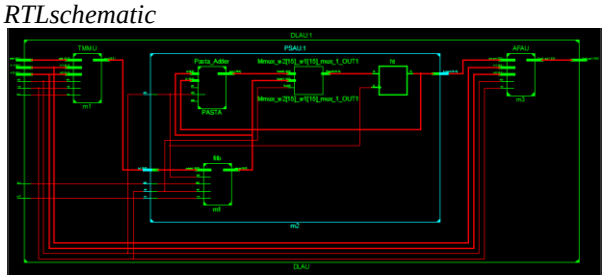
Timing report of 16 bit DLAU by Using Tile Technique

EXTENSION RESULT

Simulation

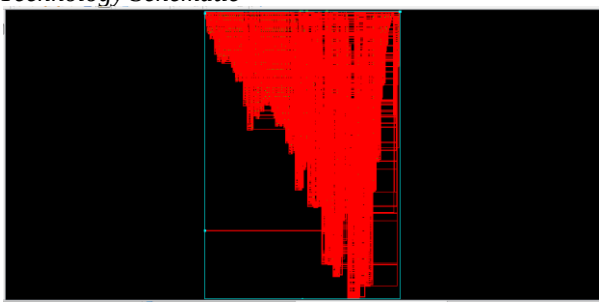


RTLschematic



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 07, July-2019

Technology Schematic



DESIGN SUMMARY

Table: 16 bit DLAU by using PASTA Technique

Logic Utilization	Used	Available	Utilization
Number of slice registers	36	4800	1%
Number of slice LUTs	38	2400	1%
Number of fully used LUT-FF pairs	20	54	37%
Number of bonded IOBs	52	102	50%
Number of BUFG/BUFGCTRL/BUFHCEs	2	16	12%
Number of DSP 48A Is	2	8	25%

TIMING REPORT

```
=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 17 / 17
-------------------------------------------------------------------------
Offset:              3.597ns (Levels of Logic = 1)
  Source:            sum_16 (FF)
  Destination:       sum<16> (PAD)
  Source Clock:      clk rising

  Data Path: sum_16 to sum<16>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FD:C->Q               1   0.447   0.579  sum_16 (sum_16)
     OBUF:I->O                 2.571          sum_16_OBUF (sum<16>)
    ----------------------------------------
    Total                      3.597ns (3.018ns logic, 0.579ns route)
                                       (83.9% logic, 16.1% route)

=========================================================================
```

Timing report of 16 bit DLAU by Using PASTA Technique

Parameters	Delay(ns)	Area(No of Slices)
Tile Technique	**4.350**	**67**
PASTA Technique	**3.597**	**36**

CONCLUSION

The Deep Learning Accelerator is a scalable and flexible. The DLAU includes three pipelined processing units, which can be reused for large scale neural networks. DLAU uses TILE techniques to partition the input node data into smaller sets and computer repeatedly by time sharing the arithmetic logic. Experimental results on Xilinx 14.5 ISE are observed. The DLAU can achieve the delay of 3.597ns by using PASTA Technique when compared to the Tile Technique with delay of 4.350ns and also compare the number of slice registers in Tile technique and PASTA technique by comparing the these two technique area can reduced by 67 slice registers to 36 slice registers.

REFERENCES

[1] chao wang,lei gong, Qi Yu, Xi Li, yuanXie, Fellow and Xuehai Zhou,DLAU "A Scalable Deep Learning Accelerator Unit On FPGA",IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 36, NO. 3, MARCH 2016

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

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] J. Qiu et al., "Going deeper with embedded FPGA platform for convolution neural network," in Proc. FPGA, Monterey, CA, USA, 2016, pp. 26–35.