DLAU: Modern Approach To FPGA Based Compatible Deep Learning Acceleration Unit

[1]Yelsetty Prashanth [2]Mr.Rajkumar D Bhure
prashanthyelsetty@gmail.com[1] bhure.rajkumar@gmail.com[2]
[1]M.Tech (VLSI-SD), J.B Institute of Engineering and Technology
[2]Associate Professor J.B Institute of Engineering and Technology.

ABSTRACT: Machine learning is widely used in applications and cloud services. And as the emerging field of machine learning, deep learning shows excellent ability in solving complex learning problems. To give users better experience, high performance implementations of deep learning applications seem very important. As a common means to accelerate algorithms, FPGA has high performance, low power consumption, small size and other characteristics. The DLAU accelerator employs three pipelined processing units to improve the throughput and utilizes tile techniques to explore locality for deep learning applications. As the extension of DLAU in the place of PASTA adder Brent Kung adder is used and experimental results on the state-of-the-art Xilinx FPGA board demonstrate that the DLAU accelerator is able to achieve speedup comparing to DLAU with PASTA.

Index Terms— Deep learning, Prediction process, accelerator, neural network

1. INTRODUCTION

Deep Learning Accelerator (DLA) is a free and open architecture that promotes a standard way to design deep learning inference accelerator with its modular architecture. Recently, machine learning is widely used in applications and cloud services, such as image search, face identification, speech recognition and so on. Since 2006, a subset of artificial neural networks has emerged as achieving higher accuracy and better results across a broad set of machine learning applications, compared with the traditional state-of the-art algorithms. Deep Learning is multilayer neural networks which are compute intensive and memory intensive. However, with the increasing of accuracy requirements and complexity of the practical applications, the size of Deep Learning networks becomes increasingly large scale, examples include the Google cat recognizing system (1 Billion neuronal connections) and Baidu Brain system (100 Billion neuronal connections). Therefore, the high performance implementation of large-scale deep learning neural networks is particularly important and becomes one of the research hotspots.

To tackle these problems, we present a scalable deep learning accelerator unit named DLAU to speed up the kernel computational parts of deep learning algorithms. In particular, we utilize the tile techniques, FIFO buffers, and pipelines to minimize memory transfer operations, and reuse the computing units to implement the large-size neural networks. This approach distinguishes itself from previous literatures with following contributions:

1. In order to explore the locality of the deep learning application, we employ tile techniques to partition the large scale input data
2. The DLAU accelerator is composed of three fully pipelined processing units, including TMMU, PSAU, and AFAU

2. LITERATURE SURVEY

Deep learning accelerator unit with high efficiency on FPGA.

The deep learning network the size of the networks becomes increasingly large scale due to the demands of the practical applications .The DLAU architecture can be configured to operate different sizes of tile data to leverage the trade-offs between speedup and hardware costs. Consequently the FPGA based accelerator is more scalable to accommodate different machines. The DLAU includes three pipelined processing units, which can be reused for large scale neural networks. High performance implementation of deep learning neural network is maintain the low power cost and less delay .in this work is we are done the samples of network signals which attain data optimization, upgraded the speed.

An Adaptable Deep Learning Accelerator Unit (DLAU) for FPGA.

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 as to maintain the low power cost, in this paper we design deep learning accelerator unit (DLAU), which is a scalable accelerator architecture for large-scale deep learning networks using field-programmable gate array (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 36.1× speedup comparing to the Intel Core2 processors, with the power consumption at 234 mW.

3. EXISTING SYSTEM

In the past few years, machine learning has become pervasive in various research fields and commercial applications, and achieved satisfactory products. The emergence of deep learning speeded up the development of machine learning and artificial intelligence. Consequently, deep learning has become a research hot spot in research organizations. In general, deep learning uses a multi-layer neural network model to extract high-level features which are a combination of low level abstractions to find the distributed data features, in order to solve complex problems in machine learning. Currently the most widely used neural models of deep learning are Deep Neural Networks (DNNs) and Convolution Neural Networks (CNNs) , which have been proved to have excellent capability in solving picture recognition, voice recognition and other complex machine learning tasks.

4. PROPOSED SYSTEM

We present a scalable deep learning accelerator unit named DLAU to speed up the kernel computational parts of deep learning algorithms. In particular, we utilize the tile techniques, FIFO buffers, and pipelines to minimize memory transfer operations, and reuse the computing units to implement the large-size neural networks.

DLAU ARCHITECTURE AND EXECUTION MODEL:

The DLAU system architecture which contains an embedded processor, a DDR3 memory controller, a DMA module, and the DLAU accelerator: The embedded processor is responsible for providing programming interface to the users and communicating with DLAU via JTAG-UART. In particular it transfers the input data and the weight matrix to internal BRAM blocks, activates the DLAU accelerator, and returns the results to the user after execution.

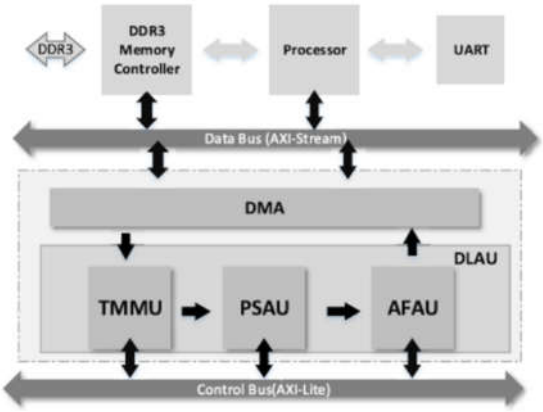

Fig 1: DLAU accelerator architecture.

The DLAU is integrated as a standalone unit which is flexible and adaptive to accommodate different applications with configurations.

1. Memory controller**:** Memory controller is a logical block that performs reads/writes from a memory based on the memory technology.
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 stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data. One of the best things about UART is that it only uses two wires to transmit data between devices.

4. DMA: Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or writes operation, and is thus unavailable to perform other work. With DMA, the CPU first initiates the transfer, and then it does other operations while the transfer is in progress, and it finally receives an interrupt from the DMA controller when the operation is done.

The DLAU consists of 3 processing units organized in a pipeline manner: Tiled Matrix Multiplication Unit (TMMU), Part Sum Accumulation Unit (PSAU), and Activation Function Acceleration Unit (AFAU). For execution, DLAU reads the tiled data from the memory by DMA, computes with all the three processing units in turn, and then writes the results back to the memory.

FIFO BUFFER:

Each processing unit in DLAU has an input buffer and an output buffer to receive or send the data in FIFO. These buffers are employed to prevent the data loss caused by the inconsistent throughput between each processing unit.

TILED TECHNIQUES:

Different machine learning applications may require specific neural net-work sizes. The tile technique is employed to divide the large volume of data into small tiles that can be cached on chip; therefore the accelerator can be adopted to different neural network size. Consequently the FPGA based accelerator is more scalable to accommodate different machine learning applications.

PIPELINE ACCELERATOR:

We use stream-like data passing mechanism (e.g. AXI-Stream for demonstration) to transfer data between the adjacent processing units, therefore TMMU, PSAU, and AFAU can compute in streaming-like manner. Of these three computational modules, TMMU is the primary computational unit, which reads the total weights and tiled nodes data through DMA, performs the calculations, and then transfers the intermediate Part Sum results to PSAU. PSAU collects Part Sums and performs accumulation. When the accumulation is completed, results will be passed to AFAU. AFAU performs the activation function using piecewise linear interpolation methods. In the rest of this section, we will detail the implementation of these three processing units respectively.

TMMU (Tiled Matrix Multiplication Unit)
ARCHITECTURE:

We are now in a position to describe matrix multiplication. The product of these two matrices is the matrix showing the number of paths of length between each pair of vertices.

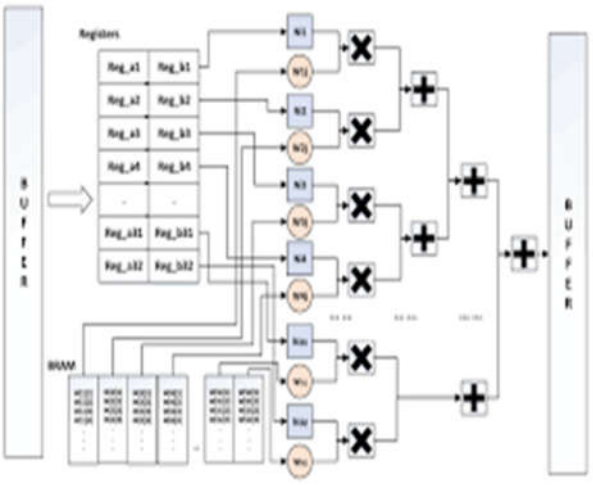

Fig 2: TMMU Architecture

Discover its row and column within the matrix. It is a good idea to make sure you understand those two lines before moving on if the statement in the next line terminates the thread if its row or column places it outside the bounds of the product matrix. This will happen only in those blocks that overhang either the right or bottom side of the matrix. We use the BRAM resources to cache the weight coefficients between two adjacent layers. The accelerator reads the matrix of weight coefficients data from input buffer, and loops to save into different BRAMs in 32 by the row number of the weight matrix (n=i%32 n refers to the number of BRAM, and i indicates the row number of weight matrix). So, the accelerator can read 32 weight values in parallel. To reduce the impact on performance of the data access time, we design two registers set to read the required data for next iteration computation or be used in current iteration alternately. In our test, the time to cache 32 input values is much less than the time of the computation of 32 values. So, the iteration computing will start without waiting except the first iteration.

PSAU (Part Sum Accumulation Unit):

Part Sum Accumulation Unit (PSAU) is responsible for the accumulation operation. Presents the PSAU architecture, the TMMU is produced the accumulation parts. Then part sum is take that value s and add that values using PSAU .will write the value to output buffer and send results to AFAU in a pipeline manner PSAU can accumulate one Part Sum every clock cycle, therefore the throughput of

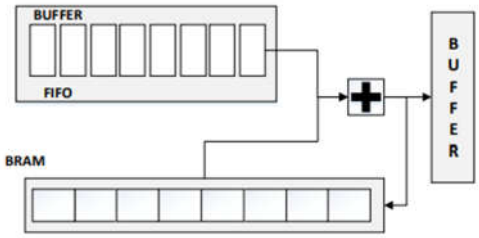

Fig 3: PSAU Architecture

PSAU accumulation matches the generation of the Part Sum in TMMU.

During PSAU addition operation Brent Kung is used in the proposed and as extenson Pasta adder ader is used.

Brentkung Adder:

The proposed Brent-kung adder is flexible to speed up the binary addition and the arrangement looks like tree structure for the high performance of arithmetic operations. Field programmable gate arrays [FPGA's] are mostly used in recent years because they improve the speed of microprocessor-based applications like mobile communication, DSP and telecommunication. Research on binary operation fundamentals and motivation gives development of devices. The construction of efficient Brent-kung adder consists of two stages. They are pre-processing stage and generation stage.

Pre-Processing Stage:

In the pre-processing stage, generate and propagate are from each pair of the inputs. The propagate gives "XOR" operation of input bits and generates gives "AND" operation of input bits [7]. The propagate (Pi) and generate (Gi) are shown in below equations 1 and 2

Pi=Ai XOR Bi ------ (1)

Gi=Ai AND Bi ------ (2)

Generation Stage:

In this stage, carry is generated for each bit is called carry generate (Cg) and carry is propagating for each bit is called carry generate (Cp). The carry propagates, and carry generate is generated for the further operation, final cell present in each bit operate gives carry. The last bit carry will help to sum of the next bit simultaneously till the last bit. The carry generates, and carry propagate are given in below equations3 & 4.

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

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

The first input bits go under pre-processing stage and they will produce propagate and generate. These propagates and generates undergoes generation stage produces carry generates and carry propagates then gives final sum. The step by step process of efficient Brent-kung adder is shown in Fig.4.

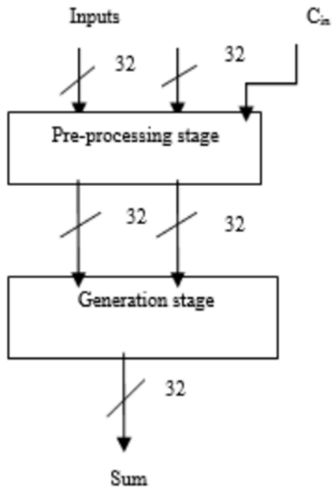

Fig 4: Block Diagram of Brent-kung Adder

The efficient Brent-kung adder arrangement is looking like tree structure for the high performance of arithmetic operations and it is the high-speed adder which focuses on gate level logic. It designs with a reduction of number of gates. So, it decreases the delay and memory used in this architecture

AFAU (Activation Function Acceleration Unit):

Activation Function Acceleration Unit (AFAU) implements the computation of DNNs does

not need high precision; Piecewise linear interpolation can achieve better performance than other methods, such as binomial expansion. The piecewise linear interpolation (y=ai*x+bi, x2[x1, xi+1)). Can implement any activation function with negligible loss of accuracy when the interval k between xi and xi+1 is insignificant is enough small.

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

Implementation of sigmoid function this implementation takes advantage of symmetry and bounded range. For x>8 and x_-8, the results are sufficiently close to the bounds of 1 and 0, respectively. For the cases in -8<x_0 and 0<x_8, different functions are configured. In total we divide the sigmoid function into four segments.

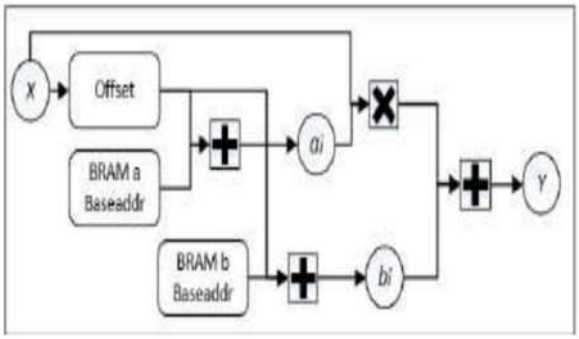

Fig 5: AFAU Architecture

Using BRAMs to store the values of a set and b set. The variables a, b and k are fixed .So find the corresponding a value and b value according the value x, and get y after multiplication and addition. The computation process is pipelined and we can get a value every clock cycle.

Extension:

PASTA (PARALLEL SELF-TIMED ADDER):

The adder first accepts two input operands to perform half additions for each bit. Subsequently, it iterates using earlier generated carry and sums to perform half-additions repeatedly until all carry bits are consumed and settled at zero level.

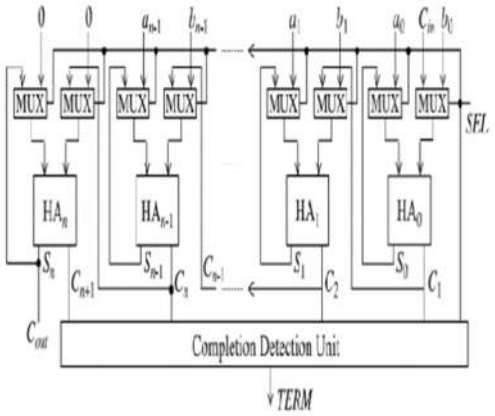

Fig 6: PASTA Architecture

The general architecture of the adder is shown in Fig.6. The selection input for two-input multiplexers corresponds to the Req handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL = 0 and will switch to feedback/carry paths for subsequent iterations using SEL = 1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values.

State Diagrams:

In Fig. 7, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represents carry out and sum values, respectively, from the ith bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state cannot appear. During the iterative phase (SEL = 1) the feedback path through multiplexer block is activated. The carry transitions (Ci) are allowed as many times as needed to complete the recursion. From the definition of fundamental mode circuits, the present design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output. It is not a Muller circuit working outside the fundamental mode either as internally; several transitions will take place, as shown in the state diagram. This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states.

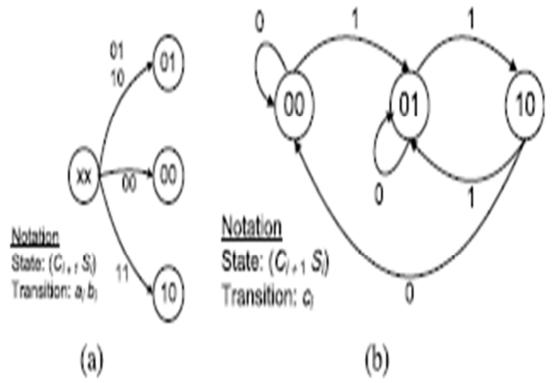

Figure 7: State diagram of PASTA (a) initial phase (b) iterative phase

5. RESULTS

Different blocks of proposed system are designed coded in VERILOG HDL, simulated in I simulator and Xilinx ISE is the software tool used for FPGA synthesis.

SIMULATION:

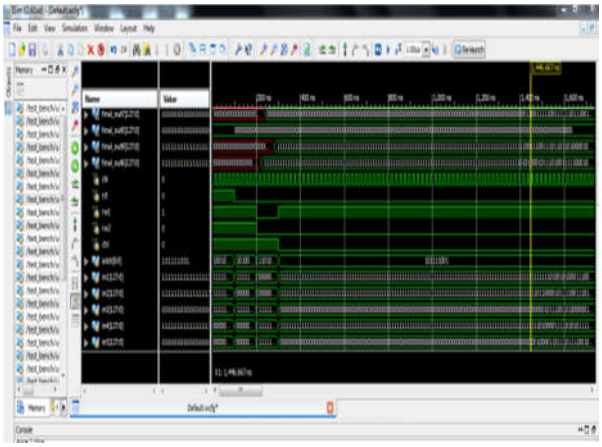

Fig 5.1Proposed Simulation output

RTL SCHEMATIC:

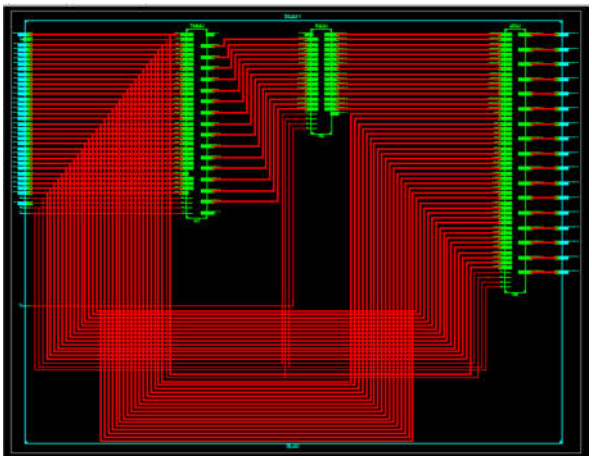

Fig 5.2 Proposed Rtl schematic

TECHNOLOGY SCHEMATIC:

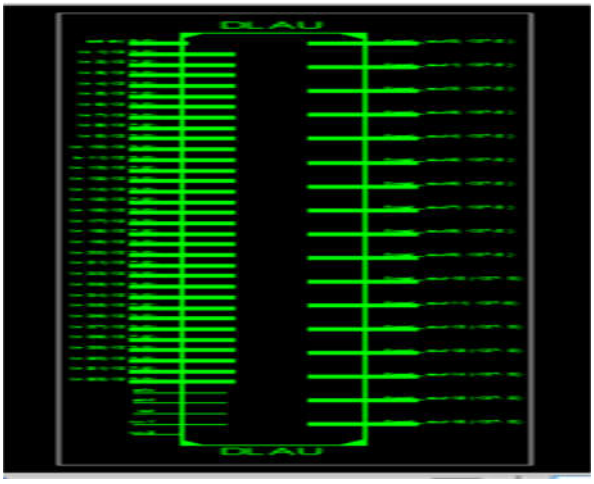

Fig 5.3 Proposed technology schematic

DESIGN SUMMARY:

```
=========================================================
*                      Design Summary
=========================================================

Top Level Output File Name         : matrix_mul.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 3089
#      GND                         : 1
#      LUT1                        : 257
#      LUT2                        : 771
#      MUXCY                       : 1028
#      VCC                         : 1
#      XORCY                       : 1031
# IO Buffers                       : 513
#      IBUF                        : 256
#      OBUF                        : 257
# DSPs                             : 56
#      DSP48E1                     : 56
```

Fig 5.4 Proposed summury

TIMING REPORT:

```
---------------------------------------------------------
Delay:               10.366ns (Levels of Logic = 224)
  Source:            matrix_input1<16> (PAD)
  Destination:       matrix_output<255> (PAD)
```

Fig 5.5 Proposed timing report

EXTENSION RESULT:

SIMULATION:

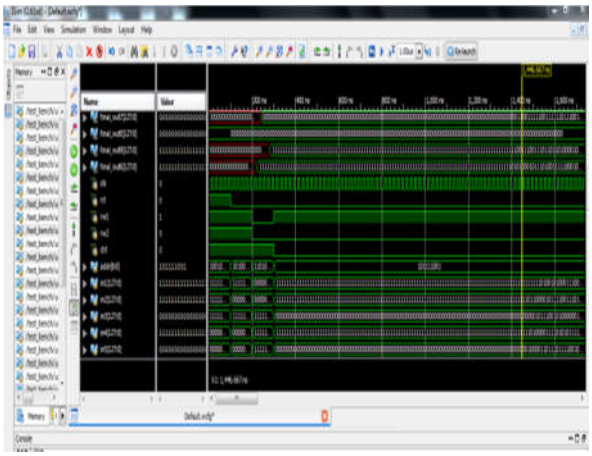

Fig 5.6 Extention simulated output

RTL SCHEMATIC:

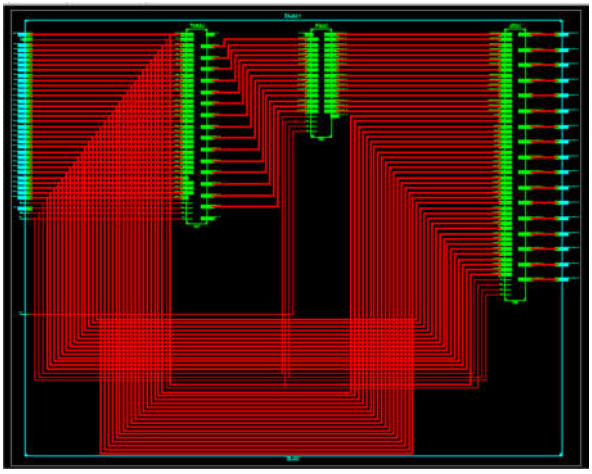

Fig 5.7 Extention RTL Schematic

TECHNOLOGY SCHEMATIC:

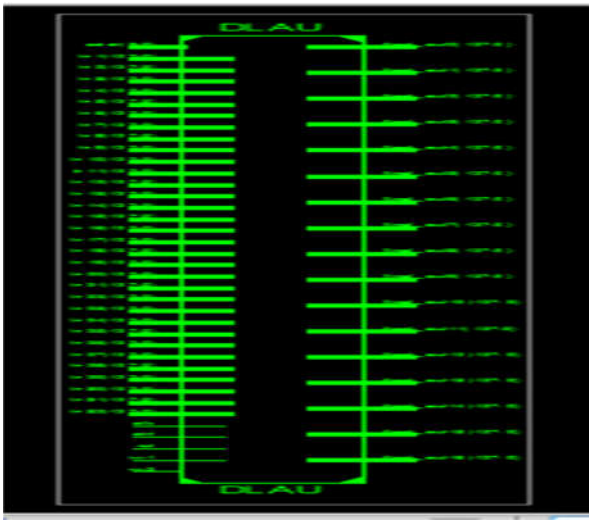

Fig 5.8 Extention Technology Schematic

DESIGN SUMMARY:

```
=========================================================
*                      Design Summary
=========================================================

Top Level Output File Name         : matrix_mul.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 3089
#      GND                         : 1
#      LUT1                        : 257
#      LUT2                        : 771
#      MUXCY                       : 1028
#      VCC                         : 1
#      XORCY                       : 1031
# IO Buffers                       : 513
#      IBUF                        : 256
#      OBUF                        : 257
# DSPs                             : 56
#      DSP48E1                     : 56
```

Fig 5.9 Design Summary

TIMING REPORT:

```
=========================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'rst'
  Total number of paths / destination ports: 4096 / 4096
---------------------------------------------------------
Offset:              3.044ns (Levels of Logic = 1)
  Source:            p66/rdata9_127 (LATCH)
  Destination:       out10<127> (PAD)
  Source Clock:      rst rising

  Data Path: p66/rdata9_127 to out10<127>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name
    ----------------------------------------  ------------
     LDE_1:G->Q           2   0.610   0.290  p66/rdata9_12
     OBUF:I->O                2.144          out10_127_OBU
    ----------------------------------------
    Total                     3.044ns (2.754ns logic, 0.29
                                      (90.5% logic, 9.5% r

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

Fig 5.10 Timing Report

Comparison Table:

SYSTEM	DELAY(ns)
PROPSED (Brent Kung Adder)	10.36ns
EXTENSION (PASTA)	3.04ns

6. CONCLUSION

In this paper, we presented DLAU, which is a scalable and flexible deep learning accelerator based on FPGA. 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 compute repeatedly by time-sharing the arithmetic logic. Experimental results on Xilinx FPGA prototype show that DLAU that speeds up with reasonable hardware cost and low power utilization.

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.

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

BIOGRAPHIES:

GUIDE DETAILS:

Mr.Rajkumar D Bhure is working as an Associate Professor in E.C.E.Department in J.B Institute of Engineering and Technology. He has 17 years of teaching experience in various technical fields. He previously worked as head of the department for E.C.E in J.B Institute of Engineering and Technology. He has delivered guest lectures in National and international level technical workshops organized by different colleges. He has presented his technical papers in various journals and conferences. He has Published book named "Basics of Electricals and Electronics Engineering" in Professional Publications.

STUDENT DETAILS:

Yelsetty Prashanth is pursuing his M.tech (VLSI-SD) in J.B Institute of Engineering and Technology.