# FPGA Data Flow Architecture for Deep Learning

Ashutosh Modi (modia), MSCS
*Department of Computer Science and Engineering*
*University of South Florida*
*Tampa, Florida 33620*
*Email: modia@mail.usf.edu*

*Abstract*— **Algorithms which try to mimic the brain were developed in late 1980s and early 1990s known as neural networks [1]. But those could not be implemented on hardware because of the technology limitations. Recent resurgence in use of neural networks is because of availability of large memory and high performance processors. And most recent advancements in GPUs made training of neural networks, up to several layers deeper than before, feasible. For some networks GPUs take training time just up to mere days for which ordinary CPUs would take months. However, field programmable gate arrays (FPGA) are emerging as an alternative to GPUs. This shift to FPGAs is because of their power efficiency while running new deep learning algorithms. Although FPGAs perform much lower than GPU, this shift is because those are more power efficient [8]. In this report we will see what change in dataflow hardware architecture helped FPGAs to convert algorithms to machine code and take 10 times less power as compared to other processors. Also see what new advancements in data flow architectures in current research are under focus.**

**Index Terms— Convolutional Neural Network, Field Programmable Gate Array, Deep Learning**

## 1. Introduction

For the last couple of years, graphics processing units (GPU) have become the benchmark for implementing artificial intelligence (AI) algorithms. Those are also being used for deep learning and computer vision application. The specialized electronic circuitry has been designed to support OpenCL and other tools to develop applications and contains large number of processing elements. But as technology in AI is improving need for fast and energy efficient processors is eminent. When it comes to multi-layer convolutional neural networks (CNN) breakthrough in specialized hardware led to state-of-the-art improvements in accuracy of traditional recognition tasks. To train and evaluate these many layered big and complex neural network requires significant computing resources. But recent growths in traditional processors are falling short to give the required performance. Field programmable gate arrays (FPGAs) are now emerging rapidly in competition to GPUs for fulfilling this hunger of deep learning algorithms with new architecture.

Recently, in view of specializing, current challenge in Microsoft research is being done to accelerate deep convolutional neural networks with servers augmented with FPGAs [3]. And they have also proved that performance of mid-range FPGA for a single node CNN accelerator is better than prior FPGA designs and high-end GPUs. Combining multiple these designs further promises to train and evaluate complex CNN model which of much larger size. In addition, this [10] paper presents scalable dataflow architecture optimized for machine vision algorithms. neuFlow and luaFlow transforms high level flow graph representation in to machine code for neuFlow.

So the idea is, for data flow architecture instruction move to their execution state when they are ready to execute. This is different from the command driven control flow execution from CPU. But there were challenges to this new paradigm and were addressed in last decade [9]. FPGAs are now much developed and are being researched for new technologies like accelerator design [2].

### 1.1. FPGA

Integrated circuits FPGA are very similar to Application Specific integrated circuits (ASICs). It provides array of programmable logic blocks which are used to implement different logic configurations. And this allows performing complex combinational tasks. This complex tasks are done with simple logic gates like AND and XOR. As claimed by company Xilinx there are many reasons to switch to Programmable devices over ASICs. FPGAs are being used for and not limited to computer vision, digital signal processing, medical imaging etc. Configuration of FPGAs is generally done using hardware description language (HDL). Current research is developing Coarse-grained architecture which connects these logic blocks with embedded microprocessors and connected peripherals [10]. With the increase in different application demands this flexibility to program the chip enables system architects and embedded software developers to apply combination of serial and parallel processing. This is the main advantage for increasingly complexity involved in processing new algorithms. The new trend provides software configurable microprocessors which provide series of processors cores and FPGA like programmable cores. To note this interesting hybrid approach is fabricated on a single chip. Mainly cost, complexity and per Watt performance. Reason FPGAs being more popular over Complex programmable logic devices (CPLDs) is it's restrictive structure leading to less flexibility. Even though there is trade-off for usage of volatile flash memory, only FPGAs can produce complex embedded functions. The new 3D or stacked architecture introduced by Tabula and Xilinx helps in shrinking the size and power consumption of FPGAs. And with modern technology it is fundamentally re-imagine the dataflow machine and eliminates the granularity caused by fine grained dataflow architecture.

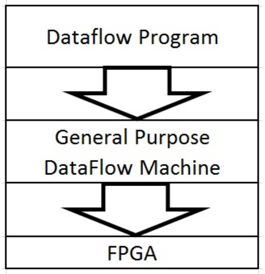
As deep learning is catching attention in machine vision industry it is evident that better hardware techniques are required to handle incredible amount of data and power involved in computing them. And FPGA provide an alternative to GPUs. Advantage of flexibility in architecture on FPGA allows researchers to explore on model level architecture which is not possible on GPUs having fixed architecture. On other hand, when deployed on large scale server based application FPGA perform better than general purpose CPUs in terms of power wattage [2].

| | Intel Xeon 2.20 GHz | | FPGA |
|---|---|---|---|
| | 1 thread - O3 | 16 thread - O3 | |
| Power (Watt) | 95.00 | 95.00 | 18.61 |
| Comparison | 5.1x | 5.1x | 1x |
| Energy (J) | 35.77 | 9.83 | 0.40 |
| Comparison | 89.4x | 24.6x | 1x |

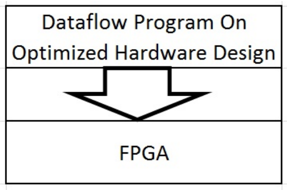
TABLE 1: Power Consumption and Energy

## 1.2. Data Flow Architecture

In computer architecture Dataflow does not follow the traditional von Neumann architecture and does not have Program counter. Order of execution is unpredictable because as and when input arguments of the instructions are available execution is done. This is achieved by having special enabling units which track the data availability. Once it is available it is sent to functional unit for processing. And hence these types of architectures are mainly used for parallel computation and database engines wherein specialized hardware is developed. As these architectures made deterministic it facilitates programmer to manage complex tasks. Since dataflow machines are combination of functional and enabling units it can be made on FPGAs. And ability of FPGAs to adopt hardware to the program, which is going to run, caught attention of researchers [6]. Merging machine logic with the program logic is the main idea. Figure 1 elicits the concept of modifying hardware for specific tasks on FPGA. This is to create a single hardware design that computes the instructions in line with the analogy of dataflow paradigm. As shown in figure 4 on page 4 approach of FIFO, dataflow architecture is suitable allowing grid of computation tiles. This paper [10] suggests the interesting aspect of grid's configuration capability. And shows a 2D grid connection of chips in which each unit shares data with 4 neighbors and also with off chip memory. And this is runtime configurable like connections, operators, Smart DMA (Direct Memory Access) modes etc. But this facility to change at run time comes at a price of reconfiguration delay.



(a) Hardware design for general purpose data flow machine



(b) Data flow program converted to an optimized design

Figure 1: Data Flow Approaches

## 1.3. Neural Network

There are many classification and regression algorithms have been developed and optimized in the field of AI. Some of the methods are linear regression, Support Vector Machines (SVM), ANN, Decision Tree, Naïve Bayes etc. Depending on the dataset size, number of features, required accuracy selection of algorithm is made. ANN is used for classification purpose and mainly when no linear classifier is possible in given dimension. Many kernel tricks are available to modify classifiers to make prediction in high dimension. High dimensionality means large number features which gets difficult to imagine for humans when more than 3 features. Looking at the constraint on other methods which uses basis functions concept of perceptron can be extended to give a multilayer neural network. This consists of perceptrons at each level connected by weight attribute. This model can be learned using back propagation and forward propagation. Goal is to find the value at connection i.e. weight to increase prediction accuracy. So when number of perceptrons increases the manipulation required is also increased.

Moreover, when network has layers more than two then all in-between levels are called as hidden layers. Input and output to these hidden layers are not intuitive to consider as any specific feature. And hence no check can be performed to grade level while training. Only final output can be evaluated to rate the trained model. So, consider if there are hundreds of layers with hundreds of perceptrons involved in some problem. This demands high performing machine along with huge memory. Parallel computation with clustering of machine is used to ameliorate the time to some extent. And when input dataset is large and can be divided to sub problems GPUs play an important role. These sub problems can be generalized to each GPU core to expedite

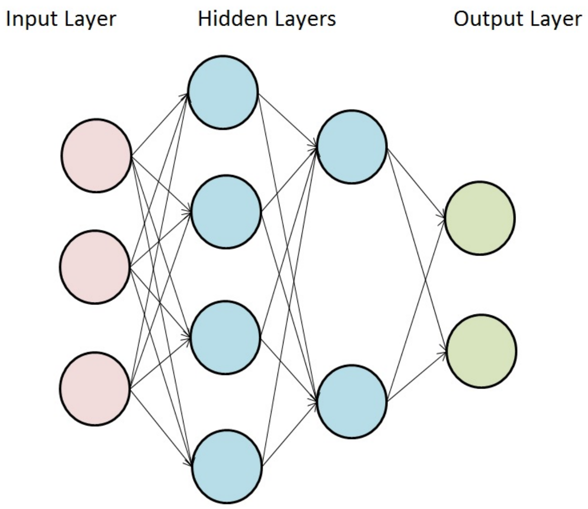


Figure 2: Neural Network

the computations. So the Deep Neural networks are these networks with large number of layers giving them a deep insight into instances. These each layer learns distinct set of features and feeds forward to get additional information from them. In CNN change is in first couple of layers where in convolution is done on windowed data. Followed by recurrent perceptrons and max pooling layers. Research has been going on to improve these models for 3D convolution [1]. This paper extends this idea to datacenters and gives example of Microsoft's 2014 Catapult project. This is a good example where in acceleration by a factor of two have been achieved using FPGA. With low power consumption and high performance FPGA accelerator increased throughput. This is one of the many examples which have made architect's interests in improving underlined hardware architecture.

## 2. Details

### 2.1. Convolutional Neural Network

Mankind has always been fascinated by the work done by human brain and it is being studied for many years. Ambition to implement a machine which can perform task of speech and image recognition was a dream in past. Today, to no surprise, with artificial neural network (ANN) this is very easy. These neural nets are consisting of many layered different computational units and resemble to what neurons do in human brain. These units are connected by weights which are learned with different algorithms and methods like backward and forward propagation. Nowadays, one of the hot topics of research in machine learning is Deep Learning which has multi-level processing and creates high level abstraction model. This emerging field is promising better results over traditional efficient algorithms like unsupervised learning or semi-supervised learning. These have revolutionized machine vision applications. Convolutional networks are the inspiration from biological process and use multilayer perceptron design. CNNs are similar to ANNs. With deep learning algorithms they are not restricted to detect one object but as layers grows data is checked very finely to identify many individual objects [4]. Figure 4 on the next page depicts the naïve implementation of CNN on FPGA.

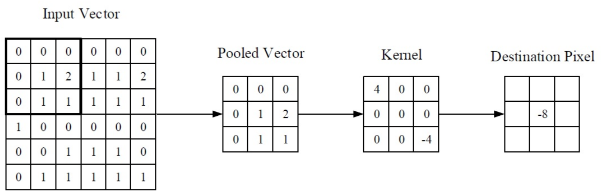


Figure 3: Convolution layer

As shown in the figure 3 Kernel is nothing but a mask which is being moved over the image. Manipulation of weighted average or depending upon the defination of kernel centre pixel gets replaced at each iteration. Idea is to make the dimensions of an image small and retaining the relevant information. [11]

CNN are based on how the visionary mechanisms work in most of the living beings. Visual processing part in cortex responsible for operating small part of captured light i.e. on sub regions. In 1980 Kunihiko Fukushima introduced CNN. And because of the unavailability of the computing infrastructure it was remain as concept until GPU made it possible [12]. Most of the deep learning involves ANN as their model. Recurrent Neural Networks are used in language modelling and CNN are used in computer vision as they are very successful in those. But CNN are not limited for vision and used for speech recognition. CNN also has pooling layers and connected by weights but the only difference is that they have fully connected convolutional layers. Structure of CNN allows input data 2D format to process easily. Since these have convolutional layer in start less weight training remains as compared to other NN. This is one of the reasons why CNNs are getting attraction nowadays.

Idea is to use small windows and scan the data sequentially. This small portion under observation is called as receptive fields. Results from these windows are then grouped to form better representation of the input data. This may not be a good indication to humans in terms of appearance but for to understand the no intuitive features. This transition may contain local or global pooling layers. All these arrangement is inspired by how the biological process happens. But unlike billions of connections, CNNs have convolution method to reduce them. Moreover, convolution involves of data required for all windows, memory requirement reduces and helps in improving performance.

Different types of layer:

1) Convolutional layer:
2) ReLU layer:

3) Pooling layer:
4) DropoutLayer:

Caffe is the most popular library for CNN which is developed by Berkeley Vision and Learning Centre. This works on CPU as well as GPU and has easy to understand architecture with faster speed.

One of the most popular research projects in deep Learning in Google Brain which is started in 2011 as Google X project. This was the initiation by Andrew Ng, Jeff Dean, Greg Corrado et. Al. In june 2012 the New York Times reported the success of cluster of 16000 computers. Result was based on 10 million images of cat captured from YouTube and system was able to mimic some aspects of human brain [13].

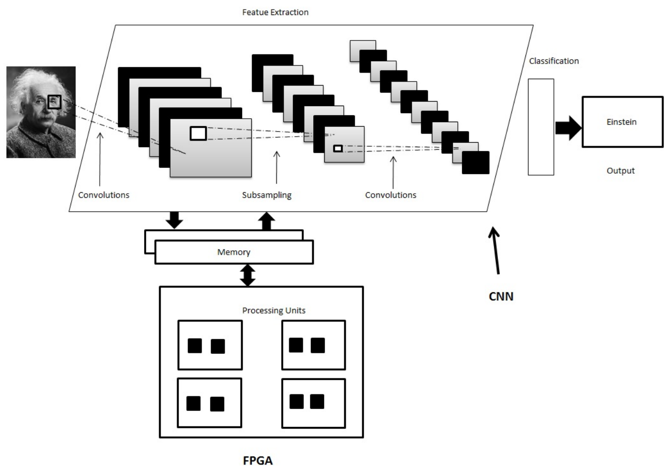


Figure 4: CNN on FPGA

Along with Caffe there are many deep learning tools like Torch, Deeplearning4j, Theano etc. Theano uses NumPy-like syntax and are compiled to run on any architecture like GPU or CPU. Deeplearning4j is another example of open source library for deep learning for Java and Java Virtual Machine [14]. With increase in interest for Deep learning integration with Apache Spark, Hadoop/YARN, integration with CUDA has increased. And many applications are now configured with Python, Scala etc for distributed computing.

Algorithm[1] 1 is one of the examples proposed by Chen Zang et.al.

## 2.2. FPGA History and Development

These can be programmed as per the customer's requirement once developed. Is consists of logic blocks and distributed with routing channels, interconnected by data buses and direct connections. Each logic block consists of a look up table and a flip flop. All modern chips are based on

1. This algorithm is with reference to the one discussed in paper by Chen Zang [2]. Paper also propose algorithms which are optimized and use concept of tiling; also considers loop carried dependencies.

**Algorithm 1** CNN Pseudo Code

```
//START
for(row=0;row<R;++row)
  for(col=0;col<C;++col)
    for(to=0;to<M;++to)
      for(ti=0;ti<N;++ti)
        for(i=0;i<K;++i)
          for(j=0;j<K;++j)
L:          output_fm[to][row][col]+=
            weights[to][ti][i][j]*
            input_fm[ti][S*row+i][S*col+j];
//END
```

this basic building block. Generic DSP, Memory chips, processors logic units etc. Most of the FPGA uses synchronous clocking. And to have a minimal skew global lines provide the clocking to each block. VHDL and Verilog are popular to reduce complexity of designing HDL. Labview is also an important tool for graphical programming language.

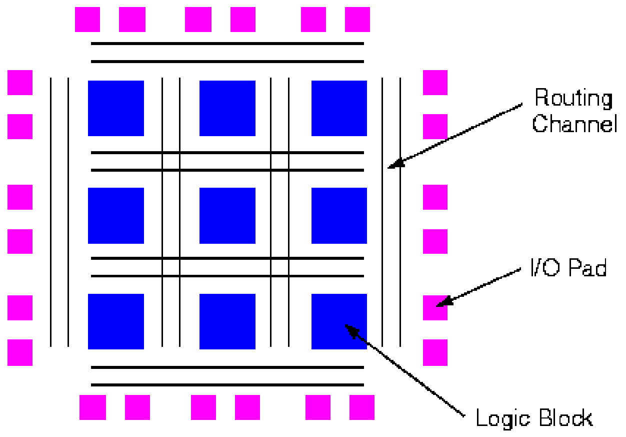


Figure 5: FPGA_structure

Following are the reasons why FPGA are being used for Deep Learning:

1) User Programmable
2) Dedicated logic for all unit
3) Easily reconfigurable
4) Ideal solution for dataflow architecture
5) Recent development in technology
6) Cost effective and fast turnaround time
7) Adaptability to new innovations in architecture and design
8) Independency in logic blocks

But there still exists circuit delays. This [9] paper discusses the usage of Altera Corp FPGA for dataflow computation with intelligent memory. This discuss about how FPGA can be used as a good substitution for memory where in hardware changes frequently. Adaptability of programmable device makes task simple. Idea of data flow memory has been discussed along with the intermediate buffer Instruction queue (IQ) and the execution unit processor pool (PP). To develop they have used Altera hardware description language (AHDL).

Survey by Jihan Zhu and Peter Sutton [7] discusses the challenges faced in implementing first ANN on FPGA. FPGA devices permit the execution of ANNs with modifiable topologies. But implementing floating point number for weights was very challenging. Which is a problem as discussed by Nichols et al. [15]. In addition, Bit-Stream arithmetic helped in reducing synoptic multiplication to simple logic operation. Weight precision and Transfer function implementation were the main issues.

## 2.3. Data Flow History and Development

Data Flow architecture is very easy for parallel computing and simple. As there is no stalling caused by branch prediction misses. As and when data is available execution is started by an actor. Flow of data and this architecture was developed in 1970 [16], [19]. This has exploited parallel computing using graphs. These graphs contain dependence of data and let actors know what data to fire. So there is a decision actor along with control actors. Then after different architectures were proposed e.g. Kahn process networks, Dennis dataflow etc. Kahn was used by concurrence theorist and Dennis was applied to architecture design.

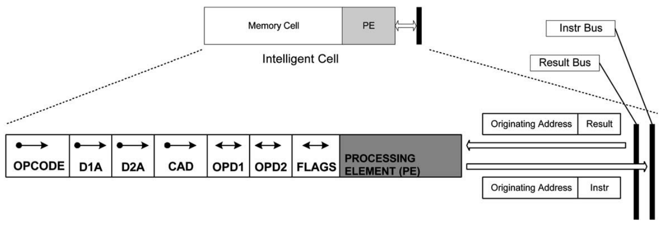


Figure 6: DataFlowMemory

So there are two main types of architecture:

- Static
- Dynamic

Static can have only single result value i.e. one instance of actor at a time in execution. Dynamic type exposes more data parallelism by carrying multiple tokens which are associated with different actors. Inherent nature of parallelism solves two problems of Neumann computer mainly memory latency and synchronization overhead [17]. This is also called as latency tolerant architecture. Whole new fields have evolved inspired on dataflow methodology. Signal processing, multithreading even the programming language are inspired on the same [18]. Later on region of actors are grouped together to create a hybrid model. This combines dataflow and control flow which helps in exploitation of multithreading. McGill data flow architecture model is based on argument-fetching principle [16], [19]. Instruction Set Architecture (ISA) architecture defines program execution model which serves and interface between hardware architecture and software. ISA includes set of instruction which operates on registers to perform certain actions defined by program.

Development in FPGA and VLSI in last couple of decades has renewed attention to reconfigurable computing machines (RCM). But there is a tradeoff between generality and efficiency. Operation density of von Neumann based program model is much higher than general purpose processors. RCM consists of array of programmable blocks and a processor. And can be distinguished based on granularity of these programmable blocks as Fine grained and Coarse grained.

## 3. neuFlow and luaFlow [10]

These are specially focused because of proliferation of research in machine vision and applications in the area of AI. This was designed to make predictions in real time. With only 10 Watts of power consumption there is a significant speedup in object detection and categorization in complex image. Like CNN Scale-Invariant Feature Transform and Histograms of gradients requires special GPUs. neuFlow and luaFlow are designed mainly for convolutional neural network consist of many layers of computation. Image processing algorithms usually have sequences, trees or graphs of transformations. Modular approach is best to implement recognition task. Figure 4 on the preceding page demonstrates the same. So any layer can start processing only when previous layer's all components are finished. This nature of model prevents parallel computation however it can be achieved within module.

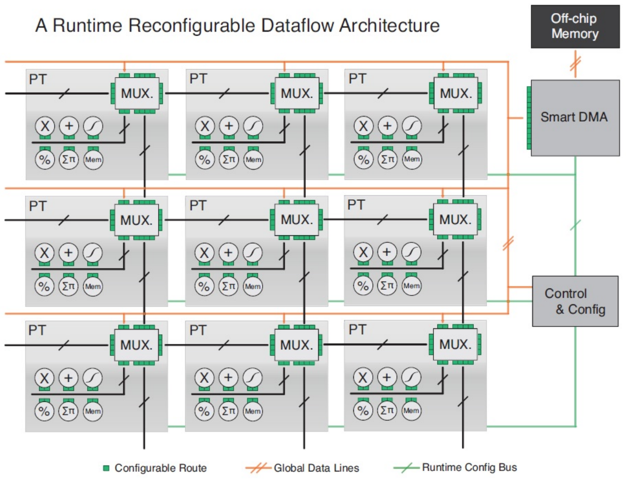


Figure 7: Reconfigurable Data Flow

FPGAs offer most versatile grid processing elements this one of the important aspect of the grid shown in figure 7. This advantage is with tradeoff of synthesis time and reconfiguration time. As shown in figure 4 on the preceding page last part of classification is done on multicore processors. To work program on these multicore no explicate synthesis is required. Existing programming language can express the parallelism. Cl´ement Farabet et al. [10] proposes a structure of grid which provides a flexible processing framework. And discusses how this architecture is useful in avoiding the constraints faced. Not only the module but connections also have configurable parameter routes or setting. These grids can not only perform the simple unary operation but also performs complex nested operations.

Figure 7 on the previous page presents several important aspects:

- Processing Tiles (PTs): It is a bank of processing operators. It can be arithmetic or FIFO or combination of operators. Operators are connected to local bus and eventually to MUX. Mux connects this data lines to local as well as global bus.
- Smart DMA connects to off chip memory. This provides asynchronous data transfer with the implementation of the priority management.
- Idea is to interconnect all the modules. So PTs are connected to neighboring 4 units.
- Connections to memory DMA, operators are being reconfigurable maintained by runtime configuration bus.

## 3.1. Runtime Reconfiguration

As mentioned earlier main feature of the grid is its configurability. And there are many similar grid structures proposed having re-configurability in interconnection. FPGAs are most important among them as they can pack elements of order $10^4$ to $10^5$. And main drawback is reconfiguration time. So fully dataflow architecture creation is a difficult task. And following constraints can be defined:

- Throughput is the main concern over low latency. This is because many operations which are being performed on the image are repeated on different segments. And latencies occurring pipelining processing are not of much significance.
- Hence each of these operators must provide a high throughput. This can be achieved by executing one operation per clock cycle. In addition the operations need to be stallable.
- It is necessary to have low configuration time and as an order of latency. So it has to be as negligible as possible.
- Grid must have processing elements as coarse as possible. This will ensure maximum ratio of computing and routing logic. This is important because when you make architecture application specific some of the hardware elements need to be developed for application unique operations. And this should in handshake with other general purpose elements.

Several research have been done to make sure of first two points above and to solve the latency vs throughput tradeoff. The proposed architecture here is stallable and has more advantages. So each operator can stall pipeline instead of breaking. FIFO implementation is used to monitor bubbles and stalling. Without being skeptical about the bandwidth issue sequence of operators can be created. Runtime configuration of bus makes third point achievable. As shown in the figure 7 on the preceding page each unit has parameters which can be tuned. Operation of the system can be illustrated as below:

1) Control unit is responsible for making connections local to neighbors and on global to others. Run time configuration bus os used to send the commands in doing so.
2) It then configures smart DMA to fetch the data and after processing sends back the data to off-chip memory.
3) Out streaming start as and when DMA is ready to do so.
4) Each tile then starts processing. It then forwards the data to another tie if required or sends back it to smart DMA.
5) Once the operation by DMA is complete control units notifies it.

Independently monitoring the flow of data by FIFO, operators can cascade data.

## 3.2. Optimization for FPGAs

Recent advances in processors make prototyping fast and easy to simulate real life applications. In this section it has been discussed that how filter based algorithms are configured on FPGA. As done for many applications here also HDL is used for programming and for synthesis. Paper talks about implementation of 2D convolutional network. As shown in the figure 8 on the next page following specializations are achieved:

- Top row of the diagram implements multiplication and addition operation. It can also perform dot product (Kernel trick) and spatial pooling.
- General mathematical operations like division, squaring etc. are performed in the second row
- Bottom layer implements the mathematical functions like trigonometric functions, absolute and normalization.

2D convolver itself is analogues to dataflow grid. Integration of convolver within the tile is only to make the ratio computing by routing logic better. Doing this there is tradeoff but then for CNN application it is beneficial. Kernel and the images are loaded from the memory. When applying a $K \times K$ matrix of filter convolver triggers same dimension of parallel operation.

DMA is a critical part of the architecture. It is proposed to access the external memory by asynchronous port. A dedicated arbiter is used. Because of the implementation of control unit it is called as smart. Using an optimal stride read and writes operations are performed. Relation between

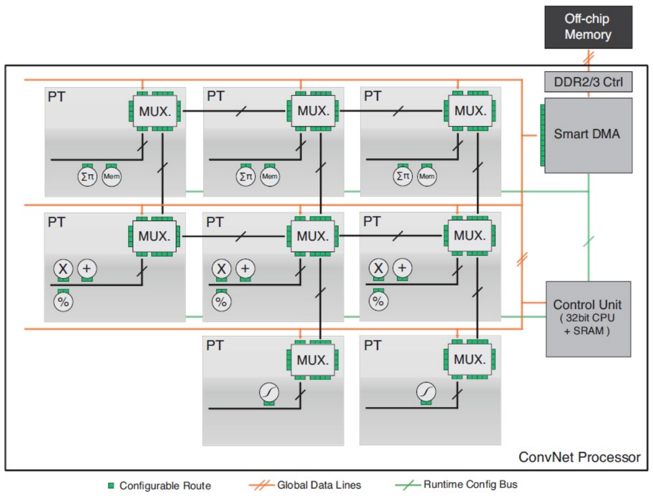


Figure 8: Filter Application of FPGA

memory bandwidth and parallel data transfer with bits is given as:

$$Max\,(N_{DMA}) = \frac{B_{EXT}}{P_{bits}}$$

e.g.

$$P_{bits} = 16 \ and \ B_{EXT} = 128 bits/cycle$$

$$\implies Max\,(N_{DMA}) \approx 7 \, parallel \, transfer$$

### 3.3. LuaFlow: Compiler for neuFlow

So the discussion ends at how to convert the code to control unit understandable. API luaFlow is a compiler and a dataflow has been the key to this problem. Developed in Torch5 environment it is a compiler parse code to different levels of parallelism. And the main question in designing an algorithm is what sequence of grid computation is chosen to have minimum computation time for given grid? So to parallelize the computation we can have three levels:

- Across Modules
- Across Images
- Within Image

Example of parallelization done by luaFlow is as follows:

$$\#\, PTs\, 2D\, convolvers = 4$$

$$\#\, PTs\, standard\, operators = 4$$

$$\#\, PTs\, non\, linear\, mapper = 4$$

Objective is to create fully connected filter bank

$$y_i = \sum_{i=0}^{3} k_{ij} * x_i \quad for \ \ j \in [0, 7]$$

where 8 outputs is a sum of 4 inputs.
And each convolved with a different kernel.

### 3.4. Remarks

Recent DSP oriented FPGAs have many MAC units and thousands of programmable blocks. This allows real time simulation of circuits. Figure 7 on page 5 can be modified using HDL to target ASIC and FPGAs. PTs are independent processing tiles consists of routing mux and local operators. This is mainly for 2D heavy data. So the architecture consists of 2D convolver that is a data flow grid itself. It maximizes ratio between computing logic and routing logic. Not only the local usage of caches but this approach also implements smart Direct Memory Access module. Smart because it complements control unit. A dedicated arbiter is used as memory interface to multiplex and demultiplex external memory. In Torch5 environment luaFlow is a compiler which parses inputs to different level of parallelism. Main area research is how to schedule dataflow computation and how to represent stream and computation on stream. There are many applications implemented on neuFlow and can be found at www.neuflow.org.

Paper has represented the filter-bank based data flow architecture approach for optimization. Performing a convolutional neural network on this type of architecture is easy. Since these are easy to configure they have wide range of applicability. And it is expected to have new research and inventions done in the area. Cl´ement Farabet et. al. [10] have developed a run time street view parser. They have used neuFlow to train the CNN.

## 4. FPGA-based Accelerator Design [2], [8]

As we have discussed earlier fulfilling memory bandwidth has been a challenging task of all time. Underutilization of it and logical resources does not match FPGA requirement. Gap of data transmission from CPU to GPU is still has scope to research and improvement. Chen Zhang et. al. have proposed a roofline model which supports analytical design scheme. Various optimization techniques are used such as loop tiling and transformation to measure computing throughput and required memory bandwidth. This roofline model helps identify solution with best performance and lowest resource requirement. The have implemented CNN accelerator on VC707 FPGA board. Performance achieved was 61.62 GFLOPS under 100MHz working frequency.

There have been many proposals to improve CNN performance like accelerated based FPGA, GPU and ASIC. Because of good performance, high energy efficient, speedup in development round and capability of reconfiguration FGPA has gained more attraction. And hence there are a lot potential solutions that result in a large design space for research. Finding these solutions are sometimes not trivial. If the design is not done carefully then the throughput does

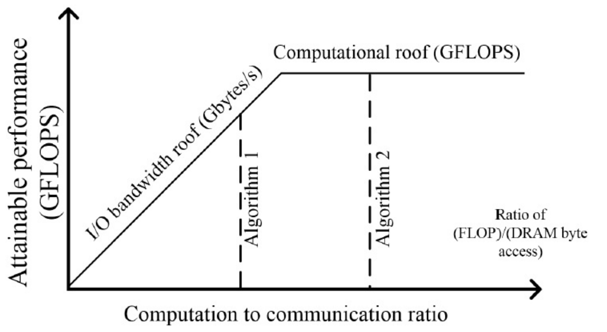


Figure 9: Roofline Model

not match FPGA memory bandwidth. So efficiency degrades because of underutilization of resources. Technology advances in FPGA and CNN algorithms improving together to handshake and aggravate the problem. As loop tiling is used design space increases and further improvement in algorithm demands more space. Hence an urgent solution was required to address the problem of small FPGA based CNN design space.

Work done by Chen Zhang et. al. [2] outperforms previous approaches because of:

1) Computation engine optimization.
2) Accelerates previous CNN applications by reducing external data access with delicate data reuse.

In both cases some modifications have been done. Main contribution of their work is as follows:

- Quantitatively analyze computing throughput and required memory bandwidth
- Given constraints identify all possible solutions in the design space using a roofline model and how to do that
- Propose loop unroll factors for CNN accelerator design
- Experiment resulted in highest performance result

With reference to the CNN code 90% of the time is spent in convolutions of the data. Making this operation fast is very critical in achieving time efficiency.

### 4.1. Roofline model:

Main constraints in the throughput optimization are computation and communication. So there is constraint because of memory or computation time. In analysis, roofline model gives arithmetic intensity vs GFLOP/s curve. It is very easy to compare two algorithms given a roofline model.

$$Arithmatic\,Intensity = \frac{Floating\,point\,operation}{Byte\,of\,memory\,access}$$

Using this formula throughput of the algorithm on given hardware can be formulated. GFLOPs are Giga Floating point operations per second. Computation to communication ratio (CTC) gives traffic for DRAM data traffic. As shown in figure 9 computation and I/O bandwidth roof is given and algorithm 2 is better than algorithm 1. It means that utilization of the hardware resources is better for algorithm 2.

### 4.2. Accelerator design details [2]

We will looks at the overview of accelerator structure and explore design challenges. Paper [2] highlights the components of the CNN accelerator based chip, where in main modules are processing elements (PE), external memory and buffer along with chip interconnections. A PE is the basic computation unit for convolution. Data needs processing are stored on externa memory. So to process the data those are first fetched in on-chip buffer and then forwarded to PE. So the challenges for this CNN accelerator to implementing FPGA are:

1) For small data loop tiling is required.
2) To process on chip data efficiently, interconnections need to be carefully considered.
3) PE processing throughput should match with bandwidth of the FPGA platform.

### 4.3. Optimization in Computation

Implementation of CNN is also done using loop scheduling and loop tile size enumeration. Once the code is scheduled loop unrolling is an important step to utilize FPGA hardware and maximize the resources availability. We will see loop unrolling, loop pipelining and tile size selection.

- Loop Unrolling: Massive amount of resource provided by FPGA needs to be utilized and to make sure that it is being achieved, loop unrolling plays an important role. How many instructions are exactly affecting the execution time and hardware in turn, whether they are affecting to what extent. And so the loop dimension can be classified as irrelevant, independent and dependent.
- Loop Pipelining: Overlapping the execution of code from different loops is idea behind loop pipelining. Resource constraints and data dependencies and data dependency would drive the implementation. Loop carried dependency prevents from having pipelined execution.
- Tile size selection: Performance depends on tile size, structure. It also depends on combination of specific sized tile.

### 4.4. Memory access optimization

So far it has been explained that data which needs to be computed needs to be buffered on to the chip. And we know the fact that even if the computation poser is high it is being harmed by the low memory bandwidth. So, code can be transformed so as to give the manipulated data as quick as possible to main memory. And so couple of terminologies we need to focus on

- Local memory promotions: We need to check if loops being operated are irrelevant or not. Especially the innermost loop iterating for many times. Depending on the measuring factor memory access to outer loops can be given.
- Loop transformation for Data Reuse: All allowed loop transformations are identified by polyhedral-based optimization framework.
- CTC Ratio: Computation operation per memory access is described by CTC i.e. Computation to communication. Data reuse increases the CTC because number of memory accesses gets reduced.

$$Computation to\, Communication\, Ratio =$$

$$\frac{total\, number\, of\, operations}{total\, amount\, of\, external\, data\, access}$$

### 4.5. CNN accelerator design

So far whet we have seen is for one layer and within specific task. But for CNN, parameters need to be considered for each layer. And it is very challenging to understand designing task to work for many convolutional layer. One of the approaches is to design complex hardware with newly configured interconnections. Uniformly unrolling factors across different convolutional layers is another way. So the system used consists of DDR3 memory for external storage. MicroBlaze, a RISC processor developed for Xilinx FPGAs along with other CPU for startup acceleration and communication. AXI4lite bus is for command transfer and AXI4 bus is for data transfer. MicroBlaze sends commands and configuration parameters and communicated to FPGA via AXI4 bus.

Performance comparison to that of CPU is as displayed in Table 2.

| float | CPU 2.2 Ghz | | FPGA | |
|---|---|---|---|---|
| 32 bit | 1thd | 16thd | ms | GFLOPS |
| layer 1 | 98.18 | 19.36 | 7.67 | 27.50 |
| layer 2 | 94.66 | 27.00 | 5.35 | 83.79 |
| layer 3 | 77.38 | 24.30 | 3.79 | 78.81 |
| layer 4 | 65.58 | 18.64 | 2.88 | 77.94 |
| layer 5 | 40.70 | 14.18 | 1.93 | 77.61 |
| Total | 376.50 | 103.48 | 21.61 | - |
| Overall GFLOPs | 3.54 | 12.87 | 61.62 | |
| Speedup | 1.00x | 3.64x | 17.42x | |

TABLE 2: Performance comparison with CPU

## 5. Remarks

Segreen Ingersoll et. al. have presented a new hardware for data flow architecture. They have proposed the intelligent memory kept aside making separation between processing and data flow operation. Their design may get more attraction in the future considering the rate at which FPGA technology is being boosted.

Najjar et. al. have described the long history behind the data flow architecture progression. Data Flow architecture has made a tremendous impact on computing machines. Their rapid evolution has large contribution to silicon industry in making high performing processors still keeping the simplicity intact. And this increases the expectation in new era; as real life applications are evolving demand for high performance and throughput is increasing. Data flow is not only been successful so far but also evolving so as to satisfy current requirement.

Zhu et. al. have surveyed how important is FPGA for artificial intelligence applications. Team has highlighted the importance of re-configurability nature of FPGA.

Lacey et. al. Presents past present and future for FPGA. When looked at the result FPGA outperform CPU and GPU and highlights that it is better in terms of power consumption.

Che et. al. developed and compared three applications on FPGA and GPU and compares pros and cons. In doing so they have considered many factors such as performance, hardware features, trade-offs etc.

### 5.1. Why the topic of Artificial Intelligence and FPGA architecture?

Machine performing some tasks like humans do with ease have always been intriguing. In today's rapidly changing world in terms of technology, industries are adopting many new methods for solving business problems. And in this competitive environment focus is on high performance, re-configurability and fast development round. Machine intelligence technology like CNN, Hierarchical Temporal Memory (HTM) which is a neocortex computational theories are emerging and hardware flexibility is a need. FPGA promises to give these essentials for pattern recognition tasks and can be a candidate. There is trade off and even though GPUs are better, FPGAs are getting attention because of the flexibility. Since current focus is on hardware improvement to be in sync with new analytical methods this area is open to research.

### 5.2. Current challenges and future work

There are many problems which are open for research. When it comes to machine learning example is Vanishing gradient problem. And there is a wide room for architecture improvement. Goal is to make time and space complexity to constant instead of some variable of number of inputs. But this may not be possible because allocating significantly more hardware may not be practical. There are various accelerators based on FPGAs have been proposed for deep CNN but the design space has not be explored [8]. Utilizing logical units in an efficient way, matching computational throughput and memory bandwidth are some of the main challenges of implementing deep learning complex algorithms.

Segreen Ingersoll et. al. have not implemented higher primitives. Hence indirect memory access are drawback for now as those cannot be done. But with some modifications in design problem can be ameliorated.

As mentioned by Zhu et.al. future work will also demand good softwares to be developed to utilize FPGA efficiently.

Lacey et. al. proposes that GPU clusters may suggest further architecture modifications to FPGA. [21]

Che et.al. proposes to have research in DSPs and network processors in order to have more insight on accelerators.

## 6. Conclusion

The report talks about the current research area in artificial intelligence and computer vision. With need of processing on much bigger data and diving deep to extract new feature algorithm demands better hardware architecture. Report cites some of the current research being done on the same and highlights current challenges involved. Even though other processor outperforms it is feasible and reliable to implement dataflow architecture on FPGA as required for latest technology in some sense. But further research will make applications faster and improve hardware performance. Idea is to develop design for each layer in neural net and interconnect them. Future work will focus on hardware-software dependence and how they affect each other leading to further benefiting performance.

## References

[1] Kalin Ovtcharov, Olatunji Ruwase, Joo-Young Kim, Jeremy Fowers, Karin Strauss, Eric S. Chung *"Accelerating Deep Convolutional Neural Networks Using Specialized Hardware"* Microsoft Research 2/22/2015

[2] Chen Zhang Peng Li Guangyu Sun Yijin Guan Bingjun Xiao Jason Cong *"Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks"*

[3] Bas Klein Essink *"Using FPGAs as Fine-Grained Static Dataflow Machines"*

[4] Lippmann, R.P., 1987. *"An introduction to computing with neural nets."* IEEE Accost. Speech Signal Process. Mag., April: 4-22.

[5] Microsoft Research: catapult, 2015.

[6] C. E. Cox and W. E. Blanz, *GANGLION-a fast field-programmable gate array implementation of a connectionist classifier*, in IEEE Journal of Solid-State Circuits, vol. 27, no. 3, pp. 288-299, Mar 1992.

[7] Zhu, Jihan, and Peter Sutton. *FPGA implementations of neural networks–a survey of a decade of progress.* International Conference on Field Programmable Logic and Applications. Springer Berlin Heidelberg, 2003.

[8] Che, Shuai, et al. *Accelerating compute-intensive applications with GPUs and FPGAs.* Application Specific Processors, 2008. SASP 2008. Symposium on. IEEE, 2008.

[9] Segreen Ingersoll, Sotirios G. Ziavras, *"Dataflow computation with intelligent memories emulated on field-programmable gate arrays (FPGAs)"*, Microprocessors and Microsystems 26 (2002) 263–280

[10] Clément Farabet, Berin Martini, Benoit Corda, Polina Akselrod, Eugenio Culurciello, Yann LeCun *"NeuFlow:A Runtime Reconfigurable Dataflow Processor for Vision"*

[11] O'Shea, Keiron, and Ryan Nash. *An Introduction to Convolutional Neural Networks.* arXiv preprint arXiv:1511.08458 (2015).

[12] Graves, Alex; and Schmidhuber, Jürgen; Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks, in Bengio, Yoshua; Schuurmans, Dale; Lafferty, John; Williams, Chris K. I.; and Culotta, Aron (eds.), Advances in Neural Information Processing Systems 22 (NIPS'22), 7–10 December 2009, Vancouver, BC, Neural Information Processing Systems (NIPS) Foundation, 2009, pp. 545–552.

[13] Markoff, John (June 25, 2012). How Many Computers to Identify a Cat? 16,000. New York Times. Retrieved February 11, 2014.

[14] Metz, Cade (2014-06-02). "The Mission to Bring Google's AI to the Rest of the World". Wired.com. Retrieved 2014-06-28.

[15] Nichols, K., M. Moussa, and S. Areibi. Feasibility of Floating-Point Arithmetic in FPGA based Artificial Neural Networks. in Proceedings of the 15th International Conference on Computer Applications in Industry and Engineering. 2002. pp San Diego, California.

[16] J.B. Dennis, First version of a dataflow procedure language, in: Proceedings of the Colloque sur la Programmation, Lecture Notes in Computer Science, vol. 19, Paris, France, April 9-11, Springer, Berlin, 1974, pp. 362-376.

[17] Arvind, R. Iannucci, A critique of multiprocessing von Neumann style, in: International Symposium on Computer Architecture, Stokholm, Sweden, 1983.

[18] Najjar, Walid A., Edward A. Lee, and Guang R. Gao. Advances in the dataflow computational model. Parallel Computing 25.13 (1999): 1907-1929.

[19] J.B. Dennis, G.R. Gao, An efficient pipelined dataflow processor architecture, in: Proceedings of the Supercomputing '88, Orlando, FL, November 1988, pp. 368-373.

[20] D. Aysegul, J. Jonghoon, G. Vinayak, K. Bharadwaj, C. Alfredo, M. Berin, and C. Eugenio. Accelerating deep neural networks on mobile processor with embedded programmable logic. In NIPS 2013. IEEE, 2013.

[21] Lacey, Griffin, Graham W. Taylor, and Shawki Areibi. Deep Learning on FPGAs: Past, Present, and Future. arXiv preprint arXiv:1602.04283 (2016).