TileNET: SCALABLE ARCHITECTURE FOR HIGH-THROUGHPUT TERNARY CONVOLUTIONAL NEURAL NETWORKS USING FPGAs

A DISSERTATION REPORT

Submitted by

BL.EN.P2VLD15024 SAHU SAI VIKRAM

In partial fulfillment of the award of the degree of

MASTER OF TECHNOLOGY

IN

VLSI DESIGN

\

DEARTMENT OF ELECTRONICS AND COMMUNICATION

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDYAPEETHAM

BANGALORE 560 035

JULY 2017

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF ENGINEERING, BANGALORE, 560035

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"TileNET: SCALABLE ARCHITECTURE FOR HIGH-THROUGHPUT TERNARY CONVOLUTIONAL NEURAL NETWORKS USING FPGAs"** submitted by "**SAHU SAI VIKRAM (BL.EN.P2VLD15024)**" in partial fulfillment of the requirements for the award of the Degree **Master of Technology** in **"VLSI Desgin"** is a bonafide record of the work carried out under our guidance and supervision at **Amrita School of Engineering, Bangalore**.

Dr. Madhura P(Project Supervisor)	**Dr. N. S. Murty**
Associate Professor	Chairperson
Department of CSE	Department of ECE
Amirta School of Engineering	Amrita School of Enginerring
Bangalore	Bangalore

This project report was evaluated by us on __7/17.

EXAMINER 1 EXAMINER 2

ACKNOWLEDGEMENT

I owe my deepest gratitude to my Guide, Dr. Madhura. P, Associate Professor, Amrita School of Engineering, Bangalore whose encouragement, supervision and support throughout the course of my project work enabled me to develop an understanding on the subject. This project work wouldn't have been possible without her. I offer my regards to all those who supported me during the completion of my project, especially the teaching and administrative staff.

I would like to express my sincere gratitude towards Dr. N. S. Murty, Head of Electronics and Communication Department, for his kind support throughout the project.

I would like to express my sincere gratitude towards our Associate Dean, Dr. Rakesh S. G ., for giving me this opportunity to manifest my ideas into a real-time project.

I would like to thank Mr. Mihir Moody, Texas Instruments, Bangalore for his immense support at every step during this project, and guiding me through various difficulties while executing this project.

ABSTRACT

Convolution Neural Networks (CNNs) have become very popular for advanced driver assistance systems (ADAS) and autonomous driving for object detection and image recognition. The choice of CNNs for ADAS necessitates a high-throughput in the order of about few 10's of TeraMACs per second (TMACS). In addition to throughput, accuracy is also of very high importance. Existing implementations become unusable with performance ranging only in the order of a few Giga-ops. This paper presents a novel tiled architecture for CNNs with ternarized weights. In this context, it has been observed that ternarization of weight vectors results in a minimal loss of accuracy, while reducing the memory utilization and eliminating multipliers in the inference phase. Our tiled architecture is generic and scalable across variations in network organization and device configurations. Hence, our objective is to trade-off accuracy to achieve high performance for CNNs to be applied in ADAS scenario. Our implementation results in 13.76 TOPS throughput for TileNet implementation of AlexNet on Virtex-7 FPGA device. The results from our estimation model show that the performance can be tuned across various application needs by catering to different CNN architectures and by providing flexibility across devices. The post implementation power results for AlexNet are around 16 W for the largest layer and are much lower than GPU power consumption.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

INTRODUCTION

1.1 ARTIFICIAL INTELLIGENCE

Artificial intelligence is a branch of computer science that aims to create intelligent machines. It has become an essential part of the technology industry.

Research associated with artificial intelligence is highly technical and specialized. The core problems of artificial intelligence include programming computers for certain traits such as:

- Knowledge
- Reasoning
- Problem solving
- Perception
- Learning
- Planning
- Ability to manipulate and move objects

Knowledge engineering is a core part of AI research. Machines can often act and react like humans only if they have abundant information relating to the world. Artificial intelligence must have access to objects, categories, properties and relations between all of them to implement knowledge engineering. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious approach.

1.2 MACHINE LEARNING

Machine learning is another core part of AI. Learning without any kind of supervision requires an ability to identify patterns in streams of inputs, whereas learning with adequate supervision involves classification and numerical regressions. Classification determines the category an object belongs to and regression deals with obtaining a set of numerical input or

output examples, thereby discovering functions enabling the generation of suitable outputs from respective inputs. Mathematical analysis of machine learning algorithms and their performance is a well-defined branch of theoretical computer science often referred to as computational learning theory.

Machine perception deals with the capability to use sensory inputs to deduce the different aspects of the world, while computer vision is the power to analyze visual inputs with a few sub-problems such as facial, object and gesture recognition.

Robotics is also a major field related to AI. Robots require intelligence to handle tasks such as object manipulation and navigation, along with sub-problems of localization, motion planning and mapping.

1.3 NEURAL NETWORKS

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions. Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

1.3.1 A simple neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not(Figure 1).

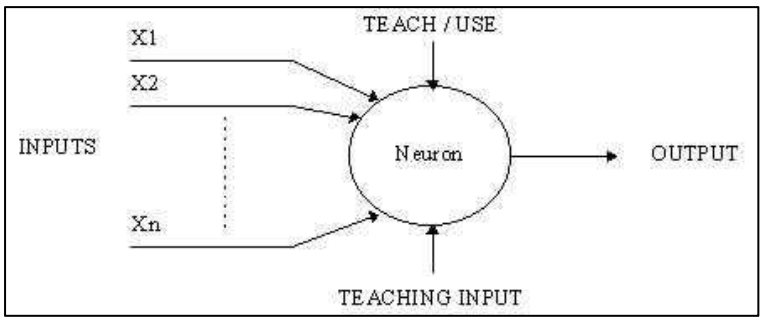


Figure 1: A simple neuron

1.4 ARCHITECTURE OF NEURAL NETWORKS

1.4.1 Feed-forward networks

Feed-forward ANNs (Figure 2) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

1.4.2 Feedback networks

Feedback networks (Figure 3) can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

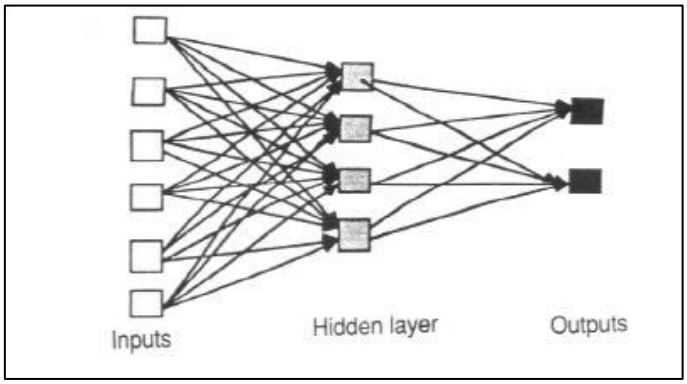


Figure 2: An example of a simple feedforward network

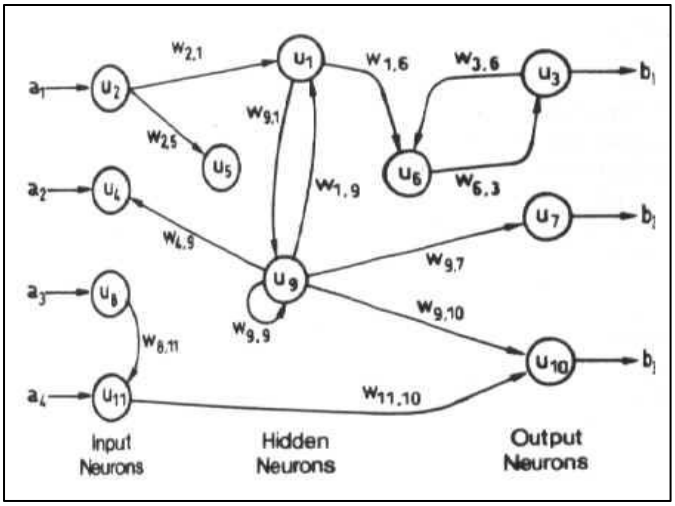


Figure 3: An example of a complicated network

1.5 NETWORK LAYERS

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units.

- The activity of the input units represents the raw information that is fed into the network.
- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power

than hierarchically structured multi-layer organisations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

CONVOLUTIONAL NEURAL NETWORKS

2.1 CONVOLUTIONAL NEURAL NETs

Convolutional Neural Networks (**ConvNets** or **CNNs**) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

Figure 4: Identifying the scenes

In Figure 4 above, a ConvNet is able to recognize scenes and the system is able to suggest relevant captions ("a soccer player is kicking a soccer ball") while Figure 5 shows an example of ConvNets being used for recognizing everyday objects, humans and animals. Lately, ConvNets have been effective in several Natural Language Processing tasks (such as sentence classification) as well.

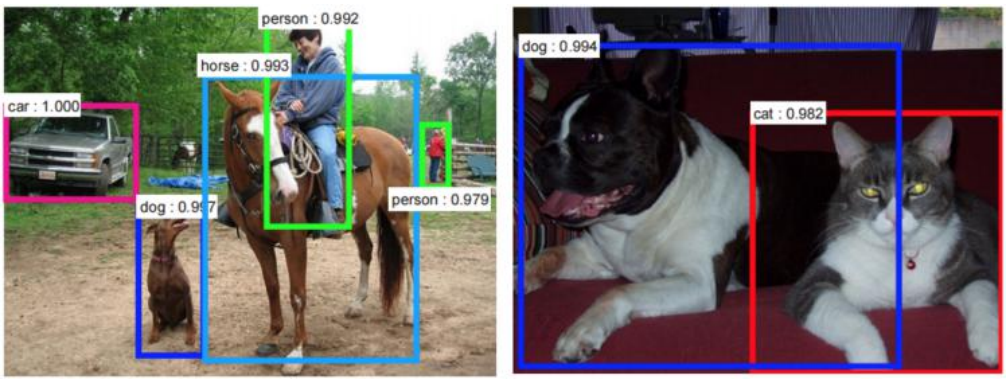


Figure 5: Recognizing everyday objects

ConvNets, therefore, are an important tool for most machine learning practitioners today.

2.2 STRUCTURE OF CNN

Fig.6 shows the basic structure of CNN. There are three kinds of layer: input layer, output layer, and hidden layer. CNN is a multi-layered neural network with multiple hidden layers, while the original artificial neural network has only a single hidden layer.

Input layer is the input interface for training and test data. An input data is represented as a vector of integer or floating point values, and each value in the vector corresponds to a neuron. In image processing, the input layer forms a 2D array. In recognition problems, output layer indicates the probability that the input data belong each classification category. On CNN, there are multiple hidden layers between the input layer and output layer.

Hidden layers consist of convolution layers, sub-sampling layers (pooling and max-out layers), and full-connected layers. In convolution layer, the value of each point is obtained by a convolution of a corresponding sub-region in the previous layer and a weight vector. CNN exploits a feature map by adopting a convolution, as well as standard image processing exploits a feature map by using filters.

Pooling layer is a condensed vector of representative values in the previous layer, which is generated by sampling of a maximum value or an average value in every small region in the previous layer for ignoring parallel displacements of data in some degree. Max-out layer is also a condensed vector, but which is generated from the previous multiple layers. In full-connected layers, all points in a layer are connected to all points in the adjacent layer, in order to determine the values of the output layer by using the feature map information in the previous convolution and sub-sampling layers.

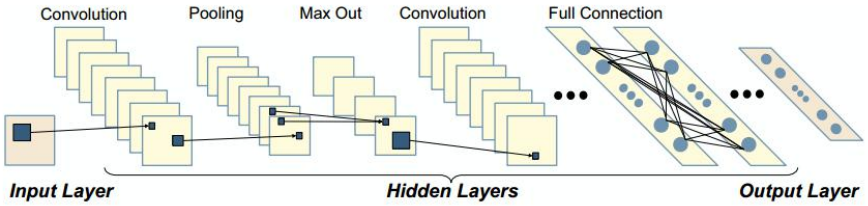


Figure 6: Structure of CNN([4])

2.3 ALEXNET

Alexnet is a Deep Convolutional Neural Network for image classification that won the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry AlexNet (designed by Krizhevsky *et al*. is one of the deep ConvNets designed to deal with complex scene classification task on Imagenet data. The task is to classify the given input into one of the 1000 classes.The main differences between LeNet and AlexNet are in Alexnet,

i) Number of processing layers and number of trainable parameters: AlexNet has 5 convolutional layers, 3 sub sampling layers, 3 fully connected layers and its total trainable parameters are in crores whereas LeNet has 2 convolutional layers, 2 Sub sampling layers and 3 fully connected layers and its total trainable parameters are in thousands.

ii) The non linearity used in the feature extractor module of the AlexNet is ReLU whereas LeNet has logistic sigmoid.

iii) AlexNet uses dropout where as no such concept is used in LeNet. A fully trained AlexNet on ImageNet data set can not only be used to classify Imagenet data set but it can also be used without the output layer to extract features from samples of any other data set.

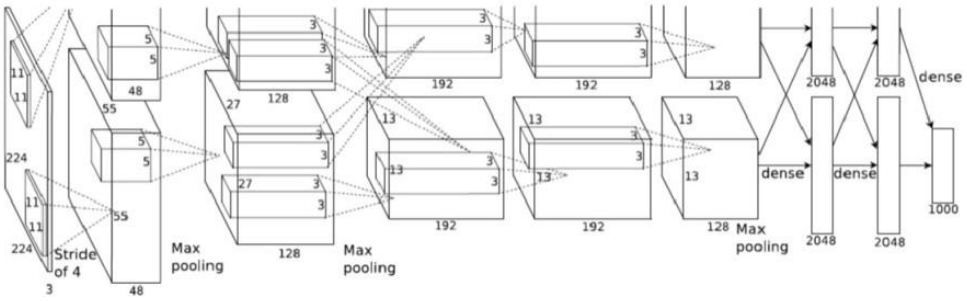


Figure 7: Structure of AlexNet([4])

Layer	N_{in}	N_{out}	$Size_{in}$	$Size_{out}$	$Size_{kernel}$	$Stride$
CONV1	3	96	227x227	55x55	11x11	4
POOL1	96	96	55x55	27x27	3x3	2
CONV2	48	256	27x27	27x27	5x5	1
POOL2	256	256	27x27	13x13	3x3	2
CONV3	256	384	13x13	13x13	3x3	1
CONV4	192	384	13x13	13x13	3x3	1
CONV5	192	256	13x13	13x13	3x3	1
POOL5	256	256	13x13	6x6	3x3	2
FC6	9216	4096	1x1	1x1	--	--
FC7	4096	4096	1x1	1x1	--	--
FC8	4096	1000	1x1	1x1	--	--

Table I: Configurations of Different Layers in AlexNet([4])

2.3.1 ReLU: Rectified Linear Unit

ReLU is a function first introduced by Hahnloser et al. It was then stated by Nair et al. that ReLU is an effective activation function for use in neural network as well. ReLU function is given by:

$$f(x) = \max(0,x)$$

LITERATURE SURVEY

To implement this CNN structure, some Reconfigurable Architectures are studied for this work. Reconfigurable architectures can bring unique capabilities to computational tasks. They offer the performance and energy efficiency of hardware with the flexibility of software.

1. DRRA(Dynamically Re-configurable Resource Array)[1][4]
2. EMAX(Energy -aware Multimode Accelerator)[3]

Reconfigurable architectures offer the parallel processing similar to ASICs, while retaining the programmability of software solutions. The architectures stated above are reconfigurable and each of these architectures they have a local memory for each of their processing Elements(PE). This feature reduces the memory transfers in the Architecture.

While the Reconfigurable architectures have the better flexibility and reconfigurability, but they are not best in terms of performance and power consumption.

Hence some application specific Hardware Accelerators are considered.

3. DianNao[2]

The accelerator researched for this work follows this spirit of targeting a specific, but broad, domain, i.e., machine learning tasks here. The focus of the accelerator is on large-scale machine learning tasks, with layers of thousands of neurons and millions of synapses, and for that reason, there is a special emphasis on interactions with memory. In DianNao[2], they operate on the small parts of data at a particular time using the process called tiling and reduce the memory requirements.

3.1 NeuroCGRA: A CGRA with support for neural networks:

The NeuroCGRA is a reconfigurable Architecture which is derived with few modifications to the DRRA Architecture. For the applications that require exact calculations, the device behaves like a normal CGRA, with

MACs/ALUs connected via circuit switched interconnect. When an application, that can tolerate approximate results, enters the platform the device dynamically

They have chosen the Dynamically Reconfigurable Resource Array (DRRA) as a vehicle to evaluate feasibility of implementing neural networks on an actual CGRA.

3.1.1 DRRA Architecture:

DRRA computational layer is shown in Fig. 8. It is composed of four elements:
(i) Register Files (reg-files),
(ii) morphable Data Path Units (DPUs),
(iii) circuit-Switched Boxes (SBs), and
(iv) sequencers.

The reg-files store data for DPUs. The DPUs are functional units responsible for performing computations. SBs provide interconnectivity between different components of DRRA. The sequencers hold the configware which corresponds to the configuration of the reg-files, DPUs, and SBs. Each sequencer can store up to 64 36-bit instructions and can reconfigure the elements only in its own cell. As shown in Fig. 8, a *cell* consists of a reg-file, a DPU, SBs, and a sequencer, all having the same row and column number as a given cell. The configware loaded in the sequencers contains a sequence of instructions (reg-file, DPU, and SB instructions) that implements the DRRA program.

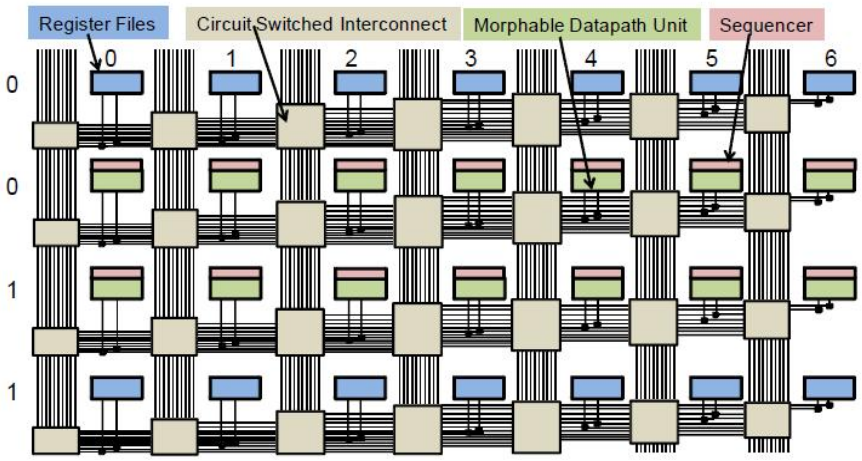


Figure 8: DRRA Physical Layer Fabric

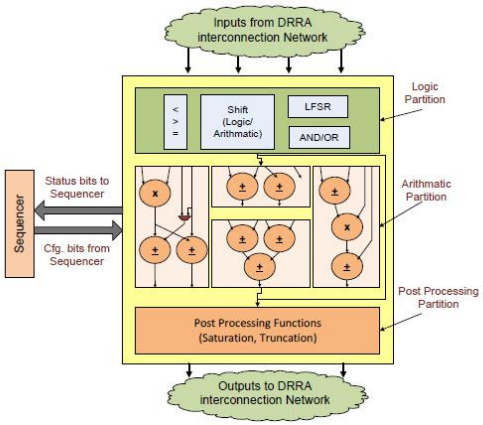


Figure 9: An illustration of a Data-Path Unit used in the DRRA fabric

.

3.1.2 Neural Networks/ DRRA Integration:

For this work they have chosen Spiking Neural Networks (SNN). Fig. 10 depicts a simple neuron model. In the figure, the neuron 4 receives input spikes from neurons 1, 2, and 3. Each connection is characterized by a weight (Wt1, Wt2, and Wt3). When a spike is generated, depending on the neuron model and weight of the connection, the neuron performs calculations. They have chosen a simple and the widely used model called leaky Integrate and Fire (I & F), to model a neuron. The model is given by the Equation 1:

$$dv/dt = (W t * I) + a - (b * V). \quad (1)$$

Where the potential V integrates input spikes I and leaks over time with $-bV$ component. Wt is the weight of the connection coefficient, a determines equilibrium point, and coefficient b is the speed of leakage. When the threshold potential is reached, a neuron outputs a spike and its potential is reset.

To realize neural networks on DRRA, they have embedded a dedicated hardware, called neuroDPU, with each DPU of DRRA. As a result of our enhancements, the DPU can be configured to either normal or neuron mode. Fig. 9 shows how the DPU functions in neuron mode.

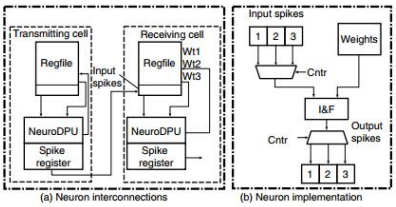


Fig. 5. Neuron realization on DRRA

Figure 10: Neuron realization on DRRA

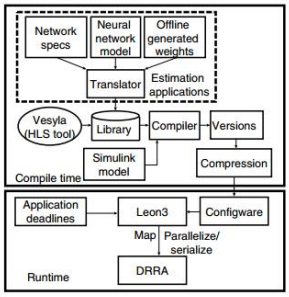


Figure 11: Modified Programming Flow

To realize FIST they have modified the configuration flow shown in the Fig.11. They have added another block to generate the configware for the estimation algorithms (shown by the dotted box). The key component of the new block is a translator. The translator takes three inputs: (i) application, (ii) weights, and (iii) application to generate the reg-file, SB, and DPU instructions for DRRA.

3.1.3 Overhead analysis:

To check the overhead imposed by the additional I & F unit, they synthesized it using 65*nm* technology with frequency of 500 MHz. Table II shows the obtained results. It can be seen that the proposed enhancement incurs 9.1% area and 4.2% power overheads.

TABLE II
AREA AND POWER CONSUMPTION OF I & F UNIT

	I & F	DRRA cell	Overhead (%)
Power mW	6.44	70.40	9.1
Area μm^2	50920	1199506	4.2

3.2 A CGRA-based Approach for Accelerating Convolutional Neural Networks

The goal of this work is to propose a high performance,low-power, and programmable approach that supports both training and classification of CNN. They presented EMAX, a CGRA with on-chip distributed memory for highly memory bandwidth efficiency by efficient temporal blocking.

3.2.1 EMAX(Energy Aware Multi-mode Accelerator)*:* Fig.12 shows the architecture of EMAX. EMAX employs an array structure of multiple processing elements (PEs) and an interconnection network. Each PE is connected by local interconnections between adjacent PEs and shared buses for each row. Computation results on the PEs are propagated to PEs on the next row. The PE array is connected with the external interconnection via a memory interface, so that the PEs access to an external memory (DRAM) and a CPU controls the EMAX.

Fig.13 shows the microarchitecture of a PE. A PE has two execution units (EX1 and EX2), an address generation unit (EAG), constant registers, two FIFOs, and a local memory (LMM). In the current implementation, the execution units are 32-bit, 2-stage floating point units. The FIFOs store temporal data employed by the next PEs. The LMM is a single-port local memory for each PE. Each PE can both read and write to the LMM by load and store instructions. Before the program is executed, instructions and constant values in the constant registers for each PE are assigned by the host CPU. Then, while the program is running, the instruction for each PE can be changed, but it can keep the contents on the LMM for temporal blocking.

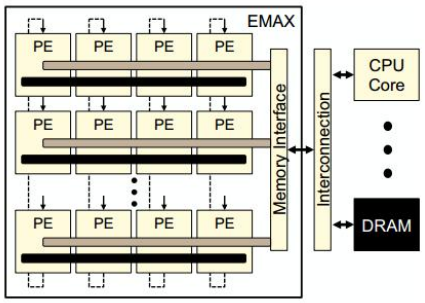


Figure 12: EMAX Architecture

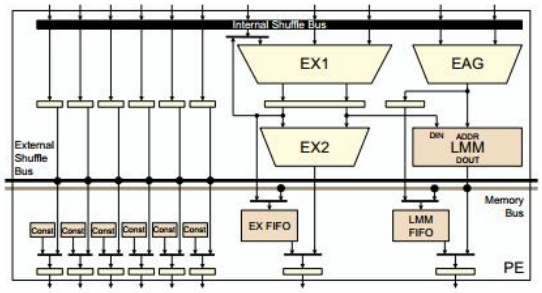


Figure 13: PE Micro-architecture

TABLE III. BENCHMARK SETUP

Dataset	Layer	Conv. Kernel	Output Neurons	Input Neurons	Input Imagesize	Mini-batch size	B/F value	Bound in GPU	Bound in EMAX
Imagenet	Alexnet-2	5x5	256	96	31	64	2.6E-03	Computation	Computation
CIFAR10	CIFAR10-1	5x5	32	3	36	64	6.0E-02	Memory	Computation
	CIFAR10-2	5x5	32	32	20	64	1.3E-02	Computation	Computation
	CIFAR10-3	5x5	64	32	12	64	2.1E-02	Computation	Computation
MNIST	Lenet-1	5x5	20	1	28	64	1.7E-01	Memory	Memory
	Lenet-2	5x5	50	20	12	64	1.6E-02	Computation	Computation

TABLE IV. HARDWARE SETUP

Hardware	GTX980	Core i7 5960x	GK20A	ARM (TegraK1)	EMAX
Number of Cores	2048	8	192	4	128 (4 rows* 32 column)
Frequency [MHz]	1253	3800	850	2300	200 (projeted.)
Peak comp. Performance[GFLOPS]	5132.288	972.8	326.4	73.6	51.2
Memory Bandwidth[GB/sec]	227	68	14.784	14.784	1.6
Number of transistors	5.2.E+09	2.1.E+09	-	-	2.6.E+07 (projected.)

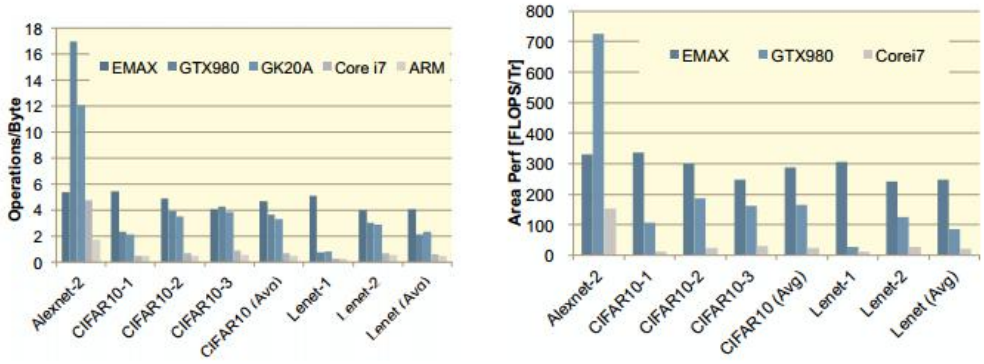


Fig 14: Performance of EMAX over CPU and GPUs

They used 3 models attached in caffe, ImageNet, CIFAR10, and MNIST (Lenet). The mini-batch size is 64. The CNN implementation on EMAX is aimed at improving efficiency in small embedded situations. They implemented 6 CNN models with 5× 5 weight matrix kernel. The parameters are listed in Table.III in detail. They evaluated the performance of CNN on EMAX in terms of normalized performance per external memory bandwidth and chip area. Although energy efficiency (performance per energy) should be evaluated, they estimated the performance per area instead of the energy efficiency, based on a simple assumption that the energy consumption is proportional to the area. They evaluated the approach by comparing some existing platforms, such as high-end and mobile GPUs, and general multicore

CPUs. The evaluation result shows that our proposal achieves 1.93x higher performance per memory bandwidth and 2.92x higher area performance, respectively.

3.3 DianNao:

In this study, they designed an accelerator for large-scale CNNs and DNNs, with a special emphasis on the impact of memory on accelerator design, performance and energy. The Accelerator achieved a high throughput of 452 GOP/s (key NN operations such as synaptic weight multiplications and neurons outputs additions) in a small footprint of 3.02 mm^2 and 485 mW; compared to a 128-bit 2GHz SIMD processor, the accelerator is 117.87x faster, and it can reduce the total energy by 21.08x. The accelerator characteristics are obtained after layout at 65nm

Tiling: To reduce the memory bandwidth, they use a process called “Tiling”. In this, instead of storing the entire data(i/p neurons or synapses), they have divided the data in the form of small Tiles. When the computation of one Tile is done. Other tile is loaded into the memory. This reduces the Bandwidth requirements and is useful as they require large memory bandwidth to store neuron values and synaptic weights.

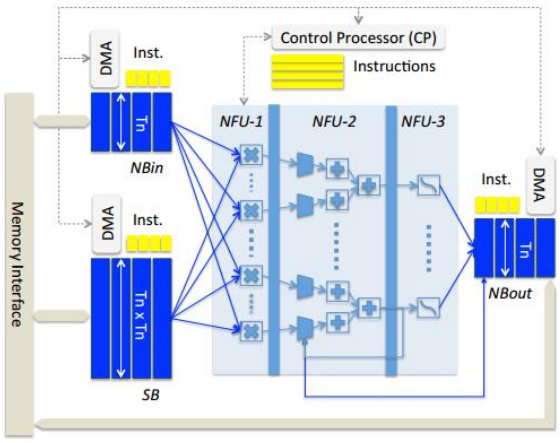


Figure 15: DianNao Accelerator

3.4 A high performance FPGA-based accelerator for large-scale convolutional neural networks

This Archetechure is designed based on the AlexNet.

AlexNet is proposed in the 2012 Image-Net LargeScale Vision Recognition Challenge (ILSVRC) with top-1 and top-5 classification error rates of 37.5% and 17% on ImageNet dataset. Fig.1 shows the AlexNet model which includes 5 CONV layers and 3 FC layers. Three max pooling layers follow the first, the second and the fifth CONV layers. ReLU function is applied on every layer except the last one. During the computation, it has been divided into two groups. The configurations of AlexNet are in Table I, where N_{in} and N_{out} are the numbers of input maps and output maps.$Size_{in}$ and $Size_{out}$ are the sizes of the input map and output map; $Size_{kernel}$ is the size for the convolution kernel. *stride* the shifting stride of the kernel during convolution operations

In the architecture, the 5 CONV layers and 3 FC layers run in the 8 modules concurrently in a pipelined style. Consequently, numerous input data and weights are required, which leads to a high data access workload. To solve this problem, intermediate results between layers are stored in on-chip buffers. In this way, the data access workload can be reduced significantly. Weights are stored in the external memory because the number of weights is too large. Furthermore, on-chip buffers also facilitate data reuse. There are 2 ping-pong buffers for each stage, where the former layer may write to or read from one buffer while the next layer reads data from the other buffer. Dual-port memory blocks are used as the ping-pong buffers. Hence, ping-pong buffers can avoid the overlap and improve the performance.

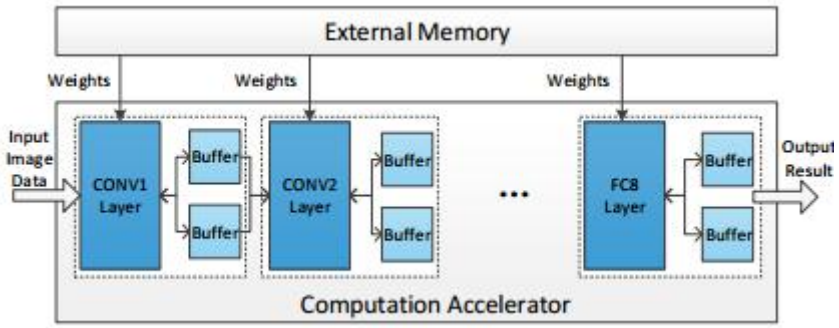


Fig 16: Overview of the Architecture

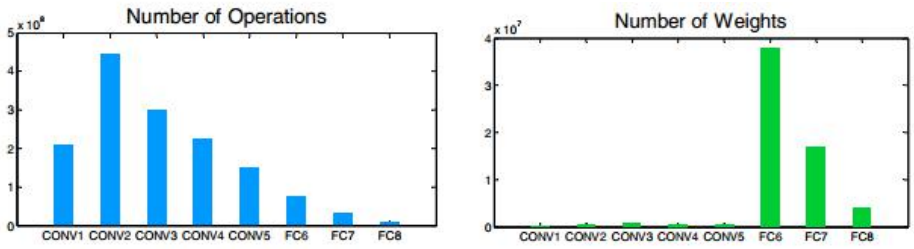
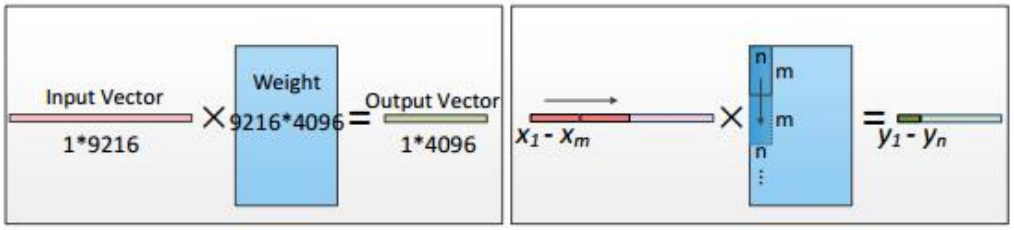


Figure 17: Number of operations & weights for different layers in AlexNet



(a) Basic operation in an FC layer (b) multiple small-scale matrix multiplications

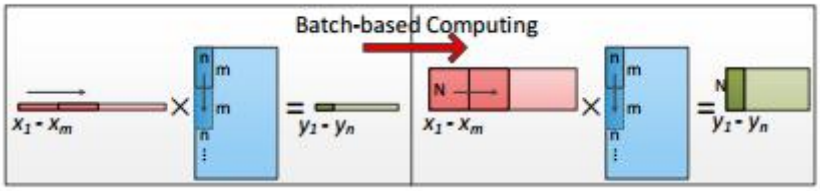


Figure 18: Batch-based-Computing

3.5 Literature Summary:

I. Computive Intensive: These structures are highly compute intensive so an Accelerator is required to improve the performance.

II. Local Memries: These Architectures used local memories for reducing the memory Transfers

III. Reconfigurable vs Accelerator: The CGRAs have the flexibility in the design and can adapt to the changes but they are not best in terms of Performance compared to Accelerator specific for Neural Networks

IV. Performance: The Performance of the NNs can be achieved by reducing the memory transfers by localising the data, by parallelisation and pipelining.

V. Tiling: Tiling technique is studied to reduce the Memory bandwidth.

VI. Ternary Weights: Improve the performance, better Area efficient.

PROPOSED ARCHITECTURE

4.1 TileNET

TileNET is a generic architecture for realizing Ternary Weighted Networks. As the name suggests, tiling of input kernel size and the weight vectors are used. A high-level representation of the TileNET accelerator template is shown in Figure 19. Since the dimensions of the input kernel and the weight vector vary with location of the layer in a model and across model, a single convolution layer may be composed of many such tiles. This modular design makes it scalable and portable across CNN models. Internally, the architecture is sub-divided into sub-blocks viz., computer, memory and control.

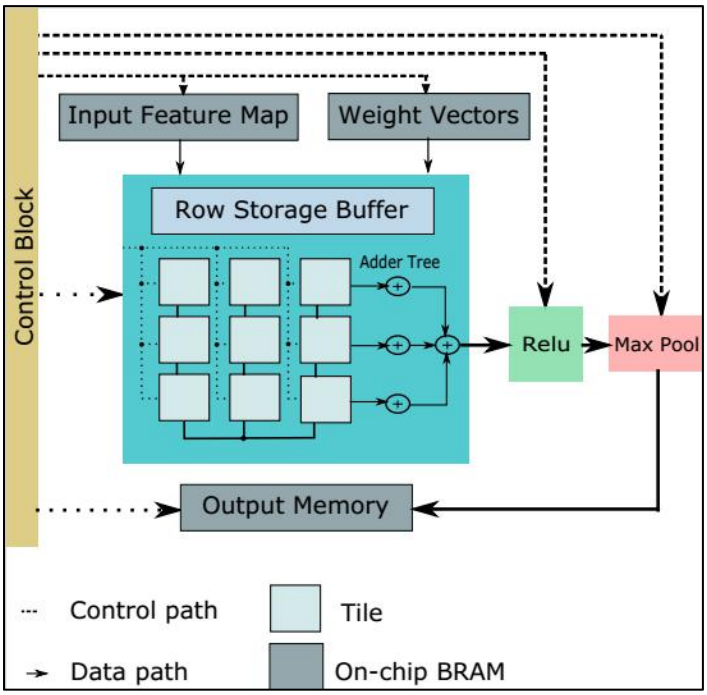


Figure 19: TileNet

4.1.1 Compute

A single tile accommodates the entire compute associated with a input kernel size of 3x3. The tile size (T) of 3x3 was considered, as for most of the layers a weight vector of size is 3x3 and it offers the optimal compute-memory ratio. In addition, the choice of 3x3 is a most frequently used weight vector size in newer CNN models. Nevertheless, weight vector sizes higher than 3x3 can be implemented by partitioning it into multiples of of 3x3 Tiles. Each tile

performs multiple accumulate operations to process an input kernel size of 3x3 and a weight vector of 3x3. The input feature vector is an 8-bit fixed point representation.

As weights are ternarized, they are represented by values -1, 0, 1. In binary format, 2 bits (00, 01, 11 for 0,1 and -1 respectively) are sufficient for ternary representation. For the ternarized weights, the multiplication operation is simply reduced to a multiplexing structure, as shown in Figure 20. For a weight value of '0', the output is a '0'. For an input weight of '1', the input kernel value itself is the result of multiplication. For an input weight of '-1', the result of multiplication is a two's complement representation of the input kernel value to represent subtraction. This form of simplification for the multiplication reduces the resource requirement for a multiplier, which is simply replaced by a 4x1 multiplexer with a 2s complement computation. As a result, the multiplier with ternarized weight vector has significantly lower resource requirements as compared to an 8-bit fixed-point multiplier. This reduction in resource requirement gives way to accommodating higher number of multiply-accumulate (MAC) units that can be realized in parallel. As shown in Figure 21, a single tile consists of 9 Ternary multipliers followed by 3-input adder tree. Multiple such tiles are instantiated in parallel for computing a part of a convolution layer. The Convolution layer is followed by Rectified Linear Unit (ReLu) and Max Pooling. The ReLu layer in essence performs a thresholding at 0 at each of its input value. The size of the vectors remains unchanged. The pooling layer is introduced between different convolution layers to reduce the dimension by downsizing. Max Pooling is performed by representing a neighborhood of values by their maximum.

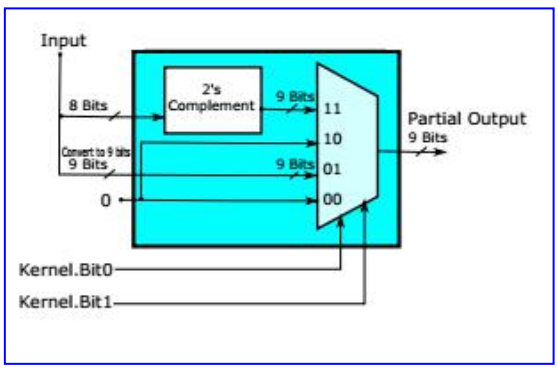


Figure 20: Ternary Compute

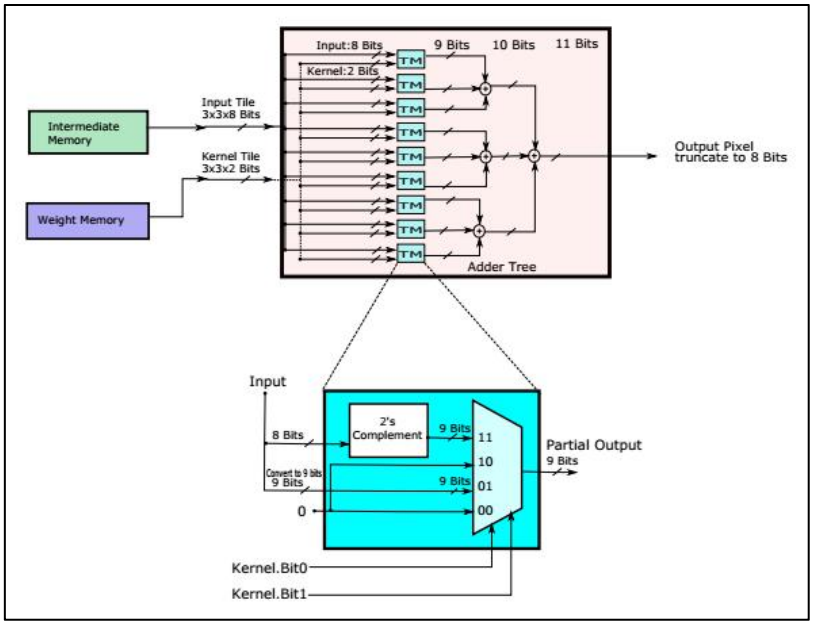


Figure 21: Processing Element

4.1.2 Memory Organisation

In general, the input kernel size is a R×C×N array of 8-bit fixed point values that is stored off-chip. The ternary weight vector has the dimensions of MxNxKxK. Data required to feed the compute elements in a tile, is stored in an intermediate storage called the Row Storage Buffer (RSB). The RSB is implemented as distributed LUT RAMs within each tile for close coupling with the ternary multipliers. The tiling and partitioning of data is shown in Figure 22. At start, the entire input image is stored in the off-Chip memory. (K+1) rows of input kernel is copied to the on-chip block RAM memory (BRAM), where K is Kernel width of the layer. For the next tile, the (K+1)th column data is fetched from the input memory to intermediate memory. The RSB stores only the NxKxK input kernel values. Each time this chunk of data is processed in the convolution layer, a new column of K values is fetched from the input memory the rest of the columns are left-shifted and left-most column is flushed out. After convolution is performed on the input, the data from the convolution layers is stored into output memory. As only M/C values are processed at a time, the input kernel values are reused C times till all the computation with the weight vectors have been completed. As at a time, M/C pixels are produced and stored to the output memory.This data from the output memory serves as input memory to next layer. As the Tile size T(3x3)remains

same across all the layers, the next layer can begin its compute after the required Tile T is available. The weight vectors are much smaller in size on account of ternarization. As a result, the entire weight vector is stored in on-chip Block RAM and as only inference is performed this memory is a read-only memory. Moving the weight vector closer to the ternary multiplier through the use of distributed memories can result in further performance improvement. $K\times$ kernels are stored in a single location with depth M/C and CxN such memories are used.

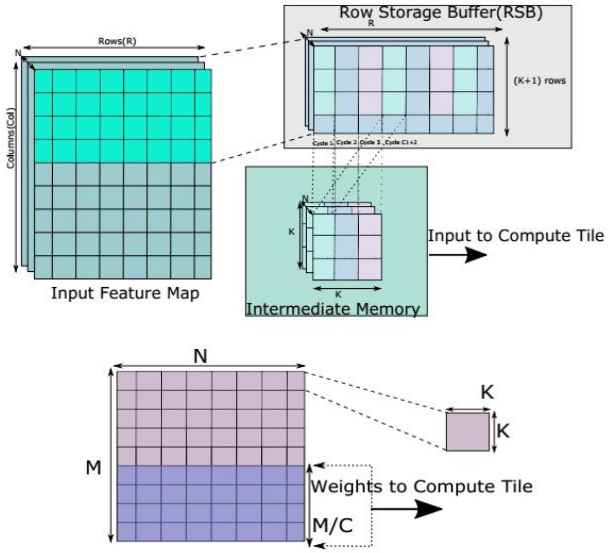


Figure 22: Memory Organisation

4.1.3 Control Path

The control path orchestrates the movement of data from memory to the compute units in the tile. The tiled organization necessitates sequentialized feeding of data (both from input kernels and weight vectors) to each tile. The resource constraints in terms of number of Compute Lookup Tables (LUTs) and the Memory (Block RAMs) available dictate the number of parallel tiles that can be accommodated on the FPGA. The factor C can be adjusted to accommodate parallel computations. The input is loaded column wise into the RSB. Once the N such 3x3 input tiles are ready in the RSB and the corresponding NxMxC KxK weight vectors are available, the signal to enable compute is sent, which performs the computation of M/C output pixels in parallel. This is then stored in the Output memory. This is repeated for C times, each time reading a new set of M/C weight vectors. The input is reused for all

of the weights for this iteration. Once all the computation using the Weight vectors are completed, the next column of input is brought into the RSB and the process is repeated. When a 3x3 tile is available in the output memory, the next layer compute can be enabled. As all the stages are pipelined, once the initial delay of filling the pipeline is done, then each stage will perform the computation of certain number of output pixels at a time.

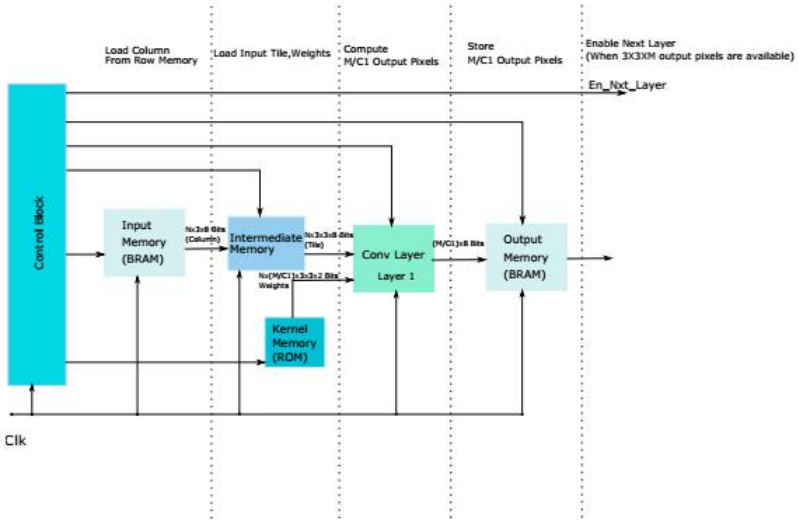


Figure 23: Control Path

4.2 Scalability and Portability

For the calculation of N input feature maps, PEs with NT tiles is replicated for N times. The parallelization factor(C) in Figure 24 determines the number of kernels that can be operated in parallel. M/C determines the number of times, the operations have to be done to compute all the M kernels. Scalability across different network architectures like that of LeNet, VGG and even ResNet-50 can be achieved by changing the parameter C. Given a set of resource constraints of a particular device, the different network architectures can be accommodated in a streaming fashion as per the maximum permissible compute and memory required it. In smaller networks like LeNet, as the inputs and weights are of smaller dimension, we can compute maximum output pixels in depth,i.e. keeping C as 1. In the case of LeNet, we can even compute pixels in parallel across rows and columns, so as to improve the compute utilization. ResNet-50 architecture has maximum number of convolution layers of all the architectures considered in this paper. To accommodate a

streaming architecture here, initial layers compute a pixel at a time and then later layers can produce only partial outputs every clock (by considering only part of the inputs at a time N/C*: c* as parallelization factor or Input Depth) to ensure that the compute utilization doesn't exceed. Portability across devices is ensured by varying C as per the resource constraints of each device. Ranging from high resource availability in Virtex Devices to sparse resources of Spartan device, C can be varied as per the amount of computation and memory available.

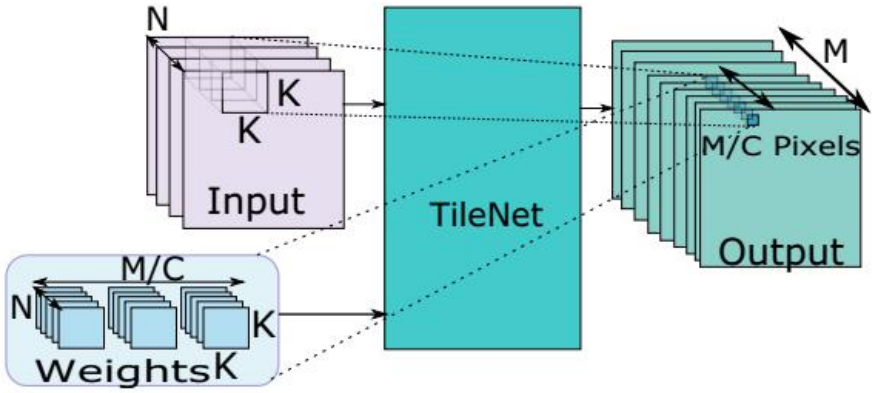


Figure 24: Scalability of TileNet

IMPLEMENTATION

The Architecture has been implemented in Xilinx FPGA.
The Tool Used: Xilinx Vivado
The Process technology of Xilinx 7 series is 28nm

5.1 Implementation of a Processing Element:

For a single PE, the implementation is done on the different devices like Virtex, Kintex, Zync and Artix. The results for the implementation are shown in Table 6 show that the utilization remains similar across devices. The total on-chip power consumption for each tile is also shown in the Table 6.

From the implementation results on the different devices, it shows that Virtex 7 have a good frequency of operation while having the maximum resources which is ideal for higher throughput when compared to other devices.

5.2 Implemenation for Convolutional Layers:

The performance estimation model provides the area and memory estimates based on the maximum amount of parallelization. Using the C values from Table 5 the convolution Layers for AlexNet using the TileNET template have been implemented on Virtex 7 FPGA device using Vivado Design Suite. The implementation results summarized the Table All the convolutional Layers are implemented in Vertex 7 (results in Chapter 6).

A Convolutional Layer has been designed in the work. This Convolutional Layer can be scalable . i,e., same design can be used for implementation of any convolutional layer by changing the design parameters like No. Of Input feature maps(N), kernel weights(M), Parallelization Factor(C), No. Of Tiles(Nt).

5.2.1 Design Parameters:

N - input feature Maps
M - output feature Maps(No. of kernel Weights)

k - kernel width

Nt - No. of tiles in Kernel

Rin,Cin - input feature map width and height

Ro,Co - output feature map width and height

C - Parallelization factor

M/c - No. of times layer has to run

By changing the parameters for the each layer. The design is Implemented in Vertex 7 w.r.t each convolutonal Layer.

The Parameters N, C are maintained for the different layers in a such a way that maximum throughput can be achieved with available resources on Virtex 7(see Table 5)

Layer	N	M	K	Nt	Rin	Cin	Ro	Co	M/C	C value	Max Pooling	Ternary Compute	LUTS
Conv1	3	96	11x11	16	224	224	55	55	16	6	3x3	2592	97344
Conv2	48	256	5x5	4	55	55	27	27	128	2	3X3	3456	129792
Conv3	256	384	3x3	1	13	13	13	13	64	2	NA	4508	173056
Conv4	192	384	3x3	1	13	13	13	13	64	2	NA	3456	129792
Conv5	192	256	3x3	1	13	13	13	13	64	2	3X3	3456	129792
Total												**17568**	**659776**

Table 5: Overall Estimation of the Ternary Compute based on the Parallelisation Factor

5.3 Throughput Calculation:

The Throughput Calculation for The Neural Networks is done w.r.t the Number of MACs.

Throughput = Number of MACs/cycle

= Number of MACs x Frequency.

The MACs are Multipliers followed by the Accumulation(addition). These MACs are calculated in our design as

No. Of MACs = Total no. Of Ternary Compute * 2.

The Total number of Ternary Compute gives the total Multiplication operations which are performed in a cycle. The addition is followed the multiplication, which is be almost same the number of Multipliers. Hence we take No. Of Macs as twice the total number of Ternary compute.

Therfore,

Throughput = Total number of Ternary Compute * 2 * Frequency

RESULTS

6.1 Processing Element on Various devices

The Processing Element has been implemented on different devices. Table 6, shows the results

DEVICE	LUTS	FFs	Delay(ns)	Power(W)
xc7vx1140(Virtex)	318	600	1.58	0.75
xc7k480(Kintex)	317	600	1.52	0.31
xc7z100(Zniq)	319	600	1.58	0.26
xc7a200(Artix)	319	600	2.39	0.23

Table 6: Implementation results for the PE on various devices

From the implementation results on the different devices, it shows that Virtex 7 have a good frequency of operation while having the maximum resources which is ideal for higher throughput when compared to other devices.

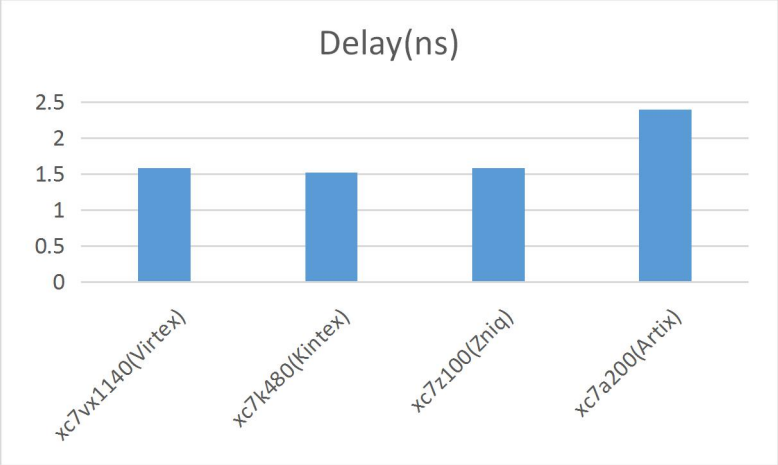


Figure 25: Comparison of Frequencies for PE on various devices.

Figure 26: Power Consumption of PE of various devices

Power consumption is less for Artix as it is low end device compared to others and operating at lower frequency.

6.2 Implementation results of Convolutional Layers of AlexNet

The results are implemented in Vertex 7

Device	xc7vx1140(Vertex 7)
LUTS	712000
FFS	1424000
BRAMS	1880

LUTS	LUTS Util %	FFs	FF Util %	BRAM	BRAM Util%	DELAY(ns)	Freq(MHz)	Power (W)
94177	13.22	187488	13.16	78	4.14	1.96	509.16	12.39
125535	17.63	249882	17.54	84	4.46	2.48	403.06	12.56
167391	23.5	333290	23.4	160	8.51	2.55	391.69	16
125535	17.63	249882	17.54	120	6.38	2.02	494.8	15.24
125535	17.63	249882	17.54	120	6.38	2.02	494.8	15.24
638173	89.63	1270424	89.21	562	29.89	2.55	391.69	71.43

Table 7: Implemented results for Convolutional Layers in AlexNet

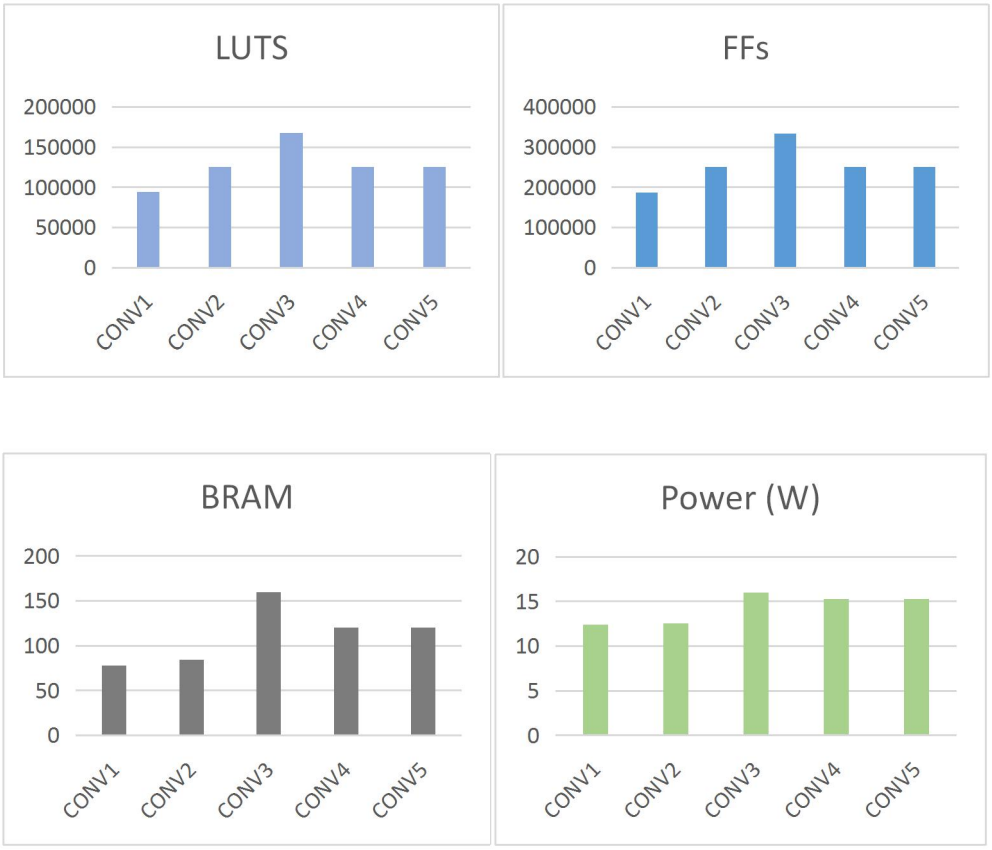


Figure 27: Comparison of LUTS, FFs, BRAMs and Power of Convoultional Layers

6.3 Overall Throughput

Based the synthesized results, the maximum frequency that can be achieved for the layers combined is 453MHz.
The Total Number of Ternary Compute is 37152.

The Overall Throughput is,
Throughput = Total ternary compute*2*Frequency

Throughput = 33.68 TOPs

CONCLUSION

ADAS necessitates high-throughput in the order of about few 10's of TeraMACs per second (TMACS). In addition to throughput, accuracy is also of very high importance. This work proposes a scalable, tile based Architecture for the Convolutional Neural Networks. Ternarization of weights is adopted to improve the Throughput. As a case study, a large-scale CNN model, AlexNet is implemented on Xilinx VC71140; it achieves 13.76 **TOPs** which outperforms the previous work.

REFERENCES

[1] FIST: A Framework to Interleave Spiking neural networks on CGRAs Tuan Ngyen, Syed M. A. H. Jafri, Masoud Daneshtalab, Ahmed Hemani, Sergei Dytckov, Juha Plosila, and Hannu Tenhune, Turku Centre for Computer Science, University of Turku, Finland, Royal Institute of Technology, Sweden, 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

[2] DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen SKLCA, ICT, China, Olivier Temam Inria, France, ASPLOS '14, March 1–5, 2014, Salt Lake City, Utah, USA.

[3] A CGRA-based Approach for Accelerating Convolutional Neural Networks Masakazu Tanomoto, Shinya Takamaeda-Yamazaki, Jun Yao, and Yasuhiko Nakashima, Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan, 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip.

[4] Li, Huimin, et al. "A high performance FPGA-based accelerator for large-scale convolutional neural networks." Field Programmable Logic and Applications (FPL), 2016 26th International Conference on. IEEE, 2016

[5] H. Alemdar, N. Caldwell, V. Leroy, A. Prost-Boucle, and F. P´etrot. Ternary Neural Networks for Resource-Efficient AI Applications. CoRR, abs/1609.00222, 2016.

[6] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. New York, NY, USA: Cambridge University Press, 2011.

[7] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 257–260.

[8] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*,

2014, pp. 682–687.