IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

A HIGH THROUGHPUT DOUBLE MAC FOR HARDWARE ACCELERATION IN DEEP LEARNING

[1]Anusha.C A, [2]Shylaja V.
[1]Student , [2]Assistant Professor
[1]*Department of VLSI and Embedded Systems, Bangalore Institute Of Technology, Bangalore,India*

Abstract: Convolutional Neural Networks which is a class of deep learning have an significant workload which increases demand for high-performance hardware acceleration. This paper is proposed to increase the rate of computation of accelerators by enclosing double multiply and accumulate (MAC) operations into single DSP block of offshelf FPGAs (called Double MAC). The method is validated synthesis by FPGAs and verilog code simulation. The distinguishing feature of Deep learning it can withstand tiny mathematical errors and works extremely fine with hardware which are less accurate. This paper presents a technique that can increase the rate of computation of a CNN layer by twice.In this paper we implement a Double Multiply Accumulate (MAC) unit based on a SIMD multiplication and 8-bit Ripple carry adder which is again replaced by 8-bit Kogge-Stone adder (KSA) for a DSP module in order to achieve high throughput. The evaluation results shows that Double MAC approach not only can increase the computation throughput of a CNN layer by twice but also has essentially reduced area, delay and power

Index Terms - Ripple Carry Adder,Kogge Stone Adder,Multiply Accumulate unit, Single MAC unit, Double MAC unit,CNN,Deep Learning.

I. INTRODUCTION

Convolution neural network (CNN) has achieved remarkable success in many applications because of its powerful capability for data processing. Their performance in computer vision have matched and in some areas even surpassed human capabilities. Convolution neural networks can capture complex nonlinear features; however this ability comes at the cost of high computational and memory requirements. State-of-art networks require billions of arithmetic operations and millions of parameters.

As machine learning algorithms are getting more popular, there is an increasing demand for developing hardware accelerators for them. In particular deep neural networks such as Convolutional Neural Networks (CNNs) have multiple traits that make them very attractive for hardware acceleration, such as high structural regularity, high computational complexity, and yet wide applicability and high recognition performance. FPGAs are one of the most preferred platforms due to their high flexibility and at the same time high parallelism. Hence much effort has been made to create better CNN accelerators on FPGAs.

A unique option available to hardware implementations of CNNs is the flexibility in data width of arithmetic operations.GP-GPUs, for instance, have long provided only two options - either single-precision or double-precision floating point since integer arithmetic on modern GP-GPUs has zero or negative performance advantage. Recently half-precision was introduced on some select models, but this is a one-time change and not customizable by user. By contrast an ASIC(Application-Specific Integrated Circuit) implementation can choose whatever precision sufficient for the target CNN application.

There is a good opportunity to increase performance for free by using lower precision without affecting output quality. FPGAs, too, have the flexibility, and using reduced precision means potentially higher throughput on the same FPGA. In practice, however, since most arithmetic operations are implemented using DSP blocks, and DSP blocks, too, support only a limited set of precisions, it is not easy to achieve higher performance through reduced arithmetic precision. For example, the DSP block of Xilinx FPGAs, DSP48E1, can perform a 25x18-bit multiplication only, and there is no way to perform two 8x8-bit multiplications simultaneously on the same DSP block for higher throughput.

The brute-force computing model of CNN often requires extremely large hardware resources, introducing severe concerns on its scalability running on traditional von Neumann architecture. The well-known memory wall, and latency brought by the long-range connectivity and communication of CNN severely constrain the computation efficiency of CNN. The acceleration techniques of CNN, either software or hardware, often suffer from poor hardware execution efficiency of the simplified model (software), or inevitable accuracy degradation and limited supportable algorithms (hardware), respectively.

The continuing advancement of the neural networks based techniques led to their exploitation in many applications, such as computer vision and the natural language processing systems where they provide high accuracy results at the cost of their high computational complexity. Hardware implemented AI accelerators provide the needed performance improvement for applications in specific areas, including robotics, autonomous systems and internet of things.

This paper is about how to turn an ordinary DSP block ofan off-the-shelf FPGA device into a 2-way SIMD (Single-Instruction Multiple-Data) MAC (Multiply-and-Accumulate)unit, that can deliver 4 ops/cycle by performing two multiply- and add operations simultaneously with reduced data width.

We propose a special class of SIMD multipliers, tailored for the kind of MAC operations found in convolutional layers of CNNs. Specifically, our Double MAC requires that the multiplications of a SIMD multiply share a common operand, viz., A×C and B×C, and that the common operand C, be an unsigned integer. We assume that only testing, as opposed to training, of a CNN is done on an FPGA. We also assume that the CNN accelerator accelerates convolution layers only, which account for the vast majority of computation.

We validate our method through Verilog simulation and FPGA synthesis, and evaluate our method by applying it to one of the state-of-the-art CNN accelerator designs. We demonstrate that our Double MAC can double the computation throughput at the MAC operation level, and also at the MAC array level when given the same number of DSP blocks, as compared to the previous state of the art.

This paper is organized as follows. Section II presents the literature survey based on the work carried out in project. i.e., topics related to applications for CNN, different implementations on MAC Unit and so on. Section III presents details carried out in implementing the proposed system of the project like the Double MAC architecture, Ripple Carry Adder and Kogge Stone Adder. Section IV presents the simulation and hardware implementation results carried out using Xilinx and FPGA Platform. Finally, conclusion and the future scope of the project is presented.

II. LITERATURE SURVEY

Convolutional Networks (ConvNets) are biologically inspired hierarchical architectures that can be trained to perform a variety of detection, recognition and segmentation tasks. ConvNets have a feed-forward architecture consisting of multiple linear convolution filters interspersed with point wise non-linear squashing functions. Cyril Poulet et al.[1] presents an efficient implementation of ConvNets on a low-end DSP oriented Field Programmable Gate Array (FPGA). The implementation exploits the inherent parallelism of ConvNets and takes full advantage of multiple hardware multiply accumulate units on the FPGA. The entire system uses a single FPGA with an external memory module, and no extra parts. A network compiler software was implemented, which takes a description of a trained ConvNet and compiles it into a sequence of instructions for the ConvNet Processor (CNP).The design can be used for low-power, lightweight embedded vision systems for micro-UAVs and other small robots.

Kyounghoon Kim et al.[2] presents an efficient DNN design with stochastic computing. Observing that directly adopting stochastic computing to DNN has some challenges including random error fluctuation, range limitation, and overhead in accumulation, we address these problems by removing near-zero weights, applying weight-scaling, and integrating the activation function with the accumulator. The approach allows an easy implementation of early decision termination with a fixed hardware design by exploiting the progressive precision characteristics of stochastic computing, which was not easy with existing approaches.

The MAC unit is a unit that is mostly demanded in DSP applications. MAC unit performs both multiply and addition functions. It operates in two stages. Firstly it computes the product of given numbers and forward the result for the second stage operation i.e. addition/accumulate. If both the computing is executed in a single rounding then it is said to be fused multiply-add/accumulate (MAC) unit. There has been a lot of research performed on MAC implementation. Priyanka Nain et al.[3] provides a comparative study and analysis of the research and investigations held till now.

Different implementations for MAC unit for various application has been observed. But the high demand of MAC unit is in Digital signal processing field. Basically it provides hardware support for number of DSP applications. It also supports signed/unsigned integers plus signed fixed-point fractional input operands. Fused MAC unit executes faster than basic MAC unit. The use of MAC unit in DSP applications is not limited upto multiplication and addition but it performs well the division, squares, and square-root operations also. It can be used in digital filters. A huge amount of data needs to be transferred quickly affects the throughput in DSP applications.

Carry-free redundant arithmetic based fused multiply-accumulate (MAC) units are designed by Ugur Cini et al.[4]. In the first design, a regular redundant carry-save MAC unit is designed using well known carry-save techniques. In the second design, a hybrid design is proposed to exploit fast carry chains of the FPGA together with double carry-save output encoding. The proposed scheme exploits fast-carry chains of the FPGA structure, and, multi-operand adders are divided into smaller blocks to increase the performance. The outputs of the multi-operand adders are not merged and the results are kept in double carry-save format where extra redundancy reduces critical path delay. Designed MAC units have 16x16- bit multiplier with 40-digit accumulate output for recursive multiply-add operations.

Deep neural networks (DNN) have been a hot research topic in recent years. Convolutional Neural Network (CNN) is the popular architecture of DNN especially for image classification. One requires an efficient implementation strategy of CNN to incorporate more computations in real time. Field Programmable Gate Array (FPGA) is considered to be the energy efficient choice for CNN as compared to Graphical Processing Units (GPUs). Fasih Ud Din Farrukh et al.[5] proposed a new idea and implemented for basic Processing Element (PE) of CNN. FPGA has limited built-in multiplier accumulator (MAC) units. In this work[5], MAC units are replaced by Wallace Tree based Multiplier which belongs to the family of log time array multipliers. The resources are saved in terms of MAC units and we can implement more processing elements on FPGA.
In [7] Roofline model based design where computation throughput and memory bandwidth is analyzed by optimization techniques.

III.PROPOSED SYSTEM

3.1 Objectives

- To design a Double Multiply Accumulate (MAC) unit to increase the throughput of the cnn accelerator.
- Compare Single MAC and Double MAC in terms of area, power and delay.
- To design a Double Multiply Accumulate (MAC) unit based on 8-bit Ripple carry adder and 8-bit Kogge Stone Adder and compare the same in terms of area, power and delay.
- Finally to conclude the best architecture that achieves high throughput Double MAC for Hardware Acceleration in Deep Learning.

3.2 SIMD Multiplication of Unsigned Numbers

Let us first consider the case where every operand is unsigned. Two unsigned multiplications with a common operand, i.e., A × C and B × C, can be done using a single multiplier as illustrated in Figure 3.1. For this to work, two conditions must be met. First, the output register must be at least 4n-bit wide for n-bit operands. Second, the two operands in one of the inputs must be separated by at least n bits. While that is enough for a single multiplication, for the addition ensuing the multiplication to work without overflow, we need one guard

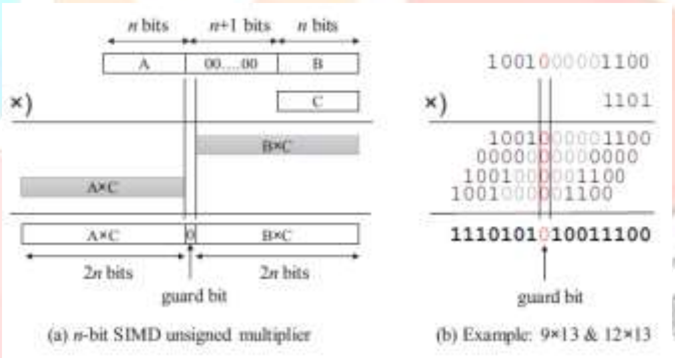

Fig. 3.1. How 2-way SIMD multiplication works (unsigned case).

bit as illustrated in the figure .In the accumulation mode, the same 1-bit guard bit can be used to detect any carry out of the lower 2n bits. On detection the carry is cleared immediately and the number of carry-out events is counted separately using a small counter. The counter value is added later once all the accumulation is done. Thus to perform two n-bit multiplications in a SIMD fashion we need one (3n+1)×n-bit multiplier. For example, with the 25x18-bit multiplier of a DSP48E in Xilinx FPGAs, n can be at most 8.

3.3 SIMD Signed-Unsigned Multiplication

Let us consider how to perform a signed-unsigned multiplication using an unsigned multiplier. Let $B = b_{n-1} \cdot\cdot\cdot b_1 b_0$ be an n-bit signed integer, and C an n-bit unsigned integer. Recalling $-2^{n-1} = 2^{n-1} - 2^n$, the product B × C can be computed as follows, where ˆB represents the value of B interpreted as an unsigned number.

$$\begin{aligned} B\times C &= \left(-b_n.2^{n-1} + \textstyle\sum_{i=0}^{n-2} b_i 2^i\right).C \\ &= \left(b_n.2^{n-1} + \textstyle\sum_{i=0}^{n-2} b_i 2^i\right).C - b_n.2^n.C \\ &= B.C - b_n.2^n.C \end{aligned}$$

In other words, B × C can be computed by performing an unsigned multiplication on B and C followed by a subtraction of n-bit left shifted C if B is negative.

Extending this to 2-way SIMD multiplication is straightforward. We can perform two n-bit signed-unsigned multiplications in a SIMD fashion by performing one (3n+1) × n-bit unsigned multiplication, followed by at most two subtractions.However we can reduce the number of subtractions to one by performing a signed multiplication for the (3n + 1) × n-bit multiplication. In this case the upper 2n-bit (i.e., A × C) is correct (no need for correction), and only the lower 2n-bit needs correction if B is negative, as demonstrated by example in Fig 3.2.

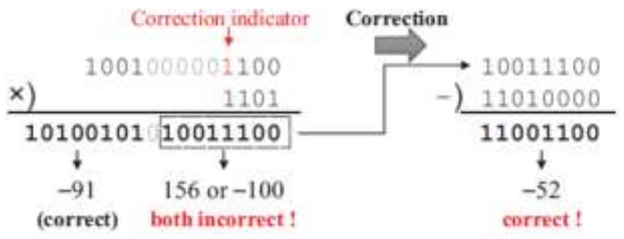

Fig. 3.2. Example SIMD execution: −7 × 13 and −4 × 13.

The correction term could be added in-place if we do just one multiply-add operation. In the accumulation (i.e., MAC) mode however we must delay doing corrections until the end of accumulation because we have only one guard bit. Instead the correction terms are accumulated separately in a small accumulator, whose value is later subtracted from the main accumulator.

3.4 Proposed Double MAC Architecture

Fig. 3.3 illustrates our Double MAC architecture. One DSP block is needed to implement both a 2-way SIMD multiplier and a 2-way SIMD adder. And each of the SIMD multiplier and the SIMD adder has a correction output signal, which is shown as thin arrows in the figure. The correction outputs are accumulated into a separate register/counter and used later for final adjustment. During adjustment there are two terms that need to be added/subtracted. The adder-correction term C1, which comes from the carry-out counter, has no overlap with the main accumulator output, and thus they can be just concatenated. Therefore we need only one addition, or the subtraction of the multiply-correction term C2 from the concatenation result.

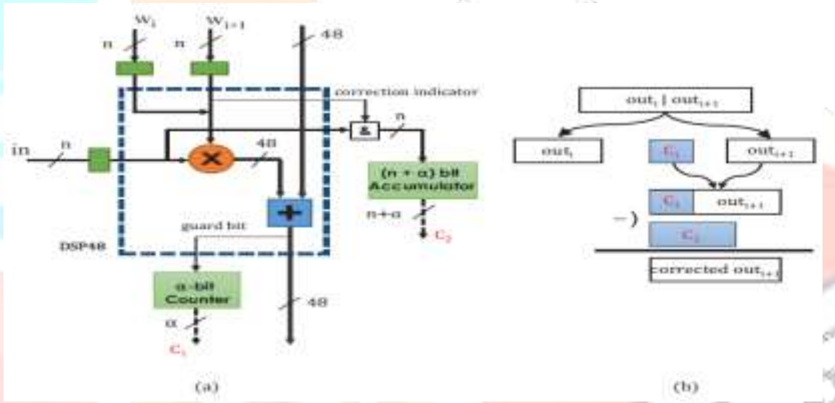

Fig. 3.3 (a) Datapath of our Double MAC architecture and (b) how it works.

The width of the carry-out counter is determined from the number of values accumulated. In the case of our baseline CNN accelerator, the width α must satisfy this constraint α ≥log2(K2N/TN) for every layer of the CNN, where K is the size of convolution filter in one dimension. Similarly the width of the accumulator for multiplier correction terms is n + α.
The post-accumulation adjustment can be done easily by using an extra adder of (n + α)-bit width that resides outside the DSP block. There is virtually no runtime overhead due to this adjustment, since the adjustment can be done simultaneously while new values are loaded into the MAC.

3.5 CNN Accelerator

A convolutional layer takes as input a number (N) of matrices called input feature maps, and generates a number (M) of matrices called output feature maps. The computation of each output feature map (Ym) involves the summation of N 2D-convolutions between each of the input feature maps (Xn) and each of the N weight parameter matrices (Wm,n). Ignoring bias addition, $Y_m = \sum_n^N W_{m,n} * X_n$ where ∗ represents convolution.

Naturally the repetitions along the M, N dimensions have been the source of parallelism exploited by multiple hardware accelerators.
Our proposed SIMD MAC architecture is applicable to other CNN accelerators as well. One important concern for a reduced-precision scheme such as ours is lower accuracy at the network level. In this section we analyze the precision impact of our Double MAC architecture and present a very low-cost mitigation scheme using Ripple Carry Adder and Kogge Stone Adder in Double MAC Architecture.

3.6 Ripple Carry Adder

A structure of multiple full adders is cascaded in a manner to gives the results of the addition of an n bit binary sequence. This adder includes cascaded full adders in its structure so, the carry will be generated at every full adder stage in a ripple-carry adder circuit. These carry output at each full adder stage is forwarded to its next full adder and there applied as a carry input to it. This process continues up to its last full adder stage. So, each carry output bit is rippled to the next stage of a full adder. By this reason, it is named as "RIPPLE CARRY ADDER". The most important feature of it is to add the input bit sequences whether the sequence is 4 bit or 5 bit or any."One of the most important point to be considered in this carry adder is the final output is known only after the carry outputs are generated by each full adder stage and forwarded to its next stage. So there will be a delay to get the result with using of this carry adder".

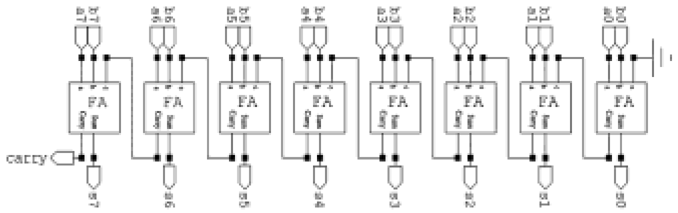

Fig. 3.4 Block Diagram of 8-Bit Ripple Carry Adder

3.6.1 Block Diagram Description

- It consists of 8 full adders which are connected in cascaded form.
- Each full adder carry output is connected as an input carry to the next stage full adder.
- The input sequences are denoted by (A0 A1 A2 A3 A4 A5 A6 A7) and (B0 B1 B2 B3 B4 B5 B6 B7) and its relevant output sequence is denoted by (S0 S1 S2 S3 S4 S5 S6 S7).
- The addition process in an 8-bit ripple-carry-adder is the same principle which is used in a 4-bit ripple-carry-adder i.e., each bit from two input sequences are going to added along with input carry.
- This will use when the addition of two 8 bit binary digits sequence.

3.6.2 Disadvantages of using RCA in Double MAC

- Ripple Carry Adder does not allow to use all the full adders simultaneously.
- Each full adder has to necessarily wait until the carry bit becomes available from its adjacent full adder.
- This increases the propagation time.
- Due to this reason, ripple carry adder becomes extremely slow.

3.7 Kogge Stone Adder

Due to the disadvantage of RCA in Double MAC it is replaced using the Kogge Stone Adder.The Kogge-Stone adder is a parallel prefix form of carry look-ahead adder. It generates the carry signals in O(log2N) time, and is widely considered as the fastest adder design possible. It is the most common architecture for high-performance adders in industry.In Kogge-stone adder, carries are generated fast by computing them in parallel at the cost of increased area. Tree structures of carry propagate and generate signals in 8-bit Kogge-Stone Adder (KSA) is shown in Figure 3.5.

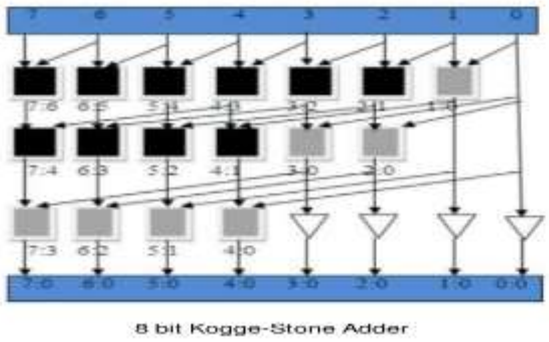

Fig.3.5 Block Diagram of Kogge-Stone Adder

3.7.1 Block Diagram Description

Kogge-Stone Adder has three processing stages for calculating the sum bits, they are:
1. Pre-processing stage
2. Carry generation (PG) network
3. Post-processing stage.

3.7.1.1. Preprocessing

This step involves computation of generate and propagate signals corresponding to each pair of bits in A and B.

Pi = Ai x or Bi--------------------------------(1)
Gi = Ai and Bi--------------------------------(2)

3.7.1.2. Carry Generation Network

In this stage we compute carries corresponding to each bit. Execution of these operations is carried out in parallel. After the computation of carries in parallel they are segmented into smaller pieces. It uses carry propagate and generate as intermediate signals which are given by the logic Equations(3 and 4):

CPi:j = Pi:k + 1 and Pk:j--------------------(3)
CGi:j = Gi:k + 1 or (Pi:k + 1 and Gk:j)---(4)

3.7.1.3. Post Processing

This is the final step and is common to all adders of this family (carry look ahead). It involves computation of sum bits.

Ci– 1 = (Pi and Cin) or Gi-------------------(5)
Si = Pi xor Ci– 1-------------------------------(6)

3.7.2 Advantages of using RCA in Double MAC

- Achieve better performance in terms of reduced power and delay.
- Low fan-outs
- High speed is achieved.

IV.RESULTS AND DISCUSSIONS

The simulation results of the conventional Single MAC Architecture for signed and unsigned multiplication operations are shown in Fig.4.1 and Fig.4.2 respectively.

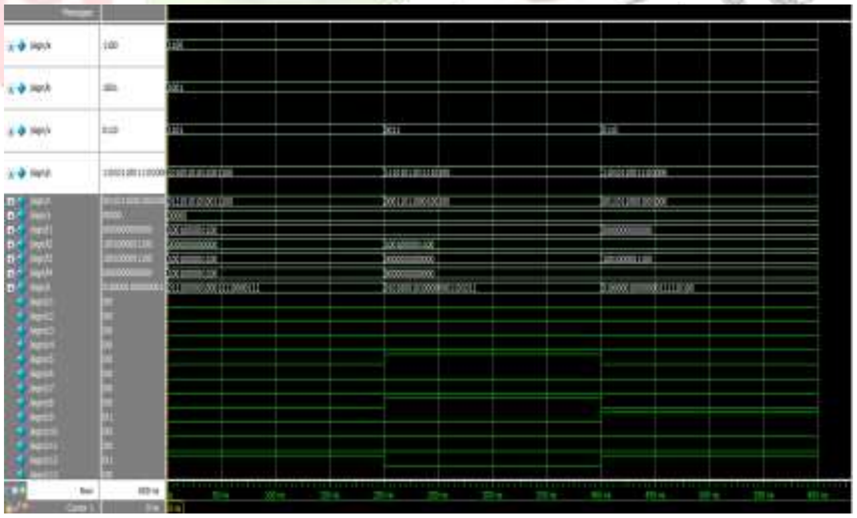

Fig.4.1 Simulation result for signed operation

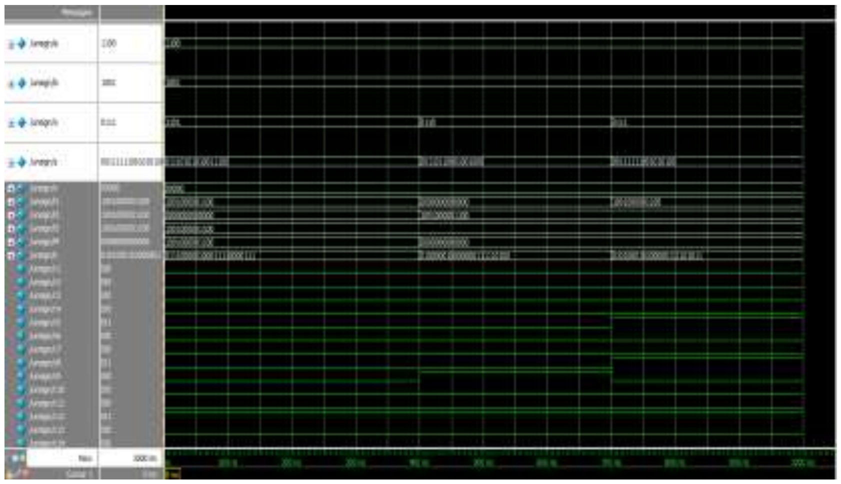

Fig.4.2 Simulation result for unsigned operation

The simulation results of the proposed Double MAC Architecture is shown in the Fig.4.3 below.

Table I shows the comparative study of area,power and delay for both single and double MAC architectures.

TABLE I: Comparison of Resource Single MAC and Double MAC Operations

	Area	**Power**	**Delay**
Single MAC	LUTs-386	535mW	22.006ns
Double MAC	LUTs-71	173mW	21.092ns

From the above Table I, it can be observed that the proposed double MAC architecture consumes minimum area,power and delay.
The modified double MAC with Ripple Carry Adder and Kogge Stone Adder are implemented and the results are compared. The simulation results of the proposed Double MAC Architecture using Ripple Carry Adder i shown in the Fig.4.4 and Fig.4.5 below.

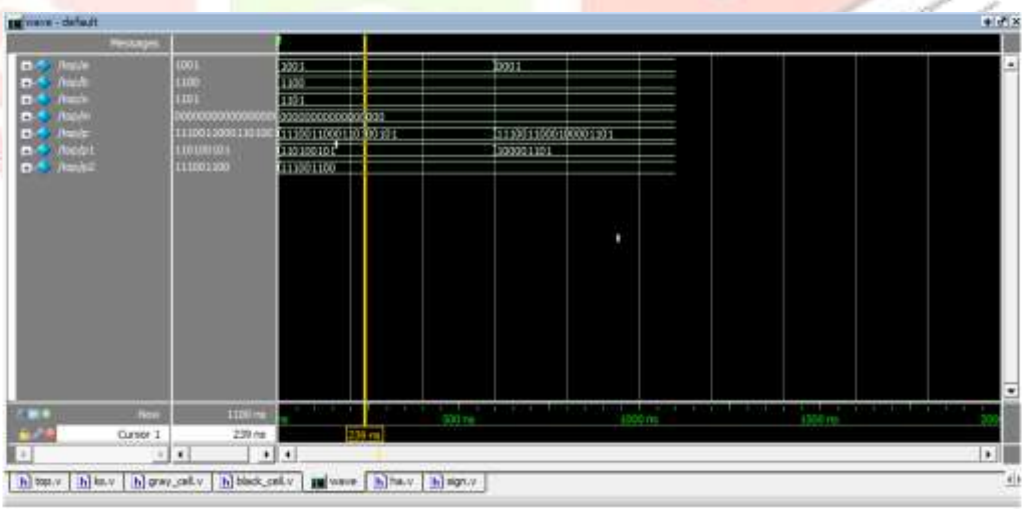

Fig.4.3. Simulation result of Double MAC Architecture

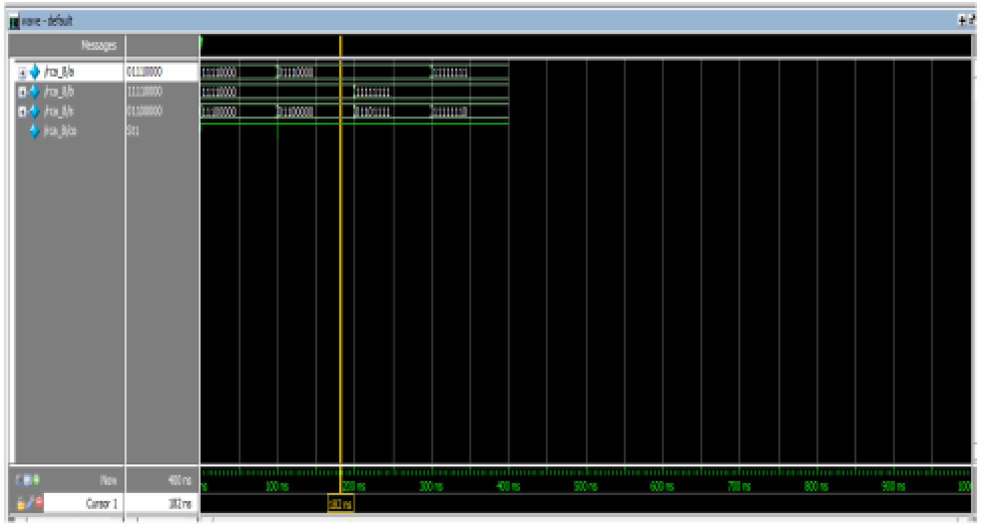

Fig.4.4 Simulation results of Double MAC using Ripple Carry Adder

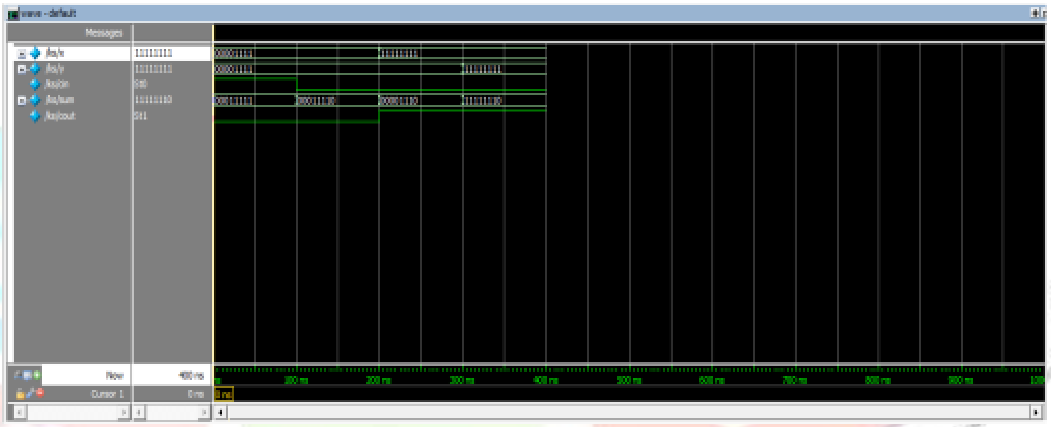

Fig.4.5. Simulation results of Double MAC using Kogge Stone Adder

Table II shows the comparative study of area,power and delay of both Ripple Carry Adder and Kogge Stone Adder.

TABLE II: Comparison of Resources of Ripple Carry Adder and Kogge Stone Adder

	Area	**Power**	**Delay**
Ripple Carry Adder	LUTs-114	112mW	38.858ns
Kogge Stone Adder	LUTs-93	108mW	35.461ns

From the above Table II, it can be observed that the proposed double MAC architecture using Kogge Stone Adder consumes minimum area, power and delay than Ripple Carry Adder.

V. CONCLUSION

We presented how to increase the computation rate of CNN accelerators on FPGAs by packing multiple MAC operations into one DSP blocks of off-the-shelf FPGAs. By exploiting the context in which MAC operations are used, our method can strike a good balance between usability and implementation overhead. Our experimental results demonstrate that our Double MAC can increase computation throughput of a CNN layer often by twice, and achieve significant performance improvements on the network level ranging from 14% to more than 80% over an already optimized accelerator solution. Our
scheme does use more LUTs, but LUTs are typically under-utilized in CNN accelerators, and our scheme can help improve balance in resource utilization. All these are done without sacrificing the output quality significantly. This improvement in performance can directly translate into energy saving, which can make accelerator solutions even more appealing as compared to GP-GPU solutions.

Future Scope

For the composition of adders, we only investigated two carry-propagation techniques: ripple-carry and kogge-stone.

The current functional units are not pipelined. For small functional units (e.g. 8-bit wide), this does not seem necessary. However, for implementations with somewhat wider functional units (such as 16-bit wide), that should operate at high clock frequencies, pipelining might help to reduce the delay of the multiplier circuit.

In future work, we can increase the number of bits and more focus may be applied to the high speed design part of the architecture.

REFERENCES

[1] D. Nguyen, D. Kim, and J. Lee, "Double MAC: Doubling the performance of convolutional neural networks on modern FPGAs," in Design, Automation and Test in Europe (DATE '17), Mar. 2017.

[2] Clement Farabet, Cyril Poulet, Jefferson Y. Han, Yann LeCun," CNP: An FPGA-Based Processor For Convolutional Networks", International Conference on Field Programmable Logic and Applications ,IEEE 2009.

[3] Kyounghoon Kim, Jungki Kim, Joonsang Yu, Jungwoo Seo, Jongeun Lee, and Kiyoung Choi, "Dynamic Energy-Accuracy Trade-off Using Stochastic Computing in Deep Neural Networks", 53nd ACM/EDAC/IEEE Design Automation Conference (DAC),2016.

[4] Priyanka Nain,G. S.Virdi,"Multiplier-Accumulator (MAC) Unit",International Journal of Digital Application & Contemporary Research, (Volume 5, Issue 3, October 2016).

[5] Ugur Cini,Olcay Kurt,"MAC Unit for Reconfigurable Systems Using Multi-Operand Adders with Double Carry-Save Encoding", International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS),2016.

[6] Fasih Ud Din Farrukh,Tuo Xie,Chun Zhang,Zhihua Wan,"Optimization for Efficient Hardware Implementation of CNN on FPGA",IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA),2018 .

[7] S. Grys, "Signed multiplication technique by means of unsigned multiply instruction," Comput. Electr. Eng., vol. 37, pp. 1212–1221, Nov. 2011.

[8]C. Zhang et al., "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ser. FPGA '15. New York, NY, USA: ACM, 2015, pp. 161–170.

[9] J. Qiu et al., "Going deeper with embedded fpga platform for convolutional neural network," in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 26–35.

[10] M. Peemen et al., "Memory-centric accelerator design for convolutional neural networks," in Computer Design (ICCD), 2013 IEEE 31st International Conference on, Oct 2013, pp. 13–19.

[11]N. Suda et al., "Throughput-optimized opencl-based fpga accelerator for large-scale convolutional neural networks," in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 16–25.