International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Benchmarking Custom Artificial Neural Network Hardware Accelerator

Prof. S. R. Ganorkar
Email: srgomom@rediffmail.com

Padmaraj A. Jain
Email: padmaraj12@gmail.com

***Abstract* – The goal of this work is to benchmark the custom design computational architecture supporting Artificial Neural Network (ANN) acceleration. The custom design optimizes the frequently used: multiply and accumulate (MAC) operations. In this work the performance of the custom design is compared with ARM and MIPS architectures supporting basic Single Instruction Multiple Data (SIMD) instructions. Benchmarking is performed by verifying the number of instructions required to compute n input neuron. The custom design is implemented using Very High Speed Integrated Circuits Hardware Description Language (VHDL) for Xilinx Spartan 6 Series FPGA Family. The execution speed is not considered as benchmarking parameter since the custom design is verified on the FPGA family. The ARM and MIPS architectures referenced here are basic architectures supporting SIMD instructions. Loading the inputs into registers and storing the results back into the memory depends upon the bus architecture supported and varies from architecture to architecture. Load and store instructions are not part of the benchmarking.**

***Keywords* – ANN, MAC, RISC, SIMD, VHDL.**

I. INTRODUCTION

ANN has wide variety of applications in areas of pattern recognition, image processing and medical diagnostic. ANN is parallel distributed information processing system. The frequently used operation in case of ANN is MAC operation [1]. Maximum computational throughput can be achieved by implementing the MAC operation in parallel. More number of bits describing the input and the output will reduce error in cumulative computations and eventually in final output. Parallel hardware and additional bit accuracy increases the cost.

Proposed custom design tries to achieve the balance between the throughput and chip cost. The proposed architecture is scalable in terms of ANN and can be used for general purpose computing as well. As this design target the computationally intense part in most types of the neural network; it is flexible to use in computing different types of neural network [1].

Similar general purpose architectures are available such as ARMv6 and MIPS SIMD series. These architectures support parallel MAC operations.

The benchmarking of the custom design can be done by implementing the n input neurons on above mentioned architectures and comparing the results. As we are comparing architectures which support SIMD instructions, the comparison can be performed with parameters such as: number of instructions to load input registers, input register precision, output register precision and number of instructions to perform MAC operations. Loading multiple registers in single cycle depends upon the bus architecture supported. As we wish to compare the computational capabilities of the hardware we are not comparing the load store mechanism in the comparing architectures. All comparing architectures are 32 bit architectures with 32 bit registers. The MAC instructions in each of the architecture differ in terms of the precision of the data manipulated during the instruction. The precision of the output depends on the instruction used in each of the architecture.

II. ARTIFICIAL NEURAL NETWORK (ANN)

A. *ANN Architecture*

Basic architecture of ANN can be described using example of multi-layer Perceptron. It consists of several layers of processing elements (PE's) or artificial neurons [5]. It has minimum three layers of artificial neurons:

- Input layer: Real time inputs are connected to this layer.
- Hidden Layers: One or more hidden layers are connected to input and output layer.
- Output Layer: Final layer processing output from hidden layers.

An input vector is presented to the neural network which determines an output. The comparison between the computed and desired output provides an output error [5]. The error is used by a learning algorithm to adapt the network parameters. Fig.1 shows diagrammatic representation of multi-layer Perceptron with one input layer, two hidden layers and one output layers. Each circle represents neuron or PE.

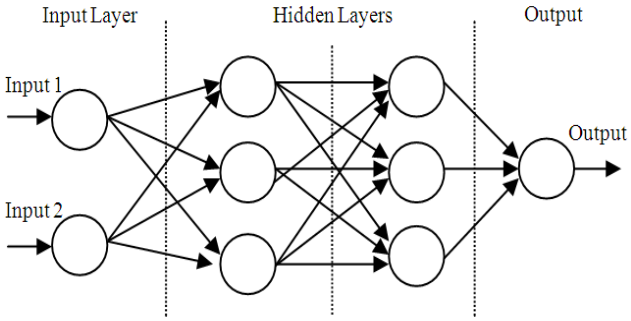

Fig.1. Multi-layer Perceptron Network

B. *Mathematical Model of Artificial Neuron*

General neuron of ANN can be described in mathematical form as shown in (1).

$$Output_n = f(\sum_{i=0}^{N}(w_i \times x_i)) \quad (1)$$

Where,

n = Index of neuron,
f = Activation function,
N = number of input or weights,
w = weights,
x = inputs

International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

The model of neuron in ANN based on above equation is shown in Fig.2

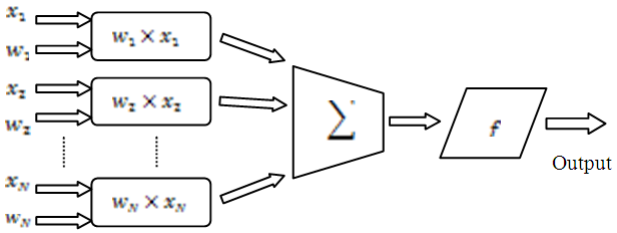

Fig.2. Mathematical Model of neuron.

III. ANN Custom Design

A. Introduction

From Fig.1 we are able to observe that there is inherent spatial parallelism in the execution of the neuron's products, called the intra-neural parallelism. Each Neuron has mathematical operation of MAC, where multiple multiplication and addition operation can be performed in parallel [1]. The proposed custom design in this paper explores parallelism in MAC operation and the property of the neuron layer where input is fixed. In case of computing single layer, input is fixed and weights change from neuron to neuron. Input is captured in input registers which are constant for one layer.

B. Hardware Design

The important components of proposed architecture which support parallelism in MAC operation are:

- 4 x 32-bit Multipliers
- 2 input, 4 x 32-bit / 4 input, 1 x 64-bit Adder
- 16 x 32-bit input registers (I0 – I15)
- 4 x 32-bit weight registers (W0 – W3)
- 1 x 32-bit address register
- 8 x 32-bit / 4 x 64-bit output registers

The block diagram of the proposed custom design is shown in Fig.3. The Input register file is a special purpose register file of 16 x 32-bit registers. At a time only 4 x 32-bit register can be accessed. Weight register file is a general purpose register file of 4x32-bit registers. One address register is also included in the architecture to load the input registers and weight registers with values from memory. Table I shows details of Operations supported by the proposed custom architecture:

Table I: ANN Accelerator Operations.

S. No.	Operation	Src 1	Src 2	Dest
1	Move	In	-	In
2	Move	Wn	-	Wn
3	Move	#immediate	-	Address register
4	Load (/repeat)	Memory	-	In
5	Load (/repeat)	Memory	-	Wt
6	Multiply	I0/I1/I2/I3	W0	O0-O1
7	Multiply	I4/I5/I6/I7	W1	O2-O3
8	Multiply	I8/I9/I10/I11	W2	O4-O5
9	Multiply	I12/I13/I14/I15	W3	O6-O7
10	Multiply	I0/I1/I2/I3	W0	O0-O1
		I4/I5/I6/I7	W1	O2-O3
		I8/I9/I10/I11	W2	O4-O5
		I12/I13/I14/I15	W3	O6-O7
11	Add	I0/I1/I2/I3	W0	O0-O1
12	Add	I4/I5/I6/I7	W1	O2-O3
13	Add	I8/I9/I10/I11	W2	O4-O5
14	Add	I12/I13/I14/I15	W3	O6-O7
15	Add	I0/I1/I2/I3	W0	O0-O1
		I4/I5/I6/I7	W1	O2-O3
		I8/I9/I10/I11	W2	O4-O5
		I12/I13/I14/I15	W3	O6-O7
16	MAC (/repeat)	I0/I1/I2/I3	W0	O0-O3
		I4/I5/I6/I7	W1	
		I8/I9/I10/I11	W2	
		I12/I13/I14/I15	W3	

IV. ARMv6 Architecture

A. Introduction

In terms of instruction set, the ARM11 builds on the preceding ARM 9 generation [3]. It supports SIMD instructions which can double video and digital signal processing algorithm speed [3]. It contains 8 stage instruction pipeline architecture. It uses ALU parallelism to implement SIMD instruction. It supports 64 bit data path. The architecture has MAC pipeline which supports enhanced multiply, multiply and accumulate instructions. It has a load and store unit. The load-store pipeline decouples loads and stores from MAC and ALU pipeline [3]. There are 31 x 32 bit general purpose registers and 6 x 32 bit status registers. These registers are not all accessible at the same time. The processor state and operating mode determine which registers are available to the programmer.

B. Multiply and MAC Instructions

ARMv6 core supports large number of instructions. We are targeting instructions which are supporting multiplication and multiplication and accumulate. Table II shows the Multiply, Multiply accumulate instructions supported by ARMv6 Architecture.

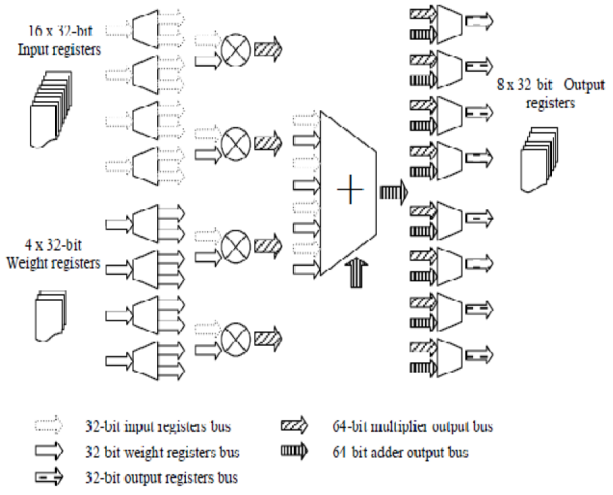

Fig.3. Block diagram of custom design.

International Journal of Electronics Communication and Computer Engineering
Volume 5, Issue 4, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Table II: ARMv6 Multiply, Multiply and Accumulate Instructions.

S.No.	Operation	Assembler
1	Multiply	MUL{cond}{S} <Rd>, <Rm>, <Rs>
2	Multiply-Accumulate	MLA{cond}{S} <Rd>, <Rm>, <Rs>, <Rn>
3	Multiply unsigned long	UMULL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
4	Multiply unsigned accumulate long	UMLAL{cond}{S} <RdLo>, <RdHi>, <Rm>, <Rs>
5	Multiply signed long	SMUL{cond}{S}<RdLo>, <RdHi>, <Rm>, <Rs>
6	Multiply signed accumulate long	SMLAL{cond}{S}<RdLo>, <RdHi>, <Rm>, <Rs>
7	Multiply 16 x 16	SMULxy{cond} <Rd>, <Rm>, <Rs>
8	Multiply-accumulate 16 x 16 + 32	SMLAxy{cond} <Rd>, <Rm>, <Rs>, <Rn>
9	Multiply 32 x 16	SMULWxy{cond} <Rd>, <Rm>, <Rs>
10	Multiply-accumulate 32 x 16 + 32	SMLAWxy{cond} <Rd>, <Rm>, <Rs>, <Rn>
11	Multiply signed accumulate long 16 x 16 + 64	SMLALxy{cond} <RdLo>, <RdHi>, <Rm>, <Rs>

Table III: ARMv6 Instruction abbreviations description.

S.No.	Abbreviation	Description
1	{cond}	Updates condition flags if cond present
2	{S}	Set condition codes
3	<RdLo>, <RdHi>	destination registers and also hold accumulating values
4	<Rd>	Destination register
5	<Rm>, <Rs>, <Rn>	Operand register

V. MIPS32 SIMD Architecture

A. Introduction

The MIPS SIMD Architecture (MSA) based on MIPS Release 5 architecture [2]. The MSA Module was implemented with strict adherence to RISC (Reduced Instruction Set Computer) design principals. The architecture allows efficient parallel processing of vector operations. The MSA operates on 32 128-bit wide vector registers [2]. The MSA architecture supports more than 150 new instructions operating on 32 vector registers of 8, 16-, 32-, and 64-bit integer, 16-and 32-bit fixed-point, or 32- and 64-bit floating-point data elements, 128-bit wide vector registers shared with the 64-bit wide floating point unit (FPU) registers [2]. The MSA operates on 32 128-bit wide vector registers. The MSA vector registers have four data formats.

B. Multiply and MAC Instructions

MSA instruction set implements the following categories of instructions: integer arithmetic, bitwise, floating-point arithmetic and non arithmetic, floating-point compare, floating point conversions, fixed-point, branch and compare, load/store and move, and element permute. In this section we will be analyzing the MSA integer instructions performing multiply, multiply and add operations.

Table IV. MSA Multiply, Multiply and Accumulate Instructions.

S.No.	Operation	Assembler
1	Multiply-Add	MADDV.B wd, ws, wt MADDV.H wd, ws, wt MADDV.W wd, ws, wt MADDV.D wd, ws, wt
2	Multiply	MULV.B wd, ws, wt MULV.H wd, ws, wt MULV.W wd, ws, wt MULV.D wd, ws, wt

Table V: MIPS SIMD Instruction abbreviations description.

S.No.	Abbreviation	Description
1	B	Byte operations
2	H	Half word operations
3	W	Word operations
4	D	Double word operations
5	Wd	Destination and accumulate register
6	Ws	Source 1 vector registers
7	Wt	Source 2 vector registers

VI. Benchmarking Custom Design

Benchmarking of custom design is performed by comparing the number of multiply and MAC instructions required implementing the n input neuron. The comparison does not take into account loading of registers and storing of registers. The n number of input requires n MAC instructions. In case of architectures supporting SIMD instructions, multiple MAC operations can be performed in one cycle. The number of MAC operations to be performed in one cycle depends upon the architecture and the source and destination data size. Table VI shows the comparison of custom design with ARMv6 and MIPS SIMD architectures in terms of number of instructions required to implement n input neuron on each of the architecture. The source size used during comparison is 32 bit signed integer value.

Table VI: Benchmarking results.

Number of neuron inputs (n)	Architecture		
	Custom Design	ARMv6	MIPS SIMD
2	1	2	1
4	1	4	1
6	2	6	2
8	2	8	2
10	3	10	3
12	3	12	3

14	4	14	4
16	4	16	4

VII. CONCLUSION

ARMv6, MSA and custom design support SIMD instructions. ARMv6 lags behind in performance comparison results in case of number of 32bit MAC instructions need to be performed in one cycle. MSA and custom design show similar results. Custom design can take advantage further, if more than 4 numbers of inputs are provided to the neural network. Sixteen inputs can be loaded in 16 input registers. MAC operations can be repeated with incremental indices of registers. Further improvements can add advantage to the custom design in terms of neural network computational performance over ARMv6 and MSA architectures.

REFERENCES

[1] S. R. Ganorkar and P. A. Jain, "Hardware Accelerator for General Purpose Artificial Neural Network," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 3, Issue 10, April, 2014.
[2] Imagination Technologies, "MIPS Architecture for Programmers," vol IV-J, Revision 1.11, April, 2014.
[3] ARM, "ARM Compiler toolchain," version 4.1, July 2011.
[4] Chandrashekhar Kalbande and Anil Bavaskar, "Implementation of FPGA-Based General Purpose Artificial Neural Network," ITSI Transactions on Electrical and Electronics Engineering (ITSI-TEEE), vol. 1, issue 3, 2013.
[5] Sandrin Ntoune Ntoune, R. and Bahoura, M., "FPGA-implementation of pipelined neural network for power amplifier modeling," New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International, pp. 109-112, June 2012.
[6] W. M. Lin, Prasanna V. K., and K. W. Przytula, "Algorithmic mapping of neural network models onto parallel SIMD machines," Computers, IEEE Transactions on, vol. 40, issue 12, Dec 1991

AUTHOR'S PROFILE

Prof. S. R. Ganorkar

joined Sinhgad Technical Education Society in September 2002. He completed M.E. (Advanced Electronics) and Ph. D. (E&TC). He is working as Professor having total experience of 25 years including 13 years of industry. Area of research is biomedical signal processing and soft computing. He got outstanding academic performance award twice. He completed BCUD research project of University of Pune on Iris recognition: An emerging biometric technology. His findings got published in 19 international journals, 16 international conferences and 41 national conferences clearly reflecting the passion towards the domain area.
In addition to teaching, he believes in playing multiple roles to bring out various facets of personality and has successfully executed various roles as coordinator for Ph.D., NBA, Purchase and Student and teacher training program. He worked as a Nodal officer (Procurement) under TEQIP-II & IEEE branch counselor.

Padmaraj A. Jain

is currently pursuing M.E. in Digital systems (Electronics Dept.) from Sinhgad College of Engineering, Pune University, Pune. He has 8 years of industry experience in field of embedded systems, low power graphics processing unit (GPU) and respective applications.