Survey of Matrix Multiplication for Digital Signal Processing Application

Rajesh Yadav[1], Prof. Satyarth Tiwari[2]
M. Tech. Scholar, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal[1]
Guide, Department of Electronics and Communication, Bhabha Engineering Research Institute, Bhopal[2]

Abstract— The fundamental function of arithmetic is Addition which is used widely in various VLSI systems such as specific application of microprocessors and DSP architectures. The key task of addition is adding the two binary numbers; it is the basis of several useful functions such as division, subtraction, multiplication, addresses calculation, etc. Matrix Multiplication (MM) is among the most customary elements in digital applications comparable to digital signal processing, control units, and photo processing on account that of its speedy and fast execution of arithmetic operations in a circuit. MM acts as the coprocessor that comprises arithmetic operations, adders, and multiplications. Speed of processor greatly depends on its multiplier as well as adder performance. In spite of complexity involved in MM operation, its implementation is increasing day by day. Due to which high speed MM architecture become important. Several MM architecture designs have been developed to decrease the delay and area.

Keywords— Matrix Multiplication, Digital Signal Processing, VLSI System

I. INTRODUCTION

We focus on the fundamental task of matrix multiplication, and use deep reinforcement learning (DRL) to search for provably correct and efficient matrix multiplication algorithms. This algorithm discovery process is particularly amenable to automation because a rich space of matrix multiplication algorithms can be formalized as low-rank decompositions of a specific three-dimensional (3D) tensor, called the matrix multiplication tensor. This space of algorithms contains the standard matrix multiplication algorithm and recursive algorithms such as Stassen's, as well as the (unknown) asymptotically optimal algorithm. Although an important body of work aims at characterizing the complexity of the asymptotically optimal algorithm, this does not yield practical algorithms [1, 2].

We focus here on practical matrix multiplication algorithms, which correspond to explicit low-rank decompositions of the matrix multiplication tensor. In contrast to two-dimensional matrices, for which efficient polynomial-time algorithms computing the rank have existed for over two centuries, finding low-rank decompositions of 3D tensors (and beyond) is NP-hard and is also hard in practice. In fact, the search space is so large that even the optimal algorithm for multiplying two 3×3 matrices is still unknown. Nevertheless, in a longstanding research effort, matrix multiplication algorithms have been discovered by attacking this tensor decomposition problem using human search, continuous optimization and combinatorial search [3].

These approaches often rely on human-designed heuristics, which are probably suboptimal. We instead use DRL to learn to recognize and generalize over patterns in tensors, and use the learned agent to predict efficient decompositions. A score is assigned based on the number of selected operations required to reach the correct multiplication result. This is a challenging game with an enormous action space (more than 1012 actions for most interesting cases) that is much larger than that of traditional board games such as chess and Go (hundreds of actions). Our framework uses a single agent to decompose matrix multiplication tensors of various sizes, yielding transfer of learned decomposition techniques across various tensors. To address the challenging nature of the game, Alpha Tensor uses specialized neural network architecture, exploits symmetries of the problem and makes use of synthetic training games [4, 5].

II. LITERATURE REVIEW

Chen Yang et al. [1], the fields of communication algorithms, digital signal processing, artificial intelligence; and so on all make extensive use of floating point operations. However, system performance and hardware overhead have been severely limited by the slow computation speed and the excessive use of resources. In order to speed up computation and save resources, floating point arithmetic units must have high area efficiency. Adder, multiplier, and reciprocal operator are among the high-performance and area-efficient floating point arithmetic units discussed in this paper. A typical communication scenario of 44 matrix inversion serves as the basis for evaluating the proposed floating point arithmetic units. Our designs improved resource overhead and performance, as demonstrated by the experiments. Our designs use only one-fourth of the computing latency of Xilinx Vivado IP and save between 20 and 45 percent of resources. Our designs improve area efficiency by 3.65 times and require only 1/4 the computing latency of Design Ware IP.

Rongyu Ding et al. [2], due to the limitations of human perception, internal operations in digital signal processing and machine learning do not require as much precision. Since its inception, approximate computing

has been viewed as an efficient means of balancing energy with precision. By using radix-8 Booth encoding on the mantissa part, a new type of approximate floating-point (FP) multiplier is proposed in this paper. In radix-8 Booth encoding, we devise the triple multiplicand addition. When compared to the IEEE-754 single precision FP multiplier, experimental results demonstrate that the proposed design can achieve significant reductions in area, delay, and power of up to 66.48%, 23.39%, and 69.02 percent, respectively, while losing only 0.18% accuracy. When applied to image smoothing and compression, the proposed multipliers exhibit minimal quality loss.

Wei Mao et al. [3], optimizing floating-point (FP) dot product units (DPUs) is becoming increasingly important for training deep learning models and high-performance scientific computing. A reconfigurable multiple-precision DPU operation has the potential to significantly reduce the cost of area and power due to the various precision requirements of applications. However, the current approaches may leave hardware resources unused for operations of varying precisions and result in redundant bits for unit multipliers. For high-performance computing (HPC) applications, a reconfigurable multiple-precision FP DPU design is proposed in this paper. The FP DPU can be changed in the following ways. For three-mode operations, a bit-partitioning technique with a programmable mixed-precision multiplier is provided to reduce the number of redundant bits: 20 operations with half-precision Dot Product (DP), 5 with single precision DP, and 1 with double precision DP. Without consuming hardware resources, any mode can be executed in two clock cycles. Using simulation results and the UMC 55-nm process, the proposed design is realized. Contrasted and the current numerous accuracy FP strategies, the proposed DPU accomplishes 88.9% and 35.8% region saving execution for FP16 and FP32 activities, individually. Furthermore, when compared to fixed FP32 and FP64 operations, the proposed reconfigurable DPU can accelerate maximum throughput rates by up to 4 and 20 percent when used in benchmarked HPC applications with multiple precisions.

Rahul Rathod et al. [4], signal processing and multimedia computations make use of floating point numbers. When compared to addition and subtraction, the process of multiplication necessitates more processing time and hardware resources. The system's execution time is determined by the multiplier processing speed because it consumes the majority of time. On VIVADO DESIGN SUITE 2018.3, Vedic multipliers, array multipliers, and CIFM multipliers are used to implement complex floating point multiplication, and their performance is compared in this paper.

S. Ross Thompson et al. [5], complete implementation that works with both normalized and denormalized numbers is also included in this paper, along with a brand-new algorithm for IEEE 754 Floating Point Multiplication. Based on injection rounding, the new rounder injects two injections into the intermediate product instead. When the product does not overflow, the first injection handles the situation, while the second injection handles the situation. To handle the two injection constants and reduce hardware duplication, a special adder is developed. The complex split between upper and lower bit paths in the single injection rounding algorithm is eliminated by dual injection rounding [1], which reduces all three key design targets; delay (1.2 percent), area (6.4 percent), and power. A standard injection rounder, Synopsys® DesignWareTM, and Cadence® ChipWareTM are compared to our novel design.

P.L. Lahari et al. [6], the less-delay-efficient multiplier and accumulator unit for inner product, filtering [3], convolution, image and video processing, and other applications are the subjects of this paper.In a digital signal processor, the multiply and accumulate unit plays an important role.On planning this consumes enormous region since it contains fractional items so Conveyed Number juggling is considered to work on the speed however for each additional information size of the ROM increments dramatically so offset twofold coding liked.The processor's overall speed will increase by using offset binary coding with floating point.Xilinx 14.7 ISE software is used to simulate and synthesize these designs.When compared to other designs, it achieves the best area and results with less delay.

Lakshmi kiran Mukkara et al. [7], for implementation of Low Power VLSI Architectures in the area of Digital Image Processing applications, Matrix Multiplication is a key arithmetic operation. To construct VLSI architectures with Low Power, High Speed and Low area, Matrix Multiplication design becomes complex. In this paper, a simple novel VLSI architecture for FPMM is presented. It is designed by considering Pseudo code for matrix multiplication, CSD multiplication algorithm for power reduction, Conventional floating point number format and Pipelining concept for improving speed. FPMM design is applicable for any arbitrary size of matrices by following matrix rules.

Soumya Havaldar et al. [8], gives an FPGA Based High Speed IEEE-754 Double Precision Floating Point Multiplier Using Verilog. This paper has implemented DPFP Multiplier using parallel Adder. A high speed floating point double precision multiplier is implemented on a Virtex-6 FPGA. In addition, the proposed design is compliant with IEEE-754 format and handles over flow, under flow, rounding and various exception conditions. The design achieved the operating frequency of 414.714 MHz with an area of 648 slices.

Ragini Parte et al. [9], IEEE point number-crunching has an immense application in DSP, advanced PCs, robots because of its capacity to speak to little numbers

and huge numbers and in addition marked numbers and unsigned numbers. Disregarding unpredictability included in gliding point number juggling, its usage is expanding step by step. Here we break down the impacts of utilizing three unique sorts of adders while figuring the single accuracy and twofold exactness skimming point increase. We additionally exhibit the increase of significand bits by disintegration of operands strategy for IEEE 754 standard.

III. DIFFERENT TYPES OF ADDER

Parallel Adder:-

Parallel adder can add all bits in parallel manner i.e. simultaneously hence increased the addition speed. In this adder multiple full adders are used to add the two corresponding bits of two binary numbers and carry bit of the previous adder. It produces sum bits and carry bit for the next stage adder. In this adder multiple carry produced by multiple adders are rippled, i.e. carry bit produced from an adder works as one of the input for the adder in its succeeding stage. Hence sometimes it is also known as Ripple Carry Adder (RCA). Generalized diagram of parallel adder is shown in figure 3.

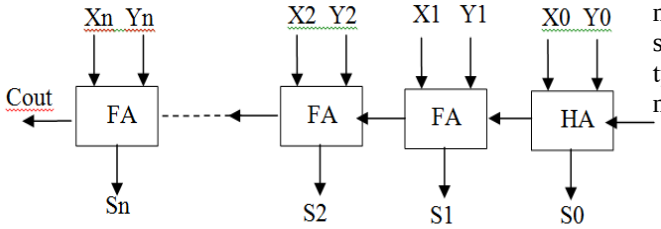

Figure 3: Parallel Adder (n=7 for SPFP and n=10 for DPFP)

An n-bit parallel adder has one half adder and n-1full adders if the last carry bit required. But in 754 multiplier's exponent adder, last carry out does not required so we can use XOR Gate instead of using the last full adder. It not only reduces the area occupied by the circuit but also reduces the delay involved in calculation. For SPFP and DPFP multiplier's exponent adder, here we Simulate 8 bit and 11 bit parallel adders respectively as show in figure 4.

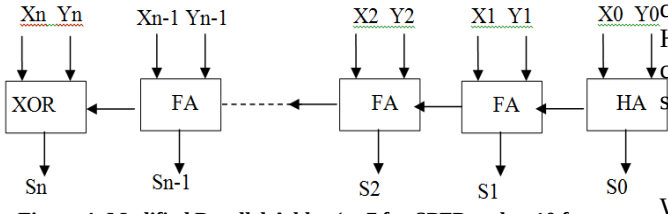

Figure 4: Modified Parallel Adder (n=7 for SPFP and n=10 for DPFP)

Carry Skip Adder:-

This adder gives the advantage of less delay over Ripple carry adder. It uses the logic of carry skip, i.e. any desired carry can skip any number of adder stages. Here carry skip logic circuitry uses two gates namely "and gate" and "or gate". Due to this fact that carry need not to ripple through each stage. It gives improved delay parameter. It is also known as Carry bypass adder. Generalized figure of Carry Skip Adder is shown in figure 5.

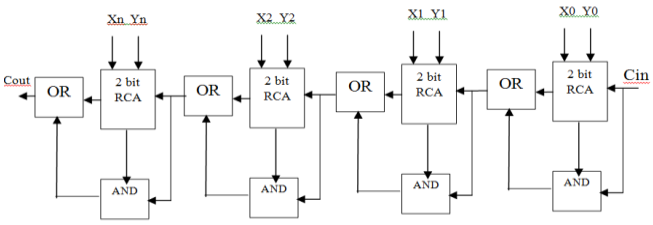

Figure 5: Carry Skip Adder

Carry Select Adder:-

Carry select adder uses multiplexer along with RCAs in which the carry is used as a select input to choose the correct output sum bits as well as carry bit. Due to this, it is called Carry select adder. In this adder two RCAs are used to calculate the sum bits simultaneously for the same bits assuming two different carry inputs i.e. '1' and '0'. It is the responsibility of multiplexer to choose correct output bits out of the two, once the correct carry input is known to it. Multiplexer delay is included in this adder. Generalized figure of Carry select adder is shown in figure 3.9. Adders are the basic building blocks of most of the ALUs (Arithmetic logic units) used in Digital signal processing and various other applications**.** Many types of adders are available in today's scenario and many more are developing day by day.

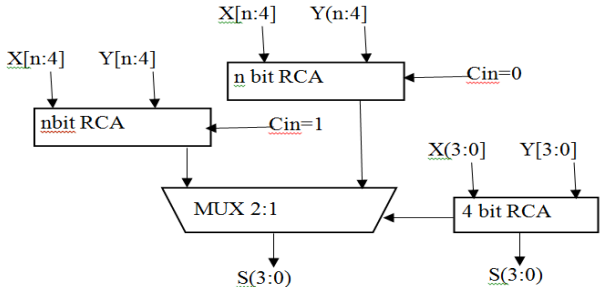

Figure 6: Carry Select Adder

Half adder and Full adder are the two basic types of adders. Almost all other adders are made with the different arrangements of these two basic adders only. Half adder is used to add two bits and produce sum and carry bits whereas full adder can add three bits simultaneously and produces sum and carry bits.

IV. CONCLUSION

When you want to solve any problem, try to choose the best method that consumes less space and time to execute efficiently. For matrix multiplication, we tested some methods Row by Row, Row By Column , Column By Column and Strassen. After our experiments we found that parallel method is the best method for implementing the matrix multiplication. For the future work, we will test many other matrix multiplication algorithms and for every algorithm we will test the space complexity.

REFRENCES

[1] Chen Yang;Siwei Xiang;Jiaxing Wang;Liyan Liang, "A High Performance and Full Utilization Hardware Implementation of Floating Point Arithmetic Units", 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS), IEEE 2021.

[2] Rongyu Ding;Yi Guo;Heming Sun;Shinji Kimura, "Energy-Efficient Approximate Floating-Point Multiplier Based on Radix-8 Booth Encoding", IEEE 14th International Conference on ASIC (ASICON), IEEE 2021.

[3] Wei Mao;Kai Li;Xinang Xie;Shirui Zhao;He Li;Hao Yu, "A Reconfigurable Multiple-Precision Floating-Point Dot Product Unit for High-Performance Computing", Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE 2021.

[4] Rahul Rathod;P Ramesh;Pratik S Zele;Annapurna K Y, "Implementation of 32-Bit Complex Floating Point Multiplier Using Vedic Multiplier, Array Multiplier and Combined integer and floating point Multiplier (CIFM)", International Conference for Innovation in Technology (INOCON), IEEE 2020.

[5] S. Ross Thompson;James E. Stine, "A Novel Rounding Algorithm for a High Performance IEEE 754 Double-Precision Floating-Point Multiplier", 38th International Conference on Computer Design (ICCD), IEEE 2020.

[6] P.L. Lahari;M. Bharathi;Yasha Jyothi M Shirur, "High Speed Floating Point Multiply Accumulate Unit using Offset Binary Coding", 7th International Conference on Smart Structures and Systems (ICSSS), IEEE 2020.

[7] Lakshmi kiran Mukkara and K.Venkata Ramanaiah, "A Simple Novel Floating Point Matrix Multiplier VLSI Architecture for Digital Image Compression Applications", 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE.

[8] Soumya Havaldar, K S Gurumurthy, "Design of Vedic IEEE 754 Floating Point Multiplier", IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016, India.

[9] Ragini Parte and Jitendra Jain, "Analysis of Effects of using Exponent Adders in IEEE- 754 Multiplier by VHDL", 2015 International Conference on Circuit, Power and Computing Technologies [ICCPCT] 978-1-4799-7074-2/15/$31.00 ©2015 IEEE.

[10] Ross Thompson and James E. Stine, "An IEEE 754 Double-Precision Floating-Point Multiplier for Denormalized and Normalized Floating-Point Numbers", International conference on IEEE 2015.

[11] M. K. Jaiswal and R. C. C. Cheung, "High Performance FPGA Implementation of Double Precision Floating Point Adder/Subtractor", in International Journal of Hybrid Information Technology, vol. 4, no. 4, (2011) October.

[12] B. Fagin and C. Renard, "Field Programmable Gate Arrays and Floating Point Arithmetic," IEEE Transactions on VLS1, vol. 2, no. 3, pp. 365-367, 1994.

[13] N. Shirazi, A. Walters, and P. Athanas, "Quantitative Analysis of Floating Point Arithmetic on FPGA Based Custom Computing Machines," Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines (FCCM"95), pp.155-162, 1995.

[14] Malik and S. -B. Ko, "A Study on the Floating-Point Adder in FPGAs", in Canadian Conference on Electrical and Computer Engineering (CCECE-06), (2006) May, pp. 86–89.

[15] D. Sangwan and M. K. Yadav, "Design and Implementation of Adder/Subtractor and Multiplication Units for Floating-Point Arithmetic", in International Journal of Electronics Engineering, (2010), pp. 197-203.

[16] L. Louca, T. A. Cook and W. H. Johnson, "Implementation of IEEE Single Precision Floating Point Addition and Multiplication on FPGAs", Proceedings of 83rd IEEE Symposium on FPGAs for Custom Computing Machines (FCCM‟96), (1996), pp. 107–116.

[17] Jaenicke and W. Luk, "Parameterized Floating-Point Arithmetic on FPGAs", Proc. of IEEE ICASSP, vol. 2, (2001), pp. 897-900.

[18] Lee and N. Burgess, "Parameterisable Floating-point Operations on FPGA", Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems, and Computers, (2002).

[19] M. Al-Ashrafy, A. Salem, W. Anis, "An Efficient Implementation of Floating Point Multiplier", Saudi International Electronics, Communications and Photonics Conference (SIECPC), (2011) April 24-26, pp. 1-5.