Survey Paper on Matrix Multiplier Module and its FPGA Implementation

Manish Kumar[1] and 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— In the present scenario, the rapid growth of wireless communication, multimedia applications, robotics and graphics increases the demand for resource efficient, high throughput and low power digital signal processing (DSP) systems. Matrix multiplication (MM) is the most widely used fundamental processing element in almost all DSP systems ranging from audio/video signal processing to wireless sensor networks. Hardware implementation of matrix multiplication requires a huge number of arithmetic operations that affect the speed and consumes more area and power. Pipelining and parallel processing are the two methods used in the DSP systems to reduce the dynamic power consumption. Demand for high performance processing element with less area and low power increases in various scientific computing applications.

Keywords: -, Matrix Multiplier, Digital Signal Processing, FPGA

I. INTRODUCTION

In many multimedia applications, fast and efficient matrix multipliers are very critical in overall operations. Matrix multiplication is an often used core operation in a wide variety of graphic, image, robotics, and signal processing applications. Traditional approaches for matrix multiplications are carried out either by software on fast processors or by hardware with multiple scalar multipliers. Software operations of the matrix multiplications can be very slow and become bottlenecks in overall system operations. On the other hand, hardware multipliers incorporate high-speed logic, but may be very costly in terms of complexity and power consumption. Moreover, these multipliers are usually designed for a specific word length and often lack flexibility where the size of scalar multipliers must correspond to the input of the matrix multiplier. The objective of this paper is to propose a flexible and energy efficient matrix multiplier, which can be extended to reconfigurable high speed processing implementation. A key concept is to apply the divide-and-conquer approach by decomposition, both at matrix level and at word level. Manuscript Received on December 2012. R.L.Bhargavi, Lecturer's in Einstein College of Engineering, Tirunelveli (Tamil Nadu), India. M.Merlin Moses, Lecturer's in Einstein College of Engineering, Tirunelveli (Tamil Nadu), India. V.Karthikeyan, Lecturer's in Einstein College of Engineering, Tirunelveli (Tamil Nadu), India. C.Karthikeyan, Assistant Professor in Einstein College of Engineering, (Tamil Nadu), India. Many studies have been done on matrix multiplications. Most notably, [1] introduced an efficient matrix multiplication algorithm for linear arrays with reconfigurable pipelined bus systems. This architecture also follows the divide-and-conquer approach for high-speed parallel processing. However, their decomposition is limited to the matrix level. A matrix multiplier for sparse matrices that reduces the input/output (I/O) and the number of trivial multiplications is introduced in [2]. This architecture is focused on the data elements of the matrices. However, the architecture is highly beneficial only when the matrices are sparse. In this paper, the proposed architecture also reduces computation energy by exploiting zeros during actual computations. Most of the previous implementations of matrix multiplication on field-programmable gate arrays (FPGAs) are focused mainly on tradeoff between area and maximum running frequency [3]–[7]. A bit-serial matrix multiplier using Booth encoding was implemented on the FPGA [3]. The multiplier discussed in [4] improved the design in [3] using modified Booth-encoder multiplication along with Wallace tree addition, where the running frequency has been doubled from the design described in [3]. These designs are further improved in terms of area and speed in [7]. All of their designs consider the input word-width directly and the size of scalar multipliers can be significantly large for practical VLSI implementation when the input word-width increases. Moreover, their designs are mainly focused on reducing FPGA resources by incorporating static memory in the implementation. This design is focused on both computational and energy efficiency, as well as structural flexibility.

II. LITERATURE REVIEW

Ankit Gupta et al. [1], approximate computing has emerged as a new paradigm for the energy-efficient design of circuits and systems. It enables highly efficient hardware and software implementations by exploiting the inherent resilience of applications to in-exactness in their computations. In this work, hardware implementation of Matrix Multiplier based on approximate computing is modeled in VERILOG Hardware Description Language (HDL). The target device used for synthesis is xc7a100t-3csg324 in Xilinx. Simulations were performed in ISIM simulator and device utilization has been presented below.

M Shanmugakumar et al. [2], matrix Multiplication features in many engineering and scientific problems, the reason which

working on efficient algorithms and architectures to perform matrix multiplication is still relevant. In this work, we bring forward an efficient algorithm that is targeted towards the hardware implementation for generating a matrix multiplier targeted according to the input parameters of the user. The proposed architecture utilizes a carry-save adder tree multiplier for multiplication and carry-lookahead adder for performing addition at the final stage. When compared with state-of-the-art matrix multiplication algorithms like Strassen and Winograd, our architecture achieves a better energy efficiency of 59% and 69% respectively, and area delay product (cycle budget) of 58% and 73% respectively. The proposed architecture is well suited for power critical applications with a negligible impact on the delay.

Chiranjit R Patel et al. [3], a simple and effective method for matrices multiplication is proposed. The determination of the resultant output matrix can either performed in parallel or sequentially, both resulting in the same output. The selection of the algorithm is application-specific and filters down to the frequency of operation or power consumption. Multipliers and adders are the main hardware blocks in a matrix multiplication system. Hence, choosing a multiplier which is not only fast but also consumes lesser power and area is vital. This study also proposes a modified 2-bit and 4-bit multiplier architectures based on Vedic mathematics, which is proven to be more efficient than the standard architectures. The simulations are performed using Cadence Virtuoso 45nm CMOS technology. The proposed 7T SRAM cell makes use separate write and reads ports which are controlled by Read Word Line (RWL) and Write Word Line (WWL). Specter simulations are performed for voltage ranges from 0.8V to 1.5V. The simulations also show that the matrix multiplier of 2-bit and 4-bit elements can operate at 2GHz and 0.7GHz at 1.2V respectively, and consumes an average of 140 μW and 350 μW. The layouts designed in 45nm are found to be matching with the schematic (LVS clean) and follows all Foundry rules (DRC clean).

Y. Huang et al. [4], large-scale floating-point matrix multiplication is widely used in many scientific and engineering applications. Most existing works focus on designing a linear array architecture for accelerating matrix multiplication on FPGAs. This paper towards the extension of this architecture by proposing a scalable and highly configurable multi-array architecture. In addition, we present a work-stealing scheme to ensure the equality in the workload partition among multiple linear arrays. Furthermore, an analytical model is developed to determine the optimal parameters for matrix multiplication acceleration. Experiments on real-life convolutional neural networks (CNNs) show that we can obtain the optimal extension of the linear array architecture.

I. Sayahi et al. [5], in diverse domains, the scientific applications requires severe computing algebra routines. The matrix multiplication presents an indispensable mathematical operation in many high performance fields. This paper presents a new FPGA design and implementation for matrix vector multiplication. The design has been implemented with Xilinx System Generator. The results of FPGA implementation were compared with similar work on VIRTEX 4 platform. It demonstrates the efficiency of our work in term of resources utilization and speed up.

W. Liu et al. [6], sparse matrix-vector multiplication (SpMV) is a fundamental building block for numerous applications. In this paper, we propose CSR5 (Compressed Sparse Row 5), a new storage format, which offers high-throughput SpMV on various platforms including CPUs, GPUs and Xeon Phi. First, the CSR5 format is insensitive to the sparsity structure of the input matrix. Thus the single format can support an SpMV algorithm that is efficient both for regular matrices and for irregular matrices. Furthermore, we show that the overhead of the format conversion from the CSR to the CSR5 can be as low as the cost of a few SpMV operations. We compare the CSR5-based SpMV algorithm with 11 state-of-the-art formats and algorithms on four mainstream processors using 14 regular and 10 irregular matrices as a benchmark suite. For the 14 regular matrices in the suite, we achieve comparable or better performance over the previous work. For the 10 irregular matrices, the CSR5 obtains average performance improvement of 17.6\%, 28.5\%, 173.0\% and 293.3\% (up to 213.3\%, 153.6\%, 405.1\% and 943.3\%) over the best existing work on dual-socket Intel CPUs, an nVidia GPU, an AMD GPU and an Intel Xeon Phi, respectively. For real-world applications such as a solver with only tens of iterations, the CSR5 format can be more practical because of its low-overhead for format conversion.

Usha Maddipati et al. [7], concentrating on the real time demands of digital signal processing, a delay and power efficient 16-tap direct form low pass FIR filter is realized using FPGA. The filter coefficients are generated using Kaiser Window function of MATLAB FDA tool. For obtaining the high speed operation at reasonable power, various adder architectures are considered for the filter design along with vedic multiplier. The designs were implemented on Artix-7 xc7a100tcsg324-1 FPGA board and debugged using Virtual Input/output IP of Xilinx Vivado to validate the results. Experimental results show that efficiency in power-delay product can be obtained by using Carry Increment Adder for FIR filter design than that of various other multi-bit adder structures.

Ankit Upadhyay et al. [8], the execution of FIR channels on FPGA taking into account conventional technique costs significant equipment assets, which conflicts with the diminishing of circuit scale and increment of framework pace. FIR channels utilizing Arithmetic is utilized to build the asset use while pipeline structure is additionally used to expand the framework speed. Moreover, the isolated LUT strategy is additionally used to diminish the required memory units. FIR filter implemented using basic Arithmetic architecture is based on bit serial operation resulting in increase in delay with decrease in speed of operation. This is because the entire co-

efficient are stored in single LUT. In Parallel DA architecture, instead of storing the co-efficient in single LUT as in traditional Arithmetic architecture, it is split into several ROM LUT's. All the LUT's are provided with different inputs at the same time, implying parallel mechanism. This increases the speed of operation.

Basant Kumar Mohanty et al. [9], transpose form finite-impulse response (FIR) filters are inherently pipelined and support multiple constant multiplications (MCM) technique that results in significant saving of computation. However, transpose form configuration does not directly support the block processing unlike direct-form configuration. In this paper, we explore the possibility of realization of block FIR filter in transpose form configuration for area-delay efficient realization of large order FIR filters for both fixed and reconfigurable applications. Based on a detailed computational analysis of transpose form configuration of FIR filter, we have derived a flow graph for transpose form block FIR filter with optimized register complexity. A generalized block formulation is presented for transpose form FIR filter. We have derived a general multiplier-based architecture for the proposed transpose form block filter for reconfigurable applications. A low-complexity design using the MCM scheme is also presented for the block implementation of fixed FIR filters. The proposed structure involves significantly less area-delay product (ADP) and less energy per sample (EPS) than the existing block implementation of direct-form structure for medium or large filter lengths, while for the short-length filters, the block implementation of direct-form FIR structure has less ADP and less EPS than the proposed structure. Application-specific integrated circuit synthesis result shows that the proposed structure for block size 4 and filter length 64 involves 42% less ADP and 40% less EPS than the best available FIR filter structure proposed for reconfigurable applications. For the same filter length and the same block size, the proposed structure involves 13% less ADP and 12.8% less EPS than those of the existing direct-form block FIR structure.

III. MATRIX MULTIPLIER

From the earlier reported works in this field, the major power consuming resource was found to be multipliers and the registers, used to store and move the intermediate data. So, we have proposed three designs which reduce as well as optimize the number of multipliers and registers being used in the matrix multiplication operation [11]. For the ease of recognition we have named the designs on the basis of input and output dataflow.

Let us consider the matrix – matrix multiplication for two n×n matrices A and B given by-

$$C\begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix} = A\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \times B\begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \quad (1)$$

Such that,

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \times b_{kj} \quad (2)$$

for all i, *j*, $a_{ik,} b_{kj}$ and c_{ij} represent elements of the n×n matrices A, B and C.

Power Reduction Techniques

The design flow of a system constitutes of various levels of abstraction. When a system is designed with the emphasis on power optimization as a performance goal, then the design must embody optimization at all levels of the design flow. In general there are three levels on which energy reduction can be incorporated - system level, architecture level and technological level. For example, at the system level inactive modules may be turned off to save power.

At the architectural level, parallel hardware may be used to reduce global interconnect and allow a reduction in supply voltage without degrading system throughput. At the technological level several optimizations can be applied at the gate level. The system and architecture have to be designed to target the possible reduction of energy consumption at the gate level. An important aspect of the design flow is the relation and feedback between the levels. Various design factors have been exploited to reduce the dynamic power of the circuit.

Clock Gating

Clock-gating is the most widely adopted technique for reducing dynamic power in digital CMOS circuits. The reasons for this are rooted, on the one hand, into the capability of clock-gating of significantly reducing power consumption with a limited penalty in area and, most important, in timing. On the other hand, clock-gating is very suitable to automatic application. Clock gating is a method where certain parts of the processor are prevented from receiving the clock signal. If a part of the processor is not needed for a given operation, then the clock signal to that part can be stopped. Since switching requires power and in the absence of the clock signal no switching will take place, gating the clock will lower power need [13].

The alternative approach to reducing wasteful activity is applying an asynchronous design methodology. CMOS is a good technology for low-power as gates only dissipate energy when they are switching. However, many gates switch because they are connected to the clock, not because they have new inputs to process. As a result, a synchronous circuit wastes power when particular blocks of logic are not utilized.

Adding clock gating may not always be accompanied by reduced power because dynamic power is also a function of clock frequency, voltage and capacitance. Even though it is not an absolute indication of power, it is a good metric for hardware designers to gain visibility into energy efficiency at RTL without time-consuming power analysis or synthesis.

Multi Threshold Voltage Cells

During logic synthesis, the design is mapped to technology gates. At this point in the process optimal logic architectures are selected, mapped to technology cells, and optimized for specific design goals. Since a range of libraries are now available and choices have to be made across architectures with different cells, logic synthesis is the ideal place to start deploying a mix of different cells into the design.

Multiple libraries are currently available with different performance, area and power utilization characteristics, and synthesis optimization can be achieved using either one or more libraries concurrently. In a single-pass flow, multiple libraries can be loaded into synthesis tool prior to synthesis optimization. In a two-pass flow, the design is initially optimized using one library, and then an incremental optimization is carried out using additional libraries.

Power Gating

It is well know that the advent of sub-100nm technologies has made the power optimization problem more difficult to address, as dynamic consumption is now paired by a non-negligible amount of static consumption, mainly due to the sub-threshold currents in the off state. Among the several solutions for reducing the sub-threshold leakage currents, power-gating is the one that, in the near past, has gained the largest momentum. As for the case of clock-gating, also power gating, or power switch-off technique, usually refers to placing switches on-chip to selectively turn off current supply based on the application requirement.

Designers are employing two types of power gating - fine-grain power gating and coarse-grain power gating. Fine-grain power gating encapsulates the switching transistor as a part of the standard cell logic. Usually these cell designs conform to the normal standard cell rules and can easily be handled by EDA tools for implementation. The size of the gate control is designed with the worst case consideration that this circuit will switch during every clock cycle resulting in a huge area impact. This is the only way to size, as the module level function is not known at this time. This is the biggest disadvantage of this method.

IV. FPGA

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

ASIC and FPGAs have different value propositions, and they must be carefully evaluated before choosing any one over the other. Information abounds that compares the two technologies. While FPGAs used to be selected for lower speed/complexity/volume designs in the past, today's FPGAs easily push the 500 MHz performance barrier. With unprecedented logic density increases and a host of other features, such as embedded processors, DSP blocks, clocking, and high-speed serial at ever lower price points, FPGAs are a compelling proposition for almost any type of design.

V. CONCLUSION

Higher order multipliers are required in image processing applications. In this paper Efficient FPGA based Matrix Multiplication using Mux and Vedic multiplier is proposed. The lower order MUX based multipliers are used with vedic multipliers to design a novel higher order multiplier of any dimensions. The proposed multiplier is used in image processing applications. It is observed that the performance parameters such as area and delay are reduced compared to existing algorithms. In future, multiplier can be designed using higher order MUX based multipliers.

REFERENCES

[1] Ankit Gupta and Ankit Gupta, "Hardware Design of Approximate Matrix Multiplier based on FPGA in Verilog", International Conference on Intelligent Computing and Control Systems (ICICCS 2020).

[2] M Shanmugakumar, Vegesna S. M. Srinivasavarma and Sk Noor Mahammad, "Energy Efficient Hardware Architecture for Matrix Multiplication", IEEE 4th Conference on Information & Communication Technology (CICT), IEEE 2018.

[3] Chiranjit R Patel, Vivek Urankar, Vivek B A and Sampath Kumar R, "2x2 Matrix Multiplication with 4-Bit elements in 45nm CMOS Technology", 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE 2020.

[4] Y. Huang J. Shen Y. Qiao M. Wen and C. Zhang "Malmm: A multi-array architecture for large-scale matrix multiplication on fpga" IEICE Electronics Express vol. 15 no. 10 pp. 20 180 286-20 180 286 2018.

[5] I. Sayahi M. Machhout and R. Tourki "Fpga implementation of matrix-vector multiplication using xilinx system generator" 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET) pp. 290-295 March 2018.

[6] W. Liu and B. Vinter "Csr5: An efficient storage format for cross-platform sparse matrix-vector multiplication" Proceedings of the 29th ACM on International Conference on Supercomputing 2015.

[7] Usha Maddipati, Shaik Ahemedali, Maddipati Sri Sai Ramya, M D Praneeth Reddy and K N J Priya, "Comparative analysis of 16-tap FIR filter design using different adders", ICCCNT, IEEE 2020.

[8] Ankit Upadhyay and Prof. Uday Panwar, "High Performance VLSI Architecture for Transpose Form FIR Filter using Integrated Module", International Conference on Computer Communication and Informatics (ICCCI), IEEE 2018.

[9] Basant Kumar Mohanty, and Pramod Kumar Meher, High-Performance FIR Filter Architecture for Fixed and Reconfigurable Applications", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 78, No.06, April 2016.

[10] Indranil Hatai, Indrajit Chakrabarti, and Swapna Banerjee, "An Efficient VLSI Architecture of a Reconfigurable Pulse-Shaping FIR Interpolation Filter for Multi-standard DUC",

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 23, No. 6, June 2015.

[11] Sang Yoon Park and Pramod Kumar Meher, "Efficient FPGA and ASIC Realizations of DA-Based Reconfigurable FIR Digital Filter", IEEE Transactions on Circuits And Systems-Ii: Express Briefs, 2014.

[12] B. K. Mohanty, P. K. Meher, S. Al-Maadeed, and A. Amira, "Memory footprint reduction for power-efficient realization of 2-D finite impulse response filters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 120–133, Jan. 2014.

[13] B. K. Mohanty and P. K. Meher, "A high-performance energy-efficient architecture for FIR adaptive filter based on new distributed arithmetic formulation of block LMS algorithm," *IEEE Trans. Signal Process.*, vol. 61, no. 4, pp. 921–932, Feb. 2013.

[14] G. Gokhale and P. D. Bahirgonde, "Design of Vedic Multiplier using Area-Efficient Carry Select Adder", 4th IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI-2015), Kochi, August 10-13, 2015, India.

[15] G. Gokhale and Mr. S. R. Gokhale, "Design of Area and Delay Efficient Vedic Multiplier Using Carry Select Adder", 4th IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI-2015), Kochi, August 10-13, 2015, India.

[16] Pavan Kumar, Saiprasad Goud A, and A Radhika had published their research with the title "FPGA Implementation of high speed 8-bit Vedic multiplier using barrel shifter", 978-1-4673-6150-7/13 IEEE.

[17] B. Madhu Latha1, B. Nageswar Rao, published their research with title "Design and Implementation of High Speed 8-Bit Vedic Multiplier on FPGA" International Journal of Advanced Research in Electrical ,Electronics and Instrumentation Engineering, Vol. 3, Issue 8, August 2014.

[18] A Murali, G Vijaya Padma, T Saritha, published their research with title "An Optimized Implementation of Vedic Multiplier Using Barrel Shifter in FPGA Technology", Journal of Innovative Engineering 2014, 2(2).