Survey of Adjoint Matrix for FPGA Implementation the Triangular Matrix

Rohit Sachan[1], Prof. Suresh. S. Gawande[2], Prof. Satyarth Tiwari[3]

M. Tech. Scholar, Department of Electronics and Communication, Rkdf college of engineering, Bhabha University, Bhopal[1]

Guide, Department of Electronics and Communication, Rkdf college of engineering, Bhabha University, Bhopal[2]

Co-guide, Department of Electronics and Communication, Rkdf college of engineering, Bhabha University, Bhopal[3]

Abstract— Due to advancement of new technology in the field of VLSI and Embedded system, there is an increasing demand of high speed and low power consumption processor. Speed of processor greatly depends on its multiplier as well as adder performance. Employing the more characteristics of both the triangular matrix and its inversion, the proposed diagonal-wise algorithm for the triangular matrix inver- sion has the high parallelism and extensibility in the hardware implementation and is suitable for the different matrix triangular factorization (QR, LDL, Cholesky and LU). Meanwhile, the recursive diagonal-wise algorithm is designed for the large scale triangular matrices. Compared with the traditional row-/column- wise methods, our algorithm has a good performance at the low computation load. In this paper, we introduce an architecture that studied of matrix multiplication using booth multiplier and different types of adder.

Keywords— Adjoint Matrix, FPGA Implementation, Booth Multiplier

I. INTRODUCTION

Matrix multiplication is commonly used in most signal processing algorithms. It is also a frequently used kernel operation in a wide variety of graphics, image processing as well as robotic applications. The matrix multiplication operation involves a large number of multiplication as well as accumulation. Multipliers have large area, longer latency and consume considerable power compared to adders. Registers, which are required to store the intermediate product values, are also major power intensive component [1]. These components pose a major challenge for designing VLSI structures for large-order matrix multipliers with optimized speed and chip-area. However, area, speed and power are usually conflicting hardware constraints such that improving upon one factor degrades the other two [2].

II. LITERATURE REVIEW

Di Yan et al. [1], due to the robustness, the matrix inversion methods based on matrix factorization are often adopted in engineering while the triangular matrix inversion is one key part of those methods. This brief reviews the adjoint matrix and presents a novel scheme for field-programmable gate arrays (FPGAs) calculating the triangular matrix inversion. Employing the more characteristics of both the triangular matrix and its inversion, the proposed diagonal-wise algorithm for the triangular matrix inversion has the high parallelism and extensibility in the hardware implementation and is suitable for the different matrix triangular factorization (QR, LDL, Cholesky and LU). Meanwhile, the recursive diagonal-wise algorithm is designed for the large scale triangular matrices. Compared with the traditional row-/column- wise methods, our algorithm has a good performance at the low computation load. Finally, the experiments are conducted on one Xilinx Virtex-7 FPGA to illustrate the performances of the four different methods.

X.-W. Zhang et al. [2], in control engineering, numerical stability and real time are the two most important requirements for the matrix inversion. This article presents an efficient and robust method for the field-programmable gate array (FPGA) calculation of the matrix inversion. We initially consider the scenario that the matrix to be processed is a nonsingular Hermitian matrix. The proposed computation procedures are composed of the matrix decomposition, triangular matrix inversion, and matrices multiplication. The first procedure is completed by LDL factorization based on the outer form of Cholesky's method, whereas the recursive algorithm for block submatrices is adopted to achieve the triangular matrix inversion. The new method has the high level in the parallel pipelining mechanism and steals the characteristics of both the upper triangular matrix and its inversion to reduce the computation load and improve the numerical stability. Furthermore, the non-Hermitian matrix inversion can be easily solved if another procedure is added in the new method. Finally, we compare our method with the exiting FPGA-based techniques on one Xilinx Virtex-7 XC7VX690T FPGA. Meanwhile, it has solved one array antenna control problem of the adaptive digital beam forming for one phased array radar successfully.

C. Zhang et al. [3], in massive multiple-input and multiple-output (M-MIMO) systems, one of the key challenges in the implementation is the large-scale matrix inversion operation, as widely used in channel estimation, equalization, detection, and decoding procedures. Traditionally, to handle this complexity issue, several low-complexity matrix inversion approximation methods have been proposed, including the classic Cholesky decomposition and the Neumann series expansion (NSE). However, the conventional approaches failed to exploit neither the special structure of channel matrices nor the critical issues in the hardware implementation, which results in poorer throughput performance and longer processing delay. In this paper, by targeting at the correlated M-MIMO systems, we propose a modified NSE based on tridiagonal matrix inversion approximation (TMA) to accommodate the complexity as well as the performance issue in the conventional hardware implementation, and analyze the corresponding approximation errors. Meanwhile, we investigate the very-large-scale integration implementation for the proposed detection algorithm based on a Xilinx Virtex-7 XC7VX690T FPGA platform. It is shown that for correlated massive MIMO systems, it can achieve near minimum mean square error performance and 630 Mb/s throughput. Compared with other benchmark systems, the proposed pipelined TMA detector can get high throughput-to-hardware ratio. Finally, we also propose a fast iteration structure for further research.

Y.-W. Xu et al. [4], cheap, efficient, and industrial-grade model predictive controllers are required for extending the use of model predictive control (MPC) to resource-constrained embedded computing platforms and practical industrial fields. In this paper, we implement the matrix inversion on the hardware to improve the computational efficiency of our previously designed MPC controller. For the specific matrices in each iteration of the active-set method, a simple positive definite symmetric matrix inversion algorithm is proposed, which can transform the matrix inversion into iterated matrix multiplication and add operations along with fewer division operations. Its similar matrix calculating formulas in each iteration are block design oriented and able to be speed up by the parallelism and pipeline in the hardware. To avoid extra data transmission, the matrix inversion algorithm and active-set method are combined through the time sequence and architectural design to form an improved predictive controller on a field-programmable gate array (FPGA). The improved predictive controller is implemented on an FPGA-based board especially made for the industrial sites and applied to a holding process of the injection molding, which achieves satisfactory control performance.

Lakshmi kiran Mukkara et al. [5], 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. [6], 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. [7], 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.

Ross Thompson et al. [8], IEEE-754 determines trade and number juggling positions also, routines for paired and decimal drifting point number juggling in PC programming world. The execution of a skimming point framework utilizing this standard should possible completely in programming, or in equipment, or in any blend of programming and equipment. This venture propose VHDL execution of IEEE - 754 Floating point unit .In proposed work the pack, unload and adjusting mode was actualized utilizing the VHDL dialect and reenactment was checked.

In this proposition work, DPFP Multiplier alongside SPFP Multiplier has been actualized with four ordinary Adders (PA, CSKA, CSA, and CSABEC). Think about their Results and CSA is known not the speediest snake among every single customary viper. Be that as it may, CSA involves more territory as it has two parallel circuits to include the same bits yet with diverse convey data. As CSA figures the whole without sitting tight for the transitional conveys to spread stage by stage. Finally it is the obligation of multiplexer to pick and give the last right yield. CSABEC is an adjusted adaptation of CSA in which one of the parallel circuits is supplanted by the arrangement of Binary to Excess-1 Converters circuit (BECs). It is turned out to be an awesome way to deal with decrease the territory.

M. K. Jaiswal et al. [9], we display a nonintrusive simultaneous mistake identification (CED) technique for ensuring the control rationale of a contemporary coasting point unit (FPU). The proposed strategy depends on the perception that control rationale mistakes lead to broad information way defilement and influence, with high likelihood, the example some portion of the IEEE-754 coasting point representation. Consequently, type observing can be used to recognize blunders in the control rationale of the FPU. Anticipating the type includes generally basic operations; in this way, our system causes fundamentally bring down overhead than the traditional methodology of copying the control rationale of the FPU. In fact, trial results on the open SPARC T1 processor utilizing SPEC2006FP benchmarks demonstrate that when contrasted with control rationale duplication, which brings about a zone overhead of 17.9 percent of the FPU size, our technique acquires a region overhead of just 5.8 percent yet still accomplishes discovery of more than 93 percent of transient mistakes in the FPU control rationale. Besides, the proposed technique offers the subordinate advantage of additionally distinguishing 98.1 percent of the information way mistakes that influence the type, which can't be recognized by means of duplication of control rationale. At long last, when joined with a traditional deposit code-based strategy for the portion, our system prompts a complete CED answer for the whole FPU which gives scope of 94.1 percent of all blunders at a region expense of 16.32 percent of the FPU size.

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.
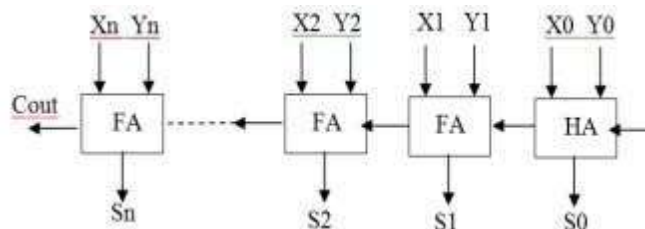

Figure 1: Parallel Adder (n=7 for SPFP and n=10 for DPFP)

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.
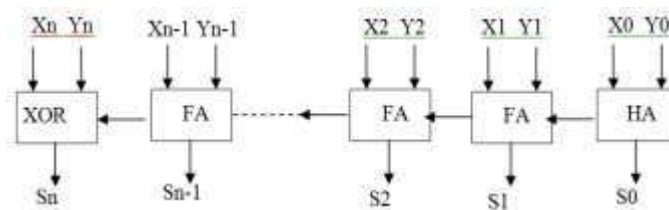

Figure 2: Modified Parallel Adder (n=7 for SPFP and n=10 for DPFP)

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.
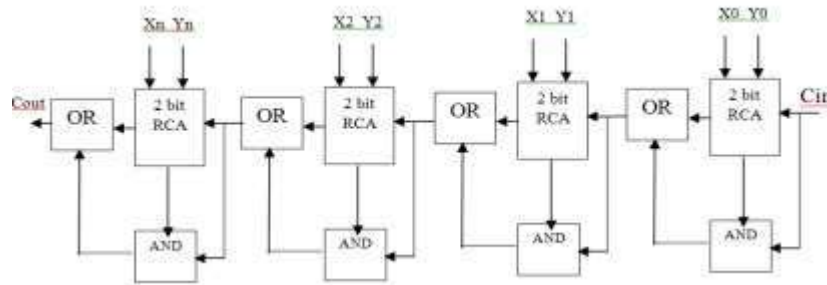
Figure 3: Carry Skip Adder

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 4. 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.
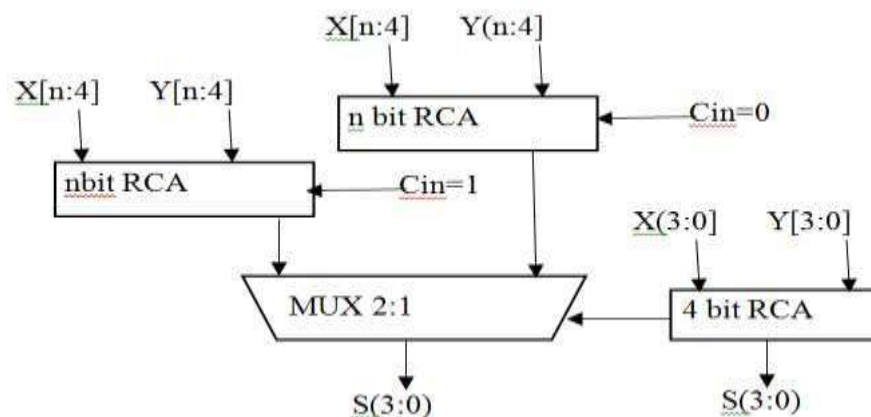


Figure 4: Carry Select Adder

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. PROPOSED DESIGN

In IEEE754 standard floating point representation, 8 bit Exponent field in single precision floating point (SP FP) representation and 11 bit in double precision floating point (DP FP) representation are need to add with another 8 bit exponent and 11 bit exponent respectively, in order to multiply floating point numbers represented in IEEE 754 standard as explained earlier. Ragini et al. [10] has used parallel adder for adding exponent bits in floating point multiplication algorithm. We proposed the use of 8-bit modified CSA with dual RCA and 8-bit modified CSA with RCA and BEC for adding the exponent bits. We have found the improved area of 8-bit modified Carry select adder with RCA and BEC over the 8-bit modified CSA with dual RCA.

o   Sign bit calculation

To calculate the sign bit of the resultant product for SP FP and DP FP multiplier, the same strategy will work. We just need to XOR together the sign bits of both the operands. If the resultant bit is „1", then the final product will be a negative number. If the resultant bit is „0", then the final product will be a positive number.
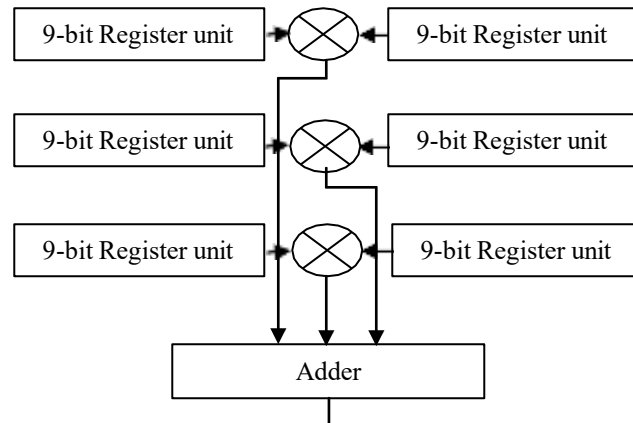
Figure 5: Proposed PPI – SO Design for n = 3

Output

o   Exponent bit calculation

Add the exponent bits of both the operands together, and then the bias value (127 for SPFP and 1023 for DPFP) is subtracted from the result of addition. This result may not be the exponent bits of the final product. After the significand multiplication, normalization has to be done for it. According to the normalized value, exponents need to be adjusted. The adjusted exponent will be the exponent bits of the final product.

o   Significand bit calculation

Significand bits including the one hidden bit are need to be multiply, but the problem is the length of the operands. Number of bits of the operand will become 24 bits in case of SP FP representation and it will be 53 bits in case of DP FP representation, which will result the 48 bits and 106 bits product value respectively. In this paper we use the technique of break up the operands into different groups then multiply them. We get many product terms, add them together carefully by shifting them according to which part of one operand is multiplied by which part of the other operand. We have decomposed the significand bits of both the operands ain four groups. Multiply each group of one operand by each group of second operand.

Matrix Multiplication

In this design we have reduced the resource utilization in terms of number of multipliers and registers in lieu of the completion time.

This design is particularly useful where resources are limited and design can be compromised on basis of increased completion time. The basic working model for a 3 × 3 matrix-matrix multiplication is shown in figure 7 below.

Considering the matrix – matrix multiplication of two n×n matrices, the calculation is performed using n number of multipliers, n number of registers and n-1 number of adders. n^2 cycles are required to perform the matrix multiplication operation. Each multiplier has two input ports: one each from matrix A and B. In each cycle, n numbers of multiplications are performed and the products are fed to the adder block to give a single element of the output matrix, C. The data flow to the multipliers are such that,

k^{th} multiplier is fed from k^{th} column of matrix A and k^{th} row of matrix B, where $1 < k < n$. At the kth multiplier, each element from matrix A is repeated for n consecutive cycles whereas the elements from matrix B are cycled back after n cycles. The partial products are then fed to the adder which computes the final result.

For a better understanding of the process, let us consider the matrix multiplication for n = 3 (as shown in figure 1). In this case, 3 multipliers and 3 registers are used to calculate and store the partial products respectively. These partial products are then fed to the adder block to compute the final result. The first multiplier receives input from the first column of matrix A (a_{k1}) and first row of matrix B (b_{1k}), where. Each element of the matrix A at the first multiplier is repeated for 3 cycles, such that the data flow can be represented as $a_{11}a_{11}a_{11}\ a_{21}a_{21}a_{21}\ a_{31}a_{31}a_{31}$.Similarly, at the first multiplier, the elements of B are repeated after 3 cycles, such that the input data-flow will be $b_{11}b_{12}b_{13}\ b_{11}b_{12}b_{13}\ b_{11}b_{12}b_{13}$. The other two multipliers receive the component of A and B in the similar order as the first multiplier. After the multiplication, the partial products are fed to the adder which computes the elements of output matrix C in row major order given by $c_{11}c_{12}\ c_{13}\ c_{21}c_{22}\ c_{23}\ c_{31}c_{32}\ c_{33}$. So the entire matrix multiplication operation is performed in n^2 =9 cycles.

V. CONCLUSION

IEEE754 standardize two basic formats for representing floating point numbers namely, single precision floating point and double precision floating point. Floating point arithmetic has vast applications in many areas like robotics and DSP. Delay provided and area required by hardware are the two key factors which are need to be consider Here we present single precision floating point multiplier by using two different adders namely modified CSA with dual RCA and modified CSA with RCA and BEC.

Among all two adders, modified CSA with RCA and BEC is the least amount of Maximum combinational path delay (MCDP). Also, it takes least number of slices i.e. occupy least area among all two adders.

REFRENCES

[1] Di Yan, Wei-Xing Wang, Lei Zuo, *Member, IEEE* and Xiao-Wei Zhang, "Revisiting the Adjoint Matrix for FPGA Calculating the Triangular Matrix Inversion", IEEE Transactions on Circuits and Systems II: Express Briefs, 2020.

[2] X.-W. Zhang, L. Zuo, M. Li and J.-X. Guo, "High-throughput FPGA implementation of Matrix inversion for control systems," Accepted by *IEEE Trans. Ind. Electron.*, 2020.

[3] C. Zhang, et al,. "On the low-complexity, hardware-friendly tridiagonal matrix inversion for correlated massive MIMO systems," *IEEE Trans. Vehic. Tech.*, vol. 68, no. 7, pp. 6272-6285, Jul. 2019.

[4] Y.-W. Xu, Y. Xi, J. Lan and T.-F. Jiang, "An improved predictive controller on the FPGA by hardware matrix inversion," *IEEE Trans. Ind. Electron.*, vol. 65, no. 9, pp. 7395–7405, Sep. 2018.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.