Error-Free Computation of Inverse Matrices in FPGA

Anatolij Sergiyenko, Volodymir Lepekha

National Technical University of Ukraine, pr. Peremogy, 37, 03056 Kiev, Ukraine

aser@comsys.kpi.ua

Abstract. *An algorithm for computing the determinant of integer matrices based on the Givens method using rational fraction numbers is considered. The algorithm is implemented in the processor array in FPGA to calculate the inverse matrices. Due to the pipelining, and the synthesis using the resynchronization approach, the processor has a high clock frequency for the integer data with the bit width up to hundreds of bits. The processor can be used in adaptive filtering, pattern recognition, computational geometry, cryptoanalysis, etc.*

Keywords

FPGA, linear algebra, Givens, pipelining.

1 Introduction

Field programmable gate array (FPGA) provide large hardware resources for implementation of many computation intensive algorithms. When the FPGA capacity reached about one million gates, it was evident that FPGA can be used for linear algebra (LA) problem solving. Since FPGAs of the first generation had no hardware multipliers, the algorithms with minimized multiply operations were preferable for configuring in them. Therefore, many LA problem solvers based on FPGA utilize the CORDIC algorithm till now [1]. When DSP units were introduced in most of FPGAs the number of processor projects for matrix computations has increased dramatically. Then such processors have to compete with microprocessor-based systems [2,3]. Modern FPGAs can effectively implement the LA double-precision floating-point computations as well as the general purpose processor can [4].

Error-free LA problem solving is used for ill conditioned matrices, integer linear programming, as well as in many other cases [5]. Only two general approaches provide the error-free LA computations. We can work with long integers - from algorithmic point of view it is many times very cumbersome. We can work with integers in finite rings using residue class or modulo arithmetic and at the end of the calculation to convert the modulo representation into rational numbers. The software systems, which provide the error-free LA computations, like Mapple, utilize the second approach [5,6,7]. But the conversion of the modulo representation into rational numbers, as well as comparing such numbers is too complex.

The advantage of FPGA is that the hardware computations of any bit width can be implemented in it. Therefore, FPGA allows for error-free LA calculations using very long integers, which represent the numerators and denominators of exact rational numbers. This feature is utilized in the processor, which is described below.

2 Determinant computation

An upper triangular matrix can be calculated using the Givens algorithm as, for example, in [1]. Assume that the vectors consisting of the pairs of elements of the matrix rows can increase in their length, which should be taken into account when calculating the determinant. Then the annealing of subdiagonal elements is obtained using only the operations of addition and multiplication. For example, the annealing of $a_{2,1}$ is derived by the formula

$$a_{2,i} = a_{2,i} \cdot t - a_{1,i} \cdot p, \tag{1}$$

where $t = a_{1,1}$, $p = a_{2,1}$. The resulting determinant is corrected by dividing by the factor t. Not to perform several division operations, the division implemented by the product of the coefficients t. The resulting algorithm for the n-dimensional matrix is shown below.

```
d = 1;
   for i = 1:n − 1
 t = A(i,i);
   for j = i+1:n
      p = A(j,i);
      if j < n   d = t * d; end
      for k = i:n
        A(j,k) =  A(j,k)*t − A(i,k)*p;
      end
   end
end
detA = A(n,n) /d;
```

Here the variable d is equal to the product of the coefficients t, and the element $A(n,n)$ of the resulting matrix is equal to the determinant numerator. Since the input data are integers, the resulting determinant $detA$ is integer as well. Since the division operation is applied only once, it can be implemented in the simplified division unit.

The numbers $A(j,k)$ and d can have a huge value. And it is the disadvantage of this algorithm. But the algorithm has a high level of parallelism. In addition, it gives consistent results for all non-singular matrices. When the algorithm was executed in Matlab for several test matrices it showed the same results as a function incorporated in this tool.

3 Processor for computation of inverse matrices

The IP core of the processor module for error-free calculation of inverse matrices was designed using the synthesizable VHDL language style. The module is parameterized by the matrix dimension n and input data bit width m. The arithmetic unit (AU) of the processor calculates the formula (1), which represents the inner loop operation of the core loop nest of the mentioned algorithm. To obtain a high clock frequency multi-level pipelining is used in AU. In the work [8] is shown that the numerator and denominator bit width of order nm is enough for the LU decomposition of the matrix in the arithmetic of rational numbers. Therefore, the AU bit width is selected which is higher than nm. Additionally, the AU has an overflow flag, which allows us to fix the incorrect result, upon which we can select the bit width empirically.

The processor was configured in the FPGA Xilinx Spartan-6. Fig.1 illustrates the processor characteristics per AU data bit width. The characteristics are hardware volume in number of multiply units DSP48, number of LUTs per one multiply unit, and the maximum clock frequency. The Fig.1 shows that the processor can perform calculations with a speed of 120 – 300 million operations of addition and multiplication of large integers per second.

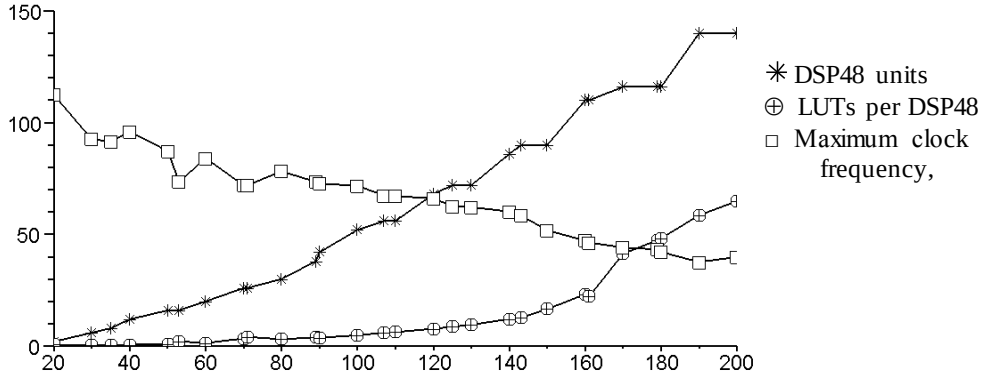

Fig. 1. Hardware volume and maximum clock frequency of the processor depending on the AU bit width

The twofold performance increase can be achieved by describing AU in the structural style, and using the high-speed adders of the DSP48 units. Another great performance increase is obtained with a processor array implementation. In this case, due to the high degree of parallelism of the algorithm, and local communications, the greater acceleration can be achieved than when implementing similar algorithms in microprocessor-based multiprocessor systems [4].

4 Conclusion

Modern FPGAs provide users with great hardware computing resources, including thousands of multiplier blocks and hundreds of thousands of LUTs. Using these resources, one can to implement hardware algorithms, which operate with very long data words. The implementation of the error-free matrix inversion algorithm in FPGA has demonstrated the feasibility and effectiveness of calculating the long data words. Processor for the matrix inversion may be effectively used in adaptive filtering, pattern recognition, cryptoanalysis etc., which require the solution of problems in real time.

References

[1] A. Sergyienko, O. Maslennikov. Implementation of Givens QR Decomposition in FPGA. *PPAM'2001, Springer, LNCS*, 2328: 453–459. 2002.

[2] A. Sergiyenko, O. Maslennikow, V. Lepekha. FPGA Implementation of the Conjugate Gradient Method. *PPAM'2005, Springer, LNCS*, 3911: 526–533. 2006.

[3] A. Sergiyenko, O. Maslennikow, V. Lepekha, A. Tomas, R. Wyrzykowski. Parallel Implementation of Cholesky LL^T Algorithm in FPGA–Based Processor. *PPAM'2007, Springer, LNCS*, 4967: 137–147. 2008.

[4] L. Zhuo, V. K. Prasanna. High-Performance Designs for Linear Algebra Operations on Reconfigurable Hardware. *IEEE Trans. On Computers*, 57(8): 1057-1073. 2008.

[5] R.T.Gregory, E.V.Krishnamurthy, Methods and applications of error-free computation. *Springer, New York, Berlin, Heidelberg, Tokyo*: 1984.

[6] C.K.Koc, R.M.Piedra. A Parallel Algorithm for Exact Solution of Linear Equations. *Proc. Int. Conf. On Parallel Processing*: 3.1-3.8. 1991.

[7] M. Mohac. Error-Free Algorithms to Solve Special and General Discrete Systems of Linear Equations. *XI Int. Workshop on Advanced Computing and Analysis Techniques in Physics Research*. Amsterdam, the Netherlands: 23-27 April, 2007.

[8] A.W. Panyukov, M.I. Germanenko. Linear Equation Error-Free Solving. *Proc. South-Ural State University, Ser.: Mathematics, Physics, Chemistry*. 10: 33–40. 2009.