System Design of an FPGA Linear Solver

Jiří Buček, Pavel Kubalík, Róbert Lórencz, Tomáš Zahradnický
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9, 166 21 Prague, Czech Republic
email: { bucekj | xkubalik | lorencz | zahradt }@fit.cvut.cz

***Abstract*—The work is focused on design of a Modular System performing error-free solution of dense linear systems using residue arithmetic in Xilinx FPGA. The designed system shall use a set of Residual Processors (RP)s for linear system solution in Residue Number System and reconstruct the set's solution afterwards. The currently proposed system's architecture has a single RP, a large DDR memory used for data transfer in between a PC and the system, and a built-in MicroBlaze processor. Future work will focus on extending the architecture to implement the entire Modular System consisting of multiple RPs and performing the backward transformation from residue representation into the rational number set.**

***Keywords*—system of linear equations; residue number system; error-free computation; FPGA**

I. Introduction

Solution of a system of linear equations (SLE) is a basic, yet often difficult task of linear algebra. On input, there is an augmented matrix of the SLE and a SLE's solution vector on the output. Whether the solution is difficult or not depends on many aspects including a dimension, density, conditioning, accuracy requirements, and used arithmetic. Since solution mostly occurs in either floating point arithmetic, roundoff errors are committed, accumulated, and possibly having severe impact at the solution. A way go around the problem is to avoid rounding at all by using an error-free arithmetic, such as the arithmetic of the Residue Number System (RNS) [1].

SLE solution in RNS is built on the Chinese Remainder Theorem (CRT) and proceeds in 3 stages [2]:

Method 1

1) Transformation of the augmented SLE matrix[1] $\mathbf{A} \mid \mathbf{y}$ into independent linear systems $(\mathbf{A} \mid \mathbf{y}) \bmod m_i$, each with a distinct prime number modulus m_i, for $i = 1, \ldots, p$; p being a number of moduli.
2) Solution of the independent systems of linear congruences (SLC)s in form $\mathbf{Ax} \equiv \mathbf{y} \pmod{m_i}$.
3) Reconstruction of the SLE solution $\mathbf{x}$ from its RNS residual representations $\mathbf{x} \pmod{m_i}$.

Method 1 has its pros and cons. The advantages are that there is no rounding, the solution process can occur in parallel, and is error-free. The disadvantages are an increased time and space complexities necessary for the solution. However it is possible to implement the method in special hardware.

The paper continues with design of a dedicated SLE solution hardware in RNS. The architectures so far designed, their testing, and implementation in FPGA and ASIC technologies were covered in papers [3] and [4]. These papers focused on design and implementation of an SLC solver (Stage 2 of Method 1), while this paper interconnects solvers at Stage 2 with transformation processes performed at Stage 1.

The paper is organized as follows: Section I (Introduction) introduces the reader into the context of the paper. Section II (Previous Work) discusses so far designed parts of the designed SLE solver. Section III (Modular System Architecture on FPGA) describes the architecture of the SLE solver using previously described SLC solvers (Residual Processors), while Section IV (Conclusion and Future Work) summarizes the paper and provides an outlook at our future efforts.

II. Previous Work

SLE solution in RNS performed in a dedicated hardware Modular System (MS) requires that the system is able to perform all Method 1 stages. An architecture of such MS is depicted at Fig. 1:

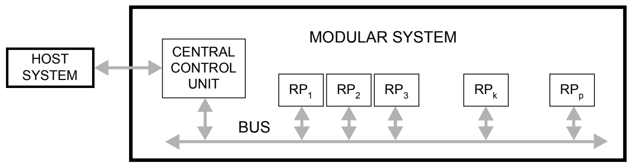

Fig. 1. Architecture of the Modular System [5]. The Central Control Unit controls processes required by Method 1 and communicates with the Host System. RP_1 through RP_p are individual Residual Processors, each with its own modulus m_i for $i = 1, \ldots, p$, supporting all three Method 1 stages. The Bus denotes an internal interconnection of all units within the MS.

The MS consists of a Central Control Unit (CCU), p hardware identical Residual Processors (RP)s, and an interconnection Bus. CCU communicates with an external Host System such as a computer, coordinates RP's work, and dispatches data to and from RPs. Each RP works with its own distinct prime number modulus m_i and is designed to support all three Method 1 stages. The Bus, not necessarily implemented as a data/control bus, denotes necessary interconnection of all units within the MS. The following paragraphs will recapitulate so the far achieved progress in papers [3] and [4], both dealing with an RP architecture.

Paper [3] presents an initial RP architecture. The architecture was designed to perform a Gauss-Jordan-Rutishauser (GJR) elimination upon individual SLCs mod m_i stored within RP's internal memory. The RP architecture contains specialized Arithmetic Units (AU)s interconnected with RP's memory.

[1]The $\mathbf{A}$ matrix generally contains either floating-point numbers or integers.

This allows performing vector (SIMD) operations corresponding to the GJR elimination. The paper also discusses dedicated hardware for residual pivoting solving a zero pivot occurrence problem. The proposed architecture was implemented in FPGA (Xilinx Virtex 6) for various SLC dimensions up to $n = 1000$ with a 24-bit m_i. The implementation in FPGA was approx. 2 times faster than its GCC-compiled software counterpart running on an Intel T9400 CPU at 2.53 GHz.

Paper [4] compares FPGA and ASIC implementations. ASIC allowed a larger memory subsystem than in FPGA. Both designs were compared and the ASIC implementation was about 4 times faster than FPGA ($n = 1000$). Both [3] and [4] discuss solving SLCs, i.e. Stage 2 of Method 1. Next, it is necessary to design Stage 1, that is, communication of the MS with the Host System and initialization/preparing RPs for the SLC solution process performed in Stage 2.

III. Modular System Architecture on FPGA

This section deals with design of a MS communication architecture in FPGA. The architecture at Fig. 2 is designed to allow interconnection of RPs with the Host System, Main Memory, and other units to perform Stage 1.

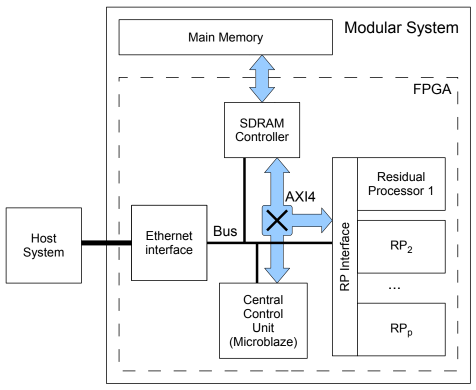

Fig. 2. Modular System in FPGA. The Central Control Unit is a MicroBlaze soft-processor. RP_1 through RP_p are individual Residual Processors, with respective moduli m_1 to m_p. RPs form a common AXI4 bus master peripheral. The AXI4 crossbar switch is used for a high speed interconnect, the Bus is used for a lower speed programmed I/O.

The matrix $\mathbf{A}|\mathbf{y}$ is sent to the MS from the Host System using a communication interface. Ethernet is suitable for its good flexibility and support in FPGA development systems in several performance options. The matrix is transferred in the form of multi-word integer numbers, assuming preprocessing by the Host System. The matrix is stored in the Main Memory and is therefore prepared for loading and conversion to $(\mathbf{A}|\mathbf{y}) \bmod m_i$ inside individual RPs.

The Main Memory is connected to the FPGA with a (DDR3) memory controller supporting burst transfers with the other parts of the system. SLC solution is done in the RPs which require synchronized fast access to the input data. For this reason, it is essential to provide a high-throughput channel from the system main memory to the residual processors. This is achieved by using a high performance interconnection (AXI4 crossbar) between the memory controller and the set of RPs. RPs share a common data channel from memory since they always get the same data. During loading of the matrix, each RP reduces the data modulo m_i. The set of RPs is connected using a common bus-master RP interface peripheral.

The Central Control Unit (CCU) controls communication, data loading, and synchronization. It is implemented using a Xilinx MicroBlaze processor. After receiving the augmented matrix, it starts the data-loading process from the main memory to the RPs. This is done by initiating a bus master read by the RP peripheral from a specified address in memory. After all data is loaded, RPs start their SLC solutions independently. The CCU can either wait for the RPs to complete, or perform other tasks such as communication to load another instance from the Host System, or other data processing.

The number of clock cycles needed for load and elimination process in the RP can be calculated by (1) and (2) for matrix size n, word length z, and word count q.

$$\mathrm{load}(n, z, q) = (2 + (10 + 3n)q + (2q - 1)z)n, \qquad (1)$$
$$\mathrm{elim}(n, z) = ((z + (4z - 2))n + 3)n + 14. \qquad (2)$$

We have implemented the system with a single RP attached to the RP interface and verified it using the Xilinx ML605 development board. Preliminary results show that times solving the SLC (i.e. elimination) agree with the prediction (2), however the loading times are slower than (1), burdened by the synchronization overhead by the CCU.

IV. Conclusion and future work

We have proposed a communication architecture of a Modular System for solving systems of linear equations. We can transfer data from PC to all RP units, calculate solution and transfer result from all RP to PC. The backward transformation from RNS to the rational numbers will be processed in PC. To solve a linear system, the more calculation than available unit must be processed and many runs for RP unit are needed. Further work will focus on analyzing the loading process and obtaining a more precise model describing the performance. We will also focus on lowering the overhead and implementing parallel processing with multiple RPs. We will implement this architecture on a Xilinx FPGA development board ML605 and verify its function with more RP units.

Acknowledgment

This research was supported by the Czech Science Foundation project no. P103/12/2377.

References

[1] R. T. Gregory and E. V. Krishnamurthy, *Methods and Application of Error-free Computation*. Springer Verlag, 1984.

[2] M. Morháč and R. Lórencz, “A modular system for solving linear equations exactly, ii. hardware realization,” *Computers and Artificial Intelligence*, vol. 11, no. 5, pp. 497–507, 1992.

[3] J. Buček, P. Kubalík, R. Lórencz, and T. Zahradnický, “Dedicated Hardware Implementation of a Linear Congruence Solver in FPGA,” in *The 19th IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2012*. Monterey: IEEE Circuits and Systems Society, 2012, pp. 689–692.

[4] ——, “Comparison of FPGA and ASIC implementation of a linear congruence solver,” in *2013 Euromicro Conference on Digital System Design*, 9 2013, pp. 284–287.

[5] R. Lórencz and M. Morháč, “A modular system for solving linear equations exactly, i. architecture and numerical algorithms,” *Computers and Artificial Intelligence*, vol. 11, no. 4, pp. 351–361, 1992.