A Scalable and Compact Systolic Architecture for Linear Solvers

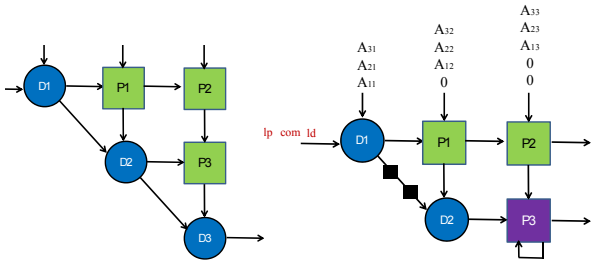
Kevin S. H. Ong*, Suhaib A. Fahmy*, Keck-Voon Ling†
*School of Computer Engineering
†School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore
Email: shong6@ntu.edu.sg

***Abstract*—We present a scalable design for accelerating the problem of solving a dense linear system of equations using LU Decomposition. A novel systolic array architecture that can be used as a building block in scientific applications is described and prototyped on a Xilinx Virtex 6 FPGA. This solver has a throughput of around 3.2 million linear systems per second for matrices of size N=4 and around 80 thousand linear systems per second for matrices of size N=16. In comparison with similar work, our design offers up to a 12-fold improvement in speed whilst requiring up to 50% less hardware resources. As a result, a linear system of size N=64 can be implemented on a single FPGA, whereas previous work was limited to a size of N=12 and resorted to complex multi-FPGA architectures to scale. Finally, the scalable design can be adapted to different sized problems with minimum effort.**

I. Introduction

Previous work in the area has focused on solving large problem sizes using iterative algorithms, such as the Conjugate Gradient Method, with a focus on sparse matrices. To achieve high linear solver performance, a digital hardware designer must be involved at all stages of the design. Hence, scientific application designers without background knowledge cannot modify such architectures for their needs. By not exploiting some of the high level design environments available, these designs represent point solutions rather than a toolkit for scientists. One of the key areas where linear solvers for large problems are being explored is in model predictive control (MPC) [1].

In this paper, we present an a scalable design for accelerating the problem of solving a dense linear system of equations using LU Decomposition. A systolic array approach facilitates easy scaling through structural regularity,allowing the linear solver to be rapidly prototyped using high-level software tools and non-experts to make use of it at a higher architecture level. While prior work avoids divide operations for the matrix inversion step, our proposed design performs floating-point division and our linear solver has a throughput of ∼3.2 million linear systems per second for matrices of size $N = 4$ and ∼80 thousand linear systems per second for matrices of size $N = 16$. This represents up to a 12× improvement in speed over previous work, while requiring up to 50% less hardware resources. Readers are reminded that the proposed scalable systolic array architecture, while suited to MPC, can be applied to general scientific computing problems where a system of linear equations is to be solved. More importantly, there is no need for a hardware designer to be on hand.



(a) Existing **(b)** Proposed

Fig. 1: Comparison of Triangular Systolic Array Architecture

II. Systolic Array Design

Due to space constraints, the reader is assumed to have prior knowledge on the subject of systolic arrays.

Existing TSA designs, as shown in Fig. 1a, require $N(N+1)/2$ PEs and the values of the L and U matrices are produced after $2N$ time-steps. A detailed examination and re-mapping of computational operations for LU Decomposition leads us to propose a triangular systolic array (TSA) design which requires a total of $(N(N+1)/2)-1$ PEs, which we call LU-TSA, and is shown in Fig. 1b. The saving of one divider PE is significant due to the area cost of those nodes. In addition to performing matrix decomposition using the LU method, both forward and back substitution steps are required to solve a system of linear equations. The total time required for the substitution steps is $2N^2$ operations.

Both [2] and [3] implemented linear solver SA designs of dimensions up to 12×12 on a single FPGA. In addition, existing work suggests that backward substitution is performed sequentially after forward substitution and no further parallelism is possible. But close examination of both substitution steps enables us to introduce parallelism in the substitution step. We propose that the diagonal division operations of the U matrix be performed independently of the back substitution step by means of the reciprocal operation. Hence, the time-consuming reciprocal latency can be hidden well within the latency of the forward substitution step, saving N time-steps, for a problem size as small as $N = 4$, and enabling hardware re-use.

978-1-4799-3609-0/14/$31.00 © 2014 IEEE

III. Implementation Setup and Results

A. System Setup

The proposed TSA-based linear solver was implemented using Xilinx System Generator (SysGen) [4] targeting a mid-range Xilinx Virtex 6 FPGA (XC6VLX240T). Our proposed design implements signed fixed-point number format with 9 integer bits and 8 binary bits, similar to [5], however our design differs in being wordlength and matrix size parameterizable at the PE level within the SA architecture. We make use of the DSP Blocks' dynamic configurability [6] to support multiple operations on the same hardware in different steps. The individual blocks are designed to be easily composable in SysGen. A mixed number representation is used, and we adopt the precision used in [5].

B. Implementation Results

From Table I, our design is $2.5\times$ faster and $\sim 6\%$ smaller than the custom linear solver reported in [7]. It also provides resource savings of up to 50% for large sizes of N. In [8] and [7], $N = 10$ was the largest problem size that could be implemented on a single FPGA. By contrast, our linear solver can be implemented for a size of $N = 16$ (pipelined design) on the smallest Virtex 6 (XC6VLX75T) FPGA. Based on the results in Table I and the structural regularity of our proposed design, we are confident that our linear solver can be implemented for up to $N = 32$ on a Virtex 6 XC6VSX475T and $N = 64$ on a Virtex 7 XC7V2000T.

Compared with [5], our TSA-based linear solver is $\sim 12\times$ faster though no resource consumption results are available for comparison of area. From the power consumption perspective, actual processing utilization for our TSA-based SA design is low when compared to a 1D SA design. Our proposed TSA-based Linear Solver can also exhibit 1D SA like power consumption through the use of clock-gating on FPGAs, which can turn on and off PEs according to their usage.

TABLE I: Linear Solver Performance & Hardware Resource Benchmarking

		Proposed (4x4)	**Proposed (16x16)**	Custom HW (4x4) [7]	MINRES (16x16) [9]
Performance	Word length	**Hybrid(18,9)**	**Hybrid(18,9)**	FXP(20,0)	Floating-Point
	Latency(cycles)	**304**	**2,650**	473	374
	Clock Freq.(MHz)	**∼247**	**∼198**	166	∼250
	Throughput (Msystems/s)	**∼0.82**	**∼0.08**	∼0.35	∼0.04 to 0.68
Resource	Slices	**1,933**	**15,622**	2,025	∼12,500
	BRAM/RAM18E1	**17**	**167**	1	∼37
	DSP48E1/DSP48	**6**	**120**	12	∼40
FPGA	Device Type	**XC6VLX240T**	**XC6VLX240T**	XC4VSX35T	XC5LX330T

The design in [9] uses an iterative floating-point linear solver with maximum iteration count to reach a solution being N, and supporting dense matrices. On the other hand, our proposed linear solver utilizes both floating-point and fixed-point representations. For $N = 16$, our novel TSA architecture is up to $2\times$ faster than their worst case scenario (see Table I). To achieve such high performance, the design in [9] employs manual deep pipelining and symmetry of the A matrix is exploited. Their dense linear solver is able to handle matrices of order up to $N = 145$ while our design has been explored for up to $N = 64$. Their design achieves this by extensive re-use of floating-point square root and division operators. However, we avoid deep pipelining and datapath customisation in order not to disrupt the structural regularity, and hence scalability, of our design.

IV. Conclusion

A triangular systolic array based linear solver has been presented and implemented on an FPGA platform. Our proposed architecture does not side-step the computationally expensive floating-point division operations as is typical in other work. Our design offers up to a $12\times$ improvement in speed whilst requiring up to 50% fewer hardware resources. As a result, a direct linear solver of size $N = 64$ can be implemented on a single FPGA. More importantly, the systolic array approach, with the optimized computational nodes we have designed, allows the design to be tailored by non-experts for their particular application.

So far, we have not exploited any special properties of the A matrix, such as symmetry, bandedness or sparsity, and this will be explored in future work. Secondly, we intend to increase the number of design parameterization options. We will also explore the use of iterative floating point computation [10] for the divisions. Lastly, a comparison study of the proposed architecture against other GPU and floating point DSP implementations will be explored.

References

[1] J. Jerez, K.-V. Ling, G. Constantinides, and E. Kerrigan, "Model predictive control for deeply pipelined field-programmable gate array implementation: algorithms and circuitry," *IET Control Theory and Applications*, vol. 6, no. 8, pp. 1029–1041, 2012.

[2] M. Karkooti, J. R. Cavallaro, and C. Dick, "FPGA implementation of matrix inversion using QRD-RLS algorithm," in *Proceedings of Asilomar Conference on Signals, Systems and Computers*, 2005, pp. 1625–1629.

[3] Y. H. Hu and S.-Y. Kung, "Systolic arrays," in *Handbook of Signal Processing Systems*. Springer, 2013, pp. 1111–1143.

[4] Xilinx, *System Generator for DSP User Guide UG640 (v 14.3)*. Xilinx, 2012.

[5] A. Mills, A. Wills, S. Weller, and B. Ninness, "Implementation of linear model predictive control using a field-programmable gate array," *IET Control Theory and Applications*, vol. 6, no. 8, pp. 1042–1054, 2012.

[6] H. Y. Cheah, S. A. Fahmy, D. L. Maskell, and C. Kulkarni, "A lean FPGA soft processor built using a DSP block," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, 2012, pp. 237–240.

[7] A. U. Irturk, "GUSTO: General architecture design utility and synthesis tool for optimization," PhD Thesis, University Of California, San Diego, 2009.

[8] C. Wan, "Systolic algorithms and applications," PhD Thesis, Loughborough University, United Kingdom, 1996.

[9] D. Boland and G. A. Constantinides, "Optimizing memory bandwidth use and performance for matrix-vector multiplication in iterative methods," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 4, no. 3, p. 22, 2011.

[10] F. Brosser, H. Y. Cheah, and S. A. Fahmy, "Iterative floating point computation using FPGA DSP blocks," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL)*, 2013.