Implementation of parallel tridiagonal solvers for a heterogeneous computing environment

H. J. Macintosh[1] D. J. Warne[2] N. A. Kelson[3]
J. E. Banks[4] T. W. Farrell[5]

(Received 3 March 2015; revised 8 February 2016)

Abstract

Tridiagonal diagonally dominant linear systems arise in many scientific and engineering applications. The standard Thomas algorithm for solving such systems is inherently serial, forming a bottleneck in computation. Algorithms such as cyclic reduction and SPIKE reduce a single large tridiagonal system into multiple small independent systems which are solved in parallel. We develop portable cyclic reduction and the SPIKE algorithm for Open Computing Language implementations on a range of co-processors in a heterogeneous computing environment, including field programmable gate arrays, graphics processing units and other multi-core processors. We evaluate these designs in the context of solver performance, resource efficiency and numerical accuracy.

Contents

1 Introduction

Tridiagonal and block tridiagonal linear systems arise in many computational applications. Some applications involve the solving of many small independent systems whilst others require solving fewer large systems [13, 18]. The focus of this article is the case when the linear system is strictly diagonally dominant.

As high performance computing (HPC) platforms become more parallel, specialised and heterogeneous, the requirement for parallel algorithms to be portable as well as efficient is increasing. We develop an Open Computing

Language (OpenCL) implementation of two well known parallel tridiagonal solvers. Our implementation is portable and capable of executing on a range of co-processor environments.

Of particular interest in this work are field programmable gate arrays (FPGAs). These are reconfigurable computing devices that consist of an interconnected array of configurable logic blocks and memory modules in the form of distributed block RAMs. The configurable logic blocks are simple units built from look-up tables, logic gates and multiplexers. Configurable logic blocks can be configured and routed at run time to define custom processor architectures.

Traditionally, FPGAs were mainly used in embedded devices or for hardware prototyping. Recently, the floating point performance of FPGAs reached a point where they have become much more relevant to the HPC arena, particularly with the approach of exascale machines [6, 11, 15]. As a result, FPGA-based linear algebra architecture design is an area of active research [12, 16, 18, 21].

The biggest barrier to effective use of FPGAs has always been the difficulty in programming them. Until recently, this entailed a hardware design process rather than software development. However, advances have been made with both major FPGA vendors releasing OpenCL compliant boards which produce comparable results to manual hardware design [1, 20].

Using the Altera based OpenCL, Warne et al. [18, 19] demonstrated the ease in which a custom tridiagonal linear system solver can be deployed. In this work, we extend our previous efforts towards a more general highly parallel solution, targeting FPGAs in particular, but also other OpenCL compliant co-processors that may be present within a heterogeneous computing environment.

2 Background

2.1 Tridiagonal linear systems

A linear system $Ax = \mathbf{b}$ is tridiagonal if the coefficient matrix $A \in \mathbb{R}^{n\times n}$ is banded with bandwidth $\beta = 1$. That is,

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & \\ & \ddots & \ddots & \ddots & \\ & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & a_{n,n-1} & a_{n,n} \end{bmatrix}.$$

Considering strictly diagonally dominant systems only, it is well known that the LU-decomposition can be performed in $\Theta(n)$ operations using the Thomas algorithm [14]. Although $\Theta(n)$ is a vast improvement on a fully dense LU-decomposition, the Thomas algorithm is inherently serial. As a result, the high performance of modern highly parallel computing architectures cannot be directly exploited.

For applications involving many small independent systems it is easy to achieve parallel computation using a standard Thomas algorithm. However, more advanced, inherently parallel methods must be applied if the problem requires solving fewer large systems.

Specialised architectures for specifically solving tridiagonal linear systems were designed using field programmable gate arrays [12, 18]. However, these systems were built with very specific uses in mind.

2.2 Heterogeneous computing with OpenCL

The ideal parallel approach depends greatly on the computing devices available to the program. HPC platforms are becoming more reliant on specialist co-processors, each with their own programming models. This naturally makes designing a general, portable solution difficult.

OpenCL is an open standard for heterogeneous computing [9]. Implementations of the OpenCL standard were developed by vendors of a wide range of computing devices including many-core CPUs, graphics processing units (GPUs), digital signal processors and, recently, FPGAs. These implementations enable a developer to build a portable computational routine which will execute on any of the available resources.

The OpenCL programming model divides program code into two partitions: 1) serial code to be executed on a host processor; and 2) parallel code to be executed on any number of co-processor devices. Each device is treated as an array of compute units which can operate asynchronously, and each unit is further divided into synchronised processing elements.

3 Parallel tridiagonal linear systems solvers

Many parallel algorithms exist for solving tridiagonal and block-tridiagonal linear systems and are implemented in well established numerical libraries such as Scalapack [3, 4, 7, 10]. In this section, we briefly present two parallel tridiagonal solver schemes which were the focus of GPU implementations, namely, cyclic reduction [8] and SPIKE [13].

3.1 Parallel cyclic reduction

The method of cyclic reduction splits a single tridiagonal system into two smaller independent tridiagonal systems [8]. This is done by transforming the ith equation,

$$a_{i,i-1}x_{i-1} + a_{i,i}x_i + a_{i,i+1}x_{i+1} = b_i \,,$$

into

$$\alpha_i a_{i,i-1} x_{i-2} + (a_{i,i} + \alpha_i a_{i-1,i} + \beta_i a_{i+1,i})\, x_i + \beta_i a_{i,i+1} x_{i+2} = b_i + \alpha_i b_{i-1} + \beta_i b_{i+1}\,,$$

where $\alpha_i = -a_{i,i-1}/a_{i-1,i-1}$ and $\beta_i = -a_{i,i+1}/a_{i+1,i+1}$. The result of cyclic reduction is two independent tridiagonal linear systems, one containing equations with even indexed unknowns and another containing equations with odd indexed unknowns. This transformation can be applied recursively to any number of independent systems.

3.2 SPIKE

Another parallel method is the so-called "SPIKE" algorithm [13]. In this method, the coefficient matrix is factorised to $A = DS$ where matrix $D = \operatorname{diag}(A_1, A_2, \ldots, A_p)$ with A_j diagonal blocks of A, and

$$S = \begin{bmatrix} I & V_1 & & & \\ W_2 & I & V_2 & & \\ & \ddots & \ddots & \ddots & \\ & & W_{p-1} & I & V_{p-1} \\ & & & W_p & I \end{bmatrix}, \tag{1}$$

where p is the number of partitions, m is the number of linear equations per partition, and n is the total number of linear equations in the system, so $n = pm$. Here $W_i, V_i \in \mathbb{R}^{m\times 1}$ are obtained by solving

$$A_i \begin{bmatrix} V_i & W_i \end{bmatrix} = \begin{bmatrix} 0 & a_{im+1,im} \\ \vdots & \vdots \\ a_{im,im+1} & 0 \end{bmatrix}. \tag{2}$$

Factorising A in this way allows the original $Ax = b$ to be solved by first solving $Dy = b$ as p independent tridiagonal systems and then solving $Sx = y$. The second step can also be transformed into p independent systems in $\Theta(p)$ operations.

Algorithm 1: : PCR compute kernel logic

```
$[i_G, i_L] \leftarrow$ GET_IDS {*work group and work item index*}
$U_c, D_c, L_c, b_c \in \mathbb{R}^n$ {*shared local cache of system $i_G$*}
$s \leftarrow 1$ {*solve for $x_{i_L}$ in $r$ recursive steps*}
**for all** $i \in [1, \cdots, r]$ **do**
  $\alpha \leftarrow -L_c[i_L]/D_c[i_L - s]$, $\beta \leftarrow -U_c[i_L]/D_c[i_L + s]$
  $U' \leftarrow \beta U_c[i_L + s]$, $L' \leftarrow \alpha L_c[i_L - s]$
  $D' \leftarrow D_c[i_L] + \alpha U_c[i_L - s] + \beta L_c[i_L + s]$
  $b' \leftarrow b_c[i_L] + \alpha b_c[i_L - s] + \beta b_c[i_L + s]$
  {*synchronise all work items*}
  $[U_c[i_L], D_c[i_L], L_c[i_L], b_c[i_L]] \leftarrow [U', D', L', b']$
  $s \leftarrow 2s$
**end for**
```

3.3 Implementation using OpenCL

For parallel cyclic reduction (PCR), the kernel code developed in this work defines the process of solving for a single unknown. The ith processing element solves for x_i. A single linear system is loaded from 'off-chip' global memory into fast local memory which is shared by all processing elements within a compute unit. Each x_i is then solved simultaneously as shown in the pseudo-code in Algorithm 1.

At the present stage of development, our SPIKE implementation uses a direct mapping of a processing element to the work required to factorise the pth partition of A. With this allocation of workload to processors, a number of processors operate in parallel on each compute unit. There is a sequential bottleneck of $\Theta(p)$ in reducing the system $Sx = y$ into a set of parallel back-substitutions. The final solution is then recovered though parallel back-substitution in the same manner as the factorisation step.

4 Evaluation

One of the main aims of this work is the exploration and development of efficient parallel tridiagonal solvers which take advantage of a range of possible compute platforms. To assess the feasibility of the design, we specifically focus on resource utilisation, compute performance and numerical accuracy. In this section, our implementations of PCR and SPIKE are evaluated against pre-existing implementations targeting NVIDIA GPUs. Not only do the results show that our designs are comparable when running on NVIDIA hardware but they also indicate that the FPGA platform is a promising alternative.

4.1 FPGA resource utilisation

Unlike fixed architectures such as GPUs and CPUs, OpenCL implementations targeting FPGAs allow the number of compute units and processing elements to be varied. However, the specific choices made for these will affect both algorithm performance and utilisation of resources on the FPGA configurable logic fabric.

We tuned the combination of both compute units and processing elements for maximum throughput for both the PCR and SPIKE implementations. Table 1 gives the optimal configurations. The estimated FPGA resource utilisation percentages are also included in the table, and these were generated for the target Altera Stratix V architecture using the Altera OpenCL software development kit. The percentages of consumed FPGA logic resources (i.e., block RAM and lookup tables) are within the total amounts available and show that each design implementation on the target architecture is feasible. However, for maximum throughput the FPGA cannot be configured to run all SPIKE stages at the same time, so instead some reconfiguration between stages is necessary.

Table 1: Logic utilisation percentage of optimum configuration for the Altera Stratix V FPGA.

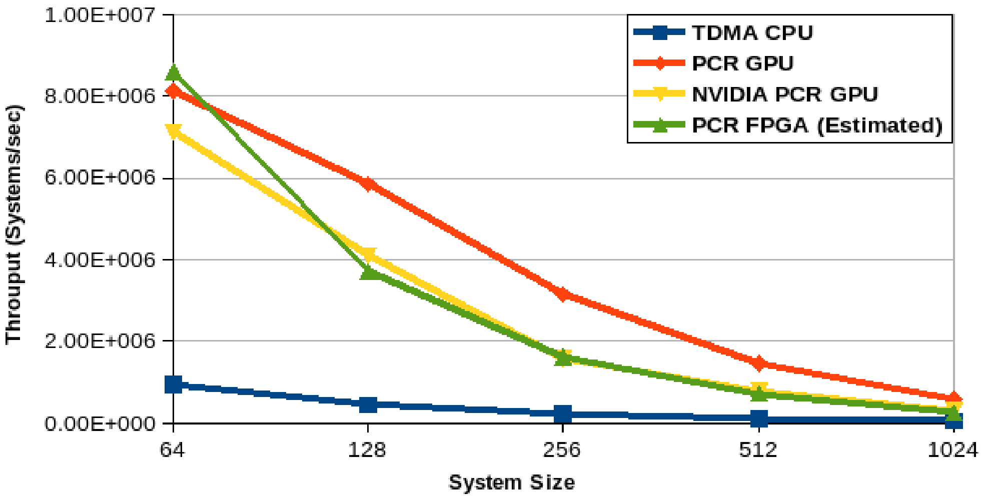
Design	Compute Units	Processing Elements	Block RAM	Lookup Tables
PCR	8	2	92%	99%
SPIKE factorise	1	4	92%	99%
SPIKE partition $Sx = y$	1	16	43%	46%
SPIKE back-solve	1	8	22%	30%

4.2 Compute performance

We are not aware of any other OpenCL implementation of PCR or SPIKE designed with an FPGA as the target architecture. This makes it difficult to benchmark. As a result, we compiled our OpenCL code for an NVIDIA GPU to aid comparison with other PCR and SPIKE implementations along with the performance for a sequential Thomas CPU based solution. In addition, our estimated FPGA performance is placed along side these results. The FPGA estimates are produced by the Altera offline compiler; an analysis of the hardware schematic generates these through the compilation process to provide the number of OpenCL work items that can be executed per second. We converted the estimated throughput to system solves per second (rather than more usual FLOP based measures) to ease comparison with the most relevant studies. In reality, data transfer overheads can impede this throughput [19]; however, there are a number of coding practices which can assist in minimising this impact [2].

NVIDIA provides an OpenCL implementation of PCR which is distributed with their CUDA software development kit. Figure 1 provides the throughput (Systems/sec) for our PCR implementation against the NVIDIA implementation for systems of size $n \in [64, 128, 256, 512, 1024]$. Our implementation outperforms NVIDIA's when running on an NVIDIA Quadro 4000 GPU. Our estimates also show that the FPGA implementation will run on-par with the NVIDIA GPU code; considering the FPGA is running at only a fraction of the

Figure 1: Compute throughput of parallel cyclic reduction implementations versus a single threaded Thomas algorithm



clock speed ($\approx 150\,\mathrm{MHz}$) this is quite impressive.

An analogous performance comparison was performed with our SPIKE implementation against a CUDA implementation. For the given system size range, our GPU based implementation has comparable performance to Chang et al. [5]. Also, the estimated performance of our FPGA design out-performs both GPU implementations by approximately a factor of four. The FPGA estimates are likely to be optimistic in this case as the re-configuration time required when switching between SPIKE stages is not taken into consideration. Although not explored here, this re-configuration time ($\leqslant 100\,\mathrm{ms}$) could be hidden in practice by, for example, processing SPIKE stages in batches.

Figure 2: Compute throughput of SPIKE implementations versus a single threaded Thomas algorithm

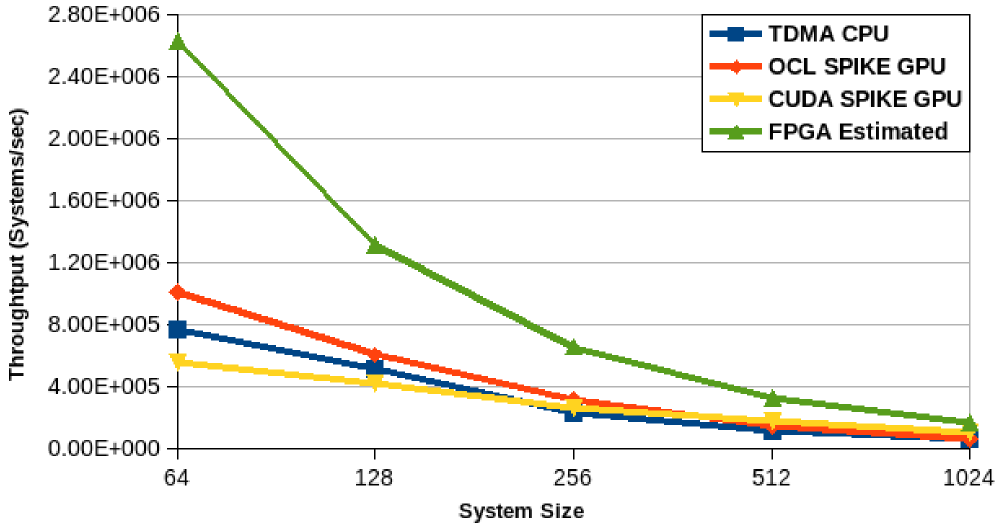


Table 2: Numerical accuracy of PCR and SPIKE ($n = 1024$) .

	Thomas	PCR	SPIKE
$\|\mathbf{x} - \mathbf{b}\|_\infty$	3.6×10^{-7}	5.4×10^{-7}	4.2×10^{-7}

4.3 Numerical accuracy

We found both our PCR and SPIKE implementations to be of similar accuracy to a standard Thomas algorithm (see Table 2). Currently, all our analysis and test cases are based on strictly diagonally dominant matrices which do not require pivot operations. In future work, we intend to perform a more general numerical stability analysis as we extend the work to other matrix types.

4.4 Power utilisation

As the purpose of this study is to assess the feasibility of building efficient linear algebra implementations for a heterogeneous computing environment, we have not performed any detailed analysis on power consumption. However, the attraction of such a heterogeneous environment is strongly motivated by potential increase in power efficiency due to the significantly (order of magnitude) lower power consumption of FPGAs [6].

To put Figures 1 and 2 into context, the Altera FPGA PCI-e card consumes a maximum of 35 W under load. By comparison, an NVIDIA Quadro 4000 has a maximum power consumption of 142 W. While a detailed examination of the various tridiagonal solvers remains for future work, we previously investigated power utilisation in the context of image processing algorithms with promising results [17].

5 Conclusion

In this article we explored the feasibility of designing truly portable multi-platform high performance linear algebra routines using OpenCL. The parallel tridiagonal solvers of cyclic reduction and SPIKE were implemented on an FPGA and a GPU and compared against other OpenCL and CUDA implementations. Results to date indicate that in terms of accuracy and computational efficiency, our designs are comparable for both FPGA and GPU cases. This suggests that further investigation is warranted, and in future work we intend to continue to develop and improve upon our designs for tridiagonal solvers in the broader context of providing enhanced platforms for high performance, low power numerical routines.

Acknowledgments This project utilised the HPC facility at the Queensland University of Technology (QUT). The facility is administered by QUT's HPC and research support group. A special thanks to the HPC group for their support, particularly in providing access to specialist FPGA and GPU resources.

References

[1] Altera. *Implementing FPGA design with the OpenCL standard.* Technical report, Altera Inc., November 2013. http://www.altera.com/literature/wp/wp-01173-opencl.pdf C448

[2] Altera. *Altera SDK for OpenCL: Best practices guide.* Technical report, Altera Inc., May 2015. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/opencl-sdk/aocl_optimization_guide.pdf C454

[3] P. Arbenz, A. Cleary, J. Dongarra, and M. Hegland. A comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems. *Parallel and Distributed Computing Practices*, 2:385–400, 1999. http://www.scpe.org/index.php/scpe/article/view/152 C450

[4] P. Arbenz, A. Cleary, J. Dongarra, and M. Hegland. A comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems II. In *Euro-Par'99 Parallel Processing*, Berlin, vol. 1685 of *Lecture Notes in Computer Science* pp. 1078–1087. Springer, 1999. doi:10.1007/3-540-48311-X_151 C450

[5] L.-W. Chang, J. A. Stratton, H.-S. Kim, and W. W. Hwu. A scalable, numerically stable, high-performance tridiagonal solver using GPUs. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11. IEEE Computer Society Press, 2012. doi:10.1109/SC.2012.12 C455

[6] Y. Dou, Y. Lei, G. Wu, S. Guo, J. Zhuo, and L. Shen. FPGA accelerating double/quad-double high precision floating-point applications for exascale computing. In *Proceedings of the 24th ACM International Conference on Supercomputing*, pp. 325–336, 2010. doi:10.1145/1810085.1810129 C448, C457

[7] C. R. Dun, M. Hegland, and M. R. Osborne. Parallel stable solution methods for tridiagonal linear systems of equations. In *Computational Techniques and Applications: CTAC95 Proceedings of the Seventh Biennial Conference*, pp. 267–274. World Scientific, 1996. doi:10.1142/9789814530651 C450

[8] W. Gander and G. H. Golub. Cyclic reduction—history and applications. In *Scientific Computing, Proceedings of the Workshop*, 1997, pp. 73–111. Springer, 1998. `http://www.springer.com/fr/book/9789813083608` C450

[9] Khronos OpenCL Working Group. *The OpenCL specification.* Technical report, Khronos Group, October 2009. `http://www.khronos.org/registry/cl/specs/opencl-1.0.pdf` C450

[10] M. Hegland. On the parallel solution of tridiagonal systems by wrap-around partitioning and incomplete LU factorization. *Numer. Math.*, 59(1):453–472, 1991. doi:10.1007/BF01385791 C450

[11] D. Jensen and A. F. Rodrigues. Embedded systems and exascale computing. *Comput. Sci. Eng.*, 12(6):20–29, 2010. doi:10.1109/MCSE.2010.95 C448

[12] S. Palmer. Accelerating implicit finite difference schemes using a hardware optimised implementation of the thomas algorithm for FPGAs. *Technical Report*, 2014. `http://arxiv.org/abs/1402.5094` C448, C449

[13] E. Polizzi and A. H. Sameh. A parallel hybrid banded system solver: the spike algorithm. *Parallel Comput.*, 32:177–194, 2006. doi:10.1016/j.parco.2005.07.005 C447, C450, C451

[14] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN: The art of Scientific Computation.* Cambridge University Press, Cambridge, England, 2nd edition, 1992. http://www.cambridge.org/au/academic/subjects/mathematics/numerical-recipes/numerical-recipes-fortran-77-art-scientific-computing-volume- C449

[15] J. Shalf, S. Dosanjh and J. Morrison. Exascale computing technology challenges. In *High Performance Computing for Computational Science—VECPAR 2010*, vol. 6449 of *Lecture Notes in Computer Science*, pp 1–25. Springer, 2011. doi:10.1007/978-3-642-19328-6_1 C448

[16] S. Skalicky, S. Lopez, M. Lukowiak, J. Letendre and M. Ryan. Performance modeling of pipelined linear algebra architectures on FPGAs. In *Reconfigurable Computing: Architectures, Tools and Applications*, vol. 7806, Lecture Notes in Computer Science pp. 146-153. Springer, 2013. doi:10.1007/978-3-642-36812-7_14 C448

[17] D. J. Warne, R. F. Hayward, N. A. Kelson, J. E. Banks, and L. Mejias. Pulse-coupled neural network performance for real-time identification of vegetation during forced landing. In *Engineering Mathematics and Applications Conference (EMAC2013)*, vol. 55 of *ANZIAM J.*, pp. C1–C16, 2014. http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/7851 C457

[18] D. J. Warne, N. A. Kelson, and R. F. Hayward. Solving tri-diagonal linear systems using field programmable gate arrays. In *Proceedings of the 4th International Conference on Computational Methods (ICCM2012)*, 2012. http://eprints.qut.edu.au/54894/ C447, C448, C449

[19] D. J. Warne, N. A. Kelson, and R. F. Hayward. Comparison of high level FPGA hardware design for solving tri-diagonal linear systems. *Proc. Comput. Sci.*, 29:95–101, 2014. doi:10.1016/j.procs.2014.05.009 C448, C454

[20] L. Wirbel. *Xilinx SDAccel: A unified development environment for tomorrow's data center.* Technical report, The Linley Group Inc., November 2014. http://www.xilinx.com/publications/prod_mktg/sdnet/sdaccel-wp.pdf C448

[21] G. Wu, Y. Dou, J. Sun and G. D. Peterson. A high performance and memory efficient LU decomposer on FPGAs. *IEEE T. Comput.*, 61:366–378, 2012. doi:10.1109/TC.2010.278 C448

Author addresses

1. **H. J. Macintosh**, School of Electrical Engineering and Computer Science, Queensland University of Technology, Queensland 4001, Australia.
   mailto:hj.macintosh@hdr.qut.edu.au

2. **D. J. Warne**, High Performance Computing and Research Support, Queensland University of Technology, Queensland 4001, Australia.
   mailto:david.warne@qut.edu.au

3. **N. A. Kelson**, High Performance Computing and Research Support, Queensland University of Technology, Queensland 4001, Australia.
   mailto:n.kelson@qut.edu.au

4. **J. E. Banks**, School of Electrical Engineering and Computer Science, Queensland University of Technology, Queensland 4001, Australia.
   mailto:j.banks@qut.edu.au

5. **T. W. Farrell**, School of Mathematical Sciences, Queensland University of Technology, Queensland 4001, Australia.

mailto:t.farrell@qut.edu.au