John von Neumann Institute for Computing

Towards an FPGA Solver for the PageRank Eigenvector Problem

Séamas McGettrick, Dermot Geraghty, Ciarán McElroy

published in

Parallel Computing: Architectures, Algorithms and Applications ,
C. Bischof, M. Bücker, P. Gibbon, G.R. Joubert, T. Lippert, B. Mohr, F. Peters (Eds.),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **38**, ISBN 978-3-9810843-4-4, pp. 793-800, 2007.
Reprinted in: Advances in Parallel Computing, Volume **15**,
ISSN 0927-5452, ISBN 978-1-58603-796-3 (IOS Press), 2008.

http://www.fz-juelich.de/nic-series/volume38

Towards an FPGA Solver for the PageRank Eigenvector Problem

Séamas McGettrick, Dermot Geraghty, and Ciarán McElroy

Dept of Mechanical and Manufacturing Engineering
Trinity College Dublin, Ireland
E-mail: {*mcgettrs, tgerghty, ciaran.mcelroy*}*@tcd.ie*

Calculating the Google PageRank eigenvector is a massive computational problem dominated by Sparse Matrix by Vector Multiplication (SMVM) where the matrix is very sparse, unsymmetrical and unstructured. The computation presents a serious challenge to general-purpose processors (GPP) and the result is a very lengthy computation. Other scientific applications have performance problems with SMVM and FPGA solutions have been proposed. However, the performance of SMVM architectures is highly dependent on the matrix structure. Architectures for finite element analysis, for example, may be highly unsuitable for use with Internet link matrices. An FPGA architecture can bring advantages such as custom numeric units, increased memory bandwidth and parallel processing to bear on a problem and can be deployed as a coprocessor to a GPP.

In this paper, we investigate the SMVM performance on GPPs for Internet matrices. We evaluate the performance of two FPGA based SMVM architectures originally designed for finite element problems on a Virtex 5 FPGA. We also show the effect of matrix reordering on these matrices. We then investigate the possibility of outperforming the GPP using parallelization of processing units, showing that FPGA based SMVM can perform better than SMVM on the GPP.

A Introduction

The Internet is the world's largest document collection. As it contains over 25 billion pages[1] to choose from, finding one desired page can seem a daunting task. Search engines have been designed to wade through the vastness of the Internet and retrieve the most useful documents for any given query. A search engine has to do much work before it can return any results. Long before a user submits a query, a search engine crawls the Internet and gathers information about all the pages that it will later search. These pages are then indexed, so that a list of related pages can be retrieved when a query is submitted. The next step is ranking the pages returned for a query. This can be done either at or before query time. In both cases, this process is handled by a ranking algorithm. Documents with relevant content get a high rank and irrelevant documents receive low ranking scores. In response to a query the search engine displays the results of a search in order of rank.

Before the advent of Google, spammers had found ways to manipulate the existing ranking algorithms. Thus, one often had to search through pages of useless results (Spam) before finding a page with useful information. Google tended to be immune to spam for every query. This improved search engine was made possible by a new ranking algorithm called PageRank. PageRank remains at the heart of the Google search engine to this day[2].

PageRank achieved a higher quality result by taking advantage of the hyperlinked structure of the Internet[2]. Each hyperlink to a page is counted as a vote for the content of the page to which the hyperlink leads. The more votes a page gets, the better its content is

assumed to be, and thus the higher its ranking. The PageRank calculation has been dubbed the largest matrix calculation in the world[3]. The matrix at the centre of the calculation is of the order of 25 billion rows and columns. It is constantly growing as new pages are added to the Google index.

In Section B, the PageRank Algorithm is discussed and the algorithm is explained. We show how the algorithm is essentially a problem in linear algebra. In Section C, some benchmark data for PCs and custom FPGA architectures, which are used in other scientific applications, is presented. These tests show the effects of reordering on Internet matrices. These results are then extrapolated to show the performance achievable using multiple processing units in parallel on a Virtex 5 FPGA. Finally, a reflection on the current work to implement the PageRank algorithm on FPGA and discussion of future progress is presented.

B PageRank Algorithm

The PageRank algorithm was developed by Sergey Brin and Lawrence Page in 1998 at Stanford University[4]. They later created Google, which is currently the world's largest and most used search engine[2]. PageRank is query independent, which means that page rankings are calculated prior to query time. This system allows Google to return query results very quickly. PageRank does not rank in order of relevance to the query, but instead it ranks by general importance of the page. By definition the rank of a page, $r(P_i)$, is calculated from the sum of the ranks of those pages, $r(P_j)$, which point to it, moderated by the number of out-links from each of these pages. Thus,

$$r(P_i) = \sum_{P_j \epsilon B_{P_i}} \frac{r(P_j)}{|P_j|} \tag{B.1}$$

where B_{P_i} is the set of pages which point to page P_i and $|P_j|$ is the number of out-links on page P_j. To begin with all pages are assigned an initial PageRank of 1/n where n is the total number of pages in the network or the indexed part of the network. The equation above can be written in iterative form, as follows:

$$r_{k+1}(P_i) = \sum_{P_j \epsilon B_{P_i}} \frac{r_k(P_j)}{|P_j|} \tag{B.2}$$

where $r_k(P_i)$ is the PageRank of P_i after k iterations. The equation is such that that after a series of iterations of this equation, the PageRank will converge to a stationary value. The PageRank equation for a full system can be more conveniently represented in terms of vectors and matrices. Thus, the graph of the web is represented by an Internet link matrix, H, which defines the interconnects between the pages. Each row in H shows the out-links from a page and each column shows the in-links to a page. When the moderating factor, $|P_j|$ is applied to a row, a probability vector is obtained. i.e. the elements of a row sum to one.

Now the PageRank vector, which is a vector, can be defined, where each element is the rank of the corresponding page, as follows:

$$\pi^{(k+1)T}(P_i) = \pi^{(k)T} H \tag{B.3}$$

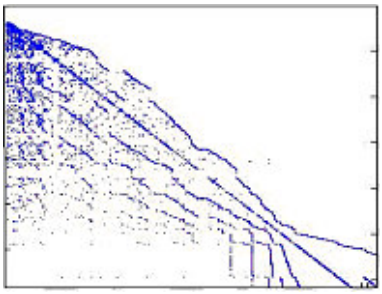

Figure 1. Example Internet Matrix (from a crawl of tcd.ie)

This equation is of the form $\lambda x = Ax$ with $\lambda = 1$. Therefore, it is an eigenvector problem. One of the most straightforward ways of solving this system is the power method. In fact, the above iterative equation is a statement of the power method. The equation clearly shows that the predominant operation needed to solve this problem is Sparse Matrix by Vector Multiplication (SMVM) between the Internet link matrix (H) and the PageRank vector.

C Performance

The matrix used in Google's PageRank calculations is a very large matrix. It has 25 billion rows and columns; this number is constantly growing as Google includes more pages in its search database[5]. The matrix is also unsymmetrical and very sparse; it has, on average, about 10 entries per column[6]. Fig. 1 shows a sample web matrix. Currently PageRank is calculated on general-purpose processors like the Intel and AMD processors. These processors often perform poorly with large SMVM problems due to the lack of structure of the matrix[7]. An FPGA solution would have a number of advantages over the traditional processor approach. An FPGA can have more than one memory channel and so can run the calculation in parallel across multiple processing units. The hardware of the FPGA can be specially optimized to compute ranking algorithms efficiently. To the best of our knowledge, no FPGA solution for search algorithms exists. To better understand how the PageRank algorithm could benefit from being implemented on an FPGA, two FPGA based architectures designed for use in Finite Element problems were benchmarked against a GPP. The field of finite element also deals with large sparse matrices. However, Internet link matrices differ from the matrices used finite element analysis as they are not symmetric or tightly banded[8]. The first of the two FPGA-based Finite Element solvers used was the column based SMVM as described by Taylor et al[9]. The second FPGA-based solver is a tile solver. This architecture is patent pending so no details will be given at this time. Some technical details on the architectures used for these benchmarks are given in Table 1. The models used to calculate the performance of the two FPGA based architectures were verified in RTL on a Virtex II device and extrapolated to Virtex 5[10].

Table 2 details the matrices used. All of these matrices come from the Stanford WebBase project[11] or the WebGraph datasets[12] with the exception of web_trinity, which was generated from a crawl of the Trinity College Website. The matrices vary in size from 2.4

	CPU/FPGA	CLK	PEAK
PC	Pent. Xeon Woodcrest	3 GHz, 1333 MHz FSB	6 GFLOPs
Column Solver	Virtex 5	222 MHz	444 MFLOPs
Tile Solver	Virtex 5	444 MHz (2x222 MHz)	888 MFLOPs

Table 1. Benchmark architecture details

Num.	Name	Nodes	No. of Links
1	arabic-2005_5K	500000	9877485
2	cnr-2000_5K	325557	3216152
3	eu-2005_5K	500000	11615380
4	in-2004_5K	500000	5422294
5	indochina-2004_5K	500000	8147699
6	it-2004_5K	500000	9387335
7	sk-2005_4K	400000	13391888
8	uk-2002_5K	500000	6998368
9	uk-2005_5K	500000	11192060
10	web_matrix_1-5M	1500000	12392081
11	web_matrix_1M	1000000	8686242
12	web_standford	281903	2312497
13	web_trinity	608358	2424439
14	webbase-2001_5K	500000	4214705

Table 2. Benchmark matrices details

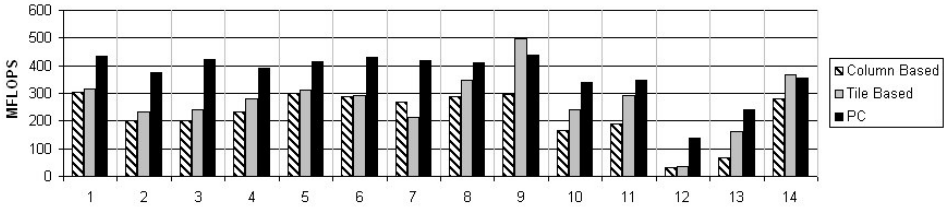

Figure 2. SMVM benchmarks for Internet Link Matrices

million non-zeros to 13.4 million non-zeros. They are much smaller than full size Internet link matrices but are large enough to test the efficiency of an architecture. The PC benchmark was a dual threaded C program. The results of the benchmarking are shown in Fig. 2.

The benchmark tests in Fig. 2 show how a single Processing Element (PE) on the FPGA-based architectures compare to the 3 GHz Intel Xeon (Woodcrest). PageRank matrices have massive storage requirements. In order to achieve a relatively cheap and scalable memory DRAM can be used. The two FPGA solvers used for benchmarking can connect to DDR DRAM. Xilinx have shown that Virtex 5 FPGAs can be connected to DDR2-667[13].

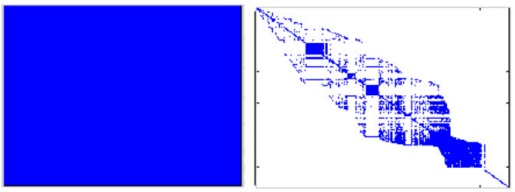

Figure 3. Web_stanford before and after RCM reordering

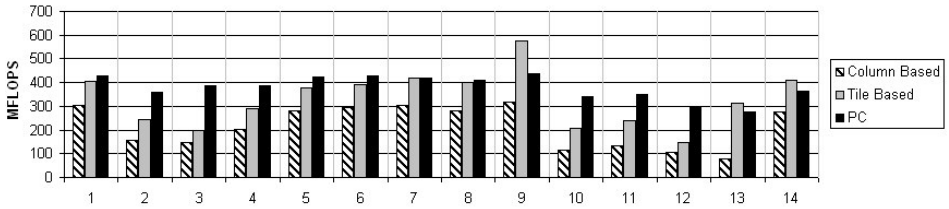

Figure 4. SMVM Benchmark Performance with RCM reordering

Using DDR2-667 RAM a 64 bit word can be read from memory every 667 MHz clock cycle. Each of the two FPGA solvers used for benchmarking requires a 96 bit word every clock cycle (64 bit non-zero entry and a 32 bit address). To maximise the memory bandwidth being used the FPGA solvers would need to run at 444 MHz, which, is unachievable with current FPGA technology. The column based architecture therefore uses a more realistic 222 MHz clock, thus, using only half the available memory bandwidth. The tile solver approaches this problem slightly differently. It internally has two MAC units running at 222 MHz in parallel which allows it to effectively run at 444 MHz. A similar system could be implemented for the column based system. However, to be consistent with Taylor's architecture it was decided to use a single MAC unit.

It is interesting to see in Fig. 2 that the FPGA-based tile solver actually outperforms the PC on two of the matrices (9,14), even though the FPGA clock is running almost 12 times slower. The GPP only achieves between 3% and 6% of its theoretical peak. Matrices 12 and 13 perform poorly on all architectures. These matrices give poor performance for two reasons. They have very few entries per row and their entries are very scattered. The left side of Fig. 3 shows a pictorial view of matrix 12. The matrix is so scattered that the picture of the whole matrix appears dense.

It was decided that some investigation should be done into how a well-known reordering scheme like Reverse Cuthill McKee (RCM)[14] would affect this performance. RCM reordering is used to reduce the bandwidth of link graphs by renumbering the nodes. Fig. 3 shows the difference in non-zero distribution before and after RCM reordering of test matrix 12. The diagram shows that simply renumbering the nodes decreased the bandwidth of this matrix dramatically. The matrices were reordered and benchmarked on the three architectures. The results of the reordering benchmarks are shown in Fig. 4. Fig. 5 shows the change in performance with reordering. Reordering the matrices caused all 3 archi-

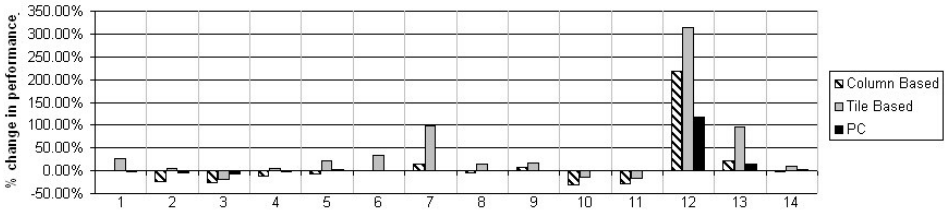

Figure 5. % change in performance with RCM reordering

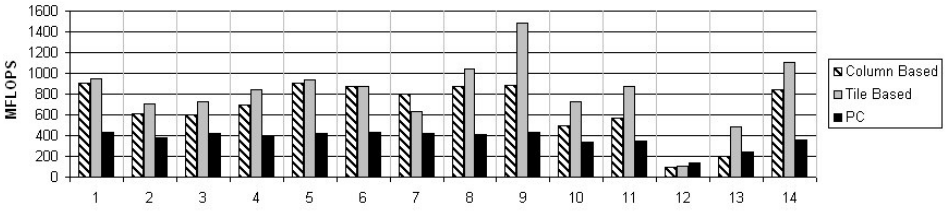

Figure 6. SMVM Benchmark Performance of 3 x PE connected via 3 DDR interfaces

tectures to improve performance on the two poorly performing matrices. Reordering also caused an increase in performance for a number of the other matrices. It also caused slight degradation in performance of others. Matrices 12 and 13 were the only matrices that were not ordered by the spiders that generated them, and thus showed the largest increase in performance when RCM was applied. This highlights the importance of reordering. The reordering could be implemented by the spider and so avoiding an extra reordering stage. The column based FPGA architecture showed a less of an increase in performance with reordering than the tile solver, which can be attributed to RAW hazards which stalls processing[15]. Since these tests use an adder pipeline 14 cycles long[10] and the average column only contains 10 entries, a large number of RAW hazards occur. There are many different types of reordering schemes. RCM is expensive to implement and is probably not the best algorithm for Internet link matrices but it does highlight that reordering is necessary.

D Parallelisation

In the previous section we saw that a single PE could outperform a 3 GHz Pentium Woodcrest processor when calculating SMVM for some, though not all, Internet link matrices. Further performance improvements are possible through the use of parallel PEs and by increasing memory bandwidth by implementing multiple parallel memory ports. In this section we investigate the performance of a system with three parallel PEs and three memory ports. As before the tile solver's PE has a pair of double precision floating point MAC units operating at 222 MHz for an effective rate of 444 MHz. Both solvers show an increase in performance.

Fig. 6 shows that the FPGA-based solutions have the ability to outperform the GPP

by as much as 3 times. This is due to their ability to take advantage of parallelism in the computations. Both FPGA-based solvers outperform the 3 GHz Pentium Woodcrest Xeon for all our test matrices except 12, even though their clock rate is over 13 times slower. The FPGA-based tile solver is the architecture that shows the best performance with a peak performance of almost 1450 MFLOPS. The FPGA-based column ordered solver performs well, but its performance is still badly hampered by RAW hazards as discussed earlier.

E Conclusions

The PageRank eigenvector calculation is a large and computationally intensive matrix problem. General-Purpose processors only achieve a fraction of their peak performance when calculating large Sparse Matrix by Vector products[16]. The GPP has a single pipe to memory and so cannot exploit parallelisms in its calculations. The large number of IO pins on modern FPGAs makes it possible to use them to achieve high memory bandwidth by implementing multiple memory ports and exploiting parallelisms in algorithms. Little performance data for Internet link matrices exists in the public domain and no co-processor architectures specialized for Internet link matrices exist to the best of the authors knowledge. Finite Element Analysis uses large sparse matrices and a number of FPGA based architectures exist for solving these problems. Some of these architectures were benchmarked against a GPP. The benchmark results show a number of interesting results. The FE FPGA based solvers were more efficient than the GPP. They achieved approximately 50% FPU utilisation when computing Internet style matrices. The GPP (3 GHz Intel Xeon (Woodcrest)) only achieved a 3-8% FPU utilisation. This increased efficiency meant that a single FPGA-based SMVM processing element could out-perform the GPP for SMVM of some internet link matrix even though the FPGA's clock rate was over 12 times slower.

A number of the matrices showed very poor performance across the three test architectures. The NZ elements in these matrices were very scattered. RCM , which is a well known reordering scheme, was applied to all matrices. RCM improved the performance of the matrices that performed very poorly by a much as 3 times , but it did not give an across the board increase in performance. RCM is not the right reordering scheme for Internet link matrices. These results highlight the need to further investigate reordering. A suitable reordering scheme has the potential to further increase the performance of an FPGA solver for the PageRank algorithm.

The large number of pins on the FPGA makes it possible implement multiple, parallel memory ports. This allows 3 PE to work in parallel and increases the performance of the FPGA solutions so that they both outperform the GPP for all our test matrices. In one case the tile solver calculated the solution 3 times quicker than the GPP despite having a clock rate 13 times slower. The tile solvers peak performance was measured at 1450 MFLOPS. This superior performance is due to the FPGAs ability to exploit parallelism in the calculation.

The solvers used in these experiments were designed for use with Finite Element matrices. FE calculations are usually done using IEEE floating-point doubles. It is thought that Internet link matrix calculations do not need to be calculated using this precision[6]. Lower precision number formats would allow for greater parallelism as well as a reduction in adder and multiplier latencies. Investigating this possibility is the next step towards designing an architecture for the PageRank eigenvector problem on FPGA. This architec-

ture will need access to large banks of memory and will rely greatly on parallelisation to achieve maximum performance for Internet link matrices.

In this paper we have shown how even FPGA based hardware not optimised for Internet link matrix SMVM can outperform the GPP ,by up to 3 times using slow but cheap and scalable DDR memory, and by using parallel processing units.

Acknowledgements

This work has been funded by Enterprise Ireland's Proof of Concept fund.

References

1. D. Austin, *How Google finds your needle in the web's haystack*, http://www.ams.org/featurecolumn/archive/pagerank.html
2. http://www.google.com/technology/
3. C. Moler, *The World's largest computation*, Matlab news and notes, **Oct.**, 12–13, (2002).
4. S. Brin, L. Page and M. Coudreuse, *The anatomy of a large-scale hypertexual web search engine*, Computer Networks and ISDN systems., **33**, 107–117, (1998).
5. A. N. Langville and C. D. Meyer, *A survey of eigenvector methods for web information retrievel*, The SIAM Review, **27**, 135–161, (2005).
6. A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond, The Science of Search Engine Rankings*, (Princeton University Press, 2006).
7. W. D. Gropp, D. K. Kasushik, D. E. Keyes and B. F. Smith, *Towards realistic bounds for implicit CFD codes*, Proceedings of Parallel Computational Fluid Dynamics, , 241–248, (1999).
8. S. McGettrick, D. Geraghty and C. McElroy, *Searching the Web with an FPGA-based search engine*, ARC-2007, LNCS, **4419**, 350–357, (2007).
9. V. E. Taylor, *Application specific architectures for large finite element applications*, PhD Thesis, Berkeley California, (1991).
10. Xilinx LogiCore, *Floating Point Operators V.3*, (2005). http://www.xilinx.com
11. http://dbpubs.stanford.edu:8091/ testbed/doc2/WebBase/
12. P. Boldi and S. Vigna, *The WebGraph framework 1: Compression techniques*, in: WWW-2004, pp. 595–601, (ACM Press, 2004).
13. K. Palanisamy and M. George, *High-Performance DDR2 SDRAM Interface in Virtex-5 Devices*, Virtex 5 Application notes, XAPP858 (v1.1), (2007).
14. http://people.scs.fsu.edu/ burkardt/f_src/rcm/rcm.html
15. C. Mc Sweeney, D. Gregg, C. McElroy, F. O'Connor, S.McGettrick, D. Moloney and D. Geraghty, *FPGA based Sparse Matrix Vector Multiplication using commodity memory*, FPL, (2007).
16. C. Mc Sweeney, *An FPGA accelerator for the iterative solution of sparse linear systems*, MSc Thesis, Trinity College Dublin, Ireland, (2006).