The Effect of First-Level Cache Improvements on the RAMpage Memory Hierarchy

Philip Machanick and Zunaid Patel
School of Computer Science
University of the Witwatersrand
{*philip,zunaid*}*@cs.wits.ac.za*

Abstract

The RAMpage memory hierarchy is an alternative memory organization which addresses the growing CPU-DRAM speed gap, by replacing the lowest-level cache by an SRAM main memory. This paper presents some modifications to the RAMpage hierarchy. More aggressive first level cache implementations are shown to improve performance of the RAMpage model, when context switches were taken on misses to DRAM.

1 INTRODUCTION

The RAMpage model has been proposed [11, 10] as an alternative to the conventional memory hierarchy model of one or more levels of cache, with a DRAM main memory and disk-based backing store for the virtual memory system. In the RAMpage model, the lowest-level cache is replaced by an SRAM main memory, and DRAM becomes the first-level paging device, which is backed by a second-level disk paging device.

This paper investigates a wider range of first-level cache (L1) variations than in previous work. Larger L1 caches are investigated, as well as variations in block (line) size and associativity.

Earlier work used a relatively small L1 cache and did not investigate variations in L1 organization. While it could be argued that for any reasonable workload, DRAM traffic (which would not vary significantly across L1 variations) is the variable of greatest interest when designing the lowest-level cache, it is important to be sure that results are not a consequence of some artifact of details of a specific L1 design.

As with previous work, trace-based simulation is used. The same SPEC92-based traces are used; as in previous work, simulating a multiprogramming workload by interleaving sections of these traces is sufficient to exercise the simulated memory hierarchy. In this paper, both the instruction (L1i) and data (L1d) caches varied from 16Kbytes to 256KB. The block size was varied from 32 to 128 bytes, and associativity varied up to 8-way.

The results show that RAMpage retains a clear advantage across these variations, and that improvements to the L1 organization generally result in a stronger improvement in the RAMpage model with context switches on misses than for a conventional 2-level cache.

The remainder of this paper starts with a brief survey of related work. Section 3 provides a little more detail of the approach, and results are presented in Section 4. Finally, Section 5 sums up the findings and outlines future work.

2 Related Work

2.1 Introduction

Since the mid-1980s, CPU speeds have been improving at an average of 55% per year, while DRAM latency has only been improving at 7% per year [5]. This growing speed gap has resulted in the view that we are approaching a "memory wall", in which future speed improvement will be dictated by DRAM speed improvement [13].

Caches have traditionally been used to bridge the CPU-DRAM speed gap. This section summarizes some improvements to the design of caches, as well as attempts at hiding basic latency of DRAM by more sophisticated organization. To place the RAMpage strategy in context, other software-based strategies are reviewed.

2.2 Improvements to Caches

Adding associativity makes it more difficult to achieve fast hits, while reducing the number of misses; increasing block size reduces misses while making each miss more expensive [5]. Another improvement which can reduce misses without making it harder to achieve fast hits is a *victim cache*, a small additional cache of recently replaced blocks [7].

Others have investigated alternatives to full associativity, including column-associative caches [1], which allow an alternative location for a block without the overhead of associativity in the best case.

Approaches to implementing associativity cheaply can be considered to be competing alternatives. However, increasing associativity only defers the memory wall.

A comparable approach to taking a context switch on a miss is simultaneous multithreading (SMT) [12]. SMT

could in principle be implemented on top of RAMpage.

2.3 Improvements to DRAM

While the underlying trend in cycle time for DRAM remains unchanged, it is possible to hide some of the latency by exploiting typical reference patterns. For example, a cache miss – the most common way of referencing DRAM from the CPU – references multiple sequential bytes. Since it is easier to achieve high bandwidth than low latency, both in general and specifically for DRAM [5], DRAM designers have offered various modes which allow multiple sequential accesses to be faster on average than a single random access. These include synchronous DRAM (SDRAM) and Rambus [4].

While latency is a key problem, bandwidth off chip is another significant issue which limits speed of transfers between DRAM and the faster parts of the memory hierarchy [9]. One approach to addressing this problem is to include DRAM on the processor chip – the IRAM (for "intelligent" RAM) approach [8].

RAMpage can exploit these improvements. Bigger SRAM pages benefit from the streaming properties of SDRAM and Rambus, and an IRAM system could page fault off-chip to a bigger DRAM.

2.4 Software-Based Approaches

Designers of both DRAM and cache hierarchies have mostly focused efforts on hardware-based improvements. Changes in cache architecture can generally be hidden from the operating system, since the cache is managed in hardware. The same is true of most variants on DRAM.

Software-based approaches on the other hand may require operating system modification. In some cases, the software may be hidden in architecture extensions (e.g., PALcode in the Alpha processor [2]).

In the 1980s, there was some work on software-based cache management, with emphasis on reduction of misses in a shared-memory system [3]. More recent work on managing the interface between cache and DRAM in software has focused on address translation [6].

These other software-based approaches have not gone as far as treating the lowest level of cache as a fully software-managed paged memory.

3 Experimental Approach

As with previous work, trace-driven simulation is used to compare a RAMpage machine with a 2-level cache hierarchy, in which both L2 in the conventional hierarchy and the RAMpage SRAM main memory are 4Mbytes.

Configurations start from previous work [11, 10], in which instruction issue rates represent speeds of current designs which issue multiple instructions per cycle. Pipeline stalls for causes other than cache misses are not simulated, so an issue rate (for example) of 8 GHz could represent a 1 GHz processor which issues 8 instructions per clock, or a 2 GHz processor which issues 4 instructions per clock.

Simulations were carried out on three different hierarchy configurations, specifically RAMpage with and without context switches on misses and a standard or conventional 2-level cache hierarchy. Several trade-offs within the first-level cache were investigated for each hierarchy. Accordingly, each of the following sections presents the results of a particular tradeoff explored.

What we expect to see is that, as L1 improves, the gap between any variations in L2 should be less significant. However, overall, the fraction of time spent in DRAM should be higher, since L1 is faster than L2, and total DRAM references should not vary significantly (since inclusion is maintained between L1 and L2). Given that the time spent in DRAM would otherwise be higher, context switches on misses should be a greater win as L1 improves.

4 Results

4.1 Introduction

This section presents results and highlights important findings, particularly the way in which RAMpage with context switches on misses gains most from improvements to the L1 cache.

The following section presents results obtained by varying the size of the L1-cache and the instruction issue rate. Section 4.3 examines the effect of varying the block or line size across the different cache sizes. Section 4.4 shows the effect of more associative L1 implementations.

4.2 L1 size variation

L1i and L1d cache sizes were varied from 16 KB to 256 KB each, resulting in total cache sizes from 32 KB to 512 KB. A block size of 32 bytes and direct-mapping was used throughout, while instruction issue rates were varied from 1 GHz to 8 GHz.

As predicted in Section 3, the fraction of time spent in DRAM increases as L1 hits increase.

Figures 1 to 6 highlight this trend. Also, as predicted in Section 3, as total L1 size is increased, the proportion of execution time in L1 increases for every issue rate shown. (TLB and L1d hits are fully pipelined, so they only account for a small fraction of total execution time.)

Comparing issue rates, the standard hierarchy spends just under 10% of execution time in the DRAM level at 1 GHz (Figure 1), while it spends about 40% of execution time in DRAM with an issue rate of 8 GHz (Figure 2).

However, the RAMpage hierarchy with no context switches on misses fares only slightly better as Figures 3 and 4 show. Only when context switches are taken on misses to DRAM are the benefits of the RAMpage approach realised. Under 1% of total execution time is spent at the DRAM level for a 1 GHz issue rate (Figure 5) and this amount does not rise as the instruction issue rate is pushed up to 8 GHz (Figure 6).

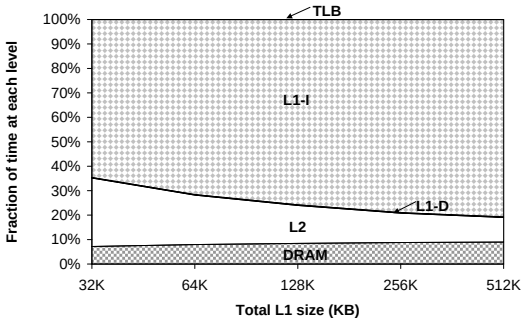


Figure 1: Standard Hierarchy: 1 GHz Issue rate. *Fraction of time spent in each level of the hierarchy.*

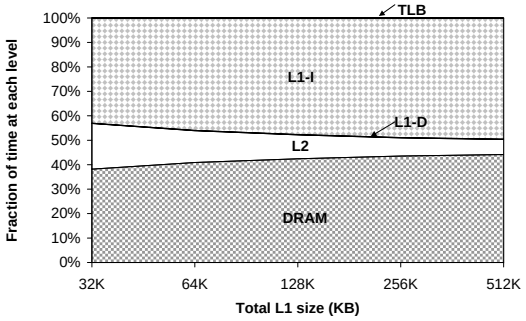


Figure 2: Standard Hierarchy: 8 GHz issue rate

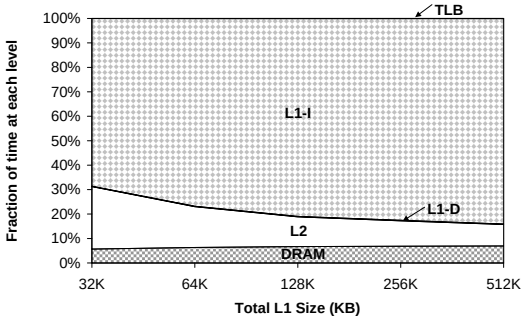


Figure 3: RAMpage hierarchy, no context switches on misses: 1 GHz issue rate

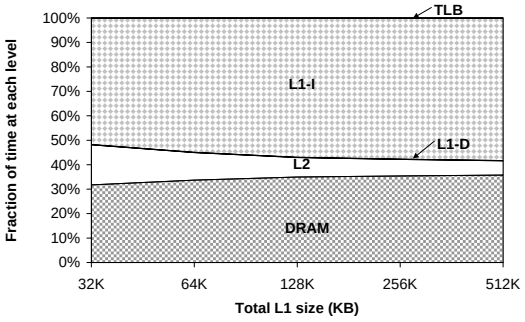


Figure 4: RAMpage hierarchy, no context switches on misses: 8 GHz issue rate

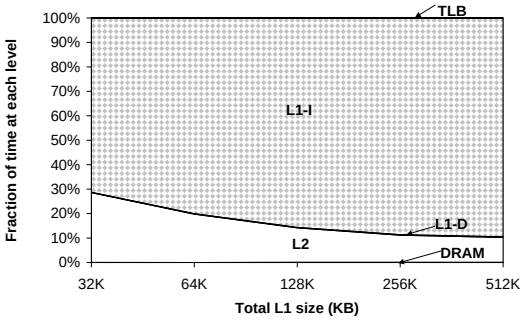


Figure 5: RAMpage hierarchy with context switches on misses: 1 GHz issue rate

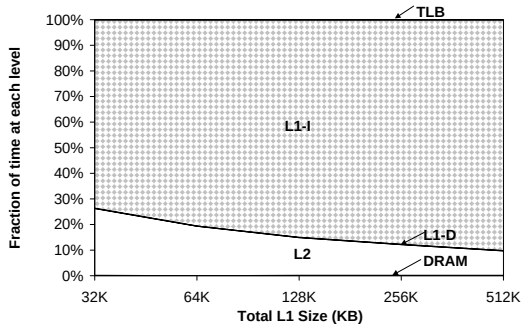


Figure 6: RAMpage hierarchy with context switches on misses: 8 GHz issue rate

As expected, larger caches decrease execution times due to the reduction in capacity misses. Table 1 shows simulated execution times. However, improvements in performance decrease as cache sizes get larger.

All of these observations are as predicted in Section 3: as L1 improves, in the limit, all references which would otherwise be found in L2 will be found in L1.

L1i and L1d Size	1 GHz	2 GHz	4 GHz	8 GHz
16 KB	1.322	0.709	0.401	0.248
	1.276	0.670	0.370	0.220
	1.229	0.623	0.302	0.152
32 KB	1.193	0.644	0.370	0.232
	1.141	0.619	0.346	0.207
	1.094	0.556	0.276	0.139
64 KB	1.127	0.611	0.353	0.224
	1.082	0.592	0.330	0.200
	1.022	0.516	0.258	0.131
128 KB	1.082	0.588	0.342	0.218
	1.061	0.580	0.325	0.197
	0.988	0.508	0.248	0.128
256 KB	1.058	0.577	0.336	0.216
	1.042	0.571	0.320	0.195
	0.978	0.489	0.245	0.124

Table 1: Cache size *vs.* issue rate. *Each row shows simulated execution times (s) for standard hierarchy (top), RAMpage with no context switches on misses (middle) and RAMpage with context switches on misses (bottom).*

4.3 Variations in L1 Block Size

Variations in L1 block (line) size present no surprises. Given the observations from increases in cache size, it is no surprise that the biggest improvements seen are with RAMpage with context switches on misses.

Times in Table 2 show significant improvements for RAMpage with context switches on misses (e.g., in the 8 GHz case, about 18% for the best case, 32byte L1 blocks) as L1 improves. By contrast, the improvement for RAMpage with no context switches on misses (Table 3) is under 12% for the best case at 8 GHz, compared with almost 14% for the best case of the conventional hierarchy (Table 4).

4.4 Associativity-speed Tradeoff

We should expect to see similar results with increasing associativity: the gap between the conventional and RAMpage hierarchy without context switches on misses should decrease, but the advantage in context switches on misses should increase. These effects can be seen in Tables 5 to 7.

Tables 5 shows how RAMpage with context switches on misses improves with more associativity in L1. Again, consider 8 GHz. The improvement in going from a direct-mapped 16KB cache to an 8-way associative 256KB cache (combined capacity of L1i and L1d) is 19%.

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
32 bytes	1.229	0.623	0.302	0.152
64 bytes	1.348	0.701	0.347	0.171
128 bytes	1.731	0.855	0.438	0.213
32 KB				
32 bytes	1.094	0.556	0.276	0.139
64 bytes	1.144	0.611	0.290	0.148
128 bytes	1.425	0.655	0.347	0.162
64 KB				
32 bytes	1.022	0.516	0.258	0.131
64 bytes	1.094	0.522	0.265	0.133
128 bytes	1.167	0.554	0.277	0.140
128 KB				
32 bytes	0.988	0.508	0.248	0.128
64 bytes	0.998	0.495	0.250	0.127
128 bytes	1.026	0.512	0.261	0.131
256 KB				
32 bytes	0.978	0.489	0.245	0.124
64 bytes	0.964	0.486	0.244	0.127
128 bytes	0.980	0.492	0.244	0.124

Table 2: RAMpage Hierarchy variations by block size. *Context switches on misses.*

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
32 bytes	1.276	0.670	0.370	0.220
64 bytes	1.398	0.716	0.392	0.232
128 bytes	1.725	0.839	0.451	0.263
32 KB				
32 bytes	1.141	0.619	0.346	0.207
64 bytes	1.189	0.641	0.356	0.213
128 bytes	1.356	0.705	0.391	0.229
64 KB				
32 bytes	1.082	0.592	0.330	0.200
64 bytes	1.089	0.592	0.330	0.200
128 bytes	1.140	0.614	0.340	0.206
128 KB				
32 bytes	1.061	0.580	0.325	0.197
64 bytes	1.054	0.576	0.323	0.196
128 bytes	1.076	0.583	0.327	0.199
256 KB				
32 bytes	1.042	0.571	0.320	0.195
64 bytes	1.027	0.564	0.317	0.194
128 bytes	1.037	0.568	0.318	0.194

Table 3: RAMpage Hierarchy variations by block size. *No context switches on misses.*

The comparable figure for RAMpage without context switches on misses (Table 6) is again under 12%, while that for the conventional hierarchy (Table 7) is almost 14%.

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
32 bytes	1.322	0.709	0.401	0.248
64 bytes	1.457	0.776	0.436	0.265
128 bytes	1.805	0.950	0.523	0.309
32 KB				
32 bytes	1.193	0.644	0.370	0.232
64 bytes	1.256	0.676	0.385	0.240
128 bytes	1.449	0.772	0.434	0.264
64 KB				
32 bytes	1.127	0.611	0.353	0.224
64 bytes	1.147	0.621	0.358	0.226
128 bytes	1.235	0.665	0.380	0.238
128 KB				
32 bytes	1.082	0.588	0.342	0.218
64 bytes	1.078	0.587	0.341	0.218
128 bytes	1.112	0.604	0.350	0.222
256 KB				
32 bytes	1.058	0.577	0.336	0.216
64 bytes	1.047	0.571	0.333	0.214
128 bytes	1.061	0.578	0.337	0.216

Table 4: Standard Hierarchy variations by block size.

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
1-way	1.229	0.623	0.302	0.152
2-way	1.120	0.563	0.281	0.142
4-way	1.099	0.551	0.275	0.138
8-way	1.069	0.537	0.269	0.137
32 KB				
1-way	1.094	0.556	0.276	0.139
2-way	1.040	0.524	0.262	0.132
4-way	1.033	0.516	0.258	0.130
8-way	1.028	0.514	0.258	0.130
64 KB				
1-way	1.022	0.516	0.258	0.131
2-way	0.998	0.501	0.250	0.127
4-way	0.994	0.497	0.248	0.126
8-way	0.991	0.498	0.248	0.126
128 KB				
1-way	0.988	0.508	0.248	0.128
2-way	0.978	0.489	0.246	0.124
4-way	0.974	0.487	0.244	0.124
8-way	0.976	0.486	0.243	0.124
256 KB				
1-way	0.978	0.489	0.245	0.124
2-way	0.969	0.481	0.241	0.123
4-way	0.963	0.482	0.240	0.123
8-way	0.963	0.481	0.240	0.123

Table 5: RAMpage (L1 associativity). *Switches on misses.*

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
1-way	1.276	0.670	0.370	0.220
2-way	1.172	0.626	0.348	0.209
4-way	1.145	0.613	0.341	0.206
8-way	1.136	0.610	0.340	0.205
32 KB				
1-way	1.141	0.619	0.346	0.207
2-way	1.093	0.593	0.331	0.201
4-way	1.083	0.589	0.330	0.200
8-way	1.082	0.588	0.329	0.199
64 KB				
1-way	1.082	0.592	0.330	0.200
2-way	1.060	0.580	0.325	0.197
4-way	1.057	0.578	0.324	0.197
8-way	1.056	0.578	0.324	0.197
128 KB				
1-way	1.061	0.580	0.325	0.197
2-way	1.045	0.573	0.321	0.196
4-way	1.042	0.572	0.321	0.195
8-way	1.042	0.572	0.321	0.195
256 KB				
1-way	1.042	0.571	0.320	0.195
2-way	1.034	0.568	0.319	0.194
4-way	1.032	0.567	0.318	0.194
8-way	1.032	0.567	0.318	0.194

Table 6: RAMpage (L1 associativity). *No context switches on misses.*

5 Conclusions

Improving L1 reduces the importance of L2, and doesn't addess the memory wall. A more aggressive L1 reduces the fraction of time spent in L2, but not actual time spent in DRAM. In fact, if L1 is more effective (to the limit of L2's effect), the fraction of time spent in DRAM increases.

Improvements to L1 reduce the value of RAMpage without context switches on misses, but increase the value of RAMpage with context switches on misses, which matches the finding of previous work, that context switches on misses are the most promising aspect of RAMpage.

The effect of TLB improvements is our next evaluation. Once a full range of measurements is complete, a more comprehensive simulation is planned. Finally, building a RAMpage machine is planned.

Given the growing scale of the memory wall problem, RAMpage is an approach showing increasing promise.

Acknowledgements

This work has been supported financially by the National Research Foundation and the University of the Witwatersrand's University Research Committee.

L1i and L1d	1 GHz	2 GHz	4 GHz	8 GHz
16 KB				
1 way	1.322	0.709	0.401	0.248
2-way	1.205	0.650	0.373	0.234
4-way	1.178	0.637	0.366	0.230
8-way	1.149	0.622	0.358	0.227
32 KB				
1-way	1.193	0.644	0.370	0.232
2-way	1.132	0.614	0.354	0.225
4-way	1.114	0.604	0.350	0.222
8-way	1.108	0.602	0.348	0.222
64 KB				
1-way	1.127	0.611	0.353	0.224
2-way	1.082	0.589	0.342	0.218
4-way	1.076	0.586	0.340	0.218
8-way	1.073	0.584	0.339	0.217
128 KB				
1-way	1.082	0.588	0.342	0.218
2-way	1.062	0.579	0.337	0.216
4-way	1.058	0.577	0.336	0.215
8-way	1.057	0.576	0.335	0.215
256 KB				
1-way	1.058	0.577	0.336	0.216
2-way	1.048	0.571	0.333	0.214
4-way	1.047	0.571	0.333	0.214
8-way	1.046	0.571	0.333	0.214

Table 7: Standard Hierarchy (L1 associativity).

References

[1] A. Agarwal and S.D. Pudar. Column-associative caches: A technique for reducing the miss rate of direct mapped caches. In *Proc. 20th Int. Symp. on Computer Architecture (ISCA '93)*, pages 179–190, May 1993.

[2] Gorden Bell and W. D. Strecker. Retrospective: what have we learned from the PDP-11–what we have learned from VAX and Alpha. In *25 years of the international symposia on Computer architecture (selected papers)*, pages 6–10, New York, NY, 1998. ACM Press.

[3] D.R. Cheriton, G. Slavenburg, and P. Boyle. Software-controlled caches in the VMP multiprocessor. In *Proc. 13th Int. Symp. on Computer Architecture (ISCA '86)*, pages 366–374, Tokyo, June 1986.

[4] V. Cuppu, B. Jacob, B. Davis, and T. Mudge. Performance comparison of contemporary DRAM architectures. In *Proc. 26th Annual Int. Symp. on Computer Architecture*, pages 222–233, Atlanta, Georgia, May 1999.

[5] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kauffmann, San Francisco, CA, 3rd edition, 2002. in press.

[6] B. Jacob and T. Mudge. Software-managed address translation. In *Proc. Third Int. Symp. on High-Performance Computer Architecture*, pages 156–167, San Antonio, Texas, February 1997.

[7] N.P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proc. 17th Int. Symp. on Computer Architecture (ISCA '90)*, pages 364–373, May 1990.

[8] C.E. Kozyrakis, S. Perissakis, D. Patterson, T. Anderson, K. Asanović, N. Cardwell, R. Fromm, J Golbus, B. Gribstad, K. Keeton, R. Thomas, N. Treuhaft, and K. Yelick. Scalable processors in the billion-transistor era: IRAM. *Computer*, 30(9):75–78, September 1997.

[9] Wei-Fen Lin, Steven Reinhardt, and Doug Burger. Reducing DRAM latencies with an integrated memory hierarchy design. In *Proc. 7th Int. Symp. on High Performance Computer Architecture (HPCA-7)*, pages 78–89, Monterrey, Mexico, January 2001.

[10] P. Machanick and P. Salverda. Scalability of the RAMpage memory hierarchy. *South African Computer Journal*, (25):68–73, August 2000.

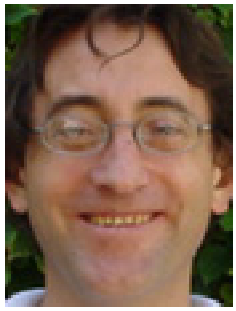
[11] P. Machanick, P. Salverda, and L. Pompe. Hardware-software trade-offs in a Direct Rambus implementation of the RAMpage memory hierarchy. In *Proc. 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, pages 105–114, San Jose, CA, October 1998.

[12] C.A. Waldspurger and W.E. Weihl. Register relocation: flexible contexts for multithreading. In *Proc. 20th Annual Int. Symp. on Computer architecture (ISCA '93)*, pages 120–130, San Diego, CA, May 1993.

[13] W.A. Wulf and S.A. McKee. Hitting the memory wall: Implications of the obvious. *Computer Architecture News*, 23(1):20–24, March 1995.

Authors

Principal Author: Philip Machanick holds a BSc degree from University of Natal, Durban, and BSc Hons and MSc degrees from the University of the Witwatersrand, and a PhD from University of Cape Town. He is a senior member of the IEEE, a member of ACM and fellow of SAICSIT. He is currently a senior lecturer in the School of IT and Electrical Engineering at the University of Queensland, Australia.



Co-Author: Zunaid Patel is a Masters student in the School of Computer Science, University of the Witwatersrand. He holds BSc and BSc Honours degrees from the University of the Witwatersrand.