Int. J. Engng Ed. Vol. 14, No. 2, p. 136–144, 1998
Printed in Great Britain.

0949-149X/91 $3.00+0.00

Simulation of Cache-based Parallel Processing Systems using Spreadsheets*

HASSAN DIAB
Department of Electrical and Computer Engineering, Faculty of Engineering and Architecture, American University of Beirut, P.O. Box 11-0236, Beirut, Lebanon. E-mail: diab@aub.edu.lb

In this paper, a spreadsheet is used for the performance analysis of cache-based multiprocessors for general-purpose computing. The Lotus 1-2-3 spreadsheet is used to study the behavior of the cache miss ratio and the bus bandwidth with respect to the cache line size. The simulation is characterized by its low cost, flexibility and simplicity. The suitability of this tool for educational purposes and its use in an advanced computer architecture course are also discussed.

INTRODUCTION

FOR THE PAST six years, the author of this paper has taught an advanced graduate-level course in computer architecture. During that period, the author observed that while most students comprehend the basic concepts underlying cache-based multiprocessor systems, they tend to have problems relating the effect of the various parameters such as cache size, block size, mapping technique, placement policy, and update policy to the miss ratio and general performance of such systems. Accordingly, the author decided that an accurate, low-cost, user-friendly simulator for the performance evaluation of cache memory systems would be very helpful for students. It is expected that students have taken the Computer Architecture pre-requisite course covering all aspects of the design of von-Neumann machines.

Simulators have been used over the past years for educational purposes such as that devised by Cutler and Eckert [1], Yen and Kim [2], Smith [3], Bic [4], Diab and Demashkieh [5], and Purvis *et al.* [6]. Furthermore, cache memory performance has been extensively researched [7–11].

The use of spreadsheet programs in solving engineering problems has proven to be an important tool for PC users who are not expert in computer programming. This provides the user with an easy environment to interact with for the simulation of specific systems. Some of the application areas include an educational tool for microprocessor systems [12], linear programming [13], analog computer simulation [14], causal filters simulation [15], high-level mixed-mode simulation [16], conditional looping [17], and partial differential equations [18].

* Accepted 5 January 1997.

SIMULATION OF CACHE-BASED SYSTEMS

The overall cache size and the cache line (block) size are the parameters that most strongly affect cache performance. Excessively large or small line sizes can raise the miss ratio; also large line sizes have long transfer times that can lead to high levels of memory traffic. Some of the factors, related to the machine architecture, that influence the line size include [7]:

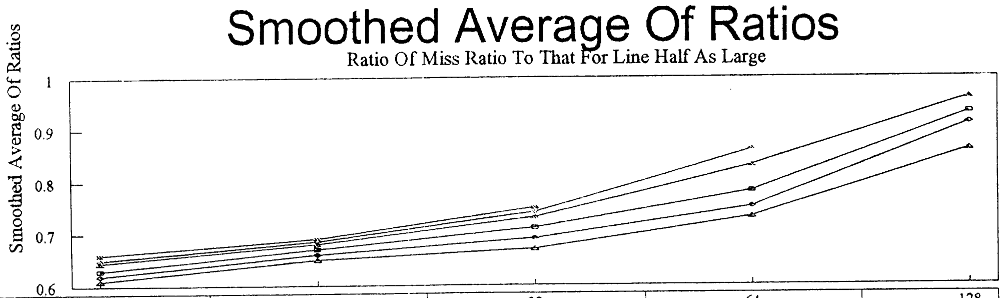
(a) width of the memory modules and degree of interleaving;
(b) bus protocol;
(c) memory interference and memory busy time;
(d) the amount of storage required to hold the address tags when the line size is small;
(e) line crossers between cache lines.

The major effect of line size choice on the performance comes from its impact on the miss ratio.

In Smith [7], trace-driven simulation was used to generate miss ratios for all cache sizes. A demand fetch, fetch on write, copy-back cache with LRU replacement and a fully associative cache was assumed. Due to the high variability of the miss ratios, the relative change in the miss ratio (r) as a function of line size provides a more stable measure than the miss ratio. Accordingly, r is computed as the ratio of the miss ratio for a particular cache size and line size to that for the same cache size but half the line size. In order to smooth out the irregularities of r, the smoothed r (R) is computed [7]. As a result, Figs 1–6 provide the R values as a function of the cache size and line size. Figures 1 and 2 are for instruction cache (miss ratio for instructions only). Figures 3 and 4 are for data cache (miss ratio for data only). Figures 5 and 6 are for unified cache (instructions and data). Figures 1–6 act as the input to the package to provide the user with values to use for estimating the performance impact of certain design choices.

	Instruction Cahe Size										
Line Size	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.660	0.650	0.645	0.630	0.620	0.610	0.600	0.595	0.581	0.577	0.573
16	0.690	0.685	0.680	0.670	0.660	0.650	0.640	0.620	0.600	0.585	0.578
32	0.749	0.740	0.730	0.710	0.690	0.670	0.650	0.630	0.610	0.589	0.581
64		0.860	0.830	0.780	0.750	0.730	0.701	0.680	0.640	0.620	0.590
128			0.960	0.931	0.910	0.860	0.832	0.750	0.690	0.657	0.634

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



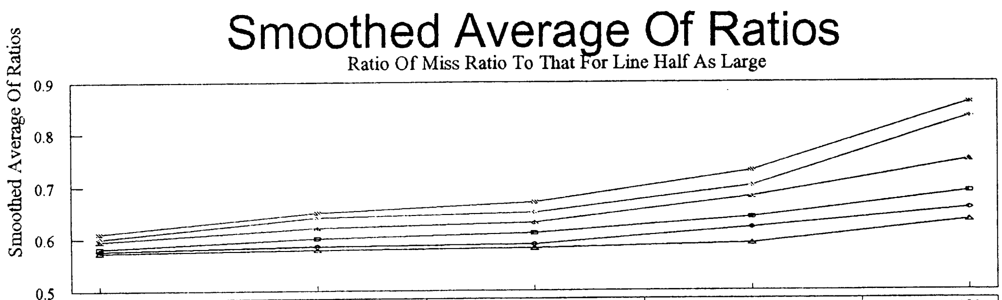
Line Size	8	16	32	64	128
32	0.660	0.690	0.749		
64	0.650	0.685	0.740	0.860	
128	0.645	0.680	0.730	0.830	0.960
256	0.630	0.670	0.710	0.780	0.931
512	0.620	0.660	0.690	0.750	0.910
1024	0.610	0.650	0.670	0.730	0.860

Instruction Cache
Small Cache Size

Fig. 1. R vs. line size, for instruction cache (cache size = 32 → 1024 bytes).

	Instruction Cahe Size										
Line Size	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.660	0.650	0.645	0.630	0.620	0.610	0.600	0.595	0.581	0.577	0.573
16	0.690	0.685	0.680	0.670	0.660	0.650	0.640	0.620	0.600	0.585	0.578
32	0.749	0.740	0.730	0.710	0.690	0.670	0.650	0.630	0.610	0.589	0.581
64		0.860	0.830	0.780	0.750	0.730	0.701	0.680	0.640	0.620	0.590
128			0.960	0.931	0.910	0.860	0.832	0.750	0.690	0.657	0.634

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



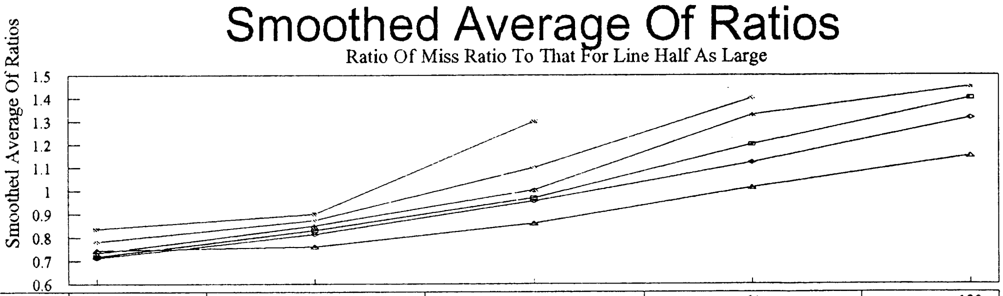
Line Size	8	16	32	64	128
1024	0.610	0.650	0.670	0.730	0.860
2048	0.600	0.640	0.650	0.701	0.832
4096	0.595	0.620	0.630	0.680	0.750
8192	0.581	0.600	0.610	0.640	0.690
16384	0.577	0.585	0.589	0.620	0.657
32768	0.573	0.578	0.581	0.590	0.634

Instruction Cache
Large Cache Size

Fig. 2. R vs. line size, for instruction cache (cache size = 1024 → 32 768 bytes).

Line Size	Data Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.836	0.781	0.734	0.718	0.712	0.744	0.660	0.619	0.603	0.602	0.602
16	0.900	0.873	0.850	0.830	0.814	0.760	0.711	0.654	0.620	0.618	0.618
32	1.300	1.100	1.004	0.970	0.956	0.860	0.787	0.700	0.667	0.646	0.630
64		1.400	1.328	1.200	1.122	1.015	0.880	0.770	0.723	0.672	0.662
128			1.450	1.400	1.314	1.150	1.071	0.900	0.817	0.747	0.697

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



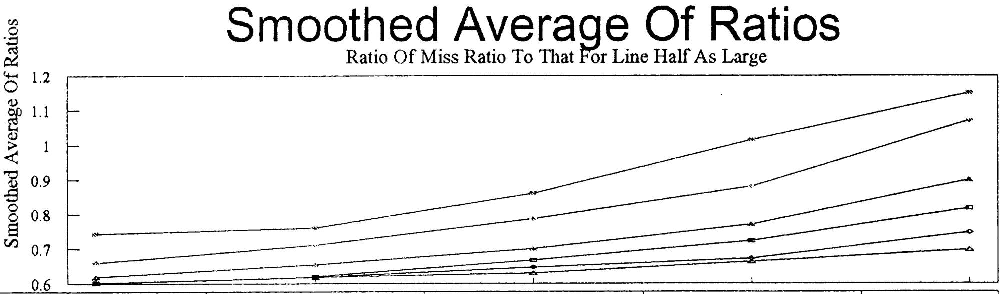
Line Size	8	16	32	64	128
32	0.836	0.900	1.300		
64	0.781	0.873	1.100	1.400	
128	0.734	0.850	1.004	1.328	1.450
256	0.718	0.830	0.970	1.200	1.400
512	0.712	0.814	0.956	1.122	1.314
1024	0.744	0.760	0.860	1.015	1.150

Data Cache
Small Cache Size

Fig. 3. R vs. line size, for data cache (cache size = 32 → 1024 bytes).

Line Size	Data Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.836	0.781	0.734	0.718	0.712	0.744	0.660	0.619	0.603	0.602	0.602
16	0.900	0.873	0.850	0.830	0.814	0.760	0.711	0.654	0.620	0.618	0.618
32	1.300	1.100	1.004	0.970	0.956	0.860	0.787	0.700	0.667	0.646	0.630
64		1.400	1.328	1.200	1.122	1.015	0.880	0.770	0.723	0.672	0.662
128			1.450	1.400	1.314	1.150	1.071	0.900	0.817	0.747	0.697

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



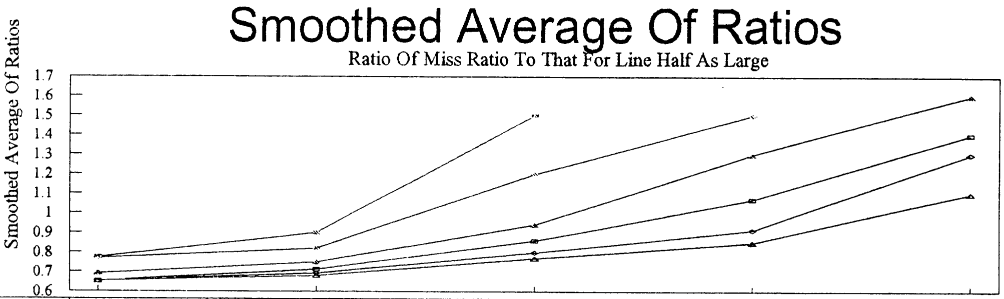
Line Size	8	16	32	64	128
1024	0.744	0.760	0.860	1.015	1.150
2048	0.660	0.711	0.787	0.880	1.071
4096	0.619	0.654	0.700	0.770	0.900
8192	0.603	0.620	0.667	0.723	0.817
16384	0.602	0.618	0.646	0.672	0.747
32768	0.602	0.618	0.630	0.662	0.697

Data Cache
Large Cache Size

Fig. 4. R vs. line size, for data cache (cache size = 1024 → 32 768 bytes).

Line Size	Unified Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.775	0.771	0.692	0.653	0.654	0.653	0.636	0.586	0.581	0.569	0.564
16	0.900	0.820	0.750	0.714	0.693	0.680	0.660	0.662	0.593	0.581	0.575
32	1.500	1.200	0.942	0.860	0.800	0.770	0.731	0.680	0.630	0.600	0.590
64		1.500	1.300	1.070	0.914	0.850	0.787	0.720	0.661	0.633	0.601
128			1.600	1.400	1.300	1.100	0.950	0.850	0.753	0.685	0.660

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



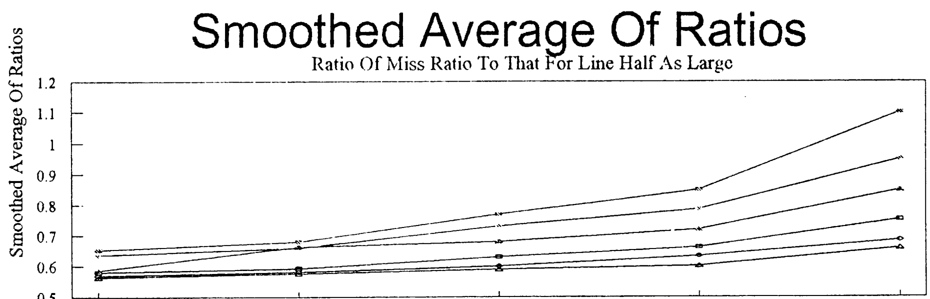
Line Size	8	16	32	64	128
32	0.775	0.900	1.500		
64	0.771	0.820	1.200	1.500	
128	0.692	0.750	0.942	1.300	1.600
256	0.653	0.714	0.860	1.070	1.400
512	0.654	0.693	0.800	0.914	1.300
1024	0.653	0.680	0.770	0.850	1.100

Unified Cache
Small Cache Size

Fig. 5. *R* vs. line size, for unified cache (cache size = 32 → 1024 bytes).

Line Size	Unified Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
8	0.775	0.771	0.692	0.653	0.654	0.653	0.636	0.586	0.581	0.569	0.564
16	0.900	0.820	0.750	0.714	0.693	0.680	0.660	0.662	0.593	0.581	0.575
32	1.500	1.200	0.942	0.860	0.800	0.770	0.731	0.680	0.630	0.600	0.590
64		1.500	1.300	1.070	0.914	0.850	0.787	0.720	0.661	0.633	0.601
128			1.600	1.400	1.300	1.100	0.950	0.850	0.753	0.685	0.660

Smoothed Average Of Ratios
Ratio Of Miss Ratio To That For Line Half As Large



Line Size	8	16	32	64	128
1024	0.653	0.680	0.770	0.850	1.100
2048	0.636	0.660	0.731	0.787	0.950
4096	0.586	0.662	0.680	0.720	0.850
8192	0.581	0.593	0.630	0.661	0.753
16384	0.569	0.581	0.600	0.633	0.685
32768	0.564	0.575	0.590	0.601	0.660

Unified Cache
Large Cache Size

Fig. 6. *R* vs. line size, for unified cache (cache size = 1024 → 32 768 bytes).

SPREADSHEET IMPLEMENTATION

The worksheets were created under the Lotus 1-2-3 for Windows. Lotus 1-2-3 will run under MS Windows on any IBM PC or compatible running MS-DOS. This provides an easy-to-use environment for:

- entering user-defined data patterns of the input parameters;
- simulating the effect on the miss ratio and cache performance as a function of the line size and overall cache size.

Figs 1–6 can be used to compute the miss ratio for one line size from the miss ratio (for a specific cache size) for another line size:

$$\text{Smoothed average of ratios} = R = \frac{m \text{ at line size } l}{m \text{ at line size } l/2} \qquad (1)$$

Figure 1 provides the input to the simulator showing the smoothed average of ratios (R) as a function of the line size (8 to 128 bytes) and instruction cache size (32 to 1024 bytes). Figure 2 is the same as Fig. 1 but covers the range of instruction cache size from 1024 to 32768 bytes. Figures 3 and 4 provide similar R values as Figs 1 and 2 but for data cache and Figs 5 and 6 for unified cache.

Accordingly, Figs 7–9 can be derived from equation (1). For example, assuming that the first row (i.e. line size = l = 4 bytes) in the top table of Fig. 7 are defined by the designer to act as the initial design target for the average miss ratio, the remaining rows can be regressively computed from the data given in Figs 1 and 2. For example, for instruction cache size of 32 bytes, we have:

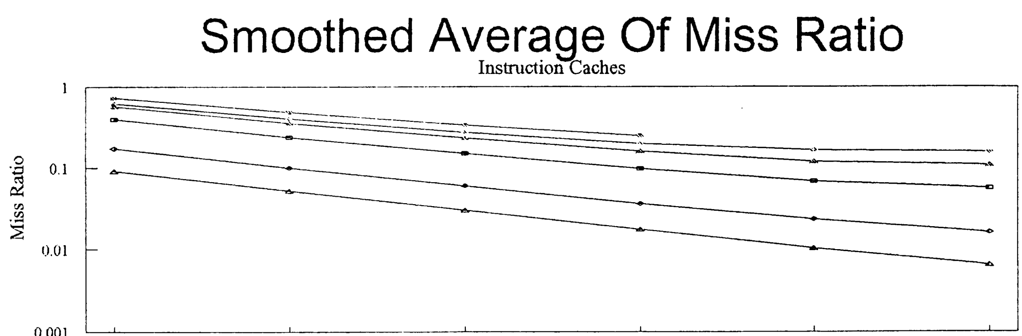
$$m_{(\text{at } l=8)} = R_{(\text{at } l=8)} \times m_{(\text{at } l=4)} = 0.66 \times 0.725 = 0.4785 \qquad (2)$$

The miss ratio can now be computed for different line sizes as a function of different cache sizes. Figures 8 and 9 can be similarly derived for data and unified cache respectively. The graphical presentations that the package facilitates, as shown in Figs 7–9, provides the user with an easy way to interpret the results and study the behavior of the miss ratio for different cache and line sizes. It should be stated at this point that the results obtained will obviously depend on the design target data and the R data tables initially defined by the user. This will provide the user with the flexibility to choose different initial data sets that may correspond to different machines.

In addition, bus bandwidth can be the limiting resource in a multi-microprocessor computer system, and thus memory traffic is a very significant performance factor. Memory traffic may be estimated by multiplying the miss ratio by the line size to yield the traffic in bytes/memory reference. Memory traffic consists of two components: fetch traffic and write or copy-back traffic. On the other hand, the traffic in the other direction, from cache

	Instruction Cahe Size										
Line Size	32	64	128	256	512	1024	2048	4096	8192	16384	32768
4	0.725000	0.674000	0.615000	0.592000	0.562000	0.504000	0.391000	0.271000	0.172000	0.148000	0.091000
8	0.478500	0.438100	0.396675	0.372960	0.348440	0.307440	0.234600	0.161245	0.099932	0.085396	0.052143
16	0.330165	0.300099	0.269739	0.249883	0.229970	0.199836	0.150144	0.099972	0.059959	0.049957	0.030139
32	0.247294	0.222073	0.196909	0.177417	0.158680	0.133890	0.097594	0.062982	0.036575	0.029424	0.017511
64		0.190983	0.163435	0.138385	0.119010	0.097740	0.068413	0.042828	0.023408	0.018243	0.010331
128			0.156897	0.128837	0.108299	0.084056	0.056920	0.032121	0.016152	0.011986	0.006550

Smoothed Average Of Miss Ratio

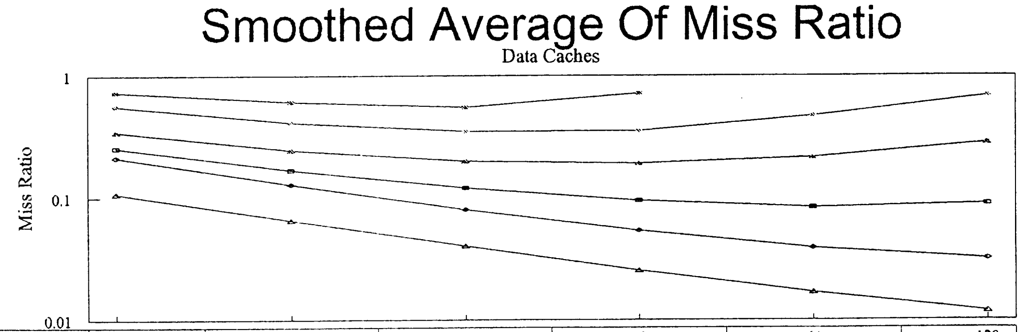


Line Size	4	8	16	32	64	128
32	0.725000	0.478500	0.330165	0.247294		
128	0.615000	0.396675	0.269739	0.196909	0.163435	0.156897
512	0.562000	0.348440	0.229970	0.158680	0.119010	0.108299
2048	0.391000	0.234600	0.150144	0.097594	0.068413	0.056920
8192	0.172000	0.099932	0.059959	0.036575	0.023408	0.016152
32768	0.091000	0.052143	0.030139	0.017511	0.010331	0.006550

Fig. 7. Smoothed average of miss ratio vs. line size, for instruction cache (cache size = 32 → 32 768 bytes).

Line Size	Data Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
4	0.731000	0.660000	0.561000	0.470000	0.345000	0.283000	0.256000	0.247000	0.214000	0.161000	0.108000
8	0.611116	0.515460	0.411774	0.337460	0.245640	0.210552	0.168960	0.152893	0.129042	0.096922	0.065016
16	0.550004	0.449997	0.350008	0.280092	0.199951	0.160020	0.120131	0.099992	0.080006	0.059898	0.040180
32	0.715006	0.494996	0.351408	0.271689	0.191153	0.137617	0.094543	0.069994	0.053364	0.038694	0.025313
64		0.692995	0.466670	0.326027	0.214474	0.139681	0.083198	0.053896	0.038582	0.026002	0.016757
128			0.676671	0.456438	0.281819	0.160633	0.089105	0.048506	0.031522	0.019424	0.011680

Smoothed Average Of Miss Ratio

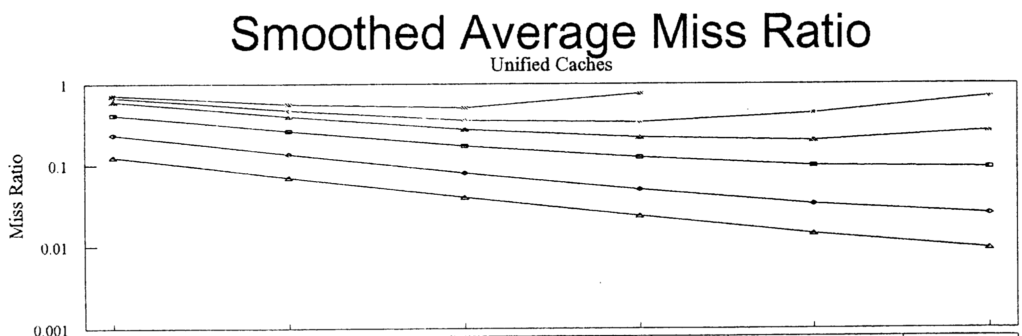


Line Size	4	8	16	32	64	128
32	0.731000	0.611116	0.550004	0.715006		
128	0.561000	0.411774	0.350008	0.351408	0.466670	0.676671
512	0.345000	0.245640	0.199951	0.191153	0.214474	0.281819
2048	0.256000	0.168960	0.120131	0.094543	0.083198	0.089105
8192	0.214000	0.129042	0.080006	0.053364	0.038582	0.031522
32768	0.108000	0.065016	0.040180	0.025313	0.016757	0.011680

Fig. 8. Smoothed average of miss ratio vs. line size, for data cache (cache size = 32 → 32 768 bytes).

Line Size	Unified Cahe Size										
	32	64	128	256	512	1024	2048	4096	8192	16384	32768
4	0.717000	0.686000	0.674000	0.643000	0.596000	0.473000	0.405000	0.329000	0.232000	0.182000	0.124000
8	0.555675	0.528906	0.466408	0.419879	0.389784	0.308869	0.257580	0.192794	0.134792	0.103558	0.069936
16	0.500107	0.433703	0.349806	0.299794	0.270120	0.210031	0.170003	0.127630	0.079932	0.060167	0.040213
32	0.750161	0.520444	0.329517	0.257823	0.216096	0.161724	0.124272	0.086788	0.050357	0.036100	0.023726
64		0.780665	0.428372	0.275870	0.197512	0.137465	0.097802	0.062487	0.033286	0.022852	0.014259
128			0.685396	0.386218	0.256766	0.151212	0.092912	0.053114	0.025064	0.015653	0.009411

Smoothed Average Of Miss Ratio



Line Size	4	8	16	32	64	128
32	0.717000	0.555675	0.500107	0.750161		
128	0.674000	0.466408	0.349806	0.329517	0.428372	0.685396
512	0.596000	0.389784	0.270120	0.216096	0.197512	0.256766
2048	0.405000	0.257580	0.170003	0.124272	0.097802	0.092912
8192	0.232000	0.134792	0.079932	0.050357	0.033286	0.025064
32768	0.124000	0.069936	0.040213	0.023726	0.014259	0.009411

Fig. 9. Smoothed average of miss ratio vs. line size, for unified cache (cache size = 32 → 32 768 bytes).

to main memory, will depend on whether the cache uses the write-through or copy-back policy.

Another worksheet was, therefore, provided on the Lotus 1-2-3 in order to study the influence of the line size on the bus bandwidth in a multicache-based multiprocessing system. The worksheet provides the user with definitions of the terms used, description of the bus timing and information on throughput bounds for different scenarios. For example, the communication bandwidth (*Traf*) for a write-through cache with hardware-enforced coherence is given by:

$$Traf = (M \times Crb) + (W \times Cw) + Xio \qquad (3)$$

where, M = total number of read misses per task executed; W = total number of writes per task executed; Xio = average number of bus cycles resulting from I/O per task; Crb = bus occupation time for reading a cache block; Cw = bus occupation time for writing a word (in bus cycles).

The upper bound on the MIPS rate is:

$$Th \leq \frac{Bw}{Dr \times (1 - Hr) \times Crb + Dw \times Cw + Xio} \qquad (4)$$

where, Hr = hit ratio for memory reads; Dr = demand ratio for read references per instruction; Dw = demand ratio for write references per instruction; Bw = maximum bandwidth that can be supported by bus(es).

The user can, therefore, use this worksheet to go through the definitions and equations pertaining to the bus throughput and can investigate the effect of the line size on the maximum throughput for different cache sizes. Figure 10 shows one such option for unified cache assuming the write-through update policy is used. Furthermore, the user can investigate the influence of the update policy on the maximum throughput. Figure 11 is similar to Fig. 10 except that the write-back update policy is used instead.

USE OF SPREADSHEETS IN ADVANCED COMPUTER ARCHITECTURE COURSES

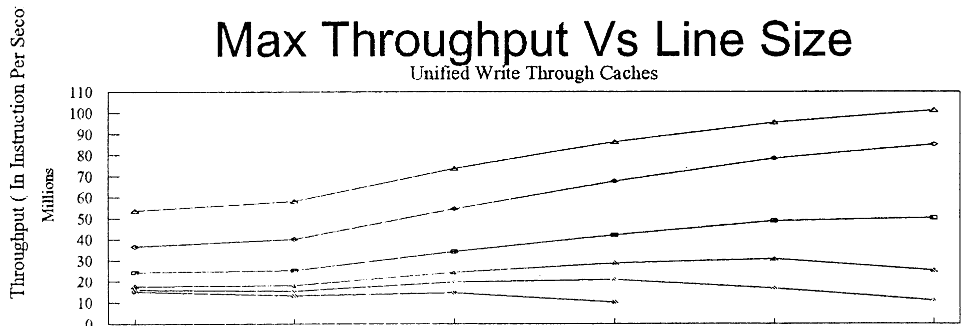
This paper has shown the use of spreadsheets as a user-friendly simulation tool for the performance evaluation of cache memory, a topic that is intensively addressed in an advanced (postgraduate level) computer architecture course. In addition to the use of the miss ratio and throughput as indicators of cache performance, a multitude of issues related to the design of cache memory may also be implemented. This may include testing the effect of the various placement policies (direct, fully-associative, set-associative, sector mapping), fetch policies (demand, anticipatory, selective), and replacement policies (LRU, FIFO, RAND, etc.).

In general, students reacted very favorably to using the package. Among the identifiable factors for this are that the package:

(a) provides the students with a graphics illustration of the simulation results. This clearly enables students to absorb and easily interpret

	Unified Cache Size										
Line Size	32	64	128	256	512	1024	2048	4096	8192	16384	32768
4	15,126,305	15,715,857	15,956,598	16,614,055	17,721,070	21,463,833	24,301,337	28,514,400	36,616,624	42,900,043	53,561,864
8	13,257,149	13,846,825	15,451,423	16,910,322	18,010,204	21,827,256	25,214,576	31,362,434	40,120,297	47,221,131	58,335,161
·16	14,542,726	16,448,897	19,713,476	22,358,743	24,292,813	29,451,862	34,304,922	41,553,165	54,520,280	62,617,158	73,661,656
32	10,124,598	9,762,781	20,707,333	25,196,088	28,833,775	35,515,328	42,260,701	52,179,600	67,600,303	76,440,688	86,228,494
64		10,124,598	16,623,807	23,892,345	30,815,277	39,610,504	48,813,223	61,544,158	78,466,373	87,015,691	95,592,207
128			10,989,312	18,150,081	25,276,865	37,181,064	50,252,672	66,121,292	85,050,521	94,087,597	101,221,454

Max Thoughput Vs Line Size , PS - W. Through



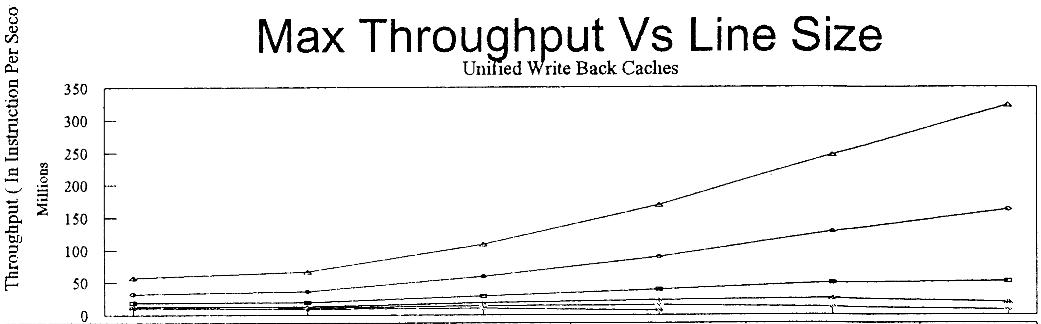
Line Size	4	8	16	32	64	128
32	15,126,305	13,257,149	14,542,726	10,124,598		
128	15,956,598	15,451,423	19,713,476	20,707,333	16,623,807	10,989,312
512	17,721,070	18,010,204	24,292,813	28,833,775	30,815,277	25,276,865
2048	24,301,337	25,214,576	34,304,922	42,260,701	48,813,223	50,252,672
8192	36,616,624	40,120,297	54,520,280	67,600,303	78,466,373	85,050,521
32768	53,561,864	58,335,161	73,661,656	86,228,494	95,592,207	101,221,454

Packet-Switched Buses

Fig. 10. Maximum throughput vs. line size, for unified cache (write-through update policy).

	Unified Cache Size										
Line Size	32	64	128	256	512	1024	2048	4096	8192	16384	32768
4	10,486,990	10,954,402	11,146,719	11,676,275	12,582,573	15,789,993	18,380,241	22,506,682	31,545,741	39,781,203	57,061,341
8	9,037,627	9,489,614	10,744,130	11,917,008	12,822,366	16,113,806	19,245,180	25,506,093	35,987,825	46,215,032	66,583,881
16	10,029,219	11,542,644	14,261,662	16,591,489	18,372,243	23,474,268	28,802,423	37,911,698	58,870,124	76,362,592	109,087,255
32	6,714,203	6,453,967	15,123,169	19,227,514	22,834,205	30,221,181	38,886,207	54,536,283	89,573,443	119,656,582	168,889,976
64		6,714,203	11,684,178	17,997,938	24,915,804	35,318,880	48,769,096	73,788,650	128,153,056	173,946,640	246,470,559
128			7,342,828	12,938,839	19,304,751	32,237,458	51,171,706	85,419,806	161,693,604	230,854,154	322,289,065

Max Thoughput Vs Line Size , PS - W. Back



Line Size	4	8	16	32	64	128
32	10,486,990	9,037,627	10,029,219	6,714,203		
128	11,146,719	10,744,130	14,261,662	15,123,169	11,684,178	7,342,828
512	12,582,573	12,822,366	18,372,243	22,834,205	24,915,804	19,304,751
2048	18,380,241	19,245,180	28,802,423	38,886,207	48,769,096	51,171,706
8192	31,545,741	35,987,825	58,870,124	89,573,443	128,153,056	161,693,604
32768	57,061,341	66,583,881	109,087,255	168,889,976	246,470,559	322,289,065

Packet-Switched Buses

Fig. 11. Maximum throughput vs. line size, for unified cache (write-back update policy).

the behavior of different case studies more efficiently;

(b) permits more individualized instruction which allows students to work at their own pace;

(c) enable students to define their own data sets for emulating different R and m values for different cache-based multiprocessor systems.

CONCLUSION

A simulation analysis tool for cache-based parallel processing systems using spreadsheets has been presented. The analysis is available in the form of a Lotus 1-2-3 worksheet under MS Windows that allows the user to select certain parameters of the system and derive the performance of cache.

The reported tool designed is motivated by the fact that a trend for using simple, cheap and user-friendly packages for system simulation, such as the Lotus 1-2-3, is becoming more commonly used.

REFERENCES

1. M. Cutler and R. Eckert, A microprogrammed computer simulator, *IEEE Trans. Educ.*, **E-30**(3), p. 135 (1987).
2. R. Yen and Y. Kim, Development and implementation of an educational simulator software package for a specific microprogramming architecture, *IEEE Trans. Educ.*, **E-29**, p. 1 (1986).
3. M. R. Smith, A microprogrammable microprocessor simulator and development system, *IEEE Trans. Educ.*, **E-27,** p. 93 (1984).
4. L. Bic, MICOS: A Microprogrammable Computer Simulator, Computer Science, Rockville MD (1985).
5. H. Diab and I. Demashkieh, A computer-aided teaching package for microprocessor systems education, *IEEE Trans. Educ.*, **E-34**(2), p. 179 (1991).
6. R. E. Purvis, R. D. Yoho and G. B. Lamont, MIME: An educational microprogrammable minicomputer emulator, *IEEE Trans. Educ.*, **E-24**, p. 257 (1981).
7. A. J. Smith, Line (block) size choice for CPU cache memories, *IEEE Trans. Comp.*, **C-36**(9) (1987).
8. S. Laha, J. Patel, and R. Iyer, Accurate low-cost methods for performance evaluation of cache memory systems, *IEEE Trans. Comp.*, **C-37**(11) (1988).
9. P. Yeh, J. Patel and E. Davidson, Shared cache for multiple-stream computer systems, *IEEE Trans. Comp.*, **C-32**(1) (1983).
10. M. Hill and A. Smith, Evaluating associativity in CPU cache, *IEEE Trans. Comp.*, **C-38**(12) (1989).
11. M. Dubois, Throughput analysis of cache-based multiprocessors with multiple buses, *IEEE Trans. Comp.*, **C-3**(1), 1988.
12. A. El-Hajj, K.Y. Kabalan, and H. Diab, A spreadsheet educational tool for microprocessor systems, *Computer Applications in Engineering Education,* **3**(3), pp. 205–211 (1995).

13. Y. Kuo and W. Kuo, Application of electronic spreadsheets to linear and integer programming, *Int. J. Applied Engineering Education,* **3**(6), pp. 563–575 (1987).
14. K. Y. Kabalan, A. El-Hajj, H. Diab, and S. Fakhreddine, Analog computer simulation using spreadsheets, *Simulation,* **68**(2), pp. 101–106 (1997).
15. K. Y. Kabalan, H. Diab, A. El-Hajj, and H. Tabbara, Stability analysis and minimum phase testing of causal filters using spreadsheets, *Int. J. Eng'g Educ*., **12**(5), pp. 382–388 (1996).
16. P. H. Saul, The spreadsheet as a high level mixed mode simulation tool, *Electronic Engineering,* pp. 549–555 (1991).
17. C. R. Thomas, Three methods of performing conditional looping in a spreadsheet, *Int. J. Eng'g Educ.,* **8**(4), pp. 271–277 (1988).
18. C. Y. Lam, Spreadsheet approach to partial differential equations. Part 1: Elliptic equation, *Int. J. Eng'g Educ*., **8**(4), pp. 278–287 (1988).

Hassan Diab is a Professor of Electrical and Computer Engineering at the ECE Department, Faculty of Engineering and Architecture, American University of Beirut. His research area is in the simulation of parallel processing systems. He is a Fellow of IEAust and a Senior Member of IEEE.