```
System:
            ------cache------                        Main
               instr     data               CPU   Memory
            ==================         ==================
Tmem             2.5      2.5                         20
Bw                16       16
Bk                                                     1
TPIb                                          2.5
A                                            0.95
B                                            0.75
pf                                              2

*****************************************************************
OUTPUT

Cache Model:
            ------cache------
               instr     data
            ==================
Cmr            0.080    0.038


P/I Model:
            ------cache------                        Main
               instr     data               CPU   Memory
            ==================         ==================
Rm-bar          1.33     1.33              1.11
Fp              0.35     0.35              1.39
Am-sub          0.49     0.49              3.85
Lint            2.03     2.03
Ce              2.41     2.41
Tpd             1.13     1.13
Dm                                                  2.20 cm

System Model:
            ------cache------
               instr     data               CPU SYSTEM
            ==================         ==================
Cet             6.36     5.52
TPI                                                 5.31
fs                                                188.23
As                                                  4.83
```

BCB MCM: U. Mich.; verification

INPUT

Cache:

	instr	data
Af	0.90	0.90
alpha	0.00	0.00
beta	0.00	0.00
S	16384	16384
(lgS)	14	14
R	16	16
(lgR)	4	4

Die: (no model for now)

	------cache------		
	instr	data	CPU
Dc	0.3	0.3	0.7
Np	48	48	300
Kb	1	1	
Wb	32	32	
Nc	4	4	2

P/I:

	------cache------		
	instr	data	CPU
FPcc	0.35	0.35	0.75
Nelt	0.65	0.65	0.65
Fc	1	1	2
Ew	0.6	0.6	0.4
Pw	0.005	0.005	0.005
Nw	2	2	2
Pp	0.02	0.02	0.02
Zo	70	70	70
Cpad	0.9	0.9	0.9
Rint	10.7	10.7	10.7
Cint	1.08	1.08	1.08
vm	18.7	18.7	18.7
Rtr	3200	3200	3200
Ctr	5	5	5
(N)	3	3	3

```
System:
            ------cache------                        Main
               instr     data               CPU   Memory
            ==================         ==================
Tmem              10       10                         60
Bw                 4        4
Bk                                                     2
TPIb                                        10
A                                         0.95
B                                         0.75
pf                                           2

*****************************************************************
OUTPUT

Cache Model:
            ------cache------
               instr     data
            ==================
Cmr            0.034    0.012



P/I Model:
            ------cache------                        Main

               instr     data               CPU   Memory
            ==================         ==================
Rm-bar          1.59     1.89              1.11
Fp              1.65     1.65              3.61
Am-sub         21.78    43.56             26.01
Lint            9.55    12.45
Ce              4.46     6.82
Tpd             6.68    10.19
Dm                                                   9.56 cm

System Model:
            ------cache------
               instr     data               CPU SYSTEM
            ==================         ==================
Cet            27.39    33.27
TPI                                                 18.63
fs                                                  53.69
As                                                  91.35
```

Appendix A. Sample Analyses

```
Advanced PCB: HP PA-RISC; verification

INPUT

Cache:

               instr     data
            ==================
Af              0.90     0.90
alpha           0.00     0.00
beta            0.00     0.00
S             131072   262144
(lgS)             17       18
R                 16       32
(lgR)              4        5

Die:        (no model for now)
            ------cache------
               instr     data               CPU
            ==================         =========
Dc              0.57     0.57              1.35
Np                52       52               300
Kb                 8        8
Wb                16       16
Nc                 8       16                 2

P/I:
            ------cache------
               instr     data               CPU
            ==================         =========
FPcc            1.65     1.65               2.2
Nelt            0.65     0.65              0.65
Fc                 1        1                 2
Ew               0.6      0.6               0.4
Pw             0.026    0.026             0.026
Nw                 4        4                 4
Pp              0.13     0.13              0.13
Zo                75       75                75
Cpad               5        5                 5
Rint             0.5      0.5               0.5
Cint             0.8      0.8               0.8
vm                17       17                17
Rtr             4000     4000              4000
Ctr               17       17                17
(N)                3        3                 3
```

[SC91a] Michael Slater and Brian Case. MIPS R4000 sets performance record. *Microprocessor Report*, October 2 1991.

[SC91b] Michael Slater and Brian Case. MIPS R4000 sets performance record. *Microprocessor Report*, 5(18), October 1991.

[SM91] Howard Sachs and Harlan McGhan. Future directions in CLIPPER processors. In *Compcon*, 1991.

[SMF+91] Hirofumi Shinohara, Noriaki Matsumoto, Kumiko Fujimori, Yoshiki Tsuhihashi, Hiroomi Nakao, Shuichi Kato, Yasutaka Horiba, and Akiharu Tada. A flexible multiport RAM compiler for data path. *IEEE Journal of Solid-state Circuits*, 26(3), March 1991.

[SRM+89] E. Seewan, S. L. Runyon, R. K Montoye, Q. Nguyen, and J. C. Ridings. VLSI circuit design for the RISC System/6000 processor. In *IBM RISC System/6000 Technology*. IBM, 1989.

[TSP+89] D. W. Terry, D. E. Stivers, K. W. Pennington, M. W. Riley, and H. C. Nguyen. Packaging for high performance. In *IBM RISC System/6000 Technology*. IBM, 1989.

References

[Bak90] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[BGM90] H. B. Bakoglu, G. F. Grohoski, and R. K. Montoye. The IBM RISC System/6000 processor: Hardware overview. *IBM Journal of Research and Development*, 34(1), January 1990.

[BW89] H. B. Bakoglu and T. Whiteside. RISC Sstem/6000 hardware overview. In *IBM RISC System/6000 Technology.* IBM, 1989.

[Cos91] Terry Costlow. IBM puts its RISC on multichip module. *Electronic Engineering Times*, July 1 1991.

[FMD+91] Mark Forsyth, Steve Mangelsdorf, Eric DeLano, Craig Gleason, and Jeff Yetter. CMOS PA-RISC Processor for a New Family of Workstations. In *Compcon*, 1991.

[Gro90] G. F. Grohoski. Machine organization of the IBM RISC Sstem/6000 processor. *IBM Journal of Research and Development*, 34(1), January 1990.

[Hig90] Lee Higbie. Quick and Easy Cache Performance Analysis. *Computer Architecture News*, 18(2):33–44, June 1990.

[HJ91] John L. Hennessy and Norman P. Jouppi. Computer technology and architecture: An evolving interaction. *Computer*, September 1991.

[HJT+91] Rob Horning, Leith Johnson, Larry Thayer, Daniel Li, Victoria Meier, Casey Dowdell, and David Roberts. Systems Design for a Low Cost PA-RISC Desktop Workstation. In *Compcon*, 1991.

[IBM90] IBM. *IBM RISC System/6000 Hardware Technical Reference General Information*, 1990.

[KSB+91] A. I. Kayssi, Karem A. Sakallah, Richard B. Brown, Ronald J. Lomax, Trevor N. Mudge, and Thomas R. Huff. Impact of MCMs on System Performance. In *Proc. Multichip Module Workshop*, pages 58–65, 1991.

[KV87] Kevin Karplus and T. B. Verghese. Sizing cMOS gates along a critical path – a tutorial. Technical Report UCSC-CRL-87-30, University of California, Santa Cruz, 1987.

[Men91] Alex Mendelsohn. Will monolithic or multichip processors win the performance race. *Computer Design*, May 1991.

[MQF91] Johannes M. Mulder, Nhon T. Quach, and Michael J. Flynn. An area model for on-chip memories and its application. *IEEE Journal of Solid-state Circuits*, 26(2), February 1991.

[OBL+91] O. A. Olukotun, R. B. Brown, R. J. Lomax, T. N. Mudge, and K. A. Sakallah. Multilevel optimization in the design of a high-performance GaAs microcomputer. *Journal of Solid-State Circuits*, 26(5), May 1991.

[PH91] Olgierd A. Palusinski and Jokob H. Hohl. Modeling of performance-related design trade-offs in multi-chip assemblies. In *IEEE/CHMT IEMT Symposium*, 1991.

with immediate feedback from technology to architecture decisions. Development risks can be reduced and more what-if alternatives can be explored.

It must be kept in mind that the results presented are valid only for the given assumptions and options. Considering first-order modeling accuracy limitations, they should be considered no more than indications of relative values and trends, which should serve to identify areas for further investigation with more refined models. They are none the less suggestive of improved design trade-offs for RISC system implementations. Although it does appear that MCM implementations will typically suffer a performance loss over monolithic, the loss can be expected to be small. The increased resistance of on-chip interconnection appears relatively closely matched to the chip-crossing delay penalty for flip-chip MCM interconnection, and should be relatively unaffected by scaling to smaller die. It even may be possible to achieve higher performance with MCM in some instances, such as when the improved die yields are used to justify larger caches and when the higher pin count and routing capacities are used to improve cache hierarchy bandwidths.

6. Conclusion

We have prototyped an early analysis tool for exploring cache architecture and implementation technology. The current study only touches on the large design space made available for trade-off analysis by early analysis tools. Although based on numerous assumptions regarding input parameters and subject to first-order model accuracy, the case studies have identified and quantified many trade-offs in cache design for RISC microprocessor based systems. Insight into various trends and expected performance differences has also been provided. After verifying the model on two test cases, we quantified MCM benefits in comparison to printed circuit boards and noted the improvement possible by utilizing early analysis bridging architectural and technology choices. We also compared monolithic and MCM implementations with one and two level caches, evaluated other cache organization parameters, and considered the impact of package pin constraints. Comparison to two additional processors was also made. From the results of this investigation, we have made suggestions for extensions to the model to improve accuracy and detail, moving the model towards a practical design tool. These preliminary results show that MCM should suffer a 10% or less performance loss in comparison to monolithic implementation. In some instances MCM may even offer higher performance than monolithic implementation, particularly where the increased yield resulting from MCM implementation with multiple smaller die is used to justify larger caches and/or when 2-level caches are both implemented on MCM removing bandwidth restrictions.

The first case studies looked at the Hewlett Packard PA-RISC system which operates at 66 MHz and utilizes printed circuit boards, and an experimental system operating at 250 MHz utilizing MCM technology [KSB+91]. Verification results were good, projected system clock frequencies being off approximately 10%. Comparison of model results for the RS/6000 and R4000 based systems showed similar agreement. Assuming aggressive leading edge printed circuit board technologies, it is projected that MCM technology would improve performance by 10 to 50%. The additional benefits of using early analysis to optimize cache sizing specifically for the MCM environment were roughly 10%. Comparative performance between monolithic (single chip) and MCM implementations differed significantly from case to case, and was particularly sensitive to refill bandwidth. For an hypothetical MCM version of the PA-RISC system, a monolithic implementation offered higher performance for cache sizes below 16 K bytes (instruction and data cache, totaling 32 K bytes). At the expense of caches several times larger, an MCM implementation was shown in one instance to out perform monolithic by 10% where refill bandwidth is restricted. Other cases show monolithic implementations exceeding MCM performance by 5 to 10%. Considering greatly increased die yield from multiple smaller die, it may be possible to justify the cost of larger caches and improve the relative performance between MCM and monolithic implementations. Case studies also suggest that a secondary cache will be desirable, even when memory with similar access time is used at both levels, with performance improvements of up to 15%. By permitting flip-chip die mounting, MCM may also reduce pin constraints on refill bandwidth, allowing a 15% performance improvement in one case.

This work has shown the importance of early analysis, optimizing the RISC memory hierarchy to the MCM environment. Our analysis approach permits quick exploration of trade-offs and identification of trends without lengthy timing analysis from layout extractions and detailed system simulations. This permits analysis in the early design phases,

5.5 Power

Die power can be estimated via the SUSPENS model. For driver power estimation C_m, the sum of all driver and line capacitances, can be calculated separately for each cache and the processor. The SUSPENS equation is again modified for the bused interconnect – $\overline{R}_M$ (average interconnection length) is replaced by L_{int} and the number of lines and drivers is not multiplied by the number of cache chips and output fanout, but the number of pad capacitances is based on the number of cache chips):

$$C_m = N_p(3\frac{1-5^N}{1-5}C_{tr} + (1+N_c)C_{pad} + L_{int}C_{int}).$$

Power per unit of frequency is then calculated for later frequency adjustment with the die power by omitting f_s in the SUSPENS equation:

$$\frac{P_m}{f_s} = \frac{1}{2}F_d C_m V_{dd}^2$$

where F_d is the average portion of output drivers switching per cycle. The SUSPENS die power model should be included, and extended to allow for short-circuit and static power. The module model includes power for the output drivers, and the power model for bipolar drivers should also be included.

Several minor refinements could also be investigated, such as sub-block refill. Since on a cache miss only one line is replaced, refill width B_w might be limited to $\leq R$ (or perhaps R times ratio of speeds). An explicit model of main memory organization (width, interleaving, bus access, etc.) may also be desirable. The early analysis model is currently set up primarily for optimization of cache miss rate versus propagation delays. Estimates of resulting system performance could be improved somewhat by taking time granularity into account, i.e. a cache miss will incur a penalty which is a multiple of the CPU cycle time.

5.4 Propagation Delay and Output Drivers

The propagation delay model in our rapid prototype tool is a simple RC model based on SUSPENS. An actual design tool should use a lumped RLC model, with multiple sections for each intermediate pad on data, address, and control buses. Distributed RLC and other more advanced signal propagation analysis should be used later in the design effort to more accurately predict timing, and to project noise and crosstalk. Within our first-order model, several extensions would be useful.

Extending the P/I model to include stacked die could be particularly important. The increased packaging density afforded by '3-d memory' could substantially improve the performance of MCM systems over planar monolithic implementations. The extension should be relatively straightforward, the development of estimations of the interconnection lengths, resistance, and loading capacitance.

Optimum output driver technology selection and driver chain sizing can be important in maximizing the advantages of MCM implementation. As discussed, output driver chain delay could be refined, the method presented in [KV87] being an alternative to that in SUSPENS. Initial review indicates that the two methods are approximately equivalent in both numerical result and calculation methodology, suggesting that the SUSPENS model remains a good first-order approximation. It may also be desirable to size the output inverter chain so that the final $R_{dr} = Z_o - R_{int}L_{int}$, maximizing speed while still approximating critical series damping.

It would be very desirable to extend the model to include the option of bipolar output drivers as a result of trends towards BiCMOS (bipolar drivers for long and/or heavily loaded on-chip lines and chip outputs). This could be especially important for on-chip cache tradeoffs. SUSPENS includes a current mode logic (CML) model for gate delay, T_g, which would have to be extended to an output driver chain if more than one stage is required. The model also includes loading capacitance. T_g is almost directly proportional to capacitive loading and would be particularly sensitive to C_{pad}, so that the propagation delay model would also have to be reconsidered. CML gate delay is also roughly proportional to gate current, I_{DC}. The SUSPENS model is for all gates being bipolar, and sets I_{DC} to the maximum possible within the package power dissipation constraint to maximize speed.[1] The addition of bipolar drivers to the model (i.e. BiCMOS) would thus allow speed-power tradeoffs, provided a power calculation is added to the model.

[1] According to SUSPENS, this power constraint gives 1μm CMOS a speed advantage over even 1μm GaAs bipolar for chips above a quarter million transistors at 10 W dissipation limit.

We can then compute the chip data width in bits C_w, and the chip depth in kilobytes, C_d,

$$C_w = \frac{8R}{N_c}$$

$$C_d = \frac{K_b}{C_w}$$

which can then be used to estimate the number of pins

$$N_p = 1.5(4 + C_w + \lg(1024C_d)).$$

The 1.5 multiplier allows for one power and one ground pin for every 4 signal lines (this could be reduced for lower power, slower memory). The next constant is an estimation for the number of control pins. The next term is the number of data pins, and the last term is the number of address pins. The first model included a term in determining the effective footprint of $\sqrt{N_p}P_p$, where P_p is the pad pitch. This is appropriate for MCM flip-chip mounting. For printed circuit boards, packaging similar to quad flat packs are assumed and the term is replaced by $\frac{N_p}{4}P_p$. Although potentially useful for estimating pin count for a die model, the pin count adjustments alone had negligible effect on model results for several test cases.

5.2 Explicit On-chip Cache

One can use the SUSPENS model and coefficients for die model interconnect, or alternatively WSI P/I model and coefficients if one wants to assume wide traces / additional metal layer for long lines on large sub-micron die. The option of BiCMOS, bipolar drivers for long (cache address, data, and control) lines, discussed in a later section, can also be important for analysis of monolithic versus MCM implementations.

5.3 Explicit Second Level Cache

In this report we did not investigate details of the second level of cache, but rather made an estimation of effective access time for a reasonable sized direct mapped unified (mixed data and instruction) cache. It would be very desirable to allow exploration of secondary cache sizing and organization, interacting with primary cache characteristics and performance.

The model in [Hig90] covers multi-level cache. Such extensions are needed to allow the option of assigning each level to on-chip, on-MCM, or on-PCB, and more importantly for analyzing sizing interactions between the two levels. Higbie's model computes miss rates per instruction executed, and assumes that each cache level is independent of the others. Under Higbie's assumptions, the same miss rate equations can be applied to each level, with no information from other levels. This may not be the case, however. In many instances trace simulations suggest that the second level cache has a shielding effect; whatever behavior is exhibited by a single level cache may show on the secondary cache in a two level cache system. One of the two levels may capture most of the locality, reducing the effects of associativity and/or line size.

5. Future Model Development

The following suggestions are for future model development. *Calibration of the existing cache model with actual RISC architectures is particularly important.* Ideally, this effort would include superscalar processors, and incorporate more specific architectural features such as pipeline breaks and replacement policy. Verification of the T_{dr} and T_{pd} models against SPICE simulations would also be appropriate. A future die model could be based on SUSPENS and [MQF91], and a detailed access time model is needed. Global place and route could be used to improve interconnect estimations, with more detailed simulations of propagation delay and noise.

5.1 Cache Die: Area, Yield, Speed, Pins

The first model makes the simplifying assumptions of constant cache access time (T_{mem}), and area directly proportional to size (S). This is not the case in practice. In discussing an area model, Mulder et al. ([MQF91]) show that line size and associativity affect cache overhead (tags and status bits) to give total area variations of 20 to 40% over data or instruction storage alone. Although the previous parametric study indicates this will affect performance by only 1 to 3% at optimally sized caches (10 to 15% for caches several times larger than optimum), die size can be critical in projecting yield for on-chip caches. It would be desirable to include the area model of [MQF91] in any subsequent early analysis model. The yield equation from SUSPENS and others could also be included.

Cache die area can in turn affect access time. A curve fit of typical results presented in [SMF+91] for CMOS SRAM gave the following relationship:

$$T_{mem} = a + b(\sqrt{B} - c)$$

where B is the number of memory bits and a, b, and c are coefficients. Quadrupling total number of bits per die roughly doubles access time, and the form of the equation suggests a fixed logic delay plus a propagation delay proportional to the dimension of one die edge. An access time model would also allow investigation of optimal die sizing for large off-CPU caches. In addition to direct use of such an empirical approach, this suggests a model could be based on the SUSPENS die model utilizing an equivalent logic depth for storage cell, driver, amplifier, and comparator. Propagation delay could be estimated by taking $L_{int} = D_c$, with propagation delay linear in L_{int} (driver resistance dominated, and/or intermediate buffers). The empirical fit is for SRAM only, and does not include the other logic of a cache system. Effect of cache organization on access time at the memory subsystem level (address decoders, control, etc.) should also be considered. This would give a model where cache area is a function of S, R, B_w, N_c, α, β, and die technology. Memory access time would be a function of die area, interconnection technology, and cache organization.

In performing parametrics on cache size and R, it was observed that the fixed memory die parameters were often invalidated, particularly for small, wide caches and large memory chips. One option considered was to ignore internal chip organization (with resulting effects on die size and access time) and add the following simple adjustments for pin count. Where K_b is now total number of kilobits per die,

$$N_c = \frac{8S}{K_b}.$$

Alternative	fs	Total Die Area	A D=0	A D=.5	A D=1.
1-level 16K, monolithic	82.7	1.797	1.80	4.38	10.57
1-level 16K, MCM	79.0	2(.56) + 2(.325)	1.77	2.24	2.86
1-level 32K, MCM	83.2	2(.56) + 4(.325)	2.42	3.00	3.77
1-level 64K, MCM	86.4	2(.56) + 8(.325)	3.72	4.53	5.58
1-level 128K, MCM	87.5	2(.56) + 16(.325)	6.32	7.59	9.19
2-level 4&32K, mono&MCM	109.6	1.293 +4(.325)	2.59	4.02	6.59
2-level 4&32K, mono&PCB	92.2	1.293 +4(.325)	2.59	4.02	6.59

Alternative	AT D=0	AT D=.5	AT D=1.
1-level 16K, monolithic	2.18	5.30	12.78
1-level 16K, MCM	2.24	2.84	3.62
1-level 32K, MCM	2.91	3.61	4.53
1-level 64K, MCM	4.31	5.2	6.46
1-level 128K, MCM	7.22	8.68	10.50
2-level 4&32K, mono&MCM	2.36	3.67	6.01
2-level 4&32K, mono&PCB	2.81	4.36	7.15

Table 4.2: Effective Area-Time for Alternatives

(area-time) advantages are similar. Indeed, for the specific cases compared, the increased cache area to give greater performance than a monolithic implementation can be justified when die yields are taken into account. The analysis also indicates that a 2-level cache can provide a large performance improvement (roughly 30%) over single-level cache even when memory speeds are the same at both levels when coupled with reducing refill width restrictions. This is, however, at a slightly poorer area-time metric.

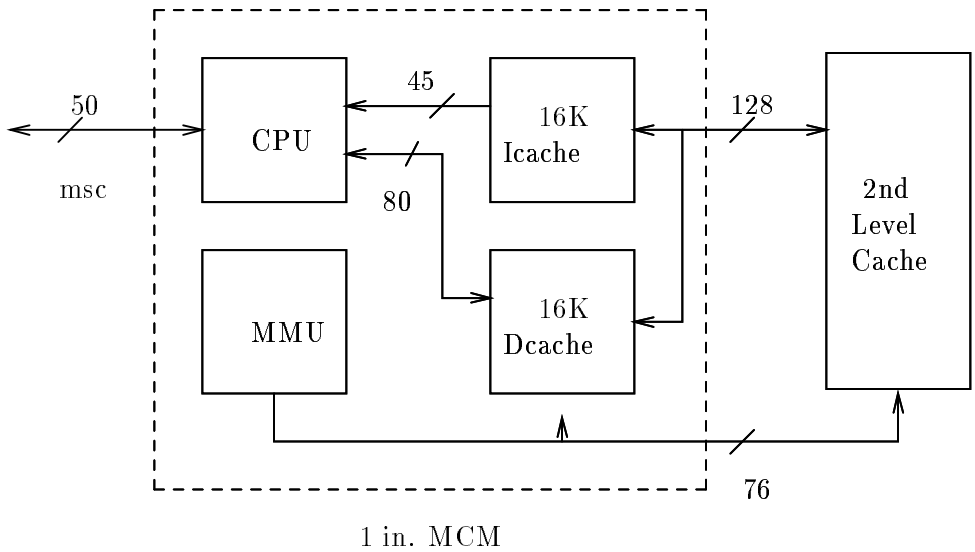


Figure 4.8: University of Michigan Case

Die Type	Area cm^2	Yield $D = .5$	Yield $D = 1.$
on-chip memory tile	.084		
off-chip memory	.325	.85	.72
processor (2 die, each)	.560	.76	.57
monolithic, 2-level cache (4K)	1.293	.52	.27
monolithic, 1-level cache (16K)	1.797	.41	.17

Table 4.1: Die Area and Yields

With the previous monolithic and 2-level cache analysis results we can look at some of the cost tradeoffs. In Table 4.1 we estimate the yields for the various die involved at .35 μm minimum feature size. A defect rate of .5 defects per cm^2 is typical of stable processes and chip designs (third year), while 1 defect per cm^2 is typical for the second year of production and an appropriate average for short lifetime products [SM91]. For the larger, monolithic die, the yield penalty is clearly visible.

Table 4.6 presents an area-time metric for several alternatives chosen from the previous case studies. Instruction and data caches are taken as the same size, so that total cache memory is twice that given in the table. For all cases except the 2-level cache, B_w is 8 bytes. For the 2-level case, B_w is 32 bytes, taking advantage of high flip- chip pin counts. Total die area shows the die area for processor and memory chips, in cm^2. The 'A' columns show the effective die processing area required, taking yield into account. For the 'D=0' column, this is the same as total die area. The 'AT' columns present the area-time metric, $100A/f_s$.

At equivalent cache sizes (16 K bytes each for instruction and data caches), MCM has an almost 2 to 1 'cost' advantage at a defect rate of .5 per cm^2. With a defect rate of 1 per cm^2, the advantage approaches 4 to 1. Even more interestingly, the 'cost-performance'

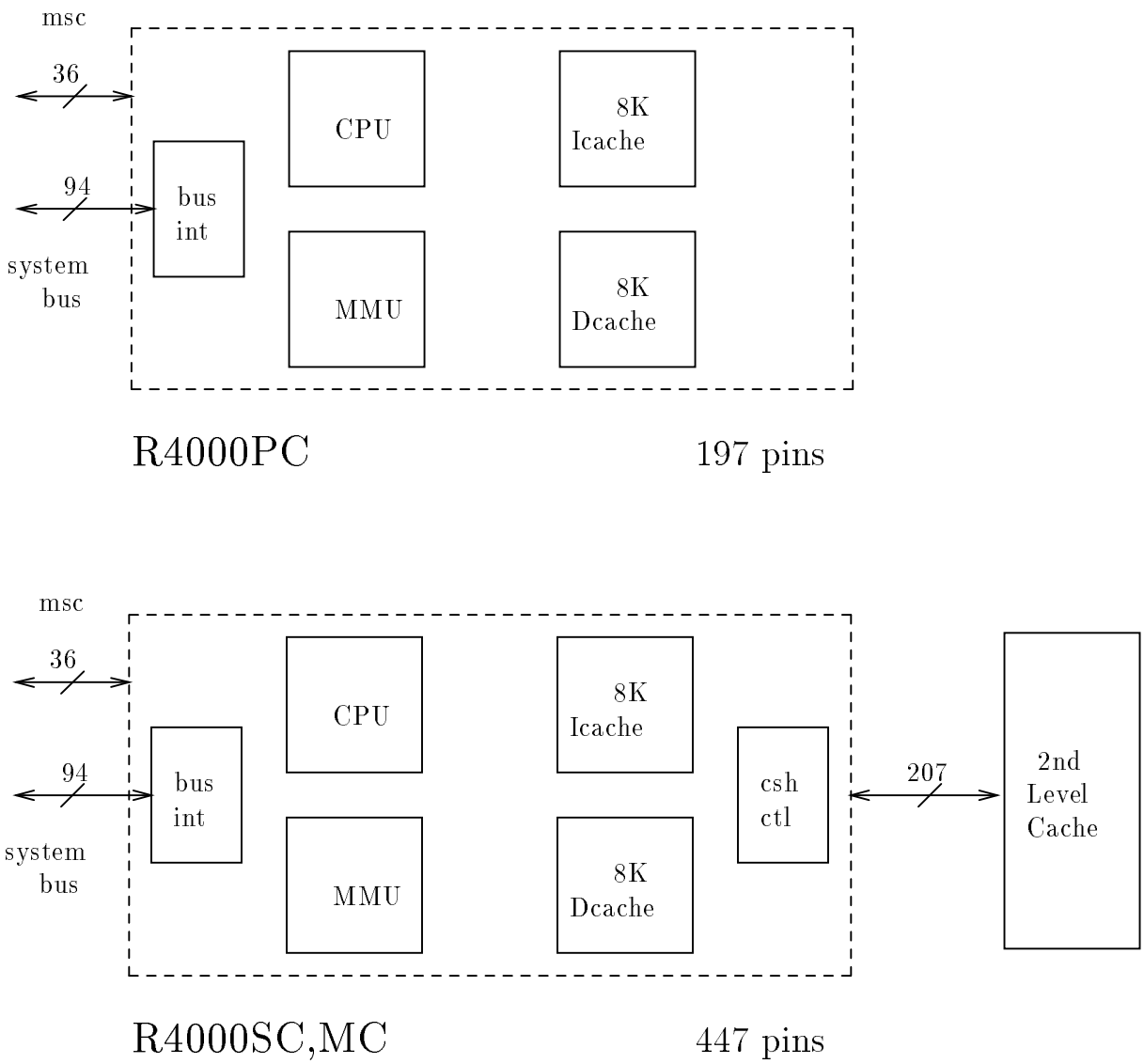


Figure 4.7: R4000 Block Diagram and Package Pins

cache levels are on PCB, but performance would be reduced over MCM and signal integrity would be a major concern.

4.6 Yield

A key advantage of MCM over monolithic implementation is the greatly increased yield possible with multiple smaller die. The standard formula for yield is given by [Bak90] and others as

$$Y = e^{-AD}$$

where A is die area in cm^2 and D is the defect rate in defects per cm^2. With yield inversely exponential in die area, the effect can be dramatic. From yield, an effective silicon area requirement can be determined and used as a cost metric. In addition to showing the cost benefits of MCM, this allows trade-off of larger MCM cache sizes to improve performance.

4.5 Pin Constraints

The prior case studies can be used to quantify the importance of packaging pin constraints. In the following analysis we assume a motherboard style workstation implementation which includes all main memory. The recently introduced R4000 RISC chip served as a basis of leading edge packaging [SC91b]. The R4000 is also similar to our other monolithic cases, with CPU, FPU, and caches integrated on a single large chip, as shown in Figure 4.7. The key pin count requirement is for access to the secondary cache. A single-level cache version of the chip, the R4000PC is available in a 179 pin package, which includes 94 pins for accessing the system bus and 36 miscellaneous control and clocking pins. Restriction to the system bus greatly increases cache refill time, and this chip version is intended for low end cost sensitive applications. The SC and MC versions of the chip add 207 lines for secondary cache access: 128 bit data path, 23 bits error correction code (ECC), 25 tag bits, 21 address bits, and 10 control lines. This results in a 447 pin package which none the less is considered economically feasible (1993 projected price \$300 to \$500 [SC91b]). Pin counts for MCM packages of the sizes considered in this study are similar. We will thereby assume that a refill width of 64 bits is economical, and a width as large as 128 bits is practical for high performance systems, across any packaging partition.

Figure 4.8 shows the high performance University of Michigan verification case, with similar data path pin count requirements across the packaging partitions[6]. Indeed, the MCM boundary is very similar to the chip boundary for the R4000, and the resulting packages would be quite similar. This is unfortunate, as the key bandwidth requirements are across the MCM package, which still faces significant pinout constraints as opposed to flip-chip mounted die crossings within the module. This suggests the potential importance of including a secondary cache on the MCM with flip-chip mounting for the processor.

Here we will ignore other parameters of cache organization which may affect pin requirements, such as cache line width and associativity, and look specifically at refill bandwidth.[7] For a single level cache with main memory on the same PCB, pin constraints are essentially the same for crossing chip or MCM boundaries, and is constrained primarily by the system bus data width (typically 32 or 64 bits). For a 2-level cache with the first level on-chip, the die boundary limits refill width. For single chip packaging (secondary cache on PCB), refill is constrained to 64 to 128 bits (B_w of 8 to 16 bytes). With flip-chip mounting on MCM, pin count is far less constrained, allowing B_w of 32 or more bytes. This can result in improvements in the range of 15% (from the prior analysis, increasing B_w from 8 to 32 bytes raises f_s from 97 to 110 MHz, a 13% improvement).

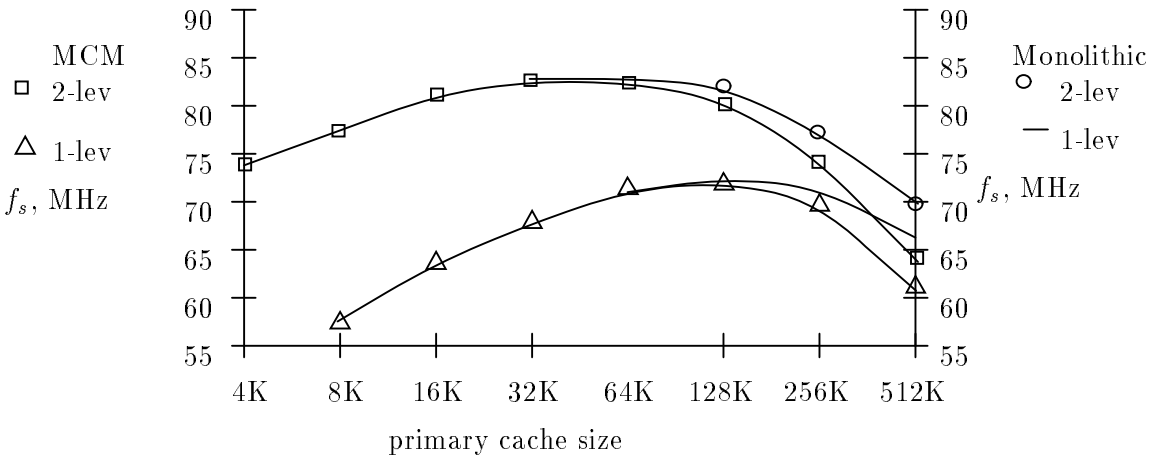
The prior analysis shows that 2-level caches are undesirable if both caches are on MCM and memory speeds are similar, which is unfortunate as such an implementation would place almost no constraints on refill width[8]. For cases where 2-level caching is used and a much faster (but more expensive or lower density memory) is used for the first level, the relaxation of B_w pin constraints offered by MCM could provide performance benefits even greater than for the above monolithic example. A similar argument would apply if both

[6] Miscellaneous and cache control pin counts were estimated based on the R4000.

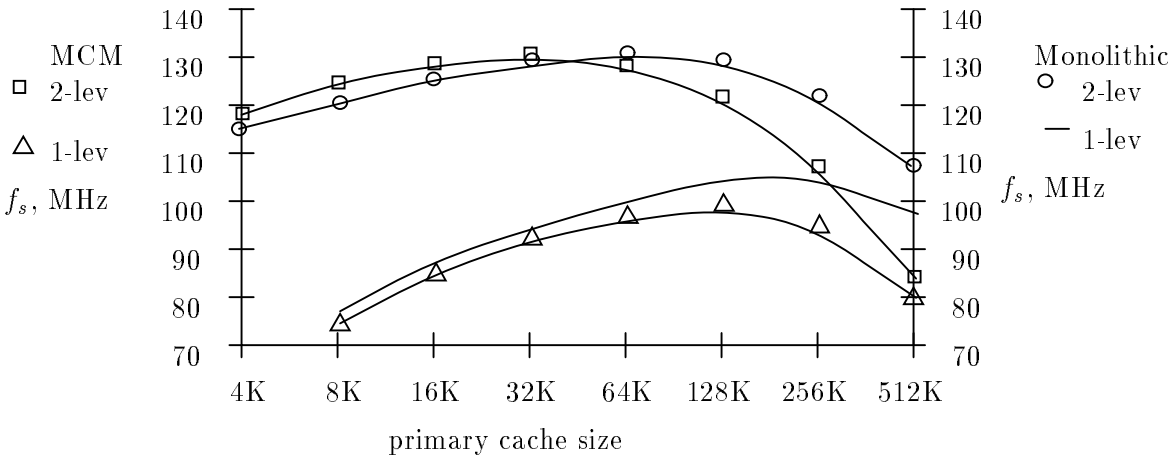
[7] Main memory organization, such as interleaving, memory, and bus data widths may also be pin and route capacity constrained, but are not within the topic of this report.

[8] Utilizing flip-chip mounting, with both levels on the MCM substrate and composed of multiple chips.

are expected in .5 μm CMOS, also shown in Figure 4.6. Here we use the optional 32 byte line width, and allow 32 byte refill bandwidth if both levels are on MCM. Results are similar to those at .8 μm. MCM primary cache has lower performance than monolithic for oversized caches, with no significant difference at near optimal sizing. Without secondary cache, on-chip primary cache offers slightly higher (approximately 4%) at the planned 64 K byte (each) planned on-chip sizes. At optimal (128 K bytes each) sizing, the performance difference is closer to 8%.

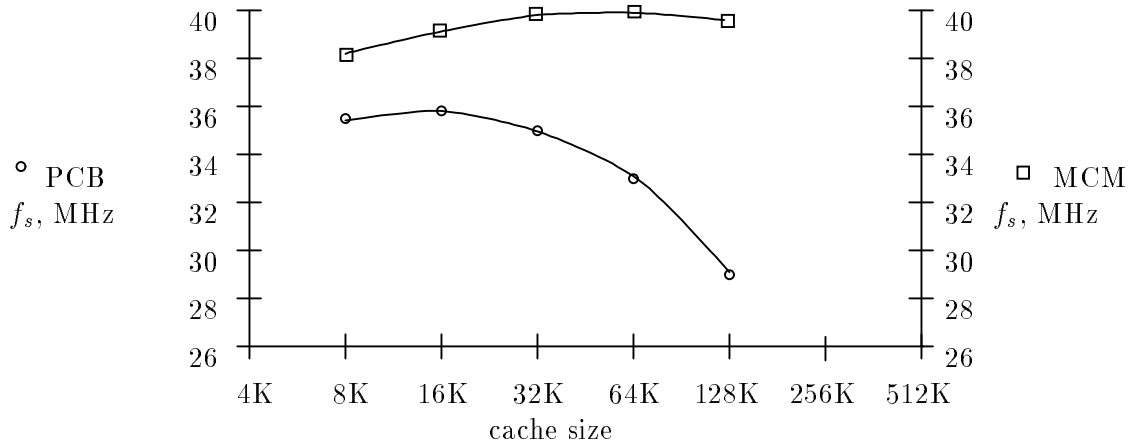


(a) Current (.8μ)

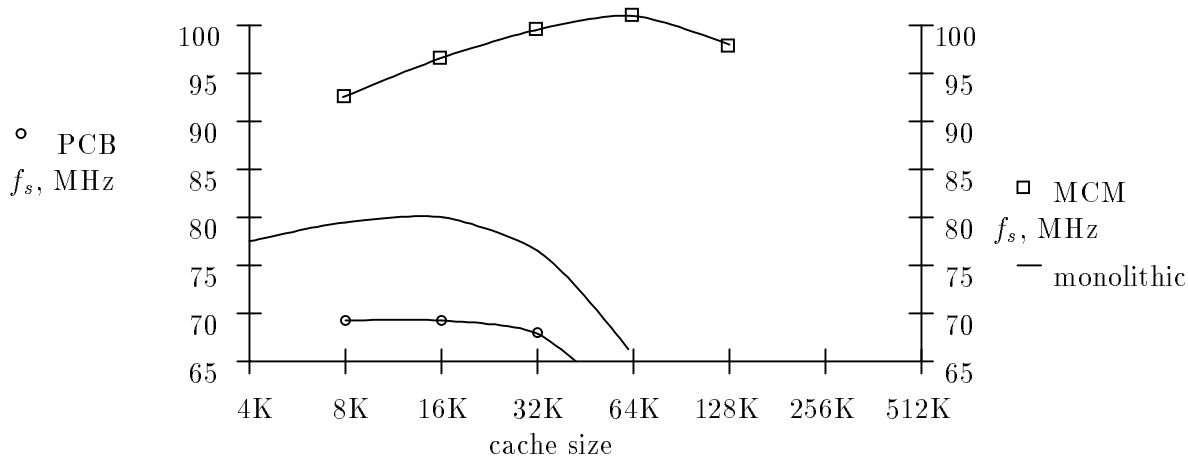


(b) .5μ

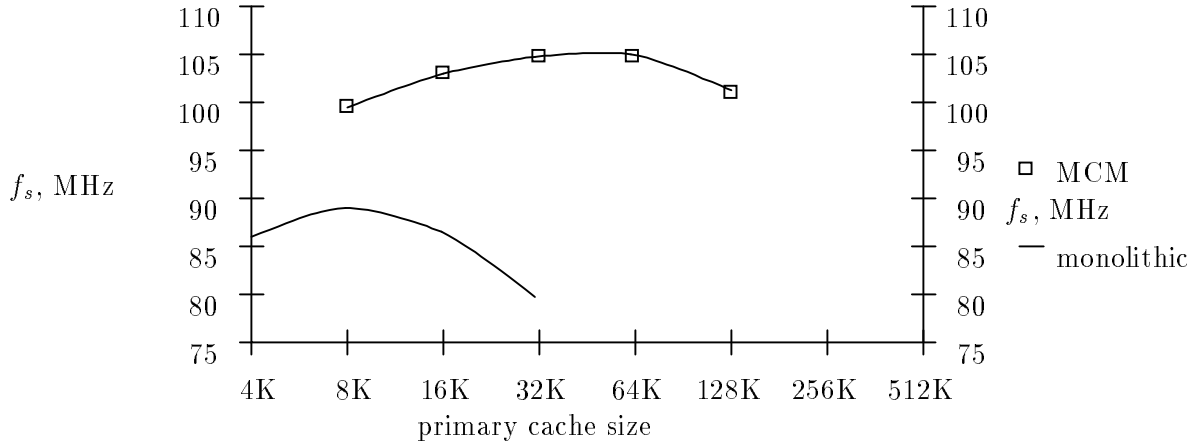
Figure 4.6: R4000 Cache Sizing



(a) Current (1μ)



(b) $.35\mu$



With Secondary Cache

Figure 4.5: IBM RS/6000 Cache Sizing

13 cm^2. Scaled to .35 μm, the base system (8 K byte instruction and 64 K byte data caches) could fit on a die 1.25 cm on a side. Figure 4.5b includes a monolithic option. Note that due to the much larger processor area (superscalar architectural enhancements, inclusion of memory management and input-output unit), the MCM implementation has much higher performance than a monolithic version. At larger cache sizing, MCM offers over 30% performance improvement, over 15% when restricted to equivalent cache size. Much of the performance advantage would be lost if the data cache were reduced due to on-chip size restrictions, and as previously discussed VLSI enhancements could be applied to improve monolithic performance. This does suggest that on-chip instruction cache and an off-chip data cache is worthy of consideration for superscalar systems.

Figure 4.5c considers the addition of a secondary cache. The MCM case places both primary and secondary cache on the MCM, and assumes a 256 K byte data cache. Improvement is slight, around 6%. With primary caches on-chip (monolithic, data cache 32 K bytes), performance improvement is greater (around 11%), but performance is still less than an all-MCM implementation.

4.4 MIPS R4000

The previous test cases have been complex, high performance systems more similar to minicomputers than microprocessor systems. Here we look at a contemporary microprocessor, the MIPS R4000. The R4000 is particularly interesting as a monolithic system with on-chip cache, with support for an off-chip secondary cache. In this section we take a first look at sizing and MCM versus monolithic implementations. The next section describes the R4000 in more detail, in terms of pin constraints for supporting secondary cache.

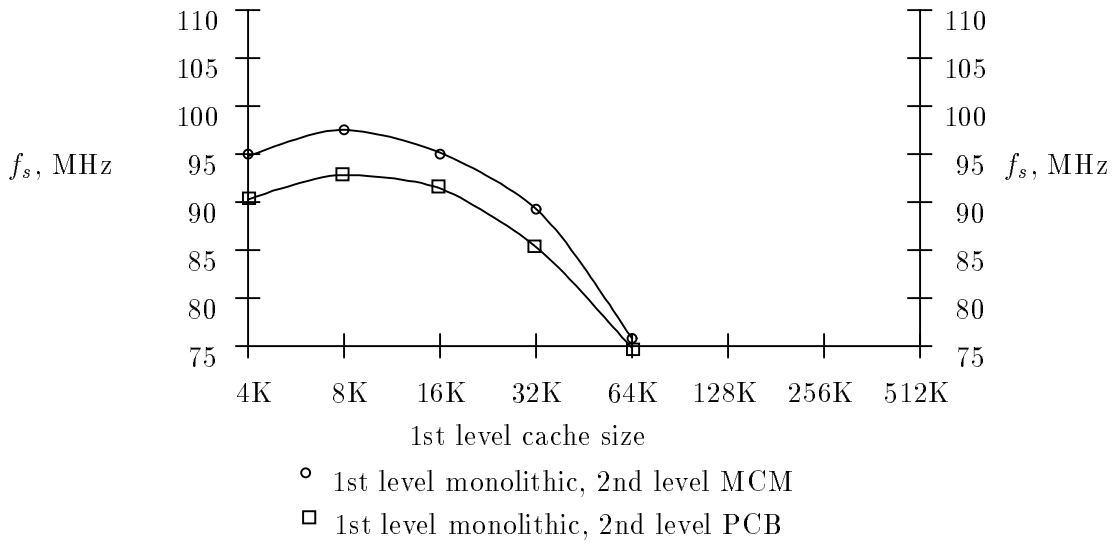
Advance information on the R4000 was taken from [SC91a]. Using .8 μm CMOS the 64 bit processor operates at 100 MHz internally and 50 MHz externally giving 62 SPECmarks with a large secondary cache. 8 K byte on-chip instruction and data caches are included, with line sizes of 16 or 32 bytes. True to the RISC philosophy, the chip is extremely small, only .29 cm^2, including primary caches, MMU, and second level cache control. The on-chip caches take up only about 11% of the die area. The cache areas were separated out, with pincount and access time estimated for input to the spreadsheet. With 50 MHz external frequency, secondary cache C_{et} was taken as 20 ns. Processor critical path delay was estimated by working backwards from our system model taking a 10 ns cycle time, equal cpu and cache times, and 1.7 ns propagation delay from a first iteration ($10 = \frac{1}{2}(T_{mem}+T_{cpu}+1.7)$). Cross checking with 'main memory' access time set to 0 ns gave 96 MHz. With 20 ns 'main memory' (secondary cache) our model gave f_s of approximately 78 MHz. This appears reasonable in comparison to the published SPECmark.

Figure 4.6a shows a sizing analysis, keeping data cache the same size as the instruction cache. The model gives optimal cache size of 32 to 64 K bytes each, indicating that restricting on-chip caches to 8 K incurs an 8% performance penalty even in the presence of a secondary cache. Due to the small processor die area, placing primary cache on MCM offers no advantage, providing performance essentially identical to on-chip cache for near optimal sizes with the cache organization given. The R4000 also comes in a version without secondary cache support. Removing secondary cache and assuming a relatively simple main memory (60 ns access chips, 2-way interleaving, and 10 ns logic and propagation delay) reduces performance significantly. At optimal sizings, the 2-level caching provides a 15% performance improvement, a few percent more at equal cache sizing. Future versions

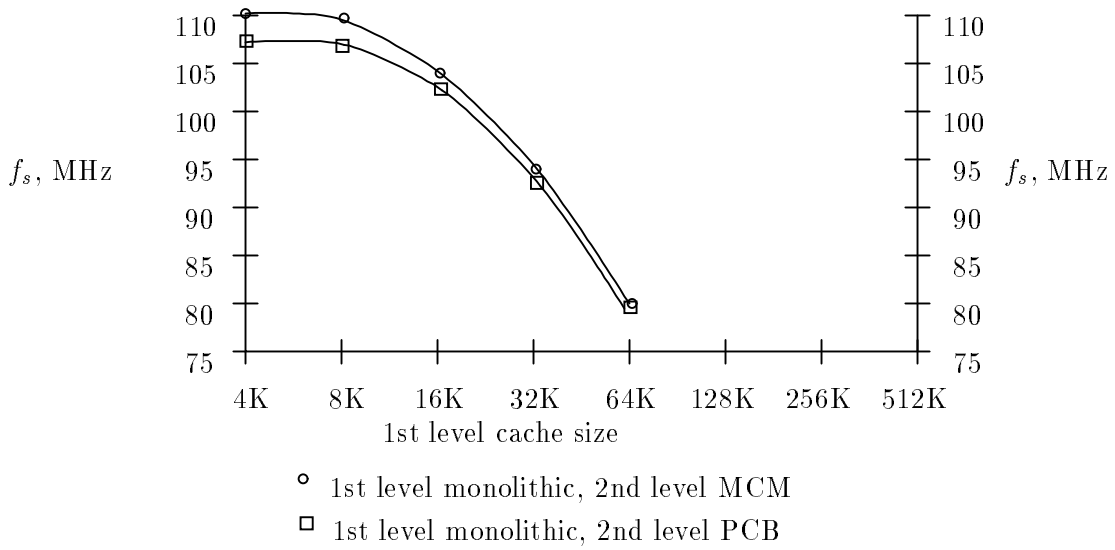
trace pitch of .0635 cm (25 mils) and .1" PGA pin centers. The processor core is composed of 9 die, including the instruction cache and the 4 data cache chips. The remaining chips are the FPU, 'FXU' (primarily integer processor), MCU, and IOU (input output). The multiple processor chips would normally require the inclusion of inter-chip propagation delays. However, block diagrams show that the system has been carefully partitioned so that the only chip crossings on critical paths are between are between the FXU (or FPU) and cache, as included in our model. Clock skew is closely managed over the processor core by the use of impedance controlled lines of balanced length and loading. Critical path for the processor was taken as the path latch – register file – comparator – multiplexor – latch, with delay for each item taken as the average of minimum and maximum values from the IBM published figures. The processor is not tied up during cache line refill. We therefore modify our system model slightly. For the other case studies we include the full refill time, for simpler microprocessor implementation. For the IBM RS/6000, *we return to Higbie's method of taking the average delay for access contention (processor versus refill) as* $\frac{1}{2}$ *the busy time, the factor of* $\frac{1}{2}$ *being returned to the term for cache refill time in the* C_{et} *calculation.* Main memory is 4 words wide, with 4 way interleaving. Access time was based on the published through put of 400 M bytes per second and the 16 byte width.

Model results are close to the actual system performance, too close to be claimed as accuracy for the model. The actual PCB area for the processor core is 361 cm^2, our model result 389 cm^2, only an 8% difference. This is particularly encouraging, as the area is wiring dominated so that the SUSPENS footprint calculation is included. Setting main main memory access time to 0 ns in order to determine clock cycle time gives 41.7 MHz. This is a negligible difference from the 41.5 MHz cycle of the recent Models 550 and 950, but almost 40% faster than the 30 MHz of the other models. The published delays for the processor macros, from which our processor critical path time was estimated, have an almost 2 to 1 range. Using the maximum times, our model projects a clock speed of 37.1 MHz, 23% faster than 30 MHz and 11% slower than 41.5 MHz. It is not known why the macro delays vary by so much, nor to what degree IBM may have derated clock speeds of earlier models for reasons such as reliability and manufacturing tolerance. A sizing analysis resulted in an optimum instruction cache size of 16 K bytes (for a constant 64 K byte data cache size), twice that of the actual system, but with a performance difference of less than 2%. Recall that f_s includes the effect of cache misses, but not superscalar execution. Interpolating MIPS figures for 30 MHz models indicates 56 MIPS for a 41 MHz system.

Limited data was also available to assess transfer of the design to MCM. Published data indicates a line pitch of 26 μm for the IBM MCM process, roughly twice of the SUSPENS generic data. Assuming that line widths were also doubled, and approximately the same thickness, we estimated one-half the line resistance and twice the line capacitance. Similar to our earlier sizing case study, we kept data cache at 64 K bytes, and varied instruction cache size as shown in Figure 4.5a. As before, MCM offers significant speed-up, but at the expense of much larger caches. It is expected that IBM will switch to MCM only in conjunction with a die scaling, to give a 100 MHz system ([Cos91]. Using a scaling to .35 μm (the current RS/6000 being 1 μm CMOS) was applied to access times and die areas, with results shown in Figure 4.5b. Here, data cache size was held constant at 128 K bytes for PCB implementation and 512 K for MCM. Performance difference for MCM is considerably improved, nearly 50% better than PCB, including larger cache sizing for MCM. At the same instruction cache size of 16 K bytes, MCM offers a 20% performance improvement. At 1 μm, the RS/6000 system core is much too large for a monolithic implementation, almost



(a) Refill Width 8 Bytes, Both Levels



(b) Refill Width 8 Bytes First Level, 16 Bytes Second Level

Figure 4.4: Two Level Cache Performance

instruction cache access time. Assuming balanced cache access times, the *4 ns estimated difference between the instruction cache and the larger data cache chips was added as a new term, T_l, into the effective access time (C_{et}) of the instruction cache.* As for the other case studies, output driver chain delays were deducted from the access times to permit more accurate and detailed modeling of interconnection delays.

The data cache is 64 K bytes, comprised of 4 chips, with 4-way set associativity and a line size of 128 bytes. Other data was also taken directly from the references, including PCB

great where "3 dimensional" memory packaging (stacked die) provides even greater improvements, and may even permit main memory on the MCM for small systems. Pin constraints also play a role in determining the utility of multi-level caches.

It is possible to analyze performance of 2-level cache using the current spreadsheet. The mixed (data and instruction) cache miss rate formula is substituted in to the model, and C_{et} calculated (the rest of the model is ignored). The resulting C_{et} can then be input into the regular spreadsheet as if T_{mem} for the main memory, and the first level of cache modeled. Here we take HP PA-RISC based model and use the same memory parameters for both levels of cache.[5]

Figure 4.4 summarizes two of the four possible combinations of first level cache on- versus off-chip and second level on-MCM versus on-PCB. With the first level cache on-chip, optimum size is now smaller (4 to 8 K bytes), and performance is increased over a single level cache by about 15%. Note also that placing secondary cache on MCM versus PCB has a smaller effect, around 5% with constrained refill width, less with wider refill.

Placing primary cache on-MCM results in a cache sizes approximately that of the secondary, which is interpreted as meaning that a 2-level cache is not appropriate for this case. This is disappointing, as placing both levels on MCM with flip-chip die mount would greatly relieve refill width constraints, increasing performance. A 2-level cache with both levels on MCM may none the less be advantageous when faster (and perhaps lower density) primary cache chips are used. The design space is very large and must await further analysis, ideally with an explicit 2-level cache model allowing interplay of sizing between the first and second levels and cache organization, along with more detailed die and driver modeling.

4.3 IBM RS/6000

We can further explore cache issues, providing data points at a few more locations in the large design space, with a case study of the 32 bit high-performance version of the IBM RS/6000 system. With a multiple chip superscalar processor, we are stretching our cache model, but results appear reasonable and only a few additional approximations were introduced. Design and performance data were taken from several sources, [BGM90,Gro90, BW89,SRM+89,TSP+89,Cos91,IBM90]. Numerous parameters differ from the previous case studies.

The instruction cache is 8 K bytes, with 2-way associativity and a 64 byte line. The processor is superscalar, and 4 words are fetched from the instruction cache each cycle and several instructions can be issued and executed each cycle. It is not clear that the cache miss ratio model is applicable in this case, although coefficients described by Higbie as appropriate for contemporary 'parallel scalar' processors are used. This is a particularly important area for future model calibration against trace based simulations. Note that our current model computes only the the cycle time, adjusted for cache misses, irrespective of cycles per instruction. The IBM instruction cache chip includes branch processor logic, so area for the cache alone was estimated for input to our spreadsheet. Cache speeds were estimated from our empirical correlation derived from [SMF+91]. Our estimate for the data cache chip coincided exactly with IBM's published data, so the correlation was used for the

[5] An actual cache would likely use faster memory at the first level, slower at the second, and still slower main memory for reasons of cost. Here we use the same speed for both cache levels to emphasize benefits without respect to reducing the size of the more expensive faster cache.

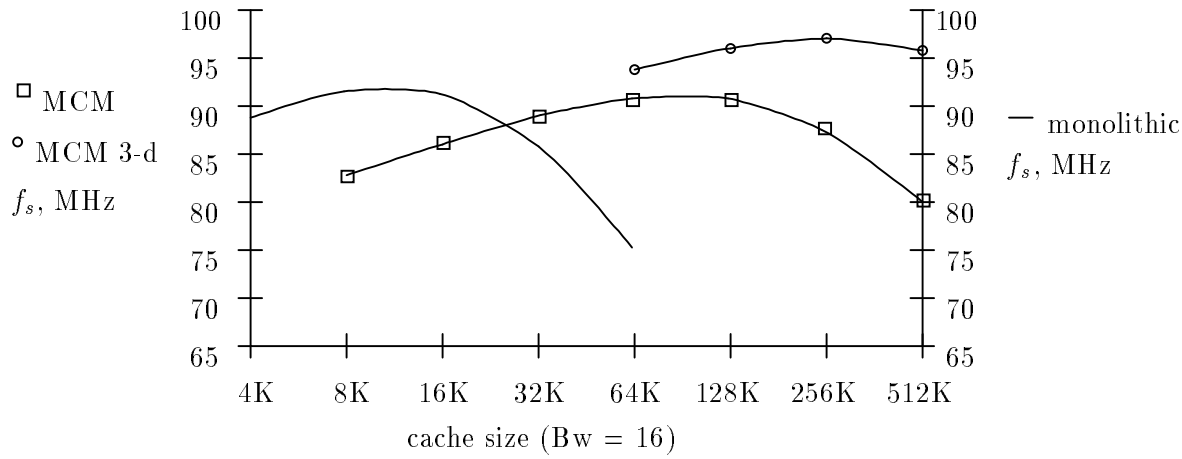


Figure 4.3: Enhanced Monolithic and MCM, .35 μm

as well as the cost-performance goals of a particular design. Early analysis tools should prove useful in guiding the development of particular technologies, identifying crucial versus less important technology parameters. In most cases, relative performance between these two alternatives is not great so that yield and other cost-related issues may dominate any choice.

4.2 Two-Level Cache

Having considered on- versus off-chip cache for systems with a single level of cache, we now consider how the trends differ with a second level of cache providing much faster refill times for the first level. Cache is often thought of as primarily a cost saving measure, exploiting locality of reference to allow a small memory of expensive fast chips with a much larger backing memory of slow inexpensive chips to perform as if the entire memory was comprised of fast chips. There are also physical constraints, however, which irrespective of cost force the use of cache and may lead to the use of multi-level caches. Larger memory requires larger area, resulting in increased propagation delay (T_{pd}). In many cases T_{pd} alone would exceed access time requirements for the memory, and a cache would be required irrespective of chip speed. In order to meet miss rate requirements, the resulting cache may be so large that even for the fastest SRAMs available, T_{pd} plus memory access time exceeds processor timing requirements, necessitating an additional level of cache. For example, the 4 ns access time for the University of Michigan system limits cache size for the available 32 K bit GaAs chips. It is not always physically possible to get a cache large enough and "close enough" to meet both hit time and miss rate requirements in order to meet the effective access time required by the processor. As Hennessy puts it, multilevel caching is required when hit time limits the size of the primary [HJ91]. This is particularly important for very high performance designs where advanced technology with relatively lower integration levels (such as GaAs) is required for speed, and where power dissipation limits packing density. [Men91] reports a trend towards a first level on-chip cache with a second level off-chip cache, the first level cache taking advantage of higher integration to reduce propagation delay and cost.

Offering significant reductions in propagation delay, chip-crossing penalty, and increased packing density, MCMs may alter these design trade-offs. Differences could be especially

for architectural enhancements such as more advanced multipliers or memory management units, superscalar and other architectural advances. This is particularly so as the miss rate curve flattens out for larger cache sizes. A further discussion of this issue, as well as other benefits of monolithic implementation (such as quicker system design) can be found in [HJ91].

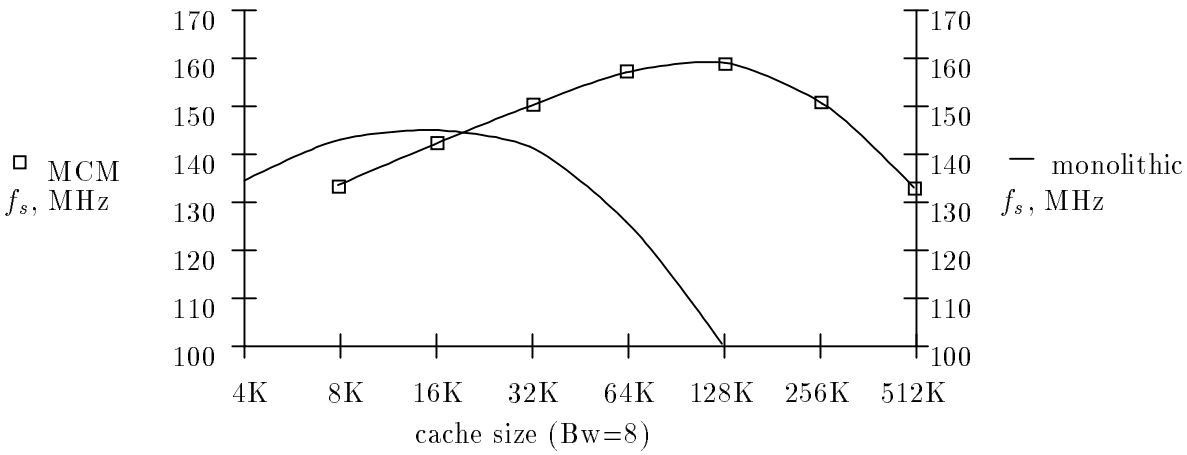


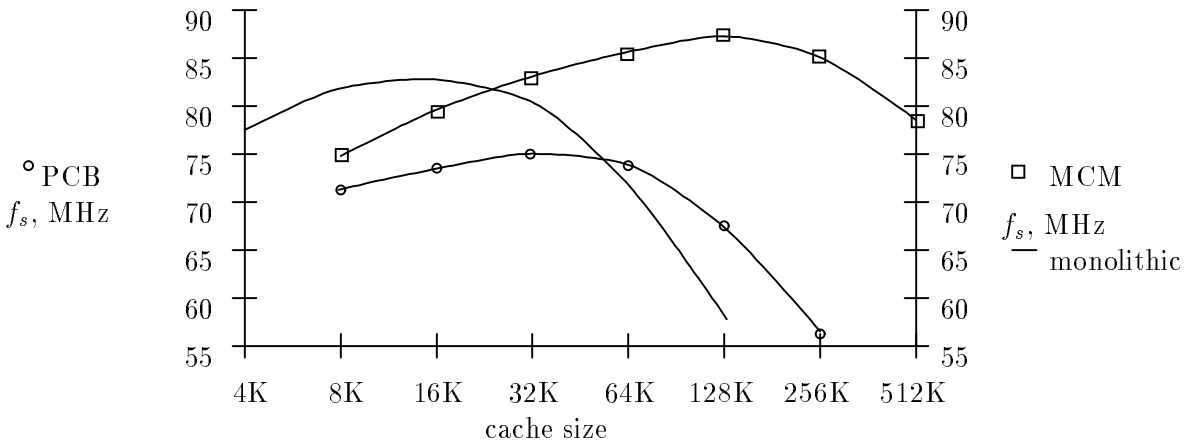
Figure 4.2: Comparative Performance, CMOS System at .35 μm

As mentioned above, VLSI designers have several options to improve monolithic performance, such as BiCMOS drivers, intermediate drivers along long lines, and additional metallization layers. MCMs also have a few more aggressive options. For MCM pad capacitance, we have been assuming 1 pF, corresponding to a more or less conventional input-output design with full esd (electro-static discharge) protection. Esd protection can be reduced, eliminating as much as $\frac{3}{4}$ of the pad capacitance. '3-D' memory, where DRAM die are stacked 16 or more chips high, can greatly reduce interconnection lengths and the associated delay.

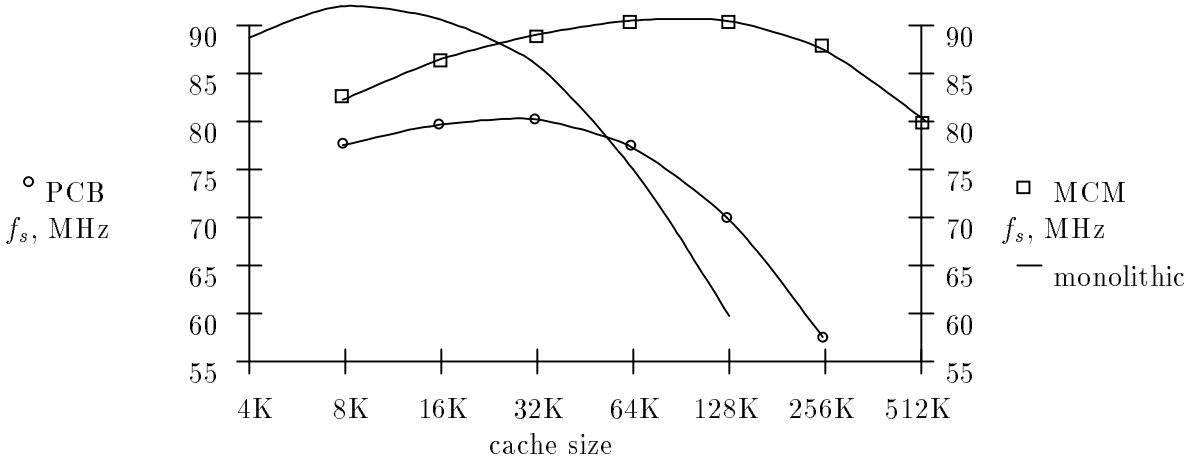
For higher power cache SRAMs, we take a stack of 8 die as an arbitrary limit for thermal dissipation. Even at .05" vertical spacing, such a stack would have a height of less than 1 cm, with propagation delay of less than .2 ns. We can therefore neglect the vertical interconnection and approximate the 3-D memory as die with 8 times the density and 8 times the pad capacitance. Reducing the esd protection as well gives $(\frac{1}{4})(8)(1) = 2$ pF equivalent pad capacitance. We can also approximate the monolithic enhancements by using the WSI parameters from SUSPENS and values for R_{tr} and C_{tr} which approximate the intrinsic gate delay for bipolar drivers. This gives the results shown in Figure 4.3.

The figure shows that more aggressive MCM packaging can indeed raise performance to exceed that of monolithic implementations, but at the cost of much larger caches. Applying BiCMOS drivers to both the monolithic and MCM cases raises the performance of each, with MCM still offering higher performance with larger cache sizes. If, however, we modify the monolithic VLSI parameters by adding a final layer of thick metallization for long interconnects, as is done in some WSI processes, the relative performance switches – monolithic implementation shows some 3% higher performance with similar cache sizes.

We have seen that with variations in VLSI and MCM technology parameters and cache organization, either MCM or monolithic implementations can show higher performance. The relative merits of each can be highly dependent on the evolution of the two technologies,



(a) Refill Width 8 Bytes



(b) Refill Width 16 Bytes

Figure 4.1: Cache Sizing and Performance, Monolithic versus MCM

One can also consider the effect of minimum feature size scaling. The SUSPENS tables scale R_{int} almost linearly[4] with λ, and D_c scales by $\frac{1}{\lambda}$. With L_{int} proportional to D_c, this implies that propagation delay will remain approximately constant as chips are scaled down. If the MCM feature sizes are held constant as scaled die are used, MCM performance should increase somewhat in relation to monolithic. This is confirmed in Figure 4.2 for the prior case at B_w of 8 bytes for instruction and data cache, scaled to .35 μm. At the larger cache sizes, MCM performs some 9% better with .35 μm die than with .7 μm. At smaller, equivalently sized caches, MCM still has lower performance than monolithic, but the gap is closing. Greater die sizes with higher transistor counts resulting from processor architecture enhancements may further close this gap. This suggests an additional consideration for monolithic cache sizing. Although future process technology should provide sufficient transistor count for large on-chip caches, it may prove preferable to use the transistors

[4]With some variation from directly linear, around 20% at a 2:1 scaling, which can be compensated by other scaling effects such as gate capacitance, and by circuit design.

To a first approximation die area, including tags, is directly proportional to S, with 2 to 1 variations of R and associativity resulting in 1 to 4% differences in area.

In this section we look at systems with a single level of cache, comparing performance for on-chip (monolithic) versus off-chip (on MCM) implementation. As sufficient information regarding die technology parameters were unavailable for the University of Michigan GaAs system, the following analysis was based on the HP PA-RISC system. Resulting cache sizes will be smaller than for conventional microprocessor systems due to the PA-RISC's advanced, high bandwidth main memory subsystem. Based on the number of transistors and die size, we estimate minimum feature size (λ) to be 1.3μm. Even with 2 chips for the processor (CPU and FPU), die area is quite large (approximately 1.7 cm^2). Further, our memory area and models are based on chips currently in use, approximately .7 μm in the instances used. Scaling memory up to the 1.3 μm for a monolithic processor would clearly penalize performance. We therefore scaled the processor to .7 μm.

Our model incorporates only overall parameters for the CPU, such as die dimension (D_c) and critical path delay (T_{cpu}), and these were directly scaled for analysis at smaller feature sizes. Cache memory die area and access time were also directly scaled with feature size, after removing output driver delay as described for the P/I model. For the monolithic case, a smaller memory 'tile' of 4 K by 8 bits (versus 8 K by 16 bits for off-chip) was chosen to allow smaller on-chip cache and better sizing resolution.[1] The parameters P_w, N_w, P_p, Z_o, and vm were taken from the SUSPENS data. C_{tr} and R_{tr} were derived as previously discussed.[2] In all cases, data cache and instruction cache were the same size to simplify analysis and presentation of results.[3] As set associativity improves performance for smaller caches, 2-way set associativity was used as a middle ground.

The results are summarized in Figure 4.1. Refill width had a significant effect, so 3 parametric runs are shown. The implications will be discussed later, in terms of pin constraints. Monolithic implementation is clearly superior to on-PCB cache in all cases and sizes. The case for monolithic versus MCM implementation is not so clear, however. At small cache sizes ($\leq$ 16 K bytes each for data and instruction cache) monolithic provides higher performance, but high on-chip line resistance as well as slightly higher line capacitance causes performance to fall off much faster than for MCM at larger caches. Depending on refill width, a larger off-chip cache on MCM may offer 10% higher performance, at the expense of caches up to 8 times larger. For the same cache sizes (8 to 16 K byte), on-chip cache shows some 10% higher performance, in line with [Men91].

Cache set associativity can also effect the relative performance. BiCMOS output drivers could improve MCM performance, but could be also be used in long-line drivers in monolithic implementations.

---

[1] Note that the off-chip access time is around twice that of the processor. Here we are making a trade-off between larger die access time versus more die and higher pin capacitance. Including propagation delay, the mismatch between cache access and processor critical path is less than 10%.

[2] Prior runs did not adjust C_{tr} and R_{tr} to the specific minimum feature size (λ) for processor and memory, but instead used an intermediate 1 μm. In the prior studies, this was irrelevant, as the values are used only for driver delay, which was deducted from chip delays, then the exact same number was added in as part of interconnection delay. Here, we will be exploring differences in λ, and the refinements are included.

[3] Reduction of data cache size from the larger optimum represents only a few percent reduction in performance and may in fact be the most cost-effective sizing.

4. Extended Case Studies

Several analyses were made extending beyond the original framework of the spreadsheet model. Although this required additional assumptions and approximations, the studies serve to further show the large design space opened up by early analysis tools. These studies also begin to quantify the various trends and tradeoffs, and identified specific requirements and priorities for further tool development.

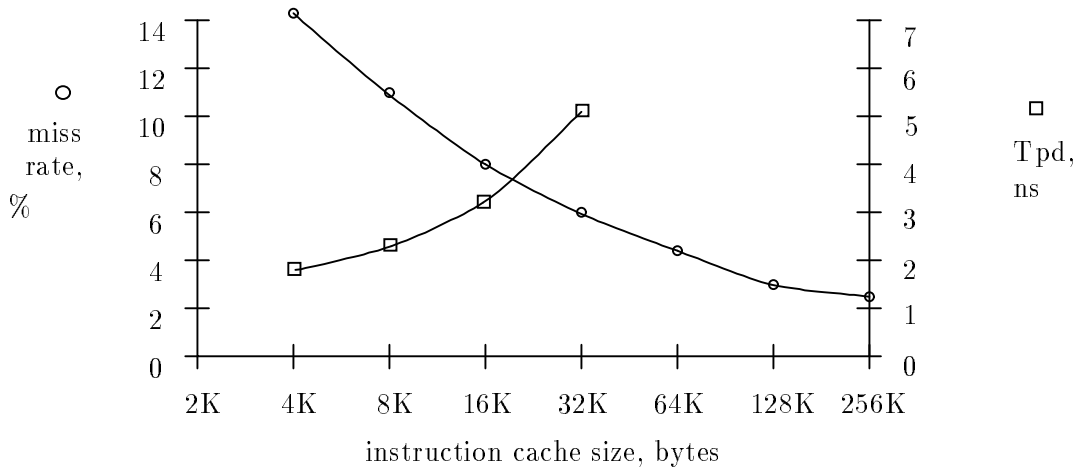
The results below suggest that MCM has a performance penalty $\leq 10\%$ over monolithic implementation, and in some cases may offer improved performance. These are, however, preliminary results, and the final vote must await more detailed models of memory and processor die speed and area, output drivers, and more detailed analysis of specific processor systems.

4.1 On-Chip Versus Off-Chip Cache

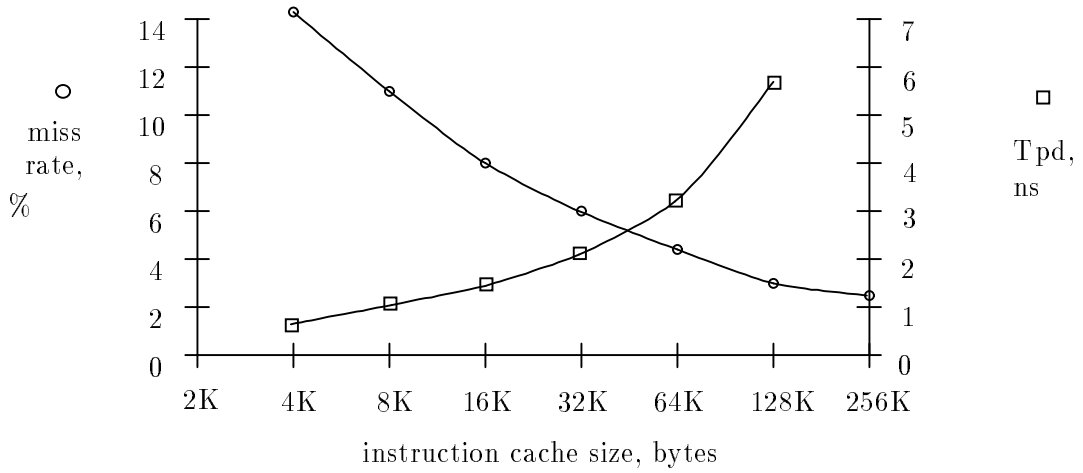
The choice between monolithic and multichip processors as discussed in [Men91] is currently a hotly debated issue. Several points concerning the placement of cache on or off the processor chip have already been mentioned, with a key issue being the size limitation with current levels of integration. Current on-chip caches, although perhaps adequate for single tasks and embedded applications, are clearly inadequate for multitasking workstations as seen in the prior cache sizing optimization. However, this will not always be the case. By the end of the decade, chips with 10 to 100 million transistors should be feasible, allowing on-chip caches of 128 K bytes to 1 M byte or more. As we will see, this should prove more than adequate, especially as a primary cache used in conjunction with an on-MCM secondary cache. With the reduced cost associated with the higher yield of smaller die, a CPU chip with cache controller and off-CPU cache memory chips may be more cost effective.

Other issues must also be addressed. Hennessy indicates on-chip cache will be faster than on-MCM, as a result of avoiding chip crossing entirely and utilizing wide on-chip buses [HJ91]. However, an on-chip cache will require a wide refill bandwidth to the off-chip secondary or main memory ([Men91]). As package pin count is expected to scale much more slowly than die device count, single chip pin count is likely to remain a severe constraint. Flip-chip mounting on an MCM may become the only way to provide the necessary pin count and routing capability required for a monolithic processor with on-chip primary caches.

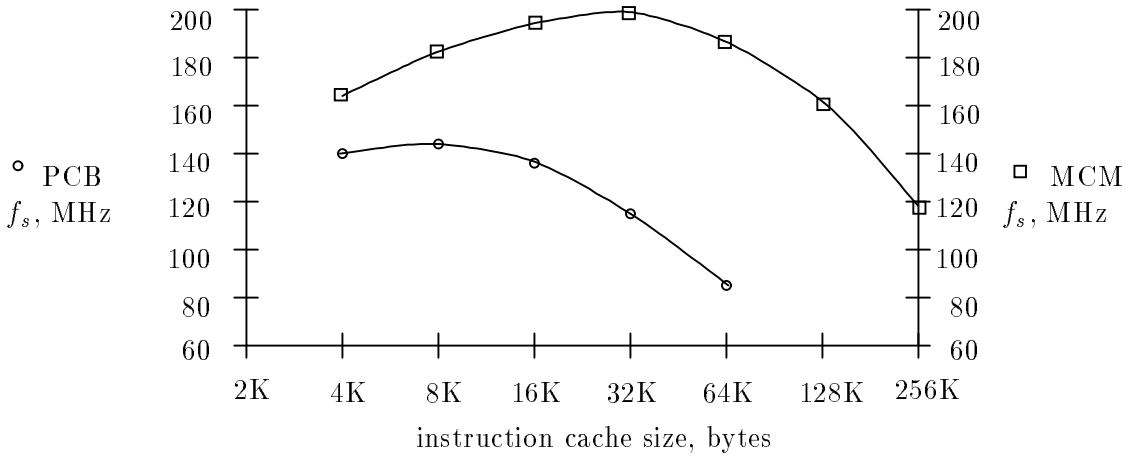
Having included all propagation delay terms in our interconnection model, as discussed previously, we can use our spreadsheet to evaluate monolithic (full processor and cache on a single chip) versus MCM implementation. For the monolithic microprocessor we treat the cache memory 'chips' as tiles (discrete area steps) on the processor die. Routing delay within a tile is included in T_{mem} (i.e. $T_{mem} = T_{logic} + T_{pd}$). A curve fit to memory access times given in [SMF+91] gave results in the form of $T_{mem} = a + b(\sqrt{B} - c)$ where B is total number of bits and a, b, and c are coefficients. This supports the first order approximation of a fixed (logic) delay plus an internal routing delay proportional to die edge dimension. The spreadsheet then adds propagation delay for interconnecting the tiles. By using interconnect parameters from CMOS die technology rather than MCM technology, performance and sizing of on-chip caches can be approximated. For the monolithic case, gate capacitance is used in lieu of pad capacitance. Die and tile areas are based on [MQF91], taking optimum S, R, and 2-way set associativity to give accurate area estimates for the mid-points of the parametric analyses.



(a) Miss Rate and Access Time, PCB

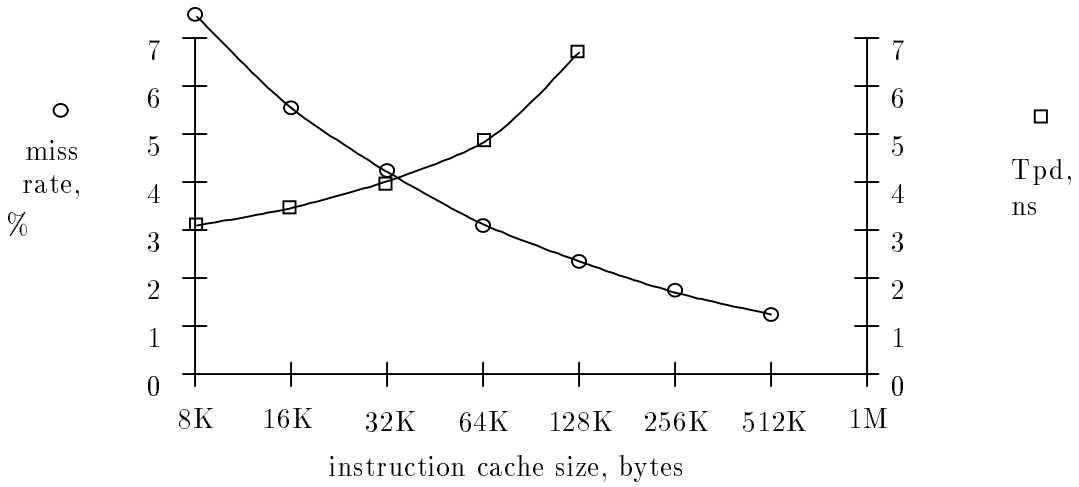


(b) Miss Rate and Access Time, MCM

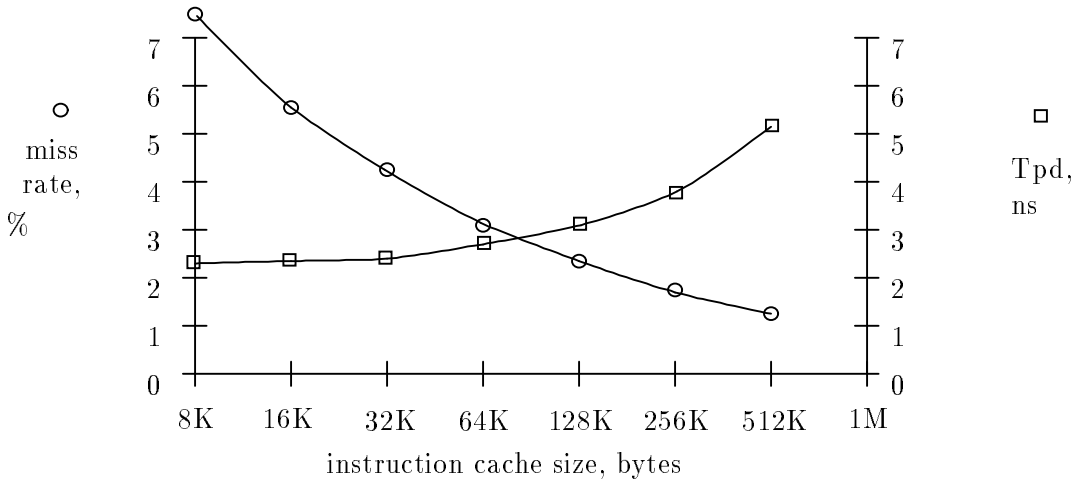


(c) Instruction Cache Optimization

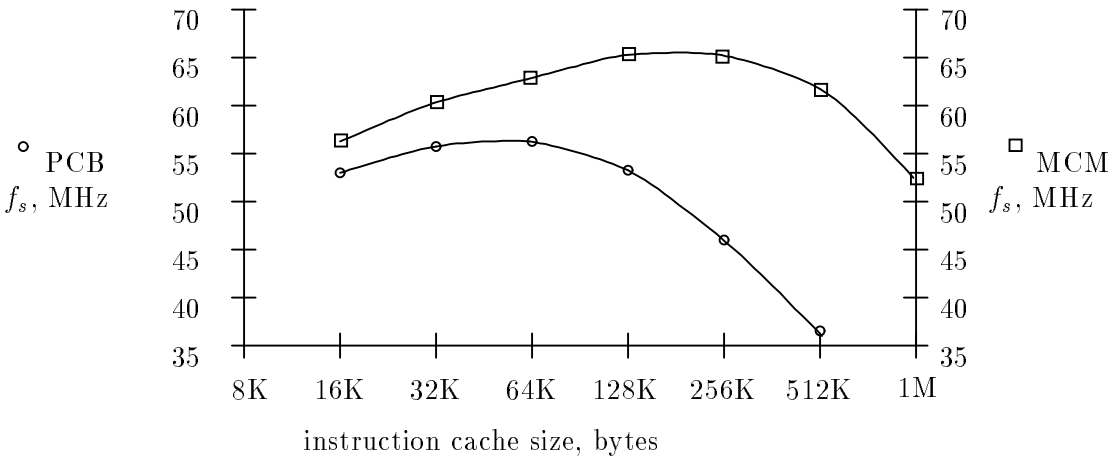
Figure 3.2: GaAs System Sizing Results



(a) Miss Rate and Access Time, PCB



(b) Miss Rate and Access Time, MCM



(c) Instruction Cache Optimization

Figure 3.1: CMOS System Sizing Results

widths, declining somewhat for large R and B_w. Optimization was made judgementally as the knee of the curve. This resulted in an R of 64 and 8 bytes for instruction and data cache respectively, B_w of 64 and 8 bytes, and a size of 64 and 256 K bytes. This optimization improved F_c from 65.6 MHz to 76.2 MHz, a 16% improvement. For the University of Michigan system, the starting assumptions (R of 16 and 16 bytes, B_w of 16 and 16 bytes, and size of 32 and 64 K bytes for instruction and data cache) proved optimum. Here, however, performance for R declined rapidly for line widths greater than 16 bytes, revealing a clear optimum. Model behavior for cache R and B_w can be complex. For R, performance often declined for increasing R, an effect sometimes called "cache pollution," and optimum line length for data cache was often equal to the data word length. Performance normally showed asymptotic improvement with B_w for data cache, but in some instances the curve was flat, and B_w equal to data word size was used.

rate for larger caches is offset by increased propagation delays resulting from longer signal lines and higher capacitive loading from pads for additional memory chips. This results in optimum cache sizes smaller than one would choose from miss rates alone. For MCM packaging, propagation delays are smaller and increase more slowly with cache size than for PCB. This results in optimal cache sizes being larger for MCM packaged systems. For the GaAs system, we ignore potential signal integrity problems of operating a PCB around 150 MHz and simply compare cache sizing and potential performance improvement. For both systems, the reduced propagation delay from MCM technology resulted in larger cache sizes being optimal. Analysis resulted in optimal cache sizes much smaller than 'conventional wisdom' would suggest. Note, however, that in addition to including propagation delay into the analysis both systems have very fast backing memory for cache misses. In the case of the HP system, an advanced main memory provides very high refill rates, and the University of Michigan system has a second level of cache.

For the 66 MHz PA-RISC system on PCB, minimal TPI occurred with a 64 K byte instruction cache and 128 K byte data cache. With MCM packaging, the sizing rose to 128 K byte instruction and 256 K byte data caches. At the smaller caches, f_s was 56.3 MHz for PCB, 63.6 for MCM. For the larger caches, f_s was 65.6 MHz for MCM. Without resizing cache specifically for the MCM environment, the technology change alone gives a 13% speed improvement. Resizing for MCM offered a 3% further improvement.

Performance improvements are more noticeable for the 250 MHz Michigan system. Here, generic MCM parameters were used rather than the BCB technology. In the PCB environment, optimal cache size was modeled to be an 8 K byte instruction cache and 16 k byte data cache. For the MCM environment, the results were 32 K bytes and 64 K bytes, respectively. For this system, the switch to MCM without cache resizing results in an 25% improvement and the "multilevel optimization" resizing results in an additional 11% improvement.

3.6 Cache Line Width, Refill Bandwidth

In addition to access and propagation times, memory hierarchy bandwidths are also important and may benefit from the reduced pin constraints and increased routing capacity offered by MCMs. Typically, propagation delay is most critical in the communication between cache and the processor (to minimize cycle time), and data width most critical between cache and main memory (to minimize refill time). At the first level, a cache line provides much of the functionality of an instruction or data prefetch. For a large R, much of the prefetch may go unused, however. Actual prefetch methods seek to avoid this penalty, and are clearly preferable over simply increasing R. Further, a larger line width reduces miss rates, but increases refill time on a miss and also reduces the number of lines in the cache, increasing conflict misses. This results in an optimum point, beyond which performance is reduced. Up to a point, increased refill width was seen in the parametric study to significantly improve performance, or alternatively permit the use of slower memory in the secondary cache or main memory. For a full cache optimization, parametrics were run iteratively for S (size), R (cache line width), and B_w (refill width) for both instruction and data caches for the HP system on MCM. Here we address performance irrespective of interconnection constraints.

For the HP PA-RISC instruction cache, performance continues to improve asymptotically with increases in R and/or B_w. For data cache the performance was level for small

and number of pins estimated. An iteration of the spreadsheet was used to calculate output driver delay, which was then deducted from memory access and cpu critical path times. The Michigan processor utilizes a two level cache, so an effective access time was calculated using the system model equations discussed previously. The MCM parameters were taken mostly from [KSB+91] and [OBL+91], for a Polycon BCB (benzocyclobutene) technology utilizing flip-chip die attach. Other parameters were taken from [Bak90].

As the processor utilizes GaAs direct-coupled FET logic, the SUSPENS CMOS output driver model was retained, with R_{tr} and C_{tr} adjusted for GaAs versus silicon. This does not appear to have introduced any significant errors, projected T_{pd}s being within 20% of that reported in [KSB+91]. A design in progress, data from the Michigan group's most recent publication was used in instances of design and analysis differences.

The details of our analysis are given in Appendix A. Here we compare results of our first-order model with those of their post layout SPICE circuit and detailed architectural simulations. Direct comparison is more difficult for this verification, as their architectural model combines cycles per instruction (CPI) with cache misses whereas our model includes only cache miss effects and not the effects of instructions which require more than one cycle. At their optimized first level cache sizing of 16 K bytes for instruction and data caches, their simulations predict a cycle time of 3.85 ns, ignoring secondary cache. Setting "main memory" cycle time to 0 in our model gives TPI 3.51 ns, a 9% difference. At their cache sizing, our effective system frequency, f_s, was 188.2 MHz. Their system performance figure of 155 MIPS includes both cache miss and instructions requiring more than 1 processor cycle. Working backwards from their CPI and cache miss simulations gives an estimated 1.37 CPI excluding cache miss. Adjusting our f_s by this factor gives 137 MIPS for a 11% error. Our individual cache miss rate projections differ significantly from their simulations, however. Our model projects miss rates of 8% and 3.8% for instruction and data caches respectively, whereas their simulation results in 1.8% and 6.9% respectively. The differences may be attributable to differences in work load and task switching as well as model error. A parametric run on cache sized gave an optimum of 32 k bytes instruction and 64 k bytes data cache when effective access time to the secondary cache is included, with a 3% difference in system performance.

3.5 Cache Size

One of the most significant parameters for memory hierarchy design is of course cache sizing. It is vital that the cache be large enough reduce the miss rate to a tolerable level. Otherwise, the miss rate penalty (main memory access and cache line refill) becomes a key bottleneck and performance suffers significantly. If it is too large, propagation delay can dominate. Historically, small on-chip caches have been backed up by larger caches off-chip. High miss rates for restricted on-chip cache sizes have resulted in the choice of off-chip cache as the sole cache on at least one occasion ([Men91]). By reducing the time penalty for chip crossing, as well as signal propagation delay, MCM can be expected to alter trade-offs in favor of larger off-chip caches.

Parametric runs were made to estimate optimum cache sizes for test cases based on the Hewlett Packard PA-RISC system, and the research system at the University of Michigan ([KSB+91]). Each case was analyzed for advanced (10 mil pitch) printed circuit board (PCB) and thin film MCM. Results are shown in Figures 3.1 and 3.2. In both figures the data cache size was held constant at the optimal sizing. For both systems, the reduced miss

change with altered cache size were greater than with parameter change alone (maximum of 25% versus 14% for A, and 15% versus 4% for$B$). This indicates these parameters are particularly important, and should be calibrated for particular processor architectures and work loads.

3.3 Hewlett Packard PA-RISC

The HP PA-RISC system was chosen as a verification point for under 100 MHz, with published design information readily available. The PA-RISC is a CMOS RISC microprocessor system, implemented on printed circuit board (PCB) technology, with large single level cache. With 128 K byte instruction cache and 256 K byte data cache, it is reported as operating at 66 MHz.

Our model verification analysis is given in Appendix A. Cache and system architectural parameters were based on [Hig90] as described above, with cache line sizes estimated at 16 and 32 bytes for instruction and data cache respectively. Additional system information was taken from [FMD+91] and [HJT+91], with processor critical path time estimated from CPU frequency. Cache chip parameters were based on Texas Instruments data sheets, combining data and tag storage, with die size estimated. An iteration of the spreadsheet was used to calculate the output driver delay, which was then deducted from the memory and cpu times. Die information for the CPU and FPU were taken from [FMD+91], and averaged for two uniform chips as input to our first-order model. Packaging and interconnect parameters were based on aggressive PCB technology at 10 mil wiring pitch and 50 mil pad pitch with 5 pF per pad.[1] R_{tr} was from MOSIS fabrication run data, and C_{tr} was determined as described previously. The remaining P/I parameters were taken from [Bak90]. Package footprints assumed PLCC chip carriers at 50 mil pitch.

At the given cache sizes (128 k byte instructions, 256 k byte data), the model predicted an operating frequency of 53.7 MHz versus the published 66 MHz. Our "system frequency", however, includes the effect of cache misses. Ignoring cache misses by setting main memory access time to 0 gives 61 MHz for an error of $7\frac{1}{2}$%. Using the model to optimize instruction cache size gave an optimum of 64 K bytes for instruction cache and 128 K byte data cache, versus the published 128 and 256 K bytes, but the difference in system performance between the two sizes was less than 5%. These differences may be attributable to system parameter assumptions, workload and task switching differences, and main memory access times in addition to model accuracy. Our model input for main memory access time was based on sustained throughput, and does not explicitly model the PA-RISC systems 155 ns setup latency.

3.4 University of Michigan RISC

An experimental GaAs direct-coupled FET RISC processor under development at the University of Michigan ([KSB+91,OBL+91]) was used as a verification case above 100 MHz utilizing MCM technology.

Cache and architectural parameters were taken from [Hig90] as described above. Cache memory speed and organization were taken from [OBL+91] and [KSB+91], with die size

[1] As suggested in [HJ91], MCM should be compared to other advanced alternatives, not "business as usual."

importance of this parameter and the importance of having a cache system that does not block the processor on write misses. As one would expect, processor and cache access times were also critical, but only affecting performance some 20% even for wide variations in time. Main memory access time had a similar degree of impact. Cache line (refill block) size and refill bandwidth are also important, with up to 20% influence on performance. As expected, 2- and 4-way cache set associativity is important for smaller caches (7 to 13% for 8 k byte), but less valuable for caches of hundreds of kilobytes (less than 2 percentage points improvement, at increased cost and access time penalties not considered here). This can be expected to vary widely with different workload assumptions, particularly those without multi-tasking. The system B parameter is also important (up to 15% influence), with the A parameter less critical (typically 4%). The architectural factor, A_f had only a 3% effect over the range of reasonable values.

Driver and propagation delay are also especially important. The off-chip driver transistor characteristics can influence performance up to 20%, although this was a somewhat extreme comparison (CMOS versus bipolar output driver chains, both sized to match Z_o). C_{pad} is also important, with a performance impact of up to 15%. Together, these suggested a closer look at the T_{dr} and T_{pd} models, as previously discussed. Subsequently, the initial model was modified to treat the multiple pads on the cache address and data busses as a distributed loads and additional capacitance was added to the C_{tr} from [Bak90] to account for drain and interconnect in addition to gate capacitance.

The number of cache memory chips, a result of device integration level (which will be lower for fast, advanced technologies such as GaAs) affected performance by some 3 to 6%. Chip footprint size affected performance by up to 5%, but only when packaging density was not routing limited.

Since memory die sizes were estimated and were not adjusted for different cache organizations, this was further investigated. A model for estimating total cache die area is presented in [MQF91]. Manual calculation of the area model agreed with our estimates to within 15%. [MQF91] indicates that tag, status bits, and control overhead cause total cache die area to vary by at most 20 to 30% with typical variations in line size and associativity. We therefore made an additional parametric run with die areas increased by 25%. This did not affect optimal sizing, but as expected performance declined faster for larger cache sizes. For the multichip module GaAs system, performance at optimal cache sizing was affected by only 1%, with only a 4% difference at instruction cache sizing 4 times the optimal. The PCB case was unaffected, single chip packaging dominating die size. Since caches are often implemented with separate data and tag memory chips, a parametric with 25% more memory chips was also made, for both MCM and PCB cases. Again, optimum cache sizing was not affected but performance at the optimum sizing reduced by 2 to 3%. At 4 times the optimum cache size, the die count change reduced performance 9 to 13%. Memory access time also changes with die size and cache organization, suggesting that these effects should be included in more detailed models.

Z_o also had influence up to 5%, largely through output driver sizing effects. Other parameters had less than 1% effect, and are mostly P/I technology factors.

In many cases, cache sizing was re-optimized to determine the effect of parameter changes on optimum cache size. With reasonable parameter changes, optimum cache size didn't change by more than a factor of 2 in any instance, and impact on performance was similar to, or less than the parameter change alone, cache size change often compensating for some of the parameter's effects. For the system parameters A and B, however, performance

3. Case Studies

This chapter presents a series of case studies using the first order model, beginning with a sanity check of the cache model followed by a parameter sensitivity study and an informal verification. Cache sizing, line width, and refill bandwidth are then explored.

3.1 Cache Model Checking

Derived from VAX simulations and having a large number of coefficients and architectural factors, the cache model was first checked against typical cache performance for RISC systems. The general literature (see [Men91], others) indicates that for these systems, cache miss rates can be as high as 25% for small on-chip caches, but can be reduced to .5-2.5% with large off-chip caches.

As covered previously, Higbie's coefficients and values for parameters were used throughout our model. Several were adjusted, in line with Higbie's discussion, to apply the model to RISC systems. A_f was taken as .9. A_f is given as 1 for the VAX, "approximately 1" for RISC, and ≤ 1 for scientific and engineering environments in [Hig90]. Similarly, β was advised as being 1 for read misses only, 0 for all cache misses, and ≤ 1 for larger register sets. A β of 0 is appropriate where a write miss does not stall the processor, and resulted in the closest calibration and verification. Also, in the mixed cache miss rate equation, the coefficient 3 was reduced to 2.8 and .75 to .73 following Higbie's suggestion that these should be somewhat less, but were rounded for "ease of remembering." Although we have not calibrated the model against trace based simulation of specific RISC systems, the discussion in [Hig90] leads us to believe the adjusted model and results are applicable. A greater variation might be expected from workload assumptions and trace sources and lengths than from the architectural parameters.

With these coefficients and parameters, miss rates of 22 to 29% were obtained for 4 to 8 k byte instruction caches with processor stalled on write misses, and $1\frac{1}{2}$ to $2\frac{1}{2}$% for 128 k byte instruction caches, in line with general experience with RISC systems. Closer agreement with published miss rates could be obtained with further adjustments but we did not wish to risk over fitting or make large changes to the coefficients and parameters. The full model was then verified against two systems, as described later.

3.2 Sensitivity Study

A parametric study was performed using a preliminary version of the spreadsheet to evaluate parameter sensitivity. This study served to indicate which options have the greatest impact and provide a better understanding of the model. In addition to identifying key issues for optimization, crucial model elements were identified for further verification and refinement. The base case was the 250 MHz University of Michigan system described below, with optimized cache sizing. This system represents an intermediary between current advanced workstations and future very high performance systems.

As one might expect, the most important parameter was cache sizing. Data cache sizing affected performance by as much as 33%, greater than even the combined differences of MCM versus PCB packaging (21%). Instruction cache sizing had a smaller impact, less than 11%. The cache model β also had a large effect, up to 27%, indicating both the

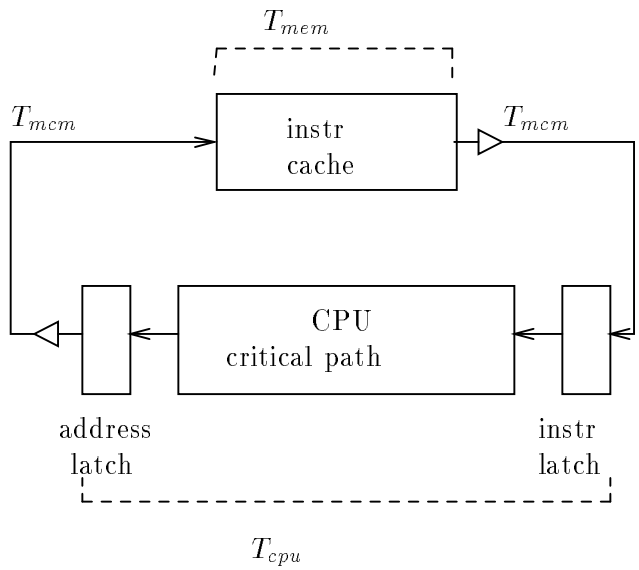


Figure 2.5: System Timing

to include the time to determine a cache miss before accessing main memory. The SUSPENS model includes the output driver chain, varying them with P/I technology choice. Their delay is therefore not included in T_{mem} and T_{cpu}. It is also assumed for the two-chip processor case studies in this report that the critical path does not cross a chip boundary, so that output driver and propagation delays between processor chips are excluded.

For a single level cache system, effective cache access time is then:

$$C_{et} = T_{mem} + 2T_{mcm} + C_{mr}\frac{R}{B_w}\frac{T_{main}}{B_k}$$

where B_w is the cache refill width (in bytes), T_{main} is effective main memory access time, and B_k is main memory interleave. The three terms thereby correspond to cache access time (excluding output drivers), output driver and interconnection delay, and line refill time.

System time per instruction becomes:

$$TPI = \frac{1}{p_f}(TPI_b + max(A \times C_{eti}, B \times C_{etd}) + \frac{T_{main}}{B_k}(A \times C_{mri} + B \times C_{mrd})$$

where p_f is the pipelining factor (number of stages), and A and B are architectural factors. The three terms multiplied by the reciprocal of the pipeline factor are processor critical path delay, TPI_b, effective cache access time, $max(A \times C_{eti}, B \times C_{etd}$, and main memory access time for cache misses. Higbie discusses A and B as being processor architecture and compiler dependent and ≤ 1. For contemporary superscalar RISC microprocessors ("highly parallel scalar"), Higbie gives values of .95 and .75 for A and B respectively. For a "fully serial" processor microarchitecture, they would both be 1.

System performance is denoted as f_s, the effective system frequency, defined as $\frac{1}{TPI}$ where TPI is time per instruction and includes cache misses. Since the cache miss rates are misses per instruction and TPI includes memory references, f_s, is a rough a approximation of system MIPS (millions of instructions per second), ignoring instructions which take more than 1 cycle, input-output, and virtual memory.

end. The number of input pads is taken as the SUSPENS coefficient for net fanout, F_c. This gives the approximation

$$(R_d + R_i)((F_c + 1)C_{pad} + C_i)$$

and multiplying out gives

$$R_d(F_c + 1)C_{pad} + R_dC_i + R_i(F_c + 1)C_{pad} + R_iC_i.$$

We observe that the output pad is not charged through R_i and that the line R_iC_i should be treated as a π-section. Letting $R_d = Z_o$ (driver matched to line impedance) and deriving R_i and C_i from per unit length values gives

$$Z_o(F_c + 1)C_{pad} + Z_oC_{int}L_{int} + R_{int}F_cC_{pad}L_{int} + \frac{1}{2}R_{int}C_{int}L_{int}^2$$

to which the same driver chain and time of flight terms as in the cache case are added.

2.5 System Model

The purpose of the system model is to combine the estimated propagation delays with the predicted cache performance and other architectural parameters to produce a projection of system level performance. In this study input-output restrictions on system performance are assumed to be a result of device speed and latency, rather than by P/I technology, and are ignored. Our system model is based largely on [Hig90], and also includes elements of the analysis in [OBL+91] and [KSB+91]. The SUSPENS system level model is too basic for our use, including only simple module power and frequency calculations, with system frequency simply the minimum of the module frequency and all chip frequencies.

[KSB+91] computes system frequency as $\frac{1}{2}(T_{mcm} + T_{cpu} + T_{mem})$, where $\frac{1}{2}$ is the cache access pipelining factor (1/number of stages), T_{mcm} is the round round trip propagation delay (address from CPU to memory, data from memory to CPU in the case of a read), T_{mem} is the access time of the cache memory, and T_{cpu} is the critical path delay through the processor. This is shown in Figure 2.5. As discussed in [OBL+91] and others, the critical path typically includes the conditional branch path, where information must pass through the instruction latch, register file, comparator, address multiplexor, and address latch. In a balanced processor design, it can be assumed that other critical path delays will be approximately equal, so a single T_{cpu} is used in our first-order analysis.

Higbie's system model bases a cache's access time on the miss rate and block refill contention, then combines these times with main memory access and other architectural factors. TPI is determined by the TPI_b (corresponding to the processor critical path delay of Kayssi) of the processor plus the memory hierarchy access time minus recovered time. Recovered time includes speedup from instruction decode overlapping miss penalties and reduced contention in an emptying pipeline after a miss. The model is more completely discussed in [Hig90], including extension to multi-level caches.

We combine the architectural model of [Hig90] with the technology (propagation delay) model of [OBL+91] and [KSB+91], and adjust the model to explicitly include refill bandwidth. We also extend the model to account for RISC microprocessors accessing data and instruction caches simultaneously (Harvard architecture). Higbie's model is also modified

Multiplying out gives

$$2R_dC_{pad} + R_dC_e + 2R_iC_{pad} + R_iC_e.$$

Referring back to the physical situation in Figure 2.4, it is necessary to correct the initial approximation. First, for the 3rd term of the above equation, we see that only one pad is charged through R_i so the factor of 2 should be removed. Second, the last term represents the distributed resistance and capacitance of the line. This can be more accurately modeled as a π-network of n sections, and taking the limit as $n \to \infty$ gives $\frac{1}{2}R_iC_e$.

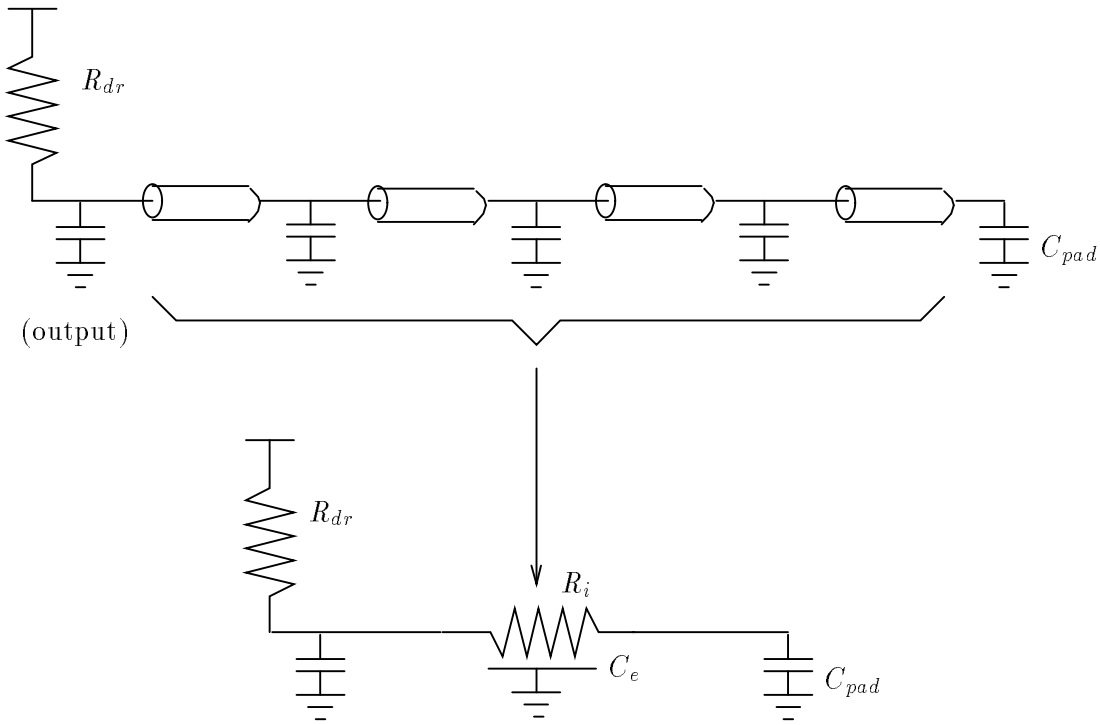


Figure 2.4: Propagation Delay Model

Adding in the output driver chain delay and time of flight from SUSPENS and calculating R_i and C_i from resistance and capacitance per unit length of interconnect and the previously computed L_{int} gives

$$T_{pd} = 15(N-1)R_{tr}C_{tr} + 2Z_oC_{pad} + Z_oC_e + R_{int}C_{pad}L_{int} + \frac{1}{2}R_{int}C_eL_{int}^2 + \frac{L_{int}}{vm}$$

where $R_d = Z_o$ for impedance matching and

$$C_e = C_{int} + \frac{(N_c - 1)C_{pad}}{L_{int}}.$$

Propagation delay is not added to T_{cpu} as it is assumed the critical path will be on a single chip in the case of the two-die processors used in the case studies. Where this not the case, a delay computation slightly different from that above can be similarly developed. Here we assume that multiple pads are not evenly distributed along the length of a line, and approximate by assuming one pad at the driver end and multiple input pads at the opposite

2.4.1 Driver Delay

Propagation delay is of course a key parameter for this early analysis, as confirmed in a sensitivity study. On inspection, the output driver delay (T_{dr}) portion of the SUSPENS calculation appeared too small. The model however is essentially the same as in [PH91], multiplying the intrinsic delay of a minimally sized inverter times the number of stages times the sizing ratio between stages. The SUSPENS data table used in the prior analysis uses the capacitance of a single gate for C_{tr}, which the model then multiplies by 3 to account for both p and n transistors and their relative sizes. Examination of an inverter layout, as well as other timing models such as in [KV87] shows that drain and interconnect capacitances should also be included. All loading capacitances together for a minimal sized gate in 1 micron CMOS was estimated at 33 to 75 fF, which corresponds to the 50 fF value used in [PH91]. A value of 17 fF (50 fF divided by the SUSPENS model factor of 3) was used for later analyses. This gave results more in line with values such as in [SM91] and [PH91]. Driver delay increased more for PCB conditions than for MCM conditions, slightly increasing the predicted advantages for MCM.

2.4.2 Propagation Delay

A sensitivity study (see Section 3.2) also showed the importance of pad capacitance. Since the first model simply lumped the multiple pads of a memory array together, a more refined model is desirable for the caches. Further, in preparing for the analysis of monolithic (cache on the processor chip) versus MCM (cache separate from the processor chip) implementation, a discrepancy between the SUSPENS die propagation and P/I propagation models was noted. The P/I equation, presented above, omitted two terms present in the die model, one for charging the line capacitance through the driver transistor and the second for charging the load (pad) capacitance through the line resistance. Although negligible for printed circuit boards, these terms could be significant for MCMs. Also, by including all terms, the same model can be used for both on-and off-chip cache propagation delays. The following derivation treats the intermediate bus pads as distributed capacitance and more closely matches typical values for line delay, such as in [SM91] and [PH91]. The model also showed good agreement with SPICE simulations of series terminated lines at critical damping, with a single terminal pad.

When the driving resistance is greater than or equal to the line characteristic impedance, a transmission line in a CMOS system can be approximated by

$$(R_d + Ri)(C_l + C_i)$$

where R_d is the thevin equivalent of the driving transistor, C_l is the load capacitance, and R_i and C_i are the resistance and capacitance of the interconnect line. Multiplying out this approximation and adding terms for output inverter chain delay and time of flight gives the SUSPENS die propagation delay model.

For a cache memory array, intermediate pads on the address, data, and control lines give the physical situation depicted in the upper part of Figure 2.4. Our model adds the intermediate pad capacitances[6] to the line capacitance, giving an equivalent interconnect capacitance, C_e. Including the driver output and final load pad gives

$$(R_d + R_i)(2C_{pad} + C_e).$$

[6]Or gate capacitance in the case of monolithic implementation.

a more detailed discussion. As described above, average interconnection length and delay are calculated separately for the instruction cache, data cache, and processor.

Average wiring length for calculating area requirements included in the die "footprint" is:

$$\overline{R}_M = \frac{2}{9}(7\frac{N_c^{\eta-0.5}-1}{4^{\eta-0.5}-1} - \frac{1-N_c^{\eta-0.75}}{1-4^{\eta-0.75}})\frac{1-4^{\eta-1}}{1-N_c^{\eta-1}}.$$

Where there are less than 2 chips in a subgroup, $\overline{R}_M$ is taken as 1.

The footprint is:

$$F_p = max\{D_c, \frac{\overline{R}_M N_p P_w}{E_w N_w}, \sqrt{N_p}P_p, FP_{cc}\}$$

for the caches, with the assumption all lines are buses (no trace fanout), and

$$F_p = max\{D_c, \frac{F_c}{F_c+1}\frac{\overline{R}_M N_p P_w}{E_w N_w}, \sqrt{Nc}P_p, FP_{cc}\}$$

for the processor (the original SUSPENS equation).

The subsection dimension (instruction cache, data cache, or processor) is simply $D_m = \sqrt{N_c}Fp$ and the average interconnection length for each given our typical floorplan is:

$$L_{int} = 1.5Dm_{cache} + .5Dm_{proc}$$

Propagation delay was originally calculated as the reciprocal of SUSPENS's f_m, as follows.

$$T_{pd} = 15(N-1)R_{tr}C_{tr} + 2C_{pad}Z_o + \frac{1}{2}R_{int}C_{int}L_{int}^2 + \frac{L_{int}}{vm}.$$

The first term represents the output driver chain delay, the second the capacitive loading of the pads, the third the distributed RC of the interconnect line[4], and the last term the time of flight. N is the number of stages in the output drivers, estimated as $1 + \ln(\frac{R_{tr}}{Z_o})/\ln(5)$. Output driver chain delay is thus based on the intrinsic delay of a minimum sized inverter times the number of stages. The factor of 15 is an implicit assumption of pMOS transistor sizing of twice that of an nMOS transistor ($R_{tr}C_{tr}$ being for a single nMOS transistor) which multiplies capacitance by 3, and an assumed factor of 5 size multiple between stages.[5] It is also implicitly assumed that the final output stage will be matched to the impedance of the interconnect, hence the use of Z_o in determining N and in the second term of the delay equation.

Initially, our model increased the number of pads to coincide with that of the bused memory chips. Inspection of results of the parametric study revealed omissions in this model, which were corrected as discussed below.

---

[4]The term 1/2 was omitted in the text of [Bak90].

[5]The ideal factor for minimizing delay is e, but a factor of 5 greatly reduces area with negligible delay penalty.

2.4 Packaging and Interconnect Model

Given the number of die with their associated speed and area, and a P/I technology, the P/I model must compute propagation delay (T_{pd}) for each of the caches separately. Ideally, global placement and topological routing could be performed for key die and buses, with other estimators for control and other lines, followed by by circuit simulation on the resulting line lengths, termination, etc. For our initial pre-netlist model, we choose estimation methods based on SUSPENS ([Bak90]).

The SUSPENS module model was modified, however, to be specific to RISC cache characteristics and to calculate propagation delays for the two caches and the processor separately. First, different chip sizes are allowed for instruction cache, data cache, and processor. Necessary wiring capacity for estimating chip "footprints[3]" are also calculated separately. Secondly, rather than basing average interconnect length on Rent's rule, a particular layout is assumed, as shown in Figure 2.3. This is topologically the same layout used in [KSB+91], and typical of processor–cache layouts. The instruction and data caches are calculated as separate, uniform arrays of chips. Area is based on the respective die footprints, and an aspect ratio of 1:1 is assumed. Lines are assumed to be data or address buses as shown in Figure 2.3, with control lines routed similarly. This gives average interconnect length as $1.5\sqrt{A_c} + .5\sqrt{A_p}$ where A_c is the total cache area and A_p the total processor area. For multilevel caches, the typical layout can be applied recursively.

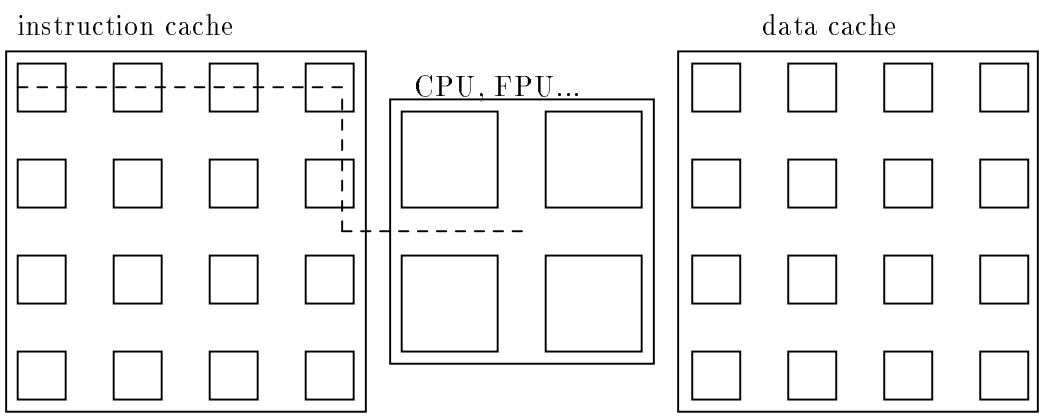


Figure 2.3: Cache Interconnection Length

In addition to the die specification, the P/I model uses the following user inputs. FP_{cc} is die footprint, in cm, which is typically the packaging dimension for printed circuit board single-chip packages, and the die size plus manufacturing clearance for MCM. A system-level Rents' coefficient, η, is also input for estimating wiring capacity area requirements for each die. F_c is chip output fanout, and E_w is an estimation of routing efficiency. Choice of P/I technology (MCM, PCB, etc.) and feature sizes determine the parameters $P_w, N_w, P_p, Z_o, C_{pad}, R_{int}, C_{int}$ and vm as described in [Bak90]. R_{tr} and C_{tr} (minimum sized transistor resistance and capacitance respectively) are also input, for use in estimating number of driver stages and the associated delay.

As with the cache model, we will simply present the model equations here, with brief mention wherever they differ from those of SUSPENS. The reader is referred to [Bak90] for

[3]In the SUSPENS P/I model, die footprint includes the routing area associated with each die.

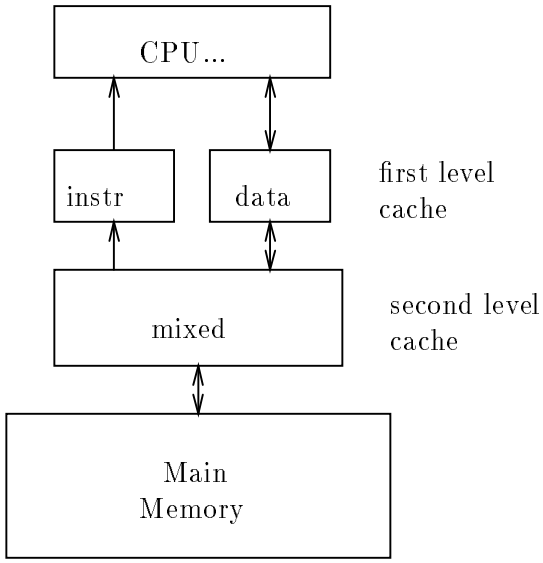


Figure 2.2: RISC Memory Hierarchy

D_c	die size, cm
N_p	number of pins
K_b	cache chip depth, kilobytes
W_b	cache chip width, bytes

Table 2.1: Die Input Parameters

2.3 Die Model

Our current tool does not include a die model. Use of existing chips is assumed, and Table 2.1 shows the inputs for instruction cache, data cache, and processor chips. The number of cache memory chips (N_c) for the instruction and data caches are calculated from the total cache size and the input cache chip depth and width. It is assumed the cache controller is included in the processor chip(s).

For systems using custom die, estimations of the above parameters (as well as power and other metrics) would be needed. It would also be necessary to incorporate models of die area, access time, and pin count based on cache organization (associativity, line width, etc.) to more accurately investigate these trade-offs. This will be particularly important for evaluating whether or not cache should be integrated into the processor chip.

The use of fixed die parameters with simple calculation of number of chips from cache total size has a potential for introducing errors. Small caches with large R and B_w may violate the memory chip organization (data width and number of words). Although total die area would be approximately correct, pin count could differ significantly. The pin count adjustment discussed later for model enhancements was tried in a few instances. The cases studies are not routing limited, however[2], with most cases showing no change. As the largest change observed was well under 1%, our current model was not modified.

[2] N_p is used only in computing die foot print area, based on routing capacity needs.

For a data cache:

$$C_{mrd} = \{1 + \beta(s-8)\}A_f.75^{s-3}\{1 - \frac{\alpha}{(10 + \frac{r}{2} - s)^2 + 2.5}\}.$$

For a mixed data and instruction cache the coefficients were adjusted slightly from those in [Hig90] as described in the Model Verification section:

$$C_{mrm} = 2.8A_f.73^{s-3}\{1 - \frac{\alpha}{(10 + \frac{r}{2} - s)^2 + 2.5}\}.$$

It is important to remember that for these equations and throughout this report *cache miss rates are per instruction, not per memory reference.*

User input includes an architectural factor, A_f, cache size (S) in bytes, and line width (R) in bytes. Two other factors are also input. The first, α, is a flag indicating cache set associativity. An α of 0 indicates direct mapped. Two way associativity is indicated by an α of .5 for instruction cache, 1 for data cache, and 1.5 for mixed cache. For four way associativity, the corresponding values are 1, 1, and 1.5. The second, β, is 0 for read misses only and 1 for all cache misses. As Higbie discusses, $\beta \leq 1$ is appropriate for processors such as RISC with large register sets, and $\beta = 0$ where a cache write miss does not stall the processor.

Although based on the VAX architecture and workload, Higbie discusses the model's applicability to RISC based systems and the appropriate parameter adjustments. In particular, the large register set for RISC processors reduces the number of memory references per instruction[1], so that the key architectural parameter, A_f, is approximately the same as for the VAX. As mentioned, workload and language have significant effect. The VAX workload used in Higbie's study is more typical of multi-user computing (by including context switching, and operating system) than a single engineering or scientific application suite. The results can therefore be taken as appropriate for general workstation use, but not necessarily for long single-task runs of computationally intensive scientific or engineering codes, or compilation.

Figure 2.2 shows a two level cache system typical of high performance RISC systems. Separate instruction and data caches are used at the first level, to provide simultaneous instruction and data access. Some systems use a secondary cache, to further buffer between the very high speed processor and much slower main memory. In the current spreadsheet implementation of our system model, a single level of separate instruction and data cache is assumed. A two-level cache system can be modeled by calculating an effective access time (as described in the System Model section) using the above mixed cache miss ratio formula, and input as if main memory access time. [Hig90] presents a full model for multi- level cache systems. For Higbie's multi-tasking work load, set associativity offers little performance advantage over direct mapped at cache sizes above 64 K bytes, while introducing significant design complexity and additional logic delay. Most of our analyses therefore assume direct mapped caches, unless small cache sizes are anticipated. Although appropriate for our early analysis research, the model should be calibrated to a particular work loads, RISC architecture and approximate cache design (via trace based simulation) before use as a design development tool.

[1] Most data reads and writes are met by the register set.

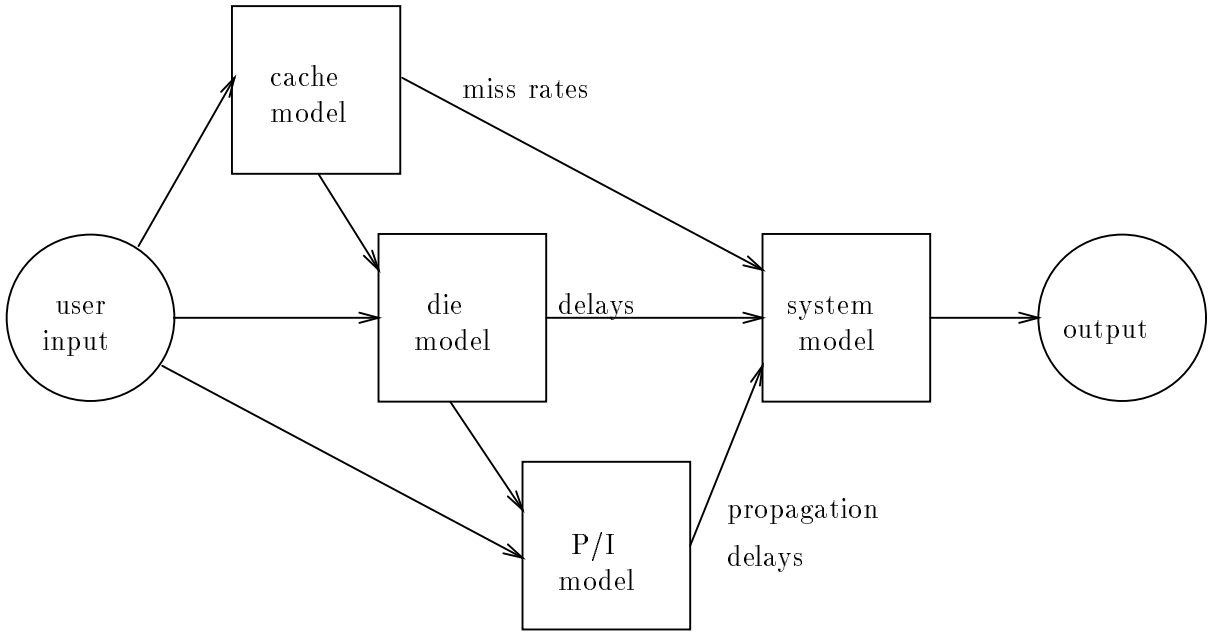


Figure 2.1: Overview of Model

2.2 Cache Model

The cache model predicts miss ratios based on user input of cache size and organization. Using memory trace based simulations would present several problems. Cache performance is highly dependent on system workload (single task scientific computation versus multi-user business applications processing are somewhat extreme examples). In addition to the often difficult task of obtaining or synthesizing an appropriate trace, formatting the trace data to be compatible with a particular system simulator can be a significant effort. Although trace based simulations of a million or so instructions require only a few minutes execution time, tens to hundreds of millions of instructions are often needed to properly characterize a multi-tasking workload and the number of parametric runs for an investigation can be in the hundreds of combinations. We believe it is beneficial for a system designer to have access to an interactive analysis tool (response time in seconds) to make initial explorations and develop a general understanding of are the key design issues, as a preface to more detailed trace based simulations later in the design effort. For these reasons, we have chosen to use the empirical analysis method presented in [Hig90] to provide quick, interactive parametric analysis. [Hig90] discusses the derivation and justification of several closed-form equations for estimating cache miss rates as a percentage of instructions executed. Rather than describe the derivation in detail, the cache equations are presented below, and the user is referred to [Hig90] for a detailed discussion.

For an instruction cache, the miss ratio is:

$$C_{mri} = 3A_f.7^{r-3}.75^{s-3}\{1 - \frac{\alpha}{(10 + \frac{r}{2} - s)^2 + 2.5}\}$$

where $s = \lg S$.

2. Analysis Method

This chapter presents the analytical model. After presenting the organization of the model, each section is discussed in detail. The cache miss-rate model is based on empirical studies by Lee Higbie, with coefficients adjusted for RISC based system. A die model is not currently included, and parameters are directly input. The interconnection model is based on Bakoglu's SUSPENS, and a new trace length model typical of memory chip arrays is given. Intermediate pads on the data, address, and control buses are taken into account, extending the SUSPENS model. The system model combines the miss-rate and propagation delays into a system performance estimate, also based on Higbie's work, modified to be more specific to RISC systems.

2.1 Overall View

Our early analysis model builds on prior work by Lee Higbie at Digital Equipment Corp. ([Hig90]), H.B. Bakoglu of IBM ([Bak90]), and researchers at the University of Michigan ([OBL+91,KSB+91]). These researchers summarize system performance as a function of cycle time, T_c, which depends on device and interconnect delays, and cycles per instruction, CPI, which depends on processor architecture and cache miss rates. As shown in Figure 2.1, our early analysis begins with estimation of cache miss rates, followed by logic and propagation delays from the die model. The packaging and interconnect (P/I) model then estimates system propagation delays. The miss-rates and delays are combined in a basic system model of performance, which includes other architectural parameters. The input parameters and outputs of each model are discussed in the following sections. The current analysis tool is a first-order model implemented as a spread sheet, which offers some important first looks at the many technology–architecture interactions of RISC system memory hierarchies. Second order effects, such as the impact of cache line size and associativity on total memory die area, pinout and access time, are not included in this initial model.

The cache performance analysis is based on correlation methods presented in [Hig90] for predicting miss rates based on cache size, line width, refill width, associativity, and write method. We make minor adjustments to the architectural coefficients to more closely match RISC system characteristics, as recommended in [Hig90]. Our packaging and interconnect (P/I) model is based on SUSPENS ([Bak90]), which incorporates empirical and first order physical models. We modify and extend the P/I model in several ways. First, we permit different die sizes and footprints for the instruction and data caches and CPU. Interconnection lengths are based on a typical floorplan rather than random placement. SUSPENS uses only gate capacitance in computing output driver delay, whereas our model also includes drain and interconnect capacitance. The model is also extended to account for the numerous pads on each address, data, and control line in a memory array. Their capacitance is treated as part of the distributed line capacitance. The model for system performance is also based on [Hig90]. We extend the model to combine driver and propagation delay in the effective cache access time and base line refill time on actual refill data path width. A factor is added to account for the pipelining of instruction and address latching, and the model is modified to account for simultaneous access to instruction and data caches in RISC architectures.

choices, and the effects of yield on cost performance. Variations on the cache systems for the IBM RS/6000 and the MIPS R4000 are also considered. Future development to more properly address these issues and further refine the model are given in Section 5. The final section summarizes our results, which demonstrate the benefits of utilizing an interactive analysis tool to quickly explore a wide variety of options very early in a system's design. Our case studies suggest that MCM will lead to larger off-chip caches. These preliminary results also suggest that the performance penalty for placing caches on MCM rather than on a monolithic microprocessor will be relatively small and that high performance systems are likely to benefit from a small on-chip cache and a large secondary cache on MCM. The increased routing capacity provided by MCM, when combined with the greater pin out capacity of flip-chip mounting, can also improve performance through architectural changes such as larger cache line size and refill bandwidth.

1. Introduction

This report presents an early analysis tool for RISC cache systems. Using a rapid prototype of the tool, we explore the implications of designing RISC system memory hierarchies specifically for MCM. We define early analysis as the evaluation of system trade offs, including implementation technology effects, at pre-netlist stages of design development. This permits analysis to begin at the highest levels of description with little or no CAD entry. This encourages the consideration of architecture and technology interactions at the earliest feasibility and specification stages. Analyzing the outcome of design decisions before actually undertaking design and fabrication offers several benefits. Risk and design time can be reduced by identifying limiting factors earlier in the design development. Further, including technology and architectural interactions permits optimization of the whole system, not just at a particular design level. This is particularly important when utilizing new packaging and interconnection technologies such as multichip modules (MCMs) where for maximum performance and cost-effectiveness it will be necessary to do more than just "miniaturize printed circuit boards." As RISC microprocessors increase in speed, it becomes increasingly difficult to provide instructions and data fast enough and the cache becomes one of the critical elements of system design. MCM's reduced propagation delay and greater interconnection capabilities can significantly affect cache sizing and organization, mitigating much of this problem. Issues of signal integrity and portion of cycle time in interconnect delays suggest MCM technology will become critical as clock speeds pass 100 MHz.

Prior optimizations combining architectural and implementation issues have for the most part relied on post-layout simulation from extracted data, as in [KSB+91]. The goal of our work has been to provide quantitative analysis more refined than rules of thumb, but requiring only high-level, pre-netlist, information as input. Existing technology evaluation tools such as SUSPENS ([Bak90]) provide early analysis of device and propagation delay, but include only minimal architectural evaluation. Cache evaluation methodologies such [Hig90] provide system level modeling, but omit interconnection delays. Our analysis combines these two models, and adds new enhancements specific to RISC and cache. The combined analysis method utilizes correlation methods to predict cache performance and empirical wiring models with basic interconnection performance analysis are used to estimate packaging and interconnection effects. The cache miss rates, signal propagation delays, and device access and cycle times are then combined in a simple system model to predict effective cycle time. This approach includes packaging and interconnection technology effects in the architectural optimization of the memory hierarchy. The resulting first order model has been implemented as a spreadsheet and used in several case studies to provide a quantitative grasp of RISC system memory hierarchy design for MCM.

In the next section we describe the analysis model, which combines an empirical model for cache miss rates with a first order calculation of propagation delays. A system model combines these results with architectural parameters to estimate system performance metrics. Section 3 describes informal verification of the model (on the Hewlett-Packard PA-RISC system and an experimental system at the University of Michigan), and discusses the results of using the prototype tool to evaluate cache sizing, line width, and refill bandwidth. In Section 4 we make use of further input approximations to evaluate design trade-offs somewhat beyond the original intent of the prototype tool. This provides further exploration in the large design space which can be made available by an early analysis tool, and includes issues such as on- versus off-chip cache, two level caching, pin constraints on architectural

List of Tables

List of Figures

Contents

Early System Analysis of Cache Performance for RISC Systems: MCM Design Trade-Offs

James D. Roberts
Dr. Wayne W.-M. Dai

UCSC-CRL-92-02
3/6/92

Board of Studies in Computer Engineering
University of California at Santa Cruz
Santa Cruz, CA 95064

ABSTRACT

This report presents a prototype early analysis tool for exploring trade-offs in cache architecture and packaging and interconnection (P/I) technology for RISC microprocessor based systems. We define early analysis as the evaluation of system trade-offs, including implementation technology effects, at pre-netlist phases of development. Prior work in cache performance estimation and P/I modeling are combined and extended to be more specific to RISC systems. After describing the model, several case studies are presented. Although limited by the accuracy of the first-order model as well as by assumptions and estimations regarding input data, these studies indicate general trends and quantify several important trade-offs for multi-chip module systems. MCM characteristics favor larger off-chip caches, with improved performance as well as yield advantages. When combined with flip-chip mounting, MCM technology also permits cache system architectural changes which can significantly improve performance. The prototype is not intended as a design tool, but rather to demonstrate the utility and importance of an interactive early analysis tool, combining architecture and implementation technology issues. Suggestions for future tool development are made.