A Performance/Area Workbench for Cache Memory Design

Osamu Okuzawa and Michael J. Flynn

Technical Report: CSL-TR-94-635

August 1994

Computer Systems Laboratory

Departments of Electrical Engineering and Computer Science

Stanford University

Stanford, CA 94305-4055

Abstract

For high performance processor design, cache memory size is an important parameter which directly affects performance and the chip area. Modeling performance and area is required for design tradeoff of cache memory. This paper describes a tool which calculates cache memory performance and area. A designer can try variety of cache parameters to complete the specification of a cache memory. Data examples calculated using this tool are shown.

Key Words and Phrases: Cache memory, Design tradeoff, Memory modeling, CAD

Contents

List of Figures

List of Tables

1. Introduction

An efficient memory hierarchy is essential for high performance processor design. Processor performance has continued to improve relative to memory speed, with cache memory being the common method to bridge the speed gap. Indeed, many high performance processors use two level of cache memory, where the first level cache is usually on the same chip as the processor. As chip area is growing due to the improvement of chip technology, so is the amount of the cache memory on a chip.

Designers should consider the tradeoff of performance and cost of cache memory at the early stage of design. The performance and cost of cache memory are affected by cache configuration, size and other parameters. Usually designers predict the area and speed using conventional tabular data. In this report, a computer aided tool for cache memory design is shown.

2. Related Work

Studies have been done about cache memory performance, especially, the studies for the relationship between cache specification and cache miss rate [1][2]. Chip area is usually used for the index of cost. A modeling method is shown in [3]. The model in [3] is mainly applied for large cache sizes. Mulder shows a cache model which is based on area analysis of actual processors using a 'register cell' as a unit[4]. The area calculation in this report is based on his model. Recently many papers have been published about design tradeoff for architectural design, including several[5][6] that also use his model.

For the prediction of the performance, usually one makes a command level simulator and executes benchmark programs. But to execute benchmark programs on a simulator requires considerable time. Flynn develops a static calculation method for processor performance[7]. The tool in this report uses the performance model based on [7]. Fu shows an analysis and the estimation using the model[8].

3. Performance/Area Workbench

3.1 Tool target

The target of this tool is to calculate cache memory speed and area using designer cache specifications. As a designer completes the specification of a cache memory, using this tool he can try variety of cache parameters. To calculate cache speed and area, this program uses the memory model shown in [4] and [7]. The model summarized in section 3.4.

3.2 Input

The input data are cache memory and processor specifications.

Detailed items are shown as follows:

(1) Processor specification

architecture - L/S, R/M or R+M

instruction set length

branch probability

chip technology

cycle time

word access time

bus cycle time

memory cycle time

timing template

data access per instruction

(2) Cache memory specification

cache kind - unified or split

cache size

line size

associativity

replacement policy (random, LRU, FIFO)

write policy (CBWA, WT)

physical word size

3.3 Output

Output data are calculation results of area and performance.

baseline CPI

excess CPI with cache

cache memory area size

3.4 Model & Calculation method

3.4.1 Area calculation[4]

A unit 'rbe' is used for area, which is the area of 1 bit register. Following variables are used in this section.

line : line length in bit

size : cache size in bit

tags : size/line

transb : band width

tsb : tag bit

$$tsb = 1 + \frac{\gamma line}{transb} + \log_2(2^{30}\frac{assoc}{size})$$

(1) Set associative

Area = PLA + Data + Tag

$$= 130 + 0.6(line \cdot assoc + 6)(\frac{tags}{assoc} + 6) + 0.6(tsb \cdot assoc + 6)(\frac{tags}{assoc} + 12) \quad \text{[rbe]}$$

(2) Full associative

Area = PLA + Data + Tag

$$= 130 + 0.6((1 + \frac{\gamma}{transb})line + 6)(tags + 6) + 0.6(\sqrt{2} \cdot tags + 6)(\sqrt{2} \cdot (27 - \log_2(line)) + 6) \quad \text{[rbe]}$$

As 1 rbe equals to $2035\,\mu m^2$ for 2um technology empirically, the tool can adjust the actual area according to the target technology.

3.4.2 CPI calculation[7][8]

Processor speed is represented with CPI (Cycle Per Instruction).

CPI = CPI_base + Miss_penalty * Miss_rate

(1) CPI_base

CPI_base is CPI without effects of cache memory. This is calculated with the effect of (i) address and data dependency, and (ii) branch penalty. For (i) empirical data are used. For (ii), the timing template and branch rate, which are both given as input, are used.

CPI_base

= Throughput + data_dependency

+ address_dependency + branch_penalty

(2) Miss penalty

A memory access model is shown as follows;

Ta : Word access time, Tbus : Bus cycle time, Tc : memory cycle time

L : Line size per physical word, M : memory Interleaved mode

w: ratio of dirty line

Tline : Line access time

$$Tline = Ta + Tc[\frac{L}{M} - 1] + Tbus((L-1) \bmod M)$$

Tmmis : Memory processing time for read request

Tcmis : Processor waiting time for read miss

Tbusy = Tmmis - Tcmis

Cache Memory Interface Schemes

i) CBWA (Copy Back Write Allocate) policy

<Scheme 1>

When hit miss, write to write buffer, and then read line and processor executes.

Tmmis = (1+w)Tline, Tcmis = (1+w)Tline, Tbusy = 0

<Scheme 2>

When hit miss, write to write buffer and read line at the same time. Then processor executes.

Tmmis = (1+w)Tline, Tcmis = Tline, Tbusy = wTline

<Scheme 3>

When hit miss, write to write buffer and read line at the same time. The processor execute along the word access.

Tmmis = (1+w)Tline, Tcmis = Ta, Tbusy = (1+w)Tline - Ta

Miss penalty is represented by the following equation.

Miss penalty = (Tcmis + Tint)/cycle

Here, Tint is time due to the misses which occur during Tbusy.

$$T\,\mathrm{int} = (Miss\_rate \cdot \frac{Data\_reference}{I \cdot CPI\_base}) \cdot \frac{Tbusy}{cycle\_time} \cdot \frac{Tbusy}{2}$$

For split cache,

$$T\,\mathrm{int} = (I\_Miss\_rate \cdot \frac{Inst\_fetch}{I \cdot CPI\_base}) \cdot \frac{Tbusy\_i}{cycle\_time} \cdot \frac{Tbusy\_i}{2}$$
$$+ (D\_Miss\_rate \cdot \frac{Data\_fetch}{I \cdot CPI\_base}) \cdot \frac{Tbusy\_d}{cycle\_time} \cdot \frac{Tbusy\_d}{2}$$

ii) WT(Write Through) policy

<Scheme 1>

When hit miss, read line and then processor executes.

Tmmis = Tline, Tcmis = Tline, Tbusy = 0

<Scheme 2>

When hit miss, processor executes during reading words.

Tmmis = Tline, Tcmis = Ta, Tbusy = Tline - Ta

Miss penalty is represented by the following equation.

Miss penalty = (Tcmis + Tint + Twint)/cycle

Here, Tint is the same as in CBWA. Twint is interference due to the write during Twmem.

$$Tw\,\mathrm{int} = \left(1-\left(1-\frac{Data\_reference}{I \cdot CPI\_base}\right)^{\frac{Twmem}{cycle\_time}}\right)\cdot\frac{Twmem}{2}$$

Here, Twmem = max(Ta, Tc).

(3) Miss rate

Miss rate is calculated by the program developed by Steve Fu. This program uses DTMR tables and adjusts parameters such as associativity of the cache and architecture type of the processor.

(4)Instruction/Data contention

In case of unified cache, instruction and data access contention occurs, which causes an increased CPI.

Contention = (Instruction fetch/I)(Data access/I)/CPI

3.5 Tool implementation

The tool consists of two parts. One is a C program and shell script on UNIX Workstation. This calculates one set of cache data. Fig.1 shows the configuration. The other is Excel interface, which enables users to calculate multiple data concurrently and manipulate data easily. Fig.2 shows the whole process including Excel interface. Table1 shows the program size.

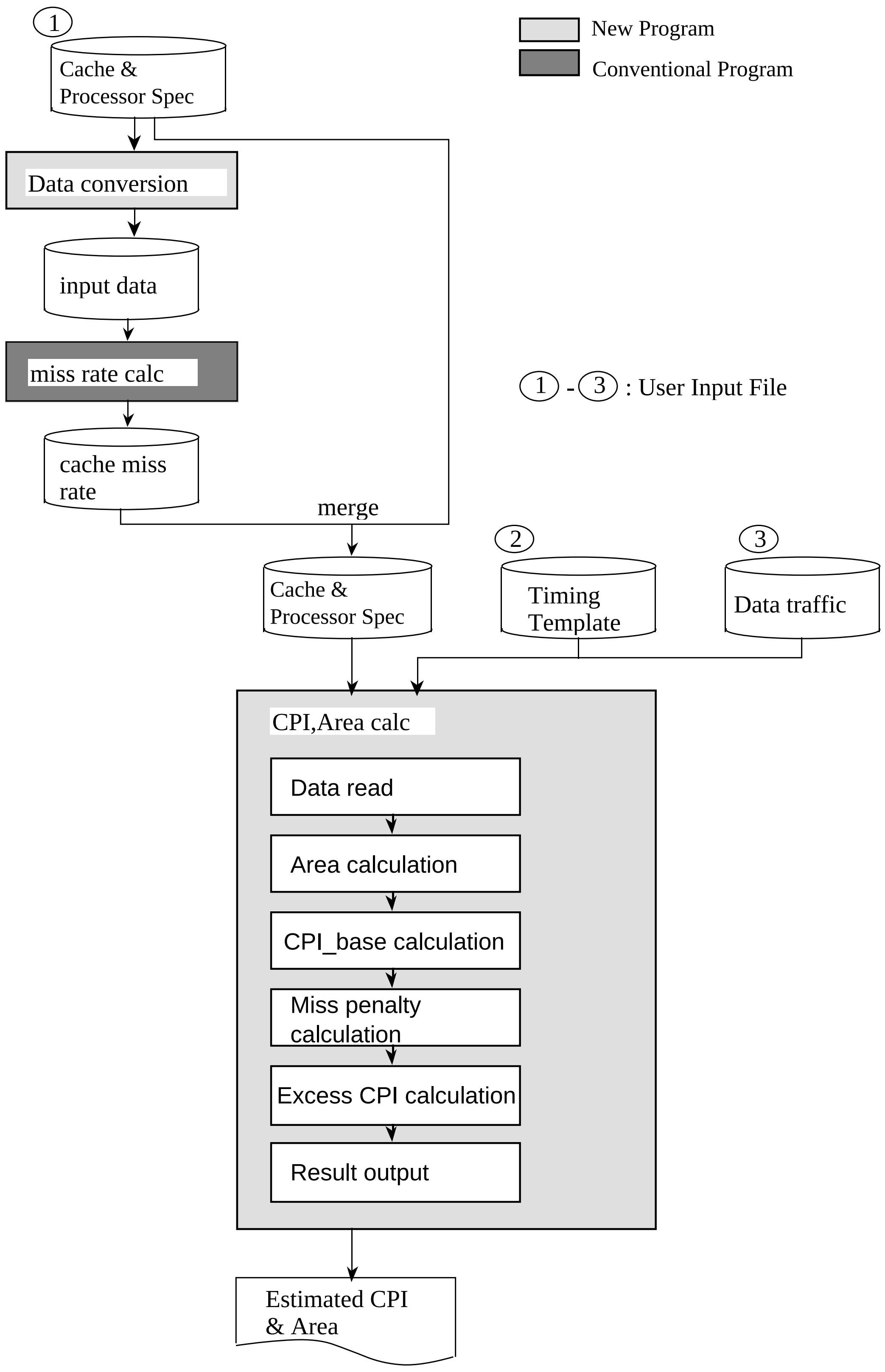

Fig.1 CPI and Area Calculation Program Configuration

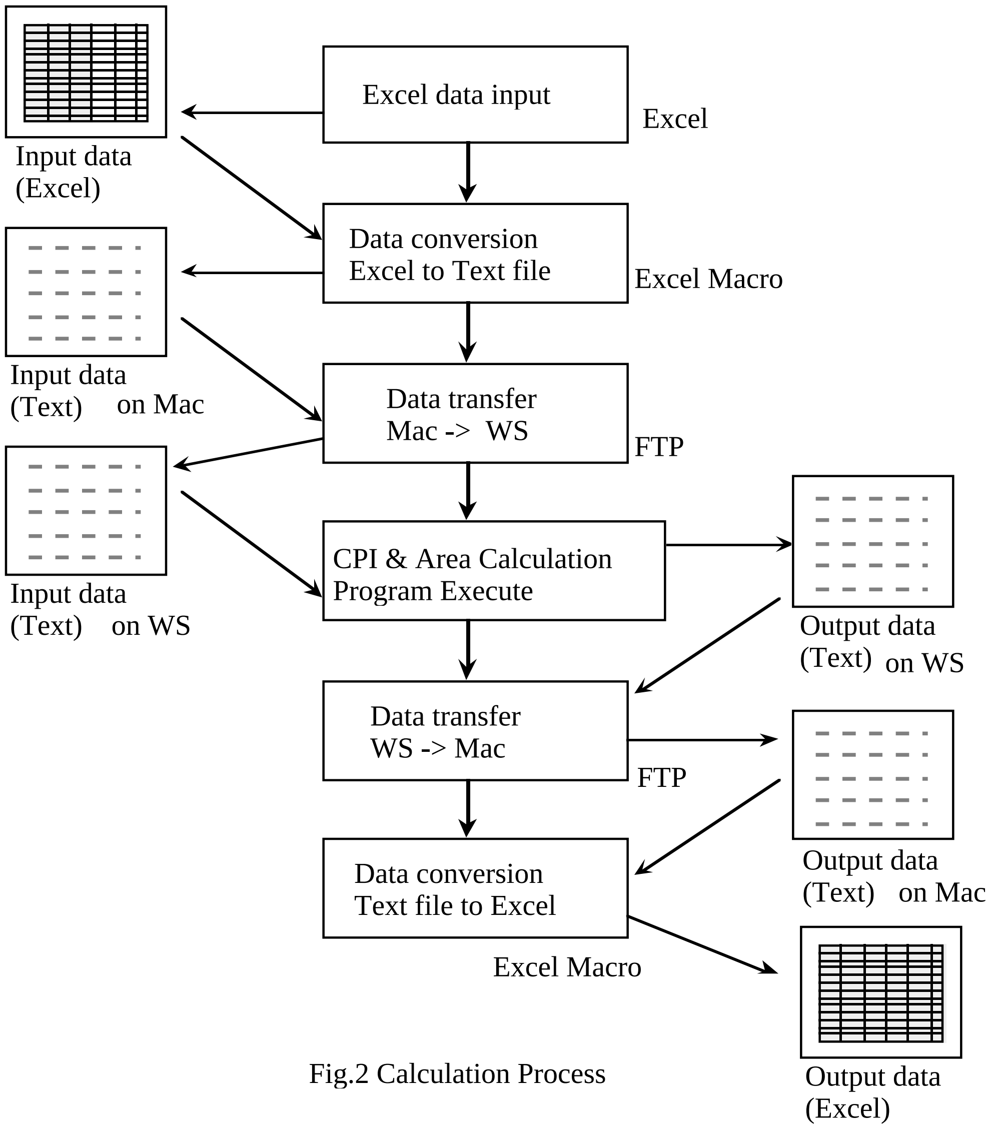

Fig.2 Calculation Process

Table 1 Program size

No	Part	language	No of lines
1	Data Conversion	C	200
2	Miss rate calculation	C	Already developed
3	CPI & Area calculation	C	800
4	Control, interface	shell, awk	50
5	Excel interface	Excel macro	20

3.6 Using manual

3.6.1 Operating Environment

Macintosh connecting with UNIX Workstation

3.6.2 Files or Programs needed

(1) on WS

datacon - program for data conversion

misscal - program for miss rate calculation

trout - program for CPI & area calculation

chexe1 - shell script 1

chexe2 - shell script 2

filediv - shell script 3

exeall - shell script 4

inawk - awk program

(2) on Macintosh

Fetch - application program for ftp

NCSA Telnet - application program for telnet

Microsoft Excel - application program

ex2tx - excel macro for conversion from excel data to text file

tx2ex - excel macro for conversion from text file to excel data

cashdata1 - excel data(template)

3.6.3 Calculation Process

(1) Making input data

Make a copy of "cashdata1". (File name is assumed "newfile")

The limitation of cache configurations which can be processed at the same time is 72.

Using Microsoft Excel on Macintosh, make input specifications on "newfile". After making data sheet, save and close the file.

(2) Convert the input file

Open "ex2tx", modify the input file name("newfile"), and execute it.

The macro create 3 text files: "input1", "input2", "input3"

(3) Data transfer

Execute "Fetch", and send the text files to WS.

(File names are assumed "input1.txt", "input2.txt", "input3.txt")

(4) Program execution(WS operation - using telnet program on Macintosh)

Execute command "exeall".

This command makes result file "out".

(5) Data transfer

Execute "Fetch", and get the text file "out" from WS.

(6) Convert the output file

Open "tx2ex", modify the output file name(assume "result1"), and execute it.

The Excel file "result1" is the final result of cache performance and area.

(7) Arrange data sheet and make graphs.

4. Results

4.1 Tool performance

Table2 shows processing time and required memory size for this tool.

Table 2 Processing time and memory size

No	Process	Machine	Processing time*	Memory
1	Excel data to text file	Macintosh IIsi	2min. 57sec.	
2	Cache calculation	DEC5000	3min. 3sec.	364KB
3	Text file to Excel data	Macintosh IIsi	15min.10sec.	

* : Total processing time for evaluating 72 cache configurations

4.2 Calculation result examples and analysis

Using this tool, a designer can try to estimate performance and area for various kinds of cache data.

(1) Example 1

Fig.3 is the performance results for 32KB split cache. Ex-CPI shows the excessive CPI due to the cache. Fig.4 shows the area corresponding to Fig.3. It is categorized by write policy and scheme. Other parameters are as follows:

I-cache 32KB, 4way set associative,

D-cache 32KB, fully associative,

Ta=96ns, Tbus=8ns, Tc=16ns

L/S architecture, chip technology=0.6um

The following observations can be made on Fig.3 and Fig.4.

i) Ex-CPI decreases when the line size increases. Generally, cache miss rate decreases and miss penalty increases when the line size increases.

ii) At the same line size, Ex-CPI is smaller for lower latency cache memory interface schemes, which is more effective than the difference between 'Write through' and 'Copy back'.

iii) Areas required is smaller when the line is larger. For larger line size, the area is reduced because tag area is less. But when the line size becomes quite large, the overhead area for drivers and amplifiers is significant.

iv) Since the area for write buffer is not accounted for in the area calculation, the areas for different cache memory interface scheme are identical.

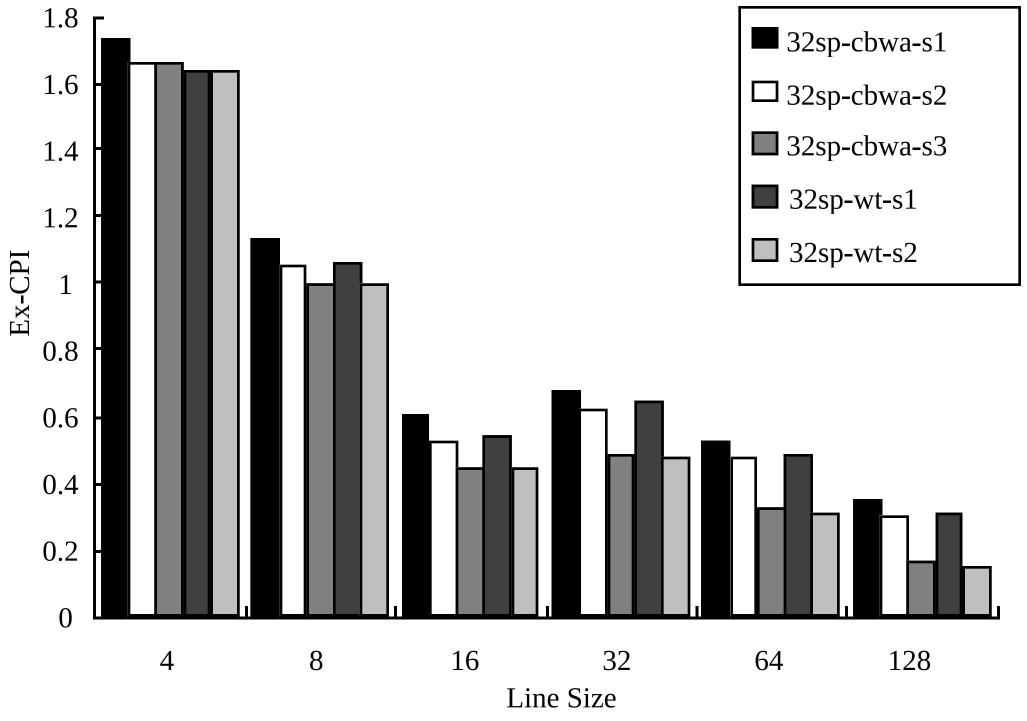

Fig.3 32kB Split cache Performance

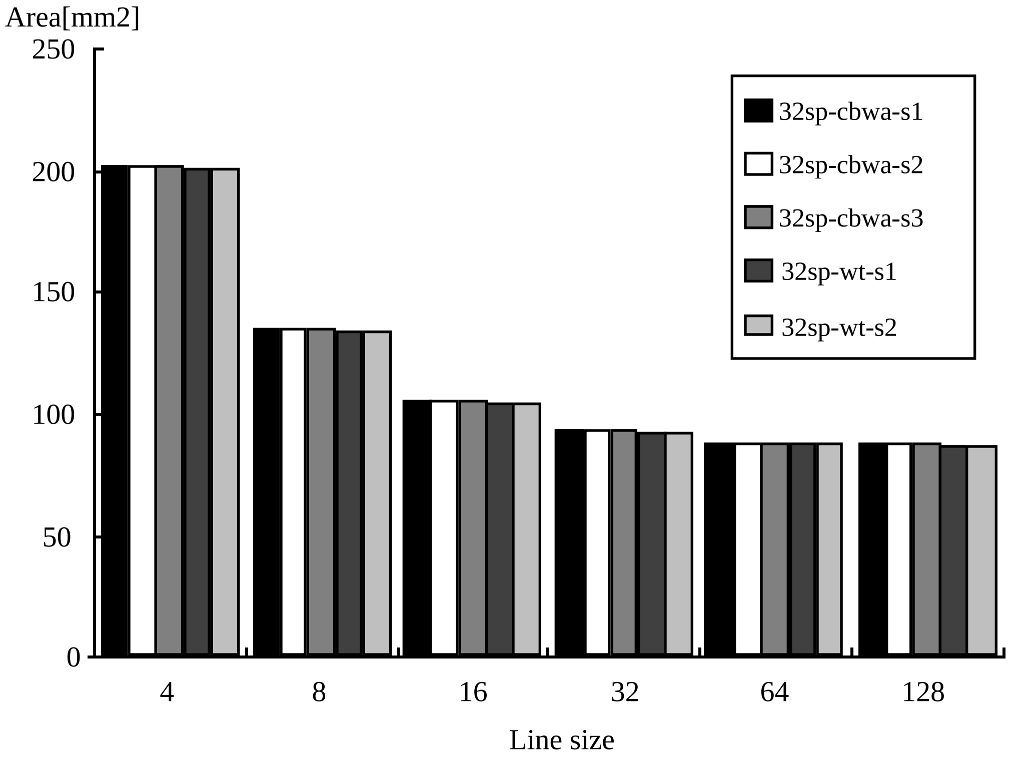

Fig.4 32kB Split cache Area

(2) Example 2

Fig.5 and Fig.6 are unified cache performance and area data. This data are categorized by cache size. associativity and line size.

Write scheme = 2

Ta=96ns, Tbus=8ns, Tc=16ns

L/S architecture, chip technology=0.6um

The following observations can be made on Fig.5 and Fig.6.

i) Ex-CPI is smaller when the associativity is larger. The Ex-CPI's for associativity 2 and 4 are almost the same.

ii) The area of a fully associative cache is larger than others, while the others are almost the same regardless of their associativity.

Fig.7 shows the relationship between performance and area for this example. Among these plots, a designer can select the best cache memory configuration. For example, if the area constraint is under $50mm^2$, best configurations shown in Table 3 can be selected. It is seen that cache #1 with 0.716 Ex-CPI and area of $42.3mm^2$ is the best in this study.

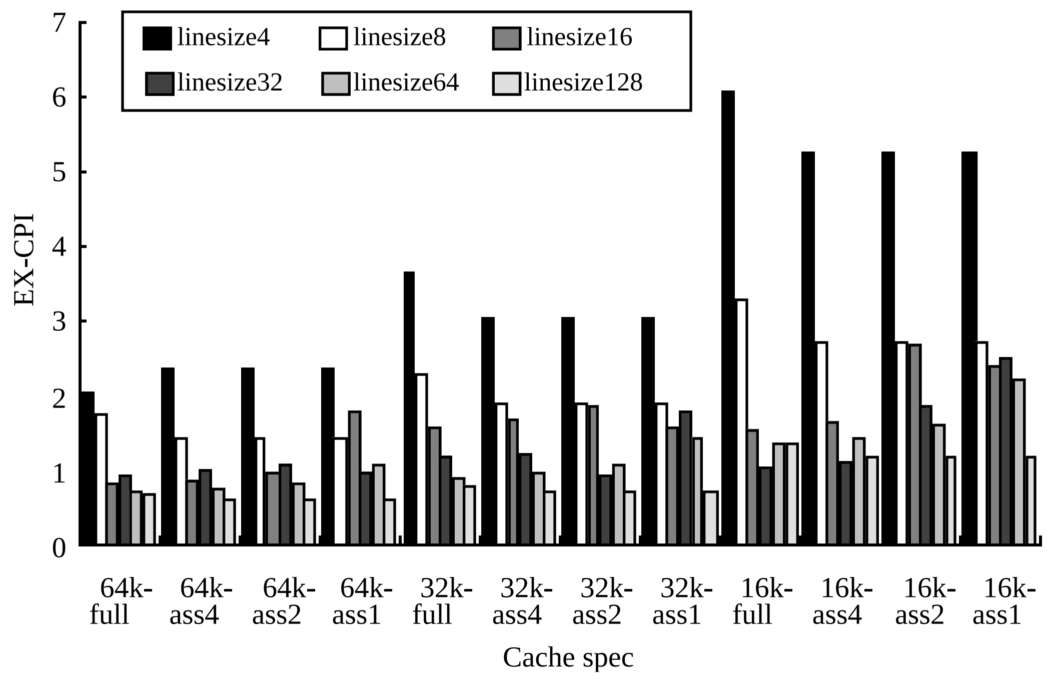

Fig.5 Unified cache Performance

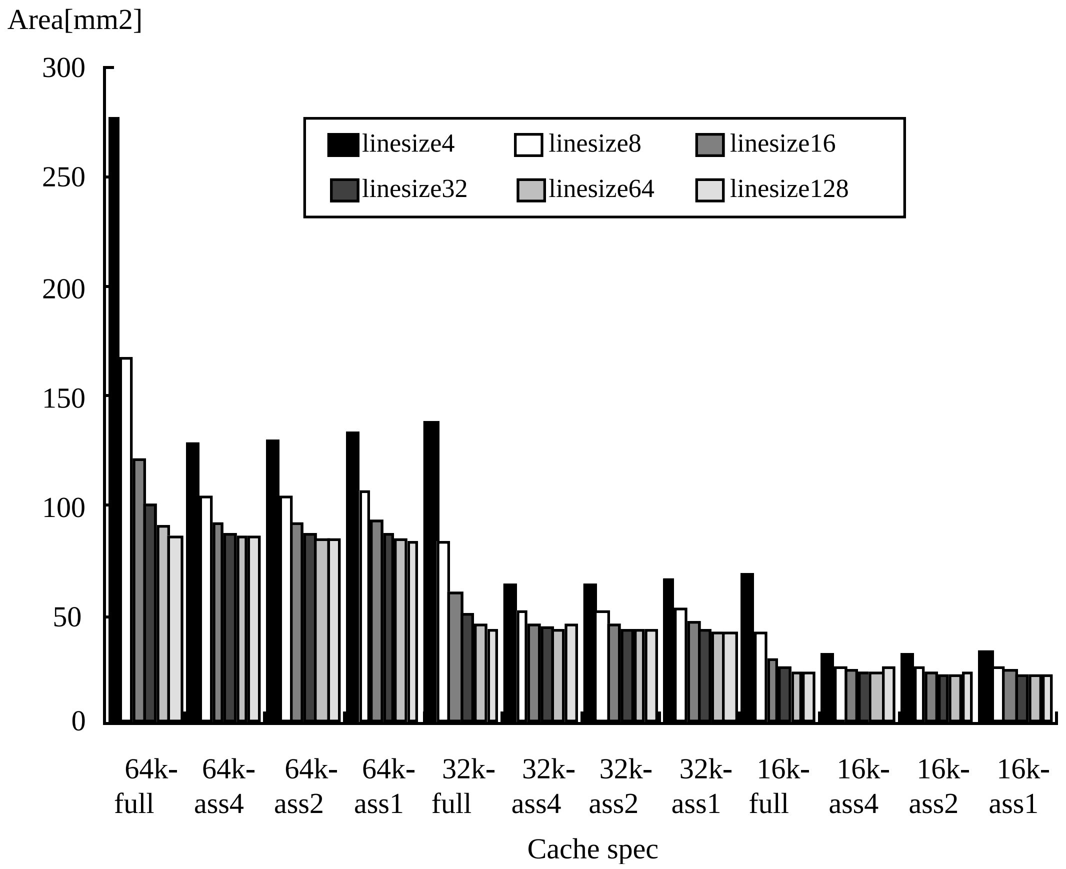

Fig.6 Unified cache Area

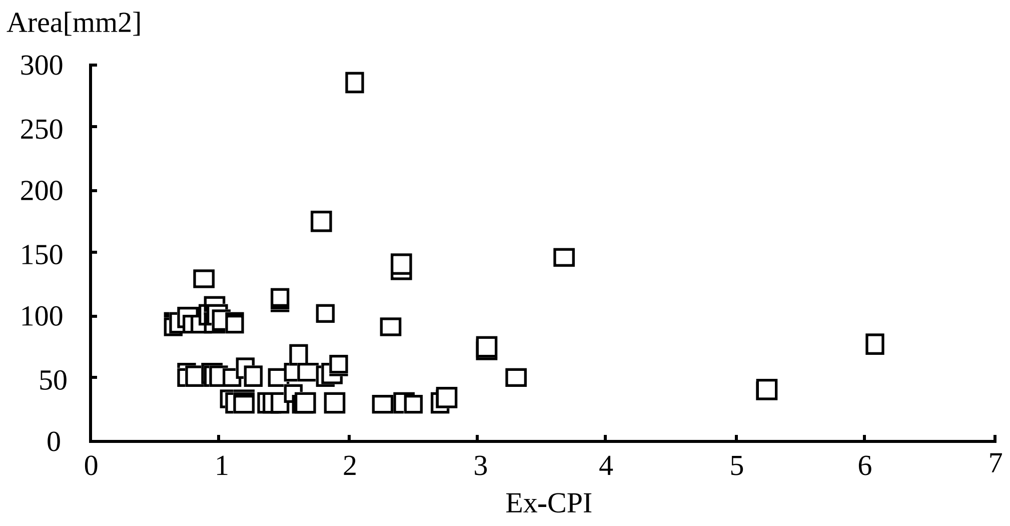

Fig.7 Unified cache Performance/Area Plots

Table 3 Best 5 Unified CBWA cache configuration under 50mm 2

No	Size(KB)	Line size(B)	associa-tivity	Ex-CPI	Area (mm2)
1	32	128	4	0.716	42.3
2	32	128	2	0.716	43.2
3	32	128	1	0.716	45.3
4	32	128	full	0.785	43.8
5	32	64	full	0.917	45.6

5. Conclusions

A tool for calculating cache memory speed and area has been developed. As the gap between processor speed and memory speed is expanding, it is very important for designers to consider tradeoffs for cost and performance of cache memories. The tool shown in this paper is useful for this tradeoff. Data examples calculated using this tool are shown. As the data are output to a spread sheet, the designer can analyze or manage them easily. For future work, more accurate performance prediction will be possible by combining this tool with an architectural simulator and cache simulator. In addition to performance and cost estimation, it is important to estimate the complete design including other factors such as power consumption in cache memory design.

Acknowledgment

The authors would like to thank Steve Fu for offering the miss rate calculation program and valuable discussion about modeling.

References

[1] A.J.Smith, "Cache Memories", ACM Computing Surveys, Vol.14, No.3, 1982, pp.473-530

[2] J.D.Gee et al., "Cache Performance of the SPEC92 Benchmark Suite", IEEE Micro, August 1993, pp.17-27

[3] D.B.Alpert and M.J.Flynn, "Performance Trade-offs for Microprocessor Cache Memories", IEEE Micro, August 1988, pp.44-54

[4] J.M.Mulder et al., "An Area Model for On-Chip Memories and its Application", IEEE Journal of Solid-State Circuits, vol.26 No.2,1991, pp.98-106

[5] N.P.Jouppi, "Tradeoffs in Two-Level On-Chip Caching", Proceedings of IEEE International Symposium on Computer Architecture, 1994, pp.34-45

[6] D. Nagle et al., "Optimal Allocation of On-chip Memory for Multiple-API Operating Systems", Proceedings of IEEE International Symposium on Computer Architecture, 1994, pp.358-369

[7] M.J.Flynn, "Computer Architecture: Concurrent and Parallel Processor Design", Jones and Bartlett, Boston, 1994

[8] S.Fu and M.J.Flynn, "Area and Performance Analysis of Processor Configurations with Scaling of Technology", Technical Report CSL-TR-94-605, Stanford University, 1994

[9] J.L.Hennessy and D.A.Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, San Mateo, 1990