Appendix

data, and in order to analyzes the shared cache performance, we have to use a different simulator, dinero, the difference of the simulation technology used in these two simulator may have introduced some error factor in our data.

8. Conclusion

Based on our study, with all the limitations in mind, we have observed that in a distributed cache environment, with 33 Million transistors, the overall system performance is led by 7 MPC620s, then followed by 19 MPC604s, and the MPC601 is clearly at the tail. We also have seen that the success of the MPC620 is not likely due to its cache performance, since all three processors' cache performance in the distributed cache environment are very close, and low. Meanwhile, the relatively much lower cache miss rate of the MPC604 in shared cache environment suggests that the overall system performance for MPC604s is most likely to be much better than the rest of the processors. Therefor, we conclude that for 33 Million transistors in a chip, the optimal configuration for cache and processors is to use the 7 most powerful microprocessor(MPC620) with a single shared cache of size 1Mbytes.

References

[1] Brewer, E.A., and Dellarocas, C.N.,. PROTEUS: A High-Performance Parallel-Architecture Simulator. *Technical Report MIT/LCS/TR-516,* 1991.

[2] Choi, C.S., DRAM transistor density versus time, Interview, Sam-sung, Nov. 1994.

[3] DiMarco, J., Spec Table, v. 3.29, University of Toronto, Nov. 1994.

[4] Nayfeh, B.A., and Olukotun, K., "Exploring the Design Space for a Shared-Cache Multiprocessor", Proc. of IEEE International Conference, 1994, pp. 166-174.

[5] Frohman, D., "VLSI technology: outlook for the year 2000 and its impact on systems", Proc. of the 1990 IEEE International Conference on Computer Systems and Software Engineering, Tel-Aviv, Israel, May 1990.

[6] Jaswinder, P.S., Truman, J., Anoop, G., and Hennessey, J., "An Empirical Comparison of the Kendall Square Research KSR-1 and Standford DASH Multiprocessors", Proc. of Supercomputing '93 Conference, Portland, Oregon, USA, Nov. 1993, pp. 214-225.

[7] Young, I., Bell, R.K., and Hennessey, J.L., "Microprocessors in the Year 2000", Proc. of IEEE International Solid-State Circuits Conference, San Francisco, Ca, USA, Feb. 1993, pp. 202-203.

6.3 Comparisons

Since the cache performance for both distributed case and shared case are quite consistent with respect to the type of processors in the system. We have plotted the cases for MPC601 to show the performance difference between shared cache and distributed cache. “Figure 6.3.1 Cache Performance: Shared vs Distributed” on page 11 shows the plot. It is clear shared cache has a much better performance than the distributed caches.

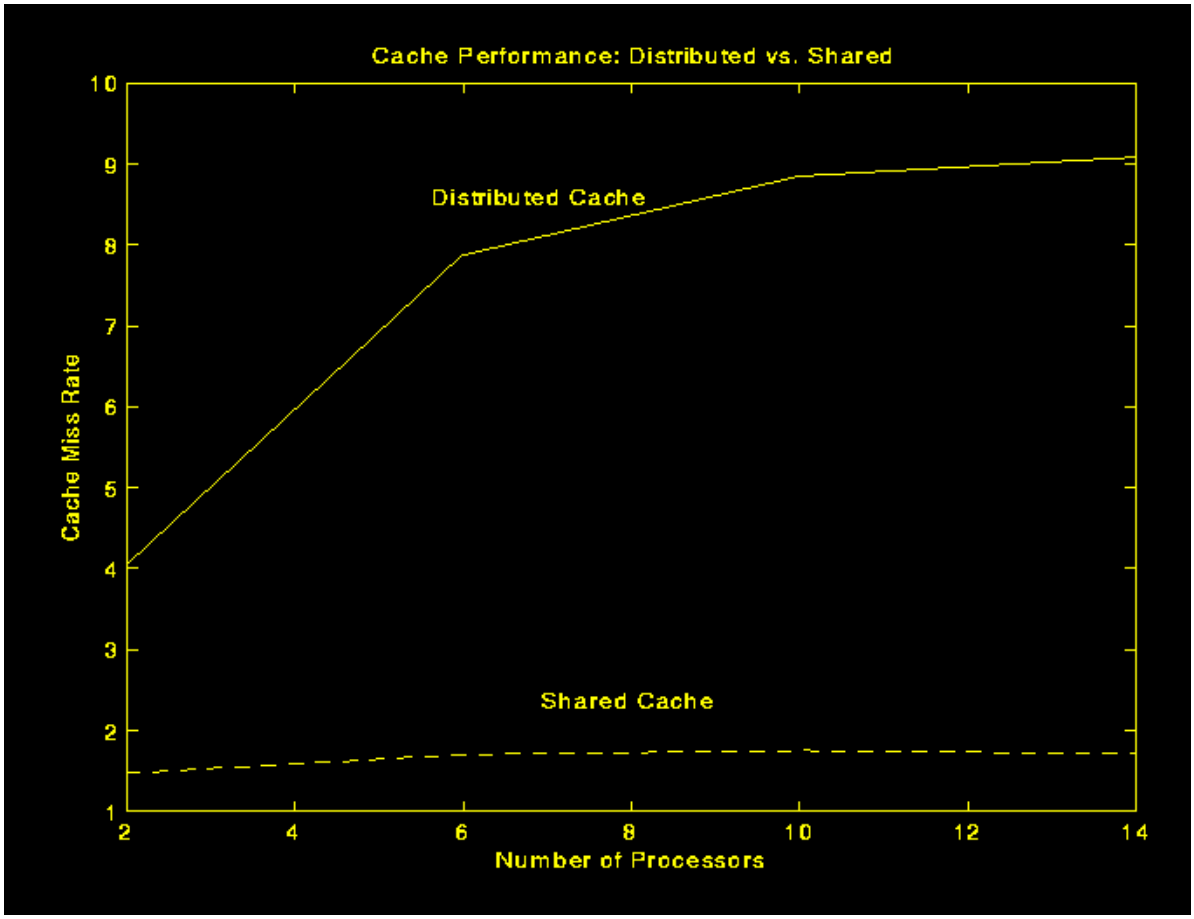

Figure 6.3.1 Cache Performance: Shared vs. Distributed

7. Assumptions And Limitations

In our study, due to the time and resource constrain, we have to make many assumptions in order to do the analysis. More specifically, we have assumed in the case of distributed caches, the miss rates are mainly come from shared data, since the Proteus only collects the cache performance statistics for the share data memory access.

One of the major limitation we have is the resource for parallel application programs for our parallel processor simulator. Meanwhile due to the nature of the simulator, we are only able to collect the distributed cache performance

6.2 The simulation Result for Shared Cache

Opposite to the cases in distributed cache, share cache does not have any extra miss rate due to the cache coherency transactions. Meanwhile, since we are running one single program across the parallel processors, the data sharing is expected to be high, and as a result, the overall cache miss rate is expected to be low. “Figure 6.2.1 Shared Cache Permance” on page 10 shows the detail.

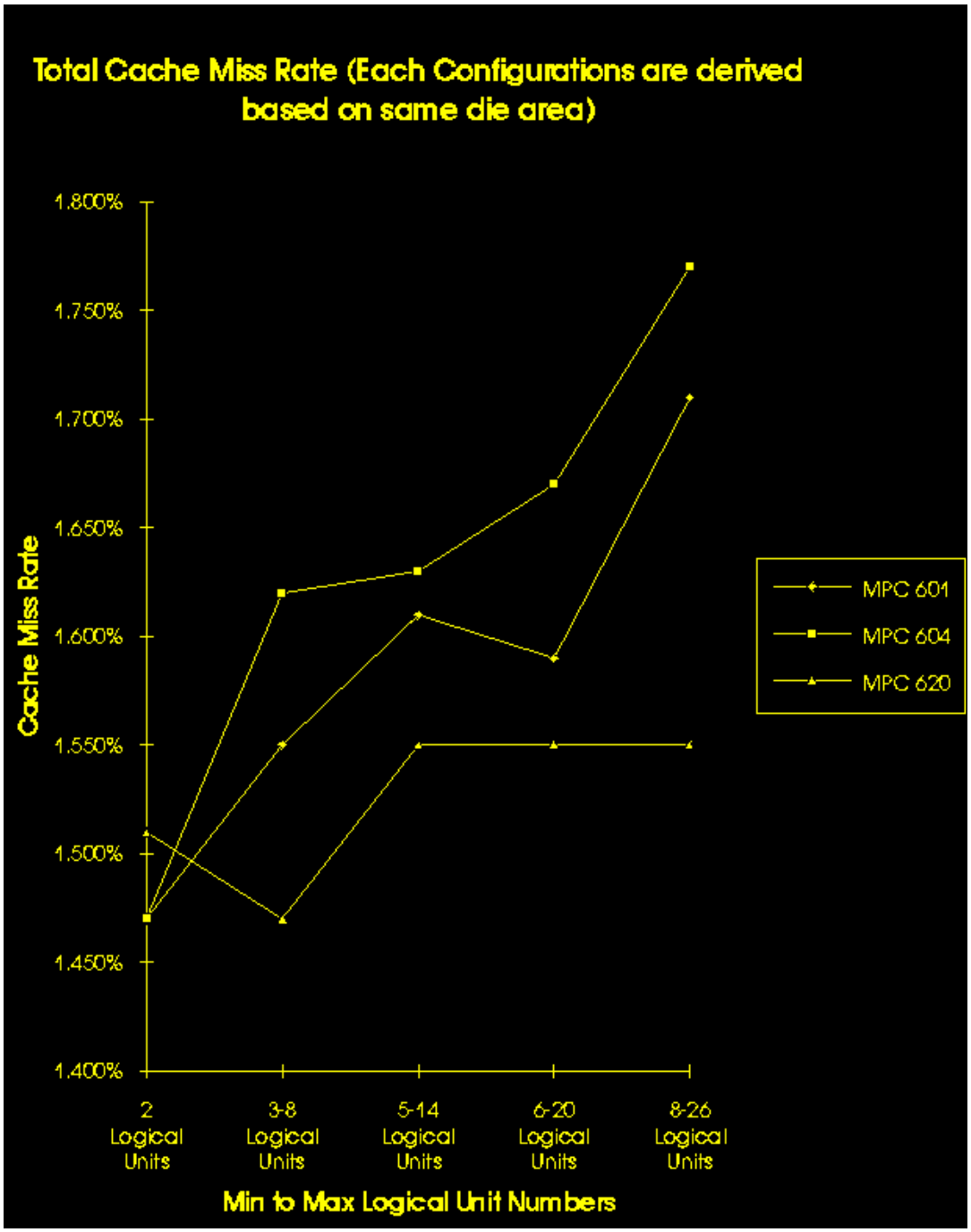

Figure 6.2.1 Shared Cache Performance

6.1 The Simulation Result for Distributed Caches

In our study we simulated one application program on the parallel processor simulator Proteus. Since the distributed caches have extra misses due to the cache coherency transactions, we expect the cache misses would be quite high. The simulation result seems to support our expectation. “Figure 6.1.1: Distributed Cache Performance” on page 10 shows the detail.

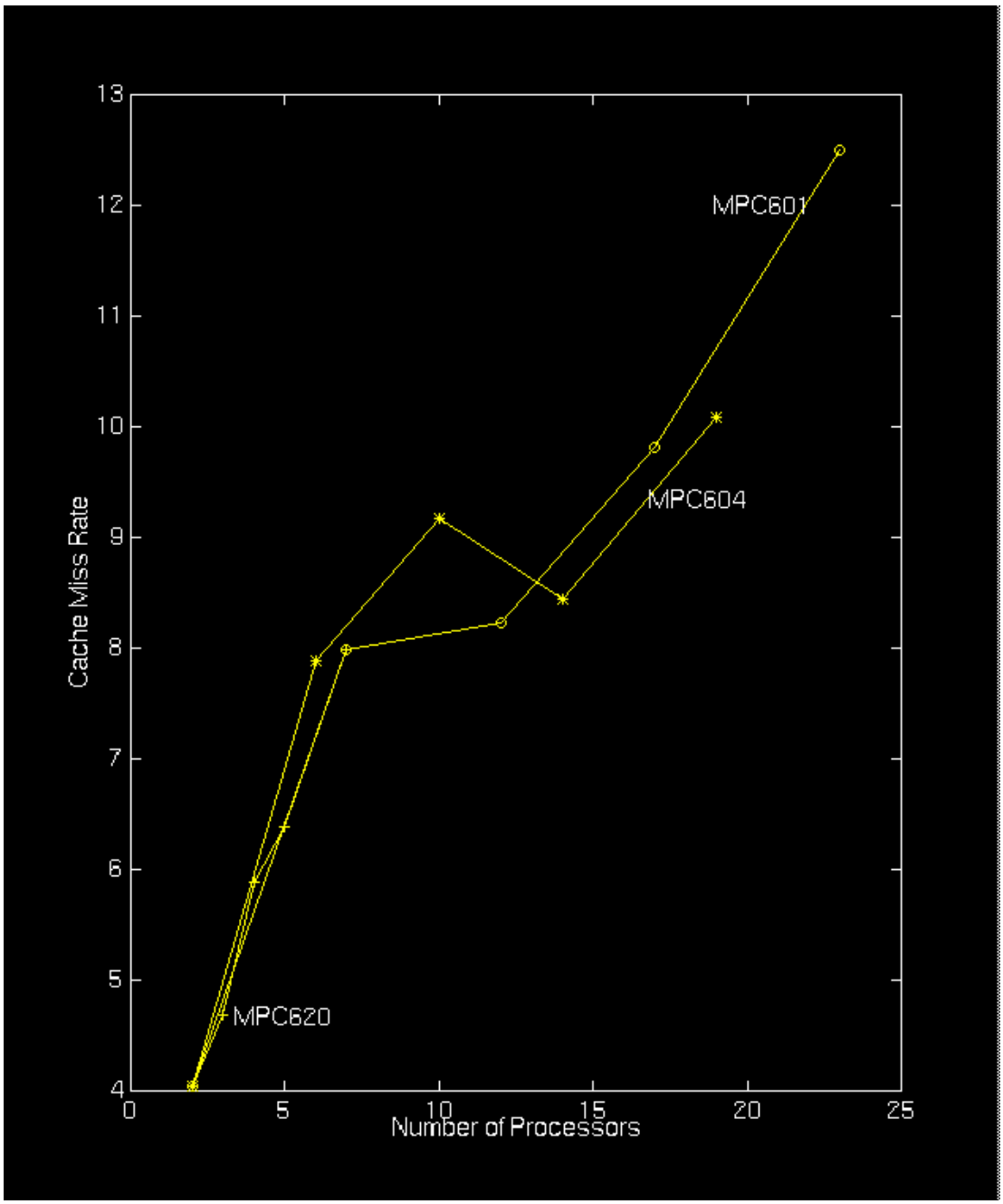

Figure 6.1.1 Distributed Cache Performance

The actual relation between total data cache size and the number of processors for MPC601, MPC604, MPC620 are clearly show in “Figure 6.0.1: Cache Size vs. Number of Processors” on page 8.

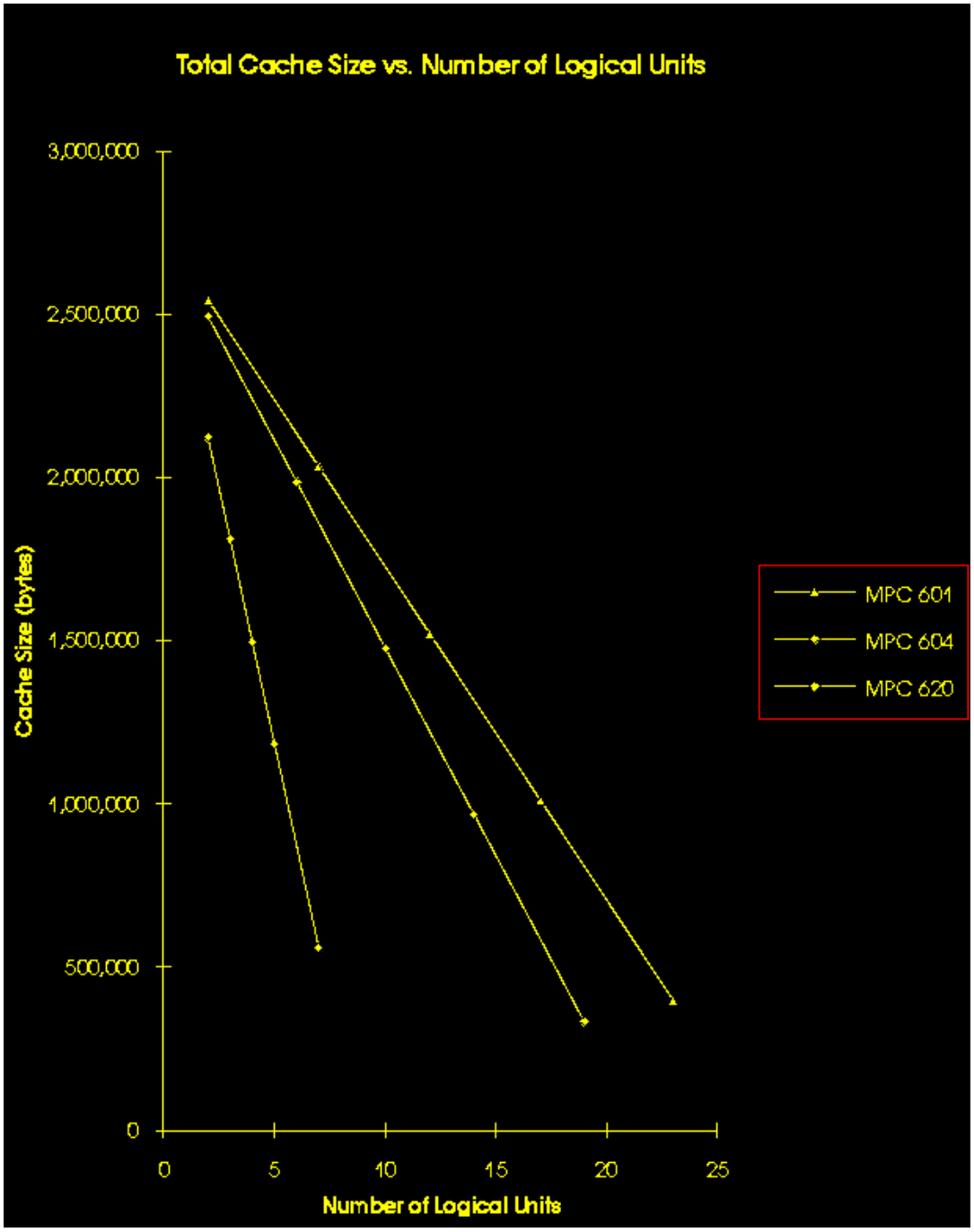

Figure 6.0.1: Cache Size vs. Number of Processors

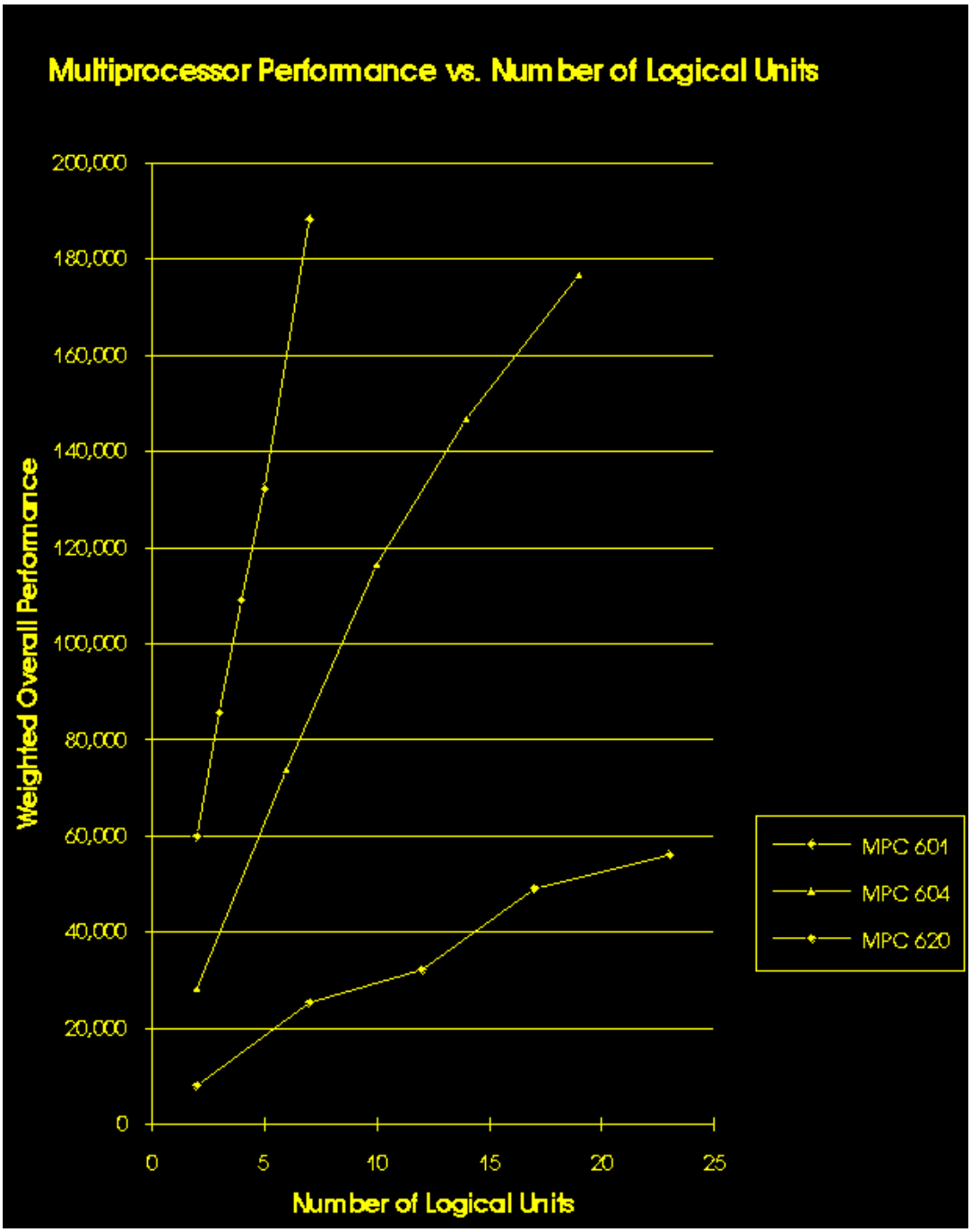

Figure 5.4.1: Processor Performance Summary

6. Cache Simulation Results

In our study we investigated the cache performance of two configurations: single shared cache and a number of distributed caches. As mentioned in previous sections, we collected the simulation result for the distributed case from our parallel processor simulator-Proteus, and collected the performance result of shared cache from the cache simulator-dineroIII.

5.3 Powerful Processors (MPC620) Simulation Results

Table 7 lists the simulation results we have derived for the different configurations for MPC620.

Table 7: MPC620 Simulation Result

Processors Count	Total Clock Cycle	Weighted Performance
2	570,480	59,916
3	398,584	85,756
4	313,356	109,080
5	258,321	132,320
7	181,462	188,365
Benchmark = 257	Clock Rate = 133MHZ	Cycle Time = 8 ns

5.4 Comparisons

We combined the results of all processors, and derived the following plot, “Figure 5.4.1: Processor Performace Summary” on page 7.

From the graph, we can see that in a distributed cache environment, as the number of processors increases, the processors’ performance increases as well for all three types of processors. And MPC620 has the highest peak among all processors when 7 MPC620 are in parallel. An interesting observation is that the MPC620 has the deepest slope which means even with more transistors (more than 33Million), the MPC620 is very likely to have the highest performance as well.

5.1 Weak Processors (MPC601) Simulation Results

Table 5 lists the simulation results we have derived for the different configurations of MPC601.

Table 5: MPC601 Simulation Result

Processors Count	Total Clock Cycle	Weighted Performance
2	570,480	8,063
7	181,462	25,350
12	143,108	32,144
17	93,697	49,094
23	81984	56,109
Benchmark = 46	Clock Rate = 100MHZ	Cycle Time = 10 ns

5.2 Medium Processors (MPC604) Simulation Results

Table 6 lists the simulation results we have derived for the different configurations for MPC604.

Table 6: MPC604 Simulation Result

Processors Count	Total Clock Cycle	Weighted Performance
2	570,480	28,397
6	218,909	74,003
10	138,901	116,630
14	110,270	146,912
19	91,620	176,817
Benchmark = 162	Clock Rate = 100MHZ	Cycle Time = 10 ns

direct mapped cache will have relatively small hit time and low miss rate as well. Table 4 gives the detailed information on our simulation parameters.

Table 4: Parameters used in Simulations

Cache Associativity	Direct Mapped
Cache Access Time	1 cycle
Cache Block Size	32 bytes
System Connection Topology	Bus Based
System Memory Access Time	4 cycles
System Reschedule Latency	4 cycles
System Interrupt Latency	2 cycles

Our goal is to investigate both the effect of varying the number of processors, and the strength of the processors. So in order to incorporate the strength of each processors into our analysis, we used the benchmark of each processor as the weighted factor, combined with the cycle count collected by the Proteus simulator, we are able to calculate the weighted performance for each processor configuration. The exact formula is:

$$WeightedPerformance = \frac{Benchmark}{ExecutionTime}$$
$$= \frac{10^9 \times Benchmark}{TotalCycles \times CycleTime}$$

5. Multiple Processors Simulation Results

In order to find out which processor configuration would give the best performance, we have simulated all possible cache/processor configurations for each type of processors: MPC601, MPC604, MPC620.

Since the designer of the Proteus simulator decided to ignore instruction cache performance in their simulator architecture, we are only able to get the data cache performance statistics for distributed caches from the simulator. As indicated before, after modifying the simulator source code, we were able to generate memory traces for data references. We used these traces as the input to our cache simulator dinero, to get the cache performance in a single shared cache environment.

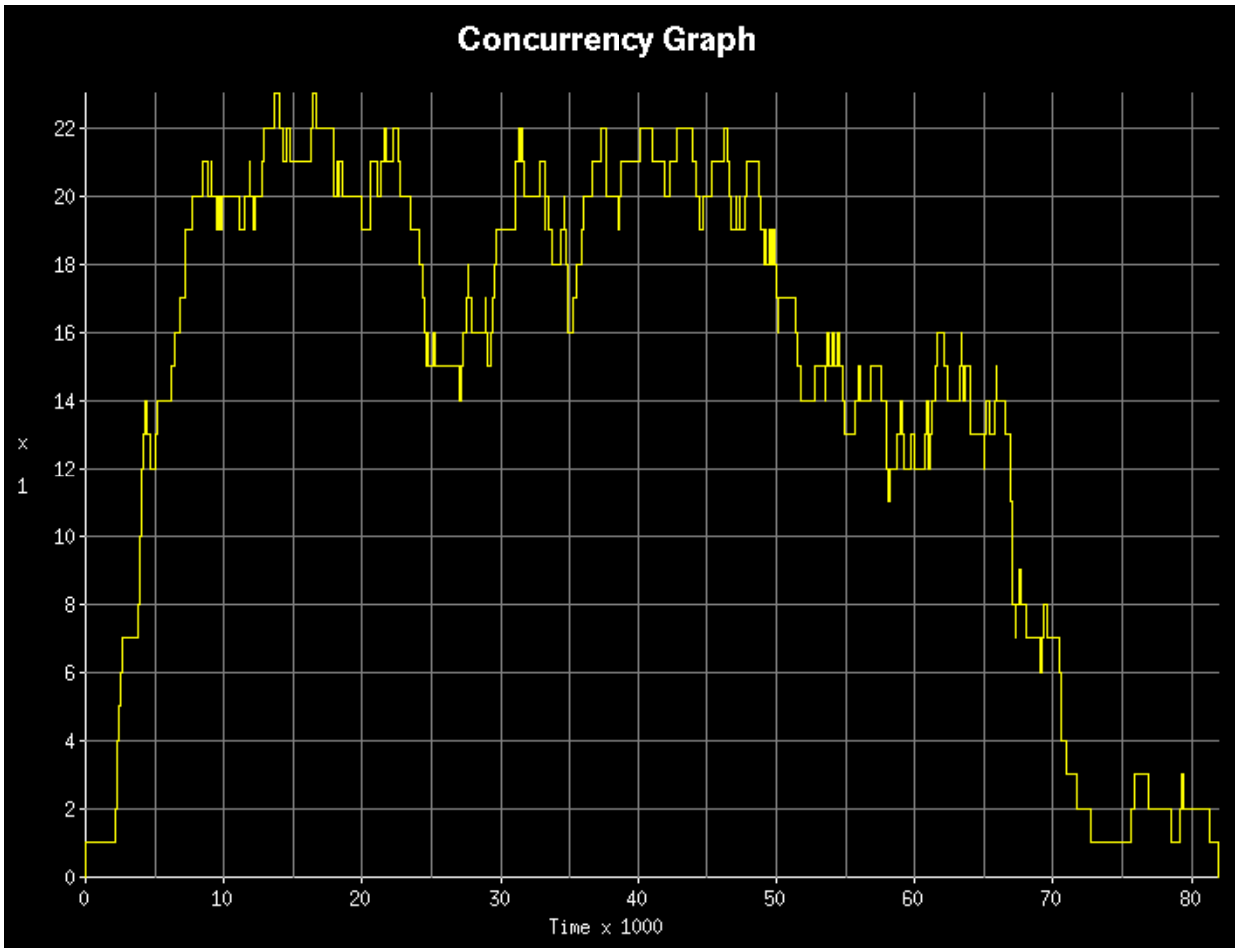

Figure 4.3.1

4.4. The Simulations

Time and resource constrains prevents us from achieving the ideal goal of exhaustively simulating many benchmark application programs. However, we have simulated all possible cache/processor configurations with our application program, as listed in Table 1 Table 2 and Table 3 . Since the number processors is always less than 25, which is technically called small scale MIMD system, we have decided to restrict the connection topology to be bus based. Meanwhile the caches in all configurations are overall very large, we believe a

Table 3: Power PC 620 Values

Number of CPU	2	3	4	5	7
Shared Cache Size	4,248,576	3,622,864	2,997,152	2,371,440	1,120016
Distributed Cache Size	2,124,288	1,207,621	749,288	474,288	160,002
Data Cache(shared)	2,124,288	1,811,432	1,498,576	1,185,720	560,008
Data Cache(Distr)	1,062,144	603,810	474,644	237,144	80,001

Since we would like to investigate both processor performance and cache performance with these different configuration, we need to collect both type of data during the simulation. Fortunately, our parallel processor simulator allows us to collect data for both processor and cache in distributed cache environment. In order to study the cache performance of a system in which all processors share a single large cache, I managed to make our parallel processor simulator generate the memory reference traces, and then used these traces as the input to another cache simulator, dinero. Thus we are able to study both cache configurations: shared vs. distributed. However, we are not able to directly collect the processor performance in a shared cache environment. Yet, with the information derived from processor performance in a distributed cache environment, and cache performance in both shared and distributed environment, we can make reasonably accurate decision in determining the processor performance in shared cache performance.

4.2 The Simulator

The system simulator used in this study is a high-performance parallel-architecture simulator, called Proteus. It simulates MIMD processors with high accuracy, and allows user to specify each cache size, associativity and access time, as well as the number of processors in a configuration, the number of memory modules and sizes, and connection topology(1).

The cache simulator used in our study is dineroIII. It allows the user to specify many parameters of a cache, including cache size, associativity, block size, write policy etc.

4.3 The Simulation Application Program and Traces

Due to limited time and resource, we were only able to make one parallel application program working on the simulator. The program solves a 8-queen problem on a system with various number of processors. The “Figure 4.3.1” on page 3 reveals the parallelism within the program on 23-processor system.

4. The Simulation Environment

Our goal for this project is to evaluate the performance of different cache/ processor configurations, so to help designers make transistor allocation decisions. In order to quantify the effectiveness of varying cache sizes, number of processors, and the strength of processors, we have designed our simulation model, collected a powerful simulator, and targeted a testing program to evaluate the different configurations. These are presented below.

4.1. The Simulation Model

The simulation model used in our study assumes multiple processors with their own individual unified caches, and are connected through a bus, and share a large memory. The processors we are interested in are MPC601, MPC604, MPC620. Since the total number of transistors available in our study is 33 Million, we have calculated most of different configurations of caches and processors based on the current technologies used in those MPC mentioned above. The Table 1 , Table 2 , and Table 3 gives the detail in describing all these configurations.

Table 1: Power PC 601 Values

Number of CPU	2	7	12	17	23
Shared Cache Size	5,090,954	4,068,341	3,045,728	2,023,114	795,978
Distributed Cache Size	2,545,477	581,192	253,811	119,007	34,608
Data Cache(shared)	2,545,477	2,034,170	1,522,864	1,011,557	397,989
Data Cache(Distr)	1,272,738	290,595	126,905	59,503	17,303

Table 2: Power PC 604 Values

Number of CPU	2	6	10	14	19
Shared Cache Size	4,990,954	3,973,864	2,954,773	1,936,682	664,069
Distributed Cache Size	2,495,477	662,144	295,477	138,334	34,951
Data Cache(shared)	2,495,477	1,986,432	1,477,386	968,341	332,034
Data Cache(Distr)	1,247,738	331,072	147,738	69,167	17,475

Using 33 million transistors we devised fifteen different combinations of multiprocessor models for our simulations. For convenience, we made a worksheet to give us exact sizes of the caches and number of logical units. This worksheet updated itself when new die size, no. of transistors for three different logical units, and minimum cache sizes were entered. But encorporating minimum cache size involved following sets of algebra:

Minimum Transistors for Cache per LU[MTC]
= Minimum Cache Size per LU(KBytes) x 6(transistors per bit) x 8(bits/Byte) x 1024(1/K)

Transistors useable for LU[TUL] = Total Transistor[TT] - (Number of LU x MTC)

Number of LU = TUL/Transistor per LU[TLU]

From above equations Number of LUs are calculated:

$$NumberLU = \frac{TT}{TLU + MTC}$$

Since Number of LU has to be whole number it was truncated to give exact number of possible logical units. Then rest of the space was used for cache:

3.2. Logical Units/Cache Configuration

Since internal caches are Static RAM(SRAM), we assumed average transistors for one bit in cache would be siz transistors. Using this assumption number of transistors used for cache became easy to calculate:

$$CacheTrans = CacheSize\,(KByte) \times 8\left(\frac{bits}{Byte}\right) \times 6\left(\frac{Trans}{bit}\right) \times 1024\left(\frac{1}{K}\right)$$

From total number of transistor, number of transistors for caches was subtracted to get an approximate number of transistor for logical units in each model CPUs. This data was verified when same method was applied to several CPUs which we were able to obtain transistor numbers of logical units. The standard deviation was approximately +/- 5% when calculated numbers were compared with several true LU transistor count.

To give optimum performance/cost for each models minimum cache size for distributed cache was set to each CPU's original first-level cache size. The detailed results for caches are later shown in section 4.1. table 1, 2, and 3.

2.4. Result

As a result of our calculations and research, our prediction concludes with following for year 2000:

A)Die Size: Ranging up to 690 mm^2 (With most die size=300mm^2)
B)Transistor Density: Approximately 110,000 transistors/mm^2
C)Clock Speed: Ranging from 250 MHz up to 2000MHz

From these results we saw that preliminary evaluation made by Frohman in his conference paper seemed to be somewhat coherent in number of transistors (50-100million transistors) but mispredicted the clock rate (200MHz).

3. Multiprocessor Models

The concept of single chip multiprocessor is unlike the multiprocessors that people in today's computer industries are familiar with. Since several Logical Units(LU) will be on same chip, communication between each processors will be very fast. In large scale multiprocessor environment, issue of having distribute memory module for each CPU would not be a question to consider because communication speed would not be much problem. But due to increase in its overall performance, cost of cache misses will be weighted more. And organization of cache and LU play a big role in overall performance of the processor.

To evaluate multiprocessor performance, we needed several different combinations of LU and Cache sizes. Overall performance could not be found due to limitations in simulators. Thus, we broke the multiprocessor components into LU performance and Cache performance.

3.1. Sample Models

For consistancy in generations of logical units, we used three CPUs of different power and size made by same manufacturer. Three CPU models were PowerPC 601(MPC 601), 604(MPC 604), and 620(MPC 620).

To set up a combinations of LUs and Caches, we needed total number of transistors we can use to place logical units and caches. Thus, we used our predictions from section 2 to construct a sample die.

$$TotalTransistors = DieSize \times TransistorDensity$$

We used our resulting die size (300mm^2) and Transistor Density (110,000 transistors/mm^2) to have 33 million transistors to construct our multiprocessor models.

To verify this prediction, we analyzed the changes in transistor density of DRAM. From this analysis we found that transistor density grew in constant rate. Using this average rate, we applied into our CPU transistor densisty change prediction. We found that the maximum density of CPU would be approximately 90,000 transistors/mm^2. Thus, transistor density was adjusted as a mean value of 130,000 and 90,000 = 110,000 transistors/mm^2.

2.3. Clock Speed

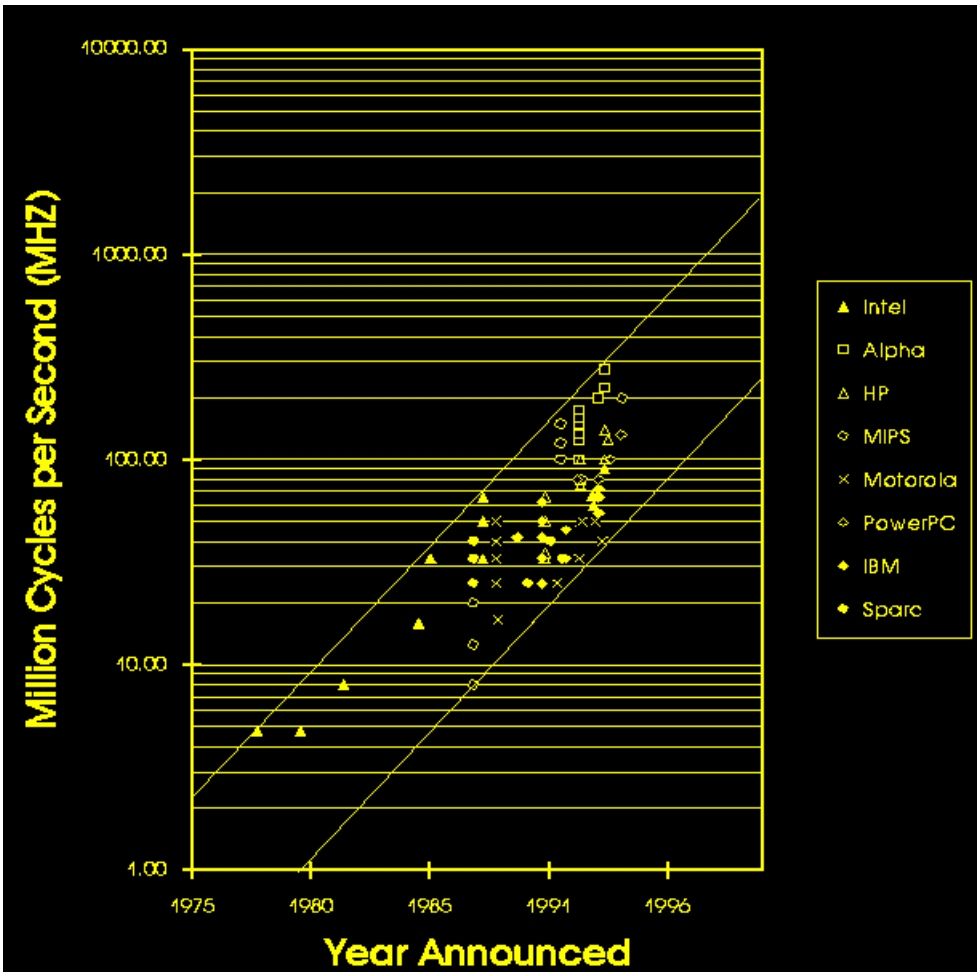

Figure 2.3.1: Clock Speed vs. Year Announced

Most easily available data for the CPUs were their clock speed. We collected over hundred CPU clock speed which gave us better chance in accurate prediction. These clock speeds were sorted in terms of time they were announced. Then the values were plotted in a log scale chart. From the graph, , we show linearity of increase in clock speed. By year 2000, we found that the clock speed will range from 250 MHz to 2000 MHz.

in its time) Thus, the average die size that we found for year 2000 was 343.5 mm^2. As it will be shown in section 3, for simplicity in our simulation, we decided to use 300 mm^2 for common die size in year 2000.

2.2. Transistor Density

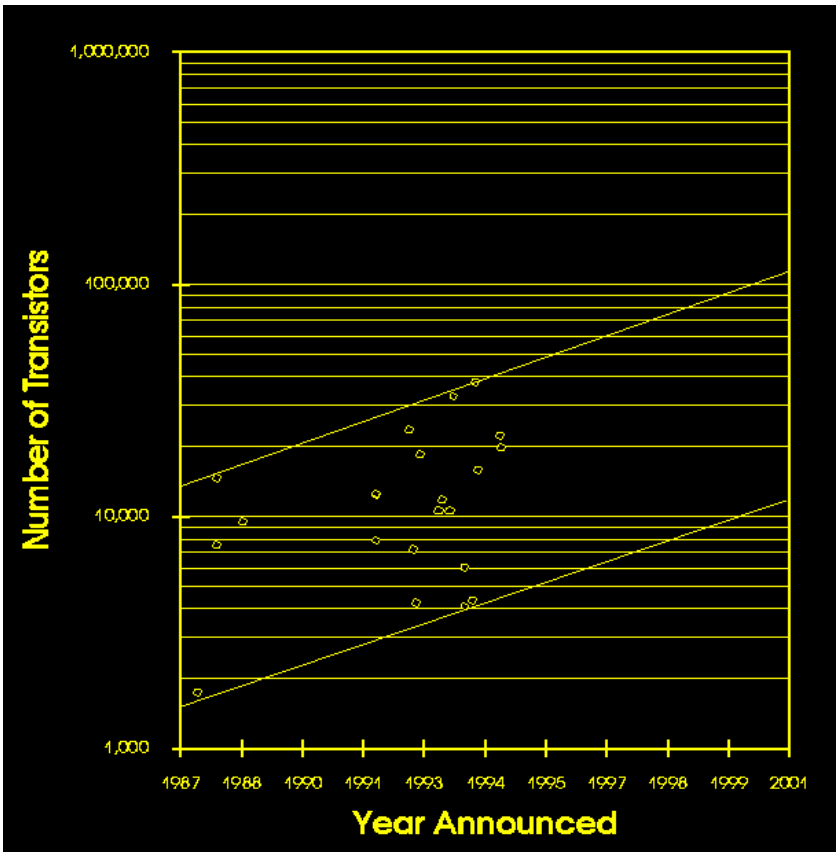

Figure 2.2.1: Transistor Density

This feature was seperately calculated to give more accurate measurement of average transistor per die. Data on number of transistors were collected. Then to determine the density each of these numbers were divided by its corresponding die size:

$$TransistorDensity = \frac{TransistorNumber}{DieSize\left(mm^2\right)}$$

With calculated transistor density we constructed a log scale scattered graph with density and time as its axis. From the figure 2.2.1 we extracted a line for average for every year. We found that this line passed through approximately 130,000 transistors/mm^2 at year 2000.

2.1. Die size in year 2000

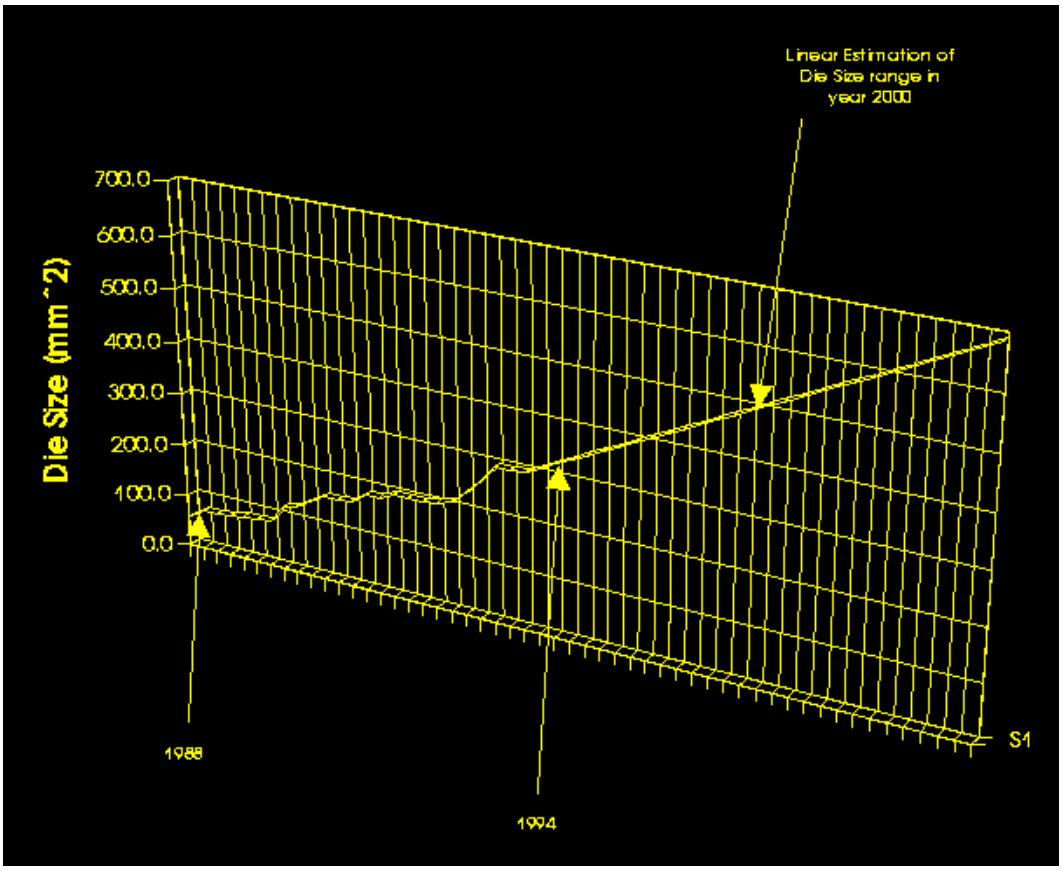

Figure 2.1.1: Die Size Prediction(Average Die Size:343.5mm^2)

To predict the die size, we collected die information on the CPUs. We found die sizes for 23 different CPUs manufactured during last six years. CPU with smallest die was MIPS R3000(72 mm^2) and the largest CPU was PowerPC 620(311.2 mm^2). Although die sizes varied between different manufacturer, we show constant growth in die size every year. Fortunately, MIPS R3000 was announced in 1988 while PowerPC 620 was announced in 1994. This data allowed us to sort the die size to show us linear growth in die size from year to year.

As shown in graph, relative rate of growth was determined. Then prediction line was drawn to give us the approximate die size in year 2000. Alluding to the fact that in 1994 the die sizes ranged from 74 mm^2 (This is due to higher transistor density every year) to 311.2 mm^2 we felt that we needed to determine average die size. Assuming that minimum die sizes in each year does not change drastically, we calculated the average die size by taking the mean for all 23 die sizes and 23 more die sizes for next six years.(This prediction is accurate because these die sizes were the result of best performance/cost for each CPUs

1. Introduction

Single chip multiprocessor is what many computer analysts expects to see in the near future. This idea is popular today because of the movement in computer architecture from large CISC machines to smaller RISC machines. Since smaller and simpler CPUs are faster, the development will be toward using several simple chips in parallel rather than developing a large single logical unit. The idea of single chip multiprocessor solved many problem concerning slow communications between CPUs since the logical units will be packed closely together.

The validity of this idea can be supported by the rate of changes in transistor density in last 15 years. Transistor density and speed is growing steadily every year, and by year 2000 we will have enough transistors on a single die to place more than one or two CPUs and its corresponding cache. Preliminary predictions were made by Frohman D.[5] in 1990 that in year 2000, transistor density per chip will reach 50-100 million with clock frequencies up to 250 MHz. The goal of our work is to re-evaluate the prediction. Then using our predicted die configurations simulate the performance of different combinations of logical units and cache to determine how effective this idea will be.

1.1 Overview

The remainder of this paper is organized as follows. In section 2, we re-evaluate the transistor density, clock speed, and average die size using datas collected on the CPUs manufactured in last 15 years. Then we carefully calculated the layout of the multiprocessor in single die in section 3; using three model logical units used today, PowerPC 601, 604, and 620. Then in section 4, 5, and 6, we simulated each configurations to find performance of logical units and cache. During the research we realized that predicting the computer technology was extremely difficult with limitations in our tools and time. Therefore we felt that in section 7, it was extremely important to mention all the limitations and our assumptions to aid and encourage others who plans to pursue this area. Finally, in section 8 we conclude with a summary of our observations and describe the future developments.

2. Prediction

Predictions in computer industry is very difficult today. It is one of the fastest growing technology. Since many preliminary predictions made were toward year 2000, we decided to re-evaluate the predictions using our own researched data. The predictions can be made on several features of the die; so we narrowed the features down to die size, transistor density and clock speed. Firstly, we collected much datas on more than hundred CPUs in last 15 years. Then calculations were made to predict the features in year 2000. Then to verify the results, accurate datas on transistor density of DRAM was encorporated into our evaluation.

Single Chip Multiprocessors

(Predictions on Die Features and Logical Unit/Cache Layout)

CS252 Final Project Report

Department of Electrical Engineering and Computer Science
University of California at Berkeley
Berkeley, California 94720

Wenchao Yang
wenchao@cory.eecs.berkeley.edu
SID#:1138-1807
Phone#:(510) 649-9804

Young H. Cho
youngc@cory.eecs.berkeley.edu
SID#:1152-8018
Phone#:(510) 540-0508

Abstract

Single chip multiprocessor is what many computer analysts expect computer research to point toward in the near future. Basic idea is that more than one CPU can be placed on single die; working in parallel like the multiprocessors available and are researched today. This idea is likely to become reality by year 2000 because of rapid growth in transistor density. Therefore we devised tests and simulations to assess and predict the validity and performance of single chip multiprocessor. First, the number of transistors and average die size in year 2000 were predicted. Then we calculated different combinations of model logical units and cache that will fit on a single die so we can simulate them. Lastly, conclusion was drawn from the simulations made with Proteus, multiprocessor simulator, and Dinero, cache simulator. Our early assessment is that powerful, but larger, logical units gave better performance with their corresponding caches. And for the cache configuration, shared cache gives dramatic performance lead against distributed cache.