Multi-Tool CAD Systems: A Performance Optimization Case Study

Jason C.Y. Lee, Daniel P. Siewiorek

EDRC 18-51-94

Multi-Tool CAD Systems: A Performance Optimization Case Study

Jason C.Y. Lee, Daniel P. Siewiorek

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

October, 1992

Abstract: As Computer-Aided Design (CAD) systems grow in size and complexity, the use of code or programs obtained from external sources is increasing. When these external tools become performance bottlenecks, traditional performance optimization methods can not be used. Instead, a black-box performance optimization approach must be used. Evaluation of the synthesis phase in the MICON system-level synthesis system revealed that over 90% of the time was spent in the commercial database. Black-box performance optimization uses a profiling method on the inputs to the database and a measurement of the synthesis time to identify areas on which to focus improvements. Modifications to the identified input areas produced over a factor of three speedup in total database processing time resulting in over a factor of two reduction in total MICON run time.

1 Introduction

Performance issues become more significant as software systems rapidly grow in size and complexity. The traditional approach to improving system performance involves four steps [1]. 1) Identify the most time consuming task or program; 2) profile the task/program to identify any bottlenecks; 3) modify the source code of the time consuming portion of the task to remove the bottleneck; and 4) re-evaluate the performance to determine if the improvement meets the initial goal. In an analogy to testing [2], performance optimization based upon source code profiling (using **tools such as *pixie* and *prof)* will be called *white box performance optimization* (WBPO). As** Computer-Aided Design (CAD) systems grow in size and complexity, programs written by different organizations are integrated. Such integrated systems provide an interesting challenge to performance optimization. The source code for some programs will not be accessible for use with WBPO. The only modifications possible for these programs is the alteration of the type and distribution of inputs. Performance tuning by input modification will be called black box performance optimization (BBPO). Frequently there are semantically equivalent requests to the black box program **(BBP).** The goal in BBPO is to identify the most efficient of the semantically equivalent input sets. Even small changes in type and distribution of inputs can have a dramatic impact on overall performance. Thus when BBPs are found to be the performance bottleneck, step three in the traditional approach to improving system performance must involve BBPO. The MICON system was examined using this modified approach.

2 Identifying the Most Time Consuming Task

The set of MICON tools allow for the capture of hardware design knowledge and the synthesis of new systems based upon this accumulated knowledge [3]. The MICON system is composed of a set of modules, as shown in Figure 1. The major modules are Ml - a knowledge-based hardware design synthesis tool, CGEN - an automated knowledge-acquisition tool, and ASSURE - a design for dependability advisor. Ml uses a hierarchical synthesis-by-composition approach. It actively supports automated knowledge acquisition by prompting the user when it needs additional knowl-

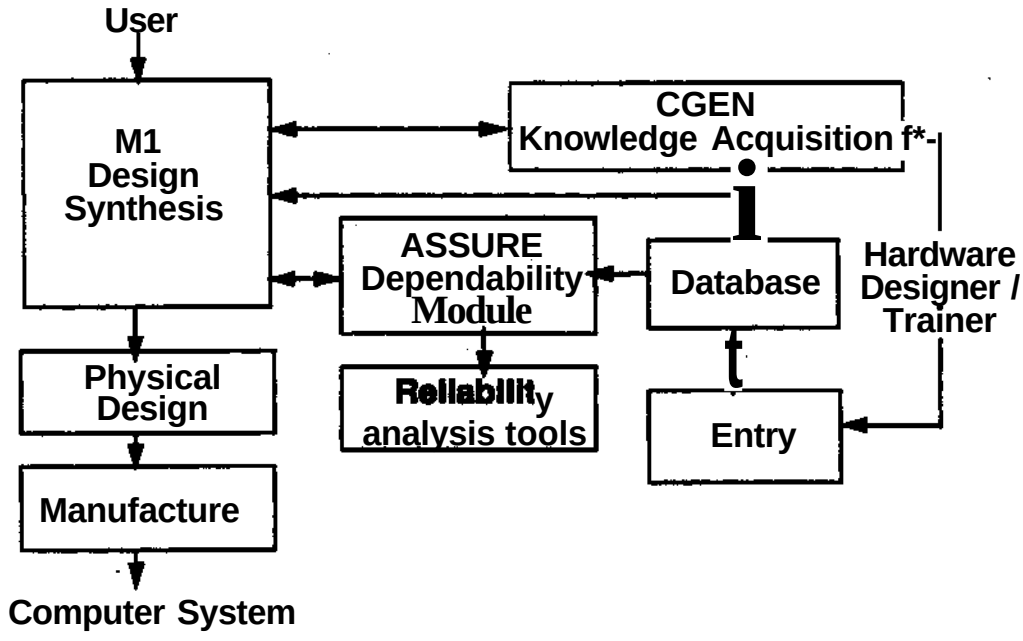


Figure 1 The MICON System

edge. CUEN takes schematics and simple equations tirom the designer and compiles them into Mi's knowledge base. ASSURE analyzes designs produced by Ml and suggests modifications for improved reliability. Figure 2 shows a sample Ml input and output. In the MICON system, synthesis is performed most often since design space exploration requires synthesizing a Iaige set of designs. For example, one application specific computer system had a design space consisting of over 5000 designs which was explored by MICON. Thus the synthesis task was easily identified as the most time consuming phase of the MICON system.

3 Profiling the Task

Four major areas were found to contribute to the execution time of the synthesis task. 1) The computation time of the synthesis tool Ml; 2) access time to the database for component information; 3) time required to build and buffer the requests to the database; and 4) network time to send and receive the data. To profile the execution time of the task. Ml was used to synthesize an application specific computer board based on the 80188 processor. This design involved 45 components

Query	User Input
Total amount of board area < square inches >	10000
Told amount of board cost < nearest dollar >	1000
Toui amount of power dissipation allowed < milliwatts >	50000
Is a CO_PROCESSOR needed	N
UPtOmadnt	Y
bSKnmlnt	Y
bTIMBRanded	N
Uafaywan Controller needed	N
Is Ethernet interface needed	N
Is DRAM needed	N
Is SRAM needed	Y
Is ROM needed	Y
Name of Processor Family	?
Processor name	68000
Minimum clock speed < Hz >	1000000
Number of ROM BYTES needed	2000
Name of ROM chip to use	?
Number of SRAM BYTES needed	40000
Name of SRAM chip to use	?
How many Ethernet Interface units are needed	0
How many PIO units are needed	?
What PIO chip would you like	?
Should this PIO generate interrupts < Y N >	N
Should this PIO preserve a resistable interrupt < Y N >	N
IMMMK Pito*y Uwd fcr MPIO	0
BnMraTAOiorMtnO	PIO.0
PIO pm^MOltaK IN OUT BIDtt >	BIDOI
ial connector type do you want	MALE
What connector size do you want	9
What drive type do you want < TTL etc >	NONB
How many SIO units are needed	1
Enternal InnMM» for dM pMt < Rs.232 «B >	RS-232
Bad m» tor tf* 3K) < 4MO m >	4800
SK) drip MM	7
ShouU Mt SK> M M i MBIMW < Y N >	N
SMMlfeMMMiMHMUMK YN>	N
Pri«'nykw«dfiork interrupts	0
Enter a tag for this SIO	SIOJD
External port size < 3 5 9 >	5
PMC«M<DTBDCB>	DTB
Cm MOM < SYNCHRONOUS ASYNCHRONOUS BOTH >	ASYNCHRONOUS
BMI MM < PIXBD HWJWITCH SW_SWITCH >	HWJW1TCH
Dniwil c—lar potofcy < MALB PBMALB >	PBMALB
External connector size < 9 25 >	25
How many TIMER units are needed 2	0
Do you want reliability analysis (y/n)	N

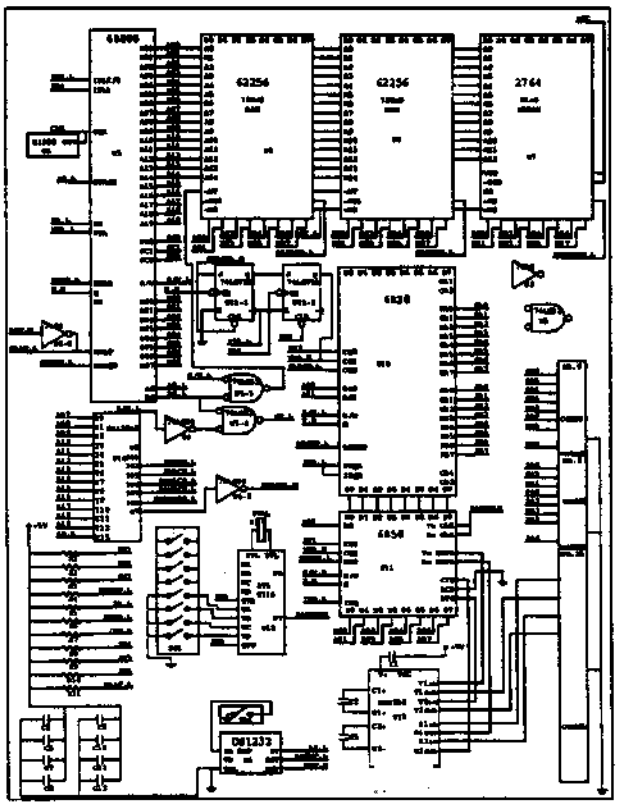


Figure 2 Sample Ml Input (top) and Schematic for Mi's Output Netlist (bottom)

and 80 nets. The database access time was found to dominate the execution time[1] (Figure 3). Of

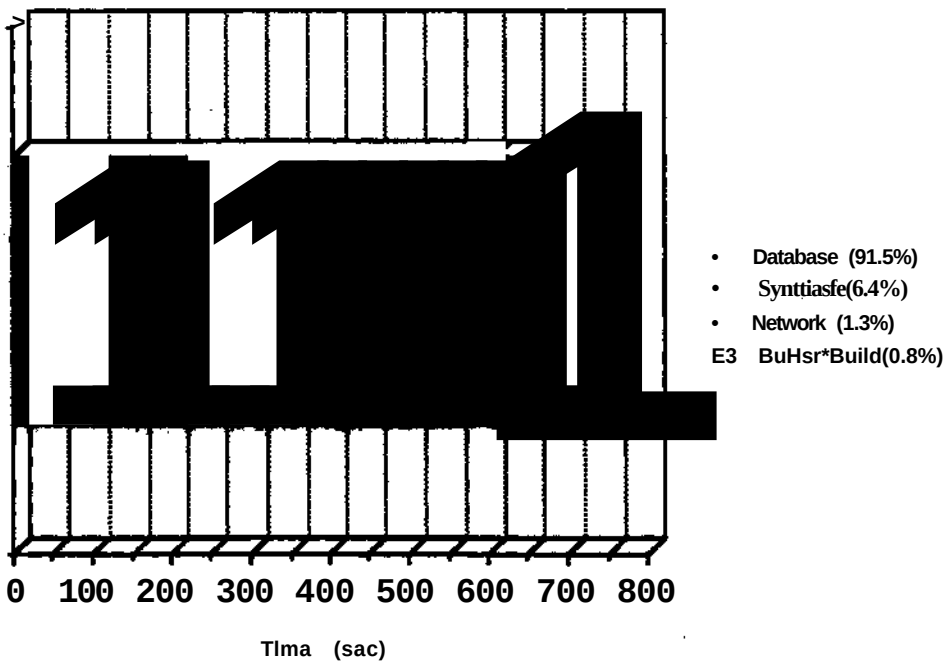


Figure 3 Ml Execution Time for the 80188-based Design

the total 665 seconds, 42.6 seconds were spent on synthesis, 608.5 seconds on accessing the database, 8.8 seconds were spent over the network, and 5.1 seconds were spent building and buffering the data. With over 90% of the execution time spent accessing the database, it is clear that the database is the bottleneck for the synthesis task.

3.1 Database Organization

The database which MICON uses is built upon the table-based relational model [4]. Each row in a table represents a relationship among the values in the row. Expressions which identify sets of information are formulated using relational algebra and relational calculus. Expressions for sets of information are converted into database *queries* (requests for information). Access to the database

1. All measurements were taken from a Sun3 running SunOS.

is done using query languages. SQL (Structured Query Language) is an ANSI standard that is based on both relational algebra and relational calculus.

MICON employs the Informix Relational Database System and interacts with the database through SQL queries. Thus MICON's information is organized into tables. For example, MICON has a part table composed of abstract and physical parts. For example, a two input NAND gate would be an abstract part while a 74F00 would be a physical part The part table contains information about the name of the chip, an associated id, package type, etc. Other tables contain characteristics of the parts, the function of the parts, the pin information, and so on. One way to represent the logical structure of the information is through an entity-relationship diagram (E-R diagram) illustrating entities and the relationships between them [4]. An entity is an object that exists and is distinguishable from other objects. In an E-R diagram, entities are in boxes, relationships are in diamonds, and attributes are in ovals. A directed line from relationship A to entity B means that there is a one-to-one or a many-to-one relationship from A to B. An undirected line means that there is a many-to-many or many-to one relationship. Figure 4 is an E-R diagram for part of the MICON database. The diagram shows that entities are parts, characteristics, functions, specifications, logical pins, and physical pins. Each entity has associated attributes. Some of the key attributes have been included in Figure 4. For example, the part entity has a unique id, chip name, id for shared characteristics, and id to map a physical pin to a logical pin. The relationships between the entities are shown by the diamonds. For example, the function entity has a relationship with the part entity but not with the characteristic entity.

The manner in which the MICON tools interact with the database is through a client-server model built on IPC sockets (Figure 5). None of the tools make direct calls to the database. A set of client routines are provided to allow for simplified interaction with the database. The client routines package the requests into buffers which are sent through IPC packets. This interface provides a way to specify the type of information (part, characteristics, pins, etc.), the specific attributes (chip name, id, type, etc.), and the qualifications (part function is a NAND, etc.) that are required.

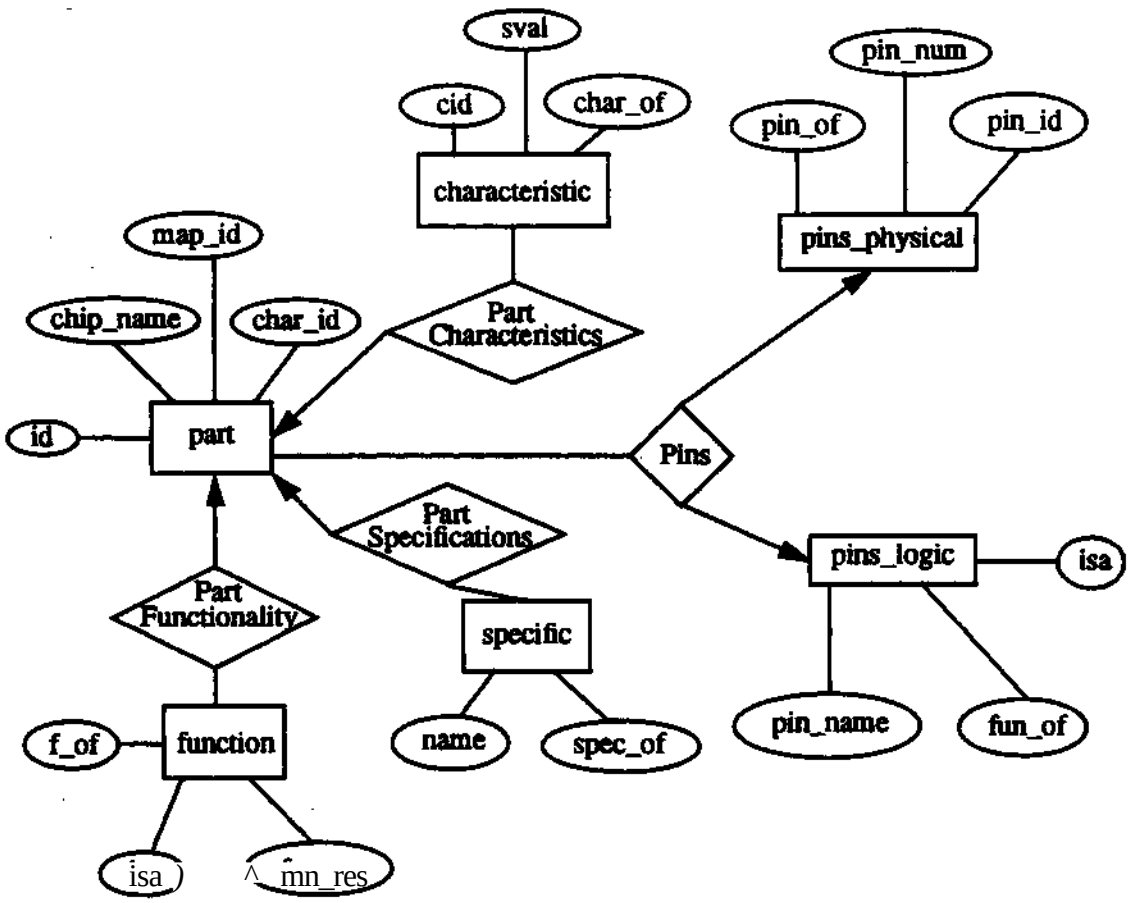
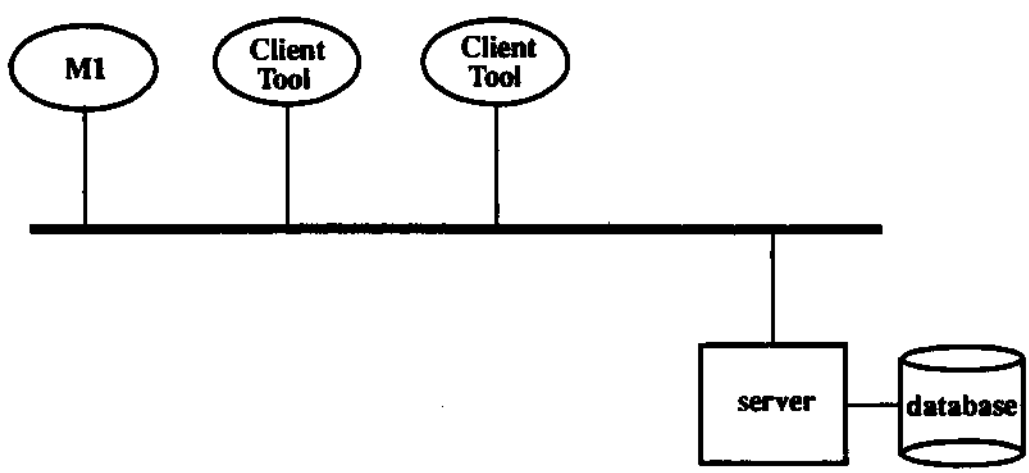


Figure 4 E-R diagram for part of the MICON database



Figures Client - Server Model

The actual database access is through a server process that interprets the client's request and transforms these requests into database queries. The results are then repackaged into a buffer and sent

back to the client. Thus besides spending time on its own processing, a client program must wait for the network, wait for the queries to be built and results buffered, and wait for the database to be accessed.

4 Using BBPO to Identify Bottlenecks

Without the source code, the database bottleneck clearly requires the use of BBPO since only the inputs to the database system are accessible. The input to the database system is SQL which can express a wide range of information requests. It can extract certain fields, columns, rows from a table or a collection of tables based on operators such as and, or, equals, etc. Analysis of Mi's database usage revealed that fourteen different SQL query types were being used during its execution (Figure 6). Some of the query types request the same number of fields from a table but they

```
Type 1
Retrieve the MTIOT value of * field in * table.

Type 2
Retrieve two ooluaas from a table.

Type 3
Retrieve the Miian value of a field in a table.

Type 4
Retrieve three coluans from a table.

Type 5
Retrieve two field* from a table where a field has a certain value.

Type €
Retrieve two fields from a table where a field has a certain value.

Type 7
Retrieve two fields from a table where a field has a certain value or the field has the value of the
result of retrieving another field from another table where a field has a certain value.

Type 6
Retrieve five fields from a table where a field has a certain value.

Type 9
Retrieve three fields from a table where a field has a certain value.

Type 10
Retrieve three fields from a table where a field has a value among the results of retrieving another
field from another table where a field has a certain value

Type 11
Retrieve a field from a table where a field has a certain value.

Type 12
Retrieve two fields from a table where a field has a certaom value.

Type 13
Retrieve three fields from a table where a field has a value among the results of retrieving another
field from another table where a field has a certain value and another field has a certain value

Type 14
Retrieve three fields from a table where a field has (a value among the results of retrieving another
field from another table where a field has a certain value) AMD another field has (a value among the results
of retrieving a field from a table where a field has a certain value and another field has another value)
```

Figure 6 Classification of each Ml Query Type

differ by the actual fields requested and the table requested. The fields requested and the size of

the table requested has a significant impact on the access time of the query. That is why they have been organized as different query types. Ml's method of synthesis-by-composition involves traversing a hierarchy of objects and putting together objects that fit the required criteria. MICON uses the database system to store the component hierarchy and information about each component The fourteen query types allow Ml to traverse the hierarchy and to retrieve information about each component in order to meet the required criteria. These query types will be referred to by number.

In BBPO, the first step is to understand the inputs to the BBR The same synthesis example of the application specific computer board was used. By profiling the frequency of each query type, a query mix can be found (Figure 7). This is analogous to an instruction mix for a program []. It is interesting to note that the query mix shows certain query types to be executed the same number of times. This is due to the pattern of requests by the synthesis task. In order to choose components that meet the criteria, Ml needs to make certain queries each time it looks at a component. Thus there are patterns in the query mix. Another way to look at the inputs is to measure the average time per query type (TPQ) as shown in Figure 8. This is similar to the cycles per instruction measurement used in computer instruction set performance evaluation. Looking at either of these measurements alone does not help identify the areas for improvement since they do not show the common case. Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of time the faster mode can be used []. Thus the queries where most of the time is spent should be the places with the greatest potential for improving the overall execution time. By multiplying TPQ by the query mix, the total time per query type can be examined and the queries which consume the most time can be identified (Figure 9). By focusing attention on the time consuming queries, the greatest performance improvement can be found.

In the case of Ml, query type 7 was found to be most time consuming (Figure 9). Examining query type 7 revealed it to be a compound query. Ml uses this type of query to access all the characteristics of a component. Since a component can be part of a family of components, the charac-

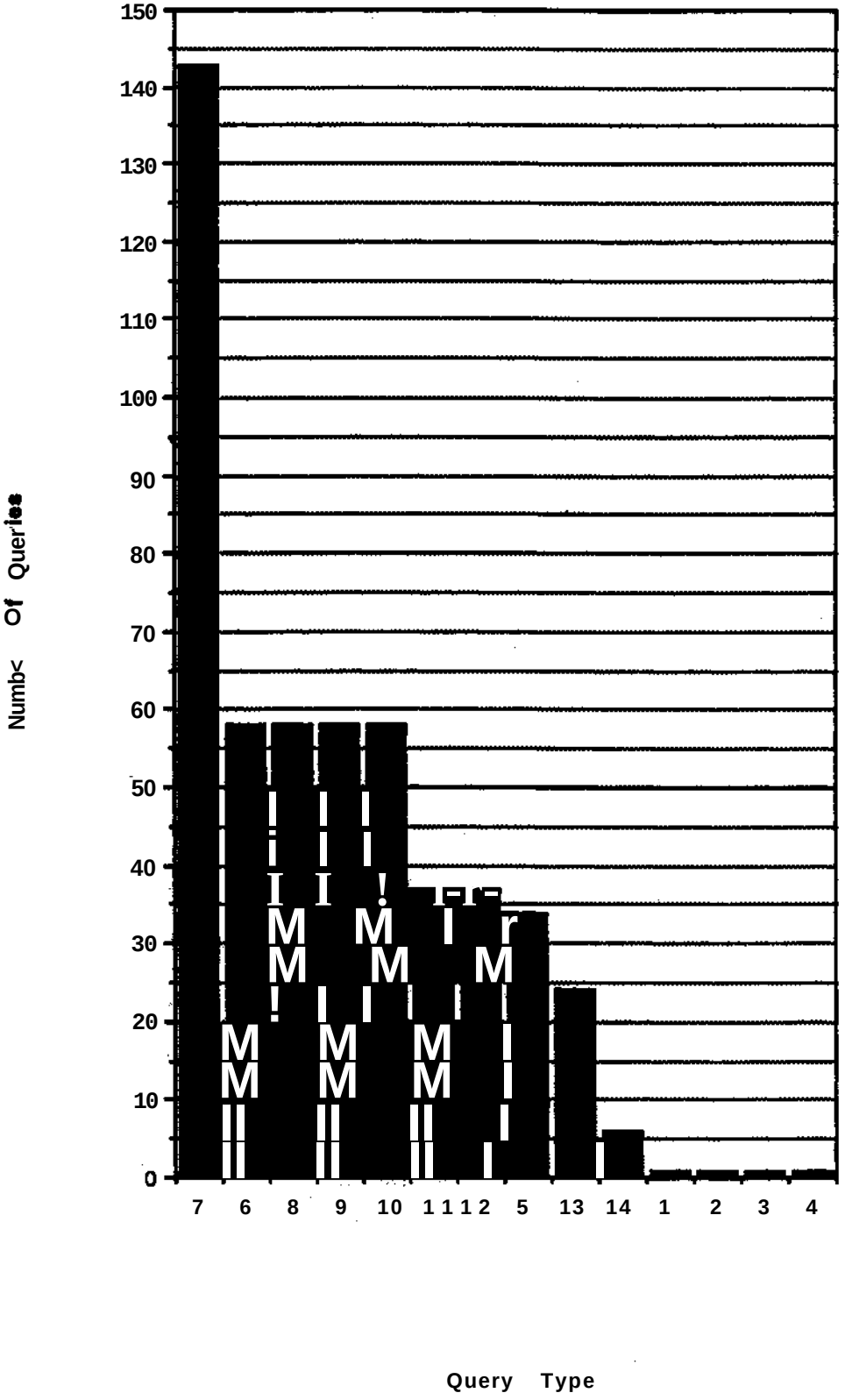


Figure 7 Query Mix for the 80188-based Design

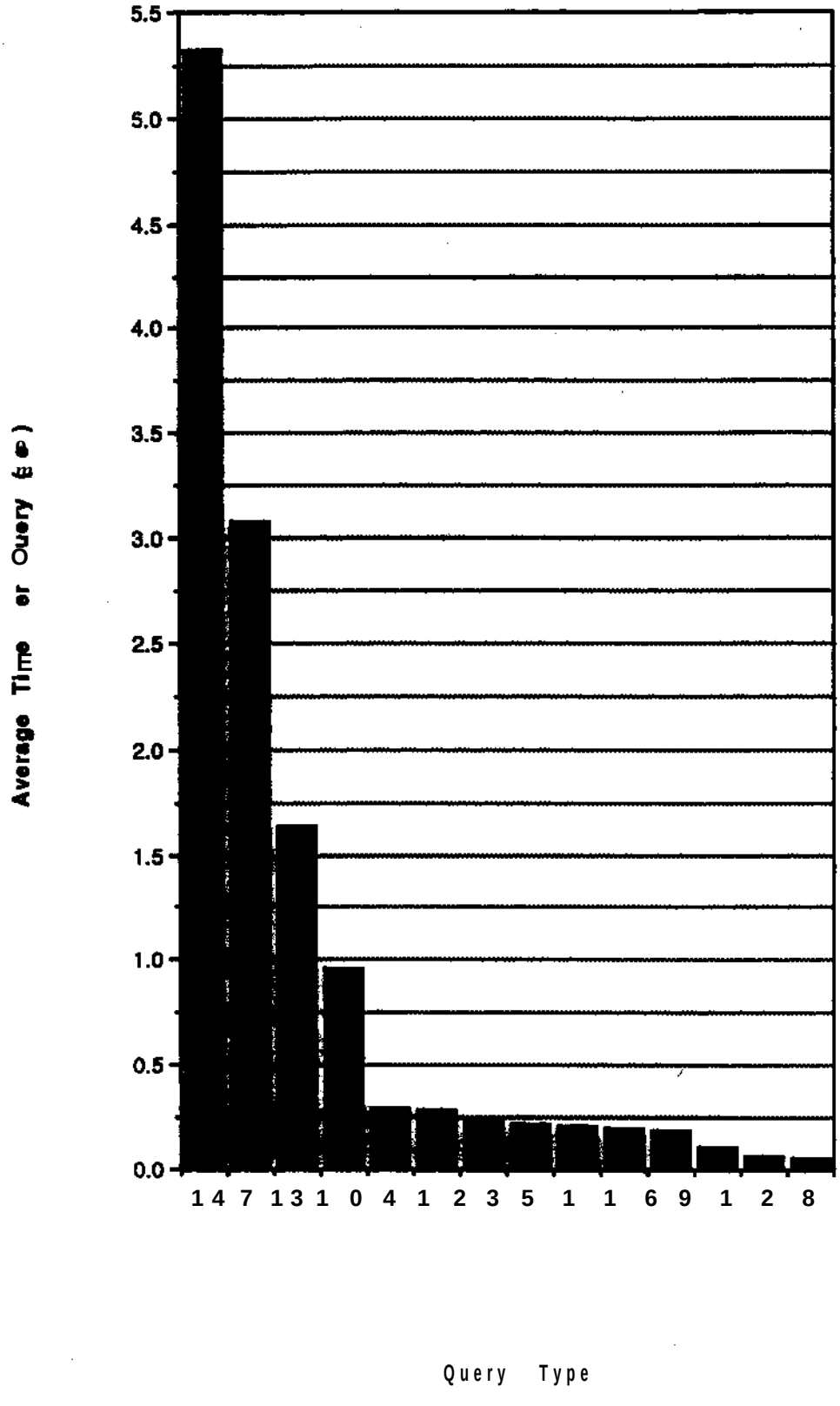


Figure 8 Average Time Per Query Type for the 80188-based Design

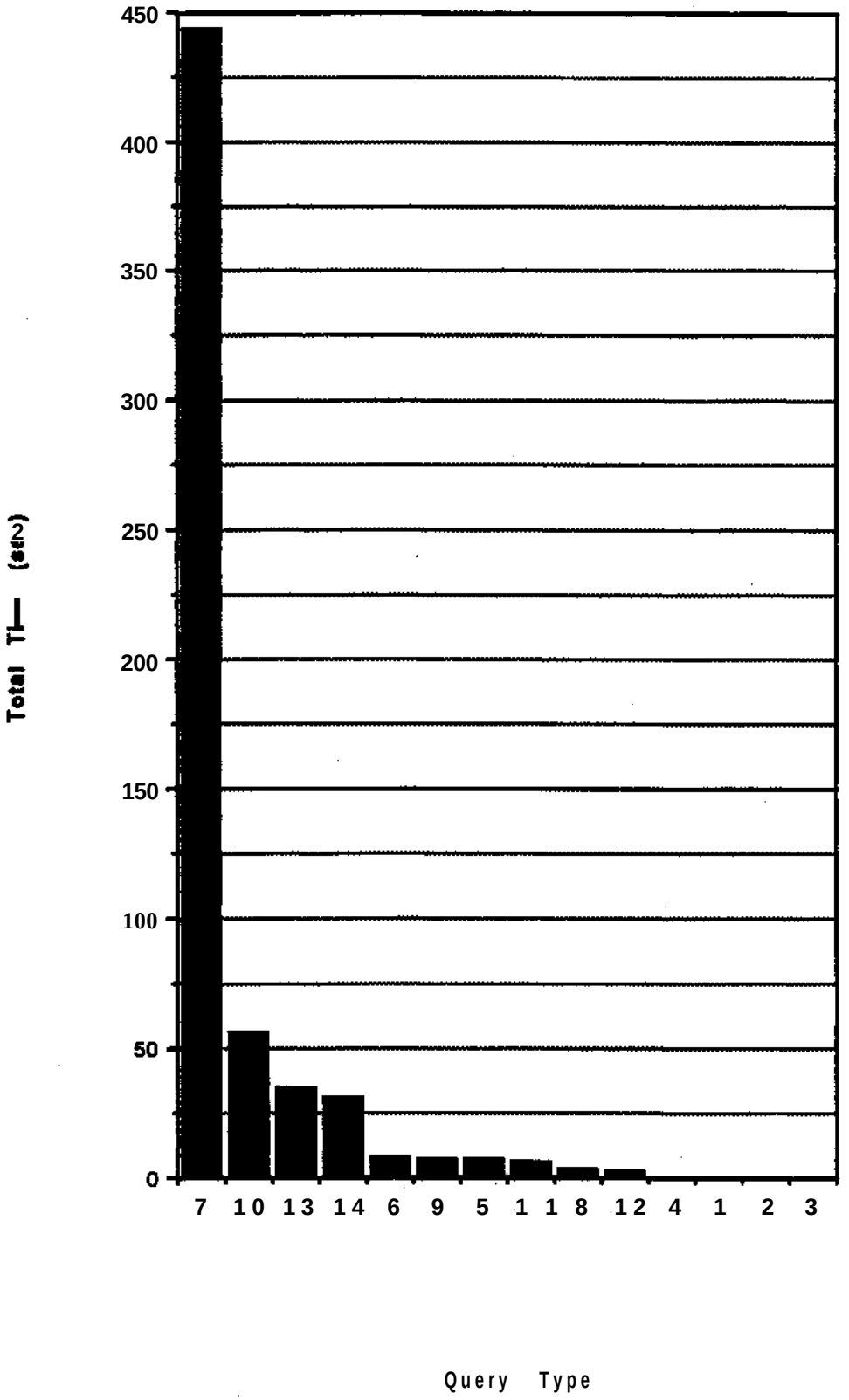


Figure 9 Total Time Per Query Type for the 80188-based Design

teristics are not only associated with the component's id, but also with the family's id. Thus this query was formulated with a subquery as part of the main query. Since the database system is a BBP, there is no way to see how the query is performed. Thus the effect of the compound query can not be understood. In order to try to improve performance of this query, the subquery could be done separately which would result in sending two simple queries to the database system. This was done by performing the subquery first and then storing the results. Then the main query was executed using the results of the subquery. The result was a 92% decrease in TPQ for query type 7. The TPQ went from 3.15 seconds to 0.15 seconds. Next, the impact of this tuning needs to be studied by re-evaluating the performance.

5 Re-evaluating the Performance

The speedup of the total database access time due to the performance improvement is found using:

$$Speedup = \frac{1}{(1 - Fraction_{enhanced}) + \frac{FraCti0n_{enhanced}}{Speedup_{enhanced}}}$$

The fraction enhanced was found by dividing the total time for query type 7 in the current version by the total query time for all queries. The speedup enhanced was found by dividing the current average time for the query by the average time for the improved query. The result was a 3.35 speedup for the database.

$$Speedup_{CGA\{Informix)} = \frac{1}{(1 - 0.737) + \frac{TM}{0.8}} = 3.35$$

Executing the synthesis task using the tuned inputs to the database resulted in execution time of the overall system to go from 10.92 minutes to 3.95 minutes which is a 2.76 speedup (Figure 10). The

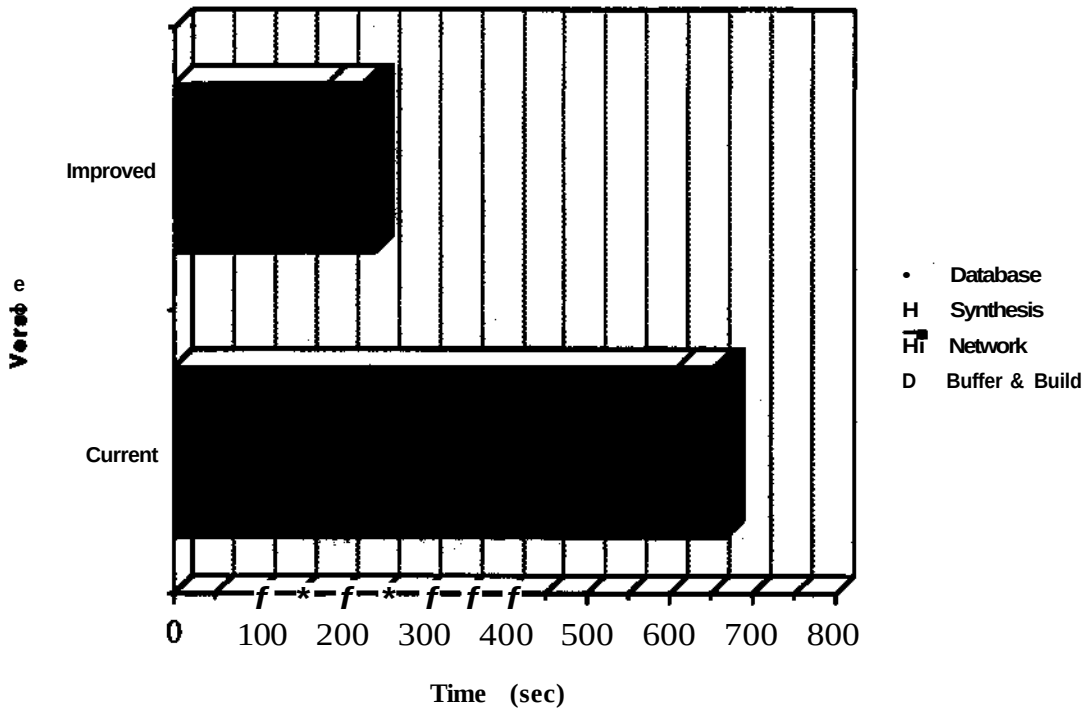


Figure 10 Execution Time Improvement Comparison for the 80188-based Design

full 3.35 speedup was not realized because the non-database execution time was not affected by the improvement to the database system

6 Other Uses of Database Query Profiling

It is often useful to predict the execution time of a system given a set of inputs. The method of examining database queries in order to improve database performance can also be applied to prediction. If the frequency of each query type is available for a task and the TPQ for the queries arc known, the performance time of the task using a different database system can be predicted. This is very useful for predicting the performance of a task under a different database system prior to porting the whole application. The frequency of each query type can be collected from the current

system. A simple driver program can be written to collect the TPQ for each query type on the new system. The predicted total database execution time would be:

$$ExecutionTime = \sum_{i=1}^{n} TPQ_i \times Q_i$$

Where Q_i is the number of times query type i is executed.

Using this method, the database execution time for Ml to synthesize a 68008-processor-based single board computer system was predicted. This design contained 33 physical parts, 104 abstract parts, and 92 nets. Figure 11 is the query mix for the design obtained by running Ml with the Informix database system. A prediction for the database execution time for Ml using the Sybase relational database system was perfoimed. The execution time for each query type was predicted. The system was then ported to use the Sybase database system. A comparison by query type between the predicted times and the actual times showed that the predictions were very close (Figure 12). A time of 351 seconds for Sybase was predicted where the actual time was 347 seconds. Thus the prediction was 1.2% higher than the actual time.

7 Conclusions

Tuning systems with black-box tools can result in significant performance improvement. Often performance issues are over-looked when dealing with black-box tools. Typically, performance improvements involving database systems result in replacing the database system. Here it was found that the inputs to the system must also be examined. The efficiency of certain queries must be considered.

Database implementation was also found to have tremendous impact on performance. The same queries that were used for Informix were made using Sybase. For the application specific com-

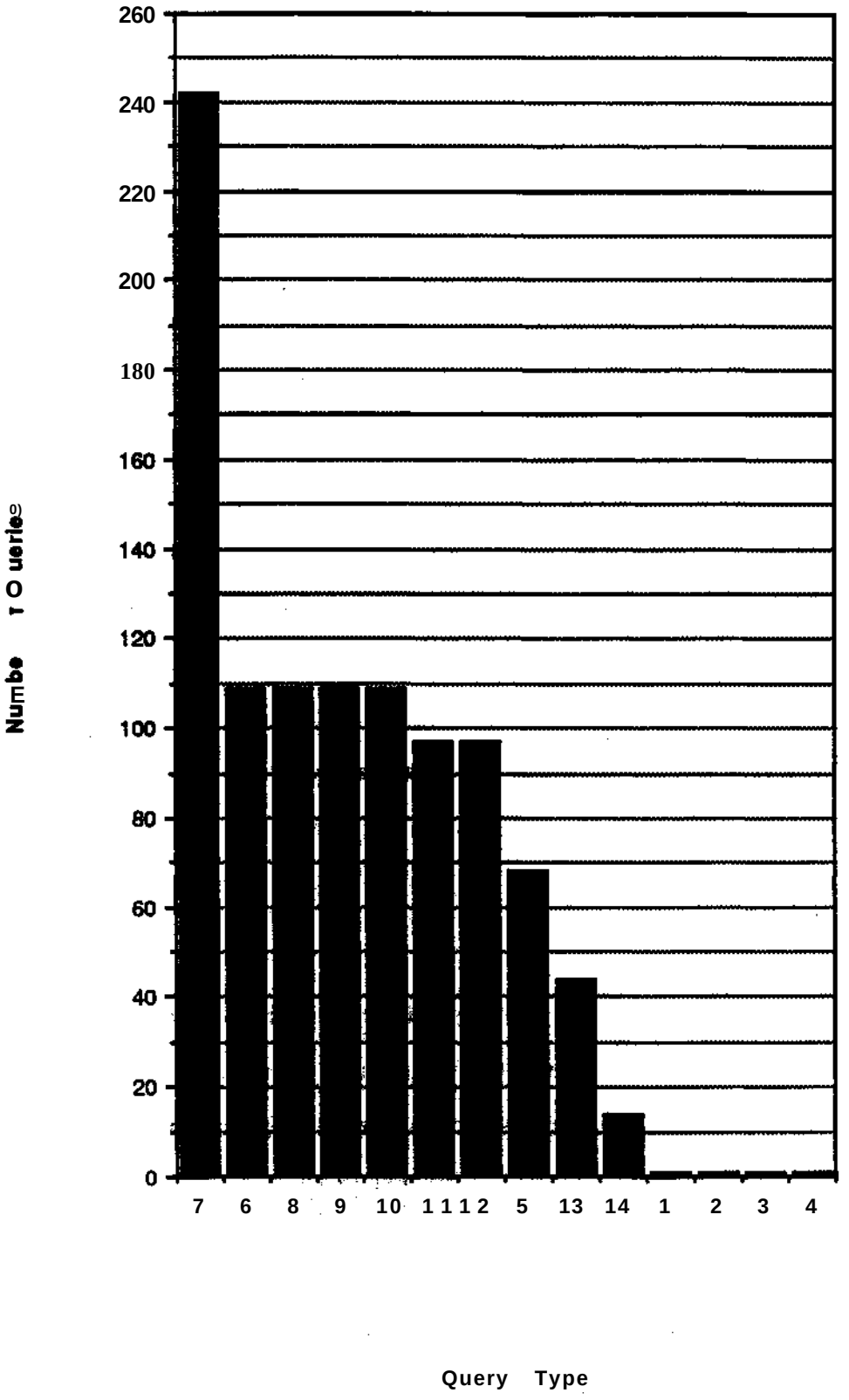


Figure 11 Query Mix for the 68008-based Computer Design

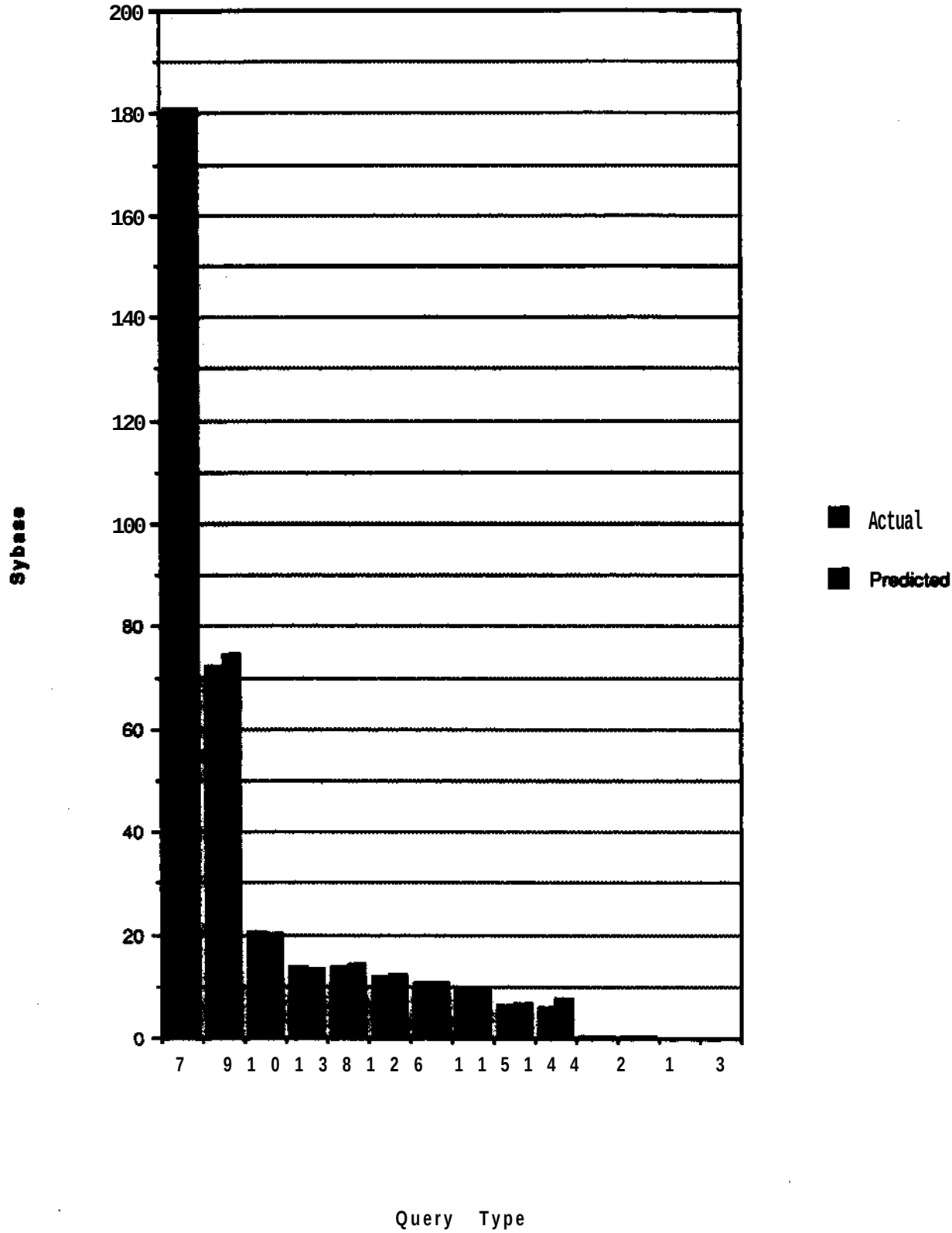


Figure 12 Actual and Predicted Performance for the 68008-based Design

puter board design, the synthesis task using Sybase required 246 seconds (Table 1). This is 37% of

Table 1: Comparison of Improved Query Performance

	Informix	Sybase
Current	**665 sec**	**246 sec**
Improved	**237 sec**	**221 sec**

the time required by Informix. By making the query improvement, the system under Informix went from 665 seconds to 237 seconds which is comparable to the non-improved query performance under Sybase. In this case, changing to a new database system was not the only solution. As for the performance of Sybase using the improved query, the overall execution time was reduced to 221 seconds which was the result of a 1.17 speedup of the total database time. This shows that the performance gain with the improved query probably is not confined to any particular database system but that the amount of performance gained is very dependent on the database implementation.

The performance gain achieved here was the result of changing a query to the database system. Other query structures or a different database organization could have been explored that potentially would have resulted in additional performance gains. Different techniques will apply to different BBPs. As seen here, when traditional performance optimizations cannot be performed, looking for performance gains with BBPs can be very beneficial. If mature systems such as MICON can improve performance by examining the database system, there is potential that other CAD systems can also benefit in this way.

8 References

[1] D Ferrari, "Computer Systems Performance Evalutation", Prentice Hall, (1978)

[2] D.P. Siewiorek, R.S. Swara "Reliable Computer Systems", Digital Press, Burlington, MA, (1992)

[3] W.P. Birmingham, A.P. Gupta, and D.P. Siewiorek, "Automating the Design of Computer Systems : The MICON project", Jones and Bartlett Publishers, Inc., Boston, MA, (1992)

[41HJF. Korth and A. Silberschatz, "Database System Concepts", McGraw-Hill, Inc., New Yoik City, New Yoik, (1991)

[5] J.L. Hennessy and D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, Inc., San Mateo, CA, (1990)