90-RP

RISC processors in a massively parallel database machine

Qiang Li, Naphtali Rishe and Doron Tal examine the application of RISC, in the form of transputers, to a system combining high-power processing and highly parallel I/O

The paper presents an application of RISC processors in a massively parallel database machine (LSDM). Depending on the configuration, the database machine can contain a few thousand processors, each of which is equipped with a disc drive. The database model used is the semantic binary model. The combination of the database model and the architecture of LSDM provides an environment where both massively parallel computation and massively parallel I/O are possible. This paper analyses the characteristics of LSDM and argues that RISC processors are the best choice for the database machine. This machine is presented as further evidence of the potential of RISC architecture. Performance evaluation and simulation results based on the characteristics of Inmos transputers are also presented.

microsystems database machines transputers interprocessor communication

RISC architecture[1-3] is a dramatic change from the traditional trend of improving processors by enriching their instruction sets. Many RISC machines have been built[4-6] and evaluated[7-9] and, although still controversial, the advantages of RISC architectures have become evident in many applications. The smaller instruction set characterizing a RISC processor yields fast execution speed for the hardware instructions implemented, less hardware complexity and high reliability. Since RISC processors eliminate the microprogramming level, compiler designers are given more freedom to optimize the programming code. With the rapid development of parallel processing, where more and more simple processors are needed by the system, the advantage of RISC processors becomes more evident.

School of Computer Science, Florida International University, Miami, FL 33199, USA
Revised paper received: 20 April 1990

Since RISC processors in general have primitive I/O capacities, their application to database machines has not been well studied. This paper discusses the application of RISC processors to a large-scale parallel database machine, namely the 'linear-throughput binary semantic database machine' (LSDM[10]). Below we discuss the basic structure of LSDM and the basic philosophy behind its design. The interprocessor communication network and some performance evaluation data are presented. The principal reasons for choosing RISC are outlined and the performance of the system is evaluated.

DATABASE MACHINE LSDM

Hardware structure

To meet the high demand on processing speed and throughput required by database machines, many hardware architectures have been built or proposed for database machines[11-18]. As parallel processing and architectures make important impacts on the development of computer systems, the concept is being introduced into database machines. In most cases, a multiprocessor database machine has a number of processors and a few large, high performance discs. The emphasis has been on the interprocessor networks. However, in practice, as the size of database systems and their functional complexity increase, system throughput is severely limited by the poor performance of the discs and I/O channels. Very high computation power is achievable without great difficulty using current technology, but there has not been a major breakthrough in secondary storage for a long time. As systems employ more and more processors, the I/O bottleneck becomes more severe.

To alleviate the problem, alterations were made to the processors-channels-discs formation of LSDM, and highly parallel I/O channels and secondary storage access power

0141-9331/90/06351-06

were introduced. In addition, the I/O interfaces were made as simple as possible so that a large number of RISC processors could be utilized.

The design of LSDM is based on the following observations

- A database transaction can be partitioned into several subtasks, each showing a certain locality in terms of secondary storage access.
- Data retrieved by a group of related secondary storage accesses needs substantial processor time to process.
- In a system with a large number of simultaneous users, different queries tend to access different secondary storage areas. Although it is generally agreed that 80% of queries reference 20% of the data, heavily referenced data can still be distributed over different secondary storage areas.

The basic structure of LSDM (see Figure 1) was based on these observations. It consists of many (a few hundred to a few thousand) fairly powerful processors, each equipped with local memory and a small capacity (e.g. 50 Mbyte) disc drive. The implication of the structure is that a large number of disc drives are used, which is a unique feature of this system. Since each processor is fairly powerful, it is able to process data retrieved by a sequence of secondary storage accesses, so the interprocessor communication load is a small percentage of the total I/O traffic. Massively parallel I/O operation by a large number of users is possible if the database model and its implementational data structures are properly chosen.

Semantic binary database model

The semantic binary database model[19-25] is, in the logical aspects of the representation of the application world, a variation of Abrial's binary model[26]. It represents information about an application's world as a collection of elementary facts of two types: unary facts categorizing objects of the real world and binary facts establishing relationships of various kinds between pairs of objects. The purpose of the model is to provide a simple, natural, data-independent, flexible, and non-redundant specification of information.

In this implementation, the entire database, including all indexing information, is represented by one logical coherent file partitioned into sufficiently small segments to allow storage on small discs. The number of processor-disc formations is sufficient to accommodate all segments. An important property of the implementation is that, for most queries, the necessary data is stored in a sequence of consecutive records. In most cases this localizes each simple query to a particular processor-disc unit. This helps to reduce significantly the traffic on the interprocessor communication network and also provides latitude for compiler optimization of register storage or local fast memory (on-chip) at the local processor level.

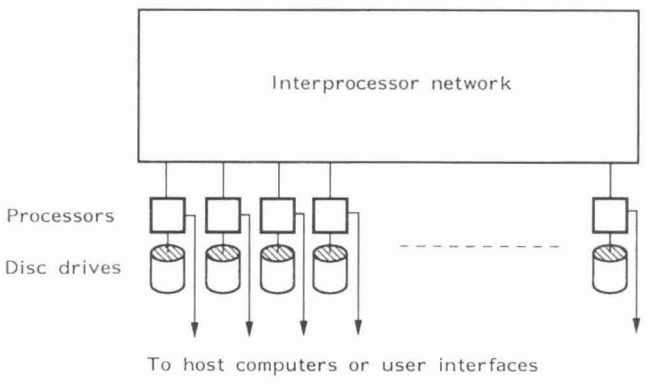

Figure 1. Structure of LSDM

INTERPROCESSOR COMMUNICATION NETWORK

Since a large number of processors are employed, interprocessor communication is an important part of the database machine. By the nature of LSDM, both short messages (a few to a few hundred bytes) and large data sets (many kbytes) must pass through the interprocessor network. The small messages are mostly the control messages or the results of small queries and appear frequently. Thus rapid and steady delivery time is desirable. For the large data sets, communication bandwidth is a more important issue. A higher bandwidth means a higher delivery speed for large data sets.

The hardware basis for the design of the network is the Inmos transputer[27]. Two types of interprocessor communication networks can be based on the available hardware components' characteristics: a circuit-switching network and a packet-switching network. These have been studied extensively and a collection of research results is surveyed in Reference 28.

Greater bandwidths can be achieved in a circuit-switching network, since a dedicated circuit is established between two processors whenever they need to communicate. However, circuit setup time becomes a disadvantage when frequent short messages are sent. Thus, circuit-switching networks are in general only suitable for use with the large data sets, where the data transmission time is much longer than the setup time.

Packet-switching networks, which normally have a static hardware connection, offer faster responses. Data is transferred as packages with destination addresses. Packages are normally sent from one node to another until they reach their destination. A packet-switching network has the advantage of quick response time and flexibility, i.e. it can send messages to any node in the system at any time. The drawback of this type of network is the low effective bandwidth when large data sets are sent, making it suitable only for small and frequent messages.

Since neither type of network could fully satisfy our requirements, a hybrid packet-switching and circuit-switching network was designed. The hybrid network, called Connect, is described here briefly; further details can be found in Reference 29.

Figure 2 shows the structure of the network. Logically, the network has two main parts: the packet-switching network and the circuit-switching network. The packet-switching network also serves as the control network of the circuit-switching network. The solid circles in the figure are the processors which do the data processing work, and are called data processors.

The packet-switching network consists of a number of communication processors connected in a tree-shaped topology. The top of the tree is not a single node; it is actually a hypercube network. Thus, the network can be viewed as a hypercube on the top with a subtree hanging out from each of the hypercube nodes. The operation of the network is straightforward. When a data processor

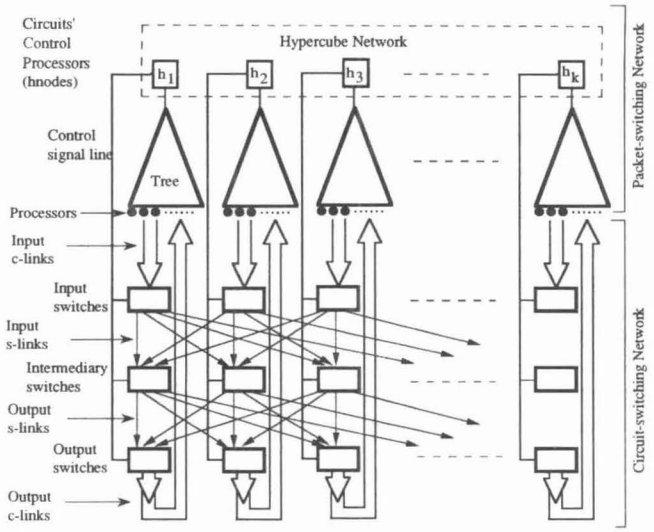

Figure 2. Structure of the network Connect

needs to send a short message to another data processor, it sends it with the destination address (the node number in the network). The messages will be sent through the network in a store-and-forward fashion.

The circuit-switching network is a three-stage Clos network[30]. When properly configured, for any given partition of the nodes into pairs, all the pairs can communicate simultaneously. When a data processor needs to send a large data set to another data processor, it first sends a circuit setup request to the hypercube network on top of the packet-switching network. The hypercube network is actually the control network of the circuit-switching network. The processor in the hypercube network will determine the best connection route for the requested connection and will send hardware signals to establish the connection. After the connection is set up, an acknowledgement message is sent to the data processor which made the request. The data processor can then communicate with its partner through the dedicated circuit.

The hybrid network inherits the advantage of both types of networks while avoiding their disadvantages. A performance evaluation of the network is given below.

USING RISC

As mentioned above, Connect is a building block-based architecture. There are two types of building blocks in Connect: data processors and communication processors. The structure has made the functions of each processor more specific. The relative advantages and disadvantages of RISC and CISC processors are discussed below.

Do we need CISC?

CISC machines have been designed mainly to close the semantic gap between operations supported by high-level languages (HLLs) and those supported at the machine level. Using this approach, computer architects hoped to reduce compiler complexity, reduce memory size, and improve execution efficiency. Unfortunately, studies show[31] that code generated by compilers is typically composed primarily of a few simple instructions. This applies strongly to the LSDM. Neither the communication processors nor the data processors in the LSDM need a large number of complex instructions. Thus, the potential advantage of supporting HLL is not compelling.

The basic function of a communication processor is to receive data packets through its input channels and forward them to appropriate output channels. The receiving operations are done by DMA engines. Each processor moves the messages from buffer to buffer in its local memory, and calculates the proper output channel numbers. Therefore, the hardware instructions needed are mostly assignments (memory copy) and simple integer comparisons. These operations are always available on RISC processors.

Owing to the implementation of the semantic binary model, the data stored in a disc drive appears to the related processor as a linear segment of sequential data. The logical operations are mainly table search and manipulation, and sorting and searching of non-floating-point data. In terms of machine language, the main operations are integer operations (e.g. comparison). Integer operations are well supported by most RISC processors.

One logical operation which appears frequently in database operations is the character string comparison or search. The operation will typically be translated by the compiler into a stream of data fetching and comparison operations. Owing to the highly pipelined nature of RISC processors, character string operations can be done much more efficiently by a RISC processor than by a CISC processor.

Why RISC?

Various types of RISC implementation have been suggested; most incorporate certain characteristic features:

- a small number of fairly simple instructions, most involving register-to-register operations, and a few load-store memory access operations
- a relatively small on-chip control area
- a large register set and single cycle execution of instructions

A discussion follows of the way in which these features enhance the performance of the LSDM.

On-chip area saved by RISC

Since RISC processors have a small number of instructions and no microprogramming level, the control circuitry is much simpler than CISC processors (e.g. the RISC I chip devotes only about 6% of its chip area to the control unit[7]). This makes available a greater area on the processor chip. Thus, other speed critical attributes can be accommodated on the chip to support the following:

- *Interprocessor communication support.* To support efficient interprocessor communication, on-chip communication channel DMA engines are needed. In general, the more independent channels a processor has, the smaller the diameter of the network and, thus, the higher the communication capacity. In addition, a built-in hardware router can increase the performance of the system significantly. RISC architecture provides area for building these supporting circuits.

- *High speed cache memory.* The most time-consuming database operations in memory are sorting and searching. The size of the code for these operations is normally small, so the code access normally shows a high locality. Moreover, the structure of implementation of the semantic binary data model in the LSDM brings with it a very high locality factor (e.g. related data segments reside in physical proximity). Therefore, a high speed cache memory is highly desirable. With RISC architecture, a large amount of local or cache memory can be accommodated on the processor chip, with access speed close to that of the CPU registers. Using this approach, the LSDM processors may view their local or cache memory as a cache front-end of the local discs, and may perform read, write-through and local (temporary) write functions. All local read/write operations can be performed rapidly enough to keep pace with execution in the local processor, and efficient algorithms should capitalize on this feature. (In fact, the transputer has a fairly large on-chip cache memory.)
- *Large register set.* The LSDM model of computation involves multiprogramming support. Each processor handles a fairly large number of tasks (data processors deal with transactions and queries simultaneously, while communication processors must respond to communication requests). This implies a need for fast and efficient context switching. The large set of registers typically available in RISC architectures is one way of meeting this demand. (It is interesting to note that the RISC-MIRIS has 2048 CPU general registers available to the user[32].)

Small instruction set and simple instructions

The fact that a RISC processor has a small set of simple instructions implies fewer bytes per instruction and no need for microprogramming technology, which leads to a higher execution speed (the instructions can be hardwired.)

Avoiding disadvantages of RISC

One fact working against RISC is its lack of support for floating-point operations. However, this is not the case in LSDM. A very small percentage of operations in typical databases involve floating-point operations. Had floating-point hardware support been built into the LSDM processors, it would remain largely idle.

Another criticism of RISC is its longer program code (though on average the code is less than 30% longer than on CISC machines). On the LSDM, the local memory attached to each processor is designed mainly for storing data, i.e. the ratio between the memory for code and memory for data is very small. Thus, small variations in code size are not significant.

Usually RISC processors can handle only primitive I/O operations, which discourages its application in database machines. However, this is not the case with LSDM. Since the entire database is mapped into a sequential file and each processing unit holds one contiguous segment, the I/O operations that each processor has to deal with are no longer complicated full scale I/O operations. Each processor deals with an exclusive disc drive on a request-and-answer basis. The basic I/O operation becomes simple read and write to a linear secondary storage space.

Other aspects

Although conclusions cannot be reached until an implementation or an extensive simulation is done, it does not seem to make sense to use CISC processors in LSDM. The utilization of the hardware used to support CISC architecture would be very low. A final and somewhat less significant point is that the large number of processors comprising the LSDM further raises the significance of having fairly simple and small processors, in order to reduce overall cost. Thus, a RISC processor as a substitute for CISC can reduce the cost of the system greatly without performance degradation. In addition, since fewer electronic components are used, the system packaging, power supply and heat dissipation are much easier.

PROTOTYPING AND SIMULATION

A prototype of the communication network consisting of 30 Inmos transputers and 6 disc drives has been built to prove the design concept and accuracy of the algorithms. The construction of a prototype of LSDM is underway.

Unfortunately, the small-scale prototype does not provide much information on performance of the real-scale system and simulation must be used to estimate the performance. There are two performance issues: performance of the overall database machine and performance of the network. A reliable performance evaluation of the former cannot be obtained until a large-scale prototype or the real system is built. However, the latter can be approximated by simulation. The result of a simulation of the network Connect appears below.

The simulated system uses a modified transputer as the hardware model. The system consists of 960 data processors and 256 communication processors. It is assumed that the data processors have four communication links and the communication processors have eight. All the communication links are serial links with an effective bandwidth of 0.8 Mbyte/s. The circuit-switching network is a three-stage Clos network, N(32,32,32), consisting of 96 Inmos C004 dynamic reconfigurable switches (32 × 32). The following assumptions are made for the data flow patterns:

- The interval time between two messages initiated by a data processor is a random variable with an exponential distribution.
- The message length follows a distribution with the density function

$$f(x) = 0.6\frac{1}{\sigma x\sqrt{2\pi}}e^{-1/2(\ln x/\sigma - \mu/\sigma)^2} + 0.4\lambda e^{-\lambda x} \quad (1)$$

  which is a combination of a log-normal distribution and an exponential distribution. With the distribution, 60% of the messages are less than 100 byte long and have a near log-normal distribution with an average message length of 50 byte; messages which are more than 100 byte long have a near exponential distribution with an average length of about 3000 byte.
- The destinations of the messages from a data processor are uniformly distributed.

Table 1 gives the performance data of the system under a total communication load of 120.1 Mbyte/s, and Figure 3

Table 1. Simulation results of Connect

Attributes	Long messages	Short messages	Total
Message rate (messages/s)	39023	56544	95567
Average message length (bytes)	3003	51	1256
Throughput (Mbyte/s)	117.2	2.9	120.1
Average delivery time (μs)	6072	917	3022
Average circuit-setup time (μs)	2101		
F-test value	1553.2	1999.1	1176.4

shows the relationship between the system throughput and the message delivery time.

To translate the Mbyte/s throughput into 'queries per second', a rough model for the system is used. In this implementation of the semantic binary model[33], the entire database is stored as a logical sequential file. Each domain in a database schema is implicitly implemented as an 'index key'. For every key, there is a consecutive segment in the logical file where the data is stored in a sorted order of that key. It has been proven that each elementary query needs, on average, approximately one disc access. This translates into the worst case of one communication access to a processor-disc unit per elementary query. The best case is that a simple query consisting of several elementary queries accesses only one processor-disc unit, in which case the communication access is much less than one per elementary query.

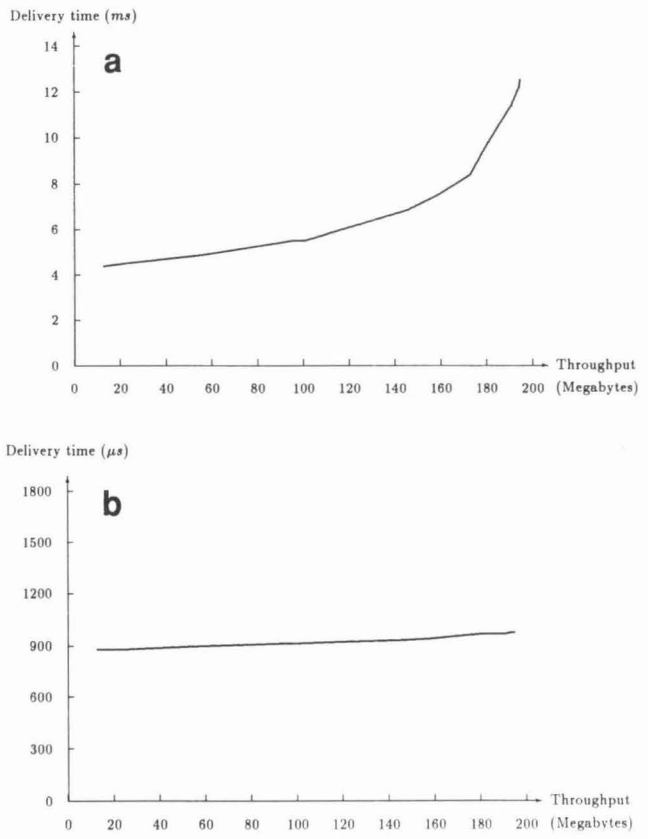

Figure 3. Throughput (delivery time). a, long messages; b, short messages

For lack of a real implementation and its details, the following assumptions were made: on average, a simple query requires access to four different processor-disc units (i.e. the simple query consists of at least four elementary queries); also, each access to a processor-disc unit results in a data transfer between processors of, on average, 3 Kbyte, which is large enough for data such as a personnel record, an insurance policy, a personal credit record, etc. The number of short messages depends on the sequence in which the processor-disc units are accessed. Since additional short messages other than the query requests are sent (e.g. synchronization messages) there should be more short messages than large data sets. Assuming 1.5 short messages per processor-disc access, there are six short messages per simple query. Note that the assumptions are pessimistic, since a simple query requires, on average, two elementary queries.

Under the system load of 120 Mbyte/s, if a simple query takes 10 interprocessor messages to solve, as assumed above, six short messages with an average length of 50 byte and four long messages with an average length of 3000 byte, the interprocessor communication network is able to transfer data for 9500 simple queries per second. If all the communication relating to a query is done serially, which is the worst case, the average additional delay for each query caused by using the parallel system is about 30 ms. This satisfies the design goal of about 5000 simple queries per second with capacity to spare. Notice that 9500 queries per second is not the limit of the system: with a 40 ms per query delay, the system can transfer data for 13 700 simple queries per second.

The above evaluation is based on a uniform random distribution of the messages. There are other cases that can be considered. When the communication patterns are patterns planned by the high-level applications instead of random patterns, the performance of the system will be improved in terms of delivery time and throughput. However, when the communication pattern is a nonuniform random distribution, the system performance can degrade significantly. Further study is needed to evaluate the system under different degrees of biased distribution and to find mechanisms that can alleviate the potential bottleneck according to the characteristics of different types of applications.

CONCLUSION

The work described represents a promising application of RISC processors to a large-scale parallel database machine. The use of RISC processors makes it possible to combine high computing power and highly parallel I/O efficiently and cost effectively.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the advice of David Barton, Nagarajan Prabhakaran and Scott Graham. This research was supported in part by the Florida High Technology and Industry Council.

REFERENCES

1 **Katevenis, M G H** *Reduced Instruction Set Computer Architecture for VLSI* MIT Press, Cambridge, MA, USA (1985)
2 **Stallings, W** *Tutorial: Reduced Instruction Set Computers (RISC)* IEEE Computer Society Press, USA (1986)
3 **Tabak, D** *Reduced Instruction Set Computer RISC Architecture* Research Studies Press, UK (1987)
4 **Radin, G** 'The 801 minicomputer' *IBM J. R. & D.* Vol 27 No 3 (May 1983) pp 237–246
5 **Hennessay, J L** 'VLSI processor architecture' *IEEE Trans. Comput.* Vol C-33 No 12 (December 1984) pp 1221–1246
6 **Seither, M** 'MIPS puts RISC processor on VMEbus, Multibus II' *Mini-Micro Syst.* (May 1986) pp 33–37
7 **Patterson, D A and Sequin, C H** 'A VLSI RISC' *IEEE Comput.* Vol 15 No 9 (September 1982) pp 8–21
8 **Piepho, R S and Patterson, D A** 'Assessing RISCS in HLL support' *IEEE Micro* Vol 2 No 4 (November 1982) pp 9–19
9 **Heath, J L** 'Reevaluation of the RISC I' *Comput. Architecture News* Vol 12 No 1 (March 1984) pp 3–10
10 **Rishe, N, Tal, D and Li, Q** 'Architecture for a massively parallel database machine' *Microprocess. Microprogramm. Euromicro J.* No 25 (1988) pp 33–38
11 **Su, S Y W, Nguyen, L, Ahmed, E and Lipovski, G** 'The architectural features and implementation techniques of multicell cassm' *IEEE Trans. Comput.* Vol C-28 No 6 (June 1979) pp 430–445
12 **Ozkarahan, E A, Schuster, S A and Smith, K C** 'RAP — an associative processor for data base management' in *Proc. AFIPS Conf.* Anaheim, CA, USA (1975) pp 370–387
13 **Lin, C S, Smith, D C P and Smith, J M** 'The design of a rotating associative memory for relational data base applications' *ACM Trans. on Database Syst.* Vol 1 No 1 (1976) pp 53–65
14 **Rudolph, J A** 'A production implementation of an associative array processor STARAN' in *Proc. Fall Joint Comput. Conf.* Las Vegas, NV, USA (1972) pp 229–241
15 **DeWitt, D J *et al*** 'GAMMA: a performance dataflow database machine' in *Proc. 12th Int. Conf. on Very Large Data Bases* Kyoto, Japan (August 1986) pp 228–237
16 **Fei, T, Baru, C K and Su, S Y W** 'SM3: a dynamically partitionable multicomputer system with switchable main memory modules' in *Proc. Int. Conf. on Comput. Data Eng.* Los Angeles, FL, USA (April 1984) pp 42–49
17 **DeWitt, D J** 'DIRECT — a multiprocessor organization for supporting relational database management systems' *IEEE Trans. Comput.* Vol C-28 (June 1979) pp 395–406
18 **Kitsuregawa, M, Tanaka, H and Moto-oka, T** 'Relational algebra machine GRACE' in *RIMS Symp. Softw. Sci. and Eng.* (1983) pp 191–212
19 **Rishe, N** *Database Design Fundamentals: A Structured Introduction to Databases and a Structured Database Design Methodology* Prentice-Hall, Englewood Cliffs, NJ, USA (1988)
20 **Rishe, N** *Database Design: The Semantic Modeling Approach* Prentice-Hall, Englewood Cliffs, NJ, USA (1990)
21 **Rishe, N** *'Data base design fundamentals: concepts and a case study' Technical Report TRCS84-16, University of California, Santa Barbara, CA, USA* (1984)
22 **Rishe, N** *Database Design Fundamentals, International Edition* Prentice-Hall International, Englewood Cliffs, NJ, USA (1988)
23 **Rishe, N** 'Semantic database management: from microcomputers to massively parallel database machines' in *Proc. 6th Symp. Microcomputer and Microprocessor Applications* Budapest (October 1989) pp 1–12
24 **Rishe, N.** 'Transaction-management system in a fourth generation language for semantic databases' in **Hamza, M H (Ed)** *Mini and Microcomputers: From Micros to Supercomputers, Proc. ISMM Int. Conf. on Mini and Microcomputers* Acta Press, Miami Beach, FL, USA (December 1988) pp 92–95
25 **Rishe, N,** 'A predicate-calculus based language for semantic databases' in *Proc. PARBASE-90: Int. Conf. on Databases, Parallel Architectures, and their Applications* IEEE Computer Society Press, USA (1990) pp 424–429
26 **Abrial, J R** 'Data semantics' in **Klimbie, J W and Koffeman, K L (Eds)** *Data Base Management* North-Holland, Amsterdam, The Netherlands (1974)
27 *Transputer Architecture Reference Manual* Inmos Corporation, Bristol, UK (1986)
28 **Wu, C and Feng, T** *Tutorial: Interconnection Networks for Parallel and Distributed Processing* IEEE Computer Society Press, USA (1984)
29 **Li, Q and Rishe, N** 'CONNECT — an architecture for a highly parallel system based on building blocks' Submitted to *IEEE Trans. on Comput.*
30 **Clos, C** 'A study of nonblocking switching networks' *Bell System Tech. J.* (March 1953) pp 406–424
31 **Fairclough, D A** 'A unique microprocessor instruction set' *IEEE Micro* Vol 2 No 2 (May 1982) pp 8–18
32 **DuBose, D K, Fotakis, D K and Tabak, D** 'A microcoded RISC' *Comput. Architecture News* Vol 14 No 3 (June 1986) pp 5–16
33 **Rishe, N** 'Efficient organization of semantic databases' *Inf. Syst.* (1988) Submitted

Qiang Li received a BS in electrical engineering from Xi'an Jiaotong University, Xi'an, Shannxi, China, and MS and PhD degrees in computer science from Florida International University, Miami, FL, USA. He is visiting assistant professor in the School of Computer Science at Florida International University. From September 1990 he will be an assistant professor in the Department of Electrical Engineering and Computer Science at Santa Clara University, CA, USA. His primary research interests are in parallel architectures, parallel algorithms and database machines.

Naphtali Rishe received BS and MS degrees in computer science from Israel Institute of Technology (Technion), and a PhD in computer science from Tel Aviv University, Israel. He is currently working as an associate professor in the School of Computer Science at Florida International University, Miami, FL, USA. His primary research interests are in databases, database machines, software engineering, and application of computer science in biomedical sciences.

Doron Tal received BS and MS degrees in electrical and computer engineering, and a PhD in computer science from Ben Gurion University, Beer-Sheva, Israel. He is currently an assistant professor in the school of computer science at Florida International University, Miami, FL, USA. His research interests are in parallel architectures, systolic arrays, database machines and operating systems.