TR-091

A Hardware Pipeline Algorithm for Relational Database Processing and Its Implementation Using a Dedicated Hardware

Shigeo Kamiya, Kazuhide Iwata,
Hiroshi Sakai, Susumu Matsuda
(Toshiba Corp.)
Kunio Murakami and Shigeki Shibayama
(ICOT)

November, 1984

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Hardware Pipeline Algorithm for Relational Database Processing and Its Implementation Using a Dedicated Hardware

Shigeo Kamiya   Kazuhide Iwata   Susumu Matsuda   Kunio Murakami
Hiroshi Sakai   Shigeki Sibayama
Toshiba R & D Center   Toshiba Ohme Works   ICOT Research Center
Kawasaki, Japan   Ohme, Japan   Tokyo, Japan

Abstract

A relational database machine Delta has been developed at the Institute for New Generation Computer Technology(ICOT). The main component of Delta is a relational database machine(RDBE) which performs the relational algebra operation consisting of a sorting operation and a relational database operation. The RDBE consists of a sorting operation hardware called the sorter, a relational database operation hardware called the merger, other special hardwares called the IN module, data input adapter, data output adapter, and a general purpose CPU(central processing unit) which controls these hardware components. The sorting operation is based on a straight two-way merge-sort algorithm[1]. The relational database operation is based on a hardware pipeline algorithm.

In this paper, firstly, we describe the design consideration of the RDBE and these algorithms. Next, we describe the design detail, implementation, and performance estimation of the sorter and the merger.

1. Introduction

Recently, the relational data model is used on some general purpose computer systems. The relational algebra operation is a time consuming operation and imposes a heavy burden on these computer systems. Intensive research has also been done on hardware implementation of various relational algebra operations.

One of them consists of the sorting operation which arranges each tuple in ascending order or in descending order and the relational database operation which selects some tuples satisfying a designated condition for each operation. We adopted these operations and designed an RDBE including the sorter and the merger.

We will describe the properties of this RDBE for implementing in the database machine enviroment as follows.

This RDBE can operate on relations including the null tuple whose key field value is null and can deal with tuples whose length and key field length are 2 through 4096 bytes. This RDBE can operate on three data types: integer number, binary number, and floating point number. The field ordering function which rotates the key field to the top of the tuple is performed in the IN module. So, the sorter can perform the pipelined sorting. The field selection function, which selects a part of the tuple and outputs it, is performed in the merger. So, this RDBE can perform a projection operation at the same time as another relational database operation. This RDBE can deal with a large relation whose stream consists of a large number of tuples, by means of chopping the stream into some substreams within the capacity of the sorter and operating them in the merger. The sorter consists of 12 sorting elements called the sorting cell or cell and a checking element called the sort checker. The sorting cell has one memory which stores a part of or all of the tuples in the stream, a comparator, and a selector. The cell stores them into the memory, reads out two tuples from the memory, compares them, selects one of them, and outputs it. But in one step, one of the two compared tuples is compared with the other tuple read out from the memory before it is written into the memory. Thus the sorting cell eliminates the loss time of writing into the memory. The hardware pipeline algorithm is based on the merging process which merges two sorted streams, and so the merger is somewhat like the sorting cell, though the merger is more complex than the cell. The merger can perform the join-equal operation by the merging process using a duplicated value, that is to say, the join-equal operation time between M tuples and N tuples is not of the order [M*N] but of the order [M+N]. The cell is controlled by the states by which different kinds of operations are defined. On the other hand, the merger is controlled by a ROM(read only memory) called the control ROM in which different kinds of operations are defined. The sort checker can detect sorting errors, and can detect a duplicated value, that is, whether the key field value of the current tuple is equal to the value of the succeeding tuple or not and can transfer the duplicated value to the merger. The duplicated value is useful in our proposed hardware pipeline algorithm.

In this paper, these properties represent important factors in the design of the RDBE and are described in detail in later sections. In sections 2 and 3, we describe an overview of Delta and its main component, the RDBE. In section 4, we describe considerations of the RDBE. In sections 5 and 6, we describe the design and implementation of the sorter and the merger. Performance estimation is described in section 7. Section 8 is the conclusion.

2. Overview of the Delta architecture

The relational data model is useful not only for conventional database systems, but also for logic database systems[6],[7]. The ICOT research center is developing a knowledge base machine which consists of a sequential inference machine(SIM)[8] and a relational database machine(Delta) as a part of the Fifth Generation Computer System(FGCS) project. This project is conducted under a program set up by the Ministry of International Trade and Industry(MITI).

We present an outline of the architecture of Delta in this section. The architecture of Delta is shown in Fig.1. Each component of Delta is described as follows.

(1) The IP provides the interface function, such as transferring the relation from the SIM to the HM or vice versa and the relational algebra level commands called Delta-commands from the SIM to the CP.

(2) The CP provides database management functions, such as translation of a

Delta-command into internal subcommands and concurrency control.

(3) The RDBE is a key component of the Delta system. There are four RDBEs which can be used in parallel. An RDBE is described in detail later.

(4) The HM stores and accesses every relation. It consists of a 128 Mbyte main memory and a 20 Gbyte disk. It transfers these relations to the RDBEs or from the RDBEs through a channel.

(5) The MP provides functions that enhance Delta's reliability and serviceability.

3. Overview of the RDBE architecture

An RDBE provides the function of performing relational database operations by means of pipelined sorting and merging algorithms. The RDBE configuration is shown in Fig.2. The RDBE consists of

(1) the CPU, which provides control of RDBE hardware and performs a part of the relational database operations, such as the Aggregate function, complexed Unique operation, Set operation, and Check operation which the RDBE hardware can not perform,

(2) the HM adapter(IN), which receives a relation from the HM and transfers it to the IN module,

(3) the IN module, which transforms the data format, rotates the field of the tuple as shown in Fig.3, and checks whether the key field value is null or not,

(4) the Sorter, which can sort up to 4096 tuples in ascending order or descending order and can achieve stable sorting, and the Sort Checker, which checks sorting errors,

(6) the Merger, which provides relational database operations,

(7) the HM adapter(OUT), which transfers the result from the merger or the CPU to the HM.

In Fig.2, DT, PT, NL, and DP indicate 16 bit data lines, two bit parity lines, one bit null line and one bit duplicated line. The NL denotes whether a null tuple(*) is on the data lines or not. The DP denotes whether a duplicated tuple(**) is on the data lines or not.

(*) A tuple whose key field value is null is called a null tuple.

(**) A tuple whose key field value is equal to the key field value of the next tuple is called a duplicated tuple.

The main data path is from the HM to the HM, through the HM adapter(IN), the IN module, the sorter, the merger and the HM adapter(OUT). If a relational database operation is concerned with two relations such as the join operation, one of them is first written into the memory in the merger through the sorter from the HM. Then, as soon as the other is input into the merger, both of them are compared and the results are output to the HM. If the result must be operated by the CPU, it is transferred to the CPU through the HM adapter(OUT). After the CPU has finished the operation, the final result is transferred to the HM through the HM adapter(OUT).

4. Design considerations of the RDBE

4.1. Treatment of the null tuple

The relational database operation usually deals with null tuples. An RDBE can process a stream including null tuples. The sorter arranges all the null tuples to the last position of the stream in the order of input tuple sequence. The merger can operate the stream including these null tuples, too. For example, the merger can eliminate all the null tuples or can pick up only null tuples.

A tuple has a tag that indicates whether the value is null or not. We prepared a piece of hardware called the IN module which checks whether the value is null or not in order to decrease the hardware complexity of the sorter and the merger. A null bit line(NL) which communicates with the sorter whether a null tuple is on the data lines or not is added between the IN module and the sorter as shown in Fig.2.

4.2. Data length

The data length varies widely depending on the application. It is desirable to deal with variable-length data. But to do so increases the hardware complexity of the sorter and the merger. So, we decided that all the tuples of a stream have to be of the same length and the tuple length is 2 through 4096 bytes restricted to multiples of two. The key field length is similar to the tuple length. Each key field length of all tuples is the same and is of an arbitrary length from 2 bytes up to 4096 bytes.

4.3. Data type

The relational database operation deals with many data types. It is difficult to deal with all of them by the RDBE. We decided that our RDBE deal with the integer number, floating point number, and binary number. The sorter deals only with the integer number in order to decrease the hardware complexity of the sorting cell. In the IN module, other data types are transformed into the binary number. In the merger, they are transformed into the original data type.

4.4. Ordering of the key field

We must sort tuples whose key field is placed in an arbitrary position. On the other hand, it is necessary that the key field is placed on the top of the tuple for the sorter to be able to perform pipelined sorting, because other fields must not be output to the next cell before the key comparison is complete. We decided that the key field is placed on the top of the tuple by the IN module. Fig.3 shows this function. Each sort cell deals with tuples whose key field is located on the top. Reordering of the key field is done by the merger.

4.5. Field selection

It often happens that some relational database operations need not output key fields. In this case, a projection operation which eliminated these fields has normally been performed after these operations were performed. In the merger, we provide a function, which is called a projection operation, that a part of the tuple is selected and output to the HM, at the same time as performing another relational database operation such as the join operation.

5. Design and implementation of the sorter

The sorter provides the function which sorts the input stream and outputs it to the merger. Until now, many hardware sorting algorithms have been investigated, but there have been very few implementations of practical sorters. Therefore, we have designed a practical sorter for application to the relational database operation. We decided to adopt a serial input, serial output sorter in order to perform the sorting process and the merging process in pipeline fashion. We could not adopt a sorter which consisted of a large number of sorting cells because we had to implement them by conventional technology. But we needed to sort as many tuples as possible. Thus, we decided to use an algorithm which could sort N tuples by [log N] sorting cells. We further needed to output the sorted tuples sequentially. So, we selected a straight two-way merge-sort algorithm. This algorithm needs sorting cells of only order[log N] to sort N tuples and requires a sorting time of order[N]. In this algorithm, one of these sorting cells has to memorize all the tuples of the stream. And, if we add N sorting cells, the memory capacity becomes 2**N times. As a result, we made the sorter to be able to sort 4096 tuples by 12 sorting cells. We made the sort checker for the purpose of high reliability.

We can have each sorting cell in common except the capacity of the memory that memorizes part of or all of the tuples of the stream for a while. It is possible to make a sorter to consist of N sorting cell LSIs and N memories whose capacities differ from each other. We developed this sorting cell LSI by using a gate array.

5.1. The straight two-way merge-sort algorithm

The straight two-way merge-sort algorithm performs the sort operation by repeating the merging process. Each sorting cell combines two streams of sorted tuples into a single stream of sorted tuples. We call this stream of sorted tuples a string. This process is shown in Fig.4. Fig.5 illustrates an outline of this sorter. In this figure, each U-memory and L-memory can memorize 2**(i-1) tuples in the i-th cell and provide the function of FIFO(first-in first-out). Therefore, the i-th sorting cell inputs two strings of sorted 2**(i-1) tuples from the(i-1)-th cell, merges them, and outputs a single string of sorted 2**i tuples to the(i+1)-th cell.

5.2. Design detail

The sorting cell takes two modes of operation. The first is the sorting mode and the other is the passing mode. In the sorting mode, the cell merges two strings into a new string. If the tuple length is larger than 16 bytes, the preceding cells take the passing mode. If the number of tuples is smaller than 2048, the succeeding cells take the passing mode. The sort checker also takes two modes, namely the checking mode and passing mode. In the checking mode, the sort checker checks the value of all two adjacent tuples. And, if it detects sorting errors, it interrupts the CPU. In the passing mode, the sort checker outputs all tuples without checking them.

5.2.1. Circuits of the sorting cell

The sorting cell is shown in Fig.6. The cell consists of four registers(INR, UR, LR, OUTR), two memories(U-memory, L-memory), an address control circuit, a comparator, a selector. We developed an LSI for sorting cells, excluding U- and L-memories. So, we needed to minimize the number of I/O pins, address pins and data pins through which this LSI accessed to these memories. So, we put the U-memory and the L-memory together into one memory called U/L memory. The U-memory is the former half of the U/L memory and the L-memory is the latter half of this memory. The U/L memory of the i-th cell is a 2**(i+4) byte RAM(random access memory). This memory is a 1.5*(2**(i+4)) byte RAM if two parity bits and a null bit are included in the written data. In a cell, the main data path is from the INR to the OUTR through either the U-memory and the UR or the L-memory and the LR, the comparator, and the selector. A string is stored into one of these memories. That is to say, the first string is written into the U-memory and the second string is written into the L-memory. After this, each string is alternately written into one of them. Two tuples called the U-tuple and L-tuple which are read out from the U-memory and L-memory are compared, selected and output via the selector and OUTR.

The address control circuit consists of five registers(RUAR, RLAR, WTAR, BUR, and BLR). These registers are shown in Fig.6(b). One of the RUAR, RLAR and WTAR is selected and the content is sent to the U/L memory. Two tuples are compared, one of them is output to the next cell, and the other is read again. We need to read again the remaining tuple. So, we need to keep the top address of this tuple. We save the top address of the U-tuple to the BUR and the top address of the L-tuple to the BLR.

The comparator is a 16 bit binary comparator. It compares the U-word(*) with the L-word(**).

(*)U-word is the word read out from the U-memory and set to the UR register.

(**)L-word is the word read out from the L-memory and set to the LR register.

The selector selects one of them as the result of the comparison, and outputs it to the next cell via the OUTR.

5.2.2. Control structure in the sorting cell

The control structure in the sorting cell contains three nested loops: a word processing loop, a tuple processing loop and a string/stream processing loop.

(1) A word processing loop has three states; the URD-state, LRD-state, and WT/C-state called URD, LRD, and WT/C. The state transition in this loop is shown in Fig.7(a). The URD is the first state of this loop and operates so that a U-word read out from the U-memory is set in the UR. The LRD operates so that a L-word from the L-memory is set in the LR. The WT/C operates so that a word in the INR is written into the U/L memory, the U-word and the L-word are

compared by the comparator, and one of them is selected and output to the next cell. The next state is the first state URD.

Each state of a word processing is controlled by a clock whose cycle time is 220ns. So, the interval of one word processing is 660ns. This interval is synchronized with the data transfer rate of 3MB/sec.

(2) A tuple processing loop has three states; the CMP-state, SEL.U-state, and SEL.L-state called CMP, SEL.U, and SEL.L. Fig.7(b) illustrates the state transition of this loop. Each state transfers to one of these states including itself at the end of the word processing loop according to the comparison result, null value, and etc.. The CMP is the first state of this loop, and is maintained until the comparison result is obtained. At this state, the U-word is selected and output to the next cell. If the U-word is selected as the result of the comparison, the next state will be the SEL.U, and the U-tuple is output. If the L-word is selected, the next state will be the SEL.L, and the L-tuple is output. These states are maintained until the last word of the tuple is output, and then the state transfers to CMP. In this loop, if the U-tuple is equal to the L-tuple, then the U-tuple is able to be output. So, the sorter can achive stable sorting.

(3) The uppermost string/stream processing loop consists of 13 states, state.01 through .06 and state.11 through .17 called S01-S06 and S11-S17. Fig.7(c) illstrates the state transition of this loop. Each state transfers to one of these states including itself when the state transfers to CMP in the tuple processing loop.

S01 operates so that the first string is written into the U-memory. As soon as the last tuple of this string is written, the state transfers to S02. Both S02 and S03 operate so that the second string is written into the L-memory, the U-tuple and the L-tuple are compared, and one of them is output to the next cell. When the state transfers to S02, a tuple compared with the U-tuple is not written into the L-memory. The comparison process between the INR and the UR is made in S02 for the purpose of eliminating the loss time of the first tuple being written into the L-memory. S02 is maintained until the U-tuple is selected and output. If the U-tuple is selected, the state transfers to S03. In this state, both the U-tuple and the L-tuple are read out from the U/L memory. After all the tuples of the second string are input in S02 or in S03, the state transfers to either S04, S05, or S06. An operation of S04 is similar to S03, except that the third string is written into the U-memory. S05(or S06) operates so that the third string is written into the U-memory and the remaining U-tuples(or L-tuples) are output. After all the remaining tuples are output, the state returns to S02. At this time, all the tuples of the third string are written into the U-memory. After this, the same process is repeated.

When all the tuples of the stream are input, the state transfers to different states, S11 through S17. From S01, the state transfers to S17. S02 and S03 transfer to either S11, S12, or S13. S04, S05 and S06 transfer to either S14, S15, or S16. The difference between S11 and S14 is that the former retains the first string in the U-memory and second string in the L-memory, and the latter retains the first string and third string in the U-memory and the second string in the L-memory. Both S11 and S14 operate so that both tuples are compared and one of them is output. In these states, if one of these streams is completely output, the state transfers to either S12, S13, S15 or S16. In these states, some tuples of either the first string or second string retain. If these tuples are output, the state transfers to End.state from S12 or S13, or to S17 from S15 or S16. S17 operates so that the remaining third string is output and transfers to the End-state.

In the passing mode, the sorting cell merely outputs the tuple from the preceding cell to the next cell via an INR, an LR, and an OUTR. The transfer time from the preceding cell to the next cell is three cycles, that is, 660ns, corresponding to the sorting mode of the word processing loop.

6. Design and implementation of the merger

An RDBE can perform the relational database operation by using the sorter and the merger. The basic operation in the merger is to merge the two sorted streams. In the merger, various relational database operations can be performed by the control whether the tuple is output or not and whether the next tuple is the current tuple, the following tuple, or the top tuple of the duplicated tuples, etc. The merger-commands shown in Fig.8 are prepared for the purpose of these operations. These commands are classified into the five types of operations. These commands are executed on the basis of the hardware pipeline algorithm. We will describe these algorithms for several database operations in detail.

6.1. The hardware pipeline algorithm

6.1.1. Sort-External operation

The sort-external operation is used when the merger sorts a long stream in excess of the capacity of the sorter. In the sort-external algorithm, a stream S is first chopped into n substreams S1,S2,.., and Sn within the capacity of the sorter and these substreams are merged repeatedly by the merger.

We will describe the basic process of this algorithm as follows.

Fig.9 represents this process which sorts a stream S. This stream consists of 20,000 tuples and the tuple length is four bytes. In Fig.9(b) and (c), this stream S is first chopped into five substreams S1,.., and S5. Each substream is sorted by the sorter, and in consecutive sequences, every two substreams S1 and S2, S3 and S4 are merged by the merger. And every two substreams is output as a single new substream S12 and S34. For the next step, the two new substreams S12 and S34 are merged as shown in Fig.9(d). Finally, the two new substreams S1234 and S5 are merged and as shown in Fig.9(e).

We will describe the detailed process of this algorithm. We will give an example of the merge(S12,S34) in Fig.9(d).

In Fig.9(d-1), two substreams S12 and S34 are chopped into S12A,S12B and S34A,S34B. S12A and S34A are the first half of these substreams. In the first step, S12A is stored into the U-memory by the load-U command as shown in

Fig.9(d-2). The next merger-command is the sort-external. In this command, as soon as the top tuple of S34A is input into the merger, the merging process is started as shown in Fig.9(d-3). If one of these substreams is entirely output, the merging process is stopped and an interruption to the CPU occurs as shown in Fig.9(d-4). When the CPU is interrupted by the merger, the CPU detects that S12A is empty, in this case. The next merger-command is the sort-external between S34A and S12B as shown in Fig.9(d-5). Similar operations are repeated until either S12 or S34 is completely exhausted as shown in Fig.9(d-6). Finally, the merger outputs the remaining tuples of the substream S12B by the pass-1 command as shown in Fig.9(d-7) and the merge(S12,S34) is finished.

6.1.2. Unique-External operation

A unique external operation merges two sorted streams except the duplicated tuples from a stream S. If a stream can be sorted by the sorter in one scan, a unique-internal merger-command is used. Otherwise, this operation is used. At first, the stream S is chopped into several substreams and sorted by the sorter, like the sort-external operation. Then, this operation is repeatedly used to make a new substream except the duplicated tuples.

6.1.3. Join operation

The type join consists of nine commands. The most important join operations is the join-equal. So, we will describe the algorithm of the join-equal operation as follows.

Fig.10(a) shows two input streams S1 and S2 in ascending order. The joined stream is shown in Fig.10(b). This algorithm uses the DP, which indicates whether the tuple is duplicated or not. Stream S1 including the DP is written into the U-memory using the load-U operation. Then, stream S2 is sorted in ascending order by the sorter. As soon as the top tuple of this stream is entered into the merger, the join equal operation is begun according to the hardware pipeline algorithm. This operation is shown in Fig.10(c) and (d). In

Fig.10(c), three dotted lines indicate that two tuples are not output and four solid lines indicate that they are output in each operation step. Fig.10(d) describes the join process of each operation step. Fig.10(e) describes the contents of the U- and L-memory address 0-7 and the contents of the RUAR, RLAR, BUR, and BLR register, at step 4. The functions of these registers are similar to the RUAR, RLAR, BUR, BLR of the sorter. The BUR and BLR are necessary when a tuple read out at the previous step is read out again. If the compared tuples are equal and both of them are duplicated, the BLR register keeps the top address of the duplicated tuples. Therefore this operation time between M tuples and N tuples is not of the order [M*N] but of the order [M+N].

When both streams SA and SB are over the capacity of the sorter, these streams are chopped into some substreams SA1,..,SAM and SB1,..,SBN within the capacity. Firstly, all the substreams SA1,..,SAM are sorted by the sorter. Next, substream SB1 is sorted and written into the U-memory. Then the join operation is applied to each substream SA1,..,SAM. This operation is repeated for all the substreams SB2,..,SBN.

6.1.4. Restrict-range operation

The restrict operation selects tuples which satisfy a certain condition. We will describe the restrict-range operation.

The restrict-range operation selects tuples between the lower limit U(i-1) and the higher limit U(i) inclusively. If U(i-1) is equal to U(i), the tuple whose value is U(i) is output. When U(1) is minimum and U(2) is maximum, all the tuples of stream S2 are output. Fig.11 describes an example of the output range of this operation.

6.1.5. Compare operation

A compare-operation compares two key field value of each tuple and output the tuples which satisfy the designated condition.

6.2 Control ROM

The relational database operations are performed by executing several merger-commands. These merger-commands are based on the hardware pipeline algorithm. Each algorithm is implemented by a control ROM. The input signals to the control ROM consist of five merger-command code bits by which the 29 merger-commands are described, two comparison result bits, and etc..

The output signals from the control ROM consist of tuple selection bits which select the next tuple at the next step, output control bits, stop operation bits, and etc..

6.3 Design details

The merger is based on the merging process. The merger is similar to the sorter in some respects. The block diagram of the merger is shown in Fig.12. The merger consists of an Operation Section and an Output Control Section. The operation section consists of an INR, two memories (U-memory and L-memory) with an address control circuit, a UR, an LR, a control ROM, and a comparator. This section performs the following steps:

(1) Writing two sorted streams from the sorter into these memories.

(2) Reading both tuples from these memories at the same time, providing them to the comparator and writing them into two tuple memories in the next section.

(3) Comparing the keys of each tuple and detecting output tuples satisfying the condition of the command.

The output control section consists of two tuple memories, two field ordering circuits, two field section circuits, two data type transformation circuits, a selector and an OUTR register.

This section performs the following functions under software control.

(1) Reordering the fields of an output tuple.

(2) Selecting fields of an output tuple.

(3) Undoing the transformation of the key field.

Each block of the operation section is described as follows.

The INR is an input register from the sorter. The content of the INR is written into the U- or L-memory. The U- or L-memory is a 64Kbyte RAM each. Each stream is written into the designated memory. One word of the U-tuple from the U-memory and one word of the L-tuple are read out and set in the UR and the LR at the same time. These tuples read out are controlled by the address controller circuit. The control ROM is described in the previous section. A comparator compares the contents of the UR and the LR. The control circuit generates several control signals which are based on the control ROM and etc.. The U- or L-tuple memory is used as the buffer memory because the comparison result is determined by the last word of the tuple at worst. These field reordering circuits rotate each key field from the top position to the original position. This function is shown in Fig.3. These data type transformation circuits transform the data types to the original data type. These field section circuits take out from the designated fields the head part, the center part, the tail part, both head and tail parts, nothing, and all as shown in Fig.13.

7. Performance estimation

7.1. Sort operation

If a stream is under the capacity of the sorter, the sorter works and the merger only passes the result. The execution time consists of an input time from the HM to the sorter, an interval time from the moment that the last tuple is entered in the sorter to the moment that the first tuple is output to the merger, and an output time from the sorter to the HM through the merger. The input time is about 22ms(=64KB/3MB) when the number of tuples in this stream is 4096 and the length of tuples is 16 bytes. The output time is the same as the input time. The 0.008ms interval time can be ignored because of minimum transferable time from the first cell to the last cell. So, the sorting time is 44ms called T. If a stream is over the capacity of the sorter, some sort-external command etc, are executed. The execution time of this operation is as follows if the interrupt processing time when either substream becomes empty, and the CPU time from one finished command to the next command are ignored. We suppose that the number of tuples in this stream is 4096*(2**N), that is, the number of substream in this stream is (2**N), and the length of tuples is 16 bytes. Then the sorting time is [3*N*(2**(N-2))+2**(N-1)]*T. For example, the operation time for 16,384 tuples is about 0.35 sec. The unique-external operation time is the same as the sorting operation time.

7.2 Join-Equal operation

In the join-equal operation, if two streams are under the capacity of the sorter, this operation time is the same as the sort-internal operation time. If the number of tuples in both streams is 4096 and the length of tuples in these streams is 16 bytes, and equal tuples are few, then this operation time is T=44ms except the time for loading one of them by the load-U command. If either stream is over this capacity, this operation consists of some sorting commands and some join-equal commands, and is described in detail in the previous section. We suppose that the number of tuples in stream S1 is 4096*M, the number in stream S2 is 4096*N, both tuple lengths are 16 bytes and output tuples are few. This operation time is (M*N+M+N)*T. For example, the operation time between two strings of 16,384 tuples is about 1.1 sec. The restrict operation time is the same as the join-equal operation.

8. Conclusion

We have described an RDBE which performs some relational database operations using special processors, the sorter and merger. Our RDBE is characterized as follows.

(1) High speed operation by using a sorter by adoption of a straight two way merge-sort algorithm and using a merger by adoption of a hardware pipeline algorithm, synchronized with the data transfer rate (3MB/sec).

(2) Many kinds of data types and a stream including null typles are supported.

(3) Relational database operation capability by using the merger with this algorithm.

The RDBE was developed using currently available technology. The developing time was about two years, including the development of a gate array. The RDBE is now incorporated in the Delta database machine and is working.

Acknowledgments

We would like to express our sincere appreciation to the members of the ICOT KBM(Knowledge Base Machine) group for their valuable discussions, and to Dr. K.Mori, manager of the Information Systems Laboratory, who provided the opportunity to conduct the present research. We would like to thank the TOSHIBA development group, in particular A.Miyoshi, K.Oda and T.Oka, for their cooperation in designing and implementing the RDBE.

References

[1] S.Todd, "Algorithm and Hardware for a Merge Sort Using Multiple Processors," IBM Journal of Res. and Develop., 22, 1978.

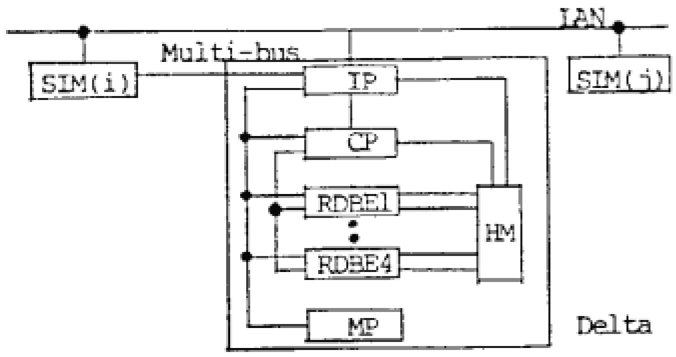
[2] S.Shibayama, T.Kakuta, N.Miyazaki, H.Yokota, K.Murakami, "Query Processing Flow on RDBM Delta's Functionally-Distributed Architecture," Proc. 2nd FGCS Conference, pp.427-435, Nov., 1984.

[3] H.Sakai, H.Iwata, K.Kamiya, S.Abe, A.Tanaka, S.Shibayama, K.Murakami, "Design and Implementation of the Relational Database Engine," Proc. 2nd FGCS Conference, pp.419-426, Nov., 1984.

[4] E.F.Codd, "A Relational Model for Large Shared Data Banks," Commun. ACM, 13, 377, June, 1970.

[5] H.Gallaire and J.Minker (eds.), Logic and Data Bases, Pleum Press, 1978.

[6] K.Taki, M.Yokota, A.Yamamoto, H.Nishikawa, S.Uchida, H.Nakashima, A.Mitsuishi, "Hardware Design and implementation of the Personal Sequential Inference Machine(PSI)," Proc. 2nd FGCS Conference, pp.398-409, Nov., 1984.



SIM	: Sequential Inference Machine
IP	: Interface Processor
CP	: Control Processor
RDBE	: Relational Database Engine
MP	: Maintenance Processor
HM	: Hierarchical Memory
LAN	: Local Area Network

Fig.1 Delta architecture

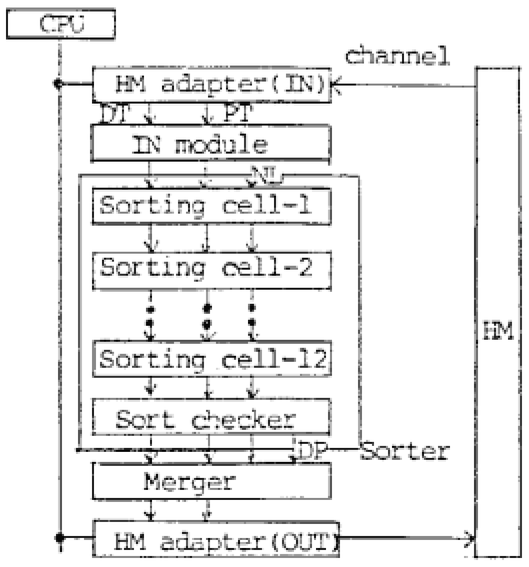


Fig.2 RDBE configuration

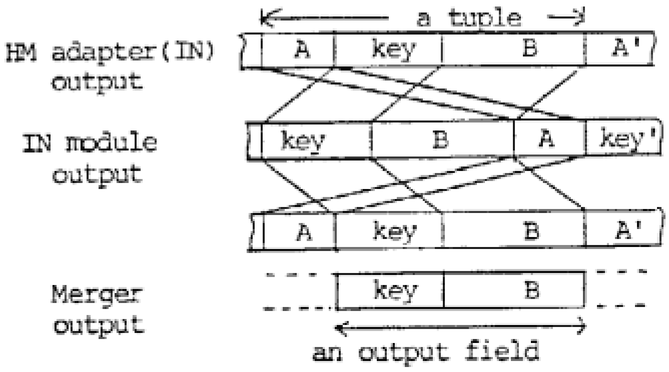


Fig.3 Function of field ordering, field reordering and field selection

[Input] 5B,1E,4A,8B,3D,2C,7F,6B

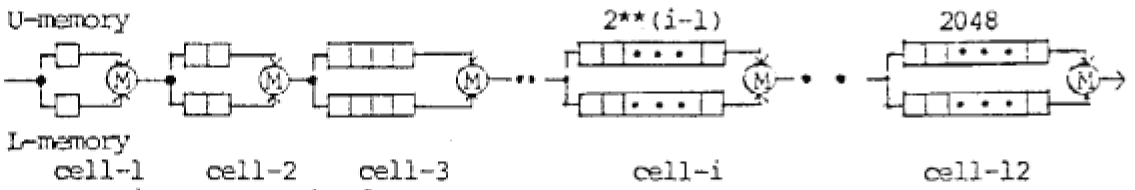
cell-1 output (1E,5B)(4A,8B)(2C,3D)(6B,7F)

cell-2 output (1E,4A,5B,8B)(2C,3D,6B,7F)

cell-3 output (1E,2C,3D,4A,5B,6B,7F,8B)

a numerical : a key field
a letter : satellite field

Fig.4 An example of sorting eight tuples by the two-way merge-sort algorithm



M : Merging process hardware

Fig.5 Outline of the sorter

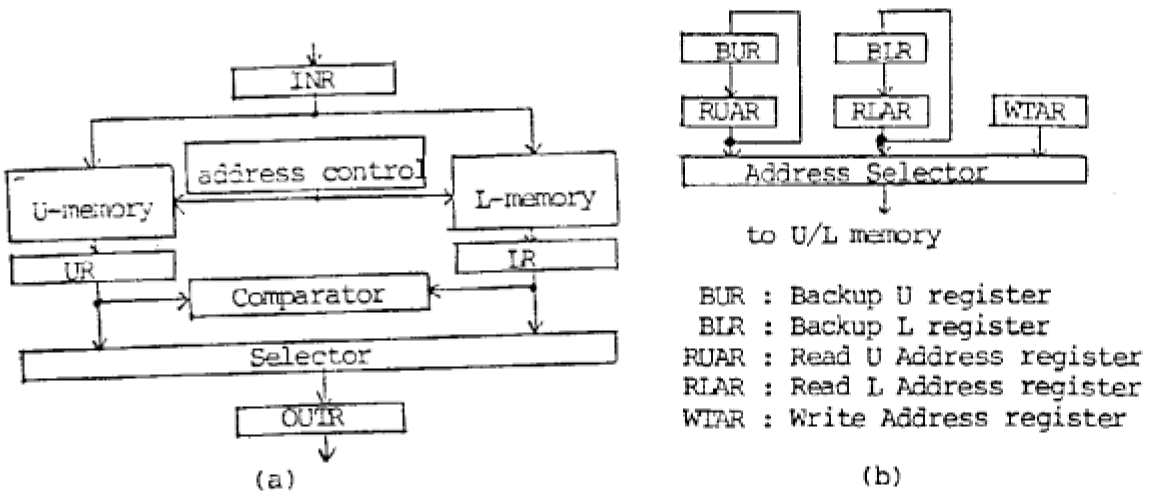


Fig.6 A sorting cell configuration

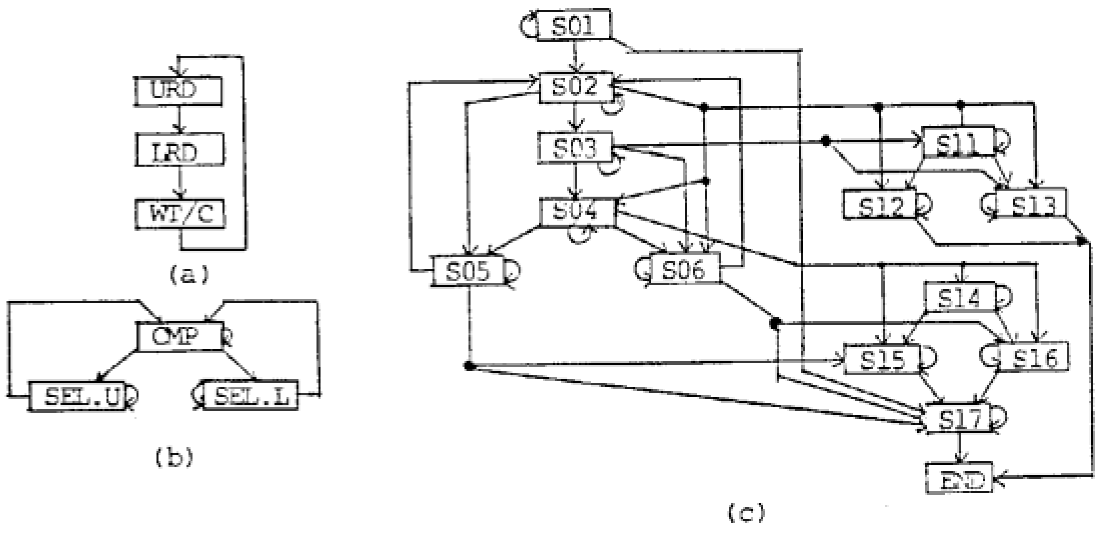


Fig.7 State transition diagram of each processing loop

PASS COMMAND	RESTRICT COMMAND	COMPARE COMMAND	JOIN COMMAND
LOAD	REST-NULL	COMP-ALL	JOIN-ALL
PASS-1(NOP)	REST-NONULL	COMP-NULL	JOIN-NULL
PASS-2(UNQ-IN)	REST-EQ	COMP-NONULL	JOIN-NONULL
SORT COMMAND	REST-NE	COMP-EQ	JOIN-EQ
SORT-IN	REST-RANGE	COMP-NE	JOIN-NE
SORT-EX		COMP-LT	JOIN-LT
UNIQ-EX		COMP-GT	JOIN-GT
		COMP-LE	JOIN-LE
		COMP-GE	JOIN-GE

NOP : No operation UNQ-IN : Unique-Internal EX : External NONULL : Not null
EQ : Equal

Fig.8 List of the merger command

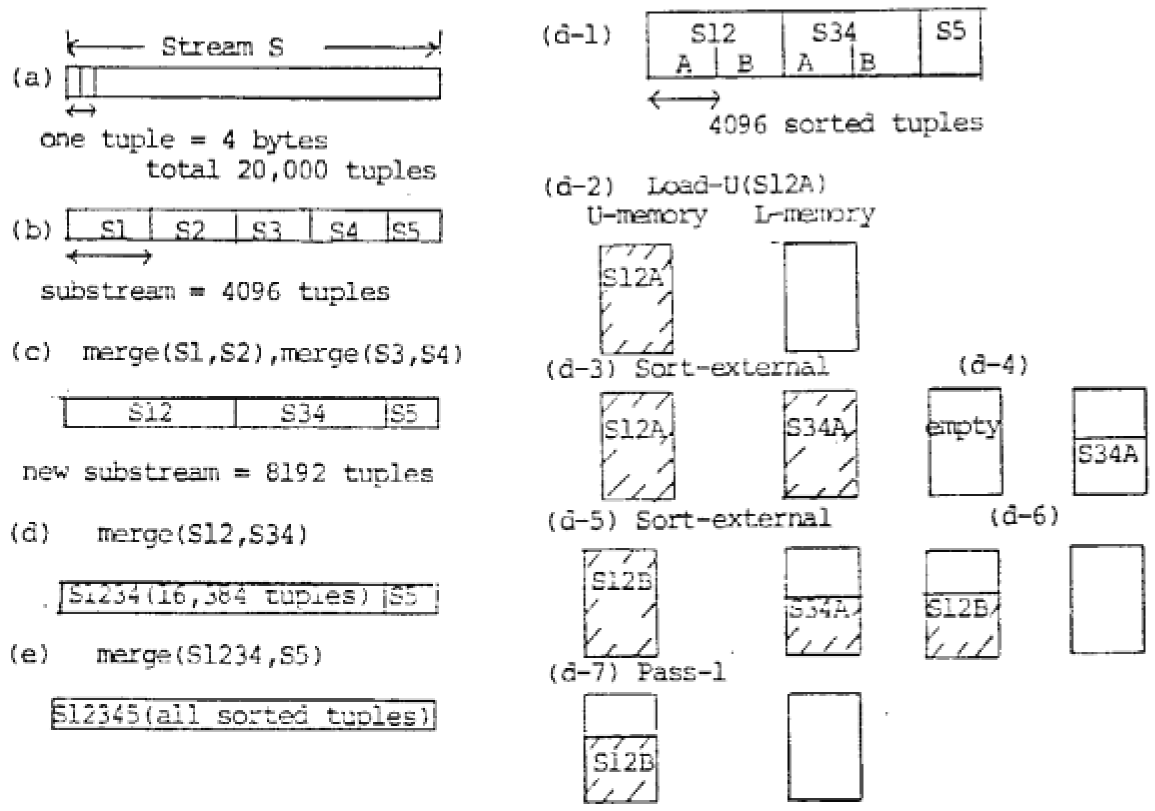
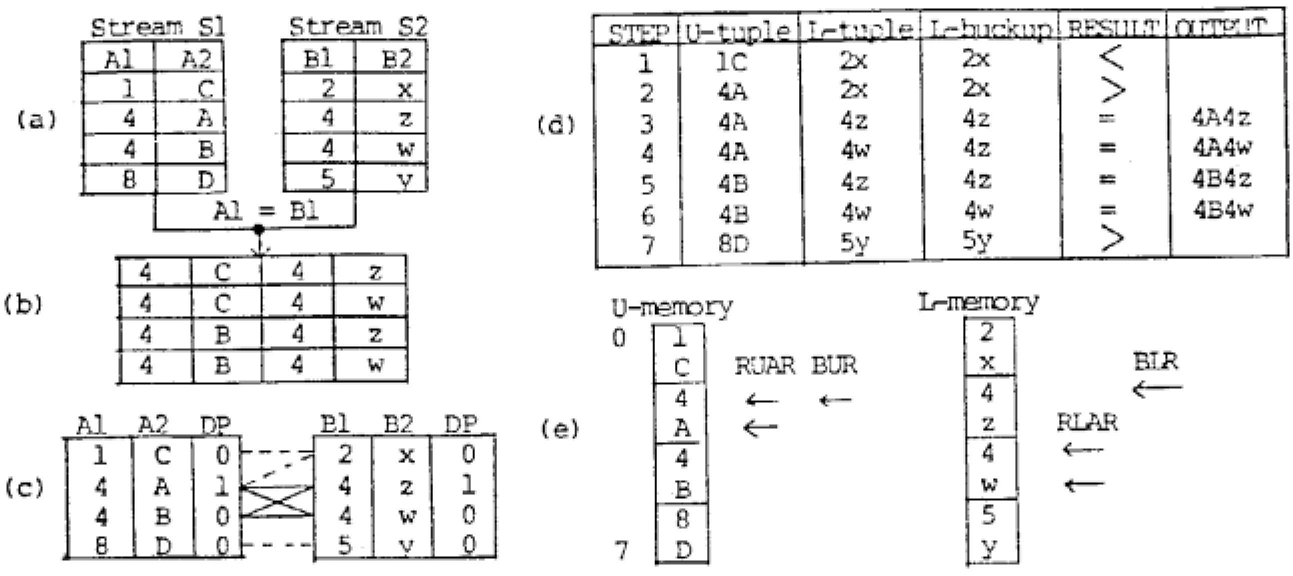


Fig.9 An example of the sort-external command



STEP	U-tuple	L-tuple	L-buckup	RESULT	OUTPUT
1	1C	2x	2x	<	
2	4A	2x	2x	>	
3	4A	4z	4z	=	4A4z
4	4A	4w	4z	=	4A4w
5	4B	4z	4z	=	4B4z
6	4B	4w	4w	=	4B4w
7	8D	5y	5y	>	

Fig.10 An example of the join-equal command

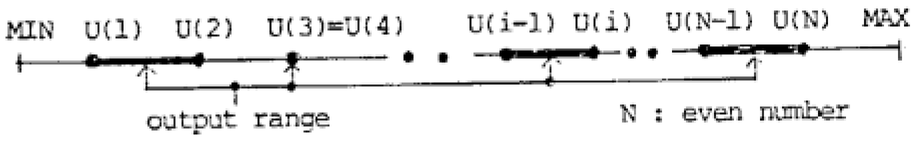


Fig.11 An example of the output range of the restrict-range merger-command

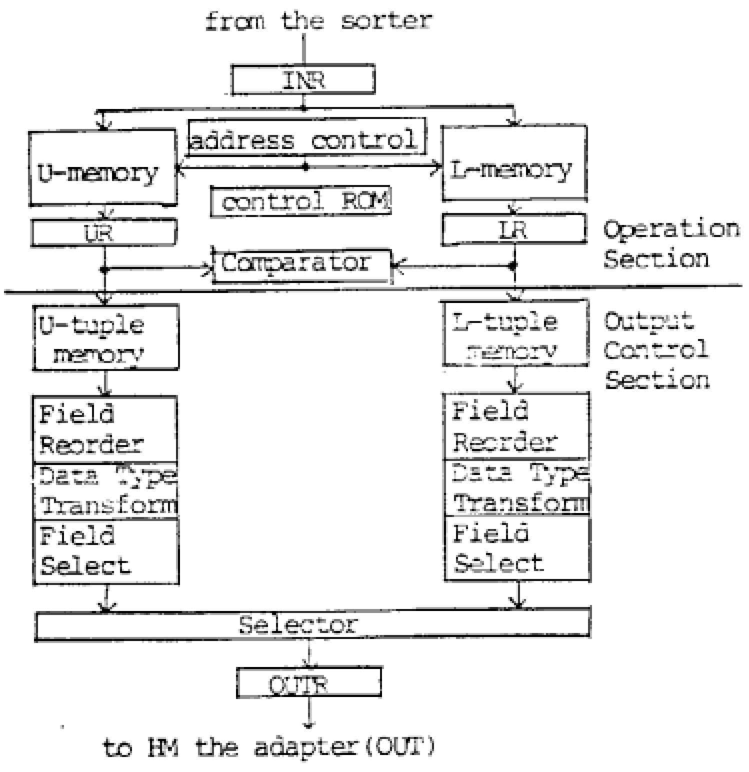


Fig.12 Block diagram of the merger

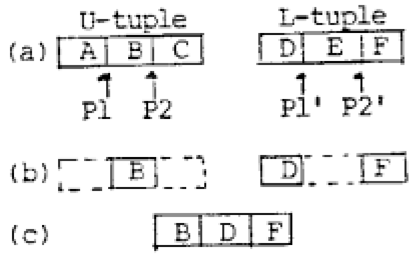


Fig.13 Function of the field selection