Invited Paper

A String Search Processor LSI

KOUSUKE TAKAHASHI*, HACHIRO YAMADA* and MASAKI HIRATA*

This paper describes a new string search processor hardware architecture, LSI chip, and application. The architecture uses the programmable sequential logic circuit (PSLC), consisting of content Addressable Memory (CAM) and Sequential Logic Circuit (SLC) parts. The design of the SLC was based on the FSA state transition diagram. The resultant string search processor LSI chip can store 64 variable-length pattern strings, search for text data at 10 Mch/sec, and provide flexible matching functions such as non-anchor matching, don't-care matching, wild-card matching, and approximate matching. It has proved functionally effective in an experimental full-text DB search system.

1. Introduction

Recent advances in word processing have created enormous stores of text data from which, because of the sheer volume of data involved, desired information can be retrieved only with difficulty. When such secondary data as keywords or content summaries are available, the search time for the text data contents may be relatively short, but the time required for preparation of the secondary data may be prohibitively long [1].

In most actual office environments, where the volume of text data produced is very large, such secondary data preparation cannot be expected to be available. The desired information must be found directly by searching text data. Many type-string search processors have been studied over a long period to accelerate text data searching [2, 3]. However, few practical processor LSI chips allow both fast and flexible string matching, because it is difficult to design processors that have the complex string-matching functions required to search for actual redundant text data.

The authors have developed a new string search hardware architecture for faster and more flexible string matching, and have also developed a practical string search processor LSI chip, using CMOS device technologies. The LSI chips were applied to an experimental text DB system for the search function check, in which methods were studied for applying them effectively to achieve faster full text DB search systems.

2. Requirements for the String Search Processor

2.1 Overall Concept of the Text DB Search System

Figure 1 shows the concept behind the full-text DB search system, in which the string search processor (SSP) is used to search the text DB memory (TDM) contents. In this system, the SSP outputs during the full text search are stored in the search result memory (SRM) before being transferred out. Such query conditions as present and past keywords, as well as Boolean logic operations, are stored in the query condition memory (QCM). The keyword part is set to the SSP and the logic operation part is set to the query resolve processor (QRP).

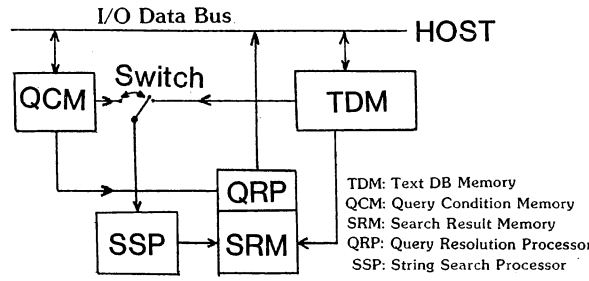

Fig. 1 Concept of the full-text DB search system.

Since a list of past query conditions in the QCM can be displayed on the CRT screen, synonymous keywords may be selected from the past query condition keywords, to be added to the present query conditions. The input text data can be searched by the SSP to store many synonymous keywords. The QRP is used to indicate the matched text record numbers by applying AND or OR operations to the SRM output.

2.2 String-Matching Functions

The SSP in such a text DB search system must have a large memory capacity for very long string matching and parallel short string matching, because it must store many kinds of synonymous keywords. Furthermore since text data may show a wide variation in formatting and may have spelling ambiguities, the SSP must have

*NEC corporation, 4-1-1, Miyazaki, Miyamae-ku, Kawasaki-city, 213 Japan.

Table 1 Basic string-matching functions.

Matching Mode	String Pattern	Text Data /Matched String
Anchor	DEFG	ABC DEFG HDEFGHI
Non-anchor	DEFG	ABCDEFGHDEFGHIJK
FLDC	DE?G	ABCDEFGABCDEXFG
VLDC	DE-FG	ABCDEFCGDEABCFGH
Wild-card	DEFG	ABCDE?GHABD?EFGH
Approximate	DEFG	ABCDEGDECFGHDEAG

such flexible string matching functions as anchor, nonanchor, fixed length don't care (FLDC), variable length don't care (VLDC), wild-card, and approximate matching functions [2, 3, 8], as shown in Table 1.

Figure 2 shows the basic function block diagram for the SSP, which contains several string matchers for parallel string matching between input text strings and stored pattern strings. Each string matcher compares the strings in the specified matching mode. The encoder is used to output the pattern string class code by detecting which string matcher generates the string match signal. This signal is output through the OR gate from either of the string matchers.

3. Conventional String Matching Hardwares

Typical string matching methods in current string search hardware architectures are categorized into four methods: (Parallel Comparator) PC, (Cellular Array) CA, (Finite State Automaton) FSA and (Dynamic Programming) DP. Typical string matching circuits based on these methods are shown in Figs. 3(a), (b), (c), and (d).

The PC method employs a character comparator array or CAM (Content Addressable Memory) and an input string shift register [SR] for string comparison [1, 2, 9]. It permits parallel string matching, because the CAM can store many pattern strings. It also allows variable-length string matching, if the CAM is designed to have the capability to handle don't-care character matching. The encoder (ENC) is used to discriminate parallelly matched strings. The advantage of the method is the speed; the disadvantages are the poor string matching functions and the required hardware size. When the wild-card and approximate matching functions are required, this kind of hardware becomes prohibitively large.

The CA method employs a set of contiguously connected cells, as shown in Fig. 3(b), each of which contains character registers R1 and R2, a character comparator [COMP], and a flag-bit register [SR] to hold a flag-bit as a matching signal for a partial string [3, 10, 11, 12]. The strict string match signal is output from the last (right-side) cell flag-bit register. Multiple string matching is realized by arranging such cell arrays in parallel. Numerical-range string matching can also be accomplished by modifying the comparator [COMP],

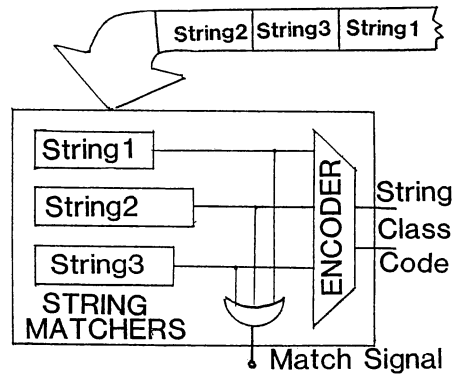

Fig. 2 Blockdiagram of the basic string search processor function.

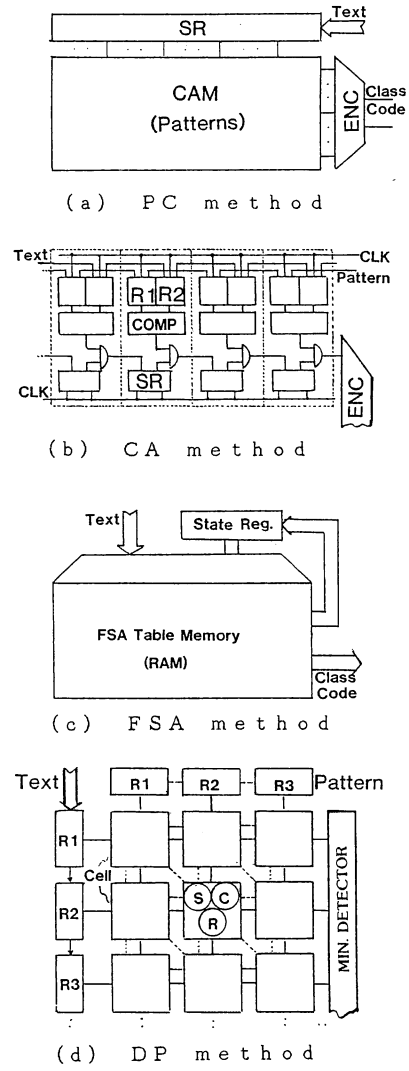

Fig. 3 Typical string-search processor configurations.

though approximate string matching is not so easily realized.

The advantage of the CA method is the speed. The disadvantage is the poor string matching functions. Though the regular circuit configuration is convenient

for VLSI design, the hardware size is not always small, since it contains discrete registers R1 and R2.

The FSA method employs the RAM to store the FSA state transition table data [13, 14]. The RAM response to each input character consists of the state code and string class code. The state code is fed back to the RAM address decoder through the state register. Complex string matching is possible, based on the FSA, though FSA transition table composition is troublesome, particularly in the nonanchor string-matching mode. The advantage of this method is the large number of string-matching functions, though this does not permit approximate string matching. The disadvantage is the amount of hardware required for parallel string matching.

The DP method employs a two-dimensional cell array [8, 15, 16]. Each cell consists of a character comparator [C], a string distance register [R], and a distance minimum selector [S], as shown in Fig. 3(d). Pattern string characters in registers R1, R2, and R3 are simultaneously compared with the text string characters in registers R1′, R2′, and R3′ over all cells. The character comparison result in each cell is used to calculate the shortest string distance together with other string distances coming from neighbouring cells, based on the DP algorithm. Therefore, though approximate string matching can be accomplished, the array hardware becomes very complicated, especially in the wild-card and don't-care matching modes. The advantage of the method is the rich matching functions, while the disadvantage is the complex hardware design.

As described above, PC and CA methods are convenient for realizing a simple SSP by means of LSI chips, and the FSA method for realizing an SSP that uses RAMs available on the market. The DP method is useful for realizing approximate string matching functions, though it is difficult to design.

4. A New String Search Hardware Architecture

4.1 Programmable Sequential Logic (PSL) Method

The design concept used in the proposed string search processor divides the string comparison process into character comparison and sequence comparison processes, so that the processor may consist of (Content Addressable Memory) CAM or Random Access Memory (RAM) character comparison, and a sequential logic circuit (SLC) for sequence comparison [4]. The design of the SLC was based on the flag-bit shift control on the shift register. Figure 4 shows an SLC used to detect the bit pattern string "1101". The D-flip registers R0, R1, R2, R3, and R4 are connected serially by AND gates, and are used as shift registers for flag-bit signals.

Let the i-th AND gate input signal at time t be expressed by $Yi(t)$, and the content of the register R_i, $i=0, 1, 2, 3, 4$ be expressed by $Si(t)$, The SLC then operates as follows:

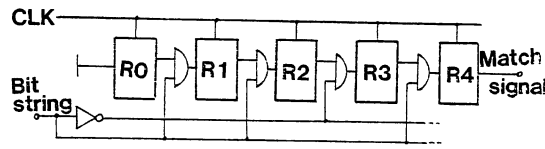

Fig. 4 Sequential logic circuits for detecting "1101".

Fig. 5 FSA state transition diagram.

$$S_0(t-1)=1,$$
$$S_1(t)=Y_1(t)*S_0(t-1),\ S_2(t)=Y_2(t)*S_1(t-1),$$
$$S_3(t)=Y_3(t)*S_2(t-1),\ S_4(t)=Y_4(t)*S_2(t-1) \quad (1)$$

where the symbol* means a logical product operation. Therefore, $S_4(t)$ becomes 1, only if $Y_4(t)=Y_3(t-1)=Y_2(t-2)=Y_1(t-3)=1$ or if the bit string "1101" is applied clock by clock. This $S_4(t)$ is used as a match signal for the bit string "1101".

This means that the SLC detects the bit pattern "1101" by checking whether or not a flag-bit "1" on R_0 can reach R_4 by applying the bit string. This flag-bit shift on the flag bit registers, connected by AND gates, is coincident with the state transition on the FSA state transition diagram to allow the specified bit pattern "1101" to be accepted, as shown in Fig. 5, because the state on the node S_0 can be transferred to the last node S_4 only when the string "1101" has been applied clock by clock. On this analogy, it is easy to realize a complex bit-string-matching FSA by using the registers and AND gates.

The fixed bit pattern "1101", shown in Fig. 4, can be programmably changed by using a RAM or CAM, instead of a wire connection. Additionally, the bit pattern can be converted to a character code pattern by widening the RAM address code or expanding the CAM word width. Figures 6(a) and (b) are examples of an SLC using a RAM and CAM, respectively. The RAM and CAM can store the character string $X_1X_2X_3X_4$. The output from the RAM or CAM can made to correspond to the bit string $Y_1Y_2Y_3Y_4$.

Let the input character code be $X(t)$ at time t:

$$Yi(t)=\text{"1" if } X(t) \text{ is equal to } Xi;$$
$$\text{other wise, } Yi(t)=\text{"0"}. \quad (2)$$

If the RAM or CAM for storing the character codes X_1, X_2, X_3, X_4 outputs bit signals Y_1, Y_2, Y_3, Y_4 for the SLC, the SLC can detect the character string instead of the bit string. Thus, Figures 6(a) and (b) are called programmable sequential logic (PSL) circuits.

In each PSL circuit, fast string matching can be achieved when character match signals output from the memory part are fed in parallel to the SL part. Though this PSL circuit is similar to the DB filter [15], it is

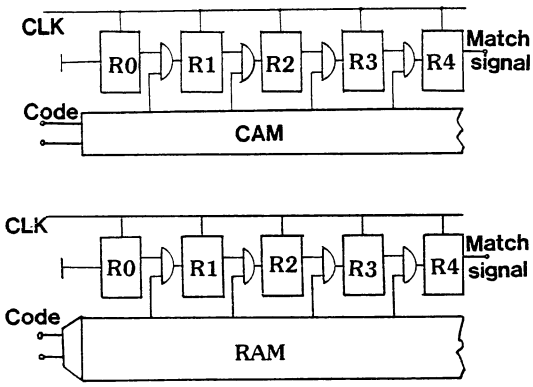

Fig. 6 Programmable Sequential Logic (PSL) circuits using CAM (a) and RAM (b).

different from a DB processor as regards the flexible correspondence to an FSA. An the analogy with the FSA, PSL circuits make it easy to design string search processors with approximate string-matching functions.

4.2 Approximate and Strict String Matching Principles

Figure 7 shows a PSL circuit for both approximate and strict string matching. The design of this PSL circuit is based on the FSA shown in Fig. 8, and allows it to accept strings with such character errors as insertion, omission or substitution, and no character error. The CAM storing the character pattern string X_1, X_2, X_3, X_4 outputs the match signal Y_1, Y_2, Y_3, or Y_4, becoming "1" when the input character code $X(t)$ is equal to X_1, X_2, X_3, or X_4.

In the state transition diagram, shown in Fig. 8, the state is initially put on node S_{10}. If the matching signal Yi is "1", the state on node S_{1_i} is sent to node $S_{1_{i+1}}$ for strict matching. The state on $S_{1_{i-1}}$ is sent to node $S_{2_{i+1}}$ for an omission error. If Yi is "0", the state on node S_{1_i} is sent to nodes S_{2_i} and $S_{2_{i+1}}$ for insertion and substitution errors, respectively. In general, the state on the j-th row nodes is expressed as follows:

$$S_{ji}(t) = Y_i(t) * \{S_{ji-1}(t-1) + S_{j-1i-2}(t-1)\} + \overline{Yi}(t) * \{S_{j-1i}(t-1) + S_{j-1i-1}(t-1), \quad (3)$$

where $i=1, 2, 3, 4$ and $j=1, 2$. Then the $S_{ji}(t)$ is the state on node S_{ji} at time t. The symbol + means a logical OR operation.

Thus, if state $S_{14}(t)$ is "1", the input character string is judged to be exactly the string $X_1X_2X_3X_4$. If the state $S_{24}(t)$ is "1", the input character string is judged to be any of the one-error strings, such as $X_1X_3X_4$, $X_1X_2X_5X_3X_4$, or $X_1X_2X_5X_4$. If $S_{14}(t)$ and $S_{24}(t)$ are not "1", the input character string is judged to be neither a strict nor an approximately matched string. These state transition processes can be realized on register arrays in the PSL circuit shown in Fig. 7.

That is, the flag-bit registers and AND gates in the SLC part in Fig. 7 are arranged according to the state nodes and state transition paths in the FSA shown in Fig. 8. Therefore, the flag-bit registers R_{s_0}, R_{s_1}, R_{s_2}, R_{s_3}, and R_{s_4} for the SLC part are used for strict string matching. Registers R_{a_1}, R_{a_2}, R_{a_3} and R_{a_4} are used for approximate string matching.

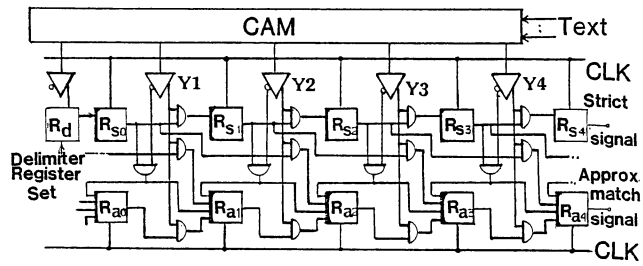

Fig. 7 PSLC for approximate string matching.

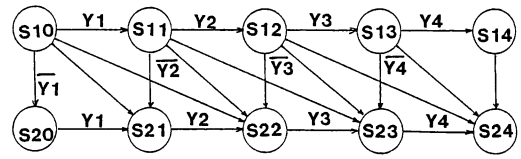

Fig. 8 FSA state transition diagram for accepting strings with single-character errors.

The flag-bit signal S_{10} on the 1st R_{s_0} is always "1". Since $Y_1(t)$ becomes "1" when $X(t)$ equals the character code X_1 stored in the CAM, the content of register R_{s_1} becomes "1". Next, if $Y_2(t)$ is "1", then registers R_{s_2} and R_{a_2} become "1". If $Y_2(t)$ is "0", the contents of both R_{a_2} and R_{s_2} become "1". When $Y_1(t)$, $Y_2(t+1)$, $Y_3(t+2)$, $Y_4(t+3)$ become "1" clock by clock, then the content of register R_{s_4} becomes "1". If there is one omission, insertion, or substitution error in each input character string, the register R_{a4} content becomes "1". This approximate string matching process can be easily expanded to allow more than two errors, by adding one or more arrays of flag-bit registers to the circuit in Fig. 7.

5. String Search Processor LSI Chip

5.1 LSI Circuit Configuration

The proposed string search processor, based on the PSL hardware architecture, has been designed as shown in Fig. 9. An LSI chip has been realized by using CMOS device technology [5, 6]. This circuit consists of a sequential logic circuit (SLC) part and a CAM part which are divided into 64 segments so that can accept both variable-length long and short strings.

The CAM part was designed to realize a large memory capacity by using eight blocks of the 2-bit SRAM. Individual bit lines for the SRAM were connected with wired AND lines, to make a match signal for 16-bit character code [4, 5, 6]. As a result, an 8-Kbit CAM part was realized by 16-Kbit SRAM cells. Using two SRAM cells per CAM cell, this CAM part stores

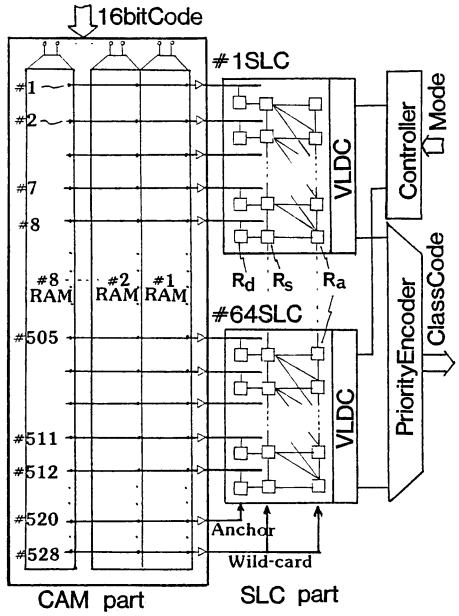

Fig. 9 SSP LSI circuit configuration.

and detects don't-care characters. A write-in operation for storing the character strings and delimiter codes is achieved by giving the character and delimiter codes together with address codes in the write mode. The erase operations is carried out by resetting all SRAM cells before the write-in operation.

The SLC part is designed to be a cascade connection of 64 SLC segments. Each SLC segment consists of 16 flag-bit registers R_s and R_a, 32 or more AND gates, and 8 delimiter registers R_d, to detect a variable-length string shorter than eight characters, since most English language words are shorter than eight characters. If the last delimiter register R_d in a segment holds "1", the flag bit for the last register R_s or R_a in the segment is used as a strict or an approximate string match signal, and is then sent to the encoder. If the register R_d holds "1" on another position of a segment, then the flag-bit register R_s on the corresponding position is set at "1" as a flag-bit signal.

The delimiter signals for R_d are supplied from the CAM when the delimiter character is input after each character string. In the anchor matching mode, the R_d contents are set to the register R_s only when the anchor code character is applied to the CAM part. In the nonanchor mode, this signal is always set to "1". To detect variable-length string longer than eight characters, the last delimiter register R_d is not "1", except on the last segment of serially connected SLCs.

As explained above, the PSL-method string search processor has such versatile string matching functions as variable-length anchor or non-anchor string matching, don't care string matching, and approximate string matching.

Additionally, the variable-length don't-care (VLDC) circuit was inserted between the SLC and the encoder.

Table 2 Hardware architecture comparison.

Matching Method	Hardware Volume	Parallelism	Fiexibility
PC	○	○	×
CA	△	△	△
FSA	×	△	○
DP	×	×	○
PSL	○	○	○

This circuit holds the string match signal for the SLC to be used as an anchor flag-bit signal for the next SLC. Thus, this processor can detect compound strings containing several strings in sequence by using serially connected VLDCs. Furthermore, the SLC circuit has hidden OR gates between the CAM part and the SLC part. These OR gates give "1" instead of the output signal Y_i from the CAM when the wild-card character is given. In this way, wild-card string matching is accomplished.

On the other hand, the CAM part always compares the input character code with all the stored characters. All character match signals are simultaneously sent to the SLC part. Therefore, fast parallel string matching can be accomplished. Table 2 shows a comparison between these conventional methods and the proposed PSL method. The PSL method is judged to be better from many view points, since the hardware is regularly configured to mate it convenient for higher-density VLSI circuit design, even if complex string matching functions are requested.

5.2 LSI Chip Characteristics

The SSP LSI chip has been fabricated by using double-metal-layer 1.6-μm CMOS technology. It consists of an 8-Kbit CAM part and a 20-K gates logic part. It was mounted on a 72-pin PGA package, as shown in Fig. 10. Its characteristics are summarized in Table 3. This LSI chip has proved successful when operated at 10 MHz. That is, 64 pattern strings can be stored at 10 Mch/sec and can be parallelly compared with input strings contiguously transferred in at 10 Mch/sec. This means that the CAM part performs about 5 billion character comparisons per sec.

5.3 Function Test in the Full Text Search System

The experimental full-text DB search system, which was developed to evaluate the matching functions for the SSP LSI chip, was realized by attaching a string search board containing two SSP LSI chips to the personal computer PC-98. In this system, all the string matching functions shown in Table 1 were accomplished by each SSP LSI chip. Though the text data transfer rate was restricted to less than 1 MB/sec, the SSP operation time was very short.

Table 3 SSP LSI chip characteristics.

Operation Modes	Strict/Approximate, Anchor/Non-anchor FLDC/VLDC, Wild-card/Priority Encode
Storage Capacity	Max. 512 characters Max. 64 strings (8 char./string)
Pattern String Length	Average < 8 characters Max. 512 characters
Search speed	Max. 10 M char./sec; 5 G char.comparison/sec
Power Consump.	800 mW (10 MHz)
LSI chip	217,600 transistors/8.62 × 12.76 mm
Device process	1.6-um CMOS double-metal-layer technology

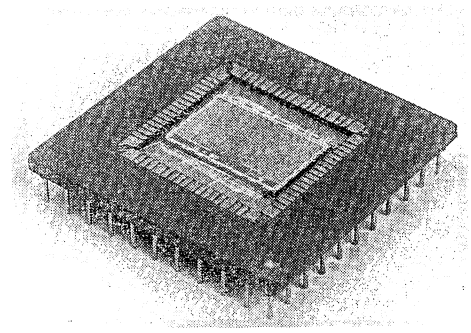

Fig. 10 A photograph of the SSP LSI package.

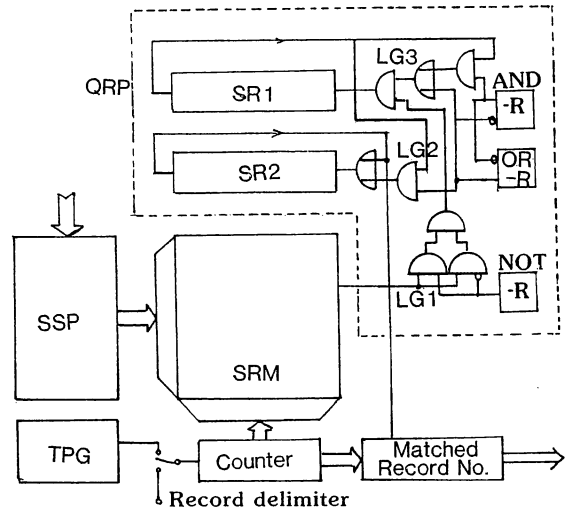

Fig. 11 A query resolution processor for the SSP.

6. Design Considerations for Faster Text DB Search Systems

For faster full-text DB search systems, the string search processor should be used, together with (search result memory) SRM, (query condition memory) QCM, (query resolve processor) QRP, and (text database memory) TDM, as shown in Fig. 1, because the performance can be improved by fast text data transfer and the query comparison process [7].

6.1 Query Resolution Process

A normal search operation consists of setting keywords to the SSP from the QCM, searching the TDM content (text data) by means of the SSP, and storing the resultant string match signals in the SRM. The QCM stores the the keywords and Boolean logic operation for the present query. The full-text search time is minimized by increasing the rate for the text data transfer from the TDM to the SSP. The final search results are output from the QRP by re-entering the keywords into the SSP after operation codes such AND, OR, and NOT were set in the QRP.

Figure 11 shows a query resolve processor (QRP) with the SSP and the SRM. The counter for records is used to make addresses for the SRM. The SRM stores match signals, indicating which text records contain keywords (patten strings) stored in the SSP. The match signals are serially output from the SRM every time the keyword is input to the SSP, under the control of the timing pulse generator TPG.

The QRP has two shift-registers, SR1 and SR2, and three logic circuits, LG1, LG2, and LG3. The Not, OR, and AND operation codes in the query conditions are set to registers NOT-R, OR-R, and AND-R in LG1, LG2, and LG3, respectively, where NOT-R, OR-R, or AND-R is set to "1" when the NOT, OR, or AND operation is specified. The SR1 is used to load the output from the SRM and to find the logical product of the content received from the SRM and the content stored in the SR1. The SR2 is used to find the logical sum of the content transferred from the SR1 and the content stored in the SR2. The final match signals are output from the SR2. That is, the timing when "1" is output is the matched record number, which is converted to the matched record address by the outside RAM.

6.2 Query Comparison Process and Learning Effect

The query comparison process is started by the operation of the SSP to store either new or old keywords, and compares the new query keywords with the past ones, which are stored together with the past search results in the QCM.

If no keyword matching occurs in the SSP, the text data in TDM must be searched by the SSP, and the search result must be stored in the SRM. If any keyword matching occurs in the SSP, the full-text search operation can be omitted, because the corresponding text search result is output directly from the SRM or QCM. Let the hit ratio h be the probability that a matching keyword is detected by the SSP. Then, the averaged search time is approximated as (1-h) times the

full-text search time T. When an incorrect hit occurs, the keywords causing the incorrect hit are added to the SSP and QCM. Therefore, the hit ratio h is enhanced as the search operations are increased. This learning effect improves the performance.

6.3 Performance of the Full-Text Search

The full-text search speed will be restricted by the rate at which text data are transferred from the TDM to the SSP, even if the SSP can operate faster. Since the transfer rate is 1 Mch/sec in such actual TDMs as disc memories, the search time for 1 billion characters of text becomes 1000 seconds in the normal mode. Using the above query comparison process, most text search times can be reduced to the time for inputting keywords. The 1000-second text search operation takes place only when a query mismatch occurs, though it seldom does so for hit ratios higher than 0.9.

7. Conclusion

This paper described a string search processor hardware architecture that uses the PSL method. The SSP LSI chip, based on the above architecture, has been developed by using CMOS device technology. String-matching functions, such as variable-length anchor, nonanchor, FLDC, VLDC, wild-card, and approximate string-matching functions have been included. The keyword store and text search operation speed in this LSI chip was 10 Mch/sec.

The application of this SSP to the full-text DB search system was evaluated and improved. It was suggested that the query comparison process in the SSP is useful for improving the performance.

Acknowledgement

The authors thank the people concerned with the SSP in the C&C system research labs., the microelectronics research labs, and the system LSI development division for discussions and support in the development of the LSI chip.

References

1. LEE, D. and LOCHOVSKY, F. Text Retrieval Machine, Sec. 14 of Office Automation, *Springer-Verlag* (1985), 338-376.
2. FALOUTSOS, C. Access Methods for Text, Computing Surverys, **17**, 1 (March 1985), 49-74.
3. HOLLAAR, L. A. Text Retrieval Computers, *IEEE Computer* (March 1979), 40-50.
4. TAKAHASHI, K. et al. A New String Search Hardware Architecture for VLSI, *Computer Architecture News*, **14**, 2 (June 1986), 20-27.
5. YAMADA, H. et al. A High-Speed String Search Engine, *J. of Solid State Circuits*, SC-**22**, 5 (Oct. 1987), 829-834.
6. HIRATA, M. et al. Versatile Data String Search Processor, *IEEE J. of Solid State Circuits*, SC-**23**, 2 (April 1988), 329-335.
7. TAKAHASHI, K. et al. Intelligent String Search Processor to Accelerate Text Information Retrieval, Proc. of IWDM '87 (Oct. 1987), 440-453.
8. HALL, P. A. V. et al. Approximate Matching, *Computing Surveys*, **12**, 4 (Dec. 1980), 381-402.
9. BEAVEN, P. A. Interactive Data Retrieval Apparatus, USP4, 433, 392 (Feb. 1984).
10. MUKHOPADHYAY, A. Hardware Algorithms for String Processing, Proc. of ICCC '80 (1980), 508-511.
11. FOSTER, M. J. and KUNG, H. T. The Design of Special Purpose VLSI Chips, *IEEE Computer* (Jan. 1980), 26-40.
12. HALAAS, A. A Systolic VLSI Matrix for a Family of Fundamental Searching Problems, INTEGRATION, *the VLSI Journal* (Aug. 1983), 1-17.
13. ROBERTS, D. C. A Specialized Computer Architecture for Text Retrieval, 4th Workshop on Computer Architecture (Aug. 1978), 51-59.
14. HASKIN, R. Hardware for Searching Very Large Text Databases, SIGIR, **15**, 2 (March 1980), 49-56.
15. CHEN, H. D. et al. VLSI Architecture for String Matching and Pattern Matching, *Pattern Recognition*, **20**, 1 (1987), 125-141.
16. YIANILOS, P. N. A Dedicated Comparator Matches Symbol String Fast and Intelligently, Electronics/Dec. 1 (1983), 113-117.
17. PRAMANIK, S. Performance Analysis of a Database Filter Search Hardware, *IEEE Trans. Comput.* C-**35**, 12 (Dec. 1986), 1077-1082.

(Received October 5, 1989)