A High-Speed String-Search Engine

HACHIRO YAMADA, MASAKI HIRATA, HAJIME NAGAI, AND KOUSUKE TAKAHASHI

Abstract—**This paper describes a newly developed character string-search engine (SSE) for rapid text retrieval. The SSE accommodates a new string-search architecture which combines 512-stage finite-state automaton (FSA) logic with a newly developed content addressable memory (CAM) to achieve an approximate string comparison of 80 million strings per second. The CAM cell consists of four conventional static RAM (SRAM) cells and a READ/WRITE circuit. Concurrent comparison of 64 stored strings with variable length has been achieved in 50 ns for an input text stream of 10 million characters per second, permitting performance despite the presence of single character errors in the form of character codes. Furthermore, this chip allows nonanchor string search and variable-length "don't care" (VLDC) string search. The SSE chip has 217 600 transistors in an 8.62×12.76-mm die area. The technology used was a double-metal 1.6-μm n-well CMOS process.**

I. INTRODUCTION

INFORMATION bulk to be retrieved is rapidly increasing in most text-retrieval applications. Retrieval of information from an essentially unformatted, huge collection of documents is generally more complex and done at a much slower search speed than that from a formatted text database [1], [2]. Hence, special hardware algorithms to search for specific pattern strings in a given serial text have been studied and developed at various organizations [3]. These algorithms use special techniques, such as an associative memory [1], [4], cellular array [5]–[7], finite state automaton [5], [8], [9], and dynamic programming [10]–[12].

Although the main goal of these activities has been to improve the searching speed, these retrieval systems still require a relatively long time to perform sophisticated search operations, such as approximate comparison for variable-length strings in the nonanchor mode. Additionally, previous hardwares have limited search capabilities for Japanese texts, in particular, because Japanese documents do not include delimiter symbols to separate individual words from one another. A string-search engine to accelerate search speed in such sophisticated text-search operations was therefore targeted for development.

This paper describes a newly developed VLSI character string-search engine (SSE) which uses a new architecture [13], [14] that combines finite-state automaton logic with a new content addressable memory to achieve a string comparison rate as fast as 80 million strings per second. This string-search performance is several hundred times faster than any previously reported.

Manuscript received March 26, 1987; revised May 26, 1987.

H. Yamada was with the System LSI Development Division, NEC Corporation. He is now with the Microelectronics Research Laboratories, NEC Corporation, 1120 Shimokuzawa, Sagmihara 229, Japan.

M. Hirata is with the System LSI Development Division, NEC Corporation, 1120 Shimokuzawa, Sagamihara 229, Japan.

H. Nagai and K. Takahashi are with the Microelectronics Research Laboratories, NEC Corporation, 1120 Shimokuzawa, Sagamihara 229, Japan.

IEEE Log Number 8715970

II. OUTLINE OF SSE

A. SSE Conceptual Structure

Fig. 1 illustrates the functional concept of the character SSE. The SSE can store 64 variable-length character strings as pattern strings to be compared in parallel with the serial-input data string. When the input data string matches either of these pattern strings, the encoder generates both a match signal as well as match address codes which indicate pattern-string class codes. Even if the input text includes overlapped pattern strings or one character error, or lacks delimiter symbols, the following complex matching operations allow the SSE to detect the pattern strings:

1) approximate comparison;
2) nonanchor search mode;
3) variable-length string comparison; and
4) multiple variable-length "don't care" search.

Since text information is prepared by humans, it often includes misspellings, thus necessitating approximate comparison for retrieval. The approximate comparison operation in the SSE detects pattern strings in input texts which include single letter misspellings.

The nonanchor string-search operation searches texts which do not include delimiter characters between words. This operation is essential for searching Japanese texts, for unlike English texts which require spaces between words, Japanese has no such requirement. For English texts, this mode may be used to detect the subword in a composite word, such as "card" in "postcard."

Variable-length string comparison is a basic function for text retrieval, but it has not been possible to achieve with conventional content addressable memories because the data size is fixed by memory structure. In the SSE, however, a combination of finite-state automaton logic with a pair-bit CAM scheme enables this complex comparison.

Multiple variable-length "don't care" search detects lengthy or complex strings which may include several variable-length "don't care" strings and substrings in sequence.

The SSE can perform not only the above matching operations but also such general matching operations as

0018-9200/87/1000-0829$01.00

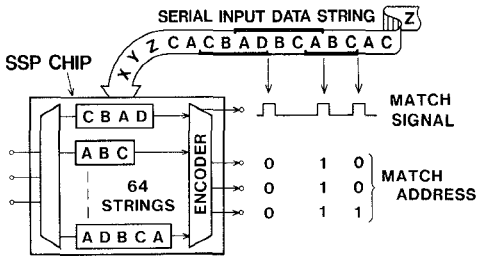


Fig. 1. Functional concept of SSE.

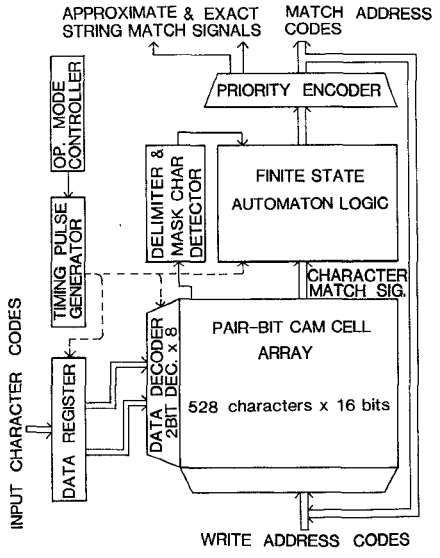


Fig. 2. SSE block diagram.

exact comparison, anchor mode, fixed-length string, and fixed-length "don't care."

Even in these complex matching operation modes, the SSE operates at a high input data rate of 10 million characters per second and a high string comparison rate of 80 million strings per second.

B. Overall Block Diagram

The string search engine (SSE) consists of a special 8K content addressable memory, 20K-gate finite-state automaton logic, a priority encoder, delimiter and mask character detectors, and data registers, as shown in Fig. 2. The content addressable memory (CAM) and finite-state automaton (FSA) logic are the key components of the SSE. The input text stream enters the CAM through a data register in the form of character codes and then is concurrently compared with the contents of the stored characters in the CAM. The FSA logic performs string comparison based on the character match signals from the CAM. In the priority encoder, the results of the string comparison are converted into match pattern string address codes, exact or approximate match signals. The encoder can generate address codes in sequence from a lower address when any input string matches multiple pattern strings.

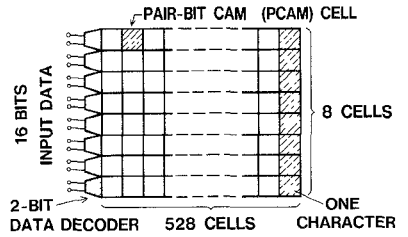


Fig. 3. Memory structure for CAM using newly developed CAM.

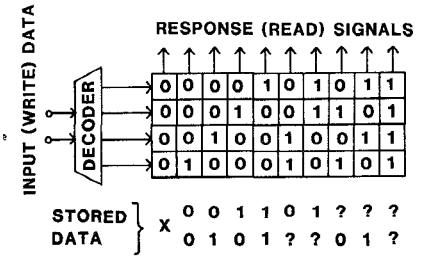


Fig. 4. Principle of PCAM constructed with conventional SRAM cells.

The total CAM capacity of 528 characters (16-bit character codes) consists of 512 characters for 64 pattern strings, eight mask characters as wild cards, and eight delimiter characters for the anchor match mode. When a defined mask character enters the SSE, it triggers the generation of the character match signals independent of the characters stored in the CAM. Thus, the mask character acts like a wild card. Delimiter and mask character detectors are used to detect these delimiters and mask characters, which are stored in 16 words of the CAM.

The SSE can also work as a conventional CAM. Parallel operation of this SSE chip would further expand word capacity and character length, while conventional CAM's are virtually incapable of expanding their data size.

III. Pair-Bit CAM Cell

The newly developed CAM consists of 8×528 pair-bit CAM (PCAM) cells and eight 2-bit data decoders, as shown in Fig. 3. Each CAM word for storing one character consists of eight PCAM cells. Each PCAM cell is basically constructed with four RAM cells. Fig. 4 shows the principle of a new CAM which utilizes a 10-word $\times$ 4-bit conventional RAM. Each column acts as a PCAM cell which can store and compare 2-bit data. A set of PCAM cells can store one of the ten different bit patterns, which include "don't care" bit as shown in Fig. 4. The marks "?" and "X" indicate, respectively, "don't care" bit and cleared condition. To write one of these bit patterns in a specific column, plural rows are energized sequentially or in parallel. The search operation is the same as the READ operation of a conventional RAM, except that the interrogation data, instead of the read address, is supplied to the decoder. The readout signals are then treated as response signals of the PCAM.

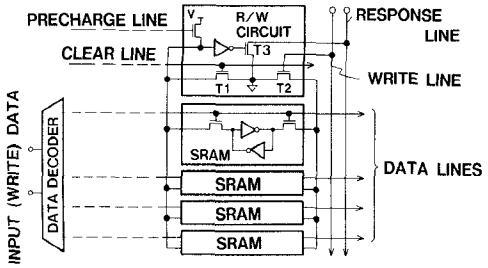


Fig. 5. Circuit diagram of PCAM.

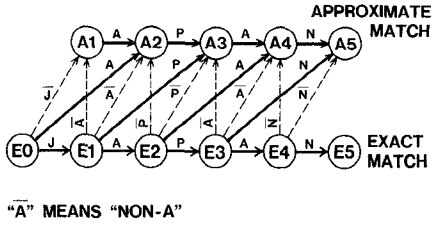


Fig. 6. Finite-state transition diagram for detecting pattern string "JAPAN."

Fig. 5 shows a practical PCAM cell structure which is quite different from the structure of a conventional CAM cell [15]. It is made up of four conventional full static RAM (SRAM) cells and a READ–WRITE circuit to control them. The output lines of a 2-bit data decoder are connected commonly to all of the 528-word PCAM cells, which are arranged horizontally. The decoder energizes one of the four data lines to select one SRAM cell from each PCAM cell. What differs from the operations of a conventional RAM is that the input data are supplied as address codes to the decoder input terminals. Initially, all the SRAM cells must be cleared by the clear line after the power has been turned on. A WRITE operation activates a write line and stores ONE's in the specific SRAM cell that is selected by a 2-bit write data. When the PCAM cell is accessed later by the same 2-bit input data as used for writing, it will turn off the transistor $T3$ and maintain the response line at a high level, indicating that the data have been matched. Access by any different data will discharge the response line and show a mismatch. Thus, this PCAM cell can detect 2-bit data. If two or four SRAM cells are stored with ONE's, the content of the PCAM will indicate one or two "don't care" bits, as shown in Fig. 4. Use of eight PCAM cells that are connected together at the drains of each $T3$ transistor to form a wired-AND logic can detect the match of a 16-bit-coded character. The 16-bit-coded character can store Chinese characters in the SSE.

The newly developed CAM has the following advantages:

1) it is suitable for implementation through VLSI technology, because the logic portion, such as the data decoder, can be easily separated from the regularized storage portion;
2) the design of the CAM makes use of the previously optimized process technology of SRAM's;
3) small stray capacity on the short bit line enables the PCAM cell to operate at a high speed; and
4) the PCAM cell can store "don't care" bits, although it needs a few more transistors than a conventional CAM cell. This means that the new CAM can store not only a "don't care" character for the fixed-length "don't care" matching mode but also two characters having 1-bit differences from each other, such as a capital letter and a small letter in the same CAM word.

IV. Finite-State Automaton Logic

A. State-Transition Diagram

Character comparisons are performed concurrently in the CAM. Character match signals from the CAM control transitions of state in order to compare the strings in the FSA logic. Fig. 6 illustrates the state-transition diagram for an FSA. The circles and arrows represent state nodes and state transitions, respectively. The automaton illustrated in the state-transition diagram in Fig. 6 can extract not only the specific string "JAPAN" but also those strings which include a one character misspelling, such as incorrect (e.g., JXPAN), extra (e.g., JXAPAN), or omitted (e.g., JPAN) characters. Each state transition is controlled by a character match signal generated from the CAM. That is, each pointer in the state node moves to the next state node, as indicated by the arrows, when the character shown on each arrow is supplied. The state nodes on the lower and upper lines correspond to exact and approximate string matches, respectively.

When a character other than the character shown on the solid arrow is supplied, the pointer does not move. Horizontal solid lines indicate the state transition caused by a character match. Diagonal thick lines indicate a one character omission error. For extra character and incorrect character errors, broken lines are used. When the input data string is correctly "JAPAN," the state pointer moves along the state nodes $E0$–$E4$, and finally reaches the last state node $E5$. On the other hand, if the input data string differs from "JAPAN" by one character, the state pointer departs from lower state nodes, and ends up at the upper last state node $A5$. Thus, these lower and upper last state nodes, $E5$ and $A5$, indicate exact and approximate matchings, respectively.

Based on the finite-state transition method, these state nodes and lines can be easily implemented by conventional flip-flops and MOS logic gates, respectively.

B. Finite-State Automaton Logic

Fig. 7 shows the details of the circuit configuration for the FSA logic cell. The SSE has 512-stage FSA logic cells mutually connected by logic gates to emulate the state-transition diagram for the FSA. The FSA logic compares the stored strings with an input text data containing errors, such as extra, omitted, or incorrect single characters. The

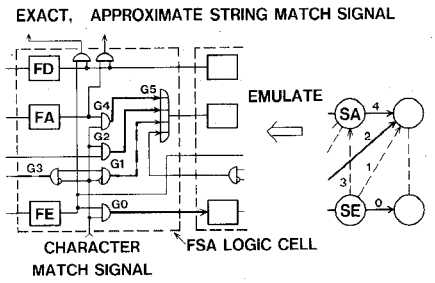


Fig. 7. Circuit diagram of FSA logic cell.

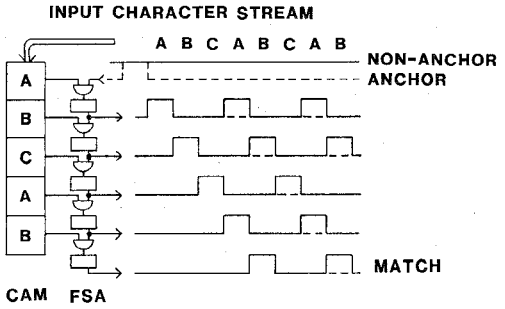


Fig. 8. Search operation in nonanchor mode.

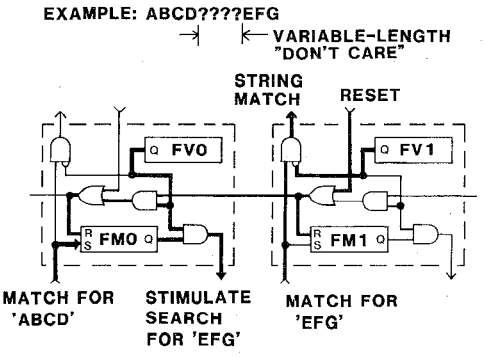


Fig. 9. Matching operation in VLDC mode.

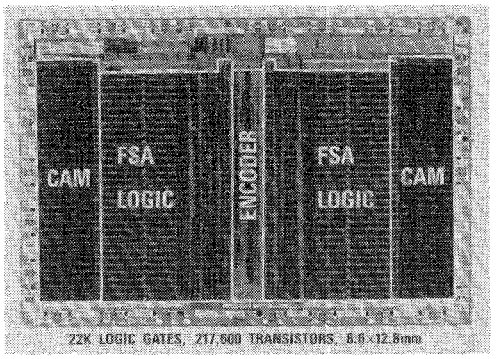


Fig. 10. SSE chip photomicrograph.

exact match flip-flop *FE* and approximate match flip-flop *FA* correspond to the state nodes *SE* and *SA*, respectively. The delimiter flip-flop *FD* stores the end mark that indicates the end point of a variable-length pattern string in the CAM. Through gates *G*0, *G*4, and *G*5, a character match signal from the CAM transfers the pointers in *FE* and *FA* to the next logic cell, with gate numbers corresponding to lines 0 and 4 on the right-hand side of the figure. On the other hand, a character mismatch signal transfers the pointer in *FE* to *FA*'s both in the same and the next logic cells, corresponding to lines 3 and 1 on the right-hand side of the figure. The *FE* (or *FA*) of a particular cell in which the *FD* has the end mark generates the exact (or approximate) match signal to the priority encoder through the variable "don't care" circuit. The SSE contains 64 sequentially connected FSA logics to emulate the state-transition diagram, each of which has eight stages as shown in Fig. 6. Connection of the adjacent FSA logics can detect strings of various character lengths.

Fig. 8 shows how the SSE detects the substring "ABCAB" stored in the CAM from within the input character stream "ABCABCAB" in the nonanchor mode. The string may be present anywhere within the input character stream. Pointer transfer operations, as described above, are carried out for every character input in the nonanchor mode.

If a user wants to start the substring search at a specific point in the input stream, he can use the anchor mode, in which the enable signal input for the first stage will be generated whenever the delimiter characters are encountered in the input. In the anchor mode, the FSA logic cells remain inactive until delimiter character is detected.

C. Matching in VLDC

Fig. 9 shows the variable-length "don't care" (VLDC) circuit in detail. Matching operations in the VLDC mode are useful to detect strings in which an arbitrary number of "don't care" characters are embedded among substrings. The flip-flops *FV*0 and *FV*1 are the VLDC marker flip-flops which define the VLDC mode. Substring match flip-flops *FM*0 and *FV*1 contain a match signal from the FSA logic. If the *FV*0 is set to ONE, the match signal for former substring "ABCD" stimulates FSA logic to search the next substring "EFG." The match signal for "EFG" then produces the string match signal in the VLDC mode. The string match signal is supplied to the priority encoder, which will generate match address codes accordingly. A reset signal for the last substring is sent to the VLDC circuit corresponding to the former substring. A sequentially connected 64 VLDC circuit can be used to detect strings including multiple "don't cares" in the VLDC mode. Hence, multiple VLDC string search allows detection of a 64-pattern substring with a defined sequence.

V. Implementation and Operation

A. Chip Photomicrograph

Fig. 10 shows a photomicrograph of the SSE chip. The geometry of the CAM portion was completely custom designed in order to minimize cell size and improve match detection speed. By contrast, the logic portion, including

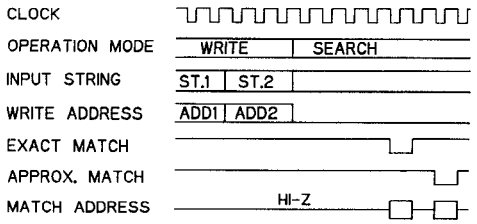


Fig. 11. Operating sequence in typical storing and matching operation.

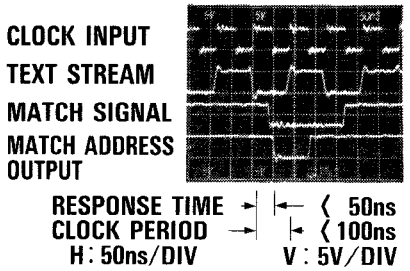


Fig. 12. Operating waveforms in search operation.

22K gates, was designed using 10 300 standard cells, though about 70 percent of the FSA logic was produced by manual cell connections because the regularity of the FSA logic was extremely high. The other logic portions, such as priority encoder and operation-mode controller, were designed by the fully automated standard cell approach. The SSE chip shown here has 217 600 transistors in an 8.62× 12.76-mm die area. It is housed in a 72-pin standard pin-grid-array package, in which 60 pins are used for signals and power supplies. The technology used was a double-metal 1.6-μm n-well CMOS process.

B. Operating Sequence

A typical operating sequence of the SSE is shown in Fig. 11. This operation is for writing two-pattern strings in two words and searching for them. Each character of the input string enters the SSE in synchronism with the clock signal. In the searching operation, an exact match signal or an approximate match signal will be generated with associated match address codes when string 1 or string 2 is entered. Match address lines will be held at high impedance in case of a mismatch.

Typical waveforms for matching operations are shown in Fig. 12. Response time for the match signal is less than 50 ns. Breakdowns of this delay are 8 ns in the input–output buffers, 7 ns in the control logics, 16 ns in the CAM, 5 ns in the FSA logic, and 12 ns in the encoder. The clock period for the text-stream input is less than 100 ns. An input rate for the text stream of 10 million characters per second means that the SSE can search specific strings at 80 million strings per second, because the chip can handle 64 strings in parallel. Power dissipation at 10-MHz clock frequency is about 500 mW. Table I summarizes the specifications.

TABLE I
SSE SPECIFICATIONS

Operation mode	Exact/Approximate match
	Anchor/Non-anchor
	FLDC/VLDC
Pattern string	Max. 64 strings
	Max. 512 character length
Character input rate	10M characters/sec
Power dissipation	500mW (10MHz)
Number of devices	217,600
Die size	8.62 x 12.76 mm
Process technogy	2 metal 1.6 μm CMOS

VI. CONCLUSIONS

Sixty-four finite-state automaton logics were first integrated on a single chip. A new pair-bit CAM cell allowed "don't care" operations without any additional bits. A combination of FSA logic with the PCAM has achieved useful intelligent matching operations, such as an approximate matching allowing one character error, anchor and nonanchor matchings, and fixed-length and variable-length "don't care" matching at high-speed. A character string-search engine based on the above architecture can directly receive document data of 10 million characters per second from communication networks or storage disks. This means that the SSE chip allows a high-speed text retrieval in which document data can be searched at a comparison rate of 80 million strings per second. This performance is several hundred times greater than those reported previously.

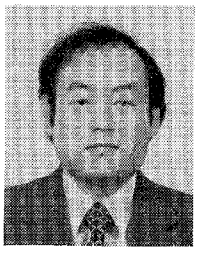
ACKNOWLEDGMENT

The authors would like to thank N. Yosida, S. Itho, K. Matsumi, K. Yoshimi, H. Sakuma, and T. Enomoto for their direction and encouragement. They also wish to thank Y. Kitamura, K. Matsumoto, H. Nakamura, T. Ishidate, N. Ohno, and project members for their technical suggestions.

REFERENCES

[1] D. Lee and F. Lochovsky, "Text retrieval machine," in *Office Automation—Concepts and Tools*. New York: Springer-Verlag, 1985, sec. 14.
[2] M. Bärtschi, "An overview of information retrieval subjects," *IEEE Computer*, vol. 18, pp. 67–84, May 1985.
[3] C. Faloutsos, "Access method for text," *Computing Surveys*, vol. 17, no. 1, pp. 49–74, Mar. 1985.
[4] F. J. Burkowski, "A hardware hashing scheme in the design of a multiterm string comparator," *IEEE Trans. Computers*, vol. C-31, no. 9, pp. 825–834, Sept. 1982.
[5] L. A. Hollaar, "Text retrieval computers," *IEEE Computer*, vol. 12, pp. 40–50, 1979.
[6] M. J. Foster and H. T Kung, "The design of special-purpose VLSI chips," *IEEE Computer*, vol. 13, pp. 26–40, Jan. 1980.
[7] A. Mukhopadhyay, "Hardware algorithms for string processing," in *Proc. ICCC*, 1980, pp. 508–511.
[8] R. L. Haskin, "Special purpose processors for text retrieval," *Database Eng.*, vol. 4, no. 1, pp. 16–29, Sept. 1981.
[9] D. C. Robert, "A specialized computer architecture for text retrieval," in *Proc. 4th Workshop Computer Architecture*, 1982, pp. 51–59.

[10] P. A. V. Hall and G. R. Dowling, "Approximate matching," *Computing Surveys*, vol. 12, no. 4, pp. 381–402, Dec. 1980.
[11] G. Salton, "Automatic information retrieval," *IEEE Computer*, vol. 13, pp. 41–55, Sept. 1980.
[12] P. N. Yianilos, "A dedicated comparator matches symbol strings fast and intelligently," *Electronics*, vol. 56, no. 5, pp. 113–117, Dec. 1983.
[13] K. Takahashi, H. Yamada, H. Nagai, and K. Matsumi, "A new string search hardware architecture for VLSI," *Computer Architecture News*, vol. 14, no. 2, pp. 20–27, June 1986.
[14] H. Yamada, M. Hirata, H. Nagai, and K. Takahashi, "A character string search processor," in *ISSCC Dig. Tech. Papers*, Feb. 1987, pp. 272–273.
[15] H. Kadota, Y. Miyake, Y. Nishimichi, H. Kudoh, and K. Kagawa, "An 8-kbit content-addressable and reentrant memory," *IEEE J. Solid-State Circuits*, vol. SC-20, no. 5, pp. 951–957, Oct. 1985.



Hachiro Yamada was born in Shizuoka, Japan, on June 6, 1949. He graduated from Shizuoka Technical High School in 1968, and from the Nippon Electric Institute of Technology in 1973.

In 1968 he joined NEC Corporation, Kawasaki, Japan, where he was engaged in the research and development of memory hierarchy systems and magnetic bubble memory systems. Since 1985 he has been engaged in the development of VLSI architecture and special-purpose LSI. He is presently a Supervisor in the Microelectronics Research Laboratories, NEC Corporation.

Mr. Yamada is a member of Institute of Electronics, Information and Communication Engineers of Japan.

Masaki Hirata was born in Hiroshima, Japan, on March 9, 1949. He received the B.S. and M.S. degrees in electronic engineering from the University of Electronics Communication, Tokyo, Japan, in 1972 and 1974, respectively.

He joined NEC Corporation, Kawasaki, Japan, in 1974, where he has been working in MOS IC design. His current research interests are in custom LSI design. He is now a Supervisor in the System LSI Development Division, NEC Corporation.

Mr. Hirata is a Member of the Institute of Electronics, Information and Communication Engineers of Japan.

Hajime Nagai was born in Hyougo, Japan, on November 1, 1951. He received the B.E. degree in electrical engineering from Saitama University in 1975.

He joined NEC Corporation, Kawasaki, Japan, in 1975 and is now a Supervisor in the Microelectronics Research Laboratories. He has been engaged in the development of a bubble memory system. Currently he is studying AI devices.

Mr. Nagai is a member of the Institute of Electronics, Information and Communication Engineers of Japan.

Kousuke Takahashi was born in Mie, Japan, on November 22, 1940. He received the M.S. degree from Keio University in 1966.

He joined NEC Corporation, Kawasaki, Japan, in 1966 and is now a Research Manager in the Microelectronics Research Laboratories. He has been engaged in the research and development of plated-wire memory, analog magnetic thin-film memory, and magnetic bubble memory. Now he is researching VLSI architecture for silicon devices.

Mr. Takahashi is a member of the Institute of Electronics, Information and Communication Engineers of Japan, the Information Processing Society of Japan, the Magnetic Society of Japan, and the Audio-Visual Information Research Group (AVIRG).