A 2K-Word Dictionary Search Processor (DISP) LSI with an Approximate Word Search Capability

Masato Motomura, Hachiro Yamada, and Tadayoshi Enomoto, *Member, IEEE*

Abstract—**A 2K-word dictionary search processor (DISP) LSI has been developed. The DISP LSI consists mainly of a 160-kb content addressable memory (CAM) to store keywords and a cellular automaton processor (CAP) to perform concurrent and approximate word searches against the stored keywords. This newly developed CAP performs distance calculation based on dynamic programming (DP), which is necessary in approximate word searches, through the use of an array of extremely simple processor elements. CAP hardware size is less than 1/10 of that of a conventional systolic array processor. The DISP LSI, which was fabricated using a 0.8-μm, triple-layer-Al, CMOS fabrication technology, has a die size of 13.02 × 12.51 mm^2 and contains about 1.2-million transistors. It operates at a clock frequency of 33 MHz with a 5-V power supply, and it typically consumes 800 mW.**

I. Introduction

Searching input word streams for specific words is an important operation in almost all information retrieval systems, including text database retrieval systems [1], dictionary search systems [2], and biomorphic database systems [3]. Much work has been done on word search technology, especially in the software domain, and state-of-the-art programs are fast enough for "exact word searches" for single keywords [4]. Here, "exact word search" refers to a search for a word that exactly matches with a given keyword.

However, there are many application fields that require 1) a concurrent search that can handle a number of keywords at the same time, and/or 2) an "approximate word search" that permits character errors in keywords and/or in retrieved words.

In full text database retrieval systems, for example, a search involving a given keyword may include, as well, a search for its synonyms. In response to being given a keyword, the system will first generate a synonym dictionary typically consisting of several tens of words that have almost the same meaning as the given keyword, and then it will start a search for all of them. Searches executed sequentially for all items produce intolerably long response times. Such searches should be done concurrently.

Furthermore, misspellings may occur in the text to be searched: an exact word search will be meaningless if such errors exist. In other words, approximate word searches are practically necessary.

The dictionary search processor (DISP) LSI [5] presented in this paper is a hardware solution for concurrent and approximate word searches. The DISP consists mainly of a 160-kb content addressable memory that can store 2048 keywords and a newly developed cellular automaton processor that performs concurrent and approximate word searches for stored keywords. We present here the DISP architecture for approximate word search in detail and compare it to conventional technology.

Manuscript received August 27, 1991; revised February 12, 1992.

M. Motomura is with the Computation Structure Group, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, on leave from the System Ultra-LSI Research Laboratory, Microelectronics Research Laboratories, NEC Corporation, 1120 Shimokuzawa, Sagamihara-shi, Kanagawa-ken, 229, Japan.

H. Yamada and T. Enomoto are with the System Ultra-LSI Research Laboratory, Microelectronics Research Laboratories, NEC Corporation, 1120 Shimokuzawa, Sagamihara-shi, Kanagawa-ken, 229, Japan.

IEEE Log Number 9108189.

II. Conventional Approximate Word Search

A. Concept

Consider a search for a single keyword. In exact word searches, equality checking is performed between the keyword and each word in an input word stream. Words that exactly match with the keyword are retrieved from the input word stream. In approximate word searches, instead of the equality checking, a degree of error, called distance, between those two words is calculated [6]. Then the words that have distance values smaller than a given criterion are recognized as approximately matching the keyword. The key to conducting approximate word searches is the method of calculating distance.

The distance between an input word and a keyword is defined as follows:

$$\begin{aligned}\text{distance} = \text{Min}\ \{&\text{No. of character substitutions}\\ &+ \text{No. of character deletions}\\ &+ \text{No. of character insertions}\}. \qquad (1)\end{aligned}$$

Here, "Min" function returns the minimum value of the addition of all possible combinations of substitutions, deletions, and insertions. In other words, distance is the minimum number of unit operations required to change a word into the keyword, unit operations being single character substitution, deletion, and insertion. For example, the distance of "registor" from "register" is 1 because a substitution of "o" for "e" is required. Other examples of distance are shown in Table I.

TABLE I
EXAMPLES OF DISTANCE CALCULATIONS
Distances are calculated from the number of unit operations required to change an input word into the keyword

Input Word	Key Word	Misspellings
SanFransisco	San Francisco	1 Deletion 1 Substitution
Mishishippi	Mississippi	2 Substitution
Masachussetts	Massachusetts	1 Deletion 1 Insertion

B. Algorithm

One well-known algorithm for calculating distance is dynamic programming (DP), commonly used in voice recognition [6]. The fundamental algorithm is as follows:

$$d_1(1) = 0, \tag{2a}$$

$$d_2(2) = \text{Min}\,\{d_1(1) + M_1(1),\ d_2(1) + 1,\ d_1(2) + 1\}, \tag{2b}$$

$$\vdots$$

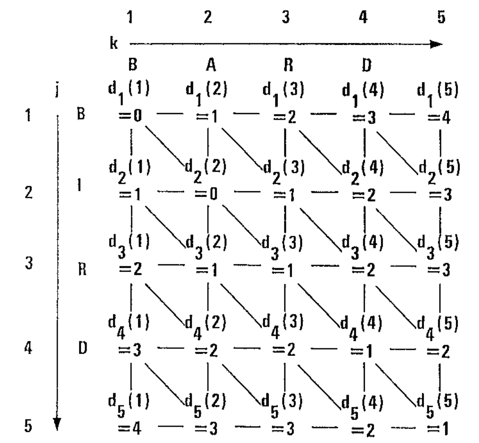
$$d_j(k) = \text{Min}\,\{d_{j-1}(k-1) + M_{j-1}(k-1),\ d_j(k-1) + 1,\ d_{j-1}(k) + 1\} \tag{2c}$$

where $d_j(k)$ is the distance between a substring, up to the $(j-1)$th character, of an input word with length m, and a substring, up to the $(k-1)$th character, of a keyword with length n. When $j < 1$ or $k < 1$, $d_j(k)$ is defined to be 0. $M_j(k)$, match signals, depend on a character comparison between the jth character of the input word and the kth character of the keyword: 0 for a match and 1 for mismatch.

Equation (2a) is the initial condition for DP calculation. Equation (2c), the general equation for DP, recursively adds 1 to the distance value whenever a unit operation is performed. The first, second, and third terms of the "Min" in (2c) correspond to a character substitution, deletion, and insertion, respectively. The "Min" function in this equation returns the minimum value of the three terms. The final result of DP calculation, i.e., the distance between the input word and the keyword, is obtained as $d_{m+1}(n+1)$. For example, the distance calculation process according to (2) between the words "BARD" (input word) and "BIRD" (keyword) is shown in Fig. 1. The grid contains $d_j(k)$ where the y axis is j and the x axis is k. The distance value at each point in the grid is calculated from the left, upper left, and upper neighbors of that point, and a corresponding match signal value according to (2). For example, $d_3(4)$ is calculated as

$$d_3(4) = \text{Min}\,\{d_2(3) + M_2(3),\ d_3(3) + 1,\ d_2(4) + 1\}$$
$$= \text{Min}\,\{1 + 1,\ 1 + 1,\ 2 + 1\} = 2.$$

These calculations compute a distance of 1 (which is found in $d_5(5)$) between these two words, which is correct because there is only one character substitution error.



k 1 2 3 4 5
B A R D
j 1 2 3 4 5
B I R D
d1(1)=0 d1(2)=1 d1(3)=2 d1(4)=3 d1(5)=4
d2(1)=1 d2(2)=0 d2(3)=1 d2(4)=2 d2(5)=3
d3(1)=2 d3(2)=1 d3(3)=1 d3(4)=2 d3(5)=3
d4(1)=3 d4(2)=2 d4(3)=2 d4(4)=1 d4(5)=2
d5(1)=4 d5(2)=3 d5(3)=3 d5(4)=2 d5(5)=1

Fig. 1. Distance calculation process in DP algorithm. The input word is "BARD" and the keyword is "BIRD." $d_j(k)$ is calculated from $d_j(k-1)$, $d_{j-1}(k-1)$, $d_{j-1}(k)$, and $M_j(k)$.

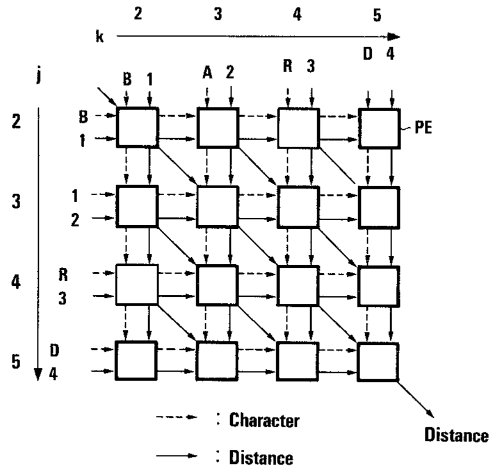


Fig. 2. The systolic array processor (SAP) architecture. The (j, k)th PE calculates $d_j(k)$ according to (2c).

C. Systolic Array Processor

The distance calculation described above can be performed by a conventional systolic array processor (SAP), formed by a two-dimensional array of processor elements (PE's) [7] (see Fig. 2). The (j, k)th PE calculates $d_j(k)$ on the basis of distance values for neighboring PE's and the match signal produced in the (j, k)th PE. In Fig. 2, PE's are provided only for the grids $j > 1$ and $k > 1$, because, as can be seen from (2c) and Fig. 1, distance values for other grids are fixed to $|j - k|$. Characters for the two words being compared are loaded into the SAP one at each time step. They continue in the same direction, vertically or horizontally, progressing one step to the next PE for each time step.

An individual PE is shown in Fig. 3. PE circuits are divided into two blocks, a character matching block, which includes two shift registers and a match circuit, and a distance calculation block, which includes three adders and a 3-to-1 minimum value detector. The character matching block produces a character match signal and sends characters to neighboring PE's, while the distance

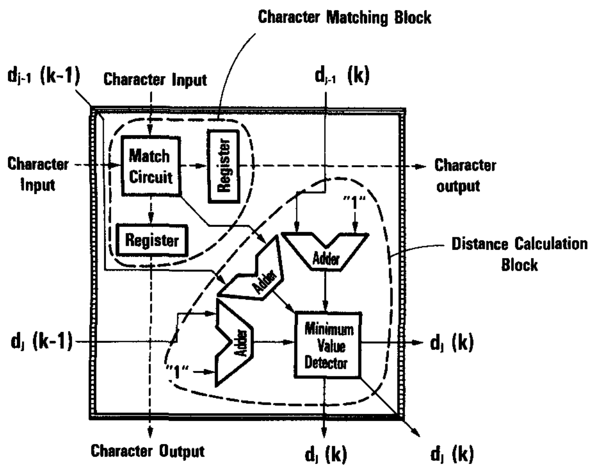


Fig. 3. Processor element of the SAP. Each PE consists of a character matching block and a distance calculation block.

calculation block calculates distance using the match signal and distance values previously calculated by neighboring PE's. Assuming that the character code is 16 b and that the maximum distance value which can be calculated is $(2^q - 1)$, the total number of gates for a single PE is roughly $340 + 40q$, while the number of gates in the critical path is roughly $20 + 4q$.

III. Efficient Architecture for Concurrent and Approximate Word Search

Though the SAP can calculate distances between two words, its hardware is too large and complicated for practical use in concurrent and approximate word searches. For example, let us assume that we are to carry out a concurrent approximate word search, whose maximum distance is 3 ($q = 2$) for 128 four-character keywords. In this case, 128 of the SAP shown in Fig. 2 would be required, and the total number of gates would reach 860K. Today's LSI technology cannot integrate such a huge number of gates onto a single chip.

In an attempt to resolve the problems encountered in working forward highly concurrent approximate word searches, we have developed an efficient architecture which features a complete separation of the character matching block from the distance calculation block.

Fig. 4 shows the architecture which, in simplified form, consists of a character matching block and a separate distance calculation block. A keyword "BIRD" is stored in registers in the character matching block. The word "BARD" is loaded one character per time step into the match circuits. Each input character is compared with all characters of the stored keyword at the match circuits simultaneously, and match signals are produced. The match signals shown in Fig. 4 are at time $t = 1$. The distance calculation block uses these match signals to calculate the distance between an input word and a stored keyword.

In this architecture, the keyword is always stored in a single set of registers; in the SAP architecture, the keyword is shifted from registers in a PE to those in another

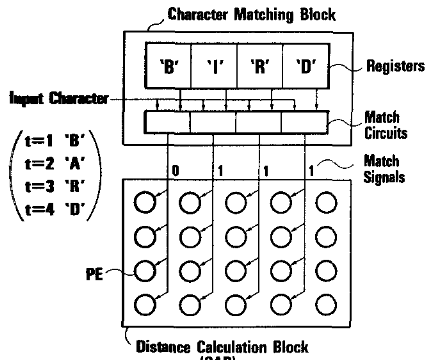


Fig. 4. Cellular automaton processor (CAP) architecture. In contrast to the SAP architecture, the character matching block and the distance calculation block are completely separated.

PE. Thus, the number of registers and match circuits is drastically reduced, which means the character matching block is much simpler than that of the SAP architecture. In the next section, it is shown that the distance calculation block can also be made extremely simple.

IV. Cellular Automaton Processor

A. Basic Concept

Our distance calculation block is newly developed and called a cellular automaton processor (CAP). In Fig. 4, the CAP is composed of an array of PE's. Each PE takes only two values, 1 or 0. For the sake of convenience, we refer to a value of 1 as a "flag." Thus, "a PE has a flag" means "a PE has a value of 1." In a PE array, the number of columns and the number of rows correspond to, respectively, (stored keyword length) + 1 and (maximum calculatable distance) + 1. Thus, the PE array shown in Fig. 4 can calculate distance up to 3 for a four-character keyword. The basic CAP operation consists of transmitting flags from PE to PE at each time step. The destination of transmitted flags is determined by the value of match signals coming into PE's that have flags. The distance value is indirectly obtained from the flag location in the PE array as explained below.

In order to understand CAP distance calculation, let us first consider it as it calculates distance based only on character substitution. The CAP can calculate this distance by imposing the following initial condition and applying the following operation rules:

Initial condition: set a flag at the (1, 1)th PE at time $t = 1$.

Flag transition rule: when a PE having a flag receives a match signal of 0 at time $t = k$, shift the flag to the right neighboring PE at $t = k + 1$, otherwise shift the flag to the right lower neighboring PE at $t = k + 1$.

Encoding rule: distance is calculated to be (the row number of the PE which has a flag in a rightmost column) − 1.

Fig. 5 shows the distance calculation process (time $t = 1$ to $t = 5$) for the example used in Fig. 4. In Fig. 5, black

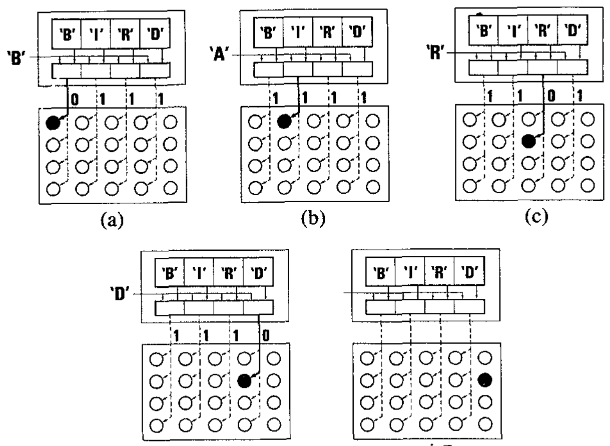


Fig. 5. A simplified CAP distance calculation process for a character substitution error. A flag changes the row downward when a substitution error occurs. (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$, (e) $t = 5$.

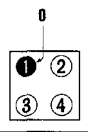
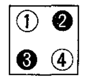
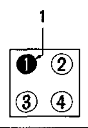
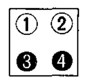
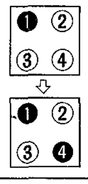
	Match Signal	Flag State at t = k	Flag State at t = k+1	Character Errors
First Rule	0			2 : Match 3 : Ins.
Second Rule	1			3 : Ins. 4 : Subs.
Third Rule	—		—	: Del.

Fig. 6. Complete flag transition rules for the CAP distance calculation. The first and second rules specify flag transitions from time $t = k$ to $t = k + 1$, while the third rule specifies a flag transition at time $t = k$.

circles represent PE's that have flags. At $t = 1$, a flag has been set according to the initial condition and, at each time step, flag transitions are carried out according to the flag transition rule whenever match signals are received by a PE having a flag. The match signals that determine the destination of flag transmissions are denoted by solid line arrows in Fig. 5. From time $t = 2$ to $t = 3$, the flag shift from the first row to the second row is determined by the match signal indicating a character substitution error of "A" for "I." The distance increases by 1 at this point. In subsequent steps, the match signal values are zero, so that a flag merely moves across to the rightmost column in the second row, which by the encoding rule means that the distance between these two words is 1.

As can be seen from the example, the basic concept here is to calculate distance by shifting a flag downward whenever a character mismatch occurs. Distance for any character substitution error can be calculated by this simple scheme.

B. CAP Operation Rules

In order to calculate distances between two words, not only character substitution errors but also insertion and deletion errors must be handled correctly. The innovative aspect of the CAP architecture is that it satisfies this requirement by simple variations of the flag transition rule and encoding rule given above. The initial condition and two new operation rules for CAP operation are exactly equivalent to the DP algorithm mathematically, as shown in the Appendix.

Fig. 6 shows the new flag transition rules. The first rule covers flag transmissions when a match signal of 0 is applied to a PE 1 which has a flag. Flags are then transmitted to the PE 2 and PE 3. The transition to the PE 2 indicates a possible character match (distance does not increase in this case), and the transition to a PE 3 indicates a possible character insertion error. This latter transition is required because of the fact that a character insertion error can take place even if the input character matches a character at a corresponding location in a keyword, such as in the input word "A$\underline{B}$BCD" against a keyword "ABCD."

The second rule covers flag transmissions when a match signal of 1 is applied to a PE 1. The transition to a PE 4 indicates a possible character substitution error, and the transition to a PE 3 covers the possibility of the mismatch being due to a character insertion error.

The third rule covers the possibility of character deletion errors. This rule says that if any PE 1 has a flag at time $t = k$, the PE to its lower right (PE 4) must also have a flag at that same time $t = k$ regardless of a match signal value.

The new encoding rule for the CAP is as follows: distance $d_j(k)$ (see Section II-A) is calculated by applying Table II for a jth column at time $t = k$. In Table II, "X" is used to indicate that the PE value may be either 1 or 0. It can be seen that by replacing "X" in Table II by 0 and by restricting application of the rule to the rightmost column, this new encoding rule is reduced to the rule described in the Section IV-A.

C. Distance Calculation Process

Fig. 7 shows the CAP calculation process of the distance between an input word "BRDI" and a keyword "BIRD" based on the operation rules. In this example, there is a deletion of "I" between "B" and "R" and an insertion of "I" after "D," i.e., the distance is 2.

As an initial state, a flag is set on the (1, 1)th PE at time $t = 1$. By the third transition rule, the (2, 2)th, (3, 3)th, and (4, 4)th PE's must also have flags at the same time (Fig. 7(a)). When the first character of the input word "B" is loaded at time $t = 1$, it is compared with all characters of the keyword, and match signals of 0, 1, 1, and 1 are produced. These match signal values and the three flag transition rules determine flag transitions from $t = 1$

TABLE II
RELATIONSHIP BETWEEN DISTANCE AND PE VALUES IN THE CAP
"X" means don't care (0 or 1). Distance is calculated as (the smallest row number of the PE which has value 1) − 1.

PE Value	Distance 0	1	2	3
1st Row	1	0	0	0
2nd Row	X	1	0	0
3rd Row	X	X	1	0
4th Row	X	X	X	1

TABLE III
PERFORMANCE COMPARISON BETWEEN THE SAP AND CAP
Conditions for the comparison are 16-b character code, 4 characters/word, and $2^q - 1$ distance.

	SAP	CAP
No. of Total Gates	$16(340 + 40q)$	$4(80 + 10 \cdot 2^q)$
No. of Gates for Critical Path	$24 + 4q$	$7 + 2^q$

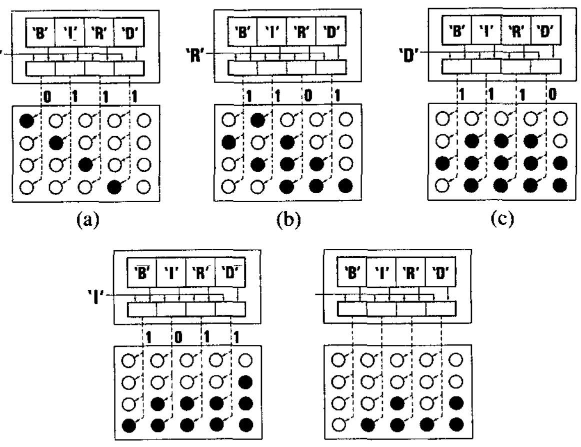


Fig. 7. CAP distance calculation process based on the complete flag transition rules. From the encoding rule, the flag states in Fig. 7(e) show that the distance between "BRDI" and "BIRD" has been determined to be 2. (a) $t = 1$, (b) $t = 2$, (c) $t = 3$, (d) $t = 4$, (e) $t = 5$.

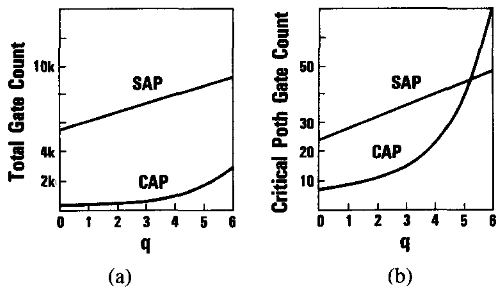


Fig. 8. Performance comparison between the SAP and the CAP: (a) total gate count, and (b) critical path gate count. Figures from Table III have been used. CAP performance greatly exceeds that of the SAP in the low distance value range.

to $t = 2$, which result in the PE states shown in Fig. 7(b). For example, a flag on the (1, 1)th PE at $t = 1$ is transmitted to both (1, 2)th and (2, 1)th PE's at $t = 2$ according to the first rule, which in turn give rise to flags on the lower right diagonal PE's at the same time according to the third rule, and so on. Similar flag transitions take place for the input characters "R" and "D" as shown in Fig. 7(c) and (d), respectively, and finally the PE state of Fig. 7(e) is obtained after the input of "I" ($t = 5$). Here the third row is the lowest row that has a flag in the fifth column. This means that $d_5(5)$, i.e., distance between the two words, is 2 according to the encoding rule. The equivalence of the distance $d_j(k)$ obtained in this way at any point in the process with those of the DP algorithm may be confirmed by comparison with results obtained from (2c).

D. Performance Comparison

As described above, it is clear that our CAP-based distance calculation hardware shown in Fig. 4 (in this section, we call it simply CAP for convenience) can perform the same operation as a SAP. Table III shows a rough performance comparison of the CAP and the SAP, where comparison conditions were: 16-b character code, four characters per one word, and $2^q - 1$ distance. The comparisons of total gate counts and critical path gate counts shown here are also depicted graphically in Fig. 8(a) and (b). As can be seen from Table III, both of the SAP totals are linear to q, while those for CAP are linear to 2^q, which is equal to (distance + 1). This means that though the CAP has a great advantage over the SAP in the small distance value range, the SAP's performance will exceed that of the CAP in the large distance value range. However, it should be noted that maximum distance value required for an approximate word search is at most 10 because word lengths are generally limited to around 10. In this distance value range, CAP hardware size is much smaller than SAP's and its operation speed is much faster. For example, for a distance of 3, the total CAP gate number (480) is 1/14 of that of the SAP (6720), and the CAP critical path gate number (11) is about 1/3 of that of the SAP (32). Thus, it can be concluded that the CAP is better suited than a SAP to approximate word searches.

V. DISP LSI

A. Block Diagram

In the DISP LSI, the new architecture has been refined to achieve enhanced concurrency. The refinement basically consists of the replacement of a character matching block, seen in Fig. 4, with a content addressable memory (CAM). Though the CAM functions in the same way as a combination of registers and match circuits, it can attain much larger densities [8], [9]. Thus, a highly concurrent word search is made possible by storing a large number of keywords in the CAM [10]–[12].

A block diagram of the DISP LSI is given in Fig. 9. It consists of a 160-kb data CAM (DCAM), a 1-kb control code CAM (C^2CAM), a cellular automaton processor (CAP), a priority encoder, and a controller. The DCAM is designed with a new architecture that enlarges CAM

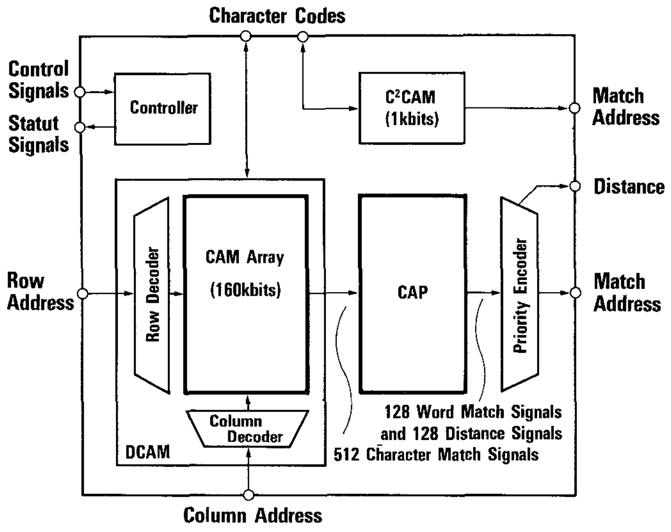


Fig. 9. Block diagram of the dictionary search processor (DISP) LSI.

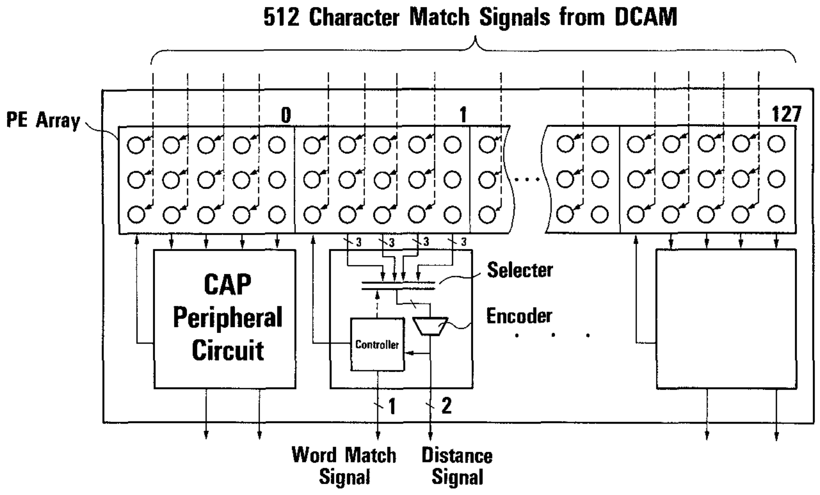


Fig. 10. Architecture of the CAP used in the DISP LSI. It consists of 128 blocks, each of which is a PE array of 3 rows × 5 columns. Each block can calculate a maximum distance of 2 for a single word whose maximum character length is 4.

capacity [13]. It can store 8192 characters. Single characters consist of 20 b each: 16 b are for storing a character code and 4 b for storing tag information. Since four characters are the minimum assignment unit for a single word (see the next section), a maximum of 2048 keywords can be stored in a DCAM. In word search operations, a block of 128 stored keywords (512 characters) is selected by a C^2CAM and compared simultaneously with an input word (the selection process is discussed in detail in [13]). Comparison between an input word and selected keywords is performed character by character in the manner seen in Fig. 7. The resulting 512 match signals are sent to the CAP, which calculates concurrently the distance between an input word and each stored keyword. A priority encoder encodes addresses of any approximately (or exactly) matched keywords.

B. CAP Implementation

Fig. 10 gives a block diagram of the type of CAP integrated on the DISP shown in Fig. 9. The PE array is divided into 128 blocks, each of which consists of fiv columns and three rows. From the explanation given i Section IV-A, one block can calculate distances up to for a four-character keyword. Each block is controlled b a CAP peripheral circuit with the following two func tions:

1) adjusting the PE array column number for a singl keyword in order to treat keywords whose lengt are not equal to 4,
2) determining whether or not an input word has bee approximately matched with the keyword and issuing a word match signal and a distance signal fo the input word.

For the first function, the PE value outputs of each of fou columns are entered into a selector in Fig. 10. The selector selects one set of PE values from these inputs, and sends them to an encoder where the encoding rule is applied. Since distance for an n-character keyword can be obtained by encoding PE values in a $(n + 1)$th column,

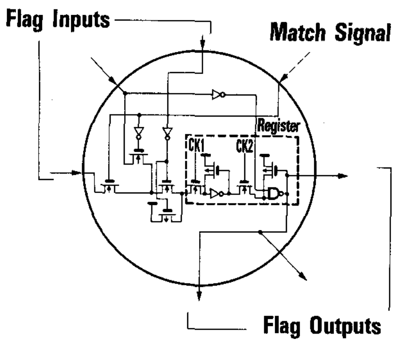


Fig. 11. Circuit diagram of PE. NMOS pass-transistor logic is used to reduce the number of transistors.

distance calculations can be performed for a keyword whose length is less than four characters. Furthermore, it is also possible to concatenate several blocks to make up a single long PE array. This allows distance calculations to be carried out for keywords of lengths from 1 to 512 characters.

The determination described for the second function above is made on the basis of a comparison between the distance produced from the encoder and a criterion stored in a controller, where word match signals are also produced. As described in Section II-A, an input word is recognized as approximately matched with a stored keyword only after its distance is calculated to be less than a given criterion. This criterion can here be arbitrarily set for each keyword individually, dramatically increasing the flexibility of our approximate word searches.

A circuit diagram for the PE is shown in Fig. 11. It is comprised of logic gates for applying the flag transition rules and a 1-b register to store and transmit flags. The logic gates are extremely simple because of the simplicity of the transition rules and also because of an nMOS pass-transistor logic scheme which was adopted for reducing transistor numbers.

VI. Chip Characteristics and Applications

Fig. 12 shows a photograph of a DISP LSI which was fabricated with a 0.8-μm rule, triple-layer-Al interconnection, CMOS process technology. 1.2 million transistors were integrated on a 13.02 × 12.51-mm^2 chip, for which 1 091 000 were for the DCAM and 93 000 for the CAP and the priority encoder. Layouts for the CAP along with the DCAM and C^{2}CAM were designed manually, while other blocks were designed using an automatic layout program based on a building-block scheme. The regular array structure of the CAP was highly convenient for manual layout.

Fig. 13 shows a schmoo plot of power supply voltage versus cycle time. Using a 5-V power supply, the operating frequency is approximately 33 MHz. That is, the DISP can perform the approximate word search at an input rate of 33-million characters per second. Typical power dissipation in this case is 800 mW. The characteristics of the DISP are summarized in Table IV.

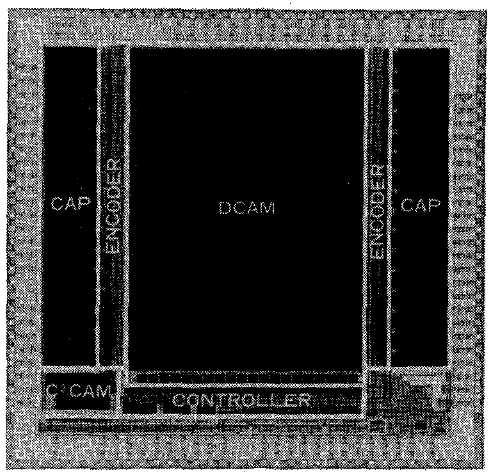


Fig. 12. Photograph of the 13.02-mm × 12.51-mm DISP LSI, which contains approximately 1.2-million transistors. The LSI was fabricated using 0.8-μm CMOS technology.

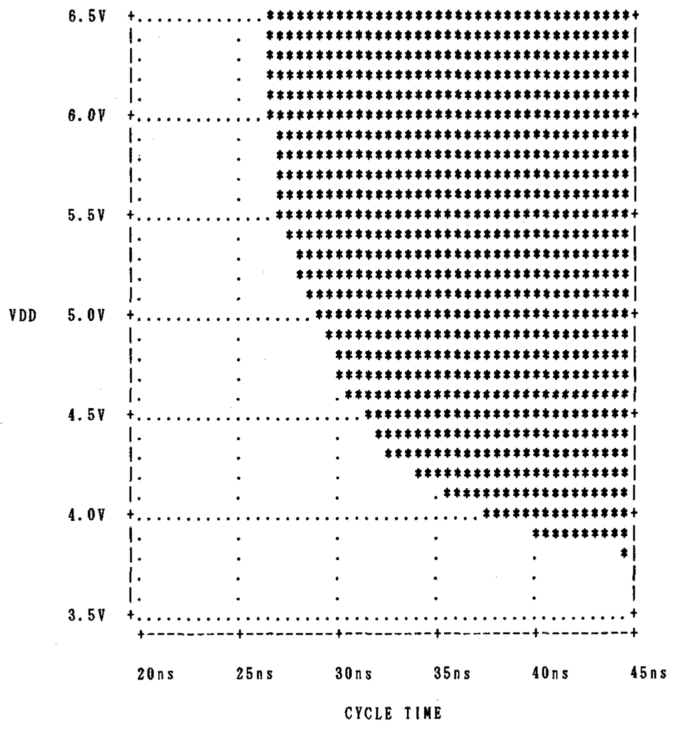


Fig. 13. A schmoo plot of power supply voltage versus cycle time.

TABLE IV
DISP LSI Features

CAM Capacity	160 kb + 1 kb (20 b/char.)
Number of Stored Key Words	Maximum 2048 (4 char./word)
Concurrent Search Ability	Maximum 128
Approximate Search Ability	Maximum Distance 2
Operating Frequency	33 MHz
Character Input Rate	33M characters/second
Power Supply Voltage	5 V
Power Dissipation	800 mW
Number of Transistors	1 208 500
Chip Size	13.02 × 12.51 mm^2
Process Technology	0.8 μm, 3-layer Al, CMOS

A full text database retrieval system, which was introduced in Section I, is a straightforward application for a DISP LSI. The capacity of a single DISP chip is enough to store keywords and their synonyms for full text database searches. Moreover, its approximate word search capability is very useful for this application, since inputs of texts either through typing or, in some cases, optical character recognition, unavoidably produce character errors in a text to be searched. Another important application is a dictionary search system used for machine translation. A realistic dictionary search system requires at least a 50K word dictionary, so that 25 DISP chips are required to store the dictionary. By simultaneously applying an input word to be translated to those DISP's in parallel, each DISP searches its stored words individually and produces match addresses if approximate matches occur. The global match address is obtained by encoding match addresses of all DISP's. Further discussion on this application can be found in [13].

VII. Summary

We have developed here a highly efficient architecture for approximate word searches. We call it the cellular automaton processor (CAP) architecture. The CAP can perform the distance calculations necessary in approximate word searches with far more compact hardware than that of conventional systolic array processors (SAP's). The advantage is especially evident in the small distance value range. For example, with a keyword length of 4 and a maximum distance of 3, CAP hardware is about 1/14 the size of a SAP performing the same function.

By combining CAP and a newly developed 160-kb large-capacity CAM, we have managed to develop a new DISP LSI that can store 2048 keywords in a single chip and that performs approximate word searches concurrently for 128 stored keywords. The maximum calculatable distance is 2. The character input rate is 33M characters per second at 5 V.

Appendix

The equivalence between the DP algorithm, (2a) and (2c), and the condition and rules for CAP distance calculation described in Section IV are proven in this appendix.

Vector representation of distance $d_j(k)$,

$$[d_{n,j}(k), d_{n-1,j}(k) \cdots d_{i,j}(k) \cdots d_{2,j}(k), d_{1,j}(k)],$$

is defined as follows:

1) each vector element $d_{i,j}(k)$ is either 0 or 1;
2) distance $d_j(k)$ is expressed by distance vector $[d_{i,j}(k)]$ as

$$\begin{aligned} d_j(k) &= 0 * d_{1,j}(k) + 1 * \overline{d_{1,j}(k)} * d_{2,j}(k) \\ &\quad + 2 * \overline{d_{1,j}(k)} * \overline{d_{2,j}(k)} * d_{3,j}(k) + \cdots \\ &= \sum_{i=1}^{n} (i-1) * \overline{d_{1,j}(k)} * \cdots * \overline{d_{i-1,j}(k)} \\ &\quad * d_{i,j}(k). \end{aligned} \tag{A1}$$

In other words,

$$d_j(k) = I - 1 \tag{A2a}$$

in case conditions

$$d_{i,j}(k) = 0, \quad \text{when } i < I \tag{A2b}$$

$$d_{i,j}(k) = 1, \quad \text{when } i = I \tag{A2c}$$

$$d_{i,j}(k) = \text{X}, \quad \text{when } i > I \tag{A2d}$$

are satisfied. Here, X means don't care, i.e., 0 or 1.

First, according to (A2), (2a) is transformed into the following equation in the vector representation:

$$\begin{aligned} d_{i,1}(1) &= 1, \quad \text{when } i = 1 \\ &= \text{X}, \quad \text{otherwise.} \end{aligned} \tag{A3}$$

Next, by utilizing the following definitions:

$$e_j(k) = d_{j-1}(k-1) + M_{j-1}(k-1) \tag{A4a}$$

$$f_j(k) = d_j(k-1) + 1 \tag{A4b}$$

$$g_j(k) = d_{j-1}(k) + 1 \tag{A4c}$$

(2c) is expressed as

$$d_j(k) = \text{Min}\ \{e_j(k), f_j(k), g_j(k)\}. \tag{A5}$$

In the vector representation, (A4) are changed into the following equations:

$$\begin{aligned} e_{i,j}(k) &= d_{i,j-1}(k-1), \quad \text{when } M_{j-1}(k-1) = 0 \\ e_{i,j}(k) &= d_{i-1,j-1}(k-1), \quad \text{otherwise} \end{aligned} \tag{A6a}$$

and

$$f_{i,j}(k) = d_{i-1,j}(k-1) \tag{A6b}$$

$$g_{i,j}(k) = d_{i-1,j-1}(k). \tag{A6c}$$

Here, $d_{i,j}(k)$ having an index value (i, j or k) less than 1 is defined to be 0. Equation (A5) can be expressed as

$$\begin{aligned} d_{i,j}(k) &= e_{i,j}(k) + \overline{e_{i,j}(k)} * f_{i,j}(k) \\ &\quad + \overline{e_{i,j}(k)} * \overline{f_{i,j}(k)} * g_{i,j}(k). \end{aligned} \tag{A7}$$

Equations (A6) and (A7) can be further simplified by utilizing Boolean expressions. Equations (A6a)–(A6c) are expressed as

$$\begin{aligned} e_{i,j}(k) &= d_{i,j-1}(k-1)\overline{M_{j-1}(k-1)} \\ &\quad + d_{i-1,j-1}(k-1)M_{j-1}(k-1) \end{aligned} \tag{A8a}$$

$$f_{i,j}(k) = d_{i-1,j}(k-1) \tag{A8b}$$

$$g_{i,j}(k) = d_{i-1,j-1}(k) \tag{A8c}$$

and (A7) is expressed as follows by using (A8):

$$\begin{aligned} d_{i,j}(k) &= e_{i,j}(k) + f_{i,j}(k) + g_{i,j}(k) \\ &= d_{i,j-1}(k-1)\overline{M_j(k-1)} \\ &\quad + d_{i-1,j-1}(k-1)M_j(k-1) \\ &\quad + d_{i-1,j}(k-1) + d_{i-1,j-1}(k). \end{aligned} \tag{A9}$$

Finally, we have three Boolean expressions in vector representation which are mathematically the same as the original (2a) and (2c):

1) (A1) for defining the relation between the distance and the distance vector,
2) (A3) for giving the initial condition for distance vector calculation which corresponds to (2a), and
3) (A9) for defining distance vector calculation which corresponds to (2c).

The approximate word search architecture described in Section IV is based on these equations. By assigning $d_{i,j}(k)$ to the state of the (i, j)th PE at $t = k$, the condition and rules described in Section IV are obtained as follows:

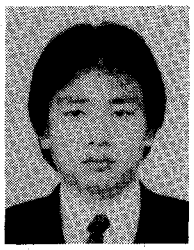
Initial condition: (A3),
Flag transition rule: (A9),
Encoding rule: (A1).

Acknowledgment

The authors are very grateful to all the people involved in developing the DISP ULSI and wish to express their thanks for all the generous help and cooperation they have received.

References

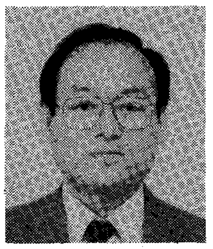
[1] L. A. Hollaar, "Text retrieval computers," *IEEE Computer*, vol. 12, pp. 40–50, 1979.
[2] T. Fukusima, Y. Ohyama, and H. Miyai, "A hardware algorithm for high speed morpheme extraction and its implementation," in *Proc. 28th Annual Meeting Assoc. Computational Linguistics*, 1990, pp. 307–314.
[3] W. D. Dettloff, R. K. Singh, C. T. White, and B. W. Erickson, "A 50MHz 1.5M transistor ASIC for biosequence analysis," in *ISSCC Dig. Tech. Papers*, Feb. 1991, pp. 40–41.
[4] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Commun. Ass. Comput. Mach.*, vol. 20, no. 10, pp. 762–772, Oct. 1977.
[5] M. Motomura *et al.*, "A 1.2-million transistor, 33-MHz, 20-bit dictionary search processor (DISP) with a 160-kb CAM," in *ISSCC Dig. Tech. Papers*, Feb. 1990, pp. 90–91.
[6] P. A. Hall and G. R. Dowling, "Approximate string matching," *Computing Surveys*, vol. 12, no. 4, pp. 381–402, Dec. 1980.
[7] H. D. Cheng and K. S. Fu, "VLSI architecture for string matching and pattern matching," *Pattern Recognition*, vol. 20, no. 1, pp. 125–141, 1987.
[8] T. Ogura, J. Yamada, and T. Nikaido, "A 4-kbit associative memory LSI," *IEEE J. Solid-State Circuits*, vol. SC-20, no. 6, pp. 1277–1282, Dec. 1985.
[9] H. Kadota, J. 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.
[10] H. Yamada, M. Hirata, H. Nagai, and K. Takahashi, "A high-speed string-search engine," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 5, pp. 829–834, Oct. 1987.
[11] M. Hirata, H. Yamada, H. Nagai, and K. Takahashi, "A versatile data string-search VLSI," *IEEE J. Solid-State Circuits*, vol. 23, no. 2, pp. 329–335, Apr. 1988.
[12] J. P. Wade and C. G. Sodini, "A ternary content addressable search engine," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1003–1013, Aug. 1989.
[13] M. Motomura *et al.*, "A 1.2-million transistor, 33-MHz, 20-b dictionary search processor (DISP) ULSI with a 160-kb CAM," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1158–1165, Oct. 1990.



Masato Motomura received the B.S. and M.S. degrees in physics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively.

In 1987 he joined the System ULSI Research Laboratory, Microelectronics Research Laboratories of NEC Corporation, Sagamihara, Japan. Since then he has been working on intelligent memory LSI's and video signal processor LSI's. He is currently on leave for one year until August 1992 at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge. His research interests include VLSI architecture and parallel processing.

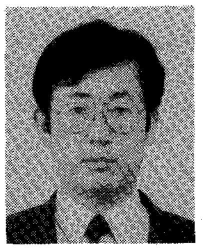
Mr. Motomura is a member of the Institute of Electronics, Information and Communication Engineers of Japan.



Hachiro Yamada was born in Shizuoka, Japan, on June 6, 1949. He graduated from Shizuoka Technical High School in 1968 and 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 content addressable memories, DSP's, vector processor LSI's, high-speed circuit technologies, and VLSI architecture. He is presently a Research Manager in the Microelectronics Research Laboratories, NEC Corporation.

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



Tadayoshi Enomoto (M'80) received the B.S.E.E. degree from Nihon University, Tokyo, Japan, in 1968, and the M.Sc. and Ph.D. degrees from the Department of Electrical Engineering, Ohio State University (OSU), Columbus, in 1972 and 1975, respectively. As a graduate research associate at OSU, he studied photoconductivity and luminescence in compound materials.

He joined NEC Corporation in 1968. In 1970 he was awarded a four-year Ohio State University Fellowship which covered all his expenses from 1970 to 1975 and permitted him to complete studies toward the M.Sc. and Ph.D. degrees. From 1975 to 1982, while he was with the Electronic Device Research Laboratory of NEC Central Research Laboratories, he worked on various analog MOS LSI's, including CCD's, switched-capacitor filters, adaptive equalizers, etc. for telecommunication applications. He also established a scaling rule for MOSFET analog circuits. From 1982 to 1986, while he was with the Ultra-LSI Research Laboratory of NEC Microelectronics Research Laboratories, he worked on two different types of CMOS multilayer IC's. One was an inter-CMOS hybrid stacked IC whose technology was named "elemental level vertically integrated circuit" (ELVIC) technology. The other was a monolithic-type stacked IC using recrystalized polysilicon with a silicon-on-insulator (SOI) structure. Since 1984 he has contributed extensively to works on CMOS and BiCMOS video signal processor (VSP) LSI's, including in 1987 the world's first VSP with full video signal processing function, in 1989 a 200-MHz 16-b super-high-speed signal processor (SSSP) core, and in 1991 a 250-MHz 16-b super-high-speed VSP (S-VSP). He was also involved in the development of various CMOS and BiCMOS ULSI's, such as the world's first 200-MFLOPS 64-b vector pipelined processor (VPP) ULSI, a dictionary search processor (DISP) ULSI, etc. Since 1986 he has been the Director of the System ULSI Research Laboratory of NEC Microelectronics Research Laboratories, Sagamihara, Japan, and his current research interests include the development of various types of CMOS and BiCMOS LSI's, including microprocessors, digital signal processors (DSP's), VSP's, DRAM's, SRAM's, CAM's, analog LSI's, and future ULSI's.

Dr. Enomoto is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan. He served as a Secretary of the technical group on Integrated Circuit and Devices, IEICE of Japan, from 1987 to 1991 and is presently the Vice President of that same group.