Data Compression by Hardware PEM (Pattern Extraction Method) Using Multi Processor Elements

TOMOYUKI KAWAMURA*

Many methods for data compression aiming at high speed processing and realtime processing have been developed.

Previously, we proposed a data compression method by automatic pattern extraction (PEM; Pattern Extraction Method), which improved compression ratio only. However, PEM wastes much CPU time for data compression.

In this paper, HPEM (Hardware PEM) is developed for high speed parallel processing of data compression from a modified PEM algorithm. A HPEM system consists of several thousands of processor elements (PE's) connected in cascade, and each PE contains about 1000 transistors. The HPEM system has a data compression speed of about 600 times faster compared with a program of the PEM algorithm on a 1 MIPS computer.

The architecture of the HPEM hardware and the evaluation of the HPEM system by simulation are reported.

1. Introduction

Many data compression algorithms have been developed as Huffman code [1] and Incremental compression code [2] for example. These algorithms aim at high speed and realtime data processing. Previously, we proposed a data compression algorithm PEM (Pattern Extraction Method) [3] intended to improve the compression ratio only. However, PEM wastes huge processing time in comparison with other algorithms.

In this paper, HPEM (Hardware PEM) is proposed which consists of hardware circuits and a modified PEM algorithm. The HPEM experiment in this paper, was simulated by software.

2. Outline of the PEM algorithm

The PEM algorithm is shown below briefly. PEM automatically determines a pattern from a text to be replaced with a compression-code by making up a PEM tree. This pattern is called S-pattern. The patterns then are replaced with new compression-codes, and the pattern is appended to the end of the text, and delimiter-code (described by ⟨128⟩ or $ in this paper) is appended to the text for separation of each S-pattern. The process shown above is repeated until no S-pattern is found in a PEM tree. Finally, a compressed text is obtained.

A small example is shown in Fig. 1. The first line shows the original text, and the last line is the compressed text. Arrows indicate replacements of the S-patterns with compression-codes. In Fig. 2, the whole PEM tree corresponding to the first line in Fig. 1 is shown.

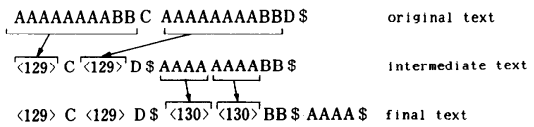

Fig. 1 Process of compression by PEM.

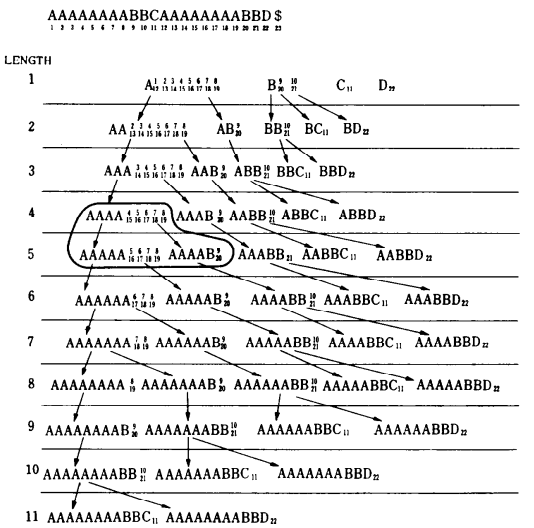

Fig. 2 Example of a PEM tree.

In the actual program of PEM, PEM makes up a partial PEM tree by cutting branches. Therefore, CPU time becomes shorter to compute a S-pattern in com-

*Department of Information Electronics, Tokuyama Technical College.

parison with making a whole PEM tree. The actual program uses list structure to represent the PEM tree, and executes a garbage collection to reuse list-cells which become unused.

A compressed text by the PEM algorithm can be decoded by a simple recursive procedure described in the paper [3]. And this decoding procedure can be executed at extremely high speed compared with the coding (=compression) procedure.

3. The algorithm of HPEM

3.1 Outline of the algorithm of HPEM

PEM is a program of list processing to make up PEM trees, while HPEM does not perform list processing and will be implemented by hardware.

(1) Outline of the hardware of HPEM.

The outline of the hardware of HPEM is illustrated in Fig. 3. MEM is a memory system keeping the TEXT to be compressed by HPEM. MP(main processer) is a CPU controlling the whole HPEM system. One MP is installed in each HPEM system. PE(processor element) is a circuit corresponding to a character-code in the TEXT compressed by HPEM. Character-code includes compression-code in this paper. The number of PE's installed in a HPEM system must be greater than the number of character-codes in the TEXT in the MEM.

The symbol ← in Fig. 3 represents 3 control signals from the MP to the PE's. The symbol ⇔ represent data lines between the PE's or between the PE and the MP. The symbol ⟺ represents a external bus to access the character-codes in the TEXT in the MEM.

(2) Determination of S-pattern

HPEM makes a PEM tree by breadth-first search as PEM. HPEM forms all patterns (length $n-1$) to get the next patterns (length n), while PEM forms meaningful patterns (length $n-1$) to get next patterns (length n). In other words, HPEM counts every pattern of length n in the TEXT, and the pattern which appears most frequently is the S-pattern for the length n. It means the counts of almost all patterns (length n) are 1 at a large n (for example $n=10, 11, 12, \ldots$) except for S-pattern, this algorithm needs much CPU time. The computation time needed for the HPEM algorithm to determine a S-pattern on a sequential execution computer, is proportional to the square of the length of the TEXT.

HPEM realizes a high speed processing by counting the number of appearances of patterns with parallel operation of the PE's. In HPEM, a PE corresponds to a character-code. Parallel degree of a HPEM system is equal to the number of character-codes in the TEXT. As a result, the computation time of HPEM is proportional to the length of the TEXT.

(3) Replacement of S-patterns

HPEM replaces the S-patterns with compression-codes by sequential processing. PEM does it by the same way.

(4) Decoding

The decoding algorithm of PEM (using a recursive procedure) is useful for HPEM. HPEM does not include a decoding algorithm.

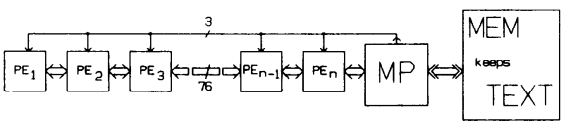

Fig. 3 Outline of HPEM.

3.2 PE (processor element)

The outline of a PE circuit is shown in Fig. 4. A, B, C are control signals from the MP to the PE's, and these signals operate to all the PE's simultaneously.

A quadrangle in this figure indicates a register. A register contains a character-code, a hash value or a counter value. Each value in the registers shifts on the PE's by the signals.

A circle (represented by "( )" hereafter) indicates a operator. "(R)" is a 8 bit left rotate operator. It is easily implemented by crossing of 8 data lines over other lines. "(+)" is a exclusive-or operator. It operates on each pair of bits from 2 inputs. The input from above contains 10 bits, and the input from left contains 25 bits. Therefore 15 significant bits from left input pass this operator. The combination of "(R)" and "(+)" realizes a hashing operator. "(=)" is comparator. The output to right is 1 when 2 inputs are equal, otherwise it is 0. "(+1)" is an incrementer. The output to right is the sum of the inputs from left and above.

A triangle is a gate which is open when the control signal is 1, otherwise it is closed.

The CH register contains a character-code. HASH and P_HASH registers contain a hash value represeting a pattern. The P_CNT register contains a counter corresponding to a pattern represented by P_HASH. The connected PE's behave like a large shift register, get data from the MP, and put data to the MP.

(1) Control signal A

The signal A makes each CH_i in PE_i have the value of CH_{i+1} in PE_{i+1}. And, CH in the PE connected to the

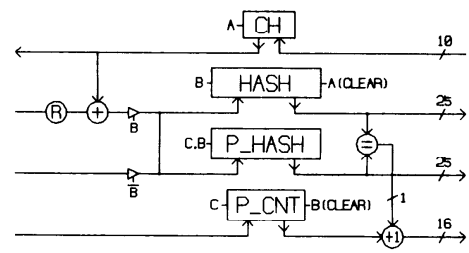

Fig. 4 Outline of a PE.

MP have the character-codes in the MEM. Also, the signal A clears each HASH register. The MP sends the signal A to as many PE's as the number of character-codes in the TEXT.

(2) Control signal B

The signal B makes each $HASH_i$ & P_HASH_i have the hash value calculated by CH_i and $HASH_{i-1}$. Also, the signal B clears each P__CNT register. The n issues of the signal B makes up a hash value to represent a pattern corresponding to character-codes from CH_{i-n+1} to CH_i, and $HASH_i$ & P_HASH_i have this hash value. Hereafter, this hash value is regarded as the corresponding pattern. The conflict of the hash values is argued in chapter 5.

(3) Control signal C

The signal C makes each P_CNT_{i+1} have the value of P_CNT_i+1 *if* $P\_HASH_i=HASH_i$, otherwise the value of P_CNT_i. Also, the signal C makes each P_HASH_{i+1} have the value of P_HASH_i.

A pair of P_HASH_i and P_CNT_i is called a $PACKET_i$. After many issues of the signal C, a $PACKET_i$ reachs the MP whose P_CNT_i indicates how many HASH's between PE_i and the MP have the same hash values as P_HASH_i.

3.3 Reception of PACKET's by the MP

The MP is a general CPU that has an additional facility to handle the PE's. The MP has circuits, shown in Fig. 5, to process PACKET's from the PE connected to the MP by the signal C, and the MP uses P__CNT only in the PACKET. This circuit works to get the number and the position of patterns which appear most frequently in the TEXT at a certain pattern-length of a PEM tree. The position of patterns is represented by the right most position of the patterns in the TEXT.

Register M__CNT keeps the maximum value of P_CNT_i from the beginning of the signal C. Register M__POS keeps the position of the pattern (P_HASH_i) corresponding to the M__CNT in the TEXT. Register POS indicates positions of P_HASH_i in $PACKET_i$ received from the PE. Initially, the POS has the number of character-codes in the TEXT. "(>=)" is a comparator. It outputs 1 if the left input >= the right input, otherwise 0. "(−1)" is a decrementer by 1.

3.4 Determination of S-pattern

The procedure to determine the S-pattern by the hardware described above is shown in Fig. 6. This procedure is executed on the MP and named TREE, because it is corresponding to the procedure making up a PEM tree in PEM. SIGNAL (X) means a issue of control signal X. This procedure assumes that each CH_i has each character-code of the TEXT before this procedure is executed.

TXTLENGTH indicates the length of the TEXT, and is equal to the number of CH's which have meaningful values. LENGTH indicates a length of patterns on processing. MEASURE(X, Y) is a function which

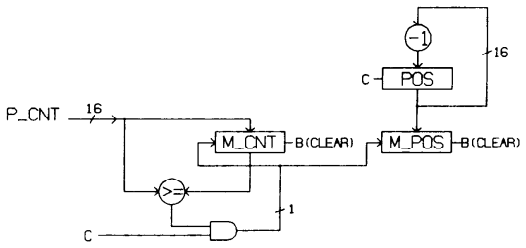

Fig. 5 Circuit for M__CNT & M__POS.

```
procedure TREE;
begin
  BEST:=0;
  BESTPOS:=0;
  BESTLENGTH:=0;
  SIGNAL(B);
  LENGTH:=1;
  loop
    POS:=TXTLENGTH;
    SIGNAL(B);
    LENGTH:=LENGTH+1;
    for I:=LENGTH to TXTLENGTH do SIGNAL(C);
    LIM:=MEASURE(M_CNT,MAXLENGTH);
    CUR:=MEASURE(M_CNT,LENGTH);
    if LIM <= BEST then return;                    (**)
    if CUR >= BEST then
      begin
        BEST:=CUR;
        BESTPOS:=M_POS;
        BESTLENGTH:=LENGTH;
      end;
    if LENGTH = MAXLENGTH then return;
  loopend;
end;
```

Fig. 6 Procedure TREE to make up a PEM tree.

calculates the profit obtained to replace patterns with compression-codes when the number of the patterns is X and the length of the patterns is Y. Higher value of this function indicates better profit. The value of this function less than or equal to 0 means no profit by the replacement. When $X=1$, MEASURE(X, Y) must have a value less than 0. An example is shown below.

$$\text{MEASURE}(X, Y)=X*Y-(X+Y+1)$$

MAXLENGTH is the maximum length of patterns which is able to appear more than twice in the TEXT. This value is 100 for the present. LIM is the profit when LENGTH of the M__CNT patterns become MAXLENGTH. This is the best profit that may be obtained from the M__CNT patterns. CUR is the profit when the M__CNT patterns whose length is LENGTH are replaced with the compression-code. BEST indicates the best value of CUR from the starting of the procedure TREE. BESTPOS and BESTLENGTH keep the position and the length of the pattern corresponding to BEST. BESTPOS and BESTLENGTH indicate the S-pattern when the procedure TREE has finished.

Figure 7 shows the flows of information by the procedure TREE for a text "xyzwyz". $\Rightarrow$ indicate the flows of HASH's by the signal B. The symbol $\rightarrow$ indicate the flow of PACKET's by the signal C. Alphabetic characters on $\Rightarrow$ and $\rightarrow$ indicate the patterns in HASH and P__HASH respectively. Numeric characters on $\rightarrow$ indicate the value of P__CNT. $\Rightarrow$ and $\rightarrow$ without characters indicate the flow of meaningless

information.

3.5 Replacement of S-patterns

After the procedure TREE finishes, HPEM replaces the S-pattern in the TEXT in the MEM with the compression-codes. At the same time, the new TEXT is sent from the MP to the PE's by the signal A for the next TREE process. The MP has the circuit in Fig. 8 to execute the processing above at high speed.

The MP makes PATTERN in Fig. 8 have the string of

TEXT[BESTPOS − BESTLENGTH + 1..BESTPOS]

that is the S-pattern. "(COMPARATOR WITH MASK)" compares BESTLENGTH character-codes input from above and below, and outputs 1 when they match, otherwise 0. CC is the compression-code. TXTLEN is the length of the TEXT before the replacement, 0 after the replacement. TXTLEN1 is 0 before the replacement, the length of the new TEXT after the replacement.

The circuit in Fig. 8 is the circuit when MAXLENGTH is 5. Character-codes in the MEM are sent to C_5 sequentially. When BESTLENGTH < MAXLENGTH, the character-codes are sent to C_4, C_3 or C_2 by DECODER in Fig. 8.

The signal A in Fig. 8 is the same as the signal A in Fig. 6. This circuit works by the signals A and D, and terminates when TXTLEN reaches 0. The signal D works only to move the character-codes from the MEM to C_i. The MEM sends delimiter-code⟨128⟩ if the MP requires character-codes over the length of the TEXT in the MEM. The processed character-codes are sent to the PE's and sent-back to the MEM sequentially by the

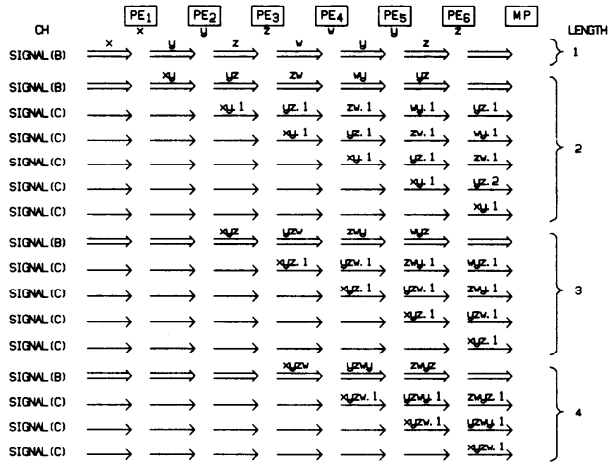

Fig. 7 Flow of information on HPEM.

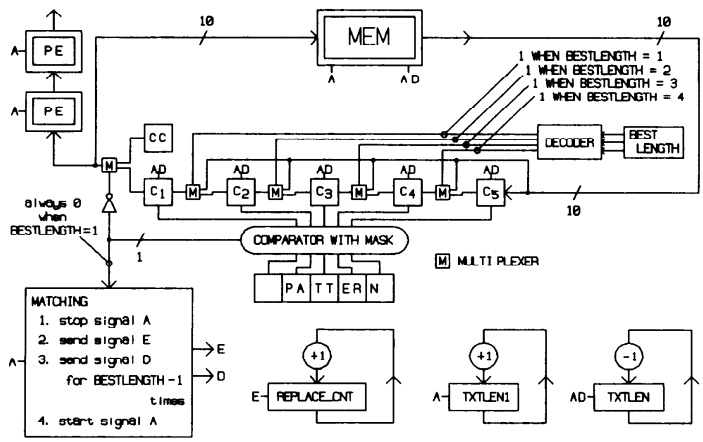

Fig. 8 Circuit for replacement.

signal A.

The procedure COMPACTION to replace the S-patterns is shown in Fig. 9. This procedure is executed on the MP.

3.6 Main program of HPEM

The main program of HPEM is shown in Fig. 10. This program is executed on the MP. The termination condition (*1) means that no S-pattern brings profit. This condition is a good condition expected for HPEM. The termination condition (*2) means that the S-pattern contains the delimiter-code. This condition is a bad condition for HPEM, because the pattern including the delimiter-code can not be replaced with a compression-code. The termination condition (*3) detects a infinite loop when 2 different patterns which have same hash values are selected as the S-pattern, and this state does not change after the procedure COMPACTION.

4. Improvements of HPEM

4.1 Improvement of termination conditions

The termination condition (*2) in Fig. 10 is improved by a simple procedure below.

IMPROVEMENT 1

When the S-pattern determined by the procedure TREE includes the delimiter-code, the main program substitutes the longest substring not including the delimiter-code in the S-pattern for the original S-pattern. For example, "AB" is substituted for "AB⟨128⟩C". If the length of the new S-pattern is 1, then HPEM terminates.

4.2 Partial exclusion of delimiter-code

Even if the IMPROVEMENT 1 is executed, the S-pattern "␣⟨128⟩" or "⟨128⟩␣" is selected by the procedure TREE to terminate HPEM unexpectedly, after many repetitions of the procedure COMPACTION. This problem is improved by the procedure below.

IMPROVEMENT 2

When the MP receives a P__HASH whose pattern-length is 2 and the P__HASH includes the delimiter-code explicitly (for example, "␣⟨128⟩", "A⟨128⟩" or "⟨128⟩B"), the circuit in Fig. 5 does not update the M__CNT or the M__POS. For this improvement, a small circuit must be added to the circuit in Fig. 5. P__HASH whose pattern-length is 2 can indicate the inclusion of the delimiter-code, if the following conditions are satisfied.

1: width of rotate (8) $>\log_2$ delimiter-code (7)
2: width of rotate *2 (16) $\leqq$ width of HASH (25)

4.3 Complete exclusion of the delimiter-code (IMPROVEMENT 3)

Even if IMPROVEMENT 1 and IMPROVEMENT 2 are excecuted, the compression ratio of HPEM is slightly worse than PEM. Because, HPEM includes the delimiter-codes in S-patterns whose length are greater than 2, while PEM completely excludes the delimiter-codes on making PEM trees.

In IMPROVEMENT 3, the patterns including the delimiter-codes are completely excluded. This improvement is achieved by the circuit in Fig. 11 to be added to Fig. 4. In this figure, DPB_i is a one bit register indicating that the pattern corresponding to $HASH_i$ includes the delimiter-code, and P_DPB_i is a one bit register indicating that the pattern corresponding to P_HASH_i includes the delimiter-code. P_DPB_i is a part of $PACKET_i$.

On sending CH from the MP to the PE by the signal A, if CH is the delimiter-code then DPB is 1, otherwise 0. And, DPB_{i+1} is shifted to DPB_i by the signal A.

On calculating hash values by the signal B, DPB_i is shifted to DPB_{i+1}. Once DPB_i becomes 1, OR operator in Fig. 11 keeps DPB_i 1 until a signal A.

P__DPB reaches the MP by the signal C as a part of PACKET. Some circuits must be added to Fig. 5 so that no update is executed in Fig. 5 when the P__DPB is 1.

The circuit in Fig. 11 seems complex compared with

```
procedure COMPACTION;
begin
  set PATTERN and MASK in Fig.8;
  for I:=1 to BESTLENGTH do SIGNAL(D);
  TXTLEN:=TXTLENGTH;
  TXTLEN1:=0;
  CC:=CC+1;
  repeat
    SIGNAL(A);          ( including SIGNAL(D) )
  until TXTLEN = 0;
  send PATTERN and <128> to MEM;
  TXTLENGTH:=TXTLEN1+BESTLENGTH+1;
end;
```

Fig. 9 Procedure COMPACTION for replacement.

```
program HPEM;
begin
  set TXTLENGTH;
  BESTLENGTH:=1;
  COMPACTION;       ( setting a initial text to the PE's )
  OLD_REPLACE_CNT:=-1;
  CC:=128;
  loop
    TREE;
    if BEST <= 0 then exit;                                    (*1)
    if ( the replacing pattern includes <128> ) then exit;     (*2)
    COMPACTION;
    if (REPLACE_CNT = 1) and (OLD_REPLACE_CNT = 1) then exit;  (*3)
    OLD_REPLACE_CNT:=REPLACE_CNT;
  loopend;
end.
```

Fig. 10 Main program for HPEM.

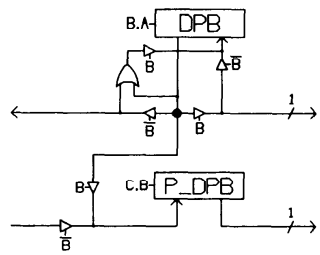

Fig. 11 Circuit to detect the delimiter-code.

Fig. 4. However, the width of the registers and the data lines are all 1. Therefore the circuit size is small.

The termination condition (*2) in Fig. 10 can be deleted, because the S-pattern determined by the procedure TREE does not include the delimiter-code at all by the IMPROVEMENT 3.

5. Conflict of hash values

HPEM substitutes hash values for patterns to compare the patterns themselves. Different patterns may have the same hash values creating a conflict of hashing. However, the TEXT in the MEM always is kept valid as an intermediate text of HPEM, because all the replacements of S-patterns in the TEXT are done by the circuit in Fig. 8.

6. Evaluation

6.1 Function MEASURE

4 functions to calculate the profit are described in the paper [3]. In this experiment, the following function is used.

MEASURE(PN, PL)=(PN*PL)−(PN*CL+PL+DL)

PN: the number of S-patterns (≧1)
PL: the length of S-patterns (≧2)
CL: the length of compression-code (1 or 2)
DL: the length of the delimiter-code (always 1)
CL: the length of compression-code (1 or 2)
DL: the length of the delimiter-code (always 1)

6.2 Correction of the number and the length of patterns

M_CNT in the function MEASURE in Fig. 6 does not indicate the number of pure appearances of patterns. That is, the M—CNT indicates the number of patterns including overlaps. For example, a text "AAAA" includes 3 patterns "AA", and the number of pure appearances "AA" in "AAAA" is 2. HPEM uses the M_CNT instead of the number of pure appearances, because HPEM can not calculate it until the procedure COMPACTION finishes.

LENGTH in the function MEASURE had better be corrected as described in the paper [3]. That is, LENGTH may have a slightly greater value than the value in Fig. 6, because the S-pattern may include 2 byte character-codes. 2 byte character-codes are created when the compression-code (CC) becomes greater than 255. Some calculation is needed per loop in Fig. 6 to correct the LENGTH. In this experiment, this correction is not implemented.

Table 1 Comparison of HPEM with other methods.

	PRG1 (1510 bytes)			PRG2 (4712 bytes)		
method	ratio	time (sec)	#CC	ratio	time (sec)	#CC
Huffman code	0.612	0.149	—	0.607	0.205	—
Incremental code	0.630	0.203	—	0.509	0.511	—
PEM	0.429	39	50	0.294	234	123
HPEM	0.437	0.075	40	0.337	0.480	64
Improvement 1	0.437	0.075	40	0.337	0.480	64
Improvement 2	0.430	0.081	50	0.2961	0.566	123
Improvement 3	0.430	0.081	50	0.2958	0.565	123

MAXLENGTH=100
Bit width of hash values=31

6.3 Effect of the IMPROVEMENT 1, 2, 3

The results of the IMPROVEMENT 1, 2, 3 are shown in Table 1 compared with Huffman code, Incremental code, PEM and HPEM for 2 sample texts (PRG1 1510 bytes and PRG2 4712 bytes). "ratio" in Table 1 is the final compression ratio of the length of the final text to the length of the original text. The length of the final text is corrected to increase slightly, if the text includes 2 byte character-codes. "time" for Huffman code, Incremental code and PEM is CPU time on a MC68000 (8 MHz) micro processor. "time" for HPEM is calculated by the expression below.

CN * 100 nano sec+AN * 400 nano sec

CN=the number of issues of the signal C+ 100 * the number of execution of loop in Fig. 6.
AN=the number of issues of the signal A.
100 nano sec=the interval of signal C.
400 nano sec=the interval of signal A.

The other parts of Fig. 6, 9 and 10 are ignored, because the execution time for these is small enough to be omitted. "#CC" is the number of compression-codes created by the procedure COMPACTION in Fig. 9.

6.4 Effect for variety of MAXLENGTH

A larger MAXLENGTH is better for the final compression ratio in the HPEM algorithm. While, smaller MAXLENGTH causes the return condition(**) in Fig. 6 becomes true sooner, and shortens the execution time of HPEM. The results of the experiments for various MAXLENGTHs are shown in Table 2. "max PL" is the maximum length of the S-patterns. "mean PL" is the mean length of the S-patterns.

6.5 Bit width of hash values

Setting the bit width of hash values smaller causes the final compression ratio to get worse, because the number of the conflicts of hash values is increasing. The relation between the bit width of hash values and the final compression ratios are shown in Table 3. "width" is the bit width of hash values. "conflict" is the number of conflicts of hash values. 25 bit width is enough to execute HPEM for about 5 K byte text, compared with PEM.

Table 2 Relation between MAXLENGTH and final compression ratio.

	PRG1 (1510 bytes)					PRG2 (4712 bytes)				
MAXLENGTH	ratio	time(sec)	#CC	max PL	mean PL	ratio	time(sec)	#CC	max PL	mean PL
100	0.430	0.081	50	61	7.5	0.296	0.565	123	42	6.4
50	0.431	0.077	51	50	7.1	0.296	0.481	123	42	6.4
30	0.436	0.073	52	30	6.9	0.298	0.425	123	30	6.0
20	0.434	0.064	53	20	6.3	0.296	0.357	123	20	5.8
10	0.438	0.052	53	10	5.4	0.300	0.269	123	10	5.1

Bit width of hash values=31

Table 3 Relation between bit width of hash values and final compression ratio.

	PRG1 (1510 bytes)				PRG2 (4712 bytes)			
width	ratio	time (sec)	#CC	conflict	ratio	time (sec)	#CC	conflict
31	0.434	0.064	53	0	0.296	0.357	123	0
25	0.434	0.064	53	0	0.296	0.357	123	0
23	0.434	0.064	53	0	0.299	0.360	123	4
21	0.572	0.032	15	3	0.373	0.246	53	6
19	0.574	0.033	15	3	0.394	0.217	46	7

MAXLENGTH=20

6.6 Implementation of the PE by VLSI

A PE in Fig. 4 can be composed of about 1000 transistors, and about 500 PE's can be installed on one chip by current VLSI technique [4]. A large scale HPEM system, composed of several thousands PE's, can be made by cascading connections of several PE VLSI chips.

By the way, the number of transistors for the circuits in Fig. 5 and 8 is no problem for the HPEM system, because only one set of these circuits is installed on the MP.

7. Conclusion

The simulation of HPEM shows that HPEM achieves similar compression ratio to PEM, and about 600 times higher speed execution in comparison with PEM executed on a 1 MIPS computer. And the number of transistors of a PE is considerably smaller so that the HPEM hardware can be made with current VLSI technique.

References

1. MIYAGAWA, H. and SHIMABARA, H. and IMAI, H. The theory of information and code, p. 266, Iwanami shoten, Tokyo, Japan (1982).
2. Japan UNIVAC, Compress of 1100, p. 32, Japan UNIVAC, Tokyo, Japan (1983).
3. KAWAMURA, T. New Data Compression Method by Automatic Pattern Extraction, *Trans. IPS Japan* **25**, 6 (1984), 1089-1094.
4. MEAD, C. and CONWAY, L. INTRODUCTION TO VLSI SYSTEMS, Addison-Wesley Publishing Company, *Inc., Reading, Massachusetts*, U.S.A. (1980).

(Received May 7, 1985; revised October 7, 1986)