Intraword Bit Operations

EIITI WADA*

Introduction

This article describes two operations which seem likely to be effective if built in a binary digital computer, the circuits which execute the above operations and, finally, the problem of size normalization for one-dimensional pattern recognition as an application of proposed operations. The two operations are

1. that which finds the number of place to normalize a word,
2. that which separates a bit configuration in a word according to the bit configuration of another word.

Sections 1 and 2 give explanations of each of the above operations and section 3 gives an example.

1. *A shift Number Detector*

1.1. *Purpose*

Suppose a fraction a which consists of m $(2 \leq m)$ places of binary digits a_i $(i=0, 1, \cdots, m-1$ and $a_i=0$ or $1)$ and is defined by

$$a=\sum_{i=1}^{m-1} 2^{-i}a_i - a_0, \tag{1}$$

is normalized by shifting to the left by c $(0 \leq c < m)$ places. Here we assume that c is represented by n $(n=1+[\log_2(m-1)]$, where [ ] means to take the largest integer not greater than the enclosed value) places of binary digits c_k $(k=0, 1 \cdots, n-1; c_k=0$ or $1)$ as

$$c=\sum_{k=0}^{n-1} 2^k c_k. \tag{2}$$

A detector which receives a_i and outputs c_k is considered.

1.2. *Circuit*

The shift number c is invariable if the entire bit pattern is transformed by replacing 0 by 1 and 1 by 0. Therefore the detector is designed to receive the original number if $a<0$ and its one's complement if $a>0$, so that input to the detector has the same boundary pattern of strings of 0 and 1 with a given number, but always starts with 1 at the left end.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 3, No. 5 (1962), pp. 247-254.

* Data Processing Center, Onoda Cement Co., Ltd., Tokyo.

Let b be an input to the detector for a. Then

(3) $b=a$ if $a<0$ or

(4) $b=-a-2^{-(m-1)}$ if $a>0$,

and each bit of b is given by

(5) $b_0=1$

(6) $b_j=1\oplus a_0\oplus a_j$ ($j=1, 2, \cdots, m-1$. $\oplus$ is 'exclusive or' notation).

As b_0 has no information, further consideration will be made only for b_j ($j=1, 2, \cdots, m-1$). Now, since c_0 is to become 1 if the shift number is 1, 3, 5, $\cdots$, i.e. odd, so $c_0=1$ if $b_1=1$ and $b_2=0$, or if $b_1b_2b_3=1$ and $b_4=0$, or if $b_1b_2\cdots b_5=1$ and $b_6=0$, or$\cdots$. Next, since c_1 is to become 1 if the shift number is, for example, 2, 3, 6, 7, $\cdots$, i.e. the quotient of the number is odd if divided by 2, so $c_1=1$ if $b_1b_2=1$ and $b_3b_4=0$, or if $b_1b_2\cdots b_6=1$ and $b_7b_8=0$, or $\cdots$.

Thus, c_k is primarily given by

(7) $c_0=b_1$ for $m=2$,

(8) $c_0=b_1\bar{b}_2$ for $m=3$,

(9) $c_0=b_1\bar{b}_2+b_1b_2b_3$ for $m=4$,

(10) $c_0=b_1\bar{b}_2+b_1b_2b_3\bar{b}_4$ for $m=5$, $\cdots$,

(11) $c_1=b_1b_2$ for $m=3, 4$,

(12) $c_1=b_1b_2\overline{b_3b_4}$ for $m=5, 6$, $\cdots$,

and generally

(13) $c_0=b_1b_2+b_1b_2b_3\bar{b}_4+\cdots+b_1b_2\cdots b_{m-1}$ for odd $m-1$,

(14) $c_0=b_1\bar{b}_2+b_1b_2b_3\bar{b}_4+\cdots+b_1b_2\cdots b_{m-2}\overline{b_{m-1}}$ for even $m-1$,

(15) $c_1=b_1b_2\overline{b_3b_4}+b_1b_2\cdots b_6\overline{b_7b_8}+\cdots+b_1b_2\cdots b_{2[m-1/2]}$ for odd $\left[\dfrac{m-1}{2}\right]$,

(16) $c_1=b_1b_2\overline{b_3b_4}+b_1b_2\cdots b_6\overline{b_7b_8}+\cdots$
$+b_1b_2\cdots b_{2[m-1/2]-2}\overline{b_{2[m-1/2]-1}b_{2[m-1/2]}}$ for even $\left[\dfrac{m-1}{2}\right]$,

(17) $c_k=b_1b_2\cdots b_{2^k}\overline{(b_{2^k+1}b_{2^k+2}\cdots b_{2\cdot 2^k})}+b_1b_2\cdots$
$b_{3\cdot 2^k}\overline{(b_{3\cdot 2^k+1}b_{3\cdot 2^k+2}\cdots b_{4\cdot 2^k})}+\cdots+b_1b_2\cdots b_{2^k[m-1/2^k]}$ for odd $\left[\dfrac{m-1}{2^k}\right]$,

(18) $c_k=b_1b_2\cdots b_{2^k}\overline{(b_{2^k+1}b_{2^k+2}\cdots b_{2\cdot 2^k})}+b_1b_2\cdots$
$b_{3\cdot 2^k}\overline{(b_{3\cdot 2^k+1}b_{3\cdot 2^k+2}\cdots b_{4\cdot 2^k})}+\cdots+b_1b_2\cdots$
$b_{2^k([m-2/2^k]-1)}\overline{(b_{2^k[m-1/2^k]-2^k+1}b_{2^k[m-1/2^k]-2^k+2}\cdots b_{2^k[m-1/2^k]})}$
for even $\left[\dfrac{m-1}{2^k}\right]$.

On the other hand, for the opposite end of c_k's, since

$$\left[\frac{m-1}{2^{n-1}}\right]=1, \quad \text{and} \quad \left[\frac{m-1}{2^{n-1}}\right]=2 \quad \text{or} \quad 3,$$

(19) $c_{n-1}=b_1b_2\cdots b_{2^{n-1}}$,

(20) $c_{n-2}=b_1b_2\cdots b_{2^{n-2}}\overline{(b_{2^{n-2}+1}b_{2^{n-2}+2}\cdots b_{2^{n-1}})}$ if $\left[\frac{m-1}{2^{n-2}}\right]=2$,

or

(21) $c_{n-2}=b_1b_2\cdots b_{2^{n-2}}\overline{(b_{2^{n-2}+1}b_{2^{n-2}+2}\cdots b_{2^{n-1}})}+b_1b_2\cdots b_{3\cdot 2^{n-2}}$ if $\left[\frac{m-1}{2^{n-2}}\right]=3$.

Recursively replacing c_k which consists of the product of continuous 2^k numbers of b_j's and their negatives by

(22) $b_p^0=b_p$ and

(23) $b_p^{q+1}=b_{2p-1}^q b_{2p}^q$,

we obtain

(24) $c_0=b_1^0\bar{b}_2^0+b_1^0b_2^0b_3^0\bar{b}_4^0+\cdots$,

(25) $c_1=b_1^1\bar{b}_2^1+b_1^1b_2^1b_3^1\bar{b}_4^1+\cdots$, $\cdots$,

(26) $c_k=b_1^k\bar{b}_2^k+b_1^kb_2^kb_3^k\bar{b}_4^k+\cdots$, $\cdots$,

(27) $c_{n-2}=b_1^{n-2}\bar{b}_2^{n-2}$ if $\left[\frac{m-1}{2^{n-2}}\right]=2$,

or

(28) $c_{n-2}=b_1^{n-2}\bar{b}_2^{n-2}+b_1^{n-2}b_2^{n-2}b_3^{n-2}$ if $\left[\frac{m-1}{2^{n-2}}\right]=3$

and

(29) $c_{n-1}=b_1^{n-1}$.

Here, q in b_p^q indicates that q stages of 'and' operation have been taken.

So far for c_{n-1}, but other c_k's need more processing. Replacing again by

(30) $\overset{(1)}{b}{}_s^k=b_{2s-1}^k b_{2s}^k$ and

(31) $\overset{(1)}{b}{}_s^{k*}=b_{2s-1}^k \bar{b}_{2s}^k$

or

(32) $\overset{(1)}{b}{}^{k*}_{\frac{[m-1/2^k]+1}{2}}=b^k_{[m-1/2^k]}$ if $\left[\frac{m-1}{2^k}\right]$ is odd,

(33) $\overset{(t+1)}{b}{}_s^k=\overset{(t)}{b}{}_{2s-1}^k\overset{(t)}{b}{}_{2s}^k$,

(34) $\bigwedge\limits_{l=1}^{s-1}\overset{(t+1)}{b}{}_l^k\ \overset{(t+1)}{b}{}_s^{k*}=\bigwedge\limits_{l=1}^{s-1}\overset{(t+1)}{b}{}_1^k\ \overset{(t)}{b}{}_{2s-1}^{k*}(+\bigwedge\limits_{l=1}^{s-1}\overset{(t+1)}{b}{}_1^k\ \overset{(t)}{b}{}_{2s-1}^k\ \overset{(t)}{b}{}_{2s}^{k*})$,

we obtain

(35) $c_{n-2}=\overset{(1)}{b}{}^{n-2*}$ if $\left[\frac{m-1}{2^{n-2}}\right]=2$,

(36) $c_{n-2}=\overset{(1)}{b}{}^{n-2*}+\overset{(1)}{b}{}_1^{n-2}\overset{(1)}{b}{}_2^{n-2*}=\overset{(2)}{b}{}_1^{n-2*}$ if $\left[\frac{m-1}{2^{n-2}}\right]=3$,

and in the same way we obtain

$$c_{n-r}=\overset{(r-1)}{b_1^{n-r*}} \quad \text{if} \quad \left[\frac{m-1}{2^{n-r}}\right]=2^{r-1}, \tag{37}$$

$$c_{n-r}=\overset{(r)}{b_1^{n-r*}} \quad \text{if} \quad 2^{r-1}+1\leq\left[\frac{m-1}{2^{n-r}}\right]<2^{r}. \tag{38}$$

Here, t in $\overset{(t)}{b_s^{k*}}$ indicates that t stages of 'and' operation or '$b_{2s-1}^{k*}+b_{2s-1}^{k}b_{2s}^{k*}$' operation have been taken. Accordingly it is understood that $c_{n-r}=\overset{(r)}{b_1^{n-r*}}$ or $c_{n-r}=\overset{(r-1)}{b_1^{n-r*}}$ will be obtained after n or $n-1$ stages of operation from b_j respectively.

1.3. *Circuit design*

This paragraph shows how to design the detector using parametron. Throughout the above description, there are only 'and' operations of two variables, negations, simple delays (they occur in the operation (32) and (34) when parenthesized terms are omitted) and '$b_{2s-1}^{k*}+b_{2s-1}^{k}b_{2s}^{k*}$' operations. The last operation can be realized in a single stage as shown in Fig. 1, if a five input parametron is available. Therefore correct representation of described recursiveness simply leads to the detector circuits for $m=$

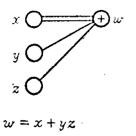


Fig. 1

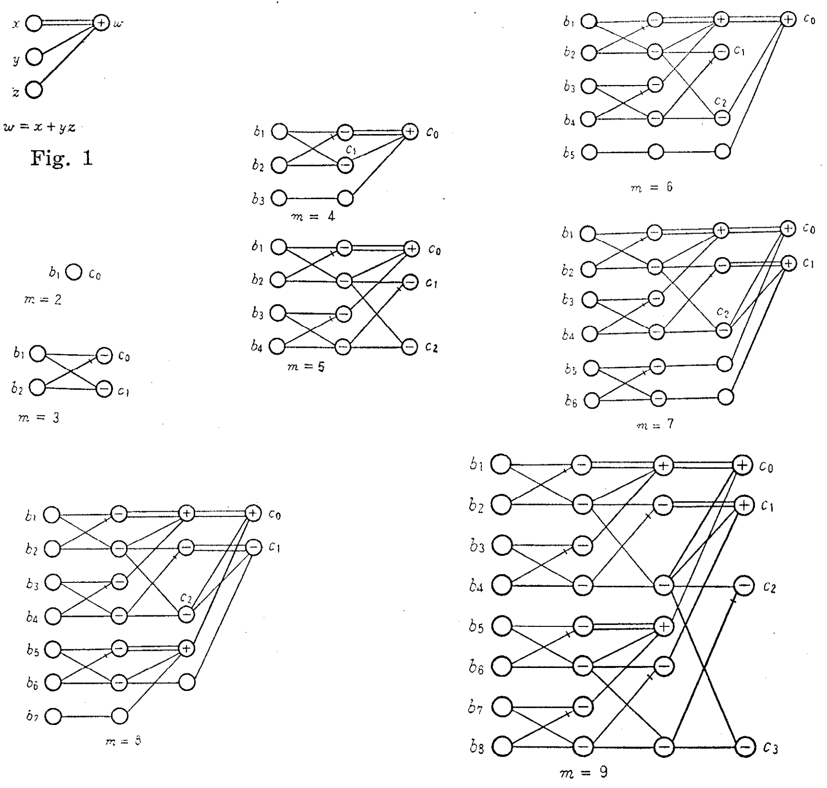


Fig. 2

2, 3, $\cdots$, 9 as shown in Fig. 2.

1.4. *Remarks on the number of stages*

As easily seen from the conditions of equations (37) and (38), and circuit diagrams, which have been shown in the preceding paragraph, the following rules can be used to get the number of stages for each bit of c.

Write down $m-1$ in binary form. Name each bit from the right end as $m_0, m_1, \cdots, m_g, \cdots, m_h, \cdots$. If right-left relation of position is taken into consideration, they look $\cdots, m_h, \cdots, m_g, \cdots, m_1, m_0$. Let the leftmost bit, which is 1, be m_h, then c is represented by $h+1$ places of binary digits $c_0, c_1, \cdots, c_h$, and c_h is obtained in h stages. Next, looking rightward from m_{h-1}, let $m_{h-1}=m_{h-2}=\cdots=m_{g+1}=0$ and $m_g=1$, i.e. let m_g be the first bit which is 1 to the right of m_h, then $c_{h-1}, c_{h-2}, \cdots, c_{g+1}$ are obtained in h stages and $c_g, c_{g-1}, \cdots, c_0$ in $h+1$ stages.

1.5. *High speed shift circuit*

We have seen that a detector can be constructed to find the shift number with a delay proportional to the logarithm of the word length. However, if the actual shifting is done in a conventional manner, 1 place at a time, there would be no point in designing such a detector. Shifting should be performed in a number of stages of $\log_2 n$. This end is achieved by the so-called high speed shift circuit, which is made of a tandem connection of 1 place shifter, 2 place shifter, 4 place shifter, $\cdots$, 2^f place shifter. It is important that shifters are connected in the decreasing order of the number of shifts performed, because c_0 cannot be obtained earlier than c_{n-1}, even if they may become ready for use simultaneously. Fig. 3 shows an example of the high speed shift circuit for $m=9$. To propagate the sign during the right shift, a_0 has so many fan-outs. But the problem of reducing the number of fan-outs can be treated independently of that of the logical design of normalizer in a logarithmic time. In the figure, l and r indicate control lines for the left shift and the right shift respectively, and the subscripts denote the number of places to be shifted.

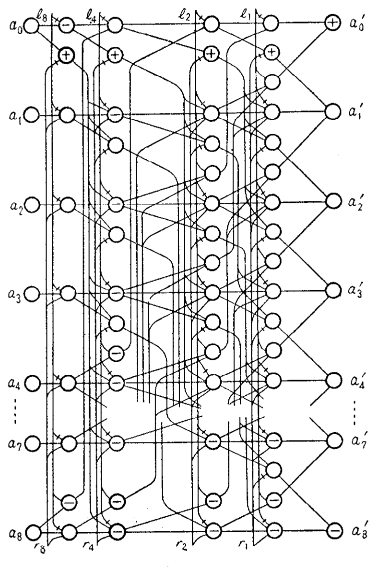


Fig. 3

2. *Bit Separator*

2.1. *Purpose*

The purpose of this circuit is first explained by an example. Suppose that two 10-bit words, C and B, are given as in Fig. 4 (C is referred to as the control word and B as the information word). Now, extract those bits in the word B which correspond to the bits 1 in C. The result would be D. The periods denote blank positions. Next, push these extracted bits together towards the left without changing their order (the result E). Extract next those bits in B whose corresponding bits in C are 0 (F). Push the extracted bits together towards the right (G). The two words in E and G are put together to form a complete word, which forms the final output (H). That is, the bit separator marks each bit of the information word by the corresponding bit of the control word and sifts out the bits left and right according to the mark. During sifting no succeeding bit gets ahead of the leading one.

C	0	1	1	0	0	0	1	1	1	1
B	1	0	1	0	0	1	0	0	0	1
D	·	0	1	·	·	·	0	0	0	1
E	0	1	0	0	0	1	·	·	·	·
F	1	·	·	0	0	1	·	·	·	·
G	·	·	·	·	·	·	1	0	0	1
H	0	1	0	0	0	1	1	0	0	1

Fig. 4

2.2. *Circuit*

Let the word length be m and each bit of the control and the information word be

$$c_0,\ c_1,\ \cdots,\ c_{m-1} \quad \text{and}$$
$$b_0,\ b_1,\ \cdots,\ b_{m-1}.$$

The whole operation is performed as a succession of simple exchanges of pairs of adjacent bits. Binary variables b_i^j and c_i^j are introduced to describe the results of successive stages of these exchange operations. Let the initial values be

$$\left.\begin{aligned} &(39) \qquad c_i^0=c_i \\ &(40) \qquad b_i^0=b_i \end{aligned}\right\} \quad (i=0,\ 1,\ \cdots,\ m-1).$$

Then each stage is defined as follows:
for an even exchange cycle

(41) $$a_{2s}^{2t}=\bar{c}_{2s}^{2t}\, c_{2s+1}^{2t},$$

(42) $$c_{2s}^{2t+1}=a_{2s}^{2t}\, c_{2s+1}^{2t}+\bar{a}_{2s}^{2t}\, c_{2s}^{2t},$$

(43) $$c_{2s+1}^{2t+1}=a_{2s}^{2t}\, c_{2s}^{2t}+\bar{a}_{2s}^{2t}\, c_{2s+1}^{2t},$$

(44) $$b_{2s}^{2t+1}=a_{2s}^{2t}\, b_{2s+1}^{2t}+\bar{a}_{2s}^{2t}\, b_{2s}^{2t},$$

(45) $$b_{2s+1}^{2t+1}=a_{2s}^{2t}\, b_{2s}^{2t}+\bar{a}_{2s}^{2t}\, b_{2s+1}^{2t}$$

$$\left(s=0,\ 1,\ \cdots,\ \left[\frac{m}{2}\right]-1,\ t=0,\ 1,\ \cdots\right),$$

and for an odd exchange cycle

(46) $$a^{2t+1}_{2s+1}=\bar{c}^{2t+1}_{2s+1}\,c^{2t+1}_{2s+2},$$

(47) $$c^{2t+2}_{2s+1}=a^{2t+1}_{2s+1}\,c^{2t+1}_{2s+2}+\bar{a}^{2t+1}_{2s+1}\,c^{2t+1}_{2s+1},$$

(48) $$c^{2t+2}_{2s+2}=a^{2t+1}_{2s+1}\,c^{2t+1}_{2s+1}+\bar{a}^{2t+1}_{2s+1}\,c^{2t+1}_{2s+2},$$

(49) $$b^{2t+2}_{2s+1}=a^{2t+1}_{2s+1}\,b^{2t+1}_{2s+2}+\bar{a}^{2t+1}_{2s+1}\,b^{2t+1}_{2s+1},$$

(50) $$b^{2t+2}_{2s+2}=a^{2t+1}_{2s+1}\,b^{2t+1}_{2s+1}+\bar{a}^{2t+1}_{2s+1}\,b^{2t+1}_{2s+2}$$

$$(s=0,\,1,\,\cdots,\left[\frac{m-1}{2}\right]-1,\ t=0,\,1,\,\cdots).$$

Above description shows that the whole exchange operation consists of alternation of odd exchange cycles and even exchange cycles. In even exchange cycles, every two adjacent bits are grouped starting from the leftmost bit; in odd cycles, every two adjacent bits are grouped except the leftmost bit. Bits in a group in the control word and bits in a corresponding group of the information word are exchanged whenever the pair in the control word is 01.

At the end of the m-th cycle, the output is the desired result. Let each bit of output word from the left be

$$h_0,\ h_1,\ \cdots,\ h_{m-1},$$

then

$$h_i=b_i^m \qquad (i=0,\,1,\,\cdots,\,m-1).$$

It is not difficult to see by inspection that the operation comes actually to an end in m cycles in the longest case.

Expression for c's can be rewritten in a much simpler form as follows: for an even exchange cycle

(51) $$c^{2t+1}_{2s}=c^{2t}_{2s}+c^{2t}_{2s+1},$$

(52) $$c^{2t+1}_{2s+1}=c^{2t}_{2s}\,c^{2t}_{2s+1},$$

and for an odd exchange cycle

(53) $$c^{2t+2}_{2s+1}=c^{2t+1}_{2s+1}+c^{2t+1}_{2s+2},$$

(54) $$c^{2t+2}_{2s+2}=c^{2t+1}_{2s+1}\,c^{2t+1}_{2s+2},$$

which can be realized by Fig. 5 circuit. An exchange circuit for one group is referred to as an AND/OR exchange circuit.

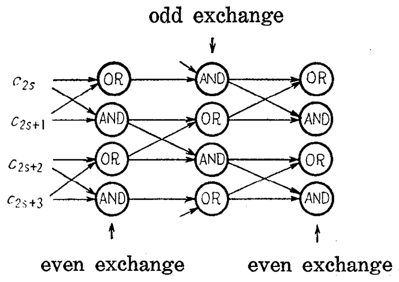


Fig. 5

2.3. *Circuit design*

A parametron AND/OR exchange circuit is shown Fig. 6. An array of exchange circuits of this type will perform shifting. However, if boundary condition is taken into account, even and odd cycles require the (a) and (b) connection of Fig. 7, respectively. Incidentally, a currently used parametron parallel shift register is very similar

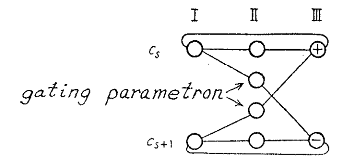


Fig. 6

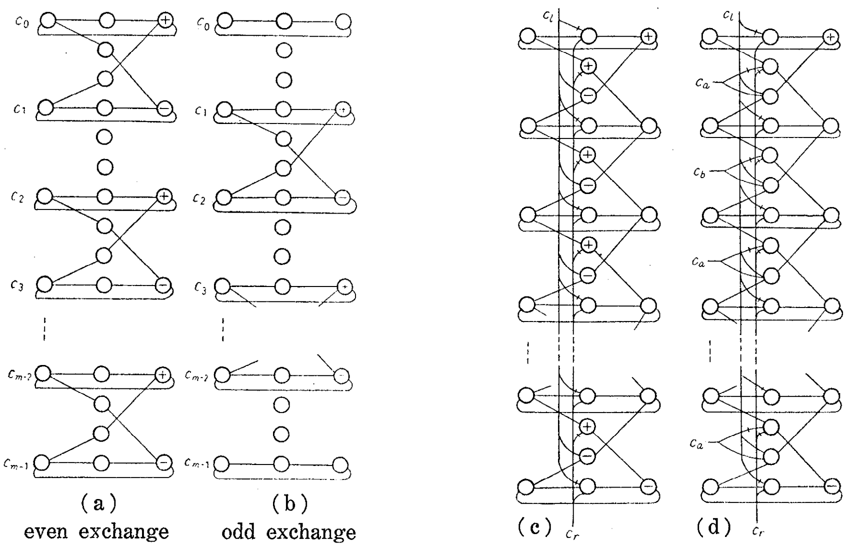


Fig. 7

to the AND/OR exchang circuit, and a modifiction of constant inputs into it yields the desired circuit. Fig. 7(c) is the ordinary parallel shift register, which shifts one place at a time. The 2^0 shifter of the aforementioned high speed shift register has the same structure. c_r and c_l, which are normally $-$, are right and left control lines. To shift right, c_r is switched to $+$; to shift left, c_l is switched to $+$. Now suppose that new control lines c_a and c_b are inserted into gating parametrons (cf. Fig. 6) in Fig. 7(d). When $c_a=c_b=-$, it is identical to Fig. 7(c), i.e. a parallel shift register. When $c_r=c_l=c_b=-$ and $c_a=+$, circuit of figure (d) acts as an even cycle AND/OR exchange circuit. Constants $c_r=c_l=c_a=-$ and $c_b=+$ provide exchange circuit for odd cycles. Similar technique can be applied for the information word with a slight modification. Parametron computers PC-1 and PC-2 at the University of Tokyo, each of which is equipped with two parallel shift registers like the above, seem easily to be modified for this operation.

2.4. *Remarks*

If the information word undergoes $m-1$ exchanges on all groups at each exchange cycle independent of the control word, the reversed word is obtained. That is

$$h_i = b_{m-1-i} \quad (i=0, 1, \cdots, m-1). \tag{55}$$

This is equivalent to the perfect 'Amitabha lottery' which gives completely symmetric destination to the starting point as drawn in Fig. 8.

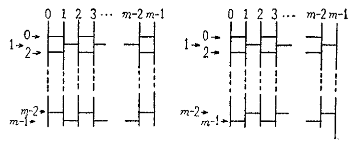


Fig. 8

To speed up the bit separation, many logical circuits which work in logarithmic time to the word length such as shift number detector, high speed shift circuit, zero detector based on the carry assimilator, must be used. However, the whole exchange time itself might not be shortened to a logarithmic scale, and in the worst case it takes full m cycles.

Define c as

$$c = \sum_{i=1}^{m-1} 2^{-i} c_i - c_0 . \tag{56}$$

① If $c=0$ (i.e. $c_0 = c_1 = \cdots = c_{m-1} = 0$) then $s\colon =0$ and proceed to ⑤.

If $c>0$ (i.e. $c_0=0$) then $s\colon=0$ and proceed to ②.

If $c<0$ (i.e. $c_0=1$) then get normalization number n of the control word, $s\colon=n+1$, shift the control word to the left logically s places, shift the information word to the left s places cyclically and proceed to ②.

② Even exchange cycle. If $c=0$ then proceed to ④.

```
1 1 1 1 1 0 0 0 0 1 1 1 0 0 1  s=0
a b c d e f g h i j k l m n o    |
                                 ↓   left cycle shift 5 places
0 0 0 0 1 1 1 0 0 1 0 0 0 0 0  s=5
f g h i j k l m n o a b c d e    |
                                 ↓   even exchange
0 0 0 0 1 1 1 0 1 0 0 0 0 0 0  s=5
f g h i j k l m o n a b c d e    |
                                 ↓   odd exchange
0 0 0 1 0 1 1 1 0 0 0 0 0 0 0  s=5
f g h j i k l o m n a b c d e    |
                                 ↓   even exchange
0 0 1 0 1 0 1 1 0 0 0 0 0 0 0  s=5
f g j h k i l o m n a b c d e    |
                                 ↓   odd exchange
0 1 0 1 0 1 0 1 0 0 0 0 0 0 0  s=5
f j g k h l i o m n a b c d e    |
                                 ↓   even exchange
0 1 0 1 0 1 0 0 0 0 0 0 0 0 0  s=6  shift
f k g l h o i m n a b c d e j    |
                                 ↓   even exchange
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0  s=7  shift
f l g o h i m n a b c d e j k    |
                                 ↓   even exchange
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0  s=8  shift
f o g h i m n a b c d e j k l    |
                                 ↓   even exchange
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  s=9  shift
f g h i m n a b c d e j k l o    |
                                 ↓   left cyclic shift 6 place

a b c d e j k l o f g h i m n
```

Fig. 9

If $c_0=c_1=0$ then perform ordinary even exchange and proceed to ③.
If $c_0=0$ and $c_1=1$ then perform even exchange, $s:=s+1$, shift the control word 1 place to the left, shift the information word 1 place to the left cyclically and proceed to ②.
③ Odd exchange cycle. Perform ordinary odd exchange and proceed to ②.
④ Shift the information word cyclically to the left $m-s$ places, and proceed to ⑤.
⑤ End of operation

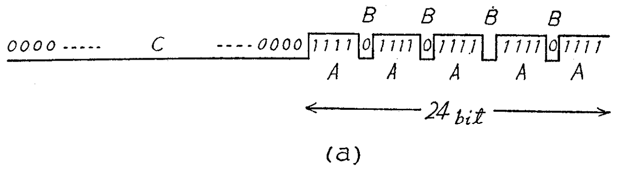
On exit from step 5, s is the number of 1's in the control word. Fig. 9 gives an example of the above process.

3. *Application to Size Normalization for One-Dimensional Pattern Recognition*

In this problem of pattern recognition, several standard patterns are given together with a test pattern. All patterns are one-dimensional array of 0's and 1's. The pattern is considered to lie between two outmost 1's in a computer word and the distance of these two 1's is called the size of pattern.

Now, the method to contract the test pattern to normalize the size using the proposed operations is considered. Use the shift number detector and the high speed shift circuit to move the test pattern to the left end of the word. Then reverse right and left by perfect 'Amitabha lottery' circuit, and use the shift number detector again to obtain the length of the test pattern. Next, suppose, for instance, the word length is 48 bits, and the length of the test pattern and that of the standard pattern are 24 bits and 20 bits respectively. The test pattern must be contracted by 4 bits in the 48 bits word. The following operation contracts the test pattern.

(a) Apply the bit separation using the control word of Fig. 10(a).



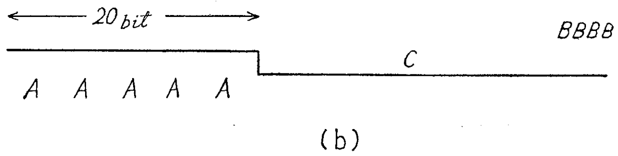


Fig. 10

(b) Shift to the right logically by 14 places.

By the control word of Fig. 10 (a), the contracted pattern appears as Fig. 10 (b) to the left of the output word and the refuse gathers to the right. Logical right shift can be used to drop off the refuse. For the other length of the test pattern, the similar control word may be used mutatis mutandis.

The normalized test pattern can be compared with the standard patterns by a bitwise addition and subsequent sideways addition. The latter can also be performed by the present operations.

It seems too optimistic to suppose that the pattern recognition could be performed by such a simple treatment as the above even in one-dimension. These applications may be understood as an example of the application of the two proposed operations.

Acknowledgment

The author wishes to express his appreciation to Prof. Hidetosi Takahashi for his helpful discussions.