UNIVERSITY OF CALIFORNIA
LOS ANGELES
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
COMPUTER SCIENCE DEPARTMENT

THEORY OF DIGITAL COMPUTER ARITHMETIC

Notes for Engineering 225A

by Algirdas Avizienis

Chapter 5: Additional Topics

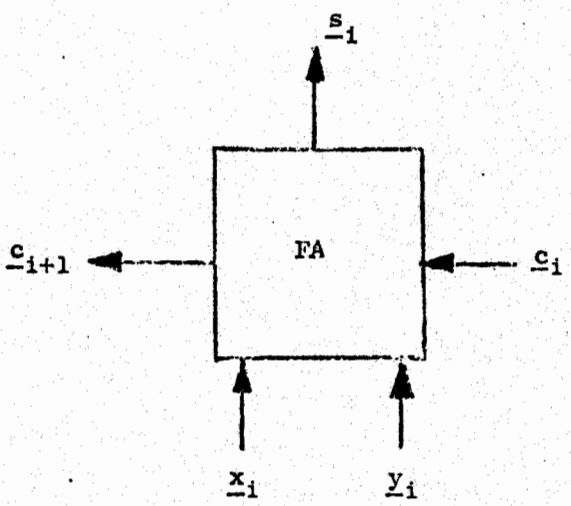
Engr. 225A 1968-69

Multi-Operand Addition (Radix 2)

A. Avizienis

The addition of more than two operands is also of interest, especially in implementing the multiplication algorithm. A fast and systematic method of adding several radix 2 operands is the "carry-save" method of cascading arrays of binary full adders.

A binary full adder (FA) is shown below:



The arithmetic transfer function of the FA is:

$$x_i + y_i + c_i \equiv 2c_{i+1} + s_i$$

with

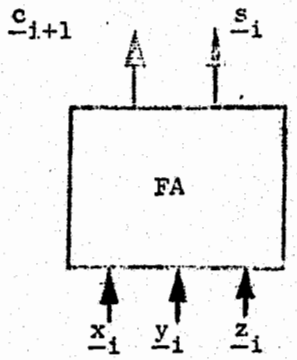
$$s_i \leftarrow 2|(x_i + y_i + c_i)$$

$$c_{i+1} \leftarrow (x_i + y_i + c_i) > 1$$

The internal logic net of the FA depends on the nature of the logic elements being used; the FA is available as a single integrated circuit package from most manufacturers.

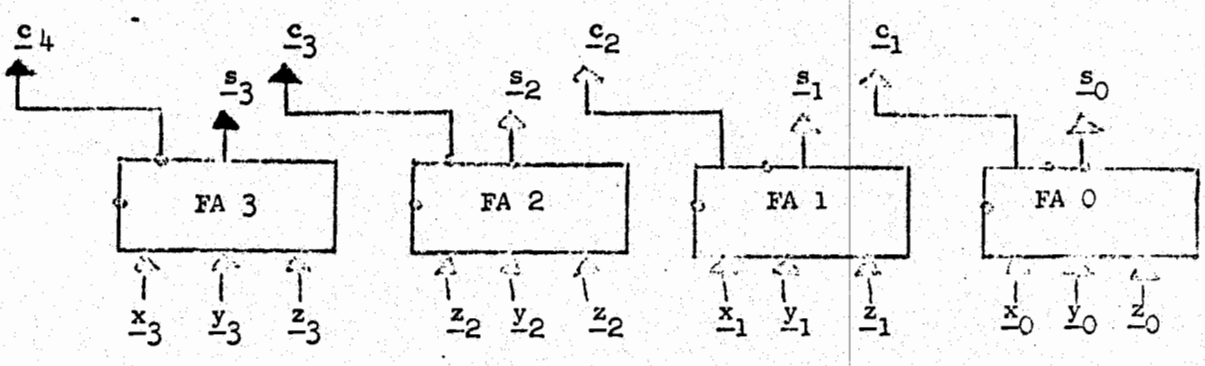
For the purpose of multi-operand summation, we consider the FA to

receive all three inputs from n-bit binary numbers $\underline{x}$, $\underline{y}$, $\underline{z}$ as follows:



The same description as before applies, with $\underline{z}_i$ replacing the input $\underline{c}_i$.

An array of n such FA's can be employed to accept the binary operands $\underline{x}$, $\underline{y}$, $\underline{z}$ as inputs and to produce two binary numbers $\underline{s}$, $\underline{c}$ as results. An example for $n = 4$ is shown below:



Both results are n bits long:

$$\underline{c} = (\underline{c}_n, \underline{c}_{n-1}, \cdots, \underline{c}_1)$$
$$\underline{s} = (\underline{s}_{n-1}, \underline{s}_{n-2}, \cdots, \underline{s}_0)$$

The vector $\underline{c}$ is displaced one position to the left; that is, it lacks $\underline{c}_0$ and has $\underline{c}_n$ as the left-most component. The relationship of natural values of the vectors is:

$$x + y + z \equiv c + s$$

that is, the vectors $\underline{c}$ and $\underline{s}$ represent the same natural value as the vectors $\underline{x}$, $\underline{y}$, and $\underline{z}$ together. Discarding $\underline{c}_n$ from $\underline{c}$ to get $\underline{c}' \equiv (\underline{c}_{n-1},\ldots,\underline{c}_1)$, we have:

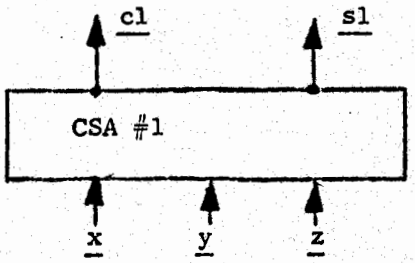
$$2^n \mid (x + y + z) \equiv c' + s$$

and if we furthermore let $\underline{c}_0 \leftarrow \underline{c}_n$ and then discard $\underline{c}_n$ to get $\underline{c}'' \equiv (\underline{c}_{n-1},\ldots,\underline{c}_0)$, we have

$$(2^n-1) \mid (x + y + z) \equiv c'' + s$$

that is, $\underline{c}'$ and $\underline{s}$ represent the modulo 2^n residue of the sum $x + y + z$, while $\underline{c}''$ and $\underline{s}$ represent the modulo 2^n-1 residue of the same sum $x + y + z$.

The array of n FA's shown in the preceding figure is called a "carry-save adder" (CSA). The CSA can be shown in a convenient block diagram form as follows:

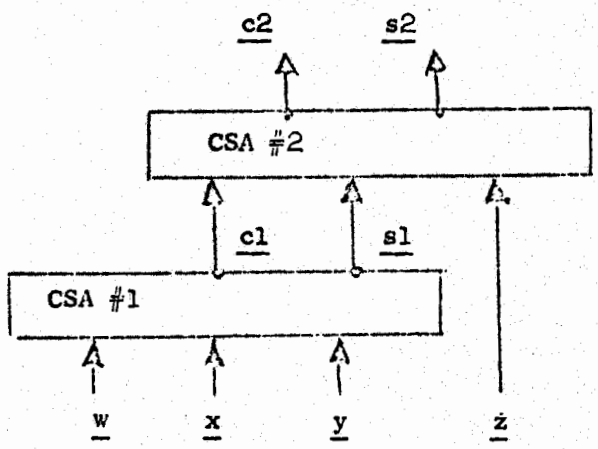


The treatment of $\underline{c}_n$ and the specification of the value of $\underline{c}_0$ remains to be stated explicitly for this diagram.

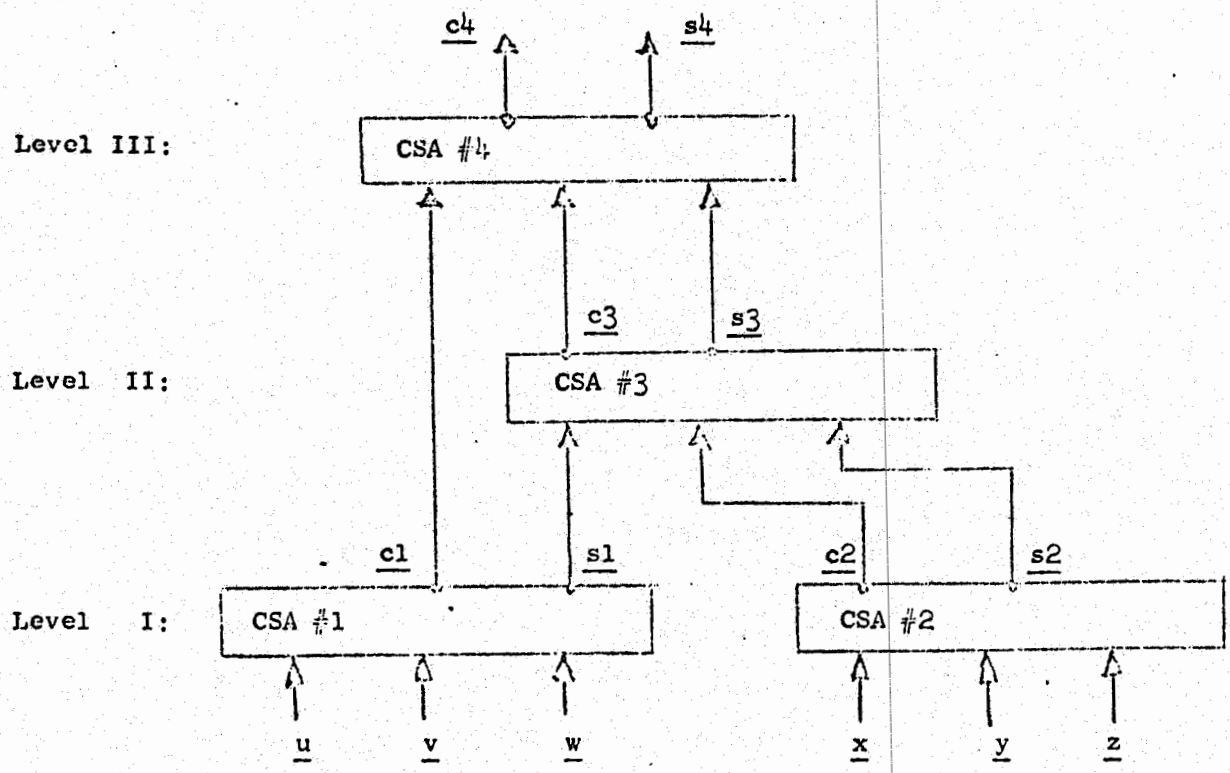
It is evident that to get a single-vector representation of the radix 2 sum of vectors $\underline{x}$, $\underline{y}$, and $\underline{z}$, the two result vectors $\underline{c}$ and $\underline{s}$ have to be added in a conventional adder with complete carry propagation.

In general, any number of binary operands can be reduced to a pair of results by the use of carry-save adders. The following figures show the

arrays for summing 4 and 6 n-bit operands, respectively, to get the two-vector representation of the natural value of the sum:



(a) Four vector CSA summation cascade (two levels)



(b) Six vector CSA summation cascade (three levels)

We note that each additional input operand increases the number of CSA's by one. To reduce m operands to a two-vector result, m-2 CSA's will be required. The maximum depth of the cascade can be m-2 levels in the case in which all CSA's are cascaded in a series arrangement. The minimum depth is attained when the largest possible number of CSA's is used in parallel at each level. The maximum number of operands which can be added using a cascade of j levels in depth is denoted by Max (j) and is given by the expression:

$$\text{Max}(j) \equiv \{\lfloor \tfrac{1}{2} * \text{Max}(j-1) \rfloor * 3\} + \{2 \mid \text{Max}(j-1)\}$$

for $j = 2, 3, \ldots$

$$\text{Max}(1) \equiv 3$$

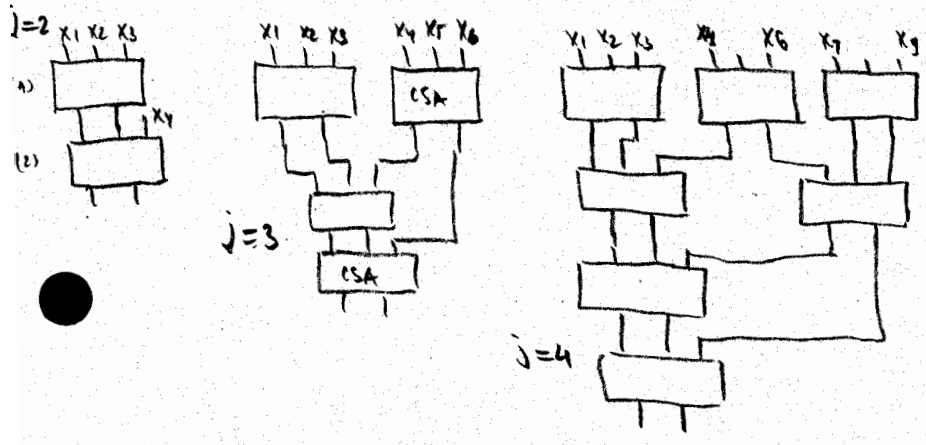
Some of the values of this function are:

j =	1	2	3	4	5	6	7	8	9	10	11	12
Max(j) =	3	4	6	9	13	19	28	42	63	94	141	211

The depth (in levels) of a CSA cascade determines the longest path (in FA delay units) for a signal between the input and output of the CSA cascade.

$$\text{Max}(j) \equiv \{3 \lfloor \tfrac{1}{2} \text{Max}(j-1) \rfloor\} + \{2 \mid \text{Max}(j-1)\}$$

$$\text{Max}(j) = \{3 \lfloor \tfrac{1}{2} \text{Max}(j-1) \rfloor\} + \{2 \mid \text{Max}(j-1)\}$$



Recoding of the Multiplier

A. Avizienis

1. Canonical Recoding

A fundamental approach to accelerating the multiplication algorithm has been the recoding of the multiplier into a signed-digit form.

In the case of radix 2, the recoded form has the set of allowed digits $\Sigma = \{1,0,\bar{1}\}$, where $\bar{1}$ is used to represent the digit value "minus one." This form of radix 2 numbers is redundant, that is, some natural values have more than one representation. For example, consider the radix 2 vectors:

$$\underline{x} = (011111)_2$$

$$\underline{y} = (10000\bar{1})_2$$

The natural values of both are given by the expressions:

$$\underline{x} = \sum_{i=0}^{5} \underline{x}_i * 2^i = 0 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 31$$

$$\underline{y} = \sum_{i=0}^{5} \underline{y}_i * 2^i = 2^5 + 0 + 0 + 0 + 0 - 2^0 = 31$$

As a matter of fact, all following 6-digit vectors also represent the same natural value 31 in radix 2:

$$(1000\bar{1}1)_2 \ , \quad (100\bar{1}11)_2 \ , \quad (10\bar{1}111)_2 \ , \quad \text{and}$$

$$(1\bar{1}1111)_2$$

A difference among these forms is found in the count of non-zero digits, which we call the multiplicity μ. The multiplicity ranges from 2 to 6 in the vectors of the preceding example.

In general the multiplicity of a binary n-digit vector $\underline{x}$ is given by the expression:

$$\mu(\underline{x}) = \sum_{i=0}^{n-1} |\underline{x}_i|$$

Of special interest in computer arithmetic is the n-digit form which has the least multiplicity among all n-digit forms representing a given natural value k . Such a form is called the "n-digit minimal form of natural value k ". The multiplicity of a minimal form of length n and natural value k is called the weight and is denoted by $\omega(n,k)$. In the preceding example the six digits long minimal form of the natural value 31 is $(100001)_2$ and its weight is $\omega(6,31) = 2$.

In some cases there may be more than one minimal form for a given vector length n and natural value k . For example, given $n = 6$ and $k = 3$, there will be two minimal forms with the weight $\omega(6,3) = 2$, that is, with the multiplicity $\mu = 2$:

$$(000011)_2 \quad \text{and} \quad (00010\bar{1})_2$$

There will also exist non-minimal forms for $n = 6$ and $k = 3$, for example:

$$(001\bar{1}0\bar{1})_2 \ , \quad (01\bar{1}\bar{1}0\bar{1})_2 \ , \quad (1\bar{1}\bar{1}\bar{1}0\bar{1})_2$$

$$(001\bar{1}\bar{1}\bar{1})_2 \ , \quad (01\bar{1}\bar{1}\bar{1}\bar{1})_2 \ , \quad (1\bar{1}\bar{1}\bar{1}\bar{1}\bar{1})_2$$

A detailed study of the existence and generation of minimal forms has been performed by G. W. Reitwiesner (Reference A5, Section 8). A binary n-digit vector $\underline{x}$ is said to possess the "unqualified property M " if and only if

$$\underline{x}_i * \underline{x}_{i-1} = 0 \quad \text{for} \quad 1 \leq i \leq n-1$$

that is, $\underline{x}$ does not contain adjacent non-zero digits.

Given the restriction that both leftmost digits cannot be +1 or -1 simultaneously, that is:

$$\underline{x}_{n-1} * \underline{x}_{n-2} \neq 1 ,$$

Reitwiesner shows:

(1) for a vector length n and natural value k there exists a unique form, (to be called canonical) which possesses the unqualified property M ;

(2) no other form of this natural value has a lower multiplicity than the canonical form.

The restriction $\underline{x}_{n-1} * \underline{x}_{n-2} \neq 1$ was imposed because of an assumed range limit for the multipliers. It can always be satisfied by affixing an additional digit $\underline{x}_n = 0$ at the left end of the operand which is to be recoded; that is, the canonical form is obtained for the n+1 digit form when $\underline{x}_{n-1}$ and $\underline{x}_{n-2}$ are both +1 or -1 .

The canonical recoding algorithm gives the rules for converting a conventional binary digit vector $\underline{x} = (\underline{x}_n, \underline{x}_{n-1}, \ldots, \underline{x}_0)$ with $\underline{x}_n = 0$ and $\underline{x}_i = 0$ or 1 for $0 \leq i \leq n-1$ to a signed-digit vector $\underline{y} = (\underline{y}_n, \ldots, \underline{y}_0)$ $(\underline{y}_i = 0, 1, \overline{1})$ which is the canonical form for the natural value represented by $\underline{x}$.

The serial recoding beginning at the low end of $\underline{x}$ requires the inspection of digits $\underline{x}_{i+1}$ and $\underline{x}_i$ and a "carry" digit $\underline{c}_i$ to generate the digit $\underline{y}_i$ of the canonical form and the "carry" digit $\underline{c}_{i+1}$ which is to be used in the next step. The algorithm is stated as

$$\underline{c}_{i+1} \leftarrow \underline{x}_{i+1} + \underline{c}_i - 2\underline{c}_{i+1} \quad \text{for} \quad 0 \leq i \leq n$$

$y_i \leftarrow x_i + c_i - 2c_{i+1}$

$0 \le i \le n$

with:

[redacted]

The following table describes details of the algorithm:

Inputs			Outputs		
$\underline{x}_{i+1}$	$\underline{x}_i$	$\underline{c}_i$	$\underline{y}_i$	$\underline{c}_{i+1}$	
0	0	0	0	0	$\underline{x}_{i+1}$ does not matter for these cases
1	0	0	0	0	
0	1	1	0	1	
1	1	1	0	1	
0	0	1	1	0	choose $\underline{c}_{i+1}$ such that $\underline{y}_{i+1} = 0$ will result
0	1	0	1	0	
1	0	1	$\bar{1}$	1	
1	1	0	$\bar{1}$	1	

The lower four entries require a non-zero value of $\underline{y}_i$; therefore $\underline{c}_{i+1}$ is chosen to guarantee that the next digit to the left in the recoded form $(\underline{y}_{i+1})$ will be a zero and $\underline{y}_{i+1} * \underline{y}_i \neq 1$ will be satisfied.

It is evident that the table may be expanded to generate more than one digit of the canonical form at once. For instance, $\underline{y}_i, \underline{y}_{i+1}$ and $\underline{c}_{i+2}$ can be generated as functions of c_i, x_i, x_{i+1}, and x_{i+2} . Since this is a canonical form, then both $\underline{y}_{i+1}$ and $\underline{y}_i$ cannot be ones. The possible combinations are; 00, 01, $0\bar{1}$, 10, and $\bar{1}0$. We may consider these two digits

$(\underline{y}_{i+1}, \underline{y}_i)$ to represent a radix 4 digit $\underline{w}_j$ with the allowed values $\{0, 1, \overline{1}, 2, \overline{2}\}$.

Two digits of the multiplier will be used up at once in a binary multiplication if the provision (sic: assurance) for adding 0, $\pm$ 1 and $\pm$ 2 times the multiplicand to the partial product is included in the arithmetic processor. The addition is followed by a two-position shift of the sum.

2. "String" Recoding (Booth's Algorithm)

Another method of recoding is known as "string" recoding and employs the observation that the two strings of binary digits

String A: $(\_\ \_\ \_\ \underbrace{01111}\ \_\ \_\ \_)_2$

String B: $(\_\ \_\ \_\ \underbrace{10001}\ \_\ \_\ \_)_2$

represent the same natural value in radix 2. A recoding will be attained if we specify the replacement of each string of ones, terminated by a zero at the left end, (such as string A above), by the string B, which contains a $\overline{1}$ instead of the right end digit 1 of string A, zeros for all other interior ones, and a 1 for the left end digit 0 of string A.

An algorithm to recode a conventional binary (n+1)-digit vector $\underline{x}$ by this "string" method is given by the table:

Inputs		Output	Observation regarding x_i
$\underline{x}_i$	$\underline{x}_{i-1}$	$\underline{z}_i$	
0	0	0	$\underline{x}_i$ does not belong to a string
0	1	1	$\underline{x}_i$ is the left end digit of a string
1	0	$\overline{1}$	$\underline{x}_i$ is the right end digit of a string
1	1	0	$\underline{x}_i$ is an interior digit of a string

The recoding requires the assumption of $\underline{x}_{\overline{1}} = 0$ and the attachment of $\underline{x}_n = 0$ to a given n-digit vector $\underline{x}$.

The algorithm above expresses the new digit $\underline{z}_i$ as the function of $\underline{x}_i$ and $\underline{x}_{i-1}$, only and therefore is suitable for a parallel recoding of $\underline{x}$ into $\underline{z}$. The new signed-digit form $\underline{z}$, however, is not always the same as the canonical form $\underline{y}$ discussed previously, since the "string" algorithm given above is inadequate to handle the occurrence of single zeros and ones. For example, consider the recodings:

$$\underline{x} = (011101110)_2 \qquad \underline{x}' = (01010101)_2$$

$$\underline{z} = (100\overline{1}100\overline{1}0)_2 \qquad \underline{z}' = (1\overline{1}1\overline{1}1\overline{1}1\overline{1})_2$$

Adjacent non-zero digits can exist and the multiplicity of the recoded form may even be higher than the multiplicity of the original form. The algorithm, however, guarantees the condition:

$$\underline{z}_{i+1} * \underline{z}_i \neq +1 \quad \text{for} \quad n-1 \leq i \leq 0$$

that is, there are no adjacent non-zero digits of the same sign in the recoded vector $\underline{z}$.

Subdividing the vector $\underline{z}$ into groups of two digits each, we encounter the following pairs $(\underline{z}_{i+1}, \underline{z}_i)$, which can be interpreted as radix 4 digit values $\underline{u}_j$:

radix 2 $(\underline{z}_{i+1}, \underline{z}_i)$	00,	01,	$0\overline{1}$,	10,	$\overline{1}0$,	$1\overline{1}$,	$\overline{1}1$
radix 4 $(\underline{u}_j)$	0,	1,	$\overline{1}$,	2,	$\overline{2}$,	1,	$\overline{1}$

The values ± 3 are avoided in the radix 4 digit vector, and the result of the parallel "string" recoding can be used in the same manner as the result of the canonical recoding.

The preceding table may be readily extended to express pairs of digits $\underline{z}_{i+1},\underline{z}_i$ as a function of $\underline{x}_{i+1},\underline{x}_i$, and $\underline{x}_{i-1}$. This will guarantee the condition $\underline{z}_{i+1}*\underline{z}_i \neq \pm 1$ for each pair, but the next pair $\underline{z}_{i+3},\underline{z}_{i+2}$ may be such that only the condition $\underline{z}_{i+2}*\underline{z}_{i+1} \neq 1$ can be guaranteed. Such a recoding has been applied in the multiplication unit of the IBM System 360/91 arithmetic processor. The recoding table is given below:

Inputs			Outputs		Observations on $\underline{x}_{i+1},\underline{x}_i$ as members of a string
$\underline{x}_{i+1}$	$\underline{x}_i$	$\underline{x}_{i-1}$	$\underline{z}_{i+1}$	$\underline{z}_i$	
0	0	0	0	0	no string in evidence
0	0	1	0	1	$\underline{x}_i$ is the left end
0	1	0	0	1	$\underline{x}_i$ is an isolated "1" (short string)*
0	1	1	1	0	$\underline{x}_{i+1}$ is the left end, $\underline{x}_i$ interior digit
1	0	0	$\bar{1}$	0	$\underline{x}_{i+1}$ is the right end
1	0	1	0	1	$\underline{x}_i$ is the left end, $\underline{x}_{i+1}$ right end**
1	1	0	0	$\bar{1}$	$\underline{x}_i$ is the right end, $\underline{x}_{i+1}$ interior digit
1	1	1	0	0	$\underline{x}_i,\underline{x}_{i+1}$ are both interior digits

Remarks:

*01 = $(\underline{x}_{i+1},\underline{x}_i)$ is a complete two-digit string; instead of $\underline{z}_{i+1} = 1$, $\underline{z}_i = \bar{1}$ we use the equivalent form 01.

**again, instead of $\underline{z}_{i+1} = \bar{1}$ $\underline{z}_i = 1$, we use the equivalent combination $\underline{z}_{i+1} = 0$, $\underline{z}_i = \bar{1}$