ARITHMETIC CIRCUIT FAULT DETECTION BY MODULAR ENCODING

Antonin Svoboda

UCLA Computer Science Department
University of California
Los Angeles, CA 90024

Abstract

Design principles of self checking digital circuits are in the focus of the general interest and many papers exist treating that subject. The use of special data encoding techniques, suitable algorithms of arithmetic, special hardware elements have been proposed long ago. The purpose of this paper is to show that the design can produce rather simple self checking circuit when the design principles are chosen which collaborate harmoniously:

1) decimal numerical system is used
2) decimal digit $d \in \{0, 1,...,9\}$ is represented in the Diamond Code by the 5-bit binary number $f = 3 \cdot d + 2$
3) decimal digits' addition algorithm introduced here is simple and effective so that 10 decimal digits can be added in parallel
4) implementation of the addition algorithm by conventional Full Adders results in in a single fault detecting circuit.

The design of a decimal adder for 10 decimal numbers, each with 10 digits, is described here as an illustration. It shows the way how to design other decimal arithmetic circuits which are single fault detecting, for instance a multiplier (derived from the adder for 10 decimal numbers).

1. Introduction

The present arithmetic design practice, especially when dealing with medium or large size computers, is based on binary numerical systems. The use of decimal system is connected (better: it is believed to be connected) with a slow-down of operation because most frequent arithmetic operation - addition - is performed in two steps: binary addition first and then decimal digit code restitution. Roughly speaking the speed is cut in half.

The slowdown of the adder for a large number of decimally encoded numbers (presented here) is insignificant because the code restitution is done only once after a parallel binary addition of many (typically 10) numbers.

*This research was supported by the National Science Foundation, Under Grant No. MCS72-03633 A04.

The adder is shown to be SFD (Single Fault Detecting) an attractive feature for some applications.

The decimal digits are encoded in Diamond Code (one of the Brown Codes [1]) so that the algorithms of the arithmetic become very simple for hardware implementation [2]. The adder can be used to design a very fast multiplier. All ten multiples $0 \cdot D$, $1 \cdot D$,..., $9 \cdot D$ are prepared during the execution of the instruction: FETCH the multiplier D*. Then the execution time of the instruction MULTIPLY D by D* with 10 decimals has only two components: time needed to add a column of decimal digits (addition of only 10 five bit binary numbers) and time needed to add two decimal numbers. Note that the result's accuracy is equivalent to the accuracy of a 33 bits × 33 bits binary multiplication.

2. Algorithmic Structure Of The Decimal Adder.

The adder must process numbers of both signs, of course.

The input format for input numbers N is shown in Figure 1. To fix ideas 10 significant digits plus sign digit are supposed as an example.

$$d^{10} \cdot d^9\ d^8\ d^7\ d^6\ d^5\ d^4\ d^3\ d^2\ d^1\ d^0$$

↑
Sign digit: 0 or 9

Figure 1: Input number format: Sign digit, decimal point, ten significant digits.

"Tenth's complement" rules are used to handle the sign:

Decimal digit:

$$d^i \in \{0, 1,...,9\} \text{ for } i = 0, 1,...,9 \qquad (2.1)$$

Sign digit:

$$d^{10} \in \{0, 9\} \qquad (2.2)$$

Zero and positive numbers:

$$N = D \geq 0 \text{ with } d^{10} = 0 \qquad (2.3)$$

Negative numbers:

$$n = D - 10 < 0 \text{ with } d^{10} = 9 \qquad (2.4)$$

The interval for N;

$$-1 \le N < 1 \qquad (2.5)$$

The value of D:

$$D = (d^{10}.10^{10} + d^{9}.10^{9} + \ldots + d^{0}.10^{0}).10^{-10} \qquad (2.6)$$

Figure 2 represents the decimal digit pattern written when 10 numbers D_k, k = 0, 1,..., 10 defined in Figure 1 are added by hand.

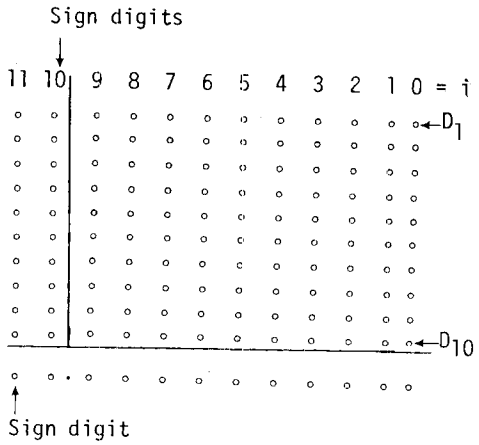

Figure 2: Addition of 10 decimal numbers D_k, k = 1,2,...10 by hand proceeds column by column (i = 0,1,...,10) The total $X = \Sigma D_k$ for all k.

The columns of the pattern are indexed by i = 0, 1,..., 11, the rows by k = 1, 2,..., 10. The row k contains the number

$$= (d_k^{11}.10^{11} + d_k^{10}.10^{10} + \ldots + d_k^{0}).10^{-10},$$

$$\text{where } d_k^{11} = d_k^{10} \qquad (2.7)$$

The digit d_k^i belongs to the row k and to the column i. The column i=11 is identical with the column i=10. That column must be added to get the correct total (Figure 3) whose digit x^{11} is necessary as sign digit.

$$x^{11}\ x^{10}\ x^{9}\ x^{8}\ x^{7}\ x^{6}\ x^{5}\ x^{4}\ x^{3}\ x^{2}\ x^{1}\ x^{0}$$

↑ Sign digit: 0 or 9.

Figure 3: Format of the total $X = \sum_{k=1}^{10} D_k$. Digit x^{10} being significant, the digit x^{11} is added as Sign digit.

The by hand addition algorithm which adds a carry from the column i-1 to the sum of decimal digits of the column i to get the digit x^i of the total as well as the carry into the column i+1, was changed slightly to improve its implementation:

BEGIN with the matrix d_k^i, i = 0, 1,..., 10, k = 1, 2,..., 10

1: EVALUATE for each column i=0, 1,..., 10 the integers C^i, A^i so that

$$\sum_{k=1}^{10} d_k^i = C^i.10+A^i \text{ with } 0 \le A^i \le 9,\ 0 \le C^i \le 9 \qquad (2.8)$$

2: EVALUATE the total X by executing X =C·10+A, where

$$A = (A^{10}.10^{11} + A^{10}.10^{10} + A^{9}.10^{9} + \ldots + A^{0}).10^{-10} \qquad (2.9)$$

$$C = (C^{10}.10^{10} + C^{9}.10^{9} + \ldots + C^{0}).10^{-10} \qquad (2.10)$$

The step 2 is illustrated in Figure 4.

$$\begin{aligned} A &= A^{10}\ A^{10}\ .\ A^{9}\ A^{8}\ A^{7}\ A^{6}\ A^{5}\ A^{4}\ A^{3}\ A^{2}\ A^{1}\ A^{0} \\ 10\cdot C &= C^{10}\ C^{9}\quad C^{8}\ C^{7}\ C^{6}\ C^{5}\ C^{4}\ C^{3}\ C^{2}\ C^{1}\ C^{0} \\ X &= x^{11}\ x^{10}\ .\ x^{9}\ x^{8}\ x^{7}\ x^{6}\ x^{5}\ x^{3}\ x^{2}\ x^{1}\ x^{0} \end{aligned}$$

↑ Sign digit

Figure 4: Second step of the addition algorithm. Note: $C^i \cdot 10 + A^i$ is the sum of digits in the column i.

Note that A^{10} occurs twice in A because columns i = 10, 11 being identical their sums of digits are equal. The execution time is composed from two items only: time to execute step 1 is equal to the time to add one column of decimal digits (addition of 10 b-bit binary numbers) because all columns are processed at the same time; time to execute step 2 is equal to the time to add two decimal numbers (Figure 4): C·10+A (about double of the time needed to add two binary numbers 55 bit long).

3. Column Addition Algorithm

To evaluate the sum Σd_k^i in (2.8), each decimal digit is represented by a binary number F. To achieve SFD (single fault detection) an encoding with redundancy must be used. The Brown Codes are very useful for that purpose because their formula:

$$F = pd + q, \qquad (3.1)$$

where p,q are integers and d ∈ {0, 1,...,9}, permits modular arithmetic checking:

$$F \equiv q \pmod{p} \qquad (3.2)$$

The decision to use the Diamond Code F = 3d+2 with 2≤F≤29 is quite natural because that code is the least redundant of the Brown Codes, needing only 5 bit format (Figure 5).

d	F	F = 3·d + 2					G				
		f_4	f_3	f_2	f_1	f_0	f_4	$\bar{f}_3$	f_2	$\bar{f}_1$	f_0
0	2	0	0	0	1	0	0	1	0	0	0
1	5	0	0	1	0	1	0	1	1	1	1
2	8	0	1	0	0	0	0	0	0	1	0
3	11	0	1	0	1	1	0	0	0	0	1
4	14	0	1	1	1	0	0	0	1	0	0
5	17	1	0	0	0	1	1	1	0	1	1
6	20	1	0	1	0	0	1	1	1	1	0
7	23	1	0	1	1	1	1	1	1	0	1
8	26	1	1	0	1	0	1	0	0	0	0
9	29	1	1	1	0	1	1	0	1	1	1

Figure 5: DIAMOND CODE for representation of decimal digits. Checking is done by counting zeros in G derived from F by complementing f_3 and f_1.

The next best code is F = 7d with 0≤F≤63 which needs 6 bits. Figure 5 tabulates the Diamond Code:

$$F = 3d + 2 = f_4 2^4 + f_3 2^3 + f_2 2^2 + f_1 2^1 + f_0 \qquad (3.3)$$

and G-numbers derived from F by complementing the bits f_3 and f_1. It is easy to prove that when the 5 bit number G has either one of four zeroes it belongs to one of meaningful F numbers tabulated in Figure 5. (Note: there are exactly 10 numbers of the form 3d+2 within the interval (0,31) and there are exactly 10 5-bit numbers written with either one or four zeroes and they are all present in the table Figure 5.)

Figure 6 shows one of possible implementations of a Diamond Code checking circuit. The output signal OK is ON only when the input signal configuration represents one of the F numbers in Figure 5.

It is difficult and unrewarding to explain the creation of the Column Addition Algorithm. Figure 7 describes a hardware implementation for an addition of 16 decimal digits d_k, k=1, 2,..., 16. For decimal digits the algorithm takes on a very simple form:

Begin with $F_k = 3d_k + 2$, k = 1, 2,..., 16 (3.4)

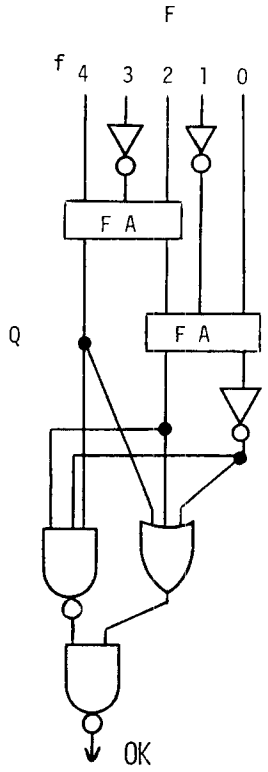

Figure 6: Diamond Code checking circuit. The output signal OK is ON only when the input signal configuration represents a decimal digit in the Diamond Code.

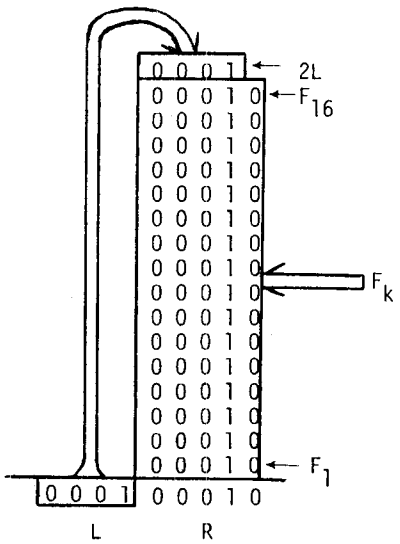

Figure 7: Column addition algorithm illustration.

$$2L + \Sigma F_k = R + 32L;\ 0 \le R \le 31;\ 0 \le L \le 15$$

1: EVALUATE L, R so that

$$2L + \sum_{1}^{16} F_k = 32L+R \text{ with } 0 \le R \le 31;\ 0 \le L \le 15 \qquad (3.5)$$

2: The resulting sum of digits will be C·10 + A, where

$$C = L-1 \qquad A = (R-2)/3 \qquad (3.6)$$

The column adder is composed from a binary adder for 17 5bit numbers: F_1, F_2,..., F_{16}, 2L; the resulting 9-bit sum is split in two parts: L (4 bits on the left), R (5 bits on the right). The value of L is fed back into the adder as 2L (one bit shift to the left). Note that A is expressed by R in Diamond Code, C being expressed by the binary number L. (Figure 7 shows the addition of 16 Zeros).

Figure 8 tabulates the numerical values bound by (3,4), (3,5), (3,6) for selected values of $\sum_{1}^{16} d_k$. When the table is completed for all

$\sum_{1}^{16} d_k = 0,1,2,\ldots,144$ the last column of the table is identical with its first column. That proves that the column addition algorithm is correct.

$\sum_{1}^{16} d_k$	$\sum_{1}^{16} F_k$	L	$2L + \sum_{1}^{16} F_k$	R	C = L-1	A = (R-2)/3	$(C\ A)_{10}$
0	32	1	34	2	0	0	00
1	35	1	37	5	0	1	01
⋮							
9	59	1	61	29	0	9	09
10	62	2	66	2	1	0	10
⋮							
19	89	2	93	29	1	9	19
20	92	3	98	2	2	0	20
⋮							
99	329	10	349	29	9	9	99
⋮							
144	464	15	494	14	14	4	144

Figure 8: The numerical values of L, R, C, A, belonging to different values of $\sum_{1}^{16} d_k$ (column addition algorithm for sixteen decimal digits).

Note that L stays constant as long as $\sum_{1}^{16} d_k$ stays between two consecutive values divisible by 10.

For each value $\sum_{1}^{16} d_k$ which is divisible by 10 the table gives R=2, A=0.

To keep the value of C below (to exclude second order carry from the column) the sum $\sum_{1}^{16} d_k$ must be kept below 99.

It is easy to use the column addition algorithm to add a number of decimal digits which is below sixteen. All what is to be done is to freeze certain number of decimal input digits to zero. Figure 9 illustrates the column adding algorithm adapted to add 10 decimal digits.

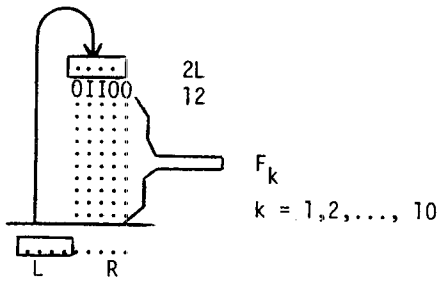

Figure 9: Column addition algorithm for ten decimal digits:

$C = L - 1, \quad A = (R - 2)/3$

The algorithm:

Begin with $F_k = 3d_k + 2$, $k = 1,2,\ldots,10$;

(3.7)

$F_k = 2$ for $k = 11,12,\ldots 16$

1: EVALUATE L,R so that

$$2L + 12 + \sum_{1}^{10} F_k = R + 32L \text{ with } 0 \le R \le 31;\ 0 \le L \le 10 \quad (3.8)$$

2: The resulting sum of digits: $C \cdot 10 + A$, where

$$C = L-1,\ A = (R-2)/3. \quad (3.9)$$

Six input digits being frozen to zero a lump sum 12 = 6·2 was added to balance the total.

BIAS. A simple trick of arithmetic is used here to simplify the column addition algorithm:

The small number -10^{-10} encoded by 99.9999999999 is added as eleventh input of the adder as suggested in Figure 11 (compare with Figure 2).

The decimal digit 9 is added to each column. To add this digit in the column addition algorithm (Figure 9), the value of F = 29 must be added as a constant (BIAS). (Note that will increase the maximal sum of decimal digits to 99, which is permitted.) In combination with the constant 12 (Figure 9) the amount to be added would be 29 + 12 = 41 = 32 + 9. It is clear, however, that 32 units (from 41) contribute in L exactly one unit. By reducing the additive constant to 9 (= 41 - 32) units, the value of L will decrease exactly by one unit. (Figure 10).

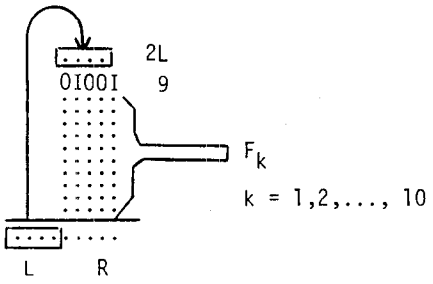

Figure 10: Column addition algorithm for ten decimal digits simplified by BIAS.

$C = L,\ A = (R - 2)/3$

That will simplify the column addition algorithm (with BIAS) as follows:

BEGIN with $F_k = 3d_k + 2$, $k = 1, 2,\ldots, 10$

(3.10)

1: EVALUATE L, R so that

$$2L = 9 + \sum_{1}^{10} F_k = 32L + R \text{ with } 0 \le L \le 9;\ 0 \le R \le 31 \quad (3.11)$$

2: The resulting sum of digits will be C.10+A, where C = L and

$$A = (R - 2)/3 \quad (3.12)$$

99.9999999999 BIAS = -10^{-10}

.. ..

.. ..

.. ..

.. ..

.. ..

.. ..

.. ..

.. ..

.. .. $= X - 10^{-10}$

d_k^i

Figure 11: BIAS of the adder by -10^{-10}

By applying the BIAS the total X will be distorted into $X - 10^{-10}$. The distorted total is corrected by adding 10^{-10} to the value 10·C as suggested in Figure 12.

$A^{10}\ A^{10}.\ A^9\ A^8\ A^7\ A^6\ A^5\ A^4\ A^3\ A^2\ A^1\ A^0 = A$

$C^{10}\ C^9\ \ C^8\ C^7\ C^6\ C^5\ C^4\ C^3\ C^2\ C^1\ C^0\ 1 = 10.C+10^{-10}$

$x^{11}\ x^{10}\ \ x^9\ x^8\ x^7\ x^6\ x^5\ x^4\ x^3\ x^2\ x^1\ x^0 = X$

Figure 12: Correction of the biased total by adding 10^{-10}.

It is important to stress the fact that (Figure 10) R defines the digit A of the sum of digits in a column directly in the Diamond Code.
L, however, defines the digit C (carry) as a binary number

$$L = L_3.2^3 + L_2.2^2 + L_1.2^1 + L_0$$

The decimal adder which evaluates the total X as suggested by Figure 12 accepts input data in the Diamond Code. For that reason the carry C must be encoded in that code:

$B = 3C + 2 = b_4 \cdot 2^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0$.
A combinational network (Carry Circuit CC) is used to do it.

4. Block Diagram Of The Adder

The column addition $\sum_{k=1}^{10} d_k^i + 9$ (biased as in Figure 10) is performed at the same time in eleven Column Adders Σ^i, i=0,1,...,10 (Figure 13).

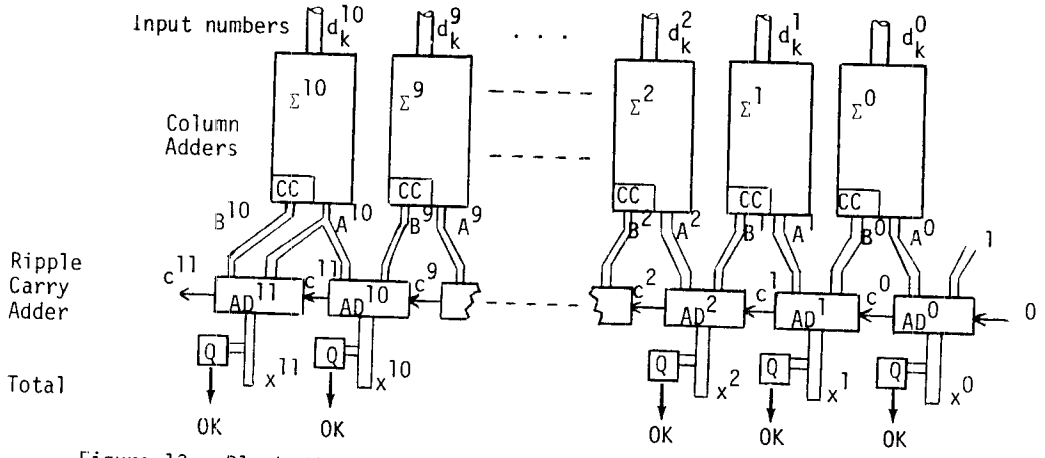

Figure 13: Block diagram of the parallel decimal adder for 10 input numbers.

The adder Σ^i is entered by all digits d_k^i belonging to the same index i.

The sum of digits in the column i is a decimal number $(C^i,A^i)_{10}$ with two decimal digits. The total X (Figure 12) is produced by a ripple carry adder (Figure 13) which works in the Diamond Code. For that reason:

1) The digit A^i comes in from the column adder on the five-wire bus A^i including output wires $a_{41}a_{31}a_{21}a_{11}a_{01}$ of the column adder (See Figure 16, note $a_j \equiv R_j$).

2) The digit $C^i = L^i$ (Figure 10) is produced by Σ^i as a 4-bit binary number $L^i = L_3^i \cdot 2^3 + L_2^i \cdot 2^2 + L_1^i \cdot 2^1 + L_0$ unsuitable as ripple carry adder's input. For that reason each column adder Σ^i is provided with a Carry Circuit CC (Figure 16) which does the necessary encoding in Diamond Code:

$$B^i = 3L^i + 2 = b_4^i \cdot 2^4 + b_3^i \cdot 2^3 + b_2^i \cdot 2^2 + b_1^i \cdot 2^1 + b_0 \qquad (4.1)$$

(five wire bus B^i in Figure 13).

The ripple carry adder includes 12 blocks AD^j, j=0,1,...,11. The block AD^j has inputs A^j, B^{j-1}, c^{j-1} (where c^{j-1} is the ripple carry: 0 or 1) from the block AD^{j-1}. The block's AD^j outputs are: x^j of the total and c^j (ripple carry into the block AD^{j+1}).

Note the correction of the BIAS which enters as a constant 1 into AD^0. Note the way in which the sign digit $x^{11} \in \{0,9\}$ is generated.

5. Single Fault Detection

The components of the adder in Figure 13 will be assembled to get an SFD (Single Fault Detecting) system. The desired goal is to connect hardware elements in a network where a single fault STH (Stuck HIGH) or STL (Stuck LOW) distorts at least one output signal configuration x^i of the total (Figure 13) making its encoding meaningless.

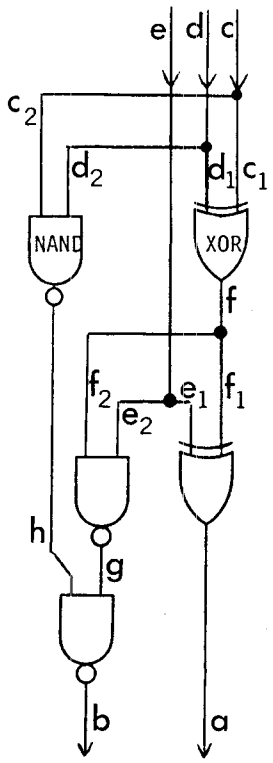

Figure 14: Full Adder. Single fault induced error in the output v = 2b + a is never divisible by 3.

The Full Adder (Figure 14) is used as the main component of the SFD adder because it has a property expressed by

Theorem 1. The change of the numerical value v = 2b+a of the output of the Full Adder in Figure 14 induced by a single fault ϕ is never equal to ± 3.

Proof. There are exactly four distinct signal configurations at the output:

$(b,a) \in \{(0,0), (0,1), (1,0), (1,1)\}$

corresponding to the output values 2b+a = $v \in \{0,1,2,3\}$. For that reason there can be only two ways how to get the change of the output v = 2b+a by 3: both outputs b,a must change either from 0 to 1 or from 1 to 0 simultaneously. Starting with the faultless state a=b=c=d=e=0 we want to place a single faulty link ϕ (STH or STL) anywhere within the circuit in Figure 14 so that both outputs would change as asked for above. First of all we must disqualify places for ϕ from which only one of the outputs b,a is affected: c_2, d_2, h, e_2, f_2, g, b (affecting only b) plus e_1, f_1, a (affecting only a). From the remaining locations c, d, e are ruled out because a change of signal on any of those changes the output v by +1. When either c_1 d_1 or f is STH the output becomes b=0, a=1. For faultless state a=b=c=d=e=0 the possibilities are exhausted. Starting with the faultless state a=b=c=d=e=1 the reader can find for himself that there is no place for ϕ which changes both outputs b,a from 1 to 0 simultaneously. QED.

Figure 15 explains the graphical symbol for full adders in figures to come. The three inputs represent the same input value 2^q with integral exponent which is written inside the square symbol for the full adder. The right hand output represents always 2^q units, the left hand output represents always 2^{q+1} units. The total value of the output v:

$$0 \le v \le 2^{q+1} + 2^q = 3 \cdot 2^q \qquad (5.1)$$

When a single fault is within this full adder the error $\pm 3.2^q$ at its output is ruled out. The fault Δv is restricted to 0, $\pm 2^q$, $\pm 2^{q+1}$.

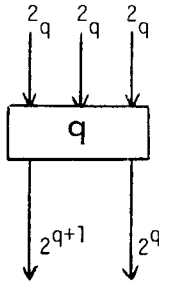

Figure 15: Symbol for the full adder. Active input signal carries 2^q.

6. Single Fault Detecting Column Adder

Figure 16 shows a SFD version of a Column Adder together with the Carry Circuit CC placed there in a special enclosure. In this section the attention will be focused on the Column Adder alone. The network is composed exclusively from Full Adders of the type described in Figure 14 and drawn according the rules given with Figure 15. An empty circle at the full adder's input means that its signal is frozen to LOW, a full circle at the input means to HIGH. (Note two full circles in the upper part of Figure 16. They are used to implement the BIAS of 9 binary units $(= 2^0 + 2^3)$ shown in Figure 10.)

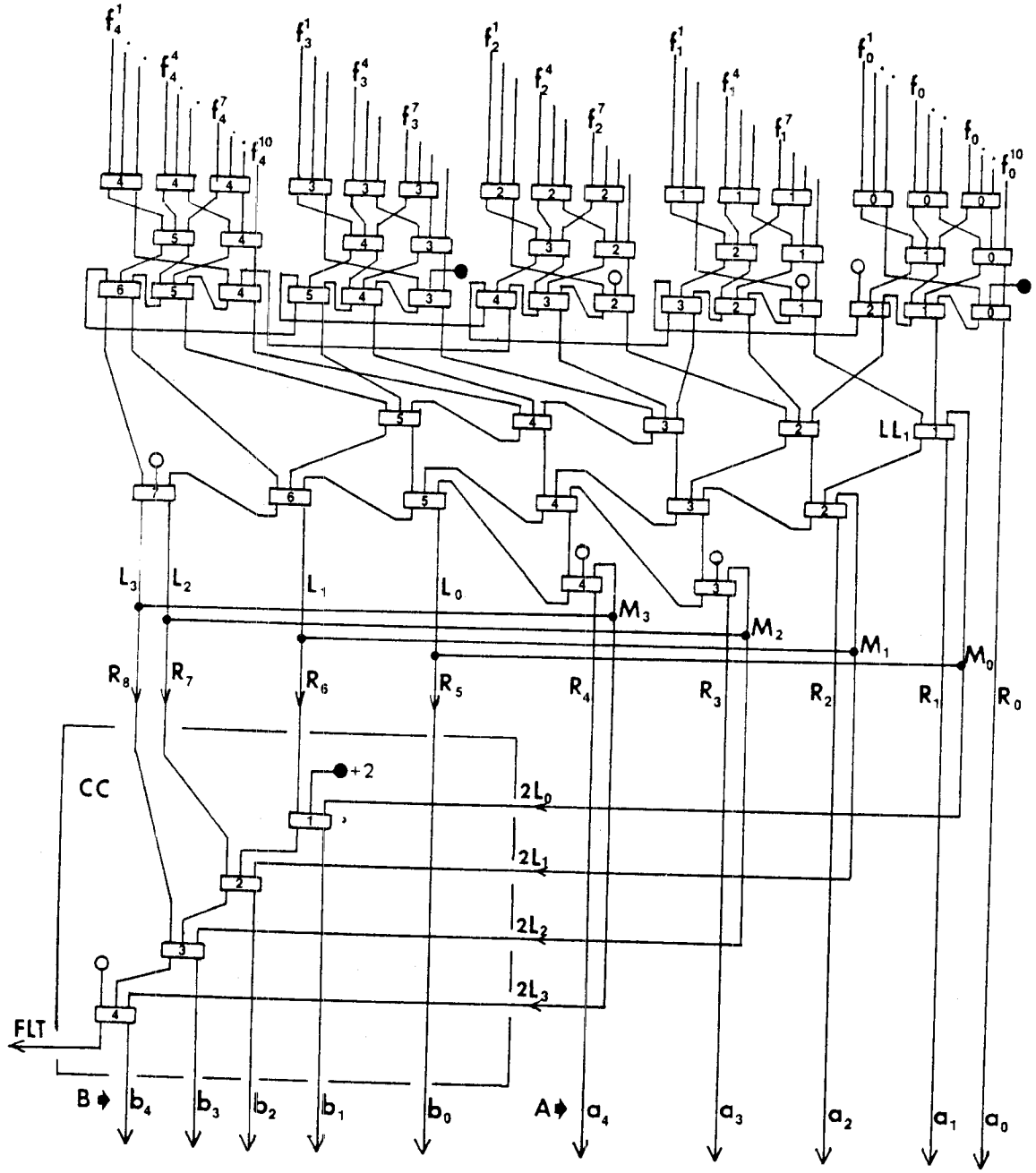

Figure 16: Column Adder and Carry Circuit.

As all column adders in Figure 13 are identical we shall not consider any particular decimal order i of a column adder. The index i will be dropped. Each column adder Σ is entered by 10 decimal digits d_k, k=1,2,..., 10 each encoded by

$$3d_k+2=f_k=f_4^k\cdot 2^4+f_3^k\cdot 2^3+f_2^k\cdot 2^2+f_1^k\cdot 2^1+f_0^k \qquad (6.1)$$

At the top of Figure 16 there are five groups of input wires each belonging to some set $f_j^1, f_j^2,\ldots, f_j^{10}$ with the same value of j. Each group looks like a set of 10 organ pipes and belongs to a fixed j= 0,1,2,3,4 the numbering proceeding from the right to the left. (Note the f_j^k enters a full adder labeled j.).

Figure 16 implements the column addition

algorithm described by (3,10), (3,11), (3,12) and illustrated in Figure 10. To bring both those figures in proper relationship we note how the signal carriers for R and L are described in Figure 16:

The right hand component R is represented by 5 signal wires:

$$R = R_4 \cdot 2^4 + R_3 \cdot 2^3 + R_2 \cdot 2^2 + R_1 \cdot 2^1 + R_0 \qquad (6.2)$$

The left hand component is represented by 4 signal wires:

$$L = L_3 \cdot 2^3 + L_2 \cdot 2^2 + L_1 \cdot 2^1 + L_0 \qquad (6.3)$$

In figure 10 the L-number components belong to the 9-bit total (Figure 16)

$$(R_8\ R_7\ R_6\ R_5\ R_4\ R_3\ R_2\ R_1\ R_0)_2 \qquad (6.4)$$

in the following way:

$$R_8 \equiv L_3,\ R_7 \equiv L_2,\ R_6 \equiv L_1,\ R_5 \equiv L_0 \qquad (6.5)$$

The network adds 2L to the column of digits (Figure 10). To implement this operation $L_t (\equiv R_{t+5})$; t=0,1,2,3 is fed back to be added into a full adder with q=t+1 (notice a shift by 4 binary orders to the right as in Figure 10). For instance the signal of the wire L_0 is fed in the full adder LL_1 with q=1.

Theorem 2. A single fault Φ incident with any hardware element (either a full adder or a connecting link) causes the configuration of output signals $(R_4\ R_3\ R_2\ R_1\ R_0)$ to fall out of the Diamond Code (Figure 5).

Proof. When the circuit is faultless the equations (3. 10,11,12) of the addition algorithm can be written in the form:

$$29 + 3 \sum_{1}^{10} d_k = 30L + R \quad \text{with } 0 \le L \le 9;\ 0 \le R \le 29 \qquad (6.6)$$

Taking this equation modulo 3:

$$2 \equiv R \pmod 3 \text{ for } 0 \le R \le 29 \qquad (6.7)$$

it is proven that the configuration of bits in $R = (R_4 R_3 R_2 R_1 R_0)_2$ belongs to the Diamond Code (see Figure 5).

A single fault ϕ located within any full adder can produce an arithmetic error $\Delta v \in \{0, \pm 2^q, \pm 2^{q+1}\}$ at its output. A single fault ϕ connected with a link induces an arithmetic error $\pm 2^h$. In both cases when the error is non-zero it has the form $\pm 2^j$, where h is an integer. This value becomes added to the column of encoded digits so that (6.6) changes into:

$$29 + 3 \sum_{1}^{10} d_k \pm 2^h = 30\ L + R \text{ with } 0 \le L \le 9;\ 0 \le R \le 29 \qquad (6.8)$$

so that

$$R \equiv 2 \pm 2^h \not\equiv 2 \pmod 3 \qquad (6.9)$$

It is obvious, however, that

$$2^h \not\equiv 0 \pmod 3 \qquad (6.10)$$

so that the number $(R_4\ R_3\ R_2\ R_1\ R_0)_2 = R$ has a configuration of bits which does not belong to the Diamond Code. QED.

Remark. The proof supposed that ϕ is located anywhere except the output wires R_4, R_3, R_2, R_1, R_0 (Figure 16). It is clear that ϕ affecting any of those output wires is detected. On the other hand, where fault ϕ has been detected the value of L either stays correct or is distorted.
Conclusion. Incidence of a single fault with any element of the column adder is detected by checking the output signal configuration $(R_4\ R_3\ R_2\ R_1\ R_0)$ which ceases to belong to the Diamond Code. The condition $R \equiv 2 \pmod 3$ does not hold anymore. The code checking circuit (Figure 6) can be used to do the checking.

Remark. It has been said that an error in the value of the carry C can occur when an error occurs and is detected in the column adder. When the incorrect value of C (representing by B in the Diamond Code) overflows the value 9 the fact is signalized by FLT (Figures 16, 19).

7. Carry Circuit

The decimal carry C = L is defined as the binary number $L = (L_3\ L_2\ L_1\ L_0)_2$. The ripple carry adder (Figure 13) requires an input encoded in Diamond Code. For that reason a Carry Circuit (enclosure in Figure 16) is provided to encode L in the Diamond Code:

$$B = 3L + 2 \qquad (7.1)$$

where

$$B = b_4 2^4 + b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0 \qquad (7.2)$$

The encoding algorithm is very simple and is expressed in a simplified form in Figure 17.

$$\begin{array}{rl} & \ldots\ = L \\ & \ldots\ = 2L \\ & \underline{00010 = 2} \\ \text{C in Diamond Code:} & \ldots\ = 3L + 2 = B \end{array}$$

$$B = b_4 2^4 + b_3 2^3 + b_2 2^2 + b_1 2^1 + b_0$$

Figure 17: Carry Circuit algorithm.

The implementation of the algorithm has a snag, however. Figure 18 shows an implementation which is theoretically correct, but is non SFD. The source of the trouble is the presence of the integer 3 in (7.1). The value of B changes by 3 for each unit of L. For that reason the signal configuration (b_4 b_3 b_2 b_1 b_0) stays in Diamond Code when the fault is incident with L_0, L_1, L_2 or L_3.

Before the solution of this difficulty is explained, the following example is discussed. Supposing L = C = 7 it is $L_3 = 0$, $L_2 = L_1 = L_0 = 1$ (Figure 18). The faultless output is: $b_4 = 1$, $b_3 = 0$, $b_2 = 1$, $b_1 = 1$, $b_0 = 1$ so that C = 7 in Diamond Code. Supposing that the input L_0 is STL (Stuck LOW) the resulting coding $b_4 = 1$, $b_3 = 0$, $b_2 = 1$, $b_1 = 0$, $b_0 = 0$ remains Diamond Code meaningful, corresponding to B = 6.

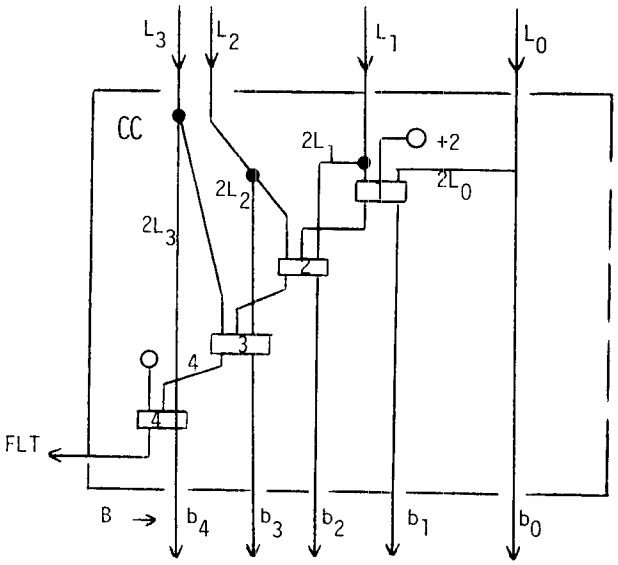

Figure 18: Carry Circuit which is not SFD!

To make this fault detectable (Figure 16) the input L_0 is not connected directly to the input $2L_0$ (as in Figure 18) but the connection is provided by a path including an element of the column adder whose signal is single fault detectable. For instance L_0 is connected with $2L_0$ (Figure 16) by a path passing through the point M_0. A fault incident with M_0, is detected by the configuration (R_4 R_3 R_2 R_1 R_0) being out of the Diamond Code.

8. Ripple Carry Adder

An SFD version of the section AD^i (Figure 13) of the ripple carry is in Figure 19.

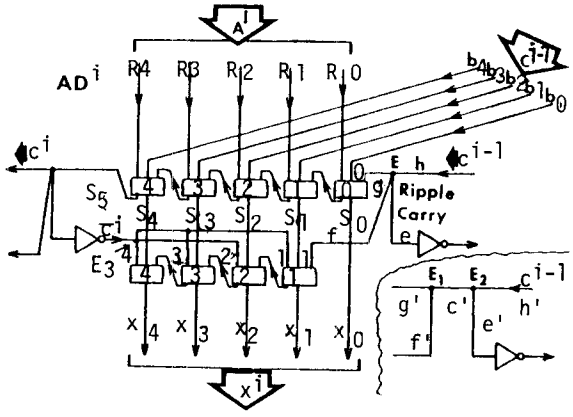

Figure 19: Ripple Carry Adder (SFD version).

The addition algorithm is quite simple:

BEGIN with inputs A^i, C^{i-1} given in Diamond Code: digit A^i enters as binary number $(R_4R_3R_2R_1R_0)_2 = 3A^i + 2$ = R digit C^{i-1} enters as binary number $(b_4\ b_3\ b_2\ b_1\ b_0)_2 = 3C^{i-1} + 2$ = B ripple carry $c^{i-1} \in \{0,1\}$ generated by AD^{i-1} enters at E.

1. EVALUATE $S = R + B + c^{i-1} = (S_5\ S_4\ S_3\ S_2\ S_1\ S_0)_2$
2. $c^i \leftarrow S_5$ and $S' = (S_4\ S_3\ S_2\ S_1\ S_0)_2$
3. IF $c^i = 1$ THEN go to 6
4. $SUM^i \equiv S' + 2c^{i-1} + 30$ (mod) with $0 \le SUM^i \le 31$
5. STOP
6. $SUM^i \equiv S' + 2c^{i-1}$ (mod 32) with $0 \le SUM^i \le 31$
7. STOP

Remarks. All carries are defined during the Step 2: as soon as c^{i-1} is evaluated. It is surprising that

$S' \rightarrow x^i$

in words: the decimal digit x^i of the total is well determined by the value of S' alone (see [2]). The table in Figure 20 shows this functional dependence.

S' =																													
	00	02	04	03	05	07	06	08	10	09	11	13	12	14	16	15	17	19	18	20	22	21	23	25	24	26	28	27	29	31
		0			1			2			3			4			5			6			7			8			9
x^i =																													

Figure 20: The decimal digit x^i of the total is well determined by the value

$$S' = (s_4\ s_3\ s_2\ s_1\ s_0)_2$$ (See Figure 19).

The network in Figure 19 is Single Fault Detecting only as long as the desing principles discussed in Chapter 7 are observed.

First of all the ripple carry c^{i-1} must change the decimal digit x^i of the output by one unit for each 3 units in the Diamond Code. One unit is added through the link g, two units through the link f (Figure 19).

When the fault ϕ affects one of the links g,f it will be detected at the output x^i, because the change in the f-value of the Diamond Code will be changed by either 2^0 or by 2^1. When the fault ϕ affects the link h, then it will affect the point E and the link e will be detected at the output x^{i-1} because the f-value of the corresponding Diamond Code will be changed either by nothing instead of correctly by -2, or by -2 instead of correctly by nothing. E is a point whose single fault is checked by x^{i-1}. The path g-f connecting inputs belonging to full adders with q=o (output S_0) and q=1 (output x_1)--note that $2^0 + 2^1 = 3$--passes through E which is fault checked by the output x^{i-1}.

Note that the network would be not SFD if the wiring is changed as shown in the insert. A fault ϕ incident with the link c' between E_1, E_2 will pass undetected by x^i (change by 3 in Diamond Code) as well as by x^{i-1} (the signal E_2 being correct).

Another structural detail needed for SFD is seen at the point E_3 (Figure 19). Fault incident with this point is detected by the distortion of the Diamond Code of x^i. A path which connects two full adders' inputs whose combined weight is divisible by 3 must pass through E_3. For instance $2^4 + 2^3 = 24$ so that the path connecting the inputs to full adders labeled 4,3 must pass through E_3. On the other hand $2^3 + 2^1 = 10$ so that the path connecting the inputs to full adders labeled 3,1 is not obliged to pass through E_3.

9. Single Fault Detecting Multiplier.

The adder described above can be used to design a SFD multiplier. The multiplication algorithm has two main sections.

1) FETCH the multiplier D^W

2) FETCH the multiplier D^*

During the execution of 2) the d-tuples of $|D|$ are evaluated and stored for d=0, 1, 2,..., 9.

The multiplication is executed with absolute values $|D|$, $|D^*|$ the sign processing being done separately.

Supposing that the numbers $|D|$ have the format as in Figure 1 the d-tuples d $|D| < 10$ will never overflow that format.

Double precision multiplication algorithm is based on the well know pattern in Figure 21.

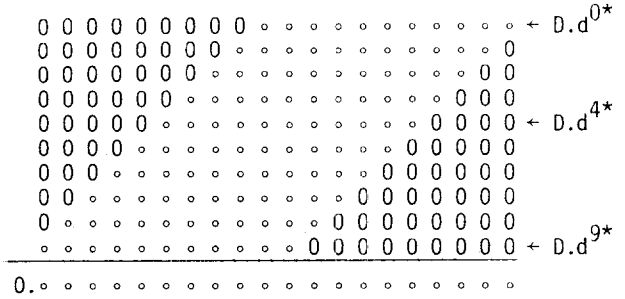

Figure 21: SFD decimally encoded multiplier for 10x10 decimal digit multiplication.

Twenty Column Adders are used. The adders with less than 10 significant digits should be designed to simplify the structure of the corresponding column adders. The execution time will not be improved by that process, however, because the 10 significant digits column's execution time will remain dominant.

To assess the performance of the multiplier we suppose that SN54H183 chips are used to build the column adder. Then the execution time of that adder, including the delay in the carry circuit CC (Figure 16) will be about 100ns. Using the chips SN5483A and SN74H183 to build the Rippled Carry Adder we get the delay of less than 450ns. The total execution time (double arithmetic) with MSI design will be less than 550ns. (twenty decimals product).

References

1. R.M. Diamond, "Checking Codes for digital Computers," Proc. IRE, Vol. 43, pp. 487-488, April 1955.

2. David T. Brown, "Error detecting and Correcting Binary Codes for Arithmetic Operations," Trans. IRE, Vol. EC-9, pp. 333-337, September 1960.

3. A. Svoboda, "Arithmetic Properties of Brown Codes," Symposium of IFIP Congress, New York 1965.

4. A. Avizienis, "Digital Fault Diagnosis by Low-Cost Arithmetical Coding Techniques," Tech. Report No. 32-1476, Jet Propulsion Laboratory, August 1970.

5. D.A. Anderson, "Design of self-checking digital Networks using coding Techniques," Technical Report R-527, Coordinated Science Lab., University of Illinois, Urbana, Illinois.