Computation With Simultaneously Concurrent Error Detection Using Bi-Directional Operands

L. G. Chen and T. H. Chen

Department of Electrical Engineering,
National Taiwan University,
Taipei, Taiwan, R. O. C.

ABSTRACT

In this paper a novel original method of computation with simultaneously concurrent error detection using bi-directional operands (BIDO) has been presented. This technique will provide an efficient and optimal design for arithmetic unit with fault tolerance. The proposed technique holds all capabilities of both space redundancy and time redundancy methods, but does not inherit their drawbacks. The major features of BIDO are: (i) There is less hardware overhead than RESO styles; and (ii) It needs hardly any redundant time. These capabilities and features will make BIDO more suitable for designing arithmetic units of VLSI systems, which basically require high performance and high reliability.

I. Introduction

With the advance of VLSI technology, large integration of processing elements, which can cooperate with each other to achieve high-speed computation, is economically feasible. Since any fault may seriously destroy the operation of the system, high reliability will be the most important requirement for high-speed system. Efficient fault-tolerant techniques should be incorporated in order to ensure the valid results after long computation. Lately, lots of general fault-tolerant techniques of ALU [1-6] by using either space redundancy or time redundancy, have been presented. The typical structures of these methods can be classified into two categories: space-redundant system and time-redundant system. The space-redundant system takes complete duplications of its full fundamental blocks. The obvious drawback of this category is its expensive hardware. The time-redundant system takes double time to recompute the result. In this category, some blind points would inherently exist by using same computing procedure, and the extra recomputing time will seriously degenerate the throughput. The RESO-style structure is the most popular system, which also utilizes time-redundant technique but intends an encoding-decoding operation. Although RESO has been implemented in several systems, there still are a lot of theoretical drawbacks, such as: great hardware overhead, long recomputing cycle, and lower throughput. Fortunately, an efficient and optimal solution for CED will be proposed here, which can provide parallel processing of recomputing and normal computing.

II. Methodology

Due to the data dependence and computing sequence, in general processing units, it always exists that some computing elements are idle in several time intervals. For example, the FAs in the last row of an array multiplier have no effective operations in the beginning of the multiplication. These elements with idle or ineffective operations are called NOP elements. In conventional researches, the capability of fault tolerance is got by means of time redundancy or space redundancy. In space redundancy, they used duplicated hardware as redundancy. In time redundancy, they used extra recomputing as redundancy. In our proposed method, the NOP elements are used as redundant devices. Duplicated data are concurrently input as space redundancy techniques do, but with the opposite direction. Recomputation is executed in the same time that normal operation is progressed. This takes a significant difference with time-redundant techniques, which need another recomputing cycle. Because the computing is performed with bi-directional operands, it is called BIDO technique, which also implies the bi-directional operation (=DO). For more understanding, we will apply the BIDO on an array multiplier as an example. The operation of general array multiplier can be illustrated in Fig. 1. The processing flow is from upper-right corner to the lower-left corner. The line b-d acts as an interface line, the array can be partitioned into M_1 and M_2. Assume that the operation time of a complete multiplication cycle is T_{mul}. It can be written as $T_{mul} = T_1 + T_2$, where T_1 and T_2 are the execution time of parts M_1 and M_2, respectively. Obviously, in time interval T_1, elements in M_1 are active and elements in M_2 are NOP elements. In time interval T_2, elements in M_2 are active and alternately elements in M_1 are NOP elements.

Fig. 2 shows the computing sequences in an array multiplier with BIDO style. At first, the design of FA elements used in general array multiplier is slightly modified. Bi-switches are added to each element. The array multiplier is then partitioned into three parts. Two of them are symmetrical parts M_1 and M_2. The other one is central part C, which includes two identical column arrays C_1 and C_2. The new diagram is shown in Fig. 3. A complete multiplying operation, either normal computing or recomputing, can then be split into two half-computations. In normal operation, the elements in M_1 and C_1 are active and elements in M_2 and C_2 are idle, at time interval T_1. These actions are just like the executing process in general array multiplier. Since M_2 and C_2 are NOP elements in T_1, by accurately controlling the bi-switches in these elements, duplicated data of multiplicand and multiplier can then be simultaneously input to array in opposite direction. It means the first half-computation of the recomputing can be executed at the same time as that of normal computing do. The progressing flows of them are in opposite direction as shown in Fig. 2.

After their first half-computations are both executed completely, these two partial results are held at the C_1 and C_2, respectively. All bi-directional switches are then turned on to change the propagation direction. Thus, C_1 and M_2 are

connected and so as C_2 and M_1. It is clear that the data flows of two computations are not changed, regardless of these switches' actions. Therefore, at time interval T_2, M_1 and C_2 are active for recomputing, and M_2 and C_1 are active for normal computing. Finally, both results can be obtained at the same time. The total computing time, including normal computing and recomputing, will be almost equal to that of general multiplying. It means that the redundant time for recomputing approaches zero. Comparing these two calculated results, any mismatch will indicate the existence of faulty state during computation. Hence, error detection can be achieved simultaneously during computation.

III. Concurrent error detection

The BIDO architecture uses two data sets: (m, q) and ($\overline{m}$, $\overline{q}$), where $\overline{m}$ = m and $\overline{q}$ = q. The results can be expressed as P and $\overline{P}$. Although both P and $\overline{P}$ are calculated by the same array, their processing sequences are in opposite direction. Thus, we can let P =f(m, q) and $\overline{P}$ = f'($\overline{m}$, $\overline{q}$).

For simplicity, the discussion of fault model is based on abstraction level (or functional level). It means that our emphasis is on the verification of the functional behavior regardless of the precise parts in a functional block. This assumption is reasonable because the major requirement of most fault-tolerant systems is to certify whether every functional unit (FU) is right. Otherwise, a reconfiguration procedure is utilized to isolate the faulty FU. Based on the principles of time redundancy, mappable correct output and disjoint error set had been described before. Let E and E' denote the sets of all possible erroneous output bits of two computation functions $f_F(x)$ and $f'_F(x)$, with a fault F. An error is detectable if and only if $f_F(x) \neq c^{-1}(f_F(c(x))) = f'_F(x)$, i.e. $E \cap E' = \phi$.

For convenience, let ER represent the range of all possible errors of $f_F(x)$, with a fault F. Then ER can be written as ER = (e_{max}, e_{min}), where e_{max} is the maximum magnitude of errors and e_{min} is the smallest nonzero magnitude. Typically, a faulty $BFA_{(i)}$ will affect its output sum and carry-out of bit-slice i. This faulty BFA can be caused by FA faults, interconnection line faults, and bi-switches faults. For example, suppose there is a faulty BFA located at the column 1 of a 4-bit array multiplier, as shown in Fig. 4. This fault will induce the off values $\pm 2^1$, $\pm 2^2$, $\pm 3*2^1$ of the partial products in normal computing and the off values $\pm 2^5$, $\pm 2^6$, $\pm 3*2^5$ in recomputing. Thus, the primary error set is E = {$\pm 2^1$, $\pm 2^2$, $\pm 3*2^1$}, and the error range is ER = (e_{max}, e_{min}) = ($3*2^1$, 2^1). The recomputed error set is E' = {$\pm 2^5$, $\pm 2^6$, $\pm 3*2^5$}, and its error range is ER' = (e'_{max}, e'_{min}) = ($3*2^5$, 2^5). Since $e_{max} < e'_{min}$, it is satisfied that $E \cap E' = \phi$. It means that the fault in Fig. 4 is detectable. The error detection capabilities of BIDO can be denoted by the following significant theorems. All statements and proofs only refer to multiplication problems and can be trivially extended to other operations, such as addition, subtraction and division.

[Theorem 1] All possible errors due to any single faulty BFA can be detected.

proof: Suppose that there is a faulty BFA in column i. This faulty BFA may be at M_1, C, or M_2. If it is in M_1, then $0 \leq i \leq n-2$. We can obtain $e_{max} = 2^i + 2^{i+1} = 3*2^i$ and $e_{min} = 2^i$. Hence, the primary error set is E = {$\pm 2^i$, $\pm 2^{i+1}$, $\pm 3*2^i$} and it's range is ER = ($3*2^i$, 2^i). During recomputation, the generated error weighting terms are 2^{2n-i-2} and 2^{2n-i-1}, which can be concluded with E' = {$\pm 2^{2n-i-2}$, $\pm 2^{2n-i-1}$, $\pm 3*2^{2n-i-2}$} and ER' = ($3*2^{2n-i-2}$, 2^{2n-i-2}). Since $e_{max} < e'_{min}$, we can obtain $E \cap E' = \phi$ and the possible errors due to BFA failure at arbitrary bit-slice i can be detected. For the same reason, if the faulty BFA is in M_2, we can devise ER = ($3*2^i$, 2^i) and ER' = ($3*2^{2n-i-2}$, 2^{2n-i-2}) under the case $n \leq i \leq 2n-2$. Alternately, there is $e'_{max} < e_{min}$, so errors resulting from faulty BFA located in M2 are also detectable. The last case is i = n-1 which denotes the faulty BFA is in the central part C. Thus, the primary result P is incorrect and the recomputed result $\overline{P}$ is correct, since column n-1 is good. So, we can achieve error detection owing to $P \neq \overline{P}$. It is clear that the failed BFA occurred in column n-1 can also be detected by good column n-1. It means that the columns in C have the capability of self-detecting. If both columns are not failed simultaneously, they are detectable. Therefore, it can be concluded that all possible errors due to any arbitrary single faulty BFA in bit-slice i, where $0 \leq i \leq 2n-2$, can be detected.

[Theorem 2] The errors resulting from single fault region with adjacent multiple faulty BFAs confined to M_1 or M_2 can be detected.

[Lemma 2.1] The maximum detectable error range of adjacent multiple faulty BFAs is ER = (2^n - 1, 1).

[Theorem 3] The errors resulting from discrete faulty BFAs, except the symmetric constraint, can be detected.

[Symmetric Constraint] Suppose that there exists a faulty BFA in column i, it may be undetectable if any BFA located in column 2n-i-1, 2n-i-2, or 2n-i-3 is faulty.

As described in section II, the most significant feature of BIDO is that the both parts M_1 and M_2 utilize each other to certify their results. It's clear that we will have the inherent drawback that the error may not be detectable, if the symmetric constraint is met. This is intractable because it always needs at least one good element to certify a failed element. Since there is the larger robust detectable error range (= (2^n - 1,1)) of BIDO, it should be enough to tolerate the general faulty cases including all single-bit errors and various multiple-bit errors.

IV. Performance analysis

The performance of the BIDO approach has been analyzed. For the purpose of clearly pointing out the special features of the BIDO method, the most popular techniques, RESO-r and enhanced RESO-r [2], are referenced. The value of hardware redundancy is calculated by utilizing the data of CMOS technology, denoted in Table I. The analyzed results are shown in Table II and Table III. Because there is only one duplicated column in the central part, the extra hardware requirement is reduced as possible. The bi-directional switches duplicated column in the central part, the extra hardware requirement is reduced as possible. The bi-directional switches are another hardware overhead, but they occupy small area and therefore provide small weight on the hardware overhead ratio. Due to the simultaneous operations of normal computing and recomputing, the redundant time nears zero. This should be the most significant advantage of BIDO. The only delay is the holding time in central part before switches turn on. The fault model accepted in this architecture is unlimited. All faults including temporary faults and permanent faults (especially stuck-at faults) can be detected concurrently. Larger fault region can be allowed than previous researches. With all these mentioned features, the control strategy of BIDO is still very simple. Only the control signal of the bi-switches is required. Of course, there must be some sacrifice, the number of interconnection lines in each element may increase. The reliability of the interconnections and the increment of the

routing complexity should be under consideration. Nevertheless, it has been shown that the BIDO technique will provide a good alternative design for fault-tolerant computing.

Table III shows the compared results of BIDO, RESO-r and enhanced RESO-$\bar{r}$, under the same detectable error region, it means that the shifting parameters: r and r are supposed equal to n/2 and n, respectively. The comparison is concentrated on the hardware overhead and time redundancy in the 16-bit and 32-bit systems. The hardware overhead ratio of BIDO structure is near one fifth of those of RESO structures. Neglecting the switching time, the time overhead in BIDO approaches to zero. But for the same detecting capability, the time overhead is near 200% in RESO-styles. Based on these features, it has been shown that BIDO approach will be more suitable for CED systems.

V. Conclusion

A novel CED method for fault tolerance has been presented. It has been shown that the proposed technique, BIDO, can hold all capabilities of both space redundancy and time redundancy methods. Due to the lack of an extra recomputing cycle, it almost doesn't need any redundant time. Only the elements in central part C require a redundancy, it is obvious that hardware redundancy is also small. In this paper, multiplication is shown as an example. As the experimental results show, BIDO should be the most available technique for various significant fault-tolerant processing.

References

[1] S. W. Chan and C. L. Wey, "The design of concurrent error diagnosable systolic array for band matrix multiplications," *IEEE Trans. CAD*, Vol. 7, No. 1, pp.21-37, Jan. 1988.

[2] S. W. Chan, S. S. Leung, and C. L. Wey, "Systematic design strategy for concurrent error diagnosable iterative logic arrays," *IEE Proceedings*, Vol. 135, Pt. E, No. 2, pp.87-94, March 1988.

[3] Ronald J. Cosentino, "Concurrent error correction in systolic architectures," *IEEE Trans. CAD*, Vol. 7, No. 1, pp.117-125, Jan. 1988.

[4] Jacob A. Abraham, Prithviraj Banerjee, Chien-Yi Chen, W. Kent Fuches, S. Y. Kuo and A. L. Narasimha Reddy, "Fault tolerance techniques for systolic arrays," *IEEE Computer*, pp.65-75, July 1987.

[5] J. H. Patel and L. Y. Fung, "Concurrent error detection in ALU's by recomputing with shifted operands," *IEEE Trans. Computer*, Vol. C-32, pp.589-595, July 1982.

[6] J. H. Patel and L. Y. Fung, "Concurrent error detection in multiply and divide arrays," *IEEE Trans. Computer*, Vol. C-32, pp.417-422, April 1983.

TABLE III

Comparison Of The SPACE/TIME Redundancy By * RESO&BIDO Under The Same Detectable Error Region.

way / n bits	RESO-r		Enhanced RESO-$\bar{r}$		BIDO	
	space	time	space	time	space	time
16	142%	200%	113%	200%	24%	~ 0
32	133%	200%	106%	200%	20%	~ 0

* : denotes r = n/2, $\bar{r}$ = n.

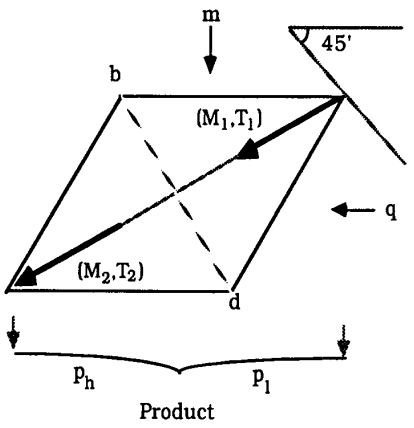

Fig.1. The operating process of general array multiplier.

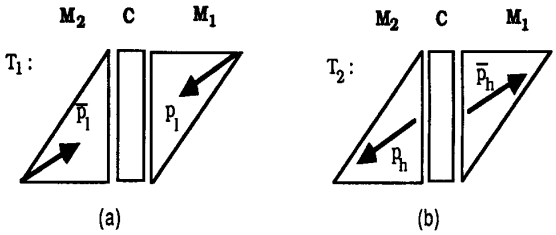

Fig2. (a) The first half-computation.
(b) The second half-computation.

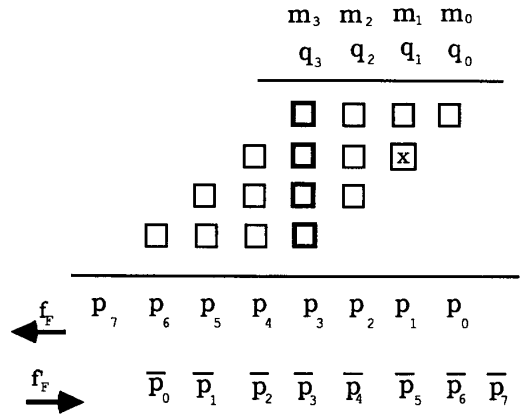

Fig.4. A 4-bit multiply operation with a faulty BFA.

TABLE I

Hardware Requirement of Components

Logic Function /per bit	FA	BFA_x	Shifter	Equality checker	Bi-switch
No. of required CMOS gate-pair	31	36, 32	23	10	1

TABLE II

Comparison Of n-bit Multiplier By RESO&BIDO.

Item \ Way	RESO-r	Enhanced RESO-$\bar{r}$	BIDO
Hardware overhead ratio	$(r/n)^2 + 2r/n + 8/3n$	$\bar{r}/n + 2/n$	$1/7 + 3/2n$
Time redundancy ratio	$1 + 2r/n$	$1 + \bar{r}/n$	~0
Maximal detectable error region*	$2r-1$	$\bar{r}-1$	$n-1$
Round-off error	no		no
Detectable fault model	temporary & permanent		temporary & permanent
Extra connections	less		more
Control complexity	easy (shifting way)		more easy (bi-switching way)

* : denotes permitted maximal adjacent faulty bit-slices for BIDO.

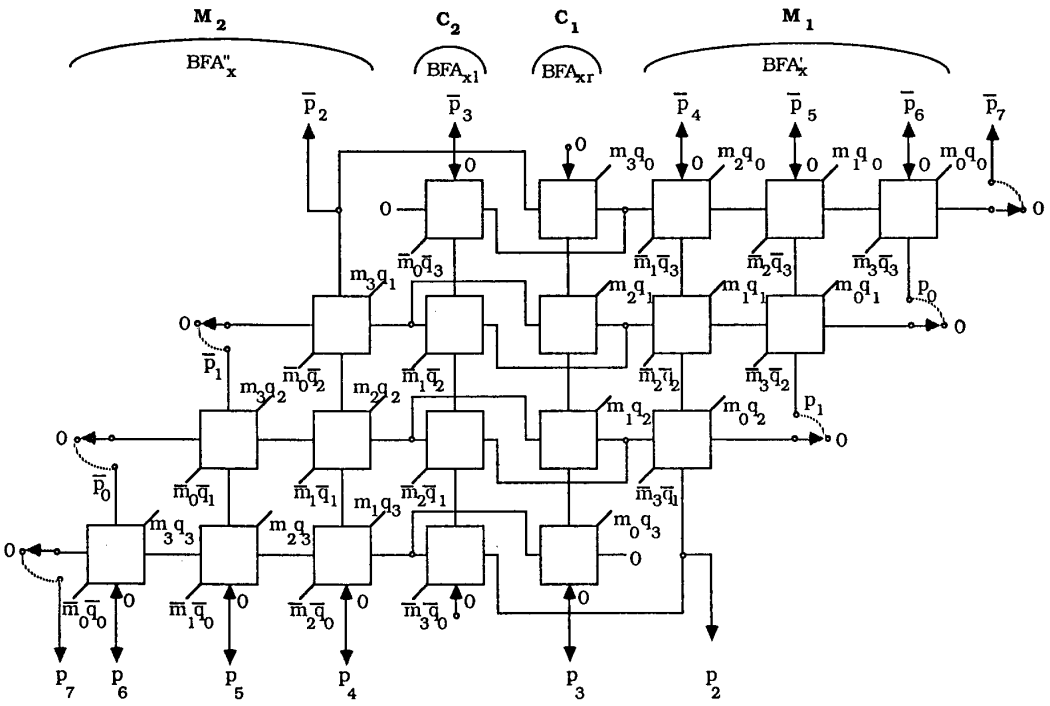

Fig.3. A 4-bit array multiplier with bi-directional full adders(BFA).