Fault-tolerant serial-parallel multiplier

Prof. L.G. Chen, PhD
T.H. Chen

Indexing terms: Concurrent error detection, Digital circuits, Fault tolerance

Abstract: The paper presents a novel fault-tolerant circuit design using a time-redundancy method for a serial-parallel multiplier, which is useful in DSP applications with serial data transmission. It utilises the RECO (REcomputing with Circularly shifted Operands) technique to detect errors concurrently. A simple OR-gate based circuit is used as the location table to identify faulty bit-slice pairs. The reconfiguration technique is then introduced to bypass the potential faulty bit-slices. This design can have the maximum detectable error region ($\cong n/2$ bits), without appending extra computing elements. The latency from error detection to location is only about two clock cycles, i.e. almost real-time detecting can be achieved. Pipelined timing for two computations is illustrated. The analyses of performance and complexity are described. The results show that this is an efficient design methodology for fault-tolerant multiplication with serial data.

1 Introduction

To achieve high performance in digital signal processing, almost all processors for this application require multipliers to perform massive computation operations. Taking into consideration circuit complexity, hardware area, pin limitations, execution speed, and data transfer type, the serial-parallel multiplier is most suitable for digital signal processing. Some serial-parallel multipliers with pipelined structure, high speed, or low cost have been presented [1–3]. With the advance of VLSI technology, large integration of the processing elements, which can cooperate with each other to accomplish massive computation tasks, have resulted in higher speed. However, owing to the increasing complexity and density of VLSI circuits, any failure of the chip may seriously affect all the operations of the system, and hence high reliability will be of increasing importance in future design. Consequently, a fault-tolerant serial-parallel multiplier becomes more and more attractive and should be incorporated to ensure valid results in long computations.

Many fault-tolerant techniques in arithmetic units, using redundancy, have been presented [4–14] in this decade. Unfortunately, some serious problems still exist. The N -tuple modular redundancy (NMR) system [14] is a well-known technique, using a voting method to mask errors resulting from failed modules. The situation of

NMR is $R = 2 * E + 1$ where R denotes the number of redundant modules and E is the number of faulty modules to be tolerated. For example, three-tuple modular redundancy (TMR) with $R = 3$ will ensure correct results in spite of any fault existing in one of three modules, but it requires at least a 200% hardware overhead. Another effective approach is to use a concurrent error detection (CED) scheme, which can detect errors during normal operation and may need lower cost to achieve fault tolerance by means of fault location and error correction.

According to the redundancy strategy, the CED techniques are divided into two categories: space-redundancy systems and time-redundancy systems. The space-redundancy system takes complete duplications of its full fundamental blocks. The obvious drawback of this category is its expensive hardware. The time-redundancy system takes twice the time to recompute the result, and some blind points would inherently exist, owing to the use of the same computing procedure, and the extra recomputing time will seriously degenerate the throughput. The self-checking technique [4] requires a 73% hardware overhead for detecting single-bit errors and 94% for detecting unidirectional errors. It is appropriate for conventional stuck-at-fault models, which are not fully suitable for VLSI failure.

The alternating logic design [5], a time-redundancy approach, utilises the complement of the original data during recomputation and requires about 85% hardware overhead to achieve its single-fault detection. It is only applicable to combinational logic circuits with a self-duality property, and hence is not suitable for some complex operations, such as serial-parallel multiplication. The BIDO technique [15] can perform normal computation and recomputation at the same time to support the real concurrent error detection with near zero time overhead. In particular, it requires only about 24% hardware overhead for a parallel-array multiplier. However, it requires basic symmetric processing and a cellular array structure and so it cannot be applied to a serial-parallel multiplier.

RESO [7, 8, 12, 15] is a widely-used CED technique which utilises time-redundancy to provide efficient concurrent error detection for ALUs. It can detect errors with unknown failure properties in a VLSI circuit. But this technique requires a large area-time overhead, especially for high numbers of shifting bits. The use of rotated operands [10] can eliminate additional bit-slices, but it is cost-effective only for logic operations and ripple-carry addition.

RESWO [15] is an improved RESO method and solves some of the problems associated with RESO, but it is difficult to apply to a serial-parallel multiplier. Obviously, it is necessary to develop an efficient method for the fault-tolerant design of a serial-parallel multiplier.

Paper 8025E (C2), first received 15th January 1990 and in revised form 18th June 1990

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

In this paper, we propose an error-detecting technique, called RECO (REcomputing with Circularly shifted Operands), for designing a general serial-parallel multiplier. This technique not only overcomes the previously mentioned difficulties but also provides the shortest detecting latency and is the most suitable technique for serial-parallel multipliers so far developed. In the following section, we will describe the RECO scheme. Section 3 shows how to design a serial-parallel multiplier with RECO. In Section 4, we discuss the capabilities of error detection in a serial-parallel multiplier using RECO. The procedures for fault location and recovery are described in Section 5.

2 Methodology of RECO scheme

The essence of fault tolerance is in redundancy. Therefore, hardware redundancy or time redundancy is inherently required. In the time-redundancy method, it is efficient to use a recomputation step. In the past, time redundancy has been applied primarily to the detection of transient faults. However, there now appears one important potential of time redundancy which implies the ability to detect permanent faults while using some extra hardware. Fig. 1 shows this encoding-decoding circuit, where c is the encoding function and c^{-1} is the decoding function. But this induces an intractable and challenging problem, i.e. how to derive a suitable encoding-decoding function with more robust detecting capability but less extra hardware. It is also difficult to improve the recognition capability of disjoint error sets [13], also explained in Section 4, to improve the detection capability on the adjacent-fault, which always exists in VLSI circuits. As previously presented [7], in bit-sliced or cellular architectures with VLSI technology, it is cost-effective to perform concurrent error detection using RESO techniques. But this would require a great hardware overhead and extra computing time for large numbers of shifting bits.

Fig. 1 Time redundancy with coding technique to detect permanent faults

- $\odot$ encoding function
- $\odot^{-1}$ decoding function

There are two observations: for recognising disjoint error sets, adding extra bit-slices is not a good solution, rather fold the data and recompute it by other independent bit-slices which are sufficient and available; for static design, carry splitting is easy to route. According to these observations, the recomputing methodologies using alternative operands with a modified input scheme can be proposed. Let us use a ripple-carry adder as an example, shown in Fig. 2, to illustrate this scheme. Two single-switch gates and one interconnection are inserted into the carry-out terminal of the MSB bit-sliced full adder and the carry-in terminal of the LSB bit-sliced full adder. For the purpose of error detection, we need two computations: primary computation and recomputation. In primary computation, a result is calculated and stored in a register in order to be referenced. Then, the operands are shifted circularly by $n/2$ bits and the switches are turned on alternately. An auxiliary result is derived by

recomputing and inverse shifting. Thus, any error can be detected when inconsistency exists between two results.

Fig. 2 RECO- $n/2$ scheme of an n -bit ripple-carry adder

At a glance, the result of this example seems to be the same as in RESO with a rotated approach, and as in RESWO. But for a serial-parallel multiplier, one of its two data inputs is transmitted bit-by-bit and hence the partial product is derived bit-by-bit. This makes both methods impracticable.

RECO can allow partial results to be shifted circularly. Also, the number of shifting bits is variable and can be determined by the degree of detection capability and fault location which will be described later. These are the special features which other methods cannot provide. Fig. 3 compares the three methods: RESO, RESWO and RECO. Because RECO utilises the original n bit-sliced operators to obtain the auxiliary result for checking, the large area-time overhead can be reduced. The application of the RECO technique to the serial-parallel multiplier is described in the next section.

Fig. 3 Two-step computations of three CED techniques: RESO, RESWO and RECO

- $\boxtimes$ extra r bit-sliced operators with zero data input
- UF upper-half bit-slices
- LF lower-half bit-slices

3 Designing a serial-parallel multiplier with RECO

A general serial-parallel multiplier (Fig. 4) takes $2n - 1$ clock cycles to multiply two n -bit data. The clock period is about βt_{FA} where β is the margin required to ensure the adequate decay of transients so that the correct infor-

Fig. 4 n -Bit serial-parallel multiplier

- $\oplus$ 1-bit full adder
- $\square$ 1-bit carry flip-flop
- $\square$ 1-bit D flip-flop

mation is set into the storage cells at each clock cycle, and typically ranges from 2 to 6; t_{FA} is the full adder delay [2]. Because of the utilisation of the carry-save add-shift method, all bit-sliced operators calculate simul-

When designing with RECO, because of the inherent constraint [10], the number of circularly-shifted bits cannot be more than half the original bit-slices. An n bit-sliced serial-parallel multiplier of RECO- $n/2$, shown in

Fig. 5 Fault-tolerant n -bit serial-parallel multiplier

taneously and produce the partial products at the end of each clock interval. In other CED techniques, error detection is always performed after the whole multiply operation is completed. Its detecting latency, at least $2n - 1$ clock cycles, may be so long as to cause heavy damage in real-time computing systems. For this reason, we take each partial product sum from the output of the full adder once after every addition. That is, the multiplier will compare both the primary result and the recomputed result of each partial product addition. Any error should be detected in every adding procedure. As a consequence, it can reduce the error detection latency to about two clock cycles.

Fig. 5, is implemented by first replacing the original carry flip-flops and delay elements with circular shifters C, D. Two extra circular shifters A and E are added to shift the input data for recomputation and hold the partial product to be checked, respectively. Two pairs of switch gates are inserted into the MSB bit-sliced adder and the central bit-sliced adder. They provide two different computing procedures for primary computation and recomputation. First, each summand $x_i y_j$ is loaded into shifter A, and in the mean time, is added to the data from C, D during primary computation. Then the resultant sum is loaded into E for checking and the carry-out is discarded. This is because the contents of shifters C and D should

be original operands for recomputation. Second, shifters A, C, D and E start to shift left circularly by $n/2$ bits. Third, recomputation is carried out and the result is compared with the primary sum stored in E by the equality checker. When any mismatch exists in both results, an error signal is produced. Otherwise, the auxiliary sum will be loaded into shifters C and D for the next partial-product addition. Then, C, D and E are shifted back to their normal states, and the next operating cycle begins. The timing sequences of the n -bit multiply operation is shown in Fig. 6. From these results, it is obvious that concurrent error detection can be achieved in about two clock cycles.

4 Error detection capability

Some common concepts of error detection and the error detection capability of the RECO scheme are described in this section. In general, the nature of failure in VLSI is not precisely known, but the effects of faults within the higher functional modules can be described with accurate and tractable models, as in [14]. For simplicity, we use the functional fault model throughout this paper. The fault model accepted by this architecture covers temporary faults and permanent faults (especially stuck-at faults). In addition to FA faults, the detectable fault cases also include AND-gate faults, connection line faults and shifter faults. Let E and E' represent the error sets (not including zero, here) of primary computation $f_F(x)$ and recomputation $f'_F(x)$ respectively, with the fault F . In Fig. 1, based on the principles of mappable correct output and disjoint error sets in time redundancy [13], it is obvious that an error is detectable if and only if, $f_F(x) \neq c^{-1}(f_F(c(x))) = f'_F(x)$, i.e. $E \cap E' = \phi$.

Owing to the data dependence and computing sequence, error propagation problems may exist when operating on a faulty bit-sliced functional unit. In RECO, the error detection capability can be classified into logic and arithmetic categories by operation. Since there is no problem of error propagation existing in logic operators, RECO- k can detect all errors generated by k adjacent faulty bit-slices. Most of the bit-sliced arithmetic operators have the inherent characteristic of intermediate result transfer, and then even a single faulty bit-slice may cause erroneous output bits. Because of this, the number of detectable failed bit-slices may be reduced so that it is smaller than k . For example, in a ripple-carry adder, it requires at least $k+1$ bits of circular shift to detect k adjacent faulty FAs. But because there is no carry propagation in a serial-parallel multiplier, errors resulting from a single faulty bit-slice will be confined to one erroneous output bit only. Therefore, a serial-parallel multiplier

with RECO- k can detect at most k adjacent faulty bit-slices. This is proved in the following theorem.

Theorem: In an n bit-sliced serial-parallel multiplier of RECO- k , the errors resulting from k adjacent faulty bit-slices can be detected.

Proof: Suppose that the related adding circuits of bit-slice i have some faults; then only partial-product bit i may be erroneous during the primary computation and we can obtain $E = \{\pm 2^i\}$. In the recomputation procedure, this faulty adding circuit will result in $E' = \{\pm 2^{i+k}\}$. Thus, $E \cap E' = \phi$ and the equality checker will give error signals. Next, we assume that k faulty bit-slices are distributed adjacently from bit-slice i to bit-slice $k+i-1$, as shown in Fig. 7. Since there is no carry error propagation problem, the partial products $pp_i \sim pp_{k+i-1}$ may be erroneous for the primary computation. But during recomputation, we can obtain correct partial products $pp'_i \sim pp'_{k+i-1}$, since those computing elements which are located from bit-slice 0 to bit-slice $i-1$ and from bit-slice $k+i$ to bit-slice $n-1$ are valid. Therefore, the architecture of RECO- k can detect at most k adjacent faulty bit-slices.

There is still the inherent constraint $k \leq n/2$ in an n bit-sliced serial-parallel multiplier with RECO- k . But, the allowable fault region of almost half of the original structure should be enough to tolerate the general faulty cases including all single-bit errors and various multiple-bit errors for concurrent error detection.

5 Procedure of fault location and recovery

Fault tolerance involves error detection, fault location, and error correction or reconfiguration by bypassing the faulty unit. Before a system can correct errors or reconfigure to recover normal operation, it must first identify faulty components. As mentioned in many previously presented papers, a single fault is difficult to locate, because of its rare occurrence and the lack of further available information. For example, a RESO-based method of time redundant fault-location in bit-sliced ALUs [17] takes a larger set of suspicious faulty bit-slices as a sacrifice for fault-isolation. In practice, many fault-tolerant architectures, because of their great-circuit complexity and the extra delay required for fault location and recovery, have not to date been implemented owing to the large area-time overhead.

By using the RECO technique, it is easy to achieve error location and to recover the normal operation when an error is detected. As described in previous sections, there is no carry propagation in a serial-parallel multi-

Fig. 7 Distributed-fault with k adjacent faulty bit-slices in the serial-parallel multiplier of RECO- k

$pp_i \sim pp_{k+i-1}$ and $pp'_{i-k} \sim pp'_{i-1}$ are incorrect

plier. Hence, any failure occurring in bit-slice i will only result in erroneous partial product pp_i in primary computation, incorrect pp'_{i-k} during recomputation, or both. In

Fig. 8 Fault transition in the serial-parallel multiplier of RECO- k
 • faulty bit-slice
 pp possible erroneous partial product

Fig. 8, if there are two faulty bit-slices, located at position bit i and bit $i - k$, it will generate at most three error signals e_{i-k} , e_i and e_{i+k} . In those error signals, e_i and e_{i+k} are induced by erroneous pp_{i-k} and pp_i respectively during primary computation, and e_{i-k} and e_i induced by incorrect pp'_{i-2k} and pp'_{i-k} during recomputation. Apparently, e_i is the common error signal for both computations and we cannot determine which failed bit-slice it belongs to. This implies that we do not know which bit-slice, i or $i + k$, has failed when there is only one error signal e_i detected. In conclusion, if there is only one faulty bit-slice and we detect an error signal e_i , we will have two possible candidates, bit-slice i and bit-slice $i - k$ for $i - k \geq 0$ (or bit-slice i and bit-slice $i - k + n$ for $i - k < 0$).

Based on the above knowledge, we have the following formula to locate the potential faulty bit-slice pair. In an n -bit serial-parallel multiplier of RECO- $n/2$, the error signal set $\{e_i, e_{i+n/2}, (e_i, e_{i+n/2})\}$ for fault location can result from faulty bit-slice i , bit-slice $i + n/2$, or both, under the condition $0 \leq i \leq n/2 - 1$.

As soon as one or all of these error signals are detected, we can replace the potential faulty bit-slice pair, while in position bit i and bit $i + n/2$, with two spare bit-slices in order to recover normal operation. This circuit design for fault location can be implemented using simple OR-gates. A fault-tolerant serial-parallel multiplier with spare allocation is shown in Fig. 5. We can give a practical design for an 8-bit serial-parallel multiplier using RECO-4. The faulty bit-slice 0 will generate the error signal set $\{e_0, e_4, (e_0, e_4)\}$, where (e_0, e_4) denotes error signals e_0 and e_4 detected at the same time, and the faulty bit-slice 4 generates the same error signal set. If the single faulty bit is at bit-slice 0 or bit-slice 4, and two faulty bit-slices occur in bit 0 and bit 4, then these have the same set of error signals $\{e_0, e_4, (e_0, e_4)\}$. For two potential faulty bit-slices 1 and 5, there is still the same error signal set $\{e_1, e_5, (e_1, e_5)\}$. For the same deduction, attributes of total fault cases are tabulated in Table 1.

Table 1: Fault location in an 8-bit serial-parallel multiplier using RECO-4

Error signal set	Potential faulty bit-slice pair
$\{e_0, e_4, (e_0, e_4)\}$	0, 4
$\{e_1, e_5, (e_1, e_5)\}$	1, 5
$\{e_2, e_6, (e_2, e_6)\}$	2, 6
$\{e_3, e_7, (e_3, e_7)\}$	3, 7

6 Conclusion

In this paper, an efficient CED technique has been extended to the serial-parallel multiplier for accomplishing fault tolerance. This technique is called RECO because the key approach involves recomputing with circularly-shifted operands. It is more suitable for DSP applications where the serial data are available. Because an extra operator for recomputation is not required, hardware overhead can be reduced. The redundant delay of error detection is only two clock cycles, which required a complete period of full operands recomputation in previous researches. Under the potential faulty bit-slice pair, the simple location table is easy to implement and two spare allocations are introduced for achieving error correction. In these practical experiments, it has been shown that the proposed method requires reduced hardware overhead. But for the parallel-array multiplier, the circuitry required to handle the carry bits from the connection junction can become more complex than the extra bit-slices added to accommodate the arithmetic shifts. Therefore, the circular-shift approach is not suitable for parallel multiplication structures.

7 Acknowledgement

This work was supported by the National Science Council under Grant NSC78-0404-E002-47.

8 References

- 1 LYON, R.F.: 'Two's complement pipelined multipliers', *IEEE Trans. Commun.*, 1976, C-12, pp. 418-425
- 2 GNANASEKARAN, R.: 'A fast serial-parallel binary multiplier', *IEEE Trans. Comput.*, 1985, C-34, pp. 741-744
- 3 CIMINIERA, L., and VALENZANO, A.: 'Low cost serial multiplier for high-speed specialised processors', *IEE Proc. E, Comput. Digit. Tech.*, 1988, 135, pp. 259-265
- 4 ANELLO, A.J., FUREY, R.G., MINERO, R.H., BEARD, J.Q., and PALOUNEK, L.R.: 'Error checking by pseudoduplication', *IBM Tech. Disclosure Bull.*, 1971, 14, pp. 16-17
- 5 CROUZET, Y., and LANDRAULT, C.: 'Design of self-checking MOS LSI circuits, application to a four-bit microprocessor', *IEEE Trans. Comput.*, 1980, C-29, pp. 532-537
- 6 REYNOLDS, D., and METZE, G.: 'Fault detection capabilities of alternating logic', *IEEE Trans. Comput.*, 1978, C-27, pp. 1093-1098
- 7 PATEL, J.H., and FUNG, L.Y.: 'Concurrent error detection in ALUs by recomputing with shifted operands', *IEEE Trans. Comput.*, 1982, C-31, pp. 589-595
- 8 PATEL, J.H., and FUNG, L.Y.: 'Concurrent error detection in multiply and divider arrays', *IEEE Trans. Comput.*, 1983, C-32, pp. 417-422
- 9 LAHA, S., and PATEL, J.H.: 'Error correction in arithmetic operations using time redundancy'. Proceedings of the 13th Annual International Symposium on Fault-Tolerant Computing, 1983, pp. 298-305
- 10 CHENG, W.T., and PATEL, J.H.: 'Concurrent error detection in iterative logic arrays'. IEEE Fault Tolerant Computing Symposium, 1984, pp. 10-15
- 11 LALA, P.K.: 'Fault-tolerant and fault-testable hardware design' (Prentice Hall, Inc., Englewood Cliffs, NJ, 1985)
- 12 PRADHAN, D.K.: 'Fault-tolerant computing theory and techniques, Vol. I, II' (Prentice Hall, Inc., Englewood Cliffs, NJ, 1986)
- 13 CHAN, S.W., LEUNG, S.S., and WEY, C.L.: 'Systematic design strategy for concurrent error diagnosable iterative logic arrays', *IEE Proc. E, Comput. Dig. Tech.*, 1988, 135, pp. 87-94
- 14 JOHNSON, B.W.: 'Design and analysis of fault tolerant digital systems' (Addison-Wesley, 1989)
- 15 CHEN, L.G., and CHEN, T.H.: 'Computation with simultaneously concurrent error detection using bi-directional operands', *Proc. IEEE ICCD*, 1989, pp. 128-131
- 16 ABRAHAM, J.A., and FUCHS, W.K.: 'Fault and error models for VLSI', *Proc. IEEE*, 1986, pp. 639-654
- 17 WU, C.C.: 'Time redundant fault-location in bit-sliced ALU's', *IEEE Trans. Comput.*, 1987, C-36, pp. 1387-1389