設計自動化　72－13
（1994. 10. 28）

Combining spatial placement with time redundancy to achieve CED and error correction

Teresa M. Murray

Guidance and Control Technology Laboratory
Space Subsystems and Technology Department
Office of Research and Development
National Space Development Agency of Japan (NASDA)

Abstract:

In order to accomplish concurrent error detection (CED) there has to be at least two computations of every computation and the duplicate computations cannot contain the same fault in both computations. Two methods are available — differing the spatial placement of the logical processing elements (PEs) or encoding and decoding the duplicate's input and output. The former is employed with time redundancy.

CED and correction is gained by placing duplicate logical PEs on physically different PEs that are in a circular sequence. The cycle between communication phases of the mesh is lengthened to two computation phases. The scheme is described and then its costs are explored showing that it gains an efficient use of time and of hardware for fault tolerant meshes and for 3 PEs.

Key Words:　Concurrent Error Dection, WSI, mesh, self-testing, Single Event Upsets

同時誤り検出（ＣＥＤ）および誤り訂正のための空間配置と時間冗長の組み合わせ

テレザ　マリー

宇宙開発事業団　技術研究本部　要素技術研究部　制御技術研究室
〒３０５　茨城県つくば市千現２丁目１－１

あらまし

同時誤り検出（ＣＥＤ）のためには各計算毎に少なくとも２回の計算を行う必要がある。２重の計算は両方の計算に同じ誤りを含むことはありえない。２つの方法が利用できる。１つは論理的処理要素（ＰＥ）の空間配置を違えることで、もうひとつは２度目の計算の入出力をエンコード、デコードすることである。前者は時間冗長と共に用いられる。

同時誤り検出および訂正は２重の論理的処理要素を環状に連結された物理的に異なる処理要素上に配置することで達成される。メッシュの通信フェーズのサイクルは計算フェーズの２倍に延長される。本方式の概要が述べられ、そのコスト調査により、この方式がフォールトトレラントメッシュと３個の処理要素にたいして、時間とハードウェアの効率的な利用を達成することが示される。

和文キーワード：同時誤り検出、ＷＳＩ、メッシュ、セルフテスティング、シングルイベントアップセット

1 Introduction

Meshes, or 2-dimensional arrays of Processing Elements (PEs), provide a linear computation speedup for many matrix operations. When meshes are implemented in Wafer Scale Integration (WSI) they need to detect the occurrence of faults and overcome the faults for the continuation of fault free computing. This paper presents a new scheme for Concurrent Error Detection (CED) in meshes for use in spacecraft.

Two contributions are made in the work presented here. The first is the inclusion of Single Event Upsets (SEUs) in the fault model for meshes in WSI. SEUs must be incorporated into the fault model if meshes are to be used in spacecraft. The error detection within a mesh must change when SEUs are included and it is better not to assume only permanent faults, which is wide spread in the literature[16]. Inclusion of SEUs is explored further in [12]. The second contribution, and the focus of this paper, is a new CED scheme that detects SEUs, transient faults and permanent faults and that can locate faults which persist.

The paper is organized as follows: Section 2 gives some background information; Section 3 gives a description of the new CED scheme with Section 3.2 providing some probabilities of the success of the scheme; Section 4 evaluates the cost of the scheme; and Section 5 discusses the scheme.

2 Background

The basic theme of the research pursued was to find some improvement for testing results in PEs and especially testing PEs in a WSI mesh for the use of these WSI devices in spacecraft. The context of the work being WSI Fault Tolerant (FT) meshes. The work can also be applied to N modular redundancy (NMR) and specifically, Triple Modular Redundancy (TMR). The issue of greatest variance from the usual investigation of FT meshes in WSI is that the fault model must be widened to include SEUs.

SEUs occur in electronic circuitry of spacecraft due to radiation effects. The nature of SEUs is well known [15] and is discussed in [8].

2.1 Extant Work In CED

Duplication of computations is required for CED and the implementation must ensure that the equality of the computations truly indicates a correct result and that inequality detects an error correctly. To achieve this either (a) the computation must be carried out by separate circuitry or (b) the operands in the computation must be sufficiently different to detect errors with the encoding of the operands and the decoding of the output providing the values to be compared. Either model has the potential to detect SEUs, transient faults and permanent faults.

Implementation of model (a) can be by either duplicating the computation module in hardware (hardware redundancy) or duplicating the computation in time on differing computation modules (time redundancy) with a single amount of hardware.

When hardware redundancy is employed the two computation modules are usually within the same PE [5,6,7,10,13]. On an inequality an error signal is set.

When time redundancy is used the duplicate computations are calculated in different PEs and a comparator compares the values either periodically for every x ($x \geq 1$) computations or at the output values of the mesh [9,11,16,20].

For model (b) the encoding can be as simple as bit shifting which is done in RESO [1,3,14] where the criteria is self duality to ensure the shifting does detect errors. Another encoding scheme is Alternating Logic (AL) [18] which alternates the bits for the encoding and decoding. These two schemes check every computation for a specific number of faults.

FT algorithms are another way of detecting and correcting errors in mesh calculations. Essentially, the idle cycles present in the mesh calculation are utilized for the spatial placement of the duplicate computations[2,11,17,19]. For efficiency in the algorithm an ad hoc design is required but there is also a general method for altering the algorithms [20].

In the next section, a new CED scheme will be described for model (a) that uses time redundancy and for which the duplicate computations spread over a sequence of physical PEs.

3 Details of the scheme.

A new CED scheme for meshes will be now described. The scheme places the duplicate calculations of physically different PEs in such a manner that a small set of the duplicates reside over a small number of PEs. This scheme is a time redundancy scheme and for which the physical placement of the duplicates is important.

The time redundancy lengthens the full computation cycle of the mesh to provide a second computation phase between communication phases. Hence, a full computation cycle includes: communication phase, computation phase 1, computation phase 2, comparison of results checking for mismatches, and finally, if mismatches exist then decide on retry and communicate that a retry is needed. A direct result of the retry is that the memory of the input cannot be overwritten during either computation phase.

PEs in meshes usually have this separation of input and output memory [5,6,7,13,14].

The placement scheme needs some notions, which will be described now. Figure 1 shows the scheme as graphs. Underneath the CED scheme there is a logical n x m mesh mapping onto a $(n + s_r)$ x $(m + s_c)$ physical mesh with $s_r, s_c > 0$. The logical PEs of the mesh are (i,j) with $1 \leq i \leq n$ and $1 \leq j \leq m$. A reconfiguration scheme maps the logical mesh onto the physical mesh.

There is a sequence of logical PEs, $S = < S(r_0, c_0), S(r_1, c_1), \ldots, S(r_{/S/-1}, c_{/S/-1}) >$, such that each logical PE in the sequence is mapped to a PE in the logical mesh. $/S/$ is the number of logical PEs in the sequence and there are $^{nm}/_{/S/}$ distinct sequences partitioning the mesh. The duplicates that give the logical PE $S(r,c)$ are $S^1(r,c)$ and $S^2(r,c)$. These duplicates of the sequence elements are mapped such that

$S^1(r_k,c_k) \longrightarrow S(r_k,c_k)$ and

$S^2(r_k,c_k) \longrightarrow S(r_{(k+1) \bmod /S/},c_{(k+1) \bmod /S/})$.

In Figure 1 the nodes of the graphs represent $S(r,c)$ which is then mapped to a physical PE. The upper hemisphere represents the first computation phase (computing $S^1(r,c)$) and the lower hemisphere represents the second computation phase (computing $S^2(r',c')$). The arcs represent the duplication and so link $S^1(r,c)$ with $S^2(r,c)$. The hemispheres are shaded when that physical PE during that computation phase is faulty. Figure 1 gives the graphical representation when $/S/=3$ and 4.

3.1 Description of the scheme.

With these basic notions the scheme can now be stated. The logical PEs are mapped to two places (one in computation phase 1 and the other in computation phase 2) according to a circular sequence, S, of length $/S/$. The detection of a mismatch by a voter causes a broadcast across the mesh that a retry must be undertaken. All PEs execute the retry. If faults persisting for two computation phases are detected in the retry then the need for a reconfiguration is broadcast. Otherwise, the calculation in the mesh can continue undisturbed after the retry. The cycle that first detects the fault is called the trigger cycle.

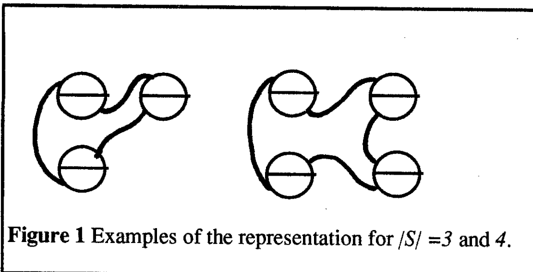

Figure 1 Examples of the representation for $/S/ =3$ and 4.

3.1.1 The Vote and Retry Procedures

The required procedures for the scheme are now given. Each is a distributed procedure assuming a single fault in the two full computation cycles. [12] gives centralized procedures and procedures that assume more than one fault.

```
procedure error_detect
begin
    sum the number of mismatches
    save the sum
    if either logical pairs do not match then
    begin
        broadcast retry
        call retry procedure
    end
end

procedure retryD
begin
    if mismatch occurred last cycle then
        if 2 mismatches this cycle then
        begin
            set error status
            broadcast need for reconfiguration which
                will cause an interrupt and instigate
                procedure reconfiguration_preparation
        end
        else if 1 mismatch this cycle then
            set error status
        else clear error status
    else clear error status
end

procedure reconfiguration_preparation
begin
    if number of mismatches in the most recent cycle
        is 2 then
        set error status
    else clear error status
end
```

3.1.2 Fault Masking

How does the scheme mask faults? The answer is trivial if the fault does not persist into the retry. If the fault does persist into the retry unconfirmed values can be utilized. This is because the fault that triggered the retry can be located when assuming one fault per sequence is valid. Observing all the possible fault occurrences in the sequence one can see that the sums of mismatches in a fault free PE are $<0,0>$, $<1,0>$ or $<0,1>$. A PE with any of these sums of mismatches is fault free and its output values may be used. By using the values in the fault free PE, we are using unconfirmed values to mask faults.

3.1.3 Assumptions

There is a reconfiguration scheme to provide a FT mesh. No assumptions need be made for

which reconfiguration scheme is chosen for this discussion. In this paper only a sequence of PEs will be considered as within the mesh the sequence providing CED is of interest.

Permanent faults, transient faults and SEUs are assumed to occur. Permanent faults occur in a PE and persist for ever more. Transient faults occur and may persist for 1 to 4 computation phases.[1] SEUs occur and persist only until a new signal is received by the corrupted circuitry. SEUs persist in memory until a new write operation writes to that memory cell, and in other circuits until a new signal is passed through the circuit. This means that they persist for only one computation phase unless they occur in the input memory locations of a PE.

Faults are functional faults. That is, they cause an error in some output of the PE. If they do not affect the output they are not detected and are not considered to be a fault. All faults are considered to be random and independent.

Fault free communication between PEs is assumed.

3.2 Calculation of probabilities

Several probabilities are of interest in order to evaluate this scheme. The method used to determine the relevant probabilities will be given in this section. $|S|=3$ for the purposes of explanation.

The binomial distribution is used to calculate the probability of failure in the sequence of duplicates. The Poisson approximation and the normal distribution approximation to the binomial distribution are not used since the product $|S|p$ is not large. Since $|S| \longrightarrow n \times m$ in a mesh, $|S|p$ becomes large and so approximations could be used for the limiting case. However, here $|S|p$ is very small since the number of PEs of interest is the number of physical PEs over which the cycle of duplicates reside. The probability of a failure is also very small with a permanent fault taken to be 10^{-15}. In the calculations the following definitions will apply:

p_{seu} = the probability of an SEU in a PE in one computation phase,

p_p = the probability of a permanent fault in one PE in one computation phase,

p_t = the probability of a transient fault in one PE in one computation phase, and

q = $1 - p_{seu} - p_p - p_t$.

Three probabilities are considered for evaluation purposes and they are:

the probability that a single fault triggers a retry and no new faults occur in the retry,

the probability that a single fault triggers a retry and a second fault occurs in the retry,

the probability that a single fault triggers a retry and that an unconfirmed value used in the retry procedure is faulty.

Notice that these probabilities are for two full computation cycles which is equivalent to four computation phases. Also note that an unconfirmed value exists in the retry since each logical PE is duplicated. One being faulty and the other being an unconfirmed value. These unconfirmed values can be used when the location of the fault is known.

Since the binomial distribution is being used and the faults are random and independent, one needs to count the number of ways events can occur for all three fault types.

The probability of a single SEU occurring in the triplet in the first full computation cycle causing a retry and with no other faults occurring is $6q^{11}p_{seu}$ from the combination ${}^6C_1$. It can be considered to be placing one black ball and five white balls in six indistinguishable bins.

When counting the ways that a transient fault can occur one needs to consider the different life expectancies of the transient fault. Over two full computation cycles the transient fault could occur for between 1 to 4 computation phases. All these need to be included in the count. There are seven cases to be considered such that the probability of a transient fault occurring in the first full computation cycle to cause a retry and no second fault occurring is:

$$3q^{11}p_t + 3q^{11}p_t + 3q^{10}p_t + 3q^9p_t + 3q^{10}p_t + 3q^9p_t + 3q^8p_t$$
$$= 6q^{11}p_t + 6q^{10}p_t + 6q^9p_t + 3q^8p_t$$

The probability of a permanent fault occurring is the probability that a permanent fault begins in the first full computation cycle in either the first or second computation phase. The probability of a permanent fault occurring in the first full computation cycle causing a retry and no new second fault occurring is then $3q^9p_p + 3q^8p_p$.

The probability of a single failure is then the summation of the three expressions. These expressions can be simplified and the predominate terms found by expressing p_{seu} and p_t in terms of p_p. This allows the summation to reduce to a final approximation.

Pr{1 fault and no other faults in 1st full computation cycle and retry}

$$= 6q^{11}p_{seu} + 6q^{11}p_t + 6q^{10}p_t + 6q^9p_t + 3q^8p_t + 3q^9p_p + 3q^8p_p$$
$$= -3p_p(c_1p_p{}^{11} - c_2p_p{}^{10} + c_3p_p{}^9 - c_4p_p{}^8 + c_5p_p{}^7 - c_6p_p{}^6 + c_7p_p{}^5 - c_8p_p{}^4 + c_9p_p{}^3 - c_{10}p_p{}^2 + c_{11}p_p - c_{12})$$
$$\approx 3p_pc_{12} \qquad (3.1)$$

[1] When a transient fault persists for 2 computation phases in the retry the fault detection and location scheme will treat it as a permanent fault. However, for the probability calculations there can be transient faults existing for 1, 2, 3 and 4 computation phases.

Using similar counting methods, the probabilities for a second fault occurring in the retry can be gained.

Pr{1 fault in 1st cycle and 1 other fault in the retry} $\approx 18p_p{}^2c_{11}$ (3.2)

Pr{1 fault in 1st cycle and unconfirmed value in retry is faulty} $\approx 3p_p{}^2c_{11}$ (3.3)

Using similar methods the probabilities can be gained for the varying $/S/$ for the three probabilities.

4 Cost Evaluation

The costs that need to be included in the evaluation are time, hardware and signal delay. Letting the time taken for the calculation on the bare FT mesh without CED be T, then the time taken on a FT mesh with this CED scheme is $T(2+\varepsilon)+E$, where E is the time for the expected number of retries in a whole mesh calculation.

The expected number of retries is very low, being $2n^2m^2(p_p+p_{seu}+p_t) < 1$ where the number of calculations in a matrix operation is taken to be $O(nm)$ on the $n \times m$ mesh. The time cost is then $T(2+\varepsilon)$, approximately a slow down of two.

4.1 Hardware Costs For The Mesh

The hardware costs for the scheme in a FT mesh have a component dependent on $/S/$ and a component independent of $/S/$. The costs that are independent of $/S/$ include within each PE the memory for two sets of input data, the memory for two sets of output data and the output data of one neighbor, the voter, and the retry procedure. Also, a 1-bit network for the broadcasting the need of a retry or reconfiguration.

These requirements are above those of a FT mesh without the spatial placement scheme developed in this paper. Let M_I and M_O be the memory required in an unaltered mesh for input and output values, C_{retry} be the area of the circuitry for the retry procedure, and G_{11} be the area used by a 1-bit grid over the whole mesh. The added area consumed by the spatial placement scheme is greater than that of an unaltered mesh by $2M_I+3M_O+C_{retry}+G_{11}$. However, this is not necessarily the total additional cost of the spatial placement scheme as we need to include the area that is dependent on the value of $/S/$.

As $/S/$ varies the time required for the communication of the output values changes so that the comparison can take place. For example, when $/S/=4$, the signal delay experienced on passing information between the PEs is bounded by the delay caused by k switches for all pairs of PEs in the mesh and in the sequence. The delay of k switches being caused by the reconfiguration scheme. However, when $/S/=3$, the signal delay is not bounded by the delay caused by k switches for all pairs of PEs in the mesh but is bounded by $2k$ between two of the PEs in the sequence.

When $/S/=3$ the connections within the sequence cannot be made on the communication network that is provided for the mesh's North-South and West-East communication. This is due to the triangular nature of the cycle of duplicates for comparison. To use the mesh's communication network one value would have to pass through a PE that is not in the cycle causing the data to travel twice the distance of the other data values. To reduce the delay a second network would most probably be required.

When the sequence of logical PEs, S, is such that it can map onto the mesh such that all pairs of logical PEs in the sequence are logical NSEW neighbors then the communication for comparison of duplicates can be managed via the mesh's communication network. Hence, only one communication network is required for the FT mesh and for the CED. All signal delay for communication is bounded by the k switches of the reconfiguration strategy. The additional hardware cost is only $2M_I+3M_o+C_{retry}+G_{11}$.

4.2 Hardware Costs For A Single Triplet

Taking $/S/=3$ and $nm=3$, we have a triplet of PEs using CED and using retry to mask faults. In this case the voter and comparison would be centralized rather than distributed. The costs of such a proposal above the cost of TMR include within each PE the memory for two sets of input data, and the memory for two sets of output data. Also, the voter would need to enlarge in order to compare three pairs of output data sets rather than three sets of output data, and to execute the retry procedure.

These requirements are above those of a TMR triplet without the spatial placement scheme developed in this paper. Using the same definitions, the total added area consumed by the spatial placement scheme is greater than that of a TMR triplet by $3M_I+3M_O+C_{retry}$. The time of the scheme $^2/_3T+E$ with $E<1$ and T being the time a TMR system would use.

5 Discussion

The scheme presented here is effective and efficient in detecting errors in PEs. The PEs may be found in either a FT mesh implemented in WSI or a sequence of PEs implemented in VLSI. In the context of WSI the optimum size of the sequence is four and in VLSI the optimum size of the sequence is likely to be three.

Let us consider the case of when $/S/=4$ and the scheme is used for a FT mesh in WSI. This size is optimum because when the circular sequence can be mapped onto NSEW neighbors the hardware cost in silicon area and signal delay is

constant and, the smaller the sequence, the less likely that more than one error will occur. When we can assume only one fault we gain fault masking on the retry.

Using the mapping of the logical duplicates as specified in Section 3, for a low cost in hardware there is CED, and when p_p is low, masking of the fault in the retry is available. The masking is possible when the assumption of only a single fault being present in the sequence is valid. Note that this is a single fault in a sequence and not a single fault in the mesh.

The scheme of this paper copes with more faults per mesh than other schemes presented in literature. Sami and Stefanelli [16] can only locate permanent faults and mask no faults. Jean and Kung *et al* [5–7] treat SEUs, transient faults and permanent faults in identical fashion causing fatal failure earlier than is required. CORP by Manolakos and Kung [10] can detect and mask SEUs and transient and permanent faults. However, the occurrence of a fault requires that the region of the mesh before (or behind) the affected wavefront be fault free. Here only one sequence of PEs is affected.

A disadvantage of the scheme is the slow down of 2 for the whole calculations on the mesh. This slow down could very well be absorbed in the speedup obtained from using a mesh. This slow down is common with RESO and AL CED schemes also causing such a slow down.

Let us consider the case of when $/S/=3$ and $nm=3$ such that the scheme is used as a modified form of TMR. The major advantage is that when the tasks can be divided in such a way that three different tasks are being computed concurrently, a time reduction to $^2/_3T$ is gained. Since the expected number of retries is low the speedup is gained most of the time. The disadvantage is in the increased memory required such that the input and output are separate, there are two sets of input and output memory and that the input memory is not over written by the computation during the PE computation. These memory requirements will probably require custom chips to be used increasing the device development cost.

The other major disadvantage is that confirmed values will not be gained if the fault persists where as masking is provided in TMR. However, the probability of a second fault occurring is small allowing the location of the fault to be known and so the retry may still be successful since an unconfirmed value can be used.

Bibliography

[1] S.-W. Chan and C.-L. Wey. The Design Of Concurrent Error Diagnosable Systolic Arrays For Band Matrix Multiplications, *IEEE Trans. Computer-Aided Design,* Vol. 7, No. 1, 1988, pp. 21 - 37.

[2] Chien-Yi Chen and J.A. Abraham. Fault-Tolerant Systems For The Computation Of Eigenvalues And Singular Values, *SPIE,* Vol. 696, 1986 ,pp. 228 -- 37.

[3] W.-T. Cheng and J. H. Patel. Concurrent Error Detection In Iterative Logic Arrays, *14th Int'l Conf. Fault-tolerant Computing,* 1984, pp. 10 - 15.

[4] T. Goka, S. Kuboyama, Y. Shimano and T. Kawanishi. The On-Orbit Measurements Of Single Event Phenomena By ETS-VI Spacecraft, *IEEE Trans. Nuclei. Science.,* Vol. NS-38, 1991, pp. 1693 - 1699.

[5] J. S. N. Jean. Fault-Tolerant Array Processors Using N-And-Half-Track Switches, *Int'l Conf. Application Specific Array Processors,* 1990, pp. 426 - 437.

[6] S. Y. Kung, C-W. Chang C. W. Jen. Fault-Tolerance Design In Real-Time VLSI Array Processors, *Proc. 6th Annual Phoenix Conference,* 1987, pp. 110-115.

[7] S. Y. Kung, J. S. N. Jean and C-W. Chang. Fault Tolerant Array Processors Using Single-Track Switches, *IEEE Trans. Computers,* Vol. C-38, No. 4, 1989, pp. 501 - 514.

[8] M. Lauriente (Ed.) and J. Gaudet (Ed.). Environmentally Induced Spacecraft Anomalies, *Journal of Spacecraft and Rockets,* Vol. 31, No. 2, 1994, pp. 153 - 185.

[9] A. Majumdar, C. S. Raghavendra and M. A. Breuer. Fault Tolerance In Linear Systolic Arrays Using Time Redundancy, *IEEE Trans. Computers,* Vol. 39, No. 2, 1990, pp. 269 – 276.

[10] E. S. Manolakos and S. Y. Kung. CORP - A New Recovery Procedure For VLSI Processor Arrays, *Proc. Symp. Engineering of Computer-Based Medical Systems,* 1988, pp. 91 - 95.

[11] R. G. Melhem. Bi-Level Reconfiguration of Fault Tolerant Arrays, *IEEE Trans. Computers,* Vol. 41, No. 2, 1992, pp. 231 – 239.

[12] T. M. Murray. Incorporating Comparison For Testing To Achieve A Real-Time Fault Tolerant Mesh, *NASDA Report for STA Postdoctoral Fellowship,* in preparation.

[13] R. Negrini, M. Sami, and R. Stefanelli. Fault-Tolerance Approaches For VLSI/WSI Arrays, *Phoenix Conf. Computers and Communication,* 1985, pp. 460 - 468.

[14] J. H. Patel and L. Y. Fung. Concurrent Error Detection In Multiply And Divide Arrays, *IEEE Trans. Computers,* Vol. C-32, No. 4, 1983, pp. 417 – 422.

[15] P. Robinson, W. Lee, R. Aguero and S. Gabriel. Anomalies Due To Single Event Upset, *Journal of Spacecraft and Rockets,* Vol. 31, No. 2, 1994, pp. 166 - 171.

[16] M. G. Sami, and R. Stefanelli,. Self-testing Array Structures, *Proc. IEEE Int'l Conf. Computer Design: VLSI in Computers ICCD '84,* October 1984, pp. 677-82.

[17] D. L. Tao, Y. S. Han and C. R. P. Hartmann. New Encoding/Decoding Methods For Designing Fault-Tolerant Matrix Operations, *SPIE,* Vol. 1770, 1992, pp. 72 - 83.

[18] C.-L. Wey. Concurrent Error Detection In Array Dividers By Alternating Input Data, *IEE Proc.-E,* Vol. 139, No. 2, 1992, pp. 123 - 130.

[19] S. Yurttas and F. Lombardi. New Approaches For The Reconfiguration Of Two-Dimensional VLSI Arrays Using Time-Redundancy, *Proc. Real-time Systems Symp.,* 1988, pp. 212 - 221.

[20] C. N. Zhang, H. F. Li and R. Jayakumar. A Systematic Approach For Designing Concurrent Error-Detecting Systolic Arrays Using Redundancy, *Parallel Computing,* Vol. 19, No.7, 1993, pp. 745 - 64.