TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER

Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Real-Time Algorithm-Based Fault-Tolerance for QRD Recursive Least-Squares Systolic Array: A Graceful Degradation Approach

by K.J.R. Liu and K. Yao

TR 91-10

Real-Time Algorithm-Based Fault-Tolerance for QRD Recursive Least-Squares Systolic Array: A Graceful Degradation Approach

K.J. Ray Liu and Kung Yao
Electrical Engineering Department
University of Maryland
College Park, MD 20742

ABSTRACT

In this paper, we propose a new algorithm-based fault-tolerant method derived from the inherent nature of the QR least-squares systolic algorithm. Since the residuals of different desired responses can be computed simultaneously, an artificial desired response can be designed to detect an error produced by a faulty processor. We show that if the artificial desired response is designed as some proper combinations of the input data, the output residual of the system will be zero if there is no fault. However, any occurring fault in the system will cause the residual to be non-zero and the fault can be detected in real-time. Once the fault has been detected, the system enters into the fault diagnosis phase from the concurrent error detection phase. Two methods, the flushing fault location and the checksum encoding methods, can be used to diagnose the location of the faulty row. When the faulty row is determined, this row and the associated column with the same boundary cell are eliminated by a reconfiguration operation. Then the system degrades in a graceful manner which is generally acceptable for many least-squares applications. Those eliminated processors enter into a self-checking phase, and when the transient fault condition is removed, a reconfiguration is performed to resume the normal full order operation. The analysis of error propagation and recovery latency is also considered in this paper.

Index Terms - Algorithm-based fault-tolerance, concurrent error detection, fault diagnosis, least-squares, QR decomposition, real-time processing, systolic array.

This work is partially supported by UC MICRO grant and the NSF grant NCR-8814407.

1 Introduction

Rapid advances in VLSI microelectronics make it practical to build low-cost and high-density application-specific integrated circuits (ASIC) to meet the demands of speed and performance of modern signal processing. Recent VLSI/WSI technology permits the building of million of transistors in a single chip, while a large system may require hundreds of these chips to function properly. For a complex system, a single fault from any part of the system can make the whole system useless. For various critical applications, highly-reliable computations are demanded. Fault-tolerance is therefore needed in many of these problems.

Fault-tolerance has been defined as *the ability of a system to execute specified algorithms correctly regardless of hardware failures and program errors* [18]. In order to achieve the goal of fault-tolerance, redundancy has to be introduced. When we encounter a specific VLSI signal proessing problem, an inherent nature of that signal processing algorithm can be used to develop a highly efficient specific fault-tolerant technique named *algorithm-based fault-tolerance.* The term *algorithm-based* means it is an algorithm-oriented but not a general scheme that can be applied to all general problems. A recently reported algorithm-based fault-tolerant technique, called checksum encoding (and weighted checksum) scheme proposed by Hwang, Abraham, Jou, Chen et al., has evolved from the study of VLSI matrix computation systems [3,4,5,6]. This scheme belongs to the category of information redundancy [21]. Since few hardware and time redundancies are necessary, it is promising for its low-cost and low overhead for VLSI/WSI multiprocessor systems. Many applications of the checksum (weighted checksum) scheme have been successfully applied to various signal processing and linear algebra operations [13]. The major drawback of the checksum scheme proposed in [6] is that the system throughput will be slowed down because the system clock has to be extended long enough to accomodate the longer signal path of non-local interconnection caused by the checksum scheme. Unfortunately, local connection is one of the basic desirable requirements of implementations.

Least-squares (LS) algorithms are among the most important tools for communication and signal processing applications. Gentleman and Kung [12] and McWhirter [1] showed that the QR Decomposition (QRD) LS algorithm can be successfully implemented using systolic arrays. For adaptive filtering or array processing applications in radar and sonar systems, signals are continueously inputted and intermediate results are also continuously propagated in the systolic array [15]. Since error detection is the only capability that has to be active during the normal operation (other capabilities such as fault diagnosis and reconfiguration become activated only when a fault is detected) and the frequency of faults in a well-designed system is relatively low, the overhead of implementing a fault-tolerant scheme is dominated primarily by error detection operations [25]. Due to the major drawback of the checksum addressed above, we are motivated to look for a more efficient error detection scheme for real-time applications of these problems.

In this paper, we propose a new algorithm-based fault-tolerant scheme derived from the inherent nature of the QRD LS systolic algorithm called *residual method.* For a LS problem, especially in communication and signal processing applications, we abstract information from the residuals which are the difference of the optimal LS estimations from input data and the desired responses. Based on the fact that a QRD LS systolic array can compute the residuals of different desired responses simultaneously, an artificial desired response can be designed to detect any error produced by a faulty processor. We show that if the artificial

desired response is designed as a non-zero linear combination of all data inputs, the residual output of this response will be zero if no fault occurred. Any fault in the system will cause the residual to be non-zero and thus the fault is detected in real-time. Thus, the residual method can be easily incorporated with the systolic array antenna beamforming systems such as that considered in [14] and will have a great impact on the next generation of radar and sonar systems where the fault-tolerance scheme is quite essential for reliable real-time operation.

Once the fault has been detected, two methods to diagnose the location of the faulty row are addressed. The first method, called the flushing fault location method, is based on the weight flushing technique to flush the weight vector out by using an identity matrix input. The second method is to use the checksum encoding property to detect the row which does not meet the checksum. When the faulty row is determined, this row and the column associated with the same boundary cell are eliminated by a reconfiguration operation. Then the system operates in an order-degraded manner which is acceptable in many least-squares applications.

This paper is organized in the following manner. In section 2, we review the QRD LS algorithm and its systolic array implementation. In section 3, we address some previous works and the fault model. The fault-tolerance schemes, including fault diagnosis, are presented in section 4. In section 5, we prove the residual method is robust and discuss some important latencies. Finally, in section 6, we present some conservation tests which may be used to detect fault as well.

2 QRD Recursive Least-Squares Algorithm

Consider a $n \times p$ real-valued data matrix $A(n)$ denoted by

$$A(n) = [\underline{u}(1), \underline{u}(2), \cdots, \underline{u}(n)]^T = [\underline{a}(1), \underline{a}(2), \cdots, \underline{a}(p)], \tag{1}$$

a $n \times 1$ desired response vector $\underline{y}(n) = [d(1), d(2), \cdots, d(n)]^T$, a $p \times 1$ weight vector $\underline{w}(n)$ and a $n \times 1$ residual vector

$$\underline{\epsilon}(n) = [e(1), e(2), \cdots, e(n)]^T = A(n)\underline{w}(n) - \underline{y}(n).$$

Let the index of performance be defined by the weighted l_2 norm of

$$\xi(n) = \|\underline{\epsilon}(n)\|_\Lambda^2 = \|\Lambda\underline{\epsilon}(n)\|^2 = \underline{\epsilon}^H(n)\Lambda^2(n)\underline{\epsilon}(n),$$

where $\Lambda(n) = diag[\lambda^{(n-1)}, \lambda^{(n-2)}, \cdots, \lambda, 1]$ with a real-valued forgetting factor $0 < \lambda \leq 1$. Then the LS solution, satisfies

$$\xi_{min}(n) = \min_{\underline{w}} \|A(n)\underline{w}(n) - \underline{y}(n)\|_\Lambda^2 = \|A(n)\underline{\hat{w}}(n) - \underline{y}(n)\|_\Lambda^2. \tag{2}$$

Assume A to be full rank with $n \geq p$. Then the QR Decomposition of $\Lambda(n)A(n)$ yields

$$Q(n)\Lambda(n)A(n) = [R(n)^T, \mathbf{0}]^T,$$

where $R(n)$ is a $p \times p$ upper triangular matrix and $Q(n)$ is an $n \times n$ unitary matrix. Thus

$$\xi(n) = \|Q(n)\underline{\epsilon}(n)\|_\Lambda^2 = \|[R(n)\underline{w}(n) - \underline{P}(n)\|^2 + \|\underline{v}(n)\|^2,$$

where

$$[\underline{P}^T(n), \underline{v}^T(n)]^T = Q(n)\Lambda(n)\underline{y}(n),$$

with a $p \times 1$ vector $\underline{P}(n)$ and a $n \times 1$ vector $\underline{v}(n)$. The LS solution $\underline{\hat{w}}(n)$ can be obtained from

$$R(n)\underline{\hat{w}}(n) = \underline{P}(n). \tag{3}$$

Then $\xi_{min}(n) = \|\underline{v}(n)\|^2$.

For some radar/sonar and communication problems, the weight vector $\underline{w}(n)$ may not be of direct interest. For example, the residual is of interest in the multiple sidelobe canceller adaptive array problem. Let $\underline{u}(n)$ be the nth new incoming row vector of data in $A(n)$ of (1). Then the LS residual

$$e(n) = \underline{u}^T(n)\underline{\hat{w}}(n) - d(n), \quad n = p, p+1, \cdots \tag{4}$$

can be obtained without computing $\underline{\hat{w}}(n)$ explicitly. Thus, the complexity of the LS problem is further reduced and the solution of $e(n)$ is feasible with a systolic array implementation [1].

To compute the linear LS problem recursively, a unitary matrix $\bar{Q}(n-1)$ is defined as

$$\bar{Q}(n-1) = \left[\begin{array}{c|c} Q(n-1) & \underline{0}_{n-1} \\ \hline \underline{0}_{n-1}^T & 1 \end{array}\right]$$

and we have

$$\bar{Q}(n-1)\Lambda(n)A(n) = \left[\begin{array}{c} \lambda R(n-1) \\ \hline \mathbf{0} \\ \hline \underline{u}^T(n) \end{array}\right].$$

Suppose the Givens rotation method is used for the QRD, then $\underline{u}^T(n)$ can be annihilated by applying a sequence of Givens rotations

$$G(n) = G_p(n) \cdots G_2(n) G_1(n), \tag{5}$$

where the $n \times n$ transformation matrix G_i is defined by

$$G_i = \left[\begin{array}{c:c:c:c:c} I_{i-1} & 0 & 0 & 0 \\ \hline 0 & c_i & 0 & s_i \\ \hline 0 & 0 & I_{n-i-1} & 0 \\ \hline 0 & -s_i & 0 & c_i \end{array}\right],$$

with

$$c_i = \frac{a}{\sqrt{a^2+b^2}}, \quad s_i = \frac{b}{\sqrt{a^2+b^2}},$$

where a and b are elements of vectors in the ith and n^{th} rows under rotation. The matrix $G(n)$ can be shown to have the form

$$G(n) = \begin{bmatrix} K(n) & | & \mathbf{0} & | & \underline{h}(n) \\ \hline \mathbf{0} & | & I_{n-p-1} & | & \mathbf{0} \\ \hline \underline{h}^H(n) & | & \mathbf{0} & | & \gamma(n) \end{bmatrix},$$

where $K(n)$ is a $p \times p$ matrix, $\underline{h}(n)$ is a $p \times 1$ vector, I_{n-p-1} is the $(n-p-1)\times(n-p-1)$ identity matrix, and $\gamma(n)$ is a scalar given by $\gamma(n) = \prod_{i=1}^{p} c_i(n), \quad n \geq p$ with $c_i(n)$ is the cosine parameter associated with the ith Given rotation. The error vector can be transformed by $\bar{Q}(n-1)$ to

$$\bar{Q}(n-1)\Lambda\underline{\epsilon}(n) = \begin{bmatrix} \lambda R(n-1) \\ \hline \mathbf{0} \\ \hline \underline{u}^H(n) \end{bmatrix} \underline{w}(n) - \begin{bmatrix} \lambda \underline{P}(n-1) \\ \hline \lambda \underline{v}(n-1) \\ \hline d(n) \end{bmatrix}.$$

When the Given rotation is applied on it, we get

$$G(n)\bar{Q}(n-1)\Lambda\underline{\epsilon}(n) = \begin{bmatrix} R(n) \\ \hline \mathbf{0} \end{bmatrix} \underline{w}(n) - \begin{bmatrix} \underline{P}(n) \\ \hline \underline{v}(n) \end{bmatrix}.$$

It can be easily seen that the last element of $\underline{v}(n)$, $v_n(n)$, can be obtained naturally during the triangularization process [1] and is given by

$$v_n(n) = \lambda \underline{h}^H(n)\underline{P}(n-1) + d(n)\gamma(n).$$

In [1], McWhirter has shown the LS residual $e(n)$ can be expressed as

$$e(n) = v_n(n)\gamma(n).$$

The above results lead to the systolic implementation of QRD LS algorithm without computing weight vector explicitly [1]. The systolic array is shown in Fig.1. It consists of two parts: a triangular array for computing QRD and a linear column array called response array (RA) for computing LS residual. When a new data vector is updated, a new triangular matrix R sits in the triangular QR array and a new vector $\underline{P}$ sits in the RA.

3 Previous Works and Fault Model

Recently, the checksum and weighted checksum fault-tolerance schemes have been proposed by Abraham et al [3,4,5,6] to various signal processing and linear algebra operations such as matrix multiplication and inversion, QR decomposition, LU decomposition and singular value decomposition (SVD) et al.. If there is any fault occurring during the computation, the checksum criterion is not met and thus the fault is detected. The weighted checksum scheme can be further used to correct errors [6,17]. The major advantages of (weighted) checksum scheme are low-cost and low overhead for error detection and correction of multiple processor systems. However, the disadvanges are:

1. There is the problem to distinguish roundoff errors from those caused by failures [6];
2. If the intermediate results of each processor are continuously propagatetd to other processors such as QRD LS systolic array, the system throughput will be slowed down because the system clock has to be extended long enough to accomodate the longer signal path of non-local interconnection caused by the checksum scheme.

There are some difficulties in applying checksum schemes to system problems such as communication data equalization, and radar/sonar adaptive antenna array, where the signal arrives continuously and high throughput rate is required. As an example, for a recursive QRD LS systolic array (see Fig.1), the data is coming in row by row. As pointed out in [3], the QRD of a row checksum encoding matrix A_r results in a row checksum encoding upper triangular matrix R_r. That is, $A_r = QR_r$. If the checksum scheme is used to detect error, during the recursive operation, when a new upper triangular matrix R is formed, to obtain a new checksum for each row, we need to sum up each elements of the corresponding row. Suppose only local connection, which is one of the basic desirable requirements of VLSI implementations, is allowed, to prevent severe throughput degradation, a new channel used to pipe out the partial sum of matrix R has to be built. While a new content of each cell is available, this content has to be added to the partial checksum sent from the previous cells through the new checksum channel and then the partial checksum has to be passed to the right for the next cell. With these requirements and operations, not only the complexity is increased but also the throughput is hindered. Even when global communications of the processors are allowed, the checksum scheme for fault detection still reduces the throughput of the system as pointed out above.

Most recently, [33] has proposed a new algorithm-based fault-tolerant technique applicable to the recursive LS triangular systolic array. With the addition of one extra column of processor array, errors in the tri-array can be detected by observing the scalar output of this column array. While there are some similarities between the results reported in [33] and that considered in this paper (as well as that in an earlier version of our paper [34]), there are much differences in assumptions, techniques, and results between these two approaches. In any case, these two works were performed independently of each other.

As the VLSI technology progresses, the geometric features become smaller. Any defect affecting a given part of the circuity may cause an entire module or a logic block to become faulty and to produce arbitrary errors. Thus, the traditional gate-level single stuck-at fault model is no longer appropriate for VLSI/WSI system. A cell or module is allowed to produce arbitrary errors if any part of the cell is under failures [3]. However, we assume that at most one cell can be faulty at a given short period of time. This is based on the assumption that the system reliability is such that the mean time between failures is long enough and the probability of more than one fault occurring is very small. Some basic assumptions we need are as follows:

1. If any part of the cell become faulty, the whole cell will not function correctly;
2. The probability of the communication links and registers failing is very small and thus negligible [24].

The second assumption is reasonable since these components are typically much simpler and smaller than the processing cells themselves [24]. In addition, they can be implemented conservatively with high redundancy or with self-testing circuitry to mask a possible fault.

4 Graceful Degradation Fault-Tolerance Approach

While a demanding system is usually expected to operate under a high performance condition, when a fault has occurred, a slightly degraded performance is often acceptable under the circumstance. For a recursive QRD LS systolic array operating under a normal fault-free condition, the optimum LS solution is attained. A reduced order degraded LS solution can still yield a reasonable performance if the row and column with the faulty processor can be eliminated from the computation. This concept leads to our proposed design of a gracefully degraded fault-tolerance approach for the QRD LS systolic array.

4.1 Concurrent Error Detection - Residual Method

An inherent nature of the QRD LS systolic algorithm is that for a given data matrix A, the minimization of $\|A\underline{w}_i(n) - \underline{y}_i(n)\|_\Lambda$ for many desired response vectors $\underline{y}_i, i \in I$, can be performed concurrently by appending some more RAs to the systolic array such as shown in Fig.1. The output of the ith RA, $e_i(n)$, is the minimal residual of $\underline{u}^T(n)\hat{\underline{w}}_i(n) - d_i(n)$, where $d_i(n)$ is the nth desired response of ith input vector $\underline{y}_i$. Let $\underline{y}_0$ belong to the column space of A. That is, $\underline{y}_0 \in span\{\underline{a}(i), 1 \leq i \leq p\}$, then $\underline{y}_0 = \sum_{j=1}^{p} c_j \underline{a}(j)$. The optimal LS residual and the associated weight vector for $\underline{y}_0$ are $e_0(n) = 0$ and $\underline{w}_i(n) = [c_1, c_2, \cdots, c_p]^T$, for $n \geq p$. The actual selection of $\{c_1, c_2, \cdots, c_p\}$ will be given later. Various extentions of this fundamental property of the optimum residual of a LS estimation problem form the basis of the proposed *residual method* approach toward concurrent error detection. Denote $\underline{y}_0$ to be the **artificial desired response** (ADR) and the associated RA as the **error detection array** (EDA).

Lemma 4.1: Given the ADR $\underline{y}_0 = \underline{a}(p)$, the contents of the EDA are identical to the contents of the pth column array of the QRD triangular array, and the optimal residual e_0, the output of the pth cell of the EDA, is always zero.

Proof: Both arrays are $p \times 1$ vector. The first $p-1$ elements of both arrays are identical given by the fact that the same data are rotated by same c_i and s_i generated by boundary cells $PE_{ii}, 1 \leq i \leq p-1$, where PE_{ii} denoted the processor at position (i,i). Thus the outputs of the $(p-1)^{th}$ cells of both arrays are identical. Initially, the contents of the pth cells of both array are zeros. Let the first non-zero output of $(p-1)^{th}$ cells be x, by the update equations of both cells, the first non-zero contents of both p^{th} cells equal x. If the second non-zero output of $(p-1)^{th}$ cells is z, the updated content of the pth boundary cell equals $\sqrt{\lambda^2 x^2 + z^2}$ and that of the pth cell of the EDA is $s_p z + c_p \lambda = \sqrt{\lambda^2 x^2 + z^2}$, where $s_p = z/\sqrt{\lambda^2 x^2 + z^2}$ and $c_p = \lambda x/\sqrt{\lambda^2 x^2 + z^2}$. Therefore, the contents of both array are identical. Since the rotation coefficients, c_p and s_p, generated by PE_{pp} boundary cell are proportional to x and z respectively, the output of the pth cell of the EDA, e_0, is $c_p z - s_p x$ which is always zero. □

If there is a fault in either the pth column of the array or the EDA, these contents are no longer identical and then lead to a non-zero e_0. Thus a fault is detected. However, if there is any fault outside of these two arrays, then the errors produced by that fault will affect both of these arrays in the same manner (*i.e.*, contents of both arrays are still identical) and resulting in a zero e_0. Thus, these faults will not be detected by the $\underline{y}_0 = \underline{a}(p)$ design. Clearly, we can generalize the above results by the following Lemma.

Lemma 4.2: Given the ADR $\underline{y}_0 = \underline{a}(k)$, for $1 \leq k \leq p$, the contents of the kth column

array of the QRD triarray and the first k cells of the EDA are identical. The output of the kth cell of the EDA are zero. The contents of cell l, $k+1 \leq l \leq p$, of the error detection array are all zeros.

Corollary 4.1 A fault occurring outside of the kth column of the QRD triarray and the EDA will not be detected if the ADR is designed as $\underline{y}_0 = \underline{a}(k)$.

Proof: From the previous discussion, a fault occurring in the ith, $1 \leq i \leq k-1$, column of QR array will not be detected. From Lemma 4.2, the output of the kth cell and the contents of cell l, $k+1 \leq l \leq p$, of the EDA are all zeros. Thus, any fault occurring to the ith, $k+1 \leq i \leq p$, column of QR array will be masked by these zeros. The optimal residual e_0 is always zero unless there is any inconsistency between the kth column of the QR array and the EDA.□

From all the above observations, by selecting $\underline{y}_0$ properly as given in the following theorem, we can detect the presence of a fault in any location of the system.

Theorem 4.1 (Concurrent Error Detection Theorem) Consider the selection of the artificial desired response $\underline{y}_0 = \sum_{i=1}^{p} \underline{a}(i)$. If there is no fault in the system, then the output of the EDA with $\underline{y}_0$ as an input yields $e_0 = 0$. If there is a fault in the system, then $e_0 \neq 0$.

Proof: From Lemma 4.2, each $\underline{a}(i)$ is "zeroed out" by the ith cell of the EDA. Any error produced by a faulty processor, say in the jth column of the QR array, will not be zeroed out by the jth cells of the EDA. The output of the jth cell is then non-zero and propagates down to the output. Therefore, whenever $e_1 \neq 0$, there is a fault in the system.□

The ADR $\underline{y}_0 = \sum_{i=1}^{p} \underline{a}(i)$ is obtained by implementing a top row encoding array (EA) consisting of p summing cells as shown in Fig.2. The response array (RA), with the desired response $\underline{y}$ as input and e as output, located at the right of the EDA is incorporated with the system to produce the desired residual. Once $e_0 \neq 0$, which indicates the system had a fault, then $e(n)$ is considered to be in error and will not be used. The error detection is thus achieved in real-time.

Example 1: An adaptive filter using QRD LS systolic array with order $p = 3$ is simulated. In between $t = 25$ and $t = 35$, a fault occurred in cell PE_{23} in such a way that random noise within range $[-1, 1]$ is generated. From Table 1, we can see, due to the propagation delay which will be considered in section 5.1.1, from $t = 28$ to $t = 38$, $e_0 \neq 0$ results from errors generated by the faulty cell. The optimal residuals obtained from e from $t = 28$ to $t = 38$ are then considered faulty. After $t = 39$, e_0 then decays down due to the adaptive nature of the algorithm. Fig.3(a) plots $|e_0|$ versus t and shows the adaptive effect of the algorithm. A threshold device can be used with e_0 to provide a decision on the size of the error that can be tolerated. Fig.3(b) shows the $|e_0|$ in Fig.3(a) with threshold set at 0.3. A generalization of the proposed scheme is stated below.

Theorem 4.2 (Generalized Error Detection Theorem) Any fault occurring in the system can be detected if the ADR is given by $\underline{y}_0 = \sum_{i=1}^{p} c_i \underline{a}(i)$, where $c_i \neq 0, 1 \leq i \leq p$.□

The simplest ADR that can detect fault is indeed a checksum encoded data (given by *Theorem 4.1*) which is a special case of the set of ADR given by *Theorem 4.2*. However, unlike the checksum fault detection scheme in [3,6], *Theorem 4.1 and 4.2* provide a real-time fault detection scheme using the inherent nature of QR LS systolic array.

4.1.1 Optimal Efficiency

Since a column of linear EDA and a row of linear EA are required, the complexity of this fault detection scheme is $2p$. That is, $2p$ redundant processors is required. Compare to the complexity of the triangular QR LS array, $(p^2 + 3p)/2$, it is a cost-effective real-time fault detection scheme. Here we do not count the final output multiplier cells in the EDA and RA.

We define the *hardware efficiency* Ω_h to be the ratio of the hardware cost of implementing the algorithm to the cost of implementing the algorithm with an error-detection capability. We see $\Omega_h(p) = (p^2 + 3p)/(p^2 + 7p)$. Thus $1/2 \leq \Omega_h(p) \leq 1$ since a single error can be detected by duplicating the hardware. When $\Omega_h(p) = 1$, we say the error-detection scheme is *most hardware efficient.* Define the *time efficiency* Ω_t to be the ratio of the time to implement the algorithm and the time to implement the algorithm incorporating the error-detection scheme. Obviously, time efficiency is bounded by $0 \leq \Omega_t \leq 1$. When $\Omega_t = 1$, we say the error-detection scheme is *most time efficient.* If an error-detection scheme is both most hardware and time efficiency, then it is said to be *optimal.* For the proposed residual method, clearly $\lim_{p\to\infty} \Omega_h(p) = 1$ as shown in Fig.4. That is, it is asymptotically most hardware efficient. However, the time efficiency $\Omega_t(p) = 1, \forall p$, so that it is also most time efficient. Therefore, the residual method is an asymptotically optimal error-detection scheme for the recursive LS systolic array.

4.2 Fault Diagnosis

When a fault is detected, the system leaves the concurrent error detection phase and enters the fault diagnosis phase. The main purpose of this phase is to find the faulty processor row. Either of two methods, the flushing fault location (FFL) method or the checksum encoding (CSE) method, can be used to diagnose and locate the faulty row. The FFL method is developed under the assumption that only the residual output e_0 can be accessed externally, while the CSE method assumes that all the cells of the EDA can be accessed.

4.2.1 Flushing Fault Location Method

During the concurrent error detection phase, a fault is detected based on unknown values of the incoming data in $\underline{a}(k)$, $1 \leq k \leq p$, and contents of all the cells. However, in the FFL method, we will control the desired incoming data as well as the contents of the tri-array and the EDA, in order to obtain an appropriate value in e_0 to locate the faulty processor row. In the FFL method, we do not use the operations of the RA cells or the EA cells. From (3), the weight vector $\underline{\hat{w}} = (\hat{w}_1, \hat{w}_2, \cdots, \hat{w}_p)$ can be solved by using the back substitution method of

$$\hat{w}_i = \frac{P_i - \sum_{j=i+1}^{p} r_{ij}\hat{w}_j}{r_{ii}}, \quad i = p,\ p-1, \cdots, 1 \tag{6}$$

where P_i and r_{ij} are elements of vector $\underline{P}$ and matrix R. A linear array to performed the back substitution as in [12] can be prevented by using the *weight flushing technique* [2,20].

In a fault-free triangular LS systolic array, by "freezing" the QRD upper triangular matrix $R(n)$ and the associated column vector $\underline{P}(n)$ in (3), the optimum solution $\underline{\hat{w}}(n)$ can be "flushed" out sequentially with a skewed identity matrix input. In these operations, the internal cells (defined in Fig.2) in a given row, act as pure bypass elements with no Givens

rotations, when the input to the boundary cell of that row is zero and $c = 1$ and $s = 0$ are being propagated. For example, let $p = 3$ and $\underline{P} = (P_1, P_2, P_3)$, then

$$\begin{aligned} \hat{w}_3 &= P_3/r_{33}, \\ \hat{w}_2 &= (P_2 - r_{23}\hat{w}_3)/r_{22}, \\ \hat{w}_1 &= (P_1 - r_{12}\hat{w}_2 - r_{13}\hat{w}_3)/r_{11}. \end{aligned} \tag{7}$$

A data flow graph is shown in Fig.5 to illustrate this computation.

However, due to errors generated by the faulty processor, various parts of $R(n)$ and $\underline{P}$ stored in the array are no longer correct. A new test triangular matrix T and a test vector $\underline{P}_t$ are load to the tri-array and EDA respectively. The values of T and $\underline{P}_t$ can be either pre-stored or distributed by an host computer when a fault is detected. Specifically, define $T = [\underline{t}_1, \underline{t}_2, \cdots, \underline{t}_p]$ as a $p \times p$ all-1's upper triangular matrix, where $\underline{t}_i$ is a $p \times 1$ vector, and $\underline{P}_t$ as a $p \times 1$ all-1's vector. Since $\underline{P}_t = \underline{t}_p$, the optimal solution vector $\underline{\hat{w}}_0 = [0, 0, \cdots, 1]$, as given by (3). One of the reasons for the selection of these all 1's in T and $\underline{P}_t$ is to reduce memory requirement. Only one-bit register is required for each cell or distribution requirement.

With T and $\underline{P}_t$ frozen, consider a skewed $p \times p$ identity matrix input to the first p columns and all zeros input to the EDA. In the absence of a fault, components of the optimum solution vector, $\underline{\hat{w}}_0 = [0, 0, \cdots, 1]$, are outputted sequentially at e_0 [2]. Denote $\underline{\phi}_i^T = [0, \cdots, 0, 1, 0, \cdots, 0]$ as a $1 \times p$ vector of all zeros except for an one at the ith position, representing the ith row of the skewed identity matrix input to the array. Then the output $e_0(1)$ in response to $\underline{\phi}_1^T$ is processed by all the cells from the first to the pth row of the array. In general, $e_0(i)$ is the response to $\underline{\phi}_i^T$ (*i.e.* $e_0(i) = \underline{\phi}_i^T \underline{\hat{w}}_0$), and is processed by all the cells from the ith to the pth row. As considered above, in the absence of a fault, $e_0(i) = 0,\ 1 \le i \le p-1$, and $e_0(p) = 1$. However, with a fault in the kth row, then $e_0(i) \ne 0,\ 1 \le i \le k$, but responses due to $\underline{\phi}_j^T$ for $k+1 \le j \le p$, encountering only fault free processing cells from the jth to the pth row, will yield the correct value. This property can be used to locate the faulty row in the FFL method.

Theorem 4.3: (Flushing Fault Location Theorem) When a fault is detected and the system enters the fault diagnosis phase, both the EA and RA arrays are made inoperative and all 1's are loaded into the contents of the processor cell. A skewed identity $p \times p$ matrix is flushed into the system with all zeros input to the EDA. The EDA output $e_0(i), 1 \le i \le p$ is obtained sequentially. Assume the first zero output at $e_0(k+1) = 0$, occurs for some k, $1 \le k \le p-2$, then the faulty processor is in the kth row. If there is no such k for $e_0(k+1) = 0, 1 \le k \le p-2$, and $e_0(p) = 1$, then the $(p-1)^{th}$ row is the faulty row; otherwise, with $e_0(p) \ne 1$, the pth row is the faulty row.□

Example 2: A QRD RLS array with order $p = 5$ is considered. Suppose a fault has occurred in PE_{34}. When a skewed 5×5 identity matrix is flushed into the system, due to the randomly generated noise from the faulty cell PE_{34}, the outputs from e_0 are given by $[0.2127, -0.5714, 0.7453, 0., 1.]$. In the absance of an error, the outputs should be $[0., 0., 0., 0., 1.]$. Since the first three elements are erroneous, based on *Theorem 4.3*, the faulty cell is in the third row.

It is obvious the flushing of $\underline{\phi}_1$ is unnecessary since computations involving the entire QR array are definitely incorrect in the fault diagnosis phase.

4.2.2 Checksum Encoding Method

The basic assumption of the CSE method is that all the cells of the EDA can be accessed. Further more, all the contents of the tri-array can be piped out in the diagnosis phase. In this paper, instead of using the CSE as a fault detector as in [3], it is used to diagnose fault location when a fault has been detected. The disadvantages of the checksum scheme for real-time application is thus prevented. It has been shown in [3,6] that the QRD of a row checksum matrix A_r results in a row checksum upper triangular matrix R_r. Let $r_{ij}(n)$ be the content of processor PE_{ij} of the tri-array at time n and $P_i(n)$ be the content of the ith processor of the EDA at time n.

Theorem 4.4 (Checksum Encoding Theorem) Given the artificial desired response $y_0 = \sum_{i=1}^{p} \alpha_i \underline{a}(i)$, for $\alpha_i \neq 0$, the checksum

$$\sum_{k=0}^{p-i} \alpha_i r_{i(i+k)}(n+k) = P_i(n+p-i+1), \tag{8}$$

holds for $i = 1, 2, \cdots, p, \quad n = p, p+1, p+2, \cdots$, if no fault has occurred. If there is an m such that the checksum does not hold for $m \leq i \leq p$, then there is a fault in the system and the faulty processor is in the first row that does not meet the checksum. □

The time indexs are introduced to describe the time difference for a given row input of data to the array. Each column of input, say $\underline{a}(k)$, is zeroed out by the kth cell of EDA. Thus the content of the kth cell of EDA is affected only by $\underline{a}(i), \quad k \leq i \leq p$. Therefore, if there is a faulty processor, say in row m, then all the rows below do not satisfy the checksum because of the error produced by the faulty one which cannot be zeroed out by the mth cell of EDA.

Example 3: Consider a $[R \vdots \underline{P}_0]$ matrix of intermediate results for a QRD RLS array with order $p = 4$,

$$\begin{bmatrix} 0.7930 & 0.7462 & 0.4655 & 0.9774 & \vdots & 2.9821 \\ 0. & 1.2973 & 1.0816 & 1.0379 & \vdots & 3.4168 \\ 0. & 0. & 0.2729 & 0.3643 & \vdots & 1.7132 \\ 0. & 0. & 0. & 0.1675 & \vdots & 0.8375 \end{bmatrix},$$

where the $(i, i+k)$ element, $r_{i,(i+k)}$, of R, takes the value at time $n+k$ due to time skewing of the input. That is, $r_{i,(i+k)}(n+k)$. The ith element of $\underline{P}$ takes the value at time $n+5-i$. That is, $P_i(n+5-i)$. As we can see, after the third row, the checksums are no longer met for each row. From *Theorem 4.4*, the faulty cell is in the third row.

Unlike the FFL method, the CSE method cannot stop the concurrent error detection phase immediately when a fault is detected. Because of the skewed manner of inputting the data, if we stop the operation immediately, the checksum property will not hold according to *Theorem 4.4*. Each processor $PE_{ij}, \quad 1 \leq i \leq j \leq p$, of the QR tri-array has to take $j-i$ more data and the ith cell of the EDA has to take $p+1-i$ more data so that the checksum is satisfied for each row of the systolic array. If each data requires one system clock, we observe that at most p more system clocks are needed to process those unfinished data after the moment a fault is detected. The last row takes one clock and the first row

takes p clocks. Generally, the ith row takes $p+1-i$ clocks to process the unfinished data. Thus, those rows which take fewer clocks can pipe their final results out to the right to check their checksum while others rows are still working on their unfinished data.

4.2.3 Order-Degraded Reconfiguration

An order-degraded performance is reasonable and often acceptable in many LS applications. A reconfiguration is needed to reroute data paths for order-degraded operation. Many models and approaches can be found in the literatures [7,26,30,36] for the reconfiguration of VLSI array processors. Here we use a similar model described in [7]. When the faulty row, say row k, is determined, the cells in the kth column and row become connection elements and enter a dormant state. In the dormant state, each cell tests itself to check its status repeatedly [7]. The reduced $(p-1) \times (p-1)$ tri-array then operates in an order-degraded LS computational manner. Fig.6 shows an example of bypassing the faulty row and the associated column to become an order-degraded LS array. When the (transient) fault is removed, the dormant cells reactivate and generate an interrupt immediately. The reactivation scheme recovers all of the cells which become connection elements before and turns them into active cells. Then the full-order LS operation is resumed. Details on various schemes and technologies of reconfiguration can be found in [7,26,30,36].

5 Error Propagation and Recovery Latency

In this section, we derive and compare the performances of the proposed fault-tolerant schemes.

5.1 Robust Error Detection

Some definitions needed in the analysis are first given. A processor is said to be *faulty* if a fault has occurred in it and is said to have been *contaminated* if it contains erroneous data or states. The *faulty moment* is the time instant a fault first occurs within the processor, and the *contaminating moment* is the time instant the first error occurs in the processor [16]. A fault is said to be *observable* if some of the observable states , such as signal data and system output, of the processor can be affected by the fault. It may take some latent period to change the observable states incorrectly once the fault occurs. An *immediately observable fault* is a fault which affects some of the observable states right after its occurrence. To make the analysis tractable, we first assume all faults are immediately observable faults such that when the fault occurs, it produces error at the output of the faulty processor immediately. Later, we will relax this assumption for more general results. The error is then propagated and contaminates those processors affected. The moment the error is first observed at the output of the system is called the *error observed moment.* First, assume a fault occurs in an internal cell $PE_{ij}, i \neq j$, at a faulty moment. The output of this faulty cell is thus erroneous and can be described by $x^{\epsilon}_{out} = x_{out} + \delta$, where x_{out} is the fault-free output and δ is the error generated by the fault. The error propagation path can be described by

$$PE_{ij} \rightarrow PE_{(i+1)j} \rightarrow \cdots \rightarrow PE_{jj},$$

and then $PE_{kl}, \;\; k \geq j, \;\; l \geq j$ are all contaminated.

From the operations executed by the internal cell, the error is modified to $c_{i+1}\delta$ by $PE_{(i+1)j}$ and the cumulative modifications of the error before reaching the boundary cell, PE_{jj}, is

$$\eta = \delta \prod_{k=i+1}^{j-1} c_k, \tag{9}$$

where c_i is the cosine parameter generated by the boundary cell PE_{ii}. Therefore, c_j and s_j are erroneous and are given by

$$c_j = \lambda r / \sqrt{\lambda^2 r^2 + (x_{in} + \eta)^2}, \qquad s_j = (x_{in} + \eta) / \sqrt{\lambda^2 r^2 + (x_{in} + \eta)^2}. \tag{10}$$

In this case, s_j is no longer proportional to x_{in}, $\underline{a}(j)$ will not be zeroed out by the jth cell of the EDA. The size of the error generated by this cell is

$$\eta_j = c_j x_{in} - s_j \lambda r = -\lambda r \eta / \sqrt{r'^2 + 2\eta x_{in} + \eta^2}, \tag{11}$$

where $r' = \sqrt{\lambda^2 r^2 + x_{in}^2}$ is the new updated uncontaminatedvalue of the content of PE_{jj}. Although those c_k's and s_k's for $j \le k \le p$ are contaminated, $\underline{a}(k)$, $j \le k \le p$, are zeroed out by the kth cell of the EDA because of the consistency of the sine and cosine parameters used by the kth column array of the QR tri-array and the EDA. When η_j propagates down to the output of the EDA, η_j is influenced by the contaminated cosines c' of each following row. The error output at e_0 due to an error δ generated at PE_{ij} is then given by

$$\begin{aligned} e_0^\delta(i,j) &= -\gamma \prod_{m=j+1}^{p} c'_m \frac{\eta \lambda r}{\sqrt{r'^2 + 2\eta x_{in} + \eta^2}} \\ &= \begin{cases} \dfrac{-\gamma\lambda r \prod_{k=i+1}^{j-1} c_k \prod_{m=j+1}^{p} c'_m \delta}{\sqrt{r'^2 + 2x_{in} \prod_{k+1}^{j-1} c_k \delta + \prod_{k=i+1}^{j-1} c_k^2 \delta^2}} & i \le j-2, \\ \dfrac{-\gamma\lambda r \prod_{m=j+1}^{p} c'_m \delta}{\sqrt{r'^2 + 2x_{in}\delta + \delta^2}} & i = j-1. \end{cases} \end{aligned} \tag{12}$$

where $\gamma = \prod_{i=1}^{j-1} c_i \prod_{k=j}^{p} c'_k$ [1]. Next, assume a fault occurs in a boundary cell, PE_{jj}, $1 \le j \le p$, at the faulty moment. Both erroneous c'_j and s'_j produced by PE_{jj} can be written by

$$c'_j = \frac{\lambda r + \delta_c}{r'_\epsilon}, \qquad s'_j = \frac{x_{in} + \delta_s}{r'_\epsilon}, \tag{13}$$

where δ_c and δ_s represent errors in the numerators while r'_ϵ represents the erroneous content of the denominators of c_j and s_j. The error produced by the jth cell of the EDA is then given by

$$\eta_j = c'_j x_{in} - s'_j \lambda r = (x_{in}\delta_c - \lambda r \delta_s)/r'_\epsilon, \tag{14}$$

and the output error at e_0 due to a faulty boundary cell is given by

$$e_0^\delta(j,j) = \gamma \prod_{m=j+1}^{p} c'_m \cdot \frac{x_{in}\delta_c - \lambda r \delta_s}{r'_\epsilon}. \tag{15}$$

A fault can occur either in the internal or boundary cell. First assume the fault occurs in an internal cell. The cosine parameter, c_j, produced by PE_{jj} is bounded by $0 \le c_j \le 1$.

Lemma 5.1: For some c_j, if there exists an $n \in I$ such that $c_j(n) \neq 0$, then $c_j(m) > 0$ for all $m > n$, $m \in I$.
Proof: The cosine parameter is given by $c_{k+1} = \lambda r(k)/r'(k)$, where $r(k+1) = r'(k) = \sqrt{\lambda^2 r^2(k) + x_{in}^2(k)}$. $\exists n \rightarrow c_j(n) \neq 0$ is equivalent to say that $\exists n \rightarrow r(n) \neq 0$ or $r(n) > 0$. Since $r(k+1) = r'(k) \geq \lambda r(k)$, we have $r(k) > 0$ for all $k \geq n$. Therefore, $c_j(k) > 0$ for all $k > n$.□

For linearly independent input column vectors, all $c_j(n) \neq 0$, and from (12), we see $e_0^\delta \neq 0$ if an error has been introduced. Thus the fault can always be detected in this case. When the fault occurs in a boundary cell, from (15) we conclude that the only chance of a loss of detection is that $e_0^\delta = 0$. This happens only if $\delta_c/\delta_s = \lambda r/x_{in}$, which is very unlikely since the value of δ_c, δ_s, r, and x_{in} are not related in any manner. Furthermore, even if δ_c/δ_s equals $\lambda r(n)/x_{in}(n)$ for some instant of time at n, it is even more unlikely that δ_c/δ_s equal $\lambda r(n+1)/x_{in}(n+1)$ at time $n+1$. We can summarize these results in the following theorem. Note that the statement of *Theorem 5.1* is valid only for infinite-precision implementation.

Theorem 5.1 (Robust Error Detection Theorem): An error produced by a faulty processor at the faulty moment will be detected at the EDA output e_0 and the probability of error detection given a fault occurs equal one. That is,

$$Pr(\text{error detected at } e_0 \mid \text{a fault occurred}) = 1.\square \tag{16}$$

5.1.1 Latency

Now, we consider some basic issues related to latencies in the array.
Definition 5.1: The *system latency*, t_s, is the time between the moment of data input to the system and the moment of the output of this data from the system.
Definition 5.2: The *processing latency*, t_p, of processor PE_{ij} is the time between the moment a data in a wavefront inputs to the system and the moment PE_{ij} is processing data from that wavefront.
Definition 5.3: The *error propagation latency*, t_ε, is the time between the faulty moment and the error observed moment.
It is clear that the system latency of the QR recursive LS array depends on the number of processors and delay elements on the boundary and is given by $t_s = 2p+1$. The processing latency of processor PE_{ij}, $1 \leq i \leq j \leq p+1$, is given by $t_p = (i+j)-1$. Since there are a totally of $p(p+3)/2$ processors, the expected processing latency is

$$E(t_p) = \frac{2}{p(p+3)} \sum_{i=1}^{p} \sum_{j=i}^{p+1} (i+j-1) = \frac{p^2+4p+1}{p+3}. \tag{17}$$

The error propogation latency is given by

$$t_\varepsilon = t_s - t_p = 2(p+1) - (i+j), \tag{18}$$

and the expected value is

$$E(t_\varepsilon) = (p+1)(p+2)/(p+3). \tag{19}$$

Fig.7 shows the plots of $E(t_p)$ and $E(t_\varepsilon)$.
Definition 5.4: The *fault diagnosis time*, t_f, of a faulty processor PE_{ij} be the minimum

time required to locate the faulty row right after the error observed moment.
Definition 5.5: The *recovery latency*, t_r, be the time between the faulty moment and the moment the faulty row is determined.
Since the system latency is $2p+1$, for the FFL method the fault diagnosis time of processor PE_{ij} for the array is $t_f^{FFL} = (2p+1)+i$. We can show that the fault diagnosis time for the CSE method is $t_f^{CSE} = p+2-i$. The expected value for fault diagnosis time are $E(t_f^{FFL}) = (5/2)p+1$ and $E(t_f^{CSE}) = (1/2)p+2$ respectively. By the definition of the recovery latency, we have $t_r = t_\varepsilon + t_f$. Therefore, the recovery latency are $t_r^{FFL} = 4p-j+3$ and $t_r^{CSE} = 3p-2i-j+4$, while the expected recovery latency are

$$E(t_r^{FFL}) = (7p^2+23p+10)/(2p+6), \qquad E(t_r^{CSE}) = (3p^2+13p+16)/(2p+6) \tag{20}$$

respectively. Due to the facts that multiple ports can be accessed externally and we can use the parallel pipe-out feature of the CSE method, it is not surprised that the performance of the CSE method is better than that of the FFL method as indicated in Fig.8. However, for both cases, the order of the expected recovery latency is $O(p)$, which is linear with respect to p. In practice, a (transcient) fault may not be necessarily an immediately observable fault. Without this assumption, all the values obtained in this section become the lower bounds of those parameters. That is, performance obtained by the assumption of immediately observable fault is the best performance we can achieve.

6 Conservation Test

Thus far the fault-tolerance system discussed above is designed to detect the fault occurring in the QR triarray and the EDA. It is not applicable to the RA which computes the real desired response. In this section, we introduce some conservation properties to tackle this problem and the data is still assumed to be real.

A: <u>Cell Level Test</u>

A1: **Energy Conservation** The rotation operation of internal cell is described in Fig.1. Denote the updated r as r'. Then it can be seen that

$$x_{out}^2 + r'^2 = x_{in}^2 + \lambda^2 r^2, \tag{21}$$

which means the energy *i.e.*, (the 2-norm) is conserved before and after the operation. This is due to the Givens rotation transformation which is an unitary transformation that conserves the energy.

A2: **Inverse of Unitary Matrix** The energy conservation test above requires the square operation which is different from the operation of the internal cell. In a VLSI/WSI system, the number of different kinds of processors or cells is usually kept as low as possible [8,29]. Observe that the inverse of a unitary matrix is the Hermitian transpose of it. If there is no fault occurring during the computations, the inverse computation should give the original values. That is

$$\begin{bmatrix} x_{in} \\ \lambda r \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_{out} \\ r' \end{bmatrix}. \tag{22}$$

One of the above inverse computation is enough to detect a fault. For example, the inverse computation of $x_{in} = cx_{out} + sr'$ will not match the original x_{in}, if x_{out} or r' is erroreous. This kind of test can be carried out by the internal cell.

B: System Level Test

Consider the RA which computes the real desired response. Based on the previous discussion, the energy is conserved for unitary transformation. Denote $P_i(n)$, $1 \leq i \leq p$, to be the content of the ith cell of the array at time n and d as defined in section 2. The system level energy conservation is described by

$$\sum_{i=1}^{p} P_i^2(n+i-1) = \sum_{j=1}^{n} d^2(j). \tag{23}$$

That is, the energy of the input signal is equal to the energy in the RA. Note that the introduced time index is used to describe the operations of continuously updating the incoming signal. Both cell and system level tests can be built either in the system or applied externally depending on the requirements of the application.

7 Conclusions

In this paper, special inherent natures of a QRD recursive LS systolic array are used to design a real-time, low-cost algorithm-based fault-tolerant system. By proper design of the artificial desired response, the residual output serves as a concurrent error detector. The disadvantages of the CSE scheme to detect fault are thus prevented. The proposed method requires the same complexity as that of the CSE scheme in [3] without hindering the throughput rate of the system. Two methods, FFL and CSE, are then introduced to locate the faulty row. For both methods, the recovery latency is achieved in $O(p)$ time. However, the recovery latency of the CSE method is generally less than that of the FFL method in that parallel execution is possible and multiple ports can be accessed. Once the faulty row is determined, an order-degraded reconfiguration is performed so that the system is operating in an gracefully degraded manner. Any single fault occuring in the system will not cause a catastrophic degradation of the system and thus one critical requirement for many high performance and demanding VLSI signal processing tasks is met.

This fault-tolerance system is general in the sense that it is independent of the implementation algorithms. It is applicable to Given rotation method as well as those methods such as Modified Gram-Schmidt, square-root free Given methods [19], and Householder trnasformations [35] etc. The LS problems with constraints such as MVDR beamforming is also applicable [11]. The residual method for concurrent error detection is robust in the sense that the probability of error detection given by a fault occurred equals one (for infinite-precision implementation). This scheme could have great impact on next geneation of radar systems which may use VLSI array processors as their central signal processing units.

8 Acknowledgement

We thank several anonymous reviewers for their suggestions and comments that improved the quality of this paper.

References

[1] J.G. McWhirter, "Recursive least-squares minimization using a systolic array", Proc. SPIE, Vol 431, Real Time Signal Processing VI, pp.105-112, 1983.

[2] C.R. Ward, P.J. Hargrave and J.G. McWhirter, "A novel algorithm and architecture for adaptive digital beamforming", IEEE Trans. Antennas Propagat., Vol AP-34, pp.338-346, March, 1986.

[3] C.-Y. Chen and J.A. Abraham, "Fault-tolerant systems for the computation of eigenvalues and singular values", Proc. SPIE, Vol 696, Advanced Algorithms and Architectures for Signal Processing, pp.228-237, 1986.

[4] C.-Y. Chen and J.A. Abraham, "Current error detection in VLSI processor arrays", Proc. SPIE, Vol 826, Advanced Algorithms and Architectures for Signal Processing II, pp. 205-214, 1987.

[5] K.-H. Huang and J.A. Abraham, "Algorithm-based fault-tolerance for matrix operations", IEEE Trans. Computer, Vol C-33, pp.518-528, June, 1984.

[6] J.-Y Jou and J.A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures", Proc. IEEE, Vol 74, pp.732-741, May, 1986.

[7] S.Y. Kung, **VLSI Array Processors**, Prentice Hall, 1988.

[8] F.T. Luk and H. Park, "Fault-tolerant matrix triangulization on systolic array", IEEE Trans. Computer, Vol 37, pp.1434-1438, Nov. 1988.

[9] S.-W. Chan and C.-L. Wey, "The design of concurrent error diagnosable systolic arrays for band matrix multiplications", IEEE Trans. CAD, Vol 7, pp.21-37, Jan. 1988.

[10] R.J. Cosentino, "Concurrent error corrections in systolic architecture", IEEE Trans. CAD, Vol 7, pp.117-125, Jan. 1988.

[11] J.G. McWhitter and T.J. Shepherd, "Systolic array processor for MVDR beamforming", IEE Proc. Vol 135, Pt. F, pp.75-80, 1989.

[12] W.M. Gentleman and H.T. Kung, "Matrix triangularization by systolic arrays", Proc. SPIE, Vol 298, Real Time Signal Processing IV, pp.298-303, 1981.

[13] J.A. Abraham et al., "Fault tolerance techniques for systolic array", IEEE Computer, Vol 20, pp.65-76, July 1987.

[14] J.V. McCanny and J.G. McWhitter, "Some systolic array developments in the United Kingdom", IEEE Computer, Vol 20, pp.51-64, July 1987.

[15] S. Haykin, **Adaptive Filter Theory**, Prentice Hall, 1986.

[16] K.G. Shin and T.-H. Lin, "Modeling and measurement of error propogation in a multimodule computing system", IEEE Trans. Computer, Vol 37, pp.1053-1066, Sep. 1988.

[17] C.J. Anfinson and F.T. Luk, "A linear algebraic model of algorithm-based fault tolerance", Proc. IEEE Int'l Conf. Systolic Array, pp.483-493, May 1988.

[18] A. Avizienis, "Fault tolerant computing - An overview", IEEE Computer, Vol 4, pp.5, Jan. 1971.

[19] M.J. Chen, "On realizations and performances of least-squares estimation and Kalman filterring by systolic array", Ph.D. dissertation, Electrical Engineering Dept., UCLA, 1987.

[20] K. Yao and T.Y. Yan, **Processing and Algorithms in Communication and Radar Systems**, to be published, Springer-Verlag.

[21] J.P. Hayes, **Computer Architecture and Organization**, McGraw-Hill, 1987.

[22] J.A.B. Fortes and C.S. Raghavendra, "Gracefully degradable processor arrays", IEEE Trans. Computer, Vol C-34, pp.1033-1044, Nov. 1985.

[23] S.Y. Kung *et al.*, **VLSI and Mordern Signal Processing**, Prentice-Hall, 1985.

[24] H.T. Kung and M.S. Lam, "Wafer-scale integration and two-level pipelined implementaiton of systolic array", J. Parallel Distrib. Comput., Vol 1, pp.32-63, 1984.

[25] H. Lev-Ari and B. Friedlander, "On the systematic design of fault-tolerant processor arrays with application to digital filter", Proc. IEEE Workshop on VLSI Signal Processing, pp.483-494, Nov. 1988.

[26] S.N. Jean, C.W. Chang and S.Y. Kung, "Graceful degradation schemes for static/dynamic wavefront array", Proc. Int'l Conf. Parallel Processing, pp.249-255, Aug. 1988.

[27] J.-Y. Jou and J.A. Abraham, "Fault-tolerant algorithms and architectures for real time signal processing", Proc. Int'l Conf. Parallel Processing, pp.359-362, Aug. 1988.

[28] J.-Y. Jou and J.A. Abraham, "Fault-tolerant matrix operation on multiple processor system using weigted checksums", Proc. SPIE Vol 495 Real Time Signal Processing VII, pp.94-101, 1984.

[29] H.T. Kung, "Why systolic architectures?", IEEE Computer, Vol 15, pp.37, Jan. 1982.

[30] S.Y. Kung, C.W. Chang, and C.W. Jen, "Real-time reconfiguration for fault-tolerant VLSI array processors", Proc. Real-Time Systems Symposium, pp.46-54, 1986.

[31] I. Koren and D.K. Pradham, "Yield and performance enhancement through redundancy in VLSI and WSI multiprocessor systems", IEEE Proc. Vol 74, pp.699-711, May, 1986.

[32] I. Koren and M.A. Breuer, "On area and yield considerations for fault-tolerant VLSI processor arrays", IEEE Trans. Computer, Vol C-33, pp.21-27, Jan. 1984.

[33] C.J. Anfinson et al., "Algorithm-based fault-tolerant techniques for MVDR beamforming", Proc. IEEE ICASSP, pp.2417-2420, May, 1989.

[34] K.J.R. Liu and K. Yao, "Gracefully degradable real-time algorithm-based fault-tolerant method for QR recursive least-squares systolic array", Proc. International Conference on Systolic Array, pp. 401-410, Killarney, Ireland, May, 1989.

[35] K.J.R. Liu, S.F. Hsieh, and K. Yao, "Recursive LS filtering using block Householder transformations", Proc. IEEE Int'l Conf. Acoustic, Speech, and Signal Processing, Albuguerque, 1990.

[36] M. Sami and R. Stefanelli, "Reconfigurable architectures for VLSI processing arrays", Proc. IEEE, pp.712-722, May 1986.

Figure Captions:

Table 1. The output of e_0 and e of an adaptive QRD LS filter with order $p = 3$ when a fault occurred in PE_{23} from $t = 25$ to $t = 35$.

Fig. 1(a). Recursive LS systolic array with optimum residual e.

Fig. 1(b). Processing cells of LS systolic array.

Fig. 2(a). Fault-tolerant recursive LS systolic array based on linear combination of input data in top row array feeding into column error detection array with output fault indication variable e_0.

Fig. 2(b). Processing cells of fault-tolerant systolic array.

Fig. 3.(a) Plot of $|e_0|$ of an adaptive QRD LS filter with order $p = 3$ when a fault occurred in PE_{23} from $t = 25$ to $t = 35$.

Fig. 3.(b) Plot of $|e_0|$ in Fig.3(a) with threshold set at 0.3.

Fig. 4. Plot of the hardware efficiency of the residual method versus the order of the LS estimation.

Fig. 5. Data flow graph of the optimal weight vector $\underline{\hat{w}}(n)$ with a skewed input identity matrix.

Fig. 6. Reduced $(p-1) \times (p-1)$ tri-array after the deletion of the row and column with a faulty cell.

Fig. 7. Plot of expected processing latency and expected error propagation latency versus the order of the LS estimation.

Fig. 8. Comparisons of expected recovery latency for FFL and CSE methods.

t=1	e0=0.000000	e=0.000000
t=2	e0=0.000000	e=0.000000
t=3	e0=0.000000	e=0.000000
t=4	e0=0.000000	e=0.000000
t=5	e0=0.000000	e=0.000000
t=6	e0=0.000000	e=0.000000
t=7	e0=-0.000000	e=0.000000
t=8	e0=-0.000000	e=-0.000000
t=9	e0=-0.000000	e=-0.000000
t=10	e0=-0.000000	e=-0.000000
t=11	e0=0.000000	e=-0.000000
t=12	e0=0.000000	e=0.000684
t=13	e0=0.000000	e=-0.153411
t=14	e0=0.000000	e=-0.138241
t=15	e0=0.000000	e=0.155885
t=16	e0=-0.000000	e=0.065978
t=17	e0=-0.000000	e=0.507390
t=18	e0=-0.000000	e=0.000090
t=19	e0=0.000000	e=-0.704896
t=20	e0=-0.000000	e=0.018910
t=21	e0=0.000000	e=0.053229
t=22	e0=0.000000	e=-0.787624
t=23	e0=-0.000000	e=0.015445
t=24	e0=0.000000	e=-0.287406
t=25	e0=0.000000	e=-0.677166
t=26	e0=0.000000	e=-0.050505
t=27	e0=0.000000	e=0.817616
t=28	e0=-0.966001	e=-0.553705
t=29	e0=-0.611989	e=0.325937
t=30	e0=1.083665	e=0.102332
t=31	e0=-0.506277	e=-0.100179
t=32	e0=-0.391151	e=-0.430207
t=33	e0=0.538918	e=0.689049
t=34	e0=-0.426679	e=-0.220855
t=35	e0=-0.558384	e=0.312747
t=36	e0=0.975315	e=-0.117668
t=37	e0=-0.748951	e=0.038787
t=38	e0=0.144688	e=0.219978
t=39	e0=-0.053715	e=0.386494
t=40	e0=-0.014142	e=0.095640

Table 1

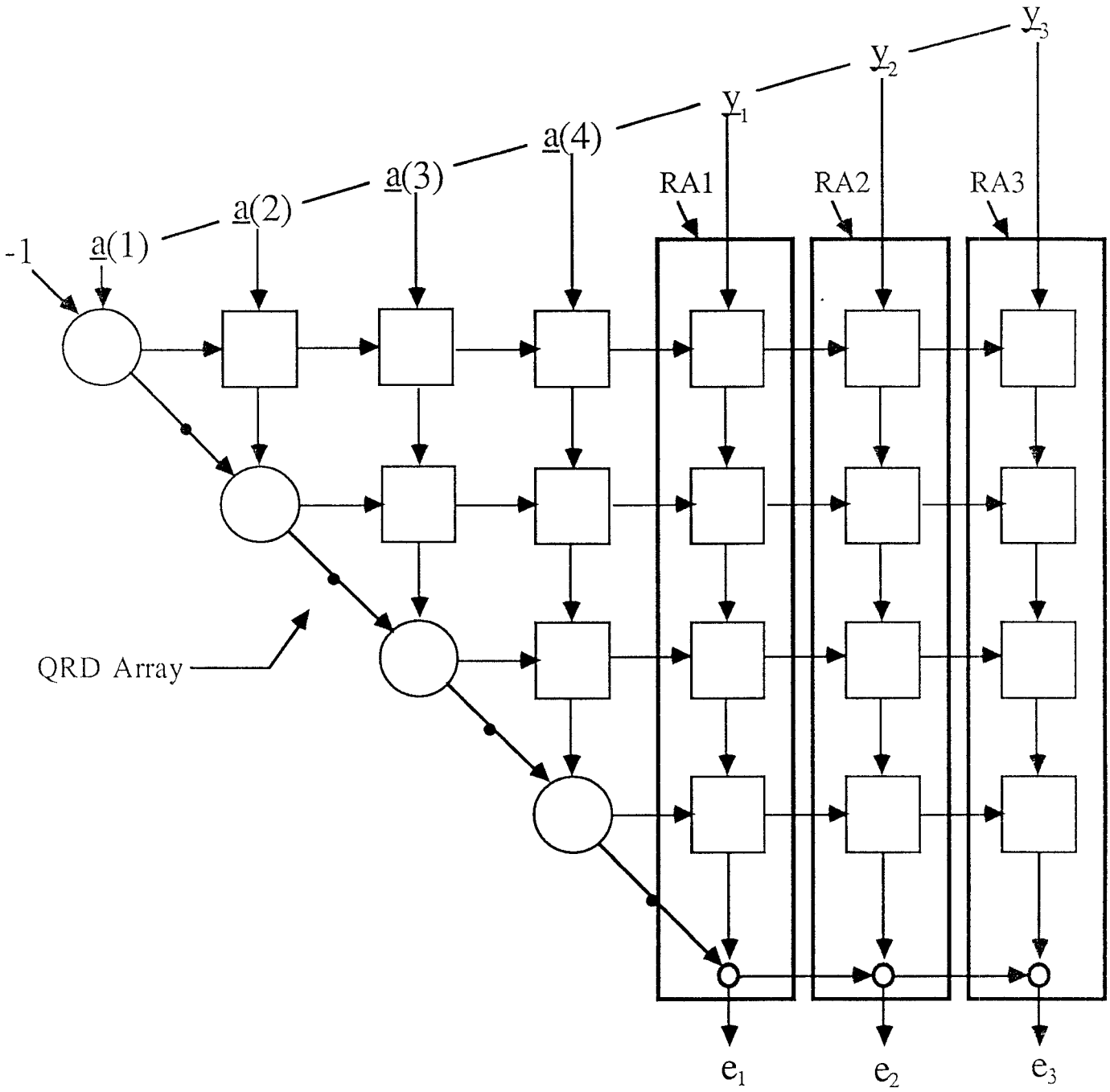
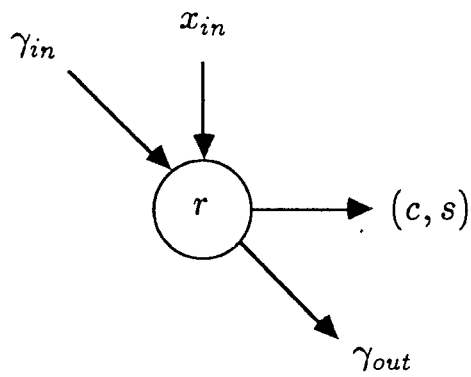


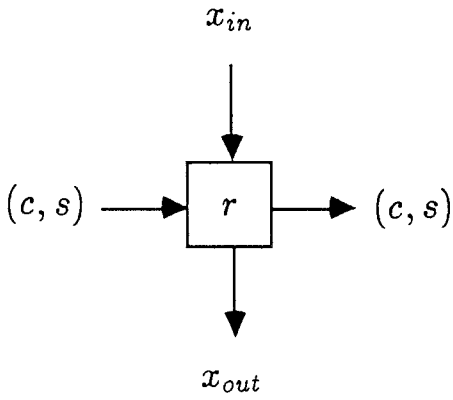
Fig. 1a

(1) Boundary Cell



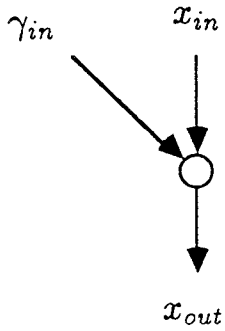
If $x_{in} = 0$ **then**
$c \leftarrow 1; \quad s \leftarrow 0; \quad \gamma_{out} \leftarrow \gamma_{in}$
otherwise
$r' = \sqrt{\lambda^2 r^2 + x_{in}^2};$
$c \leftarrow \lambda r / r'; \quad s \leftarrow x_{in} / r'$
$r \leftarrow r'; \quad \gamma_{out} = c\gamma_{in}$
end

(2) Internal Cell



$$x_{out} \leftarrow c x_{in} - s\lambda r$$
$$r \leftarrow s x_{in} + c\lambda r$$

(3) Final Cell



$$x_{out} \leftarrow \gamma_{in} x_{in}$$

Fig.1(b)

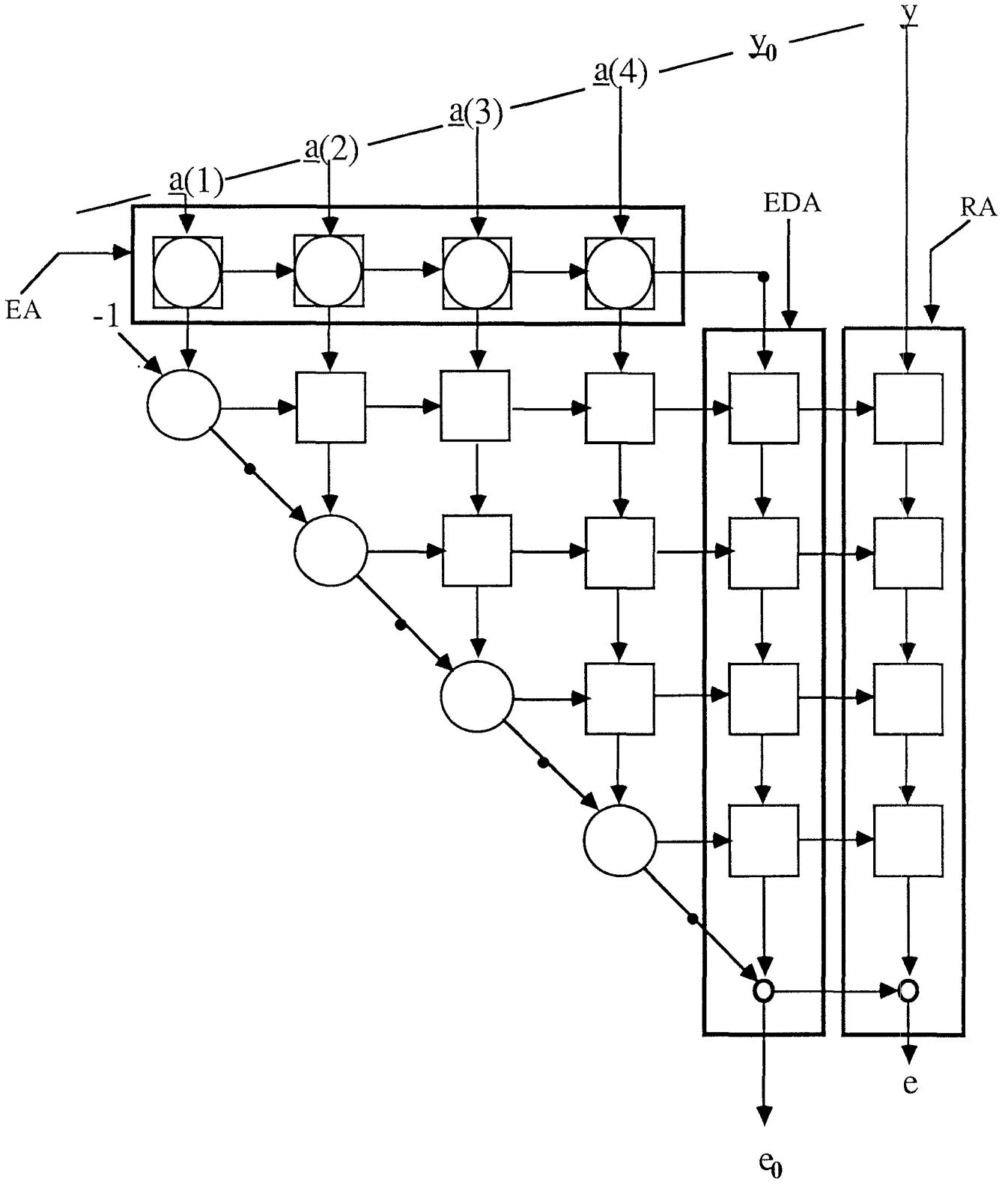
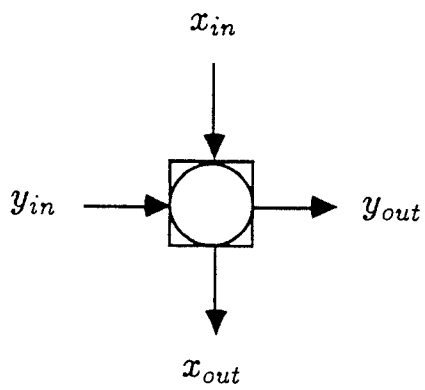


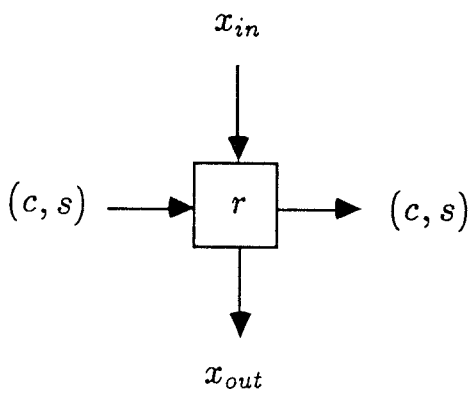
Fig. 2(a)

(1) Encoding Cell



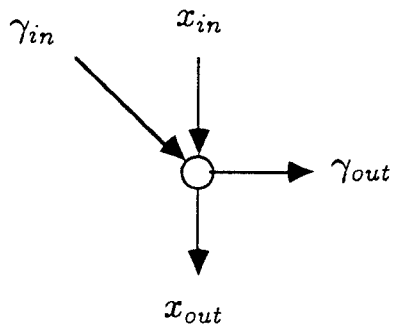
$$x_{out} \leftarrow x_{in}$$
$$y_{out} \leftarrow x_{in} + y_{in}$$

(2) Internal Cell



If $(c = 1$ and $s = 0)$ **then**
$\quad x_{out} \leftarrow x_{in}, \quad r \leftarrow \lambda r$
else
$\quad x_{out} \leftarrow c x_{in} - s\lambda r$
$\quad r \leftarrow s x_{in} + c\lambda r$
end if

(3) Final Cell



$$x_{out} \leftarrow \gamma_{in} x_{in}$$
$$\gamma_{out} \leftarrow \gamma_{in}$$

Fig.2 (b)

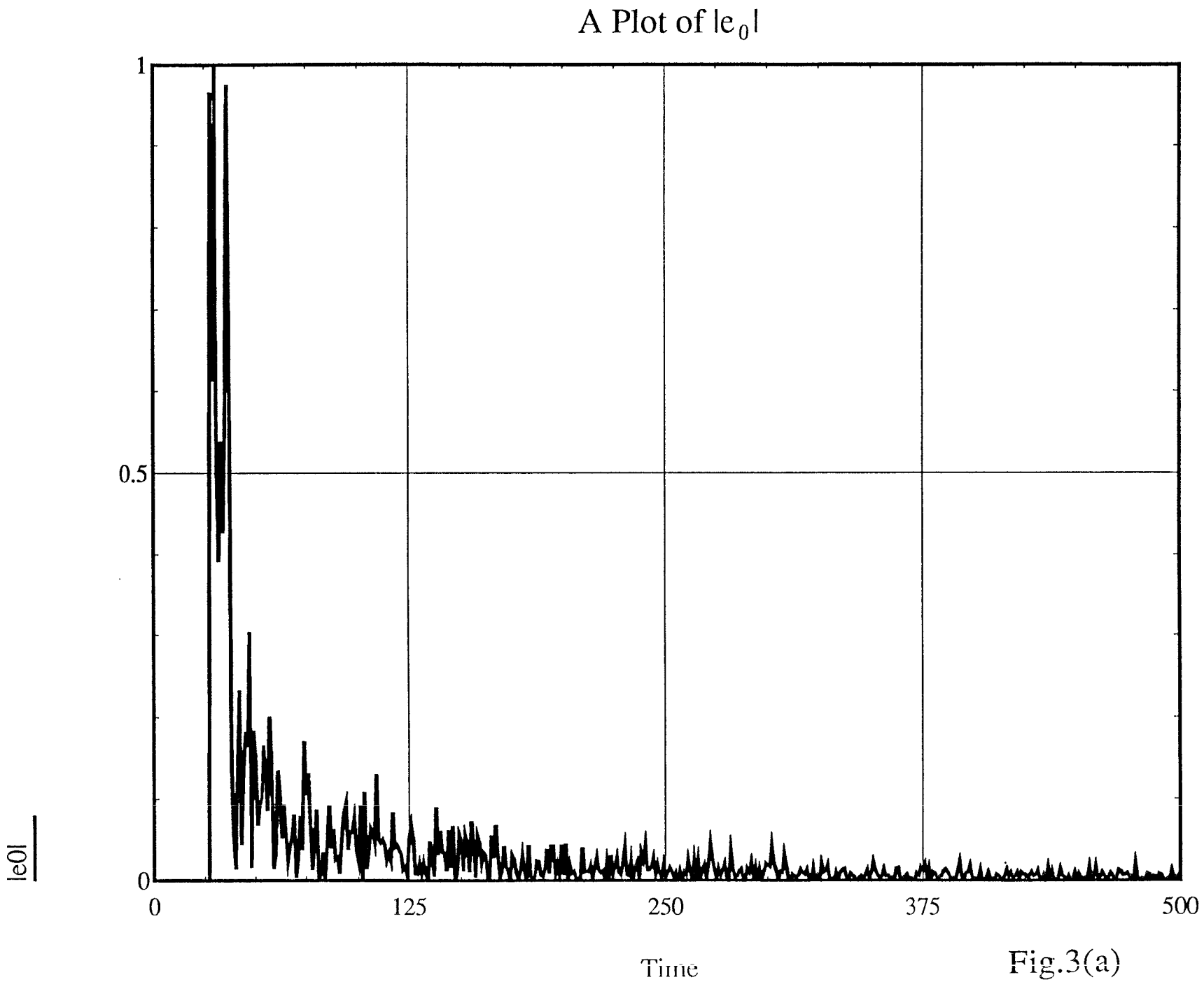


Fig.3(a)

A Plot of $|e_0|$ with threshold set at 0.3

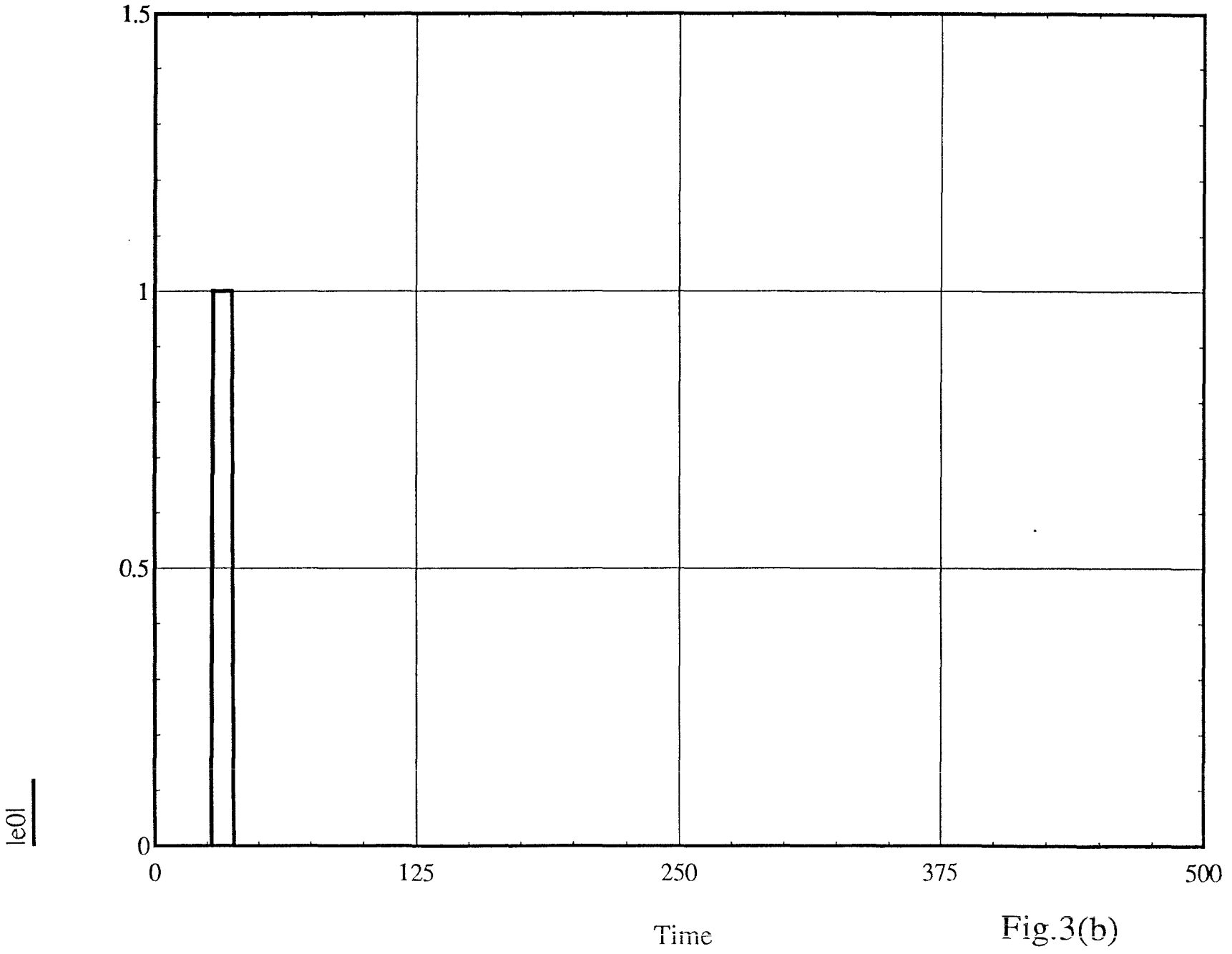


Fig.3(b)

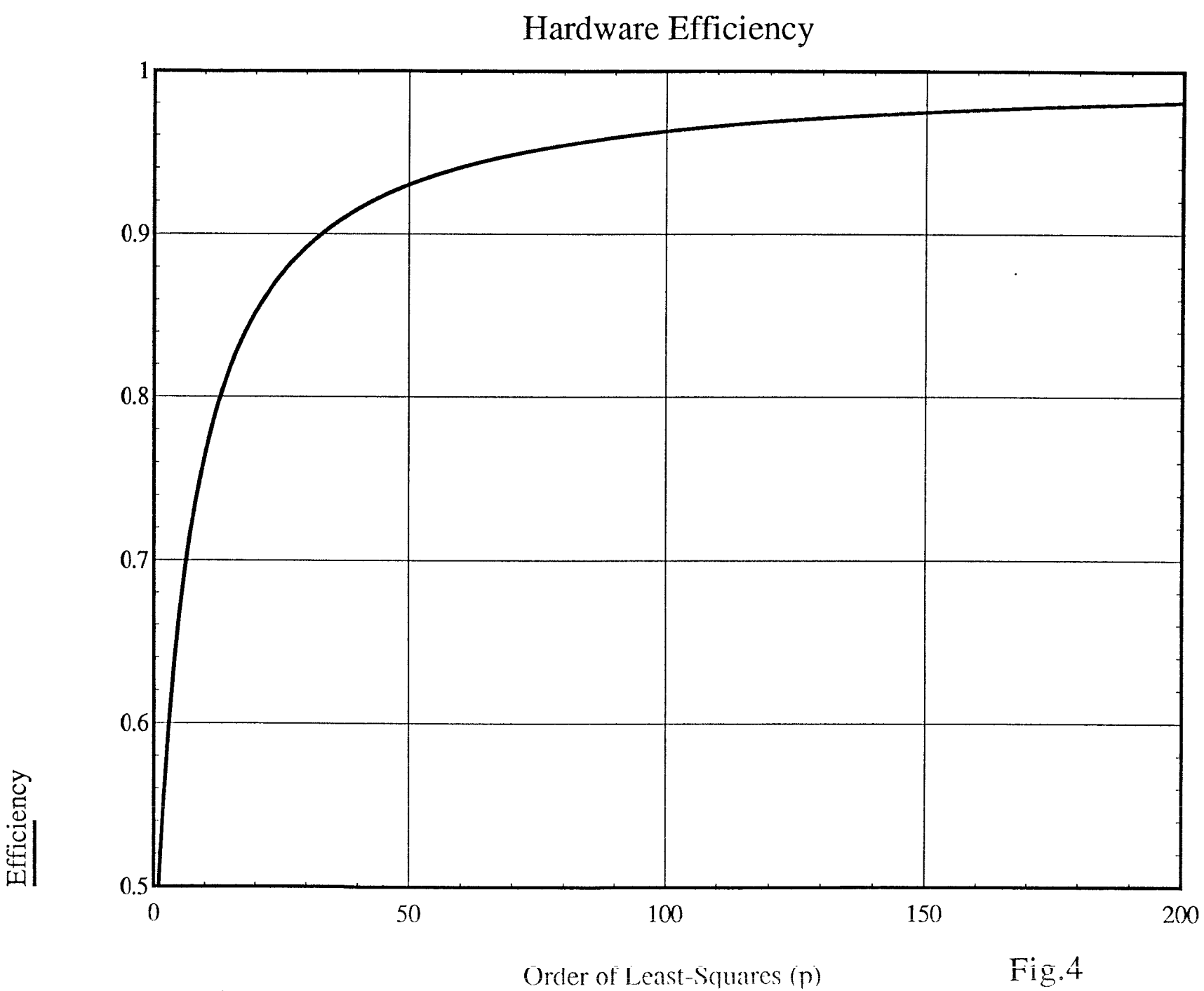

Hardware Efficiency
1
0.9
0.8
0.7
0.6
0.5
Efficiency
0
50
100
150
200
Order of Least-Squares (p)

Fig.4

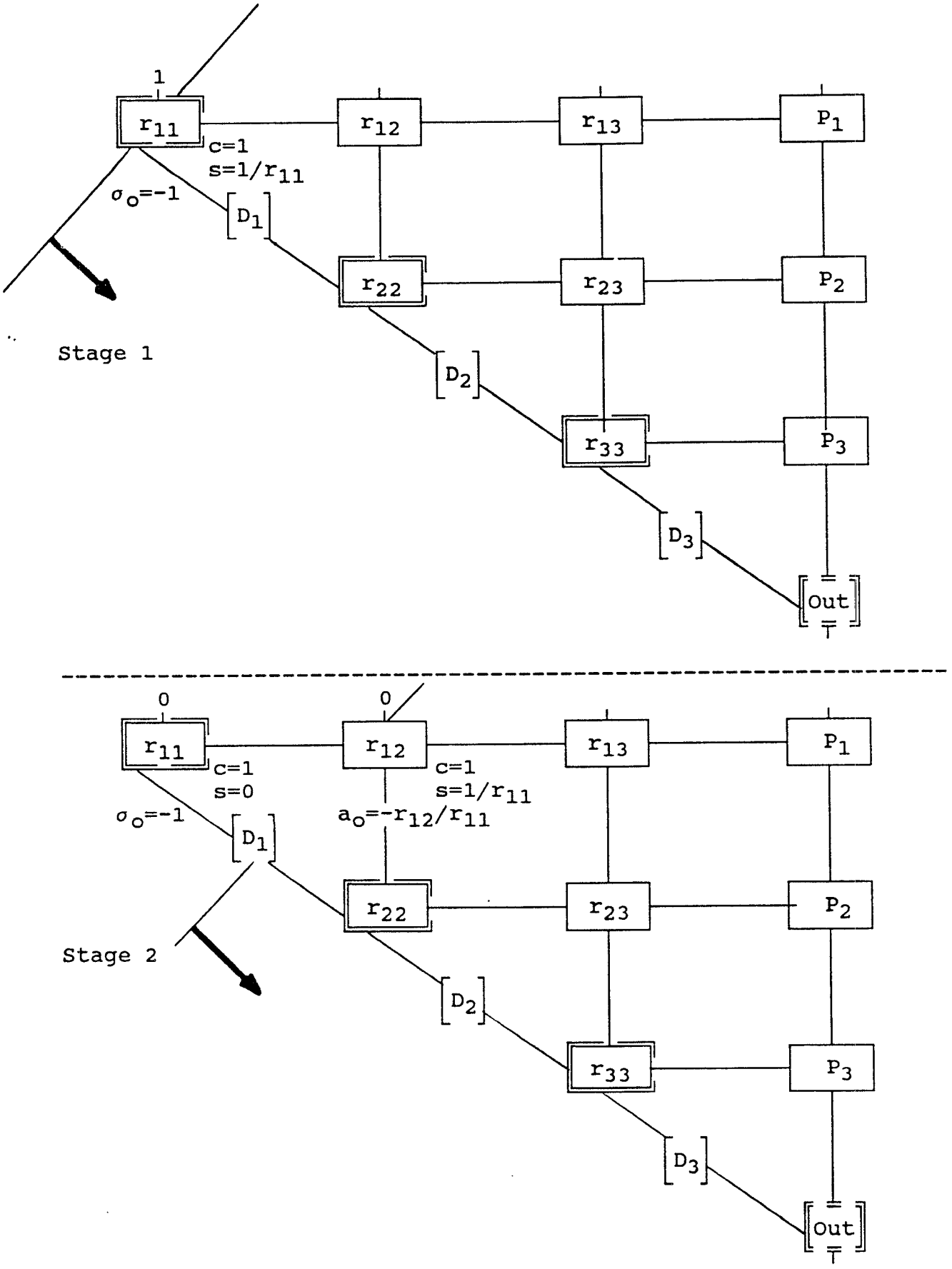
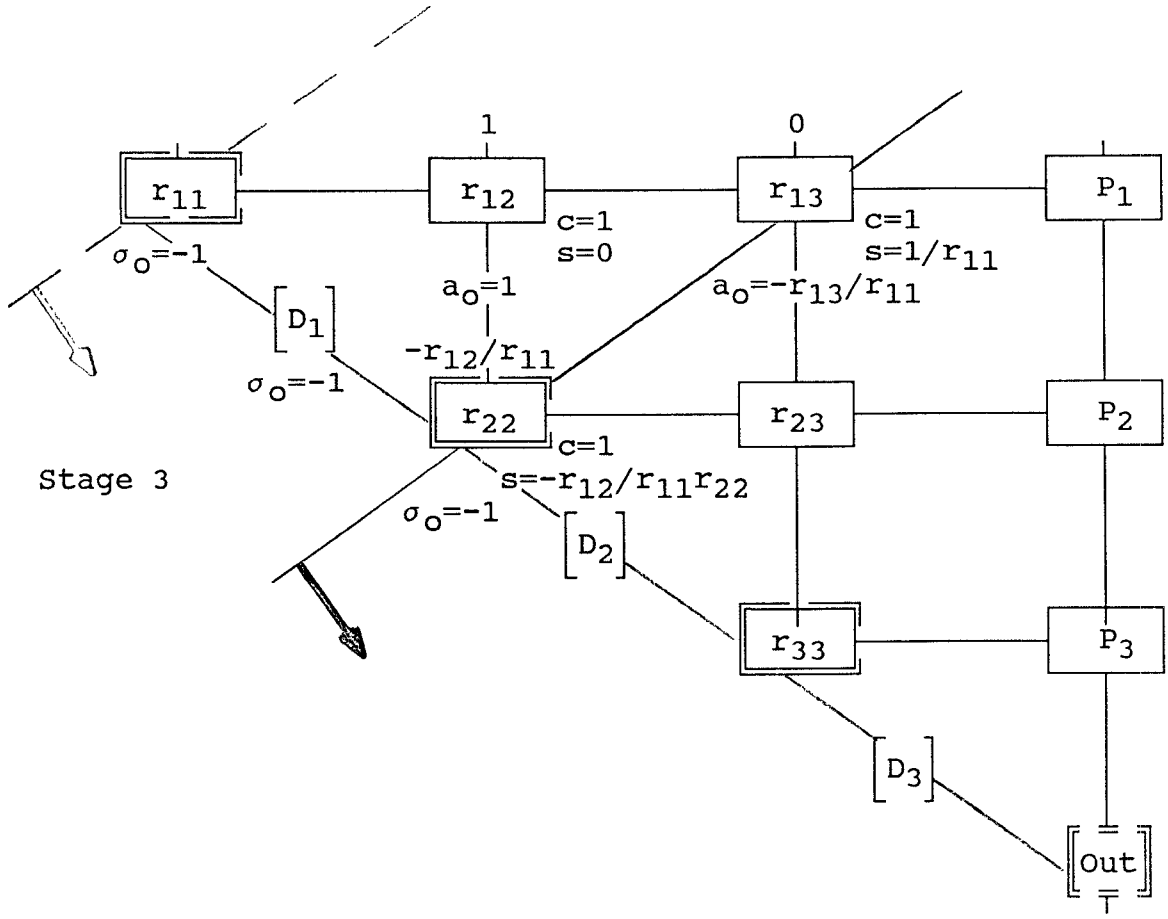
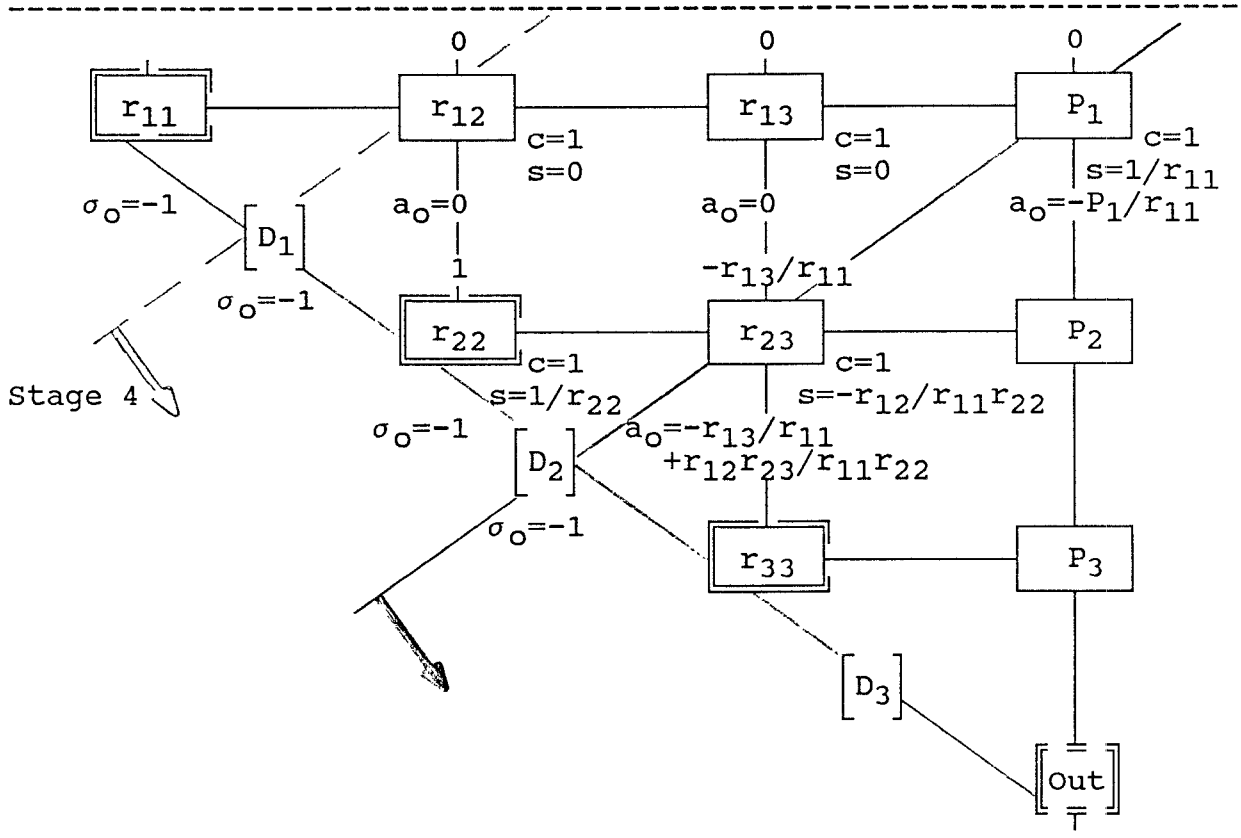

1
r11
c=1
s=1/r11
σo=-1
D1
r12
r13
P1
r22
r23
P2
Stage 1
D2
r33
P3
D3
Out
0
r11
c=1
s=0
σo=-1
D1
0
r12
c=1
s=1/r11
ao=-r12/r11
r13
P1
r22
r23
P2
Stage 2
D2
r33
P3
D3
Out

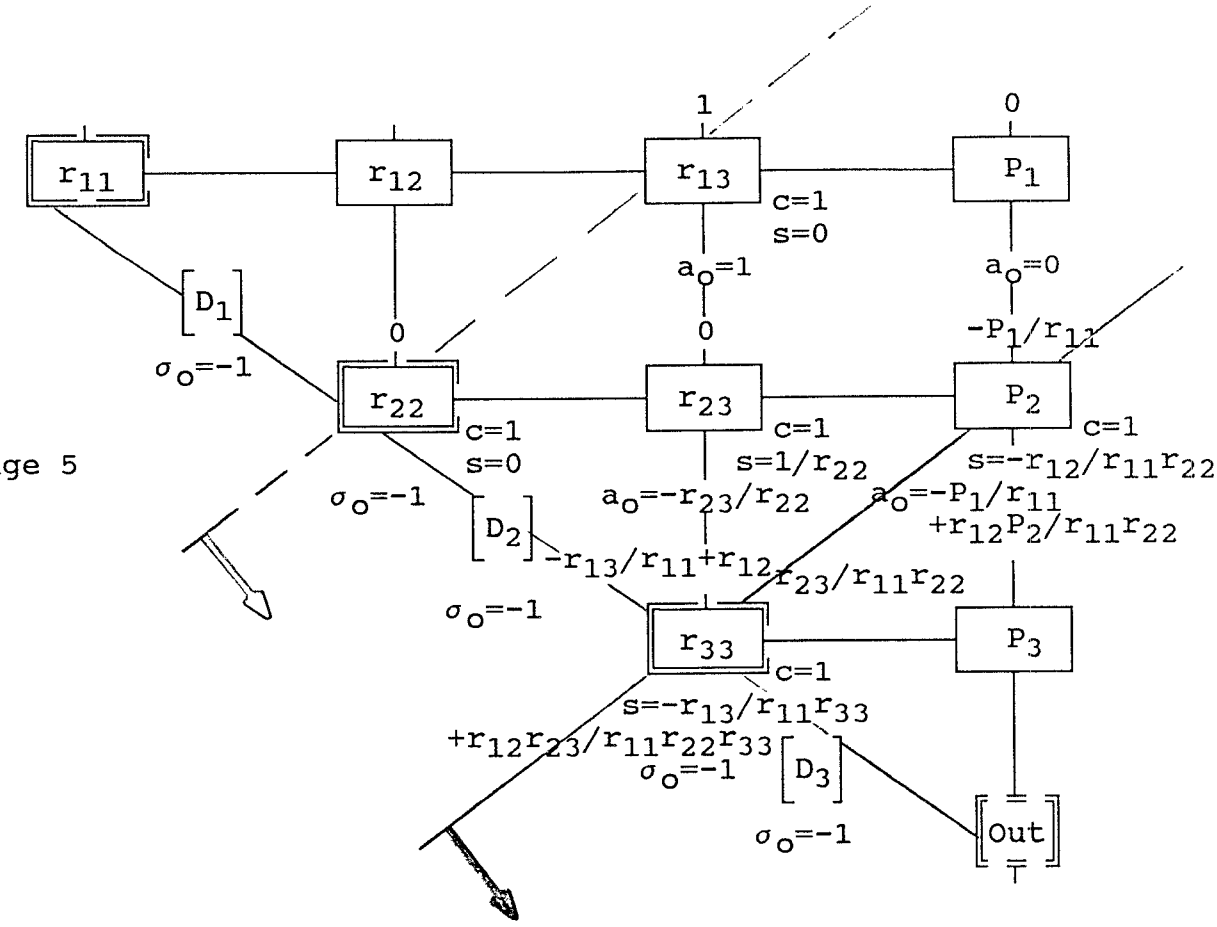
Fig.5


1
0
r11
r12
r13
P1
c=1
s=0
c=1
s=1/r11
σo=-1
ao=1
ao=-r13/r11
D1
-r12/r11
σo=-1
r22
r23
P2
c=1
Stage 3
s=-r12/r11r22
σo=-1
D2
r33
P3
D3
Out

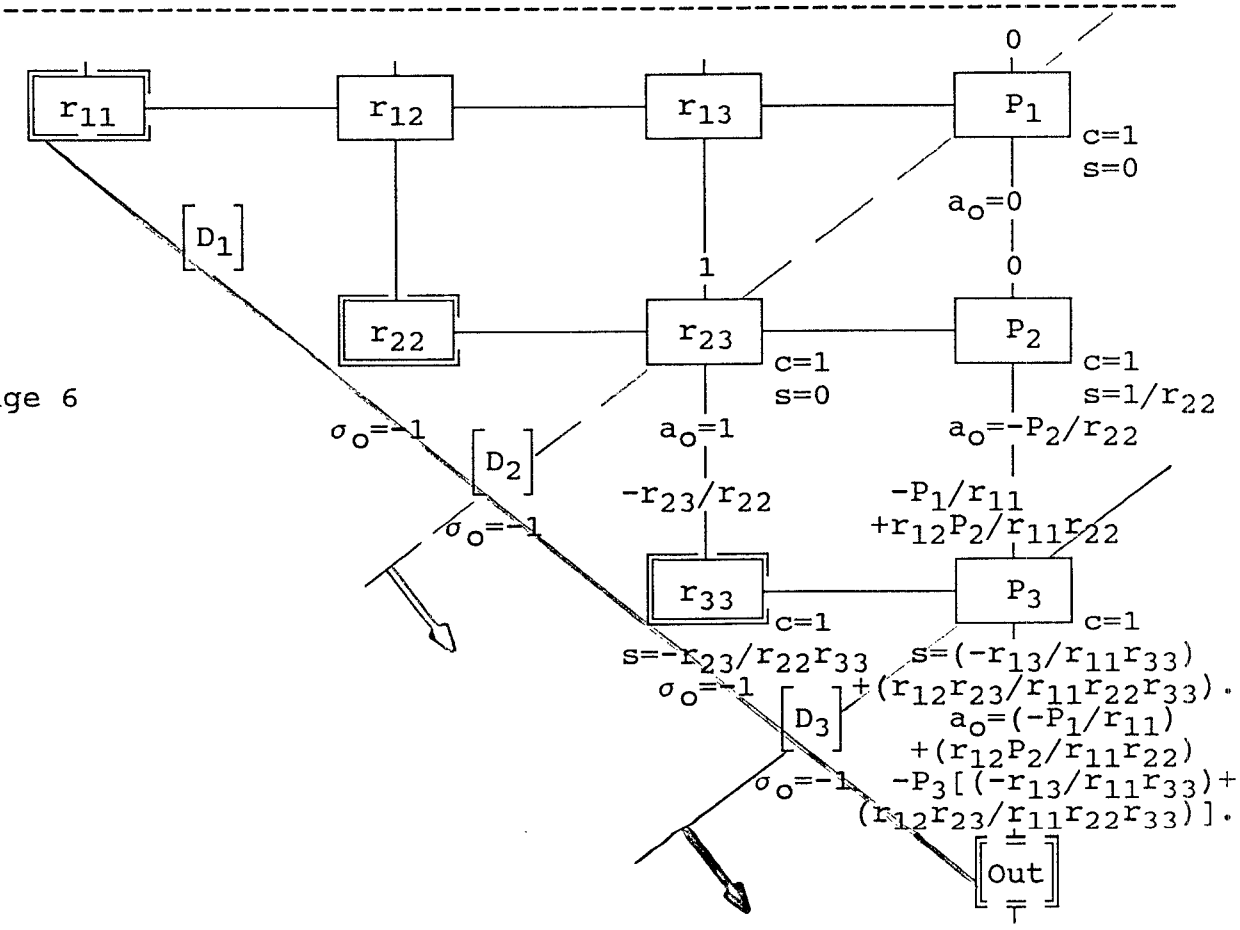

0
0
0
r11
r12
r13
P1
c=1
s=0
c=1
s=0
c=1
s=1/r11
σo=-1
ao=0
ao=0
ao=-P1/r11
D1
1
-r13/r11
σo=-1
r22
r23
P2
c=1
c=1
Stage 4
s=1/r22
s=-r12/r11r22
σo=-1
ao=-r13/r11
+r12r23/r11r22
D2
σo=-1
r33
P3
D3
Out

Cont'l.

Stage 5

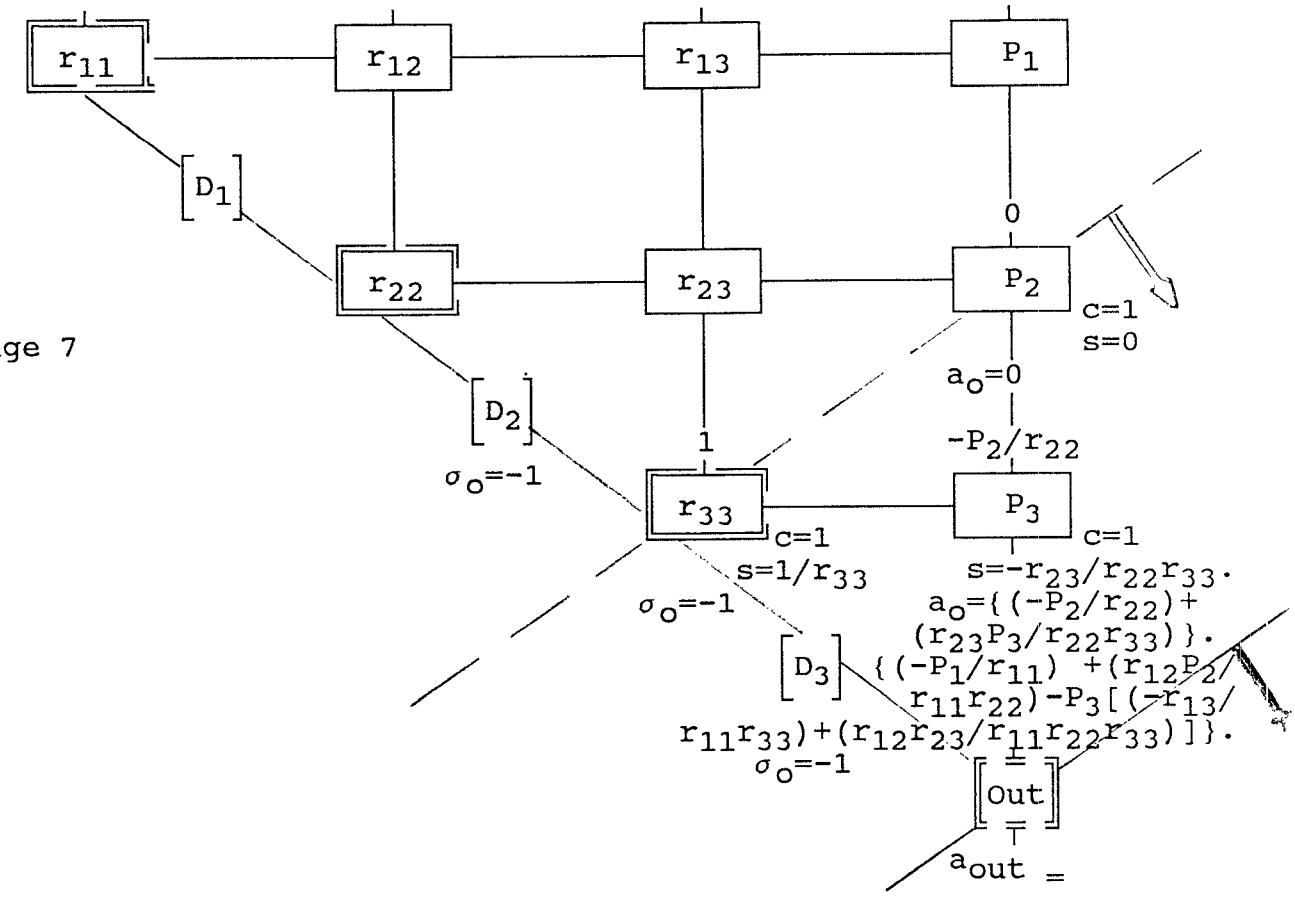


Stage 6



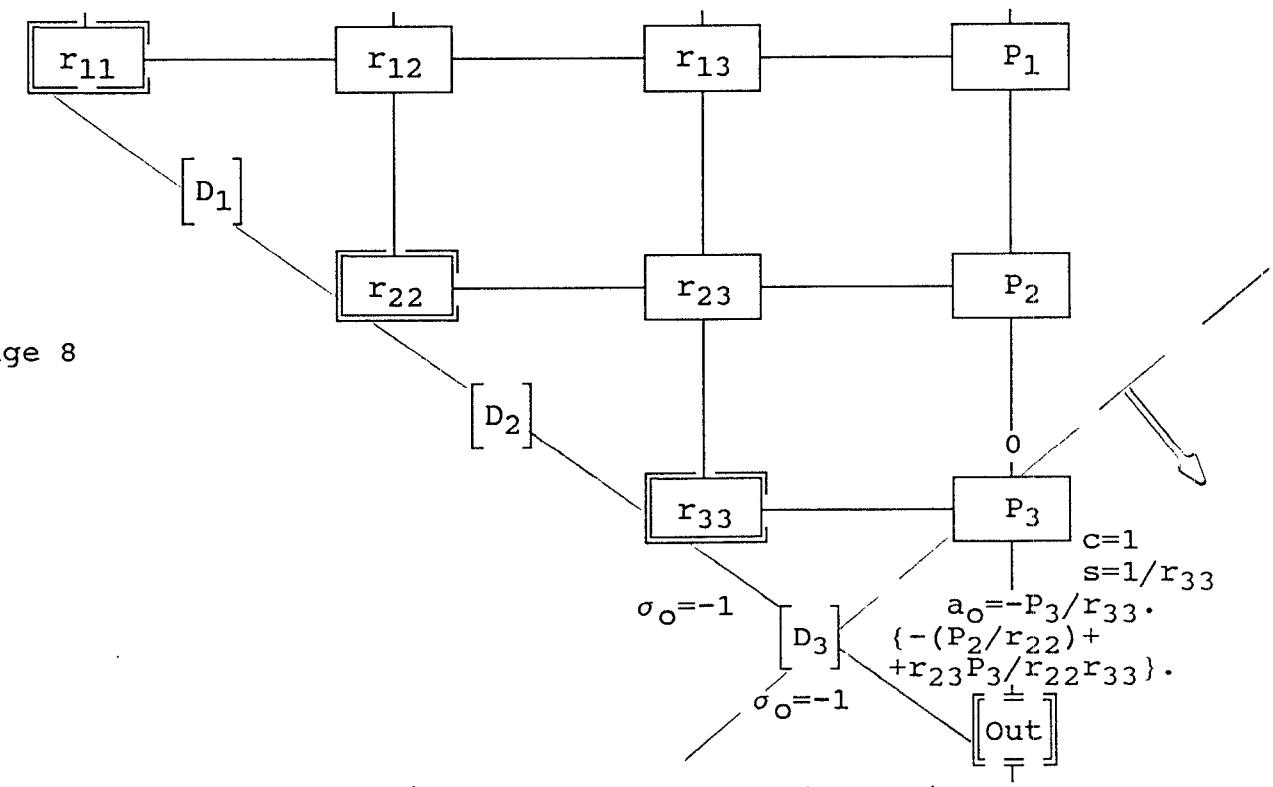
Cont'l

Stage 7



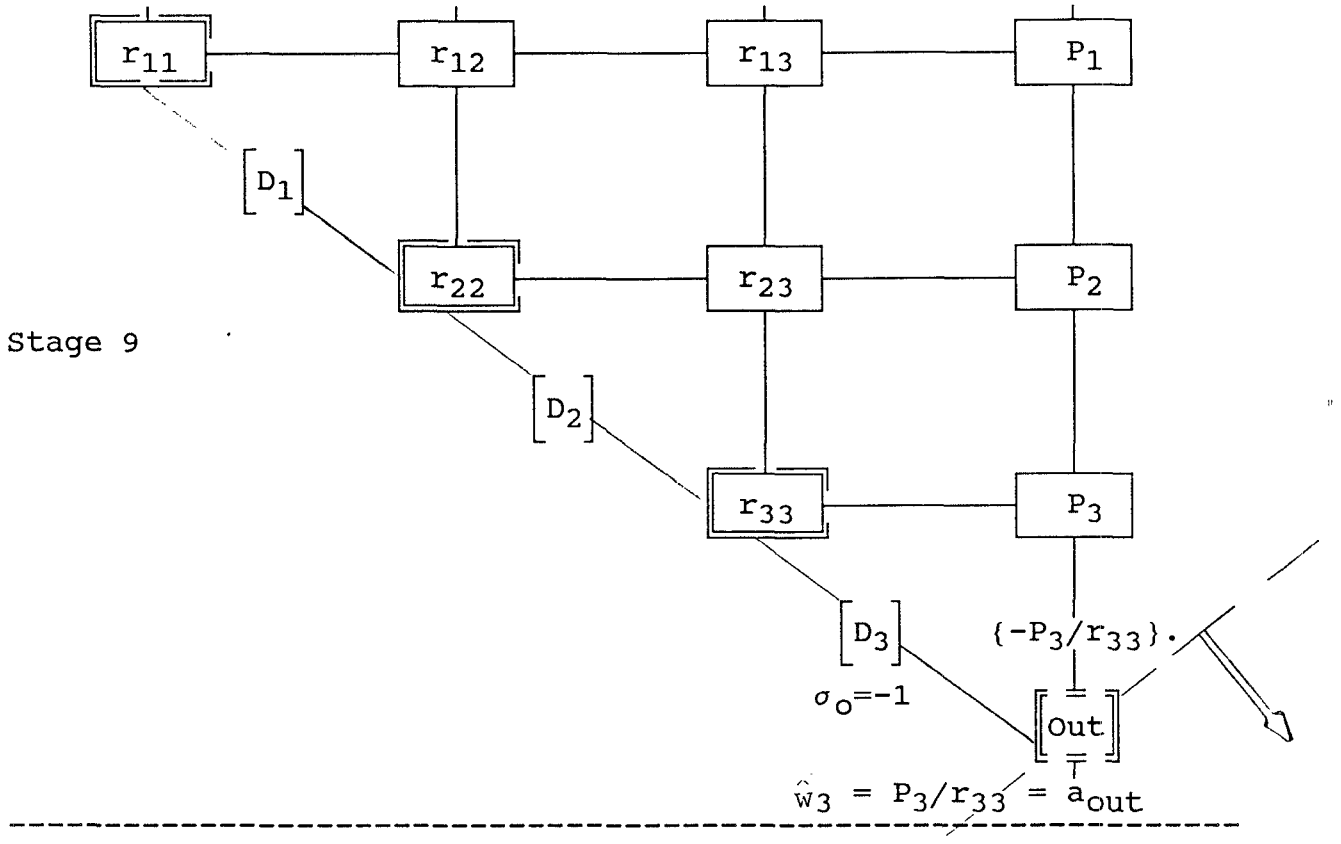
$$\hat{w}_1 = \{(P_1/r_{11})-(r_{12}P_2/r_{11}r_{22})+P_3[(-r_{13}/r_{11}r_{33})+(r_{12}r_{23}/r_{11}r_{22}r_{33})]\}$$

Stage 8



$$\hat{w}_2 = (P_2/r_{22})-(r_{23}P_3/r_{22}r_{33}) = a_{out}$$

Cont'd


r_{11}
r_{12}
r_{13}
P_1
$[D_1]$
r_{22}
r_{23}
P_2
Stage 9
$[D_2]$
r_{33}
P_3
$[D_3]$
$\{-P_3/r_{33}\}$.
$\sigma_o=-1$
[Out]
$\hat{w}_3 = P_3/r_{33} = a_{out}$

Cont'l

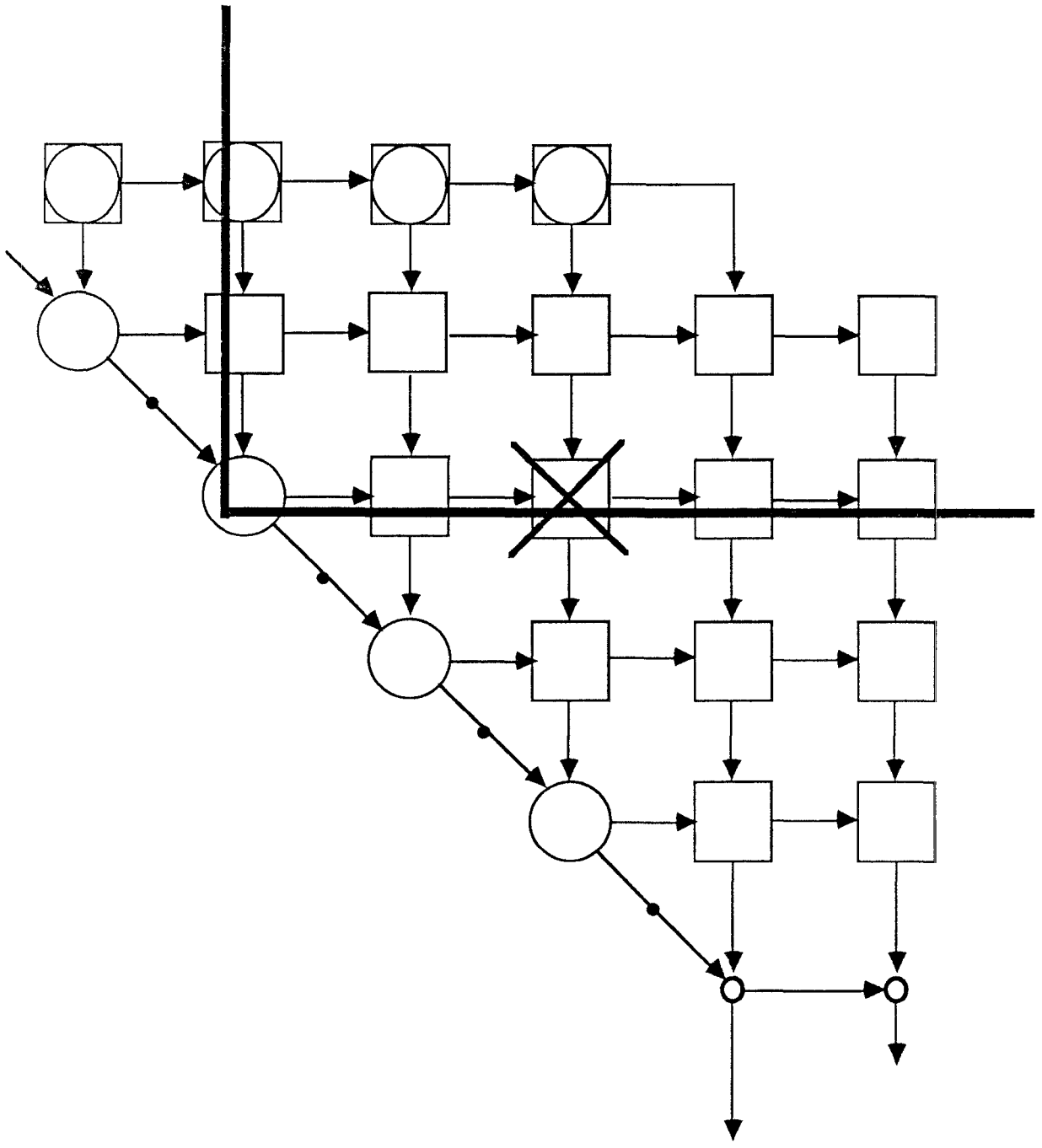


Fig.6

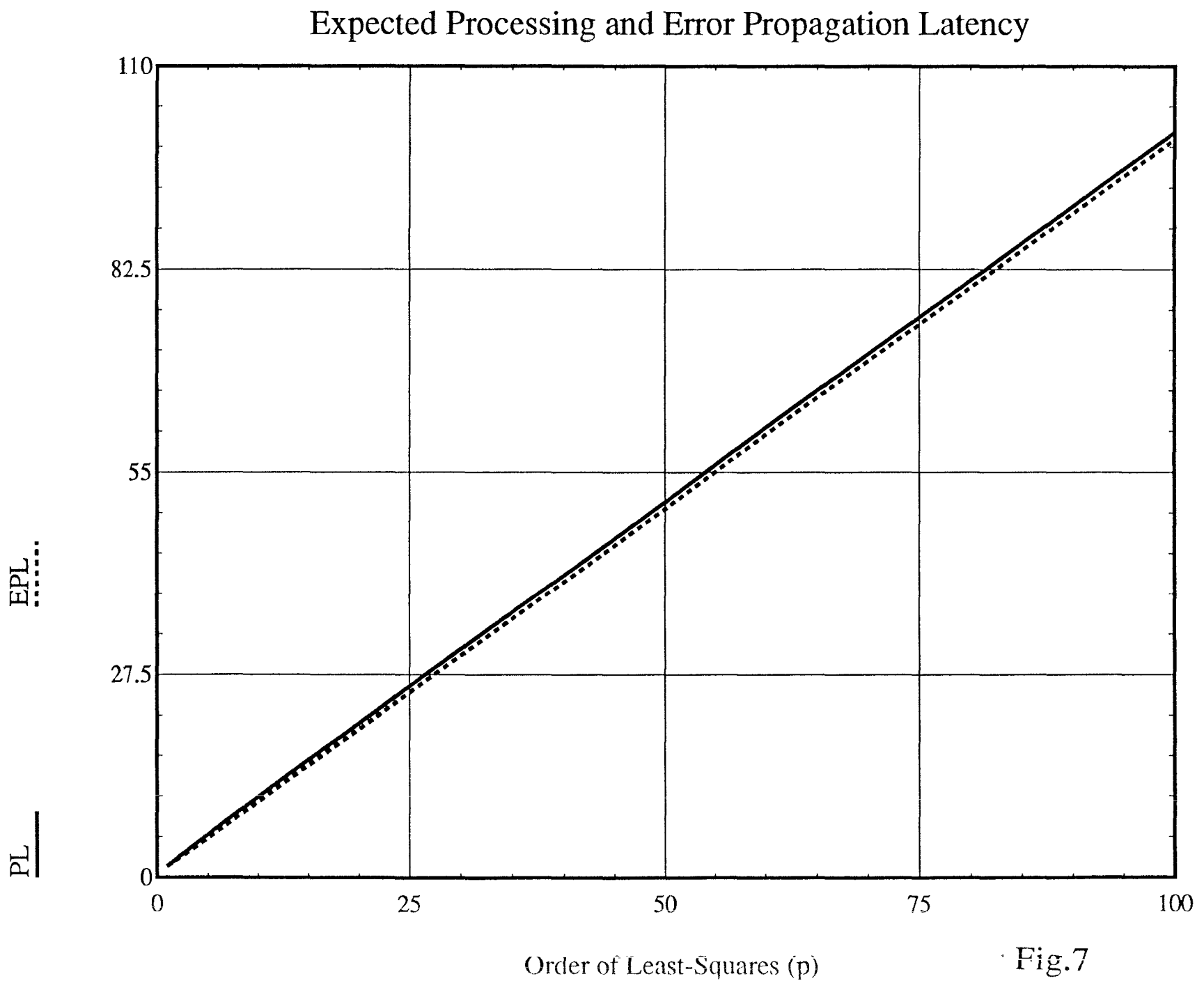

Expected Processing and Error Propagation Latency
110
82.5
55
27.5
0
0
25
50
75
100
EPL
PL
Order of Least-Squares (p)

Fig.7

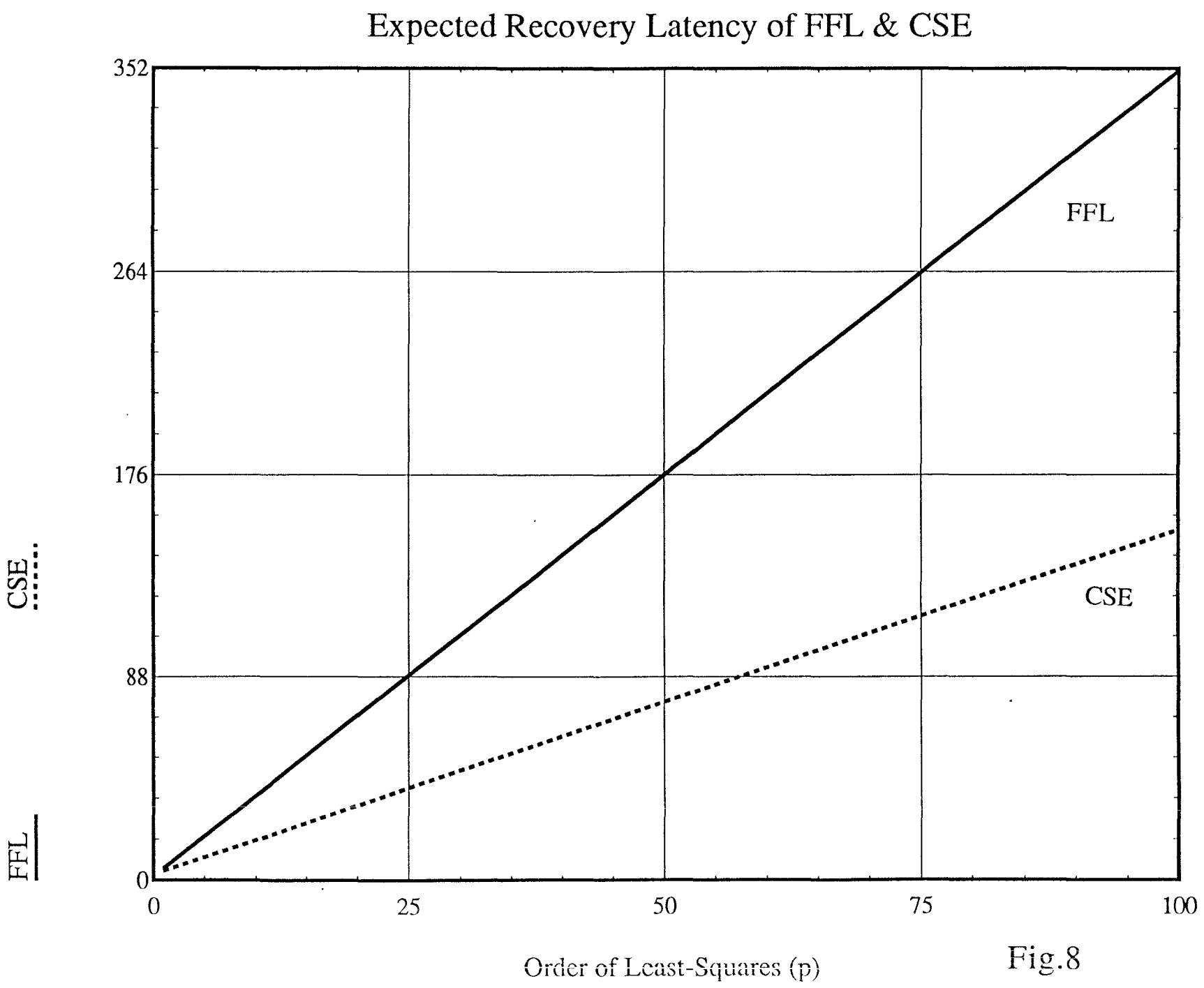

Expected Recovery Latency of FFL & CSE
352
264
176
88
0
0
25
50
75
100
FFL
CSE
CSE
FFL
Order of Least-Squares (p)

Fig.8