Proceedings

1989 IEEE INTERNATIONAL CONFERENCE ON
Computer Design: VLSI in Computers & Processors

Sponsored by IEEE Computer Society and IEEE Circuits and Systems Society
In cooperation with IEEE Electron Devices Society

ICCD '89

Hyatt Regency Cambridge, Cambridge, Massachusetts
October 2-4, 1989
Fault-Tolerant VLSI Processor Array for the SVD

Joseph R. Cavallaro
Dept. of Electrical & Computer Engineering
Rice University
Houston, TX 77251

Christopher D. Near
M. Ümit Uyar
AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

Dynamic reconfiguration techniques are presented for a two-dimensional systolic array for the SVD of a matrix. Extra computation time is not required, since idle time inherent in the array is exploited. This scheme does not require additional spare processors and is easily implemented in VLSI. Only minor hardware and communication time increases within each processing element are required.

1. Introduction

The Singular Value Decomposition (SVD) of a matrix is an important numerical algorithm used extensively in real-time signal processing and image processing applications. Special purpose parallel processor arrays for computing the SVD can significantly improve the performance of this computationally intensive algorithm. The systolic array architecture of Brent, Luk, and Van Loan [3] uses an expandable square array of processors to compute the SVD of a large matrix. Recently, the high-speed CORDIC arithmetic algorithms have been applied to the SVD [5] and a VLSI implementation of a processor array element is in progress at Rice University.

A major application of a CORDIC SVD array is space and underwater systems where the processor cannot be physically repaired. In this paper, a fault tolerant CORDIC SVD array which uses dynamic reconfigurability is presented. Faults are considered permanent rendering a processor and its associated communication links unusable.

The techniques reported in the literature for fault reconfiguration require additional processors to replace the faulty ones and can only handle a limited number of faults before the array becomes unusable [2, 10]. In the fault reconfiguration technique presented here, the computation load of the faulty processors is handled by its neighbors. This scheme requires no additional processors and handles \( O(p) \) faulty processors in a \( \lceil p/2 \rceil \times \lceil p/2 \rceil \) array operating on a \( p \times p \) matrix. Utilizing idle time inherent in the SVD algorithm, the fault reconfiguration scheme requires no additional computation time. However, the extra communication time of an array with faulty processors leads to 5% more total time than the fault-free case. Although the processor hardware is increased by 5%, no additional communication links are needed. Dynamic reconfiguration is a difficult problem and the subject of extensive research. This work is an attempt to provide an algorithm for the SVD array.

2. The Singular Value Decomposition Array

The singular value decomposition of an \( p \times p \) matrix \( M \) is

\[
M = U \Sigma V^T,
\]

where \( U \) and \( V \) are orthogonal matrices and \( \Sigma \) is a diagonal matrix of singular values. In the Brent, Luk, Van Loan array, the matrix is divided into \( 2 \times 2 \) submatrices. Each processor element contains a \( 2 \times 2 \) submatrix. The array architecture is scalable. As an example, Figure 1 shows a small array to solve a \( 6 \times 6 \) problem. There are

![Figure 1. Data and Angle Flow of the Brent-Luk-Van Loan SVD method.](image-url)
two types of data flowing in this array. Rotation angles
generated by the diagonal processors flow systolically
along the rows and columns of the array. Matrix data ele-
ments are exchanged diagonally, after the diagonal neigh-
bor has received and applied the necessary rotation angles.
This leads to “waves” of activity moving diagonally away
from the main array diagonal. Each wave is separated in
time from the next by two time periods. In this paper, this
idle time will be exploited to allow a neighboring processor
to carry the computational load of a faulty processor.

In the CORDIC SVD array, the processor is active once
in every three major cycles. The idle second and third
major cycles are necessary for the systolic propagation of
the angle data and the associated rotations. Within each
major cycle, there are three CORDIC minor cycles. At an
even lower level, there are n “micro” cycles, which
correspond to the shifts and additions that compose a
CORDIC iteration. Data transmission between processors
will contain n-bit values. The assumption is made that
one “micro” cycle, (the time for an n-bit addition), will be
more than sufficient to transmit data to an immediate
neighbor.

3. Dynamic Fault Reconfiguration for the SVD

In our fault model, only processor and/or link faults
which render a processor and its associated links unusable
are considered. Various fault detection schemes may be
employed to locate and isolate the faults, including self-
implicating structures [7], voting schemes based on reduct-
dancy [9], algorithmic checksum techniques [1], and spe-
cial SVD fault detection schemes [6]. Detection of the
faults is not addressed in this paper.

Once a fault has been detected, the faulty processor(s)
and/or links are isolated and the array is dynamically
reconfigured. A traditional approach to dynamic recon-
figuration has been to view the mesh as a graph and then
perform a cut to remove the entire row, column, or diagno-
mal with the faulty processor. The remaining processors
are rejoin and processing resumes after the algorithm
has been modified to deal with the new dimensions of the
mesh. This scheme has been used in the Massively Paral-
lel Processor (MPP) [8] and has been discussed by Kung
and Lam [10]. These redundancy-based fault tolerance
techniques are also reviewed by Negrini, Sami, and
Stefanelli [11]. However, only a few errors can be
tolerated before the mesh is reduced to an unusable size.

In this paper, a more versatile fault reconfiguration
technique which removes just the faulty processor and
reassigns the faulty processor’s data to the fault-free
neighbors by a uniform data redistribution [13] is intro-
duced. Reconfiguration through uniform data redistribu-
tion has been developed for a variety of algorithms and
interconnection networks [8, 12]. In this scheme, the
faulty processor’s computations are performed by neighbor-
ing processors and interprocessor communication is then
routed around the location of the faulty processor. For
example, if a diagonal CORDIC SVD processor is faulty, its
data will be assigned to its neighbor for computation.

4. Reconfiguration Algorithm

Each processor in the array can operate in either the
fault-free, (FF), or fault-tolerant, (FT), mode. Each proces-
sor contains a register which holds its horizontal and verti-
cal location within the array. When a fault has been
detected, the address of the faulty processor is sent to
every processor in the array. Each processor compares the
faulty processor location with its own location.

If it is a near-neighbor to a faulty processor, then it
modifies its communication pattern to talk to different
neighbors around the faulty processor. In addition, if it is
the neighbor which is to perform the additional computa-
tion, then it uses one of its idle cycles to operate on the
faulty processor’s data.

If the processor is not a near-neighbor to a faulty pro-
cessor, then it performs essentially the same operations as
in the fault-free case. However, the processor will insert
delays in communication to preserve total array synchroni-
zation.

In the event of a fault, a pre-determined neighbor is
assigned to process the 2 x 2 matrix of data points of the
faulty processor. The neighbor is chosen in order to
minimize the communication time, since it is close to the
source of the angles passed between the processors. An
alternative path is established around the faulty processor
to preserve the systolic transmission of the rotation angles
and the interchange of matrix data elements, since the
communication links to the faulty processor are also
assumed to be faulty.

Figure 2 shows the communication rerouting for a
faulty super-diagonal processor. In this example, the pro-
cessor to the west of the faulty processor will compute the
application of the rotations. The rotation angles are
routed to it and then further routed around the faulty pro-
cessor to preserve the systolic data flow as in Figure 2a.
The data routing necessary to preserve the interchange of
the matrix data elements is shown in Figure 2b. The
results produced by the processor to the west of the faulty
processor are routed to the diagonal neighbors of the faulty
processor.

Figure 2a. Angle Routing Around a Super-Diagonal Fault.
The actual sequence of events for super-diagonal processors in the CORDIC SVD array is illustrated in Table 1. Timing for sub-diagonal processors is similar; however, the direction of angle and data flow is reversed. Communication and computation events are shown for each processor. Each column of the table refers to a diagonal band of processors as shown in Figures 2a and 2b. There are two critical timing events: the rotation angle transfer and the matrix data transfer. The rotation angle transfer will control the synchronization of the systolic array since computation in any processor element occurs directly after the receipt of the rotation angles. The rotation angles are generated in the diagonal processors of the array. These angles propagate horizontally and vertically and require a transmission, TD, and reception, TR, time.

The second critical timing event in the systolic array is the transmission, TD, and reception, TR, of matrix data elements. This data is exchanged diagonally between processors. For example, a processor in Diagonal 3 requires four data elements. The last data element to arrive is from the Northeast neighbor on Diagonal 5. The processor on Diagonal 3 then applies the rotation angles and computes new matrix data elements. The results are passed to the diagonal neighbors and in particular to the Southwest neighbor on Diagonal 1.

In developing a fault reconfiguration strategy, it is important that the backup processor is near the source of angles and data, and that it is idle during the cycle. The processor which is shown as faulty in Figures 2a and 2b is on Diagonal 3. In Table 1, the west processor in Diagonal 2 will perform the computation during the idle period marked "FT". Two extra data transfer steps (not shown in Table 1) will also be needed to complete the data communication.

The CORDIC SVD processor array is toroidally connected in order to allow greater reconfiguration flexibility. Since the array contains \([p/2] \times [p/2]\) processors, the following arithmetic is modulo \([p/2]\). The following scheme is used to determine the backup processor in the event of a single fault:

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timing Events in a CORDIC SVD Processor Array</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
</tr>
<tr>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
</tr>
<tr>
<td>T0</td>
<td>T0</td>
<td>T0</td>
<td>T0</td>
<td>T0</td>
</tr>
<tr>
<td>TD</td>
<td>TD</td>
<td>TD</td>
<td>TD</td>
<td>TD</td>
</tr>
<tr>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
<td>TRD</td>
</tr>
<tr>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
<td>TRB</td>
</tr>
<tr>
<td>T0</td>
<td>T0</td>
<td>T0</td>
<td>T0</td>
<td>T0</td>
</tr>
<tr>
<td>TD</td>
<td>TD</td>
<td>TD</td>
<td>TD</td>
<td>TD</td>
</tr>
</tbody>
</table>

TRR: Receive rotation angles  
TRB: Transmit rotation angles  
TRD: Receive matrix data  
TRD: Transmit matrix data  
T0: Compute rotation angles (CORDIC cycle)  
T0: Compute new matrix data (2 CORDIC cycles)  
"FT": Processor acts as Backup Processor for D3
Diagonal Processors:
For all $P_{i\pm}$: Backup is West neighbor, $P_{i,i-1}$.
Super-Diagonal Processors:
For $P_{ij}$ where $j > i$: Backup is West neighbor, $P_{i,j-1}$.
Sub-Diagonal Processors:
For $P_{ij}$ where $i > j$: Backup is North neighbor, $P_{1,i-j}$.

The case of more than one disjoint fault (i.e., faulty processors that are not neighbors) is also handled in this manner. This scheme can handle many disjoint faults within the array, since hardware is not consumed in the reconfiguration. No conflicts will occur near the main diagonal of the array, since faults are assumed to be disjoint. However, adjacent faulty processors may require a more difficult fault reconfiguration scheme.

5. Fault Tolerant Architectural and Processor Requirements

In order to bypass the faulty processors, the 8-connected mesh architecture required by the Brent-Luk-Van Loan SVD algorithm will be exploited. Therefore, there are no additional communication links required for the dynamic fault reconfiguration scheme.

5.1. Communication Time Costs

The fault free CORDIC SVD processor time, $T_{FF, SVD}$, contains two major components, communication, $T_{COMM}$, and computation, $T_{COMP}$, time. Analysis of the computation time [5] has shown that $T_{COMP} = 3.25 TC$ where $TC = n(T_{ADD})$ for an $n$-bit implementation and $n$-bit addition. Since idle time within the array is exploited in the fault reconfiguration scheme, the computation time does not increase. However, extra communication steps are needed because of the re routing of data on the critical timing path. The extra communication time will include two extra data transfers, $T_{DT}$. Each of the data transfer steps shown in Table 1, $T_{RB}$, $T_{RT}$, $T_{HL}$, and $T_{DH}$ are assumed to be equal to $T_{DT}$. In the fault free case, $T_{COMM} = 5T_{DT}$. Based upon the performance of 2$\mu$m CMOS I/O pads, $T_{DT} = T_{ADD}$ and therefore $T_{COMM} < T_{COMP}$. The total fault free time is:

$$T_{FF, SVD} = T_{COMP} + T_{COMM} = 3.25 TC + 5T_{DT}.$$ 

The total time for a $2 \times 2$ fault-tolerant CORDIC SVD, $T_{FT, SVD}$, will be:

$$T_{FT, SVD} = T_{FF, SVD} + 2T_{DT}.$$ 

For a typical value of $n$, $n = 32$, the increase in time is less than 5%.

5.2. Hardware Costs

In order for the processors to handle the extra data of a faulty near neighbor, additional hardware is required. Each processor will require 6 extra registers, 4 for data and 2 for angles. Also, a multiplexer will be used to handle the two different data sets. The fault free area, $A_{FF, SVD}$, is:

$$A_{FF, SVD} = 2AC + A_{CONT} + A_{COMP},$$

where $A_{COMP}$ is the area of the SVD control programmable logic array (PLA), and the area of a CORDIC processor is $A_{C} = A_{CPLA} + A_{ROM} + 2A_{SH} + 3A_{ADD} + 3A_{REG}$. The various areas correspond to the internal CORDIC PLA, ROM, barrel shifter, adder, and registers, respectively.

The extra hardware units, shown in Figure 3, will include the 6 registers in the CORDIC processors, labelled $FT$, plus a bus communication controller, $A_{FT, BC}$, for the fault-tolerant communication protocol. This controller will be implemented as a PLA. The area of a fault tolerant CORDIC SVD processor, $A_{FT, SVD}$ is:

$$A_{FT, SVD} = A_{FF, SVD} + 6A_{REG} + A_{FT, BC}.$$ 

Considering the complexity of the adders and barrel shifters required by a non-reconfigurable SVD processor (4), the extra hardware for fault reconfiguration is less than 5%.

6. Summary

An efficient scheme for dynamic fault reconfiguration for the CORDIC SVD processor array is presented. This scheme does not require either additional processors or communication links. The procedure utilizes the idle time

![Figure 3. Block Diagram of Fault Tolerant CORDIC SVD processor array element.](image-url)
inherent in the CORDIC SVD algorithm to achieve fault tolerance with only 5% additional time. Each processor will have an increase in hardware complexity of 5%. The design of this fault reconfiguration method preserves the array regularity necessary for VLSI implementation. A VLSI prototype implementation of this scheme is in progress at Rice University.

References