for Real time DSP Applications

Ahmad Falih Mahmood

Lecturer

Dept. of Computer Eng. / Technical college

Ahmadalallaf@yahoo.com

Abstract

This paper presents a new, expandable, pipelined linear array architecture designed for transparently tolerating processor failures for real-time DSP applications. The proposed system use twelve TMS320C40 DSP processors (Processor Modules PMs) to construct ten stages pipelined system with two spare processors (SPs). However, the system can be expanded to increase the pipeline stages and the performance, and adding more spare processors to increase the dependability and reliability of the system. In Proposed scheme, the system can automatically reconfigure itself in the event of failure in one or two of its DSP processors and the computations continue unhindered without noticeable performance degradation. Each DSP processor communicates with neighboring processors through a high speed communication ports (commport). Some of these commports in every processor are used as a bypass links in case of failure of one or two processors. The system uses the forward-task-shift (FTS) mechanism to tolerate the fault by assigning the function of the failed processor to the next fault-free processor.

Keywords- Linear processor array, fault tolerant, bypass links, pipelining, TMS320C40, DSP processors.

معمارية الخط الإنتاجي متسامحة الأخطاء لتطبيقات معالجة الإشارة الرقمية

احمد فالح محمود - مدرس-

الكلية التقنية – الموصل

الخلاصة

يُبين هذا البحث معمارية مصفوفة خطيّةِ لمعالجات الإشارة الرقمية، جديدة وقابلة للتوسيع صمّمتْ لتَتَحَمُّل حا
لزيَادَة مراحلِ الخط الإنتاجي وتحسين الأداءَ وكذلكزيادة المعالجات الاحتياطيةً لزيَادَة الاعتمادية و المَوثُوقِيِّةِ النظام
مُلجَاوَرَة عن طريق موانئ إتصال ذات سرعةِ قسم من هذه الموانئ تستخدم كـ وصلات عبور
بين المعالجات في حالة حدوث عطل في واحد واثنان من معالجات الظلم. يَستعمل النظام آلية
لتجاوز الأعطال سميت بالية إزاحة المهمة للإمام وذلك بتَخصيص مهمة المعالج ألعطال إلى
مُلمعالج خالي مِنْ ألعطل التالي.

Received 14 Feb. 2007 **Accepted 20 Nov. 2007**

1.0 Introduction

Fault tolerance in highly parallel processing systems is important in order to achieve high dependability and sustained high performance computing. The fault tolerance provisions in the system, have to incorporate mechanisms to detect and localize errors as well as mechanisms to reconfigure the system (to isolate faulty nodes), and to recover from erroneous states,. thus limiting the effect of the fault on performance.

The fault may be transient or permanent. The transient fault mainly due to temporary environmental change, and to distinguish between transient and permanent faults, a typical technique is to perform a limit number of retry and declare a fault as permanent if it persists beyond on

average duration of transient fault. Pipeline multiprocessor systems usually have large number of processing elements, so the probability that some processor fails can be high.

Several techniques have been used in the designs to make the linear array pipeline system fault tolerance. Some of these techniques employed switching network [1][2], to achieve fault tolerance in linear array pipeline system. Other techniques use graph-based bypass connection [3], or replacement circuits [4]. However most of these approaches involves incorporating spare processor elements and use large number of external switches, control processor or complicated control circuit to detect and locate errors, reconfigure the system and recover from error.

In this paper, we propose a fault tolerance system that can automatically reconfigure itself in case of fault. Reconfiguration process in our design in response to any processor fault is easy and can be performed in a distributed manner without using centralized external host processor or large interconnection switch. Also the proposed system performs fault detection protocol implicit within the message transfer protocol and do not require separate fault detection algorithm. This minimizes the overhead time cased by fault detection algorithms. The system uses Texas Instruments TMS320C40 DSP processor as processing node. The parallel processing capability of the TMS320C40, which was designed specifically to facilities system reconfigurability, makes the processor ideal for a parallel processing node and particularly for scientific and DSP applications. The TMS320C40 contains six built-in, high speed communication ports (links) and each one may be configured as an input, output or bidirectional port. In our proposed scheme we exploits the inherent redundancy links (communication ports) provided in the TMS320C40 with extra spare processors to achieve fault tolerance in the pipeline system.

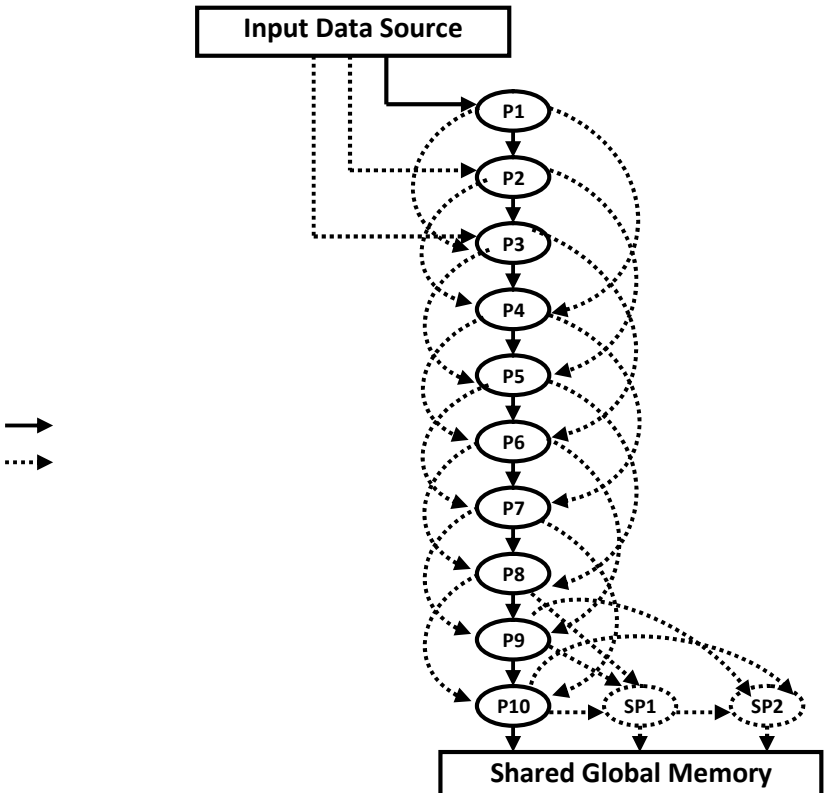
The Paper is organized as follows: section (2) describes the architecture of the system along with the principle of operation. Implementation of message transfer protocol between PMs is provided in section (3). Section (4) and (5) introduce the operation of the system in

normal and fault cases. Section (6) discusses the performance of the system. Finally, some conclusions of this works are mentioned in section (7).

2.0 The Proposed Architecture

The proposed scheme uses twelve TMS320C40 DSP processors connected in a linear array pipelined parallel processing system as shown in figure (1). The basic configuration of the system consists of ten stages (one DSP processor in each stage from P1 to P10) .The spare processors (SP1 and SP2) are linked with the last stage of the pipeline to be ready for use in event of failure of one or two processors in the basic configuration.

Each processor in the pipeline use one of their output communication ports as a primary link to connect it with its successor processor in the array. And use another two output communication ports as bypass links to connect it with the two processors that follow its successor processor. These bypass links are used to connect the processor to the next processor in the pipeline in case of failure of one or two of successor processors. Processor P1 is the processor of stage-1 in the pipeline and also serves as the input node for the pipeline. While processor P10 is the processor of the stage-10 and also serves as the output node for the pipeline**.**



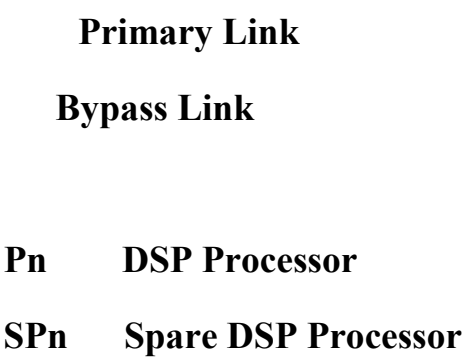


Fig.(1) Proposed system architecture

All of these comports are configured as input or as output port depending on the function of that port. For example, communication ports that link the PM to the successor processors are configured as output ports. While these communication ports that connecting the PM to their predecessor processors are configured as input ports.

Since the input and output processors are subject to failure, it is necessary to be able to input information into the array and out of the array through more than one processor. For this purpose processor P2 and P3 have bypass links to the input data source. So that, P2 may be used as

the processor of the first stage and as the input processor (in case of faulty of P1) and P3 may be used as the processor of the first stage and as the input processor (in case of faulty of P1 and P2). Processors P8, P9, and P10 have bypass links connecting them to the spare processors (SP1 and SP2). By this method, SP1 can be used as the processor of stage-10 and as output processor if any of processors in the basic configuration is faulty. And SP2 can be used as the processor of the final stage and as output processor of the any of two processors in the basic configuration are faulty. This connection strategy allows the pipeline to continue in the operation even in the failure any two processors in the system.

P10 and the spare processors (SP1 and SP2) all share a common global external memory which is used to hold the result of the processed data. Only one processor uses the shared memory at any time. In a fault free operation, P10 only use the shared memory. While in case of one processor fault, then SP1 use the shared memory to hold the result, and in case of two processor faults the SP2 use the shared memory.

3.0 Message Transfer Protocol

In this section, a message transfer protocol is developed for transferring messages between processors in the array. The C‘40 has six parallel bidirectional communication ports with built-in arbitration units to handle data transfer between PMs. For software these commport can be treated as 32-bit on-chip data I/O FIFO buffers. Processor read data from/writes data to commport is simple[5] :

```
LDI  @ Comm_port0_input, R0      ;  Read data from
commport0

STI   R0; @ Comm_port0_output    ;  Write data to
commport0
```

Every PM send a message to the following PM using the commport and receive message from the preceding PM using another commport. The format of data frame consists of three parts as shown in fig(2). Two pointers pointing to the frame length and address of the next stage to be processed and the last part contains the data to be processed. The address

of the next stage is important for the receiving processor to know which stage must be executed (specialty in case of fault).

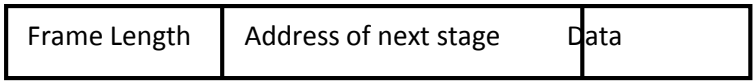


Fig (2) Format of the data frame

Each of the TMS320C40 communication port is used in a unidirectional configuration. Figure (3) is an example of two ’C4x DSPs connected via their communication ports. This simple communication interface consists of the following bidirectional control and data lines[5]:

- **CREQ x :** communication-port token request. A ’C4x activates this signal to request the use of the communication-port data bus.
- **CACK x :** communication-port token acknowledge. A ’C4x activates this signal to relinquish ownership of the communication-port data bus upon receiving a CREQ x from another ’C4x.
- **CSTRB x :** communication-port strobe. A sending ’C4x activates this signal to indicate that it has placed a valid data byte on the communication port data bus.
- **CRDY x :** communication-port ready. A receiving ’C4x activates this signal to indicate that it has received a data byte via the communication port data bus.
- **CxD(7–0) :** communication-port data bus. This bus carries data bidirectionally, one byte at a time, between two ’C4xs or between a ’C4x and some other device.

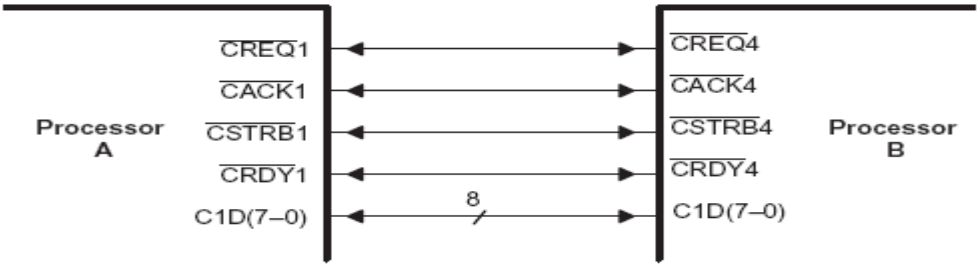


Fig. (3) ’C4x Communication-Port Interface-Connection

4.0 Operation Of The System In Fault-Free Case

In fault free operation, every processor becomes able to communicate with its adjacent processor in the pipeline using the primary commport links. Processor P1 starts processing by taking a new frame of data from the source and processing stage-1 of the task. After that P1 send the processed frame to P2 to process stage-2 of the task on frame-1 via the primary link connecting them and using the message transfer protocol described in previous section. And P1 proceed directly to receive new frame from the input data source and execute stage-1 of the task on this frame. The procedure and then continue through the rest of pipeline processors using the primary links for communication between processors. P10 execute the final stage of the task (stage-10) of each frame and puts the result in the shared memory. In this case (fault free operation), the spare processors (SP1 and SP2) remain in idle state and they not share in processing.

5.0 Operation Of The System In Fault Case

There are two main types of fault detection protocols in the literature: The first is group membership-based [6] and the second is gossip-based fault detection protocols[7]. In the fist type each node in a distributed system monitors the state of every other node in the group by direct communication with it. This is an effective method of fault detection on small systems, but since it requires all-to-all communication, network congestion becomes a significant bottleneck as the system becomes larger. An alternative to group membership protocols, in the second type of fault detection protocol, each node communicates with a subset of the other nodes in the system. Its view of the system is determined by a combination of its own information and that received from these other nodes. This requires only selected point-to-point communication. In addition, it is easy to design a hierarchical structure for the gossip-based protocol. For these reasons, a gossip protocol is more scalable than a group membership approach and thus, more appropriate for fault tolerance on a large system. In our proposed scheme, gossip-based fault detection protocol has been adopted.

The PMs in the system can become faulty or can be repaired on an arbitrary number of times during the operation. The concept of a " Forward Task Shift (FTS) " mechanism is introduced in which the task of each PM is shifted to the next PM in the pipeline in event of failure in one of the PMs. And the task of the last PM is shifter to spare PM without loss or restart the execution comparing with other approaches. There is no required for a global clock or synchronized clock in the system. Hence, the fault detection algorithm is embedded in the message transfer protocol used in the system, which minimize system overhead due to node diagnosis and avoiding the using of check-pointing or replication processes.

The following subsections presents the mechanism of using the gossip-based fault detection protocol in the system to transparently tolerating the fault in one or more DSP processors in the array.

i) One Processor Fault

Each processor is responsible for monitoring the state of its neighboring processor by checking the health of its successor processors. This checking is done implicit during the normal message transfer protocol by sending a strobe signal (CSTRB') through the primary communication link that connecting them as mentioned previously. The successor processor must respond by sending a ready (CRDY') signal within a predetermined time interval. If the strobe signal is acknowledged by the ready signal, then the next neighbor is still functioning. But if the sending processor dose not receive the ready signal (CRDY') within the stipulated time interval, the processing node concerned is assumed to be faulty. In such situation, the processor first breaks its connection with the faulty node and bypassing it by sending a strobe signal to the processor that follow its faulty successor using the bypass link connecting them. For example, the bypass link between P1 and P3 is used to link P1 to P3 when P2 is the faulty processor. Using FTS mechanism, P3 in this case will execute the task of P2 while P4 will execute the task of P3 and so on. At last, the spare processor SP1 will enter in the basic configuration of the pipeline to execute the task of P10 and writes the result in the shared

memory. The time-out periods are determined as a function of maximum response-ready (CSTRB‘-CRDY‘) delay time which depends on the operating frequency of the processor.

When the PM detecting the fault, issues a FAIL message, identifying which PM has failed, to the next PM connecting with it. And this message then propagates only in the forward direction to all the following PMs. The PMs before the faulty node need not to know which PM faulty, because failure will not affect on the number of stage that must be processed. However, the PM that detect the failure in its neighboring PM, must repeat the test of every and for a limit number of times (for example; ten times) to see if the faulty node has been repaired (transient fault). If the faulty PM responds during these retries, then PM that perform the test will send a REPAIR message to their neighboring PMs to inform them that the faulty node has been repaired. And this message propagates in forward direction to every PM in the array and the repaired PM reintegrated in the system. But if the faulty PM dose not responds during these retries, it considered as permanent fault, and stopping testing it.

The FAIL and REPAIR messages consists information about the address of the faulty node. The FAIL and REPAIR message are important specially for each spare PM to know when it will enter or exit from the pipeline. Also it is important to know which PM will be the final stage and then will use the shared memory to put the result.

In addition to mentioned above, PM2 in the array must monitor the activity of its predecessor. If a failure occur in PM1 during the processing, then PM2 will detect that if it dose not receive a strobe signal from PM1 within the predetermine time interval (which is equal to the time of processing one frame), in this case PM1 is considered to be fault. In such case PM2 will use the bypass link to receive the data frame from the input data source directly and start processing task of PM1. However,

in the event of the fault, the processor that follows the faulty processor will be in a wait state until it receives a strobe signal and then the data frame through one of the bypass links connecting it with the predecessor processor.

Finally, it is necessary that each individual processor in the array keep updating the information concerning the configurations incurred in its neighborhood. This information must be updated with any FALL or REPAIR message and includes mainly which link is used to connecting it with the adjacent processors and address of the task that must be processed and address of any faulty PM.

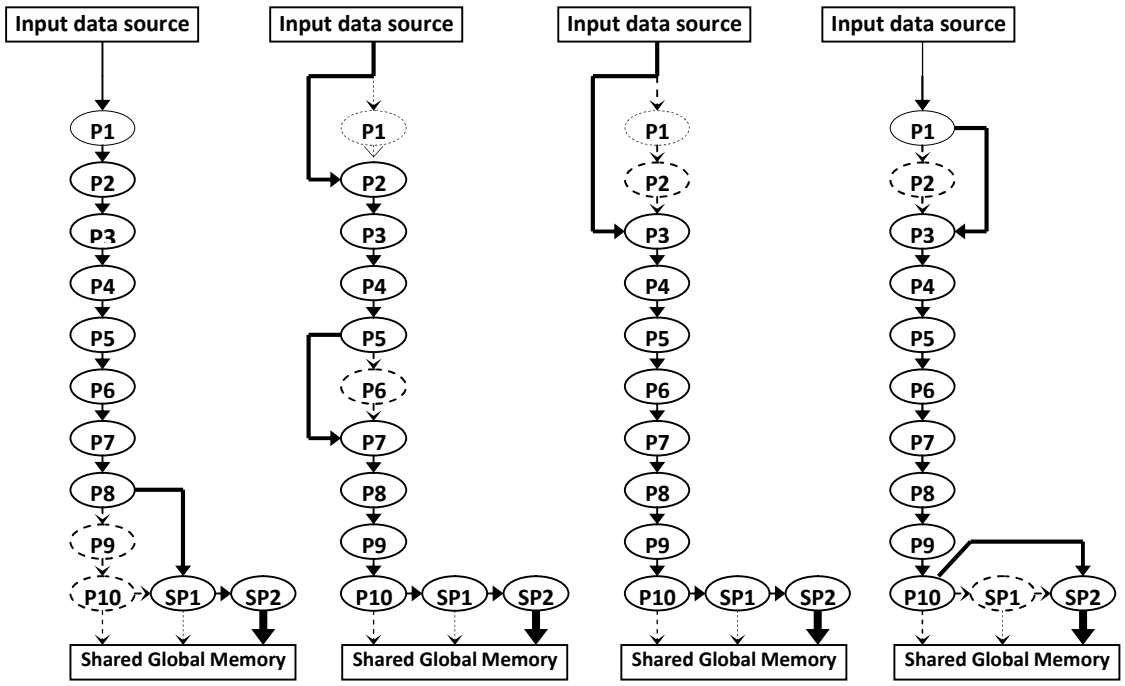
ii) Two Processor Faults

The same protocol in fault detection described in previous section will be used if two or more processors are fails. And the system will automatically reconfigure itself to allow the two spare processors to contribute with the pipeline instead the faulty nodes and SP2 in this case will serve as the output processor. Fig(4) shows examples of the data frame routing in the reconfigured array upon the failure of two processors in different locations in the array. As explained in previous section, PM3 (in addition to PM2) must monitor the functionality of its predecessor processor. For example, if PM1 and PM2 are failed, then PM3 will not receive a strobe signal within a predetermine time interval. In such case PM1 and PM2 are regarded as faulty nodes. And PM3 use the bypass link to receive new frame of data from the data source and start processing task of PM1.

iii) Three Processor Faults

There is a bound to the maximum number of faulty processor in the system such that the pipeline operates correctly without affecting to number of pipeline stages. This bound depend on the number of the spare processor and the number and locations of the faults. In present system, with two spare processors, this bound is determined by two faults.

However, to maintain the same number of pipeline stages as in the basic pipeline configuration, the number of faulty node should not more than the number of spare



(a) Processor P9 & P10 are fails

(b) Processor P1 & P6 are fails

(c) Processor P1 & P2 & SP1 are fails

(d) Processor P2 are fails

Fig. (4) Reconfigured array upon failure of two processors

processors. Also the adjacent faulty nodes (that connected directly one to another) should be not more than two under any condition. This due to the limit number of bypass links available in each PM. However, if the number of faulty processors is greater than the number of spare processors, then the performance of the system will degraded due to decreasing number of the pipeline stages.

In present system with ten processors and two spare processors, failure in three processors, for example, leaves the system with only nine processors. So that the system can function as a nine stage pipeline unless that the three faulty processors are adjacent such as (P1, P2 and P3) or (P2, P3 and P4) or (P3, P4 and P5) and so on . In this case there is no bypass link can be established to link the processor before the faulty processors with the processor that follow the faulty processors and the array is declared failed.

6.0 System Performance Discussion

Failure in one of the PMs will introduce a temporary short delay (due to fault diagnoses overhead time) and this time is equal to the time-out period. The overhead time caused by the fault diagnosis is depend to number of faulty processors. However, fault diagnosis procedure that adopted in this system is not executed periodically during run-time, but it performed only during the fault. Therefore this overhead has no significant effect on degradation of system performance. Also the overhead time due to fault diagnosis is depending on the location of the fault. For example, the overhead due to the failure in P1 is greater than

the overhead time due to failure in P2-P10. Since, the overhead of detecting the fault if the fault occurs during the task processing is greater than if the fault occurs before the task processing.

In the present proposed scheme (with 12 PM), failure in one or two PMs does not result in system failure or significant degradation in performance. Failure in more than two PM will also dose not result in system failure unless the faulty processors are adjacent. However, failure in more than two PM will case degradation in the performance because the last spare PM will then be required to take over the additional computational task of the failed node.

The overhead time due to reconfiguration is independent of the array size. Another advantage of our design is the ability, due to its distributed processing, to handle multiple faults which occur at the same time. Finally, because of using the inherent communication redundancy of the TMS320C40, the scheme of fault tolerating offers a reduced hardware cost and increased system reliability.

7.0 Conclusion

A fault tolerance pipeline architecture based on the TMS320C40 DSP processor is presented in this paper. The system consists of twelve processing modules to construct ten pipeline stages (one PM in each stage) with two spare processing modules. The system achieves the fault tolerance by using the spare PMs and the bypass links. In case of failure in one of the PMs, the system reconfigures itself and automatically incorporating one of the spare PMs instead of the faulty PM. The function of the faulty node is assigned to the next fault-free PM (FTS mechanism) without noticeable degradation in performance level.

The fault model described in this paper can be applied to cover both permanent and transient faults. Our approach combines advantages of using minimum hardware with high performance fault tolerance parallel

processing system. Since the system exploiting the communication ports of the TMS320C40 to establish the bypass links to limit the fault tolerance cost without using extra complicated connection, switching networks, or host processor. Also the fault diagnoses software is embedded within the message transfer communication protocol to minimize the fault diagnoses overhead time.

Every PM in the pipeline test the functionality of its adjacent PM by sending a strobe and receiving an response in predetermined time. The PM is treated as a faulty PM if it is inaccessible from its former PM. In general, the system can withstand failure of more than two PMs and still operate as a pipeline array unless the failed PMs are adjacent. However and with the using of two spare PMs , failure in more than two PMs will result in degradation of system performance.

References

[1] R.Mazzaferri and T.M.Murray " The connection network class for fault tolerant meshes " IEEE trans. on computers, Vol. 44, No. 1, Jan 1995.

[2] ---"Constant time fault tolerant algorithms for a linear array with a reconfigurable pipelined bus system", J. Parallel Distrib. Comput. 65 (2005) pp. 374 – 381

[3] N.Tsuda **"**Fault tolerant processor array using additional bypass linking allocated by graph-node coloring", IEEE Trans. on computers, Vol.49, No.5, May 2000.

[4] S.H. Hasseini " On fault-tolerant structure, distributed fault diagnosis, reconfiguration, and recovery of the array processor", IEEE Trans. on computers, Vol.38, No. 7, July 1989.

[5] Texas Instruments "TMS320C4X User's Guide" , Texas Instruments , may 1999,

[6] K. P. Birman, “The Process Group Approach to Reliable Distributed Computing”, Communications of the ACM, Vol. 36, No. 12, pp. 37-53, Dec.1993.

[7] R. van Renesse, Y. Minsky, M. Hayden, “A Gossip-Style Failure Detection Service”, In Proceedings of Middleware ’98, 1998.

[8] R.K.Kumar, S.K.Sinha and L.M.Patnaik “A fault tolerant multi-transputer architecture”, Microprocessors and Microsystems, Vol.17, No.2, 1993.

[9] M.G. Karpovsky, T.D.Roziner and C.Moraga “Fault detection in multiprocessor systems and array processor”, IEEE Trans. on computers, Vol. 44, No.3, 1995.

[10] Racine, LeBlanc and Beilin, "Design of a Fault-Tolerant Parallel Processor", 21st IEEE Digital Avionics Systems Conference, Irvine, CA, 2002.

[11] Tadayoshi Horita and Itsuo Takanami “Full Fault-Tolerant Processor Arrays Based on the Track Switches with Flexible Spare Distributions”**,** IEEE Computer, June 2000, Vol. 49, No. 6, pp. 542-552

[12] S. Dutt, and J.P. Hayes, “Some practical issues in the design of fault-tolerant multiprocessors,” *IEEE Trans. Computers,* Vol. 41, No. 5, pp. 588–598, May 1992.

[13] Anu G. Bourgeois ,Yi Pan, Sushil K. Prasad “Constant time fault tolerant algorithms for a linear array with a reconfigurable pipelined bus system”, J. Parallel Distrib. Comput. 65 (2005) , pp 374 – 381.