Defence Science Journal, Vol 44, No 2, April 1994, pp 99-104

Inter-Processor Communication for Fault Diagnosis in Multiprocessor Systems*

C.D. Malleswar

Naval Science and Technological Laboratory, Visakhapatnam-530 027

and

Ashok Jhunjhunwala

Department of Electrical Engineering Indian Institute of Technology, Madras-600 036

ABSTRACT

In the present paper a simple technique is proposed for fault diagnosis for multiprocessor and multiple system environments, wherein all microprocessors in the system are used in part to check the health of their neighbouring processors. It involves building simple fail-safe serial communication links between processors. Processors communicate with each other over these links and each processor is made to go through certain sequences of actions intended for diagnosis, under the observation of another processor. With limited overheads, fault detection can be done by this method. Also outlined are some of the popular techniques used for health check of processor-based systems.

1. INTRODUCTION

It has become difficult to check the health of microprocessors in application environment because of their becoming complex with large repertoire of instructions, addressing modes and data manipulation abilities.

Several techniques exist to check the health of microprocessors. A few of them include use of watchdog timers, in-built redundant systems for health checking, test generation with signature analysis, and a more recent technique called boundary scan architecture (IEEE P1149.1). To check the health of micro-processors in application environment, a simple technique is proposed for multiprocessor multiple system environments. In this method, the micro-processors, apart from their routine work, are also used to check the health of their neighbouring processors by building fail safe serial communication links (akin to computer networks), over which they communicate with each other. Each processor goes through certain definite motions or sequence of actions intended for diagnosis, under the observation of another processor. With limited overheads, fault detection can be achieved by this method.

2. CURRENTLY USED TECHNIQUES

The increased complexity of VLSI microprocessors makes them less reliable and more difficult to test in an application environment. This is more so with CISC VLSIs. As the microprocessor is the heart of the system it drives, it is essential to ensure by periodic checks the health of the processor. While the microprocessor itself may be used to check the health of the rest of the system by diagnostics or other methods, the health of the microprocessor and small core of electronics around it is difficult to check. This core which comprises of microprocessor, clock and reset circuit, buffers, decoder and ROM used in diagnostics, must be ensured to be fault-free to run reliable diagnostic checks. Some of the popular techniques are outlined in the following sections.

Received 18 August 1992

* Invited paper for the Special Issue of DSJ on Computer Applications in Defence (April 1993)

2.1. Use of Watchdog Timers

Figure 1 indicates basic idea used in this method. The microprocessor gets periodic interrupts from the timer. The timer interrupts are processed by the microprocessor to activate retriggerable mono shots in the watchdog timer. During error-free operation, the output of the watchdog timer is held in logic '1' state. When the microprocessor fails to trigger the watchdog timer, the output of the watchdog timer changes state to low which is used as an error signal. A duplicate watchdog timer is used so that one of them can be put in test mode while the other drives the outputs. Periodically the roles of the two watchdog timers is interchanged. Two mono shots are used in each watchdog timer for fail-safe operation. If one of them is stuck-at high, the other input can be used to de-assert the watchdog timer output. At least three one-shots must work to generate the fail signal. The latch at the output which is cleared on power-up will latch the error signal on the first occurrence of rising edge at its clock input which will be caused by processor failure. Only rest can restart the watchdog timers. The fail signal is used to take fail-safe shut down action.

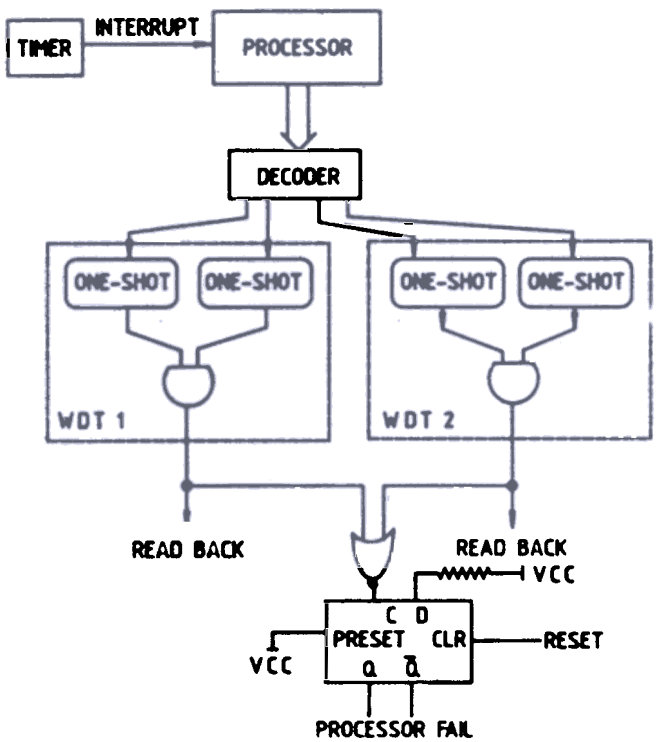

Figure Watchdog timer for detecting processor failure

The fault coverage of watchdog timers is low considering the complexity of the microprocessor. Therefore this technique may be used in conjunction with other techniques for fail-safe system applications.

2.2 Built-in Redundency for Health Checking

This technique is widely used and preferred as a high degree of fault coverage is possible. This method is used in safety critical applications. In this method, the processor hardware and software are duplicated and the signals from both processors are compared using totally self-checking comparators. The redundancy is primarily intended for ensuring testability/fault detection. A computer module using Viper 1A processor illustrates this approach[1]. As with any duplicate and compare systems, caution should be exercised in dealing with common mode errors which cannot be detected. An extension of this method is to have several redundant modules with a reliable voter for fault diagnosis. The main disadvantage of this technique is substantial increase in hardware (>100 per cent) resulting in increased space and power requirements which are major constraints in Defence applications. The systems using redundancy are also very expensive.

2.3 Test Generation with Signature Analysis

In this method, a test program is generated using the instruction sets and functions performed by the processor. This test is generated for a fault model of the microporcessor. The test program is then executed and the signatures generated by the processor are captured from the external pins of the microprocessor. These signatures are compared with the expected ones either in real-time or off-line and a decision is taken regarding the health of the microprocessor[2]. This method requires an external signature analyser and a fault model with a test generation program. A fault simulation is also needed to validate the model for the specific processor in use.

2.4 Boundary Scan Architecture

This methodology requires testability to be built into the hardware and, can be used for chip level (VLSI), board level and functional level testability of systems. ANSI/IEEE Standard 1149.1, IEEE standard test access port and boundary scan architecture, defines this architecture and method. At the chip level, this technique requires the scan cells to be implanted at I/O

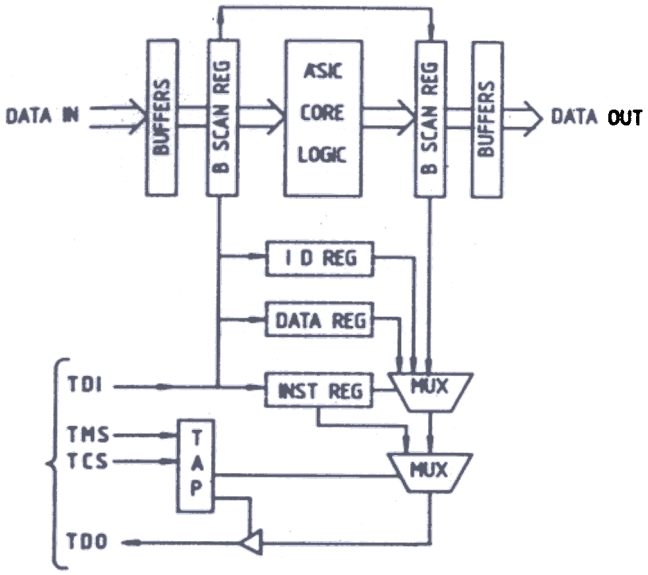

Figure 2. An ASIC VLSI with in-built test access port using boundary scan architecture.

points of the chip, which can be used to isolate external signals by control signals applied through a test access port (Fig. 2) and then exercising the chip by shifting in test data serially through test access port. The responses of the chip to the test data is also shifted out serially and collected by a signature analyser. This technique permits in-circuit testing of devices. At board level, the PC board can be partitioned depending on functionality and test requirement. Then boundary scan cells are placed along the partition boundary and are serially linked. Each circuit within a partition can be independently tested through a test access port by shifting in test data serially and collecting back responses. The test access port has 5 signals: the test data input (TDI) which is a serial input line; test clock (TCK) which is used for shifting data; test mode select (TMS); test data out (TDO) which is used for serially shifting the data out; and an optional (fifth signal) test reset input (TRST). More details on this aspect were reported earlier[3-5].

The boundary scan architecture method clearly requires special purpose hardware to be built into the devices and printed circuit boards for accessing and testing them. Also an external intelligent unit is needed to exercise the tests and to analyse the signatures collected for fault diagnosis. This method would be of immense use for ATE-based testing. However, testing is time consuming as the test data is shifted serially. A lot of storage space is needed to store all the test inputs and reference signatures. With these constraints it may be difficult to use this technique for online health monitoring.

3. THE PROPOSED TECHNIQUE FOR HEALTH MONITORING

A simple technique is proposed using reliable communication between two processors for checking health of each other. Consider two processors *A*, *B* which are linked by a serial link as shown in Fig. 3. The two processors periodically exchange data packets which contain test data field. The check process is initiated by the processor *A* sending a packet to processor *B*. On receiving this packet, processor *B* has

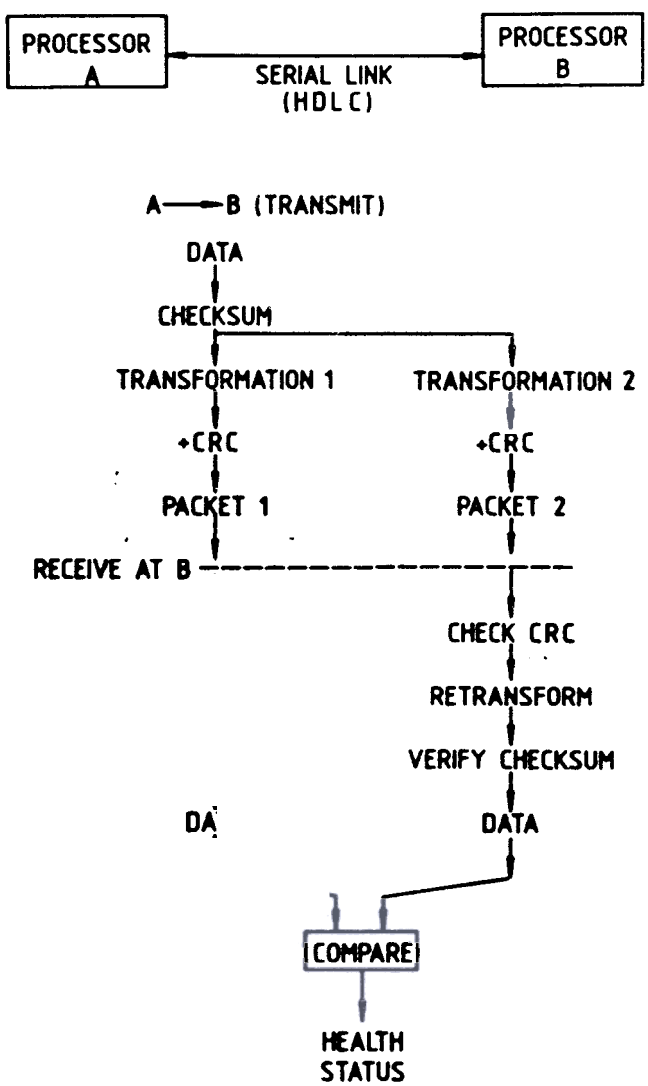

Figure 3. Health checking by simple serial communication links and data transformations.

to carry out a transformation on this data involving a variety of processor actions. These actions may include memory read/write, DMA transfer, besides arithmetic and logical operations. The transformed data is sent back to *A*. The processor at *A* will also parallelly carry out the same transformation. This end result is matched by processor *A* with the result received from *B*, to declare *B* to be healthy or not. Similarly processor *B* will also send a test data packet to ascertain whether *A* is healthy.

Three key factors to this technique are: (i) the communication link must be error-free; (ii) the transformation carried out should be such that any fault in the processor will affect the end result. Thus all critical sections/instructions that need to be checked must be involved in a critical way in the transformation; and (iii) care should be taken that test data pattern is not repeatedly sent which should be generated afresh every time (either reading from external channel or generating and using random data).

The technique can be generalised involving a group of processors, connected in any fashion, for example, as shown in Fig. 4, *B* could check the health of *A, C*; *C* could check the health of *B, D*; and *D* could check the health of *C, E*; etc. The test process involves the (i) generating test data; (ii) communication link handling; and (iii) transformation and comparison.

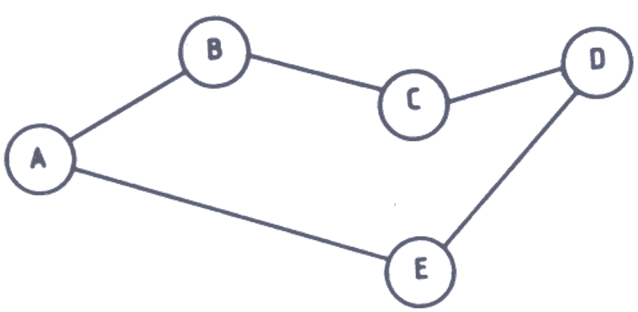

Figure 4. Arbitrarily connected processors.

This may need 10-20 per cent of processor resources and time. Each processor has two results of the test: self diagnosis result which involves only self, a result which involves health of both the processors involved in each of the tests. The two processors also time out messages to detect loss of communication and other fatal failures at each node.

We need to ensure that communication between the processors should be fail-safe/reliable for which well-established tenchniques are available. The exact scheme depends on the kind of link (specified expected bit error rate) and the final undetected error rate that one wishes to tolerate. Accordingly a powerful CRC with right frame length or message redundancy for error detection and correction can be used. Simple serial links with protocols like HDLC may be used which can be implemented with addition of a single chip. Messages must be timed out at receivers for detecting loss of messages or failure of communication links. Positive acknowledgements are used for reliability. When messages are received in error, retransmissions are used for ensuring error-free delivery of messages.

4. THE TRANSFORMATIONS

The objectives of test transformations are:

(a) The transformation used must exercise at least all those features of the processor that will be used by the application program,

(b) The transformation must be able to precipitate stuck-at faults that may affect normal execution of the application program, and

(c) The transformation should not use such operations which may result in truncation, over flow, under flow, etc, which cause irrecoverable loss to the data. When the reverse transformation is done the data must be obtainable bit to bit without error.

The complexity of these transformations depend on the complexity of the processor and its features. To write a suitable test program to exercise a CISC processor for the features utilised by the application, one can make use of the static instruction listing of the application program which forms a subset of instruction set of the processor. This set can be reduced further by taking dynamic instruction counts into consideration. Less frequently used instructions may be realised by the more frequently used instructions to shrink the instruction subset used by the application. This optimisation will help to reduce the time needed for each transformation. Now the transformation is generated using this subset of instructions, which exercises the processor when the test program is executed. this task is simpler with RISC processors which have small instruction sets and limited addressing modes. In case the test program only covers part of the

processor features used by the application, the remaining features will have to be carefully exercised alongwith the test program and their result suitably used. Examples of some of the simpler transformations, all or some of which may be used on the data are:

(a) *Calculation of check sum*: Sum of all bytes of message including check sum is zero.

(b) *Bit reversal*: Byte-wise bit reversal.

(c) *Rotate through scrambling*: (i) Cumulative parity of previous bytes in a message are used to decide whether rotation is to right or left, and (ii) the number of times the rotation itself is done is obtained by taking modulo 4 of the value of the previous byte of the same message.

(d) *Memory (RAM) test*: Write 55 in memory, read back in register *A*, write *AA* in memory, read back in register *B*, EOR *AB* (if there is no stuck-at fault must give a result of *FF*), compliment result, and add result to data.

At the end of this operation if there is a stuck-at fault in the memory or its data path the data gets corrupted. As a consequence of this, when the receiver tries to retransform the message back to data it will indicate error.

(e) *ROM check*: Compute check sum of data in ROM, negate the result, add to original check sum, and subtract the result from data bytes of message.

If the check sum was right then the result would be zero and subtracting it from data would not make any difference. If the check sum did not match then the data of the message would to be affected.

(f) *I/O check*: Loop back and read back checks may be performed on I/O and the results may be used as discussed in previous cases.

In transformation (c), the contents of some bytes of data (i.e. parity and modulo value) was used as a parameter for transformation of remaining bytes. This causes single error to appear as multiple errors in the message, thereby enhancing error detection.

These transformations check arithmetic, logical, branching features, internal and external RAM, EPROM/ROM, and some testable features of I/O. By suitable design of test software, the addressing modes, internal registers, and interrupt features can be checked.

5. APPLICATION FOR DEFENCE USE

The proposed technique can be used in Defence in multiprocessor environment such as advanced SONAR, RADAR and tactical weapon control systems, etc; and multiple system environment such as C^3I systems. In most advanced submarine combat systems, a few hundreds of processors are being used[6].

The connectivity of the processors in a weapon control system having 8 CISC processors linked by

PROCESSOR NUMBER

PROCESSOR NUMBER	1	2	3	4	5	6	7	8
1	D	X	X	X	X	X	X	X
2	X	D	-	-	X	X	-	-
3	X	-	D	-	-	X	X	-
4	X	-	-	D	X	-	-	X
5	X	X	-	X	D	-	-	-
6	X	X	X	-	-	D	-	-
7	X	-	X	-	-	-	D	-
8	X	-	-	X	-	-	-	D

Figure 5. Processor connectivity diagram for TWCS example.

HDLC serial communication links, is shown in Fig. 5. The system is organised to have 5 clusters of processors and can tolerate one or more processor failures at each cluster. Critical data paths are duplicated. Broad function of these clusters are: (a) data acquisition, (b) coordination, (c) target motion analysis , (d) display, and (e) weapon control. The failures that can be tolerated are total or partial failure of cluster (a), partial failures at (b) and (c), total failure of one of (b), (c) and (d), (e). Other features include incorporation of test access ports for ATE purpose, hot insertion for online repair, watchdog timers for processors, internal and external loop back features for I/O channels. Referring to Fig. 5, a '×' indicates that the processors in that particular row and column have a communication link. The communication links are designed to be fail-safe and reliable. A '-' indicates that corresponding

processors are not linked. All the processors which have communication links between them use the proposed technique for checking each other. *A 'D'* in the entry at the diagonal indicates that local self-check is carried out by each processor in addition to mutual checks. Processor 1 is linked to all the other processors in the system. If processor 1 has a fault, the results from all the other processors indicate the same. In the specific application considered, there are only two types of processors. Therefore, the transformation programs need be written only for the two processors. The links use 16-bit CRC and fail-safe message transmission between them. This application has both multiple processor and multiple system environments. Processor 8 forms part of an advanced SONAR system and processors 1 to 7 belong to multiprocessor tactical weapon control system.

6. CONCLUSION

In the proposed simple health check, the extent of fault converage depends on complexity of processors and the transformations that are used for check purpose. This technique may also be tried with computer networks, with lightly loaded links and spare processing power at nodes. Utilisation of a non busy processor in a multiprocessor system for a diagnostic purpose was dealt in earlier studies[7]. A practical application of distributed system level diagnosis for large networks was reported earlier[8]. Instead of having exclusive packets for health checking, one can use regular data packets also for this purpose. If every data packet undergoes transformation the processing power at nodes may become a bottle neck. It is therefore suggested that periodically some packets may be used to check health of the processors in the network. If a packet passes through several nodes before reaching its destination, care should be taken to avoid mix up of health check information of the nodes in the path. To avoid this problem nearest neighbours may only carry out this operation.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Timothy A Gonsalves of Department of Computer Science and Dr Bhaskar Ramamurti of Department of Electrical Engineering, IIT, Madras, for the useful discussions and their valuable suggestions.

REFERENCES

1. Hallbert, Max. P. Selfchecking computer module based on the viper microprocessor. *Microprocessors and Microsystems*, 1988, **12**(5), 264-70.
2. Thatte, Satish M & Abraham, Jacob A. Test generation for microprocessors. *IEEE Trans. Computers*, 1980, **C-29**(6), 429-41.
3. Maunder, Collin M. & Tulloss, Rod E. Testability on TAP. *IEEE Spectrum*, 1992, 34-37.
4. Riessen, R.P. Van; Kerkhoff, H.G. & Kloppenburg, A. Designing and implementing an architecture with boundary scan. *IEEE Design & Test of Computers*, 1990, 9-19.
5. Ellis, Milton Jr. & Bell, Bill. Bottom-up techniques propel board testability. *Electronic Design*, 1990, **24**, 57-64.
6. Submarine Fire Control Systems. *In* Jane's Underwater Warfare Systems, Ed. 2. 1990-91. pp. 28-42.
7. Simoncini, Luca; Saheban, F. & Friedman, A.D. Design of self-diagnosable multiprocessor systems with concurrent computation and diagnosis. *IEEE. Trans. Computers*, 1980, **C-29**(6), 540-46.
8. Bianchini, Ronald Jr.; Goodwin, Ken & Nydick, Daniel S. Practical application and implementation of distributed system-level diagnosis theory, *In* 20th IEEE FTCS Symposium, 1990. pp. 332-39.