On Designing and Evaluating a High Reliability Microcomputer System for Real–Time Industry Applications

Yen–Tseng Hsu and Chen–Fa Hsu
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract – The requirements for fault–tolerant computer systems in industry applications are very different from the requirements for those in transactions processing. One of the most obvious differences is the demand for faster processing speed. In addition to high reliability, the coverage, the error latency and the retry time are also the important factors in the design of fault–tolerant computer systems. In this proposed architecture we design an error–reporting circuit and a faulty address latch to detect the failure and to minimize the retry time, respectively. Furthermore, the paper employs the MIL–HDBK–217E model to predict the failure rates of system module, processor module, memory module and hard core, and shows that the mission time improvement factor of proposed system is almost independent of a given reliability. Finally we use a Markov process model to evaluate the reliability and analyze the coverage, the error latency, and the retry time of proposed system. The proposed architecture possesses the following characteristics: short retry time, high reliability, high coverage, low error latency and is well suitable for the real–time industry applications.

1. INTRODUCTION

The goal of fault–tolerant computation is to develop and certify computing systems which can perform satisfactorily in the presence of faults. For the different requirements of fault–tolerant systems, the fault–tolerant machines can be classified as follows. (1) Transaction processing systems, such as reservation system for hotels, airlines, lottery and banking system. Their retry time and error detection latency are too long to fulfill the requirements of critical real–time industry applications. (2) Critical real–time industry applications, such as August system [1]. It annihilates interruptions with three processors and two–out–of–three voting in the system level. The systems of this area, which either vote data on the system level or detect failures noncurrently, need long retry time and have long error latency compared with the proposed system.

A real time system is the one that responds immediately to the commands and processes data as soon as the data appears. If the commands or the data are produced very fast, the system can't have latency. In this area a computer system is integrated into its environment through sensors that allow it to supervise the states of practical devices and processes [2]. Therefore, for the industry applications the computations are critical and the operation can't be interrupted. They need short retry time system. Usually there is a fault detection mechanism to take the correct action when the system failed. The nonself–detecting circuit has longer retry time and lower coverage than the self–detecting one. For some applications, the retry time must be short as it is necessary to carry out at least two self–detecting computations or three nonself–detecting computations concurrently. The error latency is the time between the occurrence of the fault and the moment that the fault is detected. In general, the TMR with concurrent detection has the shortest error latency time.

Manuscript accepted on February 4, 1991.

0-7803-0453-5/91$1.00© 1991IEEE

The measure of detection quality could be the percentage of system failures called the detection coverage. It is defined by the conditional probability given that a fault occurs, and the system will detect the fault properly by the detection circuit or diagnostic program. For some applications, high coverage may be expected for high reliability. It is obvious that the standby redundancy has a major advantage that is suitable for the long–life unmaintained systems [3,4], but its coverage is lower than those of the voted system and duplex system. Since the TMR (triple modular redundancy) would not cover faults when there are three modules or two modules failed simultaneously and the coverage of duplex system depends on the design of the diagnostic program, the duplex approach usually has lower coverage than TMR scheme. Therefore, if the high coverage is required, TMR must be used.

Reliability models of fault–tolerant system usually have two types: combinatorial models and Markov models. The former modeling method may estimate the system by combinational means but it isn't suitable for a more complicate system. Moreover, the latter one concentrates on the rate at which transitions take place between different system states. Although this modeling method would obtain analytic solutions by solving the state equations of simple systems, for complicated systems ,repairable systems as well as the systems with hard faults and soft faults, the numerical solutions can be obtained from the numerical method implemented in the computer program [4]. Therefore we use the Markov model to analyze the proposed system. Section 2 reviews the conventional TMR/SLV (system level voting) system and describes the new design of a TMR/BLV (bus level voting) architecture. Section 3 outlines the system software which includes the retry procedure to mask transient errors. Section 4 evaluates the system reliability and analyzes the parameters of coverage, retry time and error latency in the proposed system, thus we can show this proposed fault–tolerant microcomputer architecture is very suitable for the controller of real–time industry applications.

2. DESIGN OF SYSTEM ARCHITECTURE

2.1 TMR/SLV System

The use of fault detection techniques alone does not supply practical tolerance of faults. On the other hand, the fault masking techniques employ redundancy which provides fault tolerance by either isolating or correcting faults before they reach the module outputs. The concept diagram of TMR/SLV system, also known as masking redundancy system, is illustrated in figure 1, where "M" represents a computer system including the processor, memory, address decoder, bus controller, I/O interface, ... ,etc, and "V" symbolize a voter

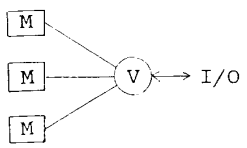

Fig. 1 Architecture of TMR/SLV.

which votes at the I/O operation therefore masks faults occurring in any module. The voter is similar to a full adder which has three inputs and one carry out [5].The reliability of TMR/SLV is $R=(3R_m^2-2R_m^3)R_v$, where R_m is the reliability of computer system and R_v is the reliability of voter. In this scheme the retry time and fault detection latency are shorter than those of the duplication–comparison and standby–replacement systems because the failure can be masked and detected immediately by the voter and fault detection circuit, respectively.

2.2 TMR/BLV Architecture

The pure fault masking usually gives no warning of a deteriorating hardware state, so we design the fault detection circuit to provide the function of tolerating permanent faults and transient faults. The architecture of TMR/BLV system is shown in figure 2. The voter "V" is built up across the processor modules (P1/P2/P3) and memory modules (M1/M2/M3). There are two ERCs (Error–Reporting Circuit) for detecting errors. If the data are read from memory modules or I/O interfaces, the ERCm (for memory) is enabled and the ERCp (for processor) is disabled. On the contrary, if the data are written into memory modules or I/O interfaces, the ERCm is disabled and the ERCp is enabled. The C.vmp system [6] has three operational modes, and the switching between independent and voting modes allows the user to perform a performance/reliability tradeoff. Its unidirectional voter and bidirectional voter that use many multiplexers are more complicated than those of the proposed system.

A more detailed structure is shown in figure 3. We use the common clock generating method to synchronize the three processor modules. Each processor module consists of a clock generator, a microprocessor, an address latch, a data transceiver and a bus controller. Each memory module includes an address decoder (Dec), SRAM and EPROM. There are no triplicated devices on the data bus 2; that is, the I/O interface is on one module and connects data bus 2.

To diagnose transient faults quickly, which will be described in next section, we design a faulty address latch (FAL) to catch the faulty address during memory failure. FAL is put between address bus and data bus. V3 can mask the failure coming from three bus controllers, and a comparator (Comp) monitors the disagreement between input and output of V3. If the comparator detects a mismatch, a signal that is connected ERCp informs the interrupt controller to diagnose the faults. While the processor modules write data, the ERCp and ERCm can simultaneously supervise the operating V1a and V1d, respectively. Furthermore, if the processor modules read data from memory modules or I/O devices, the V2 operates and a comparator monitors the disagreements of V2 too.

3. DESIGN OF SYSTEM SOFTWARE

The hardware faults of digital computer systems may be classified into two broad categories: transient (soft) fault and permanent (hard) fault. Some studies of fault diagnosis indicate that most of digital system malfunctions are caused by

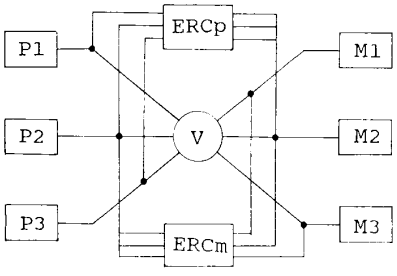

Fig. 2 Architecture of TMR/BLV.

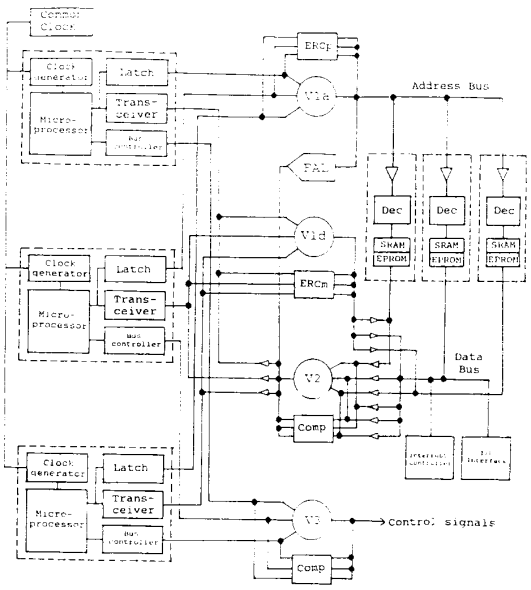

Fig. 3 Detailed architecture of TMR/BLV.

transient faults. In some systems 80 to 90 percent of the faults are estimated to be transient. The transient faults, though of short duration, may cause permanent damage. It is important that the damage caused by the transient faults is confined immediately and an appropriate retry procedure is performed. In time–critical applications such as process control or air–traffic control systems it is vital to detect the error as soon as possible.

In this paper we use software method to define a threshold number (T_i); if a consecutive retry number (C_i) is larger than the threshold number within a user–specified time interval (T), then the faulty module will be considered as permanent fault, and an on–line repair action will be issued; that is, if $C_i \geq T_i$ during the time $t \leq T$, then the faulty module is permanently faulty, otherwise the faulty module is in transient state. The flowchart of system software is shown in figure 4.

The system software reads a set of threshold numbers (T1, T2,T3) and then clears a set of retry numbers (C1,C2,C3) at the beginning of the retry procedure. If a failure occurs, there are three kinds of interrupt requests for retrying the failures. First, if it is a processor failure; that is, when the ERCp detects the failure, the ERCp will inform interrupt controller to run an interrupt service routine of CPU retry. This routine will "PUSH" and "POP" all the internal registers so that all the CPUs are forced to be consistent. This procedure is called as "processor consistency", which is very simple and will be introduced in next section. Second, if it is a ROM failure; that is, when the ERCm detects the failure, the ERCm will inform interrupt controller to run an subroutine of ROM retry. In this procedure the system reads faulty address (FA1) from port address of FAL, reads memory data from faulty address, then reads address data (FA2) from port address of FAL, and finally, checks if it is changed. With this method the transient faults can be diagnosed easily. Third, if it is a RAM failure; the RAM retry counter will be incremented by one. The faulty address will be latched by using the FAL. The CPU reads the faulty address (FA1) from the port address of FAL, writes the data (FF) to the faulty address, inverses the memory data of faulty address, then reads it again, and finally checks if it equals to (00). With this method it is also easy to diagnose the transient faults.

Furthermore, if the failure is a permanent one, the faulty condition will be displayed and alarmed. Since all the retry procedures of the TMR/BLV system are very simple, this

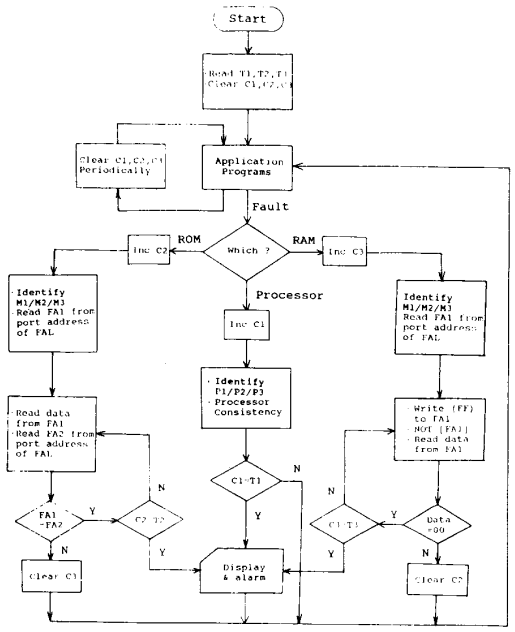

Fig. 4 The flowchart of system software.

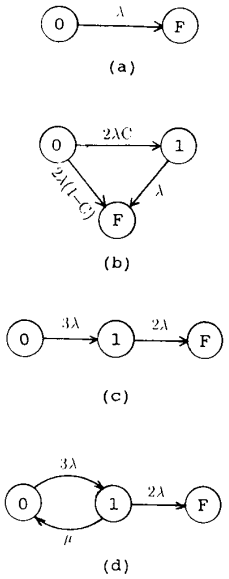

Fig. 5 Markov model for (a) simplex system, (b) duplex system, (c) TMR/SLV without on–line repair, and (d) TMR/SLV with on–line repair.

architecture has short retry time and will be demonstrated in next section.

4. EVALUATIONS

4.1 Reliability Analysis

The reliability analysis of any system is a key element in the design process. The Markov process model is usually used for analyzing the probabilistic system to evaluate the reliability [5,7]. For continuous–time Markov process, the length of time already spent in a state does not influence the probability distribution of the next state. The following will review the simplex system, duplex system, and TMR/SLV system, evaluate TMR/BLV system, and compare their reliability improvement factors (RIF) and mission time improvement factor (MTIF). With the comparison we can show the TMR/BLV system is more reliable than the other systems. In order that our analysis is more accurate and practical, we may consider the reliability of hard core,

$$R_h(t) = \exp(-\sum_{i=1}^{h} \lambda_i)t$$

where h is the number of components of hard core. We make the following assumptions :
(a) Each state is either working or failed.
(b) The failure rate and repair rate are constant.
(c) No common failure occurs.
(d) All modules of a system are good at time t=0.
(e) The repaired system acts as a new one.
(f) There is only one repair man to fix a faulty module.

4.1.1 Simplex System

For the simplex constant failure rate model, the system failure rate is the sum of the component failure rates. The Markov model of simplex system is shown in figure 5(a), where λ is the failure rate of the system module. There is no exit from the failed state.

<< State Descriptions >>
State 0: The simplex system is fault–free.
State F: The simplex system failed.

Therefore, the probability of state 0 at time t is $P_0(t) = \exp(-\lambda t)$, and the reliability of simplex system at time t is

$$R_s(t) = \Pr\{\text{No failure in time } [0,t]\} = \exp(-\sum_{i=1}^{s} \lambda_i)t \qquad (1)$$

where s is the number of components of system module and its mean time to failure is

$$MTTF_s = (\sum_{i=1}^{s} \lambda_i)^{-1} \qquad (2)$$

4.1.2 Duplex System

If the signals of duplex system are mismatched, the diagnostic routine will be issued to locate the errors, otherwise the results are sent to the outputs. With this scheme the coverage is a very sensitive factor for determining the system reliability [8] and the safety of duplex system is superior to that of standby approaches [3]. If the high coverage is required, the duplex system may have long suspending time. If the diagnostic procedure determines the faulty copy, which is then removed, the problem of rollback may take much time. As a result, the duplex system is not suitable for critical real–time applications. The figure 5(b) shows the Markov model of duplex system, where C is the detection coverage.

<< State Descriptions >>
State 0: All the modules are fault–free.
State 1: One of the module has failed.
State F: The duplex system failed.

Hence, the reliability of duplex system at time t is

$$\begin{aligned} R_d(t) = {} & \Pr\{\text{No failure or one module failure in the duplex system in time } [0,t]\} \cap \Pr\{\text{The hard core of duplex system is fault–free in time } [0,t]\} \\ = {} & 2C \exp(-\sum_{i,j=1}^{d,h} (\lambda_i+\lambda_j))t + \\ & (1-2C) \exp(-\sum_{i,j=1}^{d,h} (2\lambda_i+\lambda_j))t \end{aligned} \qquad (3)$$

where d is number of components of the duplex system. And its mean time to failure is

$$MTTF_d = 2C(\sum_{i,j=1}^{d,h}(\lambda_i+\lambda_j))^{-1} + (1-2C)(\sum_{i,j=1}^{d,h}(2\lambda_i+\lambda_j))^{-1} \quad (4)$$

4.1.3 TMR/SLV System

The TMR system, also known as masking redundancy, uses extra modules so that the effect of a faulty module is masked instantaneously. If a TMR system does not equipped with ERC for fault detection, then the effects of faults are automatically neutralized without notifying of their occurrence. Usually the TMR system is voted by system level, such as the August system [1] which has three control computer modules, a distributor and an unidirectional voter that votes on the I/O interface.

At first, we consider the Markov model of the TMR/SLV without on–line repair shown as figure 5(c).

<< State Descriptions >>
State 0: All the modules are fault–free.
State 1: One of the system module has failed.
State F: The overall system failed in this state.

As a result, the reliability of TMR/SLV system without on–line repair at time t is

$$R_{sn}(t) = \Pr\{\text{No failure or one module failure in the TMR/SLV in time } [0,t]\} \cap \Pr\{\text{The hard core of TMR/SLV system is fault–free in time } [0,t]\}$$
$$= 3\exp(-\sum_{i,j=1}^{s1,h}(2\lambda_i+\lambda_j))t - 2\exp(-\sum_{i,j=1}^{s1,h}(3\lambda_i+\lambda_j))t \quad (5)$$

where s_1 is number of components of the TMR/SLV without repairing capability. And its mean time to failure is

$$MTTF_{sn} = 3(\sum_{i,j=1}^{s1,h}(2\lambda_i+\lambda_j))^{-1} - 2(\sum_{i,j=1}^{s1,h}(3\lambda_i+\lambda_j))^{-1} \quad (6)$$

Next, the TMR/SLV system with on–line repair is considered and its Markov model is shown in figure 5(d), where μ is the repair rate. If one of the module fails, but a repairman fixs it before two of the three modules fails, then the system is reliable. Consequently, the reliability of TMR/SLV system with on–line repair at time t is

$$R_{sw}(t) = \Pr\{\text{No failure or one module failure in the TMR/SLV/Repair in time } [0,t]\} \cap \Pr\{\text{The hard core of TMR/SLV/Repair system is good in time } [0,t]\}$$
$$= \frac{A+B}{2B}\exp(-1/2\sum_{j=1}^{h}(A-B+2\lambda_j))t - \frac{A-B}{2B}\exp(-1/2\sum_{j=1}^{h}(A+B+2\lambda_j))t \quad (7)$$

where $A = \mu+5\sum_{i=1}^{s2}\lambda_i$ and $B = ((\mu^2+10\mu\sum_{i=1}^{s2}\lambda_i+(\sum_{i=1}^{s2}\lambda_i)^2)^{1/2}$, and s_2 is the number of components of the TMR/SLV with repairing capability. And its mean time to failure is

$$MTTF_{sw} = \frac{A+B}{2B}(1/2\sum_{j=1}^{h}(A-B+2\lambda_j))^{-1} - \frac{A-B}{2B}(1/2\sum_{j=1}^{h}(A+B+2\lambda_j))^{-1} \quad (8)$$

4.1.4 TMR/BLV System

The following discusses the reliability block diagram (RBD) of the TMR/BLV system to help analyzing of it's Markov model. The hard core connects the memory modules and processor modules serially so that the overall system will fail while the hard core fails. The TMR/BLV system requires at least two processor modules and two memory modules for the survival to work correctly. The BV performs the voting operation and the ERC detects the errors concurrently with the BV. The RBD of the TMR/BLV is shown in figure 6 and the Markov model of TMR/BLV is shown in figure 7. Since it is voted on the bus level, each individual module is either a processor module or a memory module.

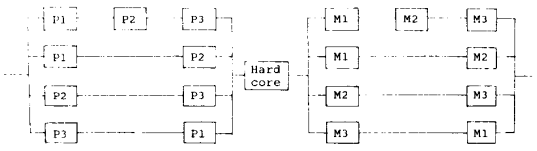

Fig. 6 Reliability block diagram of TMR/BLV.

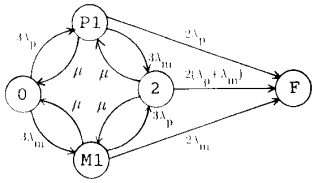

Fig. 7 Markov model of TMR/BLV.

<< State Descriptions >>
State 0: All the modules are fault–free.
State P1: One of the processor module has failed.
State M1: One of the memory module has failed.
State 2: One of the processor module and one of the memory module have failed.
State F: The TMR/BLV system fails.

<< State Equations >>

$$\dot{P}_0(t) = -3(\lambda_P+\lambda_m)P_0(t)+\mu P_{p1}(t)+\mu P_{m1}(t)$$
$$\dot{P}_{p1}(t) = 3\lambda_P P_0(t)-(\mu+3\lambda_m+2\lambda_P)P_{p1}(t)+\mu P_2(t)$$
$$\dot{P}_{m1}(t) = 3\lambda_m P_0(t)-(\mu+3\lambda_P+2\lambda_m)P_{m1}(t)+\mu P_2(t)$$
$$\dot{P}_2(t) = 3\lambda_m P_{p1}(t)+3\lambda_P P_{m1}(t)-2(\mu+\lambda_P+\lambda_m)P_2(t)$$
$$\dot{P}_f(t) = 2\lambda_P P_{p1}(t)+2\lambda_m P_{m1}(t)+2(\lambda_P+\lambda_m)P_2(t)$$

where λ_P and λ_m are the failure rate of processor module and memory module, respectively. Hence, the reliability of TMR/BLV system at time t is

$$R_b(t) = \Pr\{(\text{No failure}) \text{ or } (\text{one processor module failure}) \text{ or } (\text{one memory module failure}) \text{ or } (\text{one processor module and one memory module failure}) \text{ in the TMR/BLV in time } [0,t]\} \cap \Pr\{\text{The hard core of TMR/BLV is fault–free in time } [0,t]\} = (1-P_f(t))\exp(-\sum_{i=1}^{h}\lambda_i)t \quad (9)$$

It is difficult to solve the above state equations into analytic forms. But the graphical representation can directly be used to describe a system by a set of state equations.

4.1.5 Predictions of Failure Rate and System Comparisons

The failure rate prediction is very important to the system reliability. In general, the MIL–HDBK–217E model [9,10] is widely used for the evaluation of failure rates in the electronic equipments. In the proposed system we use this model to estimate the failure rates of system module, hard core, memory module and processor module for the evaluation of reliability. There are two failure rate prediction techniques presented in the MIL–HDBK–217E model: (1) parts count reliability prediction, this method is applicable during bid proposal and early design phases. (2) part stress reliability prediction, this method is applicable when most of the design is completed and a detailed parts list including part stresses is available. Since there are many factors to be referred, the result should be more accurate than that of using the parts count method. As a result the failure rate of electronic device in this study is predicted by this method. The failure rate of electronic device is

$\lambda_P = \pi_q(C_1\pi_t\pi_v+C_2\pi_e)\pi_l$ (failures per million hours) (10)

where π_q is the quality factor, π_t is the temperature acceleration factor, π_v is the voltage stress derating factor, π_e is the application environment factor, C_1 is the circuit complexity factor, C_2 is the package complexity failure rate and π_l is the device learning factor. Since the failure rate of system module depends on many informations, the authors made the following reasonable assumptions:

(a) CPU : The Intel 8086 contains approximately 29,000 transistors and is fabricated using the HMOS technology.
(b) Memory : The static RAM includes 80K bytes and uses ten HM6264. The EPROM also includes 80K bytes and uses five TMS27128.
(c) Application environment : ground, fixed (G_f); π_e=2.5.
(d) D–1 class quality; π_q=20.
(e) Learning factor π_l=1.
(f) Maximum supply voltage rating : V_{dd}=5 Volts; that is, the voltage stress derating factor π_v=1.
(g) The repair rate μ is 1/24; that is, one time a day.

From above assumptions all the factors relative to various electronic device and the failure rate that can be calculated by Eq.(10) are listed in Table 1. We substitute the failure rates of various devices into the equation (1),(3),(5),(7), and (9). And all cases of the system reliability are plotted in the figure 8.

Furthermore, when comparing a highly reliable system with a simplex system the reliability improvement factor at time t [RIF(t)] is proposed. It is a reliability measure estimating how well the redundant system improves with respect to simplex system and is defined as $[1-R_s(t)]/[1-R_r(t)]$, where $R_s(t)$ is the reliability of simplex system at time t and

TABLE 1
Failure Rate Predictions (failures per million hours)

	Function	IC	Number	C_1	π_l	C_2
System Module	Clock generator	8284	1	0.02	1.4	0.0064
	Microprocessor	8086	1	0.06	11.0	0.015
	Latch	74LS374	3	0.01	0.55	0.0071
	Transceiver	74LS245	2	0.01	1.00	0.0071
	Bus controller	8288	1	0.02	2.7	0.0071
	Interrupt controller	8259	1	0.02	1.00	0.01
	I/O Interface	8255	1	0.02	1.6	0.015
	Decoder	ASIC	1	0.01	0.93	0.0064
	SRAM	HM6264	10	0.2	1.3	0.01
	EPROM	TM27128	5	0.24	0.89	0.01
	Total Failure Rate λ_{sm}=102.4					
Processor Module	Clock generator	8284	1	0.02	1.4	0.0064
	Microprocessor	8086	1	0.06	11.0	0.015
	Latch	74LS374	3	0.01	0.55	0.0071
	Transceiver	74LS245	2	0.01	1.00	0.0071
	Bus controller	8288	1	0.02	2.7	0.0071
	Total Failure Rate λ_p=18.8					
Memory Module	SRAM	HM6264	10	0.2	1.3	0.01
	Decoder	ASIC	1	0.01	0.93	0.0064
	EPROM	TM27128	5	0.24	0.89	0.01
	Total Failure Rate λ_m=81.4					
Hard Core	Clock generator	8284	1	0.02	1.4	0.0064
	Interrupt controller	8259	1	0.02	1.00	0.01
	V1,V2,V3,ERC	ASIC	4	0.02	0.85	0.025
	FAL	ASIC	1	0.01	0.48	0.009
	I/O Interface	8255	1	0.02	1.6	0.015
	Total Failure Rate λ_h=10.1					

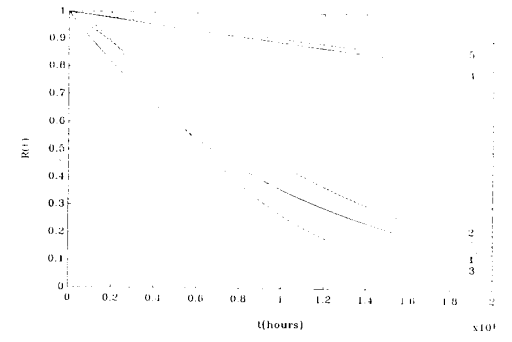

Fig. 8 Reliability of (1) simplex system,(2) duplex system,(3) TMR/SLV without on–line repair,(4)TMR/SLV with on–line repair, and (5) TMR/BLV.

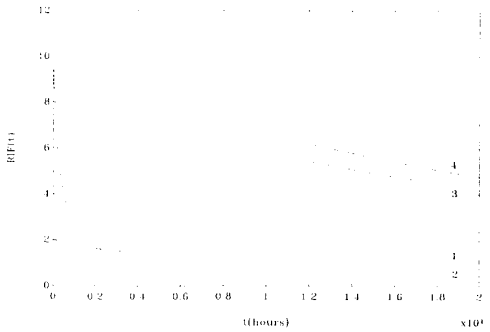

Fig. 9 RIF(t) of (1)duplex system, (2)TMR/SLV without on–line repair, (3)TMR/SLV with on–line repair, and (4)TMR/BLV, with respect to simplex system.

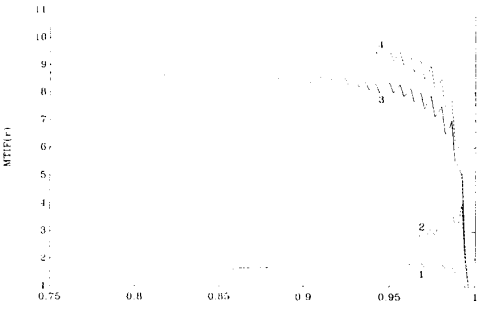

Fig. 10 MTIF(r) of (1)duplex system, (2)TMR/SLV without on–line repair, (3) TMR/SLV with on–line repair, and (4)TMR/BLV, with respect to simplex system.

$R_r(t)$ is the reliability of redundant system at time t. The figure 9 shows the RIF(t) of various systems.

Finally, for a given reliability r the mission time improvement factor [MTIF(r)] is also widely used to evaluate reliability improvement. It is defined as $MT_r(r)/MT_s(r)$ for a given reliability r, where MT_r and MT_s are the mission time of redundant system and simplex system, respectively. The mission time of simplex system for a given reliability r is $MT_s(r) = -(\ln r)/\lambda$. The figure 10 shows the MTIF(r) of various systems with respect to the simplex system. It is important that the $MTIF_{sw}(r)$ and $MTIF_b(r)$ are almost independent of a given reliability r.

4.2 Detection Coverage

Detection coverage is the probability that a fault is successfully detected before a complete system corruption occurs. If the system is required of higher coverage, the diagnostic time is usually longer under the same algorithm; i.e., the system suspending time is longer. The coverage of duplex system and standby system usually range from 0.8 to 0.9 [5], but the coverage of TMR/SLV system is higher than that of the duplex system and that of standby system because the errors in the TMR/SLV system can not be detected only when : (1) both two modules fail, (2) all three modules fail, and (3) the hard core fails. Hence the coverage of TMR/SLV is nearly perfect; that is, if the system works correctly, the errors must be immediately detected by the ERC. The TMR/BLV system also has high coverage because an error will be detected while the complete system operates properly.

If we do not consider that the detection coverage is perfect in the TMR/SLV and TMR/BLV system, their Markov models are slightly different. Although these types of Markov model are more sophisticated, they are more accurate. Figure

11 (a) and (b) show the Markov models of TMR/SLV and TMR/BLV, respectively, where C_s is the detection coverage of TMR/SLV, C_p is the detection coverage of ERC_P of TMR/BLV, and C_m is the detection coverage of ERC_m of TMR/BLV. The state equations of TMR/SLV system involving detection coverage are:

$$\dot{P}_0(t) = -3\lambda P_0(t)+\mu P_1(t)$$

$$\dot{P}_1(t) = 3\lambda C_s P_0(t)-(\mu+2\lambda)P_1(t)$$

$$\dot{P}_f(t) = 3\lambda(1-C_s)P_0(t)+2\lambda P_1(t)$$

The state equations of TMR/BLV system involving detection coverage are:

$$\dot{P}_0(t) = -3(\lambda_P C_P+\lambda_m C_m+\alpha)P_0(t)+\mu P_{P1}(t)+\mu P_{m1}(t)$$

$$\dot{P}_{P1}(t) = 3\lambda_P C_1 P_0(t)-(\mu+\beta+3\lambda_m C_m)P_{P1}(t)+\mu P_2(t)$$

$$\dot{P}_{m1}(t) = 3\lambda_m C_m P_0(t)-(\mu+\sigma+3\lambda_P C_P)P_{m1}(t)+\mu P_2(t)$$

$$\dot{P}_2(t) = 3\lambda_m C_m P_{P1}(t)+3\lambda_P C_P P_{m1}(t)-(2\mu+\gamma)P_2(t)$$

$$\dot{P}_f(t) = \alpha P_0(t)+\beta P_{P1}(t)+\sigma P_{m1}(t)+\gamma P_2(t)$$

where $\alpha = 3[\lambda_P(1-C_P)+\lambda_m(1-C_m)]$, $\beta = 2\lambda_P+3\lambda_m(1-C_m)$, $\gamma = 2(\lambda_P+\lambda_m)$, and $\sigma = 2\lambda_m+3\lambda_P(1-C_P)$. The figure 12 shows the R(t) of these two models.

4.3 Error Latency

The error latency, which is illustrated in the figure 13, is the time between the occurrence and the detection of the fault. In general, the TMR/SLV system detects the error while it performs an I/O operation; that is, the CPU must perform many memory accesses (one access requires several bus cycles),

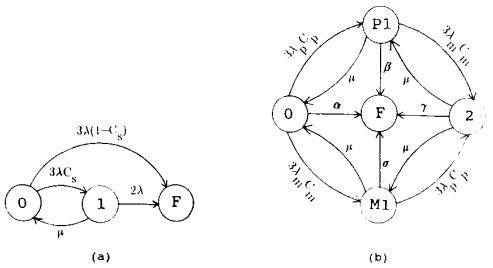

Fig. 11 Markov model of (a)TMR/SLV and (b)TMR/BLV involving detection coverage.

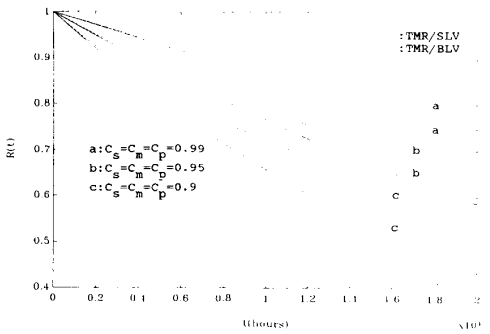

Fig. 12 Reliability of TMR/SLV and TMR/BLV for various detection coverage.

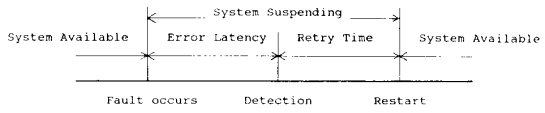

Fig. 13 Scenario for the error latency and retry time.

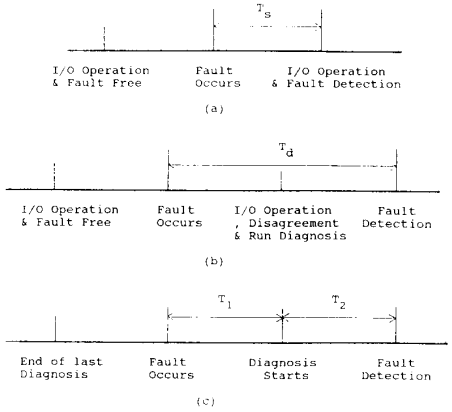

Fig. 14 Error latency of the (a) TMR/SLV, (b) duplex system, and (c)standby system.

and then one I/O operation may be issued by the processor modules. Since the errors can be detected at the system level (or I/O level), the time interval of error latency of the TMR/SLV system T_s can be illustrated in figure 14(a), where $T_s \equiv K_s T$; T is the time interval of one bus cycle and K_s is a latency factor of the TMR/SLV.

The duplex system detects the error by means of a comparator. If the output results mismatch, the disagreement signal will be sent to the interrupt controller which performs an diagnostic program to detect the errors and to locate the faulty copy. Consequently, the results are compared at the I/O operation and the fault is detected during the diagnostic period. The time interval of error latency of the duplex system, T_d, is illustrated in figure 14(b), where $T_d \equiv K_d T$ and K_d is a latency factor of the duplex system.

The standby system also detects the error during the diagnostic period. In general, there are three methods to detect a fault in an individual module of the standby system: (1) Periodic tests: The normal operation of the functional module is suspended and the diagnostic routine is operated to determine if errors are present in the module. (2) Self–checking circuits: This method has an advantage of short testing time, but it has an overhead of the cost and the system reliability may be reduced. (3) Watchdog timer, the approach is cost–effective and is a popular method of fault detection. Therefore, the time interval of error latency of the standby system T_{ss} can be illustrated in figure 14(c), and $T_{ss} = T_1+T_2 \equiv K_{ss}T$, where K_{ss} is a latency factor of the standby system.

The TMR/BLV system detects the errors on the bus level by means of the ERCs, that is, the errors can be detected during one bus cycle time. Hence the time interval of error latency of the TMR/BLV system T_b equals to T. In the Intel's 8086 CPU, if the "no wait state" is used, the time of one bus cycle equals to four clock cycles, hence T=0.84 μs for the system clock rate of 4.77Mhz. From above discussions, We conclude : $T_b << T_s << T_d < T_{ss}$.

4.4 Retry Time

Usually the retry process can determine whether the error is a transient fault or a permanent fault, and the retry time,

which is also illustrated in figure 13, is the time from the error detection, resynchronization, to the system restart. It is noted that the system suspends during the retry period, consequently, if it belongs to the real–time application, this period should be as short as possible in order to have faster response. The two retry procedures of the TMR/BLV system are stated as follows: (1) CPU retry procedure: If the CPUs write a word into a memory or an I/O port, the V1 is active and the ERCp detects the errors concurrently with the V1. At this time if there is an error, the CPU retry procedure is performed for the register consistency. Because in this system there exists a BV, the contents of all registers in the CPUs are stored in through the BV by PUSH instruction. After that, the contents of the registers are restored again from the memory buffer by POP instruction. We may see that the process, which is shown in figure 15(a), is easy significantly and requires about 100 μs with the system clock rate of 4.77Mhz. (2) Memory retry procedure: If the CPUs read a word from the memory module, the V2 is active and the ERCm detects the error concurrently with the V2. At this time if there exists an error in this system, the memory retry procedure is performed for the memory consistency. Because there is a faulty address latch (FAL) on the address bus and a BV in the TMR/BLV system, the faulty address can be latched and the system can read the data word from the latched address, and then writes into the same address, therefore, the contents of memory will be in agreement. This process, which need not be required to retry all the memory addresses is also very simple and shown in figure 15(b).

Since the time interval of system suspending during the error occurrence equals to the that of error latency plus the that of retry time, the system suspending time is short during the system failure and the TMR/BLV is suitable for the real–time industry applications.

```
                PUSH    AX
                PUSH    BX
                PUSH    CX
                PUSH    DX
                PUSH    SP
                PUSH    BP
                PUSH    SI
                PUSH    DI
                PUSH    DS
                PUSH    SS
                PUSH    ES
                PUSHF
                CALL    CS_IP
CS_IP:          POP     AX
                POPF
                POP     ES
                POP     SS
                POP     DS
                POP     DI
                POP     SI
                POP     BP
                POP     SP
                ADD     SP,#02
                POP     DX
                POP     CX
                POP     BX
                POP     AX
                RET
```

(a)

```
        IN      AX,PORT_OF_FAL          ;STORE FAULTY ADDRESS INTO AX
        MOV     BX,AX                   ;STORE DATA OF AX INTO BX
        MOV     SI,AX                   ;STORE DATA OF AX INTO SI
        MOV     CL,#00                  ;SET RETRY_NUMBER TO 0
RETRY:  MOV     DL,[SI]                 ;READ DATA OF FAULTY ADDRESS
        IN      AX,PORT_OF_FAL          ;READ FROM PORT_OF_FAL
        CMP     AX,BX                   ;CHECK IF FAIL AGAIN
        JNE     RETURN                  ;IF ERROR DISAPPEAR,RETURN
        INC     CL                      ;IF ERROR,RETRY_NUMBER + 1
        CMP     CL,THRESHOLD_NUMBER     ;CHECK IF CL THRESHOLD_NUMBER
        JB      RETRY                   ;IF BELOW,RETRY AGAIN
        CALL    DISPLAY_ALARM           ;PERMANENT FAULT OCCURE,ALARM
RETURN: MOV     [SI],DL                 ;STORE DATA TO FAULTY ADDRESS
        RET                             ;RETURN TO CALLING PROGRAM
```

(b)

Fig. 15 Assembly codes for (a) CPU retry and (b) Memory retry.

5. CONCLUSIONS

Generally a transaction processing computers can recover from a fault in several seconds and then continue its normal operation. But when it was put into an industry application environments, where fault retry time must be within milliseconds [11], this type of computer is not satisfactory. With the help of ERC and simple PUSH and POP instructions, the content of CPU's registers can be consistent, and the FAL can latch the faulty address to detect errors, thus, the retry time is only about 100 μs with the clock rate of 4.77Mhz in the microprocessor system. Because this proposed TMR scheme is voted on bus level, the failure can be detected quickly and it require only 0.84 μs with the same clock rate. Furthermore, the reliability of TMR/BLV system is higher than that of TMR/SLV system regardless of the detection coverage and its MTIF(r) with respect to the simplex system approximates to 10. In conclusion, this proposed system is proved to have high reliability and very suitable for the controller of real–time industry applications.

REFERENCES

[1] John H.Wensley,"Industrial–control system does things in threes for safety", Electronics, January 1983.

[2] Hector P. Polenta, Asok Ray and John A. Bernard, "Microcomputer–based fault detection using redundant sensors", IEEE Transactions on Industry Applications, September/October, 1988.

[3] Yen–Tseng Hsu and Chen–Fa Hsu,"Evaluation of reliability and safety of the long–life unmaintained computer systems", Accepted, in press, International Journal of Electronics, 1991.

[4] Yen–Tseng Hsu and Chen–Fa Hsu,"A novel model of intermittent faults for reliability and safety measures in the long–life computer systems", Accepted, in press, International Journal of Electronics, 1991.

[5] D.P.Siewiorek and R.S.Swarz,"The theory and practice reliable system design", 1982, Digital press.

[6] D.P.Siewiorek,"A case study of C.mmp, Cm* and C.vmp: part1, experiences with fault tolerance in multiprocessor systems", International Symposium of Fault Tolerant Computing,1978.

[7] K. Takaragi, R. Sasaki and S. Shingai, "A method of rapid markov reliability Calculation", IEEE Transactions on Reliability, August 1985.

[8] T. F. Arnold, "The concept of coverage and its effect on the reliability model of a repairable system", IEEE Transactions on Computer, March 1973.

[9] "MIL–HDBK–217E, reliability prediction of electronic equipment", Department of Defense, USA, October 1986.

[10] M. Pecht and W. C. Kang, "A critique of MIL–HDBK–217E reliability prediction methods", IEEE Transactions on Reliability, December 1988.

[11] R. Allan, "For fault–tolerant computing, software is finding – a powerful ally in hardware", Electronic Design, October 1985.