Comparing Permanent and Transient Fault Tolerance of Multiple-core based Dependable ECUs

Masashi Imai, Tomohiro Yoneda

Comparing Permanent and Transient Fault Tolerance of Multiple-core based Dependable ECUs

Masashi Imai
Hirosaki University
E-mail: miyabi@eit.hirosaki-u.ac.jp

Tomohiro Yoneda
National Institute of Informatics
E-mail: yoneda@nii.ac.jp

***Abstract*—Several dependability improvement methods using multiple-core based systems have been proposed. In this paper, the *lock-step pair* scheme that is popular in car companies these days, the traditional *TMR-spare* scheme, and our proposed *DTTR (Duplication with Temporary TMR and Reconfiguration)* scheme are modeled using the Markov chains considering both permanent and transient faults, and are compared in the viewpoint of the average system failure rate.**

***Keywords*-Dependable ECU; Reliability; Duplication with Temporary TMR and Reconfiguration; Markov chain**

I. Introduction

It has been recognized that multiple-core systems like chip multiprocessor (CMPs) and multi-processor system-on-a-chip (MPSoC), which integrate multiple processor cores and intellectual property cores in a single chip, can be used not only for performance improvement, but also for dependability improvement [1–5]. Actually, car companies use tightly-coupled dual microprocessors with a comparator, which are called *lock-step components*, in order to implement highly reliable ECUs (Electronic Control Units). The lock-step components can detect errors, but cannot tolerate faults. Thus, a pair of lock-step components (called a *lock-step pair* in this paper) is also used to tolerate one processor fault. The lock-step pair scheme may work for small dedicated ECUs, but is not so suitable, due to its highly redundant and fixed configurations, for ECUs or a set of ECUs that perform more complicated functions, for example, needed for autonomous cars.

Traditionally, a TMR (Triple Modular Redundancy) scheme has been used in order to implement highly reliable systems like airplane control systems. For multiple-core systems, several trios and hot spare cores are configured, and the trios perform application tasks in a TMR manner. When an error is detected, a dynamic reconfiguration method in which the faulty core is replaced with one of the spare cores is applied. This scheme (called a *TMR-spare* in this paper) is suitable for ECUs that perform complicated functions as mentioned above, while a specific reconfiguration mechanism is needed.

We have proposed another dependable scheme called *DTTR (Duplication with Temporary TMR and Reconfiguration)* for highly reliable platforms that is applicable to systems where many ECUs work in a coordinated manner. Several related basic ideas have been reported in [5–7]. In our dependable task execution scheme, each application task is loaded in several processor cores redundantly and statically, and usually two processor cores execute the same task simultaneously using the same inputs. Then, the results of the task are compared. If a mismatch is found, the task is executed again, but using three processor cores, to find the correct results and detect a faulty core, which is called *a temporary TMR*. If a faulty core is successfully detected, it is excluded from the system, and tasks are continuously executed on a reconfigured system.

Recently, it has also been recognized that the probability of occurrence of transient faults becomes significantly large as the VLSI fabrication technology shrinks. The former two schemes can tolerate transient faults more efficiently by slightly modifying the schemes as follows.

In the lock-step pair scheme, when no permanent faults exist, a transient fault (as well as a permanent fault) can be masked since one of two lock-step components can provide a matched result. However, in a situation where a permanent fault already occurred, another (transient or permanent) fault on the remaining lock-step component cannot be masked. In this case, if the unmatched task is executed again on the same remaining lock-step component, a transient fault can be masked. The lock-step pair scheme with this re-execution is called the *modified lock-step pair* scheme in this paper.

In the TMR-spare scheme, a transient fault that occurs in some trio can be masked (as well as a permanent fault). Note that a dynamic reconfiguration to exclude the faulty core should not be performed immediately to avoid wasting a processor core in case of a transient fault. Since our applications are periodic control programs, the same trio is used for executing the same task within a short period. If an error is detected again there, then the fault is considered to be permanent, and the dynamic reconfiguration is performed. Since it is reasonable to assume that the probability that two or more permanent faults occur within a short period is very low, this approach should work fine. On the other hand, when the number of remaining cores is only two, and no trio can be configured, those two cores can still perform tasks in a DMR (Dual Modular Redundancy) manner, but no transient fault can be masked (*original TMR-spare*). In this case, similarly to the modified lock-step pair scheme, the unmatched task can be executed again using the same cores in order to tolerate a transient fault. We call it the *modified TMR-spare* scheme.

On the contrary, in DTTR scheme, a permanent fault or a transient fault can be masked without any modification, since an unmatched task is always executed again in the temporary TMR or DMR [1]. Note that the above modified two schemes as well as DTTR scheme need an additional time slot for the task re-execution. This influence on the system reliability may not be ignored, because real-time

[1]When the number of the remaining cores is two, the temporary TMR is degraded to DMR.

applications are targeted in our research project. However, this issue is left for the future work.

In this paper, the quantitative comparison among the above five (the original and modified lock-step pair and TMR-spare, and DTTR) schemes is discussed based on Markov models [8], considering both permanent and transient faults.

II. Markov models and Reliability

A. Evaluation Assumptions

We assume the following requirements for the fair comparison.

- The number of processor cores is the same among the five schemes. It is assumed to be a multiple of 4 since it is required for the lock-step pair schemes. Our evaluation uses 4 and 8 for it.
- Each processor core has a sufficient amount of local memory so that it can load several tasks statically. The amount of the local memory is the same among the five schemes.
- There is a unit that performs several fault management functions such as the comparison of the results obtained by processor cores and the data dispatch to appropriate processor cores based on the faulty processor core information. This unit is denoted by a *diagnostics and reconfiguration (DnR) component* in this paper.
- Some high-speed low-latency communication mechanism between the DnR component and the processor cores is available. An NoC based approach is suitable since it scales well. Furthermore, that communication mechanism has suitable dependability.
- The failure rate of a processor core which represents how often a permanent fault occurs is λ. The failure rate of a transient fault is λ_{tr}. The (permanent) failure rates of the DnR components are also considered. Those for DTTR, the lock-step pair, and the TMR-spare schemes are λ_D, λ_L, and λ_T, respectively, where $\lambda_{tr} > \lambda \gg \lambda_D \approx \lambda_T > \lambda_L$. Here, the coverage of the DnR components can be defined as $C_D = e^{-\lambda_D t}$, $C_T = e^{-\lambda_T t}$, and $C_L = e^{-\lambda_L t}$, respectively.
- For simplifying the analysis, our application program consists of 8 tasks. The reliability of the above schemes depends on the number of copies of those tasks which are statically stored in local private memories. When the number of processor cores is 4, it is assumed that the number of task copies is 4, and when the number of processor cores is 8, it is assumed that it is 4 or 8. Note that in case that the numbers of processor cores and task copies are the same, each processor core has all the tasks.
- The acceptable maximum number of accumulated faulty cores is M. That is, when the (M+1)-th permanent fault occurs, it is reported with an alert, and the system is stopped for maintenance. In this paper, we use $M = 2$ for the evaluation.

B. Systems with 4 processor cores

Figure 1 shows the abstracted models of the above five schemes with 4 processor cores. As for the original and modified lock-step pair schemes, it is the minimum configuration. In this figure, rectangles labeled with Pn represent

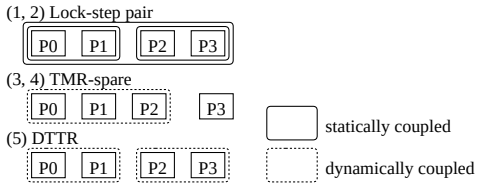


Figure 1. Multiple-core platform models with 4 processor cores.

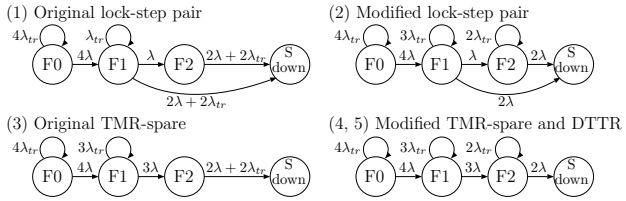


Figure 2. Markov models with 4 processor cores.

processor cores. Rounded rectangles with a solid line and a dotted line represent a statically coupled component and a dynamically coupled component, respectively. In the original and modified lock-step pair schemes, P_{2n} is statically coupled with P_{2n+1} to form a lock-step component for $n \geq 0$. Then, P0, P1, P2, and P3 statically compose a lock-step pair. In the original and modified TMR-spare schemes, a trio initially consists of three processor cores P0, P1, and P2, and the remaining processor core P3 is used as a hot spare processor core. In DTTR scheme, P_{2n} is initially coupled with P_{2n+1} to form a DMR for $n \geq 0$. In the TMR-spare and DTTR schemes, those initial configurations dynamically change to others according to the occurrence of faults.

In this paper, the Markov chain [8] is used for the evaluation. It is assumed that the probability of a given state transition in Markov models depends only on the current state. Figure 2 shows the markov models of the five schemes. In this figure, "Fn" represents that n processor cores have become permanently faulty. "S down" represents that the system is down because it cannot produce any correct output results. Under $M = 2$, the systems go down before being stopped for maintenance. Note that an arc labeled with λ_L (λ_T or λ_D) from each state to the down state, which represents the DnR component failure, is omitted in this figure for simplicity.

In the original lock-step pair scheme, when one lock-step component has a permanently faulty processor core, neither a transient nor permanent fault on the remaining lock-step component cannot be masked. Thus, in "F1" state, such a transient or permanent fault causes the system down. This transition is shown as an arc from "F1" to "S down" with probability $2\lambda + 2\lambda_{tr}$ in Figure 2 (1). On the other hand, in the modified lock-step pair scheme, a transient fault can be masked by the re-execution in "F1". Thus, the markov model is represented as shown in Figure 2 (2). "F2" is a state where both processor cores are permanently faulty in one lock-step component. In the original TMR-spare scheme, a transient fault cannot be masked when only two processor cores remain. This transition is shown as an arc from "F2" to "S down" with probability $2\lambda+2\lambda_{tr}$ in Figure 2 (3). In both the modified TMR-spare and DTTR schemes, a transient fault can be masked as long as the number of remaining

cores is two or more. Therefore, their markov models are the same as shown in Figure 2 (4, 5).

From these markov models, the reliability of each scheme is obtained as follows;

(1) Original lock-step pair scheme

$$R(t) = e^{-\lambda_L t}\left(\frac{2\lambda}{\lambda-\lambda_{tr}}e^{-(2\lambda+2\lambda_{tr})t} - \frac{\lambda+\lambda_{tr}}{\lambda-\lambda_{tr}}e^{-4\lambda t}\right)$$

(2) Modified lock-step pair scheme

$$R(t) = e^{-\lambda_L t}\left(2e^{-2\lambda t} - e^{-4\lambda t}\right)$$

(3) Original TMR-spare scheme

$$R(t) = e^{-\lambda_T t}\left(\frac{6\lambda^2}{(\lambda-\lambda_{tr})(\lambda-2\lambda_{tr})}e^{-(2\lambda+2\lambda_{tr})t} - \frac{8(\lambda+\lambda_{tr})}{\lambda-2\lambda_{tr}}e^{-3\lambda t} + \frac{3(\lambda+\lambda_{tr})}{\lambda-\lambda_{tr}}e^{-4\lambda t}\right)$$

(4) Modified TMR-spare scheme

$$R(t) = e^{-\lambda_T t}\left(6e^{-2\lambda t} - 8e^{-3\lambda t} + 3e^{-4\lambda t}\right)$$

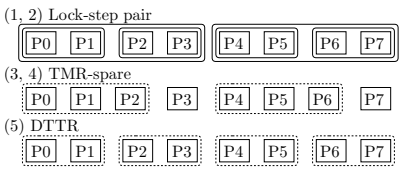
(5) DTTR scheme

$$R(t) = e^{-\lambda_D t}\left(6e^{-2\lambda t} - 8e^{-3\lambda t} + 3e^{-4\lambda t}\right)$$

It can be easily confirmed that the reliability of the modified lock-step pair (TMR-spare, resp.) scheme is the same as that of the original lock-step pair (TMR-spare) scheme when $\lambda_{tr} = 0$. The quantitative comparison is presented later.

C. Systems with 8 processor cores

Figure 3 (a) shows the abstracted models of the systems with 8 processor cores. As shown in Figure 3 (a), at the initial state, the lock-step pair schemes and the TMR-spare schemes can perform two tasks simultaneously, while DTTR scheme can perform four tasks simultaneously. When the number of task copies is 4, the task allocation table is assumed as shown in Figure 3 (b) for all the schemes.



(a) Comparison model

	P0	P1	P2	P3	P4	P5	P6	P7
T_0	x	x	x	x				
T_1					x	x	x	x
T_2	x	x	x	x				
T_3					x	x	x	x
T_4	x	x	x	x				
T_5					x	x	x	x
T_6	x	x	x	x				
T_7					x	x	x	x

(b) Task allocation table (*#copy* = 4)

Figure 3. Multiple-core platform models with 8 processor cores.

Figure 4 shows the markov models when the number of task copies is 4. In this figure, the numbers after "F" in each node represent how many processor cores have been permanently faulty in either P0 $\cdots$ P3 or P4 $\cdots$ P7 (e.g., F02 represents both the case where 2 permanent faults are in P0 $\cdots$ P3 and no permanent fault in P4 $\cdots$ P7 and the case where 2 permanent faults are in P4 $\cdots$ P7 and no permanent fault in P0 $\cdots$ P3). Compared with Figure 2, "S repair" state is added. This state means that the system is stopped for maintenance. In this evaluation, this state is also considered to be a down state. The equations of their reliability are omitted due to the limited space.

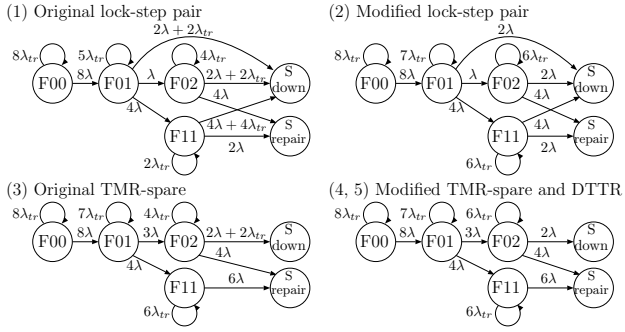


Figure 4. Markov models with 8 processor cores (#copy is 4).

When the number of task copies is 8, the markov models of the original and modified lock-step pair schemes are the same as Figure 4 (1) and (2) respectively, since the behavior of the lock-step pair schemes only depends on the initial active copy assignment and a dynamic reconfiguration is not performed. On the other hand, in the TMR-spare and DTTR schemes, the dynamic reconfiguration is applied considering the faulty processor core information. Furthermore, $M = 2$ is assumed. Thus, "S down" state is not reached, and a transient fault can be completely masked. Hence, their markov models are obtained as shown in Figure 5.

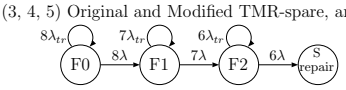


Figure 5. Markov models with 8 processor cores (#copy is 8).

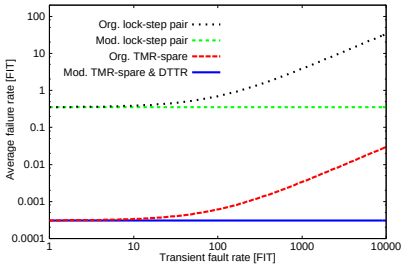
III. Reliability Comparison

We evaluate the influence of occurrence of transient faults on the system reliability. It is assumed that the permanent failure rate of a processor core and the mission time (*MT*) are 100 [FIT] and 1 year (=8760 hours), respectively. Figure 6 shows the average system failure rates at time *MT* under the cases mentioned above[2]. In this figure, the failure rates of DnR components are assumed to be 0 ($\lambda_L = \lambda_T = \lambda_D = 0$). As shown in Figure 6, DTTR scheme always has better reliability than the lock-step pair schemes. It is also shown that the average system failure rates of the original lock-step pair and the original TMR-spare schemes increase exponentially as the probability of occurrence of a transient fault increases. Thus, it can be said the re-execution is very important feature even if the additional time slot is needed. The reliability of the modified TMR-spare and DTTR schemes is concluded to be identical in this evaluation. However, they have different performance (*i.e.,* the number of tasks executed concurrently). Thus, when real-time applications with hard deadlines are considered, they may have different system reliabilities. This issue is also in the scope of our future work.

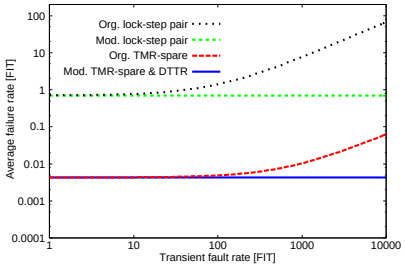
We also evaluate the influence of the failure rates of the DnR components. Figure 7 shows the average system failure rates for different λ_L, λ_T, and λ_D values with 4 processor

[2]The average failure rate of a given system A at time T is a failure rate of a simplex system B such that A and B have the identical reliability at time T.

(a) With 4 processor cores



(b) With 8 processor cores (#copy is 4)



(c) With 8 processor cores (#copy is 8)

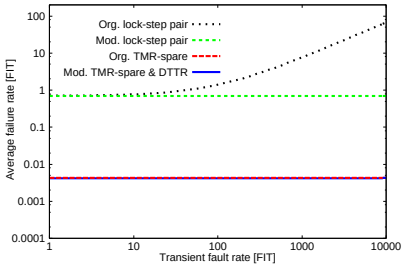


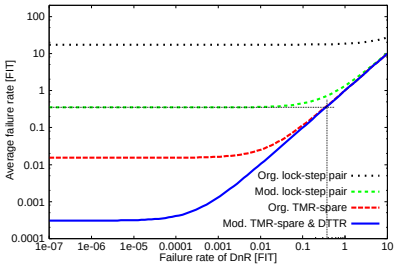
Figure 6. Average system failure rates for various transient fault rates where $\lambda = 100$.

cores and 8 processor cores where $\lambda = 100$ and $\lambda_{tr} = 5000$. Since $\lambda_D \approx \lambda_T > \lambda_L$, some appropriate horizontal point should be taken for each scheme for the fair comparison. For example, when the original TMR-spare schemes and DTTR scheme are compared, from the fact that the complexity of the DnR component for each scheme is almost the same, the same horizontal point should be taken. Thus, it is shown that the average system failure rate of DTTR scheme is almost the same or lower than that of the original TMR-spare scheme. In addition, if the failure rate of the DnR component of DTTR scheme is less than about 0.4 [FIT] (with 4 cores) and 0.7 [FIT] (with 8 cores), respectively, then DTTR scheme can achieve better average system failure rate than the lock-step pair schemes even if they use always-correct DnR components.

IV. Conclusion

We have been working for DTTR scheme which is one of dependability improvement methods for multiple-core based ECUs. In this paper, two lock-step pair schemes, two TMR-spare schemes, and DTTR scheme are modeled using the Markov chains considering both permanent and transient faults. Then, their average system failure rates are compared. As the result, DTTR scheme can achieve equivalent or better

(a) With 4 processor cores



(b) With 8 processor cores (#copy is 4)

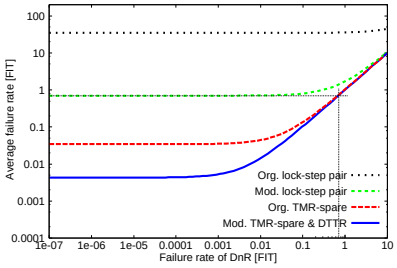


Figure 7. Average system failure rates for various failure rates of DnR components where $\lambda = 100$ and $\lambda_{tr} = 5000$.

reliability compared with the other schemes. We believe that our scheme is well suited to, for example, control programs in a little higher layer that manage lower-layer small dedicated ECUs in integrative manners, because multiple-core systems with flexible configuration may be essential for such applications.

Acknowledgment

We thank Yuichi Nakamura, Hiroshi Saito, Kenji Kise, and Takahiro Hanyu for technical discussions. This work was partially supported by CREST of JST. This work was also supported by JSPS KAKENHI Grant Numbers 15K00179, 15H02254.

References

[1] Mohamed Gomaa, Chad Scarbrough, T. N. Vijaykumar, and Irith Pomeranz. Transient-fault recovery for chip multiprocessors. *Proc. ISCA03*, pages 98–109, Jun. 2003.

[2] Christopher LaFrieda, Engin Ipek, Jose F. Martinez, and Rajit Manohar. Utilizing dynamically coupled cores to form a resilient chip multiprocessor. *Proc. DSN07*, pages 317–326, Jun. 2007.

[3] Rui Gong, Kui Dai, and Zhiying Wang. Transient fault tolerance on chip multiprocessor based on dual and triple core redundancy. *Proc. PRDC08*, pages 273–280, Dec. 2008.

[4] Mohammad Hosseinabady and Jose Nunez-Yanez. Fault-tolerant dynamically reconfigurable NoC-based SoC. *Proc. ASAP08*, pages 31–36, Jul. 2008.

[5] Masashi Imai and Tomohiro Yoneda. Fault diagnosis and reconfiguration method for network-on-chip based multiple processor systems with restricted private memories. *IEICE Trans. Inf. & Systems*, E96-D(9):1914–1925, Sep. 2013.

[6] T.Yoneda, M.Imai, N.Onizawa, A.Matsumoto, and T.Hanyu. Multi-chip NoCs for automotive applications. *Proc. of PRDC2012*, pages 105–110, 2012.

[7] T.Yoneda, M.Imai, H.Saito, T.Hanyu, K.Kise, and Y.Nakamura. An noc-based evaluation platform for safety-critical automotive applications. *Proc. of APCCAS2014*, pages 679–682, 2014.

[8] Daniel P. Siewiorek and Robert S. Swarz. *Reliable Computer Systems - Design and Evaluation* -. Digital Press, 1992.