Sequencing Tasks to Minimize the Effects of Near-Coincident Faults in TMR Controller Computers

Hagbae Kim, *Member, IEEE*, and Kang G. Shin, *Fellow, IEEE*

Abstract—Although Triple Modular Redundancy (TMR) has been widely used to mask the effects of a single faulty module, it cannot tolerate coincident faults in multiple modules caused by a common source, such as an environmental disruption or malfunction of a shared component. We propose a method to eliminate or alleviate the effects of (near) coincident faults by sequencing tasks on different modules in a TMR system. Specifically, we develop an *effective sequencing* of tasks to simply place an "optimal" distance (in the sense of minimizing the mean number of faulty tasks due to TMR failures) between the copies of a task to be executed on different modules. Several examples are presented, showing significant improvements in reducing TMR failures with the proposed task sequencing.

Index Terms—TMR failure; common-cause and independent faults; conventional, random, and effective sequencing of tasks; Task Interval (TI), task distance.

1 Introduction

TRIPLE Modular Redundancy (TMR) is one of the most popular fault-tolerance methods and uses the simplest form of static spatial redundancy. TMR can be applied to a component, subsystem, or system level. In the subsystem level, a nonredundant system is partitioned into a number of modules which are then triplicated, and majority voters are placed at module interfaces. Errors generated by any single faulty module are masked by a simple majority voter. However, TMR is effective only if (*A*1) the voter is fault-tolerant, (*A*2) module faults are statistically independent, and (*A*3) additional detection and recovery schemes are available to retain its fault masking capability, or prevent a *TMR failure* that results from sequentially-occurring faults in different modules. (A *TMR failure* is said to occur if the TMR system fails to form a majority of its module outputs.)

*A*1 can be satisfied by triplicating the voter [1], [9] and thus overcoming any critical single-point fault in the voter. Most TMR systems in the field are based on *A*2 and adopt appropriate schemes to repair/replace the faulty module (whose effects are masked) before a next module fault occurs, thus meeting *A*3. In FTMP [3], JPL-STAR [2], and C.vmp [11], disagreement detectors operate in parallel with the voters and compare the outputs of individual modules with the voted output so as to give an early warning of module exhaustion and invoke a subsequent recovery action, e.g., module replacement. The authors of [12] considered periodically-synchronizing sequences in a TMR system to tolerate multiple transient faults spaced out in time. In [10], we also proposed a method for recovering TMR failures caused by *sequentially-arriving* faults in different modules. Unlike the policies treating multiple faults only as sequential fault occurrences (under *A*2), we cannot ignore faults (near-) coincidentally occurring in different modules under certain circumstances, especially when the system is exposed to a harsh environment and/or the system is required to have very high reliability. For example, faults caused by electromagnetic interferences (EMI) are likely to induce coincident faults (or *common-cause faults*) in different modules of a TMR system, causing a TMR failure [5]. Although a dependent-fault-tolerant operating chart was produced by executing different programs on different CPUs in [4] and an optimal instruction-retry policy was proposed to recover from TMR failures due to coincident faults in [7], these approaches still focused on the behavior of independent faults, and did not present any adequate means of tolerating, or minimizing the effects of, coincident faults.

The key idea of this paper is based on the observation that multiple coincident module faults in a TMR system will *not* result in a TMR failure if

1) at any given time all three processor modules execute *different* tasks,
2) the source of the multiple coincident module faults does not last long, and
3) tasks are independent of one another.

(Condition 3 is relaxed later in Section 3.3.) This observation implies that some TMR failures can be avoided by properly sequencing tasks on the three modules of a TMR system. Such task sequencing will be able to deal with coincident faults induced by transient environmental disruptions like EMI. We first verify this idea by comparing the mean numbers of TMR failures for a *random task sequencing*, in which tasks are randomly selected for execution, and the *conventional task sequencing*, in which all three modules execute the same task. We, then, develop an *effective task sequencing* to simply place an optimal distance (in the sense of minimizing the mean number of TMR failures) between the copies of a task to be executed on different modules. We will not consider the effects of faults occurring during voting, which usually do not affect much the comparative numbers of TMR failures for both task-sequencing methods. (The voters are generally implemented with simple combinational logic components [13] and the time required for voting is relatively small compared to task execution times, and therefore, the probability of multiple fault occurrences during the voting process is very small.)

The rest of the paper is organized as follows. In Section 2, we discuss fault and task models along with the assumptions used. In Section 3, we analyze the effects of independent and common-cause faults on both the random and the conventional sequencing of tasks by computing the mean number of tasks producing incorrect executions results. An effective sequencing is also developed there. Section 4 presents demonstrative examples. Section 5 concludes the paper.

2 Basic Model and Assumptions

While independent module faults usually result from physical defects during manufacture or component-aging effects, common-cause faults occur due mainly to environmental disruptions affecting the entire system. Let $F_c(t)$ and $g_c(t)$ ($F_i(t)$ and $g_i(t)$) be the Probability Distribution Function (*PDF*) of occurrences of common-cause (independent) faults and the probability density function (*pdf*) of durations of external disruptions inducing these faults, respectively. We approximate error/failure occurrences and durations using the knowledge of fault-occurrence/duration information. (Both are not always equal though, because a fault may disappear without causing any error/failure or the latter may persist even after the former's disappearance.)

We define a *task interval* (TI), a basic time unit, as the time required to execute a task, the set of which composes a *mission phase*. Beginning with a basic task model having all independent tasks with an identical execution time (= one TI = Δt), we further cover realistic tasks of various execution times and/or dependent tasks in Section 3.3.

- *H. Kim is with the Department of Electrical Engineering, Yonsei University, 134 Shinchon-Dong, Sudaemoon-Ku, Seoul 120-749, Korea.*
- *K.G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Sciences, the University of Michigan, Ann Arbor, MI 48109-2122. E-mail: kgshin@eecs.umich.edu.*

Manuscript accepted Jan. 27, 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96086.

TABLE 1
$E(N_f)$S WHEN $\lambda_c = 1e-5$ ($\lambda_i = 1e-4$) AND $\mu_c = 1/3$ ($\mu_i = 1/3$)

Sequencing method\N	5	10	15	20	25
Conventional	1.150e−4	3.008e−4	5.001e−4	7.018e−4	9.041e−4
Random	8.051e−5	1.691e−4	2.652e−4	3.546e−4	4.286e−4
Method of [4]	9.475e−5	2.374e−4	3.879e−4	5.400e−4	6.923e−4

When the three modules execute different tasks during each TI according to a certain sequencing policy, it is not so simple to determine when to vote on the execution results as the conventional sequencing that relies on immediate voting. To circumvent this difficulty, we assume that all task execution results are saved and voted on later. Our idea and analysis are based on the assumption that the saved data is made immune to faults during the wait for voting through well-developed memory fault-tolerance schemes like error detection and correction codes (EDCCs). Tasks with tight timing constraints will be given priority to be executed early and their execution results will be voted on immediately. (More on this will be discussed in Section 3.4.)

3 Task Sequencing Policies

Let a mission phase consist of N tasks and N_f be the number of tasks producing incorrect results in the presence of TMR failures. Occurrences of TMR failures depend on fault behaviors as well as on the method of sequencing tasks. We consider only one occurrence of common-cause or coincidentally-occurring independent faults during the time interval of interest due to the rareness of their multiple occurrences. (Fault interarrival times are much larger than the interval of interest.)

Let P^c_{jk} be the probability that a TMR failure occurs in the jth TI and the duration of common-cause faults (including permanent ones) to induce this TMR failure is between $(k-1)\Delta t$ and $k\Delta t$ (i.e., k TIs). Then, by using $F_c(t)$ and $g_c(t)$, P^c_{jk} can be derived for $1 \le j \le N-1$ and $1 \le k \le N-j+1$ as:

$$P^c_{jk} = \left(F_c(j\Delta t) - F_c((j-1)\Delta t)\right)\int_0^{\Delta t} \frac{1}{\Delta t}\int_{(k-1)\Delta t - x}^{k\Delta t - x} g_c(t)dt\,dx \quad \text{for } 2 \le k \le N-j. \tag{3.1}$$

where P^c_{j1} and $P^c_{j(N-j+1)}$ are also obtained by integrating t for $[0, \Delta t]$ and $[(N-j)\Delta t - x, \infty]$, respectively. If F_c and g_c are replaced with F_i and g_i, respectively, all of the above are also applicable to independent faults, resulting in P^i_{jk}. We now consider the simplest task model having independent tasks with an identical execution time and an arbitrary execution order as $\{t_1, t_2, \cdots, t_N\}$, which will be extended to more general cases.

3.1 Comparison of Conventional and Random Task Sequencing

First, we derive and compare the mean of N_f (defined by $E(N_f)$) for the conventional task sequencing and the random task sequencing to show the effects/benefits of sequencing tasks. To do this, we compute the probability mass functions (*pmf*) of N_f and let ℓ be a dummy variable for N_f.

In the conventional sequencing, the N_f due to common-cause faults is simply equal to k (i.e., $\ell = k$) because only one possible occurrence of those faults was assumed during the mission phase. A TI of a module M_i is said to be *faulty* if M_i is faulty during that TI. The N_f due to coincidentally-occurring independent faults is the number of TIs during which two or three[1] modules are faulty (*overlapped faulty*). Let P^i_{jk} and P^i_{uv} be the probabilities of independent faults occurring in two modules. For any pair of j and k, when $u \le j$, $\ell = v - j + u$ for $1 \le \ell \le k-1$, and all faults lasting longer than $(j-u+k)$ TIs cause ℓ $(= k)$ TMR failures. When $u > j$, $\ell = v$ for $1 \le \ell \le k-u+j-1$, and all faults lasting longer than $(k-u+j)$ TIs cause ℓ $(= k-u+j)$ TMR failures. Thus, by using P^c_{jk} and P^i_{jk}, the *pmf* of N_f is obtained as:

$$P(N_f = \ell) = \sum_{j=1}^{N-\ell+1} P^c_{j\ell} + 3\sum_{j=1}^{N}\sum_{k=1}^{N-j+1} P^i_{jk}\left[\sum_{u=1}^{j}\left(P^i_{u(j-u+\ell)}\Pi_\ell(1,k-1) + \sum_{v=j-u+k}^{N-u+1} P_{uv}\delta_\ell(k)\right)\right.$$

$$\left. + \sum_{u=j+1}^{j+k-1}\left(P^i_{u\ell}\Pi_\ell(1,k-u+j-1) + \sum_{v=k-u+j}^{N-u+1} P_{uv}\delta_\ell(k-u+j)\right)\right], \tag{3.2}$$

where $\Pi_\ell(1, k)$ is a rectangular function of ℓ over $[1, k]$ (i.e., $\Pi_\ell(1, k) = 1$ if $\ell \in \{1, 2, \cdots, k\}$ and 0 otherwise), and $\delta_\ell(k)$ is a delta function such that $\delta_\ell(k) = 1$ if $\ell = k$ and 0 otherwise.

In the random task sequencing, the N_f due to common-cause faults is an integer smaller than, or equal to, k. That is, the *pmf* of N_f due to common-cause faults is derived as:

$$P(N_f = \ell) = \sum_{j=1}^{N}\sum_{k=\ell}^{N-j+1} P^c_{jk} P(k, \ell), \tag{3.3}$$

where $P(k, \ell)$ is the probability that any pair of modules executes ℓ tasks during k consecutive faulty TIs, as derived in the Appendix. The N_f due to coincidentally-occurring independent faults is also an integer not greater than the number of overlapped-faulty TIs. The *pmf* of N_f is thus derived from (3.2) and (3.3) as:

$$P(N_f = \ell) = 3\sum_{j=1}^{N}\sum_{k=1}^{N-j+1} P^i_{jk}\left(\sum_{u=1}^{N}\sum_{v=\ell}^{N-u+1} P_{uv} P(k, v, \ell)\right), \tag{3.4}$$

where $P(k, v, \ell)$ is the probability of ℓ TMR failures during any pair of modules' k and v faulty TIs, as derived in the Appendix. Using the *pmf* of N_f, we finally compute $E(N_f)$ as

$$\sum_{\ell=1}^{N} \ell\, P(N_f = \ell).$$

The examples of Table 1, which compute $E(N_f)$ under some characteristics of faults (governed by Poisson processes with particular parameters), indicate not only the possibility that certain methods of sequencing tasks may be able to deal with coincident faults but also the existence of more effective task sequences with a smaller $E(N_f)$.

3.2 Proposed Task Sequencing

We now consider a simple and efficient sequencing strategy to maintain a fixed distance (in time), $d\Delta t$, called *task distance* (TD), between the copies of a task to be executed on different modules. Then we want to develop an effective sequence of task copies separated by the optimal distance so as to minimize $E(N_f)$. Such a sequence is more complex to build but yields more reliable task execution results than the random and conventional sequencing. Let M_1, M_2, and M_3 be the first, the second, and the third modules labeled arbitrarily in a TMR system. The proposed strategy is stated as follows.

1. We ignore the rare case of all three modules (near-) simultaneously becoming faulty due to independent faults with little loss of accuracy.

- In M_1, N tasks are assigned to N different TIs in an arbitrary order and let t_j denote the task assigned to the jth TI.
- In M_2 and M_3, t_j is assigned to the $(j + d)_{modN}$th TI and the $(j + 2d)_{modN}$th TI, respectively.

The symbol α_{modN} represents that the αth TI is wrapped around the Nth TI if $\alpha > N$, like $\{t_5, t_6, \cdots, t_{10}\}$ in Fig. 1A, because we consider only N consecutive TIs for executing all copies of N tasks on the three modules. (If $\alpha \le N$ then α_{modN} is equal to α, else $N - \alpha$.) TD should not be smaller than d even after wrap-around, like t_5 in Fig. 1A. Since wrap-around happens earlier in M_3 than M_2, the following condition must hold for M_3 to guarantee TD not to be smaller than d after wrap-around in both M_2 and M_3:

$$j - (j + 2d - N) \ge d \Leftrightarrow 3d \le N. \tag{3.5}$$

We now want to derive $E(N_f(d))$ in this sequencing strategy and find an integer d_{opt} that minimizes $E(N_f(d))$ for $0 \le d \le \lceil \frac{N}{3} \rceil$. Let S_1 be the subset of tasks which are not wrapped around ($\{t_1, t_2, \cdots, t_{N-2d}\}$), S_2 be the subset of tasks wrapped around only in M_3 ($\{t_{N-2d+1}, t_{N-2d+2} \cdots, t_{N-d}\}$), and S_3 be the subset of tasks wrapped around in both M_2 and M_3 ($\{t_{N-d+1}, t_{N-d+2} \cdots, t_N\}$). Let N_f^1, N_f^2, and N_f^3 be the numbers of TMR failures occurred during the execution of tasks in S_1, S_2, and S_3, respectively. Using $\sum_{i=1}^{3} N_f^i = N_f$, the *pmf* of N_f can be obtained by convolving the *pmf*s of N_f^1, N_f^2, and N_f^3:

$$P(N_f = \ell) = P(N_f^1 = \ell) * P(N_f^2 = \ell) * P(N_f^3 = \ell). \tag{3.6}$$

$P(N_f^i = \ell)$ for $i \in \{1, 2, 3\}$ can be derived by adding the *pmf*s of common-cause and independent faults, which are derived separately and presented in the Appendix. (Occurrences of common-cause and independent faults are also exclusive to each other due to the assumption of at most one fault occurrence.)

By using the constraint of d in (3.5) and the *pmf* of N_f derived as a function of d, we can determine d_{opt} minimizing $E(N_f)$.[2]

3.3 General Model for Different-Size and Dependent Tasks

Consider a general model for tasks with a variety of execution times. Let $\{t_1, t_2, \cdots, t_m\}$ be an ordered set of m tasks, and let n_i be the execution time of t_i such that $\sum_{i=1}^{m} n_i = N$. Suppose the same sequencing strategy as the basic task model is used as in Fig. 1B. Then, unlike the case of basic task model, when placing the copies of t_k in the module schedules, one is likely to encounter a case where $2d + \sum_{i=1}^{k-1} n_i < N$ but $2d + \sum_{i=1}^{k} n_i > N$ on M_3, or $d + \sum_{i=1}^{k-1} n_i < N$ but $d + \sum_{i=1}^{k} n_i > N$ on M_2. That is, none of the remaining tasks, $t_k, t_{k+1}, \cdots, t_m$, can be executed before, or at the end of, the Nth TI of M_2 or M_3. Since we cannot generally divide a task arbitrarily, we have to extend the execution of each of these tasks beyond the Nth TI. For example, on M_3, t_k is executed from the $(2d + \sum_{i=1}^{k-1} n_i)$th TI to the $(2d + \sum_{i=1}^{k} n_i)$th TI, and t_{k+1} is not wrapped around to the first TI but extended to the $(2d + \sum_{i=1}^{k} n_i - N)$th TI, as shown in Fig. 1B. This extends the cumulative execution time of tasks $t_1, \cdots, t_k$ from N TIs to

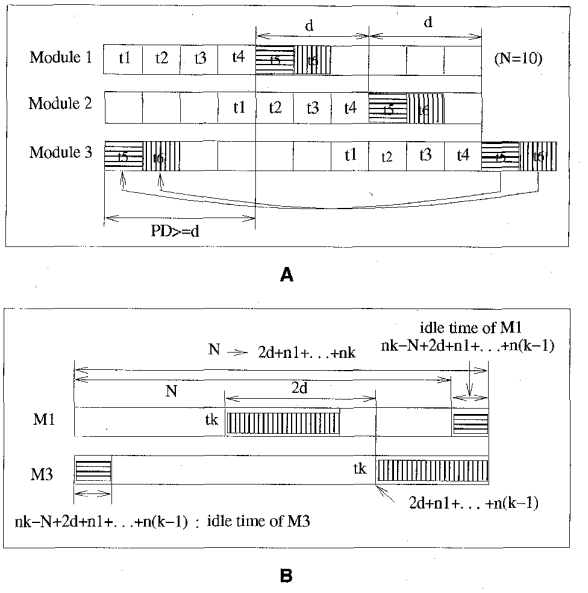

Fig. 1. **A:** Wrapping tasks around in M_3 with $N = 20$ and $d = 3$; **B:** Sequencing t_k such that $2d + \sum_{i=1}^{k-1} n_i < N$ but $2d + \sum_{i=1}^{k} n_i > N$ in M_3 by shifting time to the right by $\left(2d + \sum_{i=1}^{k} n_i - N\right)$ TIs.

$(2d + \sum_{i=1}^{k} n_i)$ TIs, by adding idle TIs. Unlike the arbitrary ordering of tasks in the basic task model, we now develop an ordering algorithm for m tasks by minimizing the extra TIs as follows:

Select any task and call it t_1 ($i = 1$).

Step 1: $i := i + 1$ (sequence tasks until tasks on M_3 need to be wrapped around)

If any task has $n_i = N - 2d - \sum_{j=0}^{i-1} n_j$, then call it t_a and go to Step 2

Else if any task has $n_i < N - 2d - \sum_{j=0}^{i-1} n_j$, then call it t_i and go to Step 1

Else select a task having n_i minimizing $2d + \sum_{j=0}^{i-1} n_j + n_i - N$ and call it t_a and go to Step 2

Step 2: $i := i + 1$ (from $i = a$ and until tasks on M_2 need to be wrapped around)

If any task has $n_i = N - d - \sum_{j=0}^{i-1} n_j$, then call it t_b and go to Step 3

Else if any task has $n_i < N - d - \sum_{j=0}^{i-1} n_j$, then call it t_i and go to Step 2

Else select a task n_i minimizing $d + \sum_{j=0}^{i-1} n_j + n_i - N$ and call it t_b and go to Step 3

Step 3: Sequence the remaining tasks (from t_{b+1}) in arbitrary order

Let n_d be the number of TMR failures in this task model. When a mission phase is composed of N TIs[3] and m tasks, we can derive

2. Although it is impossible to derive d_{opt} in a closed-form, we can determine d_{opt} numerically, i.e., computing $E(N_f)$ for every d $(0 \le d \le \lceil \frac{N}{3} \rceil)$.}

3. The effects of the extra TIs required to handle different size tasks are not considered, because 1) $2d + n_1 + \cdots + n_k - N$ is smaller than the execution time of any remaining task, 2) the total number of TIs during which tasks are actually executed in each module is still N, and 3) the number of the extra TIs depends on the given task set and thus is difficult to determine. Since $N >> 2d + n_1 + \cdots + n_k - N \ge 0$, this approxima-

the *pmf* of n_d by using $P(N_f = \ell)$ derived for N TIs from (3.6) and considering that any TI is assigned to a task having an execution time of n_i TIs with a uniform distribution (probability $\frac{n_i}{N}$) according to the proposed ordering algorithm. Note that a task would have an erroneous output if any TI composing the task is faulty.

Given $N_f = \ell$, the conditional probability of $n_d = r$ is obtained similarly to "sampling balls without replacement" in the urn containing N balls with m different colors. Since there are ℓ faulty TIs, we draw ℓ balls from the urn and derive the *pmf* for the number, r, of different colors among ℓ drawn balls. Let S_r^ℓ be a set including all possible subsets of task sizes, each subset consisting of r task sizes $n_{s1}, n_{s2} \cdots, n_{sr}$ such that $\sum_{i=1}^{r} n_{si} \geq \ell$. The number of subsets each consisting of r tasks is ${}_mC_r$, from which we obtain the set S_r^ℓ. The probability that at least one ball is drawn from every color group corresponding to a task in the subset $(n_{s1}, n_{s2} \cdots, n_{sr}) \in S_r^\ell$ is:

$$P\left(n_d = r \middle| N_f = \ell\right) = \sum_{S_r^\ell} \frac{1}{{}_N C_\ell} \left[\sum_{i_1=1}^{n_{s1}} \sum_{i_2=1}^{n_{s2}} \cdots \sum_{i_r=1}^{n_{sr}} \left(\sum_{j=1}^{r} \left(n_{sj} - i_j\right) \right) C_{\ell - r} \right], \quad (3.7)$$

which contains all the cases of r different color balls drawn by using r summations as well as all combinations of $\ell - r$ balls selected from the remaining balls in each case (i.e., from $\sum_{j=1}^{r} (n_{sj} - i_j)$ balls), which can be obtained by avoiding repeated cases. The *pmf* of n_d is thus derived as:

$$\begin{aligned} P\left(n_d = r\right) &= \sum_{l=1}^{N} P\left(n_d = r \middle| N_f = \ell\right) P\left(N_f = \ell\right) \\ &= \sum_{l=1}^{N} \sum_{S_r^\ell} \frac{1}{{}_N C_\ell} \left[\sum_{i_1=1}^{n_{s1}} \cdots \sum_{i_r=1}^{n_{sr}} \left(\sum_{j=1}^{r} \left(n_{sj} - i_j\right) \right) C_{\ell - r} \right] P\left(N_f = \ell\right). \quad (3.8) \end{aligned}$$

As a result, we compute all $E(n_d)$s for $0 \leq d \leq \left\lceil \frac{N}{3} \right\rceil$ and determine d_{opt} that minimizes $E(n_d)$ numerically as we did for the basic task model. In [8], we also proposed an efficient method to reduce the amount of computation required for (3.8), which increases rapidly as m and N increase.

We also consider a task model in which some tasks are dependent on others, i.e., there are precedence constraints among the tasks. We define a new task by merging all dependent tasks into consecutive TIs to meet the constraints, thus obtaining a set of new tasks. The size of a new task is equal to the sum of sizes of all tasks merged into the task. The problem of effectively sequencing tasks in such a newly-combined set is then equivalent to a different-size task model, while keeping same N but reduced m. However, even when no task is fitted in the remaining TIs ($= N - 2d - \sum_{i=1}^{k-1} n_i$ or $N - d - \sum_{i=1}^{k-1} n_i$) on M_3 or M_2, no extra TIs may be required unlike the different-size task model. Since a newly-defined task was made by merging some dependent tasks, a task t_k can be sliced in a way that its parts are sequenced during both the TIs near the Nth TI and the TIs near the first TI after wrap-around without violating the precedence constraints.

3.4 Tasks with Timing Constraints

The TMR system may have to deal with certain tasks with tight timing constraints; for example, the delay in executing tasks for an aircraft should be kept below a certain limit called the *control system deadline* (CSD) in [6]. Under such timing constraints, we may need to adopt a more complicated voting process—those task results must be voted on as soon as all three copies are completed.

tion makes little difference.

We can still apply the proposed method (to alleviate the effects of short-lived coincidentally-occurring faults) for those tasks by slightly modifying the case in the absence of timing constraints. For example, when tasks t_i have deadlines D_i for $1 \leq i \leq m$, the task with a shorter deadline is assigned earlier in the sequence, instead of trying to minimize the extra TIs as was done in Section 3.3. We then derive d_{opt} as before. To prevent certain tasks from missing deadlines according to this d_{opt}, we first assign the tasks with tight deadlines $d \leq \left\lceil \frac{D_i - I_i}{3} \right\rceil$, where I_i is the time to start executing t_i in the first module. We then go on applying the proposed sequencing method for the remaining tasks meeting deadlines through $d = d_{opt}$. Although this is a somewhat *ad hoc* variation of the proposed algorithm in Section 3.3, it will perform better than the conventional sequence in reducing the number of TMR failures.

4 Numerical Examples

We again assume fault behaviors governed by Poisson processes and the mean rates of fault occurrence and duration given as $\lambda_c = 10^{-6}$, $\lambda_i = 10^{-4}$, $\mu_c = 1/6$, and $\mu_i = 1/6$, all in number of TIs. Although independent faults occur more frequently than common-cause faults in each individual module, TMR failures are caused mainly by common-cause faults, as shown in Fig. 2A. Coincident independent faults occur significantly less than common-cause faults. Thus, if a large number of TMR failures occur during a certain period, their main causes are likely to be common-cause faults as shown in Fig. 2A. In Fig. 2B, the mean numbers of TMR failures are derived for several sequencing methods while varying the number of tasks. The conventional task sequencing turns out to be the worst, i.e., it has the largest $E(N_f)$ for any N and the fastest increase of $E(N_f)$ as N increases. Clearly, the proposed sequencing method with d_{opt}(= {8, 10, 11, 13, 15, 16, 18, 20, 21} for each N on x-axis) performs much better than both the method in [4] (which is similar to the special case of $d = 1$ of our method) and the random sequencing. We found the case of $6 \leq d \leq d_{opt}$ working better than the random sequencing.

In Fig. 3A, to investigate the effects of d on $E(N_f)$ under different fault-occurrence conditions, we computed $E(N_f)$s while varying d from 0 to d_{opt} (with fixed $N = 30$) for both the basic task model and the general task model, in which tasks within a pair of parentheses are dependent on each other due to the precedence constraints, and thus, a new task set {6, 3, 4, 6, 5, 8} is defined by merging dependent tasks. Cases (b) and (d), in which common-cause faults occur more frequently than cases (a) and (c), have shown relatively significant improvements in reducing the effects of coincidentally-occurring faults by using a better choice of d value. In other words, the use of a better d achieves more when the effects of common-cause faults are severer. In Fig. 3B, we also dealt with two task sets for different size models, T_1 = {1, 2, 4, 3, 2, 4, 1, 1, 3, 2, 4, 3} and T_2 = {3, 5, 7, 5, 4, 6}, which have same $N = 30$ but different number of tasks, $m_1 = 12$ and $m_2 = 6$. Although $E(N_f)$ is the same for both cases, T_1 suffers more TMR failures (larger $E(n_d)$) because $m_1 > m_2$. However, the difference of $E(n_d)$s in the two task sets decreases as d increases. This is because the increase of $E(n_d)$ as a result of increasing m becomes less pronounced as d approaches d_{opt}, which is shown similarly in Fig. 2B (increasing N).

5 Conclusion

In this paper, we have developed a new method for sequencing task copies in a TMR system (at a component, subsystem, or system level) in order to alleviate the effects of common-cause/coincident faults, the main source of TMR failures. We first proposed a simple and efficient sequencing strategy in which three copies of each task are executed on the three modules of a TMR

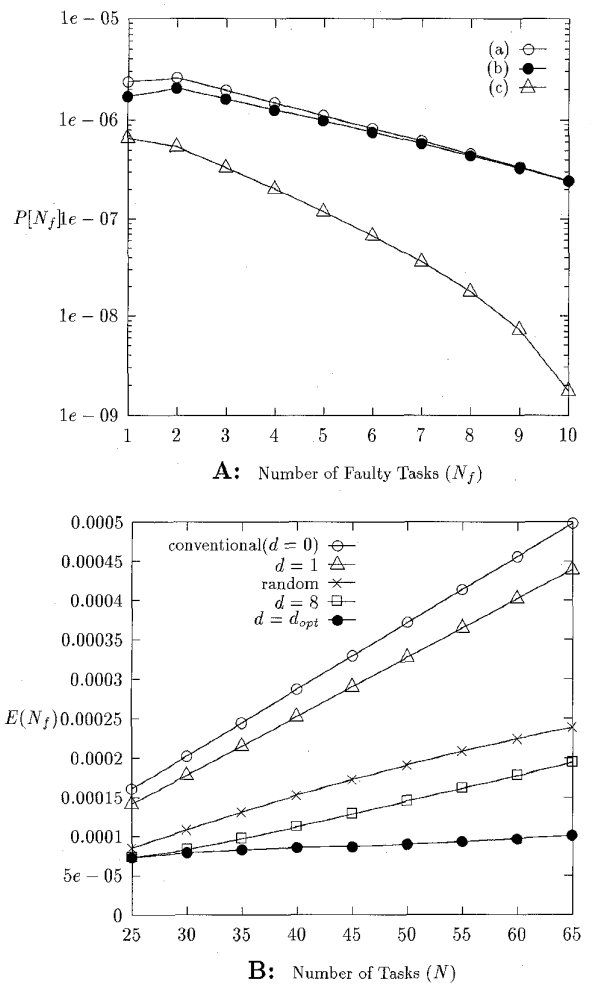

Fig. 2. **A:** *Pmf* of N_f due to (a) both faults with $P(N_f = 0) = 0.99988$, (b) only common-cause ones with $P(N_f = 0) = 0.9999$, and (c) only independent ones with $P(N_f = 0) = 0.99998$; **B:** $E(N_f)$ while varying N for various sequencing methods.

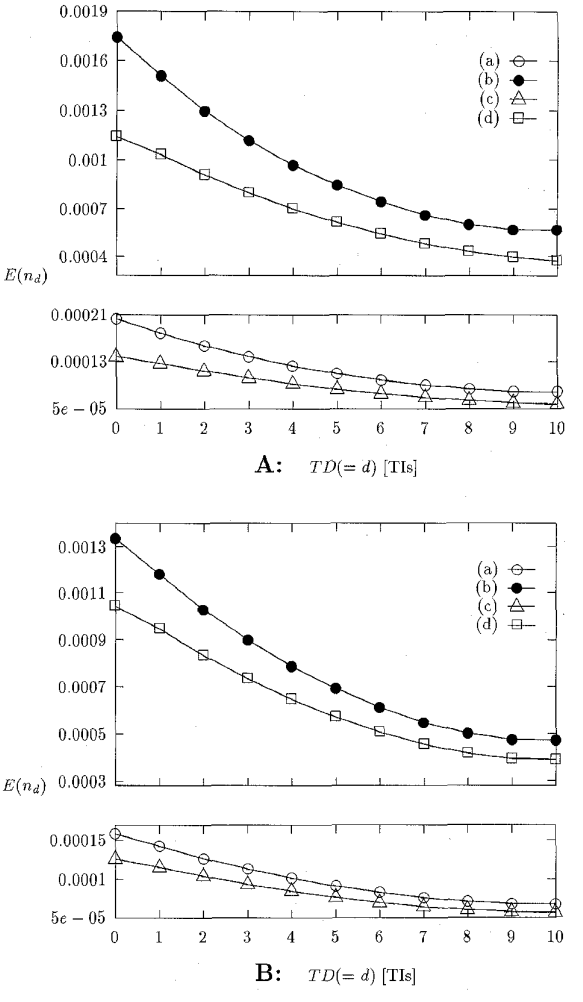

Fig. 3. **A:** $E(N_f)$ vs. d for the *basic* task model having $N = 30$ under (a) $\lambda_c = 10^{-6}$ and (b) $\lambda_c = 10^{-5}$, and for a *general* task set having {(1, 4, 1), (3), (3, 1), (4, 2), (5), (3, 5)} ($m = 11$ and $N = 33$), under (c) $\lambda_c = 10^{-6}$ and (d) $\lambda_c = 10^{-5}$; **B:** $E(n_d)$ vs. d for the *different size model* under (a) a task set $T_1 = \{1, 2, 4, 3, 2, 4, 1, 1, 3, 2, 4, 3\}$ and $\lambda_c = 10^{-6}$, (b) T_1 and $\lambda_c = 10^{-5}$, (c) a task set $T_2 = \{3, 5, 7, 5, 4, 6\}$ and $\lambda_c = 10^{-6}$, and (d) T_2 and $\lambda_c = 10^{-5}$.

system with $TD = d$, and then derived numerically the optimal d to minimize the mean number of TMR failures. Through numerical examples while varying N and d, we showed that the proposed sequencing strategy can significantly decrease the number of TMR failures under various conditions, especially when common-cause faults are likely to occur.

Although we dealt only with TMR systems, we can extend the proposed approach to the problem of sequencing tasks in NMR systems. We can also include the effects of unreliable voting, which requires a more complicated sequencing strategy to minimize the mean number of NMR failures.

APPENDIX
DERIVATION OF KEY *pmf*s

We present a concise derivation of $P(k, \ell)$, $P(k, v, \ell)$, and $P(N_f^i = \ell)$. (See [8] for full explanation.) Consider a pair of modules, say M_1/M_2, contributing ℓ_1 TMR failures. Make k tasks executed during faulty TIs of M_1 distinguishable from the remaining $N - k$ tasks. The random task sequencing on M_2 is, then, casted into the problem of *sampling of balls without replacement* in an urn (M_2) containing k black balls and $N - k$ white balls, where k balls are drawn from the urn. The probability of drawing ℓ_1 black balls is simply derived as $P(k, \ell_1) = {}_k C_{\ell_1} {}_{N-k}C_{k-\ell_1} / {}_N C_k$ for $2k - N \le \ell_1 \le k$. Let ℓ_2 be the number of TMR failures contributed by the other pairs, M_2/M_3 and M_1/M_3, then $\ell = \ell_1 + \ell_2$. Again, suppose $2(N - \ell_1)$ faulty tasks (black balls) of M_1 and M_2 are distinguishable from the remaining $N - 2(N - \ell_1)$ tasks (white balls) in M_3. The conditional probability that ℓ_2 black balls are selected among k balls drawn from M_3 holding k black balls given ℓ_1 is derived as $P(k, \ell_2 \mid \ell_1) = {}_{2(N-\ell_1)}C_{\ell_2}\, {}_{2\ell_1-N}C_{k-\ell_2} / {}_N C_k$ for $k + N - 2\ell_1 \le \ell_2 \le 2(N - \ell_1)$. From the above $P(k, \ell_1, \ell_2) = P(k, \ell_1)P(k, \ell_2 \mid \ell_1)$ is computed, and thus $P(k, \ell) = \sum_{\ell_1=1}^{\ell} P(k, \ell_1, \ell - \ell_1)$. Similarly, $P(k, v, \ell)$ is obtained as the probability that ℓ black balls are selected among v (or k) balls drawn from the urn (the other module), because there are v (or k) faulty TIs of the other module. Consequently,

$P(k, v, \ell) = {}_kC_{\ell}\,{}_{N-k}C_{v-\ell} / {}_NC_v$ for $k + v - N \le \ell \le k$.

In deriving $P(N_f^i = \ell)$ for $1 \le i \le 3$, first consider the tasks affected by common-cause faults. In S_1, the TD of a task is d for all pairs of task copies on M_1/M_2 and M_2/M_3. When $1 \le j \le d$, if $k > N - d$ then $\ell = N - 2d$, otherwise $\ell = k - d$ for $1 \le \ell \le N - 2d - 1$. When $d + 1 \le j \le N - d$, $\ell = k - d$ for $1 \le \ell \le N - d - j + 1$, similar to the above. Thus,

$$P(N_f^1 = \ell) = \sum_{j=1}^{d}\left(P^c_{j(d+\ell)}\Pi_\ell(1, N-2d-1) + \sum_{k=N-d}^{N-j+1} P^c_{jk}\delta_\ell(N-2d)\right) + \sum_{j=d+1}^{N-d} P^c_{j(d+\ell)}\Pi_\ell(1, N-d-j+1). \quad (5.1)$$

In S_2, the TDs of task copies in module pairs M_1/M_2, M_2/M_3, and M_3/M_1 are d, $N - d$, and $N - 2d$, respectively. When $1 \le j \le d$, $\ell = k - N + 2d$ only in M_3/M_1. When $d + 1 \le j \le N - 2d$, if $k > N - d - j$ then $\ell = k - N + d + j - 1$ only in M_1/M_2. When $N - 2d + 1 \le j \le N - d$, we get $\ell = k - d$ ($\le N - j - d + 1$). Thus, similar to (5.1), we obtain:

$$P(N_f^2 = \ell) = \sum_{j=1}^{d}\left(P^c_{j(N-2d+\ell)}\Pi_\ell(1, d-1) + \sum_{k=N-d}^{N-j+1} P^c_{jk}\delta_\ell(d)\right) + \sum_{j=d+1}^{N-2d} P^c_{j(N-d-j+\ell+1)}\Pi_\ell(1, d) + \sum_{j=N-2d+1}^{N-d} P^c_{j(d+\ell)}\Pi_\ell(1, N-d-j+1). \quad (5.2)$$

In S_3, the TDs of task copies on module pairs M_1/M_2, M_2/M_3, and M_3/M_1 are $N - d$, d, and $N - 2d$, respectively. When $1 \le j \le d$, ℓ is equal to $k - d$, $2d - j + 1$, $k - N + 2d$, and d, respectively, for $d < k \le 2d - j$, $2d - j < k \le N - d - j + 1$, $N - d - j + 1 < k < N - d$, and $N - d \le k \le N - j + 1$. When $d + 1 \le j \le 2d$, $\ell = k - N + 2d$ only in M_3/M_1. Considering the above facts, we have:

$$P(N_f^3 = \ell) = \sum_{j=1}^{d}\left(P^c_{j(d+\ell)}\Pi_\ell(1, d-j) + \sum_{k=2d-j+1}^{N-d-j+1} P^c_{jk}\delta_\ell(d-j+1) + P^c_{j(N-2d+\ell)}\Pi_\ell(d-j+2, d-1) + \sum_{k=N-d}^{N-j+1} P^c_{jk}\delta_\ell(d)\right) + \sum_{j=d+1}^{2d} P^c_{j(N-2d+\ell)}\Pi_\ell(1, 2d-j+1). \quad (5.3)$$

Consider the effects of independent faults, i.e., N_f^i TMR failures, similarly except for separately treating three pairs, M_1/M_2, M_2/M_3, and M_3/M_1. As shown in Fig. 4, suppose tasks in a subset, say S_1, are sequenced between the a_1th TI and the b_1th TI on one module, say M_1, and sequenced between the a_2th TI and the b_2th TI on the other module, say M_2. Then, TD = $c = a_2 - a_1$. Let P^i_{jk} (P^i_{uv}) be the probabilities that an independent fault occurs in M_1 (M_2) during the ith (uth) TI with an active duration of $k\Delta t$ ($v\Delta t$). The values of $\{a_1, a_2, b_1, b_2\}$ of three pairs of modules are given in Fig. 4. Let $N_f^k(ij) = \ell$ be the number of TMR failures for the tasks of S_k due to independent faults occurring coincidentally in a pair of modules M_i/M_j. First, consider the case of $a_1 = 1$. When $1 \le u \le c + j$, if $k, v > c + j - u + r_2$ then $\ell = r_2$, otherwise $\ell = v - c - j + u$ for $1 \le \ell \le r_2 - 1$, where $r_2 = \min\{k, b_1 - j + 1\}$. When $c + j + 1 \le u \le r_1$, where $r_1 = \min\{c + j + k - 1, b_2\}$, if $k, v > r_3$ then $\ell = r_3$, otherwise $\ell = v$ for $1 \le \ell \le r_3 - 1$, where $r_3 = \min\{k - u + c + j, b_1 - j + 1\}$. Consequently, we have the *pmf* of $N_f^k(ij) \in \{N_f^1(12), N_f^1(31), N_f^2(23), N_f^2(31), N_f^3(12), N_f^3(23)\}$:

$$P(N_f^k(ij) = \ell) = \sum_{j=1}^{b_1}\sum_{k=1}^{N-j+1} p^i_{jk}\left[\sum_{u=1}^{c+j}\left(P^i_{u(c+j-u+\ell)}\Pi_\ell(1, r_2-1) + \sum_{v=c+j-u+r_2}^{N-u+1} P^i_{uv}\delta_\ell(r_2)\right) + \sum_{u=c+j+1}^{r_1}\left(P^i_{u\ell}\Pi_\ell(1, r_3-1) + \sum_{v=r_3}^{N-u+1} P^i_{uv}\delta_\ell(r_3)\right)\right]. \quad (5.4)$$

In case of $a_1 > 1$, we should also deal with two cases of j:

1) $1 \le j \le a_1 - 1$ and
2) $a_1 \le j \le b_1$.

We can compute the probabilities of Case 2 just like (5.4). Only for $k > a_1 - j + 1$ of Case 1, we again deal with two cases according to u; when $1 \le u \le a_2 - 1$, if $v > a_2 - u + r_5$ then $\ell = r_5$, otherwise $\ell = v - a_2 + u$ for $1 \le \ell \le r_5 - 1$, where $r_5 = \min\{k - (a_1 - j), b_1 - a_1\}$, and when $a_2 \le u \le r_4$, where $r_4 (= r_1) = \min\{a_2 + (k - a_1 + j) - 1, b_2\}$, if $v > r_6$ then $\ell = r_6$, otherwise $\ell = v$ for $1 \le \ell \le r_3 - 1$, where $r_6 = \min\{k - (a_1 - j) - (u - a_2), b_1 - a_1\}$. Thus, we have the *pmf* of $N_f^k(ij) \in \{N_f^1(23), N_f^2(12), N_f^3(31)\}$:

$$P(N_f^k(ij) = \ell) = \sum_{j=1}^{a_1-1}\sum_{k=a_1-j+1}^{N-j+1} P^i_{jk}\left[\sum_{u=1}^{a_2}\left(P^i_{u(a_2-u+\ell)}\Pi_\ell(1, r_5-1) + \sum_{v=a_2-u+r_5}^{N-u+1} P^i_{uv}\delta_\ell(r_5)\right) + \sum_{u=a_2+1}^{r_4}\left(P^i_{u\ell}\Pi_\ell(1, r_6-1) + \sum_{v=r_6}^{N-u+1} P^i_{uv}\delta_\ell(r_6)\right)\right] + \sum_{j=a_1}^{b_1}\sum_{k=1}^{N-j+1} p^i_{jk}\left[\sum_{u=1}^{c+j}\left(P^i_{u(c+j-u+\ell)}\Pi_\ell(1, r_2-1) + \sum_{v=c+j-u+r_2}^{N-u+1} P^i_{uv}\delta_\ell(r_2)\right) + \sum_{u=c+j+1}^{r_1}\left(P^i_{u\ell}\Pi_\ell(1, r_3-1) + \sum_{v=r_3}^{N-u+1} P^i_{uv}\delta_\ell(r_3)\right)\right]. \quad (5.5)$$

Since TMR failures in one module pair is treated exclusively to those of other pairs, we can get:

$$P(N_f^k = \ell) = P(N_f^k(12) = \ell) + P(N_f^k(23) = \ell) + P(N_f^k(31) = \ell) \quad k \in \{1, 2, 3\}. \quad (5.6)$$

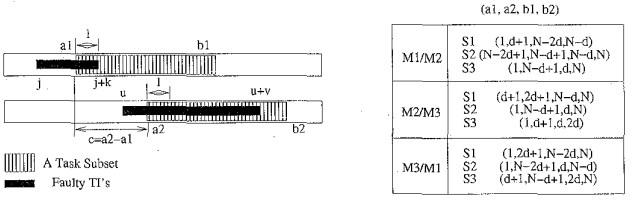

Fig. 4. An example of ℓ when $u \le c + j$ and (a_1, a_2, b_1, b_2).

ACKNOWLEDGMENTS

The work reported was supported in part by the U.S. Office of Naval Research under Grant N00014–91–J–1115 and by NASA under Grant NAG–1–1120. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the view of the funding agencies.

REFERENCES

[1] J.A. Abraham and D.P. Siewiorek, "An Algorithm for the Accurate Reliability Evaluation of Triple Modular Redundancy Networks," *IEEE Trans. Computers*, vol. 23, no. 7, pp. 682-692, July 1974.

[2] A. Avizienis and G.C. Gilley, "The STAR (Self-Testing and Repairing) Computer: An Investigation of Theory and Practice of Fault-Tolerant Computer Design," *IEEE Trans. Computers*, vol. 20, no. 11, pp. 1,312-1,321, Nov. 1971.
[3] A.L. Hopkins Jr., T.B. Smith III, and J.H. Lala, "FTMP—A Highly Fault-Tolerant Multiprocessor for Aircraft," *Proc. IEEE*, vol. 66, no. 10, pp. 1,221-1,239, Oct. 1978.
[4] M. Kameyama and T. Higuchi, "Design of Dependent-Failure-Tolerant Microcomputer Systems Using Triple-Modular Redundancy," *IEEE Trans. Computers*, vol. 29, no. 2, pp. 202-205, Feb. 1980.
[5] H. Kim and K.G. Shin, "Modeling Externally-Induced Faults in Controller Computers," *Proc. 13th IEEE/AIAA Digital Avionics Systems Conf.*, pp. 402-407, Phoenix, Ariz., Oct. 1994.
[6] H. Kim and K.G. Shin, "On the Maximum Feedback Delay in a Linear/Nonlinear Control System with Input Disturbances Caused by Controller-Computer Failures," *IEEE Trans. Control Systems Technology*, vol. 2, no. 2, pp. 110-122, June 1994.
[7] H. Kim and K.G. Shin, "Design and Analysis of an Optimal Instruction-Retry Policy for TMR Controller Computer," *IEEE Trans. Computers*, vol. 45, no. 11, pp. 1,217-1,225, Nov. 1996.
[8] H. Kim, "Design and Evaluation of Real-Time Fault-Tolerant Control Systems," PhD thesis, Univ. of Michigan, Ann Arbor, July 1994.
[9] V.B. Pradsad, "Fault-Tolerant Digital Systems," *IEEE*, pp. 17-21, Feb. 1989.
[10] K.G. Shin and H. Kim, "A Time Redundancy Approach to TMR Failures Using Fault-State Likelihoods," *IEEE Trans. Computers*, vol. 43, no. 10, pp. 1,151-1,162, Oct. 1994.
[11] D.P. Siewiorek, V. Kini, and H. Mashburn, "A Case Study of C.mmp, Cm*, and C.vmp: Part I—Experiences with Fault Tolerance in Multiprocessor Systems," *Proc. IEEE*, vol. 66, no. 10, pp. 1,178-1,199, Oct. 1978.
[12] J.F. Wakerly, "Transient Failures in Triple Modular Redundancy Systems with Sequential Modules," *IEEE Trans. Computers*, vol. 24, no. 5, pp. 570-573, May 1975.
[13] X.-Y. Zhuo and S.-L. Li, "A New Design Method of Voter in Fault-Tolerant Redundancy Multiple-Module Multi-Microcomputer System," *Digest of Papers, FTCS-13*, pp. 472-475, June 1983.

Counting Two-State Transition-Tour Sequences

Nirmal R. Saxena and Edward J. McCluskey

Abstract—This paper develops a closed-form formula, $f(k)$, to count the number of transition-tour sequences of length k for bistable machines. It is shown that the function $f(k)$ is related to Fibonacci numbers. Some applications of the results in this paper are in the areas of testable sequential machine designs, random testing of register data paths, and qualification tests for random pattern generators.

Index Terms—Transition-tours, sequential machine testing, Fibonacci numbers, checking experiments, testable synthesis.

1 Introduction

A *transition-tour sequence* is a binary sequence that includes all four transitions between adjacent binary bits. For example, 01100 is a transition-tour sequence because it has all four transitions $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$, and $0 \rightarrow 0$ between adjacent bits. The definition of transition-tour is consistent with the general definition of transition tours defined for finite state machines in [1], [2], [3], [4], [5]. We want to find a function, $f(k)$, that counts the number of distinct length k transition-tour sequences.

An application of transition-tour sequences in testing registers (for data path designs) and memory elements for finite state machines has been shown in [6]. Transition-tour sequences relate to checking experiments [7], and checking experiments can be used for an exhaustive test of sequential machines. Registers and memory elements in data path designs [6] generally model the behavior of D flip-flops or D latches [8]. In [6], transition-tour sequences were called checking sequences. There were two reasons for this change in terminology. First, the use of transition-tour sequence is consistent with earlier definitions in references [1], [2], [3], [4], [5]. Second, the term checking sequence is likely to be considered a synonym for checking experiments. Although it is argued in [6] that checking sequences for D-latches (without considering clock signals) are checking experiments, the consideration of clock signal as an explicit input to the D-latch (as shown in [9]) results in checking experiments that are different from checking sequences (as defined in [6]). However, it is important to point out that transition-tour sequence is an important part of the checking experiment because it includes all state-transitions.

The sequence 01100 is a shortest transition-tour sequence that takes the D latch or flip-flop (Fig. 1) from an initial state (0) through all four transitions. The other three shortest transition-tour sequences are: 10011, 00110, 11001.

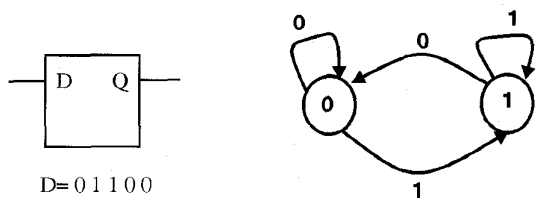

Fig. 1. D flip-flop/latch transition-tour sequence.

• *The authors are with Center for Reliable Computing, Department of Electrical Engineering, Stanford University, Stanford, CA 94305. E-mail: nsaxena@abel.mti.sgi.com.*

Manuscript received Aug. 26, 1994; revised Nov. 16, 1995.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96082.