Enhanced Performance Capability in a Dual Redundant Avionics Platform - Fault Tolerant Scheduling with Comparative Evaluation

Archana Sreekumara, Swetha K, Annam Swetha, Radhamani Pillay V

Amrita Vishwa Vidyapeetham, Coimbatore 641112, India

Abstract

For mission critical applications like Avionics, dependability is to avoid consequences of catastrophic results. Traditionally fault tolerance is implemented using hardware redundancy, the higher the redundancy, greater the cost and possibilities of more failures occurring. In this paper, an adaptive fault tolerant scheduling mechanism developed earlier with augmented performance capability and online fault recovery for a dual redundant system has been extended for an avionics mission system. An algorithm has been developed, simulated and evaluated on the practical case study vis-à-vis a traditional dual redundant system. This paper also proposes an extension of earlier scheme to schedule arrival of either/critical and non-critical aperiodic tasks. The augmented scheme helps to achieve full functionality when no fault occurs, a fail safe mechanism for a single fault and performance metrics highlights the better computational performance. It elucidates that the use of this adaptive model leverages better in terms of enhanced performance and throughput compared to the existing dual redundant systems.

Keywords: Real time Scheduling; Fault Tolerance; Safety Critical System; Avionic Missile System

1. Introduction

Multiprocessor architectures are imperative to tackle complex applications like avionics while considering the computational power and modularity issues. The traditional hardware redundancy makes the cost and complexity of
the system higher. Such redundant units being on hot standby may not necessarily seem to have an efficient use of computational resources during fault free operation and computing resources are thus underutilized. This paper examines the exploitation of such resources and simultaneously guarantees dependability assurance during fault conditions. The proposed model in\cite{1} explores this paradigm for fault tolerance in a dual redundant system with resource augmentation when there is no fault. This model has been extended for a case study of an avionics mission system. The main functions of the avionics mission system are the timely attack of the enemy and to maintain the fuel/weight ratio. An algorithm has been developed called Enhanced Fault Tolerant Scheme (EDFS) and simulated for fault tolerant real time scheduling with critical and non-critical aperiodic arrivals. The scheme ensures improved functionality under fault free operation and online fault recovery for single point permanent failure of any one system. The performance has been evaluated and compared vis-a-vis a traditional dual redundant system (DRS). The rest of the paper is organized as follows. Section 2 presents the literature survey and Section 3 gives the background study of scheduling of fault tolerant real time systems and case study of the avionic mission system. The system model with the proposed EDFS approach has been explained in Section 4, along with DRS. Sections 5 and 6 deal with analysis, simulation and performance evaluation. The conclusion and future scope are given in Section 7.

2. Literature survey

The attributes of dependable and reliable computing is given by laprie\cite{2}. In the scenario of fault tolerance, the main functions like error detection, fault diagnosis, and recovery are combined with masking by A.Avizienis\cite{3}. The fault modeling algorithm in multiprocessor environment was put forward by Mosse et al\cite{4}. Ramamritham and stankovic put forward new paradigms of static table-driven scheduling, static priority preemptive scheduling, dynamic planning-based scheduling, and dynamic best effort scheduling\cite{5}. The schedulability analysis techniques and hard real-time scheduling algorithms for homogeneous multiprocessor systems are discussed in\cite{6}. Whereas the quick recovery from failure was described by Krishna and Shin\cite{7} in their work. Oh and Son\cite{8} proposed a scheme for enhanced fault tolerance in static real time systems which was later extended for scheduling multiple tasks in minimum number of processors using RM scheduling scheme\cite{9}. Buttazo discusses basic approaches for designing predictable computing systems for critical control applications\cite{10}. Sprunt manifests an algorithm that guarantees deadlines for critical aperiodic tasks and responsiveness of soft aperiodic tasks\cite{11}.

Primary backup fault-tolerance strategy for scheduling real-time tasks on multiprocessors, a dynamic algorithm is described by Manimaran\cite{12}. Timeliness and criticality are integrated into fault tolerance by Mahmud Pathan\cite{13}. New paradigms for adaptive fault tolerance for enhanced resource management have been dealt in\cite{14,15}, by effectively utilizing the existing resources. Paradigm for handling aperiodic tasks in the context of automotive systems were dealt in\cite{16}. One mission critical application, avionics mission system, developed by the Australian Government, Department of Defence in the year 2006\cite{17} considers number of commonly used approaches for task scheduling. A description of the environment in which the mission operates and the functional description of the requirements imposed has been given in\cite{18}.

3. Background study

Dependability of a computer system can be put forward as the quality of service it provides\cite{2}. Fault tolerance for guaranteeing dependability is an important aspect of computer system, use of multiprocessor systems emphasis requirement of fault tolerance due to the increase in number of components used\cite{14}. Scheduling in multiprocessors involves global scheduling and partitioned scheduling and other scheduling techniques\cite{3}. A mixed task set of critical and noncritical tasks require periodic task allocation. The scheduling using a mixed scheduling approach is done by considering online adaptation environments. Based on the application and the type of task graph, the task allocation and scheduling is user initiated. Mission critical systems employ Static table driven approach. A table is constructed offline after identifying the start time and completion time of each task and tasks are dispatched according to this table. The use of Dynamic online best planning approach\cite{19} for scheduling in run time allows scheduling aperiodic arrivals and faults. The mixed task scheduling, during run time predictively checks for the
feasibility of schedule. The new schedule created after the arrival of the aperiodic tasks contains both previously guaranteed tasks as well as the arriving aperiodic tasks.

Redundancy helps to achieve fault tolerance, Avizienis. Hardware and software redundancy techniques ensure fault tolerance. The two main hardware redundant fault tolerance techniques are voter comparators and watchdog timers. Fault masking is a technique that prevents faults in a system from introducing errors. Examples include error correcting memories and voting. Triple Modular Redundancy is one of the commonly used fault masking. The process of regaining operational status via reconfiguration even in the presence of faults is termed as fault recovery. In the proposed method, when a fault occurs, special methods are required to switch the tasks to the spare processor or to switch out the faulty processor.

Performance metrics like Effective Utilization \(U\), signifies how far the computational resources are properly utilized, over all execution time of the process can be denoted by Process Execution Time \(C\), Average Response Time \(R\), soft aperiodic tasks executes so that the average response time are minimized, Number of tasks that missed the deadline indicated by Deadline Miss Ratio \(D\) have been effectively employed for evaluating the performance.

3.1 Case study: Avionic Mission System (AMS)

The Avionic Mission System is represented as real time embedded system having multiple processors, sensors and actuators. The functionality of the system is given by a task set with dependencies. The timely actions of such a system are important. Maintaining fuel/weight ratio during the operation seems to be a great deal of concern. Increasing the load or length of execution of tasks will cause an increase in the fuel consumed, proper scheduling and execution of tasks will help to maintain fuel/weight ratio at optimum levels.

Basic Tasks - The HUD, MPD displays, display the aircraft flight data and other important parameters. The threat response display, displays whether there is any threat or not. Weapon selection is an aperiodic task for selecting the required weapon. Control Tasks - Target Sweetening sweetens the target before the weapon release Weapon trajectory task calculates the trajectory for the weapon that is selected. HOTAS bomb button releases the weapon in CCIP mode. Monitoring Tasks - Monitors various conditions of the aircraft. Radar tracking tracks the signal from the given radar frequency and Target tracking tracks the targets. Weapon trajectory is re-evaluated using target check. The Radar Warning Receiver (RWR) is a threat warning sensor. Actuating Tasks - Based on the command, weapon release command releases the weapon at appropriate time in Auto mode as well as in CCIP mode. All task mentioned are periodic tasks except for weapon selection and hotas bomb button.

4. Approach

Based on the model developed in an algorithm has been developed to schedule the avionics mission task set for fault free and fault conditions which include critical and non-critical aperiodic arrivals. This algorithm called Enhanced Fault Tolerant Scheme (EDFS) employs the task allocation and scheduling for periodic and aperiodic tasks both critical and non-critical in the avionics environment. A comparison with the fault tolerant scheduling of a traditional dual redundant system DRS has been explored. A generic system based on has been considered for Implementing the EDFS.

Enhanced fault tolerant scheme creates a static schedule for each of the processor; tasks are scheduled in accordance to precedence. Synchronization for the execution of tasks are obtained from an external manager module called synchronization and fault check manager(SFM). The scheme proceeds by scheduling critical tasks on both processors while the aperiodic tasks are shared among the processors, so that the available slack margin has improved, which further helps to decrease deadline misses. Aperiodic task arrivals are handled in such a way that these tasks are scheduled without missing the deadlines of other tasks and are meeting all constraints. Aperiodic
critical tasks are scheduled as soon as there arrival on both processors while non-critical aperiodic tasks are scheduled on the pre assigned processor during the slack time available. Precedence constraints are considered to be important in the context of avionics mission system, rather than the fast execution of the tasks timely execution of the tasks are considered to important in safety critical systems.

4.1 Objective

- Development of EDFS and DRS algorithms.
- Simulation.
- Performance evaluation of the EDFS vis-a-vis DRS.

4.2 Assumptions

- The non critical tasks are considered to be preemptable.
- The worst case execution time for the tasks is being taken which includes all time overheads required for all pre-emption and communication.
- The watch dog timer detects the processor failures with a bounded latency.
- Exists fixed time interval between aperiodic tasks.
- The hinder to redundancy depends on the parallizable tasks that are present in the system.
- Critical aperiodic task has higher priority than other critical tasks.

4.3 Task Architecture

<table>
<thead>
<tr>
<th>Sl. No</th>
<th>Tasks</th>
<th>Nature of Tasks</th>
<th>Release Time units</th>
<th>Execution Time units</th>
<th>Deadline Time units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aircraft Flight Data</td>
<td>C</td>
<td>0</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>Steering</td>
<td>C</td>
<td>8</td>
<td>6</td>
<td>80</td>
</tr>
<tr>
<td>3</td>
<td>Radar Tracking</td>
<td>C</td>
<td>14</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Poll RWR</td>
<td>C</td>
<td>16</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>Threat Response</td>
<td>C</td>
<td>18</td>
<td>3</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HUD Display</td>
<td>NC</td>
<td>27</td>
<td>6</td>
<td>77</td>
</tr>
<tr>
<td>7</td>
<td>MPD Display</td>
<td>C</td>
<td>33</td>
<td>8</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>Target Tracking</td>
<td>C</td>
<td>21</td>
<td>4</td>
<td>61</td>
</tr>
<tr>
<td>9</td>
<td>Target Sweetening</td>
<td>NC</td>
<td>25</td>
<td>2</td>
<td>65</td>
</tr>
<tr>
<td>10</td>
<td>Auto/CCIP toggle</td>
<td>NC</td>
<td>41</td>
<td>1</td>
<td>241</td>
</tr>
<tr>
<td>11</td>
<td>Weapon Selection</td>
<td>NC</td>
<td>42</td>
<td>2</td>
<td>241</td>
</tr>
<tr>
<td>12</td>
<td>Weapon Trajectory</td>
<td>C</td>
<td>44</td>
<td>7</td>
<td>143</td>
</tr>
<tr>
<td>13</td>
<td>Control Task</td>
<td>NC</td>
<td>51</td>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>14</td>
<td>HOTAS bomb button</td>
<td>C</td>
<td>53</td>
<td>2</td>
<td>132</td>
</tr>
<tr>
<td>15</td>
<td>Weapon Release</td>
<td>C</td>
<td>55</td>
<td>1</td>
<td>62</td>
</tr>
</tbody>
</table>

The above table gives the task set of AMS with its task attributes. Criticalities of tasks are based on the effects that a task will produce in the performance of a system on missing of its deadline. Critical tasks are those which will produce catastrophic effects while non critical tasks degrade the performance of the system on missing its deadlines. All the tasks from $\tau_1$ to $\tau_5$ are critical tasks. The task $\tau_5$ is a critical task performed after scheduling $\tau_5$, it gives a "bird's eye view" of the situation. In $\tau_8$, the threat data will be superimposed on the digital map. Whereas the other display is a non critical display that is done by the task $\tau_6$. The task $\tau_{15}$ target sweetening, where the target is improved using the target designator control is classified as non critical task. The auto/CCIP toggle
switch receives the input from the aircrew to select between automatic launching of the weapon or manual launching of the weapon. The task \( \tau_{10} \) performs the function of Auto/CCIP toggle switch, which is non critical. The task \( \tau_{11} \), weapon selection is a soft aperiodic task. \( \tau_{12}, \tau_{14}, \tau_{15} \) are critical tasks. The task \( \tau_{14} \) is the critical aperiodic task which is to be executed only after the execution of task \( \tau_{13} \) due to precedence constraint. The task \( \tau_{13} \) is the periodic task which is non critical. While implementing for aperiodic arrivals this work has being extended with changes in execution time. Precedence constraints are ensured by the static order scheduler, by giving its marching orders to each processors before the commencement of the system.

The fig. 1 represents the fault tolerant model for avionics mission system with two processors and fault check manager. The static table driven scheduling is used during the normal condition while dynamic online approach is used during fault occurrence.

Both the processors continuously send a signal to the Synchronization and Fault check Manager (SFM). The critical tasks of set are being signalled by SFM, to achieve synchronization between the critical tasks in both the processors. The tasks to be executed by that particular processor are allocated in the Local Task Table. SFM maintains a global task table and it updates the table during runtime. The table consists of all the tasks of AMS with its attributes. When the SFM fails to get the signal from one of the processor, it alerts the other processor to enter the fault mode and resumes the execution of tasks. Performance of the system during the occurrence of permanent faults dealt, while possibilities of occurrence of other faults like intermittent and temporary faults are not negligible.

**The synchronization fault check manager (SFM) operation**
Input: The alive signal from the two processors
Output: The synchronization signal to both processors
1) while true
2) check the alive signal from both processors
3) signal the starting of first critical periodic task
4) end

**Algorithm for enhanced scheduling scheme**
Input: \( \tau_a, \tau_b \) are the set of tasks allocated to both processor 1 and processor 2
Output: Enhanced scheduling scheme in normal mode
1) for \( i=1 \) to \( n \) do
2) check for the ‘start’ signal from SFM for the execution of first critical task in both processors
3) if (start signal from the SFM) then
4) start the first task in both processors
5) continue the rest of the tasks
6) if (non critical aperiodic task) then
7) add task to processor 1
8) if (critical aperiodic task) then
9) add task to both processor1 and processor2
10) for each clock pulse do
11) trigger the transmission of the alive signal from processor1 and processor2 to SFM
12) end

Algorithm for scheduling arriving aperiodic tasks
Input: The aperiodic task with known time constraints
Output: The execution instant of the arrived task
1) for arriving aperiodic task do
2) check the criticality of the task
3) if (critical) then
4) schedule the task at the proper instant on both processors.
5) if (non-critical) then
6) if arrival time of non critical task < the slack time after the release of the task then
7) allocate tasks to processor1
8) execute it in the first slack period after the release of the task
9) if time of arrival of the critical task > the slack time after the release of the task then
10) allocate the task to processor2
11) execute task at the end of the all tasks after in processor1

Algorithm for fault mode operation
Input: Any kind of external or internal failure to the processor
Output: Fault tolerant mode in EDFS
1) for i=1 to hyper period do
2) if (alive signal from processor1 is absent) then
3) The SFM signals the processor 2 to allocate the non critical tasks of processor1 to processor2 without violating the precedence constraints
4) if (alive signal from processor2 is absent) then
5) The SFM signals the processor 1 to allocate the non critical tasks of processor2 to processor1 without violating the precedence constraints

Task allocation in dual processor scheme
Input: is the given task set, then
Output: in normal mode
1) for i = 1 to n do
2) schedule the tasks in both P1 and P2
3) end

The performance metrics compares the traditional dual redundancy scheme and the proposed model: Effective Utilization \( \eta \) is the summation of the ratios of execution time to time period, Process Execution Time \( C_p \) is the summation of the execution time of tasks, Average Response Time \( R_t \) is the difference between the release time and scheduling time of the soft aperiodic task, Deadline Miss Ratio \( D_t \) is the ratio of tasks that has missed the deadline to the total number of tasks in the system.
Fig. 2. DRS scheduling without aperiodic task.

Fig. 3. EDFS scheduling with soft aperiodic task.

Fig. 4. DRS scheduling with aperiodic and without fault
Fig. 5. EDFS scheduling with aperiodic tasks and without fault.

Fig. 6. DRS with fault occurring at the 10th time instant.

Fig. 7. EDFS scheduling with failure occurring in processor2 at 10th time instant.
5. Simulation

The Time Optimization Resource and Scheduling (TORSCHE) toolbox in the MATLAB is used for the task set scheduling and simulation for the proposed EDFS algorithm. Using the toolbox, one can easily investigate the application performance before the implementation. The SFM will monitor both processors in the proposed algorithm. The overall workload of 88.5% is assumed for the system, and table driven scheduling is implemented in both the processors.

Fig. 8 represents the MATLAB simulation of the Advanced Fault Tolerant Scheme (EDFS). Fig. 8(a) represents the normal scheduling of the EDFS when no aperiodic tasks are present. The tasks are scheduled in accordance to the criticality; critical tasks are duplicated in both the processors while the non-critical tasks are shared among the processors.

The fig. 8(c) shows the arrival of critical aperiodic task at the time instant 50 and the task is scheduled by EDFS scheme on both the processors at the same instant of time. It has an execution time of 2 units and deadline is 132 time units. The fig. 8(d) indicates the scheduling of soft aperiodic tasks arriving at the 54th time instant and is scheduled after the execution of all other tasks, in P1 processor.

![Fig. 8(a). Normal EDFS scheduling.](image)

![Fig. 8(b). EDFS scheduling with non critical aperiodic task.](image)
6. Evaluation

The Process Execution Time $C_p$ given in Fig. 9 shows the execution time for DRS and EDFS schemes as 56 and 47.5 time units respectively. The speed of execution has increased by 15.17% compared to the DRS scheme. Effective Utilization $U_e$ given in Fig. 10 shows an increase in slack time in EDFS. The normalized utilization for DRS and EDFS scheme are 95.9% and 88.5% respectively. The EDFS scheme has 8% more slack time for handling aperiodic arrivals. Average Response Time $R_t$ given in Table 2 shows the average response time of the EDFS has less average response time compared to DRM when a soft aperiodic task arrives. Deadline Miss Ratio $D_m$ for the DRS and EDFS are given in Table 3. In fault mode, the deadline miss ratio with respect to the change in the execution time of the critical aperiodic task is given in the table. From the same table, it can be inferred that the EDFS has improved deadline miss ratio compared to DRS.
Fig. 9. Execution time of DRS and EDFS schemes

Fig. 10. Effective Utilization of DRS and EDFS schemes

Table 2. Average Response Time of DRS and DFS schemes

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRS</td>
<td>11.6</td>
<td>11.6</td>
</tr>
<tr>
<td>EDFS</td>
<td>9.2</td>
<td>13.4</td>
</tr>
</tbody>
</table>

Table 3. Deadline Miss Ratio of DRS and ADSS schemes

<table>
<thead>
<tr>
<th>Approach Used</th>
<th>Normal Mode (time units)</th>
<th>Fault Mode (time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRS</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>EDFS</td>
<td>10</td>
<td>19</td>
</tr>
</tbody>
</table>

7. Conclusion

This paper presents an improved fault tolerant scheduling of a given Avionic missile System. The proposed EDFS scheme reduces the total execution time of the process by 15.17% over DRS under normal mode. By efficiently utilizing the extra slack margin available, response time of aperiodic tasks is reduced and thus the safety margins are improved. The functional and timing correctness have been guaranteed in the presence of faults without degradation. Future needs mandate higher complexity, functionality and reliability. Hence this approach can be effectively employed by extending it to m-processors.

References


