THE PROCESS OF SYNCHRONIZATION IN DUAL REDUNDANT FAULT-TOLERANT SYSTEM

Dong Liu, Chunyuan Zhang, Rui Li
Department of Computer, National University of Defense Technology, Changsha, Hunan 410073, P.R.China

Abstract: Synchronization is used in dual redundant fault-tolerant system to make two computers work jointly. It determines the work mode and controls the operations of the system. The paper presents a dual redundant fault-tolerant system and proposes its process of synchronization based on task. The synchronization treats task as the minimal operation unit. And it is implemented with the assist of dual-computer-controller and outer memory, the latter includes task buffer and global data region. Dual-computer-controller controls the input and output of tasks that are stored in task buffer. The switch in of backup computer is implemented by duplicating global data from global data region used by host computer. Additional resolutions for key points, such as the switch between two computers, are also put forward in conclusion. It is proved that the task-level synchronization can make two computers work in phase, and the system is applicable to critical fields requiring high dependability.

Key words: synchronization, dual redundant fault-tolerant system, task, dependability

1. INTRODUCTION

Computer systems, working in critical fields, such as bank and space, are commonly designed to be fault-tolerant to achieve high dependability, which generally increases exponentially by using linearly increasing redundant resources [1]. In engineering realizing, most of dual redundant systems are isomorphic redundant and classified to four elementary types, namely, cold backup, warm backup, hot backup and duplex. User applications in backup computer are suspended in the systems working in cold backup mode or

warm backup mode, as a result of which there is no process of synchronization between two computers. And in the systems working in duplex mode, additional comparer, or arbitrator, is adopted to compare and output the results of two computers; more dependable as the systems are, high cost and complex structure limits their applications. Hot-backup systems own the advantages of easy realizing and quick switching; however, synchronization is required to ensure proper cooperation between two computers in different situation [2-3]. This paper presents a dual redundant fault-tolerant system, which works in hot backup mode. With the intercommunion between two computers in the system, its task based synchronization is introduced.

The paper is structured as follows: after the introduction, the architecture of the dual redundant fault-tolerant system is presented. Afterwards, the process of synchronization is analyzed, including task control of the system. Finally, in Section 4 the conclusions are stated.

2. DESIGN FOR DUAL REDUNDANT FAULT-TOLERANT SYSTEM

For the purpose of decreasing the system complexity and weakening the coupling between two computers, intercommunion should be as little as possible [4-5]. As a result, the importing and exporting of one computer should be processed without the participation of the other one. In this way, two computers can make coarsely granular synchronization, named *the process of synchronization based on task* or *task-level synchronization*. According to the above statement, we designed a prototype working in hot backup mode. The system architecture is shown in figure 1.

The architecture includes two CPU-boards, one control-board and one connection-backboard, and the former two boards can be inserted into the connecter-backboard. CPU-board contains CPU, inner memory, interface circuit, self-detecting circuit and reset circuit. Therefore, one CPU-board, or computer-set, can be used as the minimal computer system that makes transactions independently. Control-board is used to administer two computer-sets and it consists of device interface, interface circuit, two outer memories and dual-computer-controller. Connection-backboard provides the channel of communications between CPU-boards and control-board; it comprises power circuit, protection circuit and bus of the system.

Outer memory is visible for outer devices and computer-sets. As one part of outer memory, task buffer stores the commands sent to the system by outer devices; and at the same time, it can be accessed by its corresponding computer-set. Since outer devices send commands to two outer memories at

the same time, the contents contained in two task buffers should be same when the system works in dual-computer-working state. Another part of outer memory is called global data region, which stores global data, such as system parameters or temporary results. Device-communication-controller controls the direction of data stream, which means that only one computer-set can export its results to outer devices, and that two computer-sets can import transaction, or task, at the same time. CPU 0/1 resets watchdog 0/1 periodically, which locates in dual-computer-controller and monitors the work state of computer-set. When both computer-sets are in well state, task-input-controller determines when tasks in task buffer are read by computer-set. If one computer-set fails, only the working computer-set can import tasks from task buffer, as is also handled by task-input-controller.

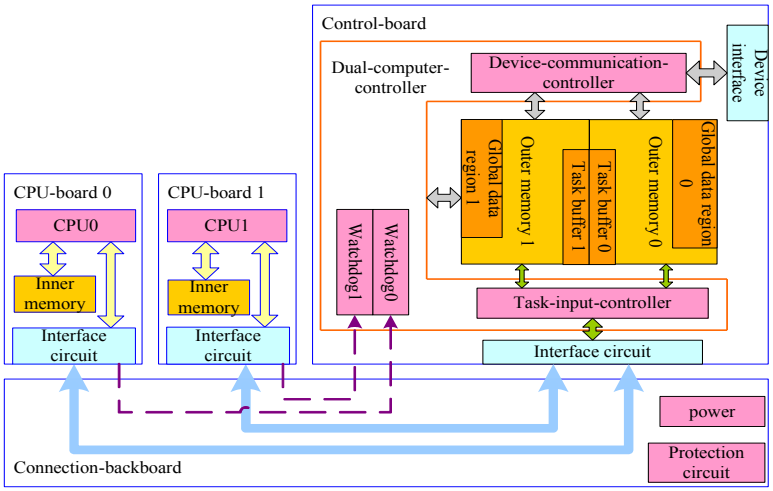

Figure 1. The architecture of the dual redundant fault-tolerant system

3. PROCESS OF SYNCHRONIZATION BASED ON TASK

If host computer fails, the system should switch from dual-computer-working state to single-computer-working state smoothly. Then, original backup computer should take over the transactions of the system with the status of new host computer. As a prerequisite, the process of switch should be invisible to outer devices, which are unaware of the architecture of the system. Therefore, two computer-sets must synchronize when they are both in well state, so as to realize the smooth switch if host computer fails.

In single-task dual redundant systems, such as dual SCMs system, the synchronization can be achieved easily, because two computers' input and output is simultaneous. However, it is different to our system, which runs multi-tasks operating system, since its data input and output is asynchronous. Moreover, with smaller data granularity, it gets more cost and worse extension ability [6-8]. In this paper, we present the task-level synchronization, which uses coarsely granular, different to signal-level synchronization and instruction-level synchronization.

For system users, task is a command sent to the system by outer devices. And it determines the anticipant operation and behavior. For the system itself, task is data with specific format, saved in the task buffer. Each task possesses its own unique signification; and there is no explicit affiliation between two different tasks. Task number or time stamp marks a task. Furthermore, task is provided with a priority, which means task with higher priority owns higher authority to be solved. Besides, task also has task head, task tail and task content. The format of task is shown in figure 2.

Figure 2. Task format

a) **Task input.** Outer devices send tasks as commands to the system through the device interface. Then, tasks are stored in task buffer temporarily, waiting for the read-requests sent by computer-set. It is the dual-computer-controller that controls when tasks are read from task buffer by computer-set.
b) **Task output.** If task has finished, computer-set write data, with given format, to outer memory accessed by outer devices.
c) **Task synchronization.** If two computer-sets run single-task operating system, own the same working frequency and import tasks simultaneously, they will complete the same task at the same time. However, in the system we present, with multi-tasks operating system running on both computer-sets, they are not likely to export their results simultaneously despite that they import a task at the same time. To tackle this problem, outer memory is divided into many blocks, each of which is used by a specific application in computer-set. In this way, there will be no interfering between two computer-sets; and the order of executing applications becomes unimportant.
d) **Switch out.** Once watchdog detects the failure of host computer, it will inform the dual-computer-controller to connect outer devices with backup computer, leading the system into the single-computer-working state. Besides, a reset signal is sent to the failed computer-set by

watchdog. The signal is also captured by dual-computer-controller to change the work state of the system. However, if backup computer fails, nothing occurs except that backup computer restarts.

e) **Switch in.** Supposing computer-set zero is in well state and computer-set one fails, watchdog one judges whether computer-set one has restarted by the periodical signals sent from CPU one. Affirming that computer-set one has restarted, dual-computer-controller will not permit computer-set zero to read new task from task buffer after the current tasks in computer-set zero finishes. Then, dual-computer-controller notifies computer-set zero to store global data to global data region zero, so that computer-set one could duplicate global data from global-data-region zero. Finally, dual-computer-controller permits both computer-sets to import and export tasks, leading the system into the state of synchronization between two computer-sets. In order to improve efficiency, the switch in of backup computer should arise at the time when host computer is in low workload; however, the system's dependability will decrease if it keeps on working in single-computer-working state for a long time.

The proposed synchronization is based on task and realized with the aids of dual-computer-controller and outer memory. And there is no monitor application running in computer-set and no communication between two computer-sets. Accordingly, with low overhead in operating system, the time of context switch decreases, as is favorable to improve the system efficiency.

4. CONCLUSIONS

It is a challenge to design dual-computer-controller for its bottle-neck position and complex functions. Therefore, in our system, dual-computer-controller is realized in FPGA with particular ability of reprogramming and reconfiguring [9]. TMR is also used to increase the reliability of key components in dual-computer-controller.

As mentioned above, once backup computer begins switching in and duplicates global data from global data region, used by host computer, to global data region, used by backup computer, host computer will stop importing new tasks. In this process, if outer devices send tasks with higher priority to the system, the system should stop duplicating data and respond to the tasks before switching in completes. Furthermore, if this kind of tasks comes forth frequently, backup computer would not complete switching in, holding the system in the state of single-computer-working for a long period, which brings the hidden trouble of system crash. However, the appearance of that kind of tasks generally indicates emergencies of the system, so they come forth infrequently. Besides, global data occupies little storage space, which decreases the time of duplicating

and expedites the process of switching in.

Watchdog being used to monitor computer-set, the count time of watchdog is distinctly the longest time that dual-computer-controller will take to become aware of computer-set's failure. In this period, dual-computer-controller believes that the system is in dual-computer-working state, but the fact is quite the reverse; as a result, the accesses from outer devices to outer memory fail and backup computer cannot export data to outer devices. However, the problem can be solved after original backup computer takes over the system.

It has been proved that our system, based on task-level synchronization, can make smooth switching between two computers. And, at the same time, high performance and reliability are preserved. It is applicable to the critical fields requiring high dependability. Furthermore, if a little change is made to the synchronization definition, it can also be used in duplex systems.

REFERENCES

1. MA Xiu-juan, CAO Xi-bin, MA Xing-rui, Reliability analysis and design of on-board computer system for small stereo mapping satellite, *Journal of Harbin Institute of Technology (New Series)*, 9(1), 79-81 (2002)
2. Surajit Dutta1, Sudip Dutta2, Riddhi Burman, et al., Design and Implementation of a Soft Real Time Fault Tolerant System, *S.K. Das and S. Bhattacharya (Eds.)*, IWDC 2002, LNCS 2571, 319-328 (2002)
3. Xiong-Fu Liu, Arthur Dexter, Fault-tolerant supervisory control of VAV air-conditioning systems, *Energy and Buildings*, 33, 379-389 (2001)
4. Vittoria de Nitto Personè, Vincenzo Grassi, An analytical model for a parallel fault–tolerant computing system, *Performance Evaluation*, 38, 201-218 (1999)
5. J.C. Campelo, F. Rodrı´guez, A. Rubio, etc., Distributed industrial control systems --- a fault-tolerant architecture, *Microprocessors and Microsystems*, 23, 103-112 (1999)
6. R. Al-Omari, A.K. Somani, G. Manimaran, An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems, *J. Parallel Distrib. Comput.*, 65, 595-608 (2005)
7. Koji Hashimoto , Tatsuhiro Tsuchiya, Tohru Kikuno, A new approach to fault-tolerant scheduling using task duplication in multiprocessor systems, *The Journal of Systems and Software*, 53, 159-171 (2000)
8. Douglas W. Caldwell, David A. Rennels, A Minimalist fault-tolerant Microcontroller Design for Embedded Spacecraft Computing, *The Journal of Supercomputing*, 16, 7-25 (2000)
9. John M. Emmert, Dinesh K. Bhatia, A Fault Tolerant Technique for FPGAs, *Journal of Electronic Testing, Theory and Applications*, 16, 591-606 (2000)