Mathematical & Computational Applications, Vol. 4, No. 2, pp. 175-184, 1999.
© Association for Scientific Research

AN APPROACH TO THE SUPPORT OF THE FAULT-TOLERANCE OF THE DOUBLE REDUNDANT COMPUTER CONTROL SYSTEMS

Rafig SAMEDOV
Pamukkale University, Camlik-20017, Denizli/TURKEY
Azerbaijan National Aerospace Agency, Azadliq av. 159, Baku/AZERBAIJAN

Abstract- Faults in computer control systems cause great economic losses and endanger human beings. In order to ensure control and monitoring tasks, the availability and safety of computer control systems in areas such as military, manufacturing plants, traffic, public health service, environment protection, banking and assurances have to be improved. The fault-tolerance of the computer control system is achieved by using the hardware and software redundancy and redundancy of processor time. In this paper, the structure of real time double redundant computer control system is updated. The highly efficient methods and device are designed for supporting of the fault-tolerance of the double redundant computer control system by using the hardware redundancy. The fault-tolerant system software enables identical application programs to work parallel on two reserved computers. The computational process executed in a computer control system is periodically interrupted at checkpoints by execution of the fault-tolerant procedure, which recognizes and eliminates the faulty computers. The recovery of computer control system is done automatically without interruption of the control services.

1. INTRODUCTION

The implementation of computer control systems (CCS) is extending over a wider and wider area and the control systems themselves are becoming highly advanced. Originally, systems were made for the use of a very limited scope or range of equipment, so that even when a system failure occurred, it was possible to prevent serious accidents and keep the plant running, by means of temporary replacement, to maintain control. However, along with the rapid advances in computer technology in both hardware and software, users have turned increasingly to control systems, requiring more sophisticated machinery over an ever-widening range. Malfunctions in control systems have thus come to invite accidents and vast expenditures of an unprecedented nature, making system reliability of vital importance.

The use of the physical system as a logic machine is based on the conventions that the values of physical variables are interpreted as the discrete values of logic variables, and that the speed with which transformations are carried out is limited by the physical properties of the hardware. The logic machine behaves in the specified manner as long as the parameters of physical components and the speed of operation remain within specified limits. However, it has been a common experience that unexpected out-of-specification physical changes in component parameters do occur in all kinds of hardware. They are usually called ***malfunction*** when the changes are temporary and ***failure*** when the changes are permanent. Their effect is to cause an unspecified and disruptive change of one or more logic variables of the logic machine. Such a change of logic values is called a ***physical fault*** [1].

The purpose of the fault-tolerance (FT) is to offer an alternate solution to the fault problem in which the detection of faults and the recovery to normal operation are carried out as internal functions of the system itself. FT is the unique attribute of a system, which makes it possible for the system to continue with its program-specified behavior as a logic machine after the occurrence of faults. It may be said that FT is the survival attribute of the logic machine because its purpose is to cause a return from error states back to specified behavior, thus assuring the survival of the information processing activities.

Existing up-to-day technologies do not allow creating absolutely reliable components for computers. Computer systems are designed and created by using existing components, which don't provide the necessary reliability. One of the solutions to the reliable problems is the creation of the fault-tolerant computer control systems (FTCCS).

There are many FTCCS already developed. Some of them are: STAR computer [2], SAPO computer [3], SAGE system [4], FTSC computer [5], C.vmp [6], PLURIBUS [7], ESS [8], Fault-tolerant computer system with three symmetric computers [9], FTMP [10], SIFT [11], The space shuttle computer [12] and DEDIX [13]. The historical progress and technical results are comprehensively summarized in the published papers [14-16]. FT is reached by using different methods of hardware, software and time redundancy [16]. The hardware redundancy is frequently used. In accordance with hardware redundancy, *N* copies of identical program are executed in *N* hardware canals. For example, STAR, FTSC, SAGE, where *N=2*, C.vmp, FTMP, SIFT, SAPO, where *N=3*, the Space Shuttle, where *N=4*, DEDIX, where *N* changes from *2* to *20*. However, in spite of a large variety of methods and provisions for FT, there is a need to new effective designs. Thus, this paper discusses the new approach to the support of FT of the double redundant computer control systems (DRCCS).

2. INVESTIGATION OF THE STRUCTURE OF DRCCS

Duplex architecture is one of the widely used methods for supporting of FT. This method was discussed at the first symposium entirely dedicated to the computer reliability questions [17]. Especially, a relevant paper for nowadays is by J.P.Eckert on "Checking circuits and diagnostic routines", describing the fault detection and duplication employed in ENIAC, BINAC and UNIVAC machines. The duplex architecture was used in structure of different systems [2, 4, 5, 7, 9, 13, and 18]. Existing duplex architectures may be evaluated in such a way:

1) Duplex architecture was realized at the component, block and processor levels. Having modern technology, it is not reasonable to use the redundancy lower than computer level. In addition, there are many computers made up of a single crystal. So, we have the architecture where separate computers turn into redundant modules. Such architecture has the important advantage having only a few numbers of communications in it. Thanks to this advantage, not only recovery mechanisms but also communication of hardware are simplified.

2) The dynamic redundancy was frequently realized. There are two types of the hardware redundancy [19]: a) Static redundancy: All redundant modules work in parallel and failures of separate modules are compensated by presence of redundant modules, b) Dynamic redundancy: The system consists of the working and redundant modules. After failure of the working modules the redundant module substitutes it. This type of redundancy requires the supplementary time for substitution of modules, which is limited in hardly real time systems.

3) The software methods were frequently used. The diagnostic and test programs were widely used in many DRCCS. It is widely accepted that the software methods work slower and less reliable than hardware methods.

Thus, the static hardware redundancy at the computer level increases the performance and reliability.

3. DESIGNING THE STRUCTURE OF DRCCS AND COMPUTATIONAL PROCESS

The structure of DRCCS for hardware implementation of FT is illustrated in Fig.1. Controlled object (CO) is the moving apparatus such as planes, satellites, space ships and other kinds of objects such as atomic power stations, continuous technological process etc., which are functioned in scale of hard real time. Control system (CS) is double redundant computer system.

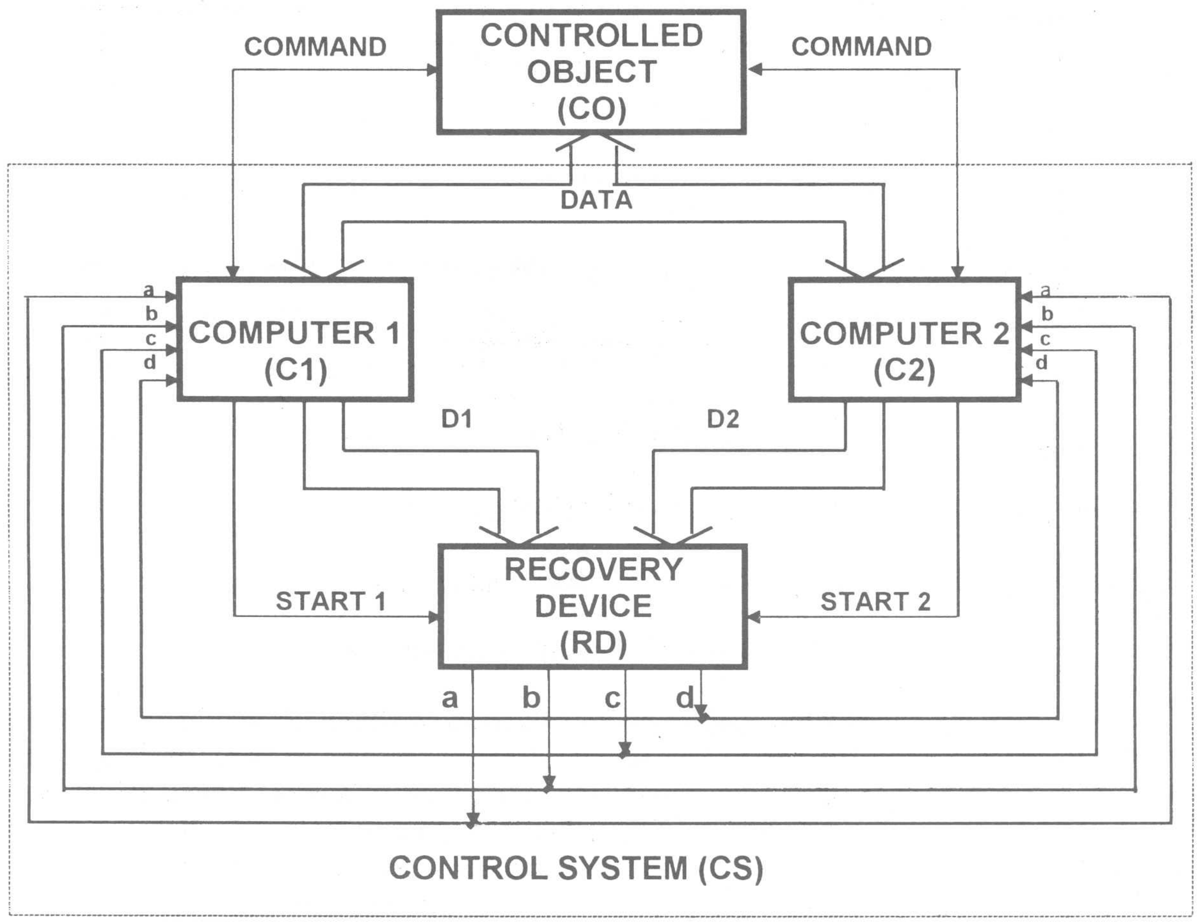

Fig.1 Structure of the Double Redundant Computer Control System

Data are received from CO to CS in parallel digital code by bus ***DATA*** in accompaniment control signal ***COMMAND***. Computer 1 (C1) and Computer 2 (C2) execute the identical serial programs on the base of received data according to computational process.

The basic principle of computational process organization is shown in Fig.2. Computational process (CP) consists of ***J*** computational cycles (CC). One of the principal properties of CP is determinate character of application task (AT) execution. The processor time is limited in each CC. It is required that summering processor time for execution of all AT and fault-tolerant procedure (FTP) in each CC was not more than planned. Consequently, the processor time for FTP is also limited.

Each CC consists of ***I*** logical segments (LS) as shown in Fig.2. The structure of LS is illustrated in Fig.3, which shows that the AT and checkpoint are executed in each LS. In checkpoint the FTP is executed.

We will now give the algorithm for supporting of FT. Suppose that the probability of occurrence of more than one fault during two LS is negligibly small. Every computer in a system with redundancy executes identical AT and the computational results (CR) are represent by a single value.

CP executed in a computer system is periodically interrupted at checkpoint by the execution of FTP. Computers send the CR to recovery device (RD) by bus ***D1*** and ***D2*** in accompaniment signals ***START1, START2*** and wait the answer from it (Fig.1).

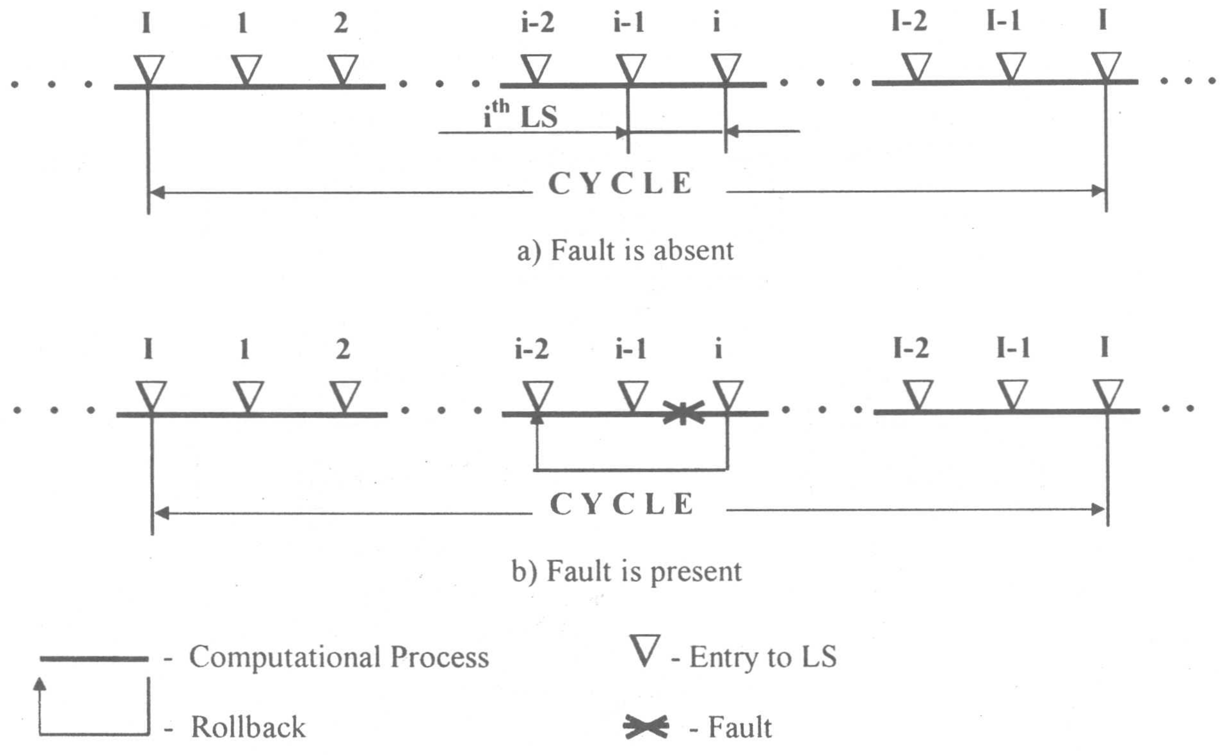

Fig.2 Structure of the Computational Process

CR of each job generated in all computers through communication of data between the computers and RD form the initial data set (IDS) which consists of $N=2$ elements. IDS is stored in RD and used to check the state of the entire system. In the absence of faults, the values of all the IDS elements agree, i.e. they are equal. In the presence of a fault, the value of one element differs from the other (disagreement appears).

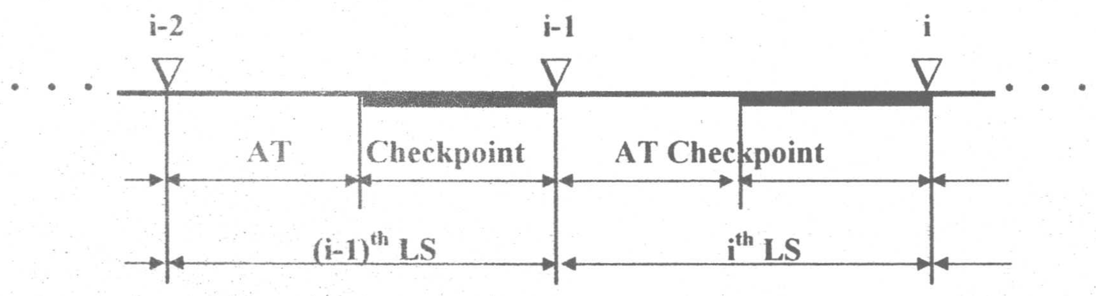

Fig.3 Structure of the Logical Segment

RD executes the FTP, which identifies the faulty computer. The results of the execution of FTP are sent to computers as control signals ***a*, *b*, *c*, *d*.** Computers choose one of following possible ways of continuation of CP:
1. ***a*** means that the fault in system is absent. Computers continue to execute the CP beginning with next LS,
2. ***b*** means that C2 is failure. C1 continues to execute the CP beginning with next LS, but C2 is turned off,
3. *c* means that C1 is failure. C2 continues to execute the CP beginning with next LS, but C1 is turned off,
4. ***d*** means that the fault is present in the system. For definition of fault types, computers roll to back to the beginning of ***(i-1)^th*** LS and repeat the execution of preceding LS.

After each LS, required data or control signals in parallel digital code are sent from CS to CO by bus ***DATA*** in accompaniment control signal ***COMMAND***.

4. DESIGNING THE RECOVERY DEVICE

Functional scheme of RD is shown in Fig.4. RD realizes the FTP, which consists of five steps:
1. Fault detection,
2. Definition of the fault types (malfunction or failure);
3. Recovery of the CP after malfunction;
4. Fault localization (defining the number of the faulty computer),
5. Reconfiguration of the system after failure.

The graphic model of FTP is illustrated in Fig.5. It is obvious that the classical sequence of execution of steps for FTP [20] differs from Fig.5. Namely, definition of the fault types and recovery of the CP after malfunction are realized before fault localization. After malfunction, the faulty computer is not localized but is masked. Recovery the CP after malfunction is realized in both computers. Fault localization is realized only after failure.

Now we will give the working principle for RD. All registers and the counter are reset by signal ***RESET*** and RD is ready to work.

4.1. Fault detection

RD detects the fault by checking the values of IDS elements for equality [20]. CR of execution of identical AT for ***i***th ***(i =1,...,I)*** LS enter from C1 and C2 to RD by busses ***D1*** and ***D2*** in accompaniment of control signal ***START1*** and ***START2*** (Fig. 3 and 4). The time diagrams for RD are shown in Fig. 6.

τ_1 is the delay element to cope with the CR of preceding LS [in this case, CR of $(i-1)^{th}$ LS in C1] to ***Reg.3,***

$$\tau_1 \geq \tau_{AND2} + \tau_{REG3} \qquad (1)$$

where τ_{AND2} is the delay time of ***AND2;*** τ_{REG3} is the write time to ***Reg.3.***

τ_2 is also the delay element for simultaneous data entry from ***Reg.1*** and ***Reg.2*** to ***X1*** and ***X1*** of ***COMPARATOR*** through ***Groups 1, 2 of AND*** and ***Groups 5, 6 of OR,***

$$\tau_2 \geq \tau_1 + \tau_{REG3} \qquad (2)$$

where τ_1 is defined from ***(1),*** τ_{REG1} is the write time to ***Reg.1.***

If ***X1=X2*** then ***Y1*** is high. Consequently, this high level signal enters to the exit of RD through ***AND4*** (which is open by high level signal of Q1) and forms ***a.*** If ***X1≠X2*** then ***Y2*** is high. Consequently, the state of ***COUNTER*** is changed, so that ***Q2*** becomes high. At the same time high level signal enters to the exit of RD through ***AND8*** (which is open by high level signal of ***Q1***) and forms ***d.***

4.2. Definition of the fault types

Suppose that data of last and preceding LS are stored in computers for the execution of AT. For definition of the fault types it is necessary to repeat the execution of the preceding LS and to check the repetition CR (RCR) for equality. If RCR are same it means that the reason of fault was malfunction. In opposite case, the reason of fault was failure. Let's discuss how RD executes this algorithm. Receiving ***d,*** both C1 and C2 roll to back and repeat the

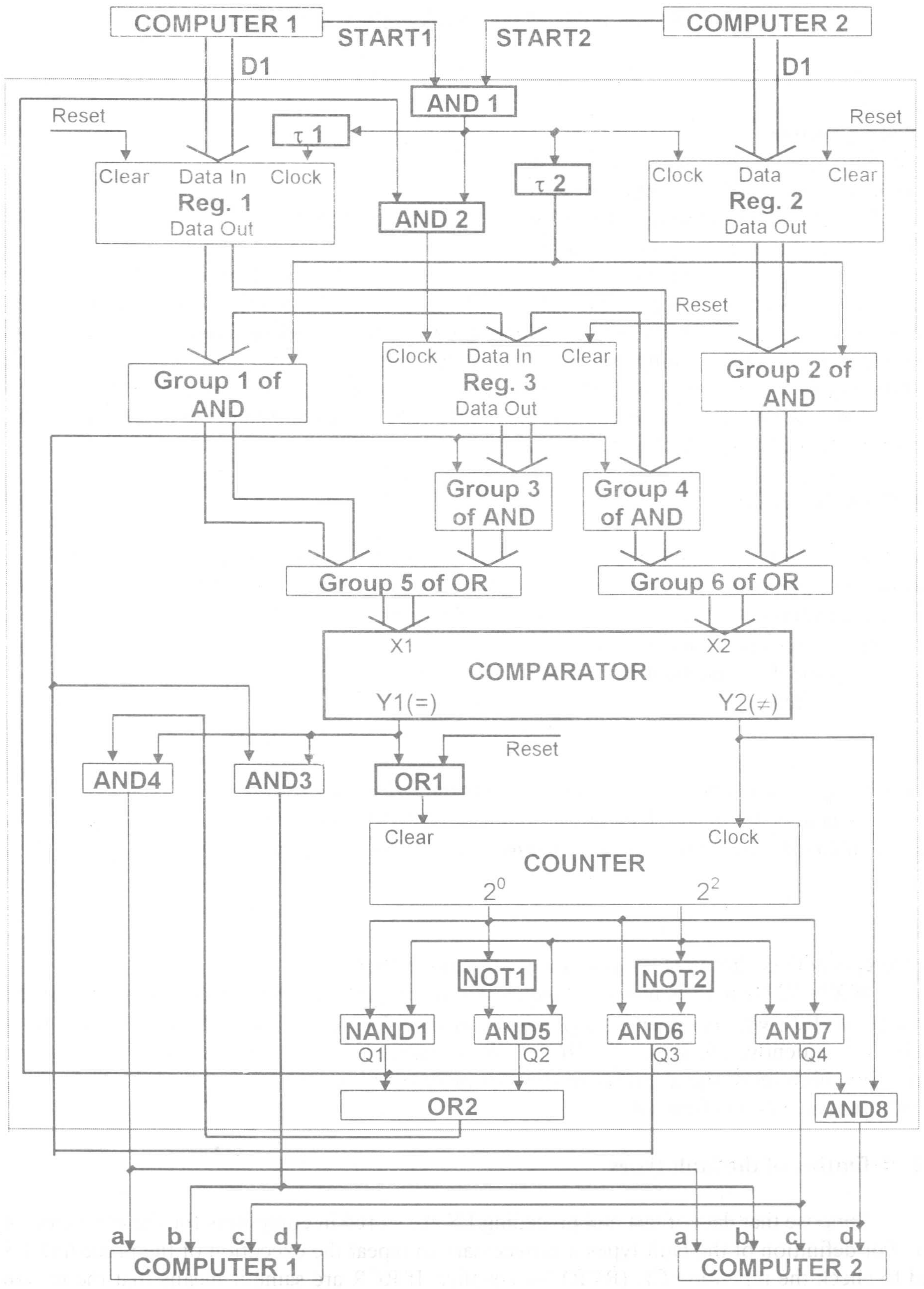

Fig.4. Functional Scheme of the Recovery Device

execution of AT of *(i-1)th* LS. The RCR enter to RD. In this case, the state of ***Reg.3*** does not change due to low level signal of ***Q1***. ***Reg.3*** stored the first CR of *(i-1)th* LS in C1. So, RCR enter from ***Reg.1*** and ***Reg.2*** through ***Groups 1, 2 of AND*** and ***Groups 5, 6 of OR*** to ***X1*** and ***X2*** of ***COMPARATOR.***

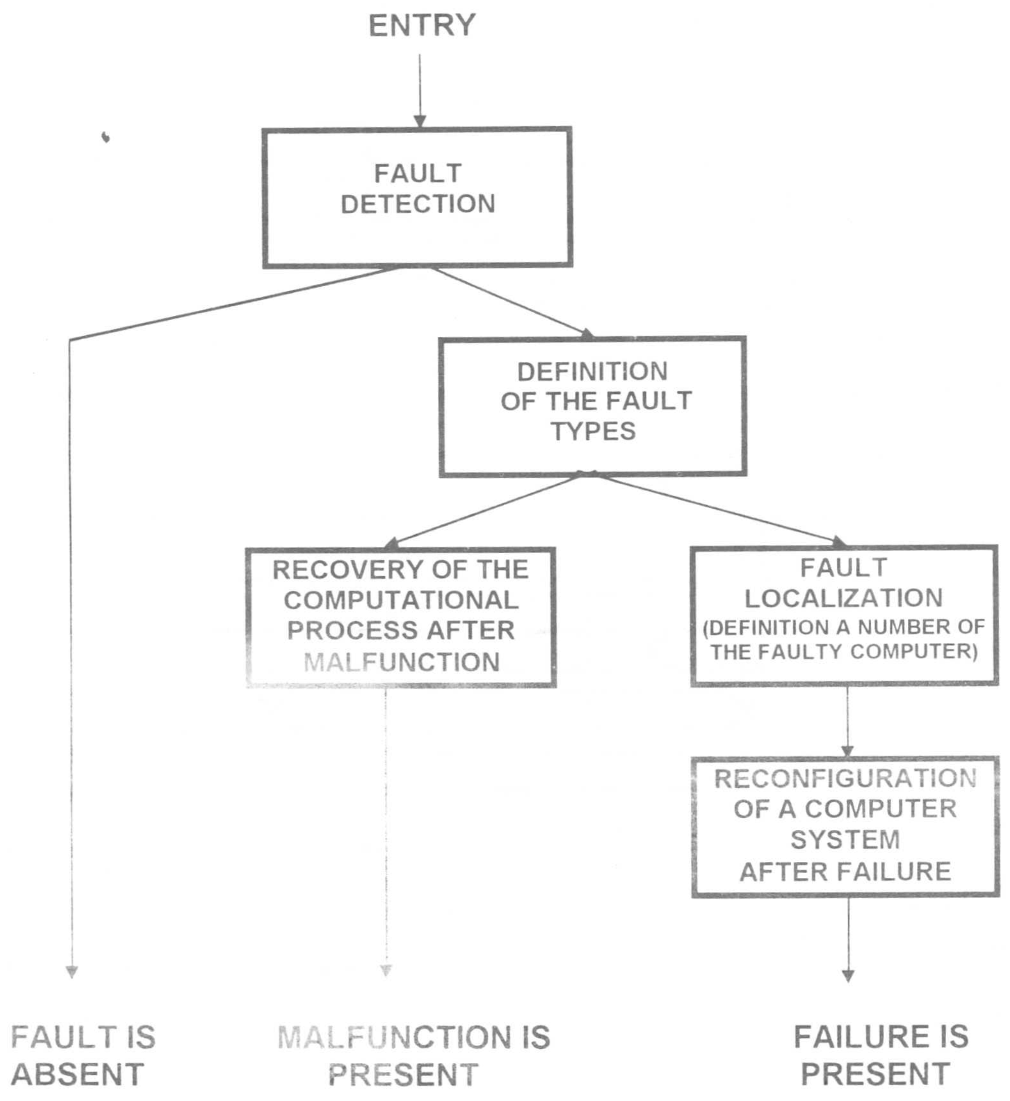
ENTRY
FAULT DETECTION
DEFINITION OF THE FAULT TYPES
RECOVERY OF THE COMPUTATIONAL PROCESS AFTER MALFUNCTION
FAULT LOCALIZATION (DEFINITION A NUMBER OF THE FAULTY COMPUTER)
RECONFIGURATION OF A COMPUTER SYSTEM AFTER FAILURE
FAULT IS ABSENT
MALFUNCTION IS PRESENT
FAILURE IS PRESENT

Fig.5 Graphic Model of the Fault-Tolerant Procedure

If ***X1=X2*** then ***Y1*** is high. Consequently, this high level signal resets the counter, passes to exit of RD through ***AND4*** (which is open by high level signal of ***Q2***) and forms ***a*** which means that the reason of fault was malfunction in one of the computers. If ***X1≠X2*** then ***Y2*** is high. Consequently, this high level signal changes the state of the Counter so that ***Q3*** becomes high which means that the reason of fault was failure in one of the computers.

4.3. Recovery of the CP after malfunction.

This step is realized by repetition of preceding LS in computers. RD executes this step during definition of the fault types. After malfunction the computers continue to execute the CP with ***ith*** LS.

4.4. Fault localization

A number of faulty computer is defined on the base of first and repetition CR of identical AT of ***(i-1)***th LS in one of the computers, which enter from ***Reg.3*** and ***Reg.1*** to ***X1*** and ***X2*** of ***COMPARATOR*** through ***Groups 3, 4 of AND*** and ***Groups 5, 6 of OR.***

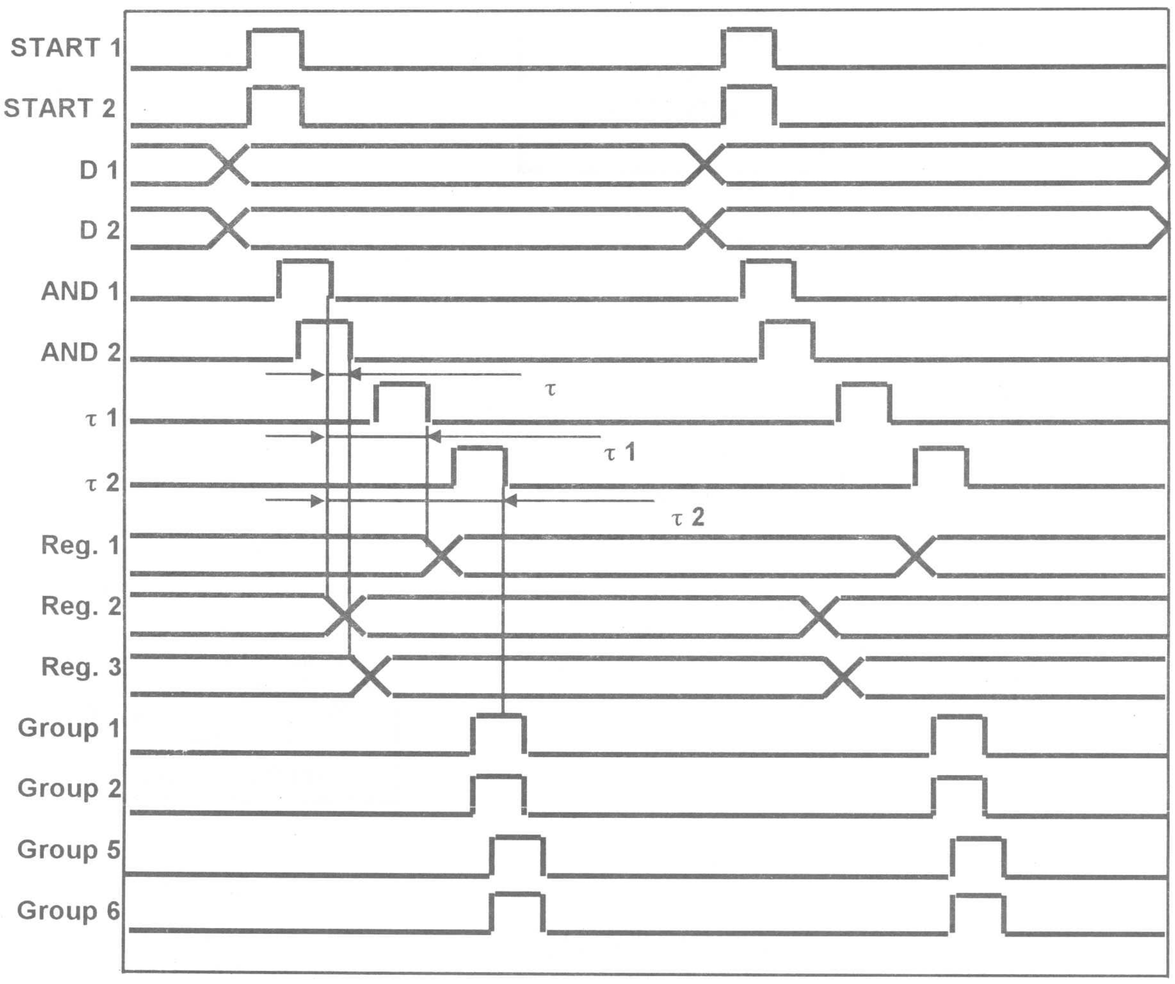

Fig. 6 Time Diagrams for Recovery Device

If ***X1=X2*** then ***Y1*** is high. Consequently, this high-level signal passes to the exit of RD through ***AND3*** (which is open by high level signal of ***Q3***) and forms ***b*** which means that the C1 is non-faulty and C2 is faulty. If ***X1≠X2*** then ***Y2*** is high. Consequently, this high level signal changes the state of ***COUNTER*** so that ***Q4*** becomes high which passes to the exit of RD and forms ***c*** which means that C1 is faulty and C2 is non-faulty.

4.5. Reconfiguration of the system after failure

The failure computer is turn off by non-faulty computer. Receiving control signal ***b (c)*** the C1 (C2) continues to the execution of CP with ***(i+1)***th LS and turns off C2 (C1). After reconfiguration, only one computer continues CP.

5. CONCLUSION

This paper gives a description of an approach to the support of the fault-tolerance of the double redundant computer control system in real time. The peculiarity of this approach is nontrivial structure of the system and computational process. In addition, proposed effective methods and designed integrated hardware - recovery device for execution of fault-tolerant procedure are the other principal results of this paper.

Double redundant computer control system has a simple structure, which consists of two computers and RD with minimal connection between them.

The organization of CP provides the repetition of maximum two LS (after malfunction) and minimum one LS (after failure) so that CO does not feel this delay.

Proposed effective methods form the nontrivial sequential of execution of the steps for fault-tolerant procedure. Namely, definition of the fault types (malfunction or failure) and recover the CP after malfunction are executed before fault localization. These methods allow supporting the fault-tolerance during two LS without using diagnostic and test programs. In addition, the memory is economized from storing different data, diagnostic and test programs.

Designed hardware - recovery device - increases the performance and reliability of system. Shortening the time of executing the fault-tolerant procedure due to recovery device increases the performance. The reliability is increased by decreasing the probability of occurrence of fault during the execution of fault-tolerant procedure.

Designed methods and hardware may be used not only for originally double redundant computer control systems but also for *N*-fold redundant systems after degradation till $N=2$. Beside it, this approach may be used for double redundant controlled objects.

As a result, we can say that the described approach may find the usage in different control systems.

6. REFERENCES

1. A. Avizienis, Fault-Tolerance: The survival attribute of digital system, Journal of the IEEE Trans. Computers, 66, 1109-1125, 1976.
2. A. Avizienis et al., The STAR (self-testing and repairing) computer: An investigation of the theory and practice of fault-tolerant computer design, Journal of the IEEE Trans. Computers, C-20, 1312-1321, 1971.
3. J. Oblonsky, Some features of the Czechoslovak relay computer SAPO, Journal of the Nachrichtentechnische Fachberichte, 4, 73-75, 1956.
4. R. R. Everett, C. A. Zraket and H. D. Benington, SAGE - A data-processing system for air defenses, In Proc. Eastern Joint Computer Conf., Washington, 148-155, 1957.
5. D. D. Burchby et. al., Specification of the fault-tolerant spaceborne computer (FTSC), In Proc. 1976 Int. Symp. Fault-Tolerant Computing, Pittsburgh, PA, 129-133, 1976.
6. D. Siewiorek, M. Canepa, and S. Clark, C.vmp: The architecture of a fault-tolerant multiprocessors, In Proc. 1977 Int. Sym. Fault-Tolerant Computing, Los Angeles, CA, 1977.
7. S. M. Ornstein et al., PLURIBUS - A reliable multiprocessor, In AFIPS Conf. Proc., 44, 551-559, 1975.
8. R. W. Downing, J. S. Nowak and L. S. Tuomenoksa, No. 1 ESS maintenance plan, Bell Syst. Tech. J., 43, 1961-2019, 1964.
9. H. Ihara, K. Fukuoka, Y. Kubo, S. Yokota, Fault-tolerant computer system with three symmetric computers, Journal of the IEEE Trans. Computers, 66, 1160-1170, 1978.
10. A. L. Hopkins, T. B. Smith, J. H. Lala, FTMP - A highly reliable fault-tolerant multiprocessor for aircraft, IEEE Trans. Computers, 66, 1221-1239, 1978.

11. J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith. R. E. Shostak, C. B. Weinstock, SIFT- Design and analysis of a fault-tolerant computer for aircraft control, Journal of the IEEE Trans. Computers, 66, 1240-1255, 1978.
12. J. R. Skalaroff, Redundancy management technique for space shuttle computers. IBM J. Res. Develop., 20, 20-28, 1976.
13. A. Avizienis et al., The UCLA DEDIX system. A distributed tested for multiple version software, In Dig. 15th Annu. Int. Symp. Fault Tolerant Computing (Ann Arbor, Ml., June 19-21, 1985), 126-134, 1985.
14. A. Avizienis, Fault-tolerant computing - progress, problems and prospects, Proc. of IFIP Congress 77, Toronto, 405-420, 1977.
15. S. Y. H. Su and R. J. Spilman, An overview of fault-tolerant digital system architecture. In AFIPS Conf. Proc., 46, 19-26, 1977.
16. A. Avizienis, J. C. Laprie, Dependable computing: From concepts to design diversity, Journal of the IEEE Trans. Computers, 76 629-638, 1986
17. Session 14: Symposium - Diagnostic programs and marginal checking for large scale digital computers, in Convention Record of the IRE 1953, Nat. Convention, Part 7, New York, N.Y., 48-71, 1953.
18. A. E. Cooper and W. T. Chow, Development of onboard space computer systems, IBM J. research and development, 20, 5-19, 1976.
19. A. Avizienis, Architecture of fault-tolerant computing systems, Proc. Int. Symp. on Fault-Tolerant Computing, 1975.
20. R. Ya. Samedov, A. Ciftci, Investigation and designing a fault-tolerant procedure for control computer system, Journal of engineering sciences, Pamukkale University, Turkey, 4, 589-601, 1998.