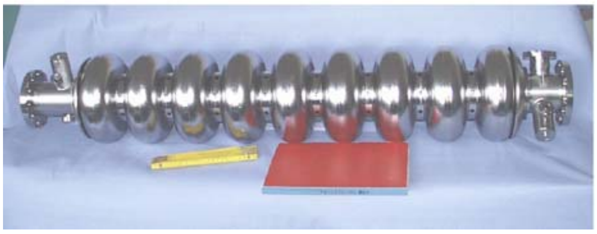
EU contract number RII3-CT-2003-506395 CARE-Conf-06-070-SRF

SRF



A NOVEL APPROACH FOR OPERATING SYSTEMS PROTECTION AGAINST SINGLE EVENT UPSET

B. Świercz, D. Makowski, A. Napieralski

Technical University of Łódź, POLAND.

Abstract

Modern high-energy physics experiments require sophisticated and complex control systems. The control systems should be able to tolerate radiation generated by high-energy accelerators and thus reliability is important feature. Reliability of control systems depends on hardware and software quality. Hardware solutions are the most effective techniques to protect system against radiation influence. Commercial of the shelf (COTS) elements are used often and protection mechanisms are moved from hardware to software layer, due to costeffective design. This paper highlights the new approach to protect systems on software level. The protection against soft errors is assured by operating system that is transparent to other applications.

Contribution to the MIXDES 2006, Gdynia (Poland)

Work supported by the European Community-Research Infrastructure Activity under the FP6 “Structuring the European Research Area” programme (CARE, contract number RII3-CT-2003-506395)

A NOVEL APPROACH FOR OPERATING SYSTEMS PROTECTION AGAINST SINGLE EVENT UPSET

B. Świercz, D. Makowski, A. Napieralski
Technical University of Łódź, POLAND

KEYWORDS: Single Event Upset, Radiation Environment, sCore, Fault Tolerant System, Virtual Memory, IA-32

Abstract: Modern high-energy physics experiments require sophisticated and complex control systems. The control systems should be able to tolerate radiation generated by high-energy accelerators and thus reliability is important feature. Reliability of control systems depends on hardware and software quality. Hardware solutions are the most effective techniques to protect system against radiation influence. Commercial of the shelf (COTS) elements are used often and protection mechanism are moved from hardware to software layer, due to cost-effective design. This paper highlights the new approach to protect systems on software level. The protection against soft errors is assured by operating system that is transparent to other applications.

INTRODUCTION

X-Ray Free Electron Laser (X-FEL)[1, 2] is currently designed at DESY Research Center in Hamburg. X-FEL is a 4^{th} generation of light source [3, 4]. Total length of the X-FEL will be 3.4 km, whereas the length of the main linear accelerator is equal to 2.1 km. Due to the complexity, distributed control system is necessary to control all subsystems of X-FEL's modules. In the past accelerating machines where consisted of two tunnels, one for the main accelerator and the other one for control electronics. The accelerator was shielding using concrete and lead, therefore the electronics were not subjected to radiation. The X-FEL consists only of one tunnel shared between accelerator and control electronics. Some part of electronic equipment (DSP and FPGA boards, embedded computers and microprocessors systems) designed for control and data acquisition will be placed inside accelerator tunnel in the nearest proximity of cavities (the main part of accelerator). Parasitic radiation is produced during normal operation of accelerator. Neutrons (especially thermal neutrons) can affect memory and registers in digital circuits and cause on illegal operation and thus loss of the system functionality [5, 6]. Single memory malfunction is known as a Single Event Upset (SEU) [7, 8, 9]. Microprocessor systems placed inside the accelerator tunnel are subjected to neutron and gamma radiation. To ensure a reliable operation it is necessary to protect digital circuits against radiation [10]. The more effective way to protect control systems against SEU is to use hardend computer systems designed for space and military application. Because of the accelerator complexity, number of control systems, sensors and computers, electronics devices design for radiation harden will be too expensive for X-FEL project. Other solution, much cheaper but more complex, is to use commercial of the shelf components (COTS) connected with system redundancy and safety algorithms. Protective algorithms can by used on hardware layer (for example embedded inside FPGA chips) or on software one. The authors decided to implement SEU-tolerant algorithms inside the sCore operating system kernel. The sCore is designed to work with standard computer architecture such as PC computers or embedded versions used by industry. There is known many frameworks and libraries supporting fault tolerant environments but all of them requires good knowledge about SEUs nature from application programmers. Moving fault tolerance from an application level to kernel level is better way to protect the application against bit-flip errors. Presented approach is to use a dedicated kernel with embedded protection algorithms which allow to develop and run applications that are not able to tolerate SEUs. Fault tolerant kernels are more comfortable for application programmers than application level libraries.

THE SCORE KERNEL

The kernel sCore was design during master thesis as a multitasking preemptive kernel based on the microkernel architecture [11]. Operating system and kernel architecture study is the goal of sCore developing. The sCore is design to work on IA-32 architecture. It can be run on standard PC platform and embedded industrial computers. The sCore is written in C++ and thus portability is assured thanks to C++ abstraction layer. The sCore provides constant time of task switching independently of number of tasks. Predictable time behaviour is important for real-time application. The sCore has more properties design for real-time systems. Other feature is scheduler with Round Robin scheduling policy and

256 priority levels queue. The most of internal structures are static and initialised during system bootstrap to ensure stability and predictable time dependences. The sCore kernel is divided into two version. First version uses flat linear memory model and second version of sCore deliver more advance memory policy management based on virtual memory model.

THE FIRST APPROACH TO PROTECT KERNEL AGAINST SEU

The first version of the sCore kernel [13] does not require Memory Management Unit (MMU) dedicated to advanced memory protection and address translation. Main memory is organized as a flat memory by sCore and all segments registers are configured to provide access to the all physical memory (see fig. 1).

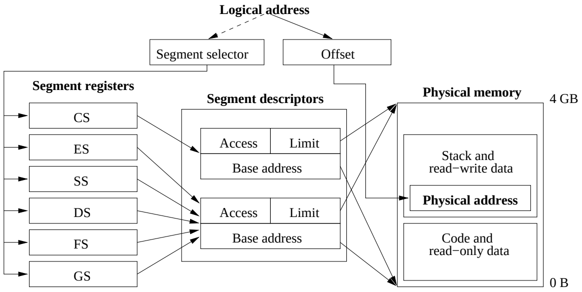


Fig. 1. The scheme of memory addressing in the first version of the sCore kernel

The flat memory model allows to access the linear memory region (on IA-32 from 0B to 4GB) by applications. The first version of the sCore uses only two segments descriptors and only one privilege level. Due to segmentation policy, all linear memory space is divided and shared between all tasks. Described behaviour of sCore kernel is dangerous for system stability and rare in modern kernels but it is necessary to run sCore on simpler architecture without MMU unit (e.g. some ARM processors).

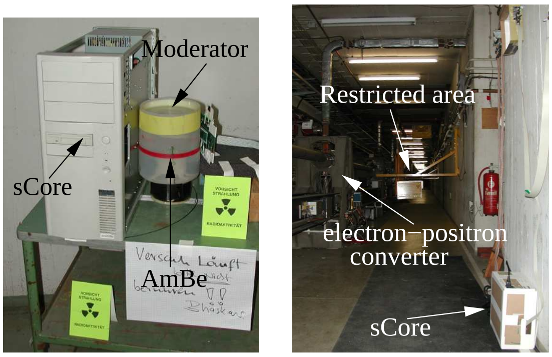


Fig. 2. Experiment with ^{241}AmBe and inside the Liniac2 tunnel

The MMU-less version of sCore kernel has a simple protection mechanism against SEU called EDAC Task [13]. EDAC Task is run periodically to scan and correct memory. The sCore system with EDAC Task was well examined during simulation with IARadSim [12] and experiments with ^{241}AmBe neutron source and inside accelerator tunnel (fig. 2). The results of experiments showed EDAC Task was able to correct bit-flip error in memory only when seldom SEU was observed. Therefore, more sophisticated protection algorithm are required.

MEMORY PAGING AND PROTECTION

The next version of sCore kernel employs MMU unit and provided by MMU paging technique to protect memory against SEU. Paging is a mechanism to divide computer memory into small parts, usually with constant size, and allocate memory using the page as a quantum part of memory [14]. Every page can be mapped into physical memory or mass storage devices. Single page is used by sCore to make copies of information and later to vote the correct one. It is necessary to know virtual memory mechanism provided by IA-32 technology to understand how algorithm works.

Virtual memory overview

The second version of the sCore kernel was based on virtual memory abstraction. Virtual memory mechanism is a method to provide individual virtual address space for every task. Memory Management Unit is required by the second version of sCore and thus sCore is not able tu run on simple MMU-less processors. The scheme of address translation on the IA-32 architecture is shown in fig. 3. The main memory is divided equally to small parts called pages. The 4 KB page size are used by sCore on the IA-32 platform. Every page is addressed by a page directory and a page table entries. The main advantage of a virtual memory is the independent address space for every task. The same linear address can be translated into a different physical address of every task. When the access to memory address is requested, MMU unit checks if exist the entry of an address inside a directory and a page table for a given linear address. If connection between a directory table or a page table and a linear address does not exist interrupt number 14 is generated. Interrupt number 14 is called by Intel page fault (PF). The page fault interrupt is handled by an interrupt service routine (ISR) and the connection between page directory and requested linear address is associated.

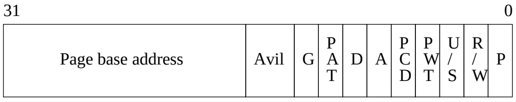


Fig. 4. A page table entry

Apart from a page address, an additional attributes are stored in the page table entry (fig. 4). From SEUs protection point of view, the most interesting flags are:

- Accessed (A) — Indicates whether a page has been accessed.

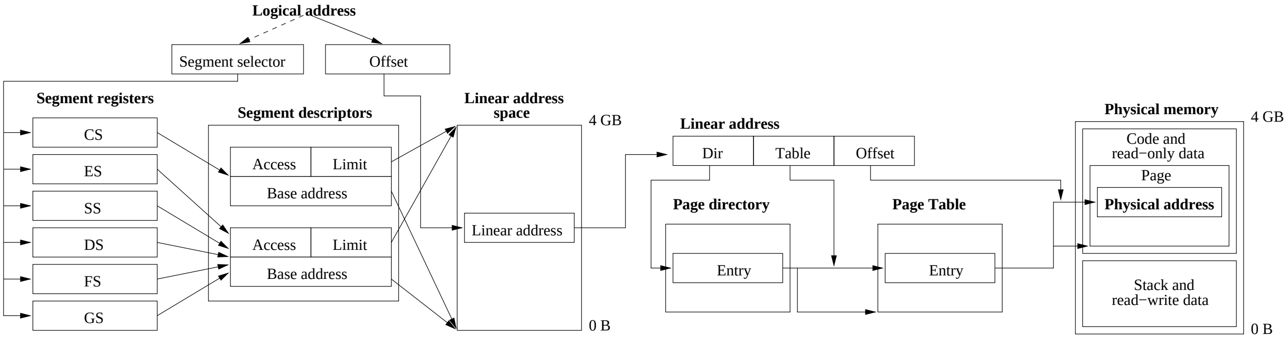


Fig. 3. The scheme of memory addressing in the second version of the sCore kernel

- Cache disable (PCD) — When the PCD flag is set caching of page is not allowed. The issue of cache will be discuss later.
- Read-write (R/W) — Indicates whether a page can be written or page can be only read.
- Present (P) — Most important flag for discussed SEUs tolerant algorithm. When the flag is set the page is in the main memory (in physical sense) and task is allowed to use it. When the flag is clear and some task tries to use it a page-fault exception (PF) is generated by processor. This flag is mostly used to realize swap technology by modern operating systems but sCore used it for SEUs protection. The present flag is always set to zero by sCore and interrupt is generated whenever page is accessed.

The address translation manner is the main difference between flat memory model and virtual memory model. The physical address is reflected by a logical address (one to one) in the first version of sCore. The second version of kernel allows to translate any logical address to a physical address. This approach provide a mechanism to realize a complete microkernel architecture by separating a address space for every modules and tasks.

The fault tolerant algorithm

The SEUs tolerant algorithm introduced by the second version of sCore is based on the memory paging mechanism. When the system is initialized (and also when new task is run or in general when a new block of memory is granted) all used pages are copied to two independent memory regions. All pages have cleared a present bit (fig. 4) in every page directory and table entries. When any task tries to use memory (code or data), page fault exception is generated by processor. The interrupt procedure (ISR) compares requested page and its copies using a triple voting technique. If contents of page is different from contents of its copies the correct value is restored. When the task is preemptive by system scheduler all pages with set RW bit, used during running time, are copied to copies memory region. The described mechanism guarantee that task has always valid contents of page (SEUs area corrected by ISR procedure) and the page is synchronized with its copies. The algorithm of protection against bit-flip errors inside memory are shown on fig. 5 This approach is more efficient and reliable, but also slow, compare to EDAC Task. In simple words, EDAC Task in second version of sCore is moved from system task to ISR procedure.

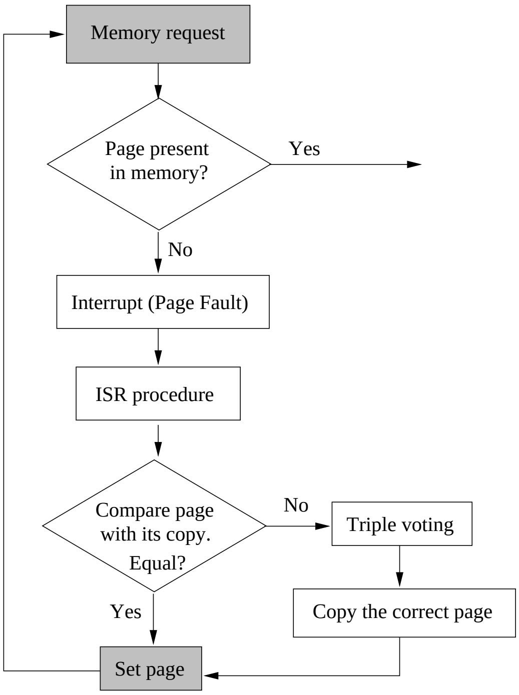


Fig. 5. A radiation protection algorithm

The influence of cache memory

The cache memory is an inseparable part of modern processors. Efficiency speed up is the goal of cache memory. A large size of cache is strongly required to decrease the number of missing hits in a standard application but not in the sCore kernel. Cache memory causes an incoherence between main memory and data used by processor when SEU is observed. Because of described cache behaviour it is not recommended to use cache memory with radiation tolerant system, like sCore. The second version of sCore disable cache memory when ever is possible.

THE FUTURE WORK

The future work should will concentrate on experiments with the second version of sCore kernel. The efficacy of new algorithm is expected to show by experiments compare to EDAC Task. It is very important to characterize the policy of cache memory management. Cache memory behaviour can be organized in many ways due the advantages of IA-32 architecture. The good cache policy should provide necessary a memory coherency and a good efficiency. The future work will be concentrated on the multiprocessor architecture, standard Symmetric Multiprocessing (SMP) as well as custom architecture design from COTS elements.

CONCLUSION

The second version of sCore provide many advantages in compare to the first version based on EDAC Task. EDAC Task can be used solely to protect read-only data. The sCore run EDAC Task in parallel to other tasks. Moreover, there is no time relation between memory checked by EDAC Task and memory used by other tasks. Very often EDAC Task compare different memory region than the one used by user task. The second version of sCore guarantee that memory region used by user task is always checked by ISR procedure, due to a small memory granularity called paging. The virtual memory technique allows to protect sCore also read-write data. Software fault-tolerant techniques will be more important in the future because of microelectronics progress. Transistors are smaller and therefore digital circuits are more susceptible to SEU and other errors, like electromagnetic interferences. Described SEU-tolerant technique can be used not only for physics experiment but also for medical devices and avionics.

ACKNOWLEDGEMENTS

We acknowledge the support of the European Community-Research Infrastructure Activity under the FP6 "Structuring the European Research Area" program (CARE, contract number RII3-CT-2003-506395), and Polish National Science Council Grant "138/E-370/SPB/6.PR UE-DIE 354/2004-2007".

THE AUTHORS

Bartłomiej Świercz, Dariusz Makowski and Andrzej Napieralski are affiliated to the Department of Microelectronics and Computer Science, Technical University of Łódź , Poland.
e-mail: swierczu@dmcs.pl

REFERENCES

[1] A. Schwarz, "The European X-Ray free electron laser project at DESY", 26th International Free-Electron Laser Conference, pp. 85-89, Agust 2004.

[2] W. Shi, "SASE X-Ray Free Electron Laser In DESY", Journal of the Society of Chinese Physysists, vol. 6, pp. 5-16, December 2000.

[3] R. Brinkmann, K. Flottmann, J. Rosbach, P. Schmuser, N. Walker, H. Weise, "TESLA Technical Design Report - The Accelerator, part II", DESY, 2001.

[4] G. Materlik, T. Tschentscher, "TESLA Technical Design Report. The X-Ray Free Electron Laser, PART V", DESY, 2001.

[5] D.Makowski, M. Grecki, B. Mukherjee, B. Świercz, S. Simrock, "Radiation Tolerant System for Neutrons Measurement", 12th Mixed Design of Integrated Circuits and Systems, MIXDES, July 2005.

[6] G. Messenger, M. Ash, "The Effects of Radiation on Electronic Systems", ISBN 0-442-25417-2. Van Nostrand Reinhold Company Inc., 1986.

[7] R. Peterson, "Radiation-induced errors in memory chips", Brazilian Journal of Physics, vol. 33, nr 2, pp. 246-249, June 2003.

[8] F. Giustino, "Radiation Effects on Semiconductor Devices", PhD thesis, Politecnico di Torino, March 2001.

[9] D.M. Fleetwood, H. A. Eisen, "Total-dose radiation hardness assurance", Nuclear Science, IEEE Transactions, vol. 50, pp. 552-564, June 2003.

[10] D. Makowski, B. Świercz, M. Grecki, A. Napieralski, "Projektowanie systemów niewrażliwych na wpływ promieniowania na potrzeby akceleratora X-FEL" (in Polish), Elektronika - Konstrukcje, Technologie, Zastosowania, nr 7/2005.

[11] B. Świercz, D. Makowski, A. Napieralski, "The sCore - Operating System for Research of Fault-Tolerant Computing", 12th Mixed Design of Integrated Circuits and Systems, MIXDES, July 2005.

[12] B. Świercz, D. Makowski, A. Napieralski, "IAradSim - IA32 architecture under high radiation environment simulator", 2005 NSTI Nanotechnology Conference and Trade Show, Nanotech 2005, Smart Sensors and Systems

[13] B. Świercz, D. Makowski, A. Napieralski, "Research of Fault-Tolerant Computing Using COTS Elements", 2006 NSTI Nanotechnology Conference and Trade Show, Nanotech 2006, Smart Sensors and Systems

[14] Intel Corporation, "IA-32 Intel Architecture Software Developer's Manual", Volume 3: System Programming Guide, 2004