Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-25 (2016) 413–418

Increasing Safety and Reliability of Roll-back and Roll-forward Lockstep Technique for Use in Real-time Systems

J. Arm [*] Z. Bradac [**] R. Stohl [***]

[*] *Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic (e-mail: jakub.arm@phd.feec.vutbr.cz)*
[**] *Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic (e-mail: bradac@feec.vutbr.cz)*
[***] *Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, Brno University of Technology, Brno, Czech Republic (e-mail: stohl@feec.vutbr.cz)*

Abstract: This work focuses on the roll-back and roll-forward lockstep architecture that are techniques to detect faults like SEU, voltage dips or another impacting to system misbehaviour. These techniques are explored using FMEA analysis and on its basis, some propose is presented that improves reliability and safety of each technique. In this case, availability is lowered as a trade of higher reliability and safety. These techniques are also explored from the point of view using in real-time systems. On this basis, some recommendations of appropriate checkpointing in lockstep roll-back technique is presented.

Keywords: Fault-tolerant, lockstep, fault detection, fault masking, roll-back, roll-forward, SEU, reliability, safety, embedded, real-time, RTOS

1. INTRODUCTION

Real-time operating systems are used in safety-critical, industrial, medical and automotive applications. Execution time of every operation has to be bounded in hard real-time systems and lowered in soft real-time systems. This OS is usually built in the embedded system.

The current challenge is to do more computational operations in lesser bounded time. Therefore more powerful devices are used as a control unit of the system. Systems are bigger and more complex, therefore more bugs can occur. Because of EM pollution, there is a higher risk of SEU (Single-Event Upsets) causing errors on memories and buses. Therefore more fault-tolerant techniques are used to eliminate and to detect errors caused by hardware and software defects. One of the main fault-tolerant technique is based on using proved hardware and software by time and by the count of applications. This technique is suppressed using new powerful processors. Therefore there has to be more sophisticated error detectors.

Fault avoidance is the preferred way how to eliminate errors. It consist of techniques like systematic architecture, hardware and software dimensioning, documentation, or system modelling and offline testing. Then it is crucial to focus on fault detection using online testing, monitoring, etc. When an error is detected, it should be masked using reconfiguration or redundancy.

The most used and the simplest technique to detect system hangs and deadlocks is a watchdog. It will raise an error when the internal time counter is not reset and exceeds a threshold. Its reaction threshold is higher when RR (Round Robin) scheduler is used. In [Pohronská and Krajčovič (2011)], one hardware watchdog per each thread is used which should eliminate false positive error detection causing by overload. On the other hand, there are many possible faulty scenarios which are not detected.

Application faults are mostly caused by SEU, voltage dips or hardware malfunction so the most used technique is so called lockstep. The basic idea is that two same CPU are assumed to perform same operations using a mutual clock source. Inputs are the same and outputs are compared. On a discrepancy, an error is signalled. There are some modifications like delaying of one core [Troppmann and Fuessl (2008)], defining one core as the leading and second as the paired [Horst et al. (2001)], or a comparison at the transaction level [James-Roxby and Wittig (2013)]. A reaction to this error is system reset or overall reconfiguration because the faulty core is not known. This technique has a vulnerability in common cause faults like software bugs, input malfunction or hardware input malfunction.

The problem of the not known faulty core is solved using TMR (Triple Modular Redundancy) that masks a fault in one of three cores using an output voter. The voter calculates median of all values three-times and the fault is masked. This technique needs three CPU cores so it is resource exhausting. In putting pressure on low price, there is a need to provide the same level of reliability and availability using less resources, e.g. enhanced mon-

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.12.083

itors or improved lock-step. In [Psarakis et al. (2014)] and [Gomez-Cornejo et al. (2013)], there are voters even on each segment of the processor pipeline.

Research about the lockstep and other fault-tolerant architectures have been risen due to FPGA. Using FPGA, more softcores can be implemented, another logic can be invented, and all can be tested using hardware logic fault injection or using special FPGA chip instructions. FPGA is used as a development platform for testing or as a final hardware solution where faults of the FPGA chip has to be considered.

Currently, there are many approaches of enhanced lockstep architecture which try to solve its drawbacks. For only error detecting, time shifted lockstep has good test results. But, when higher availability is needed, a complete fault-tolerant architecture has to be used so some automatic reaction has to be implemented like hot-swap, softcore reconfiguration (roll-forward) or lockstep roll-back (context recovery). Roll-back (see Fig. 1) is an operation which returns to the last saved fault-free context. This context called checkpoint is saved after the certain defined time or thread major period. Shortly before this operation, some tests can also be done. These techniques except hot-swap are good in comparison of reliability to price, on the other hand, provide lesser availability than *m out of n systems*.

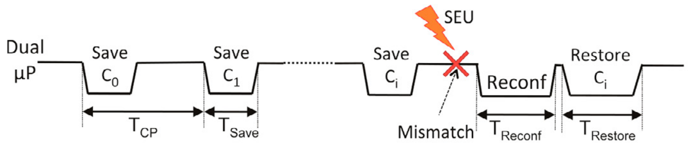

Fig. 1. Roll-back lockstep recovery process [Pham et al. (2012)]

2. RELATED WORK

In [Hong et al. (2012)], a hardware mechanism to enhance reliability using ECC (Error-Correcting Code) and TMR. ECC is usually capable of detecting double bit errors, or correcting single bit errors. TMR technique hardens highly the fault tolerance of used soft cores. ECC focuses on memories which are the most radiation susceptible resources. Using these techniques, processors are immunised to SEU without halting because of self-recoverable program memory. TMR technique provides good reliability and availability but it is much more resource demanding than other techniques.

In [Reorda et al. (2009)], an approach to an enhanced lockstep technique is presented. One of CPU is master which output is used. In hardware, the checker CPU output is compared with the master core output. In case of correspondence, the current context is saved as a checkpoint. In case of discrepancy, the context of the master core is restored. This technique is called roll-back checkpointing. When the roll-back operation is performed, program execution will go back to the last correct saved context. The context checkpointing can be triggered by bus activity or by time. The context saving and restoring are implemented using DMA (Direct Memory Access) mechanism. This solution provides the good reliability but the availability is lowered by roll-back operation. Performed experiments show that 15 % of the injected faults (random injection of SEU) are detected by OS, 1 % are transformed to permanent faults causing a faulty state and 84 % are effect-less or corrected.

In [Zabihi et al. (2015)] and [Abate et al. (2009)], a task replication-based method which can detect and correct SEUs is presented. The proposed multiprocessor system replicates a subset of tasks and their results are compared. The replication is performed on CPU idle. When an error is detected, all processors are reconfigured with fault-free bitstream. Then all processors return to the last free-error state that was verified before. According to tests, the performance overhead of the proposed method is about 70 % lower than of lockstep algorithm when the same level of reliability. Moreover, no extra area overhead is needed except the multicore processor. The redundant operations are done when a CPU is idle so there has to be a performance reserve.

In [Kottke and Steininger (2004)], a time shifted lockstep mechanism is improved. The master core is delayed by 1.5 clock source. Signals on CPU buses are set only in case of correct compared output. On used set of injected errors, this type of lockstep detects all errors.

In [Baleani et al. (2003)], some architectures using lockstep and loosely-synchronized dual processor are presented and discussed. Authors propose single-chip 4-core solution for fault tolerant automotive application based on two fail-silent channels which can be lock-stepped or loosely synchronized. All cores use all memories connected via crossbar bus architecture.

In [Safarulla and Manilal (2014)], an error is detected in lockstep architecture by providing shifted inputs to one core assuming that back-shifted outputs will be comparable.

In [Pham et al. (2012)], the roll-forward lockstep mechanism is presented. In case of error, the master core performs a test to detect the faulty core. Then the specified FPGA frame is reconfigured or the whole module is reconfigured to another place in case of persistent error. After that, recovery process is triggered using interrupt. The fault-free core saves its context, then the second core restores its context and after a synchronization signal, the both cores are started. The whole process is on Fig. 2.

A duration of the recovery process is claimed to be a value of 4 ms. According to the article, a probability of occurrence of one sensitive SEU during this time in another processor (while one is being reconfigured) is computed as a value of $2 \cdot 10^{-17}$. But another error can occur like voltage dip. Moreover, when some other SEU error occurs despite the small calculated probability, this error will be undetected and can cause error chain.

In [Zabihi et al. (2015)], the safety-critical task are replicated, executed more times and compared. This uses idle time of a CPU to create a partial time redundant system. This approach is very low cost and increases system efficiency. But in case of overload, the fault detection technique will be inaccessible.