Preprints of the 5th IEEE Conference on Energy Internet and Energy System Integration
October 22- October 25, 2021. Taiyuan, China

Fault-tolerant Design of Power Edge Computing Processor Based on Full-hardware Dual-core Lockstep

Peng Li
Digital Grid Research Institute
China Southern Power Grid
Guangzhou, China

Wei Xi
Digital Grid Research Institute
China Southern Power Grid
Guangzhou, China

Xiaowen Jiang
Institute of VLSI Design
Zhejiang University
Hangzhou, China
xiaowen_jiang@zju.edu.cn

Qun Chen
Institute of VLSI Design
Zhejiang University
Hangzhou, China

2021 IEEE 5th Conference on Energy Internet and Energy System Integration (EI2) | 978-1-6654-3425-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/EI252483.2021.9712894

***Abstract*—With the continuous reduction of integrated circuit process nodes and the increasingly complex working environment of power edge computing chips, the reliability of processors is facing severe challenges. The current fault-tolerant methods of commonly used processors either occupy a large area, or have high performance overhead and insufficient real-time performance. In response to this problem, this paper proposes a full-hardware dual-core lock-step power edge computing processor fault-tolerant design method and specific implementation, then builds a simulation-based fault injection platform, and finally conducts a lot of tests on its related indicators. The test results prove that the fault-tolerant design has excellent reliability and real-time performance, while reducing the area cost, and is particularly suitable for low-cost, real-time requirements.**

***Keywords*—power edge computing processor, fault-tolerant, hardware; dual-core, lockstep**

I. Introduction

Industrial microcontrollers are playing an important role in the development of industrial automation. Compared with general consumer applications, industrial microcontrollers have higher requirements for reliability, low cost and real-time performance. The power edge computing processor is a typical industrial microcontroller. The embedded processor is the core of the microcontroller. Due to the continuous reduction of feature size and voltage threshold [1], it is more sensitive to circuit crosstalk, atmospheric radiation, electromagnetic interference and other factors. Therefore, the fault-tolerant design of processors is the key to improving its reliability.

The basic idea of fault-tolerant design is based on redundancy, which can be divided into hardware redundancy [2], time redundancy [3] [4], software redundancy [5], and information redundancy [6]. Considering the complexity of processor functions and the requirement for performance and real-time performance, commercial processor's fault-tolerance designs are mainly based on hardware redundancy or a combination of hardware and software fault-tolerance. For example, the most common triple modular redundancy [7] [8] and multi-modular redundancy [9], which detects and recovers failures by comparing output results from multiple units, are highly reliable and real-time, but require too much area overhead. The Triple Core Lock-Step (TCLS) architecture is the natural evolution of Arm Cortex-R Dual Core Lock-Step (DCLS) processors to increase dependability, predictability, and availability in safety-critical and ultra-reliable applications [10]. However, the TCLS solutions rely on commercial semiconductor processes to reduce chip area cost and require the availability of extremely expensive radiation-resistant semiconductor processes.

To reduce area overhead, dual-mode redundancy, such as CON-MON architecture [11], is also commonly used. Dual-mode redundancy uses two cores as the main and monitoring cores respectively in the system. Kottke and Steininger proposed a general architecture in which dual cores can not only form a master-slave fault-tolerant pair for error detection of safety-critical tasks, but also work independently to perform performance-critical tasks, which improves the utilization of redundant cores [12]. Those two dual-mode redundancy can check the errors occurred by the main core, but cannot recover them. Hanafi et al. introduced dual-core error detection and recovery technology based on FPGA primitives [13]. The implementation method is simple and effective. However, considering that this method is limited to FPGA, it is expensive and the maximum supported operating frequency is low, making it difficult to use on a large scale.

In order to recover from failures under dual-mode redundancy, checkpoint-based fault-tolerant designs are often adopted. For instance, checkpoint-based dual-core lock-step fault-tolerant design [14] [15] [16] [17] detects failures by using two processors to compare operation results in real-time on hardware, saves and rollbacks processor nodes by interrupt service routine to recover failures in software. Since only the software visible processor state can be restored, its reliability is poor. And in the event of a hanging error [18], recovery may fail as the processor cannot execute the program normally. In addition, this method usually does not take into account the fault tolerance of the processor embedded cache, so although it saves area, there are shortcomings in reliability, performance, and real-time performance.

Against the above, this paper proposes full-hardware dual-core lockstep power edge computing processor fault-tolerant design, in the case that the microcontroller design company owns commercial processor soft IP core. For the failure model, as shown in TableI, this article mainly focuses on hang and SDC types of failures. By using dual modular redundancy processors and hardware fault tolerance module.

978-1-6654-3425-6/21/$31.00 ©2021 IEEE

Our scheme enables real-time fault detection and recovery, on-chip cache fault tolerance in write-through mode, and improves the reliability and real-time performance of fault tolerance at low cost.

TABLE I. PROCESSOR FAULT MODEL

Fault type	Description
benign	No effort to core normal operation,ignored
hang	This kind of error will cause the processor to be in an uncontrollable state, which generally requires a hardware reset to recove
SDC	Processor works normally but getting wrong results, which called 'Silent Data Corruption'

The remainder of the paper proceeds as follows. The methodology of checkpoint-based lockstep fault tolerance is given in Section II. Section III presents our full-hardware dual-core lockstep power edge computing processor system architecture including fault detected, fault recovery and fault isolation. Section IV shows the experimental results and analysis. Section V gives the conclusion.

II. CHECKPOINT-BASED LOCKSTEP FAULT TOLERANCE

A. Hardware Architecture

Fig.1 and Fig.2 show two common lock-step processor architectures. In Figure 1, the two processors have own separate memory, and both can access external modules. For components shared in the system, the two processors are accessed successively by handshake mechanism. In Fig.2, the two processors share the same input and memory, working in a master-slave relationship, and the main processor can access external modules, while the slave processor cannot. The checkers in the two figures are used to compare the output data of the two processor interfaces in real time, and the internal state of the processor is compared by software. The advantage of architecture in Fig.1 is that when real-time comparison's checking for errors is not required, the two processors can work independently. Since the storage units are copied, they can also detect the memory errors. The advantage of Fig.2 is that there is no need to resolve access conflicts of shared components, as well as smaller area and lower cost.

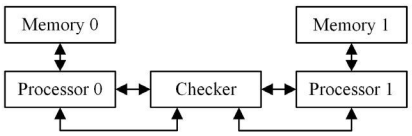

Fig. 1. Independent memory architecture.

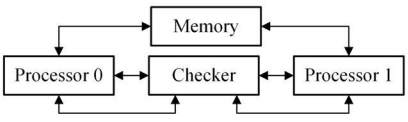

Fig. 2. Shared memory architecture.

B. Software Program

The fault tolerance processing flow [14] is shown in Fig.3. The two processors execute the same software program at the same time, and many checkpoints have been set in the program beforehand. When the execution flow reaches the checkpoint, it checks and compares if the internal states of the two processors are the same. If the comparison result is the same, it can enter the interrupt service routine for state saving, at this moment the correct node state in the processor will be saved to the reliable memory. The saved processor node state includes PSR (Processor State Register), PC (Program Counter), GPR (General-Purpose Register), etc. If it is checked that the internal states of the two processors are different, processors execute the rollback program and return to checkpoint A.

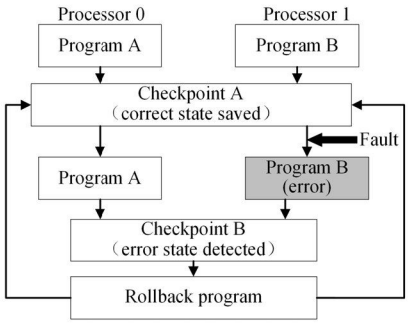

Fig. 3. Checkpoint based lockstep fault tolerance flowchart.

Checkpoint-based lockstep fault tolerance design balances cost and performance by combining hardware and software together to detect and recover failures. However, this method has some shortcomings. First of all, checkpoint-based lock-step method cannot recover from hanging failures because processors can no longer respond to subsequent recovery programs after hanging failures. Secondly, this type of fault tolerance design usually does not take into account the internal cache fault tolerance of the processor. In addition, due to the high real-time requirement of industrial microcontrollers, those two fault-tolerant methods are completed by interrupting service routine, which results in a slow response to failure recovery.

Therefore, this paper purposes a low-cost, full-hardware dual-core lockstep fault-tolerant design with shared storage which using hardware to achieve a rapidly fault recovery and solve the problem of unrecoverable hanging type faults. At the same time, cache fault tolerance in write-through mode is implemented by invaliding the error cache lines.

III. FULL-HARDWARE DUAL-CORE LOCKSTEP POWER EDGE COMPUTING PROCESSOR SYSTEM DESIGN

A. Overall Framework

Fig.4 is the architecture diagram of the fault tolerance system. The two processors operate in a master-slave mode, forming a self-monitoring pair. The fault tolerance module mainly consists of three parts: fault detection, fault recovery and fault isolation. The fault detection module is used to compare the running status of the master and slave processors in real time and detect the potential faults. The recovery module is used to save the correct node state of the main processor in real time, and in the event of a fault, return the master and slave processors to the previous correct node for re-execution. The fault isolation module is used to prevent the processors from error writing operations and external state rollback.

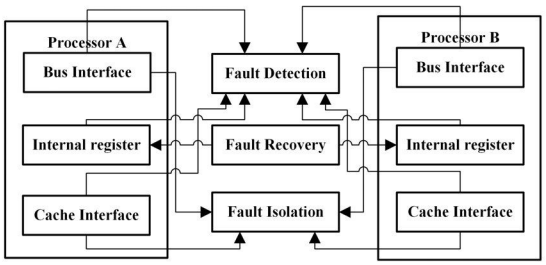

Fig. 4. Full-hardware dual-core lockstep power edge computing processor fault-tolerant design diagram.

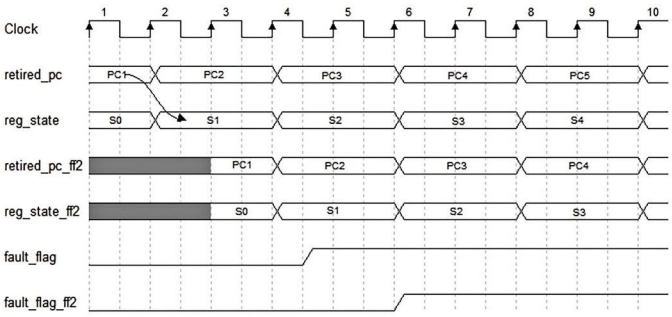

Fig. 5. Sequence diagram of state preservation of the correct node.

B. Fault Detection

Failure detection module works by using hard connections to pull related signals within the master-slave processor. Objects to compare include internal control state registers, bus interface, and cache interfaces in the master and slave processors. The internal control status registers include universal registers, control registers within the processor such as program counters, processor status registers, and related control state registers of tightly coupled IP such as timers and interrupt controllers. Since the faults are random, the error alarm signals caused by inconsistent comparison need to be synchronized from two levels as the fault isolation and recovery signals, in order to prevent uncertain state propagating from meta-stable state.

C. Fault Recovery

Fault recovery could be divided into two parts. First, the processor state information on the correct node is saved to the rollback buffer. This correct node refers to an execution point where there is no processors state inconsistency due to a soft error before, and the state information refers to the processor's internal register value. Secondly, hardware reset the master and slave processors, and put the correct node state information stored in the rollback buffer into processors. Both of the master and slave processors re-execute instructions from that node.

In order to recover the system as quickly as possible, the correct node selects the last instruction cycle when the state is inconsistent. As shown in Fig.5, after the PC1 instruction retires, the internal register state jumps to S1, and after the PC2 instruction retires, the internal register state jumps to S2. When PC3 is executed, it is detected that the master and slave processors are out of synchronization and a fault occurs. At this time, the correct node states saved at a rising edge of the clock before fault_flag is pulled high are PC2 and S1, and the processor should jump to this state when the processor fails.

Since the fault detection signal fault_flag_ff2 available to the fault-tolerant module is delayed by two beats when the abnormality is actually detected, the current processor state will be saved after each instruction is correctly retired. Therefore, when fault_flag_ff2 is pulled high for fault recovery, the content in the rollback buffer has been updated to PC3 and S2 on the sixth clock rising edge. When the fault is restored, the processor will jump to the PC3 and S2 states where the error has occurred, which does not match the expected result. In order to solve this problem, the current processor state information reg_state_ff2 and the instruction retirement signal can be delayed for two cycles, and the delayed state information reg_state_ff2 can be used as the source for storing the correct node information. At this point, when fault_flag_ff2 is pulled high after two cycles and fault recovery is required, the correct nodes stored in the rollback buffer are still PC2 and S1, which are consistent with the actual jump state.

TABLE II. PROCESSOR COMPARED SIGNALS

Signal type	Description
Core register	No effort to core normal operation, ignored
Tightly coupled IP	Such as timer and related control status register of interrupt controller
System bus interface	Signals used to access peripherals and system memory
Instruction bus interface	Used to access the signal of nuclear ICACHE
Data bus interface	Used to access the signal of the core DCACHE
Cache interface	External cache interface signal

After the fault detection module obtains the signals of Table II, it can perform fault detection through comparison operations. Once there is an inconsistency in the comparison results, the error detection signal fault_flag is raised. However, the occurrence of faults is random. The inconsistency of the signals may appear at any time. Therefore, the fault_flag signal should actually be processed asynchronously and cannot be directly used in subsequent circuit design, otherwise it may produce meta-stable state and cause system errors. Thus, fault_flag is used for two-level synchronization of the CPU clock domain as a signal for subsequent fault isolation and recovery.

D. Fault Isolation

Fault recovery can roll back the state of the master and slave processors, but it cannot reset the state outside processors, which including external memory, system IPs and ports, or the cache of processors. Because these three are different in functions and requirement modes, the fault isolation module will be designed in three parts as follows.

1) Memory Fault Isolation: Generally, the peripheral interface and the system IP are mounted on the system bus of the processor, in order to perform fault isolation, the write operation of the system bus needs to be modified. For memory, it is not important whether the data in the write operation is actually written or not, the key is that when the processor accesses the address again, it can get the previously written value. Considering this feature, fault

isolation of memory can be accomplished by creating a write operation buffer.

As shown in Fig.6, the write operation buffer consists of four buffers: the write address buffer, the write data buffer, the PC buffer, and the fault PC buffer. The write address buffer holds the corresponding write address for each write operation. The write data buffer holds the write data for each write operation. The PC buffer holds the retired PC corresponding to each write operation. The fault PC buffer saves the PC of the instructions that have been executed during the period after the fault occurs until the processor is reseted. Each buffer consists of three registers, since a maximum of two more write operations can be performed after processor fault, which means there is no more than three wrong write operations need to be isolated. Each write operation from the processor to the memory is temporarily stored in the buffer. When three write operations are stored, and the processor initiates a write operation again, it will issue a write operation with a non-zero write address stored in the write buffer for the first time. When the processor needs to read data from memory, it matches the read address to the one in the write buffer. If the address matches consistently and the address is non-zero, the data stored in the write operation buffer is returned to the processor. When a fault occurs and a state rollback is required, the write operation buffer invalidates the write operations corresponding to the same PC in the current fault PC buffer by setting the corresponding write operation address in the write operation buffer to zero. The data corresponding to invalid write operations is neither written to memory nor read by the processor. When a host other than the processor, such as DMA (Direct Memory Access), needs to access the memory, software is required to ensure that the processor performs three write operations to the useless address of the memory.

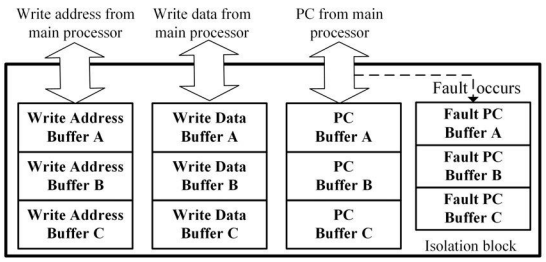

Fig. 6. Write operation buffer architecture.

2) Cache Fault Isolation: The feature of the cache in the write-through mode is that when the address to be accessed by the processor does not hit the cache, the processor obtains data from the external storage, at the same time, the data is written to the cache. Therefore, fault isolation of the cache can be accomplished by invalidating the error or lead state data in the cache. To make sure proper cache recovery and isolation while reducing recovery time, we invalidate the following eight cache rows during recovery: a) When there are no read data errors in the cache, the last four write operation addresses cached by the master and slave processors need to be invalid. b) When a read-out data error occurs in the cache, one address of the read-out data error, the last three write operation addresses of the master processor cache, and the last four write operation addresses from the slave processor cache are used as cache line addresses that need to be invalid. The specific cache fault isolation timing sequence is shown in Fig.7. When a fault occurs, the CEN on the SRAM port of the corresponding tag store is lowered during the period between fault and reset. And the corresponding cache rows are invalid by writing zero to the above eight addresses. Those achieve the cache failure isolation.

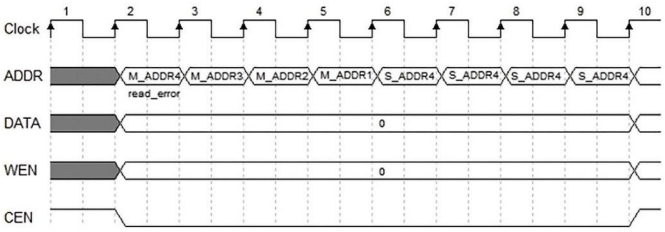

Fig. 7. Invalid operation timing diagram for cache during reset.

IV. Experiments and Analysis

A. Fault Injection Platform

Considering the cost and effectiveness of the test, this study uses simulated fault injection to test the fault tolerance performance of embedded processors, as shown in Fig.8. First, the Register Transfer Level (RTL) file of the processor is processed to extract the register signals in each hierarchical module. The output signals of the bus interface and the cache port are the fault injection points. Secondly, in the testbench, hardware macros are used instead of specific delay times and fault injection signals. When running scripts for continuous simulation, random delay times are generated and signals are randomly selected in the signal list under different random seeds. Random time and signals are passed to testbench in a macro-defined way to automate random fault injections. At the same time, in order to remove the influence of invalid fault injections that does not affect the result, testbench monitors the error alarm signal during the simulation process to determine whether each fault injection is valid. To ensure that errors are delivered internally which improves the effective error rate, each fault injection takes two system clock cycles.

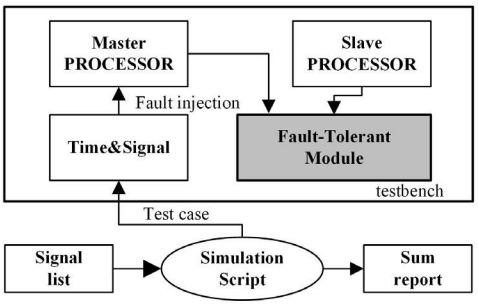

Fig. 8. Fault Injection Platform Architecture Diagram.

B. Reliability Test Results

The processor core used in this study is the CK803 series embedded processor of PingTouGe Semiconductor Co., Ltd [20]. In order to cover the fault situation as much as possible, 15000 random simulations are performed under the

conditions of 1, 2, 4 and 8 errors per injection respectively. The recovery rate and recovery time in each case are shown in TableIII, the equation of recovery rate is shown in equation (1).

$$Equation\ of\ Recovery\ Rate = \frac{PASSED}{PASSED + FAILED} \quad (1)$$

TABLE III. SINGLE AND MULTI-BIT FAULT RECOVERY.

Number of Fault Per Injection	Invalid Fault	FAILED	PASSED	Recovery Rate	Recovery Time
1	6354	481	8165	94.43	22 cycles
2	4507	913	9580	91.29	22 cycles
4	3531	1105	10364	91.36	2 cycles
8	899	2246	11855	84.07	22 cycles

From the experiment results, the more faults in a single injection, the lower the number of invalid failures. But the lower the recovery rate it is. When 8 faults are injected at the same time, the recovery rate is reduced to about 84%. This is mainly due to the fact that some unrecoverable failures occur more frequently and registers without reset pins are injected faults as the number of fault injections increases, which leads to processor enter exception and a lower overall recovery rate. Regardless of how many faults are injected in a single injection, the recovery time always stays around 20 cycles. For microcontrollers over 25 MHz, the fault response speed is within microseconds, which has a very high real-time performance.

C. Contrastive Analysis

In order to evaluate the proposed Hardware-based Dual-Core Lockstep (HDCL) fault tolerance design, we take the Triple Modular Redundancy (TMR) and Checkpoint-based Dual-Core Lockstep (CDCL) fault tolerance technology as examples to compare the area, fault tolerance rate, performance and recovery time. Based on the fault injection platform, the benchmark program Matrix is used, which calculates the product of two integer matrices of size 10x10. Meanwhile, in order to ensure consistent fault injection mode, only a single bit fault injection [8] to processor pipeline registers and internal control state registers is performed during the fault injection test. For every 300 write operations, set a checkpoint in CDCL [8]. Fig.9 shows a comparison of each of the three processor's fault tolerance schemes. To facilitate observation and analysis, the ordinate coordinates of each index are normalized. The maximum value in each index is used as the reference, and the other two values are mapped between [0,1] for comparison. The area column represents the area increase of the processor before and after fault tolerance. The fault tolerance rate column indicates the percentage of benchmarks run by fault-tolerant processors without explicit faults or faults recovered after a single-bit fault injection. The performance column shows the performance change of the executed program before and after the processor fails-tolerance when no fault occurs. The recovery time column indicates how long it takes a fault-tolerant processor to recover from a fault when an error is detected.

From Figure 9, we can see that the area overhead of HDCL is much lower than that of TMR because HDCL does not need to triple the redundancy of the processor. Compared

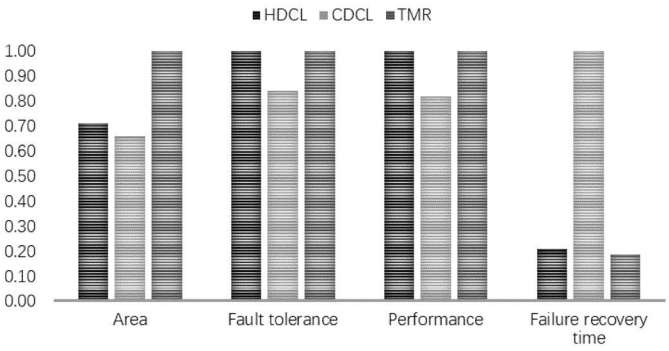

Fig. 9. Performance comparison of processor peripheral access under HDCL and TMR fault tolerance.

with CDCL, the area overhead of HDCL increased slightly, but not much, about 7% of that of CDCL due to the additional recovery logic. On the fault tolerance rate, it is equivalent to that of a highly reliable TMR. Compared with CDCL, HDCL has a higher fault tolerance rate, which is about 119%. In addition, since CDCL only performs simple port signal comparison on hardware the actual failure recovery capability of CDCL is lower. In terms of performance, HDCL is equivalent to TMR without loss of performance when the processor only performs memory and cache access. Compared with CDCL, HDCL's performance loss is much lower. About recovery time, HDCL takes slightly longer than TMR due to the state's rollback to the previous state instead of the current state after resetting the processor during recovery. Compared with CDCL, the recovery time of HDCL is much less, which is about 20%. Since the benchmark only involves access to memory and cache, and the proposed HDCL fault-tolerant scheme achieves access isolation for peripherals through delayed operations when isolating processors, so there is a performance loss when writing to peripherals. For this situation, we take further performance comparison of TMR fault-tolerant technology without performance loss. With SPIS, SPIM, I2C, UART and other peripherals as test objects, the performance of processor accessing peripherals under HDCL and TMR fault-tolerant schemes is compared by executing the same test case. The test cases are data transmission and reception in the main working modes of each peripheral, including DMA assisted data moving. Fig.9 is the peripheral access performance comparison of HDCL and TMR fault tolerant scheme.

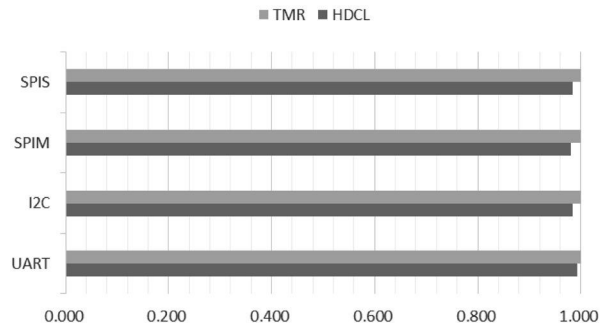

Fig. 10. Performance comparison of processor peripheral access under HDCL and TMR fault tolerance.

From the comparison in Fig.10, we can see that HDCL

has performance loss compared to TMR when accessing peripherals, and its performance is about 98% of TMR. But for microcontrollers, the configuration of peripherals usually changes less after setup. When large amounts of data are transmitted, DMA is often used to assist with data handling. Therefore, the fault isolation design for peripherals in HDCL does not affect the overall performance of the microcontroller.

V. Conclusions

The proposed full-hardware dual-core lockstep power edge computing processor fault-tolerant design has obvious advantages in fault tolerance rate, performance and recovery time with little area increasing overhead. HDCL has almost the same performances as TMR, but with much lower area cost. Therefore, the full-hardware dual-core lockstep processor fault-tolerant design has excellent reliability and real-time performance, while reducing the area cost, and is suitable for industrial microcontrollers with low cost and high real-time requirements. In this article, the fault tolerance of microcontrollers is mainly aimed at processor and memory fault tolerance, and errors in other modules cannot be detected and recovered. Therefore, further research is needed on the comprehensive fault tolerance of microcontrollers at low cost and high real-time.

Acknowledgment

This work is supported by National Key R&D Program of China (2020YFB0906000, 2020YFB0906001).

References

[1] Karlsson J , Liden P , Dahlgren P , et al. Using heavy-ion radiation to validate fault-handling mechanisms[J]. IEEE Micro, 1994, 14(1):0-23.

[2] Kanekawa N, Ibe E H, Suga T, et al. Dependability in electronic systems: mitigation of hardware failures, soft errors, and electromagnetic disturbances[M]. Springer Science & Business Media, 2010.

[3] Asghari S A , Marvasti M B , Rahmani A M . Enhancing transient fault tolerance in embedded systems through an OS task level redundancy approach[J]. Future Generation Computer Systems, 2018, 87(OCT.):58-65.

[4] Aponte-Moreno A , Pedraza C , Restrepo-Calle F . Reducing Overheads in Software-based Fault Tolerant Systems using Approximate Computing[C]// 2019 IEEE Latin American Test Symposium (LATS). IEEE, 2019,pp.1-6.

[12] Kottke T , Steininger A . A Reconfigurable Generic Dual-Core Architecture[C]// International Conference on Dependable Systems & Networks. IEEE, 2008.

[5] Li, D.; Hu, X. Redundant and fault-tolerant algorithms for real-time measurement and control systems for weapon equipment. ISA Transactions 2017, 67, 398 – 406.

[6] Ejlali, A.; Al-Hashimi, B.M.; Schmitz, M.T.; Rosinger, P.; Miremadi, S.G. Combined time and information, redundancy for SEU-tolerance in energy-efficient real-time systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 2006, 14, 323–335.

[7] Infineon. 32-bit AURIXTM Microcontroller based on TriCore? [EB/OL]. https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/?redirId=41544.

[8] Iturbe X , Venu B , Ozer E , et al. A Triple Core Lock-Step (TCLS) ARM® Cortex®-R5 Processor for Safety-Critical and Ultra-Reliable Applications[C]// IEEE/IFIP International Conference on Dependable Systems & Networks Workshop. IEEE, 2016.,pp. 246–249.

[9] Enas, A.; V., E.; N., A.; V, V. Fault-tolerant medical imaging system with quintuple modular redundancy (QMR) configurations. Journal of Ambient Intelligence and Humanized Computing 2018, pp. 1–13. 278

[10] Iturbe X , Venu B , Ozer E , et al. The Arm Triple Core Lock-Step (TCLS) Processor[J]. ACM Transactions on Computer Systems, 2019, 36(3):1-30.

[11] 8Gy″or″ok, G.; Beszédes, B. Duplicated control unit based embedded fault-masking systems. 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), 2017, pp. 000283–000288.

[13] Hanafi A , Karim M , Hammami A E . Dual-lockstep microblaze-based embedded system for error detection and recovery with reconfiguration technique[C]// Complex Systems. IEEE, 2016.

[14] Abate, F.; Sterpone, L.; Lisboa, C.A.; Carro, L.; Violante, M. New Techniques for Improving the Performance of the Lockstep Architecture for SEEs Mitigation in FPGA Embedded Processors. IEEE Transactions on Nuclear Science 2009, 56, 1992–2000.

[15] de Oliveira Á.B., T.L.; F.L., K. Exploring Performance Overhead Versus Soft Error Detection in Lockstep Dual-Core ARM Cortex-A9 Processor Embedded into Xilinx Zynq APSoC. International Symposium on Applied Reconfigurable Computing. Springer, Cham, 2017, pp. 189–201.

[16] Reorda, M.S.; Violante, M.; Meinhardt, C.; Reis, R. A low-cost SEE mitigation solution for soft-processors embedded in Systems on Pogrammable Chips. 2009 Design, Automation Test in Europe Conference Exhibition, 2009, pp. 352–357. 289

[17] Violante, M.; Meinhardt, C.; Reis, R.; Reorda, M.S. A Low-Cost Solution for Deploying Processor Cores in Harsh Environments. IEEE Transactions on Industrial Electronics 2011, 58, 2617–2626.

[18] de Oliveira, A.B.; Tambara, L.A.; Kastensmidt, F.L. Applying lockstep in dual-core ARM Cortex-A9 to mitigate radiation-induced soft errors. 2017 IEEE 8th Latin American Symposium on Circuits Systems (LASCAS), 2017, pp. 1–4.

[19] de Oliveira, A.B.; Rodrigues, G.S.; Kastensmidt, F.L. Analyzing lockstep dual-core ARM cortex-A9 soft error mitigation in FreeRTOS applications. 2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI), 2017, pp. 84–89. 297

[20] PingTouGe E803: low-power 32-bit processor. https://www.t-head.cn/product/e803?spm=a2ouz.12987052.0.0.28b848abi5uA1a, 2020.