Fault-tolerant method for anti-SEU of embedded system based on dual-core processor

Xiuhai Cui1, Qi Gao2, Ruichao Wang1, Li Liu2, Jun Liang1, Yu Peng1
1Harbin Institute of Technology, Harbin, People’s Republic of China
2China Aerospace Science and Industry Corporation, Beijing, People’s Republic of China
E-mail: cuixiuhai@hit.edu.cn

Abstract: The development of space applications based on commercial system on chip (SOC) FPGA devices has become an important direction for the development of aerospace technology, but single event upsets (SEUs) in space is a difficult problem for commercial SOC FPGAs for space applications. This article presents an anti-seu method for ARM processors in SOC FPGA. This method makes full use of the hardware resources of dual-core ARM in SoC FPGA and improves the system's anti-SEU capability through dual-core mutual-check and recovery mechanisms. At the same time, the data stream and control flow fault tolerant are used to improve the anti-SEU capability within the processor. Error detection and correction (EDAC) and triple modular redundancy (TMR) are used to improve anti-SEU capability of the data flow. A two-level watchdog and ARM exception handling are used to achieve the anti-SEU capability of the control flow. Experimental results show that the two-level fault-tolerance mechanism proposed here improves the system's anti-SEU capability without adding additional hardware resources. This method is currently carrying out satellite-borne ground application verification.

1 Introduction

The use of [1] commercial off the shelf (COTS) is a new direction for the development of small satellite [2] technology and applications. Cost-effective, high-performance, and mass-produced COTS devices meet the design needs of modern small satellites. Xilinx Zynq system on chip (SoC) is a product of development of heterogeneous multi-core SoCs. The dual-core ARM processor [3], programmable logic, and hard IP peripherals are embedded in the same chip. The perfect combination of flexibility and configurability of Zynq SoC has attracted wide attention in the field of small satellites, which has gradually been applied in engineering practice in the aerospace field [4, 5].

However, Zynq series chips have low radiation resistance and are susceptible to occur single event upsets (SEUs) [6, 7] in the space environment due to various high-energy particles and rays. SEUs can invalidate the function of the circuit and even lead to the catastrophic failure of the system. Consequently, SEUs has become a key issue to be solved in the space technology field. People have explored various methods to solve the device's SEUs problem.

Hardware triple modular redundancy (TMR) is a typical anti-SEU method. When it is used, three identical hardware modules are run at the same time, and the output results are voted in majority. If one module fails, the output result is different from that of the other two modules. Therefore, the system select the output of the two identical output results, thus shielding a module from error. TMR improves the anti-SEUs capability of the system, but it brings a large amount of hardware overhead and increases hardware costs. Error detection and correction (EDAC) [8–10] circuit is also an important anti-SEUs method. It can detect and correct SEUs error which mainly use anti-SEUs of the cache, memory, and BRAM.

Multi-version programming (MVP) is a software fault-tolerance method [11, 12] that requires N functionally equal software versions to run at the same time, which use a voter to determine the result and output it. Multi-version software is fault-tolerant and highly reliable. However, the high development cost is inconsistent with the low-cost and simple design requirements of small satellites.

With the development of the multi-core embedded processor, people use the natural redundant resources of the multi-core processor and parallel computing capabilities to provide a new solution to improve the anti-SEUs capabilities of space borne systems. In order to improve the anti-SEUs capability of space borne embedded systems, this paper proposes a new anti-SEUs tolerance method. This method improves the anti-SEUs capability of the system through hardware and software co-design. The hardware architecture is shown as Fig. 1, which includes dual ARM Cortex-A9 processors, 256KB on-chip memory (OCM), general interrupt controller (GIC) with five CPU private peripheral interrupts (PPI), 16 software-generated interrupts (SGI), and 60 shared peripheral interrupt (SPI), and other high-performance characteristics.

The work is organised as follows. The tolerant method is described in section 2, which includes system-level fault tolerance methods, control flow fault tolerance, and data flow fault tolerance. Section 3 is experiments. Section 4 is conclusion.

2 Anti-SEUs tolerant method

2.1 System-level fault tolerance method

In order to improve the system's ability of anti-SEUs, we uses a software-hardware co-design method. Dual-core mutual-check technology and roll-back recovery technology are used to implement SEUs error detection and recovery, and it is the first-
According to the Hsiao algorithm principle, the data written by the above fault tolerance mechanism in the single processor core and storage through Hsiao coding and TMR. Hsiao encoding is mainly the fault tolerance of the data flow in the satellite software. The includes the following processes: data encoding and decoding supervision code length is 5 bits. The Hsiao code implementation 2.2.1 Hsiao coding:

2.2 Data flow tolerant method

2.2.1 Hsiao coding: The paper improves the reliability of data storage through Hsiao coding and TMR. Hsiao encoding is mainly used for OCM which is for communication between dual cores, and TMR is mainly used for TMR of registers. Hsiao encoding is an extended Hamming code. The length of the Hamming code is \( n \), and the number of information bits is \( k \). Then, the number of bits of the supervision bit is \( r = n-k \), and the Hamming code is \( (n,k) \), which is expressed by the following (1).

\[
2^{r} - 1 \geq n \quad \text{or} \quad 2^{r} \geq k + r + 1 \tag{1}
\]

The Hsiao code here is \((13,8)\), the data bit width is 8 bits, and the supervision code length is 5 bits. The Hsiao code implementation includes the following processes: data encoding and decoding operations; error interrupt generation; dual-core synchronous write back.

2.2.2 Data encoding and decoding: Here, the Hsiao code is used as the error correction code for the OCM reinforcement. According to the Hsiao algorithm principle, the data written by the CPU to the OCM is encoded, and the data read by the CPU from the OCM is decoded.

When the CPU writes data to OCM, the program read the raw data from CPU0 or CPU1, split the 16-bit valid data of the original data into two sets of 8-bit data, and shift the 8-bit data of each set separately into data array of length 8. According to the data array and the generating matrix \( G \) obtained by the \((13,8)\) Hsiao algorithm, each 8-bit data is encoded, and get a check bit array which length is 5. Finally, according to the data bit allocation of the Hsiao code, the two data arrays and check arrays are shifted, and a 32-bit Hsiao code data are synthesized and stored in the OCM.

When the CPU reads data from the OCM, the program read the Hsiao code data from the OCM, record the data address, and split the Hsiao code data into two sets of \((13, 8)\) Hsiao code data. Performing the 13-bit coded data. The 13-bit coded data are stored in a 13-bit coded bit array. According to the coded bit array and the supervision matrix \( H \) obtained by the \((13,8)\) Hsiao algorithm, each coded data are decoded and a 5-bit check code is obtained. Finally, according to the syndrome \( S \), error-free, one-bit, and two-bit error status flags are obtained. For error-free and one-bit errors, the correct data can be obtained. Then, the CPU0 or CPU1 can read it.

2.2.3 Dual core synchronous write back: While data reading, if an error is detected, the correct data can be obtained through the Hsiao code decoding operation. However, in order to facilitate the use of subsequent data, correct data needs to be written back to the OCM. At the same time, another CPU cannot access OCM during the data write back to OCM. Therefore, a dual-core synchronous write-back operation is performed until the data write-back is completed. The write back process is shown as Fig. 3.

When CPU0 detects one-bit error, the CPU0 decoding operation sets an error status flag variable SEC_Flag to 1, CPU0 and CPU1 enter the interrupt error service program. At the same time, in CPU0, the correct data and the address of the write back variable are obtained. The correct data is written to the corresponding memory address. The variable SEC Flag is set to 0. CPU1 is always waiting for CPU0 to complete the write back operation until the value of variable SEC_Flag is 0. That is the loop waiting for SEC_Flag = 0 in CPU1 is terminated by CPU0.

When CPU1 detects one-bit error, the CPU1 decoding operation sets an error status flag variable SEC_Flag to 2. CPU0 and CPU1 enter the interrupt error service program. At the same time, in CPU1, the correct data and the address of the write back variable are obtained. The correct data are written to the corresponding memory address. The variable SEC Flag is set to 0. CPU0 is always waiting for CPU1 to complete the write back operation until the value of variable SEC_Flag is 0. That is the loop waiting for SEC_Flag = 0 in CPU0 is terminated by CPU1.

2.2.4 Register TMR: CPU registers can be divided into three types: read-only, write-only, and read-write. In order to use TMR for register monitoring, it is necessary to determine whether the register can be compared, refreshed, and reconfigured when refreshed.

Fig. 4 is a TMR fault-tolerant design. By monitoring the key registers, we can update the registers according to the characteristics of the registers after detecting the errors. Thus, the system can avoid abnormal system functions due to SEU changes the configuration bit information.

In order to improve the reliability of the fault-tolerant system, the contents of the key registers are divided into three backups and
stored them in three different physical addresses of the memory OCM.

During the execution of the program, the contents of the register are compared at regular intervals to detect whether there is an error in the configuration information. When the read-back comparison detects an error in the register, the register is repaired. If this register is an independent register, the true value is rewritten directly. If this register is related to other configuration registers, first set the configuration mode, then refresh the register value, and finally restore the CPU controller to normal mode.

For the SEU problem of off-chip memory DDR3, Zynq-7000 on-chip DDR controller can use the built-in ECC mode to implement the error correction function of DDR3 data which can ensure the reliability of instructions and data in CPU0 and CPU1 applications.

### 2.3 Control flow tolerant method

#### 2.3.1 Two-level watchdog mechanism:

The system aims at the problem that the SEUs causes the program to fall into an ‘out-of-cycle’. We uses a two-level watchdog mechanism to implement the monitoring of program control flow errors. The program is divided into \( N \) modules such as M1, M2, and M3 by function. When the program starts to run, first initialise PWDT0 (Private Watchdog Timer) and SWDT (System Watchdog Timer) on CPU0 and initialise PWDT0 and PWDT1 on CPU0 and CPU1, respectively. Among them, PWDT0 and PWDT1 are set to ‘timer’ mode; SWDT is set to reset output, and the SWDT value is slightly larger than \( N \). The PWDT loading value of the multiplier is shown in Fig. 5, and then the PWDT0, PWDT1, and SWDT are turned on, respectively.

Two private PWDT0 and PWDT1 are used as first-level watchdogs. Adopting the interrupt mode and taking the function module as a unit, and perform dual-core mutual-check synchronous wait detection in CPU0 and CPU1, respectively. During the running of the program, a ‘feed dog’ operation is performed after each function module is run. If the program can normally feed dog that the program continues to run forward. If the program causes a ‘dog scream’ due to a timeout during the running, an interrupt is generated by PWDT0 or PWDT1. The system perform a rollback operation, then re-run the function module until the fault is lifted, the program continues to move forward.

The system SWDT is used as the second watchdog and uses the reset method to perform software monitoring of the software in CPU0. During the running of the program, a ‘feed dog’ operation is performed once when each cycle is completed. If the program is running without any failure or the failure released by the private watchdog that the system watchdog can smoothly feed the dog. If the program fails during execution and the private watchdog fails to release, the system watchdog will generate a ‘dog scream,’ and the system will resets the system.

#### 2.3.2 Exception handling trap technology:

When the program fails to access the memory due to the influence of SEU, the program often runs off. This method aims at the problem that the SEU causes the memory data address or the instruction address to fail to access, and adopts trap technology based on ARM exception handling to implement fault-tolerant design. The fault tolerant method includes the following three steps: (i) The first is to initialise the exception handler, establish a source of prefetching instruction abort and data abort exception, and then build the connection between exception handler and exception source. (ii) Handling of exception handlers: exception handlers set the status flag of abort exception or data abort exception and record the instruction address that caused the abort. (iii) Exception return adjustment: prefetching instruction abort and data abort due to SEU cannot be resumed by repeated execution of instructions, which result in an ‘out of loop’. In order to solve this problem, the exception return address is modified again in the fault-tolerant design.

### 2.4 Dual-core mutual check and rollback recovery

Based on Zynq-7000's two-level fault-tolerant mechanism for on-board computer, dual-core ARM processors CPU0 and CPU1 operate at the same time and work in a coordinated manner. Both communicate through the OCM. They perform the same or different on-board tasks together and have the autonomous recovery capability of the on-board software after SEUs occur. The task assignment is shown in Table 1.

Rollback Recovery is a low-overhead fault-tolerant design technique. Rollback recovery is shown as Fig. 6. During the execution of the program, the entire program's valid information is saved to the checkpoint memory at regular intervals. When a fault...
when a fault is detected, the program volume returns to the previous checkpoint to recover the valid information of the checkpoint and re-execute the segment program, thereby avoiding the execution of the program from the beginning and reducing the computational losses.

3 Experiment

For testing software EDAC, fault of 1 bit and 2 bit were injected on 256 data from 0 to 255. The correct detection and repair results of the faults are shown in Table 2. As can be seen from the table, for fault-injection of 1 bit, the software EDAC can implement all fault detection and recover correctly. For fault injection with 2 bit, software EDAC can achieve all faults detection, but failure recovery. Therefore, the software EDAC’s error detection and correction capability is correct which is consistent with the expected Hsiao code error detection and correction capabilities.

For the TMR, we take a data every 0 × 100 from 0 × 00000000 to 0 × 00099000 to perform TMR redundant backup which produce a total of 1000 sets of data. For each of the three identical data, one data is injected one bit error, multiple bit error, two data are injected non-overlapping multiple bit errors, and two data are injected overlapping multiple bit errors separately. Fault injection and fault detection and recovery are shown in Table 3. It can be seen from the table that one-bit error and multiple-bit error of a data can output correct results by TMR voting. If there is different error between two data, but there is no overlap in the error positions of the two data, the correct result can be output through of TMR voting. If the error locations of the two data overlap, the results of the data recovery are wrong. The verification results are consistent with the TMR fault tolerance design.

The CPU0 private watchdog, CPU1 private watchdog, and system watchdog are performed 30 ‘dead loop’ operations, respectively, and the fault injection results are shown in Table 4. As can be seen from the table, the CPU0 and CPU1 private watchdogs enter the interrupt service routine after the watchdog ‘dog calling’ and the system watchdog resets after the watchdog ‘dog calling’. Execution results are consistent with expectations.

For data abort exceptions and prefetch instruction abort exceptions, we randomly select 30 addresses in the address mapping space of the programmable logic (PL), perform data write operations on them, and program counter PC points to the operation. The anomaly detection and error location results are shown in Table 5. The experiment results can be seen from the table, since the PL part does not write a logic module, the CPU does not have the authority to access its mapped address data, so the data access of this address all enters the data abort exception handle. Since the PC points to the mapped address of the PL part, which will cause the instruction execution to be invalid, so all the address enter the abort prefetching instruction handle.

4 Conclusion

We present a two-level anti-SEU fault tolerance algorithm for commercial SOC FPGA devices. During the algorithm design process, the internal dual-core ARM resources of the SOC FPGA are fully utilised to implement error correction and fault tolerance between the dual cores of the processor. At the same time, the data

---

**Table 2** Result of EDAC

| Content  | Injection fault | Fault detection | Rate of correction | Rate of recovery |
|----------|-----------------|-----------------|--------------------|-----------------|
| 1 bit    | 2048            | 2048            | 100\%              | 2048            |
| 2 bit    | 7168            | 7168            | 100\%              | 0               |

**Table 3** TMR test results

| Contents                | Data nums | Fault detection | Rate of correction | Rate of recovery |
|-------------------------|-----------|-----------------|--------------------|-----------------|
| one data one error      | 1000      | 1000            | 100\%              | 100\%           |
| one data Multi-errors   | 1000      | 1000            | 100\%              | 100\%           |
| two data no overlap error | 1000    | 1000            | 100\%              | 100\%           |
| two data overlap error  | 1000      | 1000            | 100\%              | 0               |

**Table 4** Watchdog test result

| Contents                | Injection faults | interruptions | Resets | Rate of correction |
|-------------------------|------------------|---------------|--------|--------------------|
| private watchdog CPU0   | 30               | 30            | —      | 100\%              |
| private watchdog CPU1   | 30               | 30            | —      | 100\%              |
| system watchdog         | 30               | —             | 30     | 100\%              |

**Table 5** ARM abort abnormal fault injection result

| Content                          | Injection faults | Enter exception handle | Correctly locate the exception | Rate of recovery |
|----------------------------------|------------------|------------------------|-------------------------------|-----------------|
| data abort exception             | 20               | 20                     | 20                            | 100\%           |
| prefetch instruction abort exception | 20            | 20                     | 20                            | 100\%           |
flow and the control flow are individually designed for fault tolerance of SEU error inside the processor. This method effectively improves the anti-SEU capability of the ARM processor in the SOC FPGA without increasing the hardware resource overhead. This method provides new solutions for SOC FPGAs for space applications. At present, the method is carrying out ground-based verification experiments for management system of satellites, which will plan to use satellites for space verification in 2019.

5 Acknowledgments

This work was partially supported by Space support fund under Grant No. JZ20170194 and Funding of Automatic Detection Technology and Instrument Key Laboratory of Guangxi under Grant No. YQ16202.

6 References

[1] Leite, F.G.H., Aguiar, V.A.P., Added, N., et al.: ‘Fast and low-cost soft error testing of a COTS microcontroller with alpha particle source’. 2017 IEEE Radiation Effects Data Workshop (REDW), New Orleans, LA, USA, 2017, pp. 18–21

[2] Liu, G.P., Zhang, S.: ‘A survey on formation control of small satellites’, Proc. IEEE, 2018, 106, (3), pp. 440–457

[3] Chielle, E., Rosa, F., Rodrigues, G. S., et al.: ‘Reliability on ARM processors against soft errors through SHFT techniques’, IEEE Trans. Nucl. Sci., 2016, 63, (4), pp. 2208–2216

[4] Ma, N., Wang, S., Ali, S.M., et al.: ‘High efficiency On-board hyperspectral image classification with zynq SoC’. 2016 7th Int. Conf. on Mechatronics and Manufacturing (ICMM 2016), New York, NY, USA, 2016, pp. 1–6

[5] Fayyaz, M., Vladimirova, T.: ‘Fault-tolerant distributed approach to satellite on-board computer design’. IEEE Aerospace Conf., Big Sky, MT, USA, 2014, pp. 1–12

[6] Benfica, J., Green, B., Porcher, B.C., et al.: ‘Analysis of FPGA SEU sensitivity to combined effects of conducted EMI and TID’. IEEE Asia-Pacific Int. Symp. on Electromagnetic Compatibility, Shenzhen, China, 2016, pp. 887–889

[7] Villata, I., Bidarte, U., Kretzschmar, U., et al.: ‘Fast and accurate SEU-tolerance characterization method for zynq SoCs’. 24th Int. Conf. on Field Programmable Logic and Applications (FPL), Munich, Germany, 2014, pp. 1–4

[8] Colodro-Conde, C., Toledo-Moreo, R.: ‘Design and analysis of efficient synthesis algorithms for EDAC functions in FPGAs’, IEEE Trans. Aerosp. Electron. Syst., 2015, 51, (4), pp. 3322–3347

[9] Abdelhai, L., Zohir, I., Samir, S., et al.: ‘Contribution to the implementation of image mosaicking algorithm on FPGA using NIOS II softcore’. 2015 3rd International Conference on Control, Engineering & Information Technology (CEIT), Tlemcen, Algeria, 2015, pp. 1–5

[10] Wilson, C., Sabogal, S., George, A., et al.: ‘Hybrid, adaptive, and reconfigurable fault tolerance’. 2017 IEEE Aerospace Conf., Big Sky, MT, USA, 2017, pp. 1–11

[11] Gasiot, G., Giot, D., Roche, P.: ‘Alpha-induced multiple cell upsets in standard and radiation hardened SRAMs manufactured in a 65 nm CMOS technology’, IEEE Trans. Nucl. Sci., 2006, 53, (6), pp. 3479–3486

[12] Shen, Z., Feng, C., Gao, S., et al.: ‘Study on FPGA SEU mitigation for the readout electronics of DAMPE BGO calorimeter in space’, IEEE Trans. Nucl. Sci., 2015, 62, (3), pp. 1010–1015