Research on SEU hardening of heterogeneous Dual-Core SoC

Kun Huang, Keliu Hu, Jun Deng and Tao Zhang

Sichuan Institute of Solid-State Circuits, China Electronics Technology Group Corp, Chongqing 400060, P. R. China
Email: hkof11@126.com

Abstract. The implementation of Single-Event Upsets (SEU) hardening has various schemes. However, some of them require a lot of human, material and financial resources. This paper proposes an easy scheme on SEU hardening for Heterogeneous Dual-core SoC (HD SoC) which contains three techniques. First, the automatic Triple Modular Redundancy (TMR) technique is adopted to harden the register heaps of the processor and the instruction-fetching module. Second, Hamming codes are used to harden the random access memory (RAM). Last, a software signature technique is applied to check the programs which are running on CPU. The scheme need not to consume additional resources, and has little influence on the performance of CPU. These technologies are very mature, easy to implement and needs low cost. According to the simulation result, the scheme can satisfy the basic demand of SEU-hardening.

1. Introduction

With the rapid development of space technology, the fault tolerance of integration circuits is increasingly important. In the harsh space environment, errors introduced by SEU occupy more than half of total radiation related errors [1]. More and more researches have focused on SEU effect. At process level, the SOI processing can make a chip single-event latch immune [2]. At circuit level, dual interlocked storage cell (DICE) achieves decent immunity against SEU effect [3]. At system level, Error Correcting Code (ECC) [4, 5] and Triple-Modular-Redundancy (TMR) [6, 7] are very helpful. At software level, control-flow checking by software signature and error detection by duplicated instructions can be a very suitable candidate [8-10].

These techniques are mature. However, there are still many shortcomings when the difficulty and the cost of implementation are considered. TMR is the most popular adopted solution, but the area cost is too high. The work [6, 7] had added TMR into the synthesized netlist, which was very inflexible. The area of DICE is smaller than TMR, but DICE requires a special digital cell library, which lead to a library to be developed afresh and result in consumption of a large number of manpower and resources. SOI is naturally immune to SEL, but the tape-out cost is expensive. The work [8] had presented a control-flow checking by software signature which required a special compiler, and would influence the performance of CPU. The work [10] adopted a redundant core to check the software signature, which still affect the performance of CPU when the current software signature is generated.

This paper proposes a novel SEU-hardening scheme for heterogeneous dual-core SoC (HD SoC). It consists of the following three techniques. First, the automatic TMR technique is adopted to harden the register heaps of the processor and the instruction-fetching module. Second, to harden the random access memory (RAM), Hamming codes are used. Last, a software signature technique is applied to check the programs running on CPU. This scheme can correct soft errors at circuit level and monitor soft errors at software level. The biggest benefit of this technology is that the implementation is very easy and the effect can satisfy the basic demand of SEU-hardening.

2. Heterogeneous Dual-Core SoC Hardening

It is common that a SoC in embedded application utilizes a DSP processor to enhance the capacity of arithmetic computing, thus a heterogeneous dual-core architecture has been established, and both performance and efficiency demand could be matched. An example of dual-core architecture is shown in Figure 1.

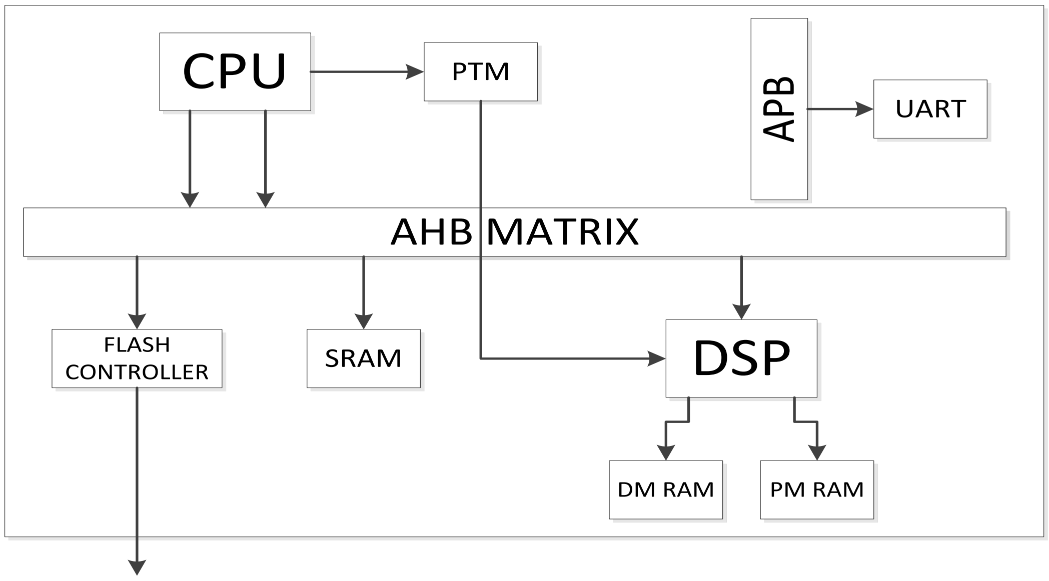

Figure 1. Arch of HD SoC

The two processers in HD SoC collaborate in a master-slave manner: CPU is the master processer which is in charge of control of the whole system, which is also a master on AHB bus matrix. DSP is a slave on AHB bus matrix and plays as a slave processer role to accomplish the data processing. The DSP has individual program ram and data ram which is also CPU-accessible. In addition, there is a program tracing module (PTM) which is used for software signature checking.

The hardening technique of HD SoC is comprised of three aspects: TMR to harden the CPU and DSP cores, ECC to protect the embedded memory, and PTM to monitor errors in a software style.

TMR is commonly used to protect sequential circuits, or storage elements from SEU, nevertheless, in the current situation, only the critical part of CPU and DSP should be TMR-hardened, in consideration of the area-consuming feature of TMR.

ECC is a widely-used method in radiation hardened memory design, and among the ECCs, Hamming code may be a simple and effective choice. In this paper, Hamming encoder and decoder will be added to fix the errors in the embedded SRAMs induced by SEU.

The function of PTM is tracing program, which records every PC value of branch instructions when CPU is running. DSP subsequently reads these messages to check the software signature, estimate whether if error occurs and run the error-fixing routine if necessary.

3. Automatic Design for TMR

TMR technique will make three copies of D flip-flops (DFF) whose outputs are connected to a voter. The structure is shown in Figure 2.

The function of the voter is:

$$Q = AB + BC + AC \tag{1}$$

If one value of (A, B, C) changes, the value of Q will not change. This makes DFF to be SEU immune.

This paper proposes a program, which can add TMR into the design of RTL automatically. It is shown in Figure 3.

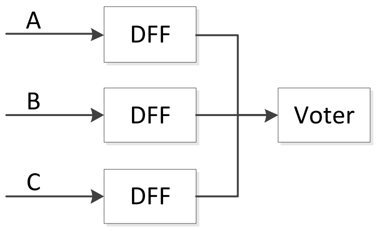

Figure 2. TMR structure

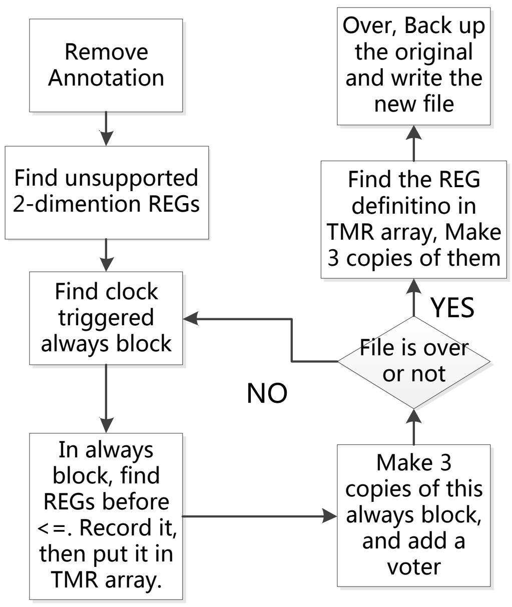
Remove Annotation
Find unsupported 2-dimention REGs
Find clock triggered always block
In always block, find REGs before <=. Record it, then put it in TMR array.
Make 3 copies of this always block, and add a voter
File is over or not
NO
YES
Find the REG definitino in TMR array, Make 3 copies of them
Over, Back up the original and write the new file

Figure 3. Automatic TMR Design

Before the automatic program to be applied, all the annotations should be removed. If 2-dimention REG arrays make 3 copies, the increase in area would not be acceptable. All 2-dimention REG arrays should be ignored. The key of this automatic algorithm is find and handle the 'always' block which is triggered by clock edge as this kind of block will be synthesized into DFF. Three copies of REG should be added when a find is hit, and the REG will be putted into a TMR array. The last step is to change the definition of REG in TMR array. All TMR REGs should be recorded, which will be used in synthesis.

4. Error-Correcting Code

Error-Correcting Code originates from Channel Coding Theory in communication, and is now widely used in radiation hardened memory design. There are many kinds of ECCs, such as Hamming code, BCH, RS (Reed-Solomon), LDPC, etc. In this paper, Hamming code is selected to apply the static RAM protection.

Hamming codes were proposed by Richard Hamming in 1950. With a k-bit message, Hamming codes generate an r-bit checksum through a matrix, and during the decoding, the checksum can be used to detect and correct 1-bit error in the message, in case it happened. The number k and r should match (2).

$$2^r \geq k + r + 1 \tag{2}$$

In this HD SoC, only the program memory of CPU and DSP is hardened, and the bit width is 32 and 24 respectively. For the 32-bit ram, choose (39, 32) Hamming codes. For the 24-bit ram, use (13, 8) and (22, 16) Hamming codes as a combination. The Hamming encoder and decoder will be added to the ram writing and reading circuit, a diagram for the circuit is shown in Figure 4.

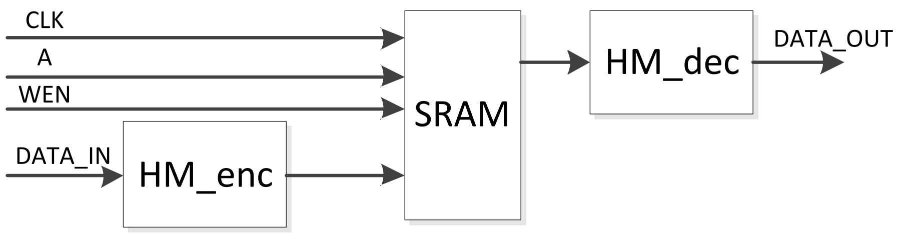

Figure 4. SRAM with Hamming Code Protection

5. Control-Flow Checking by Software Signatures

Control-flow checking by software signature is used to monitor the control-flow errors. The basic idea is as follows. The assemble source program is divided into basic block. Each block has no branch instruction except the last instruction. A unique signature is given to each block in a pre-defined way. All signatures are recorded. When the program is running into a new block, it generates a new signature and is compared with the pre-defined signature. If equal, it means the control-flow is correct. If not, it means error occurred. A special situation should be considered. When there are many blocks running into one block, Run-time Adjusting Signature should be included.

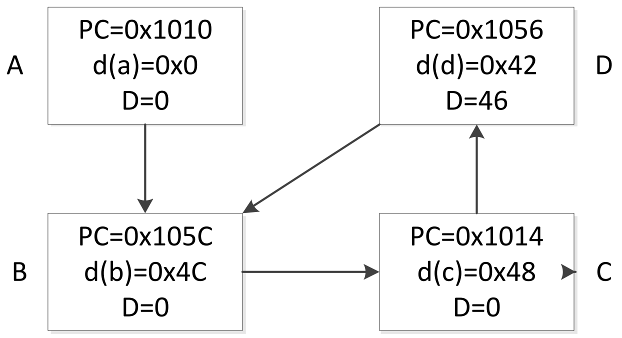

Figure 5. Example of software signature checking

In Figure 5, there are 4 program blocks, represented by node A to D. The PC of last branch instruction is the node signature. And D represents the Run-time Adjusting Signature. d(i) represents the distance (xor distance) from the source node to the destination node. For example,$d(b) = PC(B) xor PC(A) = 0\text{x}1010 \oplus 0\text{x}105\text{C} = 0\text{x}4$. d (i) is pre-defined and is restored in each node. When the program jumped from A to B, $PC(A) \oplus d(b)$ will produce the signature of node B. Then compare the signature with the PC value of current node (node B). If equals, it's right. If not, error occurs.

If the program jumped from D to B, then$PC(D) \oplus d(b) = 0x101A$. This value is not equal to the PC of node B (PC=0x105C). Node B has two source nodes, A and D. So Run-time Adjusting Signature $D = PC(A) \oplus PC(D)$ should be included. When program jumped from D to B, the signature will be$PC(D) \oplus d(b) \oplus D = 0x105C$. It is equal to the PC value of node B.

In [10], it uses a redundant CPU to check the signature of other CPU. This paper adopts a similar method. DSP is used to check the signature. The difference is how to produce the current signature, this paper use the program trace module instead of CPU which generate the signature by itself. The PTM is used to trace the signature of the current program. The PC value of the last branch instruction is defined as the signature of the block. The PTM needs to record all the PC of the branch instruction value when the program is running.

In order to record all the PC of branch instruction, it needs to find out all the branch instructions in the executing stage of pipeline and the current PC value. All these information should be organized as branch states. If a branch occurs, the PC value will be putted into a FIFO. Then the DSP reads out these PC value in the FIFO to check the software signature. The PTM is shown in Figure 6.

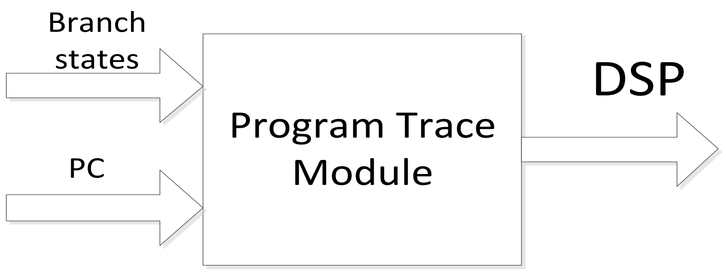

Figure 6. Program Trace Module

The algorithm of software signature checking is shown as following:

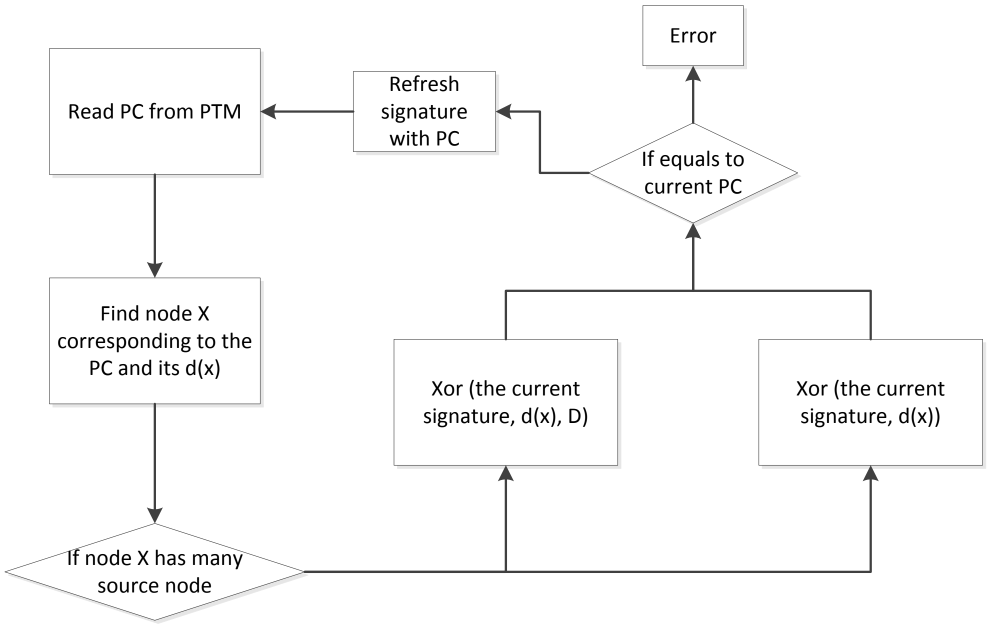
Error
Read PC from PTM
Refresh signature with PC
If equals to current PC
Find node X corresponding to the PC and its d(x)
Xor (the current signature, d(x), D)
Xor (the current signature, d(x))
If node X has many source node

Figure 7. Algorithm of software signature checking

Since the current signature PC is generated by PTM instead of CPU itself, the performance of CPU will not be affected. And the program running on CPU doesn't need to be changed.

6. Soft Error Inject Simulation

The performance of this SEU-Hardening technique is evaluated by injecting soft errors to the whole SoC during simulation. Since SEU affects DFF and SRAM, this paper only inject errors to the registers. If soft errors are injected manually, it is not efficient since the SoC is very complicated. The soft error injecting procedure is shown as follows:

• With the openkdb tools in Debussy, the full name of all REGs in SoC can be extracted by the REG triggered InstanceBased mode.

• A perl script is written to select some REGs randomly. Then a Verilog stimulus should be generated which makes the selected REGs to turn over (thus to emulate SEU errors) at predetermined time.

• CPU runs a 3x3 matrix multiplication. During this time, soft errors should be injected by the stimulus and then the result could be observed.

The running time of 3x3 matrix multiplication is 12.96us. Errors should be injected during the time. N REGs is selected randomly and errors are injected. This action is repeated 10 times.

Table 1 shows the result. When the error inject number is below 15, the program function is right which means the soft error is corrected by TMR or Hamming Code. When the error inject number is between 15 and 30, most errors could be detected which means the soft error can't be corrected but can be monitored. When the error inject number is above 30, the program run out frequently. The SEU-hardening technique is effective which meets the basic SEU-hardening requirements.

Table 1. Simulation Result

inject error num	function is right	error detected	program run out
10	10	0	0
15	8	2	0
20	7	2	1
25	7	3	0
30	6	3	1
35	4	4	2
40	4	1	5

7. Conclusion

This paper presents a novel SEU-hardening scheme for heterogeneous dual-core SoC. This scheme contains TMR tech, Hamming Code tech, and software signature checking tech. Advantages of the scheme are not only practical and uncomplicated implementation, but also little effect on the performance of CPU, as no changes are applied to the software program. The simulation result shows that this scheme can meet the basic SEU-hardening requirements.

8. Reference

[1] Changhe W. 1998 *The Influence with Reliability of Motional Satellite by the Single-Event Phenomena(Semiconductor Information vol 35)*pp 1-8
[2] Rathod S S, Saxena A K and Dasgupta S.2011*Alpha-particle-induced effects in partially depleted silicon on insulator device: with and without body contact[J](IET Circuits, Devices & Systems vol 5)*pp 52-58
[3] Jing S and Haiwei X. 2016*Design of Radiation Hardened SRAM Based on DICE(Electronics & Packaging vol 3)* pp 26-30
[4] Zhixiong S and Haixia X. 2014 *Implementation of Hamming code encoder and decoder based on VHDL(Microcomputer & Its Applications vol 12)*pp72-77
[5] JiangtingX and RuiZ. 2011 *Improved of Hamming Code and Circuits Realized in Memory (Electronics & Packaging vol 5)*pp20-22
[6] Ranran X and Hai-bo M.2014 *Triple modular redundancy design for VLSI gate level netlist(Computer Engineering & Science vol 12)*pp2356-2360
[7] Hui X and Wei T. 2015*Research of Radiation Harden Based on Triple Modular Redundancy for ASIC(MICROPROCESSORS vol 5)*pp1-4
[8] Oh NShirvani and P PMcCluskey.2002*Control-flow checking by software signatures(IEEE Transactions on Reliability vol 51)*pp111-122
[9] Oh NShirvani and P PMcCluskey.2002*Error detection by duplicated instructions in super-scalar processors(IEEE Transactions on Reliability vol 51)*pp63-75
[10] Liu T and Zhangqin H. 2014 *Fault detection approach for MPSoC by redundancy core(Journal of Computer Applications vol 1)*pp41-45