文章编号：2095-4980(2024)02-0219-08

软硬协同的嵌入式系统存储可靠性增强设计

杨　渊，邹祖伟

(中国工程物理研究院 电子工程研究所，四川 绵阳 621999)

摘　要：单粒子翻转(SEU)效应是造成空天环境中处理器故障的常见原因，需进行有效的防护设计，提升航空航天等高空设备的可靠性。传统的嵌入式可靠性防护设计一般采用单一的硬件或软件方法：采用软件三模冗余实现，需要占用大量的中央处理器(CPU)资源；采用硬件电路实现，无法输出错误信息。以PPC460处理器为目标系统，探讨了利用现场可编程门阵列(FPGA)对PPC460处理器的可靠性增强设计方法，同时使用扩展型的汉明码编解码算法、奇偶校验、三模冗余技术，利用软硬协同的方式提高了存储空间内数据的正确性，减少了CPU资源消耗，有效实现了PPC460处理器在特殊复杂环境中对重要数据的高安全、高可靠、抗干扰保护。

关键词：现场可编程门阵列(FPGA)；单粒子翻转；汉明码；三模冗余

中图分类号：TP332　　**文献标志码：**A　　**doi：**10.11805/TKYDA2022022

Soft-Hardware Coordinated Design for Enhanced Storage Reliability in Embedded Systems

YANG Yuan，ZOU Zuwei

(Institute of Electronic Engineering，China Academy of Engineering Physics，Mianyang Sichuan 621999，China)

Abstract：Single Event Upset(SEU) effects are a common cause of processor failures in aerospace environments, necessitating effective protective designs to enhance the reliability of high-altitude equipment in the fields of aviation and astronautics. Traditional embedded reliability protection designs typically employ either single hardware or software approaches: implementing Triple Modular Redundancy(TMR) through software requires substantial CPU resources; employing hardware circuits does not facilitate error reporting. This paper focuses on the PPC460 processor as the target system, discussing an advanced reliability enhancement design method utilizing Field-Programmable Gate Array(FPGA) technology for the PPC460 processor. The approach integrates an extended Hamming code encoding and decoding algorithm, parity checking, and Triple Modular Redundancy techniques. By synergistically combining software and hardware strategies, it improves the correctness of data within the storage space, reduces CPU resource consumption, and effectively realizes high-security, high-reliability, and interference-resistant protection for critical data on the PPC460 processor in special complex environments.

Keywords：Field Programmable Gate Array(FPGA)；Single Event Upset(SEU)；Hamming code；Triple Modular Redundancy

基于PowerPC(Performance optimization with enhanced RISC)架构的嵌入式处理器能够实现对空间飞行器的控制管理和数据处理，但高空环境中存在大量高能带电粒子[1]，在储存和使用过程中，空间飞行器易受高能带电粒子的影响，使处理器程序出现逻辑错误、数据失效与信号不稳定等故障[2-3]，降低空间飞行器的运行性能，导致程序执行序列紊乱、系统计算失误甚至整个系统功能失效。由于高空环境的复杂性和多样性，空间飞行器发生错误故障时难以及时修理，因此，保证空间飞行器的高安全性与高可靠性至关重要，需要采取相应的容错措施，进行有效的抗SEU加固设计提升高空设备的可靠性。

PPC460处理器包含内部静态随机存取存储器(Static Random Access Memory，SRAM)、只读存储器(Read Only

收稿日期：2022-01-17；**修回日期：**2022-03-11

基金项目：国防科工局基础科研资助项目(JCKY2019212B004)

Memory，ROM)、嵌入式闪存模块(eFlash)、直接存储器访问控制器(Direct Memory Access，DMA)、PLB(Processor Local Bus)总线到OPB总线(On-chip Peripheral Bus)的转换等。其中内部SRAM模块作为整个处理器的重要模块，所有重要数据均在SRAM空间内，因此，针对内部SRAM空间进行存储加固设计至关重要[4]。PPC460处理器传统的可靠性设计方法为：对SRAM数据采用奇偶校验保证数据完整性，写入的数据自动计算奇偶校验位，并存入SRAM中，数据从SRAM读出时，奇偶校验位也会同时被读出，并同时判断奇偶校验位是否正确，如果错误会及时报警。但仅依靠奇偶校验对SRAM存储空间进行可靠性防护不能抗单粒子翻转(SEU)现象[5]，可靠性不强。

因此，本文采用软硬件协同方式，利用FPGA对PPC460处理器进行存储器可靠性增强设计，在FPGA中对PPC460核心代码进行移植，并将奇偶校验、三模冗余、余数纠错技术结合，保障该空间内数据的正确性和可靠性，有效提高PPC460处理器在特殊的、复杂的环境中的安全性。

1 基于嵌入式系统存储可靠性的软硬件协同架构设计

1.1 基于FPGA的PPC460架构移植设计

在FPGA中对PPC460核心代码进行移植，图1为移植过程的架构示意图。

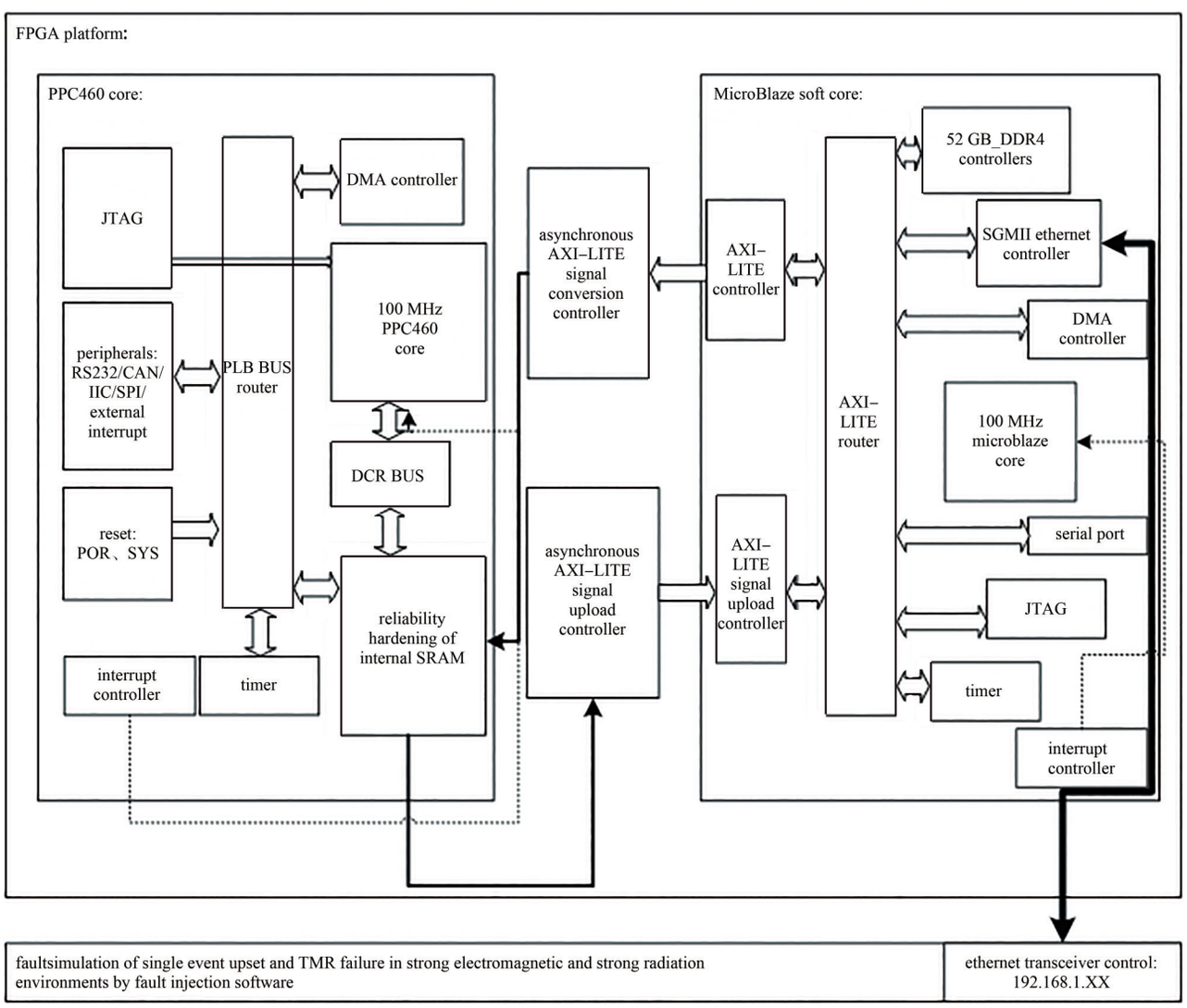

Fig.1 Schematic diagram of PPC460 architecture based on FPGA
图1 基于FPGA的PPC460架构示意图

整个系统由FPGA平台完成，由PPC460内核、内部SRAM可靠性加固代码以及MicroBlaze软核构成。其中：1) 内部SRAM可靠性加固代码包括三模冗余、错误检测与纠正(Error Detection And Correction，EDAC)、内存

监控 3 个模块，共占用 FPGA 资源为：查找表(Look–Up–Table，LUT)使用 89 023 个，触发器(Flip–Flop，FF)使用 55 345 个；

2) PPC460 内核包括原 PPC460 各功能及硬核化功能代码；

3) MicroBlaze 软核包括 SGMII 以太网控制器及其他相关外设；PPC460 和 MicroBlaze 软核间通过异步 AXI–LITE 控制器进行通信。

1.2 PPC460 内部 SRAM 空间可靠性加固设计

PPC460 内部高安全 SRAM 空间的可靠性加固设计功能示意图如图 2 所示。针对 PPC460 内部 SRAM，新增加一块 64 KB 内存空间作为高安全存储空间使用，该空间采用基于 128 位宽 PLB 总线直接挂在 PPC460 内核中供操作系统直接调用，同时采用奇偶校验技术、三模冗余技术和余数纠错技术，增强纠错能力。

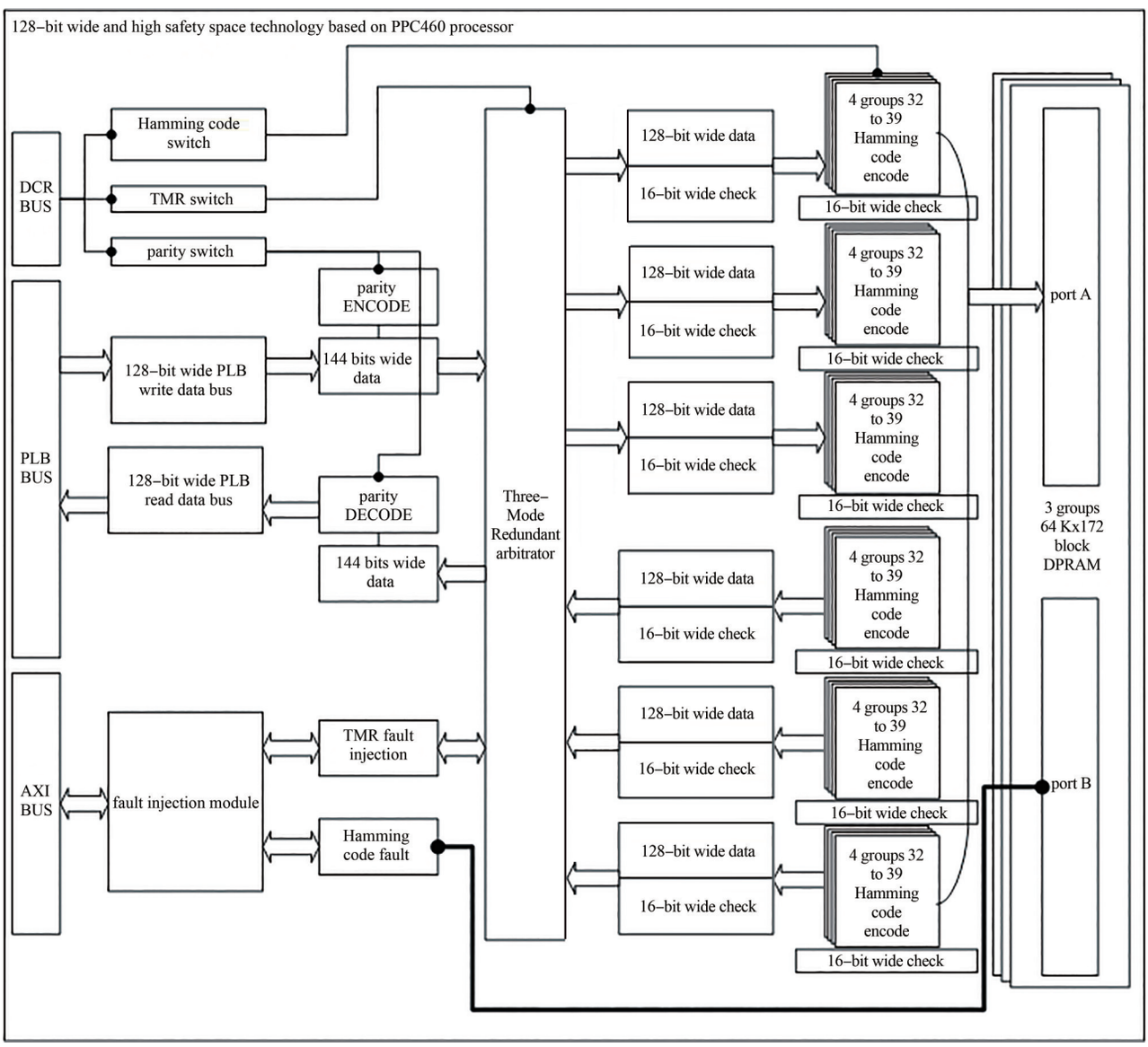

Fig.2 Functional diagram of high security storage space

图 2　高安全存储空间功能示意图

整个模块包括以下部分：

1) DCR 总线功能配置：配置奇偶校验开关、三模冗余开关、汉明码开关；

2) PLB 总线功能模块：对 128 位宽总线数据进行处理；

3) AXI 总线功能模块：对注入的故障信号进行编解码处理；

4) 原始数据奇偶校验模块：对原始 128 位宽数据进行 8~9 奇偶校验处理；

5) 三模冗余模块：对经过奇偶校验后的数据进行三模仲裁处理；

6) 汉明码数据拆分重组模块：对 128 位宽原始数据进行 4 组 32 转 39 的汉明码编解码拆分重组；

7) 32 位汉明码编解码模块：32 转汉明码编解码，实现纠一检二功能；

8) 双端口 DPRAM 控制模块：3 组 64 K×176 的 DPRAM 控制模块。

在该高安全存储空间内，通过模拟单粒子翻转现象和三模冗余失效现象，在合适的位置注入错误，并提供错误统计功能，报告无法纠正的错误异常。

2 PPC460 处理器纠检错及三模冗余加固设计

对 PPC460 处理器的增强加固设计思路为：通过 EDAC 加固控制模块进行编码，使用三模冗余技术实现纠检错[6−7]，然后将纠正后的正确数据进行解码，通过 EDAC 技术加固控制模块存入存储体中，再通过扩展型汉明码编解码，实现纠一检二功能。

2.1 扩展型汉明码纠一检二编解码算法设计

汉明码信息传输可靠性强[8]，能够完成对一位错误的纠正。但传统汉明码无法同时实现一位错误的纠正与两位错误的检测，需要在此基础上进行扩展实现纠一检二功能。本文针对 PPC460 处理器的 128 位宽原始数据进行 4 组 32 转 39 的汉明码编解码拆分重组，对 PPC460 处理器的 32 位数据位按规则编码，产生 7 位校验码，组成 39 位数据位，用于校验编码后全部码元。

2.1.1 扩展型汉明码编码设计

使用扩展型汉明码进行编码，需要在发送端给信息码元增加一定数量的多余码元(即监督码元或校验码元)[9]，形成一个新码组。(39,32)系统汉明码为 GF(26)域上(63,57)的截断码，根据伴随矩阵经计算得到生成矩阵 $\boldsymbol{G}$。设信息矢量为 $\boldsymbol{m}=[m_{31},\cdots,m_1,m_0]$，根据计算所得到的生成矩阵 $\boldsymbol{G}$ 和线性分组码的编码理论 $\boldsymbol{c}=\boldsymbol{m}\cdot\boldsymbol{G}$ 可得到：

$$c_i=m_j\ (i=38,\cdots,9,8,7;j=31,30,\cdots,2,1,0)$$

$$c_6=m_{31}\oplus m_{30}\oplus m_{29}\oplus m_{28}\oplus m_{27}\oplus m_{26}$$

$$c_5=m_{25}\oplus m_{24}\oplus m_{23}\oplus m_{22}\oplus m_{21}\oplus m_{20}\oplus m_{19}\oplus m_{18}\oplus m_{17}\oplus m_{16}\oplus m_{15}\oplus m_{14}\oplus m_{13}\oplus m_{12}\oplus m_{11}$$

$$c_4=m_{25}\oplus m_{24}\oplus m_{23}\oplus m_{22}\oplus m_{21}\oplus m_{20}\oplus m_{19}\oplus m_{18}\oplus m_{10}\oplus m_9\oplus m_8\oplus m_7\oplus m_6\oplus m_5\oplus m_4$$

$$c_3=m_{31}\oplus m_{30}\oplus m_{29}\oplus m_{25}\oplus m_{24}\oplus m_{23}\oplus m_{22}\oplus m_{17}\oplus m_{16}\oplus m_{15}\oplus m_{14}\oplus m_{10}\oplus m_9\oplus m_8\oplus m_7\oplus m_3\oplus m_2\oplus m_0$$

$$c_2=m_{31}\oplus m_{28}\oplus m_{27}\oplus m_{25}\oplus m_{24}\oplus m_{21}\oplus m_{20}\oplus m_{17}\oplus m_{16}\oplus m_{13}\oplus m_{12}\oplus m_{10}\oplus m_9\oplus m_6\oplus m_5\oplus m_3\oplus m_2\oplus m_0$$

$$c_1=m_{30}\oplus m_{28}\oplus m_{26}\oplus m_{25}\oplus m_{23}\oplus m_{21}\oplus m_{19}\oplus m_{17}\oplus m_{15}\oplus m_{13}\oplus m_{11}\oplus m_{10}\oplus m_8\oplus m_6\oplus m_4\oplus m_3\oplus m_1\oplus m_0$$

$$c_0=m_{31}\oplus m_{30}\oplus m_{29}\oplus m_{28}\oplus m_{27}\oplus m_{26}\oplus m_{25}\oplus m_{24}\oplus m_{23}\oplus m_{22}\oplus m_{21}\oplus m_{20}\oplus m_{19}\oplus m_{18}\oplus m_{17}\oplus m_{16}\oplus m_{15}\oplus m_{14}\oplus m_{13}\oplus m_{12}\oplus m_{11}\oplus m_{10}\oplus m_9\oplus m_8\oplus m_7\oplus m_6\oplus m_5\oplus m_4\oplus m_3\oplus m_2\oplus m_1\oplus m_0\oplus c_6\oplus c_5\oplus c_4\oplus c_3\oplus c_2\oplus c_1$$

为实现方便，把输出定义为 39 位，编码后的码字为：$c=(c_{38},c_{37},\cdots,c_1,c_0)$，其中，$c_{38},c_{37},\cdots,c_8,c_7$ 为信息位；$c_6,c_5,\cdots,c_1$ 为基础校验位，可实现纠一位错误的功能；c_0 为扩展校验位，加入后可以实现纠一检二功能。

2.1.2 扩展型汉明码解码设计

扩展型汉明码的解码过程，即在接收端通过检验码字的约束关系符合与否，从而判定错误位置并纠正错误。其译码过程包括伴随式的计算和错误图样的求解。

伴随式 $\boldsymbol{s}=(\boldsymbol{c}+\boldsymbol{e})\boldsymbol{H}^{\mathrm{T}}=\boldsymbol{e}\boldsymbol{H}^{\mathrm{T}}$，因此通过计算可得到伴随式 $s_6s_5s_4s_3s_2s_1s_0$ 的计算公式如下：

$$s_6=c_{38}\oplus c_{37}\oplus c_{36}\oplus c_{35}\oplus c_{34}\oplus c_{33}\oplus c_6$$

$$s_5=c_{32}\oplus c_{31}\oplus c_{30}\oplus c_{29}\oplus c_{28}\oplus c_{27}\oplus c_{26}\oplus c_{25}\oplus c_{24}\oplus c_{23}\oplus c_{22}\oplus c_{21}\oplus c_{20}\oplus c_{19}\oplus c_{18}\oplus c_5$$

$$s_4=c_{32}\oplus c_{31}\oplus c_{30}\oplus c_{29}\oplus c_{28}\oplus c_{27}\oplus c_{26}\oplus c_{25}\oplus c_{17}\oplus c_{16}\oplus c_{15}\oplus c_{14}\oplus c_{13}\oplus c_{12}\oplus c_{11}\oplus c_4$$

$$s_3=c_{38}\oplus c_{37}\oplus c_{36}\oplus c_{32}\oplus c_{31}\oplus c_{30}\oplus c_{29}\oplus c_{24}\oplus c_{23}\oplus c_{22}\oplus c_{21}\oplus c_{17}\oplus c_{16}\oplus c_{15}\oplus c_{14}\oplus c_{10}\oplus c_9\oplus c_7\oplus c_3$$

$$s_2=c_{38}\oplus c_{35}\oplus c_{34}\oplus c_{32}\oplus c_{31}\oplus c_{28}\oplus c_{27}\oplus c_{24}\oplus c_{23}\oplus c_{20}\oplus c_{19}\oplus c_{17}\oplus c_{16}\oplus c_{13}\oplus c_{12}\oplus c_{10}\oplus c_9\oplus c_7\oplus c_4\oplus c_2$$

$$s_1=c_{37}\oplus c_{35}\oplus c_{33}\oplus c_{32}\oplus c_{30}\oplus c_{28}\oplus c_{26}\oplus c_{23}\oplus c_{22}\oplus c_{20}\oplus c_{18}\oplus c_{17}\oplus c_{15}\oplus c_{13}\oplus c_{11}\oplus c_{10}\oplus c_8\oplus c_7\oplus c_1$$

$$s_0=c_{38}\oplus c_{37}\oplus c_{36}\oplus c_{35}\oplus c_{34}\oplus c_{33}\oplus c_{32}\oplus c_{31}\oplus c_{30}\oplus c_{29}\oplus c_{28}\oplus c_{27}\oplus c_{26}\oplus c_{25}\oplus c_{24}\oplus c_{23}\oplus c_{22}\oplus c_{21}\oplus c_{20}\oplus c_{19}\oplus c_{18}\oplus c_{17}\oplus c_{16}\oplus c_{15}\oplus c_{14}\oplus c_{13}\oplus c_{12}\oplus c_{11}\oplus c_{10}\oplus c_9\oplus c_8\oplus c_7\oplus c_6\oplus c_5\oplus c_4\oplus c_3\oplus c_2\oplus c_1\oplus c_0$$

根据扩展 Hamming 码的纠检错原理可知，$s_6s_5s_4s_3s_2s_1$ 与 s_0 的数值表示不同的错误类型，错误类型如表 1 所示。

计算得到(39,32)Hamming 码的伴随和与错误图样的对应关系如表 2 所示。进行纠错时，只需将发生错误的码字与对应的错误图样 $e_{38}e_{37}e_{36}\cdots e_2e_1e_0$ 进行对比异或操作，即可实现纠错。

表 1 扩展 Hamming 码的错误类型

Table1 Error type of extended Hamming code

$s_6s_5s_4s_3s_2s_1$	s_0	error type
0	0	no error
≠0	1	odd error
≠0	0	even error
0	1	parity error

表2 Hamming码错误图样

Table2 Error patterns of Hamming code

$s_6s_5s_4s_3s_2s_1s_0$	wrong pattern :$e_{38}e_{37}e_{36}\cdots e_2e_1e_0$	$s_6s_5s_4s_3s_2s_1s_0$	wrong pattern :$e_{38}e_{37}e_{36}\cdots e_2e_1e_0$
0000111	0000000000000000000000000000000001000	0101111	0000000000000001000000000000000000000
0001011	0000000000000000000000000000000100000	0110001	0000000000000010000000000000000000000
0001101	0000000000000000000000000000001000000	0110011	0000000000000100000000000000000000000
0001111	0000000000000000000000000000010000000	0110101	0000000000001000000000000000000000000
0010011	0000000000000000000000000001000000000	0110111	0000000000010000000000000000000000000
0010101	0000000000000000000000000010000000000	0111001	0000000000100000000000000000000000000
0010111	0000000000000000000000000100000000000	0111011	0000000001000000000000000000000000000
0011001	0000000000000000000000001000000000000	0111101	0000000010000000000000000000000000000
0011011	0000000000000000000000010000000000000	0111111	0000000100000000000000000000000000000
0011101	0000000000000000000000100000000000000	1000011	0000010000000000000000000000000000000
0011111	0000000000000000000001000000000000000	1000101	0000100000000000000000000000000000000
0100011	0000000000000000000010000000000000000	1000111	0001000000000000000000000000000000000
0100101	0000000000000000000100000000000000000	1001001	0010000000000000000000000000000000000
0100111	0000000000000000001000000000000000000	1001011	0100000000000000000000000000000000000
0101001	0000000000000000010000000000000000000	1001101	1000000000000000000000000000000000000
0101011	0000000000000000100000000000000000000	0000000	0000000000000000000000000000000000000
0101101	0000000000000001000000000000000000000		

2.2 基于FPGA的三模冗余仲裁设计

三模冗余基本原理是对需要加固的模块或单元进行复制，生成3个重复的电路模块或单元，再连接到多数表决器上，对其输出进行比较判决，依照少数服从多数的思想选择正确的输出值[10]。即使有一个模块产生故障，三模冗余仍能保证整个电路正常工作。由于2个模块或3个模块同时出错的概率极小，因此三模冗余的有效性与可信性较高[11]，是目前主流的抗SEU存储加固设计方法。

图3为三模冗余寄存器转换级电路(Register Transfer Level，RTL)整体示意图。本文在FPGA逻辑内部将PPC460存储器扩展出3组SRAM空间存储模块，将PLB写数据同时存入3组不同存储空间中，将SRAM的数据信息输入端连接汉明码编码器，生成具有纠一检二功能的校验位，存储至SRAM中；再将汉明码解码器与SRAM的输出端连接以译码和解码，实现对错误位的检测与纠正。基于FPGA的三模冗余仲裁器将PLB读数据进行三模冗余仲裁，即对存储器信号进行三取二的处理，找出其中发生错误的数据模块，用其他2个正确的模块数据进行数据替换，将仲裁后的数据传入PPC460内核，同时将出错模块信息上传。

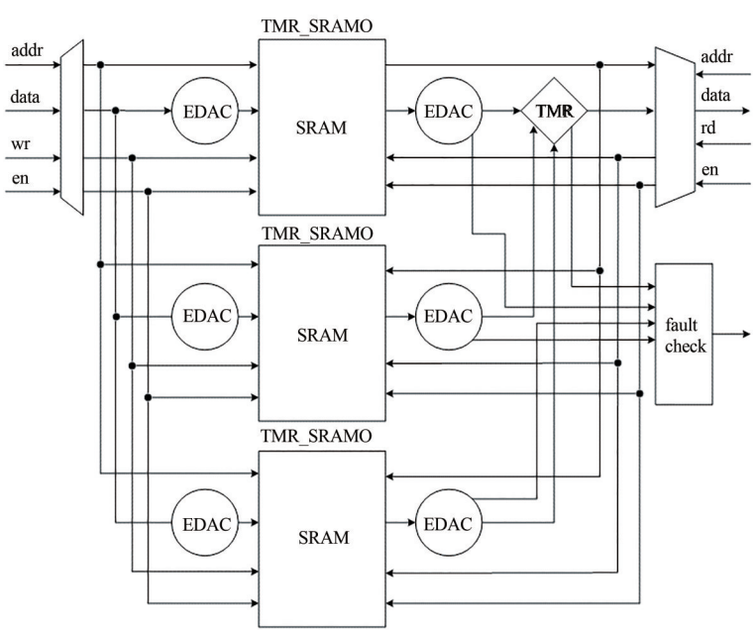

Fig3 Overall schematic of TMR RTL

图3 三模冗余RTL整体示意图

3 可靠性验证

3.1 验证流程

在FPGA仿真平台实现验证流程，通过DCR打开三模冗余、奇偶校验、汉明码编解码开关，运行PC故障注入软件进行故障数据注入，编写代码往安全空间(0x30000~0x3FFFF)内连续写入1~65 536数据，编写代码往安全空间内连续读取内存数据，比较读取数据是否异常，若读取数据正确，证明高可靠性防护设计成功。

3.2 故障注入

在FPGA中加入故障信号发生器，负责配置参数下传和数据接收显示功能，通过千兆网口(192.168.1.xx)和MicroBlaze软核进行数据交互，实现对单粒子翻转现象和三模冗余失效现象的故障模拟配合PC机软件模拟单粒子翻转现象SEU和三模冗余失效现象的故障，来验证EDAC模块是否将错误数据修复。故障信号发生器与EDAC空间的RTL关系如图4所示。

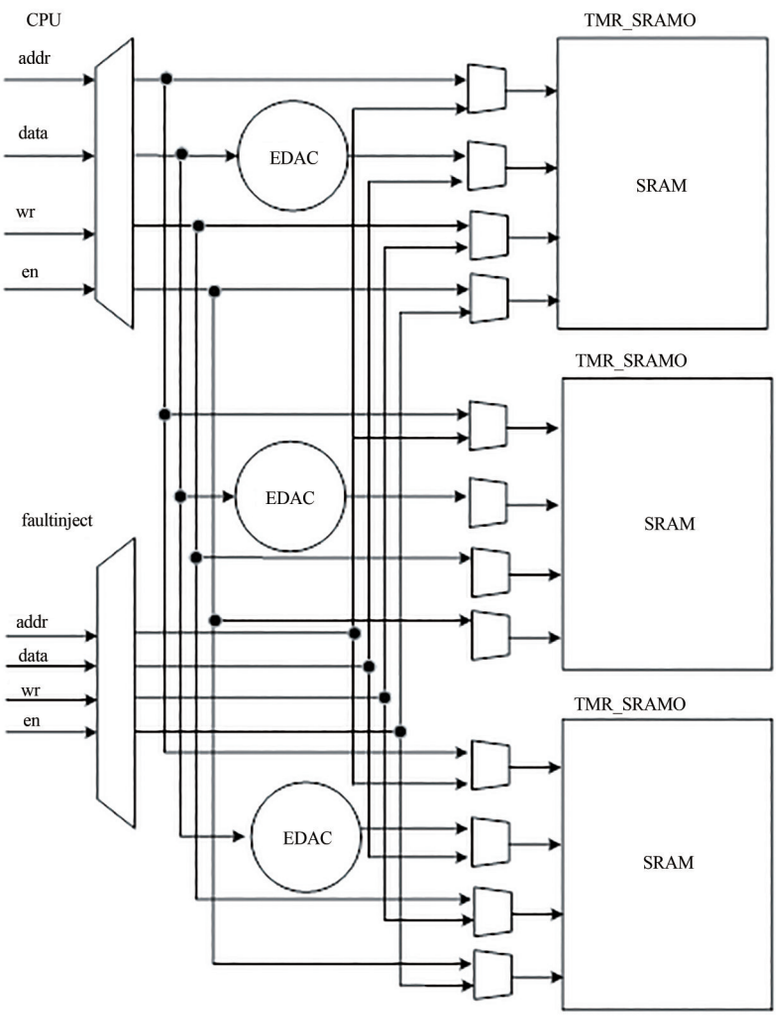

Fig.4 RTL diagram of fault signal generator and EDAC space
图4 故障信号发生器与EDAC空间的RTL关系图

图5为模拟单粒子翻转及三模冗余失效的信号图，图中：

1) cmd_edac_fault_en表示故障注入使能，产生红色高亮脉冲表示随机注入的故障位置；

2) cmd_edac_fault_addr表示故障注入的地址；

3) cmd_edac_fault_tmr表示三模冗余存储模块中发生错误的模块编号；

4) cmd_edac_fault_bitcnt表示发生错误的模块中发生翻转的bit位编号；

5) err_edac_flag表示故障标识信号，1表示出错，0表示未出错；

6) err_edac_addr表示检测出发生错误的模块地址；

7) err_edac_type 表示错误类型，可反映出错模块编号、错误检测与纠正情况。

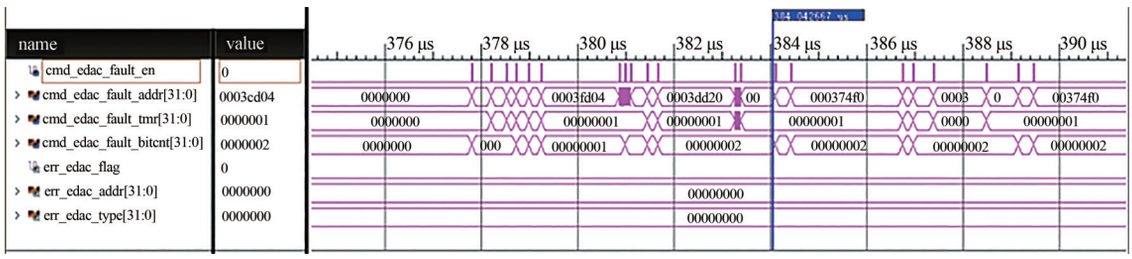

Fig.5 Simulated SEU and TMR failure signal
图 5　模拟单粒子翻转及三模冗余失效信号图

3.3　验证结果

PPC460 处理器传统的奇偶校验只能对写入的数据计算奇偶校验位，判断奇偶校验位的正确性，如果校验位错误会报警，但并不能纠正错误的数据。因此对原 SRAM 空间进行单粒子翻转故障注入，回读出来的数据仍是错误的，不能达到纠正错误的效果。

本文对安全空间(0x30000–0x3ffff)内任意一个地址进行单粒子翻转故障注入，如果三模冗余模块中 3 个存储模块中任意 2 个或 1 个模块发生 1 bit 位翻转时，可实现检错与纠错，提供错误信息并读出正确数据；当 3 个存储模块中任意 1 个模块发生 2 bit 位翻转时，可实现检错，提供错误信息并读出正确数据；当 3 个存储模块中任意 2 个模块发生 2 bit 位翻转时，不能实现检错与纠错，只能提供出错信息。三模冗余与扩展型汉明码可靠性增强设计仿真结果如图 6 所示。

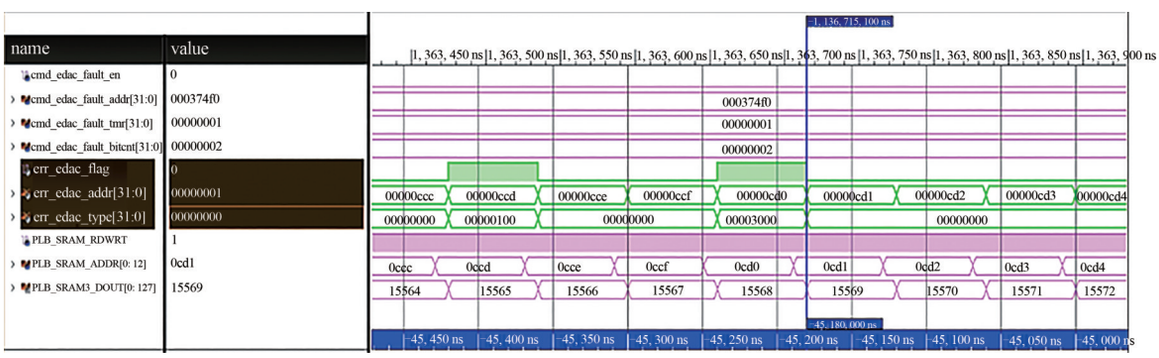

Fig.6 Simulation results of TMR and extended Hamming code reliability enhancement design
图 6　三模冗余与扩展型汉明码纠一检二验证结果

从图 6 可以看到，当有故障数据注入时，读取内存数据后故障标识信号 err_edac_flag 值会高于正常值，显示为绿色突出高亮块；当 err_edac_addr 为 0x0000ccd 时，err_edac_type 为 0x00000100，表示第 2 个模块发生了 1 bit 位翻转，并纠正了错误，正确读出了数据，PLB_SRAM3_DOUT 为 15 565，其数值与左右连贯，数据正常；当 err_edac_addr 为 0x0000cd0 时，err_edac_type 为 0x00003000，表示第 2 个模块发生了 1 bit 位翻转，并纠正了错误，正确读出了数据，PLB_SRAM3_DOUT 为 15 568，其数值与左右连贯，数据正常。综上，实现了纠一检二功能，符合预期。

4　结论

本文利用 FPGA 对 PPC460 处理器进行逻辑实现，在 FPGA 内部扩展高安全性空间，通过将故障注入系统连接 MicroBlaze 软核，对该高安全性空间内进行故障数据注入。针对 PPC460 处理器 128 位宽 PLB 总线及 DCR 总线的特点，同时采用奇偶校验、三模冗余与余数纠错技术，并通过技术改良及逻辑代码优化，提高了该空间内数据的正确性，成功实现 100 MHz 主频下高安全性空间的纠错特性验证。该可靠性增强设计方法有效降低了单粒子翻转效应对 PPC460 处理器的安全隐患，实现了对重要数据的高安全、高可靠、抗干扰保护[12]。

参考文献：

[1] RAJAEI R,ASGARI B,TABANDEH M,et al. Design of robust SRAM cells against single-event multiple effects for nanometer technologies[J]. IEEE Transactions on Device and Materials Reliability, 2015,15(3):429–436. doi:10.1109/TDMR.2015.2456832.

[2] DYER C, HANDS A, RYDEN K, et al. Extreme atmospheric radiation environments and single event effects[J]. IEEE Transactions on Nuclear Science, 2018,65(1):432-438. doi:10.1109/TNS.2017.2761258.

[3] 杨亮,李祁. 基于三模冗余架构的航天器FPGA可靠性设计[J]. 计算机测量与控制, 2019,27(12):244-248. (YANG Liang,LI Qi. Spacecraft FPGA reliability design based on three-mode redundant architecture[J]. Computer Measurement & Control, 2019, 27(12):244-248.) doi:10.16526/j.cnki.11-4762/tp.2019.12.052.

[4] ROTH D R,MCNULTY P J,ABDEL-KADER W G,et al. Monitoring SEU parameters at reduced bias(CMOS SRAM)[J]. IEEE Transactions on Nuclear Science, 1993,40(6):1721-1724. doi:10.1109/23.273487.

[5] MAVIS D G,EATON P H. SEU and SET modeling and mitigation in deep submicron technologies[C]// 2007 IEEE International Reliability Physics Symposium Proceedings. Phoenix,AZ,USA:IEEE, 2007:293-305. doi:10.1109/RELPHY.2007.369907.

[6] 肖翰珅,胡剑浩,马上. 冗余余数系统低复杂度快速纠错算法设计[J]. 电子与信息学报, 2015,37(8):1944-1949. (XIAO Hanshen,HU Jianhao,MA Shang. Low-complexity error correction algorithms for redundant residue number systems[J]. Journal of Electronics & Information Technology, 2015,37(8):1944-1949.) doi:10.11999/JEIT141454.

[7] 张玉艳,陈萍,赵京玺. 用FPGA实现纠错编码的一种方法[J]. 实验技术与管理, 2003,20(3):48-50,58. (ZHANG Yuyan,CHEN Ping,ZHAO Jingxi. A method of error correction coding with FPGA[J]. Experimental Technology and Management, 2003,20(3): 48-50,58.) doi:10.3969/j.issn.1002-4956.2003.03.014.

[8] 李海旭. 一种FPGA程序模块间距离的建模方法[J]. 单片机与嵌入式系统应用, 2019,19(10):16-17,21. (LI Haixu. Modelling method of distance between FPGA program modules[J]. Microcontrollers & Embedded Systems, 2019,19(10):16-17,21.)

[9] 钟敏,苏海冰,潘广涛. FPGA块存储器的多位翻转容错设计[J]. 计算机测量与控制, 2021,29(5):169-173,178. (ZHONG Min, SU Haibing, PAN Guangtao. Multiple bit upset fault tolerance design of FPGA block memory[J]. Computer Measurement & Control, 2021,29(5):169-173,178.) doi:10.16526/j.cnki.11-4762/tp.2021.05.034.

[10] 王鹏,刘正清,田毅. 一种面向SRAM型FPGA的三模冗余分区自修复方法研究[J]. 电子技术应用, 2021,47(6):92-95. (WANG Peng, LIU Zhengqing, TIAN Yi. A recovery methodology for SRAM-based FPGA partitioned TMR[J]. Application of Electronic Technique, 2021,47(6):92-95.) doi:10.16157/j.issn.0258-7998.200384.

[11] 段小虎,马小博,程俊强. SRAM工艺FPGA三模冗余设计故障管理与恢复[J]. 信息通信, 2020(3):139-141,143. (DUAN Xiaohu, MA Xiaobo, CHENG Junqiang. Fault management and recovery of FPGA three-modular redundancy design in SRAM process[J]. Information & Communications, 2020(3):139-141,143.) doi:10.3969/j.issn.1673-1131.2020.03.059.

[12] 严健生,杨柳青. 卫星用SRAM型FPGA抗单粒子翻转可靠性设计研究[J]. 科技创新与应用, 2021(9):48-50,53. (YAN Jiansheng, YANG Liuqing. Reliability design of anti-single event upset(SEU) of SRAM-FPGA for satellites[J]. Technology Innovation and Application, 2021(9):48-50,53.)

作者简介:

杨 渊(1995-),女,硕士,工程师,主要研究方向为嵌入式软件开发.email:492445923@qq.com.

邹祖伟(1992-),男,硕士,工程师,主要研究方向为嵌入式软件开发.