文章编号:1674-2974(2025)12-0197-09 DOI:10.16339/j.cnki.hdxbzkb.2025298

SRAM单粒子效应检错电路研究与设计

白创 ,李云杰 †
(长沙理工大学 物理与电子科学学院,湖南 长沙 410114)

摘 要:为了解决单粒子效应引起的航天器静态随机存取存储器数据出错难题,本文研究了基于错误检查纠正电路和完整性检测器相结合的静态随机存取存储器在线检测错误方法及电路实现技术.其中,错误检查纠正电路采用(39,32)汉明码设计,实现数据访问时单比特错误自动检错与纠错.完整性检测器基于哈希函数校验原理设计,实现对数据周期性循环检查.基于CMOS 0.18 μm工艺设计实现数据在线检错电路.仿真结果表明,该电路能够主动周期性对内存数据进行检查,修复单比特错误和检测出多比特错误,有效提高数据的可靠性.

关键词:单粒子效应;静态随机存取存储器;数据可靠性;错误检测纠正电路;哈希函数校验

中图分类号:TP302 **文献标志码**:A

Research and Design of SRAM Error Detection Circuit Based on Single Event Effect

BAI Chuang, LI Yunjie†
(School of Physics & Electronic Science, Changsha University of Science & Technology, Changsha 410114, China)

Abstract: To solve the problem of spacecraft static random access memory data error caused by single event effect, the static random access memory online error detection method and circuit implementation technology, based on the combination of error checking and correcting circuit and integrity detector, are studied in this paper. The error checking and correcting circuit is designed using (39, 32) Hamming code to realize automatic error detection, and error correction of volatile single-bit when data is accessed. The integrity detector is designed based on the principle of hash function verification to realize periodic cyclic checking of data. Data online error detection circuit is fabricated in CMOS 0.18 μm process. Simulation results show that the circuit can actively periodically check the memory data, correct single-bit errors and detect multi-bit errors, and effectively improve data reliability.

Key words: single event effect; static random access memory; data reliability; error checking and correcting circuit; hash function verification

---

* **收稿日期**:2024-11-21
基金项目:湖南省教育厅科研项目(22B0287), Scientific Research Program of Hunan Provincial Department of Education(22B0287)
作者简介:白创(1983—),男,陕西延安人,长沙理工大学讲师,博士
† 通信联系人,E-mail:624644099@qq.com

航天器所处环境存在大量高能粒子[1]，这些粒子轰击作用于集成电路敏感区域，引发单粒子效应(single event effects，SEE)、总剂量效应等[2]，其中SEE尤为明显，导致电路工作状态发生临时变化、功能暂时或永久性异常.SEE引起的短暂性扰动软错误可以通过电路重启或数据刷新来消除[3]，但时间和资源的消耗降低了工程上对重置方案的使用[4].而可靠性加固方法只需占用少量资源且纠检错效率高，在航天芯片领域得到广泛应用[5].

抗单粒子可靠性研究主要针对工艺级、版图级、电路级三个方面进行加固.在工艺级加固方面，叶甜春等[6]介绍了一种可配置的绝缘硅(configurable silicon on insulator，CSOI)器件技术，能有效提高器件和电路的抗辐照性，但温度和总剂量辐射增大了器件的单粒子瞬态电流，会导致电路发生短暂翻转.在版图级加固方面，苑靖爽等[7]基于敏感节点对分离和电荷补偿原理，设计了2种触发器版图结构，经过抗单粒子实验，翻转数可下降95%以上，但随着工艺发展，物理隔离加固方法会导致面积与性能损失变大.而目前针对电路的加固研究占大多数，抗SEE效果显著.电路级加固方面包括三模冗余技术、双互锁存单元(dual interlocked storage cell，DICE)加固技术、时间冗余技术、纠错码技术等.宁亚飞[8]基于三模冗余与三级错误拦截实现锁存器设计，以牺牲21.95%的面积开销为代价，使功耗降低20.77%.Maru等[9]证明了与三模冗余触发器相比，DICE在面积和速率方面具有很大优势，但DICE中敏感节点易受粒子的影响，导致功能异常；Yue等[10]提出节能有效错误检查纠正(efficient error checking and correcting，EFF-ECC)机制可抗关键错误，虽然降低传统单纠错双纠错监测(single error correction-double error detection，SEC-DED)以及错误检查纠正(error checking and correcting，ECC)86.46%的能耗，但面积损耗与错误检测的实时性还是不容乐观.综上，亟须一种低面积、纠错效率高、及时检错的监控检查方法，以确保宇航级电路的正常使用.

因此，本文构造了一个基于ECC电路和完整性检测器的高性能低代价系统级在线内存检测和修复电路，以提高航天电子系统的静态随机存取存储器(static random access memory，SRAM)可靠性.由于空间翻转多出现在单比特错误情况，本文设计在读写模式下可自动纠正1位错误，周期性检测多位错误，及时反馈至内核，以防止无法及时获取正确数据.

1 SRAM系统级加固设计

SRAM[11]通常采用6管体单元，存储单元受辐射易发生翻转，本文针对SRAM存储器研究系统级加固设计.由于空间应用中出现多位翻转的概率较小，因此本文研究了纠正一位错误和检测多位错误的解决方案，即采用内置完整性检测电路与ECC电路相结合的设计，其实现结构如图1所示.

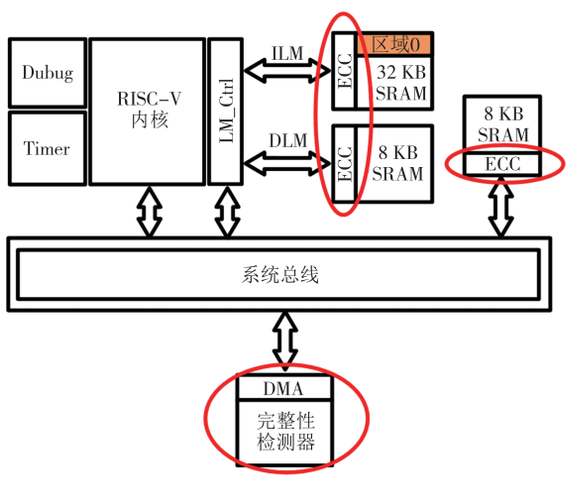

图1 系统级加固设计结构

Fig.1 System level reinforcement design structure

每个SRAM内存外均嵌入ECC电路.ECC电路采用汉明编码方式，实现纠正1位错误与检查2位错误的功能.数据在存入SRAM之前经过编码电路进行编码，读取数据时经过解码电路进行解码.若检测到1位错误，ECC电路会对可修复错误进行纠错，然后将数据存入SRAM；检测到2位错误，ECC电路会发送一个异常信号给内核.但是，在面对2位以上错误时，ECC电路会出现识别出错的可能，即SEE诱发某些多位错误情况，译码计算的校正子出现等于某个1位错对应的校正子或0，此时ECC电路则认为出现1位错误或数据正确，导致错译、错认正确的情况.

针对这些多位错误的情况，本文在ECC电路的基础上增加完整性检测电路进行周期性检测，以检查出ECC电路无法识别或识别出错的数据翻转错误.完整性检测器根据循环检测的设置周期，周期性访问SRAM的区域数据，将传输的所有数据通过内部安全散列运算(secure hach algorithm，SHA)模块计算出哈希值.若重新计算得到的哈希值与哈希

参考值比对不一致，完整性检测电路将会发送一个中断信号给内核.

2　ECC电路设计

SEE易导致单比特错误，因此本文采用(39，32)汉明编码方式，可以有效地检测和纠正1位错误，并发现2位错误，从而有效降低系统数据存储的错误率.在读取数据时，基于汉明码的ECC电路进行错误检测和修改，其工作电路结构如图2所示.

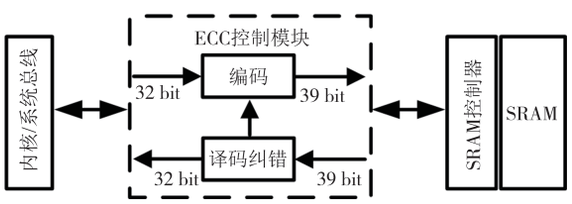

图2　处于总线与控制器之间的ECC模块

Fig.2　ECC module between the bus and the controller

ECC电路控制器调动编码和译码电路的工作，且能产生中断信号.向SRAM写入数据时，编码器将32位原始数据经过冗余编码，得到39位数据，其中包含32位原始数据和7位校验数据，并将该数据保存在存储单元阵列中.从SRAM中读取数据时，译码器对读出的39位数据进行解码校验，以获得正确的32位数据.如果发现1位错误，该电路将对数据进行纠错恢复;检测到2位错误，无法进行纠正，此时会将错误信息反馈给ECC电路的控制器.ECC电路控制器将会产生一个异常信号，随后发送给内核.

2.1　编码过程

假设原始数据为$\boldsymbol{M}=(M_{31},\cdots,M_1,M_0)$，可由纠错码原理[12]计算出生成矩阵$\boldsymbol{G}$，编码过程即利用数据矢量$\boldsymbol{M}$乘以生成矩阵$\boldsymbol{G}$，得到编码后的码字$\boldsymbol{C}=(C_{38},\cdots,C_1,C_0)$.其中$C_{31}$~$C_0$为信息位，$C_{38}$~$C_{32}$为校验位，$\boldsymbol{C}$中各位的计算表达式如公式(1)所示.

编码仿真示例如图3所示，其中32位数据i_dat[31:0]为32'h151e，经过编码后，生成7位校验位7'h5d，与原始32位数据一起构成39位数据，输出din[38:0]为39'h5d_0000_151e，存储进SRAM.

$$
\begin{aligned}
&C_i = M_i,\ \ i = 0,1,2,\cdots,31;\\
&C_{32} = M_1\oplus M_3\oplus M_4\oplus M_6\oplus M_8\oplus M_{10}\oplus M_{11}\oplus M_{13}\oplus M_{15}\oplus M_{17}\oplus M_{19}\oplus M_{21}\oplus M_{23}\oplus M_{25}\oplus M_{26}\oplus M_{28}\oplus M_{30};\\
&C_{33} = M_0\oplus M_2\oplus M_3\oplus M_5\oplus M_6\oplus M_9\oplus M_{10}\oplus M_{12}\oplus M_{13}\oplus M_{16}\oplus M_{17}\oplus M_{20}\oplus M_{21}\oplus M_{24}\oplus M_{25}\oplus M_{27}\oplus M_{28}\oplus M_{31};\\
&C_{34} = M_1\oplus M_2\oplus M_3\oplus M_7\oplus M_8\oplus M_9\oplus M_{10}\oplus M_{14}\oplus M_{15}\oplus M_{16}\oplus M_{17}\oplus M_{22}\oplus M_{23}\oplus M_{24}\oplus M_{25}\oplus M_{29}\oplus M_{30}\oplus M_{31};\\
&C_{35} = M_4\oplus M_5\oplus M_6\oplus M_7\oplus M_8\oplus M_9\oplus M_{10}\oplus M_{18}\oplus M_{19}\oplus M_{20}\oplus M_{21}\oplus M_{22}\oplus M_{23}\oplus M_{24}\oplus M_{25};\\
&C_{36} = M_{11}\oplus M_{12}\oplus M_{13}\oplus M_{14}\oplus M_{15}\oplus M_{16}\oplus M_{17}\oplus M_{18}\oplus M_{19}\oplus M_{20}\oplus M_{21}\oplus M_{22}\oplus M_{23}\oplus M_{24}\oplus M_{25};\\
&C_{37} = M_{26}\oplus M_{27}\oplus M_{28}\oplus M_{29}\oplus M_{30}\oplus M_{31};\\
&C_{38} = M_0\oplus M_1\oplus M_2\oplus M_3\oplus M_4\oplus M_5\oplus M_6\oplus M_7\oplus M_8\oplus M_9\oplus M_{10}\oplus M_{11}\oplus M_{12}\oplus M_{13}\oplus M_{14}\oplus M_{15}\oplus M_{16}\oplus M_{17}\oplus M_{18}\oplus M_{19}\oplus M_{20}\oplus M_{21}\oplus M_{22}\oplus M_{23}\oplus M_{24}\oplus M_{25}\oplus M_{26}\oplus M_{27}\oplus M_{28}\oplus M_{29}\oplus M_{30}\oplus M_{31}\oplus C_0\oplus C_1\oplus C_2\oplus C_3\oplus C_4\oplus C_5 \qquad (1)
\end{aligned}
$$

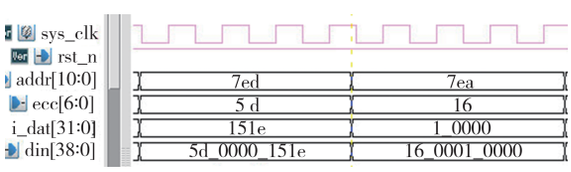

图3　编码仿真示例

Fig.3　Coding simulation example

2.2　译码过程

译码过程可以划分为以下步骤：

1)校正子的计算.

2)错误图样的求解.

3)纠错并还原为原始数据.校正子$\boldsymbol{S}$的计算方法为：

$$\boldsymbol{S} = \boldsymbol{C}\times\boldsymbol{H}^{\mathrm{T}} \qquad (2)$$

式中:$\boldsymbol{H}$为校验矩阵.根据纠错码性质构造出$\boldsymbol{H}$矩阵后[13]，可计算得到校正子$\boldsymbol{S}$，其中奇偶校验Parity、S_5~S_0的计算表达式分别为：

$$
\begin{aligned}
&S_0 = C_0\oplus C_1\oplus C_3\oplus C_4\oplus C_6\oplus C_8\oplus C_{10}\oplus C_{11}\oplus C_{13}\oplus C_{15}\oplus C_{17}\oplus C_{19}\oplus C_{21}\oplus C_{23}\oplus C_{25}\oplus C_{26}\oplus C_{28}\oplus C_{30}\oplus C_{32};\\
&S_1 = C_0\oplus C_2\oplus C_3\oplus C_5\oplus C_6\oplus C_9\oplus C_{10}\oplus C_{12}\oplus C_{13}\oplus C_{16}\oplus C_{17}\oplus C_{20}\oplus C_{21}\oplus C_{24}\oplus C_{25}\oplus C_{27}\oplus C_{28}\oplus C_{31}\oplus C_{33};\\
&S_2 = C_1\oplus C_2\oplus C_3\oplus C_7\oplus C_8\oplus C_9\oplus C_{10}\oplus C_{14}\oplus C_{15}\oplus C_{16}\oplus C_{17}\oplus C_{22}\oplus C_{23}\oplus C_{24}\oplus C_{25}\oplus C_{29}\oplus C_{30}\oplus C_{31}\oplus C_{34};\\
&S_3 = C_4\oplus C_5\oplus C_6\oplus C_7\oplus C_8\oplus C_9\oplus C_{10}\oplus C_{18}\oplus C_{19}\oplus C_{20}\oplus C_{21}\oplus C_{22}\oplus C_{23}\oplus C_{24}\oplus C_{25}\oplus C_{35}\oplus C_{31}\oplus C_{33};\\
&S_4 = C_{11}\oplus C_{12}\oplus C_{13}\oplus C_{14}\oplus C_{15}\oplus C_{16}\oplus C_{17}\oplus
\end{aligned}
$$

$C_{18}\oplus C_{19}\oplus C_{20}\oplus C_{21}\oplus C_{22}\oplus C_{23}\oplus C_{24}\oplus C_{25}\oplus C_{36}$;

$S_5 = C_{26}\oplus C_{27}\oplus C_{28}\oplus C_{29}\oplus C_{30}\oplus C_{31}\oplus C_{37}$;

$\text{Parity} = C_0\oplus C_1\oplus C_2\cdots\oplus C_{36}\oplus C_{37}\oplus C_{38}$ (3)

根据纠错码原理可知:

$\boldsymbol{S} = \boldsymbol{C}\times\boldsymbol{H}^{\mathrm{T}} = (\boldsymbol{R}+\boldsymbol{E})\times\boldsymbol{H}^{\mathrm{T}} = \boldsymbol{E}\times\boldsymbol{H}^{\mathrm{T}}$ (4)

式中:$\boldsymbol{R}$为接收向量;由伴随式$\boldsymbol{S}$和校验矩阵$\boldsymbol{H}$可求解出错误图案$\boldsymbol{E}$.本文采用查找表的方式,使得错误图案的求解过程全由组合逻辑构成.该方法原理如下:预先穷举出所有可能的错误模式,对每一种错误模式$\boldsymbol{E}_i$乘以$\boldsymbol{H}^{\mathrm{T}}$,得到对应的伴随式$\boldsymbol{S}$,从而建立起所有$\boldsymbol{E}$和$\boldsymbol{S}$之间的对应关系,保存于查找表中.在译码过程中,根据$\boldsymbol{S}$的值查找出错误模式$\boldsymbol{E}$即可,错误位置与校正子如表1所示.

表1 纠正单个误码的校正子表

Tab.1 Correction sub-table for correcting individual errors

误码位置	校正子	误码位置	校正子	误码位置	校正子	误码位置	校正子
0	6'd3	8	6'd13	16	6'd22	24	6'd30
1	6'd5	9	6'd14	17	6'd23	25	6'd31
2	6'd6	10	6'd15	18	6'd24	26	6'd33
3	6'd7	11	6'd17	19	6'd25	27	6'd34
4	6'd9	12	6'd18	20	6'd26	28	6'd35
5	6'd10	13	6'd19	21	6'd27	29	6'd36
6	6'd11	14	6'd20	22	6'd28	30	6'd37
7	6'd12	15	6'd21	23	6'd29	31	6'd38

译码仿真示例如图4,SRAM输出的39位数据dout[38:0]进入译码电路,首先计算校正子$\boldsymbol{S}$[6:0],然后查表得到错误图样$\boldsymbol{E}$[38:0],对错误进行校正,解析出正确的32位数据.

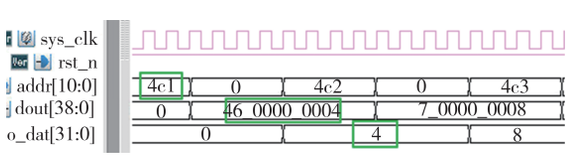

图4 译码仿真示例

Fig.4 Encoding simulation example

3 完整性检测器设计

本文使用基于哈希函数的存储器完整性校验方法.该方法基于哈希函数的校验原理,通过依据访问存储域的特征信息,实施高效的校验过程.其区域监测结构如图5所示.完整性检测器依据内存i2中域描述符的相关信息对内存区域0~3的区域进行监测,并将计算的区域哈希参考值存放在哈希区域中.

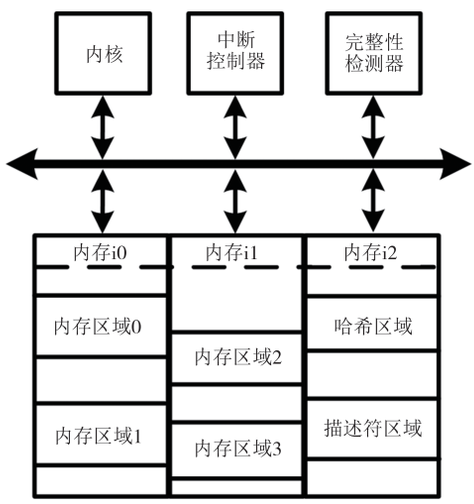

图5 4个区域监测结构

Fig.5 Four regional monitoring structures

当完整性检测器使能后,根据域描述符的信息,检测指定存储区域,将存储区域中的数据传输至SHA引擎中,计算出哈希摘要值,并将其存放于控制器和处理器均可访问的连续系统存储区域中.当内存检查器配置为主动持续监控时,域描述符列表所指向的存储域会被主动监控,即周期性对存储域数据进行哈希计算,计算其摘要值与参考摘要进行比较.若发生摘要值不匹配的情况,则触发一次中断.

在SRAM中存放域描述符列表如图6所示,每个域描述符由4个字组成,对应存储区域的相关信息.区域起始地址为该描述符关联的数据存储区域的起始地址;区域配置信息包括哈希算法配置、回绕监控等配置;区域控制信息为当前描述符对应当前存储区域块的大小;区域下一地址为下一区域描述符的起始地址.

3.1 内部结构

完整性检测器是一个依据位于内存描述符区域的区域描述符信息,对多个内存区域执行哈希计算的直接内存访问控制器.其内部结构如图7所示,完整性检测器集成了一个直接内存访问(direct memory access, DMA)接口,一个监控有限状态机(finite-state machine, FSM),一个完整性调度模块,一个上下文寄存器组,一个SHA运算模块,一个可配置接口和寄存器组.

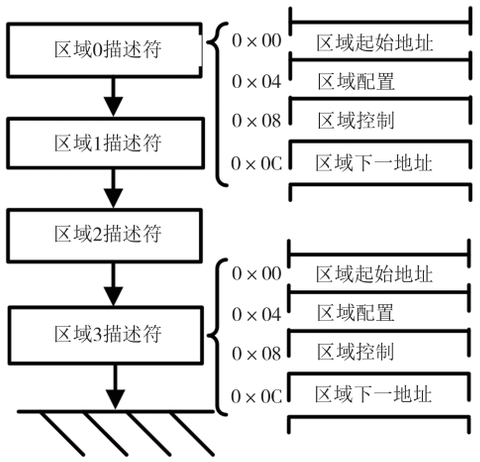

图 6　域描述符列表内部结构

Fig.6　Domain descriptor list internal structure

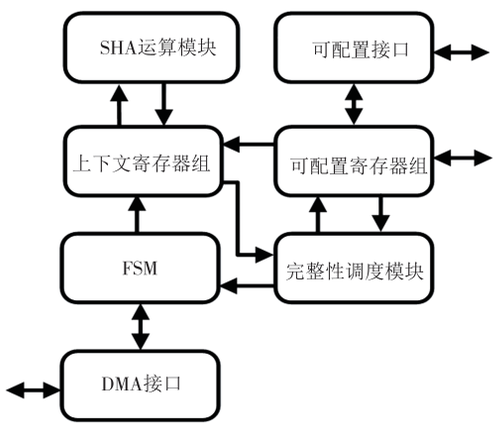

图 7　完整性检测器内部结构

Fig.7　Integrity detector internal structure

SHA 运算模块集成了 SHA-1、SHA-224 以及 SHA-256 算法. 数据被送至 SHA 运算模块，根据描述符提供的算法类型，计算出哈希摘要值.

上下文寄存器组在读取描述符阶段存放和更新对应的描述符，输出描述符的配置和状态信息分别提供 SHA 运算模块和完整性调度使用；在哈希运算阶段用于填写待运算数据；在摘要比对阶段用于读取和填写目标摘要值.

FSM 根据 DMA 接口状态控制当前区域的描述符读取、数据搬运和哈希结果读写更新，同时向 DMA 接口提供当前控制信息. DMA 用于对 FSM 当前状态的控制进行系统总线读写，完成 DMA 数据传输功能，单次写/读操作数据为 32 位.

通过可配置接口对寄存器组进行读写操作. 可配置寄存器组再将配置信息传送至完整性调度模块，同时根据反馈的中断原始信号和配置的中断掩码信号合并输出中断信号.

完整性调度模块根据寄存器配置信息向 FSM 提供必要的控制信息，以便进行描述符读取、数据读取和哈希结果对比和写回等操作的宏观调度. 依据系统配置和描述符配置和状态信息，产生中断原始信号和调度控制信号.

3.2　工作流程

完整性检测器工作流程如图 8 所示. 首先通过可配置接口获取来自应用的配置信息，其中包括使能配置、区域描述符起始地址、哈希运算结果存储起始位置、等待时间周期等. 其中等待时间周期为当前处理结束到下一块传输之间的系统时钟循环，可最多插入 32 768 个循环，用于控制内存完整性检查器的访存行为.

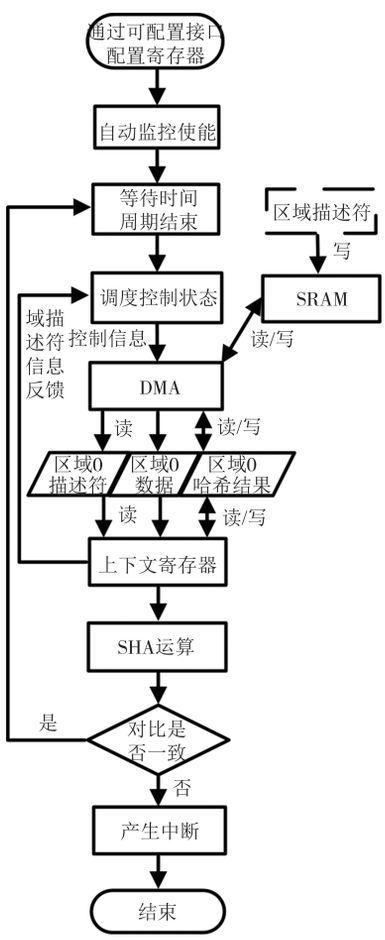

图 8　完整性检测器监控流程图

Fig.8　Integrity detector monitoring flowchart

当读取监控使能信号后，开始自动的周期性监控. 执行等待时间后，由调度器发送指令控制 DMA 依据区域描述符起始地址数据读取区域描述符并存入上下文寄存器. 此后，调度器读取区域描述符中的区域配置信息，并执行相关指令，即控制 DMA 读

取指定位置的内存数据，存入上下文寄存器等. 读取完内存数据后，调度器控制SHA运算模块对读取到的内存数据进行哈希运算，并保存其为哈希参考值. 此后，在每个周期内，完整性检测器再次依照同样流程对内存数据进行SHA运算，并将运算后的结果与哈希初始值比对，若比对结果不一致，则发送中断信号给CPU. 若比对结果一致，则继续周期性检查.

4 仿真

为验证本文设计电路对空间中常见的SEE造成的错误的检测、纠正能力，分别设计单比特错误、两比特错误和多比特错误仿真，错误注入和周期性检测实现过程如图9所示，原始数据通过前门访问方式写入SRAM，错误注入是通过后门访问方式强行修改原始数据，模拟单比特错误、两比特错误和多比特错误. 此外，还设计了软硬件纠检错效率比对仿真. 所有仿真数据均采用示例区域0的数据结构.

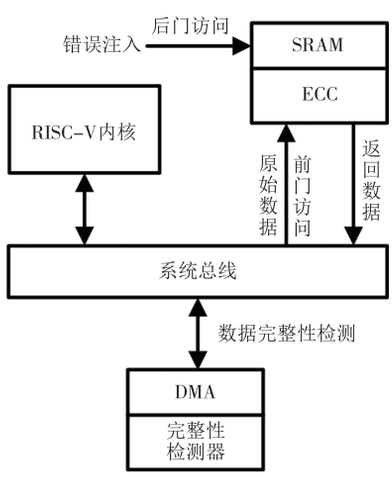

图9 错误注入和周期性检测实现过程

Fig.9 Error injection and periodic detection

在存储区域内存放一个区域0的描述符，如图10所示，包括区域起始地址为0x0000_7600；区域配置为0x00001002表示哈希算法配置SHA-256、回绕监控同一块存储域；区域控制为0x00000003表示对应数据块大小是(3+1)块；存储域次级链表起始地址为0x0000_0000表示没有次级存储域. 区域0描述符对应存储的存储区域0的4块数据块如图11所示.

Addr/Hint	[0][26:0]	[1][26:0]	[2][26:0]	[3][26:0]
[1c00][26:0]	5400007600	1700001002	0600000003	0000000000

图10 区域0描述符

Fig.10 Area 0 descriptor

Addr/Hint	[0][26:0]	[1][26:0]	[2][26:0]	[3][26:0]
[1d80][26:0]	0000000000	4600000004	0700000008	410000000c
[1d84][26:0]	4900000010	0f00000014	4e00000018	080000001c
[1d88][26:0]	4a00000020	0c00000024	4d00000028	0b0000002c
[1d8c][26:0]	0300000030	4500000034	0400000038	420000003c
[1d90][26:0]	0000000000	0000000000	0000000000	0000000000
[1d94][26:0]	0000000000	0000000000	0000000000	0000000000
[1d98][26:0]	0000000000	0000000000	0000000000	0000000000
[1d9c][26:0]	0000000000	0000000000	0000000000	0000000000
[1da0][26:0]	0000000000	0000000000	0000000000	0000000000
[1da4][26:0]	0000000000	0000000000	0000000000	0000000000
[1da8][26:0]	0000000000	0000000000	0000000000	0000000000
[1dac][26:0]	0000000000	0000000000	0000000000	0000000000
[1db0][26:0]	2680000000	0000000000	0000000000	0000000000
[1db4][26:0]	0000000000	0000000000	0000000000	0000000000
[1db8][26:0]	0000000000	0000000000	0000000000	0000000000
[1dbc][26:0]	0000000000	0000000000	0000000000	4100000600

图11 区域0数据

Fig.11 Region 0 data

完整性检测器根据区域0域描述符的信息，取出区域0对应位置的数据. 每次取512位数据送入SHA计算单元中，经过多次迭代计算，直至将数据传输完毕，计算得出区域0的哈希参考值5af3_90e8_64b7_c824_c363_6816_5a13_c941_6ff9_3b9b_6885_abbe_c9e0_af41_7e26. 根据配置的哈希区域起始地址1d00进行保存，如图12所示.

Addr/Hint	[1c00][26:0]	[1c01][26:0]	[1c02][26:0]	[1c03][26:0]
[1d00][26:0]	46af417e26	7ec9e0413b	2f6885abbe	1b6ff93b9b
[1d04][26:0]	755a13c941	18c3636816	3964b7c824	775af390e8

图12 哈希参考值存储区域

Fig.12 Hash reference storage area

4.1 单比特错误仿真

为验证本文方法对单比特错误的纠检错能力，将区域0地址1d82的数据07_0000_0008通过后门访问方式强制转换成07_0000_0000，以模拟SRAM内部发生1 bit数据翻转情况，如图13所示.

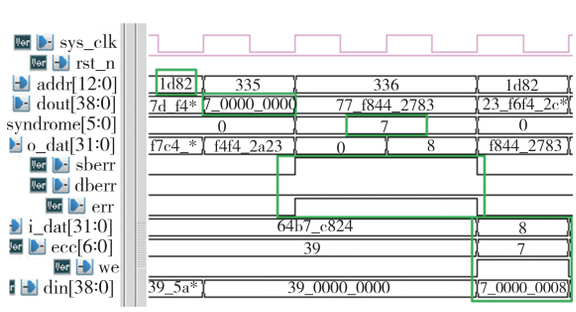

图13 ECC检查1 bit错误

Fig.13 ECC check for 1 bit errors

当完整性检测器使能循环访问SRAM区域0数据或内核访问区域0的数据时，ECC电路会访问到地址1d82的数据，SRAM输出dout[38:0]为7_0000_0000，计算出的校正子syndrome[5:0]等于7，对应单比特错误位置为第3位，ECC电路发现数据错误，置起相应的sberr和err标志. 对1 bit错误进行纠正

后的数据 o_dat[31:0]为 0000_0008，传送给 ECC 电路编码块重新编码，i_dat[38:0]和 ecc[6:0]组成 din[38:0]等于 7_0000_0008，写入地址 1d82 中.

此外,完整性检测器对区域 0 的检查结果如图 14 所示.图中 icm_irq 并未生成中断. 其原因为:当循环检测该内存区域时，ECC 电路已自动对单比特错误进行纠错，地址 1d82 译码的数据通过总线输入完整性检查模块的 wadta[31:0]为 0000_0008. 因此，完整性检查器仍对未错误的数据进行哈希运算得到哈希计算值 cur_digest[255:0]，与哈希参考值 ref_digest[255:0]比对结果一致，因此未产生中断.

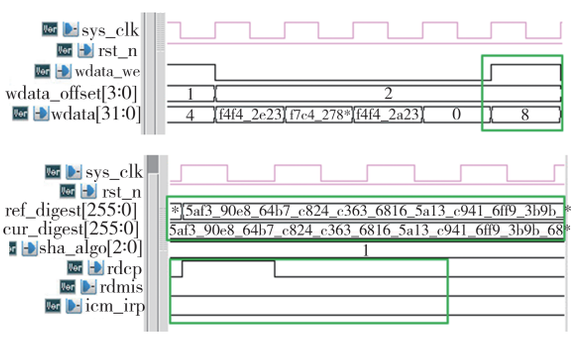

图 14　1 bit 错误完整性检测

Fig.14　Integrity detection 1 bit error

因此，当发生单比特错误时，ECC 电路能自动检错并纠错，完整性监控检查器会进行巡检判断数据是否正确.

4.2　2 bits 错误仿真

为验证本文设计对 2 bits 错误的检错能力，将区域 0 地址 1d82 的数据 07_0000_0008 通过后门访问方式强制转换为 07_0000_0001，以模拟 SRAM 内部发生 2 bits 数据翻转情况. ECC 电路工作过程如　图 15 所示.

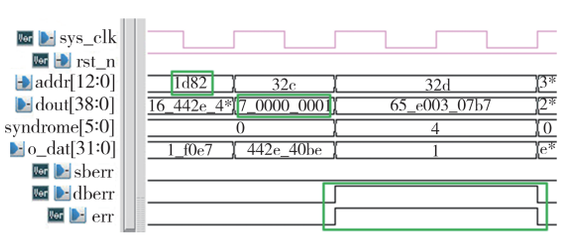

图 15　ECC 检查 2 bits 错误

Fig.15　ECC checks for 2 bits errors

当完整性检测器使能循环访问 SRAM 区域 0 的数据时，ECC 电路访问地址 1d82 的数据，SRAM 输出 dout[38:0]为 7_0000_0001，计算得到校正子等于 4，校正子表没有该校正子值则无法纠错，ECC 电路发现数据错误，置起相应的 dberr 和 err 标志.

此外，完整性检测器对区域 0 的检测结果如图 16 所示. 图中 icm_irq 生成中断. 其原因为:当检测内存区域 0 时，ECC 电路无法对两位错误进行纠错，地址 1d82 译码的数据通过总线传输到完整性检查模块的 wadta[31:0]为 0000_0001，因此，完整性检查器仍然对未纠错的数据进行哈希运算得到哈希计算值 cur_digest[255:0]，与哈希参考值 ref_digest[255:0]比对结果不一致，因此产生中断.

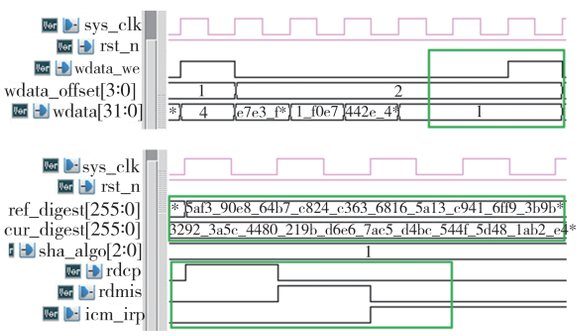

图 16　2 bits 错误完整性检测

Fig.16　Integrity check 2 bits error

4.3　多比特错误仿真

如前文所述，存在 ECC 电路将多比特错误误认成单比特错误和数据正确的情况. 本文以 3 bits 错误为例进行仿真，验证该情况下本文所研究的设计依然能正确检错. 将区域 0 地址 1d82 的数据 07_0000_0008 通过后门访问方式强制转换为 07_0000_0011，以模拟 SRAM 内部发生 3 bits 数据翻转情况. ECC 电路仿真如图 17 所示.

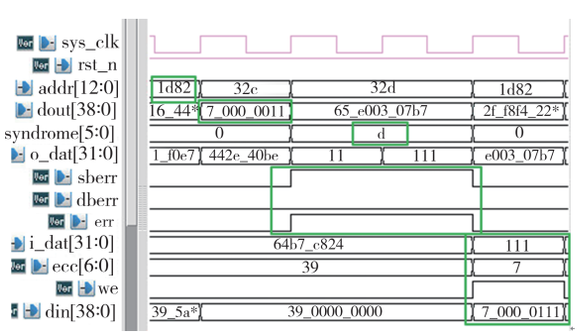

图 17　ECC 检查 3 bits 错误

Fig.17　ECC checks for 3 bits errors

当完整性检测器使能循环访问 SRAM 区域 0 的数据时,ECC 电路访问到地址 1d82 的数据，SRAM 输出 dout[38:0]为 7_0000_0011,计算出的校正子等于 d,对应单比特错误位置为第 8 位,将 3 bits 错误错认为是“1 bit 错误”，置起 sberr 和 err 标志. 对“1 bit 错误”进行纠正后的数据 o_dat[31:0]为 0000_0111,传送给 ECC 电路编码块重新编码，i_dat[38:0]和 ecc

[6:0]组成 din[38:0]等于 7_0000_0111，写入地址 1d82 中.

此时，ECC 电路已执行了错误的纠正过程. 在本文设计中，完整性检测器仍然会对区域 0 内存数据进行检测，其结果如图 18 所示.

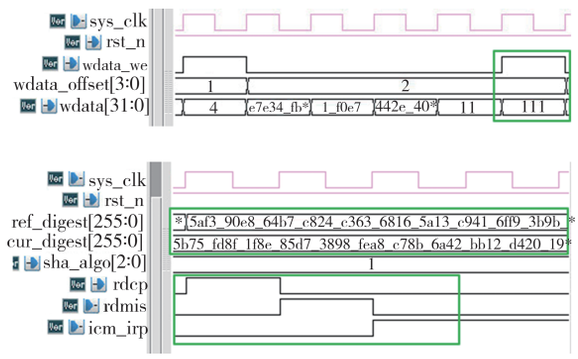

图 18 3 bits 错误完整性检测

Fig.18 Integrity check 3 bits error

图中 icm_irq 上拉，产生中断. 其原因为：当完整性检查检测该内存区域时，ECC 电路无法对 3 bits 错误进行检错纠错，地址 1d82 译码的数据通过总线传输到完整性检查模块的 wadta[31:0]为 0000_0111，因此，完整性检查器仍然对 ECC 电路纠错后的数据进行哈希运算得到哈希计算值 cur_digest[255:0]，与哈希参考值 ref_digest[255:0]比对结果不一致，因此产生中断.

ECC 电路存在将多比特错误识别成正确数据的情况，例如将 07_0000_0008 强制转换为 07_0000_0111 等，但本文方法中均能通过完整性检测器巡检检测出错误，原理与检测 3 bits 错误类似，在此不再赘述.

4.4 软硬件速率仿真

此外，软件层面也可以实现完整性检测器功能，Verderame 等[14]提供了一种基于 PARIOT 固件信息完整性校验方法，并验证了其检测的可行性和鲁棒性，但从软件层面进行检测仍然需要更多在存储和时间方面的开销. 因此，设计了软硬件方法之间的性能对比仿真案例. 软件方案为在硬件 ECC 电路的基础上使用软件实现完整性检测器功能，硬件方案为本文方法. 两种方法均执行了写入数据、完整性检测器计算原始哈希值、引入错误、ECC 电路纠检错、完整性检测器计算新的哈希值、哈希值对比出结果的整个过程.

其时间对比如图 19 所示. 其中，图 19(a)、图 19(c)分别表示在 DLM 和 ILM 寄存器中引入 1 bit 错误的结果. 两者完整性检测器检测结果均为数据正常，原因在于 ECC 电路对 1 bit 错误进行了自动纠错，完整性检测器扫描后，对比哈希值相同. 图 19(b)、图 19(d)

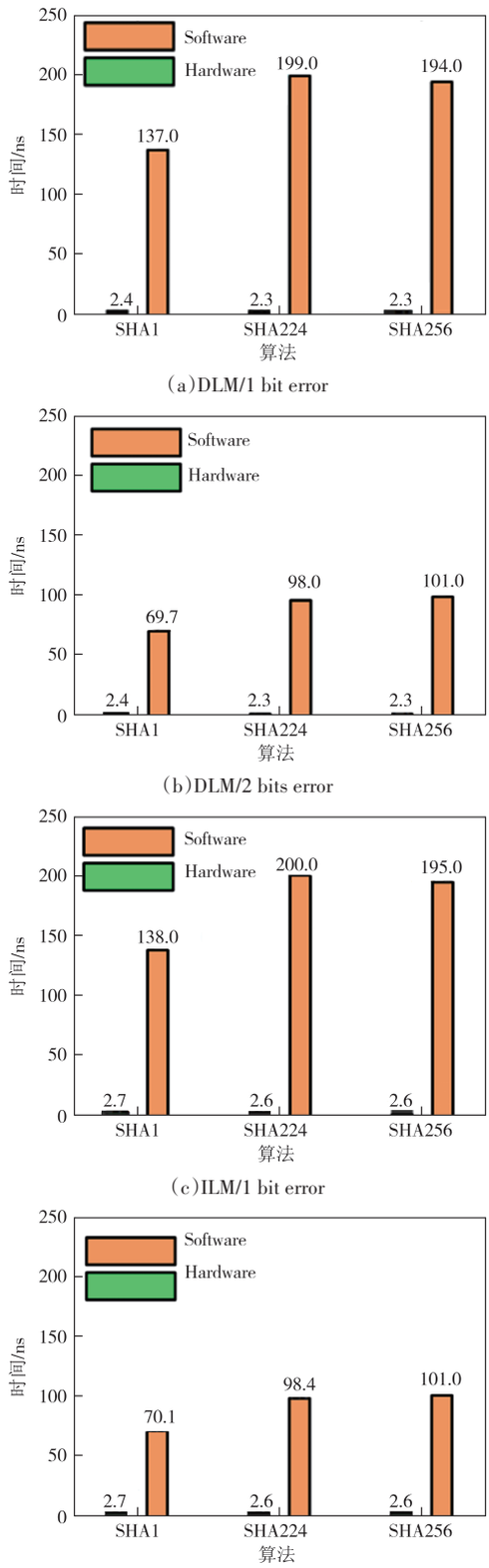

图 19 软硬件速率仿真对比

Fig.19 Hardware and software rate simulation comparison

分别表示在DLM和ILM寄存器中引入2 bits错误的结果.其完整性检测器检测结果均为数据错误,原因在于ECC电路无法对1 bit错误进行自动纠错,完整性检测器扫描后,对比哈希值不同,然后返回错误.从时间上来看,无论是1 bit错误还是2 bits错误的纠检错,硬件方案的时间远低于软件方案,是软件方案的1/50.

5 结论

SEE引起的航天器集成电路数据错误严重影响芯片的功能、性能.本文研究了一种采用(39,32)汉明码与完整性检测器的系统级在线检纠错误设计,在实现对1 bit错误进行自动纠正的同时,周期性地巡检所有数据;在检测出2 bits及以上位数错误后,提供错误信息以便处理器进行中断,防止在使用数据前发现错误需要重新写入的情况.仿真证明,该设计能自动纠正一位错误,检测多位错误,并通过软硬件性能比对,测试了该硬件方案在时间上是软件方案的1/50.

参考文献

[1] 李列文,桂卫华,胡小龙.一种基于FPGA的低功耗、容错状态机设计方法[J].湖南大学学报(自然科学版),2010,37(6):77-82.
LI L W, GUI W H, HU X L. A FPGA-based design method of low power fault-tolerance finite state machine [J]. Journal of Hunan University (Natural Sciences), 2010, 37(6): 77-82. (in Chinese)

[2] 王红敏,董涛,宁生科.超大规模集成电路内部单粒子翻转效应仿真[J].计算机仿真, 2021, 38(8): 277-281.
WANG H M, DONG T, NING S K. Simulation of single event upset effect in VLSI [J]. Computer Simulation, 2021, 38(8): 277-281. (in Chinese)

[3] JUNG J, KO Y, SO H, et al. Root cause analysis of soft-error-induced failures from hardware and software perspectives [J]. Journal of Systems Architecture, 2022, 130:102652.

[4] 吴驰, 毕津顺, 滕瑞, 等. 复杂数字电路中的单粒子效应建模综述[J]. 微电子学, 2016, 46(1): 117-123.
WU C, BI J S, TENG R, et al. Review of modeling for single event effect in complex digital circuits [J]. Microelectronics, 2016, 46(1): 117-123. (in Chinese)

[5] 杨旭,范煜川,范宝峡.龙芯X微处理器抗辐照加固设计[J].中国科学:信息科学, 2015, 45(4): 501-512.
YANG X, FAN Y C, FAN B X. Loongson X-CPU radiation hardening by design [J]. Scientia Sinica (Informationis), 2015, 45(4): 501-512. (in Chinese)

[6] 叶甜春, 李博, 刘凡宇, 等. 一种新型高抗辐照可配置SOI器件技术[J]. 原子能科学技术, 2023, 57(12): 2241-2253.
YE T C, LI B, LIU F Y, et al. A novel configurable SOI technology with extremely high radiation tolerance [J]. Atomic Energy Science and Technology, 2023, 57 (12): 2241-2253. (in Chinese)

[7] 苑靖爽,赵元富,王亮,等.28 nm工艺触发器抗单粒子翻转版图加固技术[J].现代应用物理, 2022, 13(01): 114-119.
YUAN J S, ZHAO Y F, WANG L, et al. Anti-single event upset hardened technology of flip-flops fabricated in 28 nm process [J]. Modern Applied Physics, 2022, 13(1): 114-119. (in Chinese)

[8] 宁亚飞.基于三模冗余和三级错误拦截的四节点翻转容忍锁存器设计[J].河南科技, 2022, 41(16): 19-22.
NING Y F. Design of quadruple node upsets tolerant latch based on the triple-modular-redundancy and triple-level error-interception [J]. Henan Science and Technology, 2022, 41(16): 19-22. (in Chinese)

[9] MARU A, SHINDOU H, EBIHARA T, et al. DICE-based flip-flop with SET pulse discriminator on a 90 nm bulk CMOS process [J]. IEEE Transactions on Nuclear Science, 2010, 57(6): 3602-3608.

[10] YUE H S, WEI X H, TAN J, et al. Eff-ECC: protecting GPGPUs register file with a unified energy-efficient ECC mechanism [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022, 41(7): 2080-2093.

[11] 曾健平,王振宇,袁甲,等.一种改进的SRAM故障内建自检测算法[J].湖南大学学报(自然科学版),2019,46(4):97-101.
ZENG J P, WANG Z Y, YUAN J, et al. An improved SRAM fault built-in-self-test algorithm [J]. Journal of Hunan University (Natural Sciences), 2019, 46(4): 97-101. (in Chinese)

[12] 高文才,陈小文.一种基于混合加固的容软错误NoC路由器[J].计算机工程与科学, 2023, 45(8): 1376-1382.
GAO W C, CHEN X W. A hybrid-hardening soft error tolerant NoC router[J].Computer Engineering & Science , 2023, 45(8): 1376-1382.(in Chinses)

[13] 刘金枝,邹彬,周丹阳.基于交织编码的抗单粒子翻转加固技术研究[J].电子测量技术, 2023, 46(2): 7-13.
LIU J Z, ZOU B, ZHOU D Y. Research on anti-single event upset reinforcement based on interleaving coding [J]. Electronic Measurement Technology, 2023, 46(2): 7-13. (in Chinese)

[14] VERDERAME L, RUGGIA A , MERLO A. PARIOT: Anti-repackaging for IoT firmware integrity [J]. Journal of Network and Computer Applications, 2023, 217: 103699.