中国集成电路 CIC
China Integrated Circuit

S698M SoC芯片中EDAC模块的设计与实现

黄琳[1],陈第虎[1],梁宝玉[2],颜军[2]
(1.中山大学理工学院,广东 广州 510275;
2.欧比特(珠海)软件工程有限公司,广东 珠海 519080)

摘要: EDAC检错纠错模块在电子、通信以及航空航天等领域有着广泛的应用。本文主要介绍了利用[39,32]扩展海明码的EDAC模块的基本原理和用VHDL语言设计实现EDAC的设计实现,该模块在XILINX ISE软件开发环境下通过设计、综合、仿真,验证了设计的正确性。

关键词: 错误检测与校正(EDAC);单粒子翻转(SEU);VHDL;扩展海明码

Design and Realization of Error Detection and Correction Circuit in S698M SoC Processor

HUANG Lin[1], CHEN Dihu[1], LIANG Bao-yu[2], YAN Jun[2]
(1.School of Physics & Engineering, Sun Yat-Sen University, Guangzhou, 510275, China;
2.Orbita Software Engineering Inc. Zhuhai, 519080, China)

Abstract: Error Detection And Correction is widely applied in many fields, such as space navigation, aviation and communication. This paper emphasized on the basic principles and design procedure by VHDL in detail. The implementation of design, simulation and synthesis is under the software of XILINX ISE. A reasonable simulation result is given.

Key words: Error Detection And Correction (EDAC); single event upset (SEU); VHDL;Extended hamming code

1 引言

随着半导体技术的快速发展和其设计工艺的不断改进提高,集成电路的应用领域正在不断扩大和深入,各个行业也对其提出了更高、更新的要求。许多应用场合都要求系统能长期稳定、可靠地运行。尤其在军事、工业、野外作业及航空航天等领域。例如在电磁环境比较恶劣或者存在强辐射的情况下,一些大规模的集成电路常常会受到干扰,导致系统不能正常工作。特别是由辐射引起的单粒子翻转效应SEU(Single Event Upset),会使得RAM这种利用双稳态进行存储的器件的某一位的数据从一个稳态变

化为另一个稳态，即造成存储的“0”或“1”翻转成“1”或“0”的差错。在航空航天等特殊领域，单粒子翻转问题尤为突出，直接影响到系统的正常运行，一旦发生错误会导致严重的后果。

针对这种情况，国内外近年来采用EDAC（Error Detection and Correction，简称为EDAC）电路来避免或减少这种情况地发生，在测试中证明了这是一种行之有效的方法。常见的EDAC电路设计大部分是采用（这些）抗干扰的EDAC专用芯片加上时序控制电路来完成检错纠错的功能，但是这种实现方式不利于系统的集成和小型化，难以满足一些需要高集成度低功耗的场合。本文将讨论基于S698M处理器下，利用VHDL硬件描述语言实现EDAC功能，这种解决方案不仅可以解决传统电路的缺点，也简化了电路，使得整个设计稳定、可靠、集成度更高。

2 S698M简介

目前大多具有EDAC功能的存储器都是在8086处理器控制下进行工作的，而该项目的中央处理器是采用珠海欧比特公司设计研制的S698M，这款32位RISC嵌入式处理器是属于S698系列中的一员。S698系列处理器芯片是珠海欧比特公司与哈尔滨工业大学联合开发的，是继“龙芯”、“方舟”和“众志”之后，又一具有自主知识产权的高端的“中国芯”家族成员。而S698M是在S698基础上设计研制的专用于恶劣环境的32位的SOC芯片。

S698M遵循SPARC V8标准。图1给出了S698M处理器的结构框图，如图1所示，S698M内部配置了32位整型单元（IU）和64位浮点单元（FPU）。采用可裁减的AMBA总线作为片内系统架构总线，片上各模块通过AMBA总线进行数据交换和通信。AMBA总线配置了PCI总线接口、存储器总线接口、UART、定时器、中断管理器、I/O、看门狗、配置寄存器等，大大地提高了S698M的集成度和性能。S698M处理器CPU内部指令实行单指令发射流水线，具备五级流水（Pipeline），分别是取指、译码、执行、存储和回写五个阶段。这样，每个时钟周期就执行一条指令，充分体现了RISC芯片的优势。采用“Harvard”结构，地址总线和数据总线分开，分别连接到独立的“Cache”控制器上。同时，S698M处理器采用先进的时钟配置及管理机制以及低功耗优化设计。S698M处理器具备强实时处理能力，全面支持RERMS、ORION等嵌入式实时操作系统，具有完整的芯片及应用开发系统。

S698M处理器为航空航天、高端工业控制等领域提供了可靠，稳定，高集成度的解决方案，它的研发成功打破了国外同类产品对我国在以上诸多领域的技术垄断。

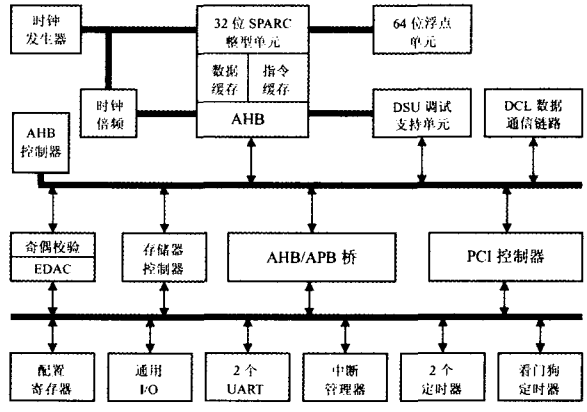

图1 S698M处理器结构图

3 EDAC检错纠错原理及算法

3.1 EDAC检错纠错原理

EDAC电路是采用扩展的海明校验码来完成自动纠正一位错检测任意两位错的功能。海明码是由R.Hamming在1950年提出，由于其简单高效的特性，目前还被广泛应用的一种可以纠正单个错误的线性分组码。它的实现原理是在传输的数据源码中加入一些冗余码，使这些数据源码与冗余码之间根据某种规则建立一定的关系，一旦合法数据编码出现某些错误时，数据源码与检验码之间的关系被破坏，就形成非法编码。而接收端可以通过检测编码的合法性来发现错误直至纠正错误。

常用的海明码算法应该满足不等式：$2^r \geq k + r +$

1,其中 k 为数据码位数,r 为校验码位数。如果要求既能自动检测并纠正一位错,同时能发现两位错,应在原算法的基础上再增加一位校验码,应满足以下关系式:$2^r - 1 \geq k + r$。

3.2 EDAC 编解码算法

以 32 位数据为例,根据纠错编码原理构造一种能对 32 位二进制数据进行纠一检二的编码方式。根据海明码算法,至少需要有 7 位校验位才能达到设计要求。这样就构成了 39 位的新码字。

按照海明码编码规则,应该在数据的第 $2^n - 1$(n=0,1,2,3,4,5)的位置上即第 0、1、3、7、15、31 位上放置校验码,以下是校验位的具体编码方程:

$P_0 = D_0 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{11} \oplus D_{13} \oplus D_{15} \oplus D_{17} \oplus D_{19} \oplus D_{21} \oplus D_{23} \oplus D_{25} \oplus D_{26} \oplus D_{28} \oplus D_{30}$

$P_1 = D_0 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_9 \oplus D_{10} \oplus D_{12} \oplus D_{13} \oplus D_{16} \oplus D_{17} \oplus D_{20} \oplus D_{21} \oplus D_{24} \oplus D_{25} \oplus D_{27} \oplus D_{28} \oplus D_{31}$

$P_2 = D_1 \oplus D_2 \oplus D_3 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25} \oplus D_{29} \oplus D_{30} \oplus D_{31}$

$P_3 = D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25}$

$P_4 = D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25}$

$P_5 = D_{26} \oplus D_{27} \oplus D_{28} \oplus D_{29} \oplus D_{30} \oplus D_{31}$

如果要区分一位错还是两位错还需要增加一位总校验位。另外,它也是奇数位出错和偶数位出错的依据,如果该位为 1 表示奇数位出错;为 0 表示无错或者偶数位出错。编码如下:

$P_6 = D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25} \oplus D_{26} \oplus D_{27} \oplus D_{28} \oplus D_{29} \oplus D_{30} \oplus D_{31} \oplus P_0 \oplus P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus P_5$

而解码过程就是利用已生成的校验码和编码过的数据码再进行一次编码,即用一个校验码和形成这个校验码的编码方程再执行一次异或,也就是又一次执行偶校验。如果解码时的校验码和编码时计算的结果一致,说明数据码没有发生错误。否则数据码有错。根据生成的伴随式 S_i($S_0S_1S_2S_3S_4S_5S_6$)结果,可以完成检查数据一位错或两位错的设计要求。以下是伴随式 S_i 的编码方程:

$S_0 = P_0 = D_0 \oplus D_1 \oplus D_3 \oplus D_4 \oplus D_6 \oplus D_8 \oplus D_{10} \oplus D_{11} \oplus D_{13} \oplus D_{15} \oplus D_{17} \oplus D_{19} \oplus D_{21} \oplus D_{23} \oplus D_{25} \oplus D_{26} \oplus D_{28} \oplus D_{30}$

$S_1 = P_1 = D_0 \oplus D_2 \oplus D_3 \oplus D_5 \oplus D_6 \oplus D_9 \oplus D_{10} \oplus D_{12} \oplus D_{13} \oplus D_{16} \oplus D_{17} \oplus D_{20} \oplus D_{21} \oplus D_{24} \oplus D_{25} \oplus D_{27} \oplus D_{28} \oplus D_{31}$

$S_2 = P_2 = D_1 \oplus D_2 \oplus D_3 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25} \oplus D_{29} \oplus D_{30} \oplus D_{31}$

$S_3 = P_3 = D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25}$

$S_4 = P_4 = D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25}$

$S5 = P_5 = D_{26} \oplus D_{27} \oplus D_{28} \oplus D_{29} \oplus D_{30} \oplus D_{31}$

$S_6 = P_0 \oplus P_1 \oplus P_2 \oplus P_3 \oplus P_4 \oplus P_5 \oplus P_6 \oplus D_0 \oplus D_1 \oplus D_2 \oplus D_3 \oplus D_4 \oplus D_5 \oplus D_6 \oplus D_7 \oplus D_8 \oplus D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus D_{13} \oplus D_{14} \oplus D_{15} \oplus D_{16} \oplus D_{17} \oplus D_{18} \oplus D_{19} \oplus D_{20} \oplus D_{21} \oplus D_{22} \oplus D_{23} \oplus D_{24} \oplus D_{25} \oplus D_{26} \oplus D_{27} \oplus D_{28} \oplus D_{29} \oplus D_{30} \oplus D_{31}$

根据伴随式 S_i 进行判断,有以下几种情况:

1)如果 $S_6 \sim S_0 = 0000000$,说明代码无错,数据位和校验位都正确;

2)如果单位数据码出错,必然有三位或三位以上的 S_i 为 1;

例如:$S_6 \sim S_0 = 1000011$,表示一位出错,出错发生在数据位 D_0,把该数据位取反纠正得到正确编码输出

$S_6 \sim S_0 = 1000101$,表示数据位 D_1 出错,该位取反得到正确编码输出

……

$S_6 \sim S_0 = 1100110$,表示数据位 D_{31} 出错,该位取反得到正确编码输出

3)如果单位校验位出错,其对应的 S_i 一定为 1,并且 S_6 一定为 1,其余的 S_i 为 0;

例如:$S_6 \sim S_0 = 1000001$,表示校验位 P_0 出错,发送错误标志,数据直接输出

$S_6 \sim S_0 = 1000010$,表示校验位 P_1 出错,发送错误标志,数据直接输出

……

S_6~S_0 = 1000000，表示校验位 P_6 出错，发送错误标志，数据直接输出

4）如果任意两位错误（包含数据位、校验位），S_6 一定为 0，其余 S_i 不全为 0，此时，根据编码已无法确认具体出错的位置，因此无法纠正错误，应发送中断标志至 CPU，由 CPU 做出相应处理；

5）其他情况超出编码的纠错能力，按照不能纠错情况处理。

4 基于 VHDL 的 EDAC 模块的设计实现

本设计与以往的设计的最大区别是，没有采用传统的检错芯片，而是用 SoC 的方式，利用 VHDL 语言实现 EDAC 电路的功能。现在多数 EDAC 电路是采用检错芯片和通用的中小规模集成电路来完成检错纠错的功能，这种解决方案虽然较为简单，但是会占用很大的电路板空间，同时，由中小规模 IC 组成的 EDAC 电路还存在着功耗和速度等问题。而利用 VHDL 语言实现的 EDAC 电路不但大大地简化了电路，也能更快速地实现检错纠错功能，提高系统的整体性能。

4.1 设计思想

EDAC 模块位于 CPU 与 RAM 之间，对 RAM 中的数据进行保护。CPU 对 RAM 进行读写时，先经过 EDAC 模块对数据进行处理。在设计中也可加入 EDAC 功能使能信号，即在系统复位时，对寄存器相应的功能选择位写入 1 或 0 即可。当 EDAC 关闭，CPU 直接对 RAM 写数据，读数据时也无需经过 EDAC 纠错。RAM 中的数据可直接读入 CPU。CPU 的片选读写等控制信号一样对 EDAC 有效。EDAC 开启时主要包括以下两个工作模式。

写周期：在 CPU 向 RAM 写入数据时，EDAC 根据算法对写入寄存器的数据位 D_{31}~D_0 进行编码产生校验位 P_6~P_0，然后数据位和产生的校验位一起写入存储器。

读周期：在 CPU 从 RAM 读出数据时，由 EDAC 从存储器中读出数据位和校验位，利用从 RAM 中读出的数据位生成新的校验位，该电路与写周期时校验位产生电路一样。但是如果从寄存器读出的码字有误，则产生的新校验位与原校验不同，$C_r \neq C''_r$；如果码字无误，新校验位与原校验位一致，即 $C_r = C''_r$。新校验位与写周期产生的校验位进行异或运算生成伴随式 S_i，然后根据伴随式的不同组合状态判断数据是否出错，数据送入锁存器保证在 CPU 读取 RAM 时不发生抖动保持稳定。如果发现 1 位错，输出单位错标志，并送入纠错单元自动对错位取反纠正错误，然后把校正后的数据位输出；如果发现 2 位错，产生中断标志送至 CPU；如果没有错误，数据直接从锁存器输出。

EDAC 电路功能框图如图 2 所示。

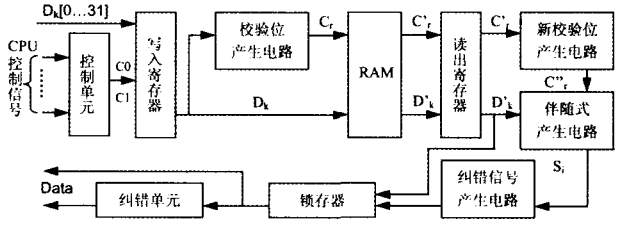

图 2 EDAC 电路框图

4.2 系统描述和功能仿真

根据 EDAC 模块的特点，在用 VHDL 语言进行系统描述时，主要分为两个进程描述，一个进程用来完成 EDAC 工作时序控制，控制信号对 EDAC 在读写操作进行控制，该进程是时序电路。另一个进程用以描述 EDAC 算法，完成 EDAC 对数据源码的编解码过程，是组合电路。两种电路描述分别在不同的进程，可以节省资源。

为了检查该设计在存储器发生错误时能否检测纠正输出正确数据，在测试过程中定义了两个测试信号 Dbin_test 和 Cbin_test，强制将有误差的数据位和校验位写入到存储器中，进行测试。

在 Xilinx ISE 8.1i 平台上对 VHDL 源码进行综合，使用 ModelSim SE 6.1b 编写测试程序产生输入

图 3 EDAC 仿真波形图

激励，对 EDAC 设计源码进行功能仿真，验证设计的正确性。仿真波形图如图 3 所示。

其中仿真电路信号分别为：

Clk（输入）：时钟信号；

Ramsn（输入）：片选信号；

Testmode（输入）：测试使能；

Oen（输入）：读信号；

Writen（输入）：写信号；

Dbin_test [31…0]（输入）：写入数据位；

Cbin_test [6…0]（输入）：写入校验位；

Dbout [31…0]（输出）：校验后的写出数据位；

Cbout [6…0]（输出）：伴随式输出；

SingleErr（输出）：1-bit 错误标志；

DoubleErr（输出）：2-bit 以上错误标志。

如图 3 所示，Clk、Ramsn、Oen 等控制信号由 S698M 给出。在 A 时刻 Ramsn 低有效，Writen 为低，CPU 向存储器写入数据。B、C、D、E、F 时刻分别是 CPU 向存储器读出数据的过程，Ramsn、Writen 低有效。在实际仿真过程中进行了多种状况测试，这里只截选了其中的一小部分过程进行说明。B 时刻数据位和校验位正确，无错。C 时刻数据位 Dbin31 有错，单错标志 SingleErr 为 1，Dbout 输出修改后的正确数据位；D 时刻数据位无错，校验位 Cbin0 有错，虽然此时单错标志 SingleErr 为 1，但是数据无需修改直接输出；E 时刻数据位 Dbin0 和 Dbin1 两位出错，SingleErr 和 DoubleErr 同时置 1，此时 Dbout 不能输出正确数据；F 时刻 SingleErr 和 DoubleErr 同时置 1，虽然此时数据只有一位数据位 Dbin0 出错，但是 EDAC 不仅检测数据位，校验位也被检测，此时校验位 Cbin6 出错，双错状态 Dbout 不能输出正确数据。通过上述仿真结果分析可以反映电路的检错纠错能力，证明设计完成了预定的功能要求。

5 结束语

本文主要研究了 EDAC 电路的原理、算法和实现，并且针对 S698M 处理器的体系结构特点，设计了能达到纠正 32 位数据中的一位错误和检测任意两位错误的 EDAC 模块。并用 ModelSim 工具对其进行模拟仿真，基本证明了设计的正确性。而且硬件实现不需要大量资源，也比较灵活地嵌入到 FPGA/CPLD 实现。该模块实现纠正单粒子翻转的目的，有效地降低了 SEU 对存储器造成的影响，保证了数据和系统的稳定和可靠。另外采用 VHDL 语言实现 EDAC 功能提高了设计的灵活性，可以推广到需要对存储系统进行保护的不同嵌入式计系统的设计中。CIC

参考文献

[1] Orbita Software Engineering Inc. 32-bit SPARC V8 Embedded Processor Sailing S698M User's Manual, www.myorbita.net, 2004

[2] 王爱英. 计算机组成与结构. 清华大学出版社，2001

[3] 刘富全. 纠错编码技术及应用. 哈尔滨船舶工程学院出版社，1993

[4] 周盛雨. 基于 386EX CPU 的实时 EDAC 设计. 硕士学位论文，2003

[5] 李飞，张志敏，王岩飞. 错误检测与纠正电路的设计与实现. 单片机与嵌入式系统应用，2003，2：34-39

[6] 刘淑芬，崔星. 计算机 RAM 检错纠错电路的设计与实现. 航天控制，2003，4：59-67

[7] 边计年，薛宏熙. 用 VHDL 设计电子线路. 清华大学出版社，2000