



一种基于扩展汉明码的动态纠错机制

陈天培^{1,2}, 吴校生¹, 强小燕²

(1. 上海交通大学微米/纳米加工技术国家级重点实验室, 上海 200240; 2. 中科芯集成电路有限公司, 江苏 无锡 214072)

摘要: 错误纠正码 (Error Correction Code, ECC) 是解决存储器数据出现一位或两位错误的有效手段, 然而过于复杂的编码方式将降低读写性能。为解决该问题, 提出了一种基于扩展汉明码的动态纠错机制。利用扩展汉明码的奇偶校验位对检错模块进行了优化, 能够监测错误发生的频率, 并可动态切换进行扩展汉明码校验与奇偶校验。运用 Verilog 硬件描述语言实现了该机制并进行了仿真分析。分析结果表明, 使用该机制与使用扩展汉明码校验相比, 能够有效降低路径延时与动态功耗。

关键词: 错误纠正码; 动态纠错机制; 扩展汉明码; 路径延时; 动态功耗

中图分类号: TN402 **文献标志码:** A **文章编号:** 1681-1070 (2022) 05-050304

DOI: 10.16257/j.cnki.1681-1070.2022.0507

中文引用格式: 陈天培, 吴校生, 强小燕. 一种基于扩展汉明码的动态纠错机制[J]. 电子与封装, 2022, 22 (5): 050304.

英文引用格式: CHEN Tianpei, WU Xiaosheng, QIANG Xiaoyan. A dynamic error correction mechanism based on extended Hamming code[J]. Electronics & Packaging, 2022, 22(5): 050304.

A Dynamic Error Correction Mechanism Based on Extended Hamming Code

CHEN Tianpei^{1,2}, WU Xiaosheng¹, QIANG Xiaoyan²

(1. National Key Laboratory of Science and Technology on Micro/Nano Fabrication, Shanghai Jiao Tong University, Shanghai 200240, China; 2. China Key System & Integrated Circuit Co., Ltd., Wuxi 214072, China)

Abstract: Error correction codes (ECC) are effective means to deal with single-bit or double-bit errors in data. However, an overly complex encoding method will reduce read and write performance. To solve this problem, a dynamic error correction mechanism based on the extended Hamming code is proposed. The parity bit included in the extended Hamming code is used to optimize the error detection module, and it can perform extended Hamming code and parity check dynamically. The hardware description language Verilog is used to realize the mechanism, and the simulation is carried out. The analysis results show that using this mechanism can effectively reduce dynamic power consumption compared with using extended Hamming code verification.

Keywords: error correction codes; dynamic error correction mechanism; extended Hamming code; path delay; dynamic power

收稿日期: 2021-09-22

E-mail: 陈天培 stevensense@foxmail.com; 强小燕 (通信作者) 24784372@qq.com

1 引言

存储器作为储存大量数据的电路元件,对数据可靠性的要求很高。提升存储器可靠性的一种常见手段是通过增加冗余硬件设计来对工作电路进行保护^[1]。给数据增加纠错码(Error Correcting Code, ECC)或检错码(Error Detecting Code, EDC)是冗余硬件设计的一种思路。常见的ECC或EDC包括奇偶校验码、CRC校验、汉明码、BCH码、RS码^[2-3]等。各种码的复杂度不同,纠错能力也不尽相同。其中,汉明码最为常见。汉明码是一种线性分组码^[4],能够实现一位比特纠错。在电路中,一位错误出现的概率最大^[5],因此汉明码得到了广泛的应用。

尽管ECC码是一种有效的增加存储器可靠性的保护措施,但同时也带来了一些负面影响。增加冗余信息码会带来额外的面积开销、编解码带来的功耗提升以及较长的组合逻辑时延。如何进行优化,研究人员对其进行了深入的探索,并得到了大量的研究成果。文献[6]提出了两种基于汉明码的新的编码机制SEC-DAED以及SEC-DED-TAED,在不增加校验位位数的情况下,前者能够实现一位纠错、连续两位检错,后者能够实现一位纠错、两位检错以及连续三位检错。文献[7]进一步提出了一种可变范围的连续位纠错的编码机制,同时不增加同等情况下汉明码的校验位数。文献[8-9]则研究了对纠错码编解码模块的优化。本文基于扩展汉明码这一编码机制进行了深入的研究,提出了一种动态纠错机制。

2 扩展汉明码原理与设计

2.1 扩展汉明码原理

汉明码的基本思想是将待检验的数据码元进行分组,再分别对各组码元进行编码生产对应校验码。若是某一个数据位出现错误,则该数据位所属的组校验码将无法与当前数据对应,即发生错误。通过计算出校验码无法对应的分组来确定最终出错的数据位。

汉明码能够实现一位检错与纠错。以 n 位数据为例,若附加的冗余校验位位数为 r ,对于校验位来说,需要唯一表示任何一位的出错情况以及数据正确的情况,因此 n 、 r 两者的关系由式(1)约束:

$$2^r \geq n + r + 1 \quad (1)$$

当数据位位数为32位时,则所需要的冗余位为6

位。基于式(1),每当数据位位数增加一倍,所需的冗余位位数增加一位。

普通汉明码只能检测并修正一位错误。通过再增加一位奇偶校验码,可以实现两位错误的识别,该码被称为扩展汉明码^[4]。

2.2 校验码生成算法

本文进行校验的数据位宽为32位。扩展汉明码的编码方式如式(2)~(8)所示。其中 b_n 为校验位 n , a_n 为数据位 n 。可以看到每位校验位的生成是对特定的数据位异或运算后生成的。而增加的一位奇偶校验位则是对所有的数据位进行异或运算。

$$b_0 = a_0 \oplus a_1 \oplus a_3 \oplus a_4 \oplus a_8 \oplus a_{10} \oplus a_{11} \oplus a_{13} \oplus a_{15} \oplus a_{17} \oplus a_{19} \oplus a_{21} \oplus a_{23} \oplus a_{25} \oplus a_{26} \oplus a_{28} \oplus a_{30} \quad (2)$$

$$b_1 = a_0 \oplus a_2 \oplus a_3 \oplus a_5 \oplus a_8 \oplus a_{10} \oplus a_{12} \oplus a_{13} \oplus a_{16} \oplus a_{17} \oplus a_{20} \oplus a_{21} \oplus a_{24} \oplus a_{25} \oplus a_{27} \oplus a_{28} \oplus a_{31} \quad (3)$$

$$b_2 = a_1 \oplus a_2 \oplus a_3 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{14} \oplus a_{15} \oplus a_{16} \oplus a_{17} \oplus a_{22} \oplus a_{23} \oplus a_{24} \oplus a_{25} \oplus a_{29} \oplus a_{30} \oplus a_{31} \quad (4)$$

$$b_3 = a_4 \oplus a_5 \oplus a_6 \oplus a_7 \oplus a_8 \oplus a_9 \oplus a_{10} \oplus a_{18} \oplus a_{19} \oplus a_{20} \oplus a_{21} \oplus a_{22} \oplus a_{23} \oplus a_{24} \oplus a_{25} \quad (5)$$

$$b_4 = a_{11} \oplus a_{12} \oplus a_{13} \oplus a_{14} \oplus a_{15} \oplus a_{16} \oplus a_{17} \oplus a_{18} \oplus a_{19} \oplus a_{20} \oplus a_{21} \oplus a_{22} \oplus a_{23} \oplus a_{24} \oplus a_{25} \quad (6)$$

$$b_5 = a_{26} \oplus a_{27} \oplus a_{28} \oplus a_{29} \oplus a_{30} \oplus a_{31} \quad (7)$$

$$b_6 = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus \dots \oplus a_{29} \oplus a_{30} \oplus a_{31} \quad (8)$$

基于上述算法生成的校验位不仅可以对数据进行一位纠错,同时可以对校验位本身进行一位纠错。其一位错误位由以下流程确定:进行ECC校验时,将储存的校验码以及相应数据发至检查模块,对数据重新生成校验码,将原校验码与新生成的校验码进行比较(异或),产生7位的错误编码;再将错误编码送至译码模块,确定错误位置。对于一位错误来说,错误编码第7位必然为1。而对于两位错误,错误编码的第7位必然为0,此时若剩余6位错误编码不为0时,则发生了两位或以上的错误。一位错误及两位错误检测如图1所示,其中syndrome信号表示错误编码,single_err信号表示一位错误,double_err信号表示两位错误。

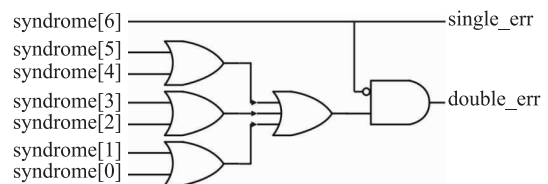


图1 一位错误及两位错误检测

2.3 ECC纠错模块设计

按照以上算法,本文基于Verilog语言进行了

ECC 纠错模块的设计。ECC 纠错模块整体分为 3 个子模块: 校验码生成模块、检查模块和修复模块。当数据要被送入到存储器前, 通过校验码生成模块生成校验码, 并与数据一起送入存储器中。当需要取用数据时, 则读出数据到检查模块, 利用生成模块生成新校验码后并得到错误编码, 根据错误编码产生一位以及两位错误信号。如若有一位错误, 则进一步送至修复模块对数据进行修正, ECC 纠错模块如图 2 所示。

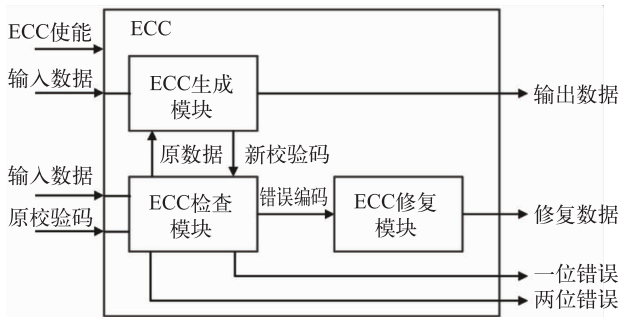


图 2 ECC 纠错模块

3 动态纠错机制原理与设计

3.1 动态纠错机制原理简述

由上述算法可以看出, 与奇偶校验相比, 汉明码校验各校验位生成与检查均需进行大量的异或计算。在具体实现上引入了较长的组合逻辑, 带来了更长的延时。同时进行数据编解码时, 逻辑门频繁的翻转也导致了更大的动态功耗。尽管奇偶校验在功耗与延时上都优于汉明码校验, 但奇偶校验本身只能检测一位错误, 而不能纠正错误。扩展汉明码则能够纠正一位错误, 检测两位错误。为了尽可能利用两种校验方式的优势, 本文提出了一种动态纠错机制, 在保证汉明码纠错强度的同时, 降低纠错过程的延时与动态功耗。

其基本原理是在检查阶段, 利用奇偶校验码对所有数据进行校验。而在纠错阶段, 则采用汉明码纠正错误。注意到扩展汉明码最高位本身是所有数据的奇偶校验位, 因此对数据进行奇偶检验时, 只需对原始校验码最高位进行计算并检查。这就使得检查的逻辑延时缩短。而其他比较逻辑在奇偶校验期间不会工作, 降低了检查模块的动态功耗。

存储器复位之后, 开启奇偶校验。当第一次一位错误出现时, 动态纠错模块将在下一个时钟周期对同一数据进行汉明码校验, 以修正错误。同时开启读请求计数, 若在设定的次数内不再发生一位错误, 则可认为该一位错误是偶发错误, 纠错模块将继续用奇偶

校验对数据进行检查。若在设定的次数内再次发生一位错误, 则可以认为工作环境发生了变化, 纠错模块将持续开启汉明码校验。在持续开启汉明码校验的阶段, 每次新发生的一位错误将重置读请求计数。若在读请求计数达到设定值前都未发生错误, 动态纠错模块将重新切换至奇偶校验。图 3 为动态纠错机制的基本控制流程图。

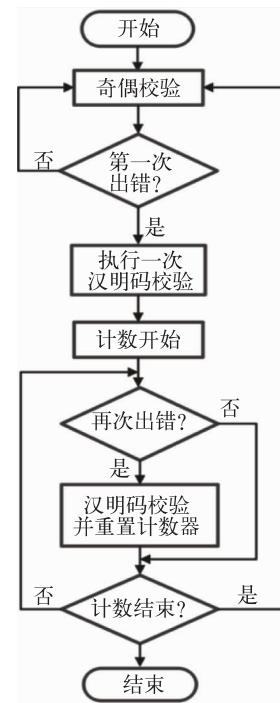


图 3 动态纠错机制控制流程

3.2 动态纠错机制模块设计

动态纠错模块由一个 ECC 监控模块以及 ECC 纠错模块组成。ECC 纠错模块的一位错误输出信号将被送至监控模块。监控模块则基于上文所述逻辑对 ECC 纠错模块进行控制。监控模块自身采用一个有限状态机实现状态转移。图 4 和 5 分别为动态纠错模块功能框图以及状态转移图。状态转移图中, fe (first error) 指示第一次出现错误, pe (parity error) 指示奇偶校验错误, ce (count end) 指示计数完毕。该状态机所有状态位均为高有效。

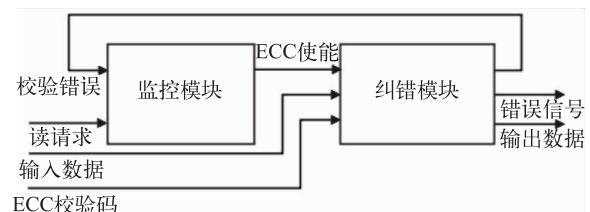


图 4 动态纠错模块功能框图

模块启动后, 状态机处于 idle (空闲) 状态, 并开启奇偶校验。当检测到奇偶校验错误时, pe 置 1, 此时状

态机切换至 first_cnt (第一次计数) 状态, 模块将进行一次纠错。pe 在计数时置 0, fe 置 1。在计数过程中, 若再次发生错误, pe 置 1, 状态机切换到 second_cnt (第二次计数) 状态。若未发生错误, 则计数完毕后 ce 置 1, 状态机回到 idle 状态。在 second_cnt 状态下, 动态纠错机制模块将持续开启汉明码校验。若在第二次计数过程中发生错误, 则重置计数器。若不发生错误, 状态机返回 idle 状态。

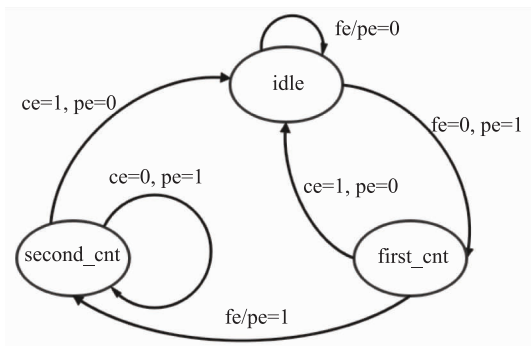


图5 动态纠错模块状态机

4 功能验证

基于上述模块搭建仿真环境, 并进行验证。动态纠错机制的主要功能点包括两个部分, 第一个部分为底层 ECC 纠错逻辑; 第二个部分为奇偶校验与扩展汉明码校验间的切换。

4.1 ECC 纠错模块的验证

ECC 纠错模块的验证分为一位数据的纠错以及二位数据的检错。为测试所有出错的可能性, 依次翻转原始数据的每一位, 并送至纠错模块。图 6 和 7 分别为一位错误情况和两位错误情况。信号 data_in[31:0] 为原始数据, data_to_ip[31:0] 为正确数据, data_from_ip[38:0] 中低 32 位为翻转后的数据, 高 7 位则为校验码。data_correct_out[31:0] 为修正后的数据。r 则指示一位错误, fatal_error 指示两位错误。当发生一位错误时, singlebit_error 置 1, 同时纠错模块将修正后的数据送出。当发生两位错误时, fatal_error 置 1, 同时纠错模块将原始数据送出。

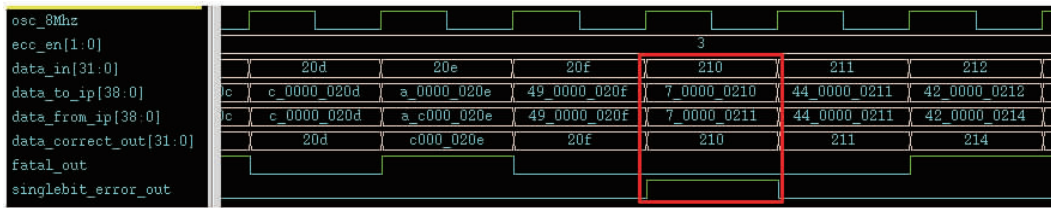


图6 一位错误

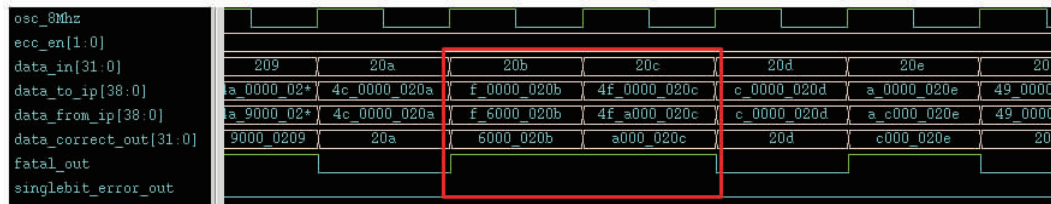


图7 两位错误

4.2 奇偶校验与扩展汉明码校验切换

根据上文所述流程, 当第一次出现错误时, 动态纠错机制将执行一次汉明码纠错。当一段时间内第二次出现错误时, 动态纠错机制将持续开启汉明码纠错。奇偶校验与汉明码校验切换如图 8 所示, 当指示奇偶校验错误的信号 parity_err 第一次出现且读请求 rd_req 拉高时, 使能信号 ecc_en 置 1, 汉明码纠错执行一次。同时计数器开始计数, 当计数器达到设定值而未出现第二次错误时, 计数器停止计数, first_err 置 0, 等待下一次奇偶校验错误。当计数器未到达设定值而出现第二次错误时, second_err 信号置 1, 同时计数器置 0, 汉明码纠错常开。当第三次错误出现时, 计数器

再次归 0, 重新计数。若后续再无错误出现且计数器达到设定值, first_err 以及 second_err 置 0, 动态纠错机制重新回到奇偶校验状态并等待下一次错误信号。图 8 中还验证了当计数器正好达到设定值时出现错误的情况, 可以看到, 设计的模块能够正常工作。

5 综合结果与分析

使用 Synopsys 公司的综合工具 Design Compiler 在中芯国际 (SMIC) 55 nm 工艺库下对设计进行综合。表 1 列出了采用扩展汉明码校验与采用动态纠错机制的关键路径延时与动态功耗。从表中可以看出, 动态

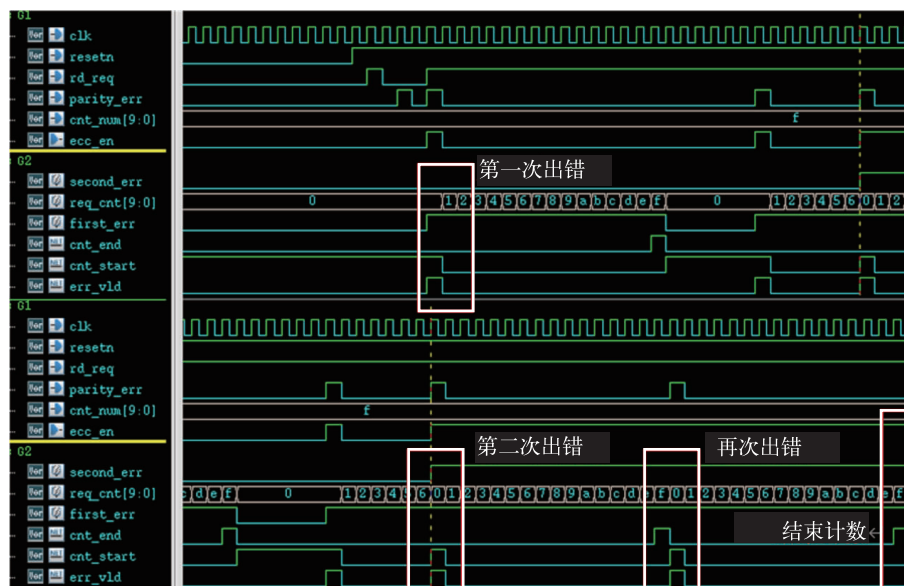


图 8 奇偶校验与汉明码校验切换

纠错机制在延时上降低了 7.49%，在动态功耗上则降低了 22.09%。延时上的降低是因为动态纠错机制将扩展汉明码校验中的奇偶校验与汉明码校验分开到两个时钟周期进行。而动态功耗上的降低则是因为动态纠错机制仅在 ECC 使能时才开启汉明码校验。

表 1 综合结果

校验类型	关键路径延时 /ns	动态功耗 / μ W
扩展汉明码	2.27	304.58
动态纠错机制	2.10	237.29

6 结论

本文提出了一种基于扩展汉明码的动态纠错机制,能够有效提高 ECC 校验检错阶段的效率,减少无效检错。本文主要对扩展汉明码的原理进行了阐述,并设计了基于扩展汉明码的 ECC 纠错模块。同时进一步阐明了动态纠错机制的原理,完成模块设计,最后对设计进行了仿真与分析。综合结果表明,相比传统扩展汉明码校验,该机制在延时与动态功耗上有明显降低。

参考文献:

- [1] ZHANG W. Replication cache: A small fully associative cache to improve data cache reliability[J]. IEEE Transactions on Computers, 2005, 54(12): 1547-1555.
- [2] 朴英琿. 基于 ECC 电路的 SRAM 自检测修复设计与验证[D]. 哈尔滨: 哈尔滨工业大学, 2019.
- [3] 王甲峰, 蒋鸿宇, 胡茂海, 等. RS 码的校验和识别方法

[J]. 太赫兹科学与电子信息学报, 2021, 19(1): 31-37.

- [4] HAMMING R W. Error detecting and error correcting codes [J]. Bell syst Tech, 1950, 26(2): 147-160.
- [5] 丁义刚. 空间辐射环境单粒子效应研究[J]. 航天器环境工程, 2007(5): 283-290, 5-6.
- [6] ALFONSO S M, PEDRO R, JUAN A M. Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement[J]. IEEE Transactions on Device and Materials Reliability, 2014, 14(1): 574-576.
- [7] ADAM N, MANOJ S. A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory[J]. IEEE Transactions on Device and Materials Reliability, 2013, 13(1): 223-230.
- [8] PEDRO R, JORGE M, SALVATORE P, et al. A method to design SEC-DED-DAEC codes with optimized decoding[J]. IEEE Transactions on Device and Materials Reliability, 2014, 14(3): 884-889.
- [9] LIU S S, LI J Q, REVIRIEGO P, et al. A double error correction code for 32-bit data words with efficient decoding [J]. IEEE Transactions on Device and Materials Reliability, 2018, 18(1): 125-127.



作者简介:

陈天培 (1998—), 男, 江苏泰州人, 硕士研究生, 现从事数字 IC 设计工作。