Computer Science and Application 计算机科学与应用, 2015, 5, 179-185
Published Online May 2015 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2015.55023

Hans汉斯

A Method of Recovering DSP Soft Error Based on Integrity Check

Guochang Zhou[1], Xiangtao Wang[2], Xiaoling Lai[1], Xiang Gao[1], Hao Wu[2], Dengyun Yu[3]

[1]Xi'an Institute of Space Radio Technology (CAST Xi'an), China Aerospace Science and Technology Corporation, Xi'an Shaanxi
[2]School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an Shaanxi
[3]The Science and Technology Committee, China Aerospace Science and Technology Corporation, Beijing
Email: zhouguochang2000@163.com

Received: May 8th, 2015; accepted: May 23rd, 2015; published: May 28th, 2015

Abstract

In aerospace, the single particle soft errors are more and more frequently reported in DSP and the other memory devices, which seriously impact on the system running safely and reliably. Aiming at the soft error occurred in DSP program storage areas, this paper presents an integrality-checking-based control-flow error recovery method, which can rapidly recovery a control-flow error after checking it by a little improvement of setting a recovery pointer and a recovery memory on the integrality-checking-based control-flow check method. The proposed method has an important guiding significance to the design and development of DSP.

Keywords

DSP, Soft Error, Control-Flow Error, Integrality Checking, Error Recovery

一种基于完整性检查的DSP软错误恢复方法

周国昌[1]，王向涛[2]，赖晓玲[1]，高　翔[1]，吴　昊[2]，于登云[3]

[1]中国空间技术研究院西安分院，陕西 西安
[2]西北工业大学计算机学院，陕西 西安
[3]中国航天科技集团公司科技委，北京
Email: zhouguochang2000@163.com

收稿日期：2015年5月8日；录用日期：2015年5月23日；发布日期：2015年5月28日

摘　要

空间环境中，DSP等器件越来越频繁地发生单粒子软错误，严重影响了系统安全可靠地运行。针对DSP程序存储区的软错误，本文提出了一种基于完整性检查的控制流错误恢复方法。该方法在基于完整性检查的控制流错误检测方法的基础上，只需设置一个恢复指针和恢复存储区，即可在检测到控制流错误之后快速地恢复控制流错误。论文提出的方法对DSP的抗软错误加固设计与开发具有重要的指导意义。

关键词

DSP，软错误，控制流错误，完整性检查，错误恢复

1. 引言

高性能 DSP 常用作空间飞行器信号处理系统的核心器件。当前，在深亚微米工艺下，集成电路的集成度不断增加、工艺尺寸越来越小、电路节点的临界电荷大大降低，使得 DSP 在空间环境中更容易发生单粒子软错误[1]。软错误并没有对 DSP 造成永久性的损伤，但可能造成存储的程序或数据发生改变，影响程序正确的执行流程，导致输出错误[2]。因此，必须采取相应的加固、容错或检错纠错等措施，增强其抗软错误能力。

DSP 是一种典型的商用现成品(COTS，commercial-off-the-shelf) [3]。COTS 产品的体系结构在形成后就不能更改，这使得在 DSP 上不能应用一些传统的集成电路加固方法，如采用抗辐照材料对敏感器件进行屏蔽加固或进行模块冗余等[4]。因此，基于 DSP 的空间信号处理系统的容错方案多采用控制流错误检测和恢复方法。控制流错误是指程序偏离正确的执行流程，即在没有出现跳转指令的地方发生了跳转，在出现跳转指令的地方却不跳转或发生错误跳转[5]。

当前，控制流错误检测主要是采取签名监测的思想。即先将程序划分为若干个基本块的集合，在编译时为每个基本块分派一个静态签名，程序运行时再计算出基本块的动态签名，并通过比较静态签名与动态签名的一致性来判断程序是否发生控制流错误[6]-[12]。R. Vemu 等人首次提出了控制流错误恢复算法及其实现，它本质上是在原程序中添加一个例程，该例程能够恢复原程序代码间的非法分支转移[13]。谭兰芳等人提出了一种类似的恢复算法 CFEDR [14] [15]，该方法能检测并恢复所有的基本块间的控制流错误并恢复绝大部分基本块内的控制流量错误。Hamid R. Zaradi 等人提出了 CDCC 和 MCP [16]两种控制流恢复算法。CDCC 主要根据数据依赖图，为全局变量赋值影子变量，当检测到错误时再将影子变量赋值给对应的全局变量，从而实现状态恢复。MCP 则是通过将控制流恢复至全局变量赋初值的位置来实现错误的恢复。

为此，本文借鉴签名检测，提出了一种基于程序执行流程完整性检查的 DSP 软错误恢复方法 ICBR (Integrity Checking Based Recovery)。该方法通过检查每个基本块的执行完整性来判断程序是否发生控制流错误；当检测出软错误后，引导程序控制流返回到出错的基本块处，重新开始执行。本文方法相对已有的控制流检测技术，移植性好、检测覆盖率高，且能以很小的存储开销来实现软错误的恢复。

2. 基于完整性检查的控制流错误恢复方法

2.1. 基于完整性检查的控制流错误检测

控制流错误是指程序偏离正确的执行流程，即在没有出现跳转指令的地方发生了跳转，在出现跳转

指令的地方却不跳转或发生错误跳转。基于完整性检查的控制流错误检测通过检测每个基本块的执行完整性来判断程序是否发生控制流错误。

1) 基本块的划分

将程序划分为基本块的集合是实现任何控制流错误检测的基础。基本块是指一段顺序执行的代码，程序的执行只能从基本块的第一条语句进入并从基本块最后一条语句退出。实现基本块的完整性检查需要获得每个基本块的结构信息。

本文选择在汇编语言一级划分基本块。在汇编语言一级划分基本块就是在顺序扫描汇编程序的过程中，不断的寻找程序控制指令如函数调用指令、函数返回指令和跳转指令的位置来确定出每个基本块的边界。由此，基本块的划分规则是：

a) 以跳转指令、函数调用指令和函数返回指令所在位置作为当前基本模块的出口；

b) 当前跳转(调用、调用返回)的后一指令所在位置作为下一个基本模块的入口；

c) 当前跳转(调用、调用返回)的目的指令所在位置作为另一个基本模块的入口。

2) 分块表的设计

分块表主要用于存储基本块的结构信息，这些信息是检查基本块完整性的重要依据。基本块的结构信息是指基本块入口(基本块第一条指令在程序中的地址)、基本块出口(基本块最后一条指令在程序中的地址)、块长度(基本块中包含的指令条数)、下一跳地址(是指跳转指令的目的地址或函数调用时，子函数的第一条指令所在的地址)。

由于在基于完整性检查的控制流错误检测采用了关键指令三模冗余技术[14]来提高控制流在基本块间的转移正确率。关键指令三模冗余是指将三条相同的指令执行一次，并对三条指令的执行结果做表决，将占优势的结果作为指令执行的正确结果来采用。为了实现关键指令三模冗余，需将每个基本块的最后一条指令在分块表中存储三份。为此，将分块表设计成一个链表，其中链表结点的结构如图 1 所示。

3) 检测点的设置

程序运行过程中，其控制流每进入一个基本块，就需在控制流退出这个基本块时对其进行完整性检查。因此，在基于完整性的检查的检测方案中，选择在每个基本块的出口处设置检测点。出于降低存储开销的考虑，将检测代码设计成一个独立的模块置于程序代码的最后，而在每个基本块的最后一条指令处设置检测点——即将每个基本块的最后一条指令替换为一条跳向检测模块的跳转指令。

这种检测点设置方式的优点是在每个检测点处复用检测代码，从而降低程序的存储开销。对于原来的基本块的最后一条指令，在划分基本块的时候已经将它的三个备份存储在分块表的相应结点中，这条指令将在检测模块的最后执行。

4) 检测模块的构建

检测模块是一个置于程序末尾的独立代码段，用于实现具体的控制流错误检测。所有基于软件实现的控制流错误检测都会“中断”原程序的执行，为了使被“中断”的程序控制流能正确返回。因此，在检测过程中开辟一个断点存储区，用于保存检测点处的程序断点信息。此处保存的断点信息除了用于程序控制流的正确返回，也将用于初始化下一个基本块的完整性检测。完成程序断点保护后，即可开始进行基本块的完整性检查。检测过程如图 2 所示。

图 2 中，计数器 counter 的值在控制流每进入一个基本块之前清零，此后每执行一条指令，counter 的值加 1 直至在检测点处停止计数。

5) 基于完整性检查的控制流错误检测

在检测过程中，设定一个分块表工作指针，该指针始终指向下一个即将进行完整性检查的基本块的信息所在的结点。基于完整性检查的控制流错误检测执行流程如图 3 所示，算法描述如下。

基本块入口	基本块出口	块长度	下一跳地址	指令备份1	指令备份2	指令备份3	链表下一结点地址

Figure 1. Structure diagram of partitioned table nodes
图 1. 分块表结点的结构示意图

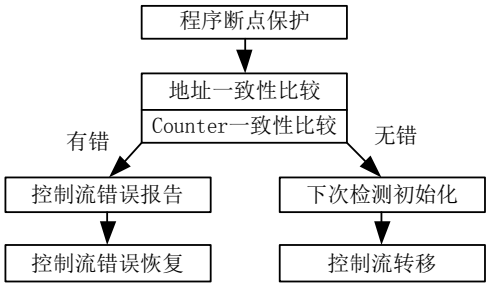

Figure 2. Flow chart of detection module implementation
图 2. 检测模块内的执行流程图

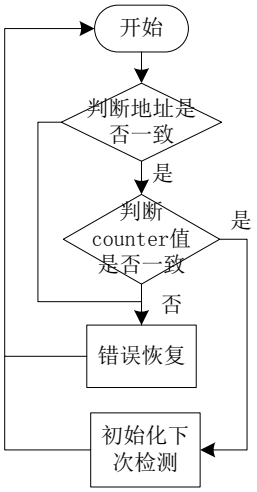

Figure 3. Integrity test flow chart
图 3. 执行完整性检测流程图

Step 1. 比较当前检测点处的地址与分块表工作指针所指的结点中的基本块出口信息是否一致。

Step 2. 比较 counter 的值是否与分块表工作指针所指的结点中的块长度信息是否一致。根据前两步的比较结果，形成以下三种结论：

a) 若地址比较和 counter 值比较均一致，表明当前基本块被完整执行，程序没有发生控制流错误；

b) 若地址比较结果不一致，表明程序发生基本块间的控制流错误跳转；

c) 若地址比较结果一致而 counter 值比较不一致，表明程序发生基本块内的错误跳转。

Step 3. 如果检测到控制流错误，则报告控制流错误并执行恢复控制流错误操作。

Step 4. 如果没有检测到控制流错误，则将 counter 的值置零，并采用“预转移”方法将分块表工作指针移向下一个即将进行完整性检查的结点。

“预转移”是指在控制流返回原程序之前恢复程序断点信息并用上一个基本块中的最后一条指令执行三模冗余，再根据三模冗余的结果决定将工作指针指向下一个即将进行完整性检查结点。“预转移”只用于实现工作指针的移动，控制流并没有真正发生转移。完成上述步骤后，须再一次恢复之前保存的程序断点信息并执行三模冗余，才能使控制流真正转移回原程序。

2.2. ICBR 的工作机制

控制流错误恢复就是指重新执行发生控制流错误的基本块。通过分块表工作指针的位置，可以准确定位出发生控制流错误的基本块。在上一个检测点处，控制流从检测模块的末尾跳向当前工作指针指向的结点所对应的基本块。为了重新执行发生控制流错误的基本块，可以再次执行上一次检测时，在检测模块末尾实现的控制流转移过程，再重新加载未受软错误影响的基本块代码并执行即可恢复之前发生的控制流错误。

为了实现上述控制流恢复方案，本文提出设立恢复指针和恢复存储区。其中，恢复指针始终指向上一次进行完整性检查的基本块所对应的分块表结点。在每次分块表工作指针转移前，将工作指针的值赋予恢复指针即可实现恢复指针的移动。恢复存储区则用于存储上一次检测时保存的程序断点信息，其大小与断点存储区的大小一致。即在系统中开辟出两个大小等于保存程序断点信息所需的存储区域，分别命名为存储区 X 和存储区 Y，将两个存储区域交替作为断点存储区和恢复存储区使用。例如，在一次完整性检测时，存储区 X 作为恢复存储区保存着上一次完整性检测时的程序断点信息，则存储区 Y 就作为当前完整性检查的断点存储区；而在下一次完整性检查时，将存储区 Y 作为恢复存储区且其内的信息不变，同时将存储区 X 作为断点存储区使用。

3. 示例与性能分析

3.1. ICBR 示例

为了验证本文方法的有效性，本节将通过一个示例来具体说明检测机制和恢复机制的工作过程。

某 DSP 汇编程序的局部控制流如图 4 所示。图 4 中矩形表示基本块，在基本块 1、2 和 3 中，标示出了检测点的位置；黑色箭头表示正确的控制流转移，红色虚线箭头表示错误的控制流转移。

若程序首先执行基本块 1，在控制流进入基本块 1 之前，分块表工作指针已经指向基本块 1 所对应的分块表结点。在检测点 1 处，控制流跳向检测模块。在检测模块内，先将程序断点信息保存在存储区 X 中，接着对基本块 1 的完整性进行检查，检查结果表明基本块 1 内没有发生控制流错误。然后，将当前工作指针的值赋予恢复指针，即此时恢复指针指向基本块 1 所对应的分块表结点，并采用“预转移”方法将工作指针指向下一个即将执行的基本块所对应的结点。

假设在执行基本块 1 后，程序控制流会进入基本块 2，则“预转移”的结果是将工作指针指向基本块 2 所对应的结点。在检测模块工作的最后，应用关键指令三模冗余技术将程序控制流转移到基本块 2 处继续执行。若基本块 2 内发生了一个基本块 2 内部到基本块 3 内部的控制流错误转移，则程序会错过检测点 2。当程序执行到检测点 3 时，再次调用检测模块。

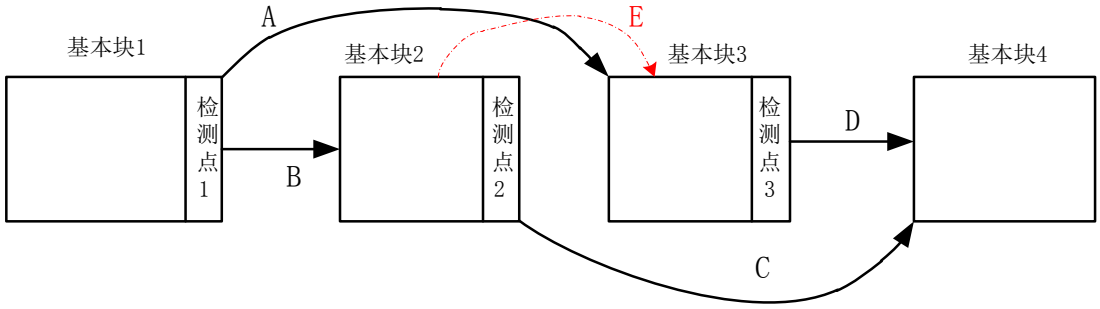

Figure 4. Example of control flow error detection and recovery
图 4. 控制流错误检测和恢复示例

由于在前一次完整性检查中，将存储区 X 作为了断点存储区，则此时应将存储区 X 作为恢复存储区并维持其信息不变，而将存储区 Y 作为当前完整性检查的断点存储区。接着，对基本块 2 的完整性进行检查，即先将当前检测点的地址(检测 3 的地址)与分块表工作指针所指的结点中的基本块出口信息(检测点 2 的地址)比较，比较的结果为不一致。由此，检测模块判断在基本块 2 内发生了控制流错误。

检测出当前控制流错误后，则进行控制流错误恢复工作。控制流错误的恢复过程与检测模块最后的控制流转移过程类似，即将恢复存储区 X 内的程序断点信息恢复一次，然后执行恢复指针所指的结点中存储的三条指令，再一次用关键指令三模冗余技术实现了基本块 1 到基本块 2 的控制流转移。接着，程序重新执行基本块 2，实现控制流错误恢复。

由此可以看出，在控制流错误检测的基础上，设置一个恢复指针和新开辟一个程序断点存储区，即可在检测到控制流错误后，快速地实现控制流错误恢复。在关键指令三模冗余技术的配合下，基于完整性检查的控制流错误检测方法能检测出所有类型的控制流错误。

上述示例中讨论了一种类型的控制流错误的恢复过程，其它类型的控制流错误的恢复过程与此完全一致。

3.2. 性能分析

目前，公开文献中提及的控制流错误恢复方法基本上都能对检测到的控制流错误进行适当的恢复，各种恢复方法之间的差异主要体现在性能开销上，性能开销一般由时间开销和存储开销来评价。值得注意的是，错误恢复方法的性能开销一般都是连同其控制流错误检测方法来分析的。ICBR 所带来的存储开销比较衡定，其运行时间开销主要受源程序中循环结构的影响，例如在 256 个样本点的 FFT 程序中，多数循环结构的执行次数为 256 次，在 128 个样本点的 FFT 程序中，多数循环结果的执行次数为 128 次，这将导致前者的时间几乎为后者的两倍。但是，在 ICBR 中，循环结构处的检测点是可以优化的，此时，通过降低 ICBR 的错误检测和恢复能力，可以有效减少运行时间的增加。

4. 总结

本文针对 DSP 程序存储区的软错误，提出了一种控制流错误恢复方法。该方法在基于完整性检查的控制流软错误检测方法的基础上，通过设置一个恢复存储区和分块表恢复指针，即可在检测到控制流错误时快速地恢复错误。

由于基于完整性检查的控制流错误检测方法具有近乎 100%的错误检测覆盖率，配合控制流错误恢复方法，则可以形成一个完整的控制流错误检测与恢复方法体系，对 DSP 等器件的抗软错误加固设计开发具有重要的指导意义。

基金项目

国家自然科学基金(61371024)、航空科学基金(2013ZD53051)、航天支撑技术基金、中航产学研项目(cxy2013XGD14)。

参考文献 (References)

[1] Saha, G.K. (2006) Software based fault tolerance: A survey. *ACM Ubiquity*, **2006**, Article No. 1.

[2] Reis, G.A., Chang, J., Vachharajani, N., et al. (2005) Software controlled fault tolerance. *ACM Transactions on Architecture and Code Optimization*, **V**, 1-28.

[3] 刑克飞 (2007) 星载信号处理平台单粒子效应检测与加固技术研究. 工学博士学位论文, 国防科技大学, 长沙.

[4] 贺兴华, 肖山竹, 张路, 张开锋, 陶华敏, 卢焕章 (2010) 空间 DSP 信息处理系统存储器 SEU 加固技术研究. 宇

航学报, **2**, 472-477.

[5] Goloubeva, O. and Rebaudengo, M. (2003) Soft-error detection using control flow assertions. *The proceedings of the* 18*th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems* (*DFT*'03), Boston, 3-5 November 2003, 581-588.

[6] Madeira, H. and Silvia, J.G. (1991) On-line signature learning and checking: experimental evaluation. *The Proceedings of the IEEE Comp-Euro* 91: *Advanced Computer Technology, Reliable Systems and Applications*, Bologna, 13-16 May 1991, 642-646

[7] Saxena, N.R. and McCluskey, E.J. (1990) Control-flow checking using watchdog assists and extended-precision checksums. *IEEE Transactions on Computers*, **39**, 554-559.

[8] Alkhalifa, Z., Nair, V.S.S., Krishnamurthy, N., et al. (1999) Design and evaluation of system-level checks for on-line control-flow error detection. *IEEE Transactions on Parallel and Distributed Systems*, **10**, 627-641.

[9] Oh, N., Shirvani, P.P. and McCluskey, E.J. (2002) Control-flow checking by software signatures. *IEEE Transactions on Reliability*, **51**, 111-122.

[10] Jafari-Nodoushan, M. (2008) Control-flow checking using branch instructions. *IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, Shanghai, 17-20 December 2008, 66-72.

[11] Borin, E., Wang, C., Wu, Y.F., et al. (2006) Software-based transparent and comprehensive control-flow error detection. *The Proceedings of the International Symposium on Code Generation and Optimization* (*CGO*), New York, 26-29 March 2006, 333-345.

[12] Chen, Y.-Y. (2005) Concurrent detection of control flow errors by hybrid signature monitoring. *IEEE Transactions on Computer*, **10**, 1298-1313.

[13] Vemu, R., Gurumurthy, S. and Abraham, J.A. (2007) ACCE: Automatic correction of control-flow errors. *The Processing of* 4*th International Symposium on Test*, Santa Clara, 21-26 October 2007, 1-10.

[14] Tan, L.F., Tan, Y. and Xu, J.J. (2013) CFEDR: Control-flow error detection and recovery using encoded signatures monitoring. *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems* (*DFTS*), New York City, 2-4 October 2013, 25-32.

[15] 谭兰芳 (2013) 面向软错误的故障恢复和验证技术研究. 工学博士论文, 国防科技大学, 长沙.

[16] Zarandi, H.R., Maghsoudloo, M. and Khoshavi, N. (2010) Two efficient software techniques to detect and correct control-flow Errors. 2010 *IEEE* 16*th Pacific Rim International Symposium on Dependable Computing* (*PRDC*), Tokyo, 13-15 December 2010, 141-148.