计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
46(2): 345-351, 2009

一种软硬件结合的控制流检测与恢复方法

龚 锐 陈 微 刘 芳 戴 葵 王志英
(国防科学技术大学计算机学院 长沙 410073)
(rgong@nudt.edu.cn)

Control Flow Checking and Recovering by Compiler Signatures and Hardware Checking

Gong Rui, Chen Wei, Liu Fang, Dai Kui, and Wang Zhiying
(*College of Computer, National University of Defense Technology, Changsha* 410073)

Abstract With the exponential increase in the transistors per chip, microprocessors are becoming more susceptible to soft errors. Control flow checking has been proved effective in promoting soft error tolerant ability of microprocessors. The conventional control flow checking method inserts large number of signature instructions in the program by compiler. So it imposes large overheads on both binary code size and program execution performance. Moreover, the conventional control flow checking method does not consider the recovery from control flow errors. A new method, control flow checking and recovering by compiler signatures and hardware checking (CFCCH), is proposed in this paper to solve the aforementioned problems. CFCCH uses a compiler to insert signature data, not signature instructions, in the program to reduce the binary code size. Hardware checking is automatically triggered after the branch/jump instruction so that the execution cycles of the checking operation can be reduced. Hardware implemented context saving and recovering is also proposed to provide fast recovering from control flow errors. CFCCH based on 8051 architecture is implemented in this paper. Random faults are injected in the 8051 microcontroller with CFCCH to evaluate the soft error tolerant ability. The experimental results demonstrate that compared with the conventional control flow checking method, CFCCH can efficiently reduce the binary code size and program execution time while keeping the same soft error tolerant ability.

Key words soft error; control flow checking; compiler signature; hardware checking; control flow recovering

摘 要 控制流检测可以有效地提高微处理器容错能力.针对传统软件实现的控制流检测时空开销大的缺点,提出了一种软硬件结合的控制流检测与恢复方法.该方法通过编译自动插入签名数据,由硬件在分支/跳转指令之后自动执行检测,并且提供了硬件现场保存和恢复机制,检测到控制流错误后无需复位系统即可以快速恢复正常控制流.基于8051体系结构实现了软硬件结合的控制流检测与恢复方法,实验结果表明与传统的软件控制流检测相比,该方法在保持相同的错误检测率的情况下,可以大幅减小二进制代码量和额外的性能开销,在发生控制流错误以后可以快速恢复正常控制流.

关键词 软错误;控制流检测;编译签名;硬件检测;控制流恢复

中图法分类号 TP302.8

收稿日期:2007-01-18;**修回日期**:2008-09-13

基金项目:国家"九七三"重点基础研究发展计划基金项目(2007CB310901);国家"八六三"高技术研究发展计划基金项目(2007AA01Z101);国家自然科学基金项目(60773024)

随着空间技术和集成电路技术的高速发展,微处理器已广泛应用于航空航天领域.在宇宙辐射环境中存在着大量的包括电子、质子、光子、α 粒子和重离子在内的高能粒子.微处理器遭到高能粒子轰击可能发生单事件效应(single event effect,SEE),引起存储器或寄存器中数据的非正常翻转,进而导致功能错误.在相同的辐射剂量下,单个器件发生SEE的概率在未来的若干年中将保持不变,甚至略有下降.但是随着工艺的进步,单片集成电路能够集成的器件数目将呈指数增长,所以单个微处理器发生SEE的概率也将呈指数增长.因此,应用于军事和航空航天领域的微处理器必须考虑高能粒子引起的SEE,以提高系统的容错能力.

由高能粒子SEE引起的错误是瞬态且可恢复的,因此也叫做软错误(soft error).软错误包括控制流错误和数据流错误.控制流错误是指程序的运行轨迹发生混乱;数据流错误是指程序可以正常结束,但是产生了错误的计算结果.有33%~77%的SEE将引起微处理器发生控制流错误,特别是对于以控制为主要应用的微控制器,其程序中有大量的控制流转移指令,将有70%以上的SEE导致控制流错误[1].

计算错误一般通过三模冗余[2]或改进的三模冗余技术[3]进行错误屏蔽.而控制流错误一般通过控制流检测的方法进行检测.控制流检测的基本思想是程序运行的可能轨迹在编译时即可确定.在程序运行过程中,检测实际的运行轨迹是否与编译预期的一致.如果一致,则认为没有发生控制流错误;否则,表明发生了错误,可能引起严重的后果,需要立即进行处理.

目前,控制流检测一般采用纯软件的方法实现,二进制代码量和性能开销都较大.针对这种情况,本文提出了软硬件结合的控制流检测方法(control flow checking by compiler signatures and hardware checking,CFCCH),该方法采用软件插入签名数据,硬件自动进行检测.每个基本块只需插入3个签名数据,检测时也只需3个时钟周期,可以减少控制流检测带来的时空开销.在CFCCH的基础上,本文提出了控制流恢复机制CFCCH-R(CFCCH-recovery).通过硬件实现的现场保存与恢复,在发现错误以后可以快速恢复正确的控制流.本文在HJTC 0.25μm工艺下基于8051体系结构实现了软硬件结合的控制流检测与恢复.实验结果表明,该方法可以有效减少代码量和性能开销.并且由于存在硬件现场保存与恢复机制,发现控制流错误以后只需1个时钟周期就可以恢复正确的控制流.

1 相关工作

早期的控制流检测采用硬件看门狗(watchdog)技术,利用专门的看门狗芯片或看门狗模块实时监控程序存储器的地址总线,以获得程序跳转等控制流信息并进行检测[4-5].对于有Cache的微处理器,若取指时指令Cache命中就不会读程序存储器.所以采用看门狗进行控制流检测可观察性有限,并不适用于所有的微处理器.

此后,研究人员提出了一系列基于软件实现的控制流检测方法.主要包括基于断言的方法和基于签名的方法.ECCA(enhanced control-flow checking using assertion)[6]在程序中插入断言,并用软件对断言进行判断.YACCA(yet another control-flow checking using assertions)[7]是另一种基于断言的控制流检测方法,该方法在检测时需要做除法运算,性能开销较大.CFCSS(control flow checking by software signatures)[8]是一种广泛使用的基于签名的检测方法,可以对95.8%以上的控制流错误进行有效检测,该方法需要由编译向源代码中插入大量检测指令.国内的研究人员也对基于签名的软件控制流检测进行了相关的研究和实现[9-10].

上诉基于断言和基于签名的方法都采用软件实现,二进制代码量和性能开销大,很多研究都致力于减少这些开销.ACFC(assertion for control flow checking)[11]对经典的基于断言的检测方法ECCA进行了改进,以减少ECCA的存储开销.SWTES(software based error detection technique using encoding signatures)[12]利用PowerPC处理器提供的分支踪迹异常机制在PowerPC体系结构上实现了基于CFCSS的软件控制流检测,由于充分利用了分支踪迹异常,所以可以有效隐藏检测时间,但是该方法只能用于PowerPC体系结构的处理器.

由于CFCSS是一种广泛使用的有效的控制流检测方法,所以本文的研究工作也是基于CFCSS的方法,采用软硬件结合的机制来降低其代码量和性能开销.另外,传统的控制流检测发现错误以后,跳转到错误处理程序进行软件处理,一般直接复位系统,这降低了系统的可用性.而本文提出了一种有效的硬件控制流恢复方法,不需复位系统即可快速恢复正确的运行现场.

2　软件签名与硬件检测

本文提出的 CFCCH 方法进行软件签名,由编译对程序流图中每一个节点赋予签名值、签名距离及运行时调整签名等签名数据,并将其植入到程序中.在程序运行的过程中,执行到控制流转移指令后触发一次硬件检测操作,对签名数据进行检测.

首先定义程序控制流图为有向图 $CFG=(V, E)$,其中 $V=\{v \mid v$ 为基本块$\}$,$E=\{\langle v_i, v_j\rangle \mid$存在从 v_i 到 v_j 的分支或跳转指令$\}$.对于某个特定的基本块 v_j,赋予其唯一的签名值 S_j,并存储于基本块 v_j 的块头.如果 $\exists\langle v_i, v_j\rangle \in E$,则将 v_i 到 v_j 的异或距离 $d_j=S_i\oplus S_j$ 作为签名距离存储在基本块 v_j 的块头.当程序执行从 v_i 到 v_j 的控制流转移时,计算实时签名值 $s_j=S_i\oplus d_j$.如果分支或转移正确,则 $s_j=S_i\oplus d_j=S_i\oplus(S_i\oplus S_j)=S_j$.如果检测发现 $s_j\neq S_j$,则表明发生了控制流错误.

由于控制流图中存在多扇入多扇出节点的情况,仅有签名值和签名距离是不够的.如图 1(a)所示的 CFG 子图中,假设存储于 v_3 节点的签名距离由 v_1 节点确定,即 $d_3=S_1\oplus S_3$.则从 v_2 到 v_3 的正确的控制流转移将被判断为非法.解决这种情况的办法是对每一个基本块 v_i 引入运行时调整签名 A_i,当发生控制流转移时,计算实时签名值 $s_j=S_i\oplus A_i\oplus d_j$.如图 1(b)所示,$v_1$ 节点运行时调整签名 $A_1=0$,并由 v_1 节点确定 v_3 节点的签名距离 $d_3=S_1\oplus S_3$.赋予 v_2 节点唯一的签名值 S_2,$S_2\neq S_1$.执行从 v_2 到 v_3 的合法控制流转移时,计算 $s_3=S_2\oplus A_2\oplus d_3=S_2\oplus A_2\oplus(S_1\oplus S_3)$,只需设定 $A_2=S_1\oplus S_2$,就可以保证 $s_3=S_3$,也就解决了正常控制流转移被判断为非法的问题.但是增加运行时调整签名将会引起混淆.图 1(b)中的单扇入节点 v_4 的签名距离由 v_2 节点唯一确定,即 $d_4=S_2\oplus S_4\oplus A_2=S_1\oplus S_4$,所以由 v_1 到 v_4 节点的非正常控制流转移将被判断为合法.全软件实现的控制流检测解决混淆的办法是从 v_i 进入单扇入节点 v_s 时只计算 $s_s=S_i\oplus d_s$,而进入多扇入节点 v_m 时计算 $s_m=S_i\oplus A_i\oplus d_m$.这样图 1(b)中的 $d_4=S_2\oplus S_4$,如果发生从 v_1 到 v_4 的非法转移也能被检测到.本文采用的是硬件检测,处理器从 v_i 节点进入 v_j 节点时无法判断 v_j 是否为多扇入节点.所以硬件进行检测时,统一计算 $s_j=S_i\oplus A_i\oplus d_j$.这样可能发生混淆的情况,但是这只发生于 v_1 恰好错误地跳转到 v_4 基本块的块头时,其发生的概率是很小的.

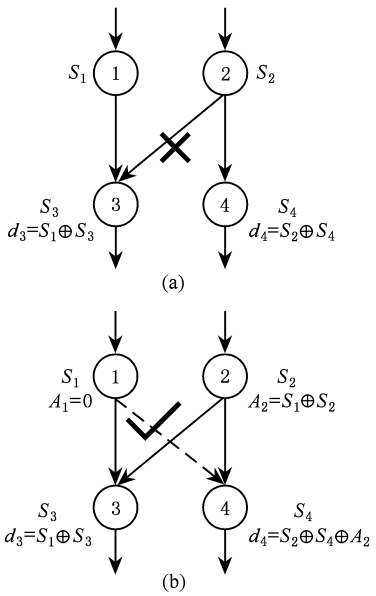


Fig. 1　Control flow graph. (a) Without run-time adjusting signature and (b) With run-time adjusting signature.

图 1　程序流图.(a)不带运行时调整签名;(b)带运行时调整签名

编译进行软件签名的算法 CompilerSignature 如下:

算法 1. 编译签名算法.

```
CompilerSignature(CFG)
Assign a unique signature value S_i to every node v_i
Insert S_i in the head of v_i
for every node v_j in CFG do
  if v_j is NOT a multi-fan-in node then
    Select {v_i}=pred(v_j)
    Calculate signature distance of v_j, d_j=S_i⊕S_j
    Insert d_j before S_j in the head of v_j
  else
    Select v_i∈pred(v_j), calculate d_j=S_i⊕S_j
    Insert d_j before S_j in the head of v_j
    for every node v_k∈pred(v_j) do
      if v_k≠v_i then
        Run-time signature D_k=S_k⊕S_i
        Inset D_k after S_k in the head of v_k
      else
        Insert D_i=0 after S_i in the head of v_i
      endif
    endfor
  endif
endfor
```

传统的 CFCSS 需要在每个基本块头插入大量的签名指令,插入后的程序代码段如图 2(a)所示. CFCCH 由硬件进行控制流检测,所以编译只需在源代码中插入签名数据,这样可以大大减小目标代码的体积,插入签名数据后的程序片断如图 2(b)所示:

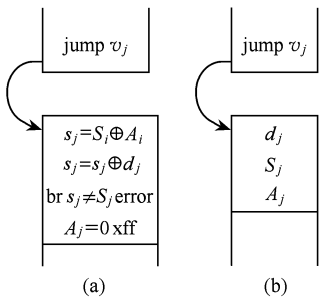


Fig. 2 Program segment after insertion. (a) CFCSS with inserted signature instructions and (b) CFCCH with inserted signature data.

图 2 插入后程序片断. (a) CFCSS 中插入签名指令; (b) CFCCH 中插入签名数据

为了实现硬件检测,增加了两个特殊寄存器 *Sreg* 和 *Areg*,分别记录当前基本块的签名值和运行时调整签名. 执行完分支/跳转指令后,从程序存储器中取出签名数据进行检测. 检测共需 3 个时钟周期,每个时钟周期的工作如表 1 所示. 与执行 CFCSS 的签名指令相比,硬件检测可以减少检测所需的时间开销.

Table 1 Operation of Hardware Checking

表 1 硬件检测操作

Cycle	Operation
1	Read d_i from instruction memory, update $Sreg=Sreg\oplus Areg\oplus d_i$
2	Read S_i from instruction memory, compare with $Sreg$. If not identical, generate error signal.
3	Read A_i from instruction memory, update $Areg=A_i$

特别需要注意的是中断的处理. 处理器发生中断后,转移到中断处理程序进行中断处理. 由主程序到中断处理程序的控制流转移不在编译的预计之内. 解决的办法是发生中断后由软件或硬件将 *Sreg* 和 *Areg* 压栈,然后将其置为 0,这样可以进行中断处理程序子控制流图的检测. 而在退出中断处理程序时进行弹栈,重新恢复主程序的控制流图上下文.

与 CFCSS 相比,本文提出的软硬件结合的方法只需插入 3 个签名数据而非检测指令,可以减小代码体积. 进行一次硬件检测只需额外增加 3 个时钟周期可以提高代码的运行速度. 在 CFCSS 方法中,如果发生控制流错误,跳转到基本块内部,仍然可以正常执行,直到下一检测指令. 而在发生控制流错误到错误被检测到这段时间内,可能已经对片外发出了错误的控制信号或写入了错误的计算结果,这对于航空航天级的高可靠实时控制系统来说是不可接受的. 本文提出的 CFCCH 方法,在每条分支或跳转指令执行之后立即触发硬件检测,可以保证检测的实时性.

3 现场保存与恢复

传统的软件实现的控制流检测,检测到控制流错误后跳转到错误处理程序进行软件处理,一般直接对芯片进行复位,这对于一些不能停机的高可用性系统来说是不可接受的. 由 SEE 引起的控制流错误只是瞬态错误,只需恢复正常的控制流再继续执行就可能消除. 恢复正常的控制流需要进行现场保存和恢复. 而由软件进行保存恢复性能开销很大. 本文提出了一种由硬件进行现场保存和恢复的机制.

处理器运行的现场是指处理器内部所有和运行状态相关的存储单元,包括特殊寄存器、通用寄存器文件和片内数据 RAM. 本文针对特殊寄存器和寄存器文件/内部 RAM 的不同特点,分别采用冗余和写缓冲机制进行现场保存与恢复. 将所有的特殊寄存器进行双模冗余,并分为运行组和备份组. 复位时两组都复位为相同的值. 处理器首先在运行组上运行,在确定没有发生控制流错误的现场保存点,将运行组的数据写入备份组特殊寄存器. 检测到控制流错误以后,根据备份组数据恢复运行组现场. 特殊寄存器的保存和恢复都只需要一个时钟周期.

目前的高性能微处理器中使用了大量的寄存器文件和片内数据 RAM,如果也像特殊寄存器一样对其进行双模冗余,将产生很大的面积开销. 解决办法是不进行冗余而引入写缓冲,在写寄存器文件或片内 RAM 时只写入写缓冲. 因此写缓冲包含了被修改的现场. 运行至现场保存点时才将写缓冲的数据写入寄存器文件和片内 RAM,保存现场的时间开销依赖于写缓冲中数据的项数. 而在发生错误时,只需花费一个时钟周期作废写缓冲中的数据就能恢复现场.

要能进行正确的恢复必须能够保存正确的现场. 在进入每个基本块进行硬件控制流检测无误后进行现场保存. 此时程序控制流正确. 当一个基本块中发生多次对寄存器文件或片内数据 RAM 的写操作导致写缓冲写满后,如果再继续写将导致写缓冲

溢出.此时需要触发一次现场保存,将写缓冲中的数据写入备份组.所以在检测到控制流错误后,可以自动回退到上一基本块块头或上一基本块内的某一条写寄存器文件/RAM 指令之后.

采用本文提出的硬件现场保存与恢复方法,在发生控制流错误以后不用复位系统就可以快速恢复正确的控制流,可以有效地保证芯片的可用性.与软件实现的现场保存和恢复相比,可以大大减少保存和恢复的时空开销.

4　基于 8051 体系结构的实现

本文基于 8051 体系结构实现了软硬件结合的控制流检测与恢复机制.所实现的高可靠 8051 体系结构兼容所有的 ASM51 指令集.指令的执行周期为 2～6 个时钟周期,采用流水的方式实现了取指和执行的并行.为便于比较,实现了 4 种 8051 微控制器:1)original:没有做任何软硬件加固,不具备检错纠错能力.2)CFCSS:采用纯软件方法进行控制流检测.没有进行硬件改动,由编译插入签名指令.具备检错能力,发生错误以后复位系统.3)CFCCH:采用软硬件结合的方法进行控制流检测.进行了硬件改动,由编译插入签名数据,硬件进行检测.具备检错能力,发生错误以后复位系统.4)CFCCH-R:带现场恢复功能的软硬件结合控制流检测.同样由编译插入签名值,执行与 CFCCH 相同的二进制代码.硬件上对 8051 的所有特殊寄存器进行了冗余,并对内部数据 RAM 增加了一个 16 个入口的写缓冲,以实现现场保存和恢复.具备检错和硬件纠错的能力,发生错误以后恢复到上一现场保存点重新执行.

将 4 种 8051 微控制器在 HJTC25 工艺下进行了综合,综合的时钟约束均为 100 MHz,其单元面积的比较如表 2 所示.与没有做任何硬件改动的 CFCSS 相比,CFCCH 由于增加了硬件检测模块,面积有少量增长.而 CFCCH-R 由于将所有的特殊寄存器全部双模冗余,并且增加了写缓冲,所以面积比 CFCSS 增加了 41%.

Table 2　Comparison of Cell Area

表 2　单元面积比较　　mm^2

Original	CFCSS	CFCCH	CFCCH-R
1.068	1.068	1.071	1.506

图 3 表示了针对不同测试程序 4 个版本的二进制代码量和性能的比较.图 3 中的数据进行了归一化处理,以未做任何软硬件加固的 original 版本为基准进行比较.CFCSS 在每个基本块头插入的签名代码,编译为 51 指令后占 11 B.CFCCH 和 CFCCH-R 二进制代码量相同,由于只在每个基本块头插入了 3 B 签名数据,其额外的代码开销约在 11%～27%之间,远低于纯软件实现的 CFCSS.CFCSS 的签名检测代码执行一次需要 13 个周期,性能开销较大.CFCCH 在每个检测点增加了额外的 3 个时钟

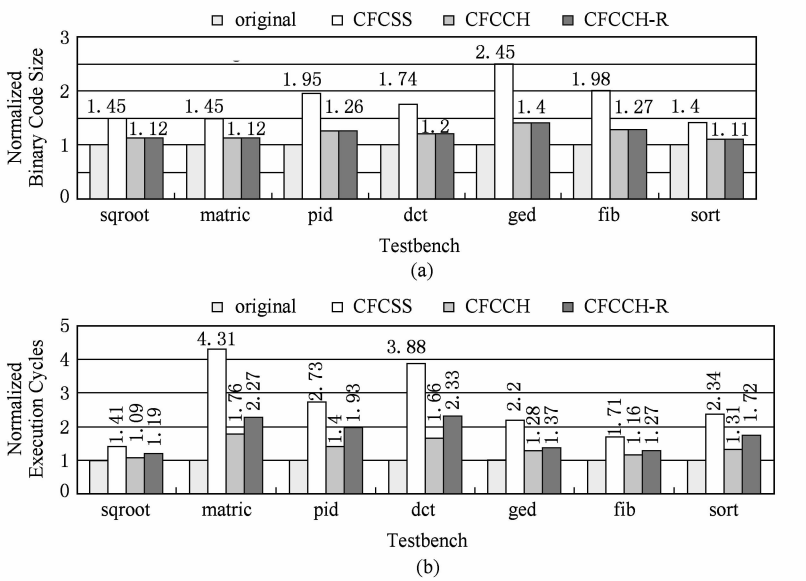


Fig. 3　Normalized comparison. (a) Binary code size comparison and (b) Performance comparison.

图 3　归一化比较.(a) 二进制代码量比较;(b) 性能比较

周期,带来了 9%～76%的性能开销,低于 CFCSS. 由于有额外的现场保存点的存在,CFCCH-R 的性能开销略大于 CFCCH,但仍低于 CFCSS.

为验证 CFCCH 及 CFCCH-R 对控制流检测的有效性,本文对计算最大公约数的 gcd 程序进行了 10000 次二进制代码随机扰乱,每次扰动后在 CFCSS,CFCCH 及 CFCCH-R 上运行,并与正常的程序执行踪迹进行比较. 图 4 比较了与总的错误注入次数相比,未检测到错误(undetected)和误报(distortion)的百分比. 由于 CFCCH 和 CFCCH-R 执行相同的二进制代码,且采用相同的检测方法,所以发生未检测错误以及误报的次数相同. 可以看出 3 个版本未检测到的控制流错误数目相当. 需要注意的是 3 个版本都存在误报的情况,即正常的控制流转移被判断为非法. 这是由于 CFCSS 中的检测语句或 CFCCH,CFCCH-R 中的签名数据被随机改动而导致的,实际的辐射环境下程序存储器中的数据发生 SEE 可能引发这样的情况. 从图 4 中可以看出,CFCCH 和 CFCCH-R 对控制流错误的检测率和误报率都与 CFCSS 相当. 而 CFCSS 已经被证明为是有效的控制流检测技术,所以本文提出的软硬件结合的控制流检测与恢复方法是有效的.

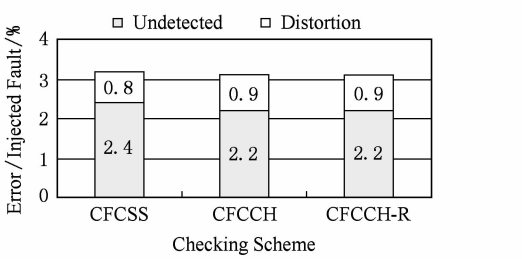


Fig. 4 Results of fault injection experiment with different checking scheme.

图 4 不同检测方法错误注入结果

5 结 论

传统的纯软件实现控制流检测具有二进制代码量大、性能开销大、检测到错误后直接复位系统等缺点. 本文提出了软硬件结合的控制流检测与恢复方法. 该方法采用编译在程序基本块头自动插入签名数据,硬件执行分支/跳转指令后触发一次检测操作. 为了在发现错误后能迅速恢复正确的控制流,采用冗余和写缓冲技术对执行现场进行保护,发现错误后可以快速回退到上一现场保存点.

本文基于 8051 体系结构实现了软硬件结合的控制流检测与恢复. 实验结果表明,软硬件结合的控制流检测方法与纯软件实现的控制流检测相比,在近似的错误检测率和面积开销下,可以大幅减小二进制代码量和性能开销. 而硬件现场保存和恢复机制牺牲了一定的芯片面积,但是提供了一种快速恢复正确控制流的途径.

参 考 文 献

[1] Czech E W, Siewiorek D. Effects of transient gate-level faults on program behavior [C] //Proc of IEEE Int Fault-Tolerant Computing Symposium. Los Alamitos, CA: IEEE Computer Society, 1990: 236-243

[2] Gaisler J. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture [C] //Proc of Int Conf. Dependable Systems and Networks. Los Alamitos, CA: IEEE Computer Society, 2002: 409-415

[3] Gong Rui, Chen Wei, Liu Fang, *et al*. Modified triple modular redundancy structure based on asynchronous circuit technique [J]. Journal of Computer Research and Development, 2006, 43(Suppl): 23-27 (in Chinese)
(龚锐, 陈微, 刘芳, 等. 基于异步电路技术改进三模冗余结构[J]. 计算机研究与发展, 2006, 43(增刊): 23-27)

[4] Mahmood A, McCluskey E J. Concurrent error detection using watchdog processors-a survey [J]. IEEE Trans on Computers, 1988, 37(2): 160-174

[5] Majzik I, Hohl W, Pataricza A, *et al*. Multiprocessor checking using watchdog processors [J]. International Journal of Computer Systems Science and Engineering, 1996, 11(5): 301-310

[6] Alkhalifa Z, Nair V S S, Krishnamurthy N, *et al*. Design and evaluation of system-level checks for on-line control flow error detection [J]. IEEE Trans on Parallel and Distributed Systems, 1999, 10(6): 627-641

[7] Goloubeva O, Rebaudengo M, Sonza R M, *et al*. Soft-error detection using control flow assertions [C] //Proc of IEEE Int Symp on Defect and Fault Tolerance in VLSI Systems. Los Alamitos, CA: IEEE Computer Society, 2003: 581-588

[8] Oh N, Shirvani P, McCluskey E J. Control flow checking by software signatures [J]. IEEE Trans on Reliability, 2002, 51(2): 111-122

[9] Li Aiguo, Hong Binrong, Wang Si. A software method for on-line control flow fault detection [J]. Journal of Astronautics, 2006, 27(6): 1424-1430 (in Chinese)
(李爱国, 洪炳熔, 王司. 一种软件实现的程序控制流错误检测方法[J]. 宇航学报, 2006, 27(6): 1424-1430)

[10] Gao Xing, Liao Minghong, Wu Xianghu, *et al*. A control flow checking algorithm based on virtual register [J]. Journal of Astronautics, 2007, 28(1): 183-187 (in Chinese)

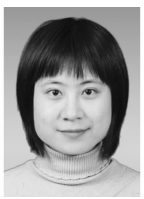
(高星,廖明宏,吴翔虎,等. 基于虚拟寄存器的控制流错误检测算法[J]. 宇航学报,2007,28(1):183-187)

[11] Venkatasubramanian R, Hayes J P, Murray B T. Low-cost on-line fault detection using control flow assertions [C] // Proc of IEEE On-Line Testing Symposium. Los Alamitos: IEEE Computer Society, 2003: 137-143

[12] Fazeli M, Farivar R, Miremadi S G. A software-based concurrent error detection technique for PowerPC processor-based embedded systems [C] //Proc of IEEE Int Symp on Defect and Fault Tolerance in VLSI Systems. Los Alamitos: IEEE Computer Society, 2005: 266-274

Gong Rui, born in 1980. Ph. D. His main research interests include high reliable microprocessor design and asynchronous integrated circuit design.

龚　锐,1980 年生,博士,主要研究方向为高可靠微处理器设计、异步集成电路设计.



Chen Wei, born in 1982. Ph. D. candidate. Her main research interests include computer architecture and high reliable microprocessor design.

陈　微, 1982 年生,博士研究生,主要研究方向为计算机体系结构、高可靠微处理器设计(chenwei@nudt.edu.cn).

Liu Fang, born in 1976. Ph. D.. Her main research interests include high reliable microprocessor design and information security.

刘　芳, 1976 年生,博士,主要研究方向为高可靠微处理器设计、信息安全(liufang@nudt.edu.cn).

Dai Kui, born in 1968. Ph. D. and associate professor. His main research interests include microprocessor design, high performance computer architecture and asynchronous integrated circuit design.

戴　葵,1968 年生,博士,副教授,主要研究方向为微处理器设计、高性能计算机体系结构、异步集成电路设计(daikui@nudt.edu.cn).

Wang Zhiying, born in 1956. Ph. D., professor and Ph. D. supervisor. His main research interests include microprocessor design, high performance computer architecture and asynchronous integrated circuit design.

王志英,1956 年生,博士,教授,博士生导师,主要研究方向为微处理器设计、高性能计算机体系结构、异步集成电路设计(zywang@nudt.edu.cn).

Research Background

Microprocessors now are widely used in space environment, which consists of various high-energy particles. The particle-induced soft errors threaten the reliability of integrated circuits and systems in space. With the development of integrated circuit, microprocessors are more and more susceptible to soft errors. About 33% to 77% soft errors are control flow errors, which means wrong execution trace. Conventional control flow checking methods are based on software, which imposes large overhead on both binary code size and execution performance. Moreover, these software based methods could not recover from control flow errors. Once detecting control flow errors, the chip is reset. This is infeasible for some real time control embedded systems.

Control flow checking by compiler signatures and hardware checking (CFCCH) is proposed in this paper. Compiler inserts signature data, not signature instructions in the program, so that the binary code size can be largely reduced. The hardware checking is triggered after the execution of branch/jump instruction to accelerate the checking process. Hardware implemented context saving and recovering is also proposed, which makes it feasible to fast recover from wrong trace without resetting the microprocessor. The experimental results demonstrate that compared with the conventional control flow checking methods, CFCCH efficiently reduces binary code size and program execution time while keeping the same fault tolerant ability.

This work is supported by the 973 Project of China under grant No. 2007CB310901, the National High-Tech Research and Development Program of China (863 Project) under grant No. 2007AA01Z101, and the National Natural Science Foundation of China under grant No. 60773024.