

一个基于微处理器功能模型的可靠度评估系统

张仕健^{1),2)} 许 彤^{1),2)} 章隆兵¹⁾ 胡伟武¹⁾

¹⁾(中国科学院计算技术研究所计算机系统结构重点实验室 北京 100080)

²⁾(中国科学院研究生院 北京 100039)

摘 要 随着以嵌入式微处理器为核心的容错系统在航空、航天、核电等高可靠领域的广泛应用,如何迅速、方便、低成本地评估这些系统中的容错机制是一个重要的问题.传统的可靠度评估方法需要一个详细的硬件原型才能进行评估,周期长、成本高,因此文中提出了一个基于微处理器功能模型的可靠度评估技术,构建了一个纯软件的可靠度评估系统.和已有的系统相比,该系统评估周期短、控制方便、成本低廉.使用该系统评估软件实现的指令冗余技术、软件实现的断言技术和硬件实现的重复取指执行技术的实验结果表明,该系统的评估是合理的.

关键词 故障注入;可靠度评估;容错技术;微处理器;瞬态故障

中图法分类号 TP303

A Dependability Evaluation System Based on Microprocessor Function Model

ZHANG Shi-Jian^{1),2)} XU Tong^{1),2)} ZHANG Long-Bing¹⁾ HU Wei-Wu¹⁾

¹⁾(Key Laboratory of Computer System and Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing 100080)

²⁾(Graduate University of the Chinese Academy of Sciences, Beijing 100039)

Abstract With the widespread adoption of embedded microprocessor-based systems in safety critical applications, such as aircrafts, spaceships and nuclear power plants, how to rapidly and conveniently evaluate these fault-tolerant mechanisms with low cost is an important problem. The traditional method requires a detailed hardware protocol to do evaluation, which lengthens evaluation period and increases the cost. A new dependability evaluation technique based on microprocessor function model is proposed, which can evaluate fault-tolerant mechanisms more rapidly, more conveniently and more economically than the conventional systems. As a case for study, the new system evaluates three fault-tolerant techniques: the software redundancy technique, the assertion validation technique and the instruction re-fetching and re-execution technique. The results show that the evaluation is reasonable.

Keywords fault injection; dependability evaluation; fault-tolerant technique; microprocessor; transient fault

1 引 言

随着以嵌入式微处理器为核心的容错系统在航

空、航天、核电等高可靠领域的广泛应用,如何检测并恢复微处理器上发生的瞬态故障是一个研究热点.一般而言,克服微处理器瞬态故障的容错技术可以分为软件容错技术和硬件容错技术.软件容错技

收稿日期:2006-07-02;最终修改稿收到日期:2007-12-21.本课题得到国家“九七三”重点基础研究发展规划项目基金(2005CB321600)、国家自然科学基金杰出青年基金项目“计算机系统结构研究”(60325205)、国家自然科学基金(60673146,60703017)、北京市自然科学基金(4072024)资助.张仕健,男,1975年生,博士研究生,研究方向为高可靠微处理器结构和高性能计算机系统. E-mail: zsj_bj@ict.ac.cn. 许 彤,男,1975年生,博士研究生,副研究员,研究方向为集成电路设计与嵌入式应用.章隆兵,男,1974年生,博士,研究方向为高性能微处理器结构.胡伟武,男,1968年生,博士,研究员,博士生导师,研究领域为高性能计算机系统结构、MPP系统和VLSI设计.

术向程序目标码插入检错指令,如指令冗余技术^[1]、断言技术^[2]等;硬件容错技术包括微处理器电路级的容错技术^[3]和微结构级的容错技术^[4].在构建一个高可靠的容错系统过程中,如何迅速、方便、低成本地评估这些容错技术是一个重要的问题.

分析建模法和故障注入法是评估系统可靠度的常用方法.文献[5]提出采用 Petri 网给微处理器建立可靠度分析模型,但是随着软硬件变得越来越复杂,使用该方法建立正确、实用的分析模型变得越来越困难.故障注入法是一种强大的可靠度评估技术,它有 3 种实现方式:(1) 硬件故障注入;(2) 软件故障注入;(3) 仿真故障注入.硬件故障注入准确度最高,但是评估周期长,评估成本高;软件故障注入需要较高的软件开发成本;仿真故障注入像硬件的仿真模型,如门级电路模型和寄存器传输级模型注入故障,能够方便地控制故障和观测故障,但是由于仿真模型开发周期长,仿真速度慢,可靠度评估也需要较长的周期.

本文提出了一个基于微处理器功能模型的可靠度评估技术.首先,将处理器的微结构抽象成指令级模型和时序级模型,缩短仿真模型的开发周期,提高仿真速度;然后,定义微处理器行为级故障模型,再现微处理器底层电路可能发生的故障,并将该故障模型映射到微处理器功能模型之中,通过该故障模型可以方便地控制故障的注入时间和注入位置;最后,将功能模型和故障模型集成在 SimOS 模拟环境中,实现了一个纯软件的可靠度评估系统.和已有的可靠度评估系统相比,本系统评估周期短,控制方便,成本低廉.

本文第 2 节对比、分析已有的故障注入技术;第 3 节详细阐述新技术的系统构架、功能模型、故障模型及系统特点;第 4 节是实验及结果分析;最后是全文总结.

2 故障注入技术简介

故障注入技术是一种按照选定的故障类型在目标计算机系统中人为地产生故障并对系统响应信息收集处理的实验过程.由于故障注入技术能够很快地再现目标系统中自然产生的物理故障,大大缩短了验证周期,因而广泛地用于可靠度评估.故障注入技术有 3 种常见的实现形式:硬件故障注入、软件故障注入和仿真故障注入.下面从评估周期、评估成本、可控度及准确度^[6] 4 个方面对这些技术进行

对比、分析.

2.1 硬件故障注入

该技术又分为接触式注入和非接触式注入^[6].接触式注入采用活动探针或专用插槽通过改变芯片引脚上的电流和电压来注入故障,虽然注入的故障发生在芯片引脚上,和芯片内部发生的固定型故障和桥接故障不完全相同,但实际效果却有很多雷同之处. MESSALINE^[7]、RIFLE^[8]都是典型的接触式注入故障的系统,用于铁路控制系统及通信系统的可靠度评估.非接触式注入采用重离子辐射或电磁场干扰的方法注入故障,常用来向芯片中注入瞬态故障.瑞典 Chalmers 科技大学开发的 FIST^[9]系统就采用重离子辐射的方式向目标芯片内部注入瞬态故障,它的目标系统是一个双 CPU 的计算机系统,真空管中的辐射源作用在其中的一个 CPU 上,另一个 CPU 作为参照,检测辐射是否导致芯片发生位翻转.硬件注入故障技术和故障实际发生的环境最接近,能够注入开路、桥接、位翻转、固定型等多种类型的故障,但是该方法也存在几个明显的缺点:(1) 通常需要昂贵的硬件设备,同时如果注入的剂量控制不好很可能损伤目标芯片,增加了评估风险并提高了评估成本;(2) 可控性和可观察性不好,难以精确控制故障注入的位置与时间,因此不利于故障分析;(3) 目标芯片必须是实际的物理芯片,因此只有等到处理器流片之后才能进行,显然大大增加评估周期;(4) 难以将故障注入过程自动化,为注入相当数量的故障以便使分析结果具有统计意义,需要耗费大量的时间.由于这些不利因素,限制了硬件故障注入方法的普遍应用.

2.2 软件故障注入

软件故障注入技术(SoftWare-Implemented Fault Injection, SWIFI)是在程序编译或运行时改变程序代码段或数据段上的内容来模拟故障的一种技术,它不需要昂贵的设备,可以向应用程序和操作系统注入故障,模拟软件故障、硬件故障和瞬态故障等故障类型,是一种低成本的故障注入方法. Austin Texas 大学开发的 FERRARI^[10]系统是一个典型的软件注入故障的系统,它采用 ptrace 接口跟踪目标进程并插入陷阱机制,当程序运行到设定的指令位置或触发了时间定时器时,该系统进入陷阱,陷阱处理程序改变目标指令的寄存器或内存位置,通过这种方式它可以向 CPU、内存和总线注入永久故障或瞬态故障. Portugal Coimbra 大学开发的 Xception^[11]系统利用现代微处理器具备的高级调试及性能监视

特征,设立了一套全面的故障触发机制,同时不需要对目标系统插入陷阱机制,减少了对目标系统造成的影响,但是仍然需要修改目标系统中部分中断响应程序. 软件故障注入技术也存在一些缺点:(1)软件只能修改微处理器体系结构中可见的寄存器、内存段,无法向软件不能访问的位置注入故障,只能部分模拟真实故障;(2)软件故障注入会不同程度地干扰目标程序甚至要改变目标系统的软件构架,显然增加了软件开发的成本;(3)时间分辨率低,对一些短延时的故障,如总线故障、CPU 故障,不能及时捕获故障的行为特征^[6]. 软件容错技术的这些缺点限制了该方法在实际中的应用.

2.3 仿真故障注入

微处理器在设计过程中涉及不同层次的抽象模型,如门级、电路级、寄存器传输级以及行为级等,基于这些模型并结合仿真工具,可以精确控制故障注入的时间和位置,方便地监控故障行为. 文献[12]将 MOS 管的干扰脉冲模拟成漏极和地之间的恒流源或恒压源,采用 HSPICE 电路级仿真工具分析了一个四位二进制计数器在出现故障时的响应情况. VERIFY 系统在硬件的 VHDL 模型上自动插入故障信号,比较模型在故障注入前后的状态来评估系统的容错能力^[13]. 文献[14]提出的 SINJECT 系统支持多种故障模型评估数字系统的可靠度,能够在门级、RTL 级、行为级、结构级有选择地注入故障,目标系统可以采用 Verilog、VHDL 或两者混合的方式描述,他们用该系统评估了一个 Verilog 语言描述的不可综合的算法处理器和一个 VHDL 语言描述的可综合的 ALU,发现不同的目标系统、不同的故障注入点都会对故障产生很大的影响. 尽管仿真故障注入技术具有良好的可控性和观测度,但是为了得到比较准确的故障分析结果,必须进行大量的故障注入实验,而处理器的上述模型仿真速度极其有限,大量的仿真故障注入常常需要花费很长的时间,另外,现代微处理器片上可以集成上亿个晶体管,对于这样规模的集成电路,HSPICE 等模块级的电路仿真工具显然力不从心. 为了提高仿真故障注入的速度,文献[15]专门开发了可以连续注入多个故障的仿真故障注入工具并采用指令集模拟器取代 RTL 模型作为运行参考.

3 基于功能模型的可靠度评估

基于微处理器功能模型的可靠度评估技术,首

先将微处理器抽象成上层的指令级模型和时序级模型,缩短抽象模型的开发周期,提高仿真速度;接着通过定义微处理器行为级故障模型,再现微处理器底层电路可能发生的故障,控制故障的注入时间和注入位置;最后,将功能模型和故障模型集成在 SimOS 环境中,模拟目标系统. 下面从系统构架、微处理器功能模型、微处理器行为级故障模型和系统特点 4 个方面进行论述.

3.1 系统构架

基于微处理器功能模型的可靠度评估系统由宿主系统和目标系统两部分构成,如图 1 所示. 宿主系统是物理系统,由 Linux 平台、故障产生器、数据分析器等部分组成;目标系统是虚拟系统,是需要研究的容错系统,它由虚拟硬件、目标操作系统和运行负载三部分组成. 在虚拟硬件部分,CPU 模拟器由微处理器功能模型实现;内存系统、硬盘、控制台以及网络部分由 SimOS 模拟^[16]. 目标操作系统是经改造的支持龙芯 1 号微处理结构的 32 位 Linux 内核.

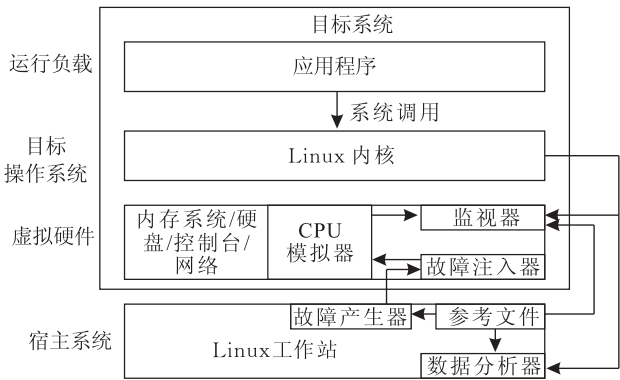


图 1 系统构架框图

系统的运行过程为

- (1) 设定需要注入的故障数;
- (2) 在无故障注入的模式下启动系统,产生参考文件,该文件包括了应用程序执行的开始时刻 `app_begincycle`、结束时刻 `app_endcycle` 以及执行结果 `correct_value`;
- (3) 故障产生器根据参考文件产生故障触发时刻 `fault_trigger_cycle`,选定故障模型 `fault_mode`,要求 $app_begincycle \leq fault_trigger_cycle \leq app_endcycle$, `fault_mode` 为 3.3 节定义的某一种故障模型;
- (4) 再次执行应用程序,当执行到 `fault_trigger_cycle` 时刻,故障注入器根据 `fault_mode` 向 CPU 模拟器注入故障,应用程序继续执行;
- (5) 应用程序在运行结束之前通过自定义的系

统调用将运行状态、运行结果复制到目标操作系统内核缓冲区；

(6) 监视器监控目标系统的运行状态,当发现系统发生异常或应用程序大幅超时,则将监控到的状态写入数据分析文件,当发现应用程序正常退出时,则将监控到的状态以及目标操作系统内核缓冲区的内容一起写入数据分析文件;

(7) 目标系统重启动,转步(3),直到注入设定的故障数;

(8) 数据分析器读取数据分析文件和参考文件中的 correct_value 进行分析统计,输出故障响应信息.

该系统能够连续注入多个故障,实现了故障注入、故障分析的自动化,从而缩短了评估周期.同时,目标系统完全虚拟化,降低了评估费用.

3.2 微处理器功能模型

微处理器功能模型是处理器微结构的一种高层次的抽象,根据模拟的详细程度,它分为指令级模型和时序级模型.指令级模型模拟指令的功能,表现微处理器取指、译码、执行的处理过程.典型的指令级模型有 SimpleScalar 的 Sim-fast 模拟器^[17]、SimOS 的 Mipsy 模拟器^[16].时序级模型除了实现指令的功能操作,还模拟指令的时序行为,SimpleScalar 的 Sim-outorder 模拟器是一个典型的时序级模型.由于功能模型抓住了关键的微结构,忽略了大量的实现细节,因此模型的开发周期大大缩短,模拟速度大大提高,例如,Sim-fast 模拟器的执行速度可达 7MIPS,Sim-outorder 的执行速度为 300KIPS,而一个 Verilog 的仿真器执行一条指令需要 1.65s^[18].

我们在 SimpleScalar 的 Sim-outorder 模拟器的基础上针对龙芯 1 号微处理器结构开发了一个时序级模型 Sim-godson1,详细模拟了龙芯 1 号取指、译码、发射、执行写回及提交五级流水线^[19].通过和龙

芯 1 号 RTL 模型的对比验证,该时序级模型的性能 IPC 偏差小于 5%,运行速度达到每秒 500k 条指令.龙芯 1 号指令级模型 Sim-mipsy 是在 SimOS 的 Mipsy 模拟器上修改得到的,因为 Mipsy 模拟器实现的指令功能和龙芯 1 号的指令比较接近,这样可以减少开发工作量.SimOS 提供了一个开放的 CPU 模块接口,通过增加一些系统态指令和例外处理函数,功能模型可以集成在 SimOS 全系统环境中.

3.3 微处理器行为级故障模型

微处理器行为级故障模型是微处理器底层电路故障在行为级的表现形式.文献[18]采用一种类 C 的寄存器传输语言给处理器建模,定义了一套基于寄存器传输语言的故障模型,他们的实验表明,该故障模型能够再现门级故障模型中 97% 的瞬态故障,同时仿真速度提高了 500 倍. Virginia 大学提出了一个通用微处理器的行为级故障模型^[20],可以对任意寄存器、内存位置、访存地址及指令注入故障,只需要考虑很少的实现细节就可以最大程度地再现处理器内部发生的故障,为系统的可靠度提供最保守的估计.他们在一个 VHDL 描述的 32 位 RISC 处理器的门级模型中注入固定型故障,发现行为级故障模型能够再现 100% 的寄存器文件故障、94.1% 的程序计数器故障、99.2% 的 ALU 故障和 82.4% 的取指译码逻辑故障,只是控制单元的故障再现率较低,只有 5.5%.综合这些研究成果,我们定义了 7 类故障模型来再现处理器流水线上可能发生的故障,本文假设片上缓存及内存单元用 ECC 或奇偶检验码保护,因此不再考虑它们发生的故障.表 1 从故障注入的位置、注入时间及故障类型三个方面刻画了这 7 类故障模型,表 2 是表 1 中相关符号的进一步说明.由于本文的研究对象是瞬态故障,表 2 中用异或操作实现位翻转来模拟瞬态故障,位翻转模型是研究瞬态故障的最常用的故障模型^[21].

表 1 微处理器行为级故障模型

	故障出现时刻	故障定位	故障值
寄存器值模型	$t_{i1} \leq t \leq t_{i2}$	$\forall r, r \in R$ 且 $r \in i$	$(r) = (r) \wedge mask(rw)$
源寄存器选择模型	$t_{i1} \leq t \leq t_{i2}$	$\forall r, r \in R$ 且 $r \in i, r=r1$ 或 $r=r2$	$r = r \wedge mask(irw)$
目标寄存器选择模型	$t_{i1} \leq t \leq t_{i2}$	$\forall r, r \in R$ 且 $r \in i, r=r3$	$r = r \wedge mask(irw)$
PC 模型	$t_{i1} \leq t \leq t_{i2}$	pc	$(pc) = (pc) \wedge mask(rw)$
指令译码模型	$t_{i1} \leq t \leq t_{i2}$	$\exists op, op \in OP$ 且 $op \in i$	$op = op \wedge mask(io\bar{p}w)$
ALU 运算模型	$t_{i1} \leq t \leq t_{i2}$	$\forall u$ 如 $u \in OP$ 且 $u \in i$; 或 $u \in R$ 且 $u \in i$; 或 $u \in i$ 且 $u = imm$	如 $u \in OP$ 或 $u = imm$, 则 $u = u \wedge mask(iurw)$ 如果 $u \in R$, 则 $(u) = (u) \wedge mask(iurw)$ 如 $u \in OP$ 或 $u = imm$, 则 $u = u \wedge mask(iurw)$
总线操作模型	$t_{i1} \leq t \leq t_{i2}$	$\forall u$ 如 $u \in OP$ 且 $u \in i$; 或 $u \in R$ 且 $u \in i$; 或 $u \in i$ 且 $u = imm$ 或 $u = data_i$ 或 $u = addr_i$	如 $u \in R$, 则 $(u) = (u) \wedge mask(iurw)$ 如 $u = data_i$, 则 $u = u \wedge mask(data\bar{r}w)$ 如 $u = addr_i$, 则 $u = u \wedge mask(addr\bar{r}w)$

表 2 符号说明表	
符号	说明
\sim	异或操作
t_{i1}	指令 i 的取指时刻
t_{i2}	指令 i 的提交时刻
$Mask(m)$	位宽 m 的向量 $\langle b_{m-1}, b_{m-2}, \dots, b_1, b_0 \rangle, \exists i, 0 \leq i \leq m-1$, 使得 $b_i = 1$, 且 $\forall j$, 如果 $j \neq i, 0 \leq j \leq m-1$, 则 $b_j = 0$;
R	寄存器 r 的集合, $R = \{r\}$
OP	操作码 op 的集合, $OP = \{op\}$
$data_i$	指令 i 的访存数据
$addr_i$	指令 i 的访存地址
$dataw$	访存数据位宽
$addrw$	访存地址位宽
i	指令 $i = \{op, r1, r2, r3, imm\}$
op	指令 i 的操作码
$r1$	指令 i 的源寄存器 1
$r2$	指令 i 的源寄存器 2
$r3$	指令 i 的目的寄存器
imm	指令 i 的立即数
rw	寄存器 r 的位宽
irw	指令 i 中 r 的编码位宽
(r)	r 的值
pc	程序计数器

定义的故障模型需要映射到功能模型中,我们以龙芯 1 号微处理器为例说明这个映射关系.图 2 是故障模型和指令级模型的映射. PC 模型反映取指过程中 PC 寄存器可能发生的故障;源寄存器选择模型、目标寄存器选择模型以及指令译码模型反映指令在指令寄存器中进行译码时可能发生的故障;ALU 运算模型、总线操作模型反映指令在 ALU 部件或 Load/Store 功能部件中可能发生的故障;寄存器值模型反映指令执行过程中通用寄存器可能发生的故障.图 3 是故障模型和时序模型的映射.发射阶段需要使用源寄存器号访问寄存器文件,可能发生源寄存器号选择故障,读取的寄存器值也有可能发生故障;在提交指令时,可能会向错误的目标寄存器写入错误的值.

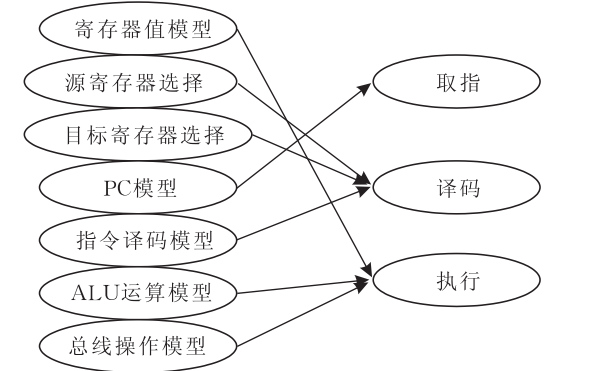


图 2 故障模型向指令级模型的映射

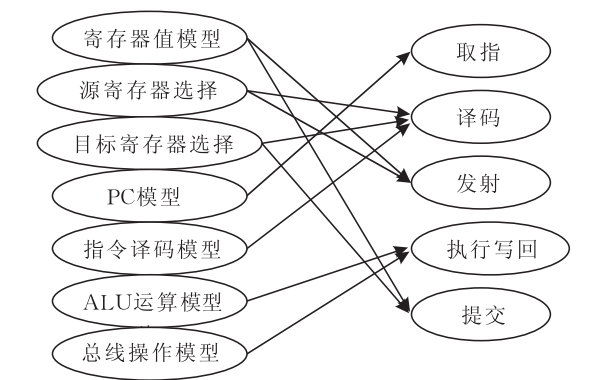


图 3 故障模型向时序模型的映射

在定义的故障模型中,通过设置故障的出现时刻,能够控制故障的注入时间;通过选择不同的故障模型,能够控制故障的注入位置.

3.4 系统特点

表 3 比较了 4 种可靠度评估系统的主要特征. MESSALINE 系统通过芯片的 IC 引脚直接注入故障; SINJECT 系统对 HDL 描述的门级模型和寄存器传输级模型注入故障; Xception 系统采用软件方法注入故障; FMFI 系统是提出的基于微处理器功能模型的可靠度评估系统. 由于 FMFI 系统既不依赖硬件原型,又不需要详细的门级或寄存器传输级模型,同时它还有较高的执行速度,因此它的评估周期最短,能够在系统设计的前期评估容错技术的容错能力,将系统的可靠度情况及时反馈给设计者. FMFI 系统集成在 SimOS 模拟环境中,一方面降低了开发的工作量,另一方面不会伤害宿主机上的软、硬件系统,降低了成本. FMFI 系统通过定义的故障模型控制故障的注入时间和注入位置,具有较好的可控度. FMFI 系统的不足之处在于准确度,这是缩短评估周期、降低成本带来的开销.

表 3 几个典型的可靠度评估系统的特征比较

	评估周期	成本	可控度	准确度
MESSALINE	很长	很高	差	很高
SINJECT	长	高	很好	高
Xception	长	高	好	一般
FMFI	短	低	好	一般

4 实验及分析

本实验在一个 Intel P4 Linux 系统上运行提出的系统评估了 3 种常用的容错技术:软件实现的指令冗余技术^[1]、软件实现的断言技术^[2]和处理器微结构实现的重复取指执行技术. 指令冗余技术和重

复取指执行技术都使用一个 16×16 的整型矩阵乘法程序作为测试程序,断言技术使用的测试程序是一个对 256 个整数进行快速排序的程序,断言为排序后队列满足 $a_i \leq a_{i+1} (0 \leq i < 255)$. 在评估指令冗余技术和断言技术时,系统采用 Sim_mipsy 模拟器,对 3.3 节定义的 7 类故障模型分别注入 2000 个故障;在评估重复取指执行技术时,系统采用模拟重复取指执行行为的模拟器 Sim-godson1,由于该技术假设寄存器文件采用 ECC 校验,所以我们只对除了寄存器值模型之外的 6 类故障模型分别注入 2000 个故障. 注入故障后,系统的响应分为以下 5 类: (1) No Error: 测试程序运行结束且结果正确; (2) Wrong Result: 测试程序运行结束但结果错误; (3) System Detected: 系统检测到异常,这些异常包括读地址异常、写地址异常、非法指令等; (4) Time Out: 规定的时间内,程序没有执行结束; (5) Fault Detected: 容错技术检测到异常. Wrong Result 是系统潜在的重大故障,本文将故障覆盖率定义为 $(1 - Wrong\ Result) \times 100\%$.

4.1 指令冗余技术的实验结果

图 4(a)和图 4(b)分别是矩阵乘法程序在采用指令冗余技术前后注入故障的结果. 从图中可以看出,容错前 Wrong Result 的百分比为 41.8%,采用容错技术后 Wrong Result 百分比下降为 0.5%,故障覆盖率为 99.5%.

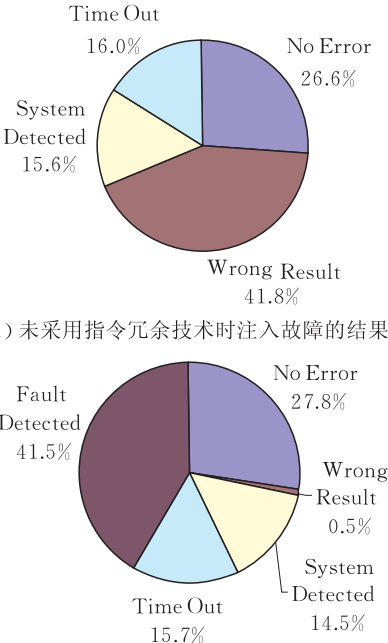


图 4 矩阵乘法程序在采用指令冗余技术前后注入故障的结果

表 4 是矩阵乘法程序在采用指令冗余技术时的故障延时. 从表中可以看出,指令冗余技术的故障延时平均为 9530 个时钟周期,占程序运行总时间的 1.8%. 未容错的矩阵乘法程序运行时间为 141375 个时钟周期,容错后运行时间为 541081 个时钟周期,指令冗余技术导致的 CPU 时间开销为 280%.

表 4 指令冗余技术的故障延时

模型	故障延时/时钟周期
寄存器值模型	9733
源寄存器选择模型	9423
目标寄存器选择模型	9399
PC 模型	9394
指令译码模型	9395
ALU 运算模型	9564
总线操作模型	9801
平均延时	9530

4.2 断言技术的实验结果

图 5(a)和图 5(b)分别是快速排序程序在采用断言技术前后注入故障的结果. 从图中可以看出,采用断言技术后 Wrong Result 的百分比从 8.1%下降到 1.1%,故障覆盖率为 98.9%.

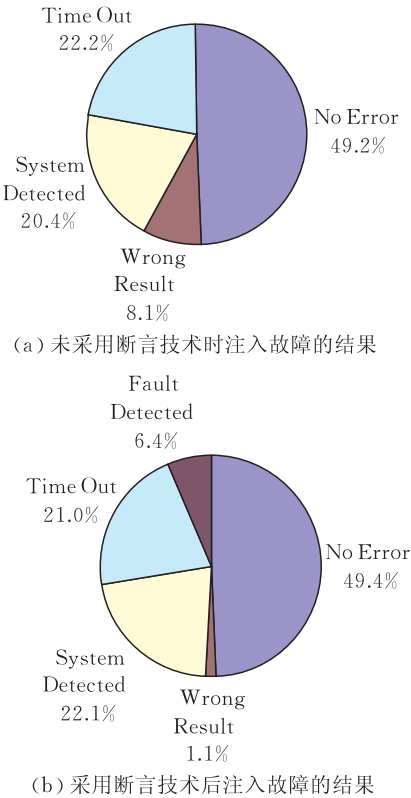


图 5 快速排序程序在采用断言技术前后注入故障的结果

表 5 是快速排序程序在采用断言技术时的故障延时. 从表中可以看出断言技术的故障延时平均为

181819 个时钟周期,占程序运行总时间的 72.6%。未容错的快速排序程序运行时间为 244768 个时钟周期,容错后的运行时间为 250384 个时钟周期,断言技术导致的 CPU 时间开销为 2.3%。

表 5 断言技术的故障延时	
模型	故障延时/时钟周期
寄存器值模型	154128
源寄存器选择模型	168656
目标寄存器选择模型	220519
PC 模型	150902
指令译码模型	215889
ALU 运算模型	161569
总线操作模型	201073
平均延时	181819

4.3 重复取指执行技术的实验结果

图 6(a)和图 6(b)分别是矩阵乘法程序在采用重复取指执行技术前后注入故障的结果。从图 6 中可以看出,采用重复取指执行技术后错误结果的百分比从 42.6%下降到 2.5%,故障覆盖率为 97.5%。

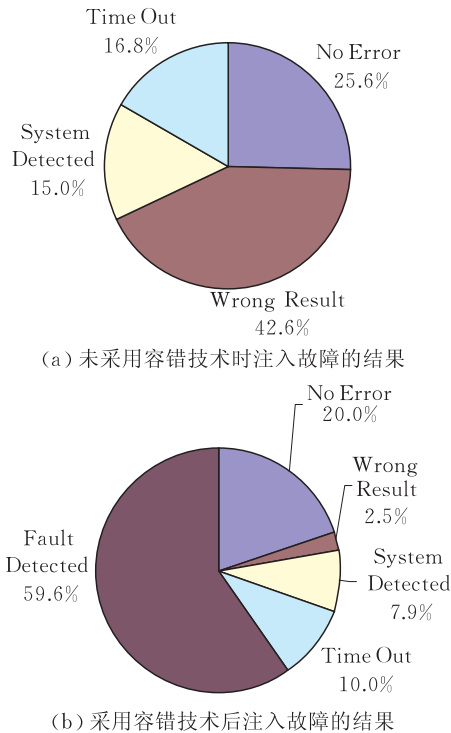


图 6 矩阵乘法程序在采用重复取指执行技术前后注入故障的结果

重复取指执行技术的故障延时平均为 16 个时钟周期,占程序运行总时间的 0.0%。未容错的矩阵乘法程序运行时间为 1042570 个时钟周期,容错后运行时间为 1300368 个时钟周期,重复取指执行技术导致的 CPU 时间开销为 24.7%。

4.4 实验结果分析

首先,我们对比指令冗余技术和断言技术的实验结果。指令冗余技术的故障覆盖率为 99.5%,故障延时平均为 9530 个时钟周期,CPU 时间开销为 280%;断言技术故障覆盖率为 98.9%,故障延时平均为 181819 个时钟周期,CPU 时间开销为 2.3%。指令冗余技术有较高的故障覆盖率,较低的故障延时,但是 CPU 时间开销较大。这是因为指令冗余技术复制了矩阵乘法程序中的每个变量和每条指令,还对每个变量及相应副本增加了比较操作和跳转指令,当发现变量和它的副本之间数据不一致时,程序马上报错;而断言技术则在快速排序完成后,再对排序结果进行比较检错。相对而言,指令冗余技术冗余的粒度较小,冗余的指令更多,所以它的故障覆盖率更高,故障延时较小,时间开销也较大。

其次,我们对比指令冗余技术和重复取指执行技术的实验结果。指令冗余技术的故障覆盖率为 99.5%,故障延时平均为 9530 个时钟周期,CPU 时间开销为 280%;重复取指执行技术故障覆盖率为 97.5%,故障延时平均为 16 个时钟周期,CPU 时间开销为 24.7%。重复取指执行技术以硬件方式实现了每条指令的冗余执行,冗余的粒度更小,同时它消除了指令冗余技术额外增加的比较指令和跳转指令,因此它的故障延时更小,CPU 时间开销更低。它的故障覆盖率不高,主要原因是该技术没有克服 PC 寄存器故障。如果 PC 寄存器采用 ECC 校验技术或在指令提交时检测该指令的 PC 值是否一致,重复取指执行技术的故障覆盖率会提高到 100%,此时故障分布如图 7 所示。

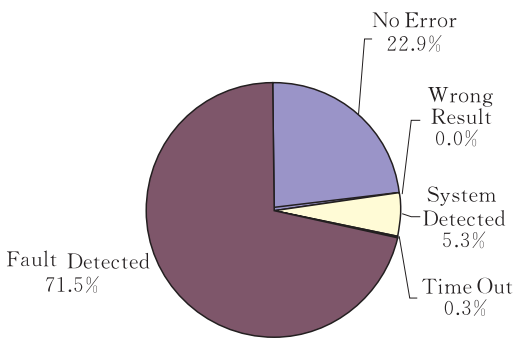


图 7 克服 PC 故障之后重复取指执行技术的故障分布

5 结 论

本文首先提出了一个基于微处理器功能模型的

可靠度评估技术,将处理器微结构抽象成指令级模型和时序级模型,减小了抽象模型的开发时间,提高了仿真速度;然后定义了一个微处理器行为级故障模型,再现了微处理器底层电路可能发生的故障,通过该故障模型可以方便地控制故障的注入时间和注入位置;最后在 SimOS 全系统模拟环境中,实现了一个纯软件的故障评估系统,和现有可靠度评估系统相比,本系统可靠度评估周期短,控制方便,成本低廉. 使用该系统评估了软件实现的指令冗余技术、软件实现的断言技术和硬件实现的重复取指执行技术,实验结果表明该系统的评估是合理的.

参 考 文 献

- [1] Rebaudengo M, Reorda M S et al. Soft-error detection through software fault-tolerance techniques//Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. Albuquerque, New Mexico, 1999: 210-218
- [2] Andrews D M. Using executable assertions for testing and fault tolerance//Proceedings of the 9th on Fault-Tolerant Computing. Madison, WI, 1979: 102-105
- [3] Calin T, Nicolaidis M, Velazco R. Upset hardened memory design for submicron CMOS technology. IEEE Transactions on Nuclear Science, 1996, 43(6): 2874-2878
- [4] Ray J, Hoe J, Falsafi B. Dual use of superscalar datapath for transient fault detection and recovery//Proceedings of the 34th Annual IEEE/ACM International Symposium on Microarchitecture. Austin, Texas, USA, 2001: 214-224
- [5] Constantinescu C. Dependability evaluation of a fault-tolerant processor by GSPN modeling. IEEE Transactions on Reliability, 2005, 54(3): 468-474
- [6] Hsueh M, Tsai T K, Iyer R K. Fault injection techniques and tools. IEEE Computer, 1997, 30(4): 75-82
- [7] Arlat J et al. Fault injection for dependability validation: A methodology and some applications. IEEE Transactions on Software Engineering, 1990, 16(2): 166-182
- [8] Madeira H, Rela M et al. FIFLE: A general purpose pin-level fault injector//Proceedings of the 1st European Dependable Computing Conference. Berlin, Germany, 1994: 199-216
- [9] Gunnetlo O, Karlsson J, Tonn J. Evaluation of error detection schemes using fault injection by heavy-ion radiation//Proceedings of the 19th Fault-Tolerant Computing. Chicago, USA, 1989: 340-347
- [10] Kanavati G A, Kanawati N A, Abraham J A. FERRARI: A flexible software-based fault and error injection system. IEEE Transactions on Computers, 1995, 44(2): 248-260
- [11] Carreira J, Madeira H, Silva J G. Xception: A technique for the experimental evaluation of dependability in modern computers. IEEE Transactions on Software Engineering, 1998, 24(2): 125-136
- [12] Maheshwari A, Koren I et al. Technique for transient fault sensitivity analysis and reduction in VLSI circuits//Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. Boston, Massachusetts, 2003: 597-604
- [13] Sieh V, Tschache O, Balbach F. VERIFY: Evaluation of reliability using VHDL-models with embedded fault descriptions//Proceedings of the 27th Annual International Symposium on Fault-Tolerant Computing. Seattle, Washington, USA, 1997: 32-36
- [14] Zarandi H R, Miremadi S G, Ejlali A. Dependability analysis using a fault injection tool based on synthesizability of HDL models//Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerant in VLSI Systems. Boston, Massachusetts, 2003: 485-492
- [15] Kim S, Somani A K. Soft error sensitivity characterization for microprocessor dependability enhancement strategy//Proceedings of the International Conference on Dependable Systems and Networks. Washington DC, USA, 2002: 416-425
- [16] Rosenblum M, Herrod S A, Witchel E et al. Complete computer system simulation: The SimOS approach. IEEE Concurrency, 1995, 3(4): 34-43
- [17] Austin T, Larson E, Ernst D. SimpleScalar: An infrastructure for computer system modeling. Computer, 2002, 35(2): 59-67
- [18] Yount C R, Daniel P S. A methodology for the rapid injection of transient hardware errors. IEEE Transactions on Computers, 1996, 38(4): 881-891
- [19] Hu Wei-Wu, Tang Zhi-Min. Architecture of the Godson 1 processor. Chinese Journal of Computers, 2003, 26(4): 385-396(in Chinese)
(胡伟武,唐志敏. 龙芯 1 号处理器结构设计. 计算机学报, 2003, 26(4): 385-396)
- [20] Cutright E, Delong T, Johnson B. Generic processor fault model. McCormick Road, Charlottesville: University of Virginia Center for Safety-Critical Systems: Technical Report UVA-CSCS-NSE-004, 2002
- [21] Sosnowski J. Transient fault tolerance in digital systems. IEEE Micro, 1994, 14(1): 24-35



ZHANG Shi-Jian, born in 1975, Ph. D. candidate. His main research interests include high reliable microarchitecture and high-performance computer system.

XU Tong, born in 1975, Ph. D. candidate, associate researcher. His main research interests focus on IC design and

embedded application.

ZHANG Long-Bing, born in 1974, Ph. D. , associate researcher. His main research interests focus on high-performance computer architecture.

HU Wei-Wu, born in 1968, Ph. D. , professor, Ph. D. supervisor. His main research interests include high-performance computer architecture, MPP system and VLSI design.

Background

With the widespread adoption of embedded microprocessor-based systems for safety critical applications, fault-tolerant mechanisms have been built into application programs and microprocessors. How to evaluate their cost and effectiveness is a difficult and challenging task. In this paper, a new fault injection technique based on microprocessor function model is proposed, which has the capability to assess system dependability in an early phase. The authors' work is an im-

portant component of the research on low power and high reliability for the Godson1 microprocessor. The project is supported by the National Basic Research Program (973 Program) of China under grant No.2005CB321600 and the National Natural Science Foundation of China under grant Nos.60325205, 60673146 and 60703017 and the Beijing Natural Science Foundation under grant No.4072024.