计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
45(2):358~366, 2008

高性能处理器的差错校正技术

王 真 江建慧 员春欣
(同济大学嵌入式系统与服务计算教育部重点实验室 上海 201804)
(同济大学计算机科学与技术系 上海 201804)
(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)
(wangzhenqq@hotmail.com)

Error-Correcting Techniques for High-Performance Processors

Wang Zhen, Jiang Jianhui, and Yuan Chunxin
(*Ministry of Education Key Laboratory of Embedded Systems and Service Computing, Tongji University, Shanghai* 201804)
(*Department of Computer Science and Technology, Tongji University, Shanghai* 201804)
(*Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100080)

Abstract The downscaling of feature size of CMOS technology results in faster transistors and lower supply voltages. This trend contributes to the overall performance improvement of integrated circuits, but it also brings more challenges to the reliability of complex circuits like microprocessors. Accordingly, the fault-tolerance design of high-performance processors becomes more and more important. Till now much work has been done for error detection and correction in processors. Some novel fault tolerant microprocessor architectures are proposed recently, such as the simultaneously and redundantly threaded processors with recovery architecture. In this paper, a comprehensive survey on conventional and up-to-date error correction techniques for high-performance processors is given. A novel taxonomy is presented, by which the fault tolerant techniques for processors are categorized into clock-level error recovery, instruction-level error recovery, thread-level error recovery and reconfiguration. Many microarchitecture schemes, prototype systems and industrial products are analyzed and detailed fault tolerant strategies and schedule algorithms are compared. It is shown that for modern processors characterized by chip multiprocessor and/or simultaneous multithreading, the reliability is mostly improved by the fault-tolerance techniques based on inherent replicated hardware resources that are designed for improving performance.

Key words high performance processor; error correction; error control code; redundancy; reconfiguration

摘 要 随着芯片密度的不断增加和对可靠性要求的不断提高,高性能处理器的容错设计越来越受到关注.对近年来高性能处理器的差错校正技术进行了分析和比较,它们被分为时钟级差错恢复、指令级差错恢复、线程级差错恢复以及重构等4类,研究对象包括研究方案、原型和产品.研究结果表明,以片上多处理器和/或同时多线程为特征的高性能处理器除了沿用传统的容错技术之外,多以固有的、旨在为改善性能而重复设置的硬件资源为基础来设计容错机制和调度方案.

关键词 高性能处理器;差错校正;差错控制码;冗余;重构

中图法分类号 TP302.8

收稿日期:2006-11-08;**修回日期**:2007-09-03
基金项目:国家"九七三"重点基础研究发展规划基金项目(2005CB321604);国家自然科学基金项目(90207021)

随着芯片密度的不断增加和对可靠性要求的不断提高,高性能处理器的容错设计越来越受到关注.采用了片上多处理器(chip multiprocessor, CMP)和同时多线程(simultaneous multithreading, SMT)结构的高性能处理器固有的硬件冗余资源,为实现容错提供了物质基础.针对数据通路和控制器,更加重视电路级容错技术的应用,如 ALU、阵列部件、Cache 和主存的容错设计.所采用的基本技术有差错控制码、冗余执行、卷回等,为了提高芯片级系统的成品率、运行时的容错性能,以硬件冗余为基础的重构技术也被采用.

高性能处理器在流水线中增设了硬件开销适中的检错和恢复两个功能级,目的是使容错与指令的执行相重叠,尽量避免处理器性能的下降.另外,在容错处理器的设计中,非常重视对已有机构的重复利用,这些机构包括针对转移预测错误的快速恢复机构、针对推测执行错误的推测执行异常恢复机构[1-2],它们都可以直接用来帮助实现容错.

已有的容错技术可以概括为三大类:一是基于时间冗余的微指令级、指令级、线程级、程序级的后向恢复方案,基本方法有重试、微卷回等;二是基于硬件冗余的多模表决前向恢复方案,基本技术运用资源复制和结果比较、表决;三是用来保证数据完整性的纠错码,例如汉明码.

本文主要对近年来的高性能处理器所采取的容错技术进行了全面的分析和总结,分为时钟级差错恢复、指令级差错恢复、线程级差错恢复以及重构等几大类.

1 时钟级差错恢复

1.1 微卷回

并发差错检测通常需要在每个时钟周期检测系统中各个模块的输出,系统需要设置专门的校验电路,这或多或少会导致时间开销的增大.为了弥补这种缺陷,1988 年 Tamir 和 Tremblay 等人提出了微卷回(micro-rollback)恢复方案[3-4].处理器采用双模冗余比较结构来检测差错,当处理器检测到差错时,程序后退几个时钟周期(称为微卷回距离),回到先前到达过的某个一致系统状态.为此必须保存每个时钟周期边界上的系统状态的快照(snapshot),包括处理器上存储有用信息的所有存储单元,即 PC、程序状态字、指令寄存器、寄存器堆等的内容、某些流水线锁存器和寄存器的内容以及 Cache 的内容.

对于超大规模(VLSI)系统而言,微卷回技术对瞬时故障是一种非常有效的恢复方法.20 世纪 90 年代前半期,Raghavendrn 和 Lursinsap 对微卷回技术进行了改进,所提出的自动微卷回自恢复综合方法通过设定约束条件,缩减了硬件开销和时间开销[5-6].Orailoglu 等人在 1996 年提出了一个自恢复系统 SYNCERE,也是通过设置校验点和卷回技术进行差错恢复[7].1999 年,Blough 等人提出的可恢复 VLSI 高层次综合算法利用图论进行系统建模,用最短路径算法设置校验点[8],通过运行基准电路的 CPU 耗时发现,优化算法在 Sun SparcStation 10 上实现最多耗费 10min 左右,而之前的算法在 Sun SparcStation 2 上的最好结果也需要几个小时.

1.2 检错纠错码

由于 Cache 占有较大的处理器芯片面积(超过 60%),它们更易受到宇宙射线的影响,而且错误的传播性强,通常采用检错纠错码.如 L1 I-Cache 和 D-Cache 通常采用奇偶校验码,L2/L3 Cache 主要采用汉明码、单差错校正双差错检测(SEC-DED)码.Itanium[9],IBM G4/G5/G6 处理器[10-13],POWER4[14],POWER5[15],Sun SPARC[16-17] 等产品都在 Cache 中采用了差错控制码.欧洲航空局在 1998 年启动的 LEON 处理机(其设计与 SPARC V8 兼容)项目[16]中,为了防止发生多差错,在 L1 Cache 中采用了两位奇偶校验位.而以 64 位超标量 SPARC V9 为基础的 HAL 计算机[17],其存储器管理单元的各种转换表使用了几种不同的并发差错检测和校正机制,如双差错校正三差错检测码.然而,Cache 因采用字节奇偶校验码和 SEC-DED 码而需要增加额外的空间(约 12.5%),还增大了操作延迟和功耗(SEC-DED 码比奇偶校码更高).在 D-Cache 中使用块复制技术,引入预测策略[14]缓解了这一问题.

寄存器的容错机制也结合了差错校验码.在基于 SPARC V8 体系结构的 LEON 处理机中[16],为了检测寄存器的单粒子翻转(single event upset, SEU)错误使用了奇偶校验码(含 1 位和 2 位校验位)、(32, 7)BCH 码.在流水线的写阶段生成校验位,并与对应的数据一起写入存储器或寄存器.差错校验在执行阶段进行,与指令执行并行.若检测到可纠正的差错,则清除流水线,校正数据差错后写回寄存器堆,然后重新启动流水线,如图 1 所示.若检测到不可校正的差错,则生成一个寄存器错误陷阱.这种方案使用正常陷阱的处理逻辑,流水线重启开销较小.除此之外,寄存器的检错和纠错还可以使用交叉奇偶校验码[18].

FETCH	INST.1	INST.2	INST.3	INST.4	INST.5	FLUSH	INST.2	INST.3
DECODE		INST.1	INST.2	INST.3	INST.4	FLUSH		INST.2
EXECUTE			INST.1	CHECK	INST.3	FLUSH		
MEMORY				INST.1	RECOV	FLUSH		
WRITE					INST.1	UPDATE		

Fig. 1 Pipeline operations during error correction.

图1 校正差错期间的流水线操作

1.3 多模表决冗余

LEON处理机[16]含有大约2500个触发器.为了防护SEU错误,每个寄存器用三模表决冗余实现.触发器在每个时钟周期进行错误检测,寄存器中的任一SEU错误都能够在一个时钟周期内消除,表决器的输出始终保持正确值.

2 指令级差错恢复

指令级容错结构主要包括双模冗余锁步结构、校验核结构、超标量处理器改进结构以及数据通路的实时容错结构,它们主要以指令或指令流为单位通过重复执行实现检错和纠错.

2.1 双模冗余锁步结构

IBM公司在1999年推出的S/390 G5处理器的结构如图2所示[10-13].它由两个完全相同的指令单元(I单元)和执行单元(E单元)、一个L1 Cache和恢复单元(R单元)组成.在每个时钟周期,R单元和L1 Cache交叉比较来自两个I单元和E单元的结果进行差错检测,同时R单元记录下校验点状态以备恢复.如果R单元或L1 Cache检测到错误,处理器将自动进入错误恢复模式,恢复步骤是R单元先冻结校验点状态,复位I单元、E单元和L1 Cache,同时差错控制码校正并更新寄存器值;再次读取R单元,若有可校正硬差错,则终止操作并重启系统.

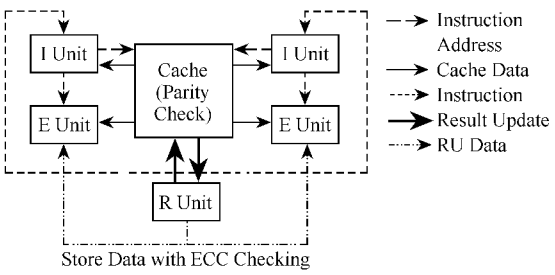

Fig. 2 Logical structure of G5 processor.

图2 G5处理器的逻辑结构

这种双模冗余锁步结构在IBM eServer z990微处理器中得到进一步发展[19],z990微处理器在单块芯片上置入了两个结构相同的完整处理器核,每个核都是如图2所示的双模冗余结构,分别在流水线的后几个阶段进行差错校验与恢复.

2.2 校验核结构

校验核结构以多核处理器为基础,目前主要采用双核(分别是核处理器和校验处理器),程序的每条指令相继在两个处理器核上执行,得到的两个结果由校验核处理器进行比较,纠错后再提交.校验核结构的一个典型例子是DIVA微处理器[20].

在DIVA处理器的基础上又提出了一些改进的方案,如选择性串联重复[21]和共享资源(SHared REsource,SHREC)串联重复[22].选择性串联重复结构的改进之处在于将处于空闲状态的处理器用做验证器.当指令准备提交时,验证器再次执行该指令,以检测错误.它由P流水线和V流水线通过重新执行队列(REQ)级联而成,其中V流水线(验证核)是P流水线的一部分,无故障的条件下它们产生相同的结果.如果检测到故障,两条流水线均被清除,并且调用子例程完成恢复.SHREC结构基于传统的超标量微结构,实现了指令非对称冗余地执行,在无需扩大指令发射队列和重排序缓冲器(ROB)的条件下减缓了共享它们的压力.实验数据表明,SHREC结构的平均性能下降得到了改善(整数基准程序是4%,浮点SPEC2K基准程序是15%).

2.3 超标量处理器的改进结构

2.3.1 对ROB的改进[22]

该方案要求每一条指令执行两次,但并不在ROB中形成两个条目,因此它不同于复制指令.ROB增加了下列标志:指令对第1次执行已经准备就绪(R)、指令对第2次执行已经准备就绪(S)、两次操作情况的结果一致且操作对提交已经准备就绪(D).它们用来指示指令是第1次执行还是第2次执行.LOAD或STORE指令与上述附加信息无关,这类指令的每个操作不需要执行两次.当且仅当同一条指令两次执行的结果一致,所有前面的结果已提交,该指令才可以退役.而当同一条指令两次执

行的结果不一致时,需要返回到该指令的一次执行就绪(R)态,直到它的结果被确认.若指令已经推测执行,但发现推测是错误的,那么系统将卷回到最后提交点.如果执行单元被反复发现错误,同时又有足够的执行单元可用,那么系统可以把该单元从活动表中删除掉.

2.3.2　对指令重发射机构的改进[23-31]

指令重发射机构以寄存器更新单元(register update unit,RUU)为基础,与基于 ROB 的故障检测类似,在 RUU 中标记两次执行结果,若检测到一个瞬时故障,则启动指令重发射机制进行恢复.改进前的 RUU 条目中仅通过 ToReissue 位标识指令是一次执行还是二次执行,为了利于故障恢复,RUU 又增加了一个 Reissued 位,如图 3 所示.若检测到瞬时故障,Reissued 位将给出指示,表明需再次重发相应的指令实现故障恢复.

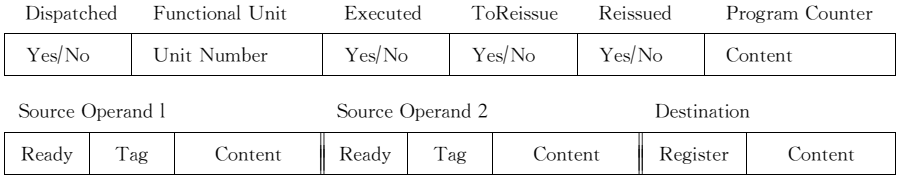

Fig. 3　Instruction format for improved RUU.

图 3　改进 RUU 的指令格式

在超标量结构上,有两种透明硬件恢复方案.方案 1 是对错误分支预测的恢复方案,它需要清除出错指令后的所有指令(即清除流水线),然后在对应的指令上重启,因此性能损失很大.方案 2 是对错误推测执行的恢复方案,采取部分选择性指令重发机制.即对与出错指令数据相关的指令进行重发,比方案 1 降低了性能损失.实验表明,在 4 路和 8 路的超标量处理器上分别采取这种容错机制性能损失分别是 12.5% 和 20.8%.若再使用指令重用技术还可进一步降低性能损失[32].

2.3.3　对整体结构的改进

2001 年 Ray 等人提出了一个能够实现瞬时故障检测和恢复的超标量处理器[21],它对具有推测机制、乱序执行的超标量处理器进行了改进.数据通路增设了指令复制逻辑、结果比较和提交逻辑.指令译码阶段暂时将单指令流变成 2 个或 3 个冗余线程,利用超标量结构所具有的多条指令同时译码和派送功能将一条指令的多个副本送到 ROB 中.在更新提交的程序状态以前,冗余线程重新合并成单一的指令流.当指令的所有副本均执行结束后,对它们进行交叉校验.如果一致,它们就正常离开 ROB,否则,将进行差错恢复.对转移类指令,一旦其副本与预计的转移目标不一致,将立即触发回退.只有预测正确的转移指令的副本才能到达提交阶段.为了保证控制流的正确性,保存最后提交指令的下一条指令的 PC 值.每一条退役(即执行完尚未提交)指令的 PC 值必须与最后提交指令的下一指令的 PC 值进行校验,若检测到不一致,则通过读取提交指令的下一指令的 PC 值再次启动执行.利用 SPEC95 和 SPEC2000 进行模拟实验,无故障情形下处理器的吞吐量降低了 2%～45%.

2002 年 Lai 等人提出 Ditto 处理器方案[33].Ditto 处理器在超标量处理器结构上附加了延迟缓冲器和验证逻辑,并对取指单元、译码单元、ROB、提交逻辑单元进行了改进,以适应复制指令流.与其他指令复制机制不同,这种处理器将长延迟操作和短延迟操作分开处理.对于长延迟操作进行推测重复执行,比较检测结果但不提交;在未被预测的指令执行退出前进行复制,再送往流水线头部重复执行,但只需验证重命名操作.对于短延迟操作,全部进行复制后再执行,并进行比较,若发现不一致,则表示发生瞬时故障.

总之,基于超标量处理器的容错结构就是要充分利用芯片内已有的硬件冗余资源实现指令的重复执行,这些资源包括回退机制、错误预测恢复机制和指令重发射机制.

2.4　数据通路的实时容错结构[34]

该实时差错恢复方案主要采用了三模表决和重试技术,同时结合了调度算法.在 VLIW 处理器内核中设置了 3 个相同的 ALU 模块和一个冗余模块,这种结构最多可以有 3 条运算指令同时发射.对每条执行的指令,一旦比较检测出差错立即重试恢复.一个周期同时发射一条或两条指令,若同时发射 3 条,则调度为 2 次顺序执行,即增加一个 ALU 周期,第 1 次发射一条指令,第 2 次同时发射 2 条.

实时容错结合调度算法展开:每个模块都设有一个差错指示位和状态标识 S_i($i=0,1,2,3$ 用以表示有 i 个模块出错).假设系统的初始状态为 S_0,执

行下面两种情况:1)一次发射一条指令,则直接在4个模块中选3个组成TMR形式执行.若检测出一个模块出错,状态转为S_1;若3个模块的执行结果各不相同,则在4个模块中重新选3个执行.2)同时发射两条指令,则每两个模块执行一条指令.若两条指令都出错,则重新顺序执行,采取操作1);若只有一条指令出错,则将出错指令调度到未出错的两个模块上重试,若无错,将正确结果与上次错误结果比较,据此转为S_1或S_2,若再次出错,则采取操作1).在整个执行过程中,根据单个模块标识的差错指示标志和所有模块所处的状态,由控制单元来对应算法中的不同情况调度相应恢复机制.

3 线程级差错恢复

SMT和CMP结构主要用于提高性能,但由于它能提供线程级冗余,所以可以将这种技术用于瞬时故障和永久故障的检错和纠错.目前,主要结构包括主动流/冗余流同时多线程(AR-SMT)结构[35]、滑流(slipstream)结构[36]、同时-冗余线程(simultaneous and redundantly threaded,SRT)结构[37]、带恢复机制的同时冗余线程(simultaneously and redundantly threaded processors with recovery,SRTR)结构[38]、芯片级冗余线程(chip-level redundant threading,CRT)结构[39]、带恢复机制的芯片级冗余线程(CRTR)结构[40]等.

3.1 基于SMT的结构

AR-SMT,SRT,SRTR等结构是在SMT技术的支撑下实现的.SRT结构[37]由AR-SMT结构改进得到,主要用于差错检测.与AR-SMT结构不同的是,它提出复制域的概念来划定冗余执行的范围,即从功能单元和指令类型两个层面界定.在复制域之内的通过冗余执行保证故障覆盖率,而在此之外的采取其他技术,例如,存储器和RAID不在复制域内则用差错控制码.对两个线程执行结果进行比较时,从复制范围输出的结果需要进行比较,而不在这个范围的结果则不需要比较.这就使系统性能的损失进一步降低,SPEC95的实验结果表明,与同等规模片上硬件冗余方案相比,SRT处理器的性能平均提升了16%.

SRTR结构在SRT处理器的基础上加入了恢复机制[38].SRT结构的首次执行的非存储指令可以在进行故障校验之前提交,但SRTR不允许任何指令在故障检测(比较)之前提交,因为一旦提交后,有故障的指令将不能被撤销.为了减小带宽压力,加快运算速度,SRTR提供具有转移预测的跟踪线程,用一个队列结构存储待校验的寄存器值,用基于相关的校验删除方法减少检测量.SPEC95定点浮点实验数据显示,SRTR结构比SRT性能下降了1%~7%.

3.2 基于SMT及CMP的结构

将CMP结构用于检错纠错是利用了它的硬件冗余特点,基于CMP的容错方案往往通过在不同的处理器核上运行同一个程序的两个版本,通过比较结果来实现检错,进而对差错进行恢复.后来的CMP中都集成了SMT技术,可以更加充分地利用芯片上的资源来实现容错,下面的CRT和CRTR技术就是基于集成了SMT的多核处理器结构.

在滑流处理机上,同一段程序的两个不同版本A-流和R-流同时运行在CMP的不同核上或者单处理器的SMT上[36].R-流是程序的原版本,A-流是经过处理之后的缩减版本.通过比较先执行完成的A-流和R-流可以检测瞬时故障.这种技术的问题在于,当发现比较结果不一致时,它无法确定这究竟是故障还是指令误删(生成A-流时需要移除指令)所造成的,因此只能通过重启实施恢复操作.滑流处理机使用差错控制码防护R流寄存器堆和D-Cache.

CRT结构把SRT结构的检错方法用在了CMP上[39].一个应用程序被复制为完全相同的独立的前导(leading)线程和跟踪(trailing)线程,前导线程的执行领先跟踪线程一个空闲期(即利用时间冗余).比较两个线程的执行结果以检测瞬时故障.在CMP中,前导线程和跟踪线程在不同的处理器上执行(即利用空间冗余),以达到负载平衡并降低污染两个线程的故障概率.CRT与SRT的区别之一是前者通过空间相异性和时间相异性的组合,以提高重复执行程序环境的相异性.另一个区别就是为了比较两个线程的执行结果(包括转移结果、装载值和存储值),CRT把数值从执行前导线程的处理器传送到执行跟踪线程的处理器.为了减小通信延迟,跟踪线程将运行于执行前导线程处理器的附近处理器上.

在CRT基础上提出的CRTR结构中[40],前导线程在校验前提交,而跟踪线程必须在校验后提交,利用跟踪线程的状态可以进行故障恢复.这种提交策略允许CRTR结构将长的空闲期用于隐藏处理器之间的延迟.由跟踪处理器保存出错指令的PC

值,一旦比较检测出错便重复执行该指令.为了不污染保存和恢复过程本身,恢复线程将寄存器状态和PC值冗余地保存到存储器单元的不同集合中.与滑流技术相比,CRTR结构的优越性体现在:1)前者不能保证发射指令地址的一致性,因为存储器地址可能在首线程和尾线程载入数据的间隔内被更改,而CRTR利用载入值队列保证了两线程的一致性;2)滑流允许首线程检测前提交,在存储器中提供备份,导致占用大量存储空间,降低了性能.而CRTR不需要存储空间复制,存储指令提交前先检测.

芯片级冗余线程也可以在具有多个处理单元的微处理器上实现,例如多标量处理器已经有了类似的容错方案[41].

3.3　无需卷回的恢复结构

卷回重算方法虽然已很成熟,但由于瞬时故障将引入一个不可预测的延迟,使得卷回的完成时间难以预测,这在某些实时系统中是不允许的.另外,许多应用系统可接受的最长错误延迟时间常常小于瞬时故障或间歇故障的持续时间,结果就容易把瞬时故障归为永久性故障,并过早地启动备份或停机[42].

为了避免这些问题,开发了基于校验点的无需卷回的故障恢复方法[42],其原理如图4所示.对应于每个校验点,有两个可能的线程集合,分别对应于正常模式和恢复模式.如果校验点没有发现差错,则运行正常模式线程,若发现查错,则转到恢复模式线程.在正常模式下,每个计算执行两个线程——计算线程和复制线程,然后比较执行结果.在恢复模式下必须先执行重算线程,该线程对应于校验点前的线程中发生错误的那一部分,然后执行紧跟其后的后继计算线程和后继复制线程.重算只能使用未被错误部分使用过的硬件单元,而且要在校验点间隔之内完成.

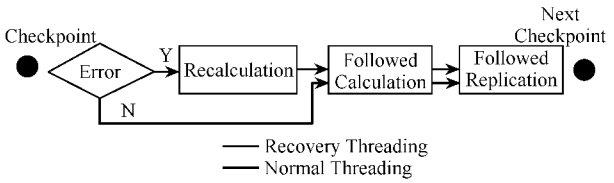

Fig. 4　Recovery structure without rollback.

图4　无需卷回的恢复结构

除了串行恢复方法之外,还可以采用并行恢复方法,即正常模式下的后继计算与前一过程在恢复模式下的重算并行执行.实验表明,并行方法比串行方法平均运行时间缩短30%,差错潜伏期缩短67%.然而并行方法在出现多故障和后继指令与前一过程存在数据相关时具有局限性,同时很可能引起功能单元的资源冲突.

4　重　　构

重构技术主要用以校正硬故障所导致的差错,结合了前向恢复和后向恢复思想,典型的重构技术有内建自修复(built-in self-repair,BISR)、面向阵列结构的备份行或列等.重构技术大多用于存储器,但也有用于处理器的.

4.1　Cache上的重构

Cache重构的一个应用是IBM的POWER4系统[14].一般情况下,Cache主要采用差错控制码防护,当出错次数超出某一阈值时,则认为是发生了硬故障,此时将启动重构操作,但处理器操作却不被中断.

在POWER4处理机的存储器中,L1和L2 Cache、L2和L3的目录(地址映象表)设置有备份,它们通过可编程的引导逻辑来控制.引导逻辑在处理器初始化期间被激活.L1/L2设有冗余位,它们之间采取写直达式设计,L1 Cache中的数据在L2 Cache都有副本.访问L1出现差错时,由系统固件控制完成恢复,使出错的L1 Cache线无效同时从L2 Cache重新取指令或数据.在L2执行ECC控制.而L3 Cache的冗余建立在Cache的行级粒度上,能够实现对故障行的删除.当可校正差错超出了某一阈值,激活L3 Cache的动态行删除功能,最多可实现2次删除达到重构.

对软错误率的分析表明,在采用SEC-DED码的超大容量Cache中(100MB~1000MB),通过周期性地访问Cache,并移去存在单差错的块,可以把瞬时双差错率降低到一个可以接受的范围.

4.2　数据通路上的重构

1993年,Guerra等人提出通过BISR来重构数据通路的方法[43].BISR最直接的办法就是对每个硬件单元都进行备份,但这将带来难以承受的硬件开销.于是他们提出了两种改进方案,一种方案是基于资源的优化分配与调度思想,首先进行BISR的初始分配,增添硬件直至找到可行的方案,再移除不必要的冗余硬件.另一种方案采用了基于转换的方法,即以不同的逻辑实现同一计算以减少冗余硬

件开销.基准电路的实验结果表明,前一种 BISR 方案增加的平均面积开销为 17%,后一种方案为 10.4%.硬件实现时,测试用例的均值表明,在 3.75 种不同类型的基本硬件单元中,前一种方案需要增加 2.3 种冗余硬件,而后一种仅需 1.5 种,成品率可以达到 87.4%.

2005 年,Breveglieri 等人提出数据通路的在线检测与重构方法[44].首先通过奇偶校验码进行差错检测并定位,再由备份的数据单元代替出错的单元,程序从开始处重新执行.在图 5 所示的结构中,一旦某个数据单元被检测到差错,重试由多路选择器控制数据流改变路径,原来通过故障单元的输入向右移一个单元直至最右端的输入从备份单元通过.使用该方案的前提是一行数据单元中一次最多只有一个存在故障.

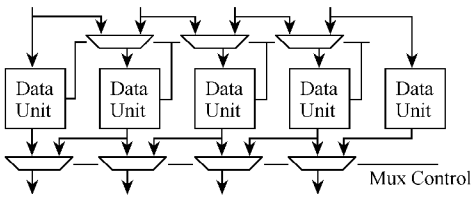

Fig. 5 Set spare unit for reconfiguration.

图 5 设置备份单元进行重构

5 结束语

"芯片级系统的在线测试技术"一文介绍了芯片系统的差错检测技术的发展情况[45],本文概述了近 10 多年来芯片系统的容错技术的研究进展.目前的研究热点是面向 SMT 和 CMP 相结合的高性能微处理器的低成本、高故障覆盖率的容错方案,大量的研究工作主要针对软差错的处理.作者希望本文能够为国内学术界和工业界在提高芯片可信性方面的研究和开发工作提供有用的参考.

参 考 文 献

[1] R A Bringmann, S A Mahlke, R E Hank, *et al*. Speculative execution exception recovery using write-back suppression [C]. The 26th Int'l Symp on Microarchiteture, Austin, 1993

[2] N J Alewine, W K Fuchs, W -M W Hwu. Application of compiler-assisted rollback recovery to speculative execution repair [C]. Workshop on Hardware and Software Architectures for Fault-Tolerance Experiences and Perspectives, Le Mont Saint Michel, France, 1994

[3] Y Tamir, M Tremblay, D A Rennels. The implementation and application of micro rollback in fault-tolerant VLSI systems [C]. IEEE 18th Int'l Symp on Fault-Tolerant Computing, Tokyo, 1988

[4] Y Tamir, M Tremblay. High-performance fault-tolerant VLSI systems using micro rollback [J]. IEEE Trans on Computers, 1990, 39(4): 548-554

[5] V Raghavendrn, C Lursinsap. Automated micro-rollback self-recovery synthesis [C]. The 28th ACM/IEEE Design-Automation Conference, San Francisco, 1991

[6] V Raghavendrn, C Lursinsap. A technique for micro-rollback self-recovery synthesis [J]. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems, 1995, 14(9): 1171-1179

[7] A Orailoglu, R Karri. Automatic synthesis of self-recovering VLSI systems [J]. IEEE Trans on Computers, 1996, 45(2): 131-142

[8] D M Blough, F J Kurdahi, Seong Yong Ohm. High-level synthesis of recoverable VLSI microarchitectures [J]. IEEE Trans on VLSI Systems, 1999, 7(4): 401-410

[9] N Quach. High availability and reliability in the Itanium processor [J]. IEEE MICRO, 2000, 20(5): 61-69

[10] T J Slegel, R M Averill, M A Check, *et al*. IBM's S/390 G5 microprocessor design [J]. IEEE MICRO, 1999, 19(2): 12-23

[11] M Mueller, L C Alves, W Fischer, *et al*. RAS strategy for IBM S/390 G5 and G6 [J]. IBM Journal of Research and Development, 1999, 43(5/6): 875-887

[12] M A Check, T J Slegel. Custom S/390 G5 and G6 microprocessors [J]. IBM Journal of Research and Development, 1999, 43(5/6): 671-680

[13] L Spainhower, T A Gregg. IBM S/390 parallel enterprise server G5 fault tolerance: A historical perspective [J]. IBM Journal of Research and Development, 1999, 43(5/6): 863-873

[14] D C Bossen, J M Tendler, K Reick. Power4 system design for high reliability [J]. IEEE MICRO, 2002, 22(2): 16-28

[15] R Kalla, B Sinharoy, J Tendler. IBM POWER5 chip: A dual-core multithreaded processor [J]. IEEE MICRO, 2004, 24(2): 40-47

[16] J Gaisler. A portable and fault-tolerant microprocessor based on the SPARC V8 architecture [C]. IEEE/IFIP Int'l Conf on Dependable Systems and Networks, Bethesda, 2002

[17] H Ando, Y Yoshida, A Inoue, *et al*. A 1.3GHZ fifth generation SPARC64 microprocessor [C]. The IEEE 40th Design Automation Conference, Anaheim, 2003

[18] M Pflanz, K Walther, C Galke, *et al*. On-line techniques for error detection and correction in processor registers with cross-parity check [J]. Journal of Electronic Testing and Applications, 2003, 19(5): 501-510

[19] T J Slegel, E Pfeffer, J A Magee. The IBM eServer z990 microprocessor [J]. IBM Journal of Research and Development, 2004, 48(3/4): 295-309

[20] T M Austin. DIVA: A dynamic approach to microprocessor verification [J]. Journal of Instruction-Level Parallelism, 2000, 2(1-6): 1-26

[21] J Ray, C H James, F Babak. Dual use of superscalar datapath for transient-fault detection and recovery [C]. The 34th ACM/IEEE Int'l Symp on Microarchitecture, Austin, 2001

[22] A Mendelson, N Suri. Designing high-performance & reliable superscalar architectures: The out of order reliable superscalar (O3RS) approach [C]. IEEE/IFIP Int'l Conf on Dependable Systems and Networks, New York, 2000

[23] T Sato. Analyzing overhead of reissued instruction on data speculative processors [C]. Workshop on Performance Analysis and Its Impact on Design, Austin, 1998

[24] T Sato. Data dependence speculation using data address prediction and its enhancement with instruction reissue [C]. Workshop on Digital Systems Design: Architectures Methods and Tools, Austin, 1998

[25] T Sato, I Arita. Comprehensive evaluation of an instruction reissue mechanism [C]. Int'l Symp on Parallel Architectures Algorithms and Networks, Richardson, 2000

[26] T Sato, I Arita. In search of efficient reliable processor design [C]. The 30th Int'l Conf on Parallel Processing, Valencia, 2001

[27] T Sato. Evaluation the impact of reissued instructions on data speculative processor performance [J]. Microprocessors and Microsystems, 2002, 25(9-10): 469-482

[28] T Sato, I Arita. Tolerating transient faults through an instruction reissue mechanism [C]. The 14th Int'l Conf on Parallel and Distributed Computing Systems, Austin, 2001

[29] T Sato. Exploiting instruction redundancy for transient fault tolerance [C]. The 18th IEEE Int'l Symp on Defect and Fault Tolerance in VLSI Systems, Boston, 2003

[30] T Sato. A transparent transient faults tolerance mechanism for superscalar processors [J]. IEICE Trans on Information and Systems, 2003, E86-D(12): 2508-2515

[31] T Sato. Exploiting trivial computation in dependable processors [C]. The 20th Int'l Conf on Computer and Their Applications, Austin, 2005

[32] A Sodani, G S Sohi. Dynamic instruction reuse [C]. The 24th Int'l Symp on Computer Architecture, Denver, 1997

[33] S-C Lai, S-L Lu, J-K Peir. Ditto processor [C]. IEEE/IFIP Int'l Conf on Dependable Systems and Networks, Bethesda, 2002

[34] Y Y Chen, S J Horng, H C Lai. An integrated fault-tolerant design framework for VLIW processors [C]. The 18th Int'l Symp on Defect and Fault Tolerance in VLSI Systems, Boston, 2003

[35] E Rotenberg. AR-SMT: A microarchitectural approach to fault tolerance in microprocessors [C]. The 29th Int'l Symp on Fault-Tolerant Computing, Madison, 1999

[36] K Sundaramoorthy, Z Purser, E Rotenberg. Slipstream processors: Improving both performance and fault tolerance [C]. The 9th Int'l Conf on Architecture Support for Programming Languages and Operating Systems, Austin, 2000

[37] S K Reinhardt, S S Mukherjee. Transient fault detection via simultaneous multithreading [C]. The 27th Int'l Symp on Computer Architecture, Vancouver, 2000

[38] T N Vijaykumar, P kith, K Cheng. Transient-fault recovery using simultaneous multithreading [J]. Computer Architecture News, 2002, 30(2): 87-98

[39] S S Mukherjee, M Kontz, S K Reinhardt. Detailed design and evaluation of redundant multithreading alternatives [C]. The 29th Int'l Symp on Computer Architecture, Anchorage, 2002

[40] M Gomaa, C Scarbrough, T N Vijaykumar, *et al*. Transient-fault recovery for chip multiprocessors [C]. The 30th Int'l Symp on Computer Architecture, San Diego, 2003

[41] F Rashid, K K Saluja, P Ramanathan, *et al*. Fault tolerance through re-execution in multiscalar architecture [C]. IEEE Int'l Conf on Dependable Systems and Networks, New York, 2000

[42] S N Hamilton, A Orailoglu. Transient and intermittent fault recovery without rollback [C]. IEEE Int'l Symp on Defect and Fault Tolerance in VLSI Systems, Austin, 1998

[43] L Guerra, M Potkonjak, J Rabaey. High level synthesis for reconfigurable datapath structures [C]. Digest of IEEE/ACM Int'l Conf on Computer-Aided Design, Santa Clara, 1993

[44] L Breveglieri, L Koren, P Maistri. Incorporating error detection and online reconfiguration into a regular architecture for the advanced encryption standard [C]. The 20th IEEE Int'l Symp on Defect and Fault Tolerance in VLSI Systems, Monterey, 2005

[45] Jiang Jianhui, Yuan Chunxin. Online testing techniques for chip-level systems [J]. Journal of Computer Research and Development, 2004, 41(9): 1593-1603 (in Chinese)
(江建慧,员春欣. 芯片级系统的在线测试技术[J]. 计算机研究与发展, 2004, 41(9): 1593-1603)

Wang Zhen, born in 1980. She received her B. S. degree in mathematics from the Shandong Normal University in 2002, and M. E. degree in computer science from Tongji University, Shanghai, China in 2005. Now she is a Ph. D. candidate of Tongji University. Her current research interests include fault-tolerant computing, reliability evaluation of high-level circuits, and microprocessor architecture.

王　真,1980年生,博士研究生,主要研究方向为容错计算、高层电路可靠性评估、微处理器体系结构.

Jiang Jianhui, born in 1964. Ph. D., professor and Ph. D. supervisor. He is currently the dean of Department of Computer Science and Technology at Tongji University. His main research interests include fault-tolerant computing, software reliability engineering, microprocessor architecture, digital

system design and testing, performance evaluation of computer systems. He is a senior member of Chinese Computer Federation.

江建慧,1964 年生,博士,教授,博士生导师,中国计算机学会高级会员,现为同济大学计算机科学与技术系主任,主要研究方向为容错计算、软件可靠性工程、微处理器体系结构、数字系统设计和测试、计算机系统性能评估.

Yuan Chunxin, born in 1936. Associate professor. His main research interests are fault-tolerant computing, microprocessor architecture.

员春欣,1936 年生,副教授,主要研究方向为容错计算、微处理器体系结构.

Research Background

As the chip density increases and the reliability demand improves, the fault-tolerance design of high-performance processors becomes more and more important. Since the technologies including simultaneous multithreading (SMT) and chip multiprocessor (CMP) are implemented, many new fault-tolerant strategies have been proposed. This paper gives a survey on error-correcting techniques for high-performance processors, which are categorized into recovery techniques at clock-level, instruction-level, thread-level, and reconfiguration technique. Many schemes, prototypes and products are analyzed. A detailed description and further comparison are given. This work is supported by the National Basic Research 973 Program of China (No. 2005CB321604) and the National Natural Science Foundation of China (No. 90207021).

第 8 届中国粗糙集与软计算学术会议、第 2 届中国 Web 智能学术研讨会、第 2 届中国粒计算学术研讨会联合学术会议(CRSSC-CWI-CGrC2008)征文通知

由中国人工智能学会粗糙集与软计算专业委员会和中国计算机学会人工智能与模式识别专业委员会主办、河南师范大学承办的"第 8 届中国 Rough 集与软计算学术会议(CRSSC2008)"、"第 2 届中国 Web 智能学术研讨会(CWI2008)"和"第 2 届中国粒计算学术研讨会(CGrC2008)"拟定于 2008 年 8 月 22 日至 8 月 24 日在河南省新乡市召开. 现将有关征文事宜通知如下,请相关研究人员踊跃投稿和参会.

征文内容(主要包括,但不局限于以下方面)

Rough 集与软计算:Rough 集理论及应用、计算智能、机器学习、神经网络、Fuzzy 集理论及应用、软计算的逻辑基础、软计算及其应用、概念格多准则决策分析、近似推理与不确定性推理、知识发现与数据挖掘、数据仓库、情感计算、模式识别与图像处理、生物信息与生物计算、演化计算、智能信息处理、其他有关领域.

Web 智能:智慧网络 Web、Farming 与 Web 挖掘、数据和知识网络、集成智能系统、认知 WI 模型、基于 Web 的智能信息系统、Web 信息安全、智能 Agent、网络支持系统、语义 Web、网络推理机、Web 信息过滤、Web 信息抽取、Web 服务、其他有关领域.

粒计算:粒计算基础、词计算、商空间理论及应用、粒逻辑与推理、信息粒化、信息粒的表示、区间分析、聚类分析、邻域系统、认知信息学、双(异质)聚类、多层次数据挖掘、数据仓库的多粒度聚合、基于粒计算的模式分析与处理、基于粒计算的 Web 智能、其他相关领域.

投稿要求

1) 投往会议的稿件必须是原始的、未发表的研究成果,研究经验或工作突破性进展报告,一般不超过 6000 字.

2) 论文包括中英文题目、作者姓名、单位、籍贯、职称、地址、邮编、E-mail 地址、联系电话、中英文摘要(一般不超过 300 字)、关键词、中图法分类号、正文和参考文献;请将基金资助项目及批准号标注于首页页脚;参考文献的著录请包含:作者、论文名、期刊名(书名、出版社、出版地)、出版年、卷、期、页码等项目.

3) 录用论文在《计算机科学》为会议出版的论文专集上发表. 根据论文评审和报告情况,确定优秀论文并推荐给各期刊正刊发表. 在各期刊正刊发表前,作者须根据审稿意见和会议交流情况,扩充并修改论文,然后提交稿件给各期刊. 推荐的期刊正刊包括:《Web Intelligence and Agent Systems》(WIAS, EI 收录)、《The Journal of Chinese Universities of Posts and Telecommunications》(JCUPT, EI 收录)、《International Journal of Computer Science and Knowledge Engineering》(IJCSKE)、《模式识别与人工智能》(中文核心, EI 网络版收录)、《计算机科学》(中文核心)、《广西师范大学学报》(中文核心)、《重庆邮电大学学报(自然科学版)》(科技核心)、《河南师范大学学报》(中文核心)等国际国内期刊.

4) 论文请用 Word 排版(具体排版格式请参考《计算机科学》),欢迎通过会议网站在线投稿.

5) 投稿请登录会议网站:http://web.htu.cn/cs/crssc2008

注意:① 请申明拟投稿的具体会议(如 CRSSC2008, CWI2008 或 CGrC2008). WIAS, JCUPT 和 IJCSKE 的投稿需要英文稿. 希望被推荐到这些期刊的论文作者,必须投稿英文稿件. ② 投稿时请务必留下详细通信地址、邮政编码、电话及 Email,以便联系. ③ 请注明第一作者是否为学生,以便于优秀学生论文评选.

重要日期 截稿日期(收到):2008 年 3 月 31 日,录用日期(发出):2008 年 4 月 30 日,论文修改稿接收和论文注册截止日期(收到):2008 年 5 月 31 日

联系方式 联系人(联系电话):徐久成(0373-3329075);薛占熬(13837350169)

电子信箱:crssc2008@henannu.edu.cn(秘书组), xjc@henannu.edu.cn (徐久成), xuezhanao@163.com(薛占熬)