一种用于容错处理器的指令复制方法

李红兵 尚利宏 周 密 金惠华
(北京航空航天大学计算机学院 北京 100191)

摘 要 介绍一种在容错处理器中实现指令复制的方法。处理器的容错机制是通过修改超标量体系结构,利用时间冗余技术实现的。指令复制是容错机制的一种重要功能。详细描述了其实现方法,同时论述了结合指令复制方法对程序控制流的正确性进行检测的问题。

关键词 容错机制,时间冗余技术,超标量处理器,控制流故障

Instruction Replication Scheme of Fault-tolerant Processor

LI Hong-bing SHANG Li-hong ZHOU Mi JIN Hui-hua
(School of Computer, Beihang University, Beijing 100191, China)

Abstract An instruction replication scheme used in a fault-tolerant processor called RSED was introduced. The processor's fault-tolerant mechanism mainly using temporal redundancy technique was implemented by modifying superscalar processor architecture. As an important function of the fault-tolerant mechanism, instruction replication scheme was introduced in detail, and the issue of control-flow error checking implemented by utilizing instruction replication was also presented.

Keywords Fault tolerant, Temporal redundancy, Superscalar, Control-flow error

1 引言

时间冗余技术利用相同硬件对同一运算执行多次,并通过比较计算结果来检测故障,是实现计算机系统容错机制的一种重要的冗余技术,主要用于检测瞬态故障的发生。文献[3]指出瞬态故障在所有引起计算机系统失效的故障中所占比重接近90%。在利用时间冗余技术实现微处理器的容错机制方面,很多相关的研究工作取得了一定的进展。其中,比较有代表性的是E. Rotenberg在文献[4]中提出的AR-SMT处理器模型,最先基于SMT(Simultaneous MultiThreading)开发了使用时间冗余技术的处理器容错机制。与空间冗余和信息冗余技术相比,利用时间冗余技术实现处理器容错机制可以节省大量硬件资源,缺点是处理器性能有较大降低。

文献[1]中提出了一种容错处理器微体系结构,称为RSED(冗余指令流执行解耦,Redundant Stream Execution Decoupled)。容错机制基于微体系结构,是通过修改超标量体系结构实现的,主要应用了时间冗余技术,可以覆盖处理器内部的全部瞬态故障及部分永久故障。RSED的微体系结构支持两个指令流(线程)并发执行,分别称为正常指令流和冗余指令流。相同程序在两个指令流中同时执行,在指令提交阶段对计算结果比较检错,必要时实施故障恢复操作。

图1是RSED的微体系结构示意图,它实现了多执行部件对指令流进行动态调度;在每个指令流内部实现了指令的顺序发射、乱序执行和顺序提交机制;处理器实现了精确的中断模型。每个时钟周期,处理器可同时发射三条指令,执行三条指令,提交两条指令。

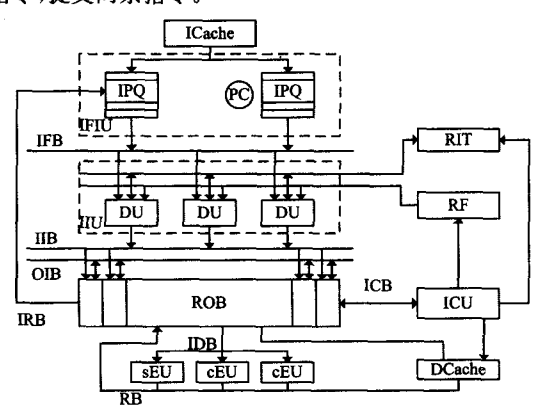

图1 微体系结构示意图

通过采取一些改进措施,与同类处理器容错机制相比,RSED结构简单易于实现,适合于嵌入式应用。在保持高故障覆盖率的同时,时间冗余技术对处理器性能造成的损失有所降低。RSED所提供的多指令流支持无需处理器实现SMT机制,因为采用时间冗余技术在两个线程中执行完全相同的程序具有特殊性,不同线程共享相同的硬件上下文,如寄存器堆等只共享一份,电路结构大为简化。

本文着重介绍容错机制的一个重要方面:指令复制的实

到稿日期:2009-06-08 返修日期:2009-09-20 本文受国家"863"计划(2007AA01Z118)资助。

李红兵(1971-),男,博士生,主要研究方向为计算机体系结构和容错技术等,E-mail:hblee0214@sina.com;尚利宏(1971-),男,讲师,主要研究方向为嵌入式系统和容错技术等;周 密(1981-),男,博士生,主要研究方向为可重构计算等;金惠华(1938-),男,教授,博士生导师,主要研究方向为计算机系统结构和容错技术等。

现。

2 背景

对于利用时间冗余技术实现的处理器容错机制，冗余指令流的产生方式是一个重要的问题，这个过程也称为指令复制。

先以 M. Franklin 所著的文献[2]为例，文中提出的处理器模型利用超标量体系结构中的空闲资源实现时间冗余技术。此方法在指令派遣或发射阶段对所有指令进行复制，生成冗余指令。所有冗余指令使用原来未充分利用的执行单元生成用于校验的检查执行结果。再以 AR-SMT 为例[4]，文中建议的处理器在两个线程中进行完全相同的计算，其中 A 流(Active Stream)首先取指执行，并将计算结果记录到一个延迟缓冲区中。R 流(Redundant Stream)跟随在 A 流后面从延迟缓冲区中读取相应指令重新译码执行，两个线程之间指令执行的最大延迟为延迟缓存区的长度，R 流执行结束之后处理器对其计算结果与保存在延迟缓冲区中的 A 流计算结果进行比较检错，只有两者相同计算结果才最终提交，否则执行卷回重新执行。对比两种指令复制实现方法，第一个例子仅仅使用时间冗余技术保护了执行单元，但结构简单；后者可以保护处理器内部完整的流水线，故障覆盖率更高，缺点是容错机制实现的代价较高。

RSED 处理器的容错机制是通过修改超标量体系结构从而实现时间冗余技术的，很多类似的实现方法都存在容错机制无法覆盖程序控制流故障的问题，主要原因是各个冗余线程并非完全独立，冗余指令流的取指地址来源自主指令流，对各个冗余计算结果进行的比较检错无法覆盖程序控制流的相关信息。

实现程序控制流错误(CFE)检测的方法有很多种，其中签名监视(Signature Monitoring)技术是一种传统的并发检测方法，可以有效检测出控制流故障[6]。对于利用时间冗余技术实现处理器容错机制的情况，以 J. Ray 等人在文献[5]中提出的处理器模型为例，这个体系结构在指令译码阶段，由一个单个指令流负责创建多个冗余指令流，各个指令流的执行结果在指令提交阶段进行交叉比较检测，比较检测期间出现的任何不一致都被认为是检测出指令执行过程中出现的故障。文献[5]中指出了程序控制流可能发生故障的问题，为此在处理器设计中引入了一个新的 PC 寄存器 next-PC，寄存器的值在每条指令提交时使用其中记录的下条指令地址进行更新。在指令的提交阶段比较和提交单元将当前提交的指令的地址与 next-PC 中存储的值进行比较，如果不同表明检测到控制流中出现的故障，由 next-PC 控制卷回逻辑从 next-PC 记录的指令地址处恢复执行。

3 指令复制算法

本节首先介绍基本的指令复制方法；随后对此方法进行扩充，提出支持控制流故障检测的改进的指令复制方法；最后使用伪代码对算法进行描述。

RSED 处理器的前端部分包含两个指令预取队列 IPQ，分别负责为正常和冗余指令流读取指令，并发送给译码单元进行后续处理。其中正常指令流包含一个独立的程序计数器 PC，负责生成取指令地址；冗余指令流对应的 IPQ 没有独立的 PC 寄存器，取指令地址来源于进入重序缓存区 ROB 中的正常指令流。RSED 处理器的基本的指令复制操作是在取指阶段进行的(见图 2)，正常指令流首先发射到 ROB 中，在这里按照程序次序将取指地址依次发送到冗余指令流所属的 IPQ 中；此 IPQ 可以视为先进先出队列，取指令地址从队尾进入，从队头退出，同时根据队头单元从指令高速缓存 ICache 中第二次读取指令，传递给译码单元进行后续处理。作为基本的指令复制方法，发射进入 ROB 的正常指令流将指令地址发往冗余指令流的 IPQ，正常指令流所发出的地址数据是自身的指令地址。与其他类似指令复制方法相比，RSED 处理器的方法有两个主要特点：一方面，冗余指令流取指令地址的更新提前到指令派遣(Dispatch)阶段进行；另一方面，推测执行被取消的分支等不必要的指令不再重新执行。采取这些措施以后，两个指令流之间的执行距离得以缩短，同时冗余指令的执行数量得以最小化，从而 RSED 处理器的指令复制机制既提高了处理器吞吐率，又降低了带宽需求。

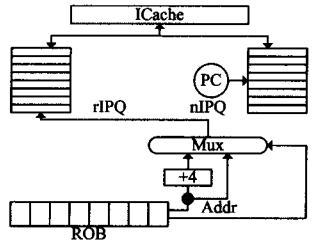

图 2 指令复制简化示意图

使用基本的指令复制方法，RSED 的冗余指令流可以充分利用正常指令流提供的相关性信息，减少被调度执行的冗余指令数量，从而提高处理器的整体性能。

通过上节中的分析可知，RSED 处理器的容错机制无法覆盖程序控制流故障。具体而言，处理器的 PC 寄存器存在发生单点故障的可能性：PC 寄存器只有一个，冗余指令流的取指令地址完全接收自正常指令流，在正常指令流自身存在程序控制流故障的情况下，容错机制无法检测出来。造成控制流故障的原因很多，包括 PC 寄存器电路中，或者传递跳转指令地址的数据通路中发生瞬态故障等等。

保证程序控制流的正确性是处理器容错机制必须具备的功能。为此，通过增加少量电路，对 RSED 的基本指令复制方法进行修改，将程序控制流错误检测与指令复制、重新执行机制结合起来，作为比较检错操作的一部分，在指令提交阶段实施控制流故障检测。主要的修改是，取指地址值不是直接取自正常指令流在 ROB 中记录的正常指令地址，而是取自每条指令记录的 next-pc 值，即每条正常指令的后继指令地址。通过这个方法，在指令提交阶段对两个指令流的对应指令地址进行比较，同时完成了对两条毗邻指令的控制流关系的正确性进行检查。此方法的另一个优点是可以加快冗余指令流中指令的生成速度，有利于提高处理器性能。

在本节的随后部分，首先提出一个经过简化的 RSED 处理器模型，其主要构成都与指令复制相关，随后在此基础上进行算法描述，最后进行算法分析。下面介绍一些后文中用到的术语、数据结构、操作的定义。

定义 1 冗余指令流指令预取队列，$rIPQ=(a_1,a_2,\cdots a_n)$，是一个先进先出队列，其每个元素 $a_i=\langle PC,rob\_no\rangle$，$i=0,1,\cdots,n$。记录指令地址和正常指令流对应的 ROB 单元编号。可以对 $rIPQ$ 实施两个操作：

ripq_enqueue(*addr*,*rob_no*)，指令地址 *addr* 和 ROB 单元编号 *rob_no* 从队尾进队；

⟨*addr*,*rob_no*⟩←*ripq_dequeue*()，指令地址 *addr* 和 ROB 单元编号 *rob_no* 从队头出队。

定义 2 译码单元，*du*=⟨*stream_no*,*rob_no*,*PC*⟩。选取与指令复制有关的信息构成一个三元组，其中 *stream_no* 是指令流编号，取值为{*normal*,*redundant*}，分别代表正常和冗余指令流；*rob_no* 是 ROB 单元编号，对冗余指令流有效，指定冗余指令应该进入的 ROB 单元；*PC* 是指令地址。

定义 3 重序缓存区 *ROB*=(re_1,re_2,…,re_n)，是一个先进先出队列，re_1 是队头元素，re_n 是队尾元素。其中元素 re_i=⟨*avail*,*head*,*inst_type*,*ns_PC*,*rs_PC*,*br_valid*,*target_addr*⟩，$i=1,2,\cdots,n$，记录与指令复制有关的每个 ROB 单元的信息。其中，*avail* 是单元可用标志，值为 0 表示此单元包含正在处理的指令；*head* 是指令复制头结点标志，值为 1 表示此单元是队头；*inst_type* 是指令类型，取值为{*seq*,*br*}，分别表示顺序执行指令和跳转指令；*ns_PC* 和 *rs_PC* 是正常和冗余指令流的指令地址；*br_resolved* 是所属分支方向是否确定的标志，值为 1 表示所属分支方向已经确定；*target_addr* 是指令跳转的目标地址，在指令执行阶段进行计算。

定义 4 指令提交单元与控制流故障检测有关的数据结构，*cu*=⟨*ns_PC*,*rs_PC*⟩，其中 *ns_PC* 是正常指令流指令地址；*rs_PC* 是冗余指令流指令地址。

定义 5 操作 *Replicate*(*type*,*addr*,*rob_no*)，由 ROB 向冗余指令流的 IPQ 发送取指地址。其中 *type* 是操作类型，取值为{*seq*,*br*}，分别表示顺序执行指令和跳转指令；*addr* 是指令地址，对于顺序执行指令值为 ROB 单元自身的指令地址，以及跳转指令值为 ROB 单元在执行阶段计算出的跳转目标地址，操作的实现见图 3。

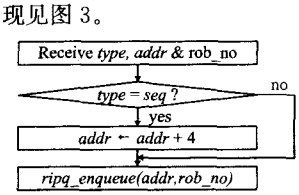

图 3 *Replicate* 操作的执行流程图

算法 1 指令复制算法。初始条件：正常指令流中的指令已经发射进入 ROB 中。

表 1 中涉及的 3 个操作 *fetch*()、*Commit*()和 *FaultRecovery*()分别是完成取指令、指令提交和故障恢复功能。

表 1 故障恢复算法

Stage	Action
Fetch	if(rIPQ is selected by fetch schedule logic) {⟨addr,rob_no⟩←ripq_dequeue();inst←fetch(addr); du←⟨redundant,rob_no,addr⟩;}
Issue	∀i(if(du. stream_no=redundant) {j←du. rob_no;re_j. rs_PC←du. PC});
Operand	∀i(if(re_i. inst_type=seq & re_i. br_resolved=1 & re_i. head=1) {Replicate(seq,re_i. PC); re_i. head←0; re_{i+1}. head←1;}); ∀i(if(re_i. inst_type=seq) {re_{i+1}. br_resolved←re_i. br_resolved;}); ∀i(if(re_i. inst_type=br) {if(branch has been resolved) {re_{i+1}. br_resolved←re_i. br_resolved;} else {re_{i+1}. br_resolved←0;}});
Finish	∀i(if(re_i. inst_type=br & re_i. br_resolved=1 & re_i. head=1) {Replicate(br,re_i. target_addr); re_i. head←0;re_{i+1}. head←1;});

Commit	if(cu. ns_PC≠cu. rs_PC | any other fault is found)
	{FaultRecovery(cu. ns_PC,cu. rs_PC,next_PC);}
	else
	{Commit();}

冗余指令流的 IPQ 是一个先进先出的队列，从队尾接收 ROB 发来的指令地址，从队头读取指令。当冗余指令流被取指调度逻辑选中时，它根据 IPQ 队头记录的地址从 ICache 中读取指令，传递给译码单元。指令发射是指令从译码单元中发射进入可用 ROB 单元的过程，不同指令流的发射过程有所不同，对于冗余指令流而言，译码单元根据其携带的 ROB 编号直接发射给指定的 ROB 单元，并以其携带的指令地址值设置 ROB 单元中记录的冗余指令地址值 *rs_PC*。译码阶段之后是取操作数阶段，此时处于已经确定的有效指令分支以及尚未实施指令复制的顺序执行指令，按照程序次序启动指令复制操作：通过一个专用电路进行指令地址+4 的运算，将计算结果发送给冗余指令流。此外处于读操作数阶段的各个正常指令还要额外判断所处分支是否有效。另一方面，在正常指令流中指令执行结束，结果返回给原来的 ROB 单元后，尚未实施指令复制的指令中最早的一个若是跳转指令，并且处于已经确定的有效指令分支，则这条指令启动指令复制操作：将已经在执行阶段计算出的跳转目标地址值发给冗余指令流。在指令提交阶段，作为比较检错操作的一部分，提交单元对两个指令流的指令地址进行比较，结果一致说明程序控制流中未发生错误，在其他比较检错也通过的情况下，此指令的执行结果将被提交；反之说明发现了程序控制流错误，处理器实施故障恢复/微卷回操作。

可以证明使用算法 1 可以覆盖 RSED 处理器中出现的各种类型的程序控制流错误。

4 试验结果

为了对 RSED 处理器进行功能验证和性能评估，在 SimpleScalar[7]模拟平台的基础上开发了一种执行驱动的处理器模拟器，并在模拟器上运行了多个 Spec95 整数程序进行测试。结果表明，与无容错机制的基线处理器相比，RSED 容错处理器的运行性能下降 17%～31%，优于作为参照的其他容错处理器模型[1]。可见与同类方案相比，RSED 处理器的容错机制可以更为充分地利用冗余的硬件资源，对处理器性能造成的降低有所减少。

另一方面，为了验证容错机制，开发了一种完全使用软件实现的系统仿真和故障注入工具。使用 SystemC 与处理器的 Verilog 结构模型混合仿真的方式完成了处理器容错机制的验证。其中使用了 SystemC 开发人机界面、故障管理、故障分析等高层模块，Verilog 模型本身是可综合的处理器模型。在验证过程中，针对处理器的不同模块设计了一组测试用例，按照不同概率分布对 RSED 处理器的各个模块进行了单粒子翻转(SEU)类型的故障注入。通过对验证程序运行结果的分析，发现控制流故障的故障覆盖率到达 99.3%，证明指令复制方法同时实现的控制流错误检测机制能够有效检测出程序控制流中发生的故障。

结束语 我们提出了一种通过修改超标量处理器体系结

构，利用时间冗余技术实现微处理器容错机制的方法。介绍了其中指令复制方法的实现，并着重分析了利用指令复制实现控制流检测。从仿真结果看，这种设计方法可以有效检测出处理器正常指令流的程序计数器及相关电路中出现的瞬态故障。提出的方法将程序控制流检测和指令复制机制融合起来，与类似的利用时间冗余技术实现的容错处理器相比，进一步节省了硬件资源，减少了额外的执行时间。

参 考 文 献

[1] 李红兵，尚利宏，周密，等. 一种执行解耦的容错处理器[C]//第十三届全国容错计算学术会议. 2009

[2] Franklin M. A Study of Time Redundant Fault Tolerance Techniques for Superscalar Processors[C]// Digest of Papers in the 25th International Symposium on Fault Tolerant Computing. 1995:207-215

[3] Sosnowski J. Transient fault tolerance in digital systems [J]. IEEE Micro, Feb. 1994:24-35

[4] Rotenberg E. AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors [C] // Proceeding of the 29th Fault-Tolerant Computing Symposium. June 1999

[5] Ray J, Hoe J C, Falsafi B. Dual Use of Superscalar Datapath for Transient-Fault Detection and Recovery[C]// Proceeding of 34th Microarchitecture. Dec. 2001

[6] Frohwerk R. Signature analysis: a new digital field service method[J]. Hewlett-Packard J, May 1977:2-8

[7] Burger D, Austin T M. Simplescalar Tool Set Version 2.0[J]. Computer Architecture News, 1997, 25(3):13-25

(上接第 277 页)

集成多个动态模块，多个处理模块的配置可以采用 configuration prefetching, configuration caching 等技术来屏蔽重构延时。

参 考 文 献

[1] Dally W J, Balfour J, et al. efficient Embedded Computing[J]. Computer, 2008, 41(7):27-32

[2] Jou Jer Min, Lee Yun-Lung, et al. A Novel Reconfigurable Computation Unit for DSP Applications[C]// IEEE Computer Society Annual Symposium on VLSI. Porto Alegre, Brazil, 2007

[3] Dutt N, Choi Kiyoung. Configurablef processors for embedded computing[J]. Computer, 2003, 36(1):120-123

[4] Wan M, Zhang H, George V, et al. Design methodology of a low-energy reconfigurable single-chip DSP system[J]. Journal of VLSI Signal Processing, 2001, 28(1):47-61

[5] Salefski B, Caglar L. Reconfigurable computing in wireless [C] // 38th Design Automation Conference. Las Vegas, Nevade, USA, June 2001

[6] Singh H, Lee M H, Lu G, et al. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications[J]. IEEE Transactions on Computers, 2000, 49(5): 465-481

[7] Xilinx Inc. Virtex-II Pro Platform FPGAs[EB/OL]. www.xilinx.com/virtex2pro

[8] Altera Inc. Excalibur Embedded Processor[EB/OL]. www.altera.com/products/devices

[9] Atmel Inc. Field Programmable System Level Integrated Circuits (FPSLIC devices)[EB/OL]. www.atmel.com/products/FPSLIC

[10] OpenCores. AES project[EB/OL]. http://www.opencores.org/

[11] AES Lounge[EB/OL]. http://www.iaik.tugraz.at/content/research/krypto/AES/

重视中、英文摘要的编写

检索数据库要依据文摘的质量来决定该文章是否被收录，读者要根据文摘提供的信息考虑是否阅读、引用原文，如能被利用，才能体现文章的学术价值，提高原文的引用频次，摘要在一篇文章中的作用非常重要！

中文摘要一般不宜超过 200～300 字，英文文摘的长度一般不超过 250words，不少于 150words，语言简洁。除了实在无变通办法可用以外，摘要中不用图、表、化学结构式、非公知公用的符号和术语。关键词一般为 3～8 个，每个关键词首字母大写。

1. 对文献进行主题分析，通过文摘体现主题概念、主题内容等该篇文献最重要的信息，使读者在没有看到全文的情况下，能够很清楚地了解到该篇文献的中心思想。

2. 英文文摘应与中文文摘相对应。

3. 信息量要完整，能够全面包含论文的关键信息，应包括目的、过程及方法、结果三方面内容。

4. 尽量简化一些措辞和重复的单元，删繁从简。

5. 注意文法(详细请见本刊网站的修改要求)。

6. 取消或减少背景信息。背景信息如果过长或占文摘篇幅的比例过大，则反映出的就是对作者所做的工作描述过于笼统和简单。

7. 作者在文献中谈及的未来计划不纳入文摘。

8. 首句不得简单重复题名中已有的信息。

9. 不用第一人称，用过去时态叙述作者工作，用现在时态叙述作者结论。

10. 文摘中的缩写名称在第一次出现时要有全称。