RISC-DSP 处理器中指令数据相关性的提前判断方法

蔡卫光　　姚庆栋　　刘　鹏
(浙江大学信息与电子工程学系　杭州　310027)

摘　要：RISC-DSP 处理器中执行周期数动态可变的指令对数据相关检测造成了困难。该文通过分布式相关检测模型将检测操作转换为依赖关系集合的计算，推测不同流水线状态下后一周期中的依赖关系集合，并根据当前指令相关性和功能单元发出的信号确定当前流水线状态，从而提前判断出下一周期中的指令相关性。按照其集合操作的特点进行逻辑优化，并以所研制的 RISC-DSP 处理器 MediaDSP64 原型机为例进行电路实现。综合结果表明，在对整体电路资源和功耗影响较小的前提下，从原先流水线关键路径中隐藏了相关检测电路，其延时下降了约 30%。

关键词：信号处理；指令相关检测；流水线技术
中图分类号：TP302; TP37　　**文献标识码**：A　　**文章编号**：1009-5896(2010)12-3046-05
DOI: 10.3724/SP.J.1146.2010.00102

Early Detection of Instruction Data Dependence for RISC-DSP Processor

Cai Wei-guang　　Yao Qing-dong　　Liu Peng
(*Department of Information Science and Electronic Engineering*, *Zhejiang University*, *Hangzhou* 310027, *China*)

Abstract: In RISC-DSP processor, the instructions with dynamic execution cycles cause difficulty to data dependence detection. Based on distributed dependence detection model, the detection operation is expressed by the calculation of the dependence relationship set. The relationship set at next cycle for each pipeline state are calculated. In the mean time, current pipeline state is detected based on the current instruction distribution and signals from each function module. Thus the instruction dependence can be determined one cycle in advance. Logic optimization based on its set operation is also proposed and implemented in our designed processor MediaDSP64. Synthesis result shows that with slight increase in hardware resource and power consumption, the detection operation can be hidden from the original pipeline critical path, and its timing delay can be decreased by about 30%.

Key words: Signal processing; Instruction dependence detection; Pipeline technique

1　引言

流水线技术通过多条指令重叠执行的方式提高了处理器性能，但也导致了不同流水级之间的指令相关性问题。随着流水线深度的提高和功能单元数量的增加，指令相关性对性能的影响也更加严重。

先前对指令相关性的研究主要包括：采用指令缓冲区[1]使流水线具有一定的乱序执行能力，减少数据相关引起的停顿。采用 2 bit 着色法[2]标识处理器状态，确定控制相关发生的时刻。利用操作表(operation table)技术[3]对指令流进行分析以确定实际执行时的指令相关，并根据流水线结构进行指令调度。虽然这些工作针对指令相关性问题采取了各种措施，但其目标主要是提高指令执行效率，并未对相关检测电路的结构进行详细的研究。

文献[4]通过指令分层次译码减少了数据相关检测电路的延时，但其流水线较短且主要针对译码级实现，限制了在其它流水级的应用。超标量处理器中数据相关检测有基于保留站的广播方式[5]和基于计数器的方法[6]，前者需要较多的寄存器号码比较逻辑，后者需要在译码时确定指令的执行周期数并维护多个计数器。并发推测多线程(Simultaneous Speculative Threading, SST)处理器通过记分牌在前行与后行线程之间传递指令相关信息[7]，但需要多个指令队列与寄存器文件。这些方法的复杂度较高，限制了在其它结构处理器中的应用。

嵌入式系统中通常允许定制指令，如 MIPS 的 CorExtend 技术[6]等。操作数较多的定制指令会增加硬件实现的复杂度，如工作频率和数据带宽等[8]。文献[9]通过基于哈希映射的影子寄存器结构提高寄存器文件带宽，但并未分析对相关检测逻辑的影响。

2010-01-26 收到，2010-07-22 改回
国家自然科学基金(60873112)和国家高技术研究发展计划专项经费(2009AA01Z109)资助课题
通信作者：刘鹏　liupeng@zju.edu.cn

本文的研究主要针对这几个方面，一是指令操作数较多增加了相关检测电路的延时，影响了处理器工作频率；二是指令执行周期数在实际运行时才能确定，无法在译码时获得；三是要求相关检测逻辑对处理器结构和资源的影响较小。本文的内容安排如下：第 1 节介绍研究背景，第 2 节针对典型流水线结构对常规检测算法进行说明，第 3 节通过流水线状态推测下一周期中的指令分布，给出了相关检测的提前判定算法及其优化方法，第 4 节以媒体处理器 MediaDSP64(以下简称 MD64)为例，进行了算法实现和电路综合，给出了实验结果和分析，最后在第 5 节给出结论。

2　相关研究

MD64 是一种 RISC-DSP 结构的媒体处理器[10,11]，它也是多核片上系统和片上互连网络研究中的面向计算的处理器核。不仅支持 RISC，DSP 和 SIMD 指令集，还能够在单个定制指令中实现多个复杂操作。支持两个操作数同时来自存储器，从而克服了定制指令不能访问存储器的不足[9]。在 400~500 MHz 的频率下，达到 3200~4000 16×16 MMAC/s (Mega MAC per second)的运算能力。

MD64 结构如图 1 所示。流水线前端为按序单发射取指，后端则将复杂指令分解成两个微操作送入相应的子流水线执行。微操作之间允许部分乱序执行，而精确中断由基于影子寄存器的检查点(checkpoint)机制实现。在由 DA，AC 和 DM 级组成的访存子流水线内与 EX 级内，用户可以添加专用硬件实现特殊功能，如针对媒体处理中比特流缓冲区的管理指令和变长码的解码指令等。这些指令的执行周期需要在运行时确定，例如变长码的解码指令其执行周期就和具体的码流有关。

原先处理器关键路径由 3 个部分串联组成，分别是数据相关检测、控制信号生成和流水级控制，其延时约为 3.53 ns。其中检测电路的延时占有较大比例(1/3 左右)，主要原因有两点。一是单条定制指令包含多个操作导致其操作数个数增加，需要检测的路径也随之增加。二是流水线深度增加导致执行

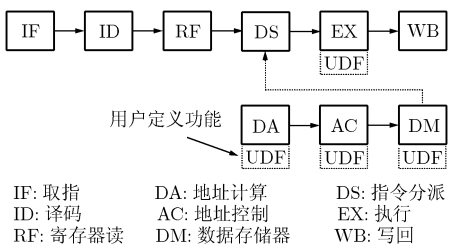

图 1 MD64 流水线结构

中的指令也随之增加，需要对更多的指令进行检测。因此，为了提高工作频率并避免对结构进行较大改动，指令相关检测成为电路设计的关键问题。

3　指令相关检测

以数据相关为例，如图 2 所示，检测单元与功能单元分布在各个流水级内，指令在 WB 流水级将其结果写入寄存器文件。针对流水级 $S(i)$内的指令 $I(i)$，检测单元 HDU(i)判断其是否存在数据相关性的公式如式(1)，式(2)所示。其中 FU_Set 表示功能单元集合，Src_Set 表示源操作数集合，Dst_Set 表示目的操作数集合。

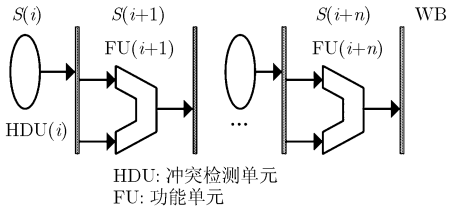

图 2 典型流水线结构

$$I(i).\text{FU\_Set} \cap \text{HDU}(i).\text{FU\_Set} \neq \phi \tag{1}$$

$$\exists j, j = 1, \cdots, n, I(i).\text{Src\_Set} \cap I(i+j).\text{Dst\_Set} \neq \phi \tag{2}$$

式(1)表示 HDU(i)所负责检测的功能单元中，包含 $I(i)$所需要使用的功能单元。式(2)表示 $I(i)$的源操作数集合与后续流水级中指令的目的操作数集合存在交集。可以通过检查指令的依赖关系集合 Dep_Set 是否为空来判断相关性，如式(3)所示，功能单元的使用信息等已隐藏在其操作数集合中。

$$\begin{aligned}\text{Dep\_Set}=&I(i).\text{Src\_Set} \cap I(i+1).\text{Dst\_Set} \\ &\cup I(i).\text{Src\_Set} \cap I(i+2).\text{Dst\_Set} \\ &\vdots \\ &\cup I(i).\text{Src\_Set} \cap I(i+n).\text{Dst\_Set}\end{aligned} \tag{3}$$

4　提前判定算法

4.1 依赖关系集合的提前判定

该方法的基本思想是计算下一周期内的依赖关系集合，并将其结果存储在流水线界面寄存器内。下个周期可以直接使用该结果，从而隐藏检测电路的延时。本文将流水线的运行状态分为 3 种，即正常运行状态(RUN)、滑行状态(SLIP)和停顿状态(STALL)。正常运行状态下每条指令在一个时钟周期后进入相邻的后一个流水级，停顿状态下指令的位置保持不变。滑行状态下，滑行点将不断插入 nop 指令并将其送入相邻的后一个流水级，滑行点之前的流水级为停顿状态，之后的流水级为正常运行状态。在分析的范围内假定流水线满足规整性，即指

令只有一个入口(来自流水线前端)和出口(WB 级)。如图 3 所示，T_0 表示当前周期的指令分布，而 T_1 表示下一周期的分布。本节将以 $S(i)$级为例，给出不同状态下其依赖关系集合的提前计算公式。

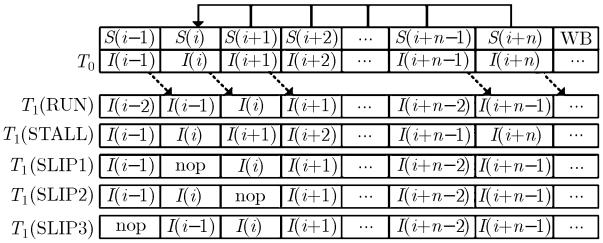

图 3 不同流水线状态下的指令传递

当 T_0时刻为正常运行状态时，T_1时刻的指令分布如 T_1(RUN)所示。T_1时刻 $S(i)$内指令的依赖关系集合如式(4)所示。

$$\begin{aligned}&\text{Dep\_Set(RUN)}\\&=I(i-1).\text{Src\_set}\cap I(i).\text{Dst\_Set}\\&\cup I(i-1).\text{Src\_Set}\cap I(i+1).\text{Dst\_Set}\\&\qquad\vdots\\&\cup I(i-1).\text{Src\_Set}\cap I(i+n-1).\text{Dst\_Set}\end{aligned}\tag{4}$$

若在 T_0 时刻计算该集合需考虑指令的位置变化，例如指令 $I(i\text{-}1)$在 T_0时刻还位于 $S(i\text{-}1)$内。将其变成基于流水级的形式后如式(5)所示，即在 T_0 时刻将 $S(i)$到 $S(i+n\text{-}1)$内的指令对 $S(i\text{-}1)$内的指令进行检测，其结果相当于 T_1时刻对 $S(i)$内指令的检测结果。

$$\begin{aligned}\text{Dep\_Set(RUN)}=&S(i).\text{Src\_set}\cap S(i).\text{Dst\_Set}\\&\cup S(i).\text{Src\_Set}\cap S(i+1).\text{Dst\_Set}\\&\qquad\vdots\\&\cup S(i).\text{Src\_Set}\\&\cap S(i+n-1).\text{Dst\_Set}\end{aligned}\tag{5}$$

当 T_0时刻为停顿状态时，T_1时刻的指令分布如图3中 T_1(STALL)所示，与 T_0时刻的分布完全相同，提前计算依赖关系的公式如式(6)所示。

$$\begin{aligned}\text{Dep\_Set(STALL)}=&S(i).\text{Src\_set}\\&\cap S(i+1).\text{Dst\_Set}\\&\cup S(i).\text{Src\_Set}\\&\cap S(i+2).\text{Dst\_Set}\\&\qquad\vdots\\&\cup S(i).\text{Src\_Set.}\\&\cap S(i+n).\text{Dst\_Set}\end{aligned}\tag{6}$$

当 T_0时刻为滑行状态时，需要确定滑行点的位置。以滑行点位于 $S(i)$流水级为例，T_1时刻的分布如 T_1(SLIP1)所示。此时 $S(i\text{-}1)$和之前流水级保持停顿，$S(i+1)$和之后流水级保持正常运行，$S(i)$内不断插入 nop 指令，所以 Dep_Set(SLIP1)为空集ϕ。

当滑行点位于 $S(i+1)$与 $S(i+n)$之间时，以位于 $S(i+1)$内为例，T_1 时刻的指令分布如 T_1(SLIP2)所示。$S(i+1)$内不断插入 nop 指令，由于 nop 指令对检测结果无影响，所以从 $S(i+1)$向后传递的 nop 指令则可作为普通指令对待。此时其计算公式如式(7)所示

$$\begin{aligned}\text{Dep\_Set(SLIP2)}=&S(i).\text{Src\_set}\\&\cap S(i+1).\text{Dst\_Set}\cup S(i).\text{Src\_Set}\\&\cap S(i+2).\text{Dst\_Set}\\&\qquad\vdots\\&\cup S(i).\text{Src\_Set}\\&\cap S(i+n-1).\text{Dst\_Set}\end{aligned}\tag{7}$$

当滑行点位于 $S(i\text{-}1)$或 $S(i\text{-}1)$之前时，T_1时刻的分布如 T_1(SLIP3)所示。此时对 $S(i)$的检测与正常运行时的效果相同，Dep_Set(SLIP3)的计算公式与 Dep_Set(RUN)相同。

4.2 电路实现与优化

若直接按照式(5)，式(6)和式(7)计算结果，再根据运行状态选择正确的值存储在界面寄存器中，则需要的检测路径较多，且检测对象和检测范围也不同，通常这意味着较大的电路延时和资源占用。可以在公式间共享检测路径进行优化，定义目的操作数集合 Dst_SubSet 如式(8)所示。

$$\begin{aligned}\text{Dep\_SubSet}=&S(i+1).\text{Dst\_Set}\cup S(i+2).\text{Dst\_Set}\\&\cup S(i+3).\text{Dst\_Set}\cup\cdots\\&\cup S(i+n-1).\text{Dst\_Set}\end{aligned}\tag{8}$$

则 3 个公式可简化为如下所示。

$$\begin{aligned}\text{Dep\_Set(RUN)}=&(S(i).\text{Dst\_Set}\cup\text{Dst\_SubSet})\\&\cap\text{S}(i-1).\text{Src\_Set}\end{aligned}\tag{9}$$

$$\text{Dep\_Set(SLIP2)}=S(i).\text{Src\_Set}\cap\text{Dst\_SubSet}\tag{10}$$

$$\begin{aligned}\text{Dep\_Set(STALL)}=&(S(i+n).\text{Dst\_Set}\\&\cup\text{Dst\_SubSet})\cap S(i).\text{Src\_Set}\end{aligned}\tag{11}$$

该公式需要对源操作数集合与目的操作数集合单独进行操作。然而在传统设计中寄存器号码采用二进制编码表示，难以表达集合操作的概念。因此在 MD64 中使用独位码(One-Hot Code)表示指令的 Src_Set 和 Dst_Set。其位宽与寄存器文件容量相

等，每比特对应一个寄存器。若指令使用到该寄存器，则将相应的比特设为高电平。于是“并集”操作转换为电路上的“逻辑或”操作，“交集”操作转换为“逻辑与”操作，从而直接实现上述算法。

5　实验结果与分析

基于 TSMC 0.13 μm (generic and worst case) 的工艺条件，我们在 MD64 内实现了该算法。综合结果表明采用该算法后检测电路自身的延时有所增加，原因在于需考虑多种流水线运行状态。新方法并未直接降低检测电路延时，而是将该延时隐藏。相当于将原先关键路径分为两个周期(如图 4 所示)。

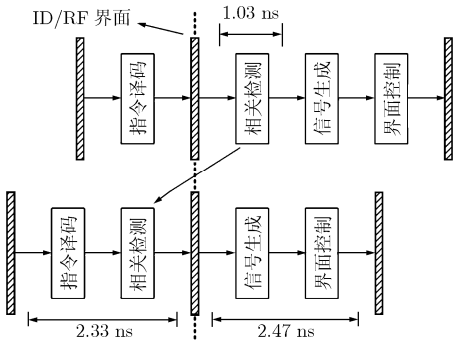

图 4 RF 级提前检测结构

图 5 列出了检测电路所在流水级的关键路径延时，共有 RF，AC 和 DS 这 3 个流水级需要检测。其中 RF 级检测路径最多，需要对 DA 到 EX 共 5 个流水级内的指令进行检测。RUN 状态下对 RF 级的提前检测需用到 ID 级指令，而 ID 级需先对指令译码和独位码编码，所以其延时最长，如图 4 所示。在 AC 级与 DS 级的检测电路内，流水线状态信号生成的延时已经超过了检测操作本身的延时。

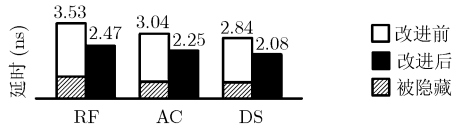

图 5 优化前后的延时对比

表 1 列出了 RF 级检测电路延时的详细信息。RF 级需检测的源操作数集合最多包含 6 个寄存器，目的操作数集合为 2 个寄存器，因此 ID 级内指令译码和独位码编码的延时较大(1.66 ns)。在相关检测部分，采用独位码不仅减少了单个指令的检测路径，并能够在流水级之间共享检测路径，从而减少了电路复杂度和驱动的延时。

表 1 RF 级检测电路延时信息

算法	译码	检测	总体
常规算法 (ns)	–	1.03	1.03
提前算法 (ns)	1.66	0.67	2.33

电路资源对比如表 2 所示，检测电路本身的资源占用有较大下降，但独位码编码器和独位码在流水线中的传递需要较多的资源(6574 门)，整体上增加了 3771 门。采用随机激励进行的峰值功耗分析表明，采用提前判断算法后相关检测电路的功耗从原先的 4.15 μW/MHz 上升到 5.51 μW/MHz(包括独位码编码器的功耗)。这些资源和功耗的增加对整个处理器的影响很小。

表 2 电路资源对比

算法	RF	AC	DS	其它	整体
常规算法 (门)	3034	627	206	–	3867
提前算法 (门)	506	375	183	6574	7638

在相关检测的延时被隐藏后，先前流水线关键路径的延时从 3.53 ns 下降到 2.47 ns，减少了约 30%，达到了设计目标的要求。

6　总结

随着流水线深度与宽度的增加，相关检测电路的延时成为制约处理器时钟频率提升的关键因素。特别是对于媒体处理器与可配置结构的处理器，指令的操作数较多，且存在执行周期数动态可变的特性。这对检测电路的设计提出了更高的要求。

本文将流水线运行方式归纳为 3 种状态。根据当前时刻指令分布和功能单元的信号，由当前流水线状态推测下一周期内的指令分布，在本周期内判断出下一周期的指令相关性，从而能够隐藏相关检测电路的延时。在 MediaDSP64 处理器上的实验表明，流水线关键路径延时下降了约 30%，资源占用增加了约 3800 门。对于 VLIW 结构和超标量结构的处理器，本文提出的方法也具有一定的参考意义。

参 考 文 献

[1] Lu Jia-jing, Zhou Xiao-fang, and Wang Jun-yu. A novel dynamic scheduling algorithm of data hazard for embedded processor [C]. Proceedings of the 7th IEEE International Conference on ASIC. Shanghai, China, IEEE, 2007: 28–31.

[2] Chang Meng-chou and Shiau Da-sen. Design of an asynchronous pipelined processor [C]. International Conference on Communications Circuits and Systems. Seoul,

Korea, IEEE, 2008: 1093–1096.

[3] Shrivastava A, Earlie E, and Dutt N D, *et al*.. Retargetable pipeline hazard detection for partially bypassed processors [J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2006, 14(8): 791–801.

[4] 余巧艳, 刘鹏. 一种面向 DSP 深度压缩指令的数据竞争检测方法[J]. 浙江大学学报 (工学版), 2005, 39(10): 1501–1506.
Yu Qiao-yan and Liu Peng. Data hazard checking method for heavily compressed DSP instruction [J]. *Journal of Zhejiang University* (*Engineering Science*), 2005, 39(10): 1501–1506.

[5] 胡伟武, 张福新, 李祖松. 龙芯 2 号处理器设计和性能分析[J]. 计算机研究与发展, 2006, 43(6): 959–966.
Hu Wei-wu, Zhang Fu-xin, and Li Zu-song. Design and performance analysis of the godson-2 processor [J]. *Journal of Computer Research and Development*, 2006, 43(6): 959–966.

[6] MIPS Technology. MIPS32 74K Processor Core Family Software User's Manual. 2008. 12.

[7] Shailender C, Robert C, and Magnus E, *et al*.. Simultaneous speculative threading: a novel pipeline architecture implemented in SUN's ROCK processor [C]. The 36th International Symposium on Computer Architecture. Austin, USA, IEEE, 2009: 484–495.

[8] Iqbal M A and Awan U S. Run-time reconfigurable instruction set processor design: RT-RISP [C]. The 2nd International Conference on Computer Control and Communication. Karachi, Pakistan, IEEE, 2009: 1–6.

[9] Jason C, Han Guo-ling, and Zhang Zhi-ru. Architecture and compiler optimizations for data bandwidth improvement in configurable processors [J]. *IEEE Transactions on Very Large Scale Integration Systems*, 2006, 14(9): 986–997.

[10] Shi Ce, Wang Wei-dong, and Zhou Li, *et al*.. 32b RISC/DSP media processor: MediaDSP3201 [C]. Proceedings of SPIE-IS & T Electronic Imaging. San Jose, USA, SPIE, 2005: 43–52.

[11] 刘鹏, 姚庆栋, 李东晓等. 32 位媒体数字信号处理器[P]. 中国专利, 200410016753.8, 2007–01–31.
Liu Peng, Yao Qing-dong, and Li Dong-xiao, *et al*.. 32 bit media DSP processor [P]. China patent, 200410016753.8, 2007-01–31.

蔡卫光： 男，1983 年生，博士生，研究方向为媒体处理器结构和媒体处理算法.

姚庆栋： 男，1932 年生，教授，博士生导师，研究方向为视频编码、数字通信、系统芯片设计和计算机体系结构.

刘 鹏： 男，1970 年生，博士，副教授，研究方向为计算机体系结构、集成电路设计、片上互连网络和并行计算.