

分支预测与值预测在 VLIW 处理器中的实现

李笑天, 郭德源, 何 虎

(清华大学 微电子所, 北京 100084)

摘 要: 为了降低超长指令字(VLIW)架构的平均跳转开销和平均访存时延,并减少 VLIW 程序的代码体积,提出了一种全新的将分支预测与值预测技术应用于 VLIW 架构的方法.首先分析现有超标量(Superscalar)架构中动态预测技术与 VLIW 架构中指令静态并行之间所存在的矛盾;通过拓展原有跳转指令和读内存指令,使之与不同的延时槽个数相对应,并根据不同的指令来阻塞流水线或延时写回寄存器,从而解决动态预测技术造成 VLIW 架构静态调度周期错乱的问题.基于 Gem5 仿真平台和清华大学 Magnolia VLIW 数字信号处理器(DSP)的基准测试程序实验表明,该分支预测与值预测技术能显著地提高 VLIW 架构的性能,缩小 VLIW 程序的代码体积.

关键词: 超长指令字架构;分支预测;值预测;动态预测

中图分类号: TP368.1

文献标识码: A

文章编号: 1000-7180(2015)01-0054-06

Realization of Branch Prediction and Value Prediction in VLIW

LI Xiao-tian, GUO De-yuan, HE Hu

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: To reduce the average branch penalty, the average memory reference latency and the program code size in VLIW (Very Long Instruction Word) architecture, A new method to implement branch prediction and value prediction in VLIW is presented. Firstly, the confliction of the dynamic prediction in Superscalar and the static instruction parallelism in VLIW is analyzed. Then the branch and load instruction is expanded, one-to-one correspondence between the delay slot and the new expanded instructions. The pipeline is stalled and the write back stage is delayed according to the given instruction. Benchmark tests based on Gem5 and Magnolia VLIW DSP of Tsinghua University are presented to prove the advantage of the branch prediction and value prediction in VLIW.

Key words: VLIW; branch prediction; value prediction; digital signal processor; dynamic prediction

1 引言

近年来,超长指令字(VLIW)架构凭借高并发、低功耗、硬件设计简单等优点,在数字信号处理器(DSP)领域得到了广泛应用^[1].如 TI C6000 系列^[2]、CEVA-X 系列、CEVA-XC 系列^[3] DSP 处理器都采用 VLIW 架构,且拥有不俗的数字信号处理性能.

VLIW 架构的缺陷同样很明显,其指令静态并发的特性决定了 VLIW 架构流控制能力偏弱,只能采用延时跳转的方式进行跳转,而延时跳转在实际使用中效率较低,跳转损失较大,且容易造成代码膨

胀,导致访问指令 Cache(缓存)效率降低,影响取指的性能.随着流水线技术的不断发展,VLIW 架构的流水线深度也在不断增加,访存指令的时延逐渐成为制约挖掘 VLIW 指令并行度的主要障碍之一.

在超标量(Superscalar)架构下被广泛应用的跳转分支预测技术^[4]能够很好地解决处理器跳转的平均性能损失.而基于值局部性原理^[5]的值预测技术^[6]能够有效降低访存的平均时延,有效地缩短 VLIW 程序的关键路径,提高处理器性能.

本文通过拓展原有 VLIW 架构下的跳转和访存指令,使之与各个不同的延时槽数目相对应,并在

预测成功与预测失败时,根据不同的指令,分别向流水线发送阻塞或者延时信号,解决在 VLIW 架构下使用动态分支预测和值预测所带来的静态指令并发周期错乱的问题。

2 分支预测

2.1 分支预测技术介绍

现今处理器流水线技术发展迅速,通用处理器和 DSP 往往具有很深的流水线级数。这导致程序的每一次跳转都将造成更大的性能损失。因为跳转指令一旦确认跳转,流水线需要被清空,重新取指,从而在流水线中引入了气泡(Bubble)。考虑到通用程序的跳转指令所占的比例高达 15%~25%,而 VLIW 架构的高并行性,更提高了每个周期遇到跳转指令的可能性,由跳转指令造成的性能损失非常可观^[7]。

在 Superscalar 架构下,分支预测是解决跳转性能损失的最好方法。分支预测是指在跳转指令执行之前,预测其跳转目标地址,并提前取指的技术。如果预测正确,则流水线无需清空,可以继续执行。只有在预测失败时,才需要清空流水线重新取指。

现今的主流 Superscalar 处理器,比如 ARM Cortex 系列^[8]、Intel X86 等,都具有强大的分支预测单元,预测的成功率达到 90% 以上,大大减少了跳转的损失。

2.2 现有 VLIW 架构跳转指令存在的问题

现今多款商用 VLIW DSP 都采用 BD^[9](跳转延时)指令代替 Superscalar 中的 B(跳转)指令,原因在于 VLIW 架构采用显示并行的方式执行指令,指令间的并行性由编译器静态决定,有别于 Superscalar 使用寄存器依赖表、重排序缓冲等复杂的动态并行检测单元来决定指令的并行。

如图 1 所示是 BD 指令的示意图。BD 指令与跳转目的地址的指令之间相隔了 i 个指令包(i 为 BD 指令延时槽的个数)。一般来说,BD 指令延时槽的个数等同于取指级到执行级的流水线级数。这样,通过有效填满 BD 指令的延时槽,可以做到不清空流水线而完成指令的跳转。

有效填满 BD 指令延时槽的前提是,BD 指令并不处于关键路径(critical path)上。对于 C 语言最常见的两种跳转方式:for 循环、while 循环,如果循环体较短,只能使用 NOP 指令(NOP 指令不执行任何操作)来填满空的延时槽。这样的做法不仅没有减少跳转指令带来的损失,相反增加了代码体积,间接影响了处理器从 ICache(指令缓存)取指的性能。

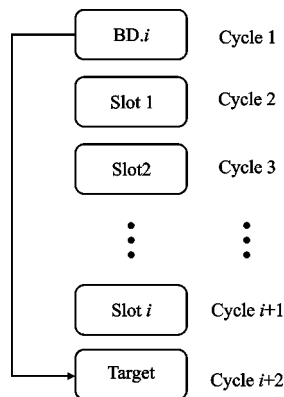


图 1 VLIW 中 BD 指令执行示意图

2.3 分支预测技术在 VLIW 中的实现

基于 VLIW 架构静态并行的考虑,直接采用 Superscalar 架构 B 指令的执行和预测方式,当 B 指令预测正确时,程序的静态并行就会出现。

如图 2 所示,MUL 指令的结果要在 4 拍之后才能得到(图中虚线表示)。但是 B 指令预测成功导致 ADD 指令在第 4 个周期读到了错误的 r0 值。而发生上述错误的根源在于 VLIW 架构的静态分发,它不会像 Superscalar 一样,在指令分发时,去检查源操作数是否已经就绪。

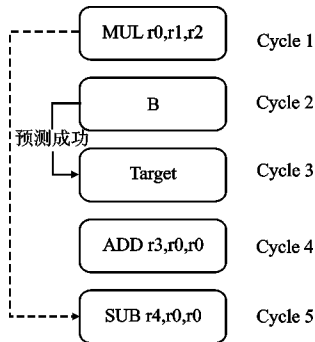


图 2 VLIW 架构分支预测成功时指令周期错乱的问题

为了避免出现上述情况,必须拓展跳转指令,以应对各种不同的静态分发情况。图 3 是一个编译器调度的基本块^[10](Basic Block)的示意图,其中 BEGIN 与 END 代表一个基本块的开始与结束,左半边代表跳转指令所在关键路径,右半边代表非跳转指令的关键路径,节点 B、P、Q 分别代表跳转指令执行时刻、非跳转关键路径上的最后一条指令的执行时刻和得到结果的时刻。

$$\textcircled{1} L_b > L_p + L_q$$

此时跳转指令所在关键路径较长,跳转指令只需要放在整个 Basic Block 的最后,就能正确执行。

$$\textcircled{2} L_p < L_b < L_p + L_q$$

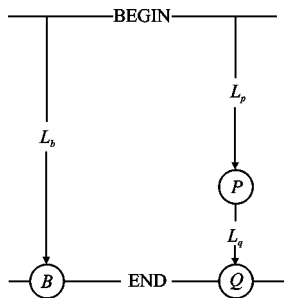


图 3 Basic Block 中 B 指令与非 B 指令关键路径

此时跳转指令可在节点 P 和节点 Q 之间执行, 预测正确时, VLIW 处理器不能直接跳转到目标地址, 需阻塞住分发级, 阻塞的周期数为 $L_q - (L_b - L_p)$ 。

$$\textcircled{3} L_b < L_p$$

这种情况可以作为第二种情况的特例, 跳转指令可以在节点 P 一起分发, 分发级阻塞 L_q 个周期。

现今的 VLIW 架构处理器中, 跳转指令往往是延时槽最长的指令。假设跳转的延时槽为 N , 可以将带有分支预测的跳转指令扩展为 $B.0, B.1, \dots, B.N$ 。B 之后所带的数字代表该条 B 指令预测正确后, 分发级需要阻塞的周期数。

图 4 所示为整个 VLIW 架构分支预测的流程。图中右侧虚线部分是 VLIW 架构分支预测相比于 Superscalar 分支预测多出的部分。如果预测错误, 则清空流水线, 重新取指。如果预测正确, 根据不同的 B 指令, 通过分发阻塞选择器向分发级发送不同周期的阻塞信号来保证 VLIW 架构静态分发的正确性。

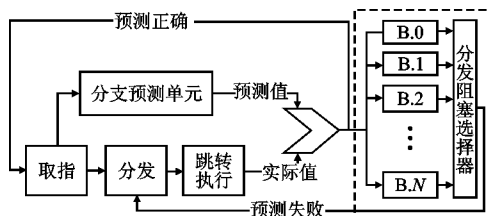


图 4 VLIW 架构分支预测流程

假设 VLIW 程序中, 原来跳转延时槽为 N , 跳转和非跳转指令关键路径各种长度可能性相同, 且分支预测成功率为 p , 则可以得到以下期望值:

无分支预测 BD 指令平均时延应为 BD 指令延时槽被有效填满的期望值:

$$E(T_{BD}) = \frac{1}{N} \sum_{i=0}^N i = \frac{N+1}{2}$$

无分支预测 BD 指令平均 NOP 冗余也为 BD 指令延时槽被有效填满的期望值:

$$E(NOP_{BD}) = \frac{1}{N} \sum_{i=0}^N i = \frac{N+1}{2}$$

有分支预测时, $B.i$ 指令预测成功时的跳转时延为 i , 预测失败则为 N , 故有 B 指令跳转平均时延:

$$E(T_{BD}) = \frac{1}{N} \sum_{i=0}^N (p * i + (1 - p)N) = \frac{p(N-1)}{2} + 1$$

有分支预测时, $B.i$ 指令之后不含 NOP 指令, 阻塞信号均动态决定, 故 B 指令平均 NOP 冗余:

$$E(NOP_B) = 0$$

比较无分支预测和有分支预测的情况, 有

$$E(T_B) < E(T_{BD}), E(NOP_B) < E(NOP_{BD})$$

对于 VLIW 程序, 由于循环出现的次数多于通用程序, 且由于 VLIW 的高并发性, 导致 BD 指令的延时槽一般很难有效填充, 因此实际应用中分支预测对于 VLIW 性能的提升比理论分析更加显著。

3 值预测技术

3.1 值预测技术介绍

现今的处理器领域, 存储设备的发展远落后于流水线技术的发展, 导致访存成为挖掘程序并行性的最主要的障碍之一。主流的 ARM Cortex-A 系列处理器, TI C6000 系列 DSP 等都有长达 3~5 个周期的访存时延。

现有计算机体系架构主要依靠三种局部性原理来降低访存的平均时延: 空间局部性、时间局部性和值局部性(Value Locality)。前两者被运用到虚拟内存和 Cache(缓存)之中, 而后者是值预测技术的基础。指的是对同一地址的多次访问有很大可能会读取到同一个值。

在 Superscalar 架构下, 值预测是指在访存指令执行之前, 预测其访存地址, 并提前进行访存的技术。如果预测成功, 则可以缩短访存时延, 如果预测失败, 则需要重新执行访存指令。

3.2 值预测对于 VLIW 的重要性

值预测技术的原理是值局部性, 以下列举的情况体现了程序的值局部性特点^[11]:

- 常量;
- Switch 语句所产生的隐式地址常量;
- 虚函数;
- 函数调用栈维护;
- 数组的访问。

对于 VLIW 架构广泛应用的 DSP 领域, 程序的

值局部性特点更显著,因为 DSP 程序往往比通用程序有更多的循环,数组线性访问,浮点计算等.这些使 VLIW 程序具有比通用程序更好的可预测性.

3.3 值预测在 VLIW 中的实现

在 VLIW 中实现值预测技术,会遇到与分支预测相同的问题.动态的值预测技术会给静态调度的程序带来周期上的错误.

图 5 左边所示的指令执行流程,在 VLIW 执行方式下,LD 指令的延时槽为 1 个周期,由实线表示.若值预测正确,LD 指令的延时槽变为 0,由虚线表示.那么在值预测正确和错误两种情况下,ADD 指令的源寄存器 r0 将读取不同的值,产生错误.

同理,图 5 右边所示的指令执行流程将 VLIW 方式下 LD 指令的延时槽改为 0,在值预测正确和错误的两种情况下,ADD 指令源操作数 r0 同样将读取到不同的值.

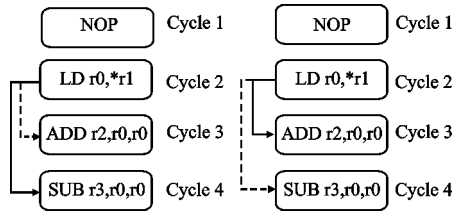


图 5 动态值预测与 VLIW 静态调度的冲突

和分支预测类似,在 VLIW 架构中,需要拓展 LD 指令来应对不同的静态调度情况.假设处理器中,LD 指令延时槽个数最多为 N ,将 LD 指令拓展成 $N+1$ 条指令,记为 LD.0, LD.1, ..., LD.N, 分别代表具有 $0 \sim N$ 个延时槽的 LD 指令.

图 6 所示为 VLIW 架构中值预测流程.图中右侧虚线部分是 VLIW 值预测相比于 Superscalar 值预测多出的部分.如果值预测成功,先经过访存地址确认,确认在值预测期间,该内存地址没有被其他的存储指令所访问,之后通过一个选择器向写回级发送 Delay 的请求,LD. i 指令则发送 i 个周期的延时请求给写回级.如果值预测失败,通过一个选择器向各级发送阻塞请求,对于 LD. i 指令,则发送 $N-i$ 个周期的阻塞信号.这样做的目的就是维护原有的 VLIW 程序的静态性,保证上下文的指令都能读到正确的操作数.

假定 VLIW 程序中,带有 $0 \sim N$ 个延时槽的 LD 指令出现的概率相同,且值预测成功率为 p ,则无值预测 LD 访存平均时延应为 LD 指令延时槽被有效填满的期望值:

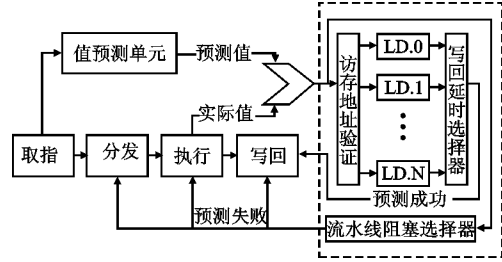


图 6 VLIW 架构值预测流程

$$E(T_{LD}) = \frac{1}{N} \sum_{i=0}^N i = \frac{N+1}{2}.$$

无值预测 LD 访存平均 NOP 冗余也为 LD 指令延时槽被有效填满的期望值:

$$E(NOP_{LD}) = \frac{1}{N} \sum_{i=0}^N i = \frac{N+1}{2}.$$

有值预测时,LD. i 指令预测成功时访存时延为 1,预测失败时为 i ,故有 LD. i 访存平均时延:

$$\begin{aligned} E(T_{LD,i}) &= \frac{1}{N} \sum_{i=0}^N (p + (1-p)i) \\ &= p + (1-p) \frac{N+1}{2}. \end{aligned}$$

有值预测时,LD. i 指令后不含 NOP 指令,阻塞和延时信号均动态决定,故 LD 指令平均 NOP 冗余:

$$E(NOP_{LD,i}) = \frac{1}{N} \sum_{i=0}^N 0 = 0.$$

比较无值预测和有值预测的情况,有

$$E(T_{LD,i}) < E(T_{LD}), E(NOP_{LD,i}) < E(NOP_{LD}).$$

对于 VLIW 程序,访存往往处于关键路径上,且由于 VLIW 程序的高并发性,使得几乎每个周期都会遇到访存指令,值预测对于 VLIW 性能的实际提升比理论分析更加显著.

4 实验结果

4.1 实验 DSP 介绍

我们选取清华大学微电子所自行设计的 VLIW 架构 DSP Magnolia 作为测试处理器^[12-13].

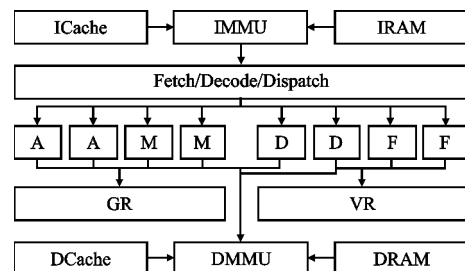


图 7 Magnolia VLIW DSP 双簇架构

如图 7 所示是 Magnolia DSP 的架构简图. Magnolia DSP 采用典型 VLIW 架构, 每周期最多并发 8 条指令, 包括 2 条 ALU 指令, 2 条 MAC 指令, 2 条访存指令和 2 条浮点指令. 其中延时跳转指令和访存指令的延时槽分别为 5 和 3.

4.2 实验平台介绍

选取 Michigan 大学开发的开源计算机软件模拟器 Gem5^[14-15] 作为实验室平台. Gem5 仿真平台采用离散事件驱动模型进行处理器的仿真, 具有周期精度的仿真能力. 现今包括 ARM、Intel 等越来越多的商业公司都在 Gem5 平台上进行包括 Cache、总线协议、流水线等架构探索工作, Gem5 已经成为业内最具影响力的计算机体系架构研究的仿真平台.

4.3 实验所用算法及测试用例

选取局部分支预测和线性值预测分别作为实验所用的分支预测和值预测算法, 并采用预测表项典型大小 2048 作为实验所用表项大小. 实验所用基准测试程序均由 Open64 编译器^[16] 在 O3 选项下编译, 链接 newlib C 函数库.

4.4 实验结果及分析

首先测试 VLIW 架构分支预测与值预测对于通用程序的性能影响. 选取 Dhrystone2.1、Whetstone 以及 Coremark 三个具有广泛应用且有参考价值的通用基准测试程序, 其中, Dhrystone2.1 和 Coremark 均为定点程序, Whetstone 为浮点程序.

从图 8 中可以看出, 分支预测的成功率基本相同. 值预测方面, 由于 Whetstone 是浮点程序, 具有较多的浮点常量, 因此具有较高的值预测成功率. 而 Coremark 有矩阵运算等, 因此值的可预测性较差.

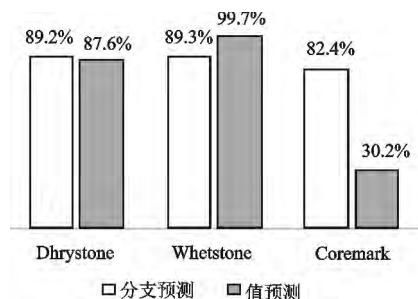


图 8 通用程序的 VLIW 分支预测与值预测预测成功率

图 9 中 Whetstone 虽然值预测成功率很高, 但是 NOP 冗余减少很少, 是因为 Whetstone 中访存指令较少, 大多数的 NOP 指令都是由于其他原因产生.

从图 9 中可以很直观地看出, VLIW 中应用动态预测技术, 可以非常显著地提升通用程序的性能,

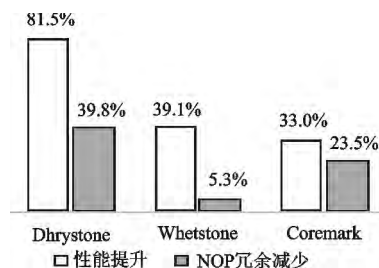


图 9 通用程序的 VLIW 分支预测与值预测带来的性能提升与 NOP 冗余减少

同时降低 NOP 的冗余.

现今商用 VLIW 架构处理器基本都是信号处理 DSP, 因此 VLIW 架构的分支预测与值预测对于 DSP 程序的性能影响尤为重要. 选取 DSPStone^[17] 中的 6 个程序作为 DSP 基准测试程序:

- convolution(卷积);
- fir(FIR);
- fir2dim(图像 FIR 滤波);
- n_real_updates(实数乘加);
- n_complex_updates(复数乘加);
- lms(最小均方).

以上 DSP 基准程序都分为定点和浮点两个版本. 不管是定点还是浮点, 分支预测与值预测的成功率都接近 99%, 考虑到 DSP 程序基本都为循环. 读 0 取数组, 计算等操作, 极高的预测成功率在情理之中.

从图 10 和图 11 中可以看出, 不管对于定点还是浮点的 DSP 测试程序, VLIW 分支预测与值预测均能普遍地提升程序的性能并有效降低 NOP 的冗余率. (convolution 程序 NOP 减少率为 0 是因为该程序几乎没有访存指令).

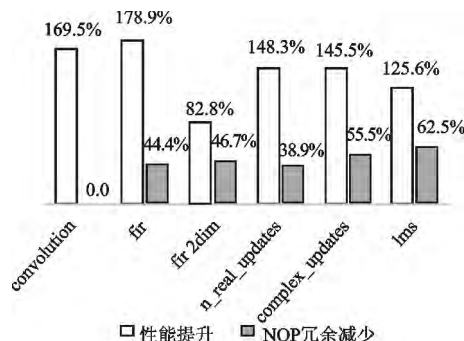


图 10 定点 DSP 程序的 VLIW 分支预测与值预测带来的性能提升与 NOP 冗余减少

而结合前文理论分析, 实验的结果性能提升比理论值更加显著, 说明原有 VLIW 程序不管是 BD 指令还是 LD 指令的延时槽的利用率都较低.

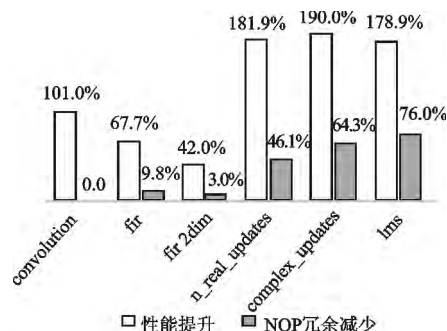


图 11 浮点 DSP 程序的 VLIW 分支预测与值预测带来的性能提升与 NOP 冗余减少

5 结束语

本文通过拓展 VLIW 指令集中的跳转指令和访存指令,使其与指令延时槽个数一一对应,并在动态预测成功或失败时根据相应的指令阻塞或延时流水线,实现了 VLIW 架构下的分支预测和值预测技术.通过在 VLIW 架构下实现动态预测技术,对于通用程序,尤其是 VLIW 化的 DSP 程序,有着显著的性能提升,同时,还可以减小 VLIW 代码的体积,提升内存利用率,间接提升了程序性能.

参考文献:

- [1] Joy W, Tremblay M. Local and global register partitioning in a VLIW processor; U. S. Patent 7,114,056 [P]. 2006-9-26.
- [2] Diao Y, Zhao X. C code optimization for TI C6000 DSPs[J]. Microcomputer Applications, 2007(5): 20.
- [3] Lin T J, Liu C N, Tseng S Y, et al. Overview of ITRI PAC project-from VLIW DSP processor to multicore computing platform[C]// VLSI Design, Automation and Test, VLSI-DAT 2008. IEEE International Symposium on. Taiwan: Hsinchu, IEEE, 2008: 188-191.
- [4] Roberts J S, Zuraski Jr G D. Hybrid branch prediction device with two levels of branch prediction cache; U. S. Patent 7,024,545[P]. 2006-4-4.
- [5] Thuresson M, Spracklen L, Stenstrom P. Memory-link compression schemes: a value locality perspective [J]. Computers, IEEE Transactions on, 2008, 57 (7): 916-927.
- [6] Calus M P L. Genomic breeding value prediction: methods and procedures[J]. Animal, 2010, 4 (2): 157-164.
- [7] Jourdan S J, Miller J A, Morgan S A. Method and apparatus for dynamic branch prediction utilizing multiple stew algorithms for indexing a global history; U. S.

Patent 7,143,273[P]. 2006-11-28.

- [8] Fick D, Dreslinski R G, Giridhar B, et al. Centip3De: A 3930DMIPS/W configurable near-threshold 3D stacked system with 64 ARM Cortex-M3 cores[C] // Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International. China: Beijing, IEEE, 2012: 190-192.
- [9] Kessler C W. Compiling for VLIW DSPs[M]. New York, Springer, 2013: 1177-1214.
- [10] Malik A M, McInnes J, Van Beek P. Optimal basic block instruction scheduling for multiple-issue processors using constraint programming[J]. International Journal on Artificial Intelligence Tools, 2008, 17(1): 37-54.
- [11] Liu S, Eisenbeis C, Gaudiot J L. Value prediction and speculative execution on GPU[J]. International Journal of Parallel Programming, 2011, 39(5): 533-552.
- [12] Yang X, He H. An advanced compiler designed for a VLIW DSP for sensors-based systems[J]. Sensors, 2012, 12(4): 4466-4478.
- [13] 陈峰扬, 杜勇, 郭德源, 等. 一种 VLIW—Superscalar 混合微处理器结构[J]. 微电子学与计算机, 2013, 30 (11): 1-5.
- [14] Binkert N, Beckmann B, Black G, et al. The gem5 simulator[J]. ACM Sigarch Computer Architecture News, 2011, 39(2): 1-7.
- [15] Butko A, Garibotti R, Ost L, et al. Accuracy evaluation of GEM5 simulator system[C] // Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on. UK: York, IEEE, 2012: 1-7.
- [16] Lin Y T, Wang S C, Shih W L, et al. Enable OpenCL compiler with Open64 infrastructures[C]// High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on. UK: Liverpool, IEEE, 2011: 863-868.
- [17] Agarwal N, Dimopoulos N. A DSP stone benchmark of CoDeL's automated clock gating platform[C]// VLSI, ISVLSI07. Brazil, IEEE Computer Society Annual Symposium on. IEEE, 2007: 508-509.

作者简介:

李笑天 男, (1989-), 硕士. 研究方向为微处理器设计. E-mail: Lixmail@163.com.

郭德源 男, (1984-), 硕士, 工程师. 研究方向为数字信号处理器软件工具链设计.

何虎 男, (1974-), 博士, 副教授. 研究方向为数字信号处理器领域.