融合动态采样剖析的可重构指令集处理器

张惠臻[1] 王 超[2]
(华侨大学计算机科学与技术学院 厦门 361021)[1]
(中国科学技术大学计算机科学与技术学院 合肥 230027)[2]

摘 要 可重构指令集处理器能够根据应用程序特点动态扩展其指令集,其硬件架构和软件工具的设计与传统设计有很大不同。在研究可重构指令集处理器软硬件特性的基础上,提出一种集成动态采样剖析硬件的可重构指令集处理器架构。该处理器具有3种不同的工作模式,它通过剖析硬件采样获取程序热点,利用配套工具链半自动地完成指令扩展生成、编译器重定向和可编程硬件逻辑配置,从而获得在不同嵌入式应用领域的硬件适应性和软件兼容性。针对性的实验结果表明,该处理器架构的采样剖析机制准确有效,并且在增加有限的硬件开销的情况下,能够很好地适应应用变化。

关键词 计算机系统结构,可重构指令集处理器,指令扩展,动态采样剖析
中图法分类号 TP302 **文献标识码** A

Reconfigurable Instruction Set Processor Integrating Dynamic Sampling Profiler

ZHANG Hui-zhen[1] WANG Chao[2]
(College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China)[1]
(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)[2]

Abstract Reconfigurable instruction set processor (RISP) can dynamically extend its instruction set based on specific applications characteristics. The design of its hardware architecture and software tools is very different from the traditional design ideas. Based on the research for the features of RISP, a RISP integrating dynamic sampling profiler (DSP-RISP) was proposed. DSP-RISP has three kinds of working modes, obtaines the hot spots of applications through a hardware profiler with sampling mechanisms and then with the software tool chain semi-automatically completes the work such as instruction extension and generating, compiler retargeting and programmable hardware logic configuration to achieve the hardware adaptability and software compatibility for various embedded applications. Experimental results show that the sampling profiler of DSP-RISP is accurate and effective and under the limited hardware overhead, DSP-RISP can well adapt to application changes.

Keywords Computer architecture, Reconfigurable instruction set processor, Instruction extended, Dynamic sampling profiling

1 引言

嵌入式处理器设计需要兼顾性能和灵活性两方面的因素,而可重构指令集处理器(Reconfigurable Instruction Set Processors, RISP)[1]被视作下一代的处理器架构。它通过可编程逻辑器件,将处理器核与可重构硬件逻辑结合,根据应用程序特点适应性地改变其硬件结构,优化原有指令集和功能部件,从而使应用程序在具有高灵活性的同时获得最佳的执行性能。

相比于传统的嵌入式处理器,RISP基于专用指令集处理器(Application Specific Instruction-set Processor, ASIP)[2]的设计思想,更专注于对指令集的可重构定制实现。其指令集一般分为两个部分:1)固定的基本指令集,在处理器初始设计完成后不更改,用于完成一般的计算和控制任务;2)扩展的定制指令集,根据应用特征的不同而变化,完成应用中频繁出现的、对系统性能有关键影响的计算任务。与ASIP不同的是,RISP的定制指令由处理器根据应用特征动态生成,所需的处理单元和数据通路也在可重构硬件逻辑中动态实现。RISP可被用作具有同一基本指令集的ASIP系列产品的快速原型系统,随着产品的不同应用和技术成熟而改进重构其硬件结构,有效降低重设计带来的成本代价。同时,RISP指令集根据应用程序特征而动态定制的特性,使得传统软件工具链设计不适合于RISP的应用开发,需要有新的软件工具链设计方法,以适应RISP动态指令集的特点,从而充分挖掘RISP的执行性能。

本文针对RISP的软硬件特性,基于前期的工作成果,提

到稿日期:2012-09-15 返修日期:2012-11-20 本文受国家自然科学基金青年基金(61202053),华侨大学引进人才科研基金(12BS214)资助。
张惠臻(1983—),男,博士,讲师,主要研究方向为嵌入式软硬件系统、可重构计算、异构多核系统、编译优化,E-mail:zhanghz1006@gmail.com;
王 超(1985—),男,博士后,主要研究方向为可重构计算、异构多核系统。

出一种在处理器内部融合动态采样硬件剖析器的 RISP 处理器架构(RISP integrating Dynamic Sampling Profiler, DSP-RISP),并给出其配套软件工具链的设计方法。该处理器利用集成的硬件剖析器,通过一定的采样策略获取程序实际执行时的热点代码用作指令扩展的备选,然后结合配套的软件工具链选择并生成扩展的定制指令,完成编译器的重定向设计和定制指令功能部件的重构配置,从而实现软硬件系统的适应性更新,使应用程序获得最佳的执行环境。

本文第 2 节介绍 RISP 软硬件架构和动态剖析技术等相关研究工作;第 3 节详细描述 DSP-RISP 的体系架构、采样剖析机制、工作模式及软件工具链设计方法;第 4 节给出针对性的实验结果及分析;最后对所做工作进行总结及展望。

2 相关工作

2.1 RISP

RISP 的研究是目前处理器设计领域的热点。Beeck[3] 研究了一种可配置的 RISP 模型,以便通过设计空间搜索的方法实现不同约束要求的 RISP。Campi[4] 研究的 XiRisc 采用载入/存储的架构,所有的数据访问都通过共享寄存器文件进行,只使用了很简单的两种扩展指令。Iqbal 等[5] 设计的 RT-RISP 将每条指令看作可加载/卸载的独立模块,利用部分可重构技术,根据当前的应用需求动态地换入换出,从而实现处理性能的提升。Vassiliadis 等[6] 设计的 Molen 不修改处理器核以支持可重构逻辑,也不采用软硬件协同设计的方法,而将硬件重构配置过程作为处理器设计的一部分,把应用于可重构逻辑的计算任务看作是原子操作,通过可重构微代码的形式来实现。此外,专门针对 RISP 的某方面性能的研究也在进行。Barat[7] 进行了低功耗 RISP 的研究,建立了一种类似 Wattch 的功耗模型,对所设计的粗粒度的 RISP 进行功耗评估,减少了 18%的功耗。表 1 是总结得出的近 10 年中学术界一些典型 RISP 处理器架构的比较[3-6,8-11]。

表 1 近 10 年出现的典型 RISP 类型处理器比较

RISP 名称	CRISP	XiRisc	Molen	AMIDAR	Vulcan	IRISP	TEMPO	ARISE	RT-RISP
时间	2003	2003	2004	2005	2006	2007	2007	2007	2009
处理器类型	RISC	VLIW	General	General	General	VLIW	General	MIPS	General
应用类型	多媒体	DSP	图形图像	一般	智能控制	一般	一般	一般	一般
可重构逻辑耦合方式	RFU	RFU	协处理器	RFU	RFU	RFU	RFU	Attached	RFU
可重构粒度	粗	细	细	粗	细	细	粗	粗	粗
定制指令类型	专用	专用	专用	一般化	专用	一般化	一般化	专用	一般化
指令编码	ID	ID	Address	—	ID	—	—	ID	ID
是否有配置表	是	是	否	否	否	是	否	是	否
操作数域	—	固定	—	固定	固定	固定	—	固定	固定
共享寄存器组	是	是	否	是	是	是	否	是	是
存储端口	有	无	有	无	有	无	有	有	有
动态可重构	不支持	支持	不支持	支持	支持	支持	支持	支持	支持
配置方式	软件	控制器	软件	控制器	控制器	控制器	控制器	软件	控制器
配置时停用 RFU	是	否	是	否	是	否	否	否	否
加速比	2.5	13.5	2.96	1.29	3.8	1.84	25	7.5	2

2.2 动态剖析技术

程序执行时 90%以上的执行时间消耗在大约 10%的代码(称为热点代码)上,找出程序的热点代码并优化将显著提高程序的执行效率,从而带来系统整体性能的提升,这一观点被业界普遍接受。剖析技术作为一种有效的确定程序运行时热点代码的性能分析技术,受到学术界的广泛关注。动态剖析则是众多剖析技术中的热点。

动态剖析技术是一个在线过程,通过跟踪程序的实际执行行为,不断收集和更新程序动态执行信息,准确找到具有性能提升潜力的热点代码。Law[12] 和 Sriraman[13] 突破传统的函数体和循环体的边界限制,在更大的程序范围内收集执行代码的运行信息以获得对热点代码的更准确定位,但是这种方法的剖析开销过大。为降低剖析开销,Yasue[14] 提出结合动态插桩技术的一次性路径剖析方法来收集程序运行信息;Bond 等[15] 基于剖析制导的思想,研究利用边剖析技术简化路径剖析的方法。此外,不少学者致力于将采样技术应用到动态剖析技术中。Anderson[16] 通过对程序计数器的采样收集程序基本块的相关信息。Bond 和 Kathryn[15] 对传统的 B-L 路径算法做了采样优化,根据剖析信息为每条路径分配唯一标识,不过这种方法需要操作系统的支持。Arnold 和 Grove[17] 提出了一种新颖的采样算法用于剖析过程,它能够准确有效地获得程序运行时信息;Bond[18] 对此算法做了优化,进一步降低了算法开销。

3 DSP-RISP 设计

3.1 DSP-RISP 的体系结构

DSP-RISP 在单块 FPGA 上实现,总体架构如图 1 所示。从逻辑上看,DSP-RISP 可分为 3 个部分:1)左侧的嵌入式通用处理器部分。主要包括了处理器核、译码器和寄存器组等一般通用处理器的常规组成。这部分主要负责完成整个 DSP-RISP 的总体控制,并执行程序中的基本指令。2)右侧的可重构逻辑部分。这部分是 DSP-RISP 的可变组成部分,包含了采样剖析器、配置控制器、配置信息存储器、封装接口和可重构阵列等模块。配置控制器是这部分的核心,控制着其他模块的使用;配置信息存储器保存配置相关的数据,用于可重构逻辑的配置;而采样剖析器用于监测程序执行,遵循一定的采样算法获取程序的热点代码;可重构阵列用于实现扩展的定制指令的功能部件,其架构与功能将在配置控制器的控制和配置信息的指导下动态实现;封装接口用于统一可重构阵列与处理器中其他模块的连接方式,使可重构阵列内部功能部件的重构实现对外透明。3)上方的指令/数据缓存及总线接口部分。这部分主要完成与片外存储设备及其他设备的

接口与缓存。

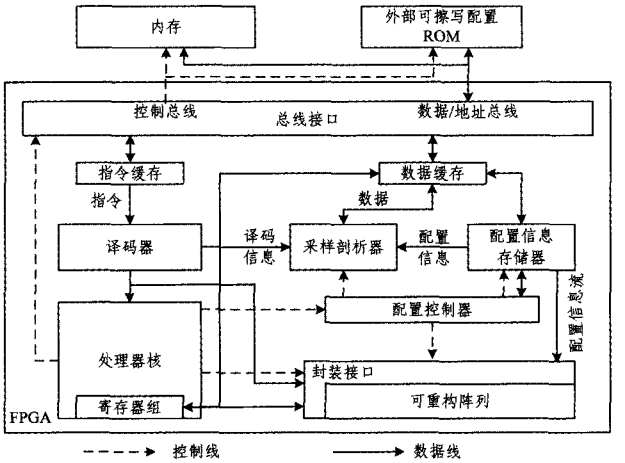

图 1　DSP-RISP 总体架构

3.2　DSP-RISP 的动态采样剖析机制

DSP-RISP 的采样剖析器是在前期工作设计的硬件路径剖析器[19]的基础上，将 Arnold-Grove(以下简称为 A-G)采样算法[17]应用到动态剖析过程中，并在处理器内部硬件化实现而得到的。其支持如图 2 所示的 3 种采样机制：(a)计时中断采样机制；(b)标准的 A-G 采样机制；(c)简化的 A-G 采样机制。图中的横轴代表程序的执行过程，轴下方的箭头表示计时器中断触发时刻；轴上的方块代表采样时机，黑色方块表示实际执行了采样。

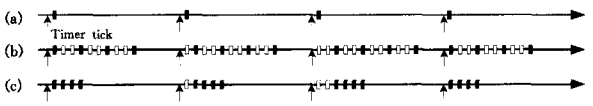

图 2　DSP-RISP 的采样机制

DSP-RISP 在 CHDPP 的控制寄存器中增加了 Timer_Enable、AG_Enable 和 SAG_Enable 3 个信号来控制上述 3 种采样机制，使用两个寄存器保存 A-G 采样算法的两个参数值：N_SAMPLE，即每个计时周期内执行的采样次数；N_STRIDE，即两次采样之间间隔的采样机会的个数。如图 2(b)中，N_SAMPLE 为 4，N_STRIDE 为 2。同时使用两个计数器 C_STRIDE 和 C_SAMPLE 分别对计时中断次数和每个计时周期内已进行的采样次数进行计数。

当计时器递减到 0 触发计时中断时，若 Timer_Enable 信号有效，则执行基本的计时中断采样一次，以更新存储的执行路径等相关信息。由于计时时间一般较长，因此其较适用于长时间执行的服务器程序。若 AG_Enable 信号有效，将进行标准的 A-G 采样。首先判断 C_STRIDE 的值是否等于 N_STRIDE，若相等，则将其清零，否则将 C_STRIDE 的值作为采样开始前的跳步值；然后剖析器采样一次，并将 C_STRIDE 和 C_SAMPLE 的值加 1；接着每隔 N_STRIDE 个采样机会采样一次，同时将 C_SAMPLE 的值加 1 直至其值等于 N_SAMPLE 时，停止采样，等待下一次计时中断。若 SAG_Enable 有效，则进行简化的 A-G 采样，过程与标准的 A-G 采样类似，只是在两次采样之间不作间隔，而在开始采样后连续进行 N_SAMPLE 次采样，如图 2(c)所示。

3.3　DSP-RISP 的工作模式

DSP-RISP 使用通用嵌入式处理器核执行普通指令，用可重构阵列来实现定制指令功能。整个处理器的执行过程有 3 种模式，如图 3 所示。

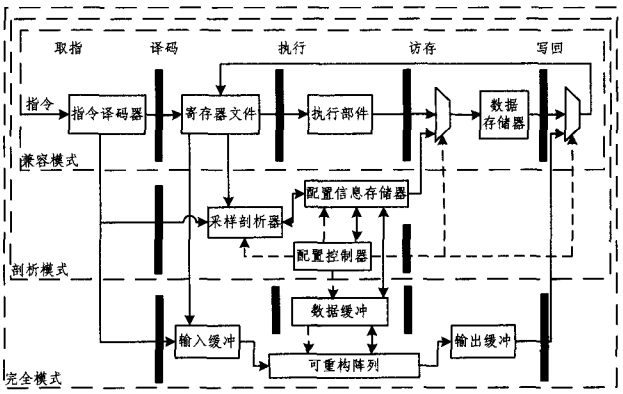

图 3　DSP-RISP 的工作模式

3.3.1　完全模式

这是 DSP-RISP 正常工作时的执行模式。程序指令经过译码之后，根据译码结果判断是发送到处理器核还是可重构阵列。若是普通指令，则由通用处理器核执行；若是定制指令，则将其发送到封装接口的输入缓冲，待可重构阵列对其进行处理，处理器核可以继续执行其后续指令。

可重构阵列的处理可能需对阵列架构进行重构，以实现当前定制指令所需的功能单元。这取决于该指令相应的功能部件是否已在可重构阵列中实现。若存在，则直接执行指令；否则，配置控制器将调用配置信息存储器中的配置数据，对可重构阵列进行重构后再执行指令。

采样剖析器在这个模式下处于采样剖析的方式，可以根据需要进行配置：其仅获取定制指令或是同时剖析普通指令，这样有助于控制整个系统的开销。

3.3.2　剖析模式

在剖析模式下，所有的指令都由通用处理器核执行，可重构阵列不工作。采样剖析器将根据应用程序预分析信息进行配置，以实现对程序执行的全过程剖析或采样剖析，其剖析工作原理与 CHDPP 相似。剖析得到的程序热点路径等相关信息保存到配置信息存储器中。配置信息存储器的数据将进入处理器基本的数据通路，进而用于指导定制指令的生成和编译过程对指令序列的修改优化。处理器一般在应用程序有较高的性能要求时进入剖析模式，以定制指令集和重构硬件架构，然后再转入完全模式执行。

3.3.3　兼容模式

这种执行模式就是基本的通用处理器执行模式，所有指令都由通用处理器核执行，不开启剖析功能。在系统刚启动时，DSP-RISP 运行于兼容模式，进行系统初始化。此时处理器加载并执行一个初始化程序，执行必要的配置，使可重构逻辑部分的资源处于就绪状态，然后转入完全模式或者剖析模式。

3.4　DSP-RISP 的软件工具链设计

DSP-RISP 的有效工作需要有配套的软件工具链支持，以将定制指令应用于程序二进制代码中。图 4 给出了设想的 DSP-RISP 的配套工具链的一种设计框架。

应用程序源代码首先经过快速编译后在 DSP-RISP 上执

行，此时 DSP-RISP 工作于完全模式或剖析模式，采样剖析器跟踪执行过程获得热点代码等相关信息，提供给指令选择生成器。指令选择生成器根据设计好的定制指令选择和生成算法[20]得出新的定制指令集，并将其与程序相关信息分别提供给代码映射注释器、可重定向编译器和可重构配置信息生成器。

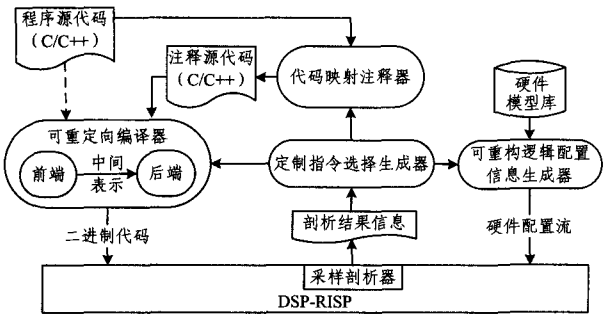

图 4　DSP-RISP 的配套工具链设计框架

代码映射注释器在程序的静态源代码中匹配定制指令可能使用的位置，进行注释标识，以辅助下一步使用定制指令的新的程序二进制代码的编译生成；可重定向编译器将根据定制指令信息修改其指令模板文件、机器描述文件等与机器体系结构相关的部分，完成编译后端的重定向，通过新的代码优化生成算法得到应用了定制指令的程序二进制代码；可重构配置信息生成器根据定制指令功能，结合硬件模型库，生成完成定制指令对应的功能部件的硬件结构信息，从而指导片上可重构阵列的配置过程。

4　实验与结果分析

4.1　采样机制的有效性实验

采样可能会降低剖析的精度，为了验证 DSP-RISP 中融合的采样机制的效果，使用不同的采样方法进行了实验，结果如图 5 所示。图中 Base 表示不进行采样剖析，AG 和 SAG 分别表示标准的 A-G 采样机制和简化的 A-G 采样机制。图示中的括号对，前一个值表示 N_STRIDE，后一个值表示 N_SAMPLE。实验使用的机器环境为 AMD Athlon™ 1GHz 处理器，内存为 384MB，定时器频率为处理器工作时钟频率的 1/1000；测试程序采用 gcc 4.1.0 版本进行编译，使用-O3 编译选项。

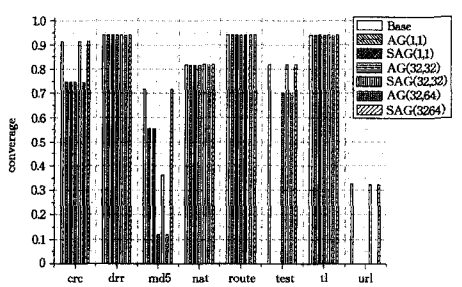

图 5　采样机制有效性实验结果

图中显示，NetBench 测试集中的 drr、nat、route 和 tl 等应用程序使用采样机制找到的热点代码的路径覆盖率与不使用采样机制得到的结果差不多。对于其他几个应用，SAG 采样机制得到的结果明显优于 AG 采样机制；同时，不同的跨步长度和采样次数得到的结果不同，相比而言，SAG(32,64)得到的结果最好，基本都能得到与不使用采样机制一样的覆盖率。此外，通过实验得知，在 SAG(32,64)的采样设置下，CHDPP 中更新剖析信息的硬件逻辑的工作时序要求可以降低为原来的 6.4%。

4.2　硬件资源开销

为了评估在处理器中融合采样剖析器的额外硬件开销，使用 Verilog 语言实现了采样处理器，并通过 ModelSim 仿真软件和综合工具 Synplify Pro 8.0 完成其硬件仿真与综合。选用的 FPGA 器件型号为 Xilinx 公司 Virtex2 系列的 XC2V500，占用资源情况如表 2 所列。

表 2　采样剖析器资源占用情况

资源类型	数目
Total LUTs	90
I/O Register bits	43
BUFGP	1
Global Clock Buffers	1
I/O primitives	100
Register bits(no I/Os)	65

结束语　可重构指令集处理器的设计与传统处理器设计有着很大的不同，本文提出了一种将动态采样剖析机制融合到处理器内部的可重构处理器 DSP-RISP，对其体系结构、工作模式以及使用的动态采样剖析机制进行了详细的设计描述，并给出了其配套的软件工具链的一种设计设想。在此基础上，对处理器的采样机制和硬件实现等关键环节进行了针对性的实验。实验结果表明了该处理器架构的有效性和实用性。

配套软件工具链的功能实现对 DSP-RISP 架构的有效工作有着重要的影响。下一步的工作将集中于对 DSP-RISP 工具链的研究。

参 考 文 献

[1] Barat F, Lauwereins R. Reconfigurable instruction set processors: a survey[A]// RSP 2000. Proceedings of 11th International Workshop on Rapid System Prototyping[C]. 2000

[2] Keutzer K, Malik S, Newton A R. From ASIC to ASIP: the next design discontinuity[A]// Computer Design: VLSI in Computers and Processors. Proceedings of 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors [C]. 2002

[3] Beeck P O, Barat F, Jayapala M, et al. CRISP: A Template for Reconfigurable Instruction Set Processors[A]// FPL 2001. Proceedings of the 11th International Conference on Field-Programmable Logic and Applications[C]. 2001

[4] Campi F, Toma M, Lodi A, et al. A VLIW processor with reconfigurable instruction set for embedded applications[A]// ISSCC 2003. Proceeding of 2003 IEEE International Solid-State Circuits Conference[C]. 2003

[5] Iqbal M A, Awan U S. Run-Time Reconfigurable Instruction Set Processor design: RT-RISP[A]// IC4 2009. Proceeding of 2nd International Conference on Computer, Control and Communication[C]. 2009

[6] Vassiliadis S, Wong S, Gaydadjiev G, et al. The MOLEN polymorphic processor[J]. IEEE Transactions on Computers, 2004, 53(11): 1363-1375

[7] Barat F, Jayapala M, Vander T, et al. Low Power Coarse-Grained Reconfigurable Instruction Set Processor[A]// FPL 2003. Proceeding of the 3th Intl. Conf. on Field Programmable Logic and Applications[C]. 2003

[8] Gatzka S, Hochberger C. The AMIDAR Class of Reconfigurable Processors[J]. Supercomput, 2005, 32(2): 163-181

[9] Ferreira V M, Gauthier L, Kando T, et al. REDEFIS: a system with a redefinable instruction set processor[A]// ICSD 2006. Proceedings of the 19th Annual Symposium on Integrated Circuits and Systems Design[C]. Ouro Preto: ACM, 2006: 14-19

[10] Ur R A, Syed A A, Iqbal M A. Intelligent Reconfigurable Instruction Set Processor (IRISP) Design[A]// INMIC 2007. Proceeding of 2007 IEEE Internationa Multitopic Conference[C]. 2007

[11] Vassiliadis N, Theodoridis G, Nikolaidis S. The ARISE Reconfigurable Instruction Set Extensions Framework[A]// IC-SAMOS 2007. Proceeding of International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation [C]. 2007

[12] Law J, Rothermel G. Whole program Path-Based dynamic impact analysis[A]// Proceedings of the 25th International Conference on Software Engineering[C]. IEEE Computer Society: Portland Oregon, 2003: 308-318

[13] Sriraman T, Zhang X, Rajiv G. Extending path profiling across loop backedges and procedure boundaries[A]// CGO 2004. Proceedings of International Symposium on Code Generation and Optimization[C]. 2004

[14] Yasue T, Suganuma T, Komatsu H, et al. An efficient online path profiling framework for Java just-in-time compilers[A]// PACT 2003. Proceeding of 12th International Conference on Parallel Architectures and Compilation Techniques[C]. 2003

[15] Bond M D, McKinley K S. Practical path profiling for dynamic optimizers[A]// CGO 2005. Proceeding of International Symposium on Code Generation and Optimization[C]. 2005

[16] Anderson J M, Berc L M, Dean J, et al. Continuous profiling: where have all the cycles gone? [J]. ACM Transactions on Computer Systems, 1997, 15(4): 357-390

[17] Arnold M, Grove D. Collecting and exploiting high-accuracy call graph profiles in virtual machines[A]// CGO 2005. Proceeding of International Symposium on Code Generation and Optimization[C]. 2005

[18] Bond M D, McKinley K S. Continuous path and edge profiling [A]// MICRO-38. Proceedings of 38th Annual IEEE/ACM International Symposium on Microarchitecture[C]. 2005

[19] 张惠臻，周学海，纪金松，等. 可配置的热点路径动态剖析器的硬件实现[J]. 系统工程与电子技术，2009，31(9)：2254-2259

[20] 纪金松. 基于动态指令集的自适应处理器的关键技术研究[D]. 合肥：中国科学技术大学，2008

（上接第 7 页）

[64] Jotsov V. Emotion-Aware Education and Research Systems [J]. Issue s in Informing Science and Information Technology, 2009, 6: 779-794

[65] Wilson B C, Shrock S. Contributing to success in an introductory computer science course: a study of twelve factors [J]. ACM SIGCSE Bulletin, 2001, 33(1): 184-188

[66] Besie C, et al. An Examination of Age, Race, and Sex as Predictors of Success in the First Programming Course [J]. Journal of Informatics Education Research, 2003, 5: 51-64

[67] Rountree N, et al. Interacting factors that predict success and failure in a CS1 course[R]. Working group reports from ITiCSE on Innovation and technology in computer science education. Leeds, United Kingdom, ACM Press, 2004

[68] Rountree N, et al. Predictors For success in studying CS[C]// Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. Norfolk, Virginia, USA: ACM Press, 2004

[69] Lister R, et al. A Multi-National Study of Reading and Tracing Skills in Novice Programmers[R]. Working group reports from ITiCSE on Innovation and technology in computer science education. New York: ACM, 2004: 119-150

[70] Fincher S, et al. Programmed to succeed? a multi-national, multi institutional study of introductory programming courses[R]. Computing Laboratory Technical Report 105. C. University of Kent, UK, 2005

[71] Raadt M D, et al. Approaches to learning in computer programming students and their effect on success[J]. Higher Education in a changing world: Research and Development in Higher Education, 2005, 28: 407-414

[72] Simon, Cutts Q, Fincher S, et al. The ability to articulate strategy as a predictor of programming skill[C]BProc Eighth Australasian Computing Education Conference. Australia, 2006: 181-188

[73] Tolhurst D, et al. Do map drawing styles of novice programmers predict success in programming? A multi-national, multi-institutional study[C]// Proc Eighth Australasian Computing Education Conference. Australia, 2006: 213-222

[74] Wray S. SQ Minus EQ can Predict Programming Aptitude[C]// PPIG'07. 2007: 243-254

[75] Simon, et al. Predictors of success in a first programming course [C]// Eighth Australasian Computing Education Conference. Hobart, 2006: 189-196

[76] Jones S, Burnett G. Spatial Ability and Learning to Program[C]// PPIG'07. 2007: 229-242

[77] Ivins J, Ong M P-S. Psychometric Assessment of Computing Undergraduates [C]// 17th Workshop of the Psychology of Programming Interest Group. Sussex University, June 2005: 305-319

[78] Swanson L H. Influence of metacognitive knowledge and aptitude on problem solving[J]. Journal of Educational Psychology, 1990, 82: 306-314

[79] Plonka L, Segal J, Sharp H, et al. Collaboration in Pair Programming: Driving and Switching [J]. Lecture Notes in Business Information Processing, 2011, 77(1): 43-59

[80] Huang Fu-qun, Liu Bin. Systematically Improving Software Reliability: Considering Human Errors of Software Practitioners [C]// Psychology of Programming Interest Group Annual Conference 2011. Doctoral Consortium, 2001