一种处理器无关的 trace 协处理器自动产生方法

桑胜田[1],邱善勤[2],李晓明[1],喻明艳[1]
(1.哈尔滨工业大学微电子中心,黑龙江哈尔滨 150001; 2.工业和信息化部软件与集成电路促进中心,北京 100038)

摘 要: 本文提出了体系结构无关的程序热 trace 提取方法,把基于 trace 的软硬件划分从机器代码层次提高到了中间代码表示的控制数据流图层次,从而实现了处理器体系结构无关的 trace 协处理器自动产生方法;提出了基于散列签名的 trace 预测方法,支持包含环路结构的 trace 预测,提高了平均命中率.基于上述工作实现的协处理器自动产生方法,可以作为系统级设计优化工具应用于现有的 SoC 软硬件开发流程.实验表明,与基于机器指令 trace 的方法相比,本文方法获得的系统平均性能提高了 22.6%.

关键词: 协处理器自动产生; 中间表示; trace; 推测执行
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2011) 02-0402-06

A Processor-Independent Trace Co-Processor Synthesis Method

SANG Sheng-tian[1], QIU Shan-qin[2], LI Xiao-ming[1], YU Ming-yan[1]
(1. *Microelectronics Center of Harbin Institute of Technology, Harbin, Heilongjiang* 150001, *China*;
2. *Software and Integrated Circuit Promotion Center, Ministry of Information Industry, Beijing* 100038, *China*)

Abstract: The architecture-independent hot trace extraction method is presented, with which the processor architecture-independent trace co-processor synthesis is achieved by promoting trace-based HW/SW partitioning from the level of machine instruction to that of control data flow graph of intermediate code. To support trace predication in cyclic structure for better average hit rate, hash-signature based trace predication is proposed. Based on the above work, a processor-independent co-processor synthesis method is implemented, which can be seamlessly integrated with the hardware and software development process as a system-level design optimization tools. The experiment reveals that the result system performance is increased by 22.6% over instruction trace based method.

Key words: co-processor synthesis; intermediate representation; trace; speculative execution

1 引言

把数字通信、多媒体等应用的算法核心以协处理器的形式实现为专用数据通路(data-path),能以很小的硬件代价显著提高系统性能、降低功耗[1,2].针对特定应用的协处理器属于专用协处理器,对采用新算法的应用,很难重用已有的专用协处理器模块,所以自动化的专用协处理器产生方法,对提高设计效率,降低设计成本具有重要意义.

很多研究工作集中在专用协处理器自动化的设计方法上,其中 Cathedral III[3]采用 SILAGE 语言描述作为输入自动综合面向数字信号处理的协处理器.Critical Blue 开发了基于二进制应用程序产生协处理器的商业工具[4].最近 Michalis D.Galanis 等人提出了基于灵活计算组件(Flex Compute Component)的协处理器综合方法[1].这些方法中协处理器模块的选择都是基于高级语言结构进行的,以程序核心循环为主,对于核心循环不明显,或循环内部结构复杂的程序,这种方法则不易获得理想的效果.

协处理器综合的核心问题是软硬件划分.文献[5~9]等分别尝试了新的软硬件划分模型和算法.目前的软硬件划分方法中,一定程度上利用了系统执行时的动态信息,例如,利用程序 profile 信息来选择系统优化的热点、估算划分对象的硬件实现时加速贡献等,或者利用 profile 信息指导设计空间探索[10].我们前期的工作探索了基于 trace 的软硬件划分方法[9],更进一步地利用系统动态特征,根据运行时的动态特征扩展程序静态视图,构造适合硬件实现的划分对象.

本文提出了基于体系结构无关的中间表示代码(IRC, intermediate representation code)的程序热 trace 提取方法,把基于 trace 的软硬件划分从机器代码层次提高到了机器无关的控制数据流图层次,并进一步提高了协

收稿日期:2009-12-24;修回日期:2010-03-28

处理器加速效果;同时本文提出了基于散列签名的软件 trace 预测算法,显著提高了软件 trace 预测的命中率.在此基础上,实现了一种由目标应用的算法描述自动产生高效协处理器的方法.

2　基于 trace 的软硬件划分

基于 trace 的软硬件划分的基本原理如图 1 所示,其中虚线表示的节点是利用热 trace 构造的对象,软硬件划分后该节点作为硬件模块实现.虚线节点内的操作来自于原始控制数据流图,但它们的组织方式却是依据动态特征优化的.

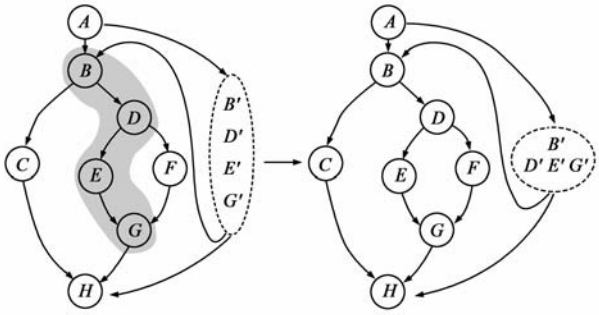

图1　基于trace的对象构造

基于 trace 的软硬件划分采用分支断言把热 trace 转化成利于硬件实现和软硬件交互的推测执行单位.对于协处理器硬件实现来说,推测执行允许更激进的并行调度,从而提高了并行度.

如果任何一个断言失败,表示推测执行错误,执行结果被放弃,此时要沿着图中由虚线节点到节点 B 的边恢复软件执行;trace 预测用来保证只在最适合的时机调用协处理器模块,提高推测执行的成功率.

文献[9]的 trace 收集与热 trace 提取以一种具体机器体系结构(SPARC v8)的指令作为表示形式,使得软硬件划分依赖于具体的编译器和特定机器的指令.文献[9]采用基于 path-profile 的软件 trace 预测方法,由于其路径记录无法跨越环路结构的反向边,该方法对无环子图的 trace 预测效果较好,但是子图中的环路会影响预测准确率.

3　体系结构无关的程序动态特征分析

3.1　程序表示层次的影响

基于 trace 的软硬件划分方法的关键步骤之一就是获取并分析程序的动态行为信息,发现程序行为模式特征,这需要对全程序 trace 进行分析,按照重复出现的频率选择关键 trace.程序分析可以在(1)高级语言源程序层次上和(2)机器语言层次上进行.

前者适合静态分析和有限的动态分析,例如基于 profile 结果反标的动态分析,这对于传统的基于核心循环的软硬件划分比较适用,但基于 trace 的程序分析希望在类似于指令的粒度上定位系统的热点,并以 trace 为工具扩展程序结构,由于高级语言程序语法要求和逻辑结构的制约,无法对程序进行细粒度动态分析和结构变换.

后者基于机器指令的 trace 收集和分析具有足够细的粒度,可以获得足够的动态信息,但可执行的机器指令表现形式,一方面与特定的机器体系结构相关,无法实现独立于特定处理器的高层次分析和变换,使得程序分析和软硬件划分都是针对特定处理器的.本文的思想是实现一种自动的系统级协处理器生成方法,系统级设计工具应处于系统开发流程的早期阶段,并且独立于特定的处理器.

另一方面程序从高级语言到机器代码,经过了编译、优化和链接等多个步骤,这些步骤(特别是机器代码生成步骤)会引入很多机器和运行环境相关的操作,例如寄存器溅出、堆栈处理等,这些对于基于 trace 的细粒度分析产生了干扰,这种编译和运行系统引入的辅助性指令(以下称为"噪声指令")会干扰程序中操作的频谱特性,影响对程序算法本身固有属性的分析.图 2 是采用 SPARC v8 gcc 编译的程序的噪声分析.

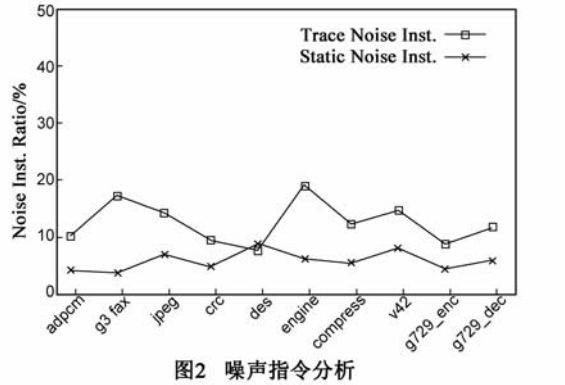

图2　噪声指令分析

10 个实验程序中,噪声指令约占总量的 4~9%,但对于程序热点分析来说影响十分可观,这些噪声指令所构成的 trace 在热 trace 中所占比例可达 19.1%,而这些"伪"热 trace 并不是程序算法本身的计算热点,完全是由特定机器的编译环境引入的.所以,基于 trace 的程序分析和软硬件划分既不能像基于程序循环的软硬件划分那样,在高级语言层次上进行,又不适合使用传统的机器指令级 trace 分析.

3.2　体系结构无关的中间表示

为了使基于 trace 推测执行的协处理器综合方法能独立于具体处理器体系结构和软硬件开发环境,需要以一种体系结构无关的中间表示代码(IRC)进行分析和程序变换.这种中间表示必须具备以下特性:

(1)完备性　包含足够丰富的操作、数据类型,能够等价表示任何 C/C++ 程序的语义.

(2)简约性、中立性 仅提供最常用的运算操作和数据类型,不包含任何处理器特有的操作和数据类型.

(3)细粒度 操作、数据类型足够原始、底层,能在最基础的原语粒度上表示和分析程序.

(4)可执行性 这种 IRC 形式的程序能够像机器指令那样执行,从而支持动态行为分析.

此外还希望这种 IRC 易于进行优化变换,并且能够与 C/C++ 程序之间进行转化.

为了满足以上需求,本文基于 LLVM[11]的中间代码设计了与体系结构无关的程序表示,并实现了针对该程序表示的仿真执行引擎,用于进行体系结构无关的 trace 收集和分析.

本文的 IRC 具有类似于 RISC 指令的抽象操作,寄存器是无限数量的虚拟寄存器,程序表示采用静态单赋值(Static Single-Assignment)形式.一组单入口单出口的 IRC 指令序列构成一个基本块,基本块内部的 IRC 指令间的数据依赖关系构成了数据流图,基本块之间显式的控制关系构成了控制流图.这种形式的指令构成的控制数据流图反映了程序算法中的计算操作的内容和逻辑关系,不含任何具体处理器引入的额外的操作,如寄存器换出(register spill)等;同时,这种 IRC 又能象机器指令一样在仿真执行引擎上执行,从而支持程序动态特性的分析.

中间代码形式表示的控制数据流图是本方法的核心数据形态,是程序静态、动态特性分析和软件 trace 预测等代码变换的基本形式.

在执行引擎上运行程序的 IRC 可以收集全程序的体系结构无关的 trace,即程序执行的全部 IR 指令构成的线性序列,热 trace 是这个指令序列中出现频率达到一定高度的子序列.在全程序 trace 中寻找热 trace 需要在分析过程中发现子序列出现的模式,并定位出每一个模式的实例,本文采用 SEQUITUR 算法[9]提取 IRC 程序执行中的热 trace.

3.3 基于散列签名的 trace 预测

实验表明热 trace 通常出现在包含环路的子图中,为此本文提出了基于散列签名的 trace 预测方案,解决了文献[9]中预测不支持环路的问题.

基于散列签名的 trace 预测方案设计思想如下,首先定义每个基本块的特征 ID,实现中这个特征 ID 为随机选取的唯一的 32 位二进制向量.硬件 trace 提取完成后,把 trace 中的所有基本块的特征 ID 提取出来,进行散列运算,最后得出一个数值,这个数值称为这条 trace 的期望签名(signature);软件执行时,每个执行了的基本块的 ID 都要进行同样的签名计算,保存到一个寄存器中,执行到控制流图子图的直接后必经节点(Immediate Post Dominators)处,签名寄存器中的值称为本次执行的实际签名.当程序执行的实际签名与某条硬件 trace 期望签名相等时,就认为最近的执行符合该 trace.预测器采用低开销的软件饱和计数器,当某一 trace 相关的路径连续执行次数达到设定的阈值时,下次执行到硬件 trace 的入口基本块处,即转入硬件执行.

程序路径与签名值是多对一的映射,存在散列碰撞的可能.这种方法可能会出现错误肯定(false positive)报告,也就是说,虽然实际执行的路径与硬件 trace 不同,但散列签名计算的结果却刚好与期望签名相等;但不会出现错误否定(false negative)报告,即,当签名不相等时两个路径决不会是同一个路径.实验表明这种方法具有很好的预测效果.

4 trace 协处理器产生方法

4.1 协处理器综合流程

图 3 是方法的工作流程,主要步骤如下:

①C/C++ 系统描述经编译器前端和基本的优化转变成可执行的控制数据流图,它是以本文第 3 节所设计的处理器无关的 IRC 表示的.

②通过 IRC 代码在仿真引擎上的 profile 执行,收集完整执行 trace,在系统设计约束条件指导下,使用热 trace 提取算法选择热 trace.

③根据系统的静态视图(控制数据流图)和动态特征(热 trace)构造扩展的控制数据流图,增加推测执行的节点和 trace 预测,并实现推测执行失败时的软件后备执行.

④扩展后的 IRC 控制数据流图经过 C 语言输出后端发射成协处理器功能单元的 C 语言描述和在处理器

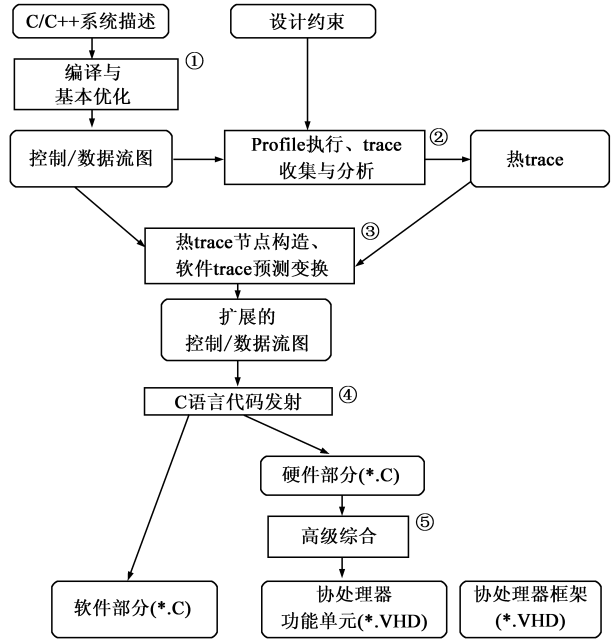

图3 协处理器自动产生流程

上执行的软件部分 C 语言代码.

⑤硬件部分的 C 语言描述经过高级综合变成专用协处理器功能单元的硬件 RTL 描述,这个步骤可以使用任何现有的支持标准 C 语言输入的高级综合工具.由于使用分支断言和推测执行机制,这里作为高级综合输入的 C 语言模块没有复杂的控制结构,相当于仅包含计算操作和数据相关的数据流图,这样的模块的高级综合比包含复杂控制逻辑的模块简单得多.

高级综合输出的协处理器模块的硬件描述和协处理器框架模板的硬件描述构成了可用于逻辑综合的专用协处理器.

4.2　协处理器模板与协处理器执行模型

图 4 是协处理器内部结构.图中虚线部分编号 1 到 n 的功能模块是自动构造的模块,其电路的 RTL 描述是软硬件划分后硬件部分的 C 语言文件经高级综合产生的.图中的其它部分是支持功能模块运行的固定设施,是人工设计实现的框架模板,可在多个不同应用中复用.其中图中上部分的模块负责根据处理器对协处理器执行请求 ID 选择相应的功能模块,并在输入操作数就绪时启动相应的模块.

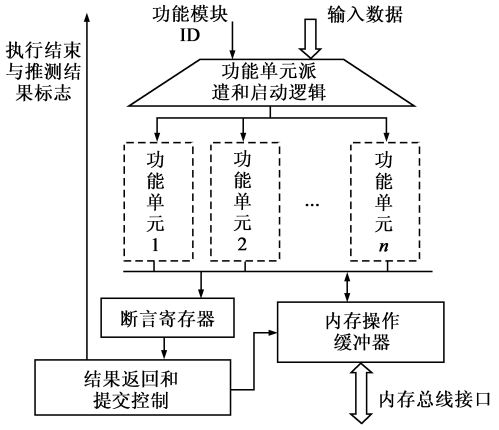

图4　协处理器结构示意图

图中的结果返回和提交控制模块根据断言寄存器的值向主处理器报告协处理器模块执行是否成功.图中最下部的推测结果写缓冲器,用于缓存功能单元执行中产生的内存写操作,如果断言寄存器表明推测执行成功,在提交控制模块的控制下将缓冲的内存写操作提交到内存,否则,所有的内存写被废弃.因此,当硬件功能模块猜测执行失败时,协处理器执行不会对内存中数据产生不当的影响.

5　实验结果

5.1　实验对象和实验方法

本文工作主要面向嵌入式应用,实验程序包括数字信号处理、控制、多媒体等典型的嵌入式应用.这些程序来自于 Powerstone benchmark、MediaBench[12] 以及实际工程项目.实验面向的目标 SoC 系统由一个 SPARC v8 处理器,共享存储器和一个包含多个功能模块的专用硬件加速协处理器组成.

实验基于 LLVM 实现了本文第 4 节图 3 的协处理器综合流程.基于 trace 构造的推测执行节点发射成标准 C,采用 SPARK 高级综合工具[13]获得对应的硬件描述语言模块;同时获得硬件实现时这些 trace 的等价逻辑门数和执行的周期数.

为了评估加速比,使用周期精确的 SPARC 处理器模拟器获得实验程序划分前软件实现和划分后的执行时间(周期数).

实验中,LLVM 为 2.3 版;C 和 C++ 编写的工具模块用 GCC v3.4.6 编译;高级综合工具为 spark-linux v1.2.用于评估加速比的 SPARC 二进制程序采用 sparc-linux-gcc 2.95.3 编译.实验环境是基于 Intel xeon 处理器的 Dell 工作站,操作系统为 Redhat Linux server 4.0.

5.2　软件 trace 预测效果评估

实验结果表明,基于散列签名的软件 trace 预测静态代码开销很小,其中程序 g3fax 静态代码增加最低,为 3.2%,最高的 v42 为 8.6%.

图 5 是基于散列签名的 trace 预测方法与文献[9]预测方法命中率的对比,从图中可看出基于 path profile 的预测方法的预测命中率则随程序特性不同变化很大;而本文的软件 trace 预测方案总体上有比较理想的命中率,平均 90.3%,比文献[9]提高了 11.3%,对环路密集的程序最多提高达 41.4%.因此,基于散列签名的 trace 预测方案,解决了包含环路的程序的路径记录和预测问题,这也是本文实验中总体加速比相对于文献[9]中实验结果提高的主要原因之一.

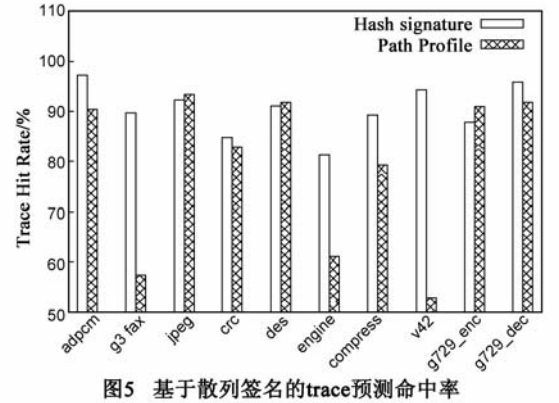

图5　基于散列签名的trace预测命中率

5.3　加速比与硬件面积评估

表 1 给出了应用本文方法自动提取协处理器硬件的实验结果,其中,trace 是程序中总共的 trace 数量,Hot-trace 是划分到硬件实现的热 trace 个数,约束条件为硬件面积不超过 2 万门;coverage 是这些热 trace 覆盖

的动态指令的百分比,最后一列是协处理器硬件实现的逻辑门数.从表中数据可以看到,虽然这些程序的总体 trace 数很多,但程序大多数时间在执行很少数量的热 trace.

表 1 实验结果

Program	Trace	Hot-trace	Coverage(%)	Area(Kgate)
adpcm	82	16	81.5	17
g3fax	567	29	79.6	19
jpeg	662	17	73.3	16.5
crc	29	6	97.0	11.5
des	63	12	94.1	13.5
engine	319	21	72.0	18
compress	258	14	88.4	16
v42	752	36	70.2	17
g729_enc	997	27	64.2	18.5
g729_dec	444	23	77.0	15.5

图 6 和图 7 是分别是同等面积约束下软硬件划分获得的加速比和最终硬件面积(等价逻辑门数,单位是千门).图中 IR Trace 是基于 IRC 的 trace 分析方法,SPARC Trace 是文献[9]的基于 SPARC 机器指令的 trace 分析;为了对比,图中同时给出了以基本块为划分对象粒度的模拟退火方法的划分结果[14].

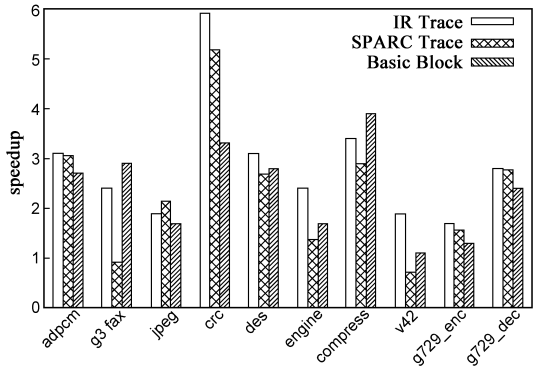

图6 基于trace推测执行的协处理器的加速比

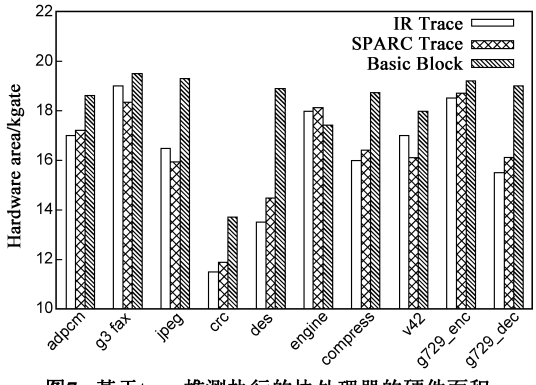

图7 基于trace推测执行的协处理器的硬件面积

实验发现,就硬件面积来说,基于中间表示的方法取得与基于机器指令的热 trace 提取相当的结果.软硬件划分后本文方法获得的平均加速效果比文献[9]方法高 22.6%,一方面,因为避免了"伪"热 trace 的干扰,更精确地捕捉到应用程序真正的性能关键部分;另一方面,更主要的原因在于 trace 预测对 trace 协处理器性能至关重要,基于散列签名的预测方法对各种类型的程序都有较好的预测精度.

采用模拟退火算法的细粒度划分最终使用的面积都很接近于约束上限;相比之下,基于 trace 的划分方法在同等成本约束条件下,最终结果面积变化很大,例如实验程序 crc,只使用了 11.5 千门,但获得了 5.9 倍的加速.深入分析实验数据发现,对于该程序,实际上 6 个很小的热 trace 就已经覆盖了支配整体运行时间的核心循环操作,剩余的 trace 则体积大,执行次数不多,在约束条件下不可能、也没有必要纳入协处理器硬件.des、g729_dec 等实验程序也呈现出了类似的特征.从结果可以看出,对于大多数实验程序,本文的方法比基于模拟退火的细粒度划分方法使用的硬件面积小,总体来说,程序获得的性能平均高 24.7%,最终结果硬件面积小 12.6%.

6 结束语

本文分析了程序表示层次对基于 trace 软硬件划分的影响,提出了体系结构无关的程序分析和热 trace 提取方法,实现了一种处理器无关的高效协处理器自动产生方法.本文的基于散列签名的软件 trace 预测方案,提高了 trace 预测命中率.实验表明,与原有的基于机器指令的 trace 提取方法相比,体系结构无关的 trace 分析结合基于散列签名的预测方法,获得的协处理器性能提高了 22.6%.

参考文献:

[1] M D Galanis, G Dimitroulakos, S Tragoudas, C E Goutis. Speedups in embedded systems with a high-performance coprocessor datapath [J]. ACM Transactions on Design Automation of Electronic Systems, 2007, 12(3): 1 – 22.

[2] 于苏东,刘雷波,尹首一,魏少军.嵌入式粗颗粒度可重构处理器的软硬件协同设计流程[J].电子学报,2009,37(5):1136 – 1140.
Yu Su-dong, et al. Hardware-software co-design flow for embedded coarse-grained reconfigurable processor [J]. Acta Electronica Sinica, 2009, 37(5): 1136 – 1140. (in Chinese)

[3] F Catthoor, H D Man, W Geurts, S Vernalde. Accelerator Data-Path Synthesis for High-Throughput Signal Processing Applications [M]. Boston: Kluwer Academic Publishers, 1997. 9 – 18.

[4] B Hounsell, R Taylor. Co-processor synthesis: A new methodology for embedded software acceleration [A]. Proc DATE '04 [C]. Paris: IEEE Computer Society Press, 2004. 682 – 683.

[5] 彭艺频,凌明,杨军,时龙兴.基于关键路径和面积预测的软硬件划分方法[J].电子学报,2005,33(2):249 – 253.

Peng Yi-pin, et al. Hardware/software partitioning method based on critical path and area prediction [J]. Acta Electronica Sinica, 2005, 33(2): 249 – 253. (in Chinese)

[6] J T Olson, J W Rozenblit, C Talarico, W Jacak. Hardware/software partitioning using bayesian belief networks [J]. IEEE Transactions On Systems, Man, And Cybernetics, 2007, 37(5): 655 – 668.

[7] G Stitt, F Vahid. Binary synthesis [J]. ACM Transactions on Design Automation of Electronic Systems, 2007, 12(3): 1 – 30.

[8] Wu Jigang, Thambipillai Srikanthan, Guangwei Zou. New model and algorithm for hardware/software partitioning [J]. Journal of Computer Science and Technology, 2008, 23(4): 644 – 651.

[9] 桑胜田,喻明艳,叶以正.基于程序执行轨迹的 SoC 软硬件划分方法[J].微电子学与计算机,2009,26(1):85 – 88.
Sang Sheng-tian, Yu Ming-yan, Ye Yi-zheng. A method to construct SoC hardware acceleration module with trace [J]. Microelectronics & Computer, 2009, 26(1): 85 – 88. (in Chinese)

[10] M Lopez-vallejo, J C Lopez. On the hardware-software partitioning problem: system modeling and partitioning techniques [J]. ACM Transactions on Design Automation of Electronic Systems, 2003, 8(3): 269 – 297.

[11] C Lattner, V Adve. LLVM: A compilation framework for lifelong program analysis & transformation [A]. Proc of the 2004 International Symposium on Code Generation and Optimization [C]. Chicago: IEEE Computer Society, 2004. 75 – 86.

[12] C Lee, M Potkonjak, W Mangione-Smith. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems [A]. Proc of the International Symposium on Microarchitecture'97 [C]. Washington, DC: IEEE Computer Society, 1997. 330 – 335.

[13] S Gupta, et al. SPARK: A high-level synthesis framework for applying parallelizing compiler transformations [A]. Proc of 16th Conferences on VLSI Design [C]. Los Alamitos, USA: IEEE Computer Society, 2003. 461 – 466.

[14] 吕雅帅,沈立,黄立波,王志英.面向嵌入式应用的指令集自动扩展[J],电子学报,2008,36(5):985 – 988.
Lü Ya-shuai, et al. Automatic instruction set extension for embedded applications [J]. Acta Electronica Sinica, 2008, 36(5): 985 – 988. (in Chinese)

作者简介:

桑胜田　男,1975 年 5 月出生于黑龙江.哈尔滨工业大学讲师,主要研究方向为 SoC 软硬件协同设计.
E-mail: stsang@hit.edu.cn

邱善勤　男,高级工程师.现主要从事软件与集成电路领域的研究工作,为重大专项、863 及多个部级项目的负责人.

李晓明(通信作者)　男,博士,2003 年 5 月获得哈尔滨工业大学微电子学与固体电子学专业工学博士学位.攻读博士期间的主要科研方向为嵌入式微处理器及 SoC 体系结构设计及相关技术研究.发表学术论文 10 余篇.目前主要从事 SoC 设计方法学、微处理器体系结构及微体系结构、cache 及总线低功耗技术等方面的研究工作.
E-mail: Lixiaoming@hit.edu.cn