计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
44(4):714~721, 2007

配置流驱动计算体系结构指导下的 ASIP 设计

李 勇 王志英 赵学秘 岳 虹
(国防科学技术大学计算机学院 长沙 410073)
(liyong.nudt@163.com)

Design of Application Specific Instruction-Set Processors Directed by Configuration Stream Driven Computing Architecture

Li Yong, Wang Zhiying, Zhao Xuemi, and Yue Hong
(*School of Computer Science, National University of Defense Technology, Changsha* 410073)

Abstract Efficiency and flexibility are crucial features of processors in embedded systems. The embedded processors need to be efficient in order to achieve real-time requirements with low power consumption for specific algorithms. And the flexibility allows design modifications in order to respond to different applications. In this paper, the configuration stream driven computing architecture (CSDCA) is proposed, which is both flexible and application specific hardware solution for implementation of embedded processors. Different from the traditional very long instruction word (VLIW) architecture or the transport triggered architecture (TTA), in the CSDCA, not only the responsibility of controlling the data transports is moved from the hardware to the compiler, but also the interconnect network between function units is visible to the compiler. So the routing can be performed by the compiler and the architecture can support the efficiency but complex interconnections to achieve low area overhead with low power dissipation. Directed by the CSDCA, an efficient design method for hardware implementation of application specific instruction-set (ASIP) processors is presented, which supports the reconfigurable segmented-bus networks. Experiment results with several practical applications show that the segmented-bus network can save 53% in power consumption and 38.7% in bus numbers, while maintaining the same speed compared with the simple-bus network.

Key words configuration stream driven computing architecture; transport triggered architecture; application specific instruction-set processor; embedded processor

摘 要 为了兼顾嵌入式处理器设计中的灵活性与高效性,提出配置流驱动计算体系结构.在体系结构设计中将软/硬件界面下移,使功能单元之间的互连网络对编译器可见,并由编译器来完成传输路由,从而支持复杂但更为高效的互连网络.在该体系结构指导下,提出一种支持段式可重构互连网络的专用指令集处理器(ASIP)设计方法.该方法应用到密码领域的3类ASIP设计中表明,与简单总线互连相比,在不影响性能的前提下,可平均节约53%的互连功耗和38.7%的总线数量,从而达到减少总线数量、降低互连功耗的目的.

关键词 配置流驱动计算体系结构;传输触发体系结构;专用指令集处理器;嵌入式处理器

中图法分类号 TP303

专用电路ASIC和通用信号处理器DSP是两种完成嵌入式信号处理的传统方法.ASIC方法功耗低、速度快,但是由于适用面窄,所以市场容量小使得成本较高.DSP方法的通用性限制了其开发特定领域内的并行性,速度偏低,功耗较大,随着市场划分越来越细,其竞争力优势正在逐渐丧失.专用

收稿日期:2006-04-26;修回日期:2006-12-22
基金项目:国家自然科学基金项目(60173040)

指令集处理器 ASIP 具有领域内的设计灵活性与高效性,有效克服了上述两种方法的缺点. 设计一个新的 ASIP 处理器,面临着针对特定应用程序的指令集生成、微结构设计和可重定向编译器开发[1]等问题.

Corporaal[2]提出的传输触发体系结构(TTA),将编译器视点下移至数据传输,可以有效解决上述问题[3]:TTA 只包括一种指令,避免了指令集生成的问题;在该结构的软件工具链中,语义翻译和调度相互独立,调度器无需关心语义,有效解决了可重定向编译的问题;微结构设计遵循统一模板,自动生成寄存器传输级描述. 但是,TTA 中功能单元之间通过简单总线来完成数据传输,使得总线成为主频提高的瓶颈和功耗消费的大户.

为了支持复杂但高效的互连网络,本文将软/硬件划分界面进一步下移,由编译器完成针对于复杂互连网络的传输路由,得到配置流驱动计算体系结构 CSDCA. 并在 CSDCA 指导下,提出一种支持可重构段式互连网络的 ASIP 设计方法.

1　配置流驱动计算体系结构

计算机程序的并行性开发包括两个方面:探测程序的并行性和在处理器资源限制下有效实现这种并行. 在单处理器上执行用高级语言编写的串行代码需要经过以下步骤[4]:

1) 前端编译. 完成词法分析、语法与语义分析、简单优化后生成基本操作.

2) 分析程序中基本操作间的数据依赖关系,生成数据相关图和控制相关图.

3) 绑定操作数到存储器中的位置. 在一定的时间周期内,内部寄存器被分配给某个变量,以减少与片外的数据交互,该过程也称为寄存器分配.

4) 绑定操作到时间槽. 指定操作完成的序列,该过程也称为指令调度.

5) 绑定操作至功能单元. 在有多个同一种功能单元时,必须对功能单元进行选择.

6) 绑定传输. 在完成功能单元绑定后,根据所需完成的操作,确定需要的数据传输.

7) 传输路由. 在传输网络上为多个并行传输找到最优路径,并生成配置.

8) 将配置加载到数据通路,完成数据处理功能.

上述步骤的顺序并非完全确定,例如寄存器分配和指令调度 2 个步骤,很多策略是寄存器分配在前,也有策略是指令调度在前. 显然,第一步需要编译器来完成,最后一步需要硬件来执行,为了达到最优化利用硬件资源或兼容性的目的,对于中间步骤,不同的体系结构采用不同的方式来完成.

超标量体系结构为了兼容性全部采用硬件来实现这些功能. 虽然能够不加修改地执行已有的二进制代码,但是由于硬件资源的限制,指令窗口太窄从而难以挖掘更多的并行.

数据流体系结构则是让编译器完成前端编译和数据相关性分析,后续的步骤则由硬件完成. 目标代码中的相关性被明确标识出来,也就是说,每个操作均指定使用其运算结果的后续操作. 这些结果通过令牌来传送,令牌中包括结果值和后继操作. 令牌必须与指令以及其他的令牌匹配,而这些匹配代价非常大.

超长指令字 VLIW 的编译器一直完成到绑定操作至功能单元,但是功能单元的源数据获取则由硬件来完成,源数据既可能来自于寄存器文件,也可能来自于定向回路. 随着功能单元的增加,寄存器文件将会成为瓶颈,而且也存在很多冗余传输.

传输触发体系结构 TTA 通过指定功能单元之间的数据传输来完成编程. 在前面几种体系结构中,指令均为明确指定,数据传输是副效应,而在 TTA 中,传输是基本的编程原语,指令则是副效应. 在硬件方面,TTA 具有译码简单、模块性好、灵活性等诸多优点;在软件方面,由于 TTA 编译器可以直接看到传输操作,可以引入多种独有的优化技术.

上述体系结构中,软件所见均为硬件的逻辑功能模型,而不涉及硬件的物理信息. 在通用计算机 GPP 领域,这样做的优势是软件开发人员从硬件中完全脱离出来,可以集中精力对现实世界中的问题进行建模. 但是在 ASIP 设计领域,这种界面却有待商榷:因为在 ASIP 设计领域,问题的复杂度相对较低,计算机系统只需要实现一些特定功能即可;而且 ASIP 在功耗和成本方面较 GPP 有更为严格的需求.

指令在硬件中,是通过对互连网络和功能单元上多个硬件编程点的配置,在特定时间(例如 1 个时钟周期或者 1 个算法时间)内形成特定的数据传送通道,进而形成计算引擎. TTA 结构下,应用软件在执行过程中指令流被翻译成传输流,而传输流可进一步被翻译成配置流.

定义 1. 将以配置流作为软硬件划分界面的体系结构,称之为配置流驱动计算体系结构 CSDCA.

图 1 显示了 CSDCA 以及上述体系结构的硬件

与编译器的责任折中. VLIW[5]和 TTA[3]是两种在 ASIP 设计中常用到的体系结构模板. 功能单元之间的互连网络是功耗消费的大户,占到整个芯片功耗耗费的 40%[6]. VLIW 和 TTA 的内部互连网络分别是交叉开关网络和多总线网络,这两种互连网络的路由简单,由硬件译码器来完成,然而这两种网络却因消费许多冗余功耗和占用较大面积而效率不高. 在 CSDCA 体系结构中,编译器可以看到完成功能单元之间连接的互连网络,并完成传输路由,从而可以支持效率更高的复杂互连网络,例如段式总线互连、异步互连、Mesh 互连等.

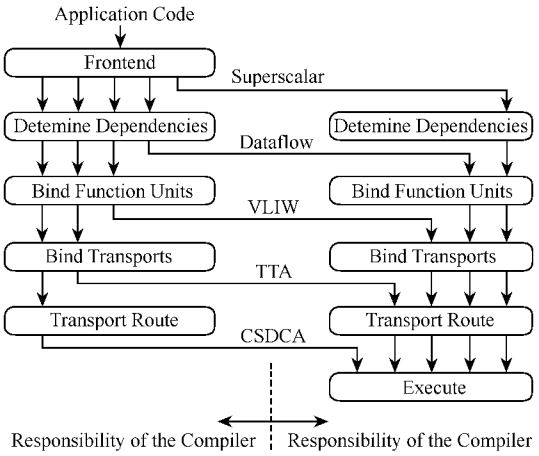

Fig. 1 The tradeoff between hardware and compiler for CSDCA.

图 1 CSDCA 的体系结构折中

2 段式总线互连的 ASIP 设计流程

在 CSDCA 指导下,本节描述一种支持段式总线互连的 ASIP 设计方法,包括 3 个步骤:首先,基于 TTA 生成简单总线互连的处理器,并基于该处理器和目标应用,得到传输流;然后,将简单总线在总线连接处分段,得到段式总线互连;最后,根据传输流需求,针对段式总线互连进行路由,得到配置流,同时减少总线数量以进一步优化互连网络.

2.1 生成传输流

本文采用基于 TTA 的 ASIP 设计流程生成传输流[7],如图 2 所示. 目标应用程序经过编译得到串行代码,串行代码输入到串行模拟器中,统计出应用程序动态运行的信息. 根据这些统计信息,资源选择器在设计者的指导下生成初始微结构机器描述. 调度器依据微结构和串行模拟器得出的统计信息对串行代码进行调度,生成并行代码. 并行模拟器则对并行代码进行模拟,得出应用程序的实际运行周期和资源的利用率. 代价估算器依据上述信息、微结构描述以及代价库,计算出 ASIP 处理器的面积、实际运行功耗和性能. 当应用程序在 ASIP 处理器上的运算周期不能满足设计者时,可以增加功能单元和总线,直到能够满足需要或者处理周期不再减少为止.

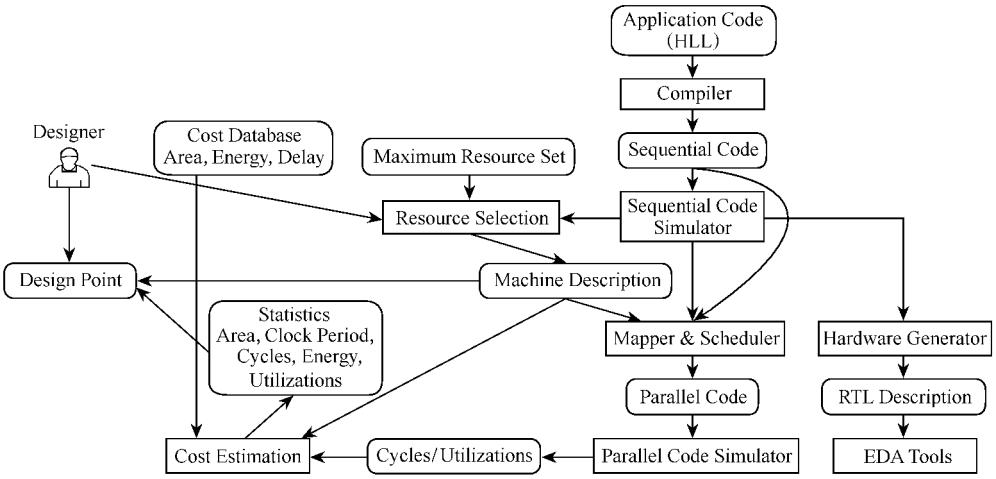

Fig. 2 The automatic design flow based on TTA.

图 2 基于 TTA 的自动生成流程

经过上述步骤后,硬件方面可得到基于简单总线的 ASIP 处理器. 软件部分则得到相应的传输流,即确定了每个周期需要完成的传输操作. 然而,简单总线互连网络是处理器的性能瓶颈和功耗消费大户.

2.2 段式总线互连网络

段式总线互连常常被用来降低总线互连的功耗[8-9]和提高其并行性[10-11]. 段式总线是指将普通总线分成多个段,通过对相邻段之间的连接点进行

配置,可以组成多条总线.段式总线较普通总线有两点优势:首先,特定传输的总线功耗降低,因为段式总线的负载仅包括传输所必须经过的路径;其次,简单总线在经过分段后,可组成多条总线,在并行度确定的前提下,可减少总线数量.

用段式总线代替已得到的 ASIP 处理器中的简单总线,如图 3 所示.图 3(a)显示了简单总线互连网络,功能单元的端口与总线有两种可能配置:断开或者连接,当前支持 4 路并行传输,需要 4 组总线.将这些简单总线在端口与总线的连接处用总线连接器(BC)断开,得到段式总线,如图 3(b)所示.BC 有 5 种可能的配置:端口与左边总线连接(简称左连)、端口与右边总线连接(简称右连)、端口与总线全连接(简称全连)、总线传输和断开.因为段式总线可支持并行,上述 4 路传输仅需两组总线.为了支持这个 4 路传输,简单总线网络的全部均为活跃的,而段式总线中只有必须的部分才是活跃的(活跃段为实线,非活跃段为虚线).段式总线的开销主要来自于 BC,在实验部分,这些开销将被考虑在内.

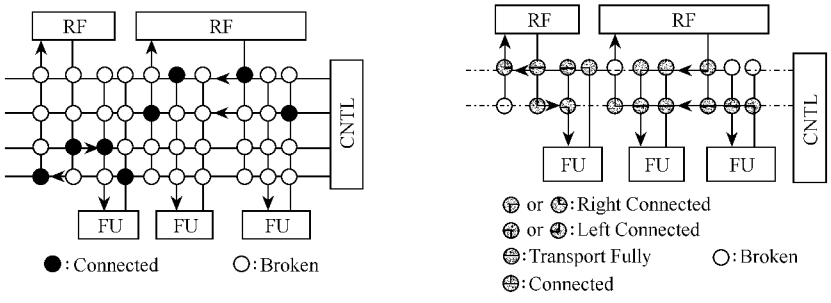

Fig. 3　Interconnect network. (a) Simple-bus interconnect network and (b) Segmented-bus interconnect network.

图 3　互连网络.(a)简单总线互连网络;(b)段式总线互连网络

2.3　路由与优化

在第 2.1 节中得到了传输流,即指定了每个周期需要完成的传输操作.在本小节中,我们的目标除了为传输指定路径外,还要尽量减少所需的总线组数,优化互连网络.首先进行基础路由,即为每一个传输分配一组段式总线;然后进行合并,以减少实际使用的总线数.

将总线端口和总线连接器按照从左到右的方向编号,它们的编号一致.基础路由为每路传输指定一组总线,并进行路由.合并包括同源 Socket 合并与最终合并:如果多路传输的源端口相同则显然应该合并;而最终合并是指,如果两组总线所支持的传输,需要经过的线段号没有交集,则可以合并.具体过程如算法 1 所示.

算法 1. 路由与优化.

数据结构:Typedef unsigned short sock;

struct MOVE {*sock source*; *sock dest*;}

enum CONFIG {左连,右连,传输,全连,断开}.

输入:MOVE *Trans*[0..*N* − 1]; sock *placement*[0..*S* − 1];

输出:CONFIG *buses_config*[0..*N* − 1][0..*S* − 1]; bool *buses_used*[0..*N* − 1]

1) for (*i* = 0; *i* < *N*; *i* ++) /* 基础路由 */

1.1 min = *MIN* {*placement*[*trans*[*i*].*source*], *placement*[*trans*[*i*].*dest*]};

1.2 max = *MAX* {*placement*[*trans*[*i*].*source*], *placement*[*trans*[*i*].*dest*]};

1.3 for(*j* = 0; *j* < min; *j* ++) *buses_config*[*i*][*j*] = 断开;

1.4 *buses_config*[*i*][*j* ++] = 右连;

1.5 for(; *j* < max; *j* ++) *buses_config*[*i*][*j*] = 传输;

1.6 *buses_config*[*i*][*j* ++] = 右连;

1.7 for(; *j* < *S*; *j* ++) *buses_config*[*i*][*j*] = 断开;

1.8 *buses_used*[*i*] = TRUE;

2) for(*i* = 0; *i* < *N*; *i* ++) {/* 同源合并 */

2.1 for(*j* = *i* + 1; *j* < *N*; *j* ++)

2.1.1 if (*trans*[*i*].*sock* == *trans*[*j*].*sock*) {

2.1.1.1 *buses_used*[*i*] = FALSE;

2.1.1.2 合并(*i*,*j*);

2.1.1.3 break;

3) for(*i* = 0; *i* < *N*; *i* ++) /* 最终合并 */

3.1 if (*buses_used*[*i*] == FALSE) continue;
3.2 *max_length* = 0; max = 0;
3.3 for(*j* = *i* + 1; *j* < *N*; *j* ++)
3.3.1 if (*buses_used*[*j*] && 碰撞检测(*i*, *j*)) *length* = 尝试合并(*i*, *j*);
3.3.2 if (*length* > *max_length*)
3.3.2.1 *max_length* = *length*;
3.3.2.2 max = *j*;
3.4 if (max == 0) continue;
3.5 合并(*i*, max);
3.6 *buses_used*[*i*] = FALSE.

算法 1 的输入是单个指令字内并行执行的 N 个传输和端口连接编号,输出是 N 组段式总线的使用情况和所用总线的相应配置.

在基础路由阶段,为第 i 路传输分配第 i 组总线. 查询传输对应的物理端口号,并记住小端口号 min 和大端口号 max. 将大于 min 和小于 max 的 BC 指定为传输,小于 min 或大于 max 的 BC 指定为断开,min 处的 BC 向右连接,max 处的 BC 向左连接. 这样就为每路传输在一组段式总线上形成了一条传送通道.

在同源合并阶段,判断第 i 路传输与其后面的传输是否有相同的源端口,如果相同则进行"合并",并删除第 i 组段式总线. 在最终合并阶段,判定两组段式总线能否合并的条件是它们的传输路径是否会碰撞(碰撞即经过相同的段),不碰撞则进行合并. 由于一组总线可能可以与后续多组总线合并,那么与谁合并选择标准就是合并后活跃的总线长度最长.合并过程中,"合并"运算的规则如图 4 所示:

	L	*R*	*F*	*T*	*B*
L	*L*	*F*	*F*	*F*	*L*
R		*R*	*F*	*F*	*R*
F			*F*	*F*	*F*
T				*T*	*T*
B					*B*

L: Left connected; R: Right connected; F: Fully connected; T: Transport; B: Broken

Fig. 4 Operation rule for merging.

图 4 合并运算规则

通过统计段式总线使用情况,可得到当前指令字需要的总线组数. 对应用程序中的所有指令字执行该算法后,需要的最大总线组数即为互连网络的总线组数,得到优化后的段式总线互连网络. 经过路由后,传输流被翻译成配置流,得到固件.

3 实验结果

本节中,首先给出段式总线的功耗和延迟模型,然后基于周期精确的模拟器统计具体应用中互连网络耗费的能量. 将段式总线互连和简单总线互连在总线数量、实际运行功耗和延迟等方面进行比较.

$$E = (C_{\text{wire}} + C_{\text{BC}})V^2 = K_{\text{L}} \cdot L + K_{\text{BC}} \cdot N_{\text{BC}}, \quad (1)$$

$$T = r(d)C_{\text{total}} + \sum_{i=1} \{r(W_i)[0.5C(W_i) + C(i)] + r(BC)C(i)\}. \quad (2)$$

3.1 功耗与延迟模型

通用的互连网络功耗模型如式(1)所示. 式(1)中 K_{L} 是单位连线消耗的能量,K_{BC} 是连接器消耗的能量. L 和 N_{BC} 分别代表活跃的传输线的长度和总线连接器的个数.

根据 Elmore 模型得到互连网络的延迟模型如式(2)所示,图 5 为延迟模型,其中 $r(d)$ 为源驱动电阻,$r(W_i)$ 为第 i 段连线的电阻,$C(W_i)$ 为第 i 段连线的电容,$r(BC)$ 为连接器电阻,$C(i)$ 为连接器电容.

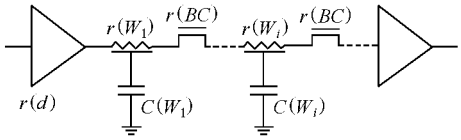

Fig. 5 Delay model of segmented bus.

图 5 段式总线延迟模型

根据上面叙述的分析模型,理论上可以直接根据式(2)计算出 ASIP 处理器的互连网络延迟,而运行功耗的估计除了需要式(1),还要依赖于模拟器的统计结果.

3.2 评估结果

目标结构中宏模块的面积和延迟信息均可通过综合得到,采用 0.18μm CMOS 1P6M 工艺. 连线的电阻与电容参数是 6 层金属连线参数的平均值. 本文从密码领域选出 3 类不同的应用来进行评估. 第 1 类应用是安全 Hash,包括 MD5,SHA-1,SHA-256 等 3 种算法,每一种算法均对 1Mb 数据生成摘要;第 2 类应用是对称密码,包括 DES,3DES,AES,RC6 等 4 种算法,每一种算法加密 512Mb 的数据;第 3 类应用是非对称密码,包括 RSA 一种算法. 前两类应用涉及的主要运算是位操作,而乘法则是第 3 类应用的主要运算. 对于此 3 类应用,利用第 2 节所述的方法得到相应的 ASIP 处理器,如表 1 所示. 从表 1 中可以看出,在引入 CSDCA 概念,支持段式总线后,总线数量平均减少 38.7%.

Table 1　Three ASIP Processors

表 1　三类 ASIP 处理器

Application	Macro Block					Number of Buses (simple bus)	Number of Buses (segmented bus)
	Access memory cell	Register files	Bit operation cell	Multiplier	Adder		
Secure Hash	2	16×3	3	0	2	10	6
Symmetric cryptogram	3	32+16×1	2	0	1	9	6
Public key cryptogram	1	16×5	1	3	1	12	7

模拟器通过统计程序运行过程中活跃的总线长度和所有的 BC 翻转次数,根据式(1)计算消耗在总线上的功耗,结果如图 6(a)所示. 互连总线的延迟估计可以根据式(2)计算得到,其结果如图 6(b)所示.

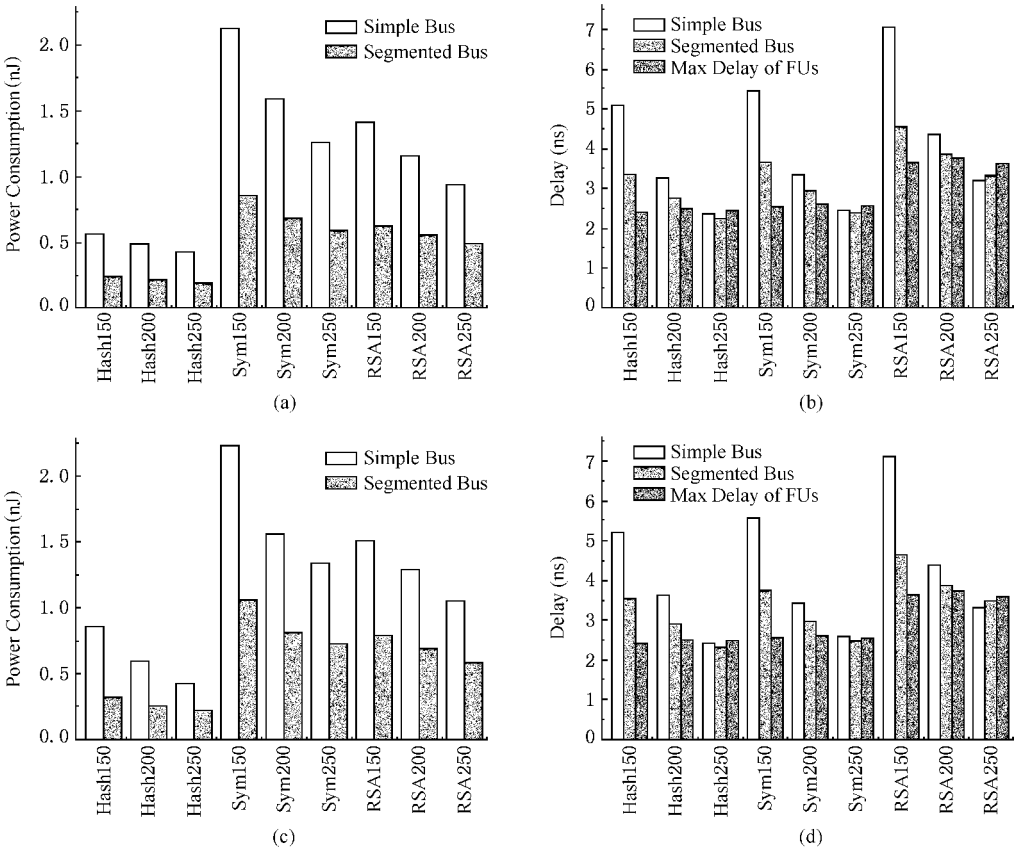

Fig. 6　Results of statistics. (a) Power consumption comes from the simulator; (b) Delay comes from simulator; (c) Power consumption comes from EDA tools; and (d) Delay comes from EDA tools.

图 6　统计结果.(a) 模拟器统计的功耗结果;(b) 模拟器统计的延迟结果;(c) EDA 工具统计的功耗结果;(d) EDA 工具统计的延迟结果

从模拟器估计结果来看,安全 Hash 和对称密码两类应用的功耗为所包含算法的平均值. 能量的绝对耗费值取决于应用、宏模块高度及总线类型这 3 个因素. 对于相同的应用和宏模块高度 h,段式总线比简单总线所消耗的能量要少得多. 随着宏模块高度的增加,绝对能量减少,因为无论对于那种总线,其相应的连线长度都在减小. 段式总线减小的速度相对较小,这是因为总线连接器的能量耗费为定值.

同时,随着宏模块高度 h 的增加,简单总线互连的延迟以 $O(1/h^2)$ 的速度递减,而段式总线网络的延迟减少要缓慢得多,因为总线连接器所引入的延迟为常数. 对于公钥密码应用,当 $h=250\mu m$ 时,简单总线的延迟小于段式总线,但是此时的关键路径在乘法器上. 其他的情况下,段式总线连接的延迟均小于简单总线. 因此,引入段式总线没有影响处理器性能.

为了验证模拟器估计结果,本文对所实现的 ASIP 电路进行了仿真分析. 生成 ASIP 版图后,对其进行版图寄生参数提取(layout parasitic extraction, LPE),可以得到反标有寄生参数的 Spice 网表,分别

输入测试向量后可以通过 EDA 工具对其进行功耗及时序分析,仿真所用电源电压为 1.8V,温度为 25℃,器件参数采用厂家提供的典型值,统计所得的功耗及延迟信息分别如图 6(c)与图 6(d)所示.

可以看出,采用支持段式可重构互连网络的 ASIP 设计方法,能够有效地减少总线数量,降低互连功耗.

4 结论与进一步工作

本文将编译器视点进一步下移,得到配置流驱动计算体系结构,编译器能够看到片内功能单元之间的互连网络并完成路由,该体系结构指导下的 ASIP 能够支持复杂但高效的互连网络. 采用段式总线互连技术,本文提出一个完整的高效 ASIP 设计流程. 将该方法应用到密码领域的 3 类 ASIP 设计中表明,与已有的设计方法相比,在不影响性能的前提下,可平均节约 53% 的互连网络功耗和 38.7% 的总线数量.

由于指令需要配置来直接控制内部传输操作,指令宽度过长,程序代码中信息冗余严重,代码密度较低,需要研究指令压缩技术.

致谢 感谢芬兰 TUT 大学的 Takala 和 Cilio 对本研究工作的进展提供大量建议!

参 考 文 献

[1] K Keutzer, S Malik, A R Newton. From ASIC to ASIP: The next design discontinuity [C]. IEEE Int'l Conf on Computer Design, Freiburg, 2002

[2] H Corporaal. Microprocessor Architectures: from VLIW to TTA [M]. Chichester, West Sussex, England: John Wiley & Sons Ltd, 1998

[3] Yue Hong, Shen Li, Dai Kui, *et al*. A TTA-based ASIP design methodology for embedded systems [J]. Journal of Computer Research and Development, 2006, 43(4): 752-758 (in Chinese)
(岳虹, 沈立, 戴葵, 等. 基于 TTA 的嵌入式 ASIP 设计[J]. 计算机研究与发展, 2006, 43(4): 752-758)

[4] B R Rau, J A Fisher. Instruction-level parallel processing: History, overview and perspective [J]. Journal of Supercomputing, 1993, 7(1): 9-50

[5] B Middha, V Raj, A Gangware, *et al*. A Trimaran based framework for exploring the design space of VLIW ASIPs with coarse grain functional units [C]. The 15th Int'l Symp on System Synthesis, Kyoto, 2002

[6] D Liu, C Svensson. Power consumption estimation in CMOS VLSI chips [J]. IEEE Journals of Solid-State Circuits, 1994, 29(6): 663-670

[7] Zhao Xuemi, Wang Zhiying, Yue Hong, *et al*. Automatic generation of applications specific instruction-set processer directed by TTA [J]. Journal of Computer-Aided Design & Computer Graphics, 2006, 18(10): 1491-1496 (in Chinese)
(赵学秘, 王志英, 岳虹, 等. TTA 指导下的 ASIP 自动生成[J]. 计算机辅助设计与图形学学报, 2006, 18(10): 1491-1496)

[8] Wang Zuodong, Wei Shaojun. Research and progress of low power design in SOC era [J]. Microelectronics, 2005, 35(2): 174-179 (in Chinese)
(王祚栋, 魏少军. SOC 时代低功耗设计的研究与进展[J]. 微电子学, 2005, 35(2): 174-179)

[9] W B Jone, J S Wang, H Lu, *et al*. Design theory and implementation for low-power segmented bus systems [J]. ACM Trans on Design Automation of Electronic Systems, 2003, 8(1): 38-54

[10] C Katsinis. A segmented-shared-bus multicomputer architecture [C]. The 9th Int'l Conf on Parallel and Distributed Computing and Systems (PDCS'97), Washington, 1997

[11] C H Yeh, B Parhami. Design of high-performance massively parallel architectures under pin limitations and non-uniform propagation delay [C]. The 2nd AIZU Int'l Symp on Parallel Algorithms/Architecture Synthesis (PAS'97), Aizu-Wakamatsu, 1997

Li Yong, born in 1978. Ph. D. candidate in computer science. His main research interests include computer architecture, asynchronous circuits design technology and VLSI design.
李 勇, 1978 年生, 博士研究生, 主要研究方向为计算机体系结构、异步电路设计及 VLSI 设计.

Wang Zhiying, born in 1956. Professor and Ph. D. supervisor of the National University of Defense Technology. His main research interests include computer architecture, microprocessor design and asynchronous circuits design technology.
王志英, 1956 年生, 教授, 博士生导师, 主要研究方向为先进计算机体系结构、微处理器设计技术研究、异步电路设计技术等.

Zhao Xuemi, born in 1979. Ph. D. His main research interests include design of efficient architectures for DSP.
赵学秘, 1979 年生, 博士, 主要研究方向为面向数字信号处理算法设计、实现高效体系结构.

Yue Hong, born in 1980. Ph. D. Her current research interests include computer architecture, high performance microprocessor design and VLSI design.

岳　虹,1980 年生,博士,主要研究方向为计算机体系结构、高性能微处理器设计等.

Research Background

We propose a design method for application specific instruction-set processor (ASIP) directed by transport triggered architecture (TTA). In TTA, software specifies data transports between function units (FUs), so application specific hardware can support more sophisticated FUs. Problems about instruction generation and retargetable compiling can be solved at the same time. Configuration stream driven computing architecture (CSDCA) is proposed, where routing is performed by the compiler to support efficient but complex interconnections. Experiment results with several practical applications show that segmented bus network save 53% in power consumption and 38.7% in bus numbers, while maintaining the same speed compared with the simple-bus network. Our work is supported by the National Natural Science Foundation of China (60173040).

2007 年全国软件与应用学术会议(NASAC '07)征文通知

中国西安

2007 年 9 月 20 日至 22 日

全国软件与应用学术会议(NASAC)由中国计算机学会系统软件专业委员会和软件工程专业委员会联合主办,是中国计算机软件领域一项重要的学术交流活动. 第 6 届全国软件与应用学术会议 NASAC2007 将由西安交通大学计算机系承办,于 2007 年 9 月 20 日至 22 日在陕西西安举行. 此次会议将由国内核心刊物(计算机科学)以增刊形式出版会议论文集,还将选择部分优秀论文推荐到核心学术刊物(EI 检索源)发表,并将评选优秀学生论文. 欢迎踊跃投稿.

征文范围(但不限于下列内容)

- 需求工程
- 构件技术与软件复用
- 面向对象与软件 Agent
- 软件体系结构与设计模式
- 软件开发方法及自动化
- 软件过程管理与改进
- 软件质量、测试与验证
- 软件再工程
- 软件工具与环境
- 软件理论与形式化方法
- 操作系统
- 软件中间件与应用集成
- 分布式系统及应用
- 软件语言与编译
- 软件标准与规范
- 软件技术教育
- 计算机应用软件

论文要求

① 论文必须未在杂志和会议上发表和录用过.

② 论文篇幅限定 6 页(A4 纸)内.

③ 会议只接受电子文档 PDF 或 PS 格式提交论文. 排版格式请访问会议网址(http://nasac07.xjtu.edu.cn)

④ 投稿方式:采用"NASAC2007 在线投稿系统"(http://nasac07.xjtu.edu.cn)投稿(待建).

重要日期

论文投稿截止日期:2007 年 5 月 31 日

论文录用通知日期:2007 年 6 月 30 日

学术会议及活动日期:2007 年 9 月 20 日至 22 日

联系方式

联系人:王换招、张华,西安交通大学计算机科学与技术系

Tel:029-82668971　　E-mail:csed@mail.xjtu.edu.cn

更详细的内容请访问 NASAC 2007 网址:http://nasac07.xjtu.edu.cn