综述:面向 SoC-FPGA 的开源处理器

余 乐[1],李任伟[2],王 瑶[1],李洋洋[1],吴 超[1],贾 瑞[2]
(1. 北京工商大学计算机与信息工程学院食品安全大数据技术北京市重点实验室,北京 100048; 2. 中国科学院自动化研究所,北京 100190)

摘 要: 近年来,随着各种 IP 核的广泛应用,SoC-FPGA 的应用领域也随之日益扩展. 处理器作为 SoC-FPGA 的核心 IP,其对系统性能的影响至关重要. 使用开源处理器 IP 能大幅度提高 SoC-FPGA 系统级设计的效率,已成为现在项目开发中常用的手段. 本文研究了现有的绝大多数开源处理器的关键技术指标,从可用性和稳定性上提出了一种选择开源处理器的方法. 根据该方法,选择出一些具有高可用性和稳定性的开源处理器. 最后,利用不同厂商提供的 FPGA EDA 工具将所述的开源处理器进行了综合与实现,并与现有 FPGA 厂商提供的商用软核 Nios II 和 Microblaze 进行了比较和讨论.

关键词: 处理器; FPGA; SoC-FPGA; 开源; 应用; 综述; 概述
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)04-0992-13
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.04.030

Open Source Processors for SoC-FPGA: A Survey

YU Le[1], LI Ren-wei[2], WANG Yao[1], LI Yang-yang[1], WU Chao[1], JIA Rui[2]
(1. *Beijing Key Laboratory of Big Data Technology for Food Safety, School of Computer and Information Engineering, Beijing Technology and Business University, Beijing* 100048, *China*; 2. *Institute of Automation, Chinese Academy of Sciences, Beijing* 100190, *China*)

Abstract: With the increasing deployments in IP Cores in FPGA applications, SoC-FPGA systems have been gaining wide popularity in recent years. Processor is no doubt the heart of a SoC-FPGA system, and has a critical impact on overall system performance. The utilization of open source processor IPs can greatly improve working efficiency and reduce the development cycle of SoC-FPGA systems, and it has been gaining favor as a commonly adopted design methodology. This paper investigates the key technical indicators of existing open-core projects, and provides an overview of open source processors. The major features of open source processors are summarized in terms of usability and stability, and the authors propose a methodology to choose appropriate processors for specific applications. Following these guidelines, some open source processors are selected and implemented on Stratix V and Virtex-7 FPGA platforms using corresponding EDA tools. The implementation results are compared and discussed.

Key words: processors; FPGA; SoC-FPGA; open source; applications; survey; overview

1 引言

FPGA 具有许多的设计优点,如降低设计成本、复杂度等. 相比于 ASIC[1],FPGA 上市时间更短. FPGA 传统的应用领域[2],包括工业控制[3~6],通信[7,8],医疗[9~12]和图像处理[13,14],以及 FPGA 加速器件[15~18]的开发. 文献[15~18]中提到,虽然由 FPGA 综合出的处理器在技术上和商业上都取得了一定的成功,但是近年来,包含愈来愈多 IP,正逐步演变成 SoC-FPGA 系统[19,20]在未来重要的发展趋势. 基于 FPGA 的 SoC 快速发展也使 FPGA 的应用领域扩展到一些新的领域,比如云计算[21~23],移动市场和数据中心[24,25]. 相应地,基于 SoC-FPGA 系统的设计与实现方法也变得越来越重要. 最吸引人的 SoC-FPGA 设计方法是选择可复用器件和 IP 来整合系统[26~28]. 处理器作为 SOC-FPGA 中最重要的组成部分,若能通过复用方法进行设计、验证、测试及共享,将显著增加 SOC-FPGA 的设计生产率[29].

开源硬件作为一种新型的共享 IP 设计模式,正在成为提高设计生产率[30]的一种有效的方法. 可用的开源硬件是一类可用的资源[31]. 开放公开可用的源代码非常利于学习和创新. 如果处理器以一个开放源的形式共享,它可以被不同的设计小组获取、优化,进而设计

收稿日期:2016-10-09;修回日期:2017-06-13;责任编辑:孙瑶
基金项目:北京市自然科学基金(No. 4174086)

出不同的嵌入式处理器.更重要的是,如果我们从中挑出设计出色的处理器,可以大大加快开发过程.显然,这对于基于嵌入式处理器的整个 SoC-FPGA 系统的性能和可靠性的保证起到了非常重要的作用.

软核处理器的架构可以通过 HDL 进行行为级抽象[32].对于设计人员来说,在基于 FPGA 的 SoC 系统上实现软核处理器是非常方便的.使用软核有以下优点:首先,能够通过使用单一的 FPGA 来减少系统的多个物理组件;第二,软核的"能见度"是非常高的,系统设计人员不仅可以了解处理器架构,同时也可以为不同应用领域[33~38]硬件的实现深度定制;第三,由于其卓越的灵活性,软核能够在任何的目标 FPGA 器件和 ASIC 技术[39~41]进行实现.

虽然软核天然具有经济性,成为许多设计者和工程师的选择.然而,现在市面上众多的开源处理器使得开发人员的选择过程变得十分困难.除此之外,在选择开源处理器时,也需要考虑其可靠性、可视性、实用性以及稳定性的特点.针对不同的 SoC 系统,软核的选择应该从 ISA、授权许可以及工具链的开源处理器多方面考虑.本综述回溯了 178 个开源处理器并给出了如何选择合适的开源处理器的一套原则.本文同时也将现有 FPGA 厂商提供的软核和开源处理器通过在 Stratix V 和 Virtex-7 FPGAs 进行实现,并对试验结果做了比较和分析.

本综述的主要贡献是:

(1)总结和分类开源社区中的开源处理器.

(2)提出一种如何从开源处理器中选择具有较高稳定性和可用性的开源处理器的方法.

(3)实现并比较了数个备选开源处理器和现有的 FPGA 厂商提供的软处理器.

2　研究现状

2.1　开源硬件的产生

1998 年,荷兰代尔夫特理工大学发起 Open Design Circuits Group[42]组织,目的是在网上开放电路设计,便是开源硬件的雏形.这启蒙了后来著名的两个开源硬件网站:Opencores 和 OpenIPcore.此后,越来越多的研究者和工程师们选择把他们的设计公布到这两个开源网站上.2000 年,OpenIPcore 合并入 Opencores[43].

Jamil Khatib 撰写了许多最初的 Opencores 奠基性文档[44],并定义了开源硬件及其商业模式,也参与了最早的 F-CPU、gEDA 等项目的开发.2006 年,他甚至建议除了 http 之外,在互联网上还需要添加 Hardware Computing Resource Protocol,比如说"hcrp://",专门用来进行硬件资源下载.

随着设计规模越来越大,芯片设计越来越向 SoC 方向迈进,以达到快速设计出原型系统的目的.这时,IP 核的可重用性和易用性就成了最关键的问题,而开源 IP 核无疑在这方面具有先天优势.因此,很多商业公司也逐渐加入了开源硬件的行列,例如,SUN 和 IBM 公司[45].

2.2　开源处理器的发展

开源处理器中知名的项目[46,47]有:(1)F-CPU,是第一个在互连网上设计的处理器;(2)OpenRISC,OpenCores 中的项目,试图开发一个完全免费的 RISC 构架的处理器以及之上的 SOC;(3)Leon2,欧洲航天局(ESA)开发兼容 Sparc 的处理器.

与开源软件的全免费不同,开放源码硬件最终要物理实现才能验证其设计是否达到预期.FPGA 以其快速灵活,初期投入成本低廉,成为开源硬件最适合的开发平台.这也为 SoC-FPGA 直接应用 Opencores 中的开源处理器 IP 提供了便利.

为了让开源硬件社区的项目易于共同开发和沟通,通常会提供统一的开发板.例如,Opencores 社区广泛使用的 Micro FPGA Board 和 OCRP-1 board 这两款开发板[48,49],就是采用 Xilinx 公司的 Vertix 系列 FPGA 设计的.大名鼎鼎的 OpenRisc1000 便是在这块板上进行了原型验证.目前,Leon2 和 Openrisc 1200 均可以成功运行 Linux 系统,这些都为开源硬件和软件的社区融合打下了基础.

2.3　中国开源处理器的发展状况

国内有案可查的开源处理器项目是 2001 年 3 月启动的 OpenARM.然而,OpenARM 并不像它的名字所隐含的那样与设计 ARM 处理器有关.实际上,该项目仅利用了现成的 ARM 芯片,外加了 MEMEC 公司(现已被 Avenet 收购)于 2002 年 3 月捐赠的一块 Xilinx FPGA.其中,FPGA 是作为 ARM 的外围扩展模块.FPGA 的引入使该设计可以很容易地用 HDL 代码的形式进行管理和跟踪.该项目后期还充分引入了 OpenCores 社区已有的成果,如 Wishbone 总线[50].

此后,随着国家对"中国芯"的大力支持,中国科学院、国防科技大学、北京君正、北大众志、苏州国芯、杭州中天、总参 56 所等研究单位逐渐崭露头角.在他们的不断努力下,国内也诞生了龙芯系列[51]、FT 系列[52]、X Burst 系列[53]、Uni-Core 系列[54]、C * Core[55]系列、CK-CPU 系列[56]、申威[57]等一批自主研发的优秀 CPU 产品.虽然其中某些处理器项目参考了开源处理器的部分源码,但更多内核设计属于自主开发、指令兼容、逆向设计、技术授权或技术引进等.

3　开源处理器的关键技术指标

近年来开源软核处理器数目迅速增加,开源软核处理器在嵌入式领域也获得了广泛的知名度.FPGA 厂商和开源社区提供了大量的开源软件处理器,这些处

理器各有各的特点. 这样一来,嵌入式的开发者从众多的处理器中选择一个合适的处理器是非常困难的. 本文从实用性和稳定性的角度,对现有开源软核处理器进行概述.

本文共讨论了 178 个处理器,他们来自 OpenCores[58] 上的处理器项目、软核处理器和 Wikipedia[59] 上的开源处理器. 来自 OpenCores 的处理器的源代码是可以从互联网上直接下载,并直接应用的. 在这 178 个处理器中,有四个处理器没有源代码,一个是针对 CPLD[60],两个是用 C/C + + 写的,两个被设计成接口转换器. 这九个处理器不在本文讨论的基于 FPGA 的开源软核处理器范围. 因此,需要进一步考虑的开源软核处理器的总数下降到 169 个.

开源处理器的开发周期持续时间较长. 一般情况下,一个开源项目的开发周期被分为 5 阶段,包括规划,Pre-α,α,β,稳定期和成熟期[61]. 本文所研究的开源处理器所处的开发周期概括在图 1 中.

不同开发周期的开源处理器的稳定性也不尽相同. 在稳定期,处理器功能完善,基本没有 bug. 因此只有当项目经历了稳定期,处理器才可信赖. 根据调查,仅有 68 个处在稳定期的开源处理器能够被方便的使用.

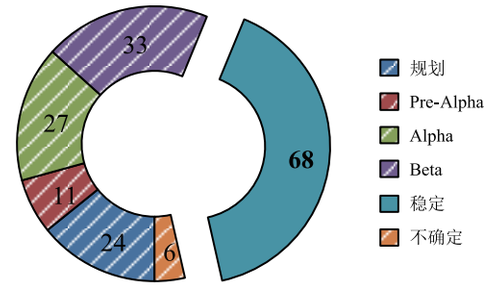
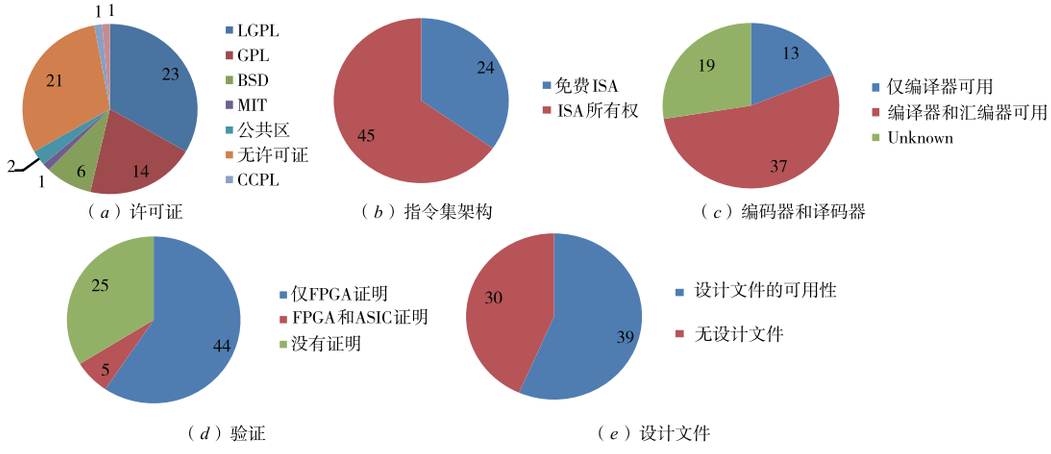


图1 所有开源软核处理器的开发周期

虽然这 68 个处理器经历了稳定期,许多其他的因素也对处理器的实用性和稳定性有影响. 针对本次调研选择的这 68 个稳定的处理器,本章的余下部分从授权许可、指令集架构(ISA)、编译器与汇编器、验证和设计文档这几个方面给出了详细的描述. 如图 2 所示.

3.1 授权许可

所有的软件在默认情况下都受版权法保护. 许可证是该软件的所有者给人们复制和修改软件[59] 的权限. 因此,处理器许可证是在选择一个开源的软核处理器时的最重要的因素之一,因为它决定哪些用户可以或不可以使用处理器. 大多数开源处理器的授权许可都参照开源软件[62] 移植的授权许可. 处理器的授权许可显示在图 2(a).



(a) 许可证 (b) 指令集架构 (c) 编码器和译码器

(d) 验证 (e) 设计文件

图2 开源处理器的详细分类

处理器授权许可证,涵盖 GNU 通用公共许可证(GPL),GNU 宽通用公共许可证(LGPL),伯克利软件部许可证(BSD)和知识共享署名许可证(CC-BY). GPL 和 LGPL 具有最严格的衍生产品许可授权. GPL 和 LGPL 许可下使用任何代码设计,要求新的设计必须继续在这个许可下使用. 但是,LGPL 不需要把所有的模块都转化成 LGPL 许可,只需要对原代码以链接的形式引用许可证就行. 与 GPL 和 LGPL 相比,允许衍生产品使用不同许可证[63] 的 BSD 更加宽容. CC-BY 是一种在知识共享组织下的通用公共许可,其允许商业作品[64] 使用. “公共许可”类别意味着没有版权,开发商已授权,这一类的处理器都是免费的,没有商业和商业用途. “其他”类别意味着作者自定义权限,而不是采用公共许可. 根据著作权法,所有的版权作品被默认的[65] 受版权保护,所以最后没有许可的类别就是指收费的.

3.2 指令集架构(ISA)

ISA 作为硬件和软件之间的接口,可以有效地描述程序语言. 除了建立带有可实现性和兼容性的整体架构,

ISA 还应该被设计成免费 ISA,而不是针对于特定开源处理器的 ISA. 本文所研究的开源处理器 ISA,如表 1 中所列. 在表中,大多数 ISA 是商业公司的知识财产(IP). 匿名 ISA 是由独立开发人员设计的,他们并没有被广泛运用. 除此之外,我们在图 2(*b*)中总结了 ISA 是否免费或者专有. 在这些 ISA 中只有 35% 是免费且不侵权的.

表 1　开源处理器的指令集架构

Supporter	Amber23	Number
Motorola	6800	5
	68000	
	68HC05	
	68HC11	
Intel	8080	5
	8088	
	80186	
	MCS-48	
Atmel	AVR	5
ARM	ARMv2	1
Hitachi	SuperH-2	1
Lattice Semiconductor	LatticeMicro32	1
Micorchip Technology	PIC	6
MIPS Technology	MIPS	11
National Semiconductor	COP4	1
Sun Microsystems	SPARC	4
Texas Instruments	MSP430	1
Zilog	Z80	4
MOS Technology	6502	3
OpenRISC	OpenRISC	3
Xilinx	MicroBlaze	1
	PicoBlaze	1
Opencores	unnamed	14
Opencores	DLX	1
Synopsys	ARC	1
Total		69

3.3　编译器和汇编器

编译器和汇编器是开发工具链最重要的组成部分. 编译器将来源于高级语言的源代码像 C、C ++ 编译成汇编语言. 汇编器用于将汇编语言代码翻译成机器码. 编译器和汇编器决定了处理器的可用性. 除此之外,一个稳定的开源编译器和汇编器能够吸引更多的用户. 在调研的这些处理器中,38 个处理器有编译器和汇编器,12 个仅提供编译器,18 个既没有提供编译器,也没有提供汇编器. 如图 2(*c*).

3.4　验证

经过板级验证或流片验证过的处理器更加的可靠和稳定. 在这 68 个处理器中大约有 68% 的处理器已经在 FPGA 上实现,如图 2(*d*). 他们在 Xilinx、Altera 或者 Actel 的 FPGA 板上实现了应用电路来测试处理器的逻辑功能,而其中的 7 个处理器也在 ASIC 上验证了. 以上经过验证的处理器都提供了验证结果,包括面积消耗和性能. 这些结果都可以作为开发者将开源处理器嵌入到复杂 SoC 系统中非常有价值的参考.

几乎所有汇编器都支持 GCC 或者是类 GCC[66],这是因为 GCC 汇编器是开源且免费的. GCC 授权是发布在 GPL 许可证下. GCC 兼容性强,可在大多数操作系统平台上运行. 最重要的是,GCC 编译出有效代码的性能是最优的. GCC 的所有这些特点吸引了最多的处理器用户和支持者.

3.5　设计文件

设计者决策使用一个开源的硬件产品而不是专有的 IP 时,设计文件和文档的可用性很重要[67].

处理器的设计文件通常包括处理器架构的描述,指令集和处理器的验证的过程. 我们调查处理器并在图 2(*e*)中展示了调查结果. 大约 43% 的处理器没有设计文档. 虽然设计文件是非必须项,但是如果处理器有一个详细的设计文件,用户就可以根据设计文件来理解所述处理器并将其加入到他们的嵌入式系统中.

3.6　总结

本文所研究的稳定的开源软核处理器的特点总结于表 2 中. 该表包含了 68 个在开源许可证下稳定的软核处理器,每个处理器的描述从许可证、ISA、编译器和汇编器、ASIC/FPGA 验证和设计文档等方面进行说明. 许可证、ISA、编译器 & 汇编器和设计文件确定了处理器的可用性,ASIC/FPGA 验证反映了其稳定性. 在表 2 中,除了“匿名”的 10 个处理器使用免费的 ISA,大多数处理器都是专用的 ISA.

4　选择开源处理器

开源处理器的全部特点总结于表 2 中,由于缺乏编译器和汇编器支持,又受到许可证和 ISA 的限制,许多开源处理器可用性非常低. 因此,在本章中,我们的选择将基于许可证、编译器/汇编器的可用性、ISA 是否免费三项最重要指标对开源处理器进行选择.

4.1　选择方法

选择开源处理器的过程如图 3. 首先我们考虑的最重要的因素是许可证. 在 68 个稳定的开源处理器中,我们选择了有开源许可证的 47 个处理器. 因为开源许可证给用户使用权限,可以修改处理器的源代码.

此外,编译器和汇编器是为用户构建一个嵌入式系统必不可少的工具. 从所选择的有开源许可证的 47 个处理器,我们选择了有可用的编译器和汇编器的 36 个处理器. 在配套的编译器和汇编器下,设计师很容易去编译应用程序,而不是开发自己的编译器和汇编程序.

表 2 开源软核处理器的特点:许可证,ISA,编译器和汇编器,验证和设计

Processorss	License	ISA	Compiler & Assembler	Verification	Documents
16-bit Microcontroller	No license	unnamed	C compiler and assembler		√
16-bit CPU based on Caxton Foxter's Blue	GPL	unnamed	A cross compiler		
68hc05	No license	Motorala's MC68HC05	unknown	FPGA	
68hc05	No license	Motorala's MC68HC05	unknown	FPGA	
8080 Compitable CPU	Public domain	Intel's 8080	MI C/Micro Basic	FPGA	
AEMB	Modified BSD	Xilinx's MicroBlaze	GCC tool chain	FPGA	√
Altor32	LGPL	OpenRISC	GCC tool chain	FPGA	
Amber	LGPL	ARMv2	Sourcery G++	FPGA	
Aquarius	GPL	Hitachi's SuperH-2	GNU C for SuperH	FPGA	√
ASPIDA sync/async DLX	LGPL	DLX	Dlxgcc+dlxassembler	ASIC&FPGA	√
AVR Core	No license	Atmel's AT mega 103	WinAVR		
AVR Hyper pipelined	LGPL	Atmel's AVR	WinAVR		√
AX8 mcu	No license	Atmel's 90S1200/2313	unknown		
ClaiRISC	No license	PIC 12-bit	HiTech PICC	FPGA	
Cpu Generator	No license	unnamed	Built-in assembler		√
cpu6502_tc – R6502	GPL	MOS Technoloty's 6502	unknown	FPGA	√
CPU86	GPL	Intel's 8088	Any 8088 assembler		
Data Flow Processor	No license	unnamed	unknown	FPGA	
Educational 16-bit MIPS	LGPL	MIPS16	A assembler written in JAVA		√
Educational RISC	No license	unnamed	unknown	FPGA	√
FORTH processor with Java compiler	LGPL	unnamed	FORTH-assembler and a java compiler	FPGA	√
HC11 Compatible –Gator μProcessor	LGPL	Motorala's 68HC11	HC11 C/C++ Compiler include DCC	FPGA	
HIVE	Others	unnamed	unknown		
LatticeMicro32	GPL	LatticeMicro32	gcc+binutils	FPGA	√
LEM1_9	LGPL	unnamed	Assembler written in C#	FPGA	√
LEON2	LGPL	SPARCv8	BCC	ASIC&FPGA	√
LEON3	GPL	SPARCv9	BCC	ASIC&FPGA	√
Leros	BSD	unnamed	Compiler written in JAVA		
Light 8080 compatible	GPL	Intel's 8080	Small -C	FPGA	√
McAdam's RISC	GPL	unnamed	spr assembler	FPGA	√
MicroSimplez	LGPL	unnamed	unknown	FPGA	
MiniMIPS	LGPL	MIPS I	gasm	FPGA	√
Mini-RISC core	No license	PIC 12-bit	Free MPLAB		
MIPS_enhanced	LGPL	MIPS I	gcc-elf-mips	FPGA	
MIPS32 Release1	LGPL	MIPS32	GCC tool chain		√
Mips 789	No license	MIPS I	gcc-elf-mips	FPGA	√
Mips-FaultTolerant	LGPL	MIPS32	unknown	FPGA	√
mipsr2000	LGPL	MIPS32	unknown	FPGA	√
Navré AVR clone (8-bit RISC)	GPL	Atmel's AVR	unknown	FPGA	
Next 80186	LGPL	Intel's 80186	Macro Assembler	FPGA	√
Next Z80	LGPL	Zilog's Z80	unknown	FPGA	
Open8 μRISC	BSD	Synopsys's ARC	Hi-Tech compiler+binutils	FPGA	
OpenMSP430	BSD	TI's MPS430	MSPGCC	ASIC&FPGA	√
OpenRISC 1000	LGPL	OpenRISC	GCC toolchain	ASIC&FPGA	√
OpenRisc 1200HP	LGPL	OpenRISC	GCC toolchain		√
OpenSPARC T1	GPL	SPARCv9	Solaris Studio	ASIC&FPGA	√
OpenSPARC T2	GPL	SPARCv9	Solaris Studio	ASIC&FPGA	√
Plasma	Public domain	MIPS I	gcc-elf-mips	FPGA	√
PPx16	No license	PIC 12-bit	HiTech PICC	FPGA	
RISC Microcontroller	No license	Atmel's AT90S1200	AVR assembler and AVR studio		√
RISC 16f84	CC-BY	PIC 12-bit	HiTech PICC	FPGA	
RISC5x	LGPL	PIC 12-bit	HiTech PICC	FPGA	
S1 Core	GPL	SPARC v9	GCC toolchain		√
SAYEH educational	LGPL	unnamed	unknown	FPGA	√
System68	No license	Motorala's 6800/6801	unknown		
T400 μController	GPL	National Semicond-uctor's COP400	Macro assembler	FPGA	√
T48 μController	GPL	Intel's MCS-48	Macro assembler	FPGA	√
T65 CPU	No License	MOS Technology's 6502 65c02 65c816	unknown	FPGA	
T80 CPU	No license	Zilog's Z80 8080	Z88dk cross compiler		
TG68	LGPL	Motorala's 68000	unknown	FPGA	
Tiny64	No license	unnamed	assembler		
UCore	No license	MIPS32R2	GCC toolchain	FPGA	
Wishbone Z80	No license	Zilog's Z80	AS80 assembler		√
Y80e	BSD	Zilog's Z80	unknown		√
YACC	No license	MIPS I	gcc-elf-mips	FPGA	√
Yellow Star	No license	MIPS I	GCC toolchain		√
ZPU	FreeBSD+GPL	unnamed	GCC toolchain	FPGA	

注:表中,红色标注的为硬核处理器

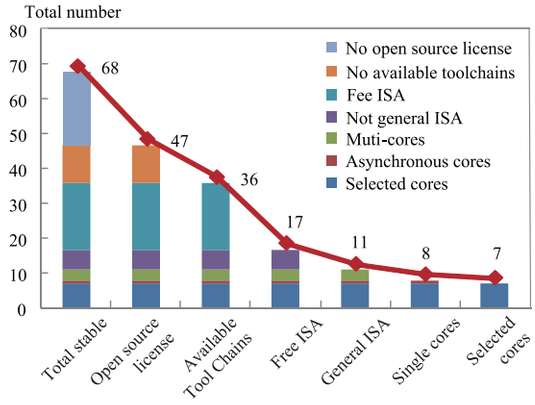


图3　开源处理器的选择过程

最后,ISA 是一个不容忽视的项目,特别当使用商业用途的处理器时. 上文选出的 35 个处理器中,我们选择了配有免费 ISA 的 17 个处理器. 其中,14 个处理器的 ISA 的是匿名的,这些都是独立开发者设计的,没有被广泛使用. 我们倾向于选择了带有通用 ISA 的 11 个处理器,因为他们具有稳定的和良好支持的软件开发工具. 这 11 个已选定的处理器以红色标记在表 2 中,是我们的目标处理器. 排除以上 4 个,剩下的开源处理器分别是 Amber[68,69], LatticeMicro32 (LM32)[70,71], S1[72~76], Altor32[77,78],OpenRISC1200 (OR1200)[79],LEON2[80] 和 LEON3[81]. 这几个备选开源处理器的特点在下一小节介绍.

继续对这 11 个选定的处理器进行分析,OpenSPARC 的 T1,OpenSPARC 的 T2 和 OpenRISC1200 HP 是多核处理器. S1 是 OpenSPARC 的 T1 的简化版本只包含一个 CPU 核心. OpenRISC 1200 HP 能够在相同的设计中找出众多 OpenRISC1200 的例子. ASPIDA 处理器实现 DLX 指令的异步 IP 组结构,它是由一组逻辑来代替一个全局时钟来控制. 因为本文专注于单核同步处理器,所以没有研究这 3 个多核处理器和 1 异步处理器.

4.2　备选开源处理器介绍

每个备选开源处理器的详细功能我们在下面将逐一介绍. 这些处理器内核都选择了哈佛结构,所以其中的许多基本通用单元是如何设计的并没有详细介绍. 位宽,流水线深度,复杂计算单元,缓存架构和内存管理单元(MMU)这几项基本通用单元是以下主要需要考虑的部分.

Amber 处理器是 32 位处理器,它用 ARM v2a ISA. Amber 处理器项目提供了两个版本. Amber23 设计有 3 级流水线,统一指令和数据高速缓存. Amber25 的资源开销更大,提供了比 Amber23 更好的性能. Amber25 有 5 级流水线,独立的指令和数据高速缓存. 两个版本都具有内置的乘法器和一个 Wishbone 总线接口. 这两个版本处理器的缓存大小都是可配置的,但是 MMU 不一样[68,69].

作为 RISC 型开源处理器,LM32 最初设计用于 Lattice FPGA 器件. 支持的 LatticeMico32 ISA,LM32 是 32 位宽,同时具有 32 个通用寄存器和 6 级流水线. LM32 乘法器和除法器单元都是可配置的. 指令缓存和数据缓存的大小也是可配置的,同时,因为 LM32 没有 MMU,也没有独立的指令和数据访问的 Wishbone-存储器接口[70,71].

S1 为 OpenSPARC T1 微处理器的单核版本. S1 实现了基于 SPARC V9 指令集的 64 位处理器体系架构,它包括内建的乘法器,除法器,浮点运算单元(FPU) 和一个 6 级流水线. S1 拥有 4 线程的硬件支持. 每个线程需要 160 个寄存器,包括 128 个通用寄存器和 4 组全局寄存器,每组全局寄存器有 8 个. 总的来说,S1 内含 640 个寄存器,具有 1 个指令 Cache,1 个数据 Cache 和完全相关的指令和页表缓存(TLB). 除了 AMBA 接口,S1 也实现了一个用 64 位数据总线和 1 字节粒度的 Wishbone 接口[72~76].

AltOR32 是一个 32 位 OpenRISC1000 体系结构,包含大部分基本的 ISA 特征,除了与向量,FPU 以及除法器相关的指令和寄存器. AltOR32 被设计成指令位宽 32 位,运行 32 位宽的数据操作. AltOR32 有内置乘法器,独立的指令 Cache 和数据 Cache,没有 MMU. AltOR32 有 5 级流水线,接口是 Wishbone[77,78].

OR1200 是一个 32 位 OpenRISC 架构的处理器. OP1200 的流水线深度为 5,内置乘法器完成基本的数字信号处理(DSP) 操作. OR1200 内部不包括除法器和 FPU. 拥有独立的可直接映射的指令 Cache 和数据 Cache. MMU 以指令 Cache 和数据 Cache TLB 形式实现,OR1200 的总线接口是 Wishbone 型[79].

LEON2 是一个 32 位 SPARC V8 架构的处理器. 5 级指令流水线,通用寄存器数量是在 SPARC 标准(2-32) 限制范围内配置的,默认配置为 8. 总的寄存器数目可以通过计算获得(通用寄存器数 * 16 + 8). 寄存器总数依据通用寄存器数不同,可从 40 变化到 520. FPU 是用硬件乘法器和除法器实现的,三种类型的 FPU 均可用于实现高性能运算操作. 独立的指令 Cache 和数据 Cache 是可配置的,每个可配置 1 ~4 组,1 ~64 K 字节/组,每行 16 ~32 个字节. MMU 可有独立指令 Cache 和数据 Cache 或者 TLB. 该 TLB 可配置为 2-32 的整个关联实体[80].

LEON3 是 32 位,SPARC V8 架构、7 级指令流水线的处理器. 指令流水线级数根据通用寄存器的数目变化而变化,总的寄存器个数范围从 40 到 520. 硬件乘法,除法器,MAC 单元和可配置 MMU 都包括在 LEON3

内.指令 Cache 和数据 Cache 也都可以单独配置.该 MMU 可以有不同的指令和数据或者一个 TLB,TLB 可配置为 2-32 完全关联项[81,82].

4.3 总结

表 3 总结了 7 个备选的开源处理器的位宽,流水线级数,包括乘法器在内的复杂计算单元,Cache 架构以及内存管理单元 MMU 等特点.这些备选的稳定开源处理器的源代码可以在开源许可协议授权下进行访问,支持兼容的编译器和汇编器.备选的开源处理器的 ISA 都是免费的,且有独立的指令 Cache 和数据 Cache,可用于针对不同应用的 SoC 中.

5 基于 FPGA 的板级验证及对比

在本章中,首先简单回顾一下备选开源处理器和由 FPGA 供应商提供的软核处理器的特点.然后,完成以下对比工作:(1)所有备选开源处理器和 Altera 提供的 Nios II 处理器都使用 Altera EDA 平台的 Quartus II 实现,并对实施结果进行了比较和讨论;(2)所有备选处理器和 Xilinx 提供的 MicroBlaze 处理器都使用 Xilinx 的 EDA 工具 ISE 设计套件来实现并比较.之后,相关的开源处理器和商业软核更为详细的讨论在下一章介绍.

5.1 处理器描述和平台配置

备选处理器的特征包括,含有乘法器的复杂计算单元,缓存的总规模和一些可以根据应用电路的不同需求配置处理器的 MMU.所有这些源代码的默认配置已经在表 3 中用“(d)”标注出来.

表 3 位宽,流水线深度,包含乘法器的复杂计算单元,缓存架构和 7 个已选择的开源处理器的内存管理单元(MMU):AMBER,LATTICEMICRO32(LM32),S1,ALTOR32,OPENRISC1200(OR1200),LEON2 和 LEON3

			Amber23	Amber25	LM32	S1	Altor32	OR1200	LEON2	LEON3
Processor Architecture	Bit-width		32	32	32	64	32	32	32	32
	Pipeline Depth		3	5	6	6	5	5	5	7
	Complex Computation Unit	Multiplier	√	√	Optional, √(d)	√	√	√	√	√
		Divider	N/A	N/A	Optional, √(d)	√	N/A	N/A	√	√
		FPU	N/A	N/A	N/A	√	N/A	N/A	√	√
	Register		15	15	32	640	32	32	40-520, 136(d)	40-520,136(d)
	Cache	Total Size/Kbytes	8(d),12,16,32	16,24,32(d)	0,1,2,4(d),8	32(d)	16(d)	16(d)	8(d)	0.008(d)-64
		Instruction Cache/ Kbytes	8(d),12,16,32	8,12,16(d),32	0,1,2(d),4,8	16(d)	8(d)	8(d)	4(d)	0.004(d)-32
		Data Cache/ Kbytes		8,12,16(d),32	0,1,2(d),4,8	16(d)	8(d)	8(d)	4(d)	0.004(d)-32
		Sets	256(d)	256(d)	128,256, 512(d),...	128(d)	256(d)	512(d)	128(d)	1(d)-256
		Associativity/ Ways	2(d),3,4 or 8	2,3,4(d) or 8	1(d),2	4(d)	1(d)	1(d)	1(d)-4	1(d)-4
		Byte-per-line/ Bytes	16	16	4(d),8,16	32(d)	32	16(d)	32(d)	4(d)-8
		Write policy	Write through	Write through	Write through	Write through	Write through	Write through	Write through	Write through
		Replacement policy	Read- miss	Read- miss	Read- miss	LRU	Read- miss	LRU	LRU,LRR, Random(d)	LRU,LRR, Random
	MMU	Shared/ Separate TLB	N/A	N/A	N/A	Separate	N/A	Separate	N/A(d),Optional	N/A(d), Optional
		No. of Instruction TLB entries	N/A	N/A	N/A	64	N/A	16,32,64(d),128	2-32,N/A(d)	2-32
		No. of Data TLB entries	N/A	N/A	N/A	64	N/A	16,32,64(d),128	2-32,N/A(d)	2-32
		No. of Shared TLB entries	N/A	N/A	N/A	N/A	N/A	N/A	2-32,N/A(d)	2-32
Interface			Wishbone	Wishbone	Wishbone	Wishbone /Amba	Wishbone	Wishbone	Amba	Wishbone /Amba

NiosII 核的三个版本是由 Altera 提供的,分别是经济型、标准型和高速型,分别被标记为 NiosII/e、NiosII/s 和 NiosII/f.NiosII/e 实现了小尺寸,它的性能有限.对于高性能,NiosII/f 能够更灵活的配置适应不同的性能需求.如果想要平衡性能和尺寸,可以选择 NiosII/s.

MicroBlaze 是由 Xilinx 提供,且被业界广泛使用的软核.MicroBlaze 可以配置以下参数,流水线深度、数据 Cache 大小、指令 Cache 大小、FPU,硬件乘法器、硬件除法器、硬件可控阵列移位寄存器以及本地存储器总线数据/指令的接口.MicroBlaze 的外围设计了大量的接口,也可以用来作为双核处理器.Nios II 与 MicroBlaze 的特点都概括在表 4 中,本文的设计实现中使用默认设置的地方标有“(d)”.

为了比较备选的开源处理器和商业软处理器,我们将所有备选处理器和 NiosII/e/s/f 分别在 Altera 的 Stratix V FPGA 和 Xilinx Virtex-V FPGA 上实现.以下两小节,本文分别讨论和分析实验结果.

5.2　在 Altera FPGA 上实现并比较性能

开源处理器与 NiosII e/s/f 在 Stratix V FPGA 上实现的对比结果如表 5 所示. NiosII/e/s/f 可以用比所有的开源处理器更少的 ALM 和寄存器来实现. 因为 S1 和 LEON2 比所有其它的处理器需要更多的 ALM、寄存器和 BRAM,而 S1 的最大运算频率是最低的,所以我们排除掉 S1 和 LEON2. 剩下所有其他的开源处理器的最高频率、动态功耗和总共消耗 BRAM 位的结果在图 4 中依次做了对比.

表 4　位宽，流水线深度，包含乘法器的复杂计算单元，缓存架构和内存管理单元（MMU）的 FPGA 供应商提供的软核处理器：Alter 的 NIOS II E / S / F 和单 / 双核的赛灵思 MicroBlaz

			Altera			Xilinx	
			NiosII/e	NiosII/s	NiosII/f	MicroBlaze/single	MicroBlaze/dual
Processor Architecture	Bit-width		32	32	32	32	32
	Pipeline Depth		N/A	5	6	3/5(d)	3/5(d)
	Complex Computation Unit	Multiplier	N/A	Optional,N/A(d)	Optional,√(d)	Optional,√(d)	Optional,√(d)
		Divider	N/A	Optional,N/A(d)	Optional,√(d)	Optional,N/A(d)	Optional,N/A(d)
		FPU	N/A	N/A	N/A	Optional,N/A(d)	Optional,N/A(d)
	Register		32	32	32	32	32
	Cache	Total Size/Kbytes	N/A	2(d),4,...	4(d),8,...	16,32,N/A(d)	16,32,N/A(d)
		Instruction Cache/KBytes	N/A	2(d),4	2(d),4,...	8,16,N/A(d)	8,16,N/A(d)
		Data Cache/KBytes	N/A	N/A	2(d),4,...	8,16,N/A(d)	8,16,N/A(d)
		Sets	N/A	64(d),128	64(d),128,...	512	512
		Associativity/Ways	N/A	1(d)	1(d)	1	1
		Byte-per-line/Bytes	N/A	32(d)	4,16,32(d),...	16(d),32	16(d),32
		Write polity	N/A	N/A	Write through	Write through(d),Write Back	Write through(d),Write Back
		Replacement polity	N/A	unkown	unkown	unkown	unkown
	MMU	Shared/Separate TLB	N/A	N/A	Shared+Separate	Separate+Shared	Separate+Shared
		No. of Instruction TLB entries	N/A	N/A	6(d)	Optional,N/A(d)	Optional,N/A(d)
		No. of Data TLB entries	N/A	N/A	4(d)	Optional,N/A(d)	Optional,N/A(d)
		No. of Shared TLB entries	N/A	N/A	128(d)	64,N/A(d)	64,N/A(d)
Interface			Avalon	Avalon	Avalon	PLB,AXI(d),LMB,FSL,XCL	PLB,AXI(d),LMB,FSL,XCL

表 5　开源处理器和 ALTERA STRATIX V FPGA 的 NIOS II E / S / F 实现结果：ALMS，总寄存器（REGS），总块存储位数（TBMBs），最大频率（F_{max}），芯核动态 / 静态的功耗（P_{CS}/ P_{CD}），I / O 的功耗（$P_{I/O}$），分析和综合的 CPU 时间（$T_{A\&S}$），CPU 时间（TF）.

	Amber23	Amber25	LM32	S1	AltOR32	OR1200	LEON2	LEON3	NiosII/e	NiosII/s	NiosII/f
ALMs	3073	5278	2141	39519	2014	2428	5678	1239	409	590	811
REGs	1875	3221	2468	55061	1754	1191	10546	1272	591	862	1347
T_{BMB}/ bits	152,575	305,664	52,736	270,432	69,632	156,288	72704	8,704	10,240	27,200	44,544
F_{max}(100℃)/ MHz	84.73	96.11	179.92	52.32	94.64	110.3	158.5	212.27	367.51	260.48	297.89
F_{max}(-40℃)/ MHz	83.08	96.15	171.14	56.15	91.99	107.4	158.55	209.95	337.04	250.38	288.68
P_{CS}/ mW	1738.22	1744.96	1735.37	1751.5	1735.08	1737.05	1738.29	1733.99	1733.48	1733.87	1734.5
P_{CD}/ mW	48.44	90.38	24.13	369.47	17.71	30.22	61.56	13.31	10.46	12.28	16.56
$P_{I/O}$/ mW	36.13	62.51	48.38	50.25	43.98	56.56	35.90	43.31	41.91	42.50	42.36
P_t/ mW	1822.78	1897.85	1807.87	2171.22	1796.77	1823.84	1835.75	1790.61	1785.85	1788.66	1793.42
$T_{A\&S}$	4min16s	7min3s	1min22s	12min52s	1min44s	1min12s	1min35s	38s	15s	15s	35s
T_F	20min35s	23min19s	29min17s	1h27min27s	29min35s	20min45s	34min36s	17min47s	25min35s	16min12s	17min19s

图 4(*a*)给出了占用资源的比较结果,其中 X 轴是的 ALM,Y 轴是总寄存器. NiosII/e/s/f 对 ALMs 和寄存器的平均占用率为 19%,而所有开源处理器的平均值为 29%. 在所有备选的开源处理器中,LEON3 处理器的资源数最少.

图 4(*b*)比较了最大频率(F_{max})与流水线深度. NiosII/e/s/f 的 F_{max} 比所有开源处理器高. 在 NiosII 的所有三个版本中,NiosII/e 的 F_{max} 的最高,NiosII/f 最低. 带 7 级流水线的 LEON3 和带 3 级流水线 Amber23 在所有开源的处理器中分别表现为处理频率最高和最低. 开源处理器的 F_{max} 与管道深度成正比.

在表 5 中,所有开源处理器的静态功耗 P_{cs} 几乎是相同的,因为它们是主要由备选的 FPGA 器件本身决定. 动态功耗 P_{cd} 和 ALM 的数量的关系如图 4(*c*)所示,P_{cd} 和 NiosII/e/s/f 成正比,而 NiosII/e/s/f 的 P_{cd} 小于所有开源处理器.

总的块存储器位(TBMBs)主要是由 Cache 大小决定. 图 4(*d*)描述了 TBMBs 和总 Cache 大小的关系. TBMBs 的值随着 Cache 线性变化.

比较 $T_{A\&S}$,NiosII/e/s/f 的分析和综合过程用时最

少,其余的几乎都在 10 分钟以内. $T_{F\&P}$ 只有 S1 超过半小时,其余的几乎都少于半个小时.

5.3 在 Xilinx FPGA 上实现并比较性能

开源处理器和单/双核 MicroBlaze 在 Virtex-7 FPGA 上的试验效果如表 6 所示. 相比单核,双核的 MicroBlaze 需要占用接近两倍的资源. MicroBlaze 占用的 LUT 和寄存器数目比所有的开源处理器更少. 在所有备选开源处理器中,LEON3 占用的资源数最少. MicroBlaze 的 LUT 和寄存器分别是所有开源处理器平均值的 41% 和 58%.

以上比较结果如图 5(*a*)所示,其中 X 轴表示 LUT 数目,Y 轴表示寄存器 Regs 数目. MicroBlaze 可以用比所有开源处理器更少的 LUT 和 Regs 实现. LEON3 是所有备选开源处理器里资源占用数最少的. MicroBlaze 的 LUT 和 Regs 分别是所有开源处理器平均值的 41% 和 58%.

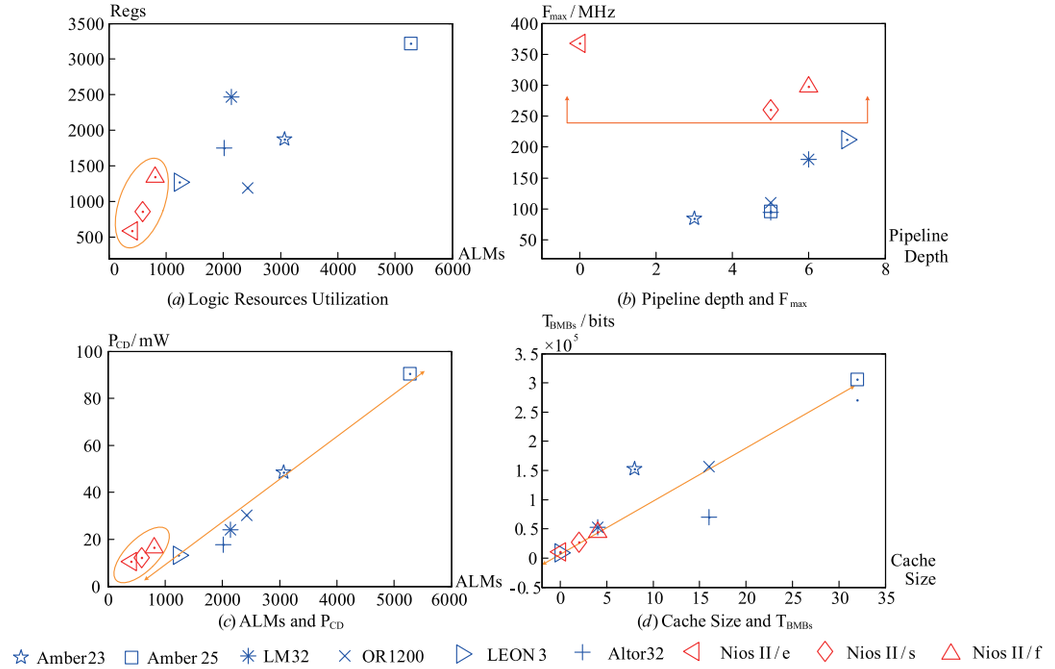


图4 开源处理器和Nios II e/s/f的比较

(*a*)寄存器对比ALMs, (*b*)最高频率(F_{max})与流水线深度, (*c*)芯核静态功耗(P_{CS}), (*d*)总块存储器位数(TBMBs)对比总缓存大小

表 6 开源处理器和单核、双核的 MicroBlaze Xilinx 的 Virtex-7 FPGA 实现结果:查找表(LUT),寄存器(REGS),RAMB36E1'S(N_{R36E1}),RAMB18E1'S(N_{R18E1}),最大频率(F_{max}),动态功率(P_d),静态功耗(P_S),合成及翻译 TIME($T_{S\&T}$)和映射及 P & R 时间($T_{M\&P}$)

		Amber23	Amber25	LM32	S1	AltOR32	OR1200	LEON2	LEON3	MicroBlaze/single	MicroBlaze/dual
LUTs		3274	6592	3281	56690	2742	3669	3674	2522	1491	3507
Regs		2306	3409	2233	38028	1629	1276	1559	1144	1114	2485
Mem	N_{R36E1}	8	16	4	48	2	4	2	0	2	2
	N_{R18E1}	4	8	0	17	1	2	3	2	0	0
F_{max}(25℃)/MHz		114.59	117.40	204.3	65.77	113.27	143.37	192.82	238.83	233.1	216.92
P_s/ mW		429	435	428	476	430	431	430	431	487	487
P_d/ mW		23	83	18	489	37	44	34	44	12	14
P_t/mW		452	519	446	965	467	475	464	474	499	501
$T_{S\&T}$		1min20s	1min45s	15s	11min41s	1min55s	1min37s	1min21s	1min20s	1min8s	2min3s
$T_{F\&P}$		5min45s	7min57s	4min24s	38min37s	7min39s	6min15s	4min42s	4min26s	3min15s	4min3s

图 5(*b*)比较了最大频率(F_{max})与流水线深度的关系. MicroBlaze 的最大频率高于所有除 LEON3 的开源处理器. 开源处理器的最大频率与流水线深度成正比. 所有开源处理器中,拥有 7 级流水线的 LEON3 和 3 级流水线的 Amber23 分别能运行在最高和最低频率.

表 6 中,所有的开源处理器的静态功耗(P_s)几乎是相同的,因为它们主要由选用的 FPGA 器件决定. 动态功耗 P_d 和 LUT 数量的关系如图 5(*c*). P_d 与 LUT 个数线性相关,而 MicroBlaze 的 P_d 小于所有备选的开源处理器.

RAMB36E1 (NR36E1) 的数目和 RAMB18E1 (NR18E1)的数目主要由总的 Cache 大小决定. 图 5(*d*)现实了等效 RAM 块(N_R)和总 Cache 大小之间的关系,其中 $N_R = 2 \times N_{R36E1} + N_{R18E1}$. N_R 的值与 Cache 大小线性相关.

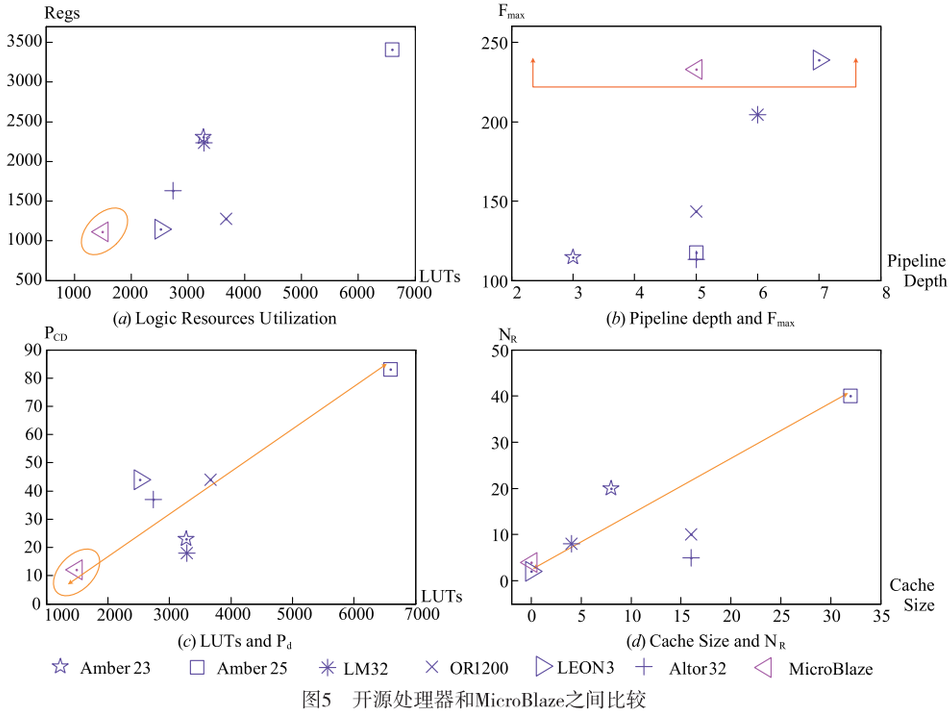


图5　开源处理器和MicroBlaze之间比较

（*a*）寄存器和查找表，（*b*）最高频率（F_{max}）与流水线深度，（*c*）动态功耗（P_d）对比LUT，（*d*）等效RAM块（$N_R=2\times N_{R36E1}+N_{R18E1}$）对比总缓存大小

除此之外,本文还比较了 $T_{S\&T}$ 和 $T_{M\&P}$,可以发现除了 S1 之外,他们几乎都相同.

6　讨论

基于上文的实现结果,我们继续讨论开源处理器和商业化处理器的差异性.

6.1　是否选择开源处理器

FPGA 厂商的商业化处理器专门做过优化. 具体而言,不仅可以减少了逻辑资源占用率,动态功耗和 CAD 工具运行时间,同时还实现高的处理性能. 虽然商业化软核不是开源的,但它们结构灵活. FPGA 的 EDA 工具提供了很多配置选择,设计人员可以根据应用程序的要求使用他们来优化自己的系统.

开源处理器的源代码公开,可自由访问和修改. 其中,无论从研究还是工程的角度,某些性能优良的开源处理器还可用于做架构优化的战略性探索. 而对那些需要深度定制的应用,更可以选择开源处理器以加速项目的开发过程.

6.2　架构与实现

在逻辑资源占用率上,商业软核均优于开源处理器. 对于开源处理器,逻辑资源占用率取决于其结构参数,例如寄存器的数量,Cache 大小,MMU 类型等等,而 ISA 会影响到逻辑资源占用率.

同样地,由供应商优化的商业软核的最大频率比开源处理器更高. 拥有更大的流水线深度的开源处理器,运算频率也更大. OR1200 和 Altor32 具有相同的流水线深度,因此,它俩的最高运算频率也相近.

芯片上的内存使用率取决于总 Cache 大小. 配置更大 Cache 的开源处理器需要更多的 FPGA 片上 BRAM. Amber25 和 S1 具有最大 32 千字节的总 Cache 大小,其结果就是,他们使用了最多的内存块.

开源处理器的动态功耗与逻辑资源数目是成正比的,而静态功耗则主要是由 FPGA 设备本身决定.

6.3　可配置性和便利性

商业软核大都用户友好,设计与使用都是通过 FPGA 厂商自己提供的 EDA 工具进行的. 这些 EDA 工具不仅提供具备极大可配置性的软核,而且提供大量优化的 IP. 通过适当的配置,系统可以采用最合适的软核,最少的外设和高度优化的硬件构造而成.

生而“开放”,开源处理器在代码的工程化优化和研究上具备有定制性和探索性的优势. 更重要的是,工程师和研究人员可以从事任何可能的手段,去试验和验证他们想要进行的深度定制优化是否可行. 所有这些开源的 IP 都可以用在 SoC-FPGA 上.

7　结论

IP 重用技术可以大幅度加速设计开发过程. 开源处理器作为可重用 IP 最重要的组成部分,应该在模块级设

计,验证,测试以及开源共享,来提高设计效率. 面向 SoC-FPGA 的处理器软核可以分成商业软核和开源软核. 本文总结了现有所有的开源处理器特点,从易用性和稳定性等方面,提出了选择开源处理器过程中的要点. 根据这些要点,我们挑选了开源处理器,并对现有的商业化软核和开源软核进行实现,比较并讨论性能.

参考文献

[1] Melo C A R A, Barros E. Oolong: A baseband processor extension to the RISC-V ISA[A]. Proceedings of IEEE International Conference on Application-Specific Systems, Architectures and Processors[C]. USA: IEEE, 2016. 241 – 242.

[2] Gray J. GRVI phalanx: A massively parallel RISC-V FPGA accelerator[A]. Proceedings of IEEE International Symposium on Field-Programmable Custom Computing Machines [C]. USA: IEEE, 2016. 17 – 20.

[3] 张惠国,王晓玲,等. 一种用于 FPGA 配置的抗干扰维持电路[J]. 电子学报,2011,39(5):1169 – 1173.
ZHANG Hui-guo, WANG Xiao-ling, et al. Antijamming holding circuit for FPGA configuration cell[J]. Acta Electronica Sinica, 2011, 39(5): 1169 – 1173. (in Chinese)

[4] Christoforos Economakos, et al. Using advanced FPGA SoC technologies for the design of industrial control applications[A]. Proceedings of the IEEE 6th International Conference on Information, Intelligence, Systems and Applications (IISA)[C]. USA: IEEE, 2015. 1 – 6.

[5] Monmasson E, Cirstea M N. FPGA design methodology for industrial control systems-A review[J]. IEEE Transactions on Industrial Electronics, 2007, 54(4): 1824 – 1842.

[6] Sahoo S K, Das G T R, Subrahmanyam V. Contributions of FPGAs to industrial drives: a review[A]. Proceedings of the International Conference on Information and Communication Technology in Electrical Sciences (ICTES)[C]. UK: IET, 2007. 343 – 348.

[7] Jiang W, Gokhale M. Real-Time classification of multimedia traffic using FPGA[A]. Proceedings of the International Conference on Field Programmable Logic and Applications[C]. USA: IEEE Computer Society, 2010. 56 – 63.

[8] Bueno E J, Hernandez A, Rodriguez F J, et al. A DSP-and FPGA-based industrial control with high-speed communication interfaces for grid converters applied to distributed power generation systems[J]. IEEE Transactions on Industrial Electronics, 2009, 56(3): 654 – 669.

[9] Kasik V, Chvostkova Z. FPGA in technical resources of medical imaging[A]. Proceedings of the IEEE International Symposium on Applied Machine Intelligence and Informatics[C]. USA: IEEE, 2013. 193 – 196.

[10] Coric S, Leeser M, Miller E, et al. Parallel-beam backprojection: an FPGA implementation optimized for medical imaging[A]. Proceedings of the ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays[C]. USA: ACM, 2002. 217 – 226.

[11] Jiang R M, Crookes D. FPGA implementation of 3D discrete wavelet transform for real-time medical imaging [A]. Proceedings of the European Conference on Circuit Theory and Design[C]. USA: IEEE, 2007. 519 – 522.

[12] 王晨旭,韩良,等. 一种适用于 RFID 标签的安全化密码算法实现[J]. 电子学报,2014,42(8):1465 – 1473.
WANG Chen-xu, HAN Liang, et al. A secure cipher implementation suitable for RFID-tags[J]. Acta Electronica Sinica, 2014, 42(8): 1465 – 1473. (in Chinese)

[13] 黄伟,王旭明,等. 一种轻小型遥感相机视频处理 FPGA 软件设计[J]. 电子学报,2014,42(11):2303 – 2309.
HUANG Wei, WANG Xu-ming, et al. Highly-integrated video processing FPGA software design of a remote sensing camera[J]. Acta Electronica Sinica, 2014, 42(11): 2303 – 2309. (in Chinese)

[14] Hasanzadeh A, Edrington C S, Stroupe N, et al. Real-time emulation of a high-speed microturbine permanent-magnet synchronous generator using multiplatform hardware-in-the-loop realization[J]. IEEE Transactions on Industrial Electronics, 2014, 61(6): 3109 – 3118.

[15] Rui J, Rui C, Yu L C, et al. Low cost 1D DCT core for multiple video codec[J]. Chinese Journal of Electronics, 2016, 25(6): 1052 – 1057.

[16] Zhang C, Li P, Sun G, et al. Optimizing FPGA-based accelerator design for deep convotional neural networks [A]. Proceedings of the IEEE/ACM International Symposium on Field-Programmable Gate Arrays (FPGA)[C]. USA: IEEE, 2015. 161 – 170.

[17] Lee J, Rahman A, Choi K. Efficient FPGA acceleration of convolutional neural networks using logical-3D compute array[A]. Proceedings of the IEEE Conference Publications[C]. USA: IEEE, 2016. 1393 – 1398.

[18] Tredennick N, Shimamoto B. Prospects for reconfigurable systems[J]. IEEE Micro, 2014, 34(1): 72 – 78.

[19] Wilson R. Cpus in fpgas: many faces to a trend (2011)[OL]. http://www.edn.com/electronics-news/4369558/CPUs-in-FPGAs-many-faces-to-a-trend. 2017-05-31.

[20] Ahmed S Z, Sassatelli G, et al. Survey of new trends in industry for programmable hardware: FPGAs, MPPAs, MPSoCs, structured ASICs, eFPGAs and new wave of innovation in FPGAs[A]. Proceedings of the International Conference on Field Programmable Logic and Applications [C]. USA: IEEE Computer Society, 2010. 291 – 297.

[21] Kuon I, Tessier R, Rose J. FPGA architecture: Survey and challenges[J]. Foundations and Trends in Electronic De-

sign Automation,2008,2(2):135 – 253.
[22] Chen C L P,Zhang C Y. Data-intensive applications,challenges,techniques and technologies:A survey on Big Data [J]. Information Sciences,2014,275:314 – 347.
[23] Ciresan D C,Meier U,Gambardella L M,et al. Deep,big,simple neural nets for handwritten digit recognition[J]. Neural Computation,2010,22(12):3207 – 3220.
[24] Fpgas everywhere – the cloud and mobile[OL]. http://www10. edacafe. com/nbc/articles. 2017-05-31.
[25] First time ever at acm/sigda conference:Fpgas for mobile apps[OL]. http://www. eetimes. com. 2017-05-31.
[26] Saleh R,Wilton S,Mirabbasi S,et al. System-on-chip:reuse and integration[J]. Proceedings of the IEEE,2006,94(6):1050 – 1069.
[27] Raveendran A,Patil V B,Selvakumar D,et al. A RISC-V instruction set processor-micro-architecture design and analysis[A]. Proceedings of the International Conference on VLSI Systems,Architectures,Technology and Applications[C]. USA:IEEE,2016. 1 – 7.
[28] Barthe L,Cargnini L V,Benoit P,et al. Optimizing an open-source processor for FPGAs:a case study[A]. Proceedings of the International Conference on Field Programmable Logic and Applications [C]. USA: IEEE Computer Society,2011. 551 – 556.
[29] Chen Q,Yi X,Wang M,et al. Design and implementation of IP core for AS5643 serial bus[J]. Electronic Measurement Technology,2016,39(3):78 – 86.
[30] Viseur R. From open source software to open source hardware[A]. Open Source Systems:Long-Term Sustainability[C]. Berlin Heidelberg:Springer,2012. 286 – 291.
[31] Davidson S. Open-source hardware[J]. Design & Test of Computers IEEE,2004,21(5):456 – 456.
[32] Tong J G,Anderson I D L,Khalid M A S. Soft-core processors for embedded systems[A]. Proceedings of the International Conference on Microelectronics [C]. USA: IEEE,2006. 170 – 173.
[33] Fletcher B H. FPGA embedded processors[A]. Proceedings of the Embedded Systems Conference[C]. San Francisco USA:Memec,2005. 1 – 18.
[34] Yiannacouras P,Steffan J G,Rose J. VESPA:portable,scalable,and flexible FPGA-based vector processors[A]. Proceedings of the International Conference on Compilers,Architectures and Synthesis for Embedded Systems [C]. USA:ACM,2008. 61 – 70.
[35] Yiannacouras P,Steffan J G,Rose J. Exploration and customization of FPGA-based soft processors [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2007,26(2):266 – 277.
[36] Meng X,Zeng H,Lawal N,et al. Portability analysis of soft microprocessor for FPGA[A]. Proceedings of the Embedded Computing[C]. USA:IEEE,2012. 5 – 8.
[37] Salgado F,Garcia P,et al. A customizable processor architecture for a design space exploration framework[A]. Proceedings of the IEEE International Conference on Industrial Technology[C]. USA:IEEE,2012. 129 – 133.
[38] Plavec F. Soft-Core Processor Design[D]. Canada:University of Toronto,2004.
[39] Prakash A,Lam S K,Singh A K,et al. Rapid design exploration framework for application-aware customization of soft core processors[A]. Proceedings of the International Conference on Field Programmable Logic and Applications[C]. USA:IEEE,2009. 539 – 542.
[40] Sheldon D,Vahid F,Lonardi S. Interactive presentation:Soft-core processor customization using the design of experiments paradigm[A]. Proceedings of the Design,Automation & Test in Europe Conference & Exhibition[C]. USA:IEEE,2007. 821 – 826.
[41] Kamte M G,Bhaskar M P. Design and implementation of reusable processor independent IP core for SoC platform [J]. International Journal of Engineering Trends and Technology,2011,5(2):69 – 72.
[42] Open Design[OL]. http://videowiki. root2020. com/index. php?yturl = Open_design. 2017-05-31.
[43] Barthe L,Cargnini L V,Benoit P,et al. Optimizing an open-source processor for FPGAs:a case study[A]. Proceedings of the International Conference on Field Programmable Logic and Applications [C]. USA: IEEE Computer Society,2011. 551 – 556.
[44] The New Site is Still Under Construction (Do Not Worry It Will be Available Soon)[OL]. http://www. geocities. ws/jamilkhatib75/. 2017-05-31.
[45] 开源硬件简史[OL]. http://www. eefocus. com/Kevin/blog/07-06/2238_5cfa2. html/. 2017-05-31.
[46] OpenRISC[OL]. https://openrisc. io/. 2017-05-31.
[47] LEON2 Processor[OL]. http://vlsicad. eecs. umich. edu/BK/Slots/cache/www. gaisler. com/products/leon2/leon. html. 2017-05-31.
[48] Production Media Networking-Coveloz[OL]. https://opencores. org/project,mfpga. 2017-05-31.
[49] OCRP-1 Board:Overview[OL]. https://opencores. org/project,ocrp-1. 2017-05-31.
[50] SoC Interconnection: WISHBONE[OL]. http://opencores. org/opencores,wishbone. 2017-05-31.
[51] 吴瑞阳,汪文祥,等. 龙芯 GS464E 处理器核架构设计[J]. 中国科学:信息科学,2015,45(4):480 – 500.
WU Rui-Yang, WANG Wen-Xiang, et al. Design of Loongson GS464E processor architecture[J]. Scientia Sinica:Informationis,2015,45(4):480 – 500. (in Chinese)

[52] 王慧慧. 片上共享 Cache 的参数化设计及验证[D]. 湖南:国防科技大学,2015.
[53] 北京君正集成电路股份有限公司. 北京君正发布新一代 65nm XBurst CPU JZ4775[J]. 集成电路应用,2013,(5):40 -40.
[54] 赵雅. 北大众志 UniCore-Ⅱ处理器中存储管理单元设计与实现[D]. 北京:北京大学,2007.
[55] 苏州国芯科技有限公司. C ~ * Core CPU 设计技术[J]. 中国集成电路,2009,18(5):24 -25.
[56] 张军. 基于 CK510 嵌入式 CPU 的 32 位 MCU 的 RTL 设计与验证[D]. 杭州:杭州电子科技大学,2011.
[57] 张海军,白书敬,尉红梅,唐大国. Java 虚拟机在国产申威处理器平台上的移植初探[A]. 全国高性能计算学术年会(HPC China2011)[C]. 山东济南,2011. 1 -7.
[58] OpenCores[OL]. http://www. opencores. org/projects. 2017-05-31.
[59] Soft Microprocessor[OL]. http://. en. wikipedia. org/wiki/Soft _ microprocessor. 2017-05-31.
[60] CPLD[OL]. http://zh. wikipedia. org/zh-cn/CPLD/. 2017-05-31.
[61] Fontana R, Kuhn B M, Moglen E, et al. A Legal Issues Primer for Open Source and Free Software Projects V1. 5. 2[M]. Software Freedom Law Center,2008.
[62] OpenCores[OL]. https://en. wikipedia. org/wiki/OpenCores. 2017-05-31.
[63] Open Source Initiative[OL]. http://www. opensource. org//license. 2017-05-31.
[64] Help Us Build a Vibrant, Collaborative Global Commons [OL]. http://www. creativecommons. org/license. 2017-05-31.
[65] Various Licenses and Comments About Them[OL]. http://www. en. gnu. org/licenses/licenselist. html. 2017-05-31.
[66] Gough B, et al. An Introduction to GCC[M]. USA: Network Theory, Ltd, 2004.
[67] OSHW Community Survey 2013[OL]. http://www. oshwa. org/oshw-community-survey-2013. 2017-05-31.
[68] Amber 2 Core Specification[OL]. http://opencores. org. 2017-05-31.
[69] Amber ARM-Compatible Core: Overview[OL]. http://opencores. org/project, amber. 2017-05-31.
[70] LatticeMico32 Processor Reference Manual[OL]. https://www. yumpu. com/en/document. 2017-05-31.
[71] LatticeMico32 Open, Free 32-Bit Soft Processor[OL]. http://www. latticesemi. com/en/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores02/LatticeMico32. aspx. 2017-05-31.
[72] Simply RISC S1 Core Specification[OL]. http://www. srisc. com/download. 2017-05-31.
[73] OpenSPARC T1 Micro Architecture Specification[OL]. http://users. ece. utexas. edu/ _ mcdermot/vlsi-2. 2017-05-31.
[74] OpenSPARC T1 Processor Datasheet[OL]. http://read. pudn. com/downloads161/ebook/726659/doc. 2017-05-31.
[75] OpenSPARC T1 Processor Megacell Specification[OL]. http://read. pudn. com/downloads161/ebook/726659/doc. 2017-05-31.
[76] S1 Core: Overview: OpenCores[OL]. http://opencores. org/project, s1 _ core. 2017-05-31.
[77] AltOr32-Alternative Lightweight OpenRisc CPU: Overview [OL]. http://opencores. org/project, altor32. 2017-05-31.
[78] OpenRISC 1000 Architecture Manual[OL]. http://www. cl. cam. ac. uk/research/srg/han/ACS-P35/documents. 2017-05-31.
[79] OpenRISC 1200 IP Core Specification[OL]. http://openrisc. net/or1200-spec. html. 2017-05-31.
[80] LEON2 Processor Users Manual[OL]. http://www. dit. upm. es/ _ str/ork/documents. 2017-05-31.
[81] Evaluation of the LEON3 Soft-Core Processor Within a Xilinxradiation-Hardened Field-Programmable Gate Array [OL]. http://prod. sandia. gov/techlib. 2017-05-31.
[82] Aeroflex Gaisler. SPARC V8 32-bit Processor LEON3/LEON3-FT Companion Core Data Sheet, March, Version 1. 1 (2010)[OL]. http://prod. sandia. gov/techlib. 2017-05-31.

作者简介

余 乐(通信作者) 男,1983 年 6 月生于湖北黄冈,现为北京工商大学计算机与信息工程学院讲师. 现在的研究兴趣:类脑计算与大数据技术.
E-mail:yule@ btbu. edu. cn

李任伟 男,1984 年 4 月出生于云南禄劝,现为中国科学院自动化研究所高级工程师,研究兴趣为微处理器体系结构研究与设计.