计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202330060
60(6): 1204−1215, 2023

基于 SoC-FPGA 的 RISC-V 处理器软硬件系统级平台

齐 乐[1] 常轶松[1,2] 陈欲晓[1,2] 张 旭[1,2] 陈明宇[1,2] 包云岗[1,2] 张 科[1,2]
[1](处理器芯片全国重点实验室(中国科学院计算技术研究所) 北京 100190)
[2](中国科学院大学计算机科学与技术学院 北京 100049)
(qile@ict.ac.cn)

A System-Level Platform with SoC-FPGA for RISC-V Hardware-Software Integration

Qi Le[1], Chang Yisong[1,2], Chen Yuxiao[1,2], Zhang Xu[1,2], Chen Mingyu[1,2], Bao Yungang[1,2], and Zhang Ke[1,2]
[1](*State Key Lab of Processors* (*Institute of Computing Technology*, *Chinese Academy of Sciences*), *Beijing* 100190)
[2](*School of Computer Science and Technology*, *University of Chinese Academy of Sciences*, *Beijing* 100049)

Abstract Building a system-level prototype platform with FPGAs for hardware-software integration of one processor design under test (DUT) is an essential step in pre-silicon evaluations of the processor chip design. In order to meet design requirements of open-source processors based on the emerging open RISC-V instruction set architecture with minimized FPGA development efforts, we propose a system-level platform with a tightly-coupled SoC-FPGA chip for agile hardware-software integration and evaluation of RISC-V DUT processors. In specific, we first elaborate the interconnect between the DUT and the SoC via the existing SoC-FPGA interfaces. Then we introduce a scheme of virtual inter-processor interrupt to support highly-efficient collaboration between the DUT and the hardcore ARM processor in the SoC-FPGA. As a result, the DUT is able to flexibly leverage various I/O peripherals for full-system evaluation. The hardcore ARM processor is also involved for acceleration of the DUT's time-consuming software workloads. Additionally, we build a configurable framework on cloud for flexible composition and system integration of the DUT's hardware and software components. Based on our evaluation results with a couple of target RISC-V processors, we believe that our proposed platform is of great significance in improving efficiency and shortening iteration period when building a system-level prototype platform.

Key words pre-silicon system-level platform; hardware-software full-system evaluation; RISC-V instruction set processor; SoC-FPGA

摘 要 构建软硬件系统级原型平台是处理器设计硅前测试中必不可少的环节.为适应基于开放指令集RISC-V的开源处理器设计需求,简化现有基于FPGA的处理器系统级原型平台构建方法,提出了一套基于SoC-FPGA的处理器敏捷软硬件原型平台,以实现目标软硬件设计的快速部署与系统级原型高效评测.针对上述目标,发掘紧耦合SoC-FPGA器件的潜力,构建了一套RISC-V软核与ARM硬核(SoC侧)之间的信息交互机制.通过共享内存和虚拟核间中断等方法,可使目标RISC-V处理器灵活使用平台丰富的I/O外设资源,并充分利用硬核ARM处理器算力协同运行复杂软件系统.此外,为提升软硬件系统级平台的敏捷性,构建了灵活可配置的云上自动化开发框架.通过对平台上目标RISC-V软核处理器各方面的分析评估,验证了该平台可有效缩短系统级测试的迭代周期,提升RISC-V处理器软硬件原型评测效率.

收稿日期:2023-01-10;**修回日期**:2023-05-04
基金项目:中国科学院战略性先导科技专项(XDA0320000, XDA0320300);国家自然科学基金重大项目(62090020).
This work was supported by the Strategic Priority Research Program of Chinese Academy of Sciences (XDA0320000, XDA0320300), and the Major Program of the National Natural Science Foundation of China (62090020).
通信作者:张科(zhangke@ict.ac.cn)

关键词 硅前系统级平台；软硬件全系统评估；RISC-V 指令集处理器；SoC-FPGA

中图法分类号 TP302

敏捷开发最初是指以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发. 而近年来随着当今处理器核心设计需求的多样性与复杂性不断增加，一些前沿领域的芯片设计团队也开始尝试借鉴敏捷开发的思想[1-3]. 处理器芯片设计在投片前，需要基于近似真实的硬件环境，敏捷地进行软硬件系统定义、集成和试错，开展软硬件系统级评估.

在早期的处理器开发流程中，由于对硬件的依赖，处理器相关软件的开发和验证需要稳定的硬件原型，这种硬件开发和软件开发之间的串行性在一定程度上降低了整体的开发与迭代效率. 为了缩短开发周期，软硬件协同开发[4]的思路被提出. 软硬件协同开发在设计初期将系统功能进行软件和硬件的划分，随后软件和硬件的开发并行进行，验证阶段采用软硬件协同验证的方式. 软硬件协同验证是一种对软硬件设计同时进行验证的方法，要求在验证框架下搭建待验证硬件环境并运行待验证软件，运行过程中软硬件的行为可以彼此提供测试向量，从而将验证工作并行化，缩短验证时间. 同时，软件应根据硬件配置进行设计，综合启动引导固件、操作系统、应用程序等各层次的代码，组织成完整的软件栈. 随着硬件设计的规模逐渐扩大，软件栈需要跟踪各种硬件接口（时钟与复位逻辑、访存接口、各类对外通信设备等）的准确版本以及从固件到用户级应用程序的特性和功能，因此软件栈的开发和测试变得更加复杂. 设计复杂性的不断增加对实验和研究软件工作负载的管理提出了挑战.

处理器设计的系统性改进升级可能涉及修改任何层次代码，包括从低级电路的硬件描述代码到操作系统和应用程序的软件代码. 综上所述，处理器设计团队需要一个硬件和软件完全集成的原型验证系统. 全系统级验证是对软件或硬件合作的尝试. 系统级验证指对系统层面的硬件设计进行验证，验证对象通常是软硬件整体. 系统级验证的方法主要采用对软硬件整体仿真或模拟的方法，如搭建现场可编程门阵列（field programmable gate array, FPGA）原型系统进行验证，就是一种目前业界广泛采用的基于可编程硬件模拟的方法. 近年来，由于 FPGA 器件成本的下降和逻辑资源规模的提升，加上 FPGA 本身与真实电路的近似性，FPGA 原型验证作为系统级验证的一种方式，其优势愈加明显.

现有的系统级原型验证平台主要基于通用型 FPGA 器件. 通用型 FPGA 器件以提供可编程逻辑资源为目标，几乎不包含硬核，设计流程上主要通过 FPGA 制造商提供的电子设计自动化（electronics design automation, EDA）支持工具来构建设计并配置器件，从而构建 FPGA 原型系统. 各制造商的 FPGA 开发过程基本相同，但开发过程往往涉及工程方面的繁杂细节. 在基于传统通用型 FPGA 的原型验证中，除了需要测试的处理器核部署在 FPGA 上之外，还需要实现一些基本的内存访问接口和 I/O 外设（包括但不限于网络接口、串行端口、USB、SD/MMC 等）. 没有 FPGA 开发经验的人需要相当长的时间来学习掌握，技术门槛较高. 这些接口的实现本身也需要占用 FPGA 的大量片上逻辑资源，从而挤压待测试的处理器核心的布局和布线自由度空间，带来时序违例等问题. 此外由于芯片底层实现工艺的原因，相较于专用集成电路（application specific integrated circuit, ASIC），通用型 FPGA 的原型验证场景中处理器核的运行频率低，执行一些计算密集型任务往往需要消耗相当长的时间，而这些等待时间也会间接影响到处理器的验证流程.

1 研究背景

基于面向开放 RISC-V 指令集[5]的开源处理器在近年来受到广泛关注. 随着开源芯片社区的不断发展和成熟，处理器内部的一些典型基础逻辑功能模块也在历经多次流片验证后趋于稳定，这些基础模块在未来爆发大规模逻辑错误和设计缺陷的可能性已大幅度降低。然而，随着系统整体复杂度的不断提升，目前的处理器集成了越来越多的基础模块部件，因此，未来不仅需要开展面向传统 RTL（register transfer logic）逻辑设计描述和门级电路的仿真测试，更需要一种快速构建软硬件原型的平台和方法，以尽快开展系统级测试评估. 设计团队亟需一种利用开源 IP 快速构建硅前系统级平台的方法，尽快开展硅前硬件纠错和软件调试工作.

随着片上系统（system-on-a-chip, SoC[6]）技术的演进，出现了在同一芯片上集成 FPGA 可编程逻辑（programmable logic, PL）和硬核处理系统（processing system, PS）的新型 SoC-FPGA[7]器件. 以 Xilinx 的 Zynq-7000/MPSoC 为例，其硬核处理系统部分是一个以 ARM 处

理器硬核为核心的 SoC，它包含硬核 ARM 处理器、平台管理单元(platform management unit, PMU)协处理器及各类丰富的外设接口资源. SoC-FPGA 可以认为是 ASIC 器件与传统 FPGA 器件的互补，将一些广泛使用的标准化处理器核与外设接口以硬核的形式直接提供给开发者复用.

如何充分利用 SoC-FPGA 提供的各类外设接口及算力资源，为开源 RISC-V 处理器核提供一套敏捷易用的原型验证环境，以更好地开展软硬件全系统评估工作，是本文着力解决的问题.

2 国内外研究现状

FireMarshal[8] 是一个用在全栈 SoC 上创建和共享可复制和可重复实验的软件工件的工具. 用户可以将配置项写入 JSON 文件，以快速生成所需的所有级别的软件栈. 然而，FireMarshal 侧重于 RISC-V 软件系统的自动化生成，在硬件上依赖 FireSim[9] 仿真加速平台，尚未充分考虑如何为 RISC-V 处理器的完整软硬件系统级验证提供丰富的 I/O 外设接口.

Freedom[10] 是 SiFive 设计的一个平台，用于将特定的 RISC-V 核——Rocketchip[11] 部署到传统通用型 FPGA 设备，为设计者提供一套典型的 RISC-V SoC 参考系统. 虽然 Freedom 开源代码仓库中提供了一些简单 I/O 接口的使用方法，但接口类型和功能极为有限，不能满足目标处理器软硬件系统的完整验证需求. 此外，Freedom 提供的 I/O 接口可选择使用 Xilinx FPGA 开发工具中提供的 IP，但这些 IP 核仍需要以逻辑电路的形式被部署在 FPGA 上，占用待验证处理器需要的逻辑资源.

ESP[12] 是用于异构 SoC 设计的开源研究平台. ESP 体系结构具有高度可扩展性，并在规则性和专业性之间取得平衡. 伴随的方法提高了系统级设计的抽象级别，并支持从软件和硬件开发到 FPGA 上的全系统原型的自动化过程. 对于应用程序开发人员，ESP 提供面向特定领域的自动化解决方案，为其软件合成新的加速器，并将复杂的工作负载映射到 SoC 架构上. 对于硬件工程师来说，ESP 提供了将加速器设计集成到完整 SoC 中的自动化解决方案.

国内也有团队开始探索基于 SoC-FPGA 构建 RISC-V 处理器的测试系统[13]. 其基本思路是通过对 RISC-V 发起的外设地址访问请求进行捕获，将 RISC-V 处理器通过 AXI 外设控制总线对外设控制器访问产生的相应数据转存，最终通过串口打印. 然而该方法仅能对 RISC-V 处理器的外设访问通路进行相关访问数据记录的收集，转存机制本身也难以保证数据交互的实时性，同时也未考虑对复杂高速 DMA 外设的访问支持还无法满足完整的系统级验证需求.

综上，尽管国内外已开始重视 RISC-V 有关的软硬件协同验证系统，但主要沿用传统的通用型 FPGA 模式，测试待验证处理器所需的接口通常需要直接在 FPGA 上实现. 特别是对于一些高速接口，即使能够应用 Xilinx 或第三方公司提供的 IP 核保证功能可靠，但这些 IP 核需要占用大量的逻辑资源，基于 Xilinx ZynqMP ZU2EG 系列器件测得的各 IP 以默认配置设置时所需的逻辑资源数量与所占总逻辑资源的比例，如表 1 所示. 而目前基于 SoC-FPGA 构建 RISC-V 处理器测试系统的技术尝试，尚存在一定的局限性.

Table 1 Logical Resources Required by the IP of Several Peripheral Interfaces

表 1 若干外设接口 IP 所需的逻辑资源

IP	查找表	触发器	块存储器
SDIO	1 097（2.3%）	9 944（10.5%）	1（3.0%）
Ethernet	3 106（6.6%）	5 256（5.5%）	4（12.0%）
USB 2.0	2 388（5.0%）	2 135（2.2%）	3（9.0%）
PCIe 2.0	16 444（35.0%）	12 421（13.2%）	20（60.0%）

针对传统通用型 FPGA 在构建复杂系统方面的这些缺点，本文面向 RISC-V 处理器提出了一种基于 SoC-FPGA 的 RISC-V 处理器软硬件系统级平台. 该平台充分发掘可编程 SoC-FPGA 软硬件协同可编程的潜力，通过构建各级硬件和软件的设计，使部署在硬件可编程逻辑部分的待验证目标 RISC-V 处理器核能够充分利用硬核 SoC 侧的 I/O 外设资源，并与 ARM 硬核处理器进行协同，以获得较好的仿真验证效果. 最后，通过与一系列定制化软硬件自动化设计框架，用户只需提供几个关键参数即可自动生成可在 SoC-FPGA 上部署的目标 RISC-V 开源处理器软硬件镜像文件，降低原型验证环境的构建开销.

3 验证平台的设计

如图 1 所示，待测 RISC-V 软核处理器位于可编程逻辑侧，可通过不同类型的总线接口(访存接口和外设接口)访问 ARM 硬核 SoC 侧的内存与 I/O 外设资源；同时 I/O 外设可通过一致性接口向 RISC-V 可见内存发起 DMA 访问. 此外，为了提高验证平台在软硬件各个层次上的兼容性，本文提出一种基于

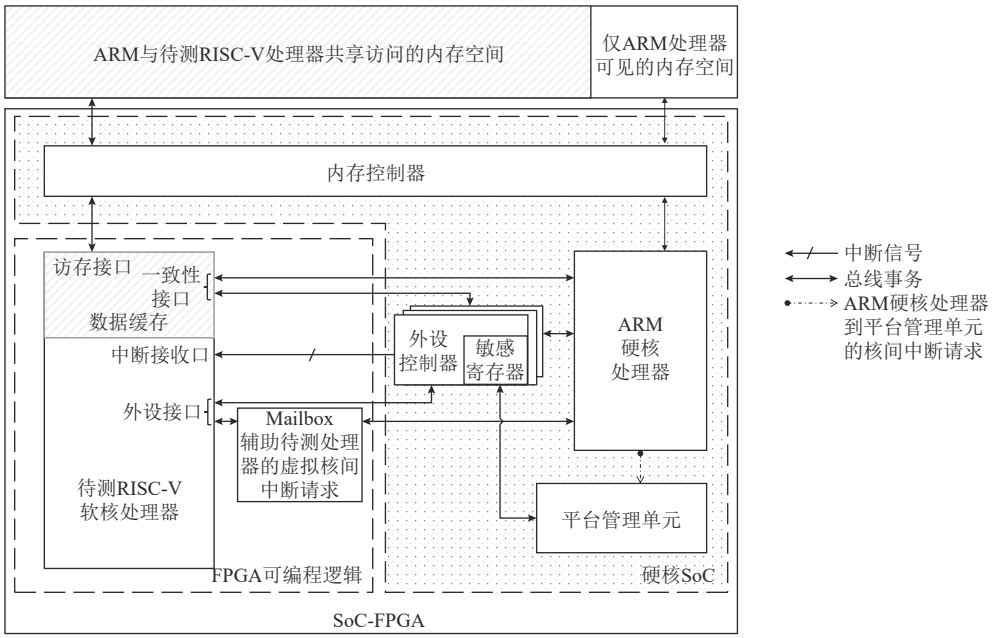


Fig. 1 Overall schematic diagram of prototype verification platform based on SoC-FPGA

图 1 基于 SoC-FPGA 的原型验证平台总体示意图

Mailbox 模块的软硬件协同的验证平台设计框架，通过 RISC-V 软核与 ARM 硬核对 Mailbox 模块的共享访问，实现硬核处理器对 RISC-V 处理器软硬件验证的高效协同.

3.1 目标 RISC-V 处理器访问内存与 I/O 外设

完整的软硬件协同验证需要待测试 RISC-V 处理器核与各种总线接口之间进行频繁交互. 一些特定的测试数据、测试结果也需要通过相关的总线接口与待验证处理器进行传输.

在验证平台框架中如图 1 所示，待验证的 RISC-V 处理器核主要包含 3 类总线接口，分别为访存接口、一致性接口和外设接口，其中外设接口一般使用内存映射输入输出（memory-mapped I/O, MMIO）的方式工作. 为了充分利用硬核处理系统侧的各项资源，本文将可编程逻辑侧待验证处理器的 MMIO 接口连接到硬核处理系统端，并允许待验证处理器访问硬核处理系统端的各种 I/O 外设；将待验证处理器通过访存接口访问硬核处理系统侧的内存控制器. 一些具有大量通信数据的外设设备，如用于访问 SD/MMC 便携存储设备的安全数字输入输出（secure digital input and output, SDIO）接口和以太网，其 DMA 交互接口也与待验证处理器的缓存一致性接口连接. 此外，Mailbox 模块可以像普通外设一样由 MMIO 接口直接寻址访问，为可编程逻辑侧的待验证处理器提供了一种间接访问 PMU 的方式在 ARM 处理器的算力协助工作中发挥重要作用.

硬核处理系统[14]中的 UART、SDIO、网络等外设既可以被可编程逻辑中的待验证处理器通过接口访问，也可以被硬核处理系统中的 ARM 处理器访问，但不应该被二者同时占用. 在验证平台启动过程中，首先由 ARM 处理器完成平台初始化和引导文件加载工作，之后各外设的控制权被移交给待验证处理器，以实现外设资源的复用. 各外设资源在平台启动过程中的使用情况为：

1）UART 接口. 在验证平台启动期间，ARM 处理器和待验证处理器的各个启动阶段的日志信息，以及挂载文件系统后的交互都可以通过 UART 接口完成输入输出.

2）SDIO 接口. 在基于 SD 卡的引导模式下，ARM 和待验证处理器引导文件都需要从 SD 卡移动到内存. 启动 Linux 内核后，待验证处理器还可能需要在 SD 卡分区上挂载发行版文件系统，也需要访问 SDIO 接口.

3）网络. 在基于网络的启动模式下，ARM 和待验证处理器都需要从网络的远端服务器读取启动镜像文件到内存. 启动 Linux 内核后，待验证处理器还可能需要在网络文件系统（network file system, NFS）服务器上挂载发行版文件系统，这也需要访问网络接口.

值得注意的是，由于 ARM 处理器在通电和启动的初始阶段仅暂时使用 SDIO 或网络接口，一旦开始执行服务程序，ARM 将不再访问硬核处理系统端的各类外设资源. 因此，待验证处理器从开始加载启动

到挂载发行版文件系统的整个过程,对硬核处理系统端的所有接口资源都具有完全权限.

3.2 基于硬核处理器协同的待测 RISC-V 处理器运行时环境

硬核处理系统侧的处理器核心主要包括 ARM 主处理器和 PMU 协处理器 2 部分. ARM 主处理器作为硬核处理系统的高性能处理器,主频较高,而且上电后立即开始工作,可直接进行访存以及各类外设接口的交互操作,能够向待测 RISC-V 处理器软核提供算力方面的协助. PMU 是一个在 SoC-FPGA 上执行管理功能的硬核处理器. PMU 可以执行其他处理器不被允许的操作,如电源管理、时钟管理和可编程逻辑配置,能够向待测 RISC-V 处理器软核提供功能上的辅助. 待测 RISC-V 核对 2 种硬核处理器资源的利用在结构上均需要基于 Mailbox 进行关键信息的交互,如图 1 所示. 3.2.1 节将对硬核处理器资源的利用方法,以及不同处理器之间的具体交互流程进行阐述说明.

3.2.1 平台管理单元协同

基于 3.1 节的方法虽然能够访问到 I/O 外设的控制与状态寄存器空间,但无法直接访问一些对系统行为较为敏感的特殊寄存器(如 I/O 外设的时钟寄存器). 这些特殊寄存器只能由 PMU 管理,其他处理器核没有直接访问权限. 当待验证 RISC-V 处理器需要读写特殊寄存器时,需要向 PMU 发出核间中断请求,委托 PMU 进行特殊寄存器的读写操作. 因此需要建立待验证 RISC-V 处理器与 PMU 的核间中断通信机制. 然而,现有 SoC-FPGA 器件往往并不允许位于可编程逻辑的待测处理器直接向 PMU 单元发出核间中断操作.

为解决这些问题,本文提出一种虚拟核间中断机制,通过 ARM 硬核处理器作为第三方,对核间中断(inter-processor interrupt, IPI)请求进行转发. 具体地,本文提出一种基于 Mailbox 的核间中断请求转发流程. 同时,为了对特殊寄存器的操作与普通寄存器操作进行隔离,本文在 RISC-V 的机器态运行模式(machine mode, M-MODE)下实现了程序调用接口,允许在其他运行模式下通过环境调用(environment call, ECALL)模式访问.

如图 2 所示,Mailbox 主要包括请求标记段和参数段,参数段包括请求类型段、输入参数段与输出参数段 3 部分,可基于 FPGA 片上小容量 BRAM 实现. 请求标记段由 RISC-V 在需要时进行置位和回读,ARM 则在默认情况下循环检测该请求标记. 为了当 RISC-V 开始执行 PMU 访问时首先通过 ECALL 陷入 M-MODE,将核间中断的相关信息通过 MMIO 总线接口写入 Mailbox 的相应位置. 在 Mailbox 模块的配合下,RISC-V 核对 PMU 等单元的操作请求将由 ARM 进行中转,代替 RISC-V 向 PMU 发出操作请求从而实现虚拟核间中断机制,最终 RISC-V 通过 Mailbox 回读核间中断操作的返回结果,图 2 中,从平台的角度上,各操作按数字序号的次序依次执行;从 RISC-V 与 ARM 处理器的角度上,各自按照其执行步骤箭头依次执行.

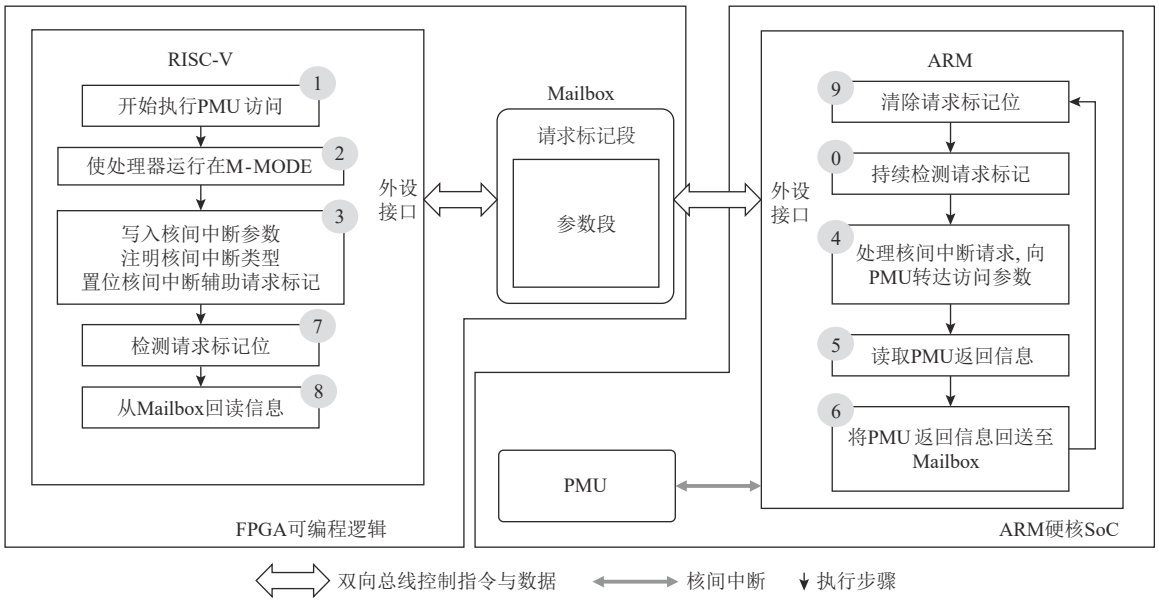


Fig. 2 Interaction process between RISC-V processor and PMU

图 2 RISC-V 处理器与 PMU 的交互流程

基于 Mailbox 构建的虚拟核间中断机制将在平台的操作系统引导过程中起到重要作用. 在待测 RISC-V 软核启动时, Linux 内核的初始化阶段需要陷入 M-MODE 的管理器二进制接口(supervisor binary interface, SBI)固件代码中执行核间中断请求. RISC-V 处理器上的 Linux 内核运行于 S-MODE, 更高级别的 M-MODE 运行 OpenSBI(OpenSBI 是 SBI 的一种开源实现, 除了作为引导程序以外, 同时也作为平台管理固件运行于处理器的最高级别). 本文兼容了现有的 SoC-FPGA 时钟驱动程序, 通过添加相应的接口函数来支持 RISC-V 从内核态陷入 M-MODE 固件, 发起对 PMU 的虚拟核间中断, 如图 3 所示.

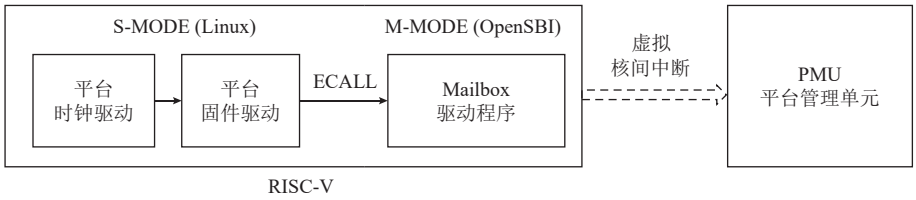


Fig. 3 Linux drivers running on RISC-V processor initiate a virtual IPI towards PMU

图 3 RISC-V 处理器上运行的 Linux 驱动程序发起对 PMU 的虚拟核间中断请求

3.2.2 ARM 处理器协同

在验证平台中, ARM 处理器负责为 RISC-V 处理器核的启动准备内存环境, 并将 RISC-V 的启动镜像文件从 SD 卡提前移动到内存, 完成可编程逻辑端的位流配置等. 并在启动和运行期间最终执行服务程序以协助 ECALL 和 RISC-V 的其他操作. 除了 RISC-V 操作所需的辅助外, ARM 处理器还可以承担一些计算任务.

例如, 在涉及 RISC-V 安全框架的研究中, ARM 处理器不仅负责协助 RISC-V 启动和转发 IPI 消息, 它本身还具有更高的工作时钟频率和更大的计算潜力. 对于一些计算密集型的任务, RISC-V 可以通过 Mailbox 向 ARM 提交计算辅助请求以及所需要计算的任务类型、原始数据信息等, 如图 4 所示. ARM 接收到计算辅助请求后, 将通过一致性接口读取所需计算的原始数据, 之后将计算结果数据由一致性接口写入共享内存区域, 并通过清除请求标记告知 RISC-V 去回读共享内存中的计算结果, 从平台的角度上, 各操作按数字序号的次序依次执行; 从 RISC-V 与 ARM 处理器的角度上, 各自按照其执行步骤箭头依次执行.

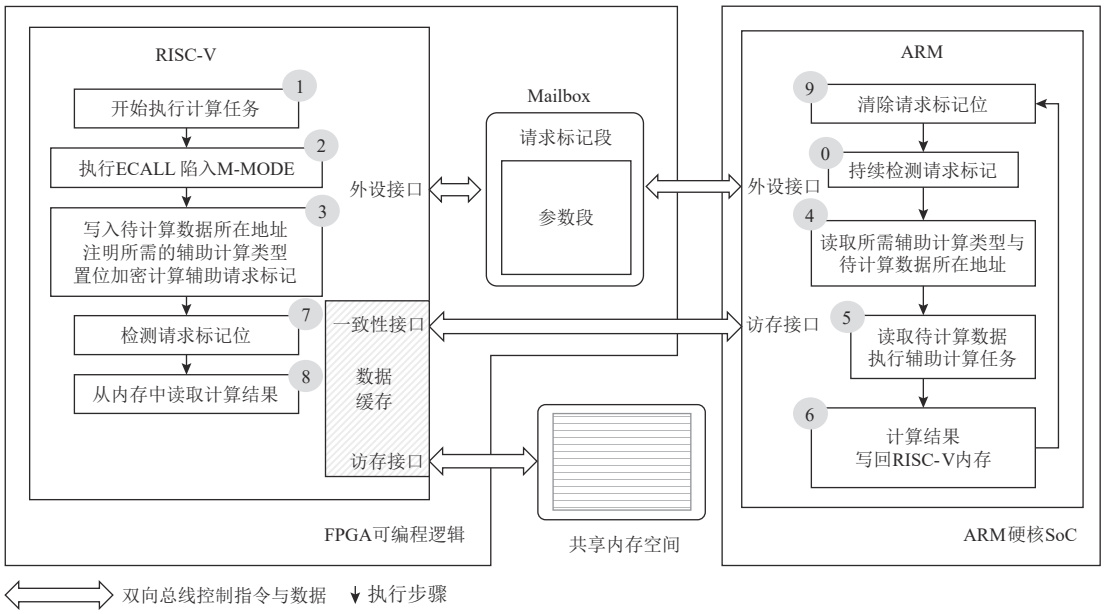


Fig. 4 ARM processor cooperates to run complex computing tasks of RISC-V

图 4 ARM 处理器协同运行 RISC-V 的复杂运算任务

3.3 敏捷验证平台开发框架

为了提升软硬件协同验证平台的敏捷性, 本文构建了 RISC-V 处理器软硬件验证的可配置、自动化开发框架.

3.3.1 可配置框架

验证平台通过多层 Git 存储库组织, 将有关的软

件和硬件仓库以高内聚、低耦合的原则组织成一个集成的设计环境. 用户仅需要向平台的接口文件提供关键配置参数, 如硬件处理器核的相关配置、软件代码的入口地址等. 如图 5 所示, 平台通过一系列逐层调用并相互协作的 Makefile 和工具命令语言(tool command language, Tcl)脚本, 将这些关键参数配至各个代码仓库的编译传参接口, 自动进行各个软硬件模块的参数修改以及 FPGA 工程的适配工作, 而不需要用户深入平台内部进行繁琐的手动修改. 当出现某些模块的更替时仅需要关注关键参数即可, 从而使得平台能够快速兼容各个软硬件层次的设计变化.

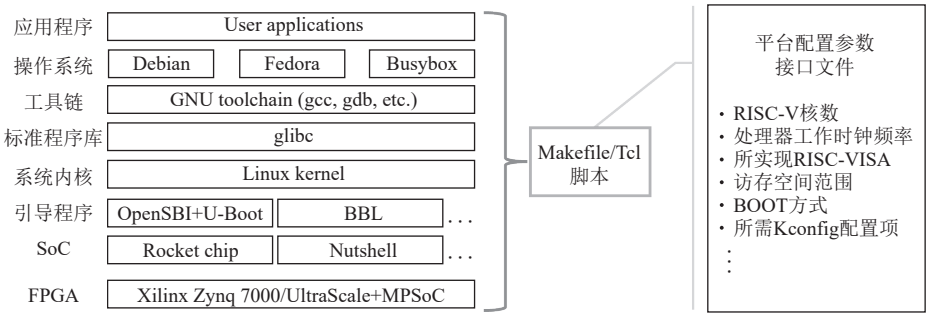


Fig. 5 Organization structure of platform on software and hardware codes

图 5 平台软硬件代码组织结构

软硬件协同验证平台的敏捷性应该体现在其可以快速高效地集成任何符合标准接口的软件与硬件模块. 软件模块包括软件协议栈的各层次代码, 例如启动引导软件、Linux 操作系统、应用层可执行程序的不同阶段的软件代码; 而硬件模块则主要包括处理器层次的逻辑设计与配置代码与板卡层次的硬件脚本代码. 对于不同的软硬件层次, 需要分别配置特定的参数, 如表 2 所示.

Table 2 Configuration Parameters Required for Integration of Different Software and Hardware Modules

表 2 不同软硬件层次整合时所需要提供的配置参数

软硬件层次	配置参数
处理器核	各总线地址范围、核数、内存入口信息
板卡硬件	器件 IO 物理约束、时钟源、内存映射信息
启动引导软件	起始地址、设备树、总线地址范围、引导方式
操作系统	内核配置项信息
应用层程序	所依赖的操作系统、标准程序库、编译链

3.3.2 自动化框架

考虑到系统级原型验证涉及到很多软硬件各层次代码, 本文探索在云端将全栈的代码仓库组织起来, 使用户在本地或服务器端, 自动化执行编译和测试流程. 由于编译需要许多特定版本的工具链, 基于持续集成(continuous integration, CI)的编译方法可以将不同的编译阶段分成特定顺序的 job, 各个 job 使用包含所需编译工具链的独立容器镜像, 相比本地编译更简单、更可靠. 除了编译阶段, 本文也探索了将编译产出的目标文件通过持续部署(continuous deployment, CD)更新到设备.

4 验证平台的评估

4.1 验证平台评估实验环境

4.1.1 硬件环境

平台的硬件环境包括 2 种自研板卡平台, 实物如图 6 所示, 分别是基础实验平台(Plat-B)板卡, 以及扩展实验平台(Plat-E)板卡, 2 种板卡的各项板载基础配置参数分别如表 3 所示. Plat-E 配备了逻辑资源更大的 SoC-FPGA 器件, 可以提供更多的片上存储资源以支持相对复杂的应用场景(如基于 RISC-V 的可信执行环境).

4.1.2 目标软核处理器

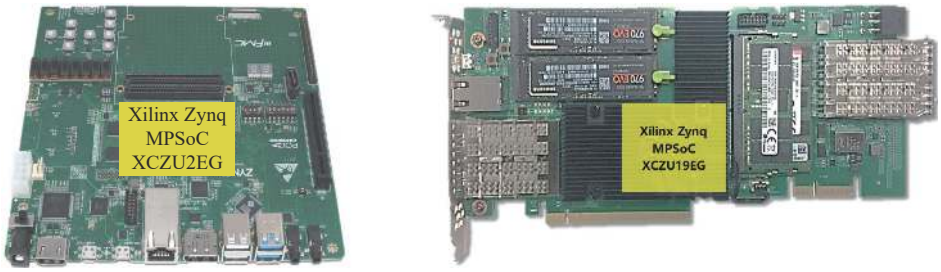
在评估实验中, Plat-E 板卡与 Plat-B 板卡上 RISC-V 软核处理器的基本配置相同, 但 Plat-E 板卡为 RISC-V 提供的访存空间更大、可编程逻辑侧提供的时钟频率更高, 如表 4 所示.

4.1.3 软件负载

基于表 3、表 4 硬件环境, 通过部署并运行一系列软件负载, 如表 5 所示, 评估本文提出的验证平台性能.

1)验证平台的 I/O 外设性能

硬核处理系统侧的各类外设接口在交予可编程逻辑侧的待测试处理器使用时, 由于片内通路较长, 需要更多的总线中转, 可能会产生一定的性能下降.



(a) 基础实验平台 (Plat-B) (b) 扩展实验平台 (Plat-E)

Fig. 6 Physical photos of platform hardware

图 6 平台硬件实物照片

Table 3 Basic Configurations of Two Experimental Platforms

表 3 2 个实验平台的基本配置

配置项	Plat-B	Plat-E
ARM 硬核处理器	ARM Cortex-A53 Quad-core @ 1.3GHz	
内存	2GiB DDR4-2133 内存颗粒	16GiB DDR4-2133 SODIMM
RISC-V 可见内存容量/GiB	1.75	15.75
UART 波特率	115 200	
SD/MMC	32GB，CLASS 10	
以太网网络接口速率/（$Mb \cdot s^{-1}$）	1 000	
ARM 和 RISC-V 间的互连接口时钟频率/MHz	100	134

Table 4 Basic Configuration of RISC-V Soft Core Processor

表 4 RISC-V 软核处理器基础配置

配置项	Plat-B	Plat-E
时钟频率/MHz	100	134
基本配置	Rocketchip v1.2 单核 BaseSubsystemModuleImp	

Table 5 Software Workload

表 5 软件工作负载

类型	基本信息
引导程序	OpenSBI v0.5 + U-Boot v2020.01
操作系统	Linux v5.4.0 + Debian 11
测试负载	①网络性能测试：iperf v3.7 ②SD/MMC 存储性能测试：fio v3.25 ③复杂运算协同：Keystone 可信执行环境，内核驱动 v0.3 + 安全监视器（提交编号 adba60d9aa3）

对于大多数待验证处理器的功能和性能测试，特别是需要运行 Linux 的场景，考虑到文件系统通常需要放置在 SD/MMC 卡或 NFS 服务器上，并且数据输入和输出也取决于 SD/MMC 存储介质或网络接口. 为了测试本文所述的原型验证平台是否可以提供足够的网络与 SD 卡访问能力以满足系统级测试的基本需求，本文基于 Plat-B 板卡的硬件环境设计了一些典型的比较试验，对验证平台的网络与 SDIO 这 2 个典型外设接口进行了性能测试和评估.

2）验证平台的算力辅助

在原型验证中，待测处理器软核往往不能以高时钟频率运行. 当待测处理器必须执行一些耗时的计算任务时，长时间的等待往往会给其他后续验证过程带来麻烦. 硬核处理系统侧的 ARM 处理器可以辅助进行一些较为复杂的计算密集型任务.

如何提供一个安全可信执行环境是未来处理器设计中的重要发展方向. 在现有可信执行框架下，基于第三代安全散列算法[15-16]（secure Hash algorithm 3，SHA-3）的度量操作属于一种计算密集型任务，它占整体运行时间的 80.4%. 本文在 RISC-V 开源处理器核上部署 UC Berkley 发布的开源可信执行环境 Keystone[17-18]，探索将 SHA-3 核心计算任务委托到 ARM 硬核处理器进行处理的方法，并进行性能评估.

4.2 验证平台 I/O 外设性能评估

在 Xilinx Zynq UltraScale+ MPSoC 设备上，本文分别在硬核处理系统侧使用 ARM 处理器，在可编程逻辑侧使用待验证处理器核来测试两侧处理器访问通信的各项性能. 网络带宽 Inbound（验证平台接收来自外界发送的数据）和 Outbound（验证平台向外界发送数据）的情况如表 6 所示. 实验评估了分别由 ARM 硬核和 RISC-V 软核访问 SDIO 接口时的性能参数，如图 7 所示，从左到右依次为 SDIO 接口的带宽、每秒读写操作次数、延迟测试结果.

Table 6 Network Performance Evaluation of Prototype Verification Platform

表 6 原型验证平台的网络性能评估

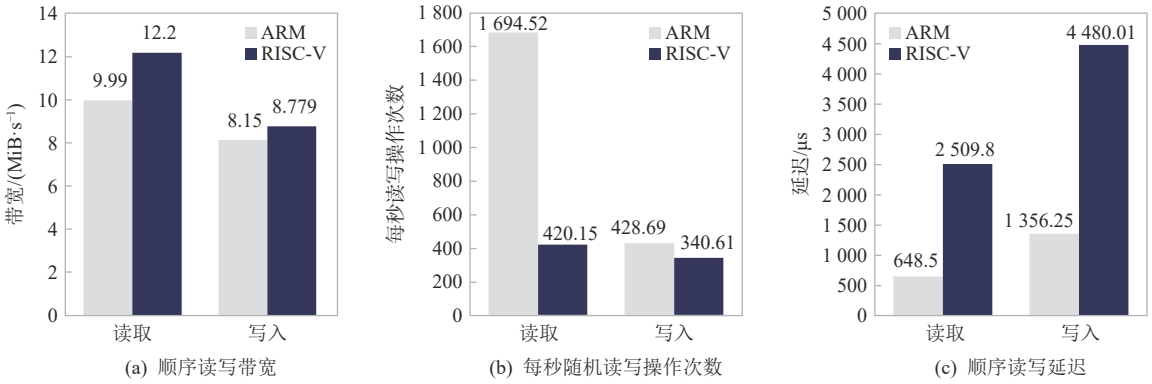
数据包长度/B	RISC-V Inbound/(Mb·s^{-1})	ARM Inbound/(Mb·s^{-1})	RISC-V Outbound/(Mb·s^{-1})	ARM Outbound/(Mb·s^{-1})
64	54.3	251	0.47	180
128	55.1	501	0.94	322
256	55.3	901	1.87	447
512	54.5	891	3.78	727
1K	54.0	893	8.42	810
2K	56.0	893	11.3	921
4K	55.3	892	20.4	941
8K	54.8	891	30.5	941
16K	54.5	893	41.1	941
32K	55.8	893	50.2	941
64K	55.4	890	51.3	941
128K	55.9	891	51.6	941

从芯片架构来看, 硬核处理系统侧的 ARM 处理器具有更高的运行主频(1.5GHz), 其到网络外设设备的数据和控制路径更短. Inbound 和 Outbound 的测试性能均优于被测的 RISC-V. 尽管如此, 被测的 RISC-V 仍然可以保证一定的网络传输带宽, 在大多数原型验证工作中不会出现明显的瓶颈, 可以满足系统级测试的基本需求.

4.3 验证平台算力辅助效果评估

基于硬核处理系统侧的 ARM 处理器来进行 SHA-3 计算的基本思路如图 4 所示. ARM 可以按 RISC-V 地址空间, 以常规访存方式, 通过 RISC-V 缓存一致性接口读取数据. 执行完成后, 通过 ARM 侧显式地执行缓存写回操作, 将 ARM 侧的运算结果主动通过 RISC-V 一致性接口直接写回 RISC-V 内存, 无需显式地进行结果数据在 ARM 与 RISC-V 内存中的搬移.

逐一运行 Keystone 的功能测试集进行算力辅助效果测试. 在实验中 RISC-V 软核处理器通过 Mailbox 将用于 SHA-3 计算的数据块的起始地址信息传递给 ARM 处理器. ARM 运行特定的加速程序对该地址的数据块执行 SHA-3 计算. ARM 辅助计算的方法显著加快了 SHA-3 计算速度, 平均将加解密计算运行时间减少了 81.13%. 由于 SHA-3 的计算得以辅助进行, Keystone 框架中各功能测试集的执行时间均显著缩短, 如图 8 所示.



(a) 顺序读写带宽 (b) 每秒随机读写操作次数 (c) 顺序读写延迟

Fig. 7 SDIO performance evaluation of prototype platform

图 7 原型平台的 SDIO 性能评估

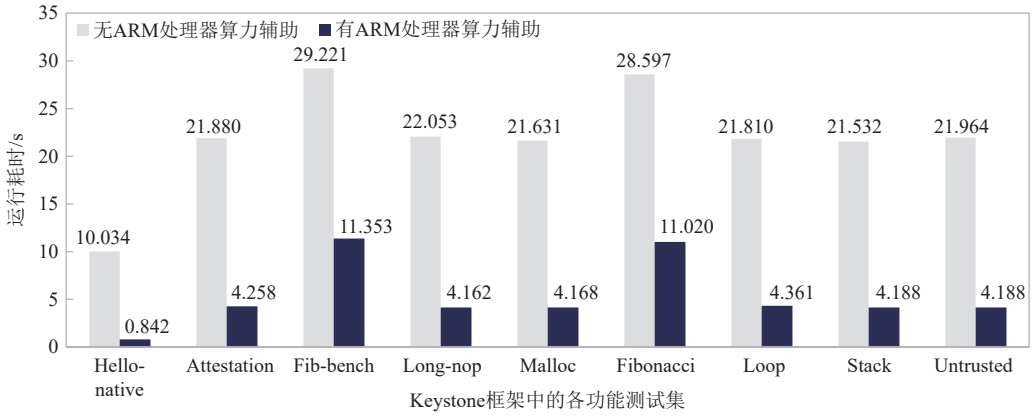


Fig. 8 Evaluation of the computing power's auxiliary effect of the ARM processor on the prototype platform

图 8 原型平台的 ARM 处理器算力辅助效果评估

4.4 验证平台敏捷性效果评估

为进一步验证本文所提出的平台在敏捷性方面的改进程度，本文在 4.1 节所述的在基准平台基础上，分别对 3 个处理器实例进行移植，如表 7 所示.

Table 7 Amount of Work Required to Integrate Different Software and Hardware Modules

表 7 整合不同软硬件模块所需要的工作量

目标处理器	平台修改项目与手动改动代码行数
实例 I 基础配置开源“果壳”处理器[19-20]	替换处理器核代码子仓库、总线地址范围；替换启动引导软件栈修改代码量：131 行
实例 II 支持脉冲神经网络加速的“果壳”处理器[21]	替换处理器核代码子仓库修改代码量：12 行
实例 III 支持虚拟化扩展的四核 Rocketchip[22]	替换处理器核代码子仓库、核数配置项；U-Boot 与 Linux 内核配置项修改代码量：77 行

如表 7 所示，当需要实现实例 I 的目标处理器时，平台在基准 Rocketchip 环境的基础上，仅需将代码仓库中的处理器核子仓库替换为“果壳”处理器子仓库，根据相应的设计要求修改 DRAM 起始地址与内存范围，此外考虑到“果壳”处理器核不包含硬件浮点计算单元，需要启动引导软件提供浮点模拟辅助，因此增加 BBL（Berkeley boot loader）代码仓库到启动引导软件栈路径下，最后修改顶层软件栈相应编译脚本即可.

当需要实现实例 II 的目标处理器时，则以实例 I 为基础，替换支持脉冲神经网络加速的“果壳”处理器子仓库，继续沿用实例 I 处理器的其他改动即可直接重新编译并部署新的处理器核到平台上.

当需要实现实例 III 的目标处理器时，平台首先将虚拟化扩展的四核 Rocketchip 子仓库替换基准 Rocketchip 子仓库，再修改 U-Boot 与 Linux 的配置项以支持 RISC-V 的虚拟化功能即可.

最后，根据 3 个实例所需的包括对平台脚本的配置项、Makefile 与 Tcl 脚本代码修改等工作量进行了评估，如表 7 所示.

综上，对于以上软硬件层次的需求变更，仅需较少的手动修改工作即可快速整合进平台，开启新一轮的部署与验证迭代.

5 结 论

处理器芯片设计在投片前需要基于近似真实的软硬件环境，敏捷地进行软硬件系统定义、集成和试错. 为更好地支持敏捷的软硬件协同验证技术，本文提出了一套基于 SoC-FPGA 的系统级全栈平台，支持主流开源 RISC-V 处理器软硬件栈，支撑软硬件协同的评测工作.

未来的工作包括进一步提升平台兼容性，以便为设计规模更大、逻辑结构更为复杂的国产香山 RISC-V 开源处理器项目[23-24]提供系统级原型环境. 对于香山处理器核所需的可编程逻辑资源规模，现有的 SoC-FPGA 器件尚不足以提供支持，本文正在探索相关解决方案，以构建逻辑资源规模更大的系统级平台.

本文所述的敏捷软硬件系统级评估平台已经开放①，并发布了平台的使用说明②，欢迎申请试用并提出宝贵建议.

作者贡献声明：齐乐实施实验并撰写论文；常轶松负责提供实验方案与详细工作思路；陈欲晓和张旭提出相关意见并修改论文；陈明宇、包云岗和张科提供总体指导.

参 考 文 献

[1] Wang Huizhe, Tang Dan, Yu Zihao, et al. Open-source chip, RISC-V and agile development[J]. Big Data Research, 2019, 5(4): 50−66 (in Chinese)
(王诲喆, 唐丹, 余子濠, 等. 开源芯片、RISC-V与敏捷开发[J]. 大数据, 2019, 5(4): 50−66)

[2] Yu Zihao, Liu Zhigang, Li Yiwei, et al. Practice of chip agile development: Labeled RISC-V[J]. Journal of Computer Research and Development, 2019, 56(1): 35−48 (in Chinese)
(余子濠, 刘志刚, 李一苇, 等. 芯片敏捷开发实践: 标签化RISC-V[J]. 计算机研究与发展, 2019, 56(1): 35−48)

[3] Lee Y, Waterman A, Cook H, et al. An agile approach to building RISC-V microprocessors[J]. IEEE Micro, 2016, 2(36): 8−20

[4] Teich J. Hardware/software co-design: The past, the present, and predicting the future[J]. Proceedings of the IEEE, 2012, 100: 1411−1430

[5] Liu Chang, Wu Yanjun, Wu Jingzheng, et al. Survey on RISC-V system architecture research[J]. Journal of Software, 2021, 32(12): 3992−4024 (in Chinese)
(刘畅, 武延军, 吴敬征, 等. RISC-V指令集架构研究综述[J]. 软件学报, 2021, 32(12): 3992−4024)

[6] Rashinkar P, Paterson P, SINGH L. System-On-A-Chip Verification: Methodology and Techniques[M]. Norwell: Kluwer Academic

① https://gitlab.agileserve.org.cn:8001.

② https://gitlab.agileserve.org.cn:8001/ICT-SERVE-FARM/serve-docs/user-manual.git.

Publishers, 2002
[7] Altera. What is An SoC-FPGA?[EB/OL]. (2020-08-13)[2023-01-01].https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ab/ab1_soc_fpga.pdf
[8] Pemberton N, Amid A. FireMarshal: Making HW/SW co-design reproducible and reliable[C]//Proc of 2021 IEEE Int Symp on Performance Analysis of Systems and Software (ISPASS). Piscataway, NJ: IEEE, 2021: 299−309
[9] Karandikar S, Mao H, Kim D, et al. FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud[C] //Proc of 2018 ACM/IEEE 45th Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2018: 29−42
[10] SiFive. Source files for SiFive's Freedom platforms [EB/OL]. (2016-11-24) [2023-01-01].https://github.com/sifive/freedom
[11] Asanovic K, Avizienis R, Bachrach J, et al. The rocket chip generator[R]. Berkeley: University of California, 2016
[12] Mantovani P, Giri D, Guglielmo G D, et al. Agile SoC development with open ESP[C] //Proc of 2020 IEEE/ACM Int Conf on Computer Aided Design (ICCAD). Piscataway, NJ: IEEE, 2020: 1−9
[13] Chong qing haiyun jiesoon science and technology. SoC FPGA-based RISC-V hardware test method and system: China, CN202210145663.7 (重庆海云捷迅科技有限公司. 基于SoC FPGA的RISC-V硬件测试方法及系统: 中国, CN202210145663.7[P]. (2022-5-24) [2023-04-01].)
[14] Xilinx. Zynq ultraScale+ device technical reference manual [EB/OL]. (2020-04-26) [2023-01-02]. https://www.xilinx.com/support/documentation/user_guides/ug1085-zynq-ultrascale-trm.pdf
[15] Li Mengdong, Shao Penglin, Li Xiaolong. A short review on SHA-3 winner: Keccak[J]. Journal of Beijing Electronic Science and Technology Institute, 2013, 21(2): 18−23 (in Chinese)
(李梦东, 邵鹏林, 李小龙. SHA-3获胜算法: Keccak评析[J]. 北京电子科技学院学报, 2013, 21(2): 18−23)
[16] Wang Hai Tao. Security analysis and implementation of Keccak in SHA-3 standard[D].Xidian University, 2017
(王海涛. SHA-3标准Keccak算法的安全性分析与实现[D]. 西安: 西安电子科技大学, 2017)
[17] Lee D, Kohlbrenner D, Shinde S, et al. Keystone: An open framework for architecting TEEs [EB/OL]. (2022-04-02) [2022-12-19]. http://docs.keystone-enclave.org/en/latest/
[18] Lee D, Kohlbrenner D, Shinde S, et al. Keystone: An open framework for architecting trusted execution environments [C] //Proc of the 15th European Conf on Computer Systems. 2020[2022-12-18].https://dl.acm.org/doi/proceedings/10.1145/3342195.
[19] Wang Huaqiang. Nutshell: A linux-compatible RISC-V processor designed by undergraduates[C/OL] //Proc of RISC-V Global Forum 2020 [2023-01-01].https://riscvglobalforum2020.sched.com/event/dO2g/nutshell-a-linux-compatible-risc-v-processor-designed-by-undergraduates-huaqiang-wang-university-of-chinese-academy-of-sciences
[20] Wang Huaqiang, Zhang Zifei, Zhang Linjuan, et al. OSCPU/NutShell: RISC-V SoC designed by students in UCAS GitHub repository [EB/OL]. (2020-07-07) [2022-12-28].https://github.com/OSCPU/NutShell
[21] Wang Jiulong, Wu Ruopu, Chen Guokai. RISC-V toolchain and agile development based open-source neuromorphic processor[EB/OL]. (2022-09-30)[2023-01-01].https://gitee.com/openmantianxing/wenquxing22a
[22] B. Sá, J. Martins and S. Pinto. A first look at RISC-V virtualization from an embedded systems perspective[J]. IEEE Transactions on Computers, 71(9): 2177−2190
[23] Xu Yinan, Yu Zihao, Tang Dan, et al. Towards developing high performance RISC-V processors using agile methodology[C] //Proc of the 55th IEEE/ACM Int Symp on Microarchitecture (MICRO) . Piscataway, NJ: IEEE, 2022: 1178−1199
[24] Wang Kaifan, Xu Yinan, Yu Zihao, et al. XiangShan open-source high performance RISC-V processor design and implementation[J]. Journal of Computer Research and Development, 2023, 60(3): 476−493 (in Chinese)
(王凯帆, 徐易难, 余子濠, 等. 香山开源高性能RISC-V处理器设计与实现[J]. 计算机研究与发展, 2023, 60(3): 476−493)

Qi Le, born in 1994. Bachelor, assistant engineer. His main research interests include system-level FPGA prototyping and FPGA cloud.
齐 乐, 1994 年生. 本科, 助理工程师. 主要研究方向为 FPGA 系统级原型验证、FPGA 云.

Chang Yisong, born in 1985. PhD, associate professor. His main research interests include computer architecture and heterogeneous computing.
常轶松, 1985 年生. 博士, 高级工程师. 主要研究方向为计算机系统结构和异构计算.

Chen Yuxiao, born in 1998. Master candidate. His main research interests include hardware design and verification, FPGA debugging, and simulation acceleration.
陈欲晓, 1998 年生. 硕士研究生. 主要研究方向为硬件设计与验证、FPGA 调试、仿真加速.

Zhang Xu, born in 1996. PhD candidate. His main research interests include memory disaggregation, distributed shared-memory, and graph computing.
张 旭, 1996 年生. 博士研究生. 主要研究方向为池化内存, 分布式共享内存系统、图计算.

Chen Mingyu, born in 1972. PhD, professor. His main research interests include memory architecture, hardware security method of processor, hardware/software optimization on network protocol of data-center.
陈明宇, 1972 年生. 博士, 研究员. 主要研究方向为计算机访存体系结构、处理器硬件安全控制机制、数据中心服务器网络协议栈软硬件优化技术.

Bao Yungang, born in 1980. PhD, professor. His main research interests include data-center architecture, agile design methodology of processor chips, and ecosystem of open-source processor chips.

包云岗, 1980 年生. 博士, 研究员. 主要研究方向为数据中心体系结构、处理器芯片敏捷设计方法论、开源处理器芯片生态.

Zhang Ke, born in 1982. PhD, professor. His main research interests include computer architecture, heterogenous acceleration, and FPGA cloud.

张　科, 1982 年生. 博士, 正高级工程师. 主要研究方向为计算机体系结构、异构加速、FPGA 云.