中国科学: 信息科学 2018 年 第 48 卷 第 3 期: 315–328
SCIENTIA SINICA Informationis
论文

新一代软件定义体系结构

吕平*, 刘勤让, 邬江兴, 陈鸿昶, 沈剑良

国家数字交换系统工程技术研究中心, 郑州 450002
* 通信作者. E-mail: lp@ndsc.com.cn

收稿日期: 2017–10–24; 接受日期: 2018–02–08; 网络出版日期: 2018–03–16
国家科技重大专项核高基项目 (批准号: 2016ZX01012101)、国家核高基重大专项核高基项目 (批准号: 2017ZX01030301) 和国家高技术研究发展计划 (863) 重点课题 (批准号: 2009AA012201) 资助项目

摘要 体系结构在信息系统中起着举足轻重的作用, 不仅决定了系统的功能和性能, 而且决定了系统的效能和安全, 因此, 体系结构直接决定着信息系统的先进性. 本文在介绍体系结构概念的基础上, 对体系结构的发展进行了归纳总结, 分析指出体系结构刚性是当前信息系统灵活性和高效能无法兼备的问题本质, 提出了以软件定义互连和软件定义节点为特征的新一代软件定义体系结构, 基于软件定义体系结构实现了 Web 服务、口令字恢复和图像识别 3 种典型系统, 对比测试表明, 软件定义体系结构系统较传统通用系统性能提升 29.4~344.5 倍, 效能提升 13.7~315.4 倍, 证明了软件定义体系结构的高灵活性和高效能.

关键词 信息系统, 软件定义互连, 软件定义节点, 软件定义体系结构

1 引言

体系结构即包括一组部件以及部件之间的联系, 在 1964 年由 Amdahl [1] 首次提出, 最早用于描述计算机系统, 并为计算机系统的设计与开发奠定了良好的基础. 40 多年来, 体系结构学科得到了长足的发展, 其内涵和外延得到了极大的丰富, 被广泛用于软件系统、硬件系统、信息系统、信息栅格、科学技术体系等领域. 当前, 体系结构是人工复杂系统的核心范畴, 体系结构在系统中起着举足轻重的作用, 它不仅决定了系统的功能与性能, 也同时直接决定着系统的效能与安全, 体系结构的优劣直接决定着信息系统的基因好坏, 体系结构围绕高速、高效、灵活、安全的目标不断地向前发展与演进. 构建一个先进的体系结构也成为开发复杂信息系统的关键, 它既是一门艺术, 也是一门科学. 构建体系结构的理论基础是系统科学; 科学方法是系统工程. 随着复杂系统体系结构研究的深入, 已经诞生了一些专门的体系结构学, 如科学体系结构学 [2]、C4ISR 系统体系结构学 [3] 等.

按照 “许氏循环” 对半导体特征的预测, 从 2018~2028 年, 半导体将会重新走向通用. 可重构计算技术的发展, 最终将推动主流应用进入 U-SoC 通用波动 [4]. 通过对原始芯片的配置编程就可以得

引用格式: 吕平, 刘勤让, 邬江兴, 等. 新一代软件定义体系结构. 中国科学: 信息科学, 2018, 48: 315–328, doi: 10.1360/N112017-00204
Lv P, Liu Q R, Wu J X, et al. New generation software-defined architecture (in Chinese). Sci Sin Inform, 2018, 48: 315–328, doi: 10.1360/N112017-00204

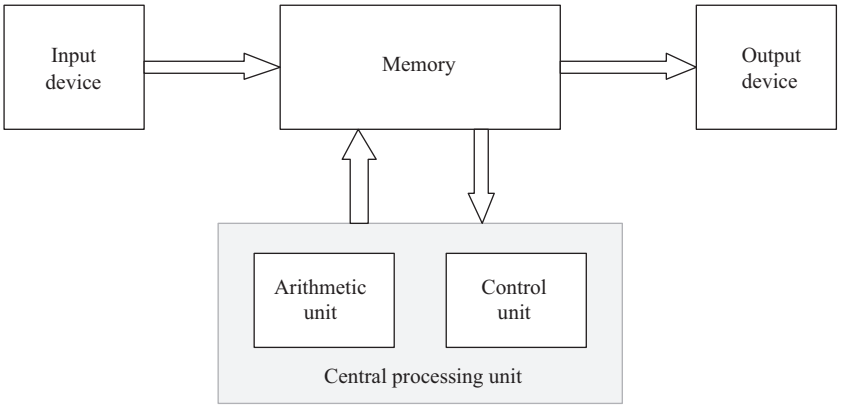


图 1　处理器为中心体系结构

Figure 1　Processor-centric architecture

到用户自定义的功能电路, 从而引导半导体产业结构演变, 最终促进芯片应用创新无设计模式的兴起, 使得 “硬”、“软” 均可编程, 即算法可编程、可重构器件也可编程的 U-SoC.

本文从体系结构的发展脉络出发, 在分析现有体系结构问题的基础上, 基于半导体特征的可编程发展趋势, 提出了软件定义体系结构, 相比现有的体系结构, 软件定义体系结构中的处理节点和互连都支持软件在线定义. 基于该体系结构构建的信息系统, 处理节点的功能可定义、性能可分配, 处理节点之间的互连拓扑可动态变化, 节点之间的处理流程可灵活重构, 从而支持信息系统体系结构适配应用需求的在线定义, 是高灵活性与高效能兼备支持的开放式 “函数化” 体系结构; 最后基于软件定义体系结构技术思想, 在 FPGA 平台上实现了 Web 服务、口令字恢复和图像识别 3 种典型应用系统, 通过与通用服务器系统的性能和效能测试对比, 验证了软件定义体系结构在灵活性和能效方面的优势.

2　体系结构发展历程及面临问题

2.1　体系结构发展历程

按照体系结构, 本文将计算机与信息系统体系结构归纳为处理器为中心、存储器为中心和总线为中心 3 代发展历程.

第 1 代. 以处理器为中心的体系结构 [5], 如图 1 所示. 该体系结构中, 通常包含单个处理器, 且处理器是体系结构的中心, 处理器包括运算器和控制器, 输入/输出设备与存储器间的数据传送都要经过处理器进行控制与传输.

第 2 代. 以存储器为中心的体系结构 [6], 如图 2 所示. 该体系结构中, 通常包括多个处理器, 为了解决处理器与存储器之间的速度矛盾, 设计了包括寄存器、高速缓存、主存储器和外部存储器等多级存储结构.

第 3 代. 以总线为中心的体系结构 [7], 如图 3 所示. 该体系结构通常包括多个处理机和多个存储系统, 借助总线进行复杂任务的分布式协同处理, 总线既是指令的汇聚中心, 也是数据的汇聚中心, 是系统的核心. 总线又经历了早期的共享总线, 到后来的交换总线两个阶段, 核心就是为了解决不同端点系统 (包括处理系统、IO 系统和存储系统等) 之间的大规模并发互连需求.

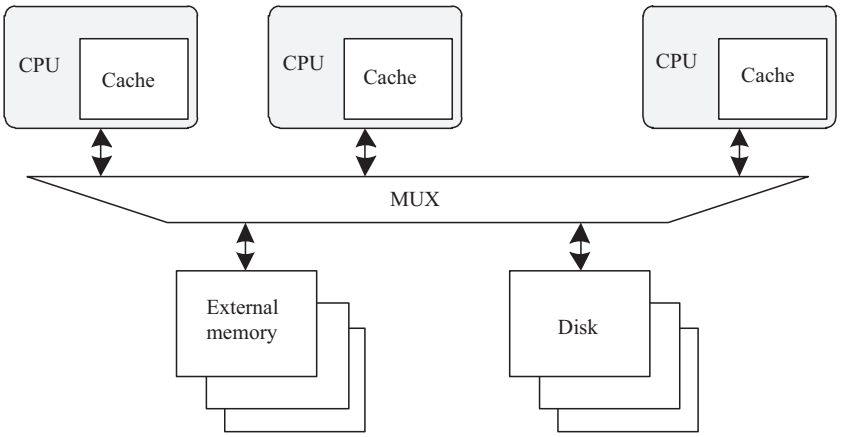


图 2 存储器为中心体系结构

Figure 2 Memory-centric architecture

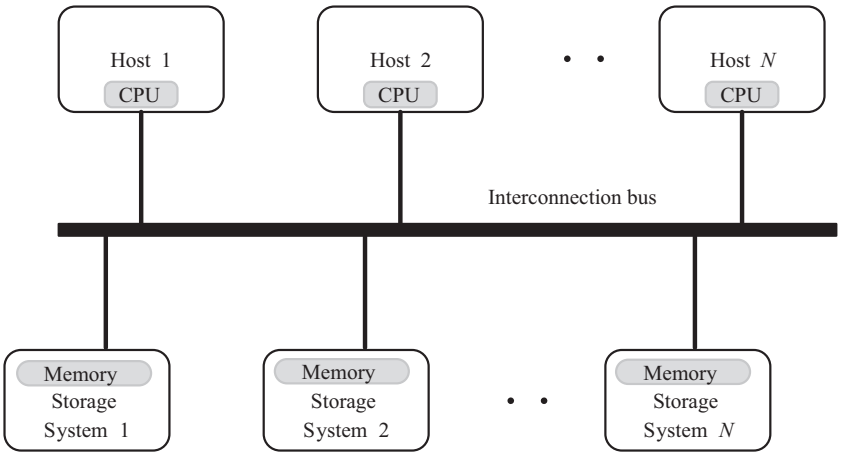


图 3 总线为中心体系结构

Figure 3 Bus-centric architecture

2.2 现有体系结构的问题分析

现有以总线为中心的体系结构, 在支持多处理机之间的分布式协同, 组建规模更大的系统方面起到了重要的作用, 尤其以交换总线为中心的体系结构, 已成为当前信息系统的主流. 交换互连在芯片内、芯片间、处理部件间、子系统间、系统间和网络中广泛采用. 但信息系统朝着多业务一体化承载、大众服务智能化个性提供、新业务快速灵活部署生成等方向发展, 尤其云计算和数据中心的快速发展, 信息系统对体系结构的捷变性和动态适应性提出了更加苛刻的要求. 一方面以超级计算机为核心的信息基础设施, 为满足这种要求, 如 CRAY X1E、BLUE GENE/L、天河二号、太湖之光等超级计算机都普遍引入异构处理和多种互连来实现体系结构的多样化; 另一方面以通用服务器为核心的基础设施, 为了应对这种应用需求, 诞生了各种各样的适配和虚拟技术, 软件定义被认为是最有发展前途的技术. 当前, 诞生了软件定义网络 [8]、软件定义存储 [9]、软件定义计算 [10], 乃至以上述 3 项软件定义技术为支撑的软件定义数据中心 [11] 和软件定义信息基础设施 [12]. 软件定义的核心是通过构件化与标准化, 借助软件虚拟化, 实现处理功能和流程的可定义, 实现信息系统灵活性、开放性和扩展性的提升.

但现有的软件定义技术存在如下问题.

一是现有的软件定义都是基于商业标准的刚性硬件结构, 通过软件的虚拟化来实现, 虽然灵活性、综合性得到较大幅度的提升, 但过度虚拟化反而使得系统效率和性能常常不能满足使用要求, 尤其是在高数据率、高传输带宽、高实时性、低时延等应用场合更难以发挥优势, 存在层层虚拟引入的效率低下、性能受损、实时性下降等突出问题. 每引入一层虚拟化, 性能损失在 10% 左右, 层层虚拟之后, 计算、存储与网络资源的利用效率大大降低, 这也是当前数据中心存在的突出问题 [13].

二是当前所有的软件定义技术, 都是在协议互连层之上进行的软件定义, 如网络层的软件定义转发行为、传输层的软件定义安全策略等, 对协议互连无法支持软件定义, 基本上还是一个刚性互连架构, 具体表现在: 现有互连都是基于特定协议, 针对不同协议之间的互连, 还需要特定桥接技术才能完成; 现有互连端口的带宽分配是固定不变的, 造成了系统组网的不灵活和扩容的不方便; 现有互连的内部交换拓扑结构也是固定的, 通常依据特定的通信模型构建, 无法适用更多的应用场景; 现有互连的内部工作模式通常也是固定的, 要么电路交换, 要么分组交换, 无法在多种工作模式之间灵活切换等. 所以, 现有总线互连在对系统性能的灵活扩容、新协议的开放支持、系统运行的在线优化等均缺少良好的支持.

三是现有处理节点基本上都是刚性的, 尽管随着处理技术的发展, 除了 CPU 在信息系统中广泛采用, 面向信号处理的 DSP、面向图像处理的 GPU、面向网络报文处理的 NPU, 以及各种专用 ASIC 和可编程的 FPGA 等不断用到信息系统之中, 将现有的信息系统演变为一个典型的异构、分布、并行、复杂信息系统. 但系统一旦建成, 各个处理节点也就随之固化, 其硬件的处理功能和处理性能也就固定下来, 只能根据应用需求进行粗颗粒度调度, 如果本处理节点支持的业务没有处理任务, 该处理节点也只能闲置, 无法复用到其他任务, 从而导致硬件资源的浪费. 缺少一个可以统一描述、统一定义的支持软件定义的处理节点技术, 能够在硬件层面通过软件定义来支持不同的细粒度处理功能, 如通过软件定义可以让硬件系统演化为 Web 服务器、加解密处理器、图像识别系统等.

3 新一代软件定义体系结构

基于本文对总线为中心体系结构存在问题的分析, 同时遵循 “许氏循环” 对半导体发展趋势的预判 [4], 本文提出了一种软件定义的新一代体系结构, 其核心思想是从硬件的硅级软件定义入手, 从芯片到软件到系统均支持体系结构的互连可软件定义及处理节点可软件定义, 从而打破现有硬件的刚性体系结构, 将刚性硬件变为柔性硬件, 在保持硬件传统高性能、高效能的同时, 拓展硬件的灵活性. 发展软硬件融合方式的可定义、可重组、可重构、可重建的新一代软件定义体系结构, 如图 4 所示.

该体系结构包括两个核心技术要素: 软件定义互连和软件定义节点. 软件定义互连在硬件层面实现了互连协议、端口类型、拓扑结构、带宽分配、互连模式等关键参数的软件定义, 可以支持不同协议、不同带宽、不同互连模式的数据交互通信需求. 软件定义节点定义了统一的标准、模型与定义流程, 使得处理节点的功能类型能根据软件定义动态可变, 实现了硬件资源支持功能的在线定义.

软件定义体系结构可以根据不同的应用需求动态改变结构形态, 可以以最佳的效能适配应用需求, 大幅度减少软件的虚拟化层次, 从而在系统层面实现高性能、高效能和灵活性的兼备支持.

3.1 软件定义互连

现有信息系统为支持不同应用场合的数据通信需求, 通常采用多种异构互连协议, 如中央处理器

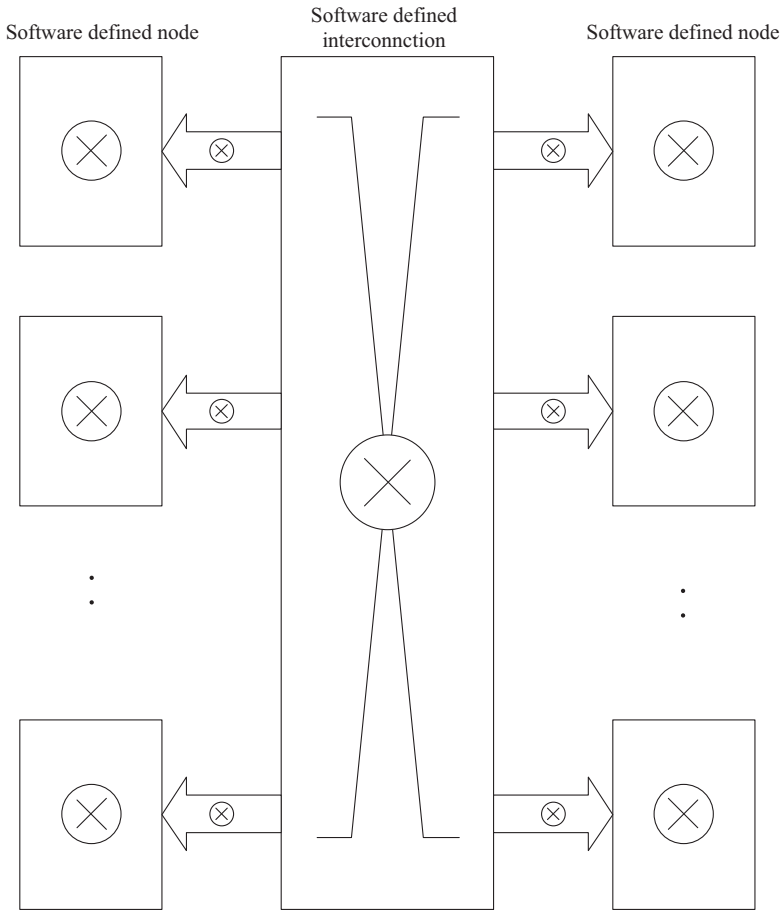


图 4 软件定义体系结构

Figure 4 Software defined architecture

之间通常采用 PCI 和 PCIE 互连协议、信号处理器之间通常采用 RapidIO 互连协议、存储器之间通常采用 FC 协议、内部网络通常采用 Ethernet 协议等. 复杂信息系统或数据中心通常需要多个异构处理系统、存储系统和通信系统协同进行工作, 不可避免地产生不同异构协议之间数据灵活交互的需求, 现在通常采用单协议交换和多种桥接技术来满足异构协议的交互需求, 采用此技术存在两个方面的突出问题: 一是需要采用多款芯片才能实现不同异构协议之间的互连互通, 集成度低, 复杂度高, 功耗大, 不符合绿色节能数据中心和信息系统的发展方向; 二是系统一旦部署完毕, 系统结构也随之固定, 带宽分配也随之固化, 系统的开放性和对新协议的接纳性差, 更无法支持私有协议和用户自定义协议, 灵活性受限. 软件定义互连可以很好解决体系结构中的互连刚性问题.

3.1.1 软件定义互连技术内涵

软件定义互连的技术内涵包括: (1) 互连协议可软件定义: 传统的互连技术只能实现两种同构协议的交换互连, 如 PCIE, Ethernet, Infiniband, RapidIO, FC 等. 软件定义互连可以实现任意软件定义协议之间的交换互连, 其每个互连端口可以定义成标准协议, 如上述的 PCIe, Ethernet, Infiniband, RapidIO, FC 等, 也可以定义成军标协议或用户私有协议. (2) 互连端口可软件定义: 互连端口可软件

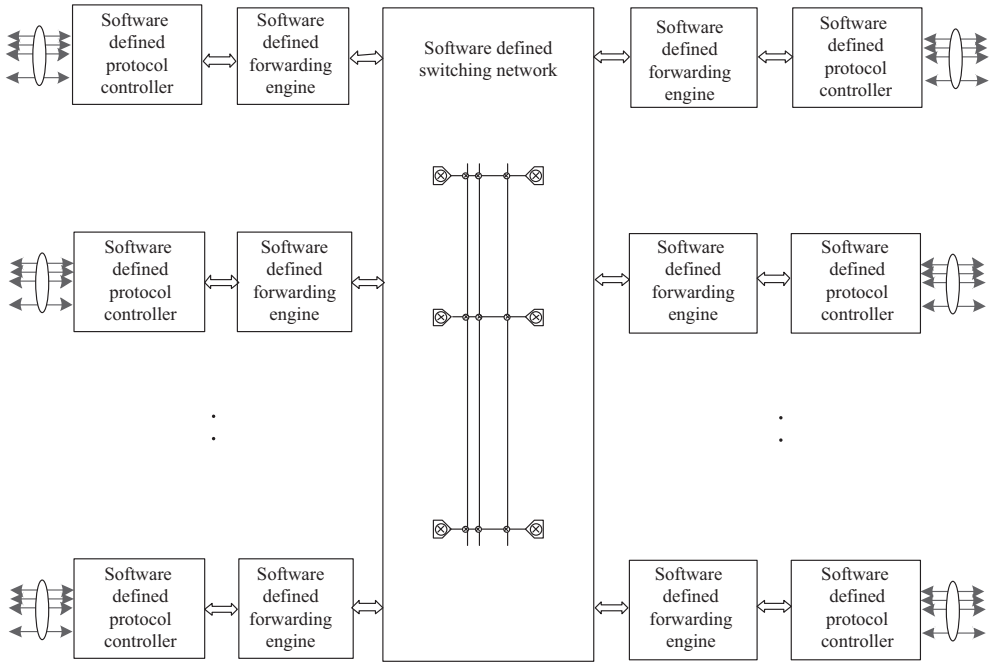


图 5　软件定义互连结构

Figure 5　Architecture of software defined interconnection

定义包括端口类型 (电路端口、分组端口等)、端口绑定模式 (1x、2x、4x、8x 等)、高速串行端口速率 (1.25, 2.125, 2.5, 3.125, 4.25, 5, 6.25, 8.5, 10.3125, 12.5, 25 Gbps 等)、端口流控方式 (发端流控、收端流控等) 都可软件定义. (3) 互连模式可软件定义: 不仅可以支持点到点的单播、单点到多点的组播、单点到所有点的广播, 还可以支持电路交换互连模式、分组交换互连模式, 以及混合交换互连模式等. (4) 互连拓扑可软件定义: 互连拓扑可依据通信业务模型的需求可软件定义为星型拓扑、树形拓扑、环形拓扑 (如带弦环形拓扑、双层环形拓扑等)、基于 Mesh 的拓扑 (支持向多维扩展的立方体和超立方体 Mesh 结构, 如 3D Torus、三角形环网等)、分级互连拓扑、混合拓扑、不规则拓扑等结构. (5) 互连带宽可软件定义: 每个互连端口所支持的速率以及整体互连带宽都可以进行定义, 包括部分利用度、全利用度, 以及不同的调度策略和服务质量等. (6) 协议转换可软件定义: 在各个互连端口上可实现所支持任意异构协议之间的相互转换. (7) 内容处理可软件定义: 支持对数据包进行加扰/解扰、加密/解密、安全监测、语法转换、正则匹配等软件定义的灵活流式处理. (8) 互连安全可软件定义: 支持各个互连端口软件定义的冗余备份、负载均衡和匹配映射, 再加上各个端口协议的受控软件定义跳变, 实现交换互连的结构可变拟态防御.

3.1.2　软件定义互连实现结构

软件定义互连的实现结构如图 5 所示. 通常包括软件定义协议控制器、软件定义转发引擎和软件定义交换结构 3 个部分组成, 软件定义协议控制器可软件配置为支持不同频点、不同绑定模式、不同协议的协议控制器; 软件定义转发引擎可软件配置为不同协议的解析与封装、不同协议的转换、不同协议的转发查表; 软件定义交换结构可软件配置为不同规模、不同结构的交换网络.

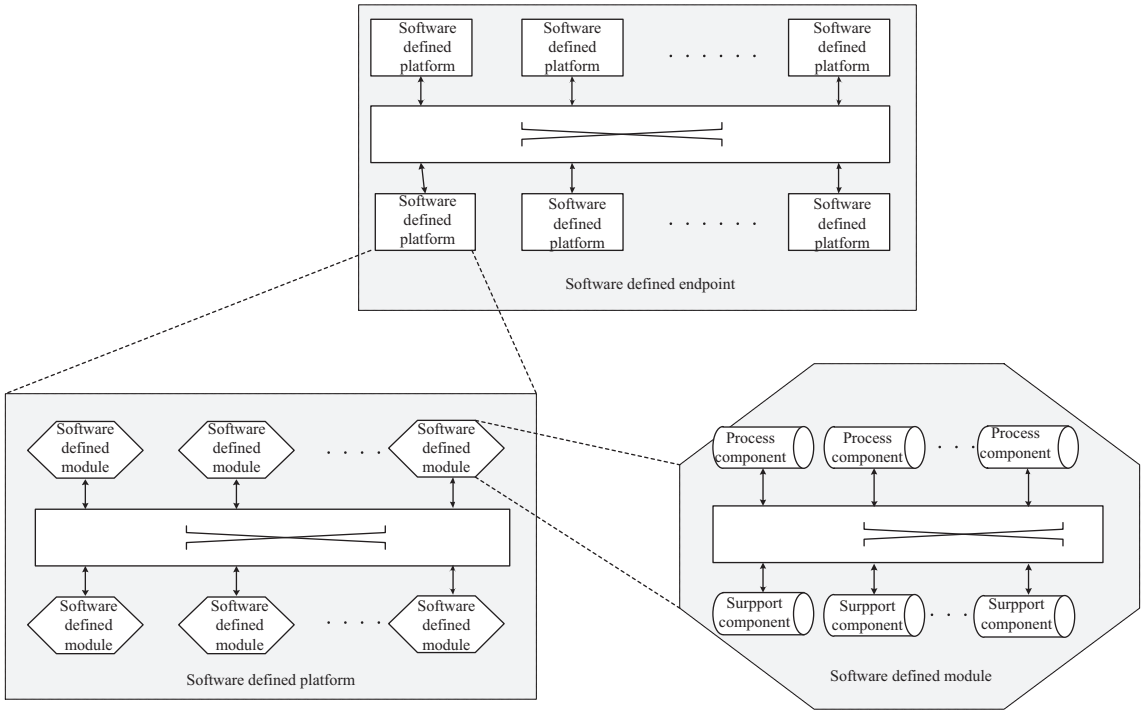


图 6 软件定义节点结构

Figure 6 Architecture of software defined processing-node

3.2 软件定义节点

当前处理器已经进入多核时代, 像 CPU、DSP 和 GPU 等都可以支持不同程度的并行计算. 但是, 这些处理器本质上还是基于大量的指令, 单核采用程序的串行方式实现应用算法, 其执行过程必须经过 “取指令→指令译码→地址生成→取操作数→执行→存储或写回” 6 个步骤, 对于 ASIC 实现, 一般无需前 3 步, 而前 3 步恰恰在处理器中消耗大量的控制逻辑, 执行代价大, 对芯片面积和功耗都有非常大的影响, 因此, 处理器在效率上远落后于 ASIC; 不同于上述指令执行方式, FPGA 作为细粒度的可重构器件, 以 LUT 为基本单元, 实现硬件的位级可编程, 具备良好的通用性, 但是, 布线资源占据高达 80% 的芯片面积, 基于 SRAM 的 LUT 单元同 ASIC 相比需要更多晶体管来执行逻辑功能. 通常 FPGA 实现相同功能电路的面积是 ASIC 的 23∼55 倍, 功耗是 6∼26 倍, 可见 FPGA 同样面临严重的效率问题 [14].

按照 “平台面向应用、组件面向功能、构件面向处理” 的技术思想, 文中提出了支持动态装配和在线定义的软件定义节点技术, 如图 6 所示, 支撑构件代表组成软件定义处理组件的电源、时钟、复位、接口、缓存、管理等基本构件, 处理构件支持混合粒度可编程, 通过装载不同的电路, 可实现不同的处理, 是组成软件定义处理组件的核心部分. 在设计中将接口和引用技术用于构件和组件的封装, 不同的构件通过数据驱动的交换灵活重组成不同功能的组件, 不同的组件通过逐级交换灵活装配不同的平台, 实现节点功能和性能均支持软件定义. 核心技术包括处理功能硬件构件灵活定义、处理结构逐级交换灵活定义、处理流程数据包驱动灵活定义 3 个部分.

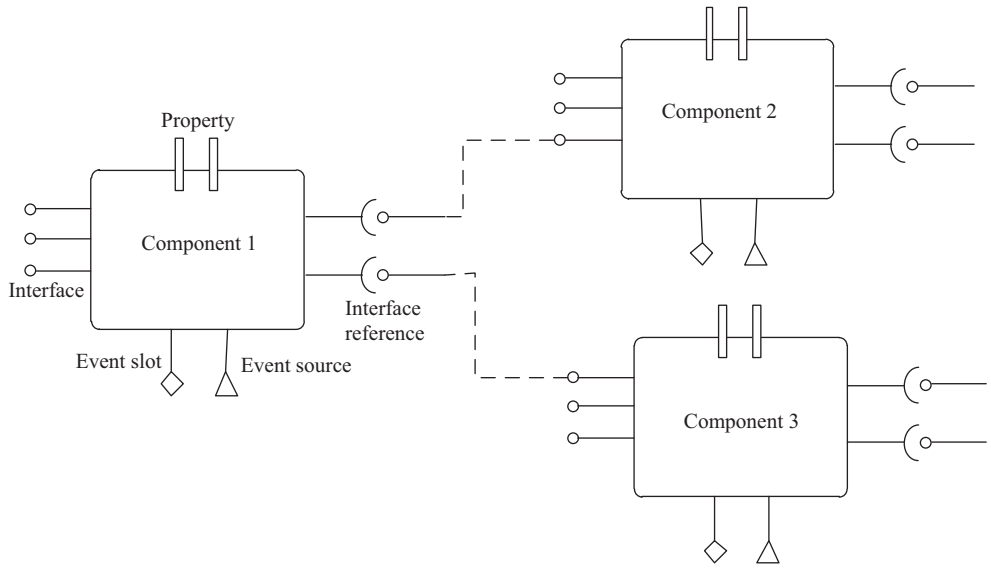


图 7　硬件构件抽象模型

Figure 7　Abstraction model of hardware component

3.2.1　处理功能硬件构件灵活定义

为了支持处理节点功能的软件定义, 本文提出了硬件构件封装定义的概念, 即支持通过基于构件的封装定义出新的构件, 从而支持硬件功能构件的拓展与衍生, 并给出了一种硬件构件抽象模型, 包括提供接口、接口引用、属性、事件源、事件槽 5 个部分, 如图 7 所示.

提供接口. 构件都是具有一定处理能力的功能模块, 并且不止一种功能, 把一个构件功能的集合称为处理集, 而这些处理能力是需要对外提供使用的, 所以把构件能提供的处理能力称之为提供接口, 处理结果将反映到提供接口中.

接口引用. 构件包含提供接口, 就意味着构件也有可能使用来自其他构件的提供接口, 把构件使用其他构件接口的情况称之为接口引用, 体现为在构件中记录引用的其他构件功能的标识. 接口引用主要是在构件组装情况下使用, 当然一个构件可能存在零个或多个接口引用.

属性. 构件在实现过程中需要将一些设置参数化, 以便更好使用运行环境, 而如何针对这些参数进行设定和修改就是构件属性的作用所在. 构件属性反映了构件的特性, 可供外部配置和查询.

事件源. 构件可以针对自身状态的变化来通知其他构件或运行环境, 使其做出正确响应和调整, 比如某个数据被创建、删除或配置更改等.

事件槽. 当检测到目标构件或运行环境发生改变, 构件需要根据情况做出反应.

硬件构件除了进行数据交互之外, 还要满足软件定义要求, 即构件能够进行组装. 构件在组装过程中会存在构件与元构件的关系, 维护并记录这些关系就需要接口引用, 构件引用可以用如下信息进行描述:

$$\text{Reference} := \text{Component\_name::Interface\_name},$$

即用被引用的构件名称加相应的接口名称的形式进行标识.

一个构件可能没有引用其他构件的接口, 但也有可能引用多个接口, 所以构件引用是需要被记录和管理的. 这些信息将会以构件描述文件 (XML 文档) 的形式保存下来. 通过这种标准化的构件模型

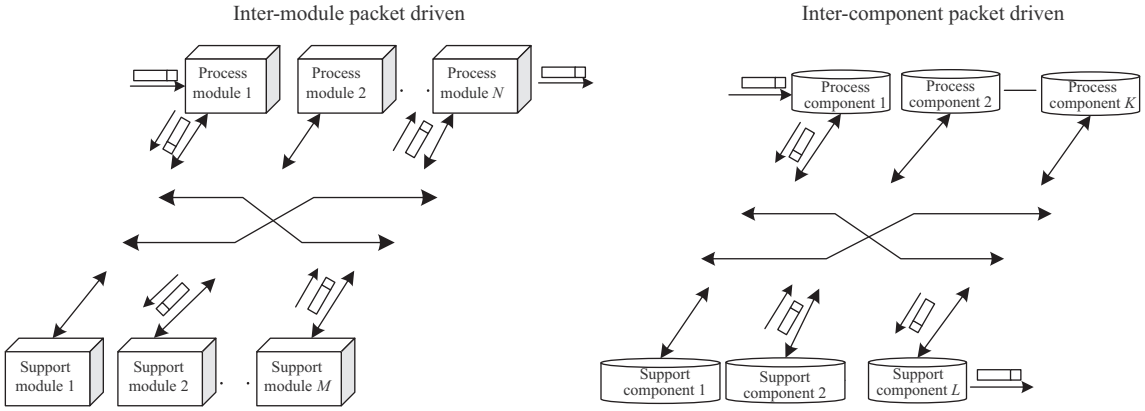


图 8 数据包驱动机制

Figure 8 Packet driven processing mechanism

以及构件引用的软件定义封装, 可以动态生成不同处理功能的硬件构件, 实现硬件处理功能的灵活软件定义.

3.2.2 处理结构逐级交换灵活定义

当前系统设计中模块依靠总线连接, 模块间连接关系不可变, 导致节点只能按照固定模式完成单一任务. 如果要升级或扩展节点功能, 只能采取离线升级和重新设计软硬件模块的方式. 处理流程设计完毕后不能发生改变, 使得节点设备不能灵活、快速地适应网络需求的变化.

为支持处理结构的灵活在线定义, 文中基于软件定义节点的思路, 提出了逐级交换机制的互连拓扑结构, 参照图 6 所示.

逐级交换互连拓扑结构中的逐级指的是将平台分为组件间、构件间和构件内部 3 个不同级别的视图; 交换指的是组件或构件间不再使用传统的固定线路连接方式, 使用交换带有标识的报文来满足组件或构件间的消息交互需求. 标识内容的多寡影响了交换控制的开销和复杂度, 根据所处视图的不同可根据实际情况确定交换标识的具体内容. 逐级交换带来的好处是用户在遵循标准接口的前提下可以灵活地替换构件, 引入新的功能. 利用交换拓扑可变来更改构件间的通信连接, 用户可专注构件的开发, 而不必受到通信对象的限制. 同样只有在可变连接拓扑的支持下, 才能发挥构件组合的优势. 这种机制的优势在于, 连接在交换网络上的构件和组件可以实现替换, 这样可以改变传统的数据处理流程难以改变的问题, 通过改变处理流程导致节点功能变化.

3.2.3 处理流程数据包驱动灵活定义

当前处理平台中单元模块间通过数据总线连接, 模块间的控制和数据交互通过电气信号驱动, 导致信息交互的语义由电信号表征, 一旦信号含义确定, 后续使用中将不能改变, 无法引入新的数据或控制语义, 不能支持节点功能的改变. 在软件定义节点中, 文中引入数据包驱动的处理流程控制机制, 即组件和构件间交互使用统一的物理接口和统一的数据格式, 通过数据报头体现处理要求. 当数据包到达后, 组件或构件根据报头语义进行相应的处理, 完成后更新报头内容传递给下一组件或构件进行后续处理. 这种机制的优势在于设计人员可以通过定义报头格式引入新的控制语义, 以适应构件、组件的替换, 支撑节点功能的改变, 如图 8 所示.

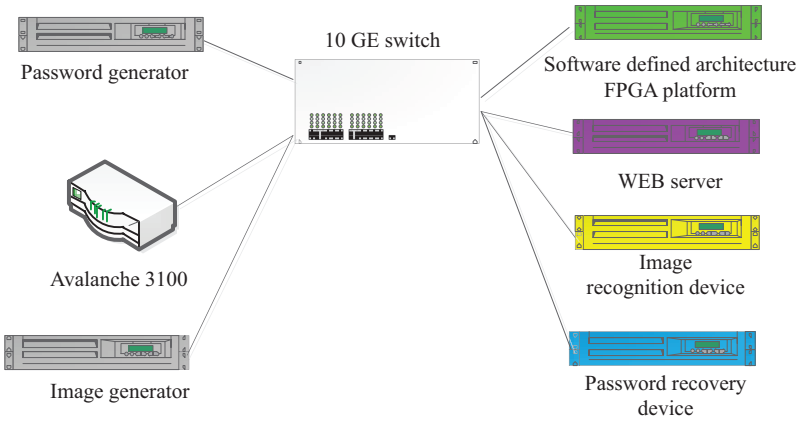


图 9　(网络版彩图) 对比测试场景

Figure 9　(Color online) Contrast test scenario

软件定义节点的特点是应用驱动、非指令执行, 且高度可扩展. 应用驱动是指通过对应用算法的分析, 提取可共用的基本操作单元 (可称为操作集, 区别于传统处理器的指令集) 并依据其硬件实现形成基础算核集, 在此基础上构建粗细粒度混合的可重构单核结构; 非指令执行是指将应用算法的硬件实现形式直接映射到结构中, 即单核内的粗细粒度单元以硬件可重构方式实现应用算法中的算核, 明显区别于以存储程序为基础的通用处理器结构, 这样就消除了传统处理器的取指令和指令译码的开销, 进而可大幅提高执行效率; 而高度可扩展是指核间和核内都采用以软件定义互连为基础的逐级互连架构, 可实现以单核为基础的高度可扩展.

与通用处理器采用周而复始的取指、取数、运算、存储中间结果的处理方式不同, 软件定义节点的处理过程是: 重构高阶运算栈、初始化、运算处理、存放结果. 软件定义节点用硬件构成的有限状态自动机代替指令, 减少了内存访问数量, 同时高阶运算栈提高了硬件利用率, 具有很高的处理性能. 基础算核通过嵌套软件定义互连, 可以构成支持应用算法各子模块的高阶运算栈, 相对通用处理器, 计算过程在高阶运算栈上的执行效率更高, 这些计算过程最终又支撑了各种应用的核心算法.

4　测试对比

为了验证本文提出软件定义体系结构构建信息系统的灵活性和高能效, 与传统通用处理器结构组成的系统进行了性能和能效对比, 对比测试场景如图 9 所示. 基于 4 片大规模 FPGA 芯片, XILINX Virtex-7 1140T 开发了软件定义体系结构的 FPGA 验证平台, FPGA 两两之间采用 4 路高速通道进行全互连, 平台包含 12 条 16 G DDR3 内存, 对外提供 4 路 SFP+ 光接口、8 路 QSFP+ 光接口、4 路 PCIE、4 路 SATA 接口. 通过软件定义, 平台可分别定义成 Web 服务器、口令恢复服务器和图像识别服务器, 与基于 IBM X3850 X6 搭建的 Web 服务器、口令破解服务器和图像识别服务器进行效能对比, Avalanche 3100 测试仪、口令字测试数据产生设备、图像测试数据产生设备与软件定义体系结构 FPGA 平台、通用 Web 服务器、图像识别设备、口令字恢复设备通过万兆交换机 H3C S5800 相连, 对比测试同等流量情况下 3 种业务在软件定义体系结构 FPGA 平台和其他设备上的性能和效能对比, 其中效能提升倍数定义为 P_SDA/P_CPU.

3 种典型应用场景的对比测试如表 1 所示.

表 1 3 种典型应用的性能/能效对比指标

Table 1 Comparison of performance/efficiency of three typical application scenarios

Typical application	Evaluating indicator	Performance	Efficiency
Web service	Number of successful transaction requests per second	Transaction/s (tps)	tps/w
Password cracking	Number of verifying password per second	Password/s (pps)	pps/w
Image recognition	Number of detecting video frames per second	Frames/s (fps)	fps/w

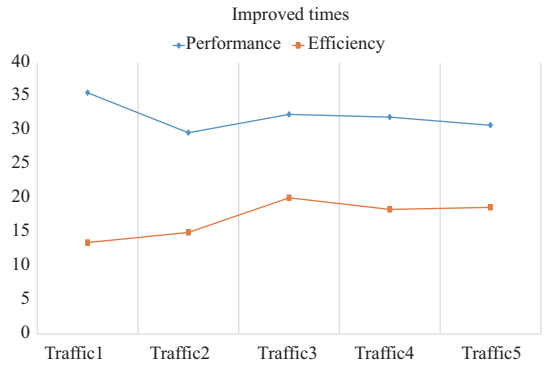


图 10 (网络版彩图) Web 服务器性能和能效对比

Figure 10 (Color online) Comparison of performance/efficiency of Web server

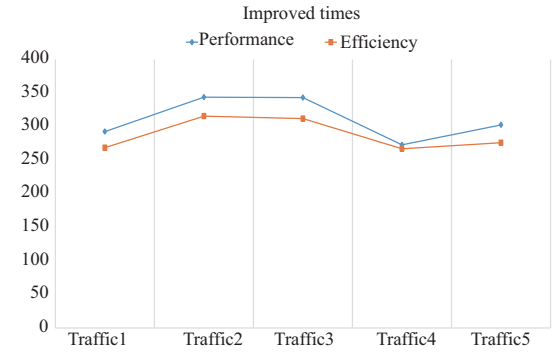


图 11 (网络版彩图) 口令破解性能和效能对比

Figure 11 (Color online) Comparison of performance/efficiency of password cracking

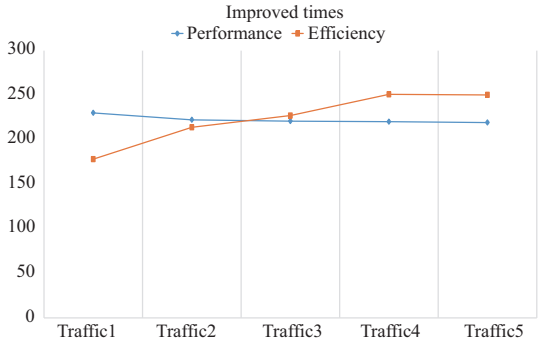


图 12 (网络版彩图) 图像识别性能和效能对比

Figure 12 (Color online) Comparison of performance/efficiency of image recognition

在不同输入负载速率下, 对 3 种典型应用场景的软件定义体系结构 FPGA 平台和通用设备性能和效能进行对比测试, 测试结果如图 10~12 所示.

测试结果表明, 软件定义体系结构 FPGA 平台的 Web 服务器相对通用服务器的性能提升倍数在 29.4 到 35.6 之间, 效能提升倍数在 13.7 到 20.3 之间; 口令破解性能提升倍数在 273.2 到 344.5 之间, 效能提升倍数在 267.5 到 315.4 之间; 图像识别性能提升倍数在 219.3 到 230.10 之间, 效能提升倍数在 178.2 到 250.9 之间.

由于软件定义体系结构的 FPGA 平台所支持的 3 种典型应用是在统一硬件平台上通过软件定义实现, 证明了该体系结构系统的强大灵活性, 同时 3 种典型应用场景的性能和效能测试对比又证明了

该体系结构系统的高效能.

5 结论

软件定义体系结构在处理和互连层面均实现了体系结构的软件定义, 是系统结构适应应用 [15] 的基础, 按照 “结构决定功能、结构决定性能、结构决定效能”, 这种体系结构在灵活性和效能方面呈现了天然的优势.

在实现层面, 软件定义体系结构涉及到硬件软件定义、软件可重构、主动业务感知、最佳匹配映射、资源调度管理等系列关键技术. 得益于硬件尤其硅可编程技术的发展与日趋成熟, 使得软件定义芯片逻辑成为可能, 从而打通了软件定义体系结构的最后一个环节, 使得系统体系结构能够按照应用需求的多样性进行动态变化, 达到灵活性与效能的兼顾. 在软件定义节点和软件定义互连的实现中, 都需要首先对系统的工作模式、业务流程进行分析, 以高效能为目标, 以多业务灵活支持为原则, 科学抽取节点处理功能的异构算粒、互连功能的异构算粒, 并优化内部互连拓扑, 开发软件定义的软硬件支撑与应用环境, 实现系列化软件定义处理芯片和软件定义互连芯片. 相比 FPGA, 软件定义芯片由于在算粒层面, 而非比特层面实现可编程, 大大节省了布线资源, 精简了系统设计, 优化了系统性能, 但其灵活性不如 FPGA 普适, 比较适用面向领域的应用; 相比 CPU, 由于软件定义芯片采用基于配置流的工作模式, 精简了 CPU 操作的固定 6 步骤操作过程, 实现了效能的大幅提升, 但灵活性不如 CPU. 软件定义芯片的前提是要有丰富和良好的基础算粒库支撑, 基础算粒的抽取是软件定义芯片的关键, 也是核心技术门槛, 由于不同领域应用的基础算粒差异较大, 软件定义芯片会从规模化服务提供、多样化业务支持的领域取得突破, 通过研制面向领域的软件定义芯片, 推出开发和应用所需的软件硬件工具和支撑环境, 就可以实现在这些领域软件定义系统的搭建, 解决效能和灵活性无法兼顾的难题, 继而逐渐扩展到其他领域, 实现多领域通用.

因此, 软件定义体系结构具有灵活性和高效能同时兼备的技术优势, 代表着体系结构的发展方向, 未来复杂信息系统将伴随系列软件定义处理芯片和软件定义互连芯片的面世步入软件定义时代.

参考文献

1 Amdahl G M. The structure of SYSTEM/360, part III: processing unit design considerations. IBM Syst J, 1964, 3: 144–164

2 Dong Z. Discussion on the structure of science system. Sci Sci Manag ST, 1992, 11: 5–8 [董忠. 对科学学体系结构的探讨. 科学学与科学技术管理, 1992, 11: 5–8]

3 Luo A M, Huang L, Luo X S. Study on architechtural description and design of C4ISR. Fire Control Command Control, 2005, 30: 25–28 [罗爱民, 黄力, 罗雪山. C4ISR 体系结构描述和设计方法. 火力与指挥控制, 2005, 30: 25–28]

4 Xu J Y. Analysis of major developments in microelectronics. J Chinese Acad Electron Sci, 2006, 1: 215–218 [许居衍. 微电子重大发展态势分析. 中国电子科学研究院学报, 2006, 1: 215–218]

5 Liu B W, Chen S M, Wang D. Survey on advance microprocessor architecture and its development trends. Appl Res Comput, 2007, 24: 16–20 [刘必慰, 陈书明, 汪东. 先进微处理器体系结构及其发展趋势. 计算机应用研究, 2007, 24: 16–20]

6 Wang J. Architecture analysis of computer cache memory. Aeronaut Comput Technique, 2006, 36: 29–33 [王珏. 计算机高速缓冲存储器体系结构分析. 航空计算技术, 2006, 36: 29–33]

7 Zhao Z S. The bus-plugin styled architecture method research. Dissertation for Master's Degree. Chongqing: Chongqing University, 2001 [赵泽松. 总线 – 插件式体系结构方法研究. 硕士学位论文. 重庆: 重庆大学, 2001]

8 Azodolmolky S. Software Defined Networking. Beijing: China Machine Press, 2014 [Azodolmolky S. 软件定义网络. 北京: 机械工业出版社, 2014]

9 Ye Y R. Software Defined Storage. Beijing: China Machine Press, 2016 [叶毓睿. 软件定义存储. 北京: 机械工业出版社, 2016]
10 Zhang X L, Zhang D, Cao L L, et al. Cloud computing virtualization platform performance research. Softw Guide, 2013, 12: 1–3 [张新玲, 张东, 曹玲玲, 等. 云计算虚拟化平台性能研究. 软件导刊, 2013, 12: 1–3]
11 Chen X, Ricky S. Software Defined Data Center. Beijing: China Machine Press, 2015 [陈熹, Ricky S. 软件定义数据中心. 北京: 机械工业出版社, 2015]
12 Wang X, Chen M, Chao H U, et al. SDICN: a software defined deployable framework of information centric networking. China Commun, 2016, 13: 53–65
13 Huber N, Quast M V, Hauck M, et al. Evaluating and modeling virtualization performance overhead for cloud environments. In: Proceedings of the International Conference on Cloud Computing and Services Science, Noordwijkerhout, 2011. 563–573
14 Wu J X, Luo X G, Cao W, et al. A reconfigurable computing array and construction method. CN 102799563 B. 2015 [邬江兴, 罗兴国, 曹伟, 等. 一种可重构计算阵列及构建方法. CN 102799563 B. 2015]
15 Wu J X, Luo X G, Si X M. The "stochastic strain" mimic computing village. Sci Technol Overview, 2014, 5: 54–58 [邬江兴, 罗兴国, 斯雪明. "随机应变" 的拟态计算村. 科技纵览, 2014, 5: 54–58]

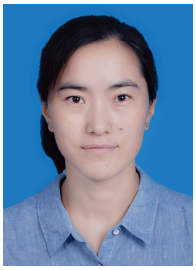
New generation software-defined architecture

Ping LV*, Qinrang LIU, Jiangxing WU, Hongchang CHEN & Jianliang SHEN

National Digital Switching System Engineering and Technological Research Center, Zhengzhou 450002*, China*
* Corresponding author. E-mail: lp@ndsc.com.cn

Abstract Architecture plays an important role in information systems. It not only determines the function and performance of a system, but the system efficiency and security as well. Therefore, the architecture directly determines the advanced nature of an information system. On the basis of introducing the concept of architecture, this paper summarizes the development of architecture, and points out that the rigidity of architecture is the essential reason behind why current information systems cannot tradeoff between flexibility and high efficiency. A new generation of software-defined architecture, which is characterized by software-defined interconnections and software-defined nodes, is proposed. Three typical systems, Web service, password recovery and image recognition, are realized based on a software-defined architecture. Compared with the traditional general systems, the comparison tests show that the performance of the software-defined architecture system increased by 29.4 to 344.5 times, and the efficiency increased by 13.7 to 315.4 times, which demonstrates the high flexibility and high efficiency of software-defined architecture.

Keywords information system, software defined interconnection, software defined node, software defined architecture



Ping LV was born in 1977. She received her Master's degree at the National Digital Switching System Engineering & Technological Research Center (NDSC) in 2007. Currently, she is pursuing her Ph.D. degree at NDSC. Her research interests include new architectures and new generations of information communication technology.

Qinrang LIU was born in 1975. He received his Ph.D. degree at the National Digital Switching System Engineering & Technological Research Center (NDSC) in 2004, Currently, he is a professor and Ph.D. supervisor at NDSC. His research interests include new architectures and cyberspace security.

Jiangxing WU was born in 1953. He is an academician of the Chinese Academy of Engineering and a Ph.D. supervisor at the National Digital Switching System Engineering & Technological Research Center (NDSC). His current research interests are information & communication technology and cyberspace security.

Hongchang CHEN was born in 1964. He is a professor and Ph.D. supervisor at the National Digital Switching System Engineering & Technological Research Center (NDSC). His current research interests are big data analysis and new generations of information communication technology.