有意思的论文FPGA Catapult（P3）

Original 2018-04-30 Accela Zhao Accela推箱子

（续前文：
有意思的论文FPGA Catapult - P2，
有意思的论文FPGA Catapult - P1）

FGPA的云虚拟化

软件模拟虚拟化如今已逐渐被Intel CPU等硬件辅助虚拟化代替。CPU、内存的虚拟化由Intel VT-x支持。IO/PCIe虚拟化有Intel VT-d，硬件外设如网卡则提供SR-IOV（PCIe上一个Root虚拟地插入多个外设；SR-IOV不仅仅适用于网卡）。GPU虚拟化有Intel GVT-g等支持；链接中还有AMD、NVIDIA各自的方法。可以看到，实用的虚拟化支持最终都落到CPU等硬件上。

[NVIDIA, AMD, and Intel: How they do their GPU virtualization](https://www.brianmadden.com/opinion/NVIDIA-AMD-and-Intel-How-they-do-their-GPU-virtualization)

	AMD MxGPU	Intel GVT-g	NVIDIA vGPU
Video RAM	Physical slice+	Physical slice	Physical slice
Shader engines	Physical slice	Time slice	Time slice*
Video decode	No	Time slice	Time slice*
Video encode	No	Time slice	Time slice*
GPU Compute	OpenCL	OpenCL	Passthru only
Hypervisor requirements	SR-IOV	SW Manager	SW Manager
Hypervisors supported	ESX	KVM/Xen	ESX/Xen

(source: Randy Groves, "Virtualized GPUs are now an Option for VDI and DaaS!" BriForum Boston 2016)

关于FPGA的虚拟化，并没看到成熟一致的方法。第一种，较易理解的，是基于FPGA的**Partial-reconfiguration（PR）**。通过PR将FPGA划分为多个区域，可独立互不影响地重新编程，对用户呈现为虚拟的小vFPGA，资源池化。完整的系统还需配套FPGA分配资源管理、Hypervisor支持、FPGA板上驱动、客户API等。下面所列的前两篇论文是同一系列作者发表；第一篇更详细地讲解了FPGA虚拟化，第二篇则着重构建完整的云框架。第三篇论文展示如何在Openstack上构建FPGA虚拟化。

[Virtualized FPGA Accelerators for Efficient Cloud Computing](https://warwick.ac.uk/fac/sci/eng/staff/saf/publications/cloudcom2015-fahmy.pdf)

[Virtualized Execution Runtime for FPGA Accelerators in theCloud](http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7840018)

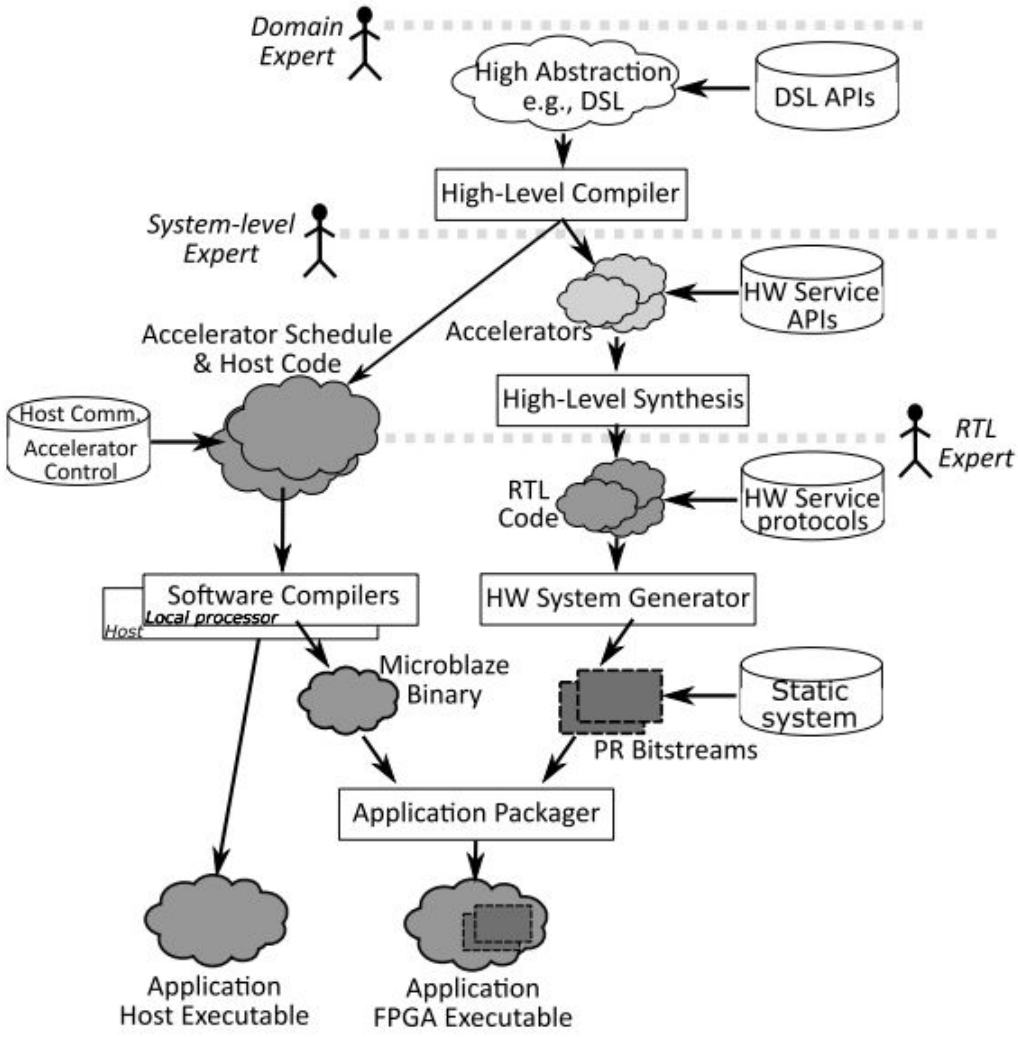


FIGURE 5. Design flow for our methodology. We define three entry points, depending on the user's expertise, which eventually converge to the same type of output.

[Enabling FPGAs in the Cloud](http://nics.ee.tsinghua.edu.cn/people/wangyu/Enabling%20FPGAs%20in%20the%20Cloud.pdf)

第二种方法则是在FPGA板上构建**虚拟的CPU核**。虚拟核可以在不同FPGA中迁移，可以缩放占用板上面积，构造处理能力大小不同的核。用户程序向这些核提交任务，有资源管理器管理核的

Placement以及任务的调度，从而实现虚拟化。但在FPGA上虚拟CPU核，似乎浪费了FPGA能够跳出CPU架构处理任务的优势。

[A platform-independent runtime methodology for mappingmultiple applications onto FPGAs through resource virtualization](https://pdfs.semanticscholar.org/3a7a/53a5530aa5b07e9e051dfd0433f9928eecca.pdf)

[Overlay Architectures For FPGA Resource Virtualization](https://hal.archives-ouvertes.fr/hal-01405912/document)

[FPGA Virtualization for Enabling General PurposeReconfigurable Computing](http://cc.doc.ic.ac.uk/fresh16/Dirk.pdf)

[An Efficient FPGA Overlay for Portable Custom InstructionSet Extensions](https://pdfs.semanticscholar.org/2d04/22cc490db8b2069134f27bde85c9458af8e8.pdf)

第三种方法称为**Overlay架构**。不同FPGA供应商提供的基础编程元件不同，导致FPGA程序难以移植。类似于Java虚拟机，Overlay架构在FPGA物理单元上构建出一致的虚拟编程单元及架构；其上的程序可自由移植。Overlay架构比较大的问题是浪费物理编程单元，一个虚拟LUT可能需要40个物理LUT来实现，即PV-ratio为40x（physical LUT : virtual LUT ratio）。

[Overlay Architectures For FPGA Resource Virtualization](https://hal.archives-ouvertes.fr/hal-01405912/document)

[ZUMA: An Open FPGA Overlay Architecture](http://www1.cse.wustl.edu/~roger/565M.f12/4699a093.pdf)

[FPGA Virtualization for Enabling General Purpose Reconfigurable Computing](http://cc.doc.ic.ac.uk/fresh16/Dirk.pdf)

[An Efficient FPGA Overlay for Portable Custom Instruction Set Extensions](https://pdfs.semanticscholar.org/2d04/22cc490db8b2069134f27bde85c9458af8e8.pdf)

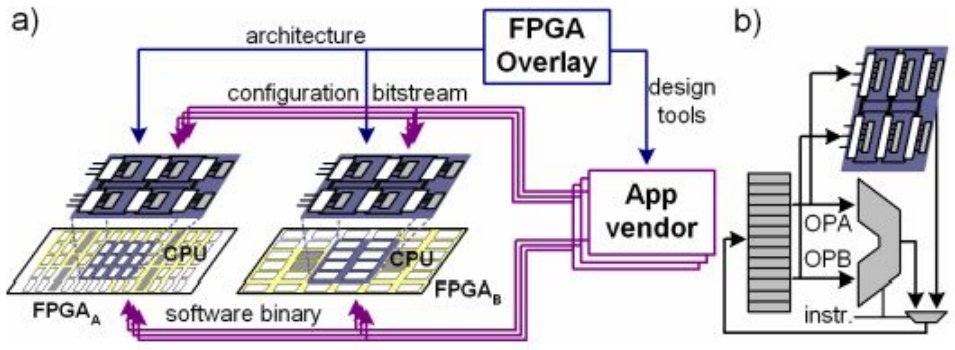


Fig. 1. a) FPGA specific overlay implementations allow posting of custom instruction bitstreams. b) Attaching a custom instruction overlay to a CPU

其它一些有意思的论文或项目，例如，用类似Boot虚拟机的方法，Boot新FPGA。FPGA的虚拟化仍然基于Partial-reconfiguration。

[FPGAs in the Cloud: Booting Virtualized HardwareAccelerators with OpenStack](https://ieeexplore.ieee.org/document/6861604/)

Openstack子项目Cyborg，用于管理各种加速硬件服务，例如GPU、FPGA等。

[OpenStack Acceleration Service: Introduction of Cyborg Project](https://www.openstack.org/videos/boston-2017/openstack-acceleration-service-introduction-of-cyborg-project)

用于HPC的，将多个物理FPGA卡聚合成更大的虚拟FPGA的方法。通过API Remoting访问远程FPGA，虚拟如本地访问一样。

[High Performance in the Cloud with FPGA Groups](http://www.globule.org/publi/HPCFV_ucc2016.pdf)

更进一步，以虚拟化为基础，FPGA有与之对应的、类似Hadoop/YARN的、标准成熟的**云级处理调度框架**吗？就我目前看到，更多的是各大厂商自研自用的框架，或者与Hadoop、Openstack的整合，而公有云一般提供的是独享的、基于PCI-Passthrough的FPGA虚拟机使用场景。另外，用于机器学习/深度学习的FPGA框架也有一些。

FPGA的应用场景

首先，比较普通的是，FPGA易用于用户**加密、压缩、编码**等任务固定、计算密集型场景。FPGA常常作为现存系统中的一个子模块，解决一些独立的问题，对已有系统架构冲击不大。

其次，相比CPU，FPGA能够高并行、低延迟、高能效、低成本地处理较为简单的任务。非常合适的使用场景就是**网络虚拟化**，如Azure SDN。前文已经看到，在网络带宽飞速增长的今天，CPU难以消化全部吞吐量，而FPGA能取得数量级的性能提升。另一方面，FPGA可安装在网卡附近，数据包无需经过PCIe到CPU处理再返回网卡。相比如今的网络速度，常见的PCIe V2的带宽实在太低。

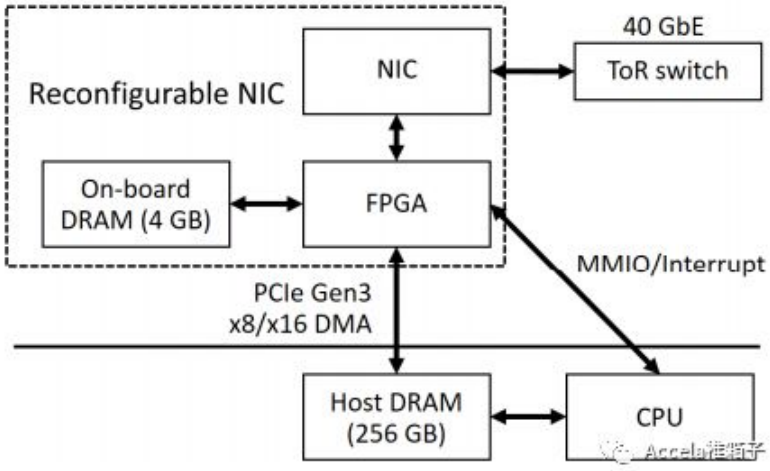
[Wikipedia: PCI Express](https://en.wikipedia.org/wiki/PCI_Express)

PCI Express link performance[30][33]

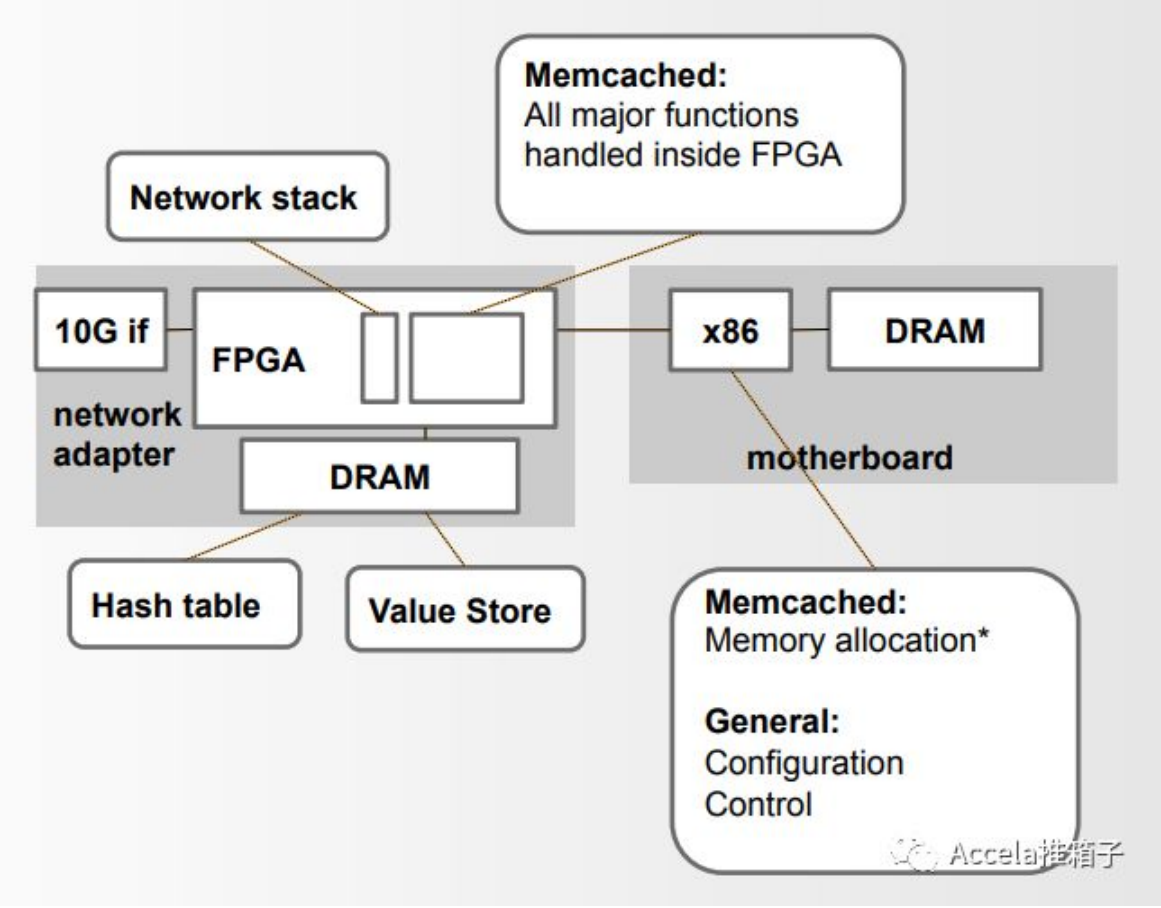
PCI Express version	Introduced	Line code	Transfer rate[i]	Throughput[i]				
				×1	×2	×4	×8	×16
1.0	2003	8b/10b	2.5 GT/s	250 MB/s	0.50 GB/s	1.0 GB/s	2.0 GB/s	4.0 GB/s
2.0	2007	8b/10b	5.0 GT/s	500 MB/s	1.0 GB/s	2.0 GB/s	4.0 GB/s	8.0 GB/s
3.0	2010	128b/130b	8.0 GT/s	984.6 MB/s	1.97 GB/s	3.94 GB/s	7.88 GB/s	15.8 GB/s
4.0	2017	128b/130b	16.0 GT/s	1969 MB/s	3.94 GB/s	7.88 GB/s	15.75 GB/s	31.5 GB/s
5.0[31][32]	*expected in Q2 2019*[34]	128b/130b	32.0 GT/s[ii]	3938 MB/s	7.88 GB/s	15.75 GB/s	31.5 GB/s	63 GB/s

另一个场景是**内存存储与计算**，如FPGAKV-Direct、Memcached。将大部分存储逻辑编程到FPGA上，将FPGA安装到网卡附近，依托高速网络；大部分处理都由FPGA完成、不需经过较慢的PCIe总线，也不需CPU参与。相比普通内存存储，这类系统往往能够提升数量级的吞吐量，更低延迟，同时还更节能。

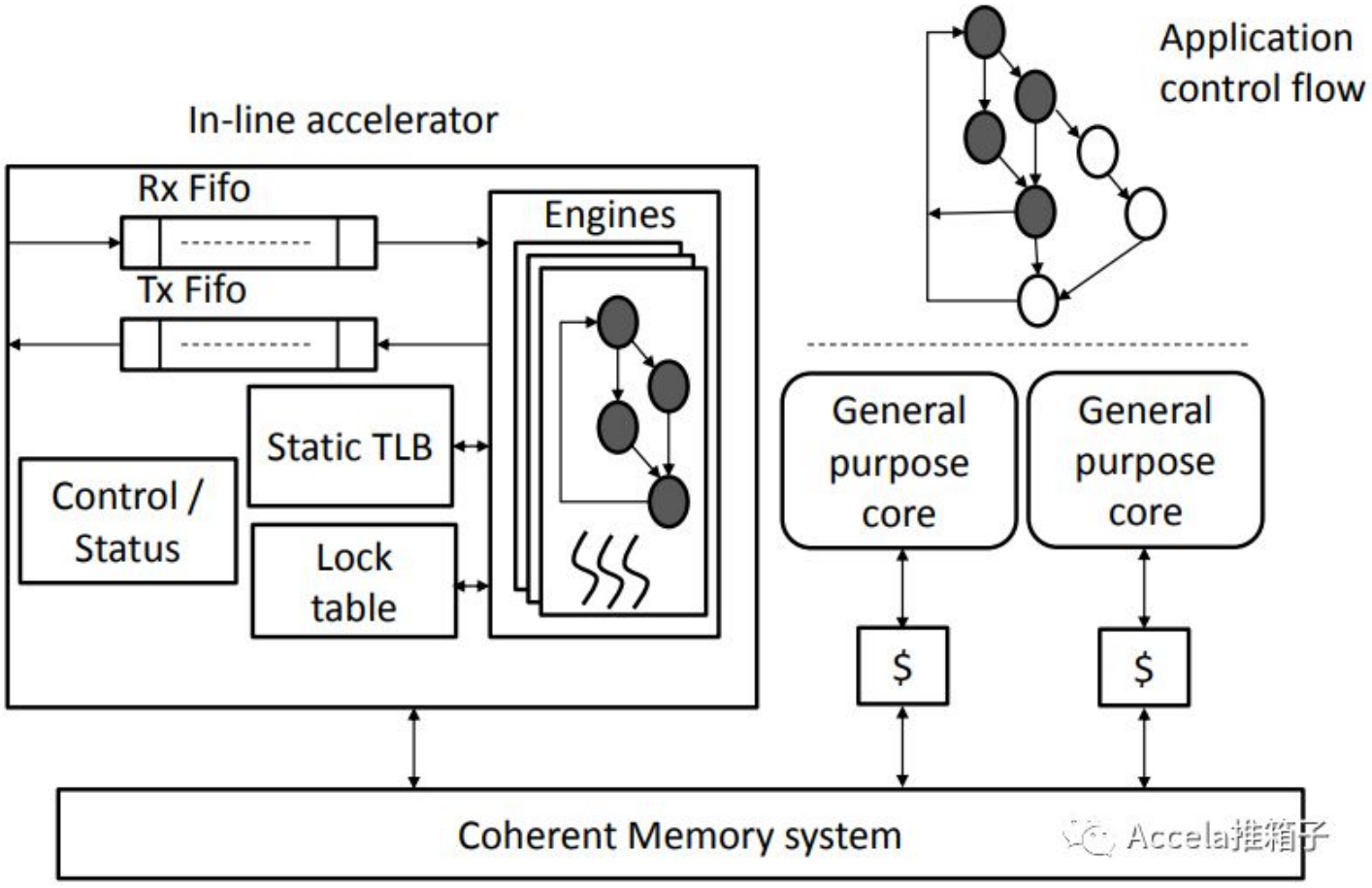
[KV-Direct: High-Performance In-Memory Key-Value Store with Programmable NIC](https://www.microsoft.com/en-us/research/wp-content/uploads/2017/12/kv-direct.pdf)



[Achieving 10 Gbps line-rate key-value stores with FPGAs]
(https://www.usenix.org/system/files/conference/hotcloud13/hotcloud13-blott.pdf)



[An FPGA-based In-line Accelerator for Memcached]
(http://www.cs.princeton.edu/courses/archive/spring16/cos598F/06560058.pdf)



基于磁盘或SSD的存储系统，往往需要复杂的逻辑来处理持久存储的异常、硬件中断等，甚至还需要处理日志（Journaling）。但FPGA更适用于简单而高并发的处理逻辑，同时磁盘/SSD和PCIe的慢速也拖累FPGA的发挥。与FPGA在内存存储取得的显著效果不同，在驱动磁盘/SSD方面似乎使用场景不多。当然，做磁盘控制器是可以的

[FPGA Drive](https://fpgadrive.com/example-designs/)

总之，硬件发展往往是存储系统架构演进的一大动力。例如CPU、内存、网络、持久存储的容量/速度的发展，以及更重要的性价对比平衡的变化。另一方面，则是新式硬件的出现和成熟，例如FPGA、GPU、ASIC等，在合适的场景里，它们能带来完全不同的架构。此外，用户场景的变化也是一大驱动，例如互联网和智慧城市对对象存储的大量需求，例如云计算对块存储的需求，例如欧盟GDPR新规带来的对归档存储（ArchivalStorage）的需求。而如何应对越来越大的数据规模（如Scale-out），同时维持高可靠性，维持成本和访问速度，则是持久的话题。

（全文完。注：本文为个人观点总结，作者工作于微软）

Report