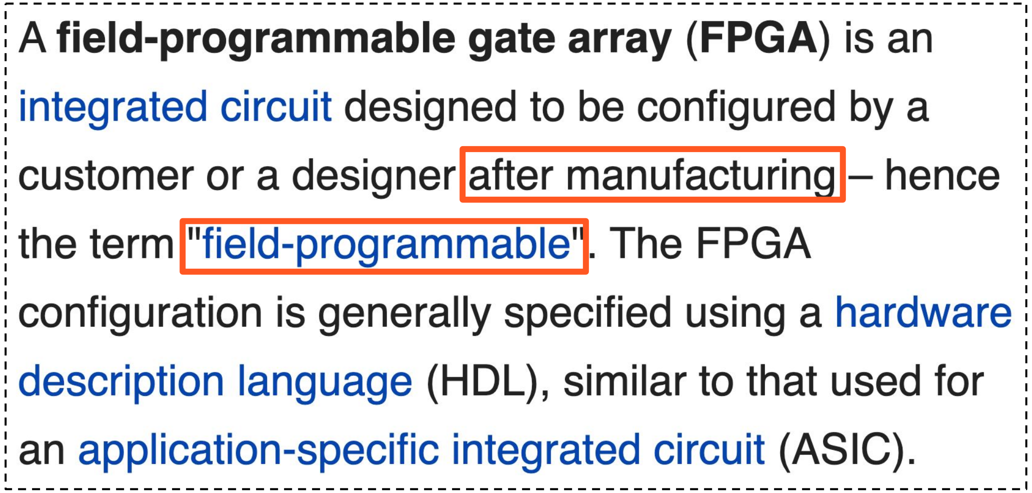
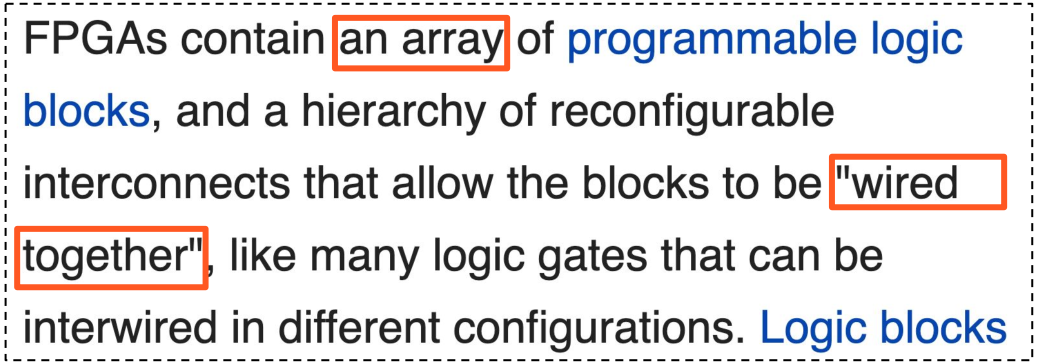
FPGA Virtualization

For CSE291J Virtualization
Yizhou Shan
Feb 27, 2020

What is FPGA?

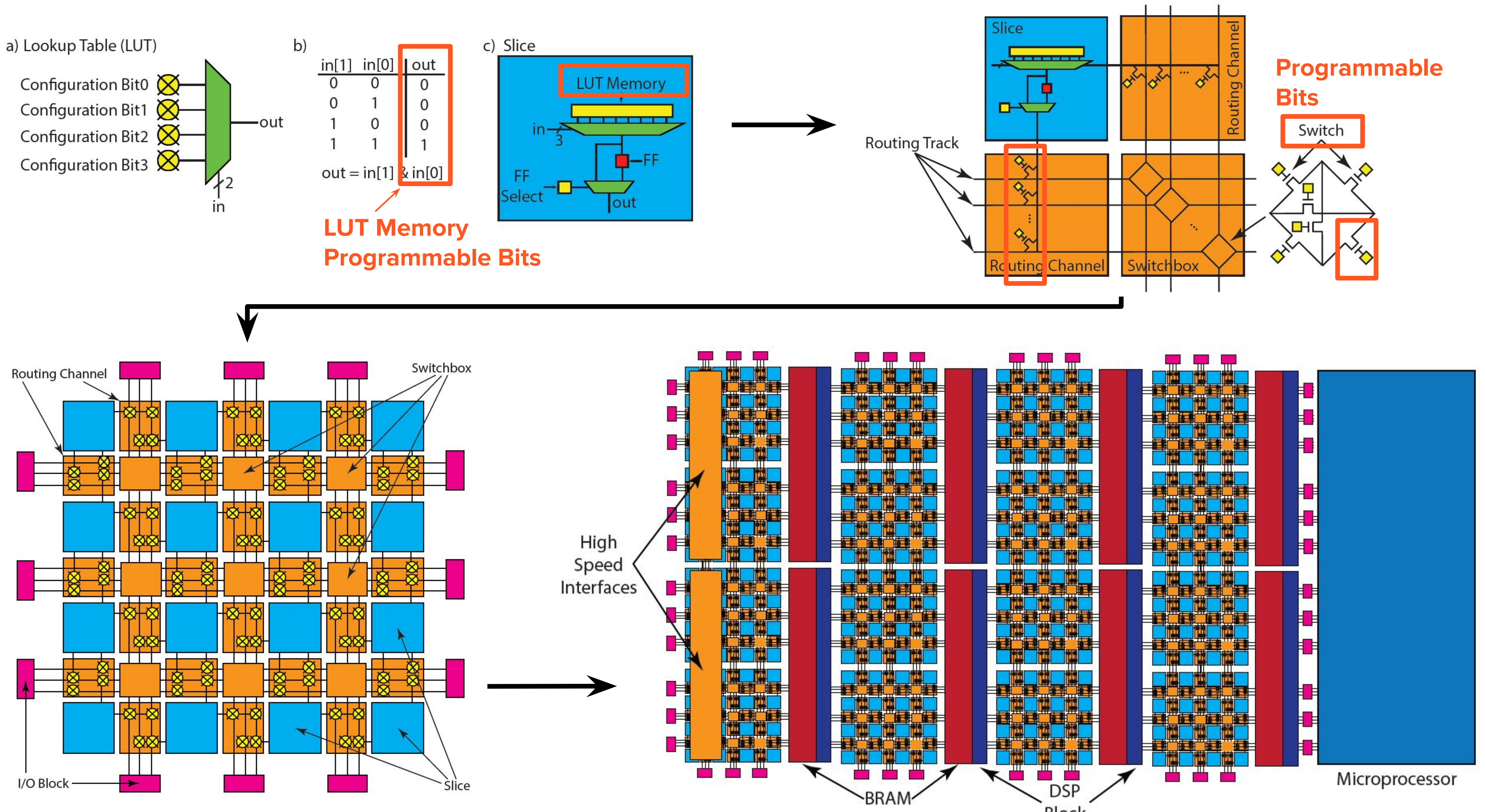


A **field-programmable gate array** (**FPGA**) is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC).



FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be interwired in different configurations. Logic blocks

WIKIPEDIA The Free Encyclopedia



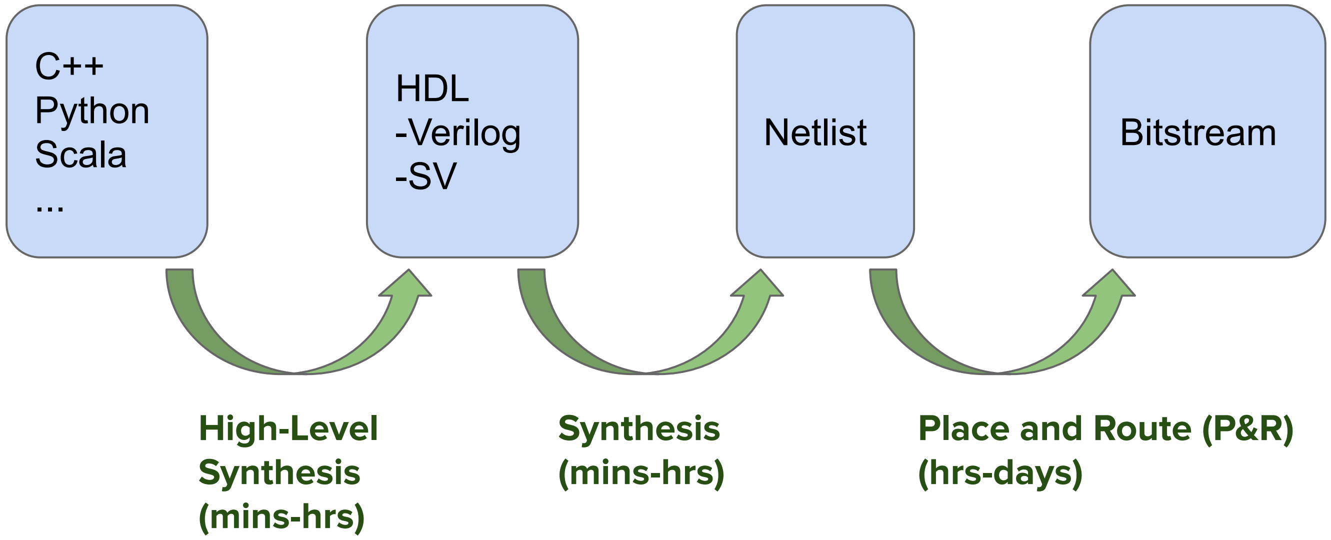
in[1]	in[0]	out
0	0	0
0	1	0
1	0	0
1	1	1

(Images from: Parallel Programming for FPGAs, http://kastner.ucsd.edu/hlsbook/)

Development Process

Optional,
You can start
from HDL

Bitstream is like a binary file!

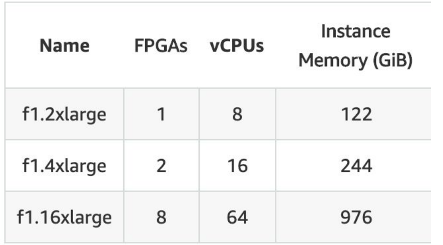


Use FPGA as an accelerator

- Image/Video Processing
- Machine Learning
  - DNN/CNN and more
  - A good alternative to GPUs
- Bio Analysis
- Network Acceleration (e.g., SmartNIC)
- Storage Acceleration (e.g., SmartSSD)
- Graph/KVS and more

You don't really need to understand FPGA in order to use it
Recent language advancement has boosted its recent adoption

Massive Deployment, Cloud FPGA



Name	FPGAs	vCPUs	Instance Memory (GiB)
f1.2xlarge	1	8	122
f1.4xlarge	2	16	244
f1.16xlarge	8	64	976

Microsoft Project Catapult

- Released at 2014 (internal, not public)
- Since then, it has been used to accelerate
  - Bing Search
  - Azure Network
  - Machine Learning

AWS, Alibaba, etc

- Public cloud FPGA
- High-end Xilinx chips
- Large scale, low-cost, and fast dev
- Current model
  - **Single user, no sharing**

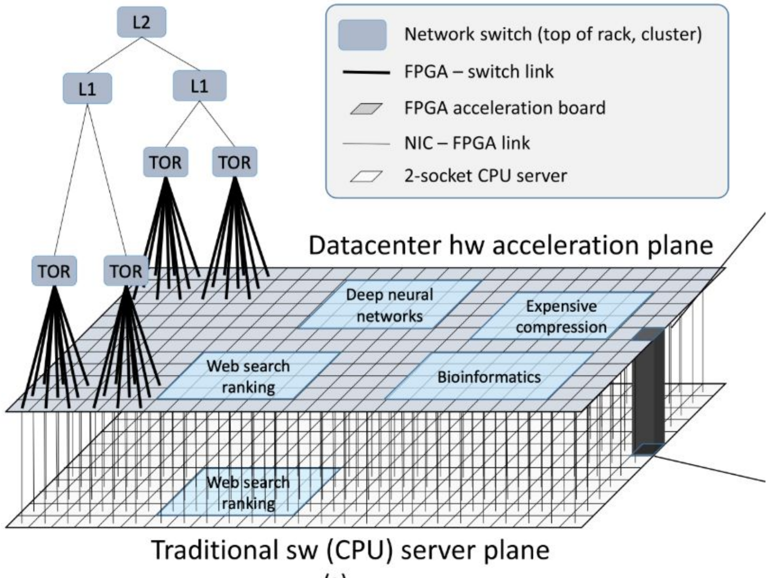
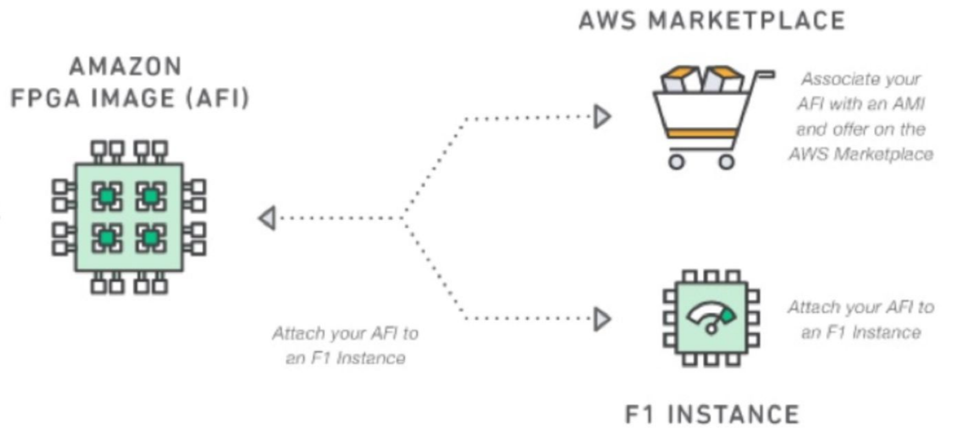


Image from Catapult, ISCA'14



AWS MARKETPLACE
AMAZON FPGA IMAGE (AFI)
Associate your AFI with an AMI and offer on the AWS Marketplace
Attach your AFI to an F1 Instance
Attach your AFI to an F1 Instance
F1 INSTANCE

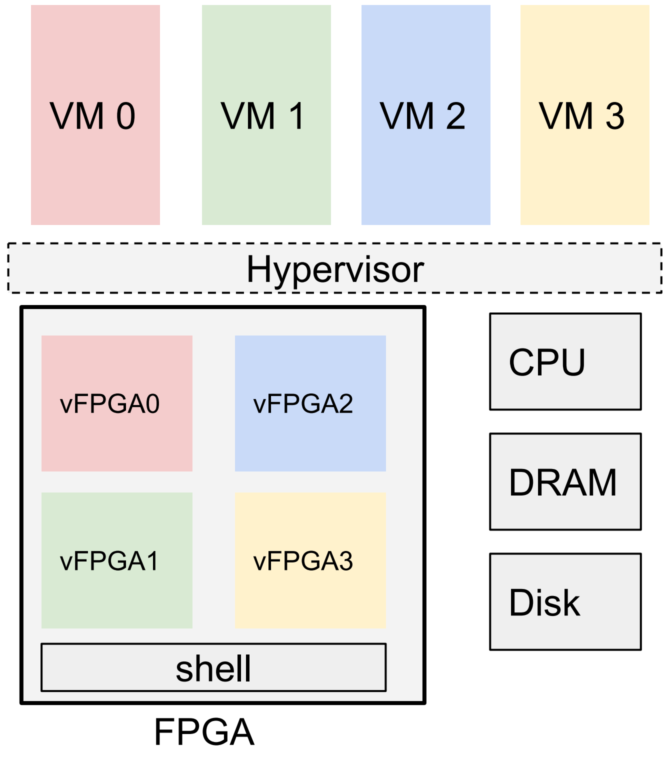
Discussion: Is FPGA the future?

- Microsoft is betting big on FPGA
- Google and Amazon are leaning towards ASIC
- Can FPGA take over GPU or Google TPU or ASICin the future?

Compared to FPGA:

	Cost	**Energy**	**Dev Velocity**	**Performance**	**Programmability**
GPU	Higher	Higher	Slower	Depends	Lower
ASIC Google TPU	Lower	Lower	Slower	Depends	Lower
ASIC Amazon Nitro	Lower	Lower	Slower	Depends	Lower

Towards sharing cloud FPGAs



Reasons for sharing
- Customer: pay-as-you-go
- Vendor: consolidation

Strawman solution:
- vFPGA on top of a physical FPGA

Key technique: **Partial Reconfiguration (PR)**
- Change a part of a running FPGA design, e.g., update `vFPGA0`, without disturbing others
- Limitation: fixed slots ➜ Fragmentation
    - Resizing needs to reprogram the whole chip

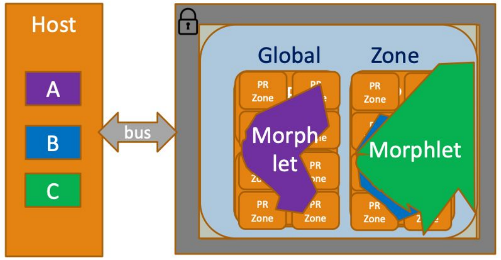
But sharing needs more:
- Protection
- Elasticity
- Compatibility

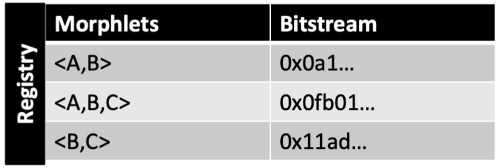
AmorphOS: enable cloud FPGA sharing

- High-level goals
  - Protection among untrusted FPGA applications
  - Dynamic Scaling, or Elasticity
  - Compatibility across vendors

AmorphOS

- A framework to efficiently share cloud FPGAs among untrusted users
  - A set of APIs
  - A way to partition the chip → ***Zone***
  - A way to scale/package FPGA apps → ***Morphlet***
  - A way to mix FPGA apps → ***High-throughput/Low-latency mode***
  - A way to protect resource from untrusted FPGA applications → ***Hull***
  - Finally a way to deploy mixed apps onto protected and partitioned chip → ***Registry***





Registry	Morphlets	Bitstream
	<A,B>	0x0a1…
	<A,B,C>	0x0fb01…
	<B,C>	0x11ad…

Zone and Morphlet

A set of APIs
A way to partition the chip → ***Zone***
A way to scale FPGA apps → ***Morphlet***
A way to mix FPGA apps → ***High-throughput/Low-latency mode***
A way to protect resource from untrusted FPGA applications → ***Hull***
Finally a way to deploy mixed apps onto protected and partitioned chip → ***Registry***

- Partition the chip into a multi-level zones
  - Global zone for the whole chip
  - Then smaller sub-zones
- Morphlet
  - An instance of a user FPGA bitstream (like a container, and scalable)
  - It can morph, i.e., dynamically change resource requirements
    - How? E.g., change the array size `N` for `int buf[N]`.

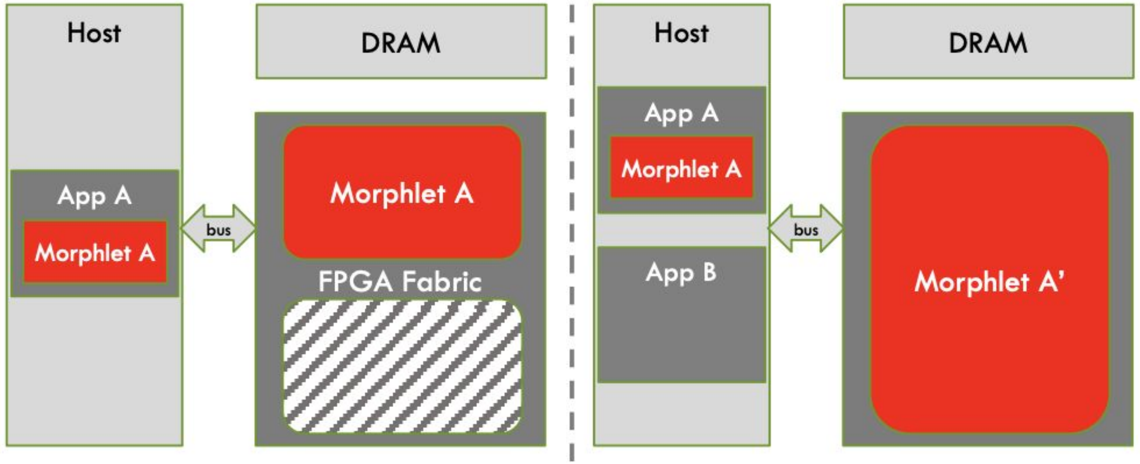


Image from AmorphOS, OSDI'18

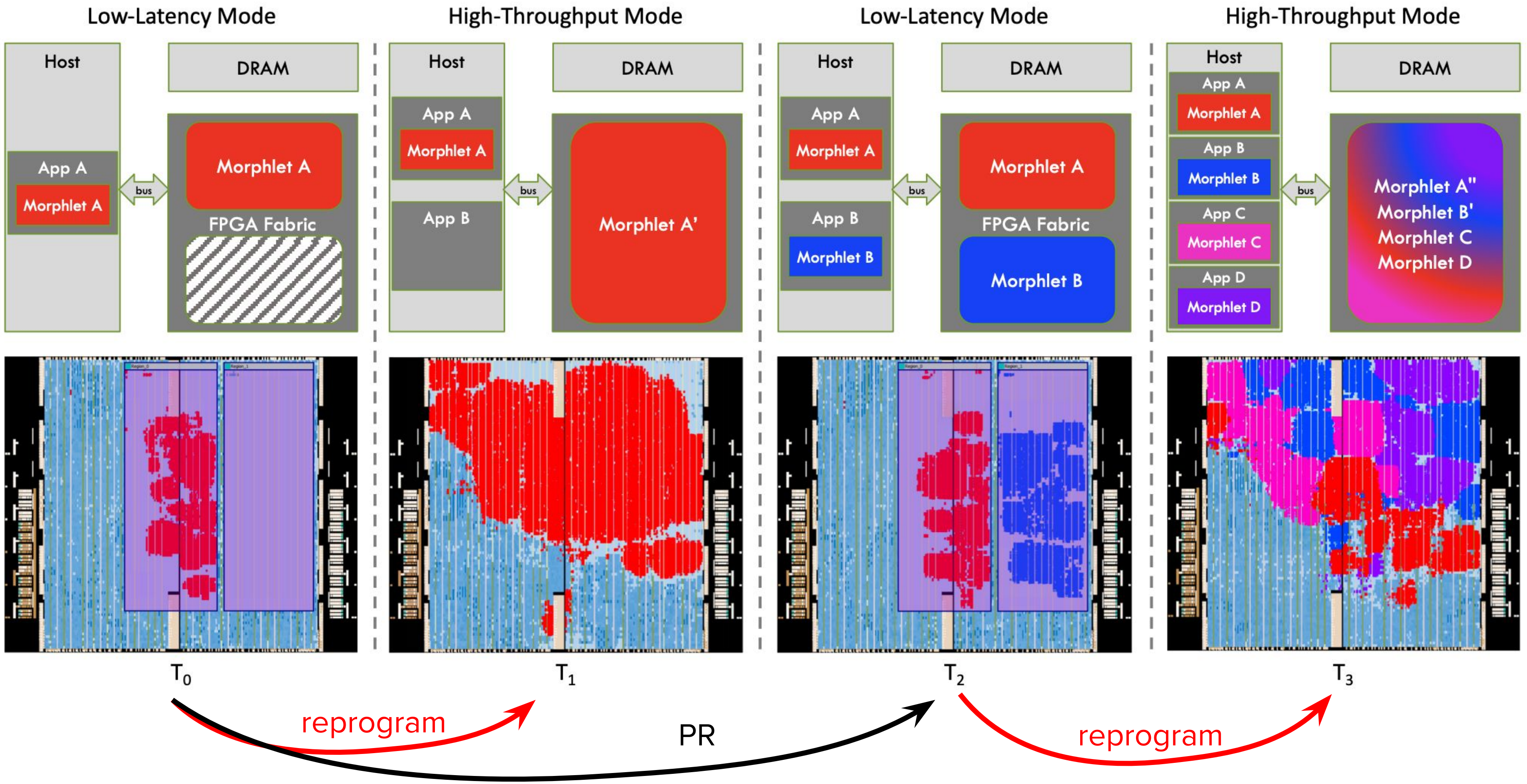
Scheduling Morphlets

A set of APIs
A way to partition the chip → ***Zone***
A way to scale FPGA apps → ***Morphlet***
A way to mix FPGA apps → ***High-throughput/Low-latency mode***
A way to protect resource from untrusted FPGA applications → ***Hull***
Finally a way to deploy mixed apps onto protected and partitioned chip → ***Registry***

- Low Latency Mode
  - Fixed zones + PR
  - Default Morphlet bitstream
- High Throughput Mode
  - Combine multiple Morphlets
  - Co-schedule on a global zone

Low latency: switching is fast
High throughput: more areas are used

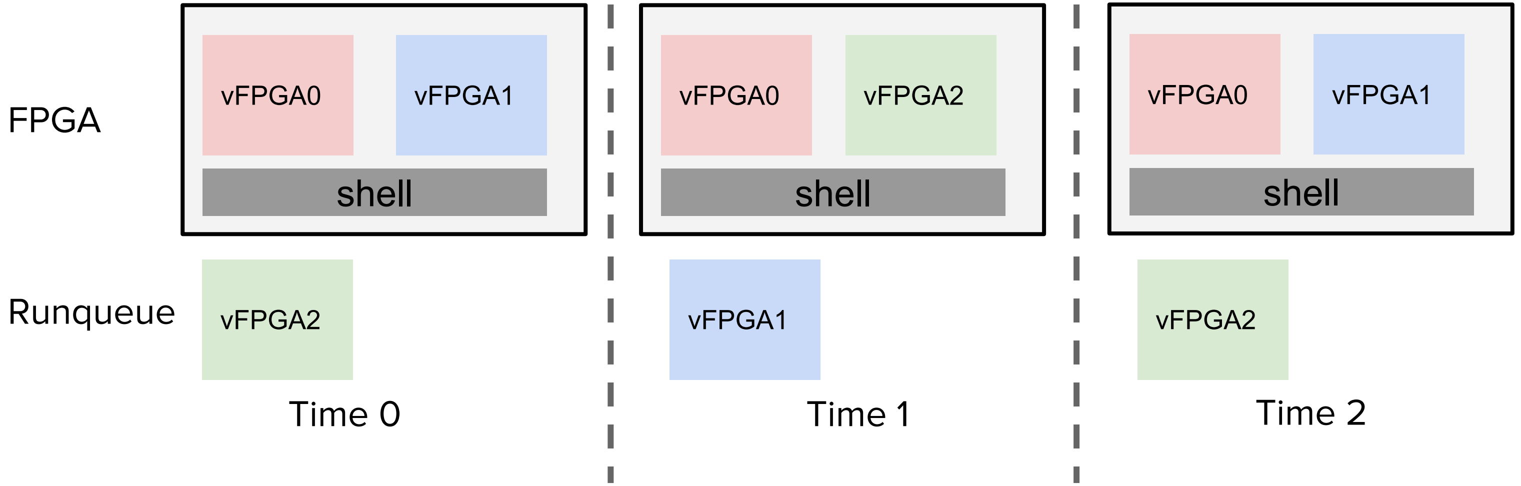
Image from AmorphOS, OSDI'18



Discussion: Scheduling Tasks in FPGA

- What are two common approaches?
- Can we do the same on FPGA?

Space sharing is easy.
Time sharing is not, esp for **preemptive time sharing**
- It's hard to readback the states, not what FPGA is designed for

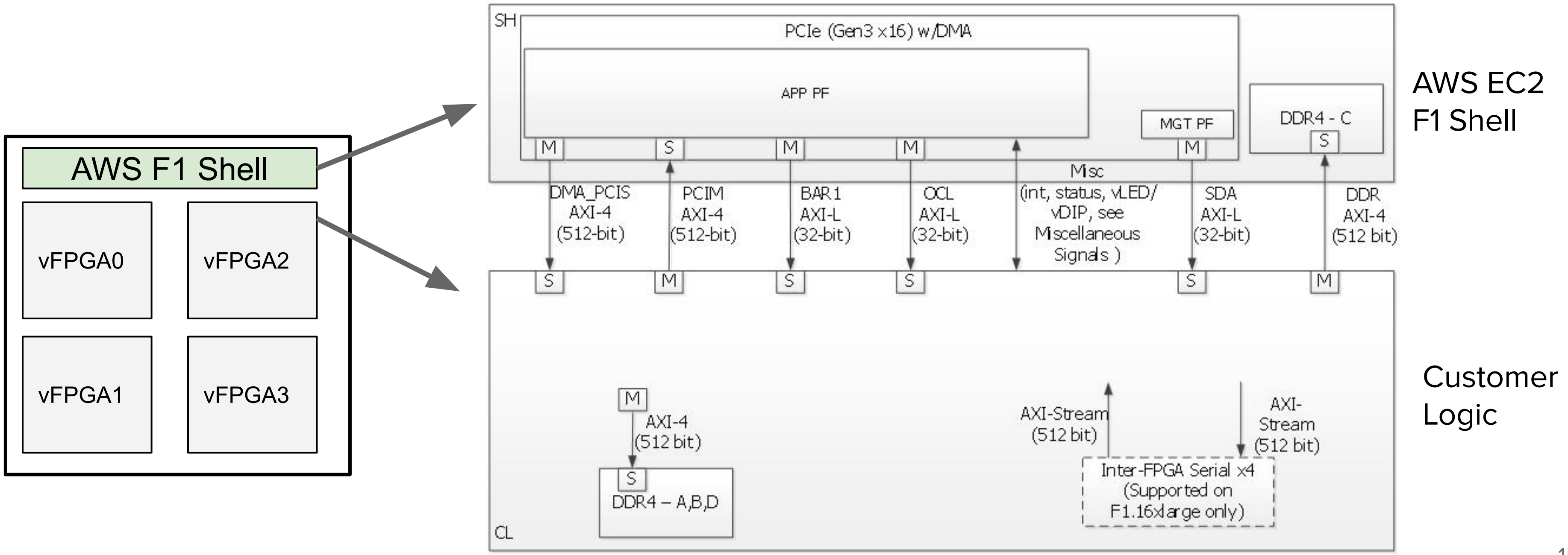


Protection

- Vendor Shells
  - Basic raw PCIe, memory controller IPs
  - Clocks, virtual LEDs, Pads etc
  - No sharing, protection, multiplexing mechanism.

Why not just let users deploy those raw IPs?

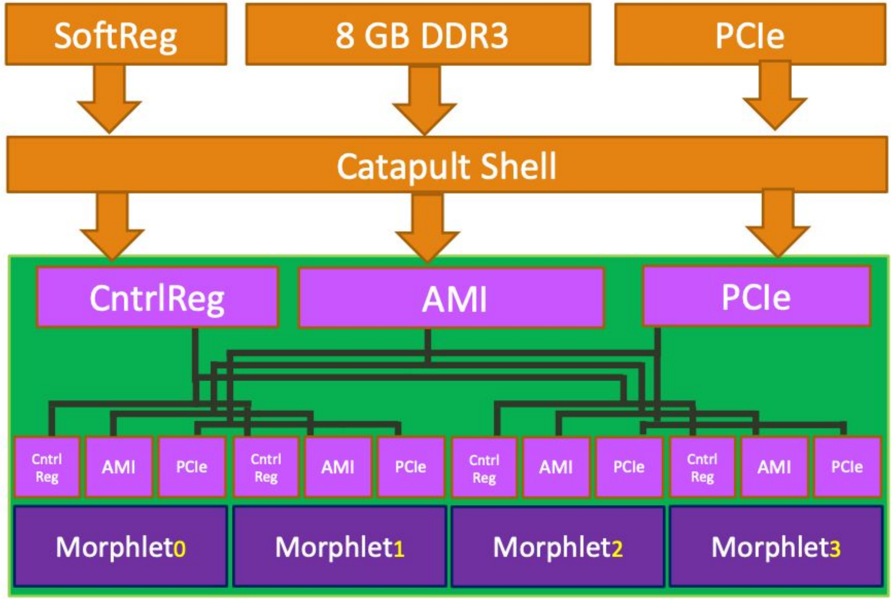
1. Those are heavy lifting tasks
   a. A novice user can easily spent days/months on just setting up
2. Safety. Misconfiguration might harm chip.



A set of APIs
A way to partition the chip → ***Zone***
A way to scale FPGA apps → ***Morphlet***
A way to mix FPGA apps → ***High-throughput/Low-latency mode***
A way to protect resource from untrusted FPGA applications → ***Hull***
Finally a way to deploy mixed apps onto protected and partitioned chip → ***Registry***

Protection

- What need to be protected?
  - Host/On-board DRAM
  - Other host PCIe devices
- AmorphOS Hull
  - Hardens and extends vendor shells
  - Isolation/Protection/Fairness
  - Interfaces
    - Control (CntrlReg)
    - Virtual Memory (AMI)
    - Data Transfer (PCIe)
- Basic idea
  - Mediate all IO requests



A set of APIs

A way to partition the chip → ***Zone***

A way to scale FPGA apps → ***Morphlet***

A way to mix FPGA apps → ***High-throughput/Low-latency mode***

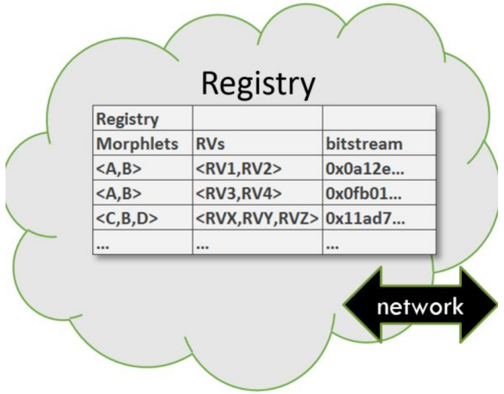
A way to protect resource from untrusted FPGA applications → ***Hull***

Finally a way to deploy mixed apps onto protected and partitioned chip → ***Registry***

Registry

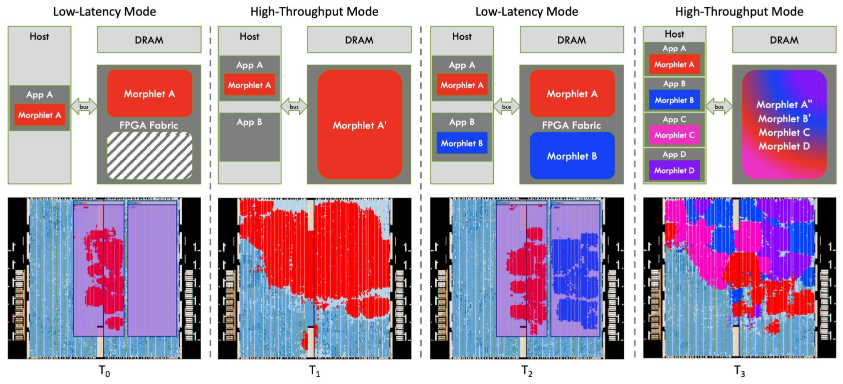
Rationale

- Compiling takes time (hours to days)
- To switch, next bitstream must be ready
- AmorphOS will do precompile and put them into a *bitstream registry*


Registry
network

Registry		
Morphlets	RVs	bitstream
<A,B>	<RV1,RV2>	0x0a12e...
<A,B>	<RV3,RV4>	0x0fb01...
<C,B,D>	<RVX,RVY,RVZ>	0x11ad7...
...	...	...

For this particular case, we need 4 bitstreams!



Limitations of AmorphOS

- App Support
  - The key to AmorphOS's success is its ability to **right-sizing apps**
  - To take advantage, apps must be written in a way that can scale
  - Thus, its solution is more on managing app rather than managing chip
- Virtualization Support
  - How to use host resource in a virtualization-enabled node? E.g., with IOMMU in place.
- Runtime
  - The hull protection is static and lacks of a runtime dynamic mgmt part
  - A SW-programmer friendly interface: e.g., malloc/free, read_file, etc.

Advancement in this field

ViTAL, ASPLOS'20

A framework that is able to partition any FPGA applications, and partition the FPGA into identical blocks.

Thus it overcomes the app support part, and also has a finer-grained scheduling unit.
(No app modification needed)

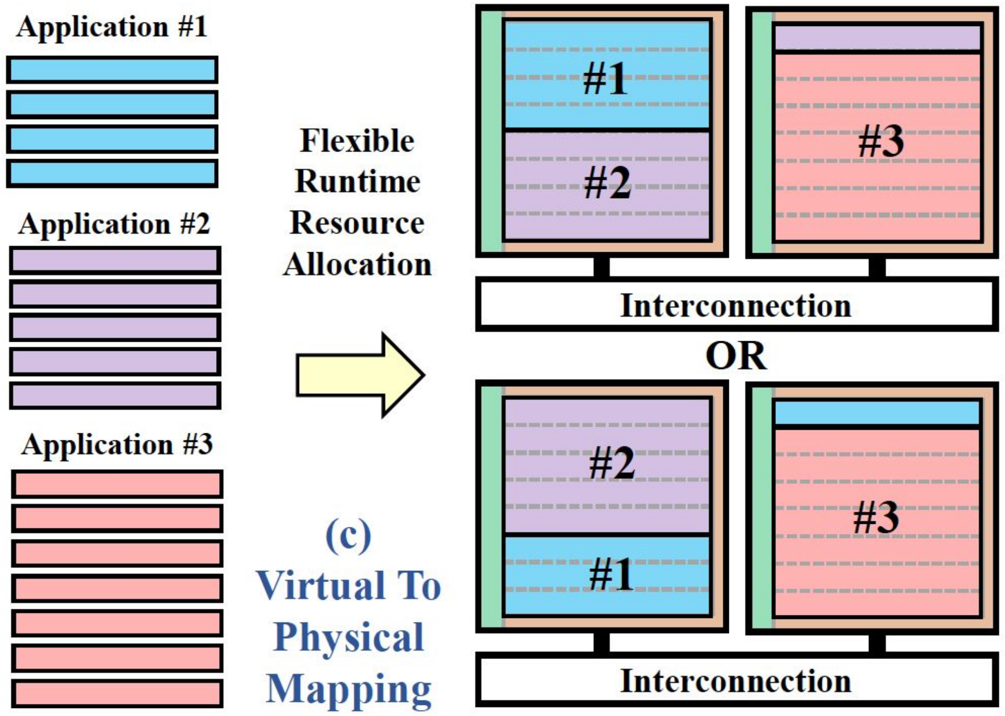


Image from ViTAL, ASPLOS'20

Advancement in this field

Optimus, ASPLOS'20

Deal with DMA+IOMMU.

Essentially implemented
an IOTLB and IO Page Table Walker in FPGA

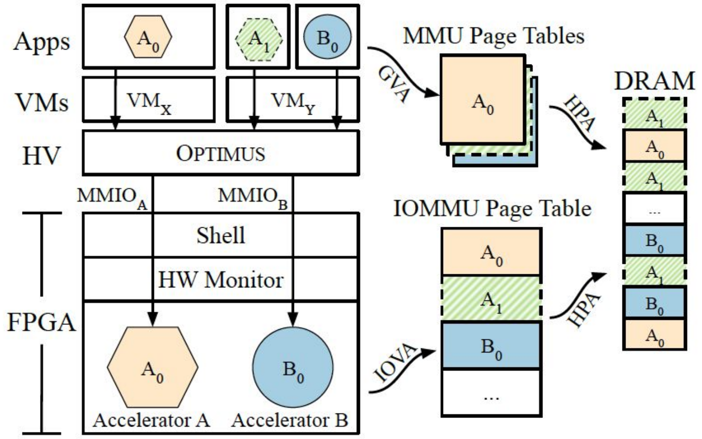


Figure 2. OPTIMUS design overview, shown with two physical accelerators for brevity. OPTIMUS spatially multiplexes a shared-memory FPGA as physical accelerators (*A* and *B*), and temporally multiplexes physical accelerators as virtual accelerators (A_0, A_1, and B_0).

Image from Optimus, ASPLOS'20

Summary

- FPGA is massively deployed in Cloud
- Shared Cloud FPGA is still in its infancy
- A lot exciting research is going on, trying to improve the model

Thank you.
Questions?