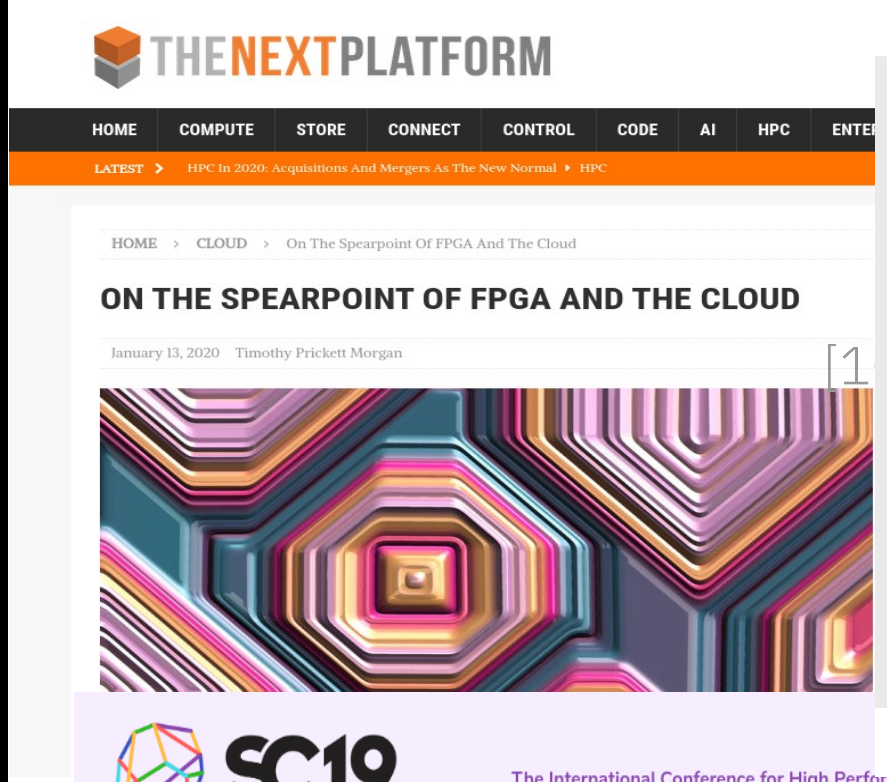
openRole: Can we bring ‘Design once, run everywhere’ to FPGAs?

—

Burkhard Ringlein & cloudFPGA Team Zurich
IBM Research Europe

FPGAs are spreading in DCs and Clouds…

- **…that's why we are here**
- Further Examples:
  - AWS FPGA instances
  - Microsoft's Project Brainwave
  - OC-Accel/Snap
  - PC², Galapagos, cloudFPGA, etc…
- Consequences:
  - Cloud and Datacenter are typically multi-user environments
    → Constrains the architecture
  - Usually, not all app developers want to deal with the I/O details of specific boards
    → Vendors provide platforms


THENEXTPLATFORM

HOME | COMPUTE | STORE | CONNECT | CONTROL | CODE | AI | HPC | ENTER

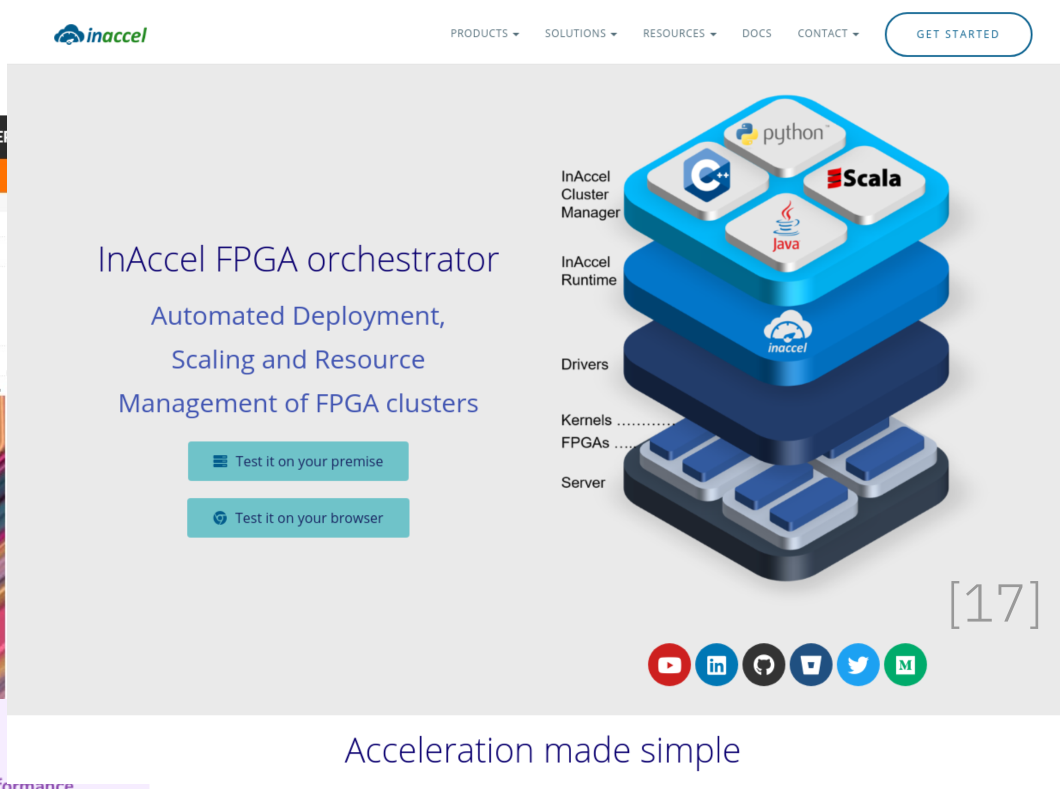
LATEST › HPC In 2020: Acquisitions And Mergers As The New Normal ▸ HPC

HOME › CLOUD › On The Spearpoint Of FPGA And The Cloud

ON THE SPEARPOINT OF FPGA AND THE CLOUD

January 13, 2020 Timothy Prickett Morgan

[1

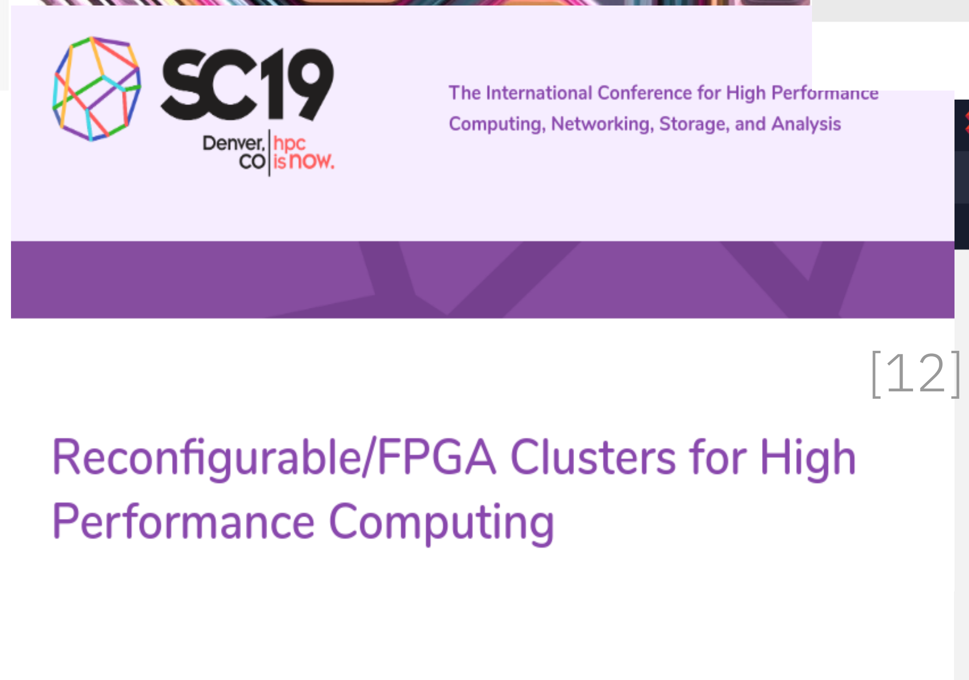


InAccel FPGA orchestrator

Automated Deployment, Scaling and Resource Management of FPGA clusters

[17]

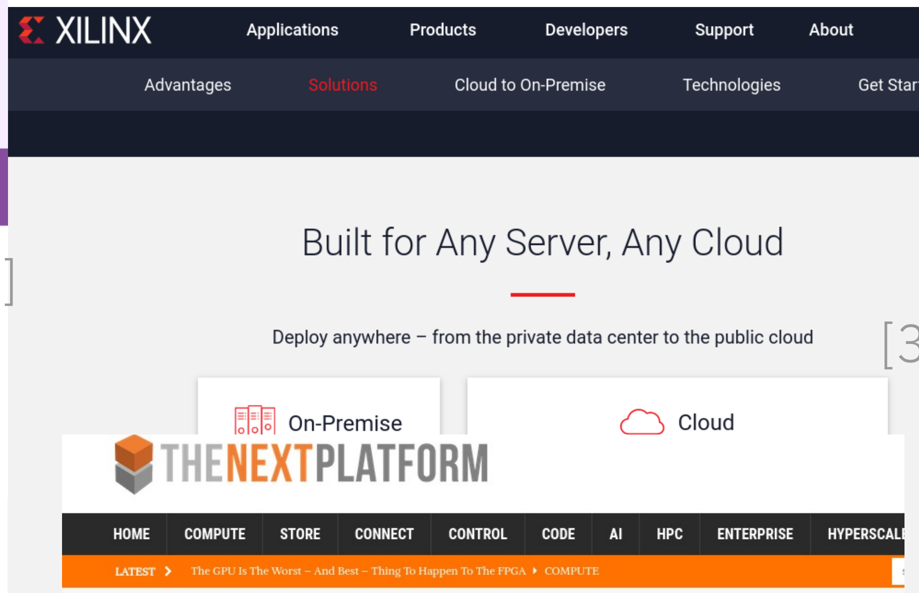
Acceleration made simple



The International Conference for High Performance Computing, Networking, Storage, and Analysis

[12]

Reconfigurable/FPGA Clusters for High Performance Computing



Built for Any Server, Any Cloud

Deploy anywhere – from the private data center to the public cloud

[3

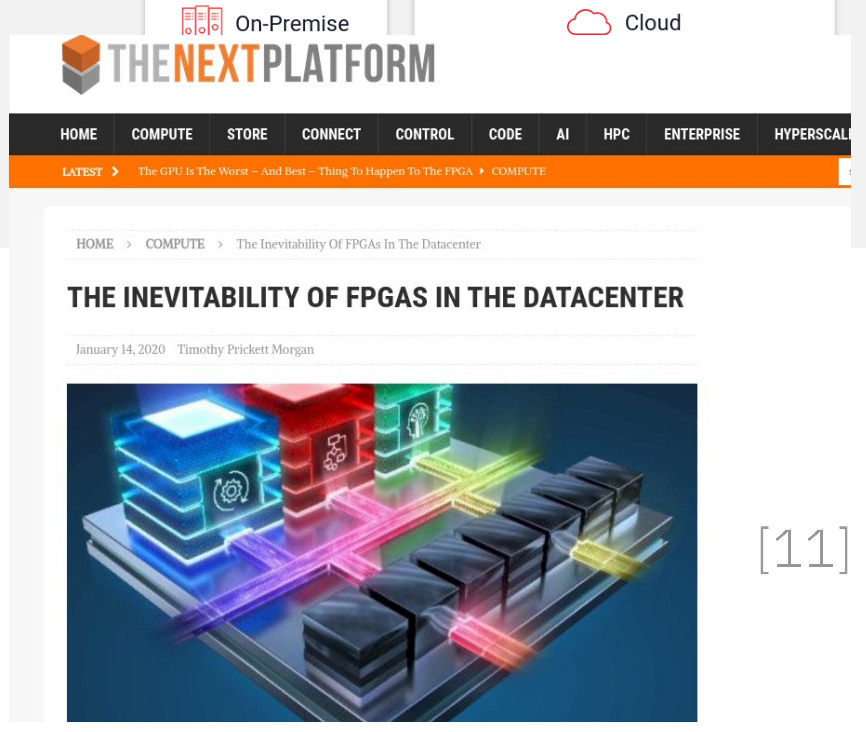
Fifth International Workshop on
Heterogeneous High-performance Reconfigurable Computing (H^{2}RC'19)

Sunday, November 17, 2019 (9:00am to 5:30pm)
Denver, CO

Held in conjunction with:

SC19: The International Conference for High Performance Computing, Networking, Storage and Analysis

[13]


THENEXTPLATFORM

HOME | COMPUTE | STORE | CONNECT | CONTROL | CODE | AI | HPC | ENTERPRISE | HYPERSCALE

LATEST › The GPU Is The Worst – And Best – Thing To Happen To The FPGA ▸ COMPUTE

HOME › COMPUTE › The Inevitability Of FPGAs In The Datacenter

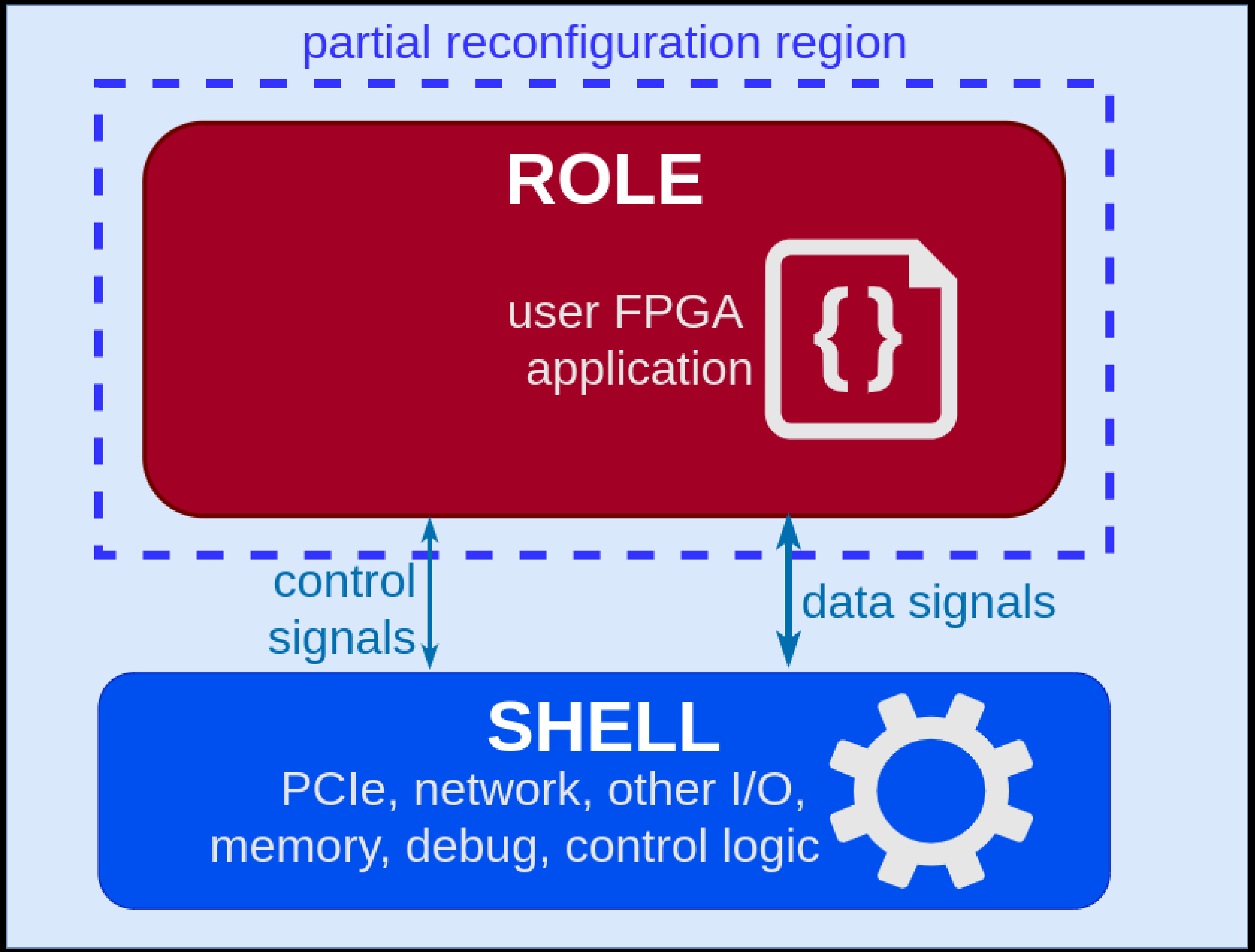
THE INEVITABILITY OF FPGAS IN THE DATACENTER

January 14, 2020 Timothy Prickett Morgan

[11]

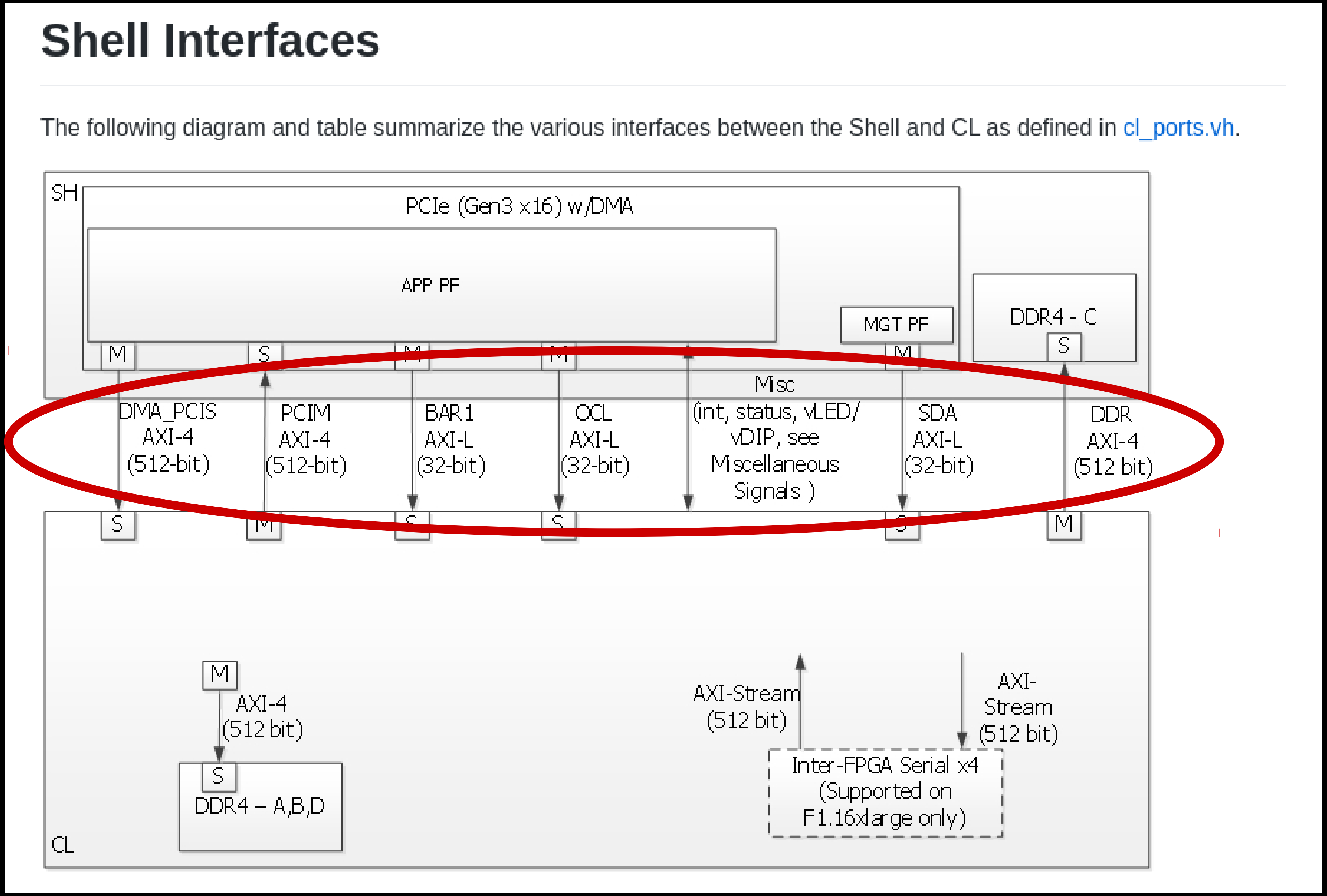
Providing a platform: The Shell Role Architecture (SRA)

- Split the FPGA design into:
  - a platform specific part: **SHELL**
  - an application specific part: **ROLE**
- The SRA design pattern offers:
  - I/O abstractions
  - different privilege levels
  - improves platform security (especially in combination with partial reconfiguration)
- Consequently: SRAs are used frequently (by all major Cloud vendors)
- SRAs are "*the APIs of FPGA app developers*"


partial reconfiguration region
ROLE
user FPGA application
control signals
data signals
SHELL
PCIe, network, other I/O, memory, debug, control logic

Some examples of SRAs

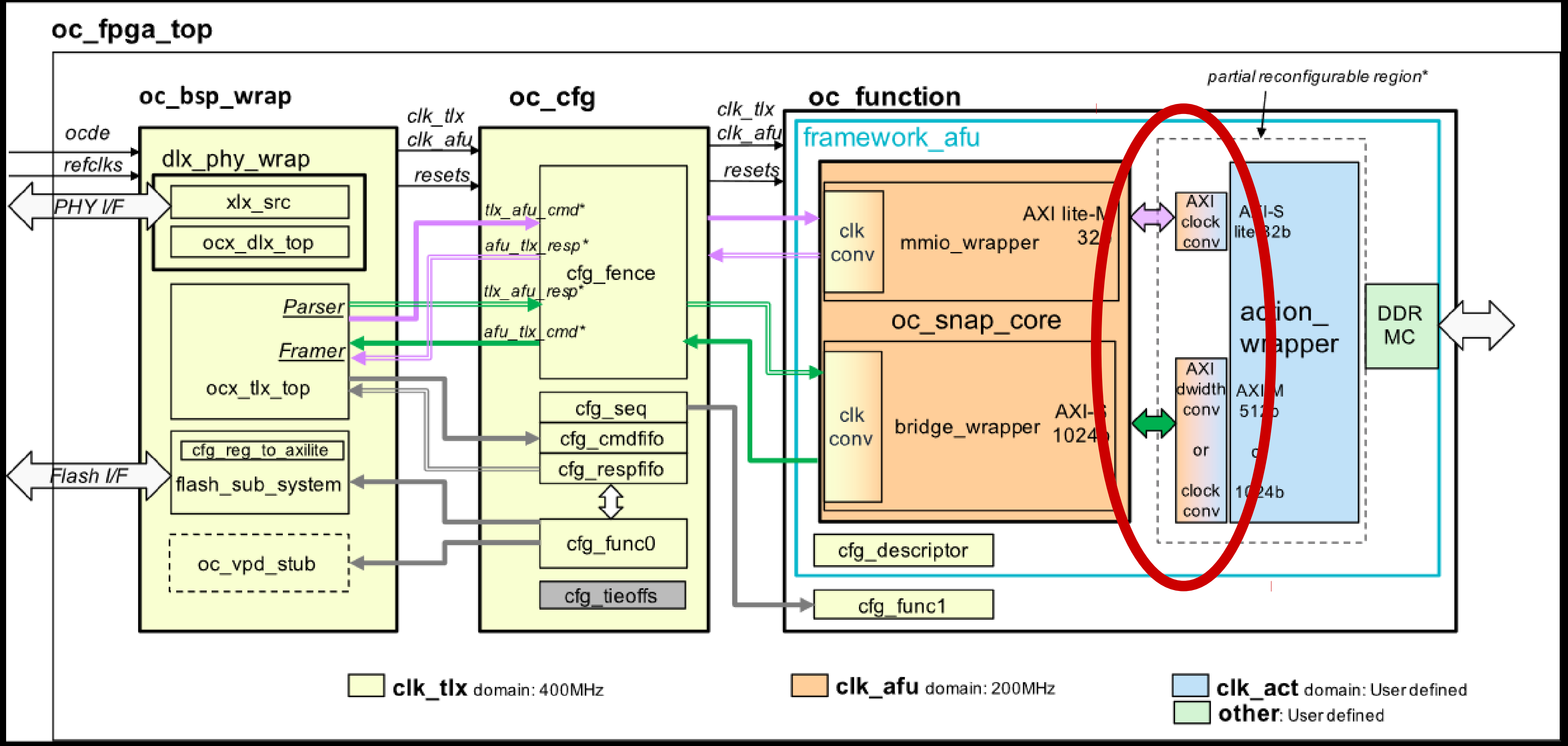
AWS [1]



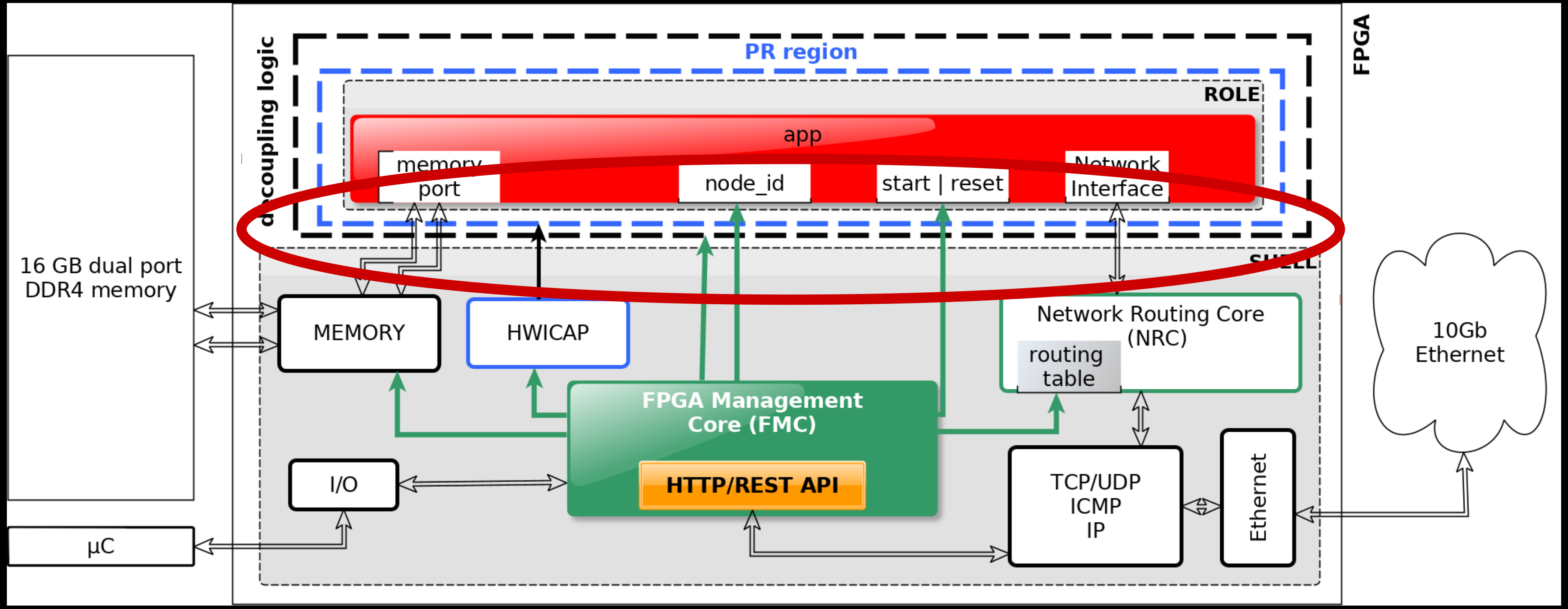
Shell Interfaces

The following diagram and table summarize the various interfaces between the Shell and CL as defined in cl_ports.vh.

OC-Accel (OpenCAPI) [2]

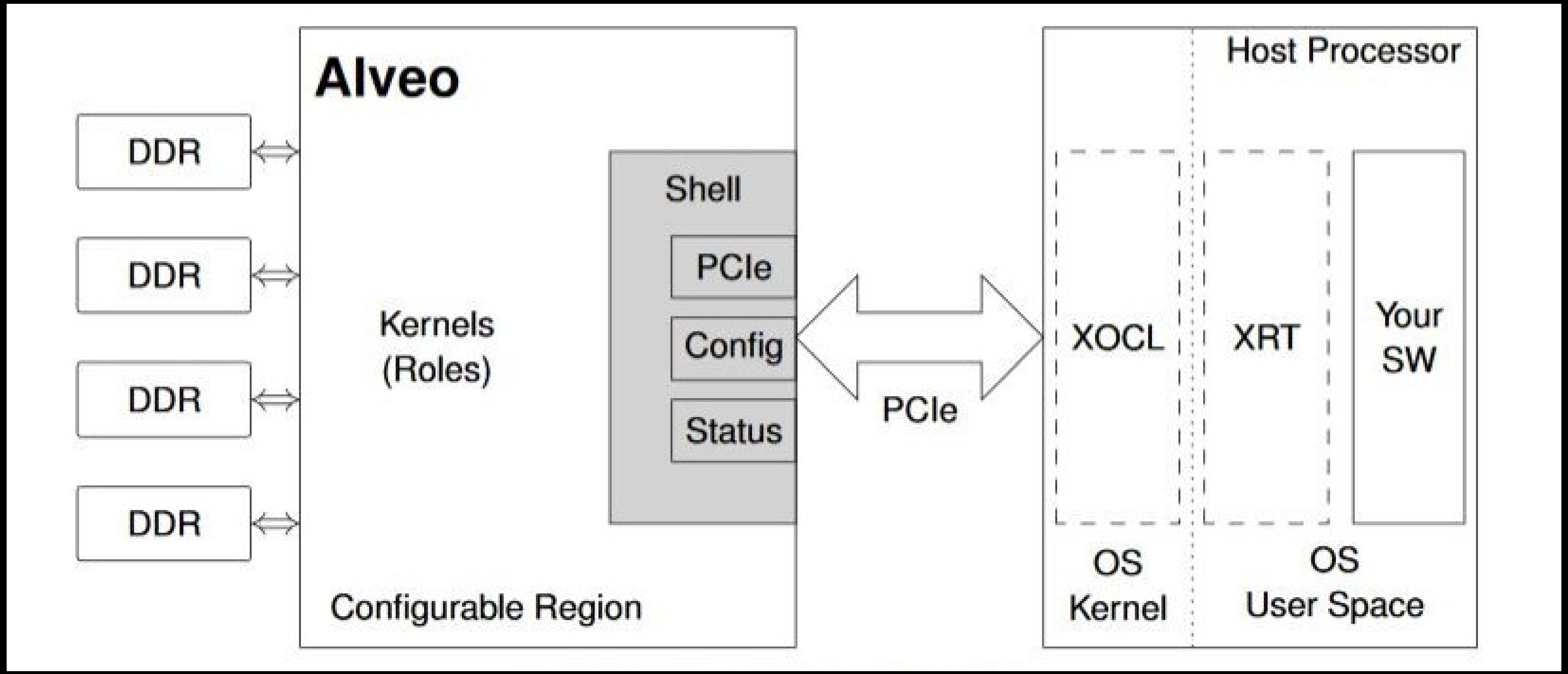


cloudFPGA [14]

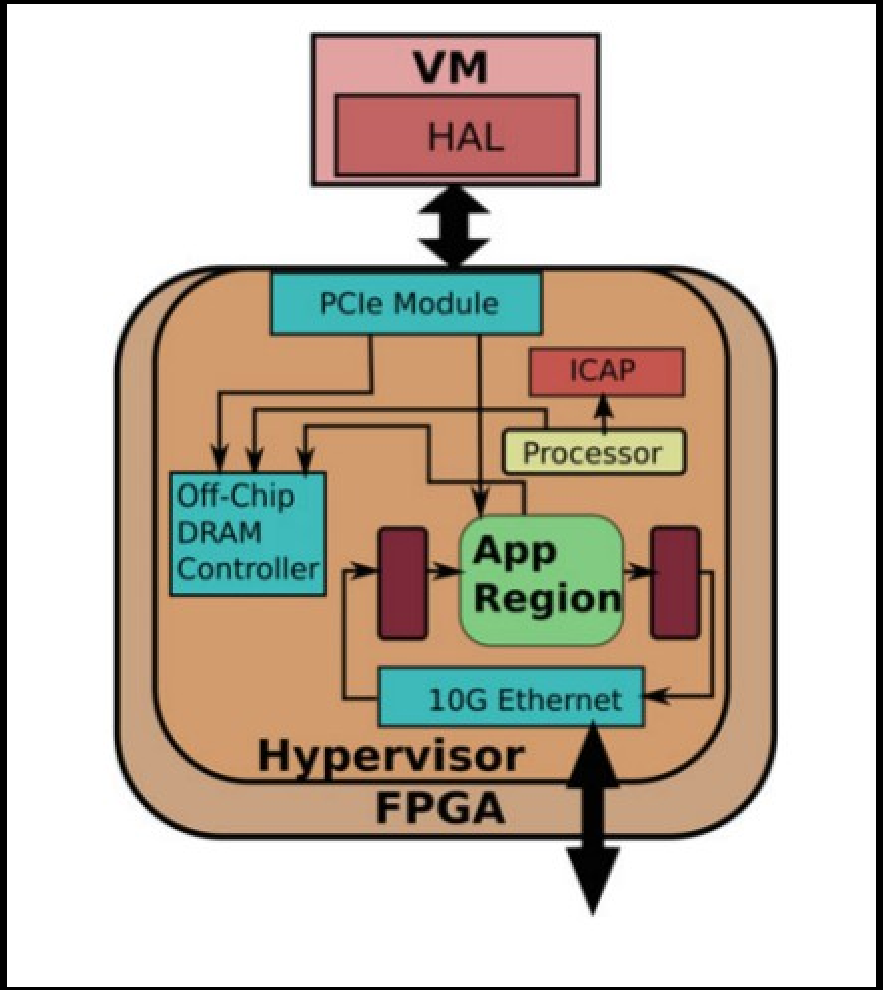


Some more examples of SRAs

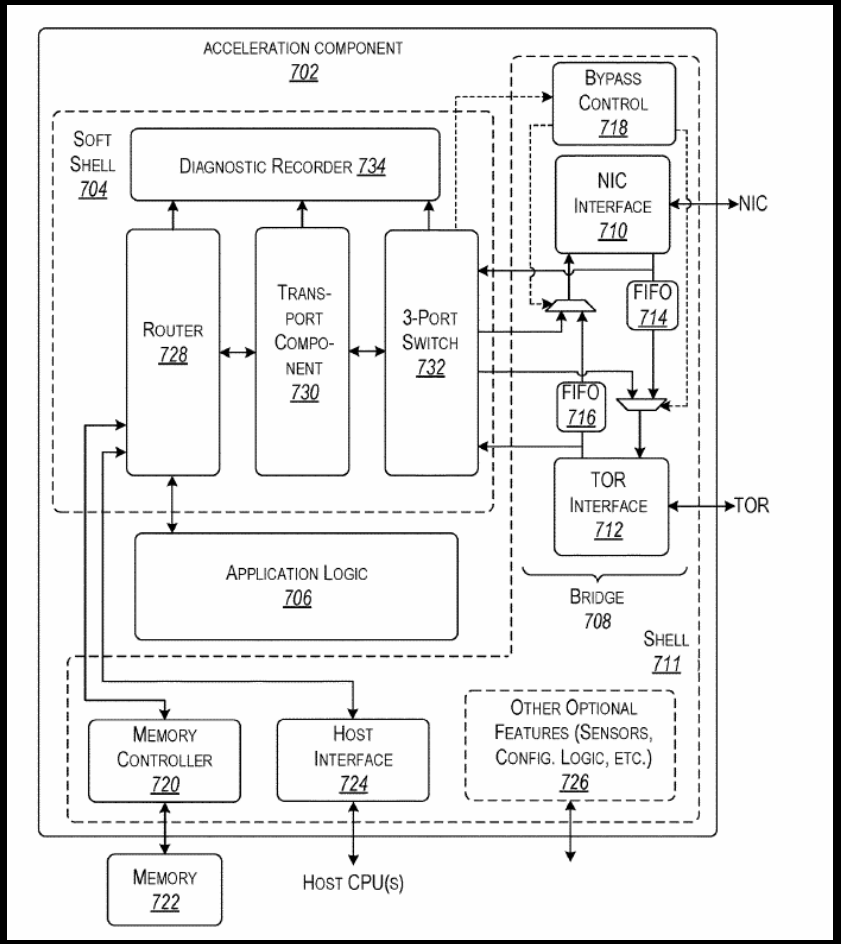
Alveo cards [7]



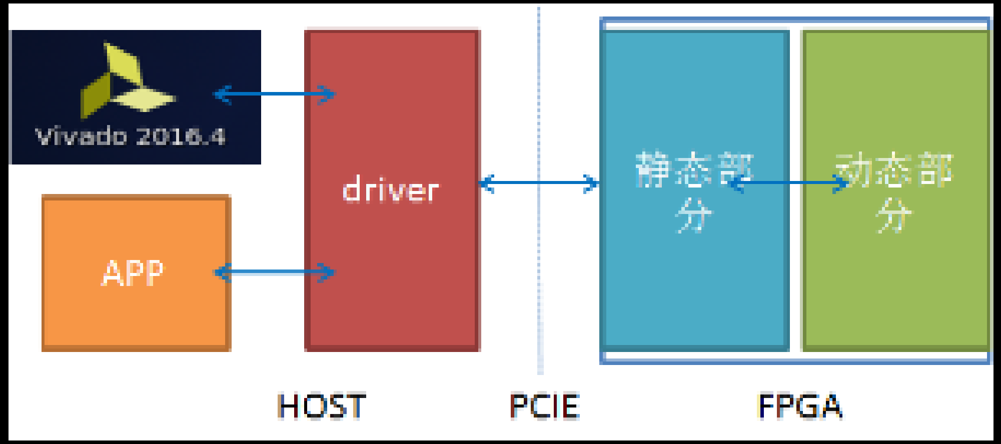
Galapagos FPGA Hypervisor [5]



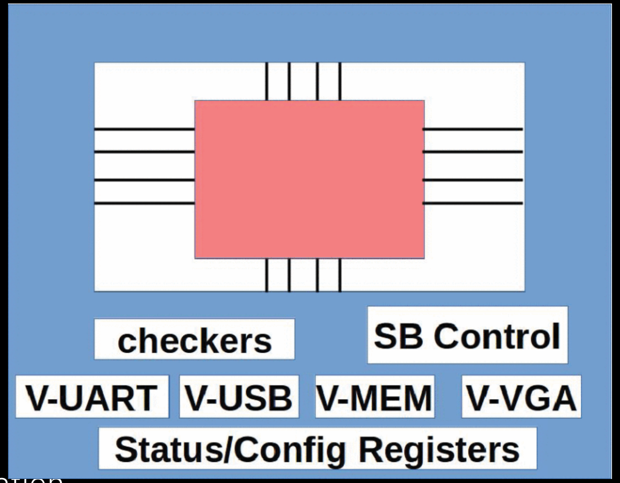
Project Brainwave [8]



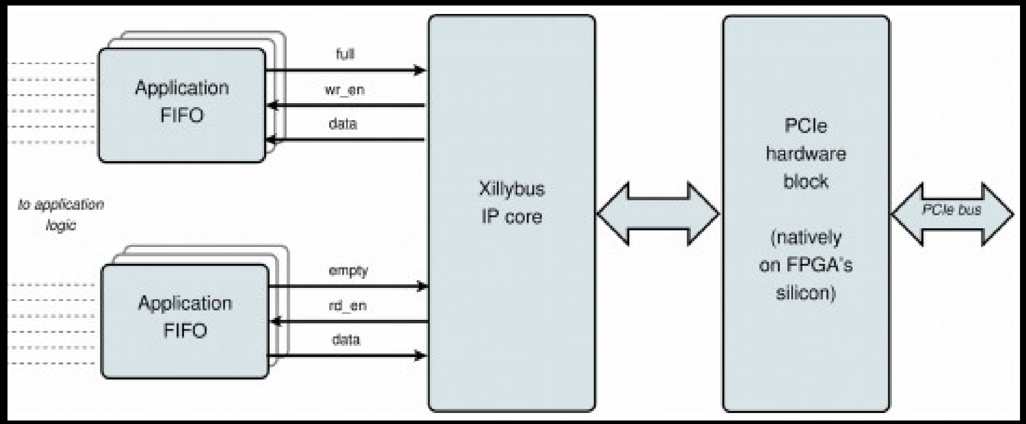
Baidu FPGA Cloud server [15]



Hardware Sandboxes (Florida) [16]



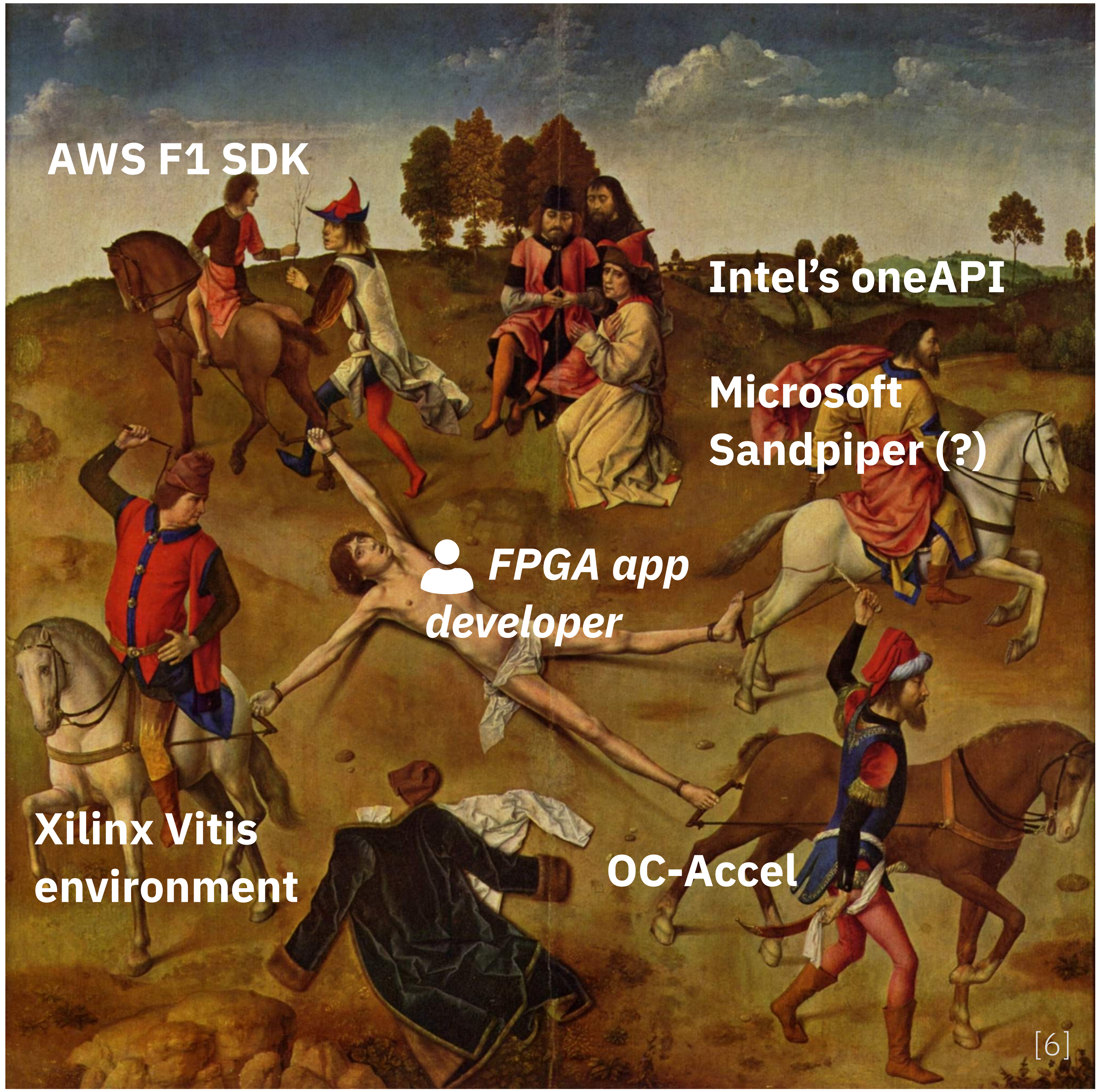
Xillybus (kind of) [9]



`/dev/xillybus_<name>`

...and counting...

The Problem: Every vendor provides a different interface

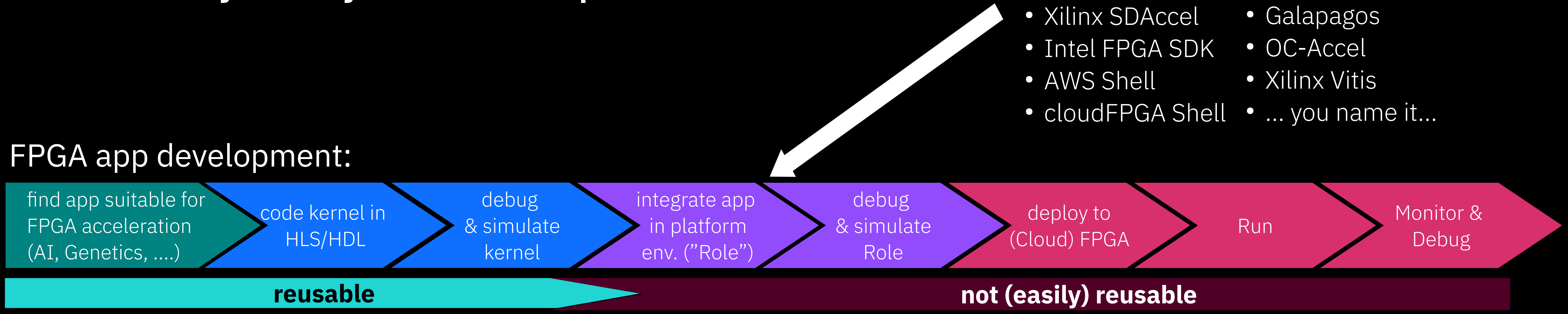


Actually, why is this a problem?

Role environment examples:

- Xilinx SDAccel
- Intel FPGA SDK
- AWS Shell
- cloudFPGA Shell
- Galapagos
- OC-Accel
- Xilinx Vitis
- … you name it…

FPGA app development:



- FPGA application development is tightly coupled to interfaces (i.e. "I/O")
- to achieve low latency / high throughput:
  run time behavior and interfaces *must be considered* at design time (stalls, request latency, bursts…)

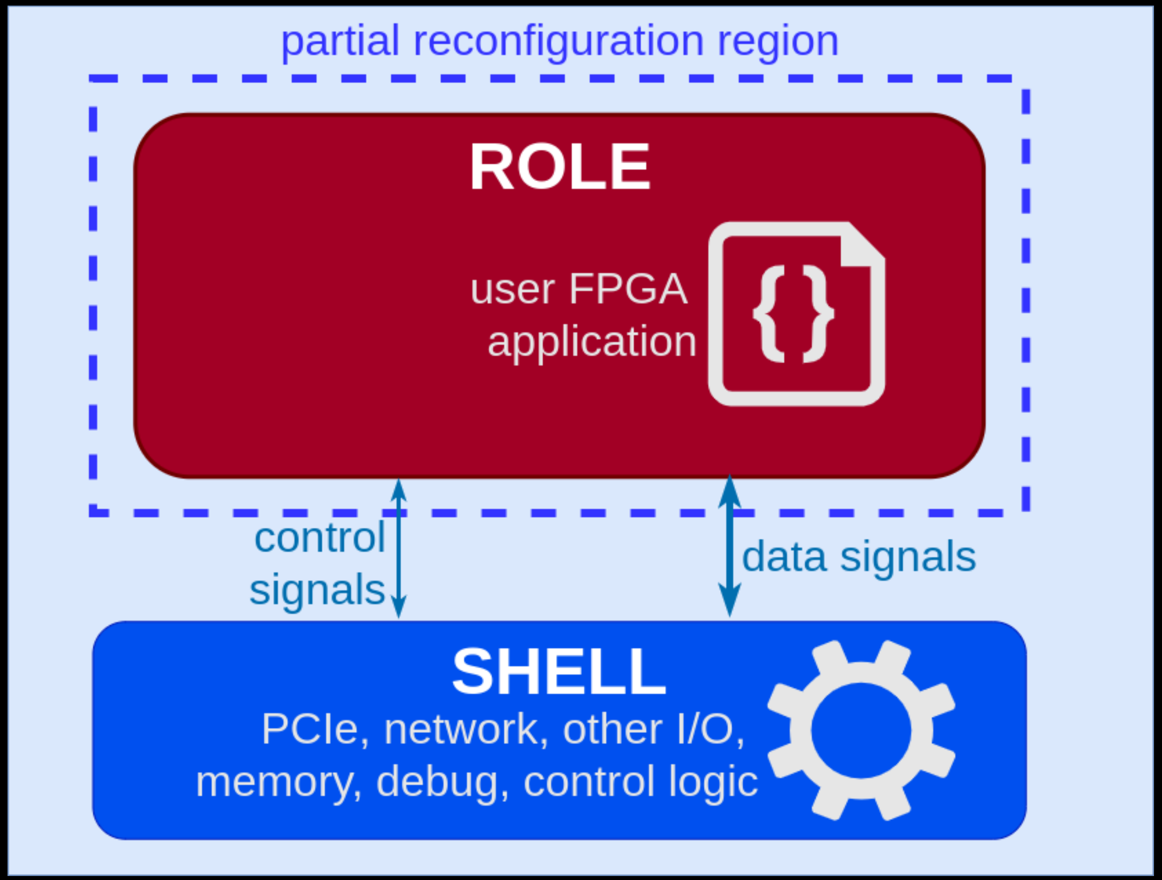
→ FPGA applications are barely interface agnostic

→ application depends on Shell interfaces ("APIs")

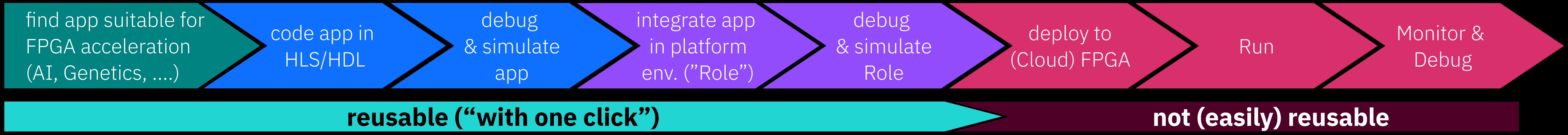
→ limited re-usability & portability

What would be a better solution?

- **limited (or no) portability** of FPGA designs is another "roadblocker" in the way of FPGAs
- (good) reasons for change of platforms:
  - elasticity, scaling
  - evolution of platforms
  - Cloud vendor change, business decisions, etc.
- re-usable **Roles** would greatly accelerate the adaption of new platforms
  → and consequently, the growth of the ecosystem

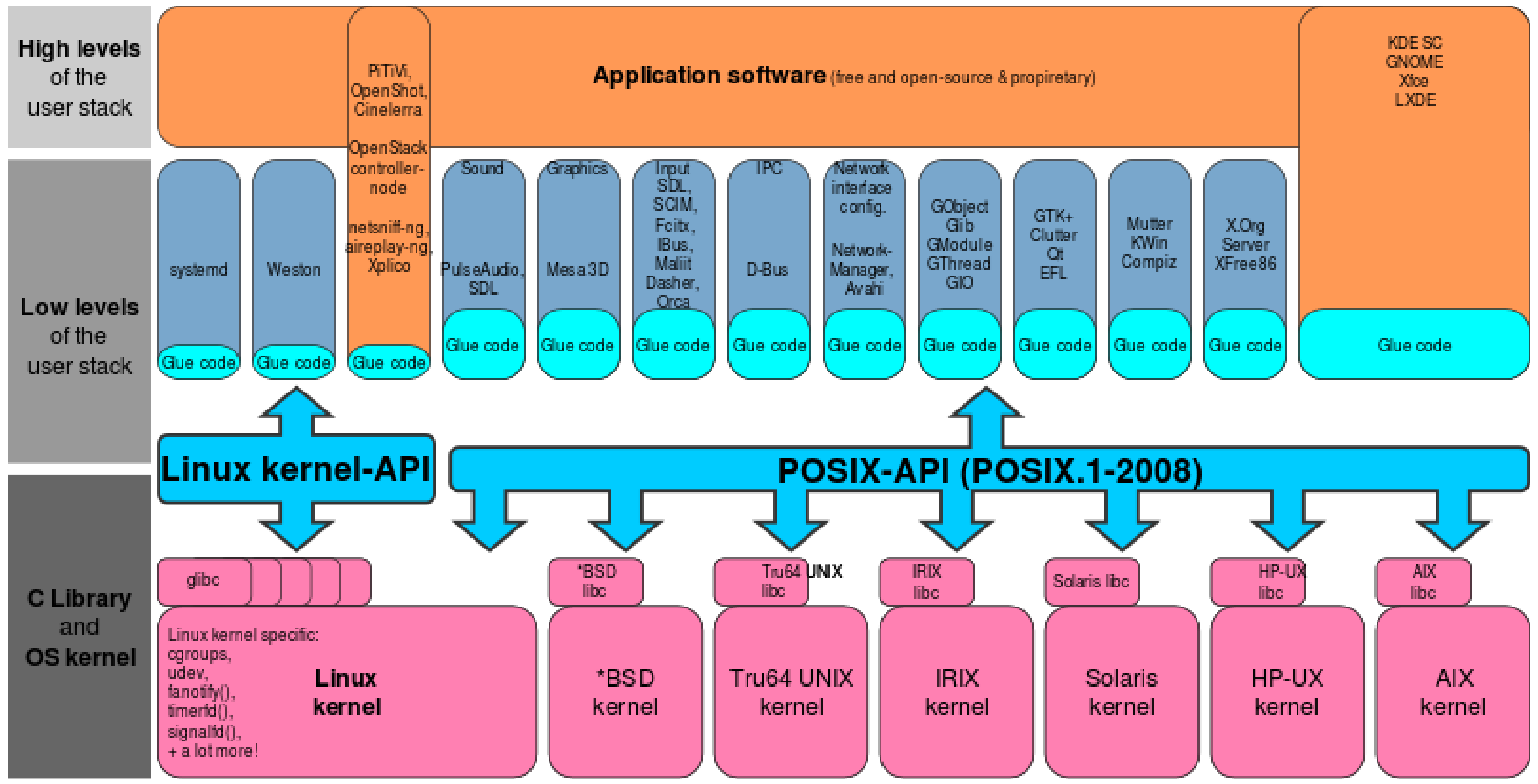

partial reconfiguration region
ROLE
user FPGA application
control signals
data signals
SHELL
PCIe, network, other I/O, memory, debug, control logic

with **portable / reusable Roles**:


find app suitable for FPGA acceleration (AI, Genetics, ...)
code app in HLS/HDL
debug & simulate app
integrate app in platform env. ("Role")
debug & simulate Role
deploy to (Cloud) FPGA
Run
Monitor & Debug
reusable ("with one click")
not (easily) reusable

Once upon a time…

- late 1970s: many disparate versions of OSs
- **POSIX: Portable Operating System Interface**
- Originated 1988 to maintain compatibility between OSs
- A POSIX compliant application can be compiled and deployed on every POSIX compliant OS, **without code changes**
- Latest update: IEEE Std 1003.1-2017
- Linux, macOS & Android are POSIX-compliant (but extend the API with custom calls)
  → a (well-designed) **SW application can be ported effortless between these platforms**
- (Windows Kernel offers a POSIX compatibility layer)


High levels of the user stack
Low levels of the user stack
C Library and OS kernel
Application software (free and open-source & proprietary)
PiTiVi, OpenShot, Cinelerra
OpenStack controller-node
netsniff-ng, aireplay-ng, Xplico
KDE SC GNOME Xfce LXDE
systemd
Weston
Sound PulseAudio, SDL
Graphics Mesa 3D
Input SDL, SCIM, Fcitx, IBus, Maliit Dasher, Orca
IPC D-Bus
Network interface config. Network-Manager, Avahi
GObject Glib GModule GThread GIO
GTK+ Clutter Qt EFL
Mutter KWin Compiz
X.Org Server XFree86
Glue code
Linux kernel-API
POSIX-API (POSIX.1-2008)
glibc
*BSD libc
Tru64 UNIX libc
IRIX libc
Solaris libc
HP-UX libc
AIX libc
Linux kernel specific: cgroups, udev, fanotify(), timerfd(), signalfd(), + a lot more!
Linux kernel
*BSD kernel
Tru64 UNIX kernel
IRIX kernel
Solaris kernel
HP-UX kernel
AIX kernel

[4]

```
POSIX = code once, run everywhere,
        if you have a compiler
```

What does POSIX bring?

- **POSIX defines APIs and data structures** for OS system calls (among other things)
- *Any program that*:
  - applies the POSIX API of the structs
  - uses a programming language that offers a POSIX compliant compiler (for the target platform)
  - → **can be compiled and executed on any POSIX compliant Operating System** without further code changes
- POSIX also **specifies the commands and flags** to invoke the compiler (e.g. **cc** or **c99** etc.)
- POSIX does **not** provide an **implementation** for the compilers or the system calls

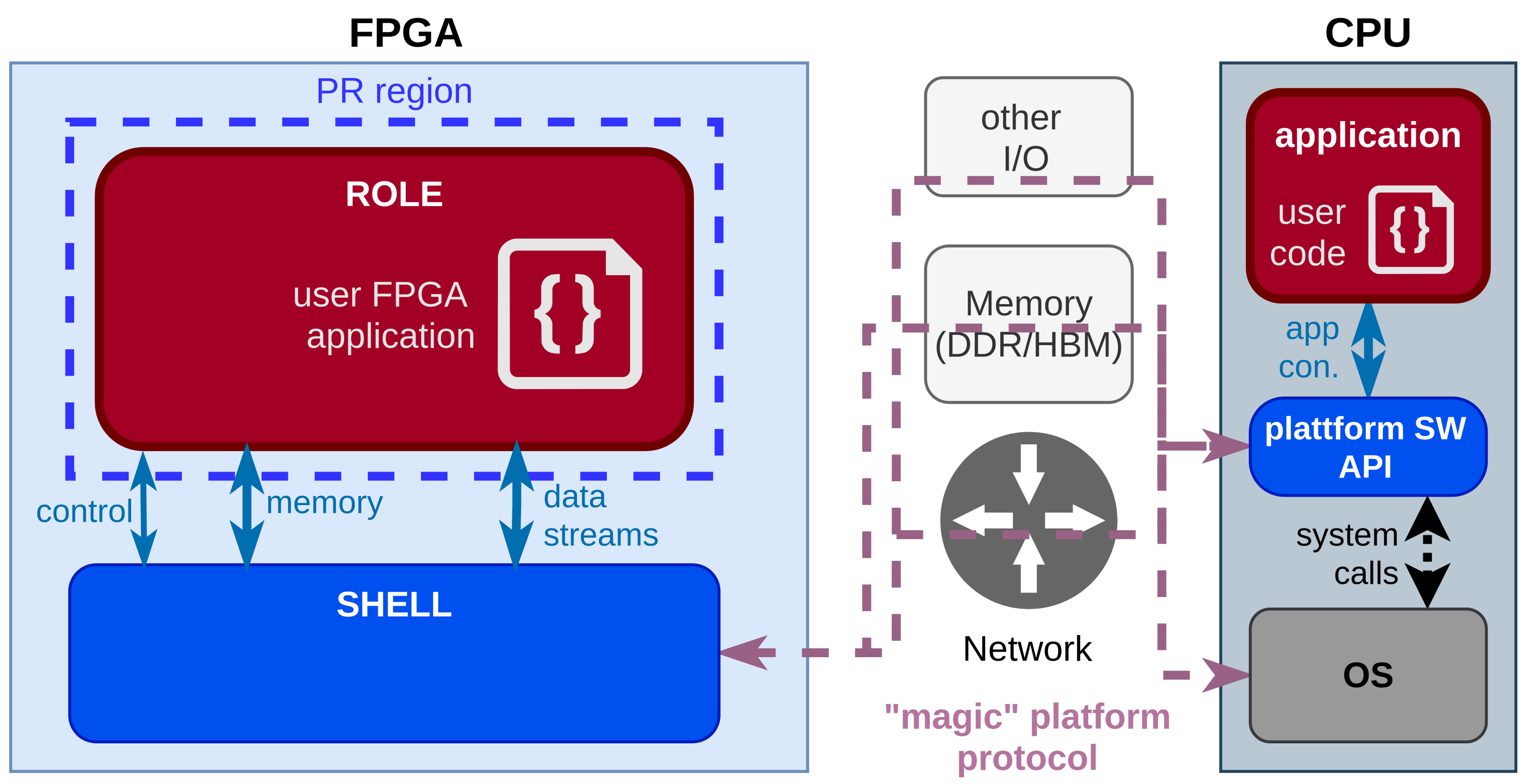
```
$ man 3p sendto
```

```
SENDTO(3P)
                     POSIX Programmer's Manual


NAME
       sendto - send a message on a socket

SYNOPSIS
       #include <sys/socket.h>

       ssize_t sendto(int socket, const void *message, size_t length,
              int flags, const struct sockaddr *dest_addr,
              socklen_t dest_len);

DESCRIPTION
       The  sendto()  function shall send a message through a
connection-mode or connectionless-mode socket. If the socket is
connectionless-mode, the message shall be sent to the address
specified by dest_addr. If the socket is connection-mode, dest_addr
shall be ignored.
       ..........

RETURN VALUE
       Upon successful completion, sendto() shall return the number
of bytes sent. Otherwise, -1 shall be returned and errno set to
indicate the error.

ERRORS
       The sendto() function shall fail if:
       ...........
```

Can we do this for FPGAs too?

Can we bring “design once, run everywhere” to FPGAs?

→ the **openRole** proposal

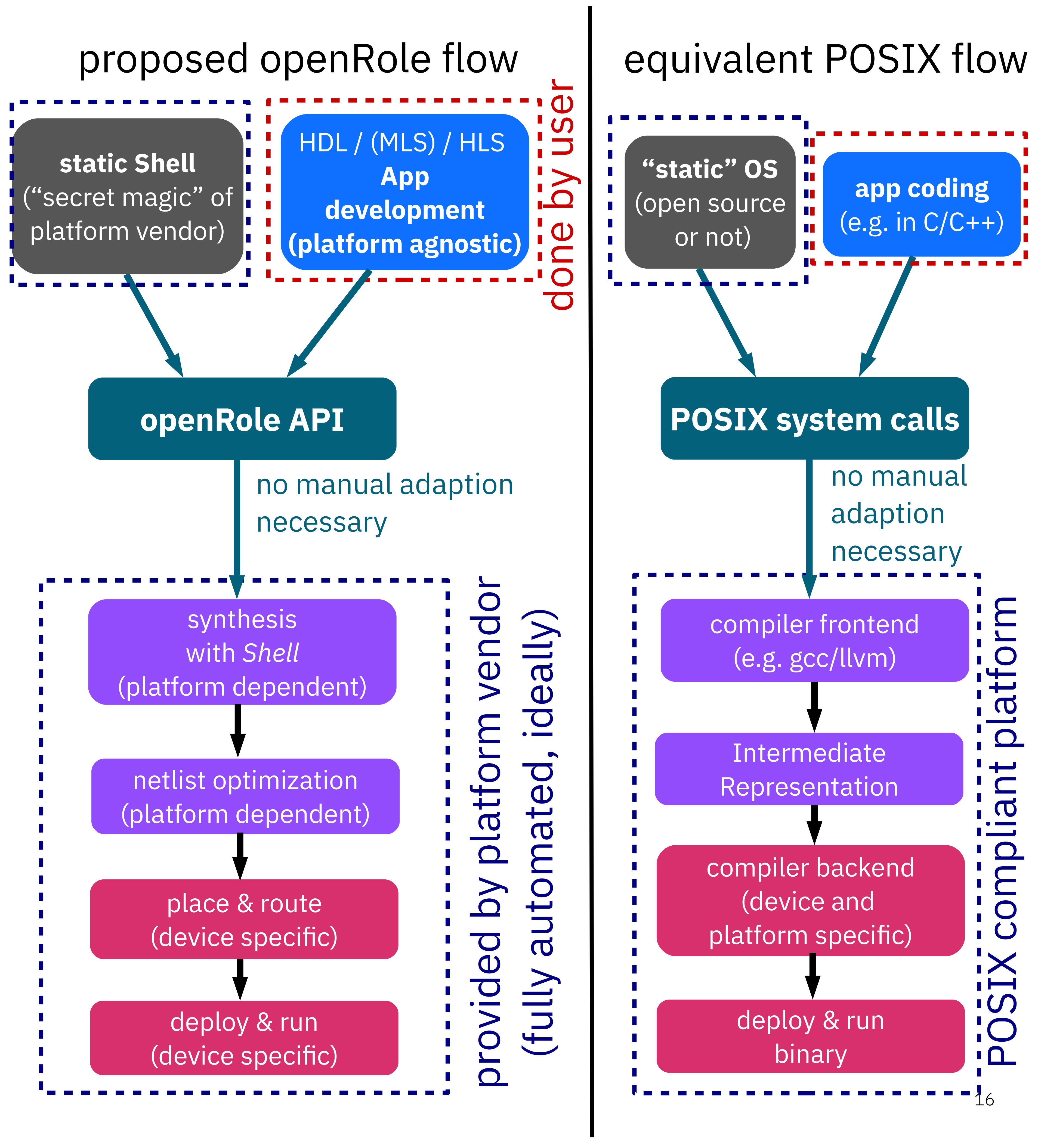


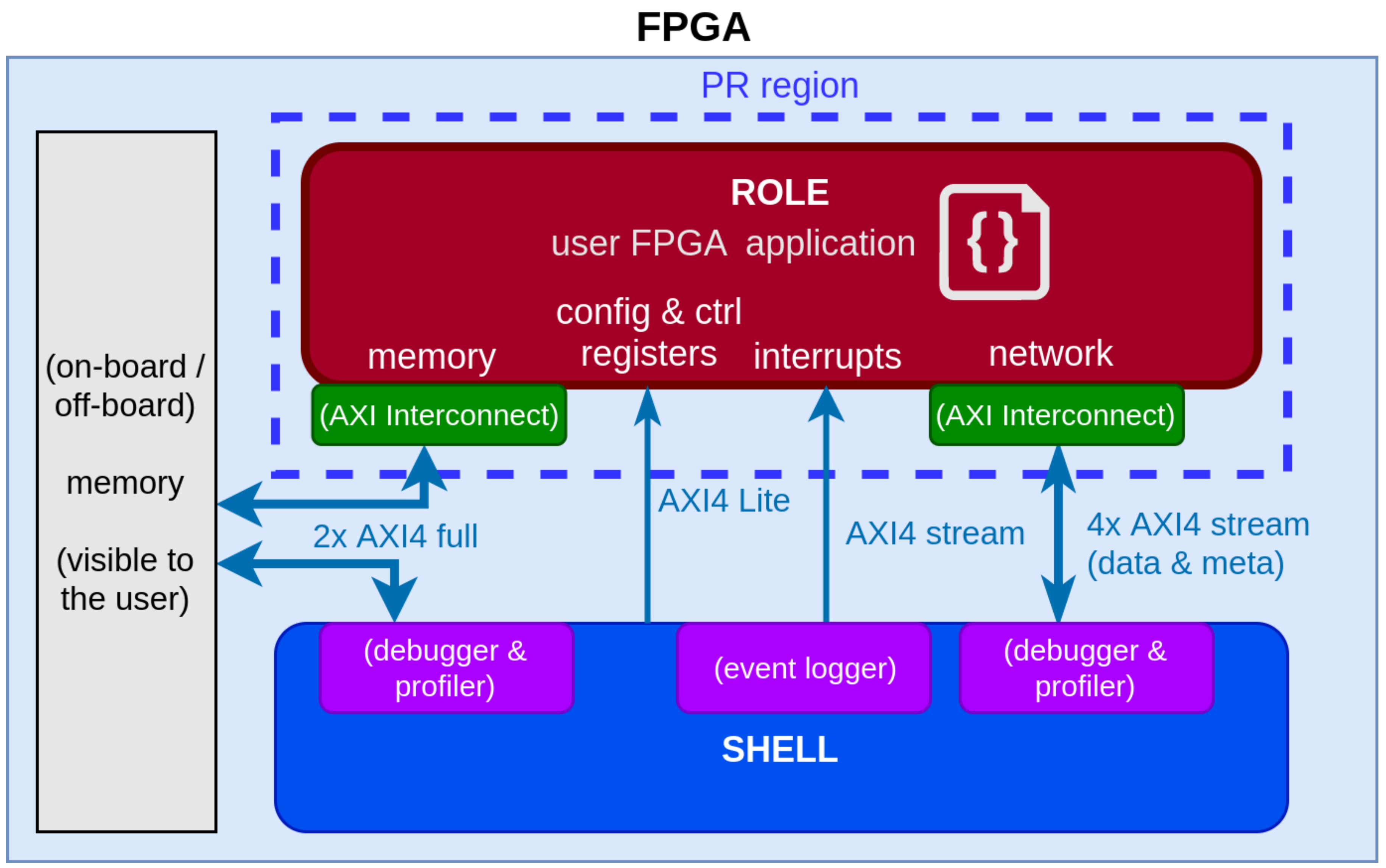
SRAs have already common concepts

- usually FPGA applications exist not alone: See as part of a (complex) application communication schema
- Besides configuration & control registers, there are usually (one of) two communication channels:
  - address based: PCIe, via Memory (AXI4 Full)
  - stream based: network or PCIe abstraction (AXI4 Stream)
- **Stream- and address-based communication can abstract every fabric** (PCIe, CXL, Ethernet, Infiniband, etc.)
- Define an interface for this level of abstraction *that is valid for all kind of platforms*

Design flow with **openRole**

- allow the **compilation/synthesis** of any Role for any compliant Platform
  - it is *not* limiting any platform or application implementation
  - it is *not* limiting any platform specific optimization
- → enables portable FPGA designs
  - SRAs have already common concepts
  - design flows are actually automated and similar across platforms and vendors
  - → we have to agree on a **common interface**
- **it is about portable FPGA designs**, not porting Software to FPGAs


proposed openRole flow
static Shell ("secret magic" of platform vendor)
HDL / (MLS) / HLS App development (platform agnostic)
done by user
openRole API
no manual adaption necessary
synthesis with Shell (platform dependent)
netlist optimization (platform dependent)
place & route (device specific)
deploy & run (device specific)
provided by platform vendor (fully automated, ideally)
equivalent POSIX flow
"static" OS (open source or not)
app coding (e.g. in C/C++)
POSIX system calls
no manual adaption necessary
compiler frontend (e.g. gcc/llvm)
Intermediate Representation
compiler backend (device and platform specific)
deploy & run binary
POSIX compliant platform



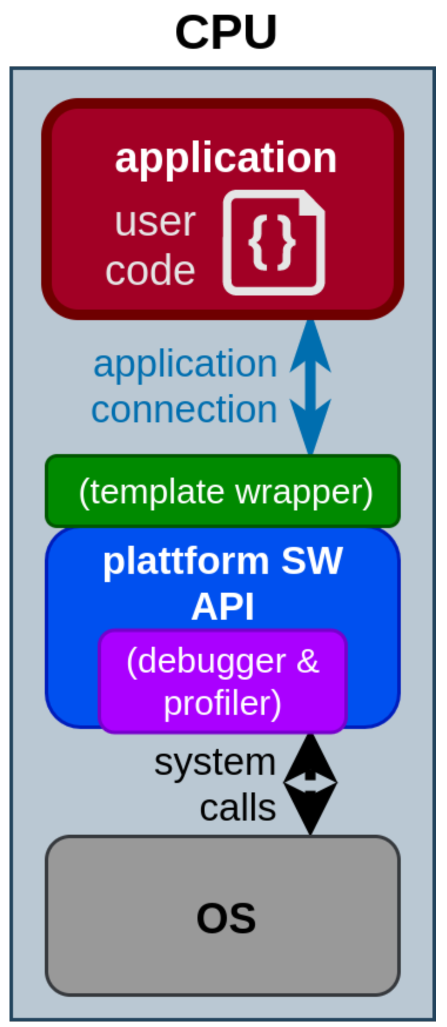
```
entity is oR_Role is
  port (
    piClk                     : in    std_ulogic;
    piRst                     : in    std_ulogic;
    ---- Configuration & Ctrl Registers AXI4-Lite ----------
    biOR_control_AXI_AWVALID            : in    std_ulogic;
    biOR_control_AXI_AWREADY            : out   std_ulogic;
    biOR_control_AXI_AWADDR             : in    std_ulogic_vector (15 downto 0);
.....
    ---- Input AXI-Write Stream Interface ----------
    siNetwork_Data_tdata                : in    std_ulogic_vector( 63 downto 0);
    siNetwork_Data_tkeep                : in    std_ulogic_vector(  7 downto 0);
    siNetwork_Data_tvalid               : in    std_ulogic;
.....
    );
end oR_Role;
```

openRole: ideas for a standard – FPGA side

- solely based on **AMBA AXI4**
- **one** defined interface (*“API”*)
- bus-width adaption done with AXI Interconnects (“Template parameter”)
- Interface includes:
  - stream based and address based communication
  - control registers (and virtual interrupts)
- The use of a standard AXI-based interface allows the straightforward integration of:
  - debug probes
  - performance profilers

openRole: ideas for a standard – CPU side

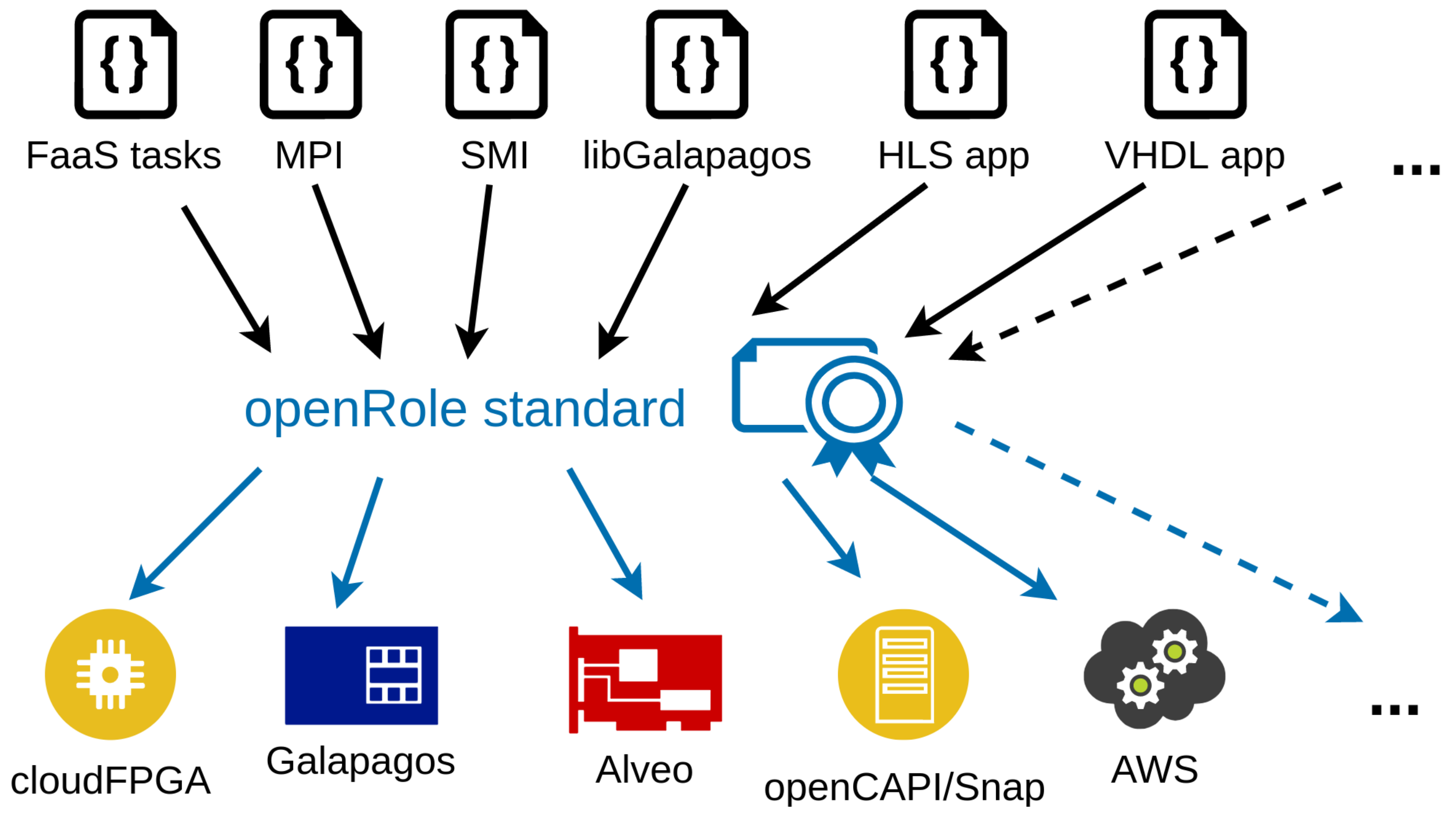
- every FPGA Role/app needs some control by the platform or communication with some SW app
- The app developer is only interested in the interface between SW app and FPGA Role, *not how the data gets there*
- We need also an unified SW interface, e.g.:
  - **oR_configure**(…)
  - **oR_write** & **oR_read** (address based)
  - **oR_send** & **oR_receive** (stream based)
- All *Roles* should be identified with an integer **role_id** (not PCIe address, or IP address, etc.) to have common *meta data*



```
oR_return oR_configure(oR_role_id role_id, const oR_word *config_to_write);

oR_return oR_write(oR_role_id role_id, const oR_word *data, oR_address start_address, oR_len length);

oR_return oR_read(oR_role_id role_id, oR_word *data, oR_address start_address, oR_len length);

oR_return oR_send(oR_role_id role_id, const oR_word *data, oR_len length);

oR_return oR_recv(oR_role_id role_id, oR_word *data, oR_len length);
```

Summary:
The openRole proposal

- Enable **“design once, run everywhere”** for heterogeneous CPU-FPGA platforms
- **NOT** an implementation, it’s a standard for abstracting configuration and communication from a particular HW
- **NOT** a programming model, it is about portable FPGA designs
- requires a “compiler” for each platform
- but does **not** require changes to the code
- published as “Header Files”, “VHDL entities” and “PDFs” by the **community**
- **interested in shaping openRole?** You would do it completely different?
  ➔ **contact us**



Thank you…

Burkhard Ringlein

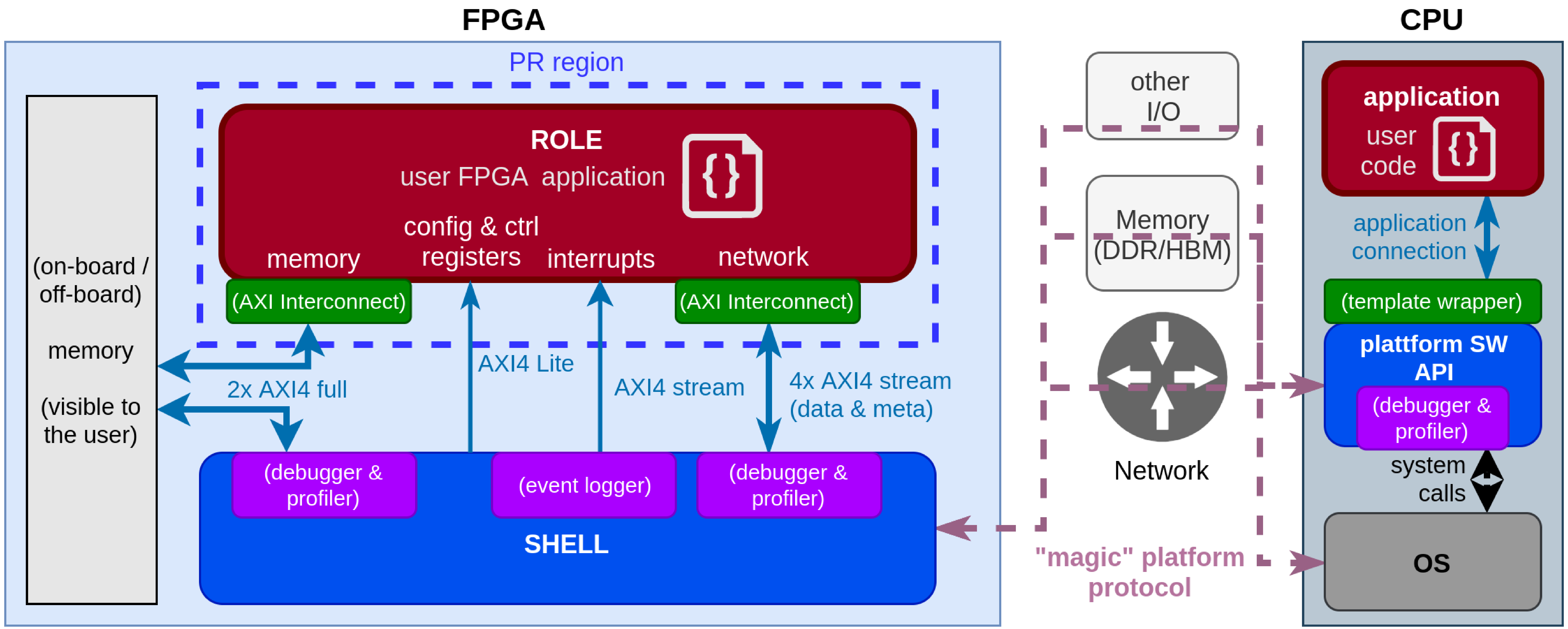
ngl@zurich.ibm.com

zurich.ibm.com/cci/cloudFPGA/

@0xcaffee

Appendix

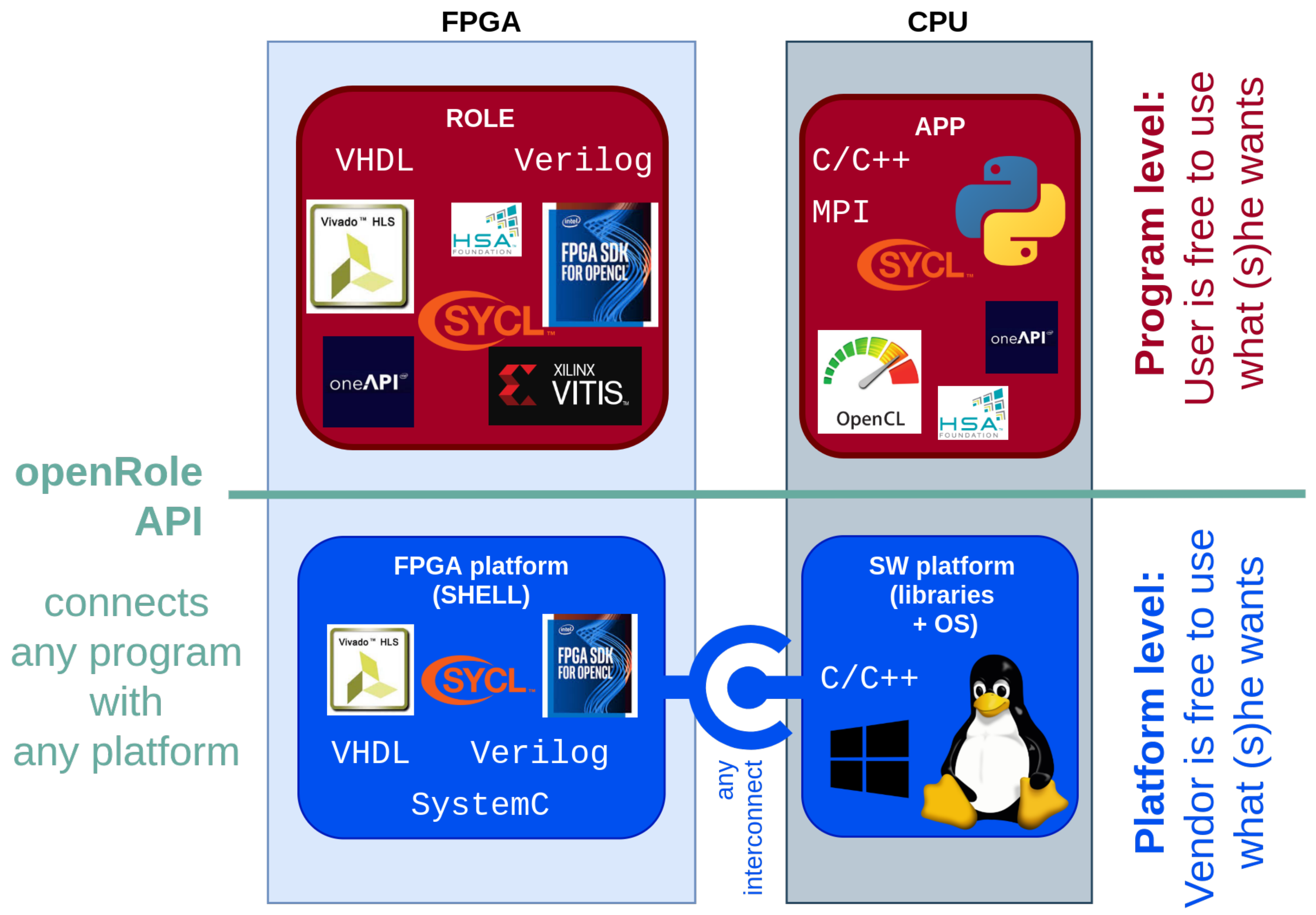
openRole: ideas for a standard – system view



openRole should **not** limit languages or tools

openRole is also **not** a programming model or a transport protocol

...it **just** connects arbitrary programs with arbitrary platforms.



References

[1] https://github.com/aws/aws-fpga/blob/master/hdk/docs/AWS_Shell_Interface_Specification.md#ShellInterfaces
[2] https://opencapi.github.io/oc-accel-doc/
[3] https://www.xilinx.com/applications/data-center.html
[4] https://commons.wikimedia.org/wiki/File:Linux_kernel_API.svg
[5] https://doi.org/10.1007/978-3-319-92792-3_2
[6] https://commons.wikimedia.org/wiki/File:Dieric_Bouts_013.jpg
[7] https://developer.xilinx.com/en/articles/acceleration-basics.html
[8] US20190190847A1: Allocating acceleration component functionality for supporting services
[9] http://xillybus.com/downloads/xillybus_product_brief.pdf
[10] https://www.nextplatform.com/2020/01/13/on-the-spearpoint-of-fpga-and-the-cloud/
[11] https://www.nextplatform.com/2020/01/14/the-inevitability-of-fpgas-in-the-datacenter/
[12] http://sc19.supercomputing.org/proceedings/bof/bof_pages/bof115.html
[13] https://h2rc.cse.sc.edu
[14] https://doi.org/10.1109/FPL.2019.00054
[15] https://cloud.baidu.com/doc/FPGA/s/Ajwvyh11e
[16] https://doi.org/10.1109/HPEC.2019.8916526
[17] https://inaccel.com

All remaining images are from IBM DAM or IBM Websites or created by the author.

cloudFPGA: Further Reading

- B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner and D. Fey, “ZRLMPI: A Unified Programming Model for Reconfigurable Heterogeneous Computing Clusters” in 28th IEEE International Symposium On Field-Programmable Custom Computing Machines (FCCM), 2020.
- B. Ringlein, F. Abel, A. Ditter, B. Weiss, C. Hagleitner and D. Fey, “ System architecture for network-attached FPGAs in the cloud using partial reconfiguration,” in 29th International Conference on Field Programmable Logic and Applications (FPL), 2019.
- F. Abel, J. Weerasinghe, C. Hagleitner, B. Weiss, S. Paredes, “An FPGA Platform for Hyperscalers,” in IEEE 25th Annual Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, pp. 29–32, 2017.
- Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, “Disaggregated FPGAs: Network performance comparison against bare-metal servers, virtual machines and Linux containers,” in IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Luxembourg, 2016.
- J. Weerasinghe, R. Polig, F. Abel, “Network-attached FPGAs for data center applications,” in IEEE International Conference on Field-Programmable Technology (FPT ’16), Xian, China, 2016.
- J. Weerasinghe, F. Abel, C. Hagleitner, A. Herkersdorf, “Enabling FPGAs in hyperscale data centers,” in IEEE International Conference on Cloud and Big Data Computing (CBDCom), Beijing, China, pp. 1078–1086, 2015.
- F. Abel, “How do you squeeze 1000 FPGAs into a DC rack?” online at LinkedIn
- The **cloudFPGA project page at ZRL:** https://www.zurich.ibm.com/cci/cloudFPGA/