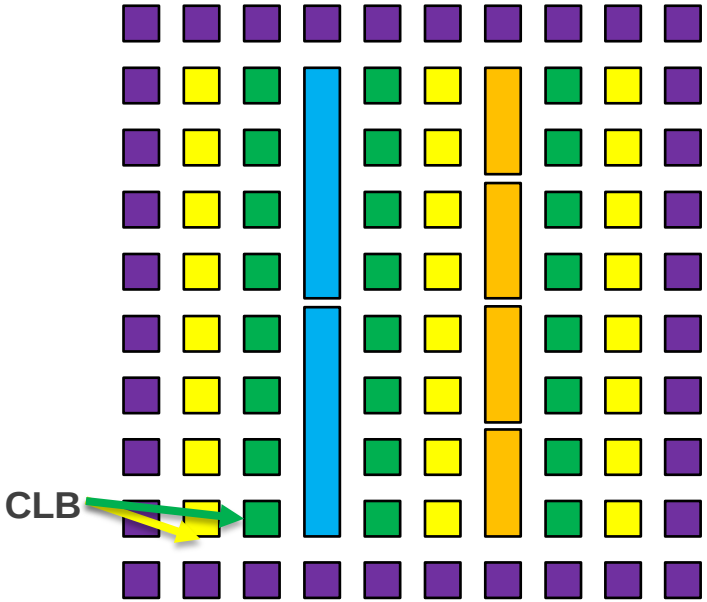
OpenPOWER™

Network Protocol Acceleration With CAPI SNAP

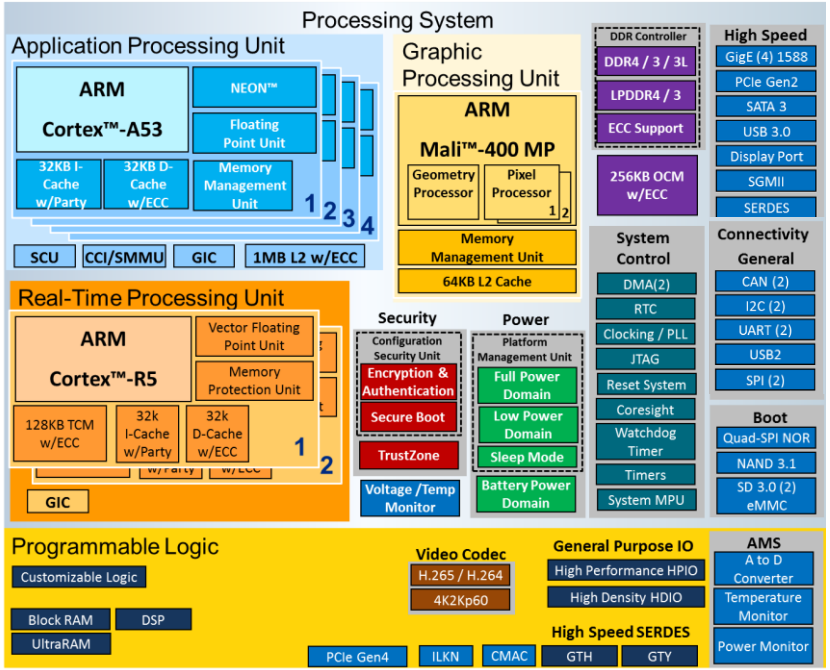
Dr. Endric Schubert, MLE
Dr. Andrew McCormick, Alpha-Data

FPGAs Are Not Just Gate-Arrays Anymore

FPGA 1990 - 2010

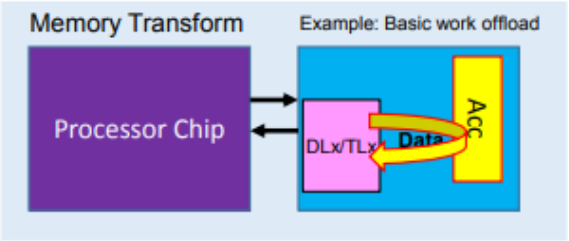


FPGA 2016 - ...



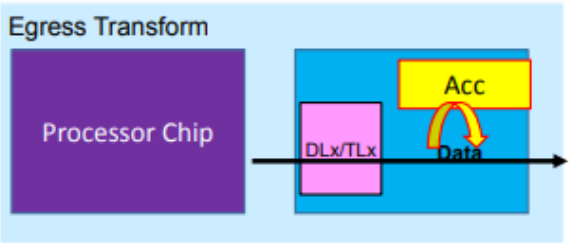
Acceleration Paradigms with Great Performance

OpenCAPI



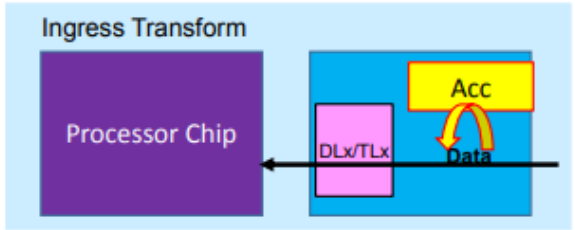
OpenCAPI is ideal for acceleration due to Bandwidth to/from accelerators, best of breed latency, and flexibility of an Open architecture

Examples: Machine or Deep Learning such as Natural Language processing, sentiment analysis or other Actionable Intelligence using OpenCAPI attached memory



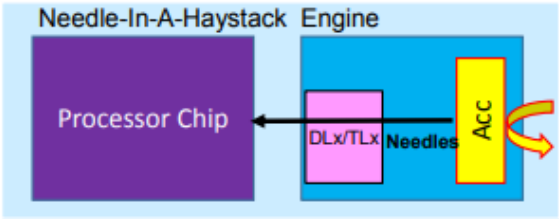
100 GigE?

Examples: Encryption, Compression, Erasure prior to delivering data to the network or storage



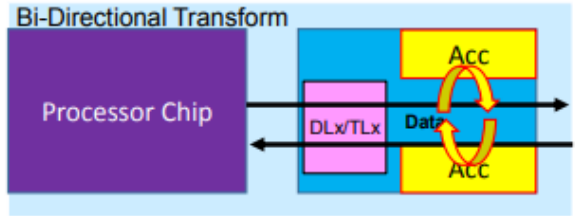
100 GigE?

Examples: Video Analytics, Network Security, Deep Packet Inspection, Data Plane Accelerator, Video Encoding (H.265), High Frequency Trading etc



Large Haystack Of Data

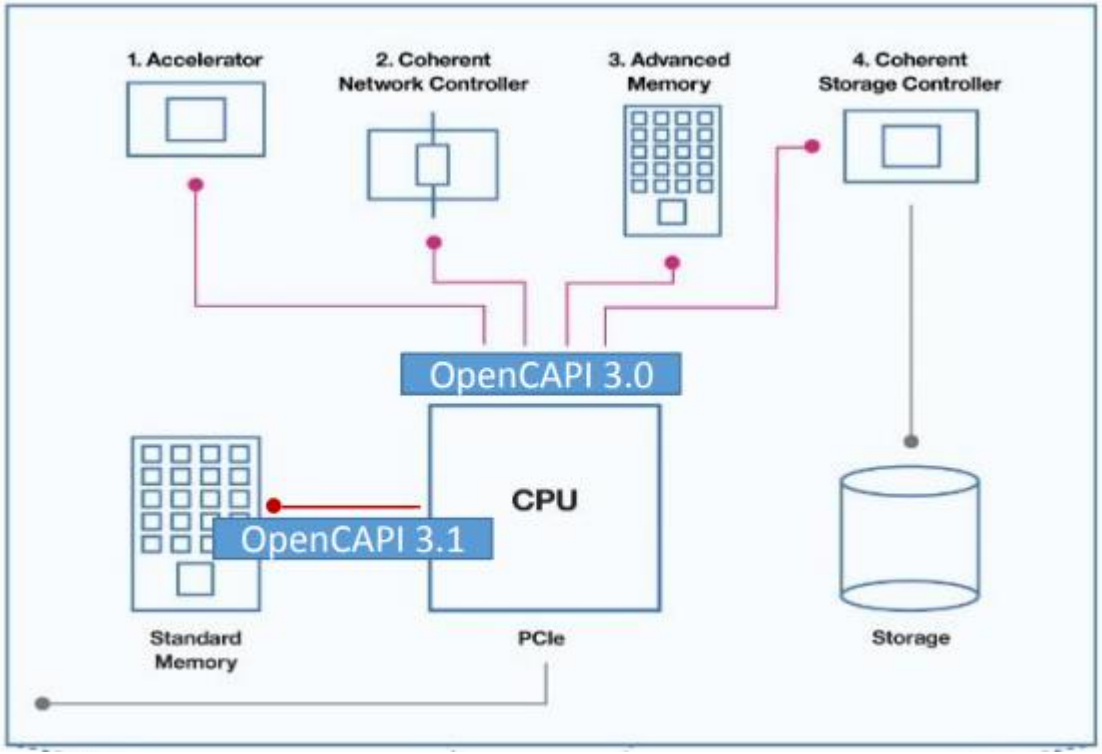
Examples: Database searches, joins, intersections, merges
Only the Needles are sent to the processor

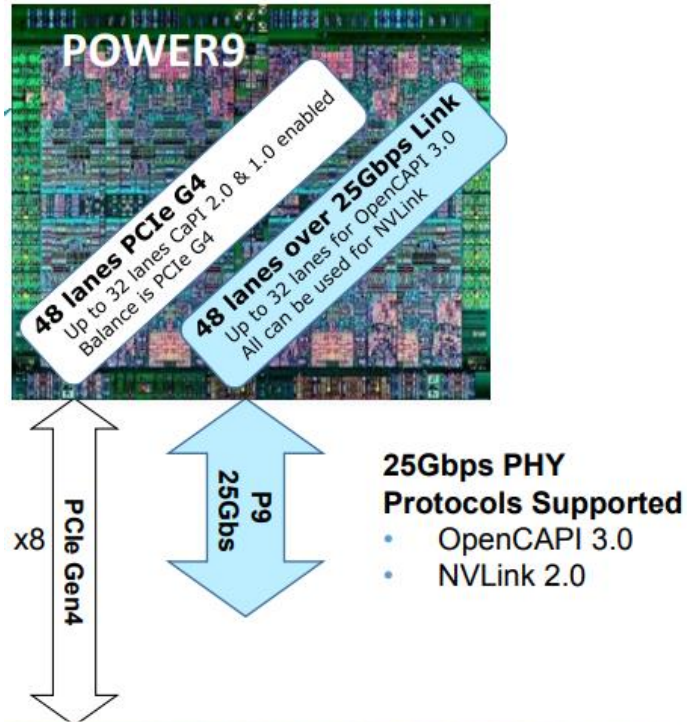


100 GigE?

Examples: NoSQL such as Neo4J with Graph Node Traversals, etc

CAPI - Coherent Accelerator Processor Interface

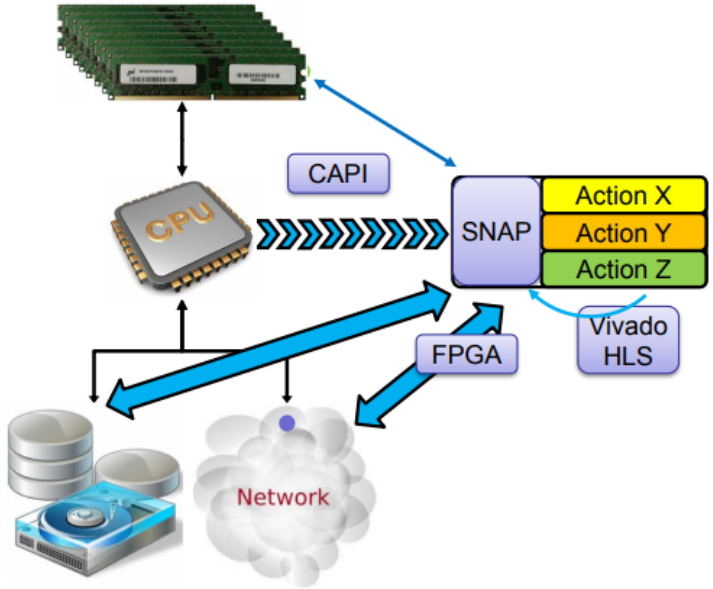




SNAP - Storage, Networking, Analytics Platform

The CAPI – SNAP concept

OpenPOWER



CAPI — FPGA becomes a peer of the CPU
➔ Action **directly** accesses host memory

\+

SNAP — Manage server threads and actions
Manage access to IOs (memory, network)
➔ Action **easily** accesses resources

\+

FPGA — Gives on-demand compute capabilities
Gives direct IOs access (storage, network)
➔ Action **directly** accesses external resource

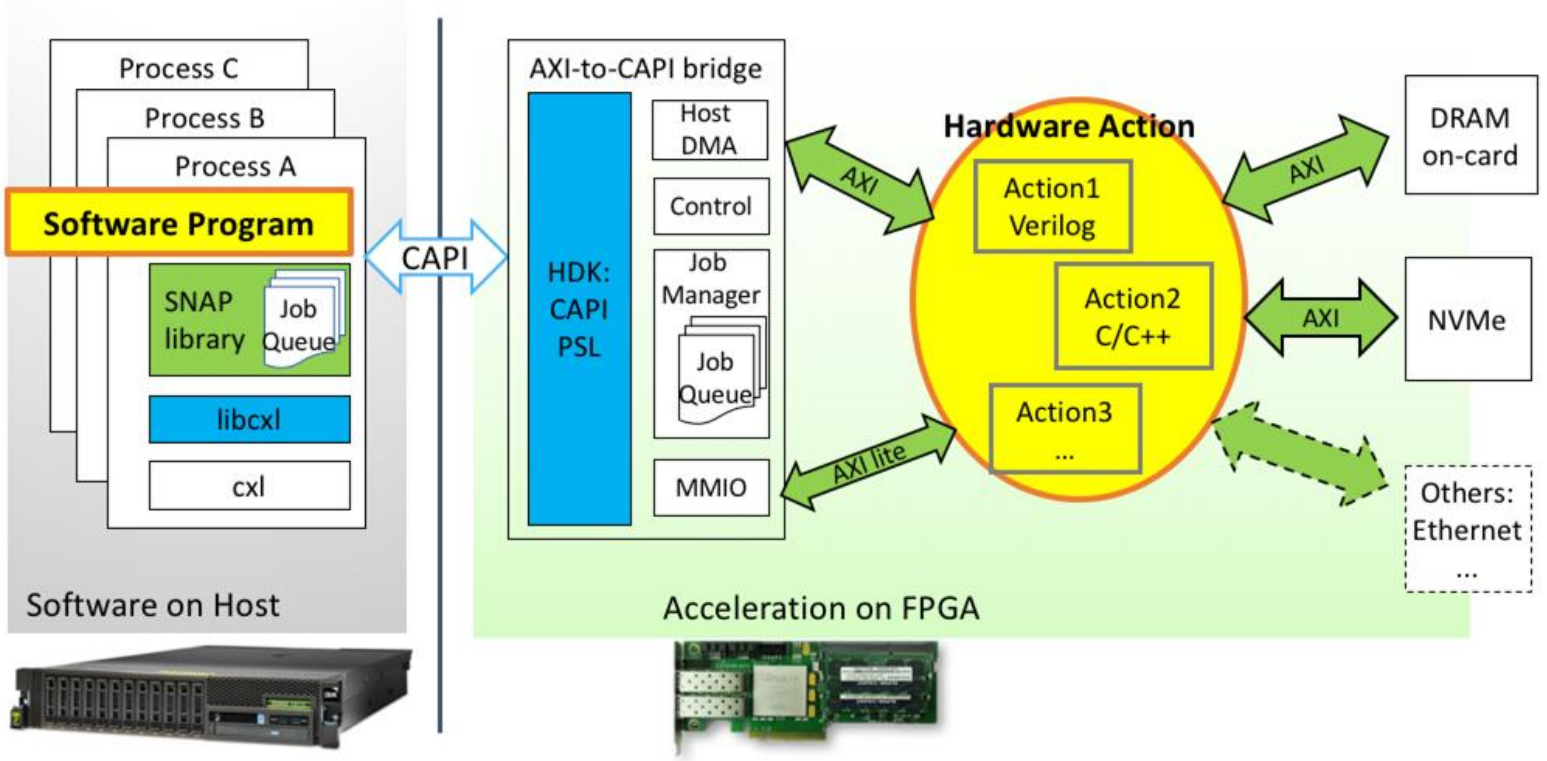
\+

Vivado HLS — Compile Action written in C/C++ code
Optimize code to get performance
➔ Action code **can be ported efficiently**

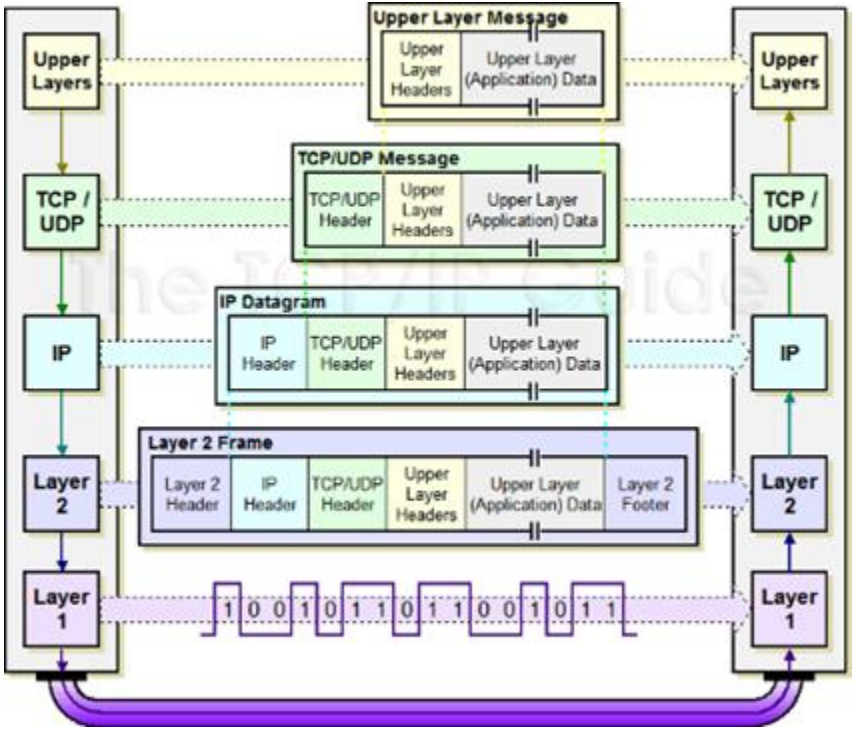
=

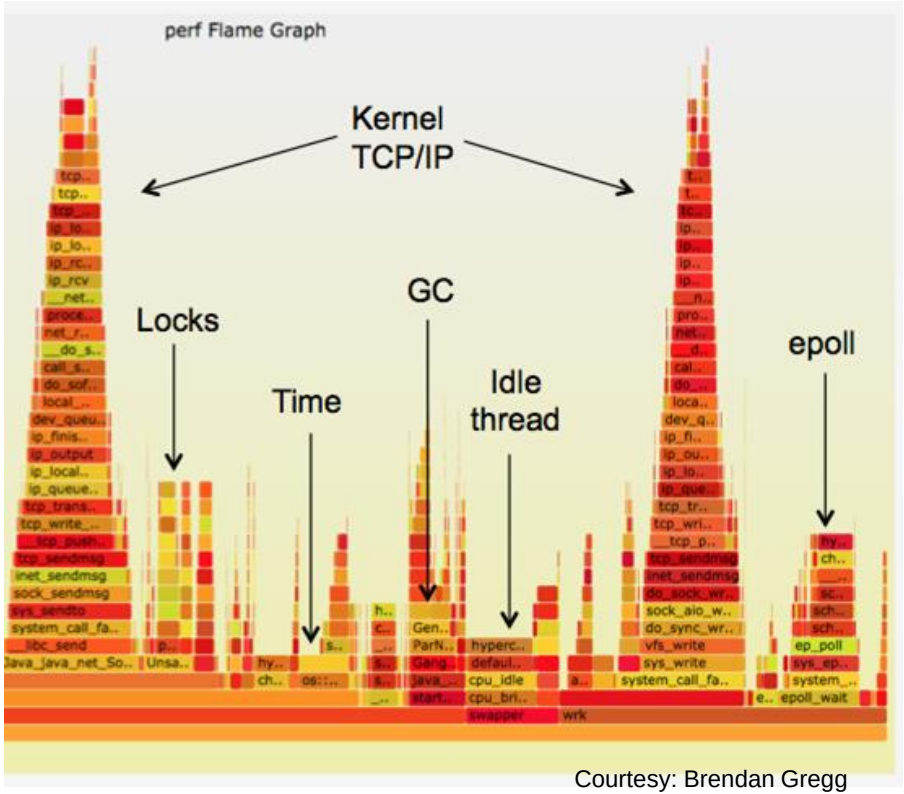
Best way to **offload/accelerate** a C/ C++ code with :
- Minimum change in code
- Quick porting
- Better performance than CPU

CAPI SNAP Framework



Computational Aspects of Network Protocols

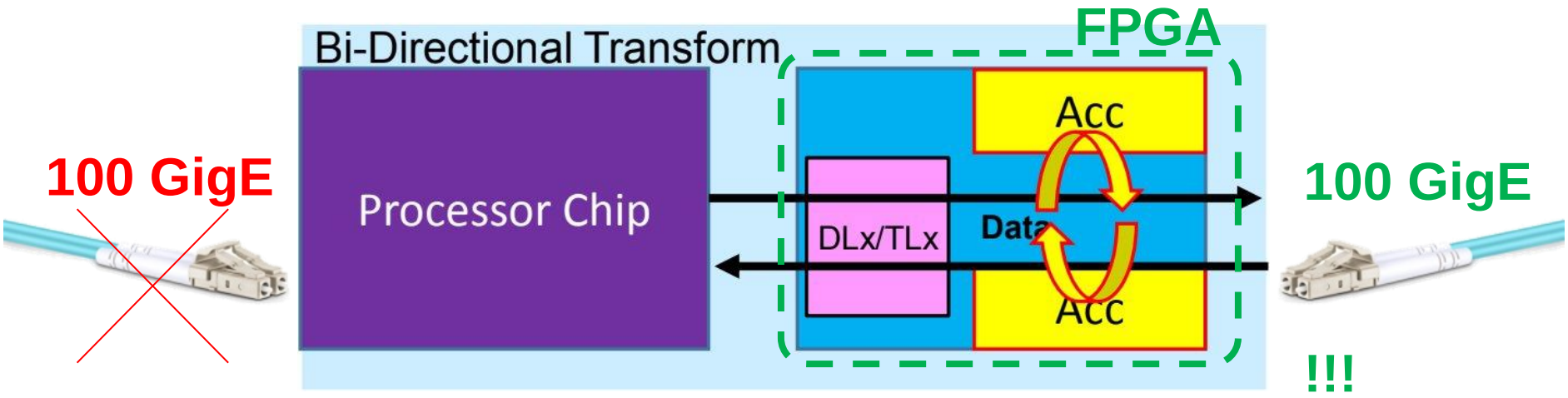




Courtesy: Brendan Gregg

Architecture Proposal for CAPI Smart NIC

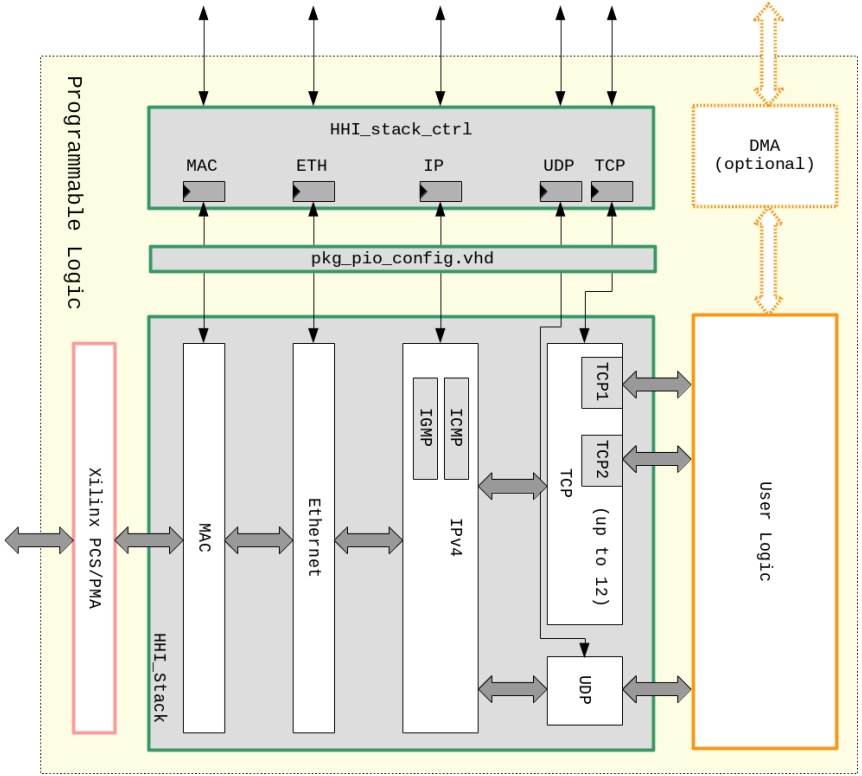
+ Offload CPUs
+ Free up PCIe bus
+ Enable “on-the-fly” egress/ingress/bi-dir processing



Capabilities of MLE / Fraunhofer HHI Technology

Network Protocol Accelerator Platform (NPAP)

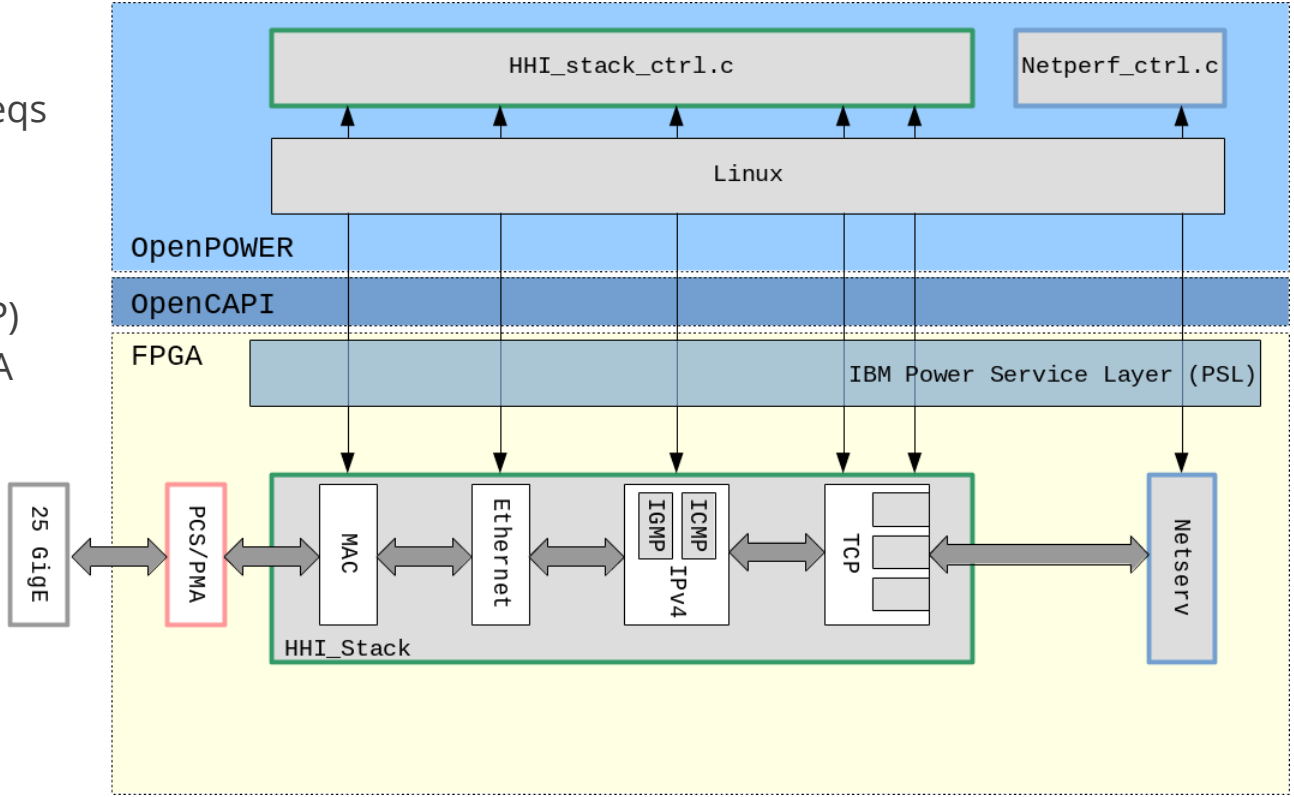
- Modular TCP/UDP/IP stack implementation in synthesizable HDL
- Parameterizable datapath architectures
  - 128-bit for 10/25 GigE (now)
  - 128-bit for 40/50 GigE (4Q18)
  - 512-bit for 100 GigE (1Q19)
- Multiple scalable single TCP session engines for scalable processing
- Network Interface Card (NIC) functionality with Bypass (optional)
- Session scaler to 10k+ sessions (roadmap)



Split between SNAP Action and SNAP Shell

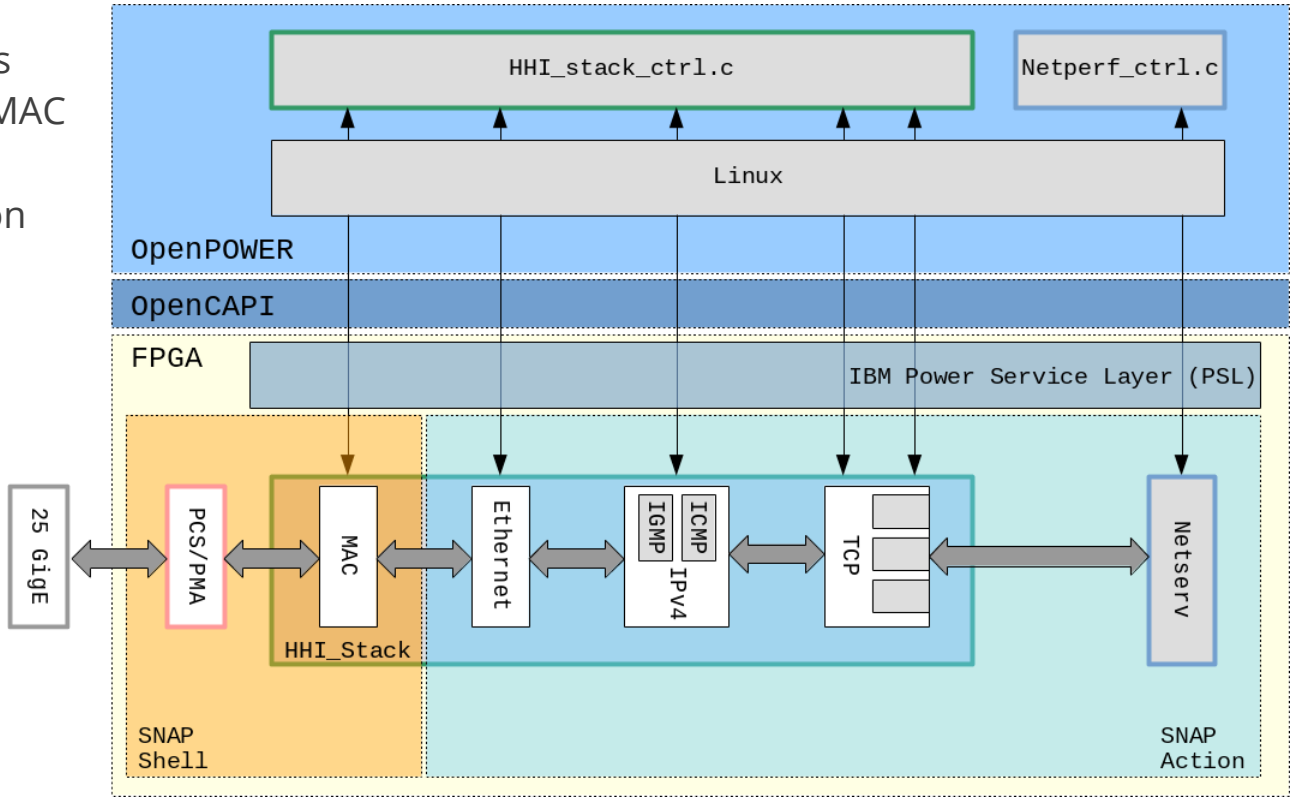
- Explore Use-case
- Adjust to License Reqs

- PCS/PMA
- MAC
- Internet Protocol (IP)
- TCP/UDP Offload/FA



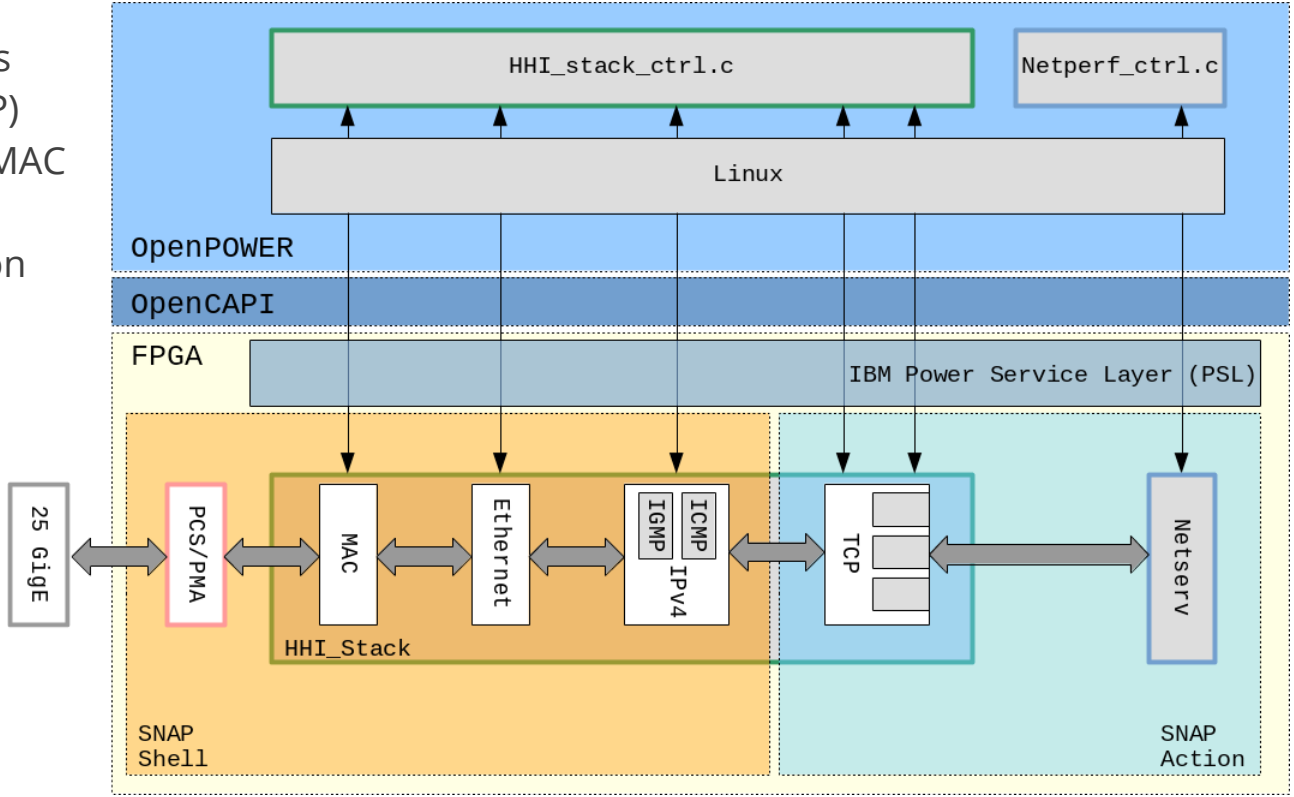
Deliver as SNAP Action – Ethernet

- SNAP Shell supports 10/25/40/50/100G MAC
- Fraunhofer HHI NPAP as SNAP Action



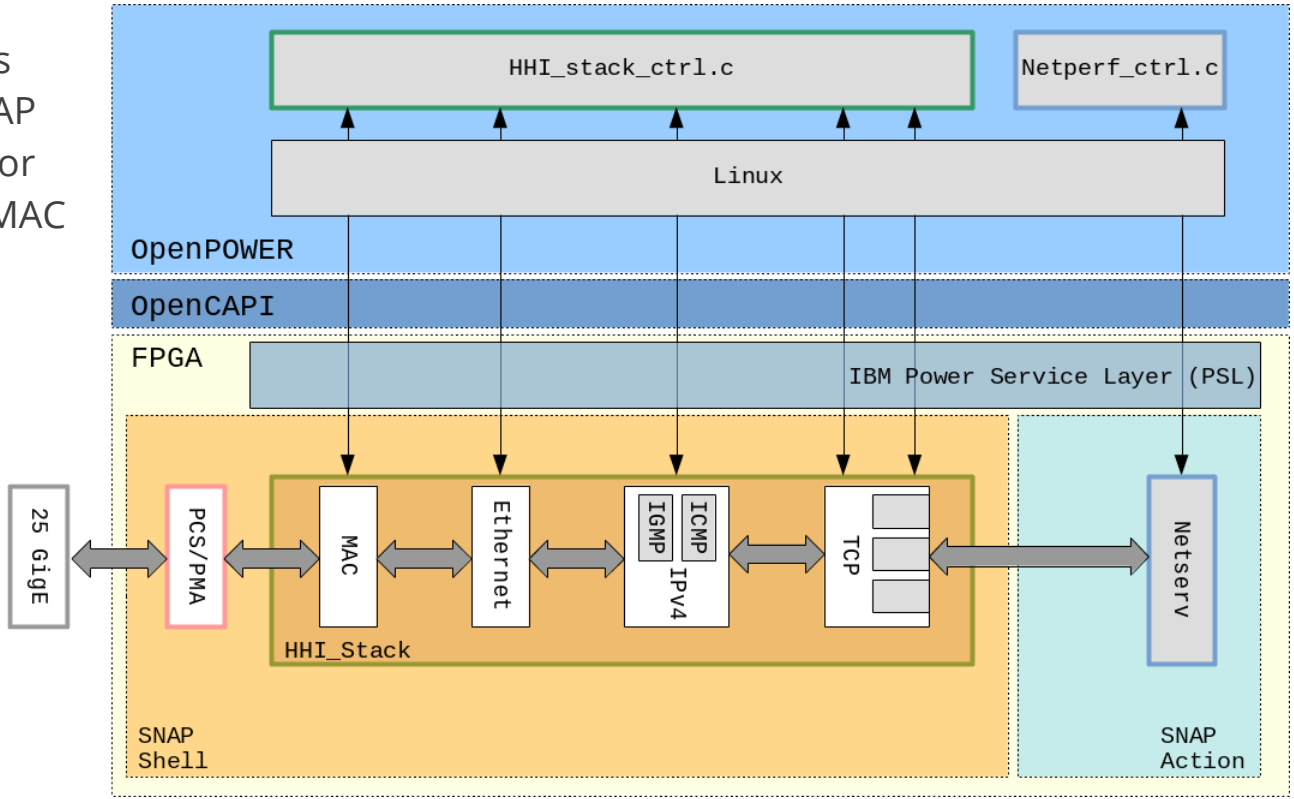
Deliver as SNAP Action – IP

- SNAP Shell supports Internet Protocol (IP) 10/25/40/50/100G MAC
- Fraunhofer HHI NPAP as SNAP Action



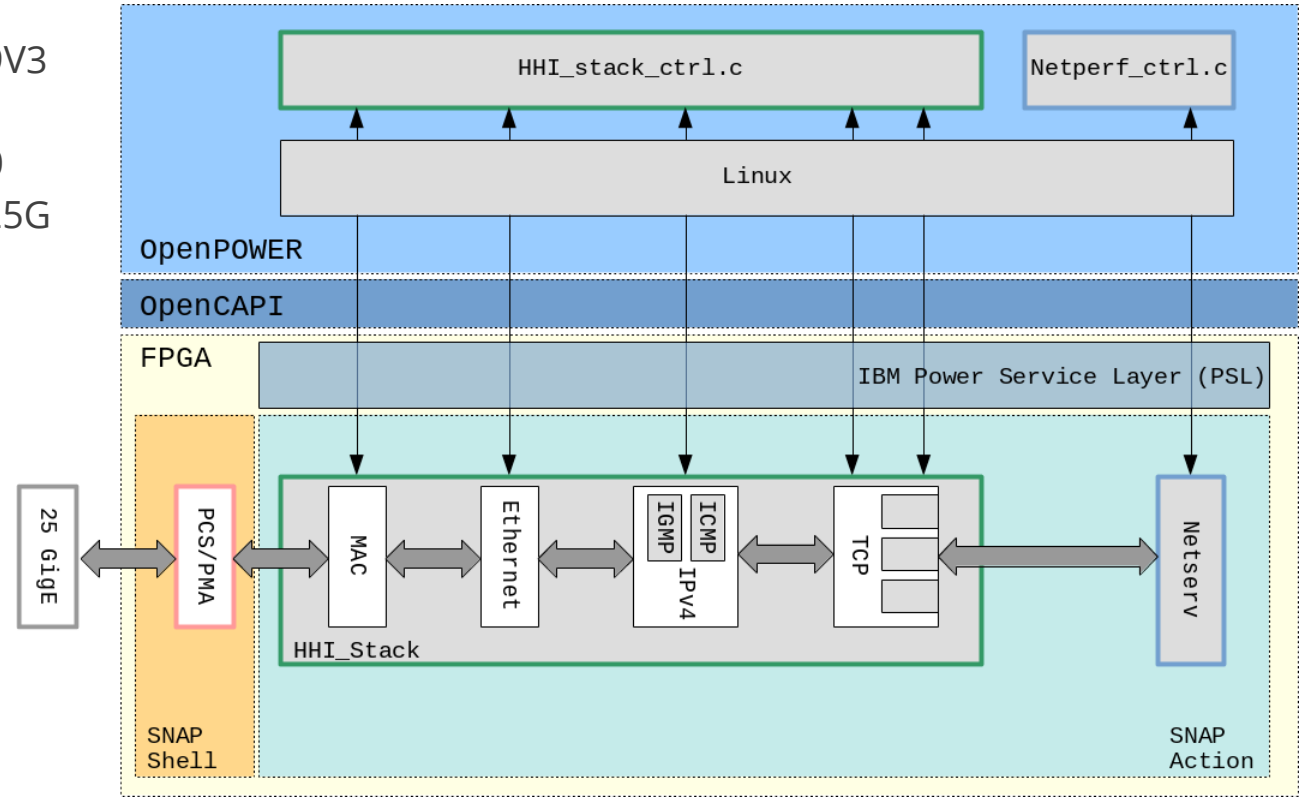
Deliver as SNAP Action – TCP/UDP

- SNAP Shell supports Fraunhofer HHI NPAP as Full Accelerator for 10/25/40/50/100G MAC
- SNAP Action runs Netperf 2.6.0 for testing



Current PoC Implementation

- AlphaData KU3 ⇒ 9V3
- Power8 ⇒ Power9
- CAPI 1.0 ⇒ CAPI 2.0
- MLE NPAP-10G ⇒ 25G
- Testbench w/ Netperf and Loopbacks



Network Proto Acceleration with CAPI SNAP

```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-33-generic ppc64le)
user@power8:~$
user@power8:~$ cd snap.git/software/tools/
```

Network Proto Acceleration with CAPI SNAP

```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-33-generic ppc64le)
user@power8:~$
user@power8:~$ cd snap.git/software/tools/
user@power8:~/snap.git/software/tools$
user@power8:~/snap.git/software/tools$ sudo ./snap_maint -v
SNAP Card Id: 0 Name: ADKU3. NVME disabled, ETH 10G PCS/PMA enabled, 0 MB DRAM available.
(Align: 64 Min_DMA: 1)
SNAP FPGA Release: v1.5.0 Distance: 83 GIT: 0x36e85e51
SNAP FPGA Build (Y/M/D): 2018/08/18 Time (H:M): 16:34
SNAP FPGA CIR Master: 1 My ID: 0
SNAP FPGA Up Time: 172 sec
   0 Max AT: 1 Found AT: 0x22db0001 --> Assign Short AT: 0
     0      0x22db0001      0x00000000  MLE HDL 10G Ethernet TCP/UDP/IP Accelerator Demo
```

Network Proto Acceleration with CAPI SNAP

```
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.15.0-33-generic ppc64le)
user@power8:~$
user@power8:~$ cd snap.git/software/tools/
user@power8:~/snap.git/software/tools$
user@power8:~/snap.git/software/tools$ sudo ./snap_maint -v
SNAP Card Id: 0 Name: ADKU3. NVME disabled, ETH 10G PCS/PMA enabled, 0 MB DRAM available.
(Align: 64 Min_DMA: 1)
SNAP FPGA Release: v1.5.0 Distance: 83 GIT: 0x36e85e51
SNAP FPGA Build (Y/M/D): 2018/08/18 Time (H:M): 16:34
SNAP FPGA CIR Master: 1 My ID: 0
SNAP FPGA Up Time: 172 sec
   0 Max AT: 1 Found AT: 0x22db0001 --> Assign Short AT: 0
     0     0x22db0001     0x00000000  MLE HDL 10G Ethernet TCP/UDP/IP Accelerator Demo

user@power8:~/snap.git/software/tools$ cd ../../actions/hdl_eth_npap_demo/sw/

user@power8:~/snap.git/actions/hdl_eth_npap_demo/sw$ sudo ./ifconfig
npap0       Link encap: npap   HW addr: 02:00:00:00:00:11
            inet address: 192.168.1.101   Bcast: 192.168.1.255   Mask: 255.255.255.0
```

Netperf 2.6.0 Performance Analysis

Intel X520 in Power 8 8247-21L Ubuntu 16.04
NPAP-10G on AlphaData KU3

```
$ timeout 15 netperf -H 192.168.1.101 -t TCP_STREAM
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.101 () port 0 AF_INET
: demo
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec
87380  16384   16384    10.00    9491.13
$ timeout 15 netperf -H 192.168.1.101 -t TCP_RR
MIGRATED TCP REQUEST/RESPONSE TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.101 () port
0 AF_INET : demo : first burst 0
Local /Remote
Socket Size   Request  Resp.   Elapsed  Trans.
Send   Recv   Size     Size    Time     Rate
bytes  Bytes  bytes    bytes   secs.    per sec
16384  87380  1        1       10.00    43345.98
```

ALPHA DATA

CAPI SNAP Networking and Acceleration Boards for Power8

ADM-PCIE-KU3

Entry level CAPI SNAP card
Xilinx KU060 FPGA
Dual QSFP+ Interfaces
Up to 8x10 GE or 2x40 GE ports
16/32GB DDR3 DIMMS

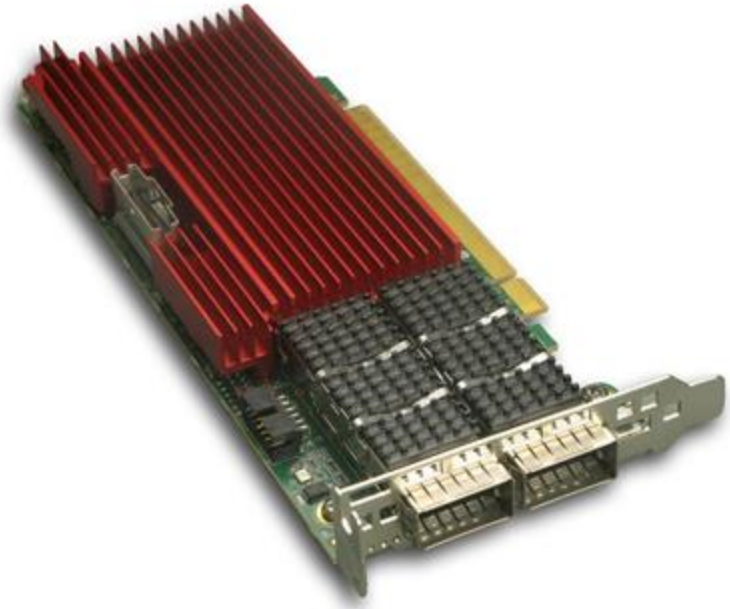
ADM-PCIE-8K5

High Performance CAPI SNAP card
Xilinx KU115 FPGA
Maximum DSP(ML) performance in 20ns
Silicon/Low Profile Form Factor
Dual SFP+ Interfaces : 2x10 GE
16/32GB DDR4

ALPHA DATA

CAPI/OpenCAPI FPGA Networking and Acceleration Board for Power9

ADM-PCIE-9V3

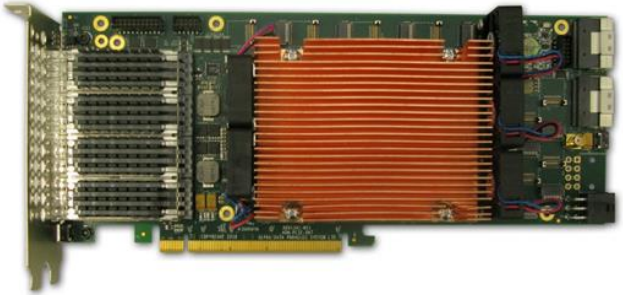


Low Profile PCIe Form Factor
Xilinx VU3P FPGA
CAPI 2.0/PCIe Gen4x8/Gen3x16
OpenCAPI 25Gb/s x8 Link
Dual QSFP28
Up to 8x10G/25G or 2x40G/100G
16/32GB DDR4

ALPHA DATA

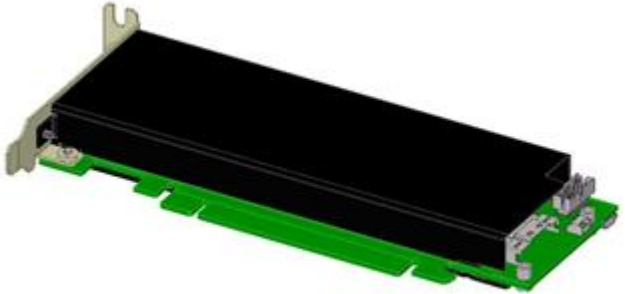
CAPI/OpenCAPI HBM Boards for Power9

ADM-PCIE-9H7



Full Height ¾ Length PCIe form factor
GPU Size, 150/225W power budget
Xilinx VU37P HBM FPGA
4xQSFP28 (4x100GE)
8xFirefly (32x28Gbps)
2xOpenCAPI links
PCIe Gen3x16/Dual Gen4x8/CAPI 2.0
8GB of 460GB/s HBM

ADM-PCIE-9H3



Low Profile PCIe form factor
Xilinx VU33P HBM FPGA
QSFP-DD (2x100GE)
OpenCAPI link
PCIe Gen3x16/Dual Gen4x8/CAPI 2.0
8GB of 460GB/s HBM

Call for Participation

- CAPI is a significant technology advancement for FPGA accelertion!
- SNAP is a fantastic acceleration design framework…
  now supports networking functionality!

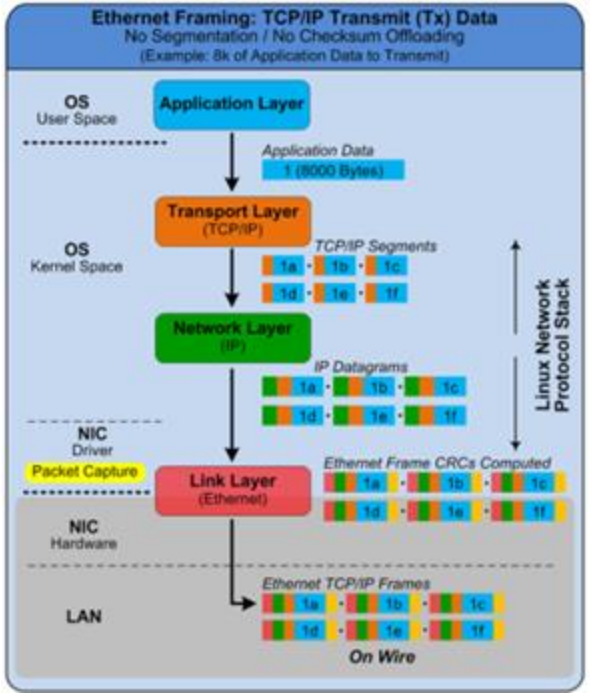
- AlphaData, IBM, MLE are looking forward to participation from the OpenPOWER ecosystem to guide further development!

Alpha Data www.alpha-data.com
Missing Link Electronics www.MissingLinkElectronics.com

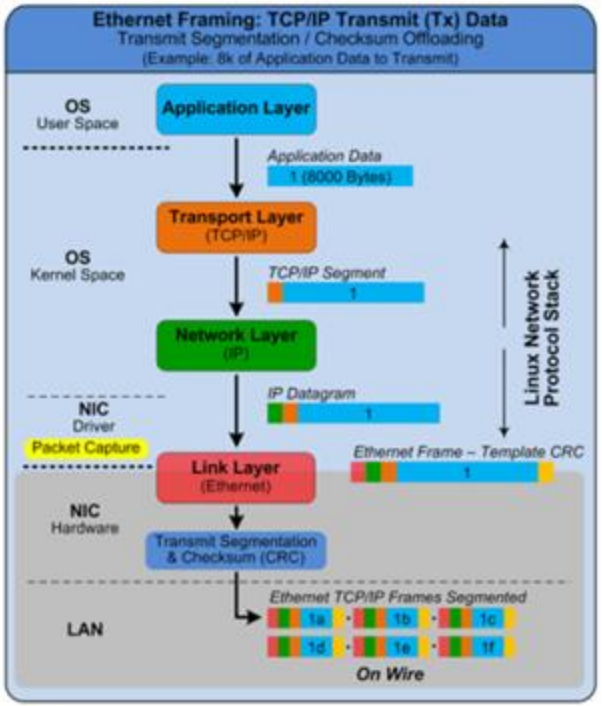
BACKUP

Network Protocol Acceleration

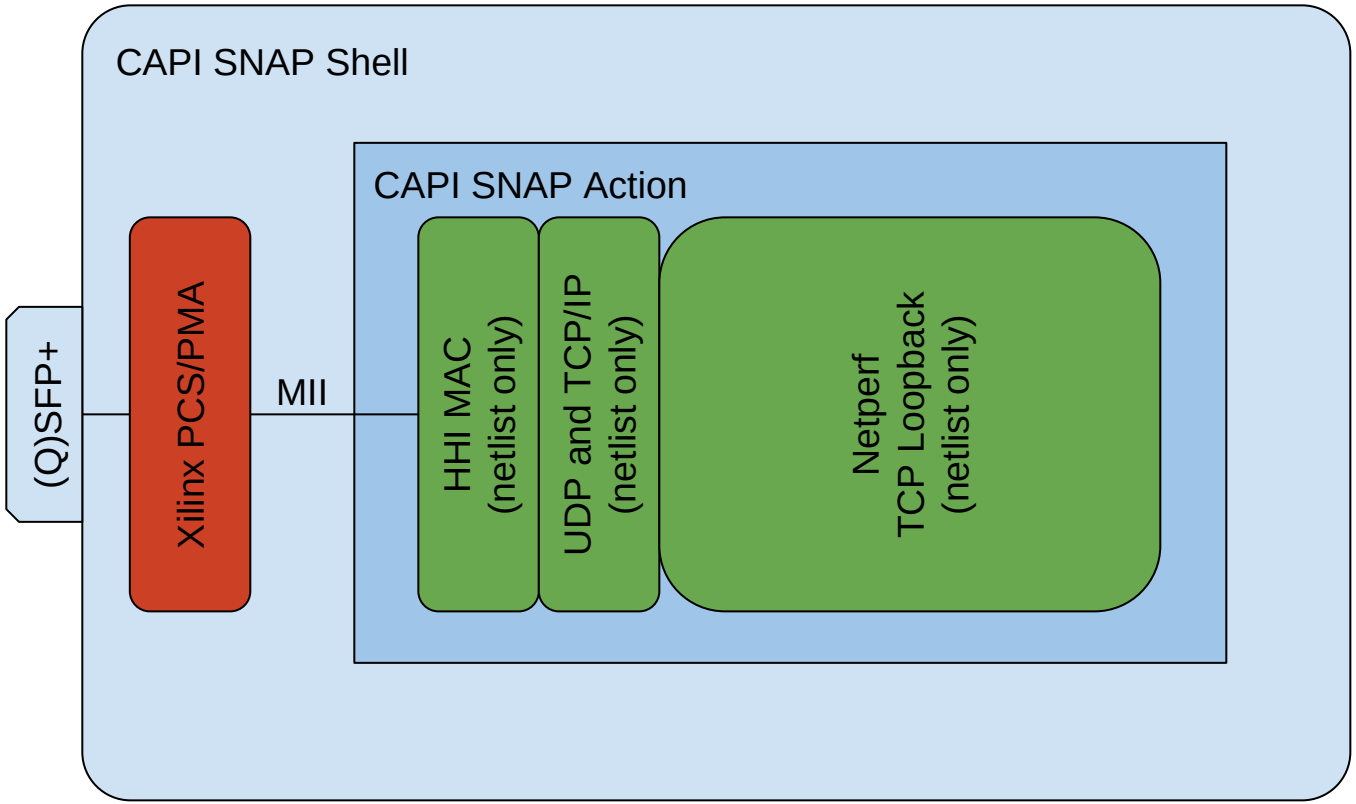
All software



FPGA Acceleration

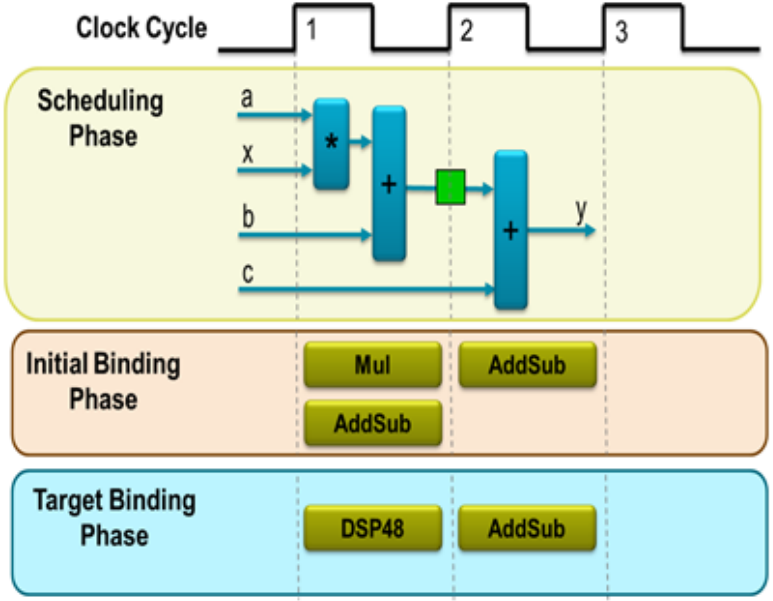


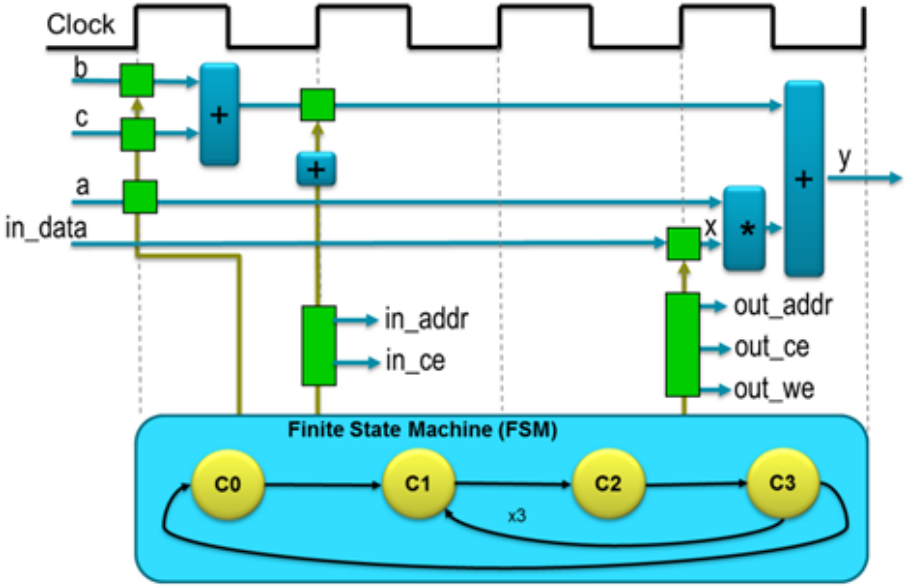
Network Options for CAPI SNAP



HOW? High-Level Synthesis Works

- Design automation runs scheduling and resource allocation to generate RTL code comprising data path plus state machines for control.



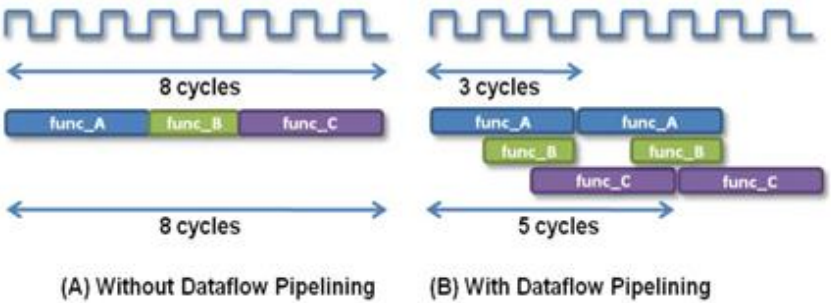


HOW? Offloading Software Functions to FPGA

- Automatic performance optimization via parallelization at dataflow level

```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```



(A) Without Dataflow Pipelining (B) With Dataflow Pipelining

- Automatic interface synthesis and code generation for variety of real-life HW/SW connectivity