Sensor Fusion and Data-in-Motion Processing for Autonomous Vehicles

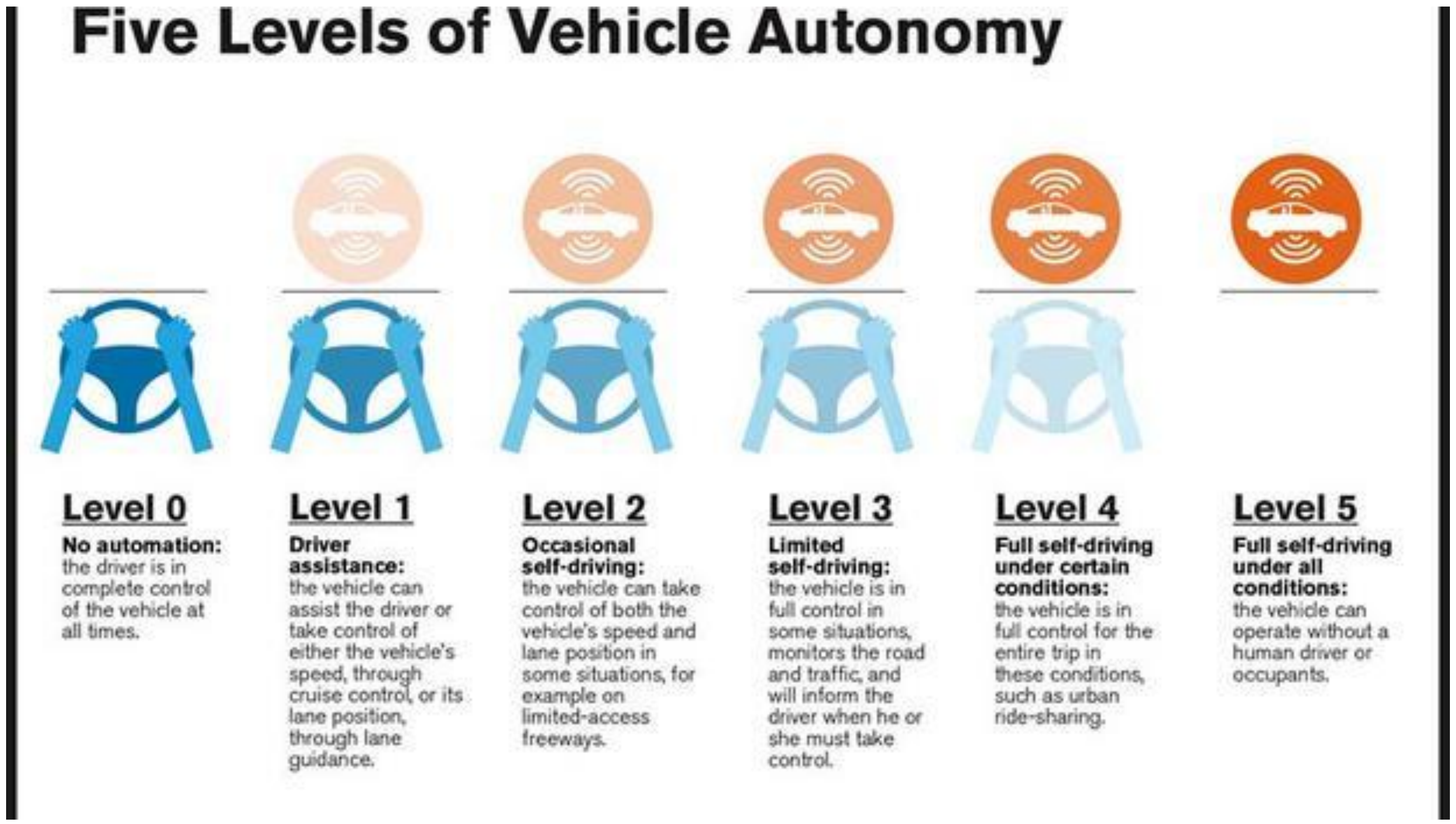
Endric Schubert, Ph. D.
CTO
Missing Link Electronics (MLE)

Disclaimer

Acknowledgements

- Ulrich Langenbach, Dir. Eng. – Missing Link Electronics
- Jim Peek, Dir. Technology – Missing Link Electronics

Autonomous Vehicles



Five Levels of Vehicle Autonomy

Level 0
No automation: the driver is in complete control of the vehicle at all times.

Level 1
Driver assistance: the vehicle can assist the driver or take control of either the vehicle's speed, through cruise control, or its lane position, through lane guidance.

Level 2
Occasional self-driving: the vehicle can take control of both the vehicle's speed and lane position in some situations, for example on limited-access freeways.

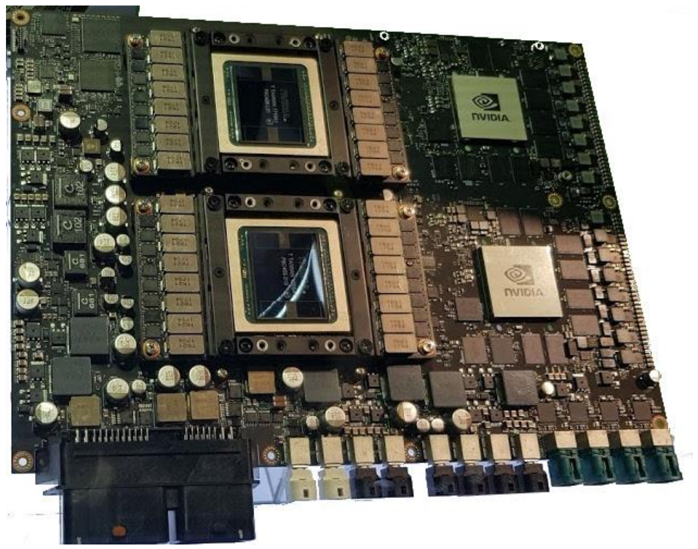
Level 3
Limited self-driving: the vehicle is in full control in some situations, monitors the road and traffic, and will inform the driver when he or she must take control.

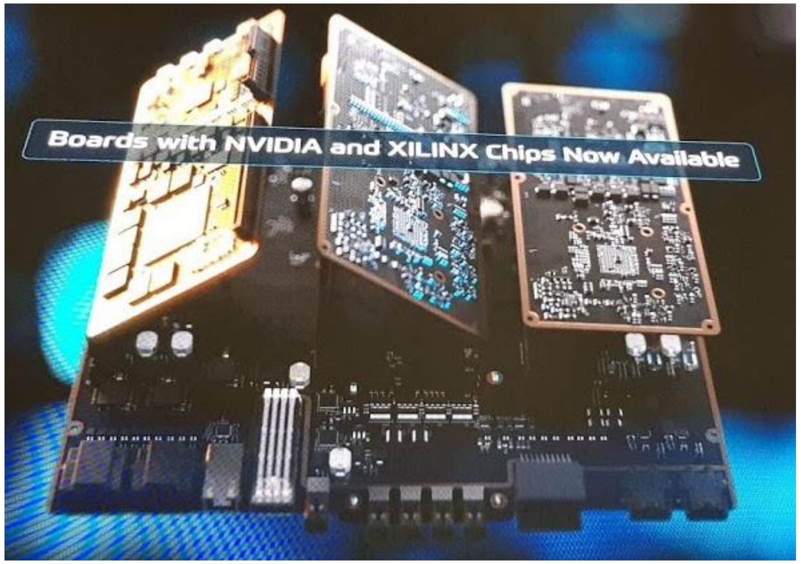
Level 4
Full self-driving under certain conditions: the vehicle is in full control for the entire trip in these conditions, such as urban ride-sharing.

Level 5
Full self-driving under all conditions: the vehicle can operate without a human driver or occupants.

Better: Automated - not autonomous - driving

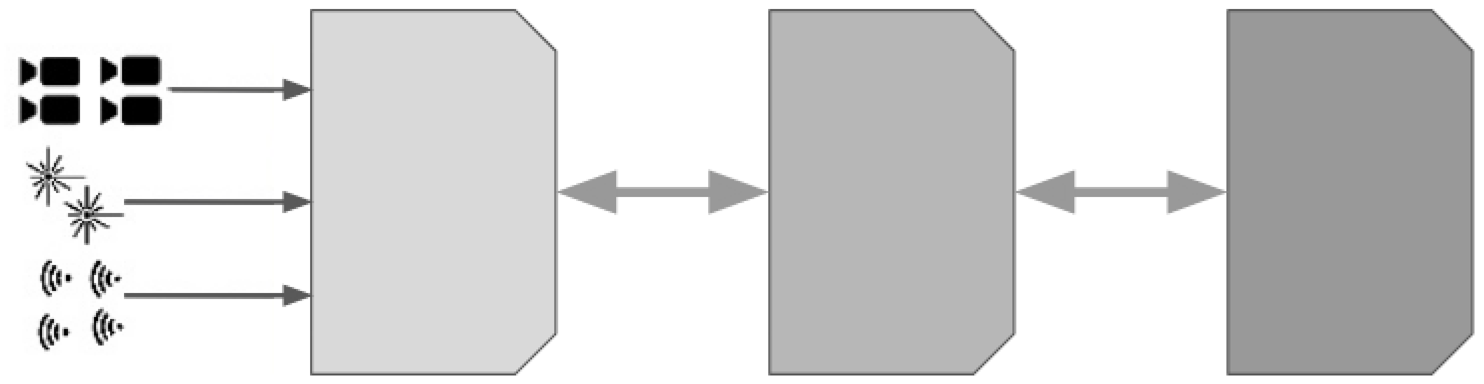
CES 2019: Multi-GPU/SoC ECUs





System Overview L4 AV

The Level-4 AV ECU Design Problem



Sensors

Sensing
- Data Fusion
- Calibration
- Object Detection

Perception & Decision Making
- Path Planning
- Safety Monitor

Vehicle Actuators

No single-chip solution available soon!

⇒ Must be implemented w/ multiple, different automotive SoCs

⇒ How to partition? How to connect?

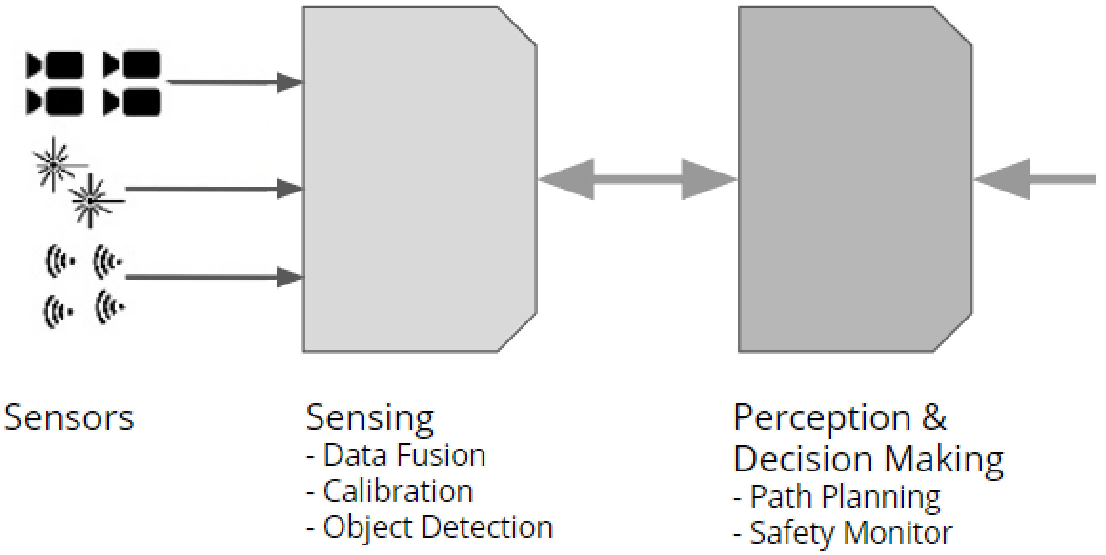
Data-in-Motion Processing

- **100 Gbps raw bandwidth, or more**

Up to 16 Cameras @2.5 Gbps

Up to 4 Radar @ 1..10 Gbps

Up to 4 Lidar @ 1..10 Gbps

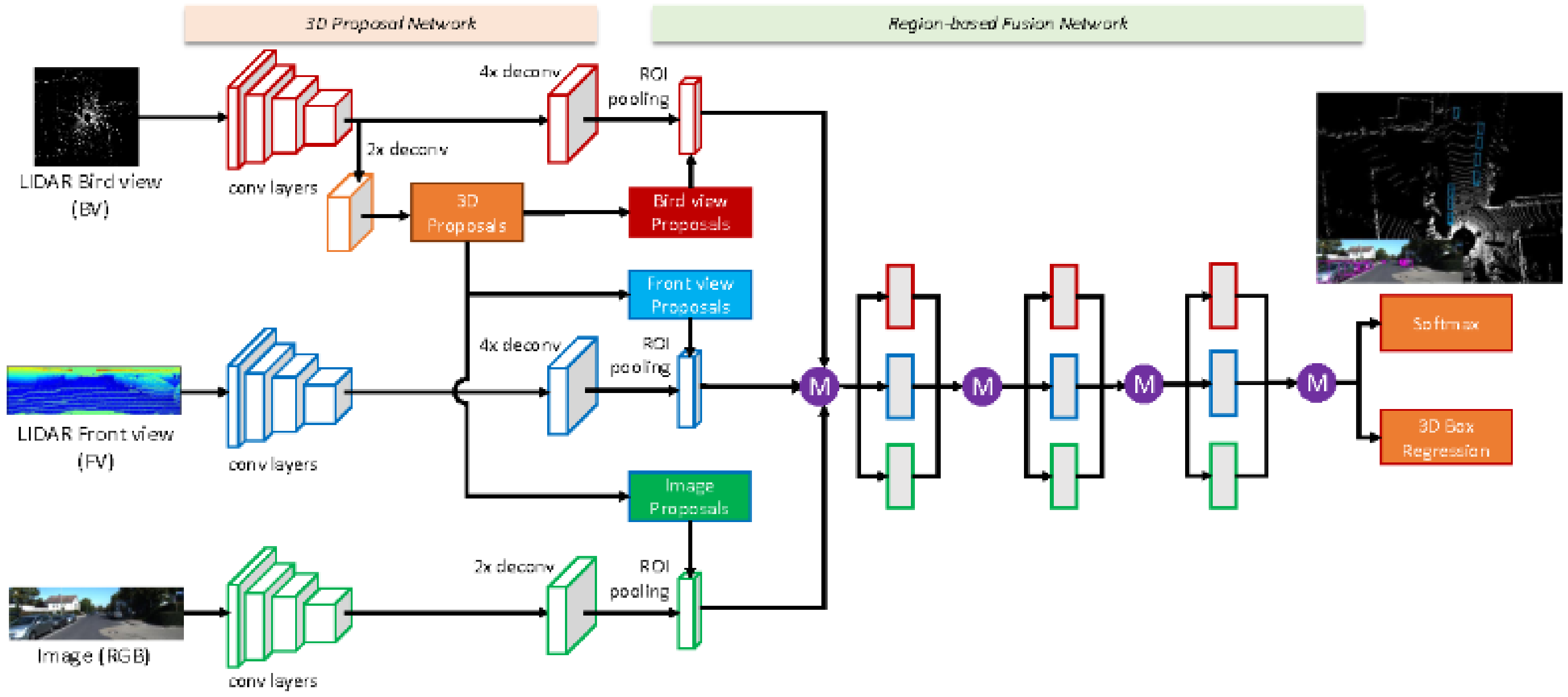


- **Data Granularity Issues**

Pixels	Lines	Frames
Bytes	Kilo Bytes	Mega Bytes

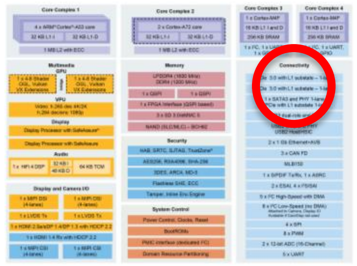
Algos Drive Compute Needs

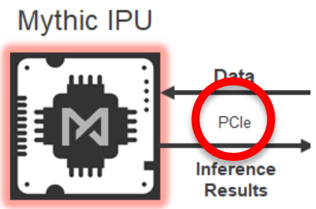
Combine classical image processing with DNN

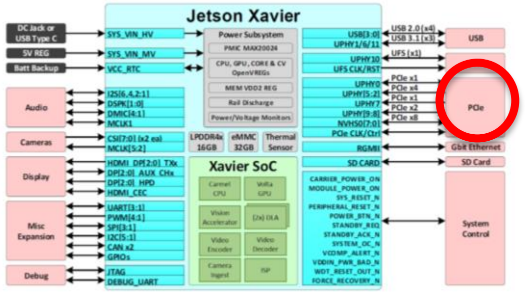


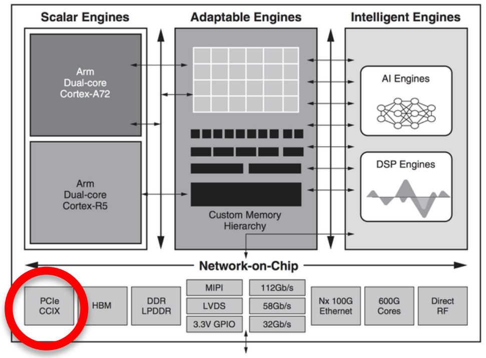
PCIe® Comes to the Rescue!

- **Relevant CPUs/GPUs/SoCs all have PCIe!**

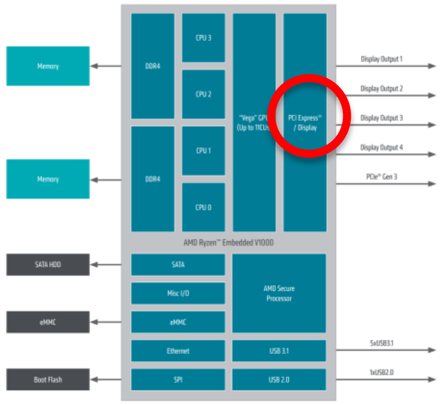






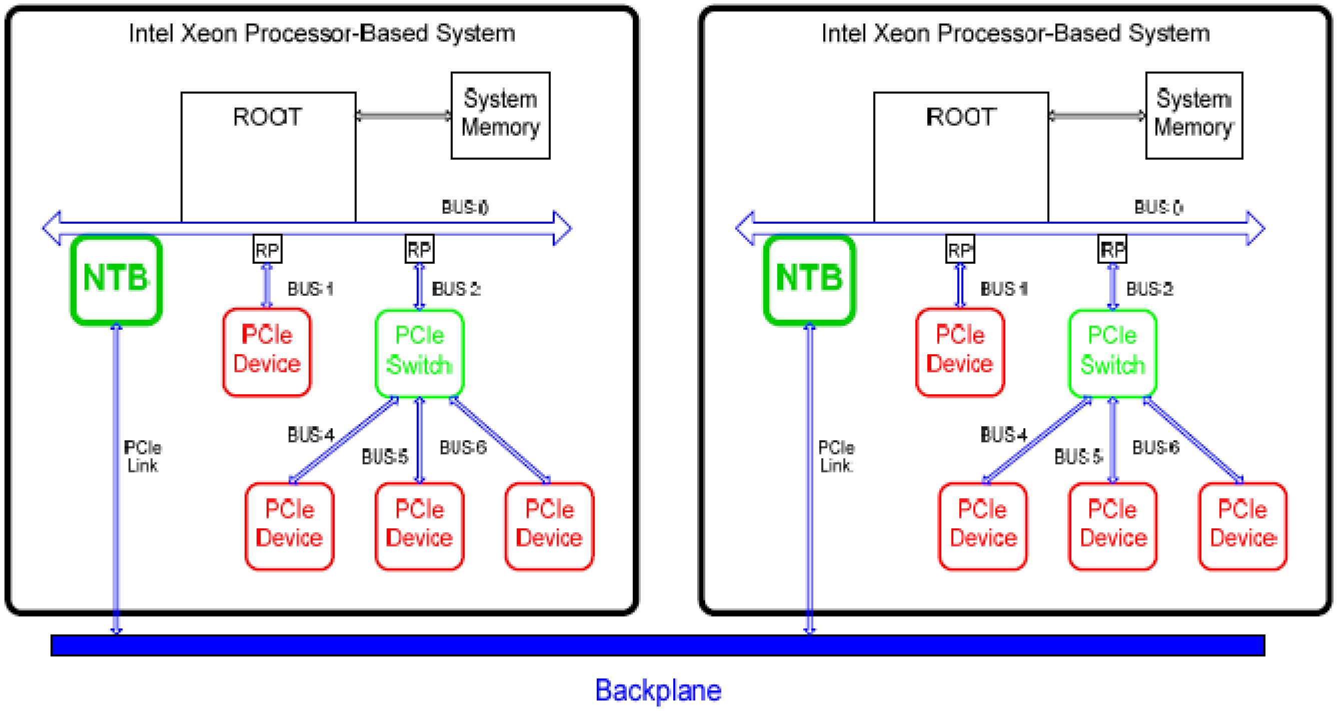


- ➔ **Build ECU with "PCIe network"**
  - Industry standard, relevant chips all have PCIe
  - Low latency (micro-seconds)
  - High bandwidth (tens of Gigabits per second)



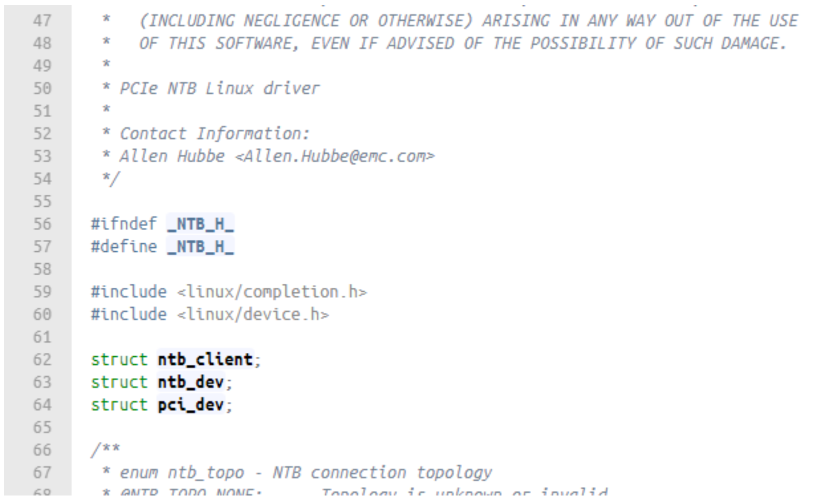
PCIe Non-Transparent Bridge

- **Non-Transparent Bridge (NTB) connects multiple Root Ports**
- **Example of NTB Back-2-Back**
  (Example from Intel Xeon C5500)



PCIe NTB – A Defacto Standard

- **“Using Non-transparent Bridging in PCI Express Systems” – Jack Regula, 2004**
- **Linux NTB from Jon Mason**
- **Supported by Linux kernel**
  - ntb.h



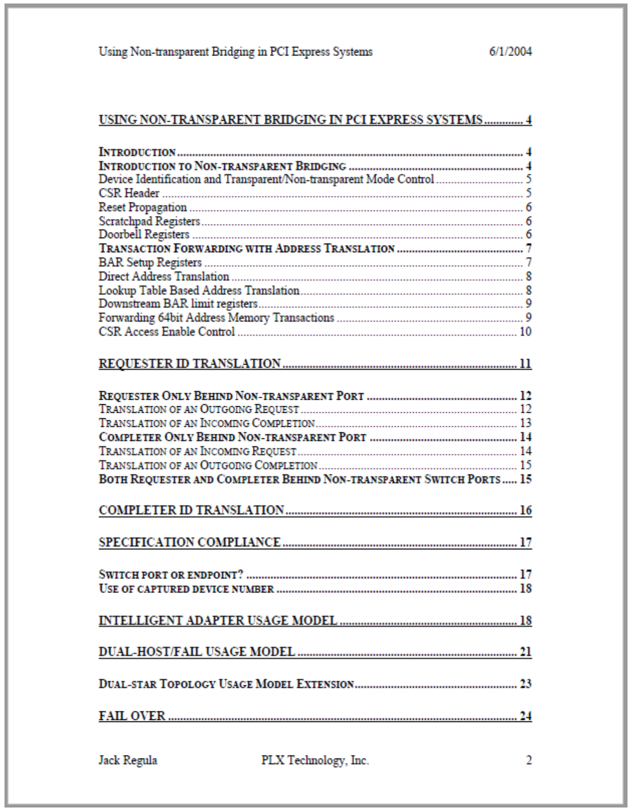
```
 *   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 *   OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * PCIe NTB Linux driver
 *
 * Contact Information:
 * Allen Hubbe <Allen.Hubbe@emc.com>
 */

#ifndef _NTB_H_
#define _NTB_H_

#include <linux/completion.h>
#include <linux/device.h>

struct ntb_client;
struct ntb_dev;
struct pci_dev;

/**
 * enum ntb_topo - NTB connection topology
```

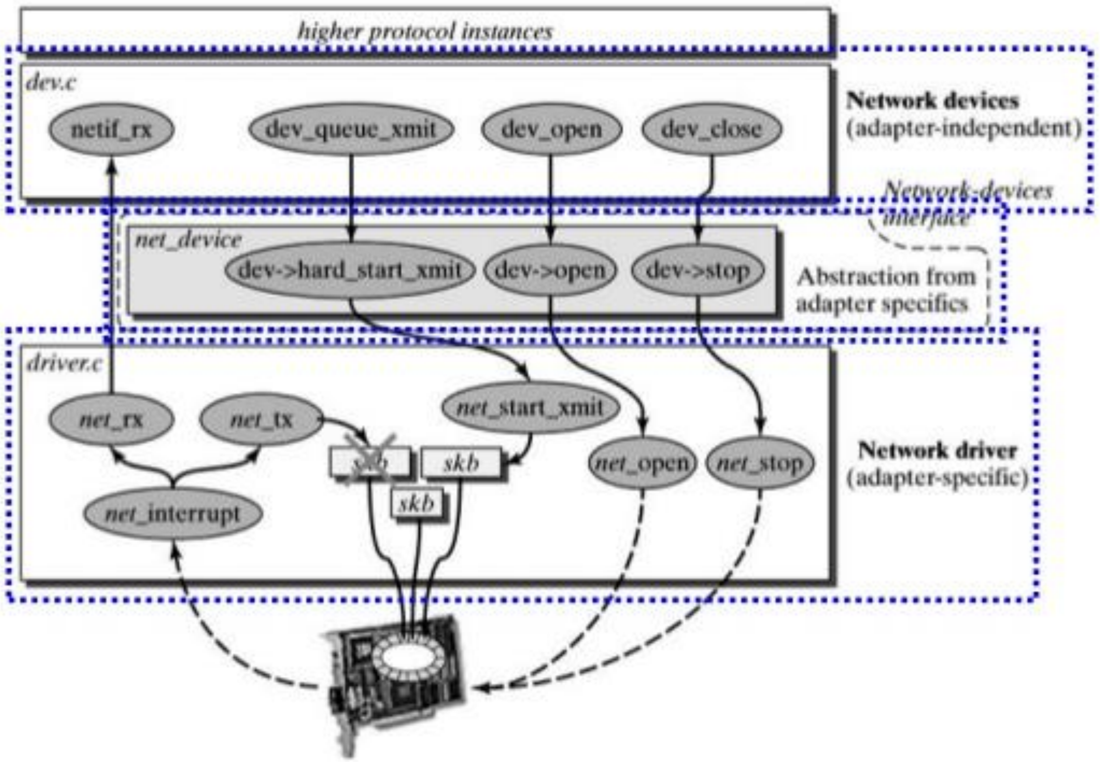


Using Non-transparent Bridging in PCI Express Systems 6/1/2004

Jack Regula PLX Technology, Inc. 2

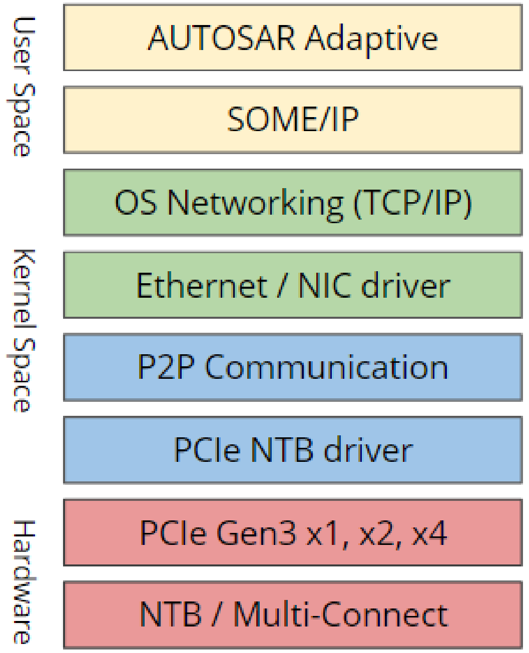
NTB - Programmer's View

- **Great, a network device!**
- **R/W via TCP sockets**



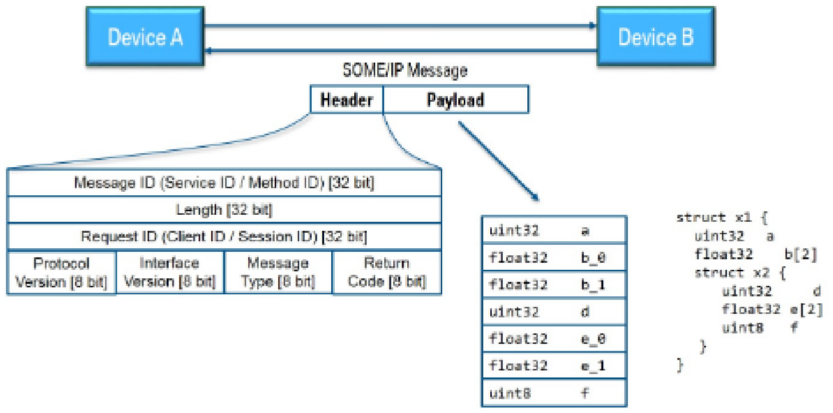
Automotive Comm. Standards

Software Stack on Compute Node
(Linux, QNX, Adaptive AUTOSAR, ...)



Application Programmer's View

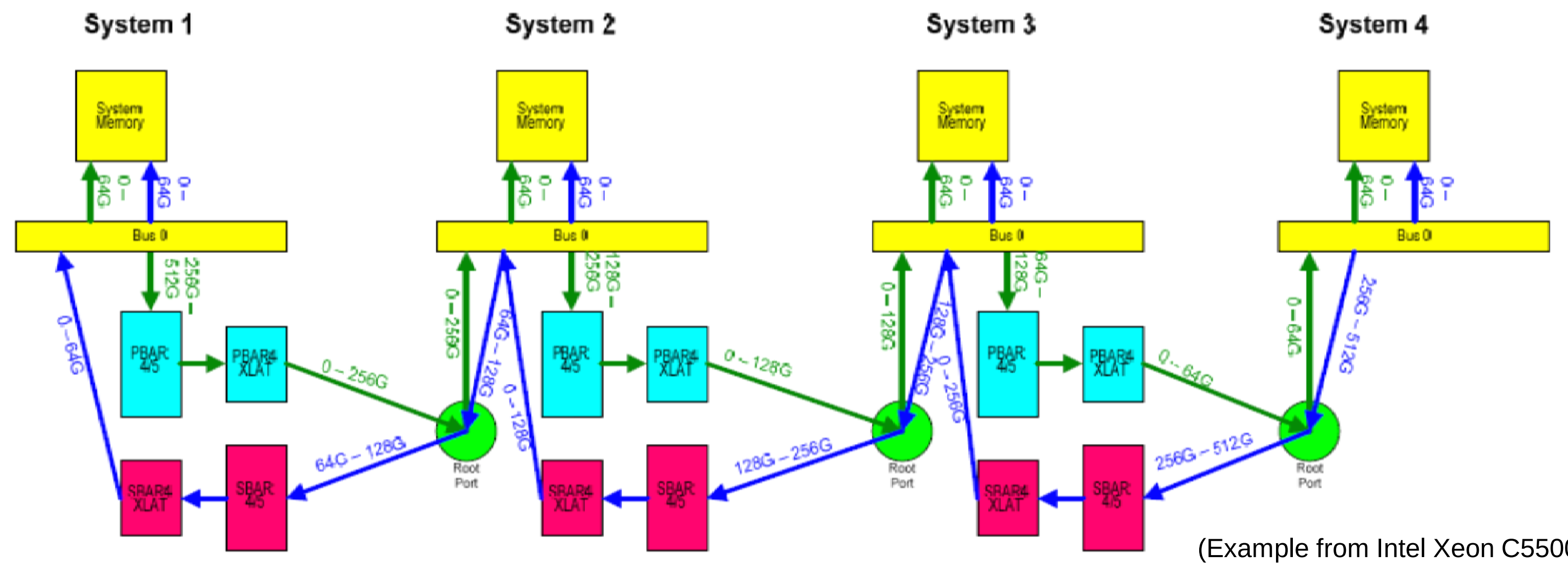
- Fully transparent comm. via PCIe NTB
  - Local (within one ECU)
  - Remote (between multiple ECUs)
- IP address for each Compute Node
- Gateway does routing, fail-over re-routing
- Send/receive TCP/IP, UDP/IP, SOME/IP messages



uint32	a
float32	b_0
float32	b_1
uint32	d
float32	e_0
float32	e_1
uint8	f

```
struct x1 {
  uint32   a
  float32    b[2]
  struct x2 {
     uint32     d
     float32 e[2]
     uint8    f
  }
}
```

PCIe NTB via Daisy-Chain

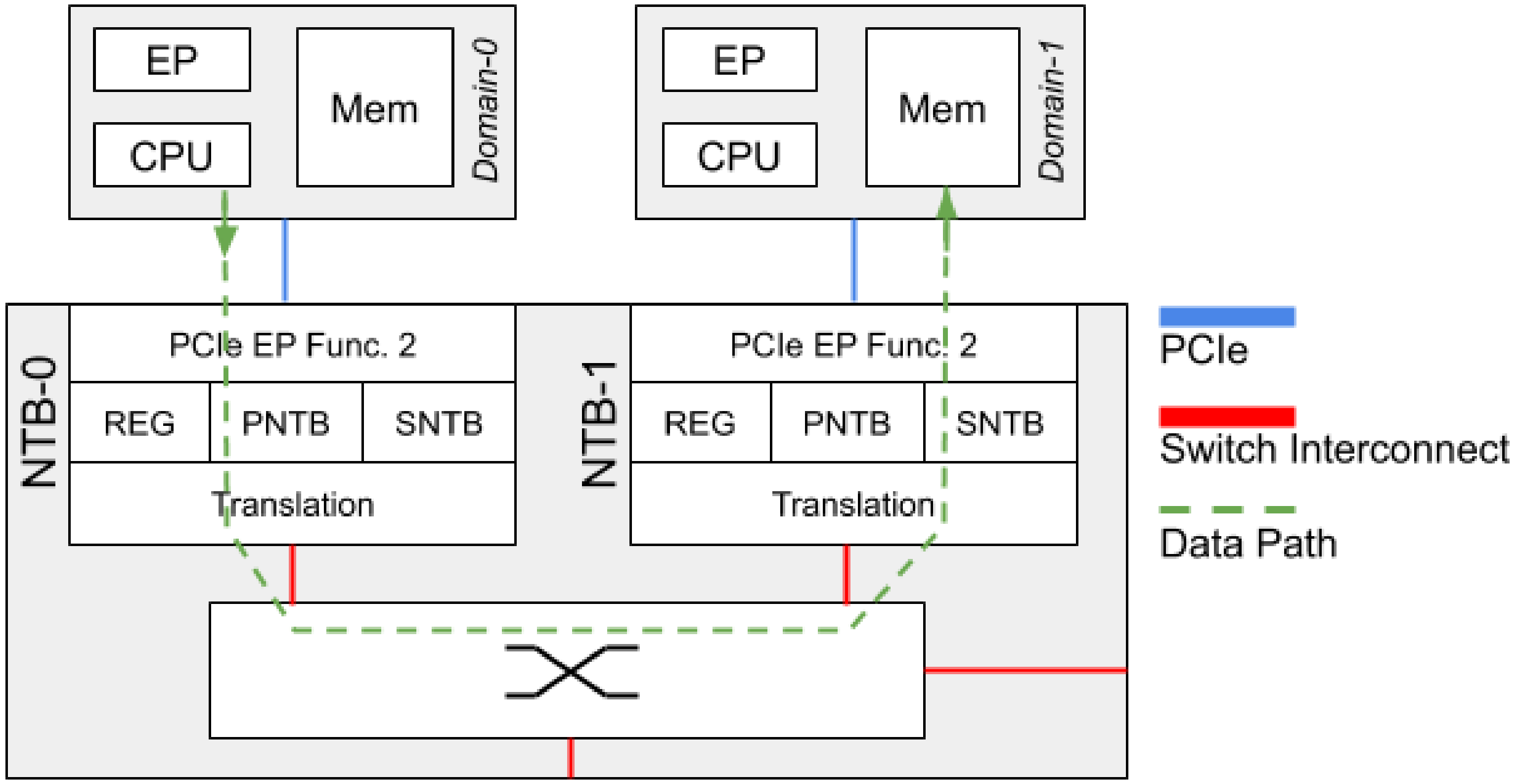


(Example from Intel Xeon C5500)

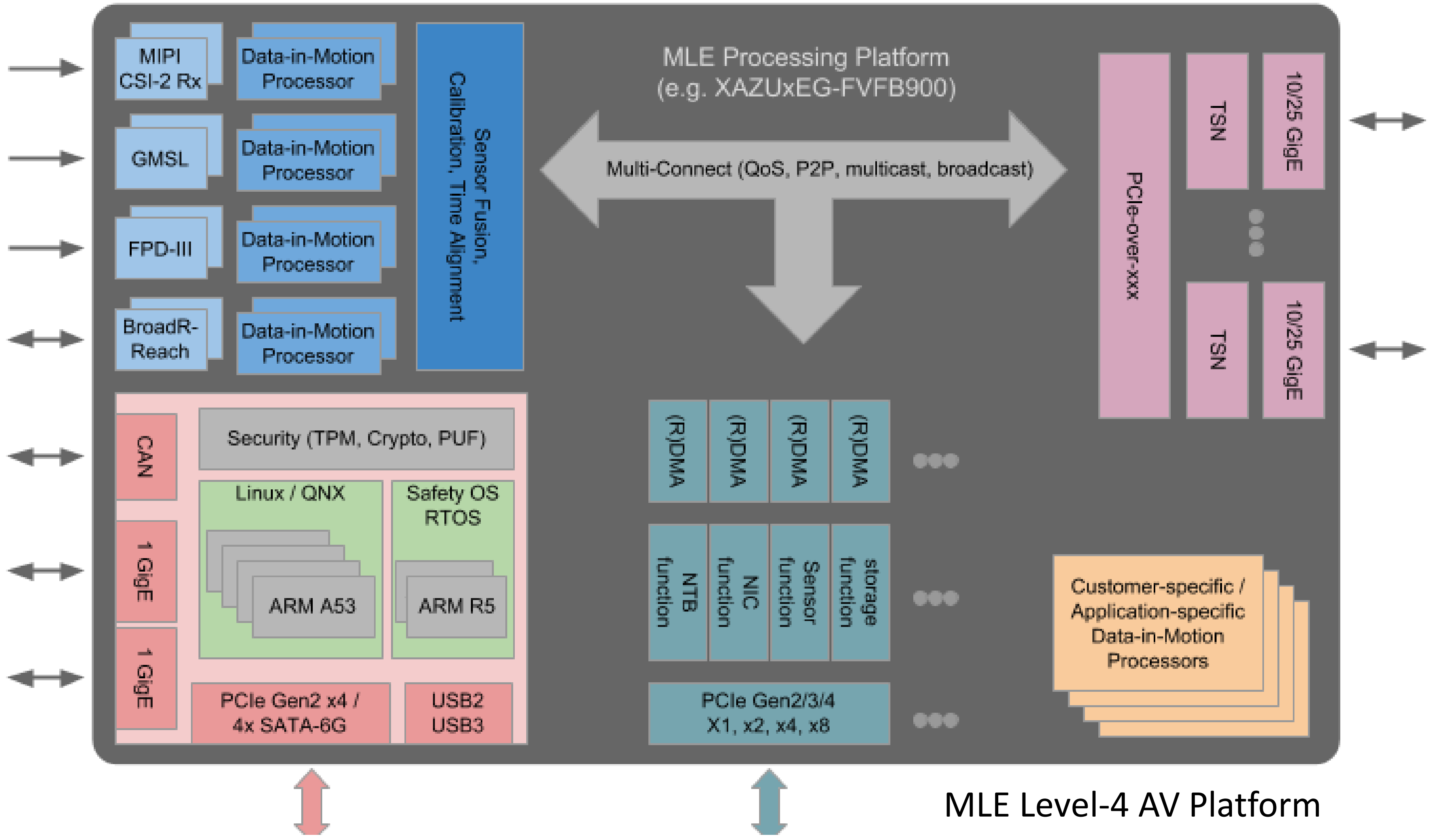
- **Not optimal for Automotive ECU**
  - Shared Bandwidth
  - Not resilient to HW failures
  - Added Latency for ID translation

PCIe Non-Transparent Bridge

Network-on-Chip for Any-2-Any Connectivity between PCIe Roots



FPGA-Based NTB Architecture

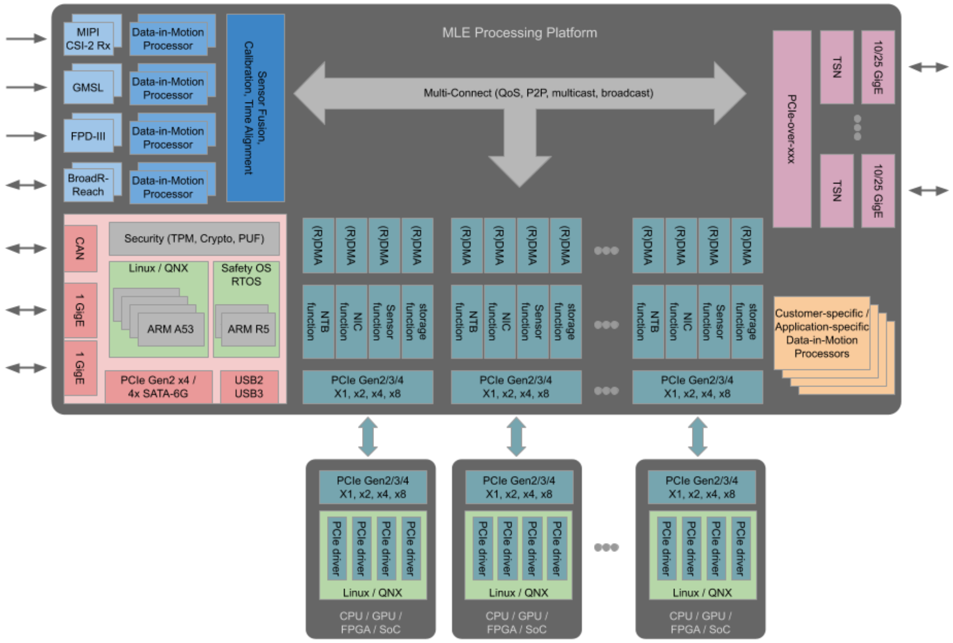


MLE Level-4 AV Platform

Data Acquisition & Preprocessing

FPGA technology enables

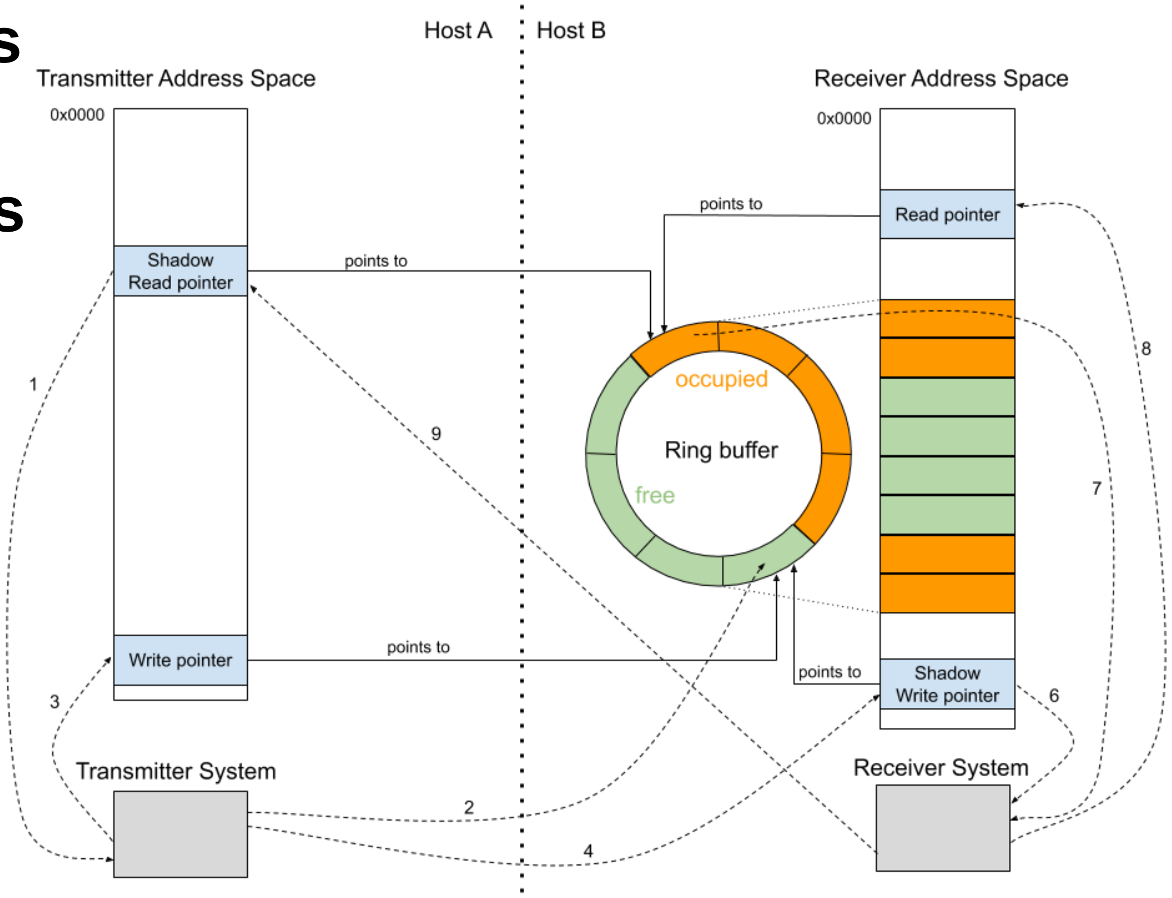
- Flexibility to deal with sensor i/f
- Sensor fusion
- Data Acquisition and Data Preprocessing (DADP)
- Data-in-motion preprocessing
- Functional safety (monitor compute nodes & re-route)
- ECU Scale up / ECU Scale out



Delivering Performance

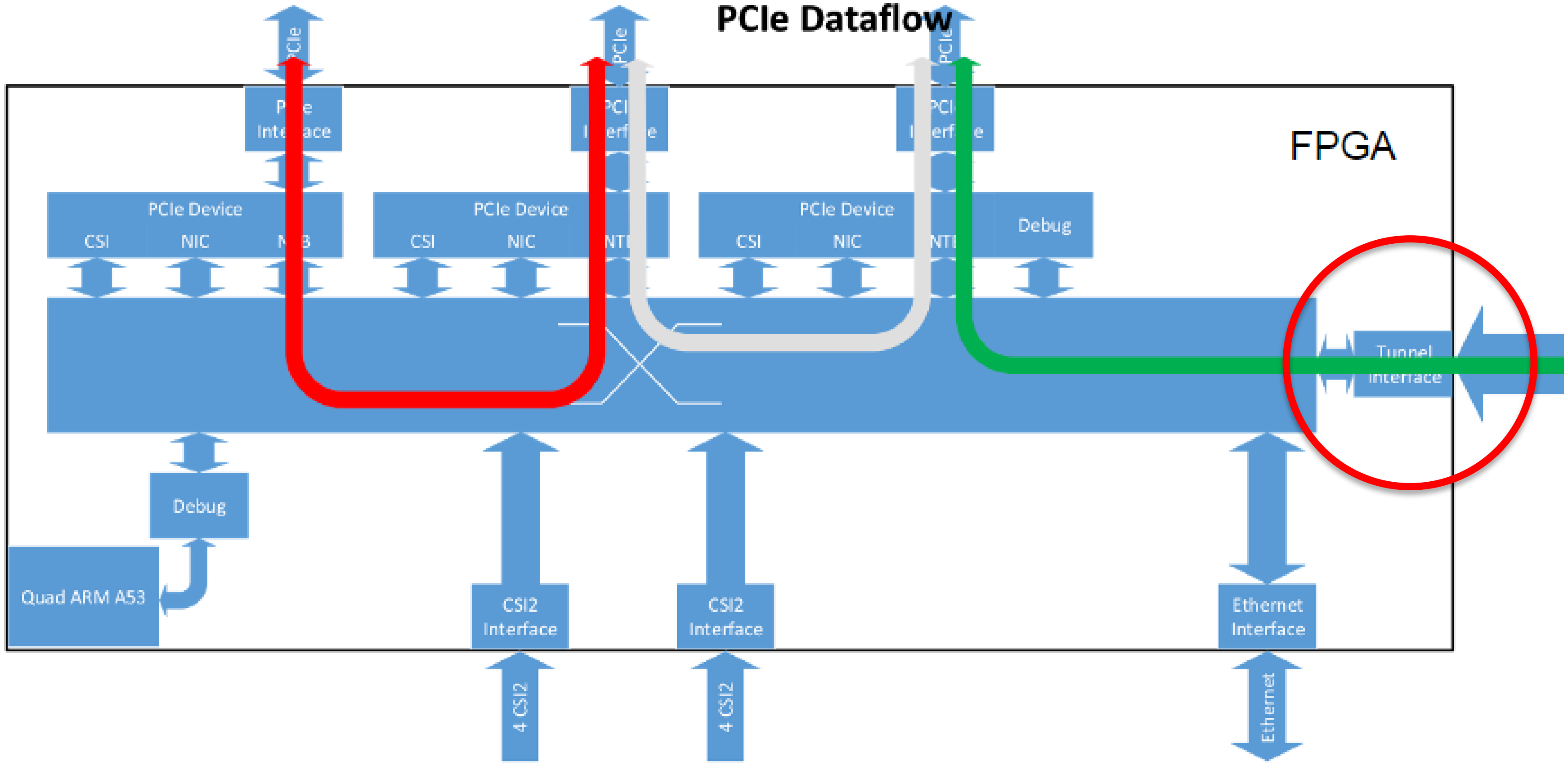
Write-Only Communication via Doorbells - NVMe-style

- Avoids difficulties of multi-device
- Scales to >32 RCs
- Posted Writes



Peer-2-Peer Communication

Custom Switch Based Design

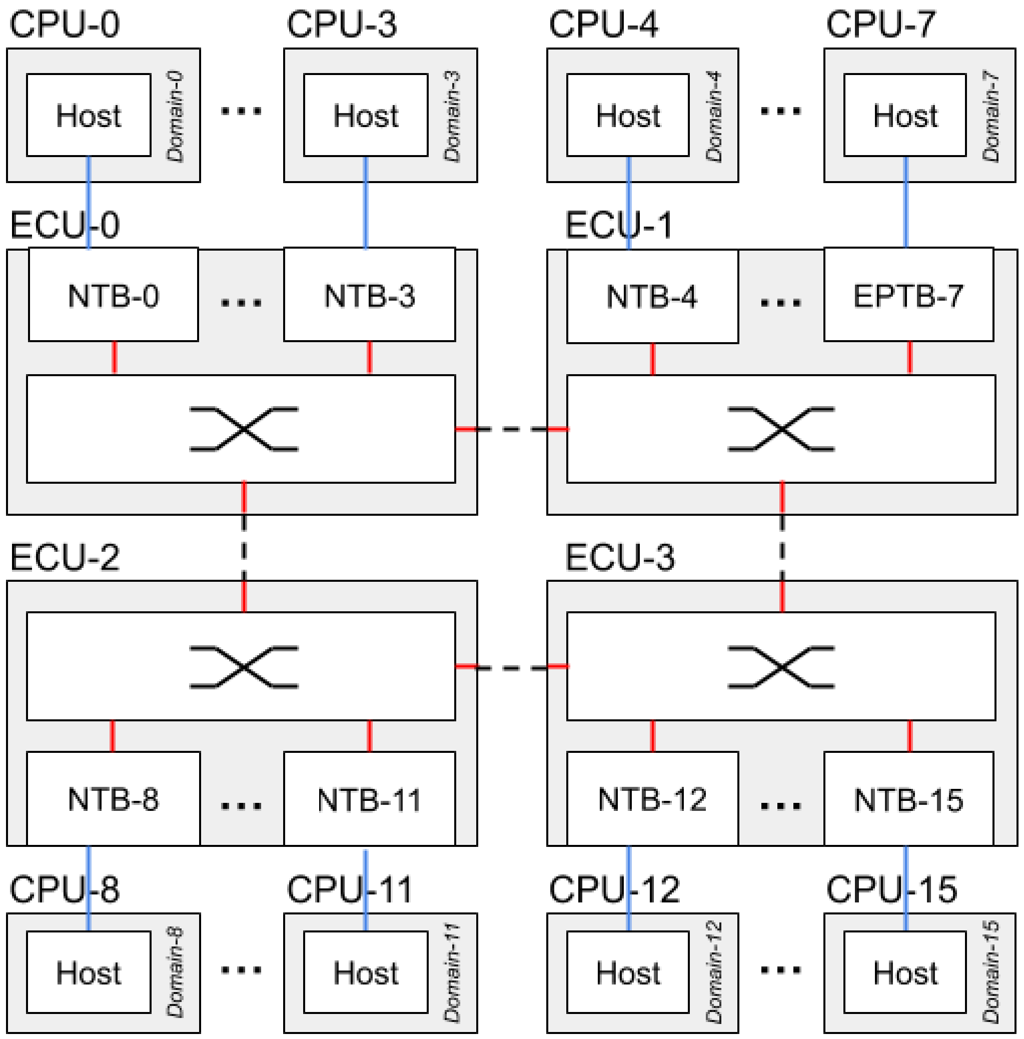


Secure, Reliable, Resilient

- Security / Reliability / Dependability:
  single host cannot take entire system down!
  - local configuration of local properties
  - receive side memory protection,
  - incoming traffic for memory space not configured is discarded and reported (especially useful for (embedded) devices without IOMMUs)
  - shielded global NTB config (orthogonal control path for inter NTB connectivity / routing from Primary NTB to Secondary NTB)

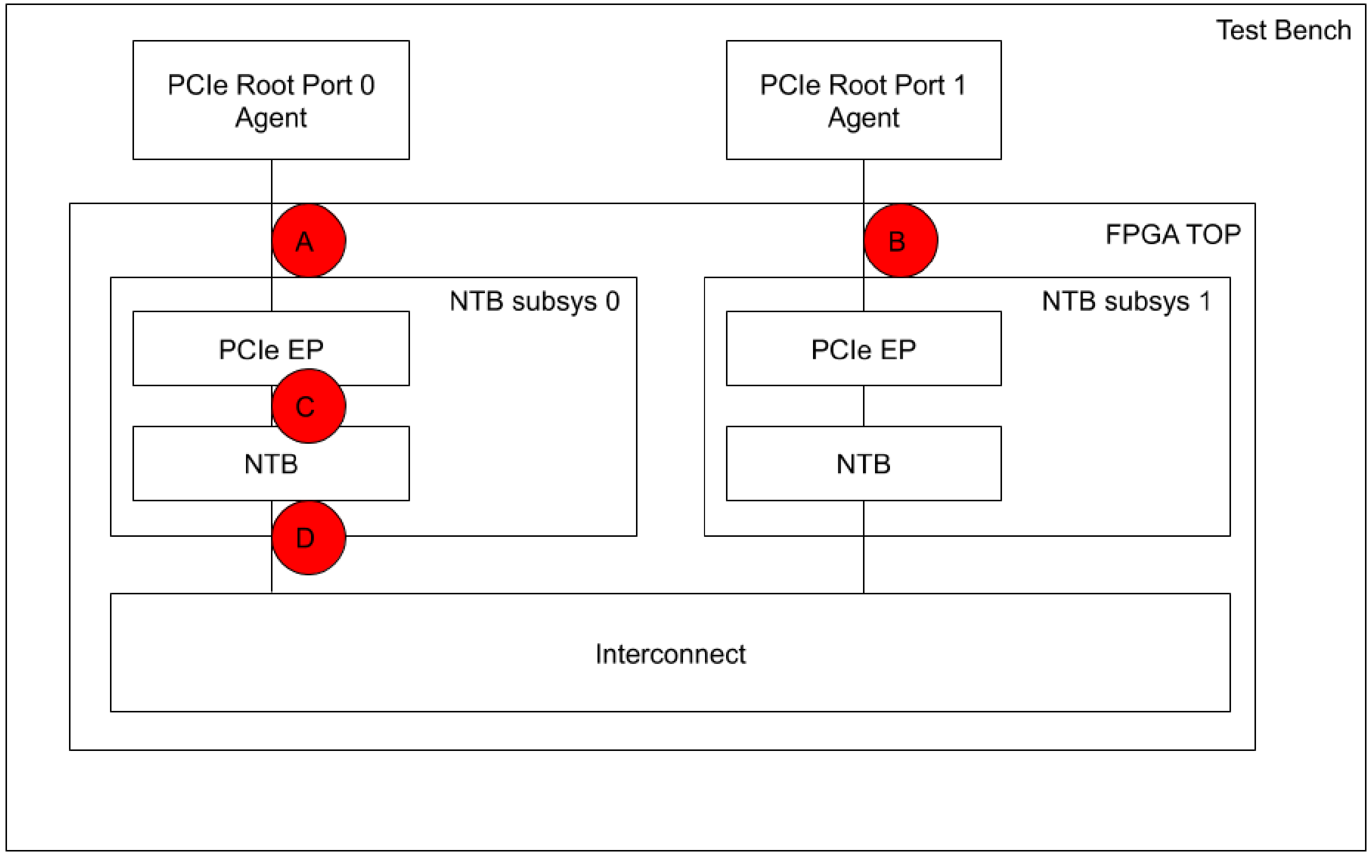
Fail-Overs

- ECU-2-ECU via automotive 10/25/50G Ethernet
- Detect PCIe failure via AER
- Intra-ECU re-route
- Inter-ECU re-route



System-Level Verification

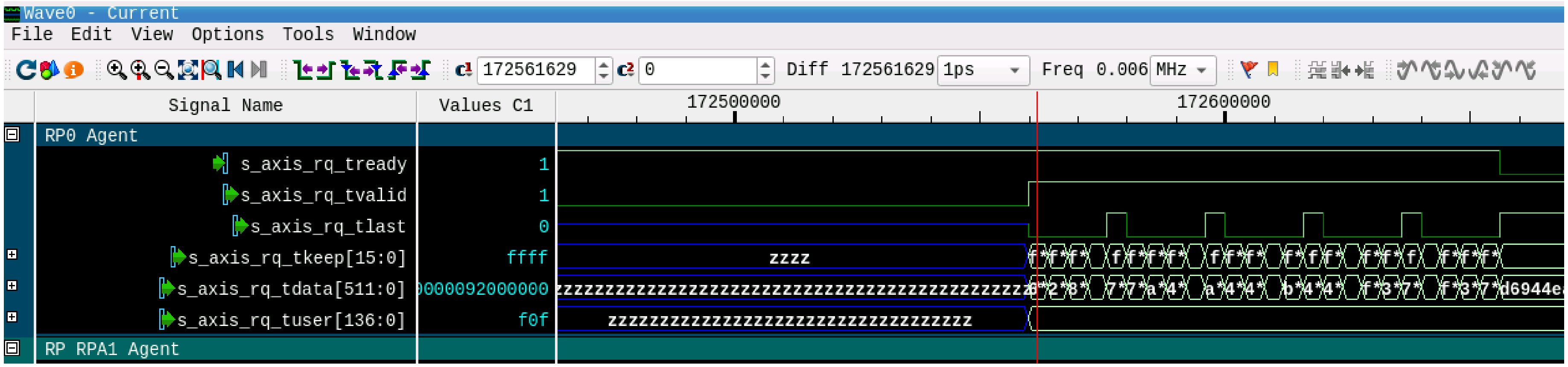
- Use of Xilinx VIP for PCIe + PCIe Root Agents with scripted testcases
- Questa Prime PIPE-Level Sim runs ~ 180 seconds



System-Level Verification

- **Simulation-based performance close to theoretical max:**
-

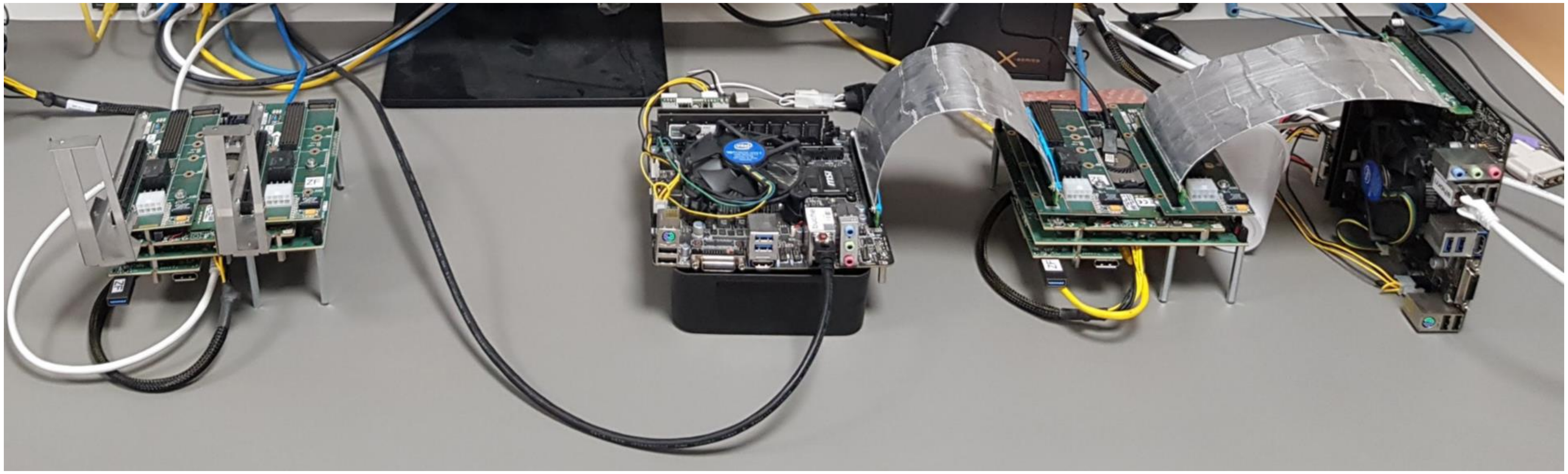
```
# Run Test Case 60, MLE NTB bandwidth, single peer to peer, TLPs w/ 256 bytes payload, 131072
bytes total
#     213928.00 ns: data transfer, RPA0 to RPA1, duration: 41828.00 ns, bandwidth: 3.133595 GB/s
#     259828.00 ns: data transfer, RPA0 to RPA2, duration: 45896.00 ns, bandwidth: 2.855848 GB/s
#     301644.00 ns: data transfer, RPA1 to RPA0, duration: 41812.00 ns, bandwidth: 3.134794 GB/s
#     347532.00 ns: data transfer, RPA1 to RPA2, duration: 45884.00 ns, bandwidth: 2.856595 GB/s
#     389304.00 ns: data transfer, RPA2 to RPA0, duration: 41768.00 ns, bandwidth: 3.138096 GB/s
#     431092.00 ns: data transfer, RPA2 to RPA1, duration: 41784.00 ns, bandwidth: 3.136895 GB/s
```



Signal Name	Values C1
RP0 Agent	
s_axis_rq_tready	1
s_axis_rq_tvalid	1
s_axis_rq_tlast	0
s_axis_rq_tkeep[15:0]	ffff
s_axis_rq_tdata[511:0]	)0000092000000
s_axis_rq_tuser[136:0]	f0f
RP RPA1 Agent	

Lab Setup

ProFPGA ZU19 Prototyping System with multiple x86 as PCIe Root and cable PCIe Edge-to-Edge, male-to-male, crossed



Results of Performance Analysis

- iPerf – TCP bandwidth measurement (x86)

```
dummy@buche:~/src/sw/ntbpi$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.3 port 5001 connected with 192.168.1.7 port 33686
------------------------------------------------------------
Client connecting to 192.168.1.7, TCP port 5001
TCP window size: 2.01 MByte (default)

dummy@ahorn:~/src/sw/ntbpi$ iperf -c 192.168.1.3   -t 28800 -i 60
------------------------------------------------------------
Client connecting to 192.168.1.3, TCP port 5001
TCP window size: 1.25 MByte (default)
------------------------------------------------------------
[  3] local 192.168.1.7 port 33690 connected with 192.168.1.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-60.0 sec  90.9 GBytes  13.0 Gbits/sec
[  3] 60.0-120.0 sec  91.0 GBytes  13.0 Gbits/sec
[  3] 120.0-180.0 sec  91.9 GBytes  13.2 Gbits/sec
```

Conclusion

- **To build multi-CPU/GPU/SoC AV ECUs we combine existing PCIe specification with defacto industry standards (Linux NTB)**
- **Single chip solution based on automotive-grade SoC-FPGA**
  - Delivers performance close to theoretical max.
- **Devil in the details:**
  - Interrupt schemes: MSI vs MSI-X
  - Functional Safety vs Surprise Hotplug
  - Where to put DMA?
    - CPU side?
    - FPGA side? Local write vs remote write?

Thank you for attending the PCI-SIG Developers Conference 2019.

For more information please go to www.pcisig.com