Advancements in Automotive Ethernet Addressing Challenges and Implementations in Asymmetric Multigigabit Ethernet-based Sensor Networks

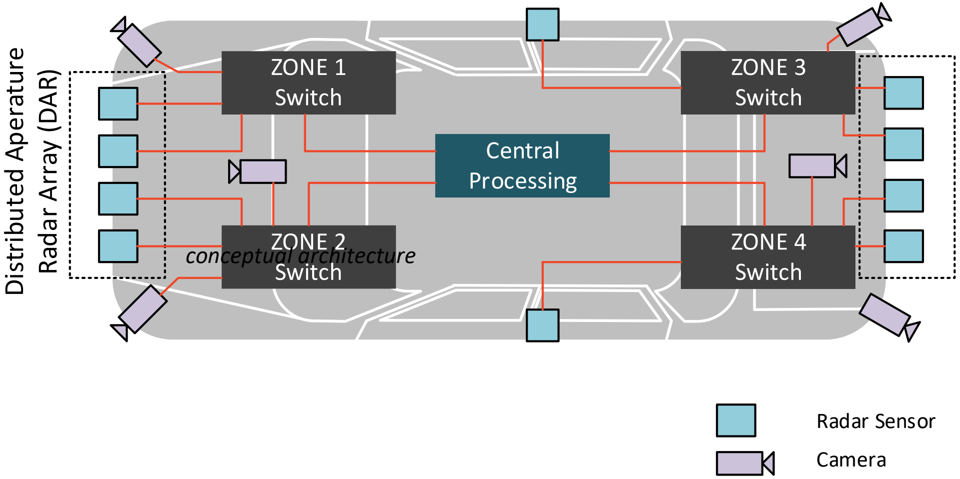
Dr. Philip Axer, Lead System Architect

Manfred Kunz, Vice President of Products

AXONNE

Zonal Architecture – All Ethernet

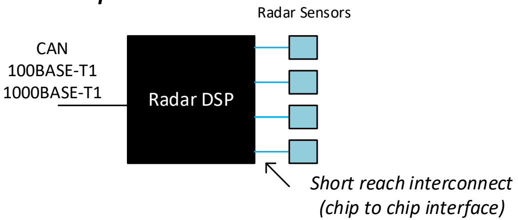
- **Trend towards Zonal**
  - Compute in center
  - Sensors in zone
  - Common network (Ethernet), where possible
- **Driving factors**
  - Availability of RAW data in the center
    → Enabler of SDV (Software Defined Vehicle)
    → Enabler of Sensor Fusion
  - Reduction of cables, shorter cables, less connectors, less copper, simpler assembly
  - Less „boxes“→ less software blobs to distribute
- **All Ethernet** → same set of protocols and mechanisms across the car

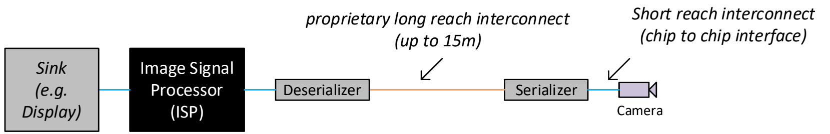


Landscape - Status Quo

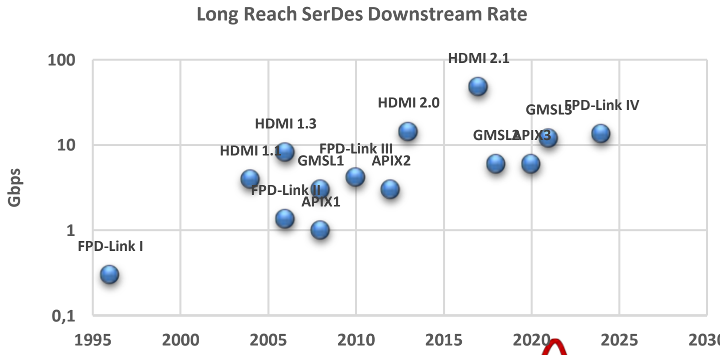
- Radar Sensors are connected over **short-reach interconnect** to Radar processor
- Radar processor sends object data via CAN or *slow* Ethernet
- Image Sensors are connected to **proprietary SerDes solutions** to bridge the spacial gap
  - APIX3, FPD-LINK III / IV, GMSL3
- **Asymmetric SerDes** solutions offer
  - Low-speed upstream rate → tunneling of GPIO, I2C, SPI, …
  - High-speed downstream rate → Data streaming
- Short-reach steaming interconnect for Imagers and Radar
  - MIPI CSI-2 D-PHY / C-PHY, MIPI DSI D-PHY
- Low-datarate sensors/actuators use CAN

Example: Radar





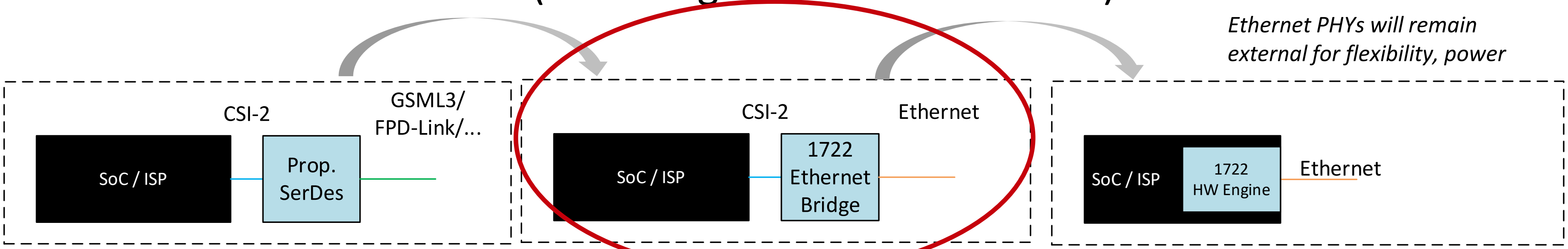
Example 2: Imager



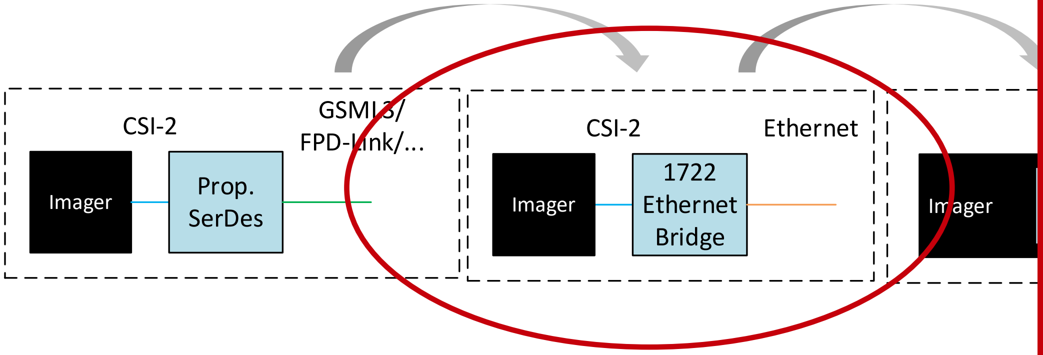
Evolution of Interfaces

Evolution of **SoC** interfaces (assuming All-Ethernet sensors)

Ethernet PHYs will remain external for flexibility, power



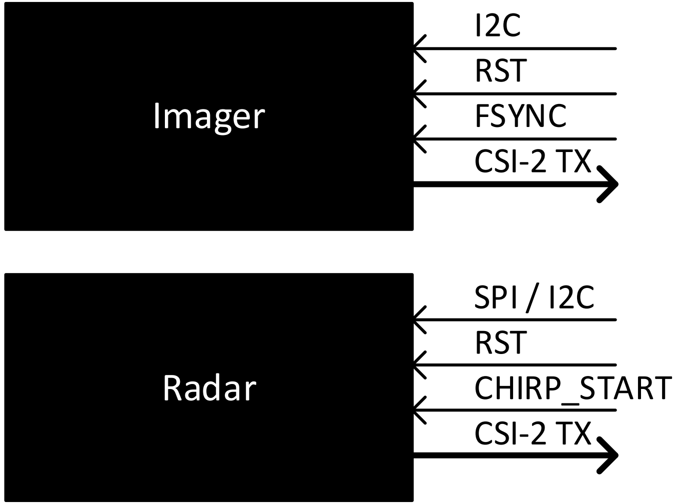
Evolution of **Sensor** interfaces (assuming All-Ethernet sensors)



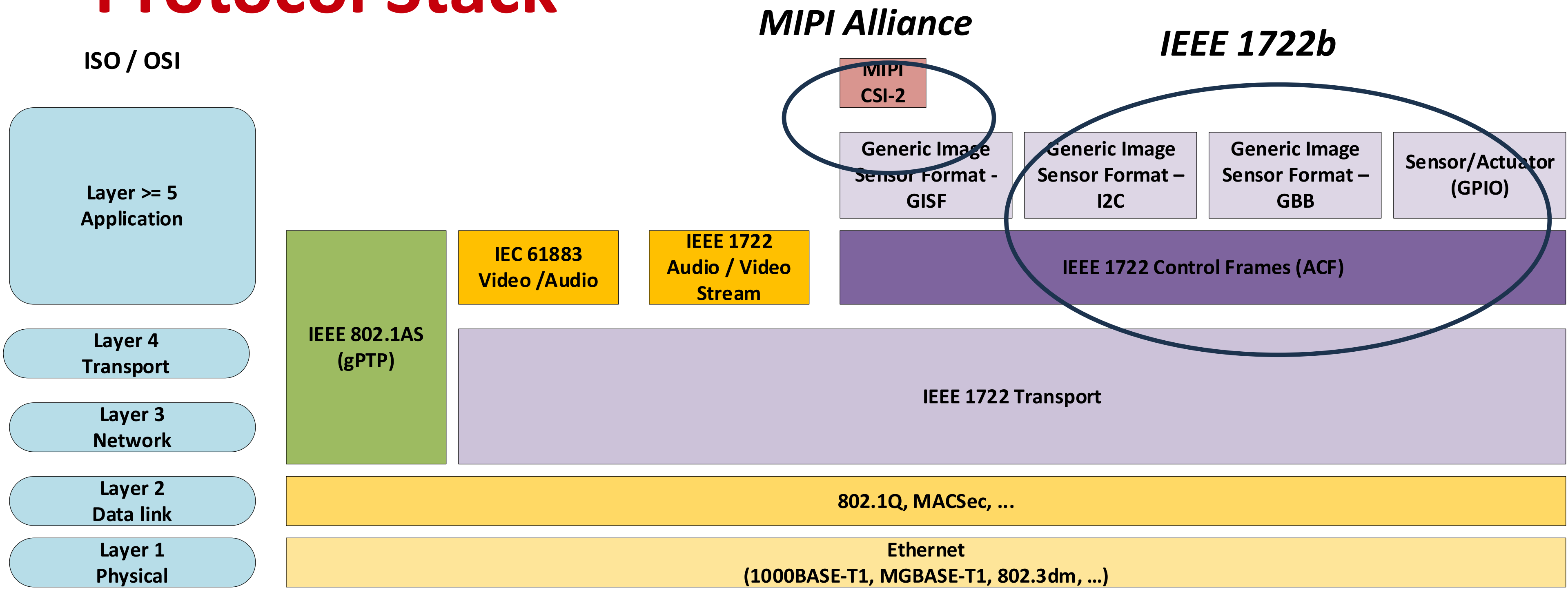
To enable the first transition in this evoluation existing protocols such as CSI-2, I2C need to be tunneled through Ethernet

What to encapsulate / communicate

- **Serial protocols**
  - I2C (mostly imagers)
  - SPI (some radar chips)
- **GPIO signals**
  - Reset, LEDs, FUSA logic
- **Synchronization signals**
  - FSYNC, CHIRP_START
- **Timing information**
  - To annotate data with correct presentation timestamps
  - Done through 802.1AS (gPTP) → already available

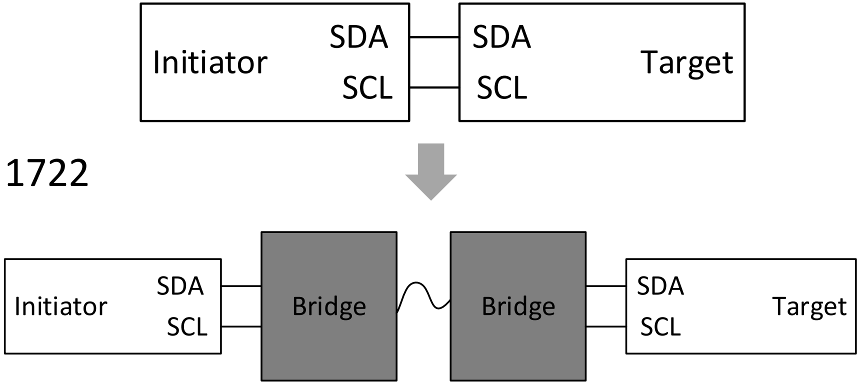


Protocol Stack

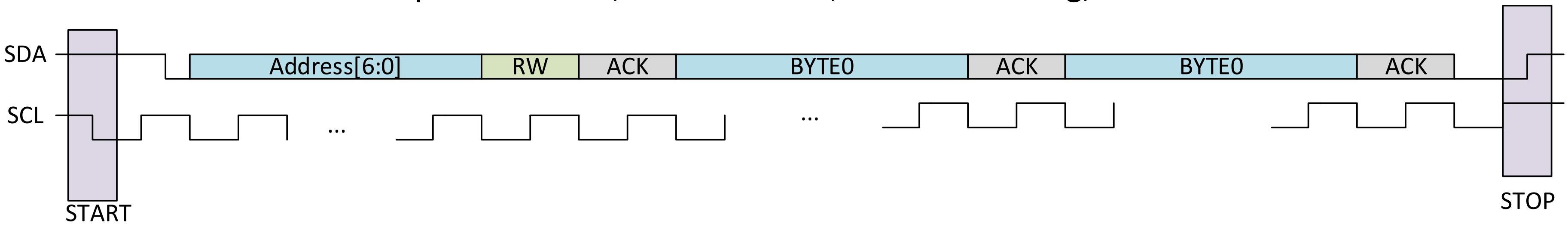


I2C ENCAPSULATION

I2C Encapsulation



- Faithfully transmit/receive SCL/SDA over Ethernet via 1722
- SDA is bidirectional open-drain
- Each byte is acknowledged by the respective receiver
  - ACK/NACK driven by target for ADDR and WRITE bytes
  - ACK/NACK driven by initiator for READ bytes
- Advanced features: Repeated START, 10bit address, clock-stretching, …

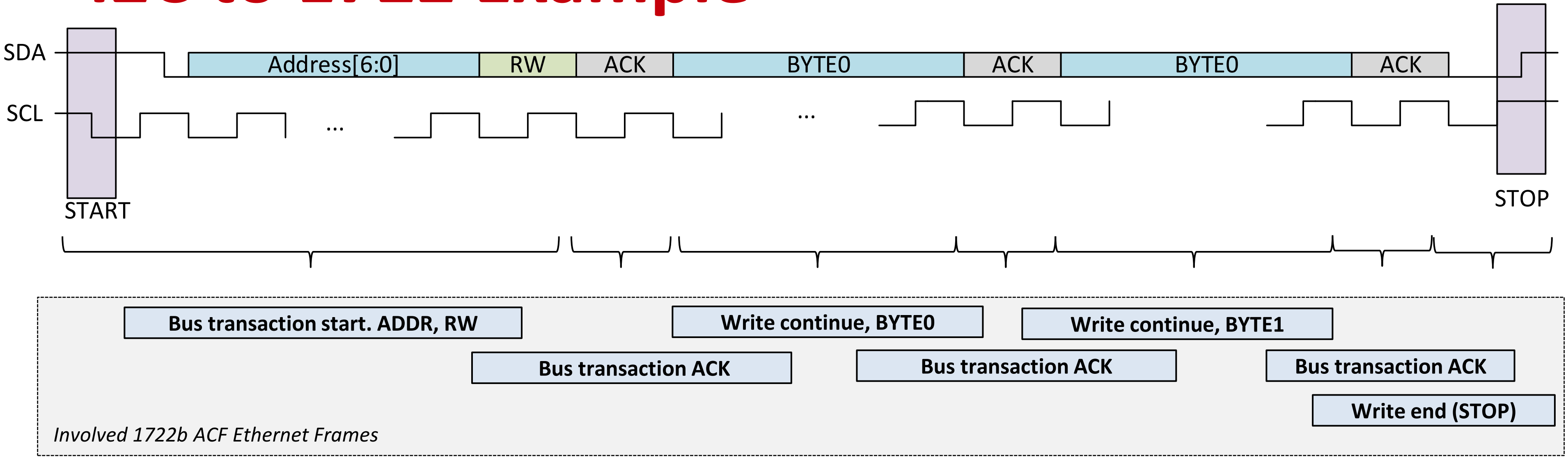


General Challenge:

- A read/write transaction cannot be send in one shot.
  - WRITE: Bridge does not know how many bytes the target is able to consume
  - READ: Bridge does not know how many bytes the initiater wants to read

→ Unroll entire I2C transaction, event-by-event and create one 1722 frame per event

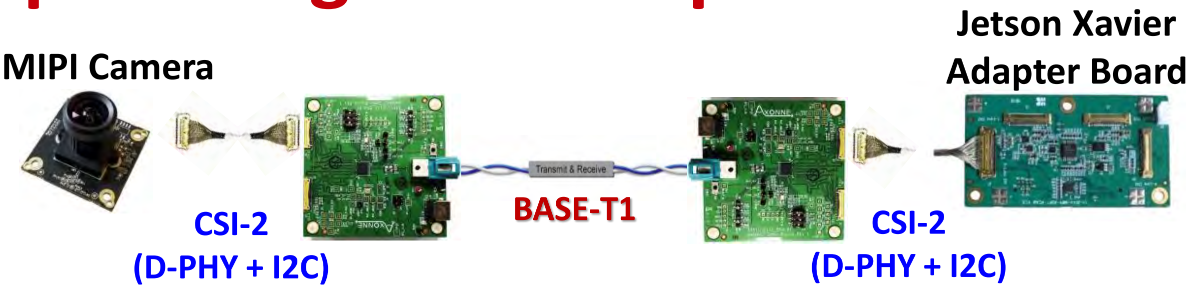
I2C to 1722 Example



1722b ACF Encapsulation:

- Each ACF frame is 64B due to minimum Ethernet framing rules
- 3 bytes I2C addr/data → 7 · 64B = 448B
- Relatively inefficient, but the Ethernet speed is abundant compared to I2C speed

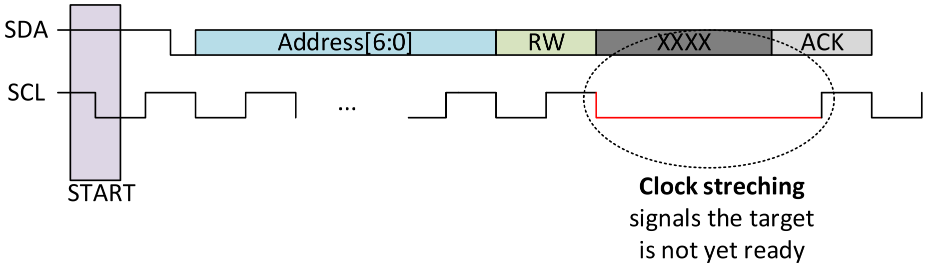
Example: Imager Init Sequence



- IMX Leopard image camera connected to NVIDA Jetson
- Actual I2C imager configuration data:
  1.5 kB over I2C (including i2c address and data)
- Encapsulated into 3696 frames, total **231 kB** over T1
  - **Factor ~150**
- *Potential* latency issue for large topologies or slower Ethernet speeds

AXONNE

Advanced Topics

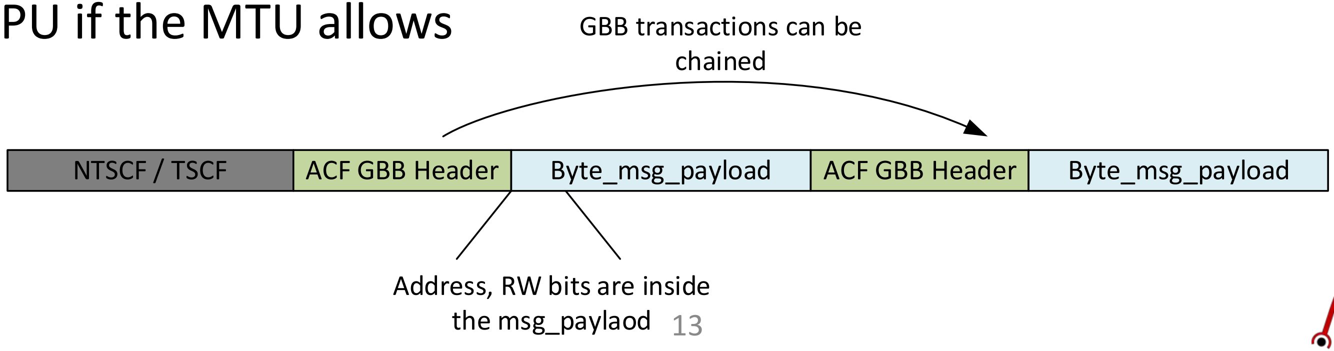


- **Backpressure**
  - Normally, I2C clock is driven from the initiator and consumed by the target
  - I2C tunneling needs to cope with the transmission latency of response frames
  - The target can backpressure by holding the clock low (clock stretching)
  - Clock stretching is optional in the I2C specification → initiator must support it
- **Error recovery**
  - Ethernet is lossy
  - ACF I2C specifies a simple error handling
  - Timeout-based approach with sequence number and retransmission

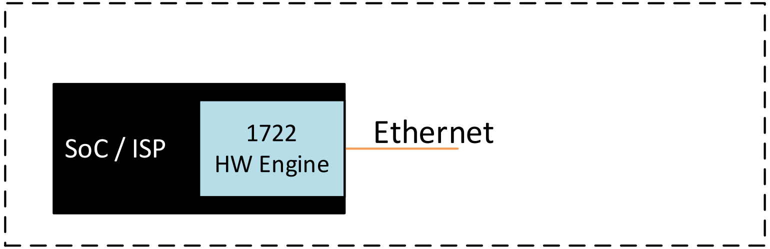
GENERIC BYTE BUS

GBB – Generic Byte Bus

- Removes overhead of I2C ACF
- Carries address, data in one frame for efficient transport
  - Support for large data bursts from kB to 2 Mb
  - Fragmentation support for large bursts
- Allows to pack multiple transfers into one AVTPDU
  → one full initialization sequence can be packed into a single AVTPU if the MTU allows



GBB Drawbacks



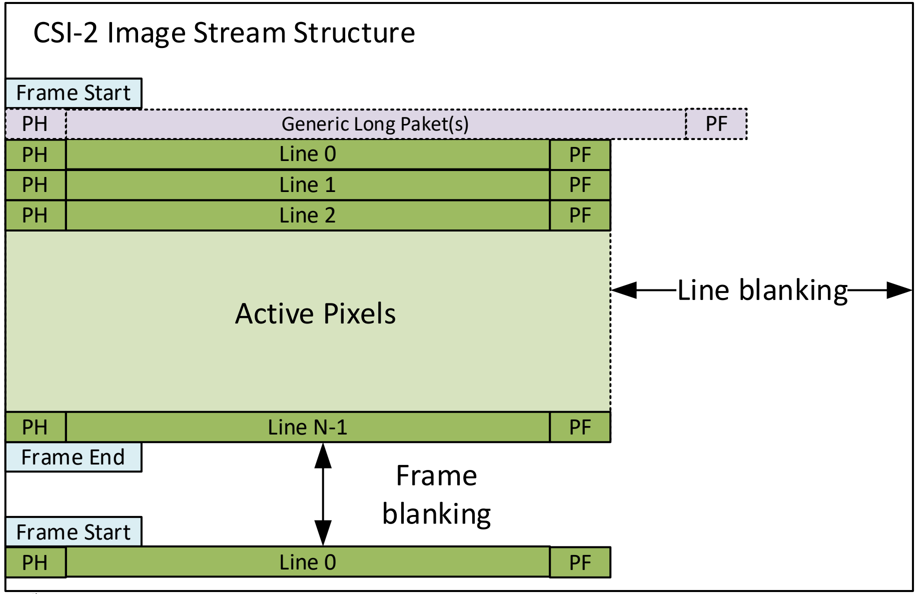
- Not able to faithfully tunnel physical interface signals
- GBB initiator must be an explicit SW/HW implementation in the host
- But: GBB target can unpacks the signals and talk to the physical I2C, SPI device

AXONNE

CSI-2 ENCAPSULATION

CSI-2 Overview

- Relevant released MIPI specs
  - **MIPI CSE** (Camera Service Exensions)
  - **MIPI CSI-2** (Camera Serial Interface 2)
  - *Draft specs to encapsulate CSI-2 over IEEE 1722*
- Paket-based protocol
  - Short pakets for control
  - Long pakets for data

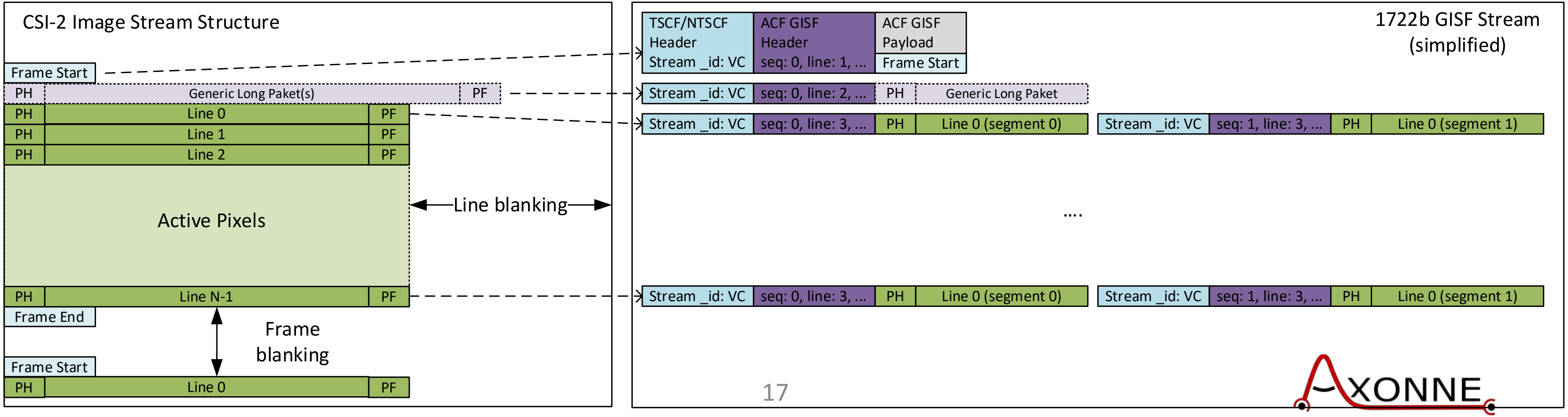


Details:
https://www.hackster.io/sam-d/csi-2-image-simulator-lattice-crosslinkfpga-to-jetson-nano-24ac21

AXONNE

CSI-2 1722b Encapsulation (simplified)

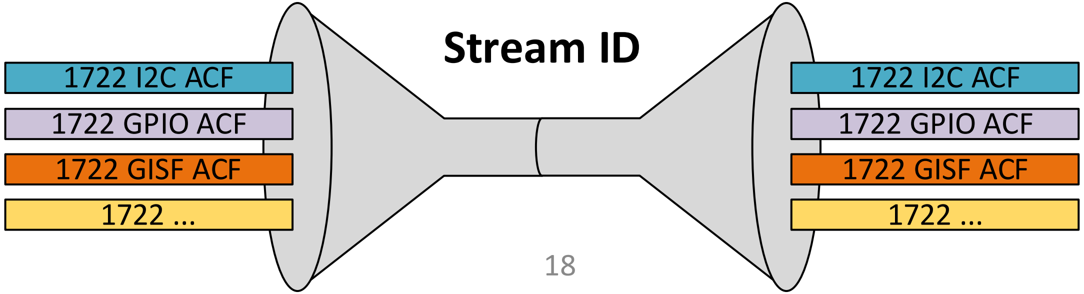
- Piggyback onto ACF GISF (Generic Image Sensor Format)
  - GISF is **agnostic of image format**, knows about
    - Stream_id, message length, sensor id, line, frame timestamps, data
  - CSI-2 **virtual channel → stream_id**
  - **Segmentation** via line number and sequence number, **reassembly** on the receiver required
  - Actual encapsulation is **more advanced**



AXONNE

1722 Protocol „Overhead“

- 1722b AVTP adds Ethernet headers
  → some protocol overhead
- However, **big benefit** of the 1722 encapsulation
  - **Associate timestamps with events** in a consistent way
  - **Associate stream ids with multiple flows that belong together**
    - FSYNC, I2C, Video CS-2 can be captured under one common stream
  - Layered protocol stack approach follows traditional way of working



AXONNE

Summary & Conclusion

- Tunneling ACF for raw image data, I2C, SPI allows a **smooth transition path to „all Ethernet“**
- Chance of legacy protocol carry over
  - Today’s 1722b protocols are all crafted around physical interfaces (e.g. CSI-2). In future the physical interfaces might disappear entirely.
    → but they remain in existence virtually.
  - Once up-integration is completed, do we need/want to optimize the transport protocols again?
- The current set of protocols (available as drafts from MIPI and IEEE) are sound and are extendible to accommodate new features and requirements