Realizing Asymmetric Datarates via Energy Efficient Ethernet (EEE)

Dr. Philip Axer

Agenda

- Introduction to EEE
- Timing and Latency
- Partitioning
- Configurations, Power & Robustness
- Summary

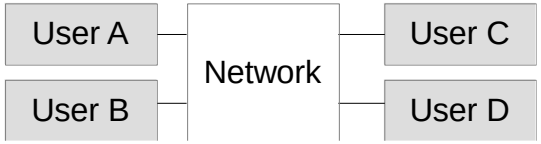
Introduction to EEE

The need for Asymmetry

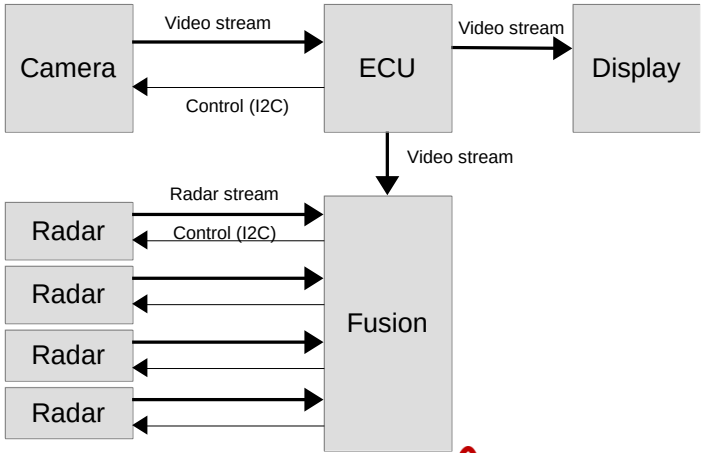
- Enterprise/Datacenter:
  - Dynamic traffic pattern. Exact use-case known by the users. Usually, service contracts that give guarantee
  - Symmetric data rates (except for last-mile)

- Automotive has obvious source/sink relation
  - Majority of data is streaming data that is generated in the Zone and flows to central processing
  - Control data flows in opposite direction (actuation, sensor control, handshaking messages such as TCP/IP, …)

Datacenter application



Automotive application



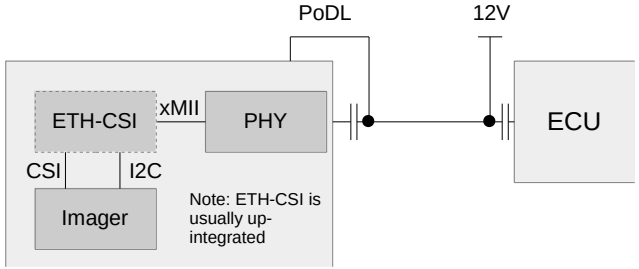
Asymmetric rates over symmetric Ethernet

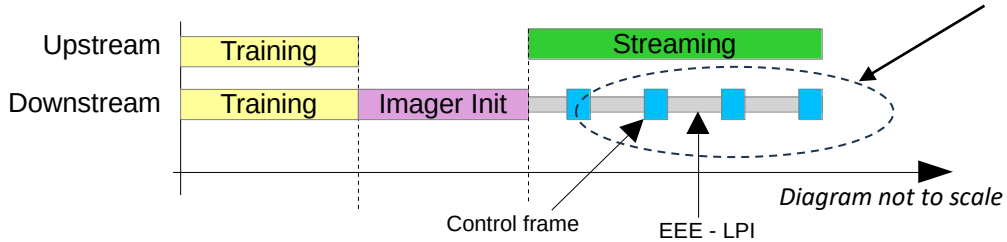
- Today‘s Ethernet (BASE-T1) is full-duplex and symmetric: same rate in both directions
  - Except for 10BASE-T1S
- All T1 speed grades >=100M use active IDLE and exchange data when no Ethernet frames are being transmitted.
  → **Power consumption is data-independent and under-utilized links use power**

- Energy Efficient Ethernet (802.3-2022 Clause 78) addresses this issue
- EEE is specified for all 1000BASE-T1, 2.5GBASE-T1, 5GBASE-T1, 10GBASE-T1 and 25GBASE–T1.

EEE Use-case: Imager

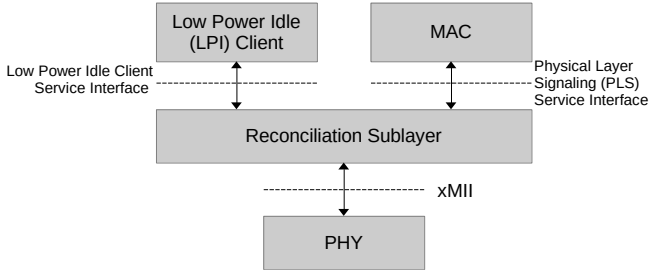
- Ethernet Imager with PoDL support
  (Example values)
  - I2C Control 400 kHz
  - 60 fps
  - 4k RAW10
- PoDL to supply radar or camera nodes
- Assuming I2C control activity linked to frame rate period of ~**17ms**, downstream link is mostly idle



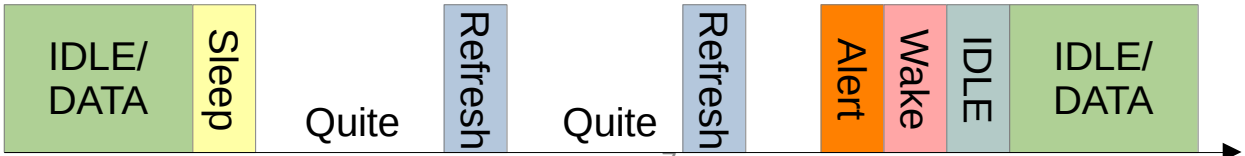


How dows this work in detail?

EEE Introduction



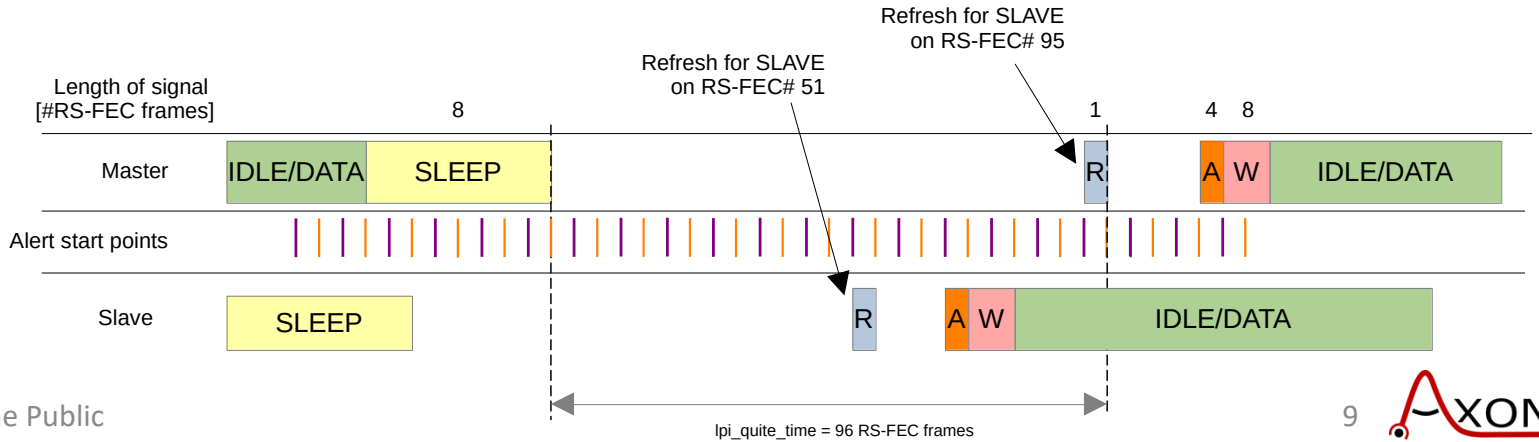
- **Low Power Idle (LPI) Client** decides when link goes into power saving mode (LPI mode)
- LPI is **asymmetric** and directions can go into LPI indepdentely, e.g.:
  - TX direction can be in LPI and RX direction remains active
- During LPI state, PHY sends **cyclic REFRESH** sequence to maintain timing lock
- Wakeup is signaled via an **ALERT** (Link sync sequence), followed by a **WAKE** (training sequence)



Timing and latency

EEE Timing

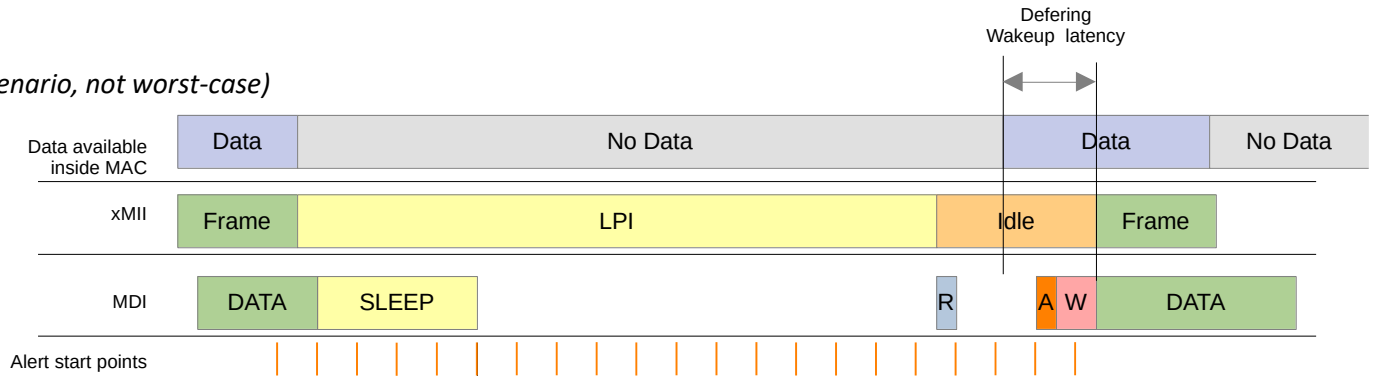
- The PCS uses Reed-Solomon frames (RS-FEC frames)
- In EEE everything happens on RS-FEC frame boundary.
  - RS-FEC frame counters serves as common time base for MASTER and SLAVE
  - Synchronized within 0 to 4 RS-FEC frames (better for speeds < 10G)
- REFRESH and ALERT are interleaved between MASTER/SLAVE to avoid interference
  - REFRESH once per cycle.
  - Distinct start points for ALERT



EEE Timing cont.

- When new data is available, MAC needs to defer until link is available → **latency**
- Two EEE flavors (user setting): **“EEE”** and “**Slow-Wake**”
  - Slow-Wake has **one** alert start points per cycle → **higher latency but more power saving potential**
- **Latency** for wakeup (assuming the wakeup happens after SLEEP signal has completed *)**:**
  - Waiting for the next ALERT opportunity + ALERT duration + WAKE duration
  - Normal: 10G: **6.4 us** 5G: **12.8us** 2.5G: **25.6us**
  - Slow: 10G: **34.56us** 5G: **69.12us** 2.5G: **138.24us**

(illustrative scenario, not worst-case)



*) based on “Case 2” and “Case 4” latency 802.3 Clause 78.5

Putting the latency into perspective

- Latency is large, when compared to Ethernet frame
- Latency is small in relation to I2C transaction (for 5G and 10G much faster)

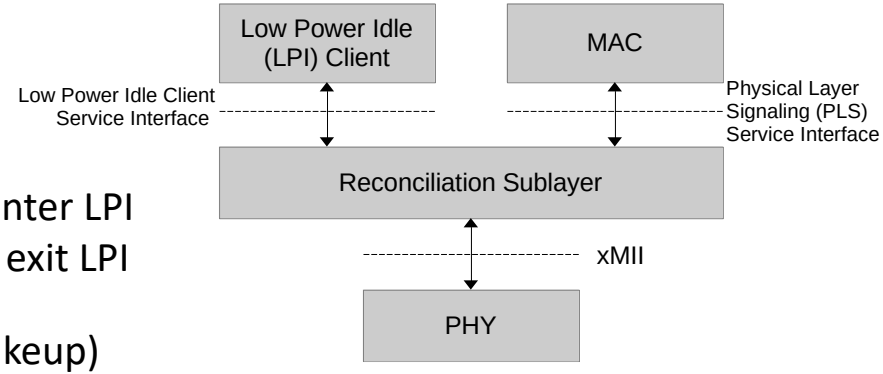
	Latency
2.5G Wakeup latency (EEE)	25.6us
2.5G Wakeup latency (Slow-Wake)	138.24us
Frame period at 60 Hz	16666 us
1500 octet frame @ 2.5G	4.8 us
64 octet frame @ 2.5G	0.2 us
I2C Read (400 kHz, 1 Byte, no clock streching)	72.5 us
I2C Write (400 kHz, 1 Byte, no clock streching)	97.5 us

Partitioning

LPI Client

- The 802.3 view is to put the LPI client on-top of the xMII interface (i.e. inside a SoC/switch)
  - MAC sends LPI control words through xMII to enter LPI
  - MAC sends IDLE control words through xMII to exit LPI
  - MAC needs to obey LPI timing (how long to send LPI, how long to defer on wakeup)

802.3 view on LPI Client



- The IEEE 802.3 view on LPI is very loose → the LPI client's behavior is not specified
- Not specified **when** LPI is entered and exited

  Heuristics to enter LPI are completely left to the implementer

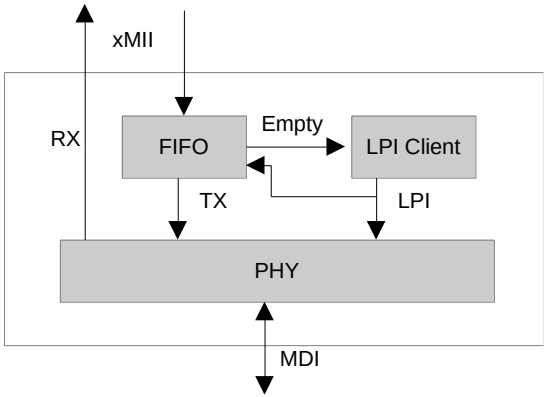
  → no relation to 802.1Q queuing, shaper, etc

LPI Client (cont.)

- Broad spectrum of EEE support in MACs
  - some MACs **do not implement EEE at all**
    - LPI codewords not generated/understood
  - some MACs **implement poor EEE**
    - For example: Software register to manually enter LPI via software
    - LPI is not *automatically* entered/left based
  - And some MACs implement **good EEE**
    - For example: LPI entered based on 802.1Q decisions (i.e. queue fill levels)

OEMs/Tier1 companies cannot pair every EEE PHY with every SoC

Autonomous EEE

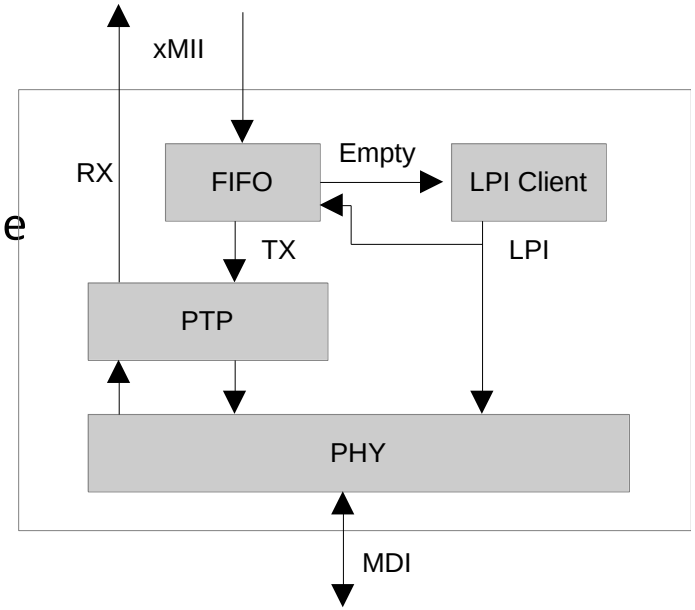


- A PHY product can easily make EEE decisions on its own, without MAC involvement
  - LPI Client inside PHY
- During the wakeup time, the PHY needs to buffer arriving TX frames *):
  - **11.2 kb** for EEE (for 2.5G T1)
  - **46.4 kB** for Slow-Wake (for 2.5G T1)
- Viable solution for EEE
- Old idea and products are already out since the beginning of EEE

*) based on "Case 1" and "Case 3" latency 802.3 Clause 78.5

Timestamping - Autonomous EEE

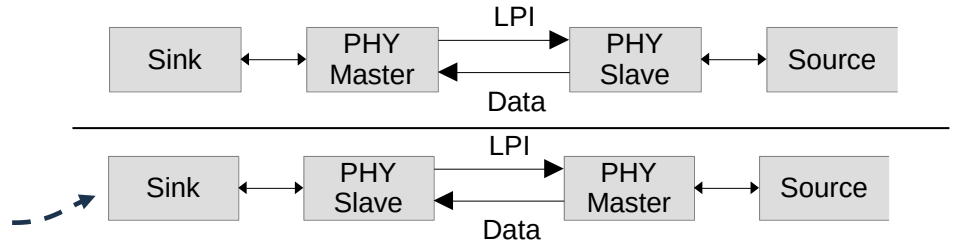
- PHY latency not constant anymore because the FIFO is buffering
  → wakeup latency
- MAC-based PTP timestamps may be incorrect due to FIFO variability to accommodate PHY wakeup
- Can be easily overcome with the following solutions
  - PHY timestamping – Timestamp the actual packet release time at the PHY
  - Meassure PHY resistence time and compensate MAC timestamp accordingly



Power

Configurations & Power consumption

- All MASTER/SLAVE configurations are possible and part of future IOPT tests
  - SyncE-like scenario possible: “controller” as MASTER distributes clock to SLAVE sensor nodes



- Power consumption depends on LPI states in TX/RX direction:

Mode	Qualitative Power consumption
TX IDLE/DATA, RX IDLE/DATA	High
TX LPI, RX IDLE/DATA	Medium
TX IDLE/DATA, RX LPI	Low
TX LPI, RX LPI	Lowest

Some loose ends…

- OPEN TC-16 is trying to fill in some blanks
  - EEE interoperability & conformance tests
  - EMI (?)
- Some points not addressed/discussed yet
  - Bridging the gap between 802.1Q and 803.2 EEE
  - *Do we need a standard when to enter/exit LPI based on the queuing/prios/shapers/gates/…?*

Dashboard Overview & Summary

Asymmetric Ethernet via 802.3ch + EEE

- Plug-and-play style, works with standard MACs Solution is ready
  → sample and near-final silicon available
- **Everything is Ethernet** & builds on-top of proven T1 tech
  → less diverse technology ecosystem
- High data rate available in upstream direction
  → Fast OTA, firmware update
  → Diagnostic scenario in which data is routed through alternative paths
- Simple Interoperability ecosystem, asymmetric but without chipset solutions
  → less combinations of things to test against

Summary

- EEE available today – contrary to other technology
- Scalability → EEE also available for future T1 PHYs (25BASE-T1)
- EEE incurs additional latency during wakeup scenario
  - Compared to I2C latency (the control channel to sensors), latency is negligible
- LPI Client (EEE decision) can be in MAC or PHY
  - Easy to retrofit SoC that do not have EEE support with Autonomous EEE PHY implementation
- Some ecosystem work (IOPT, conformance, 802.1Q, …) to be done → OPEN Alliance TC16