Link Sync, Auto-Negotiations, Crystal-less

William Lo

November 10, 2025

Agenda

- Clarify the use case for Auto-Negotiation vs Link Sync
- Show no need to negotiate speed in Link Sync use case
- Refute the notion that ACT Link Sync cannot negotiate speed if desired
  - Capability is not there because there is no need
- Show crystal-less is an issue for both ACT and TDD solutions with current definition of Auto-Negotiation
- Show a path to make Auto-Negotiation work with crystal-less
  - Not advocating for this. Just showing it is possible.

Auto-Negotiation vs Link Sync

- Auto-Negotiation
  - Generalized framework to select highest capability for devices implementing multiple PHY types.
  - Assumption is same set of cabling can support various advertised capabilities
  - Permits leader/follow negotiation in addition speed capabilities.
  - Expandable to new PHY types
  - Use case - Plug and play environments
- Link Sync
  - Fixed capability, fixed leader/follower configuration
  - Link Sync mechanics may differ between different capabilities
  - Use case – Engineered networks

Link Sync for Speed Negotiations

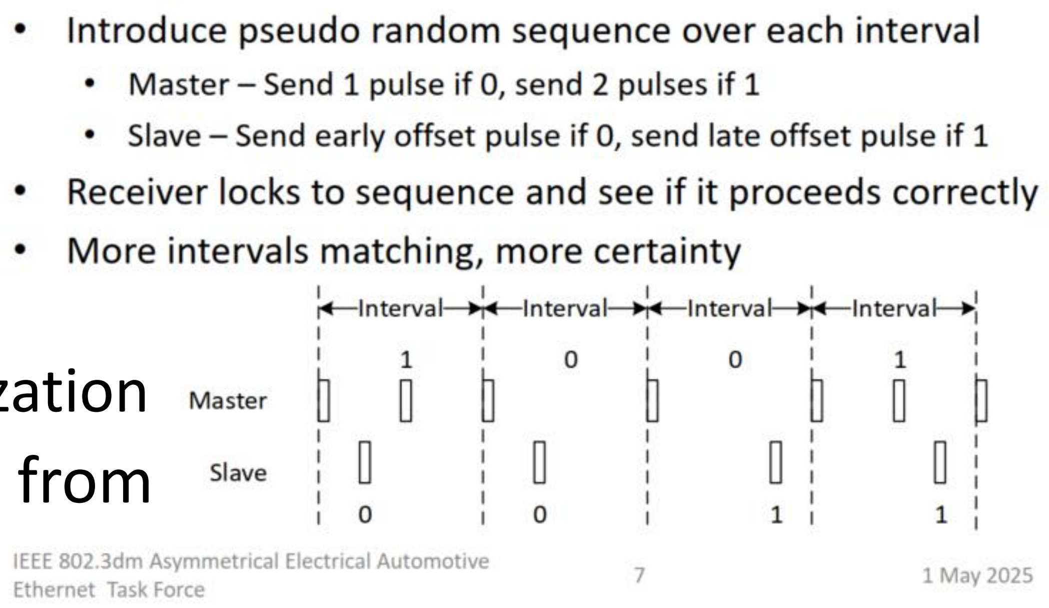
- Link Sync is for engineered systems
  - Speed and leader/follower is known a-priori
- If negotiation capability is defined then very limited set of capabilities
  - Only capabilities sharing the same link sync mechanics can be negotiated
- Increases complexity
  - At minimum need to define the negotiation mechanics for particular link sync
  - Need to test that mechanics implemented correctly

Crystal-less and Auto-negotiation

- Gauthier_Wang_3dm_01c_091525.pdf slide 20
  - Claims Clause 98 Auto-Negotiation does not work with Crystal-less
  - Claims TDD has “free” Auto-Negotiation in startup
  - Implies ACT link sync cannot exchange speed

- TDD “free” Auto-Negotiation cannot support switch side devices desiring to negotiate between 802.3ch and 802.3dm devices
  - i.e Plug and play switch port to connect to either 802.3dm camera, or 802.3ch backplane

ACT Link Sync Can Be Made To Support Speed Negotiation



- Introduce pseudo random sequence over each interval
  - Master – Send 1 pulse if 0, send 2 pulses if 1
  - Slave – Send early offset pulse if 0, send late offset pulse if 1
- Receiver locks to sequence and see if it proceeds correctly
- More intervals matching, more certainty

Interval
Interval
Interval
Interval
1
0
0
1
Master
Slave
0
0
1
1

IEEE 802.3dm Asymmetrical Electrical Automotive Ethernet Task Force 7 1 May 2025

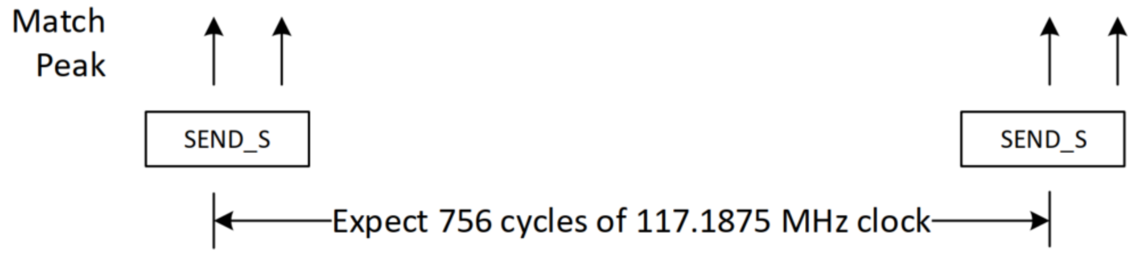
- Lo_3dm_01_050125.pdf
  - Passes randomized bits
- Lo_3dm_01_091525.pdf
  - Simplified scheme – removed randomization
  - Simplification possible based on work from
    - razavi_01_3dm_01a_July_2025.pdf
    - zherebtcov_Jonsson_3dm_01a_09_04_25.pdf
- Coded bits can be sent during ACT link sync instead of randomized bits to exchange speed info if desired
- Nevertheless use case does not call for speed exchange

Auto-Negotiation with Crystal-less

- Using Crystal-less a-priori determines who is leader/follower
  - Engineered system at minimum on which device must have crystal
- If Auto-Negotiation does not work with Crystal-less it applies equally to ACT and TDD
- It is possible to modify Auto-Negotiation to extract timing information while remaining backward compatible.

Modifying Auto-Negotiation for Crystal-less

- Use technique shown in Lo_01_0325.pdf
  - Slides 14, 15 shows how to use SEND_S spacing to extract timing
  - Key is to take advantage of exact timing is known between SEND_S



- Autoneg DME page header is Golay sequence
  - Matched filter to extract peaks much like SEND_S

Modifying Auto-Negotiation for Crystal-less

- Eliminate variability in timing between DME pages
- If device advertises 802.3dm capabilities then leader shall transmit Autoneg DME pages at exact known intervals.
  - Example tighten back_off_timer
    - from (6805 to 6925 ns) + rnd(0 to 15) x (2120 to 2240ns)
    - to 6840 + rnd(0 to 15) x 2160 ns (only 16 possible positions)
  - Still backwards compatible
- Approach requires more investigation but possible
  - Not saying use case requires this

Summary

- Auto-Negotiation plug and play generalized solution to resolve multi-capability PHYs and leader / follower
- Is there a real need to add complexity to startup outside of Auto-Negotiation to resolve speed for limited number of capabilities?
- Refuted the notion that ACT Link Sync cannot be modified to negotiate speed if desired
- Outlined a path to make Auto-Negotiation work with crystal-less

THANK YOU