ANALOG DEVICES

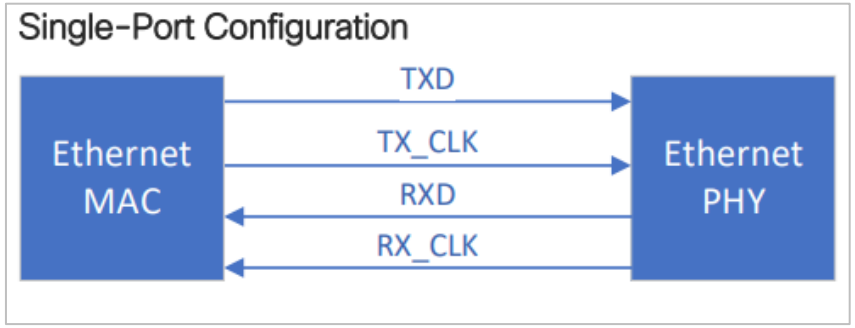
AHEAD OF WHAT'S POSSIBLE™

Symbol Encoding for POPI for 10/100M PHYs

Brian Murray
Jacobo Riesco

Overview of Proposals for the Implementation of POPI

- For a single 10M or 100M PHY, it should be possible to implement POPI as a simple 4-pin interface with single ended IOs and digital logic
  - Operate at signalling rates similar to existing GE PHY MAC interfaces with encapsulation including MII & RS[1] control codes
  - Have a sideband channel for embedded management including Ingress and Egress timestamps
  - Use a scrambler to ensure spectral integrity of the serial data
  - Source synchronous with optional clock/data recovery
  - POPI for a multi-port MAC-PHY interleaves the port data symbols with identification of Port 0 Idle symbols
  - Exchange POPI link configuration at power-up and when there is a change in the POPI signalling rate
  - Operate in a low power mode when all the PHY links are down



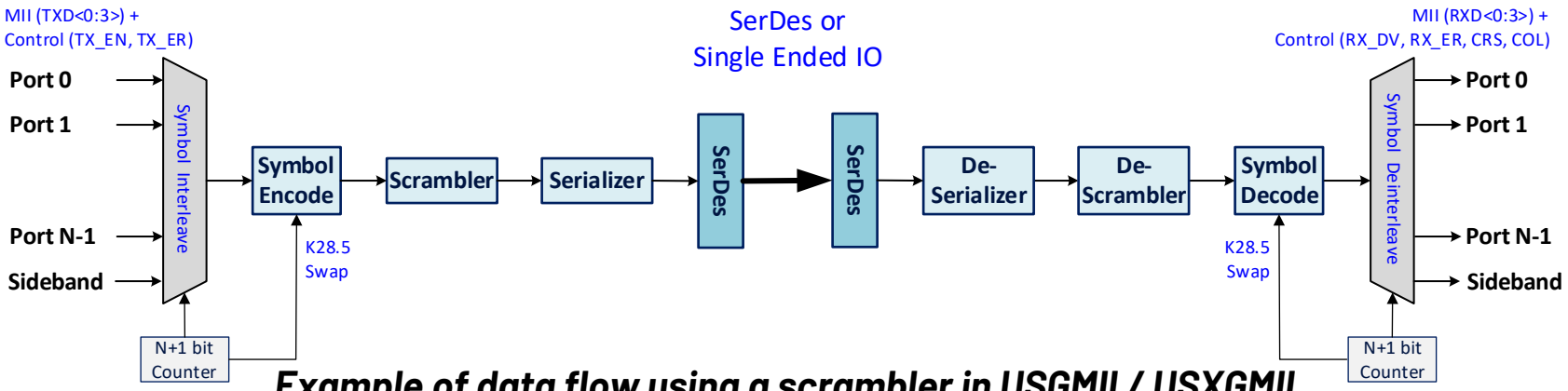
Pin Optimized PHY Interface, Call For Interest Consensus Building (page 23)

- For higher speed PHYs and multi-port (Quad / Octal) use a 4-pin SerDes
  - xMII & RS[1] control codes with sideband channel for management and Ingress and Egress timestamps
  - Source synchronous with clock/data recovery and scrambler for spectral integrity of the serial data
  - Exchange POPI link configuration at power-up and operate at lower speed when all the PHY links are down
  - Interleaves the port data symbols and sideband symbols with identification of Port 0 Idle symbols

[1]*Reconciliation Sub-layer Clause 36 control codes and control codes associated with Clause 46 sequence ordered sets*

Data Encapsulation and Encoding

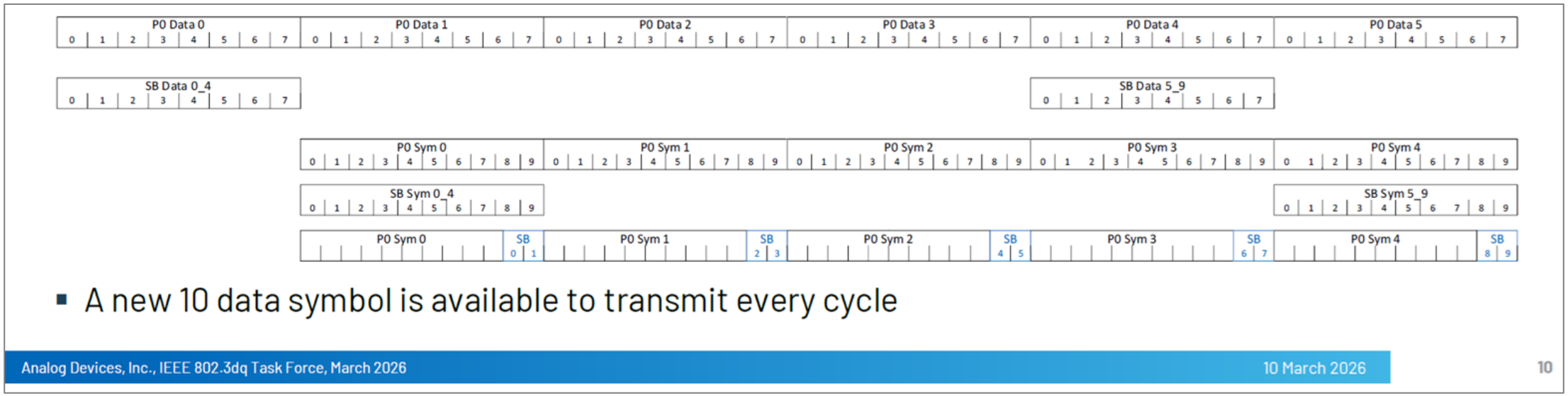
- Data encapsulation is necessary to support serialization of the data to achieve a low pin count (4-pin) interface
- Existing lower speed SerDes (up to 5Gb/s) use 8B/10B encoding and higher speed SerDes (10Gb/s and greater) use 64B/66B encoding
  - The 8B/10B encoding also provides symbol alignment, controls the signal disparity, ensures lots of transitions, is run length limited
  - The 64B/66B encoding provides encapsulation and regular synchronization information and include a multiplicative scrambler to ensure well behaved disparity and to ensure lots of transitions - this is achieved on a statistical basis



Example of data flow using a scrambler in USGMII / USXGMII

Interleave of Data and Sideband Symbols

- One proposal to support the sideband channel for the MAC-PHY POPI interface is to interleave a sideband symbol every 4 data symbols
  - See Single 10M or 100M PHY POPI Requirements – March 2026 slides 7, 8, 9 & 10
  - This approach works well for multi-port MAC-PHY interface and has no impact on latency (slide 8)
  - However, for a single port MAC-PHY interface, interleaving complete sideband symbols impacts latency (slide 9)
  - A proposed solution (slide 10) to evenly interleave the sideband bits every data symbol



  - This achieves the goal (slide 7) that "**POPI should use a block encoding that ensures a new data byte is available each cycle for encoding and transfer to the line**" which is very important at lower speeds

Disadvantage of 8B/10B Encoding for POPI

- Given that existing lower speed SerDes used 8B/10B symbol encoding it has been considered as an option in initial proposals for the implementation of POPI
- But major disadvantage of 8B/10B encoding is that it is very inefficient
  - There is a 25% overhead on the data rate encoding using 8B/10B
- A second disadvantage is error propagation from one symbol to the next
  - This is especially problematic with a multi-port MAC-PHY interface as an error on one port propagates to the next port
  - Note, using a multiplicative scrambler also has the disadvantage of error propagation
    - This is not an issue with the additive scramblers normally used in PHYs (802.3dg, cg. ….)
- Given that we already need to include scrambling to ensure the spectral integrity of the serial stream many of the features of 8B/10B are redundant
  - Using a scrambler provides control of disparity, transitions and run length on a statistical basis
- So we just need symbol encoding for encapsulation, support for the xMII & RS control codes and for symbol alignment and scrambler lock during Idle
  - At power up before we have PHY links we will always start with Idle signalling

8B/9B Symbol Encoding

- 8B/10B encodes 8-bits into 10-bits so that there is sufficient coding space to support all the 8B/10B features, including unique COMMA symbols
  - 8B/10B is also very (very) efficient from a gate implementation point of view, which is no longer as important many, many technology generations later
  - For POPI we just need to be able to encode a set of control codes and a set of idle symbols
- Encoding 8 bits into 9 bits (8B/9B) has more than enough coding space to provide data encapsulation and all the control codes we need
  - 9B encoding has a total of 512 codes available
  - 256 of these are required to encode the 8-bit data
  - Propose using 32 codes for Idle for symbol alignment and to ensure for randomization for spectral integrity as was done in 802.3dg
    - If we receive a large number of codes not in Idle codebook we know the symbol alignment is incorrect
    - In 802.3dg 32 codes for PAM-2 training proved to be sufficient for spectral integrity
  - Propose having 4 codes for each control code selected using the scrambler bits so that we also have randomization for the control codes
- Using 8B/9B encoding with one sideband bit per symbol supports a 12.5Mb/s sideband channel with POPI signalling at 125 MHz
  - 2 sideband bits for 25 Mb/s with 137.5MHz POPI signalling and 3 bit for 37.5 Mb/s and 150 MHz signalling

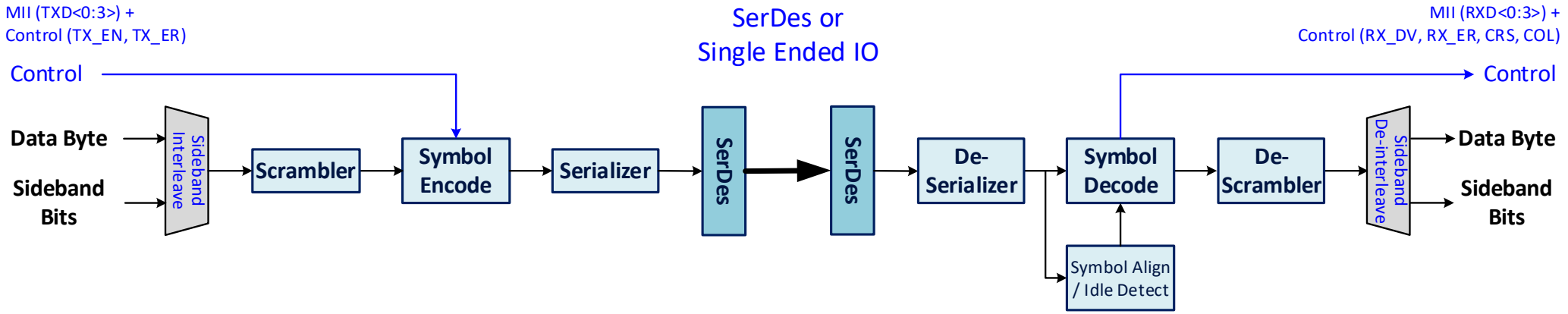
Allocation of 9B Codes

- Example allocation of 9B codes to Data, Idle and Control codes is shown here
  - Randomize Idle and Control codes based on scrambler bits to ensure spectral integrity
  - 32 entries in the Idle codebook for symbol alignment; two Idle codebooks so can identify Port 0
  - Multiple SSD and ESD codes for robustness

Type	Number	Randomization Factor	Disparity	Transition	Number of Codes
Data	256	x 1	±1, ±3	3 → 9	256
Idle 1	32	x 1	±1, ±3	3 → 9	32
Idle 2	32	x 1	±1, ±3	3 → 9	32
SSD (0→ 3)	4	x 4	±1, ±3	3 → 9	16
ESD (0→ 3)	4	x 4	±1, ±3	3 → 9	16
Control	8	x 4	±1, ±3	3 → 9	32
Spare Control 1	16	x 4	±5	2 → 9	64
Unused	31	x 2	±5, ±7	1 → 9	62
COMMA / MARK	2	x 1	±9	0	2
					512

Block Diagram Showing the Data Flow

- The following is a block diagram of the data flow for 8B/9B encoding
  - One sideband bit is interleaved with each data byte
  - The 9-bits of data + SB are scrambled using a standard 33-bit additive scrambler
  - The scrambled 8-bits corresponding to the data are encoded 8B/9B
  - Balanced disparity in the Idle and control codes can optionally be used to control running disparity
  - The encoded 9-bits plus one sideband bit are serialized, resulting in 125 MHz POPI signalling



Summary

- A proposal has been presented that uses 8B/9B symbol encoding with interleave sideband bits for POPI for a 100M single MAC-PHY interface
  - This operates at a 125 MHz signalling rate supporting a 12.5 Mb/s sideband channel with very low latency by mapping each byte at the MII to a byte at the PHY PCS
    - Achieves the goal of **ensuring a new data byte is available each cycle for encoding and transfer to the PCS** which is very important at lower speeds
  - Has the advantage of being much more efficient than using 8B/10B encoding and allows an implementation of POPI using single ended IO to use the same signalling rate as existing MAC-PHY interfaces
    - This allows POPI to be added to existing designs with only digital logic changes
  - The proposal can be extended to support higher sideband rates of 25 Mb/s and 37.5 Mb/s at lower POPI signalling rates than possible using 8B/10B encoding
    - Supports multiple sideband rates as an efficient trade-off with the POPI signalling rate

Questions ?