ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

POPI Requirements for 10/100M PHYs

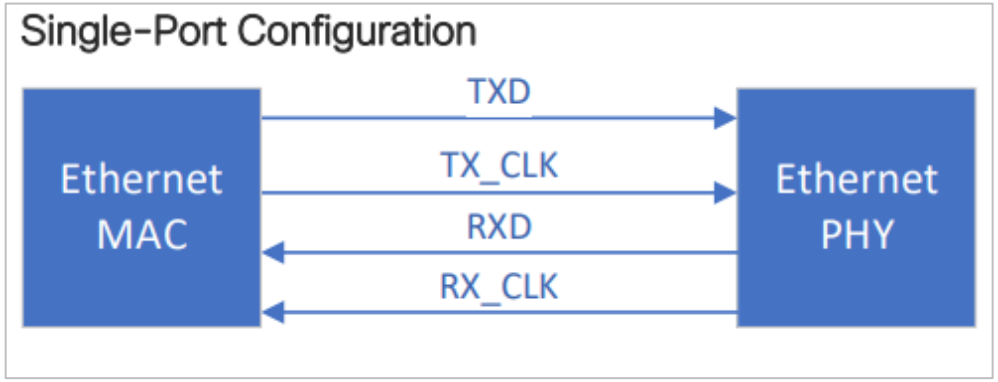
Brian Murray
Jacobo Riesco

Introduction

- This presentation is a continuation of our January, March and April presentations to develop the POPI requirements for a 10M and 100M single PHYs
  - With particular focus on the 802.3dg 100BASE-T1L PHY standard (under development)
  - Include the requirements for a full-duplex 10BASE-T1L PHY
  - Include the requirements for a half-duplex 10BASE-T1S PHY using PLCA
  - And be compatible with legacy full/half duplex 100M PHY standards, e.g. 100BASE-TX and 100BASE-T1
- The requirements discussed include the following:
  - The pin interface requirements
  - Ordered sets for encapsulation and control signalling
    - Start and end of packet, carrier extend, error propagation, LPI, remote/local faults, PLCA and configuration
  - Interleave of PHY and sideband data streams / interleave of multi-port PHY and sideband data streams
  - Embedded sideband channel or IPG for the ingress and egress time stamps for PTP
  - Embedded sideband channel for MDIO and link information
- Follow a consistent approach and logical format for POPI for lower and higher speed PHYs and for single and multi-port PHYs

Review of POPI Pin Interface for a 100M Single PHY

- For a single 10M or 100M PHY the desire is to have a simple low pin count interface using standard IOs and digital logic
  - 100M data with overhead for encoding of control/encapsulation (8B/10B would be 25% overhead)
  - Add 25% overhead for sideband channels for PTP time stamping and MDIO
  - Operate at a raw data rate on the pins of 125 to 156.25 Mb/s
  - Use single ended IOs to avoid the complexity of a SerDes while still keeping to a very low pin count
- Use a simple 4-pin interface with single ended IOs
  - TXD, TX_CLK, RXD, RX_CLK
  - Same pin count as a SerDes (4 pins)
  - Signalling rates are similar to existing GE PHY MAC interfaces (< 250 Mb/s per pin)
  - Clock/data recovery is optional
  - Source synchronous
  - Proven technology, low power/area, available in every technology node
    - An area efficient 1G SerDes might get close to the area of an RGMII MAC interface with 12 pins, but not 4 pins



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

Encapsulation and Control Signalling

- Could use 8B/10B and Clause 36 ordered sets for control and encapsulation
  - This is a proven approach used in SGMII, QSGMII and USGMII
  - IDLE ordered sets are used for idle
  - Encapsulation ordered sets are used for the start and end of packet, carrier extend, error propagation
  - LPI ordered sets are used for Energy Efficient Ethernet
  - Configuration ordered sets
    - In clause 37 these convey information about the 1000BASE-X link
    - POPI could use Configuration ordered sets to configure the POPI link
    - Information about the PHY link should be in a sideband channel
- We will need to amend/supplement these ordered sets to include fault signalling, PLCA and maybe others
  - Add Assert Local Fault and Assert Remote Fault from clauses 22, 46 and 190
  - Add PLCA Beacon and Commit Request / Indication from clause 22 and 148

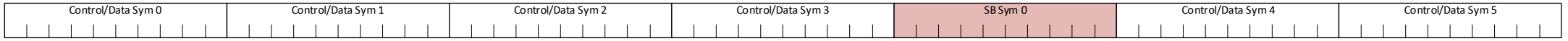
Table 36–3—Defined ordered sets

Code	Ordered Set	Number of Code-Groups	Encoding
/C/	**Configuration**		Alternating /C1/ and /C2/
/C1/	Configuration 1	4	/K28.5/D21.5/Config_Reg[a]
/C2/	Configuration 2	4	/K28.5/D2.2/Config_Reg[a]
/I/	**IDLE**		Correcting /I1/, Preserving /I2/
/I1/	IDLE 1	2	/K28.5/D5.6/
/I2/	IDLE 2	2	/K28.5/D16.2/
	Encapsulation		
/R/	Carrier_Extend	1	/K23.7/
/S/	Start_of_Packet	1	/K27.7/
/T/	End_of_Packet	1	/K29.7/
/V/	Error_Propagation	1	/K30.7/
/LI/	**LPI**		Correcting /LI1/, Preserving /LI2/
/LI1/	LPI 1	2	/K28.5/D6.5/
/LI2/	LPI 2	2	/K28.5/D26.4/

[a]Two data code-groups representing the Config_Reg value.

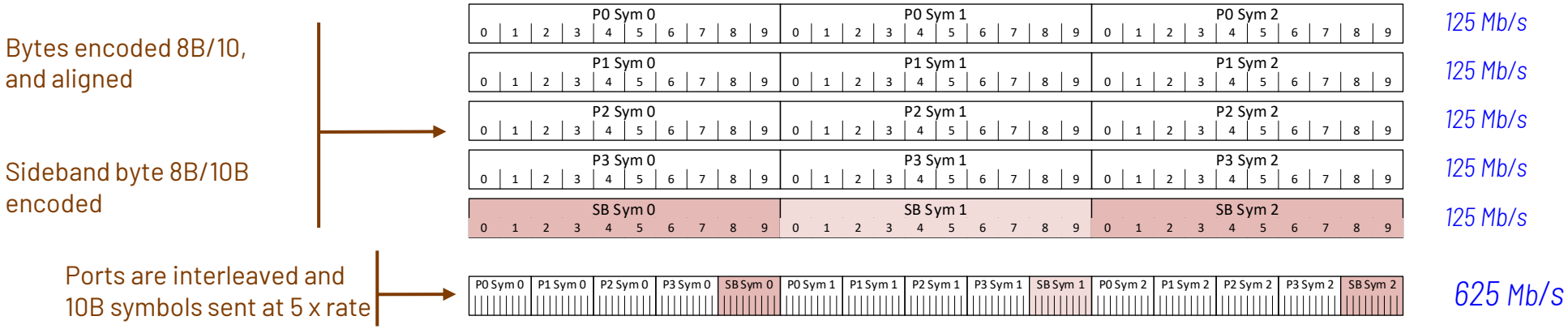
Interleave of Sideband and Data Channels

- For a single-port PHY, POPI can interleave PHY and sideband encoded symbols
  - An interleave ratio of 4:1 gives us 25 Mb/s for the embedded sideband channel
  - Run the POPI interface 25% times faster than the encoded payload rate



156.25 Mb/s

- For a multi-port PHY, POPI can share a sideband channel over a number of ports
  - So we can interleave a 100M sideband channel with the 100M PHY ports
  - A quad is a very common sub-multiple, so one obvious choice is to interleave a sideband every 4 PHY ports
  - This gives us 25 Mb/s for the embedded sideband channels for each PHY
  - Run the POPI interface 25% times faster than the multi-port encoded payload rate
    - The 4 data bytes are each encoded for example as 8B/10B and sent first, followed by the encoded sideband byte

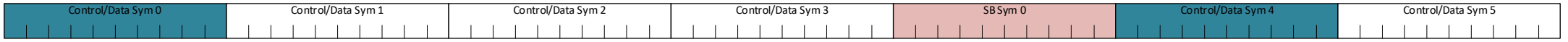


Identification of Port Number and Sideband Channel

- In 8B/10B encoding COMMA codes allow easy alignment of the 8B/10B symbols within the serial stream
  - There are guaranteed bit transitions in the 8B/10B symbols to support clock data recovery
- In POPI we interleaved the data and sideband symbols in a known ratio and in a multi-port PHY interleaved the ports for a known number of ports
  - For a single PHY we need to identify the first data symbol after the sideband symbol
  - For a multi-port PHY we need to identify the data symbols for Port 0; Port 1, 2 … follow in sequence and the sideband symbol follow the last port, Port N-1
- QSGMII and USGMII swap the K28.5 codes with K28.1 to identify Port 0
  - The exact same scheme can be used for POPI for a single and multi-port PHYs
  - For a single PHY swap the K28.5 codes with K28.1 for the first data/control symbol after the sideband symbol and every 4^{th} data symbol after that
  - For a multi-port PHY swap the K28.5 codes with K28.1 for the data/control symbols for first port

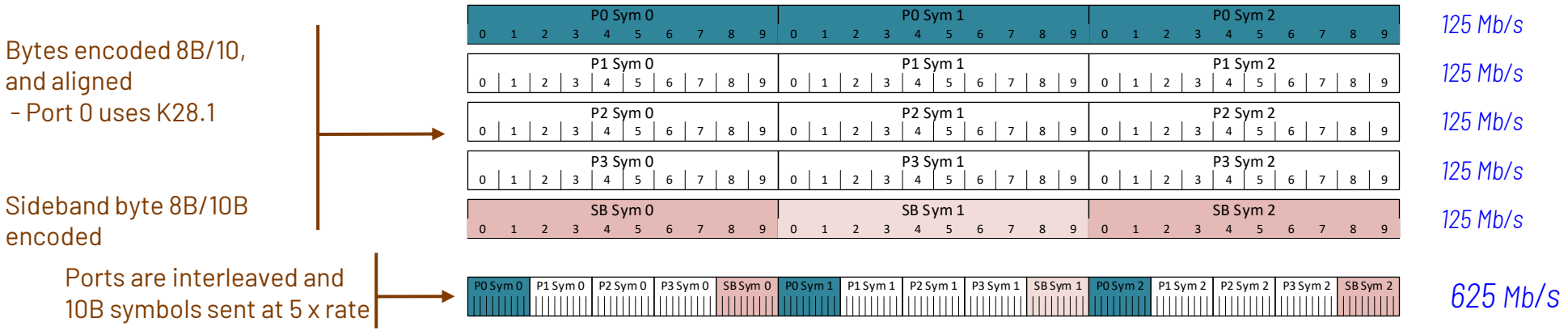
POPI for a Multi-port PHY – Interleave Ports & Sideband

- For a single-port 100M PHY with an interleave ratio of 4:1
  - Swap the K28.5 control codes with K28.1 for the first control/data symbol and every 4th control/data symbol after that



156.25 Mb/s

- For a multi-port PHY where we interleave a 100M sideband channel with the 100M PHY ports every 4 PHY ports
  - For Port 0 control/data symbols swap the K28.5 control codes with K28.1

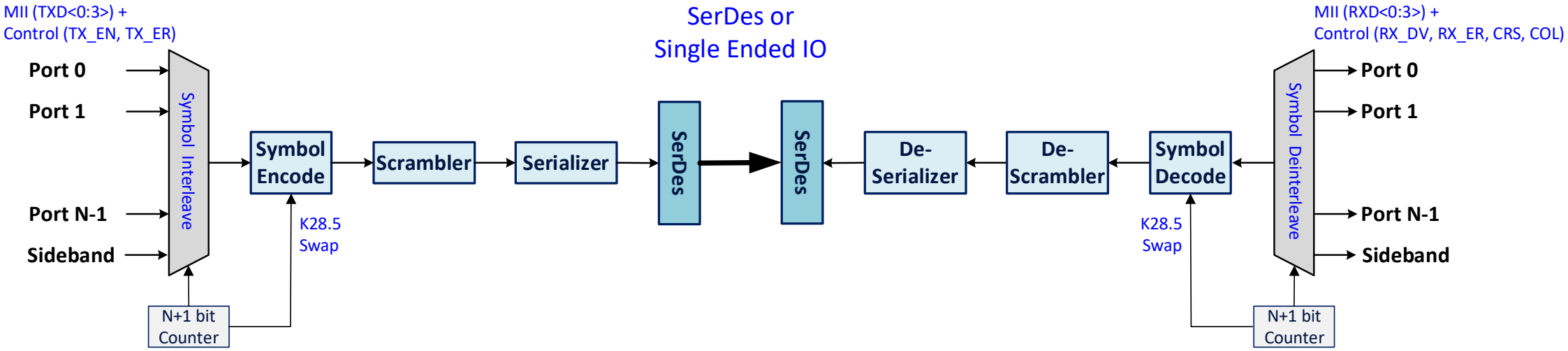


Scrambling of the Bits on the Serial Interface

- USGMII added scrambling of the signal after interleave and before been sent to the SerDes
  - This ensures randomization of signals on the line and better SerDes behaviour
  - Better EMI performance as control signals are also randomized
  - Limits run's of 1's or 0's based on statistics – which you may also get from the symbol encoding
  - Guarantees bit transitions for clock data recovery based on statistics - may also get from encoding
- USGMII uses a self-synchronizing scrambler
  - This has the advantage of being easy to lock but has the disadvantage of propagating errors
- Propose POPI use scrambling for randomization and EMI
  - Propose using a standard additive scrambler – Exclusive-OR the serial data with an LFSR
  - A PHY link takes time to come up so will always have idle signalling to start with so it is easy to lock the scrambler

Block Diagram of Scrambler on the Serial Interface

- The following block diagram illustrates the data flow for POPI
  - For simplicity just showing MAC to PHY encode/decode path
    - PHY to MAC is the inverse and obviously both are combined on POPI over single ended IO or a SerDes
- Refer to the appropriate reference documents for more detail on how this was done in USGMII / USXGMII



PHY Link Information

- In industry standard MAC-PHY interfaces like RGMII or xSGMII, the PHY link information is sent to the MAC when the PHY link status changes
  - This includes link status, speed, duplex, EEE capability, etc.
  - This is exchanged using in-band status bits or Clause 37 Configuration ordered sets or Clause 46 sequence ordered sets depending on the interface
- For POPI we could send this information at link up and any time the link status changes using the sideband channel with a specific Macro Frame type
- Some Ethernet applications have a PHY Fast Link Down requirement
  - The PHY must detect and signal link down information within 10 to 100 μs
  - Typically signalled using a LINK_ST (link status) pin and typically used for fast fail over to redundant links
- What is the best way to signal link status when the link drops?
  - Do we signal this with Link Status (from clause 22.2.4.2.13)
  - Do we signal this with Receive link status (from clause 45.2.1.2.4)
  - Do we signal this with Local Fault (from clause 46.3.4)
  - Do we signal this by conveying the LINK_ST pin over POPI – as this is a signal that is used in practice
- This has been a lot of evolution across PHY clauses – POPI could try to converge these

Backup Slide for Link Information

22.2.4.2.13 Link Status

When read as a logic one, bit 1.2 indicates that the PHY has determined that a valid link has been established. When read as a logic zero, bit 1.2 indicates that the link is not valid. The criteria for determining link validity is PHY specific. The Link Status bit shall be implemented with a latching function, such that the occurrence of a link failure condition will cause the Link Status bit to become cleared and remain cleared until it is read via the management interface. This status indication is intended to support the management attribute defined in 30.5.1.1.4, aMediaAvailable.

45.2.1.2.4 Receive link status (1.1.2)

When read as a one, bit 1.1.2 indicates that the PMA/PMD receive link is up. When read as a zero, bit 1.1.2 indicates that the PMA/PMD receive link is down. The receive link status bit shall be implemented with latching low behavior.

46.3.4 Link fault signaling

Link fault signaling operates between the remote RS and the local RS. Faults detected between the remote RS and the local RS are received by the local RS as Local Fault. Only an RS originates Remote Fault signals.

Sublayers within the PHY are capable of detecting faults that render a link unreliable for communication. Upon recognition of a fault condition a PHY sublayer indicates Local Fault status on the data path. When this Local Fault status reaches an RS, the RS stops sending MAC data or LPI, and continuously generates a Remote Fault status on the transmit data path (possibly truncating a MAC frame being transmitted). When Remote Fault or Link Interruption status is received by an RS, the RS stops sending MAC data or LPI, and continuously generates Idle control characters. When the RS no longer receives fault status messages, it returns to normal operation, sending MAC data or LPI.

POPI Low Power Mode / Power Down Mode

- When the link is down a standard MAC – PHY interface like GMII, RGMII will typically operate at a lower clock frequency to save power
  - The interface can be configured to automatically operate at a lower speed when the PHY link is down, e.g. select 10M and clock at 2.5 MHz rather than 125 MHz or even no speed and no clock
  - Usually configurable in a PHY register to select the speed that should be used when the link is down
- POPI should support a similar mechanism to allow the MAC-PHY interface to enter a lower power modes when there is no need for the POPI interface
  - When the PHY link is down it may be desirable to put the POPI interface into a low power or power down mode
    - For example when the cable is unplugged or the PHY link is disabled
    - There is no data to be transmitted or received so no need for POPI – except if the MAC wants to read or write registers or the PHY wants to indicate that the link is starting to come up
- When the link is down the MAC can request the POPI link to go into a lower power mode
  - In the low power mode POPI would run at a low speed or stop all signalling and go into a near zero power mode
  - The MAC or the PHY should be able to wake up POPI or go to full rate by sending wake-up signalling
- POPI using single ended IO could run at 12.5 MHz in low power mode, sufficient for MDIO
- POPI using a SerDes could select a lower speed SerDes, e.g. 2.5 Gb/s for low power mode
  - Could drop the SerDes link for the power down mode and use wake-up signalling to exit the power down mode
  - Note, the typical SerDes link-up time is of the order of a milliseconds

Configuration of the POPI Interface

- There may be parameters of the POPI interface that we want to configure just after power-up and initialization of the POPI interface between the MAC and PHY
  - The number of ports being interleaved, the ratio of sideband data to port data, maybe specifics about the encoding
- POPI Link-up
  - A typical SerDes connection starts with encoded idle signalling at a specific clock rate
  - Either the clock rate is fixed and known or use the default low power mode signalling rate
  - The clock is recovered using the transitions in the data guaranteed by scrambling
  - The encoded symbols / frames are aligned, the scrambler is locked and the Idle signalling is decoded
- If the POPI configuration is fixed or agreed between the MAC and PHY the decoding of the part data and sideband data can be done immediately
  - Configuration ordered sets can be used to convey information about the POPI link
    - Just as they are used in Clause 36 to convey information about the 1000BASE-X link
  - We would define a POPI Config_Reg which could include the number of ports, the port interleave, the sideband interleave

Summary of Bandwidth Requirements for Sideband

- The following table is a summary of the bandwidth required for MDIO read and write and ingress and egress time stamps for PTP
  - Peak bandwidth is under worst case assumptions of smallest packets and high utilization
  - In practice (nominal bandwidth) there will be a typical case of a mix short and long packets and utilization will be lower that 100%
    - The bandwidth required for the Ingress timestamp can be much lower in the nominal case; lower percentage and lower absolute bandwidth

Channel	Peak and Nominal Rates	Nominal Bandwidth	Peak Bandwidth
MDIO	Peak = 6.25 MHz clock, all write or all read Nominal = 5 MHz clock, 40% write and 40% read	~2.00 Mb/s	6.25 Mb/s
Ingress TS	Peak = 4 bytes every 64 byte packet + 50% OH Nominal = 4 bytes every 512 byte packet + 50% OH	1.5625 Mb/s	12.5 Mb/s
Egress TS	Peak = 4 bytes 20% of 64 byte packet + 50% OH Nominal = 4 bytes 5% of 512 byte packet + 50% OH	< 0.1 Mb/s	2.5 Mb/s
		< 4.0 Mb/s	**21.25 Mb/s**

Use Inter Packet Gap (IPG) for the Ingress Timestamp

- The bandwidth requirements for MDIO and PTP are dominated by the Worst Case assumptions for the Ingress timestamp, ~60% of total bandwidth
  - In practice the bandwidth required for the Ingress timestamp for a typical case of a mix short and long packets would be much less; a lower percentage and lower absolute bandwidth

Channel	Peak and Nominal Rates	Nominal Bandwidth	Peak Bandwidth	Peak MDIO + Egress TS
MDIO	Peak = 6.25 MHz clock, all write or all read Nominal = 5 MHz clock, 40% write and 40% read	~2.00 Mb/s	6.25 Mb/s	
Ingress TS	Peak = 4 bytes every 64 byte packet + 50% OH Nominal = 4 bytes every 512 byte packet + 50% OH	1.5625 Mb/s	12.5 Mb/s	
Egress TS	Peak = 4 bytes 20% of 64 byte packet + 50% OH Nominal = 4 bytes 5% of 512 byte packet + 50% OH	< 0.1 Mb/s	2.5 Mb/s	
		< 4.0 Mb/s	**21.25 Mb/s**	**8.75 Mb/s**

- One simple way to deal with the variation of the bandwidth required for the Ingress timestamp wrt the Rx packet length is to use the IPG
  - Guaranteed IPG of 12 bytes – use 5 to 8 of these 12 bytes for the Ingress timestamp + overhead
  - Ingress timestamp would be 5 data symbols following directly after the packet Terminate symbol
- This would reduce the Peak Bandwidth required and would allow a 1:8 interleave or an even higher data rate for MDIO or other sideband transactions

Peak and Nominal Rates for a 10M PHY

- For 10M, the bandwidth requirements for MDIO and PTP is dominated by the MDIO and Worst Case assumptionsn, ~80%of total bandwidth
  - In practice the bandwidth required for MDIO would be less for 10M links
  - But still looks marginal at a 25% overhead
  - A 10M PHY may need a 100% overhead for the sideband channel, which is a 1:1 interleave

Channel	10M Peak and Nominal Rates	Nominal Bandwidth	Peak Bandwidth	Peak MDIO + Egress TS
MDIO	Peak = 6.25 MHz clock, all write or all read Nominal = 5 MHz clock, 40% write and 40% read	~2.00 Mb/s	6.25 Mb/s	
Ingress TS	Peak = 4 bytes every 64 byte packet + 50% OH Nominal = 4 bytes every 512 byte packet + 50% OH	0.15625 Mb/s	1.25 Mb/s	
Egress TS	Peak = 4 bytes 20% of 64 byte packet + 50% OH Nominal = 4 bytes 5% of 512 byte packet + 50% OH	< 0.01 Mb/s	0.25 Mb/s	
		< 2.5 Mb/s	**7.75 Mb/s**	**6.5 Mb/s**

POPI – Half-Duplex Operation

- Half-duplex operation (COL/CRS)
  - Support half-duplex PHYs like 10BASE-T1S, 100BASE-TX & 10BASE-T
  - All legacy non-standard MAC interface infer COL/CRS at the MAC side from RX_DV and TX_EN
  - If **NOT** PLCA there is no reason to do anything different for POPI
- If Half-Duplex and **NOT** PLCA infer COL / CRS at the output of POPI
- If PLCA need to signal CARRIER_STATUS and SIGNAL_STATUS to the MAC
  - In Clause 148 these are sent as CRS/COL over the MII
  - In POPI we could add Control Codes for CARRIER_STATUS and SIGNAL_STATUS

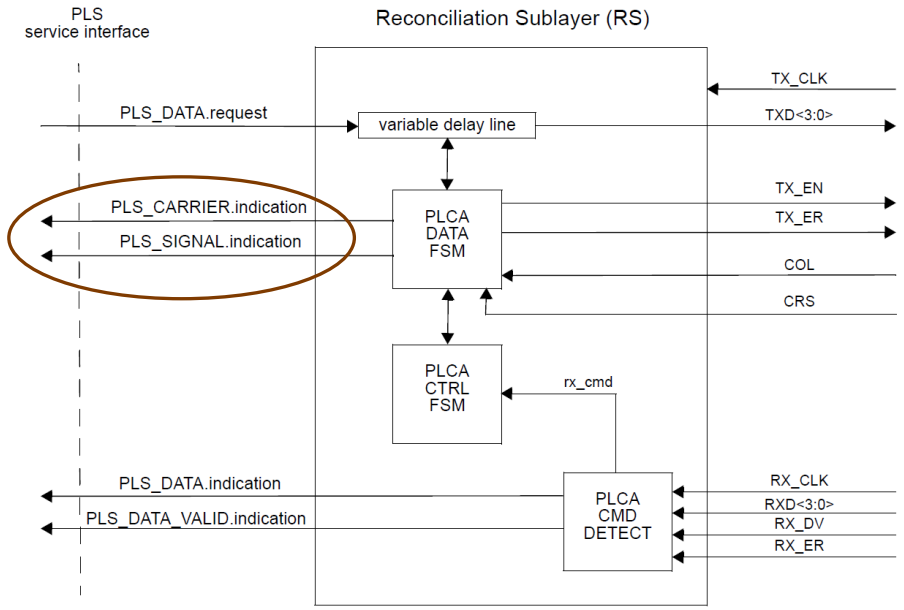


Figure 148–2—PLCA functions within the Reconciliation Sublayer (RS)

POPI for Multi-drop 10BASE-T1S

- Support a 10BASE-T1S PHY using PLCA by including PLCA primitives
  - Add Control Codes for PLCA Beacon and Commit Request / Indication from clause 22 and 148

The RS conveys the BEACON & COMMIT requests via MII interface

When the PHY receives a BEACON or COMMIT, it indicates this information to the RS by asserting MII signals

Table 22–1—Permissible encodings of TXD<3:0>, TX_EN, and TX_ER

TX_EN	TX_ER	TXD<3:0>	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Reserved
0	1	0001	Assert LPI
0	1	0010	PLCA BEACON request
0	1	0011	PLCA COMMIT request
0	1	0100 through 1111	Reserved
1	0	0000 through 1111	Normal data transmission
1	1	0000 through 1111	Transmit error propagation

Table 22–2—Permissible encoding of RXD<3:0>, RX_ER, and RX_DV

RX_DV	RX_ER	RXD<3:0>	Indication
0	0	0000 through 1111	Normal inter-frame
0	1	0000	Normal inter-frame
0	1	0001	Assert LPI
0	1	0010	PLCA BEACON indication
0	1	0011	PLCA COMMIT indication
0	1	0100 through 1101	Reserved
0	1	1110	False Carrier indication
0	1	1111	Reserved
1	0	0000 through 1111	Normal data reception
1	1	0000 through 1111	Data reception with errors

  - Add Control Codes to signal CARRIER_STATUS and SIGNAL_STATUS to the MAC?
- Note many implementations of 10BASE-T1S integrate the MAC and PHY
  - A 10BASE-T1S MAC-PHY typically communicates with a Host over SPI
  - Open Alliance have defined a 3-pin interface to an external PMD allowing the digital portion of the 10BASE-T1S PHY to be integrated with the MAC and host MCU and the analog (high voltage) portion of the PHY to be implemented as a stand alone part
  - In both of these cases POPI would not be used

Summary

- For a single 10M or 100M PHY, POPI should be implemented as a simple 4-pin interface with single ended IOs and digital logic
- Should operate at signalling rates similar to existing GE PHY MAC interfaces, e.g. 125 to 156.25 MHz
- Support MDIO, PTP and clause 36 like ordered sets as embedded interfaces with POPI to achieve the lowest pin count interface
- Swap K28.5 control codes with K28.1 to identify sideband symbols or Port 0
- Use a scrambler to randomize the serial data
- Support low power and power down modes when the PHY links are down
- Support configuration of the POPI link at power-up
- Support 10 Mb/s to 25 Mb/s for the embedded sideband channels
- Follow the same logical format for frames used for the embedded interfaces like MDIO and PTP as used by higher speed /multi-port POPI

Questions ?