ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

POPI Requirements for a 100M Single PHY

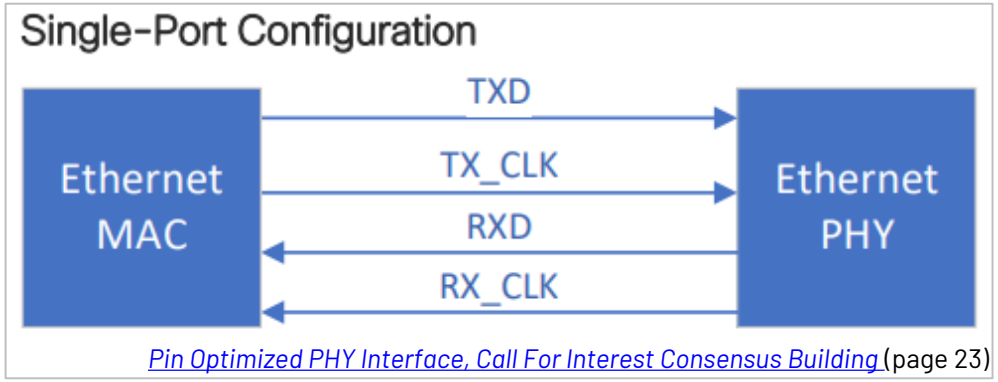
Brian Murray
Jacobo Riesco

Introduction

- This presentation describes what we believe are the requirements of POPI for a 100M single PHY
  - With particular focus on the 802.3dg 100BASE-T1L PHY standard under development
  - Also considering the requirements for a 10BASE-T1L PHY
  - And should be compatible with legacy full duplex 100M PHY standards, e.g. 100BASE-TX and 100BASE-T1
  - And ideally should also be compatible with 10BASE-T1S
- The requirements considered include the following
  - The pin interface and electrical requirements
  - The embedded interfaces
  - The features that should be supported

Single PHY POPI - Pin Interface

- For a single 100M PHY the desire is to have a simple low pin count interface using standard IOs and digital logic
  - To support 100M data plus overhead for control, PTP time stamping, MDIO and clause 36 ordered sets we expect to need a raw data rate on the pins of about 125 Mb/s, somewhere in the range of 110 – 140 Mb/s
  - At these speeds it is preferable to avoid the complexity of a SerDes while still keeping to a very low pin count
- A simple 4-pin interface with single ended IOs is sufficient
  - TXD, TX_CLK, RXD, RX_CLK
  - Same pin count as a SerDes (4 pins)
  - Signalling rates of ~125 MHz are similar to existing GE PHY MAC interfaces
  - No clock/data recovery as clocks provided
  - 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)

Single PHY POPI – Low Latency Data Requirement

- At 100M the latency of each bit is much more significant than at Multi-Gig rates
  - Highly desirable to work with single data nibbles and data bytes
  - 100BASE-T1L uses an 8b6T line code and 16B/17B block code to minimise the latency in transferring 2 x data bytes from the MII or RMII to the line (with a small buffer of a few bits due to the 16B/17B block code)
  - 100BASE-TX uses MLT-3 and 4B5B to encode a nibble and minimises the latency in transferring each data nibble from the MII or RMII to the line
  - 10BASE-T1L uses 4b3T to encode a nibble and minimises the latency in transferring each data nibble from the MII or RMII to the line
  - 100BASE-T1 uses 3b2T and 4B3B mapping to encode a nibble and minimises the latency in transferring each data nibble from the MII or RMII to the line (with a small buffer of a few bits for the 4B3B mapping)
  - 1000BASE-T uses 8b4Q to encode a data byte and minimises the latency in transferring each data byte from the GMII or SGMII to the line
- The block encoding on the pin interface, e.g. 8B10B or other block encoding used by POPI should ensure a data byte is available each cycle for encoding and transfer to the line
  - E.g. in MII or RMII every 25 MHz cycle a new data nibble is available to the PCS
  - E.g. in GMII or SGMII every 125 MHz cycle a new data byte is available to the PCS
- In POPI, ideally a new data byte should be available to the PHY PCS every 12.5 MHz cycle
  - With additional sideband bits at a lower rate to support PTP, MDIO, etc.

Single PHY POPI –MDIO Management Interface

- MDIO is a medium bandwidth interfaces that should be embedded with POPI so additional pins between the MAC and PHY are not required
  - The MAC interface typically uses a 2-pin shared bus for the MDIO management interface
    - This is a very low overhead for a multi-port switch or for a legacy RMII/RGMII MAC interface
    - However, for a single PHY using POPI this would represent 50% more pins
  - There is a strong preference to embed MDIO with POPI to achieve the lowest pin count interface
- Clause 22 specifies an interface speed of 2.5 MHz for MDIO, many PHYs support speeds up to 6.25 MHz
- In some case PHYs are required to support MDIO speeds of 12.5 MHz
  - However, this is typically driven by multi-port or PTP requirements
  - With POPI there would be a separate embedded PTP interface and each port would have its own embedded MDIO interface
- Hence, an embedded MDIO interface supporting 2.5 to 6.25 Mb/s would be sufficient for a single 100M PHY

Single PHY POPI – PTP Timestamping

- PTP timestamping is also a medium bandwidth interfaces that should be embedded with POPI so additional pins are not required
  - PTP timestamping is sometimes supported using two additional pins (Tx and Rx Start of packet pins) or is supported with register operations over the MDIO
  - There is a strong preference to embed PTP with POPI to achieve the lowest pin count interface
- PTP timestamping could require 4 bytes for a 64 byte packet, which suggests an upper bound of ~ 6.25 Mb/s
  - Only some PTP packets require timestamp information to be sent from the MAC to the PHY, not every packet – so this is a very conservative number
  - Really its is an approximate peak bandwidth requirement
  - And on the receive side a timestamp for some packets must be sent from the PHY to the MAC
- Hence, a shared embedded MDIO/PTP interface supporting 6.25 Mb/s would be sufficient for a single 100M PHY

Single PHY POPI – Clause 36 Ordered Sets

- Clause 36 ordered sets are used for encapsulation and are also used as a low bandwidth interface
- In SGMII and USGMII clause 36 ordered sets are used for control and configuration
  - E.g. Auto-Negotiation information, error propagation, faults, LPI, etc.
- Clause 36 ordered sets
  - Configuration ordered sets convey information about the link
    - Link-up, link failure, Auto-Negotiation error, FD/HD, Pause
  - Encapsulation ordered sets are used for the start and end of packet, carrier extend and error propagation
  - LPI ordered sets are used for Energy Efficient Ethernet
- Clause 36 ordered sets may have to be amended/supplemented to support PLCA
  - And we may also want to also include clause 46 sequence ordered set for fault signalling

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.

Config_Reg Base Page bits	Management register bit
Full Duplex (FD)	4.5 Full Duplex
Half Duplex (HD)	4.6 Half Duplex
PAUSE (PS1)	4.7 PAUSE
ASM_DIR (PS2)	4.8 ASM_DIR
Remote Fault (RF2, RF1)	4.13:12 Remote Fault

Table 37–3—Remote Fault encoding

RF1	RF2	Description
0	0	No error, link OK (default)
0	1	Offline
1	0	Link_Failure
1	1	Auto-Negotiation_Error

Single PHY POPI – Embedded Interfaces

- There is a strong preference to embed MDIO, PTP, link information and clause 36 like ordered sets with POPI to achieve the lowest pin count interface
- Ideally, POPI would support up to 10 Mb/s for embedded interfaces to support all 3 of these interfaces; MDIO, PTP and link information
  - This would support the speed requirements of MDIO and PTP with some margin
  - Would allow reservation of bandwidth for control information
- These embedded interfaces can all operates asynchronous to the 100M data interface and they do not have any critical latency requirements
  - They operate today at the speed of packets not bytes or bits
  - Additional sideband bits need to be transferred over the interface to support the embedded interfaces, however, these bits can be collected over a much larger number of data bytes to accumulate a frame, which can encode this set of interfaces
  - Note, these interfaces as implemented today have no error protection or CRC, which really should be addressed in POPI
  - Ideally the logical format of such a frame / interfaces would follow the same scheme at all POPI rates

Single PHY POPI – Features Supported

- In the Study group and the consensus call for interest meeting a number of different features that POPI might support were discussed
  - We have already covered the most important ones of these; MDIO, PTP, LPI
- Half-duplex operation (COL/CRS)
  - Desire here is to support half-duplex PHYs like 10BASE-T1S, 100BASE-TX & 10BASE-T in HD mode
  - All the legacy non-standard MAC interface infer COL/CRS at the MAC side from RX_DV and TX_EN
  - Why do anything different for POPI ?
  - We suggest that we should **not** try to support half-duplex operation within POPI as it adds complexity for no benefit
- PLCA support
  - PLCA defines some extra MII codes, e.g. Commit and Beacon
  - These could be added using clause 36 like ordered sets in POPI
- Two-pair 10/100 PHY compatibility
  - If POPI only supports full-duplex there is nothing else to be done to support two-pair 10/100 PHYs

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

Single PHY POPI – Local / Remote Fault

- 802.3dg added Assert Local Fault / Assert Remote Fault to clause 22 to support fault signalling between the PHY and RS
  - And added encoding of Assert Remote Fault in the PCS so that it can be sent from the local RS to the remote RS over the link
  - This was done t add the fault signalling developed in clause 46 for 10G (and Multi-Gig) PHYs to 100BASE-T1L
- Assert Local Fault and Assert Remote Fault should be added using clause 36 like ordered sets in POPI

22.2.2.4 TXD (transmit data)

Change Table 22–1 as shown (unchanged rows not shown):

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

TX_EN	TX_ER	TXD<3:0>	Indication
…			
0	1	0100	Assert remote fault
0	1	0100~~0~~1 through 1111	Reserved
…			

22.2.2.8 RXD (receive data)

Change Table 22–2 as shown (unchanged rows not shown):

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

RX_DV	RX_ER	RXD<3:0>	Indication
…			
0	1	0100	Assert remote fault
0	1	0101	Assert local fault
0	1	0110~~0~~ through 1101	Reserved
…			

Single PHY POPI – 10BASE-T1S

- It would seem desirable in order to be complete that we should also support 10BASE-T1S, especially as this is a recent PHY and new products could adopt POPI
  - However, it does seem unlikely that a 10BASE-T1S single PHY would be implemented as a stand alone PHY with a MAC interface
  - There is already a 3-pin interface defined to allow the analog (maybe high voltage) portion of the PHY to be implemented stand alone
  - It is more likely that the low voltage digital portion of 10BASE-T1S would be integrated with a MAC
- This suggests, that the 10BASE-T1S requirements should not be a high priority
  - The end result may be usable as a 10BASE-T1S MAC interface anyway

Summary

- For a single 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 MHz
- Should provide a new data byte from the MAC interface to the PHY PCS every 12.5 MHz cycle
- Support MDIO, PTP and clause 36 like ordered sets as embedded interfaces with POPI to achieve the lowest pin count interface
- 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 ?