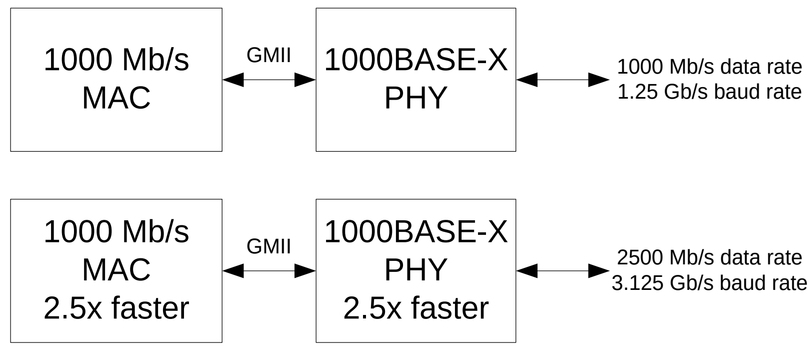
2.5GBASE backplane PCS and Auto-Negotiation Proposal

January 7, 2016

William Lo, Marvell

Define 2.5G PHY for wide adoption

- **Implementations in the field from various suppliers running 1000BASE-X at 2.5 times speed.**



- **Can leverage existing equipment if we define 2.5G backplane such that existing equipment can operate in compatible fashion.**
  - **Existing implementations would be <u>compatible</u>, but do not need to be <u>compliant</u>**

- **Plenty of margin with 1000BASE-X running 2.5 times faster**
  - **No RX DFE and no TX Equalization - wu_CU4HDDSG_01_1115.pdf**
  - **using channels from Calbone_CU4HDDsg_02_0915.pdf**

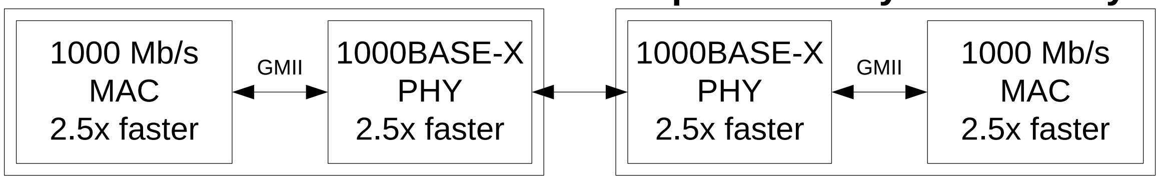
What is needed for compatibility

- Adopt subset of 1000BASE-X PCS and run 2.5 times faster
  - No need to support Clause 37 Auto-Negotiation
  - Clause 73.1 – recommends disabling Clause 37 AN if Clause 73 AN is used
  - Let's enforce this by making it mandatory to disable (or not implement) portion of the PCS that supports Clause 37 AN.
- Make Clause 73 Parallel Detect support mandatory for 2.5G Backplane
  - i.e. One PHY AN on and other PHY AN off. The PHY with AN on detects 2.5G signal from link partner and stops the AN process and proceeds to link in 2.5G
  - Parallel detection currently supported in 1000BASE-KX, 10GBASE-KX4.
- Energy Efficient Ethernet can be enabled only if both PHYs advertise the capability using Clause 73 Auto-Negotiation
  - Implication – Parallel Detect link up will not support EEE
  - EEE can be manually turned on without AN but this is outside the scope of the standard.
- Need to reconcile possibility of GMII based 1G MAC running 2.5G interacting with XGMII based 10G MAC running 2.5G
  - Existing 2.5G implementations most likely using scaled up 1G MAC
  - 802.3bz chose scaled down 10G MAC running at 2.5G. We are stuck with XGMII.

Cases To Consider

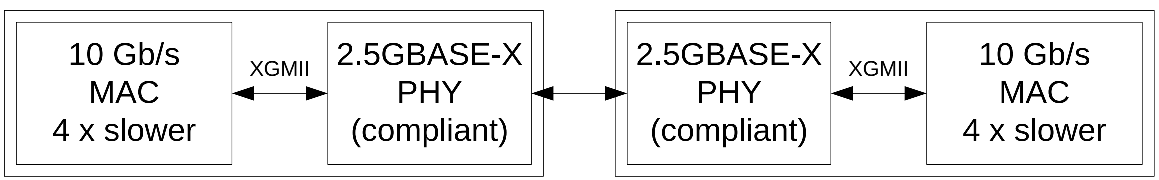
Legacy PHY to Legacy PHY

- **No Problem – 1000BASE-X at 2.5x speed already works today**



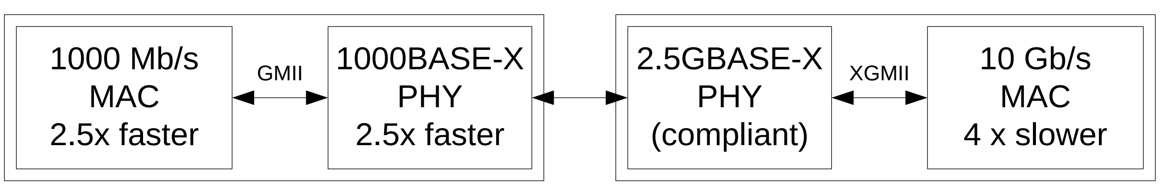
Compliant PHY to Compliant PHY

- **No Problem – We can define anything we want**
- **Need to introduce concept of passing Sequence and Signal ordered set as 10G MAC capable of sending**



Legacy PHY to Compliant PHY

- **Compliant PHY needs to compensate for non-4 byte alignment issue**
- **Compliant PHY passing transmitting Sequence and Signal ordered set does not interfere with legacy PHY**



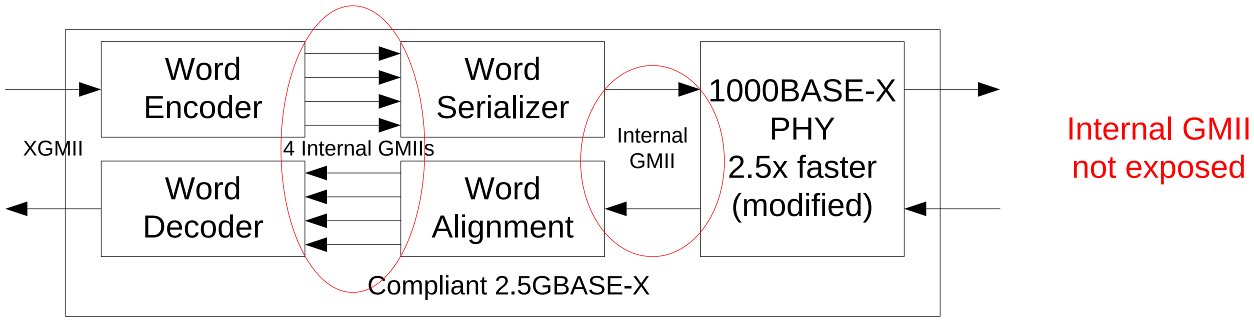
Fitting 1000BASE-X PCS using 2.5G XGMII from 802.3bz

- Implementing *MII is optional. But from standards point of view it is necessary as a fixed point of reference to the Reconciliation Sublayer
- 1000BASE-X PCS uses the GMII as reference
- 802.3bz defined the 2.5G reference using the XGMII

- To allow compatibility need to address following 2 issues
- 1 byte vs 4 byte alignment issue of start of packet
  - XGMII is 4 byte interface while GMII is 1 byte interface
- Passing Sequence and Signal ordered set
  - XGMII based MAC can pass |Q| ordered sets
  - No concept of |Q| or |Fsig| ordered sets in GMII based MAC

Proposal on PCS Specification

- **Start with base 1000BASE-X PCS and make modifications**
  - **Small state machine modification to pass |Q| and |Fsig| ordered sets**
  - **Disable/remove portions of the state machine that supports Clause 37 Auto-Negotiations**
  - **If we decide to block |Q| and |Fsig| and send as idles then no need for changes. Simply set variable to disable Clause 37.**
- **Word serializer/alignment simply handles the 4 byte to 1 byte conversion between XGMII and GMII**
- **Word encoder/decoder – mapping between XGMII to Internal GMII**
- **Implementation as shown does not require much incremental logic**



- **Does not preclude implementations that directly map XGMII into PCS**
  - **Diagram above for IEEE specification purposes only**

Word Serializer/Alignment Specification

- **Serializer**
  - Simply take four GMII bytes and send it out one GMII byte at a time
- **Alignment**
  - XGMII operates 4 bytes at a time and requires Start of Packet be on byte 0
  - GMII operates 1 byte at a time. Simply grouping 4 bytes will not guarantee Start of Packet will be on byte 0.
  - Use deficit idle counting (DIC) in Clause 46.3.1.4 to align Start of Packet

Deficit	SOP on byte 0	SOP on byte 1	SOP on byte 2	SOP on byte 3
0 byte	Do nothing	Delete 1 idle byte	Delete 2 idle bytes	Delete 3 idle bytes
1 byte	Do nothing	Delete 1 idle byte	Delete 2 idle bytes	Insert 1 idle byte
2 bytes	Do nothing	Delete 1 idle byte	Insert 2 idle bytes	Insert 1 idle byte
3 bytes	Do nothing	Insert 3 idle bytes	Insert 2 idle bytes	Insert 1 idle byte

- **Extend concept of Deficit Idle Counting to align other things to byte 0**
  - Start of Packet
  - First Low power idle when transitioning in from idles
  - Start of ordered sets
    - This is more an error condition case as ordered set can only be generated from XGMII interface which should already be aligned
    - See slide on ordered set on how to align

Word Encoder Specification

- **Internal GMII – Define Sequence code**
  - **We are NOT changing definition of GMII of Clause 35, we are only defining a new code for Internal GMII**

TX_EN	TX_ER	TXD[7:0]	Description
0	0	xx	Idle
0	1	0x01	Low Power Idle
0	1	0x0F	Carrier Extend - Not used
0	1	0x1F	Carrier Extend Error - Not used
1	0	00 to FF	Data
1	1	xx	Transmit Error
0	1	0x9C	Sequence

- **Simple XGMII to four GMII mapping except for Sequence ordered set**
- **Sequence ordered set expand from 4 bytes to 8 bytes**
  - **Throw away every other order set on XGMII – ok to do this (10GBASE-X4 throws away more than 90% of sequence order set seen on XGMII)**
  - **Use Prev Seq variable to track whether to throw away next Sequence ordered set on XGMII**
  - **Truncate ordered set transmission if one throw away is not a Sequence ordered set**

XGMII					GMII				
Byte 0	Byte 1	Byte 2	Byte 3	Prev Seq	GMII 0	GMII 1	GMII 2	GMII 3	Next Seq
Data A/Err	Data B/Err	Data C/Err	Data D/Err	X	Data A/Err	Data B/Err	Data C/Err	Data D/Err	No
Idle	Idle	Idle	Idle	X	Idle	Idle	Idle	Idle	No
LPI	LPI	LPI	LPI	X	LPI	LPI	LPI	LPI	No
SOP	Data A/Err	Data B/Err	Data C/Err	X	0x55 Data	Data A/Err	Data B/Err	Data C/Err	No
Terminate	Idle	Idle	Idle	X	Idle	Idle	Idle	Idle	No
Data A/Err	Terminate	Idle	Idle	X	Data A/Err	Idle	Idle	Idle	No
Data A/Err	Data B/Err	Terminate	Idle	X	Data A/Err	Data B/Err	Idle	Idle	No
Data A/Err	Data B/Err	Data C/Err	Terminate	X	Data A/Err	Data B/Err	Data C/Err	Idle	No
Sequence	Data X	Data Y	Data Z	No	Sequence	Data S0	Sequence	Data S1	Yes
Sequence	Data X	Data Y	Data Z	Yes	Sequence	Prev Data S2	Sequence	Prev Data S3	No
else					Error	Error	Error	Error	No

Word Decoder Specification

- State dependent mapping based on Prev Word and Next Word variables
- False Carrier, Carrier Extend Error, and out of place Carrier Extend converted to errors
- Look ahead needed to check S0, S1, S2, S3 bit 7 consistency (i.e. 0111)
- If link is down then XGMII outputs Local Fault ordered set

Internal GMII						XGMII				
GMII 0	GMII 1	GMII 2	GMII 3	Prev Word	Next Word	Byte 0	Byte 1	Byte 2	Byte 3	Prev Word
Data A/Err	Data B/Err	Data C/Err	Data D/Err	Not Idle	X	Data A/Err	Data B/Err	Data C/Err	Data D/Err	Data
Data *	Data A/Err	Data B/Err	Data C/Err	Idle	X	SOP	Data A/Err	Data B/Err	Data C/Err	Data
Idle	Idle	Idle	Idle	Not Data	X	Idle	Idle	Idle	Idle	Idle
Idle	Idle	Idle	Idle	Data	X	Terminate	Idle	Idle	Idle	Idle
Data A/Err	Idle or Carrier Extend	Idle	Idle	Data	X	Data A/Err	Terminate	Idle	Idle	Idle
Data A/Err	Data B/Err	Idle	Idle	Data	X	Data A/Err	Data B/Err	Terminate	Idle	Idle
Data A/Err	Data B/Err	Data C/Err	Idle or Carrier Extend	Data	X	Data A/Err	Data B/Err	Data C/Err	Terminate	Idle
LPI	LPI	LPI	LPI	X	X	LPI	LPI	LPI	LPI	Idle
Idle	Idle	LPI	LPI	X	X	LPI	LPI	LPI	LPI	Idle
LPI	LPI	Idle	Idle	X	X	Idle	Idle	Idle	Idle	Idle
Sequence	Data S0	Sequence	Data S1	X	Sequence S2, S3	Sequence	Data X	Data Y	Data Z	Sequence
Sequence	Data S0	Sequence	Data S1	X	Not Sequence S2, S3	Idle	Idle	Idle	Idle	Idle
Sequence	Data S2	Sequence	Data S3	Sequence	X	Sequence	Data X	Data Y	Data Z	Idle
else						Error	Error	Error	Error	Error

What Does Sequence Ordered Set Like

- XGMII – Sequence, Data X, Data Y, Data Z (4 bytes)
- Internal GMII – Sequence, S0, Sequence, S1, Sequence, S2, Sequence, S3
- Output of PCS – K28.5, Dx.y (S0), K28.5, Dx.y (S1), K28.5, Dx.y (S2), K28.5, Dx.y (S3)
- S0[7] = 0, S1[7] = S2[7] = S3[7] = 1 for |Q| Ordered Set
- S0[7] = 1, S1[7] = S2[7] = S3[7] = 0 for |Fsig| Ordered Set
  - Note: S0[7] and S1[7] opposite values can use for ordered set byte alignment
- S0[5:0] = Data X[5:0]
- S1[5:0] = Data Y[3:0], Data X[7:6]
- S2[5:0] = Data Z[1:0], Data Y[7:4]
- S3[5:0] = Data Z[7:2]
- Sn[6] = Sn[7] if Sn[2] = 0
- Sn[6] = Sn[5] if Sn[2] = 1

- Six K28.5 Dx.y to avoid
  - Forcing bit 6 to be the same as bit 7 or bit 5 guarantees this

Function	Data Code	Octet	7	6	5	4	3	2	1	0
Idle	D5.6	C5	1	1	0	0	0	1	0	1
Idle	D16.2	50	0	1	0	1	0	0	0	0
LPI	D6.5	A6	1	0	1	0	0	1	1	0
LPI	D26.4	9A	1	0	0	1	1	0	1	0
Config	D21.5	B5	1	0	1	1	0	1	0	1
Config	D2.2	42	0	1	0	0	0	0	1	0

1000BASE-X PCS Modifications

- Never set xmit = CONFIGURATION
  - Disables configuration ordered set from ever being sent
- Add state machine transitions to handle Sequence ordered set
  - Will have draft ready in Jan meeting
  - Will show how 1000BASE-X PCS (unmodified) will ignore the Sequence ordered set

THANK YOU