Marvell. Moving Forward Faster

MultiGig Auto-PHY Block Code Considerations

August 23, 2017

Brett McClellan, Marvell Semiconductor

MARVELL®

Agenda

- Media Independent Interface
- 1000BASE-T1 Block Coding
- MultiGBASE-T Block Coding
- Transcoding
- Conclusion
- Further Steps

Gig Media Independent Interface

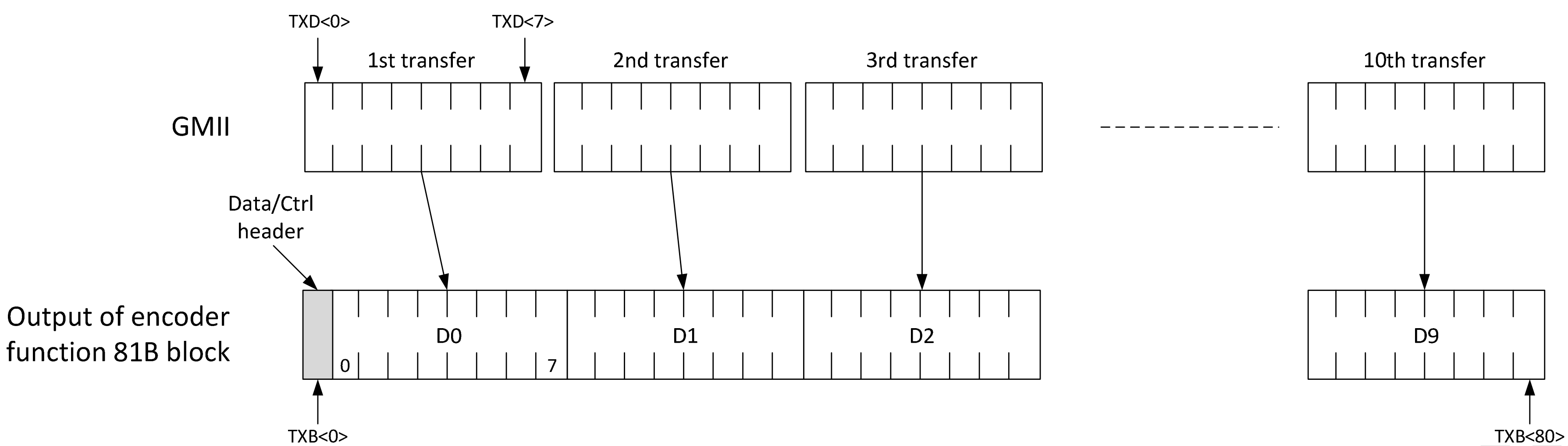
- Gig PHYs defined for GMII – Clause 35
  - 1000BASE-X, 1000BASE-T, 1000BASE-T1
  - Byte-wide at 125MHz
  - TXD<7:0>, TX_EN, TX_ER
  - RXD<7:0>, RX_ER, RX_DV, (COL, CRS)
  - Host interfaces: SGMII, RGMII

MultiGBASE-T Media Independent Interface

- Multi-Gig PHYs defined for XGMII – Clause 46
  - 2.5/5/10GBASE-T (Cl. 126, 55), 2.5GBASE-X & 5GBASE-R (802.3cb)
  - 4-Bytes at 78.125/156.25/312.5 MHz
  - TXD<31:0>, TXC<3:0>
  - RXD<31:0>, RXC<3:0>
  - Host interfaces:
    - 10G: XFI (10GBASE-R), XAUI (10GBASE-X), RXAUI
    - 5G: 5GBASE-R, USXGMII, rate adaptive from 10G
    - 2.5G: 2.5GBASE-X, USXGMII, rate adaptive from 5G/10G

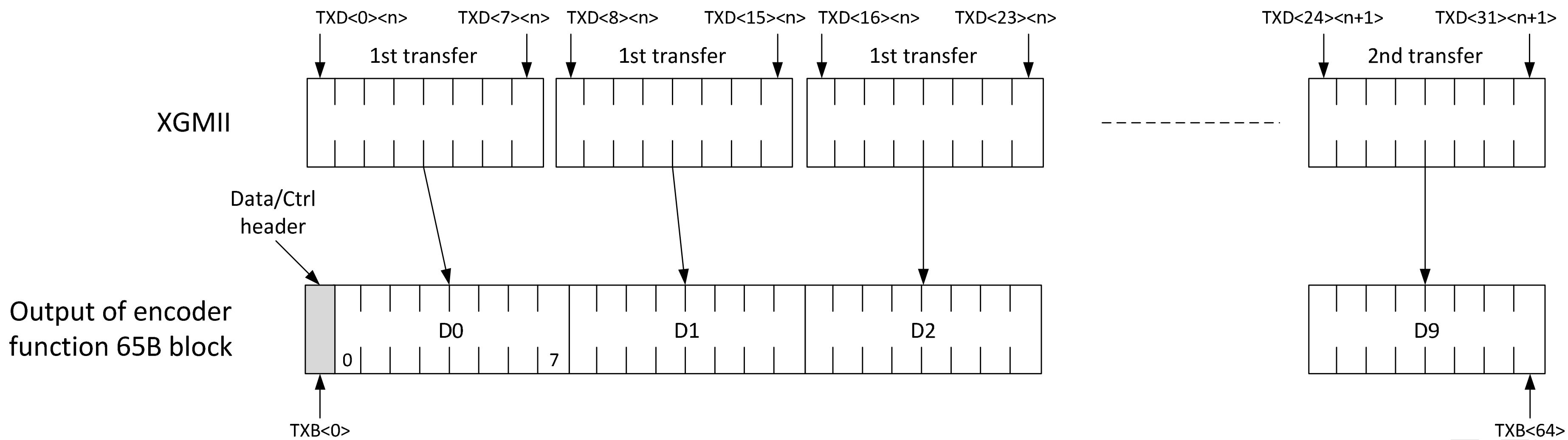
1000BASE-T1 Block Coding

- 80B/81B block code
  - http://www.ieee802.org/3/bp/public/mar14/Lo_3bp_02_0314.pdf
  - Block code designed for GMII
  - Not compatible with XGMII: fault codes & 4-byte alignment not supported
  - Header bit
    - = 0 indicates All Data
    - = 1 indicates pointers and control codes used
    - 1.25% overhead



MultiGBASE-T Block Coding

- 64B/65B block code
  - Used in 2.5/5/10GBASE-T
  - Block code designed for XGMII – aligned to 4 bytes
  - Not compatible with GMII: SOF alignment to lane 0 only
  - Header bit
    - = 0 indicates All Data
    - = 1 indicates block type and control codes used
    - 1.5625% overhead
  - Short latency, 8 bytes



64B/65B Block Code

Input Data	data ctrl header	Block Payload								
Bit Position:	0	1								64
Data Block Format:										
$D_0 D_1 D_2 D_3/D_4 D_5 D_6 D_7$	0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
Control Block Formats:		Block								
$C_0 C_1 C_2 C_3/C_4 C_5 C_6 C_7$	1	0x1E	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
$C_0 C_1 C_2 C_3/O_4 D_5 D_6 D_7$	1	0x2D	C_0	C_1	C_2	C_3	O_4	D_5	D_6	D_7
$C_0 C_1 C_2 C_3/S_4 D_5 D_6 D_7$	1	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
$O_0 D_1 D_2 D_3/S_4 D_5 D_6 D_7$	1	0x66	D_1	D_2	D_3	O_0		D_5	D_6	D_7
$O_0 D_1 D_2 D_3/O_4 D_5 D_6 D_7$	1	0x55	D_1	D_2	D_3	O_0	O_4	D_5	D_6	D_7
$S_0 D_1 D_2 D_3/D_4 D_5 D_6 D_7$	1	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
$O_0 D_1 D_2 D_3/C_4 C_5 C_6 C_7$	1	0x4B	D_1	D_2	D_3	O_0	C_4	C_5	C_6	C_7
$T_0 C_1 C_2 C_3/C_4 C_5 C_6 C_7$	1	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0 T_1 C_2 C_3/C_4 C_5 C_6 C_7$	1	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0 D_1 T_2 C_3/C_4 C_5 C_6 C_7$	1	0xAA	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0 D_1 D_2 T_3/C_4 C_5 C_6 C_7$	1	0xB4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0 D_1 D_2 D_3/T_4 C_5 C_6 C_7$	1	0xCC	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0 D_1 D_2 D_3/D_4 T_5 C_6 C_7$	1	0xD2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0 D_1 D_2 D_3/D_4 D_5 T_6 C_7$	1	0xE1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0 D_1 D_2 D_3/D_4 D_5 D_6 T_7$	1	0xFF	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

Transcoding

- Introduced for 25G/40GBASE-T - 512B/513B
  - Based on 64B/65B code
- Trade increased coding efficiency for longer latency
  - Frees more bits for forward error correction
- 512B/513B Transcode
  - Aggregate 8 x 65B blocks
  - If all blocks are data only, set header bit = 1 and send data bytes
  - If any block is a control type
    - Set header bit = 0
    - Send control blocks: block type, block position, continuation flag
    - Send remaining data bytes
  - 0.2% overhead
  - Long latency 64 bytes

Conclusion

- 80B/81B - not suited for MultiGBASE-T1
- 64B/65B – good basis for block coding
- Transcoding may be used for higher efficiency
  - 512B/513B
  - Smaller block sizes may be used for shorter latency

Further Steps

- Consider FEC options
  - Alignment between block code and FEC frame sizes
- Modulation impact on symbol size, FEC and block code