Low Latency and Long Reach PHY Proposals

William Lo

25 January 2024

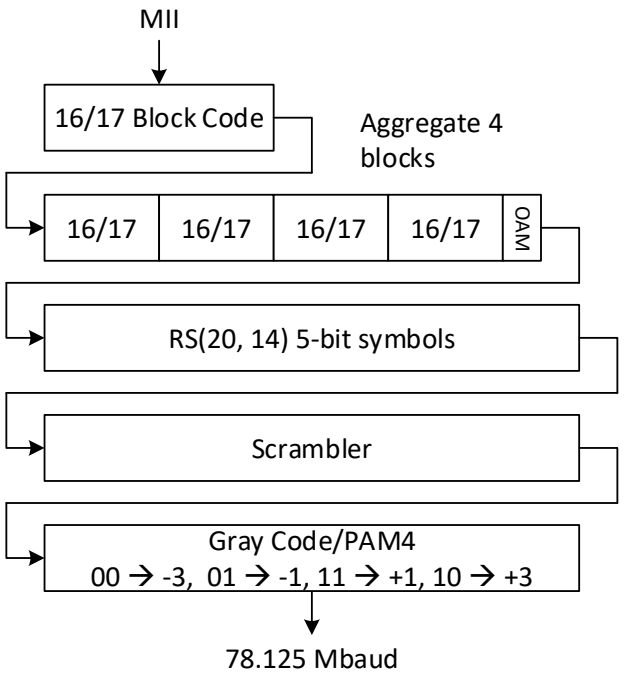
Overall Objectives

- Low Latency
  - 100 meters, < 1.5us latency
  - Can be less bandwidth efficient to decrease latency
  - FEC is a bonus
- Long Reach
  - 500 meters, latency less important
  - Bandwidth efficient to decrease baud rate, hence longer reach
- Bounded Disparity
  - Intrinsic safety
- Reduce cost to implement
  - Assume one solution does NOT fit all
  - Expensive portions keep common
  - Differentiate on less costly components between Low Latency and Long Reach

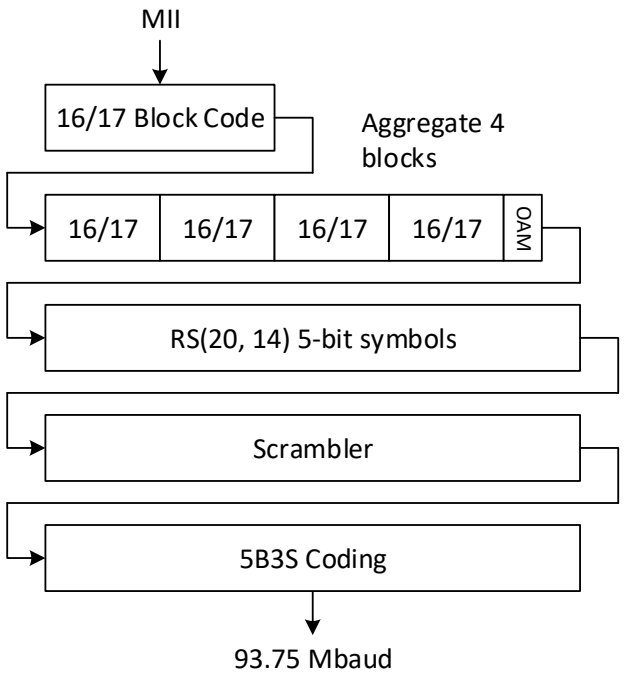
Summary of Proposed Baseline

- Transmit Path shown, Reverse arrows for Receive Path
  - No bounded disparity shown for comparison only.

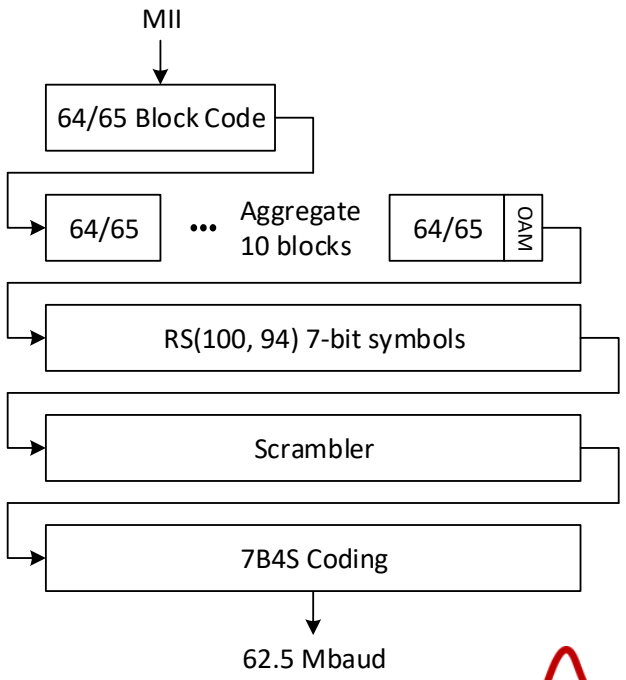
Low Latency – No Bounded Disparity



Low Latency – Bounded Disparity



Long Reach – Bounded Disparity



Encoder

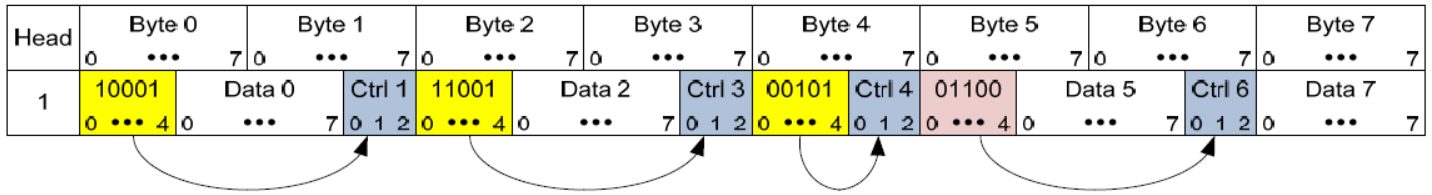
- Use 8N/(8N+1) encoding discussed in
  - https://grouper.ieee.org/groups/802/3/bp/public/mar14/Lo_3bp_02_0314.pdf
  - No lane alignment required (vs traditional 64/66 or 64/65) as this would add latency
  - N = 2 for Low Latency (Shorter delays)
  - N = 8 for Long Reach (More efficient)
- N parameterizable in

  Clause 97.3.2.2.5
- Arbitrary placement of control codes

```
N = number of GMII octets encoded into block. Octets numbered n = 0,1,2,…,N-1.
octet 0 is the first one presented on GMII.
TC[n] = 0 if octet n is data octet on GMII, 1 if octet n is control octet on GMII
TC[-1] = 1 by definition
TD[n][0:7] = GMII octet n TXD[0:7] if TC[n] = 0
TD[n][5:7] = 010 - IPG (loc_phy_ready = OK), 101 - LPI, 001 - TX Error, 000 - IPG
(loc_phy_ready = NOT_OK) if TC[n] = 1. TD[n][0:4] is undefined.
B[0:8N] is the 8N+1 block. Bit 0 transmitted first.
OR(n) = Bitwise OR of TC[n:N-1]
NEXT(n)[0:3] = bit position of lowest bit in TC[n:N-1] that is a 1. Bit 3 is MSB.
NEXT(n)[4] = 0 if Bitwise SUM of TC[n:N-1] = 1, else 1

B[0] =                 OR(0)
B[8n+1:8n+4] =         TD[n][0:3] - if OR(n) = 0
                       NEXT(n)[0:3] - if OR(n) = 1 AND TC[n-1] = 1
                       TD[n-1][3:6] - if OR(n) = 1 AND TC[n-1] = 0
B[8n+5] =              TD[n][4] - if OR(n) = 0
                       NEXT(n)[4] - if OR(n) = 1 AND TC[n-1] = 1
                       TD[n-1][7] - if OR(n) = 1 AND TC[n-1] = 0
B[8n+6:8n+8] =         TD[n][5:7] - if OR(n) = 0
                       TD[n][5:7] - if OR(n) = 1 AND TC[n] = 1
                       TD[n][0:2] - if OR(n) = 1 AND TC[n] = 0
```

- **Example: D/C/D/C/C/D/C/D**



Reed Solomon

- Low Latency
  - RS(20, 14) GF(2^5)
  - Four 16/17 blocks,
  - RS block duration is 640ns
  - Up to 96ns burst protection
  - 2 OAM bits
  - Overhead 100/64 = 56.25%
- Long Reach
  - RS(100, 94) GF(2^7)
  - Ten 64/65 blocks,
  - RS block duration is 6400ns
  - Up to 192ns burst protection
  - 8 OAM bits
  - Overhead 700/640 = 9.375%

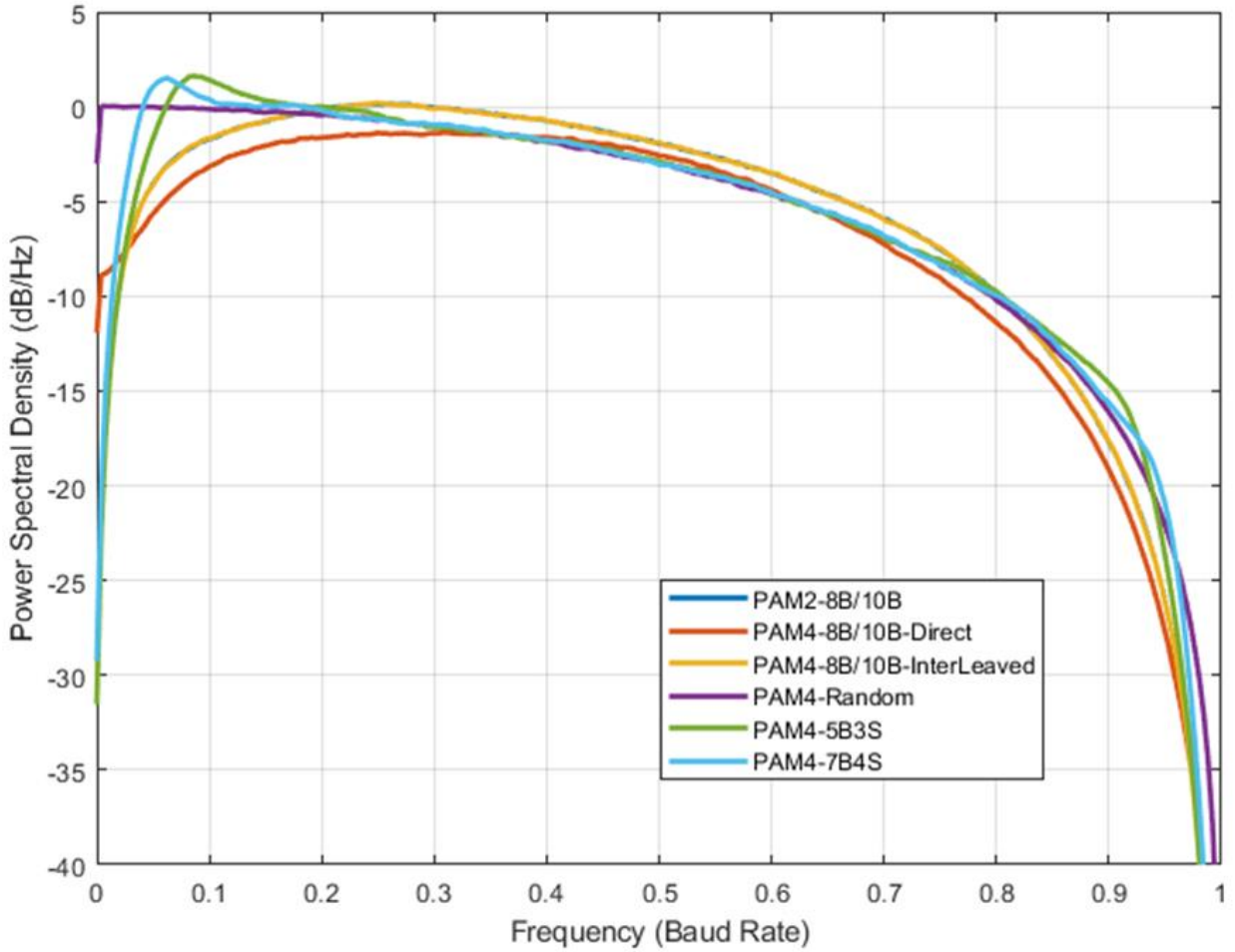
- Both have similar circuit structure with 6 parity symbols

Scrambler

- Use MGBASE-T1 scrambler Clause 149.3.4
- Easy transitioning from training in PAM2 into PAM4
  - Don’t reinvent the wheel
- Master: $gM(x) = 1 + x^{13} + x^{33}$
- Slave: $gS(x) = 1 + x^{20} + x^{33}$

Bounded Disparity Coding

- Low Latency
  - 5-bit, 3-PAM4 symbols (5B3S)
  - 20% overhead
  - (8b/10b overhead is 25%)
- Long Reach
  - 7-bit, 4-PAM4 symbols (7B4S)
  - 14.3% overhead
- Matches 5-bit and 7-bit RS Symbols
  - Any symbol corrupted affects only 1 symbol
- DC Balanced
  - More low frequency component (less attenuation) vs PAM2 8b/10b
  - 8b/10b to PAM4 assumes grouping every 2 bits to form PAM4 symbol
  - 8b/10b interleaved to PAM4 matches PAM2 8b/10b



5B/3S Table

	NS = P			NS = N		
r[4:0]	P[2]	P[1]	P[0]	N[2]	N[1]	N[0]
0	1	1	1	-1	-1	-1
1	1	1	-1	-1	-1	1
2	1	-1	1	-1	1	-1
3	-1	1	1	1	-1	-1
4	1	1	3	-1	-1	-3
5	-1	-1	3	1	1	-3
6	1	-1	3	-1	1	-3
7	-1	1	3	1	-1	-3
8	1	3	1	-1	-3	-1
9	1	3	-1	-1	-3	1
10	-1	3	-1	1	-3	1
11	-1	3	1	1	-3	-1
12	1	3	3	-1	-3	-3
13	1	3	-3	-1	-3	3
14	1	-3	3	-1	3	-3
15	-1	3	3	1	-3	-3
16	3	1	1	-3	-1	-1
17	3	1	-1	-3	-1	1
18	3	-1	1	-3	1	-1
19	3	-1	-1	-3	1	1
20	3	1	3	-3	-1	-3
21	3	1	-3	-3	-1	3
22	3	-1	3	-3	1	-3
23	-3	1	3	3	-1	-3
24	3	3	1	-3	-3	-1
25	3	3	-1	-3	-3	1
26	3	-3	1	-3	3	-1
27	-3	3	1	3	-3	-1
28	3	3	3	-3	-3	-3
29	3	3	-3	-3	-3	3
30	3	-3	3	-3	3	-3
31	-3	3	3	3	-3	-3

Table scales the PAM4 values by 3x and shows +/-3 and +/-1 instead of +/-1 and +/- 1/3 to avoid printing fractions

Easy to Implement 5B/3S and 7B/5S

- No Table Lookup
  - Code construction example below shows 5B3S. Similar concept for 7B/4S.

- Let r[4:0] be the 5-bit binary code. There are 32 possible codes.
  Let this set of codes be called the set **R**.

- The PAM4 side each output is a tuple of three PAM4 symbols.
  There are 64 possible codes (4x4x4 = 64).

- We'll use +3, +1, -1, -3 instead of +1, + 1/3, -1/3, -1 to avoid writing fractions in the notation below

- Let L[2:0] be a tuple {a, b, c} where a, b, c is either 1 or 3 where
  - if r[i+2] = 0 then L[i] = 1 and if r[i+2] = 1 then L[i] = 3
  - Essentially do a simple mapping where a 0 will map to either PAM4 +/- 1 and a 1 maps to either PAM4 +/- 3
  - i.e. r[4:2] = 011 maps to {1, 3, 3}

5B/3S construction continued

- Let Q[2:0] be a tuple {d, e, f} where d, e, f is either +1 or -1 where
  - if r[1:0] = 00 then Q = {+1, +1, +1}
  - if r[1:0] = 01 then Q = {+1, +1, -1}
  - if r[1:0] = 10 then Q = {+1, -1, +1}
  - if r[1:0] = 11 then Q = {-1, +1, +1}
  - Exception, if r = 00101 or 00110 then invert all three tuples
    - ie r = 00101 then Q = {-1, -1, +1}. r = 00110 then Q = {-1, +1, -1}

- Let P[2:0] = {L[2] x Q[2], L[1] x Q[1], L[0] x Q[0]}
  - Essentially set the sign of each element in L with the sign in Q.
  - i.e. r = 10001 then L = {3, 1, 1}, Q = {+1, +1, -1} hence P = {+3, +1, -1}

5B/3S construction continued

- There are 32 possible codes of P[2:0]. Call this the set **P** or positive set.
  Each code in P has a 1 to 1 mapping to a code in **R**.
  The sum of the three PAM4 symbols are all non-negative.
  Define this sum as the disparity of this code.
  i.e. {+3, -1, +3} has a disparity of +5.

- Let set **N** or negative set contain 32 possible codes N[2:0] with each code having a 1 to 1 mapping to a code in **P** where each symbol in the tuple N[2:0] is the negative of its corresponding P[2:0].
  i.e. N[2:0] of {-1, +3, -3} corresponds to P[2:0] of {+1, -3, +3}
  Implication of 1:1 mapping is that codes **N** also has 1:1 mapping to codes in **R**
  The disparity of the codes in set **N** are all non-positive

Running Disparity

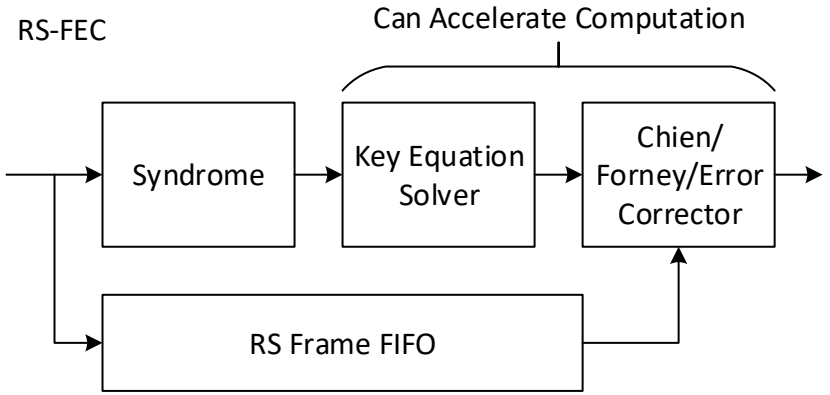
- Given stream of r from **R**, output the corresponding code from either **P** or **N**
- Define the following variables:
  - RD = Running disparity
  - NS = Next Set to use (N or P)
- 0) Initialize RD = 0 and NS = N (Doesn't really matter if it is N or P)
- 1) Output the 3 PAM4 symbols of the code corresponding to r in the set indicated by NS
  - Output symbol 2, followed by 1, followed by 0. (Order doesn't really matter as long as we are consistent)
- 2) Update RD = RD + disparity of the code output above.
  - Disparity bounded to between -9 to +9 inclusive at update point
- 3) Determine the next set to use
  - If RD is positive set NS = N
  - If RD is negative set NS = P
  - If RD is zero, no change to NS.

- Steps 1 to 3 repeats for every code output.

Gray Code

- Gray code needed only for case without bounded disparity.
- 5-bit RS symbol mapped to 2.5 PAM4 symbols or two RS symbols mapped to 5 PAM4 symbol
- If PAM4 symbol that straddles the two 5-bit codes is damaged, it corrupts two RS symbols.
- Gray code mitigates this by minimizing the chance that both bits are inverted.
  - i.e. (-1) 00 ⇔ 01 (-1/3), (-1/3) 01 ⇔ 11 (+1/3), (+1/3) 11 ⇔ 10 (+1)

Algorithm Latency



- All values in ns
- 400 ns+ sufficient for implementation
- 8n/(8n+1) Decoder
  - If FEC on, Algorithm latency is 0 since there is no waiting once syndrome is computed
  - If FEC off, need 12 bits to decode first byte. First PAM4 unmap presents 5 bits, so need 2 more unmap or 2 x 32ns = 64ns

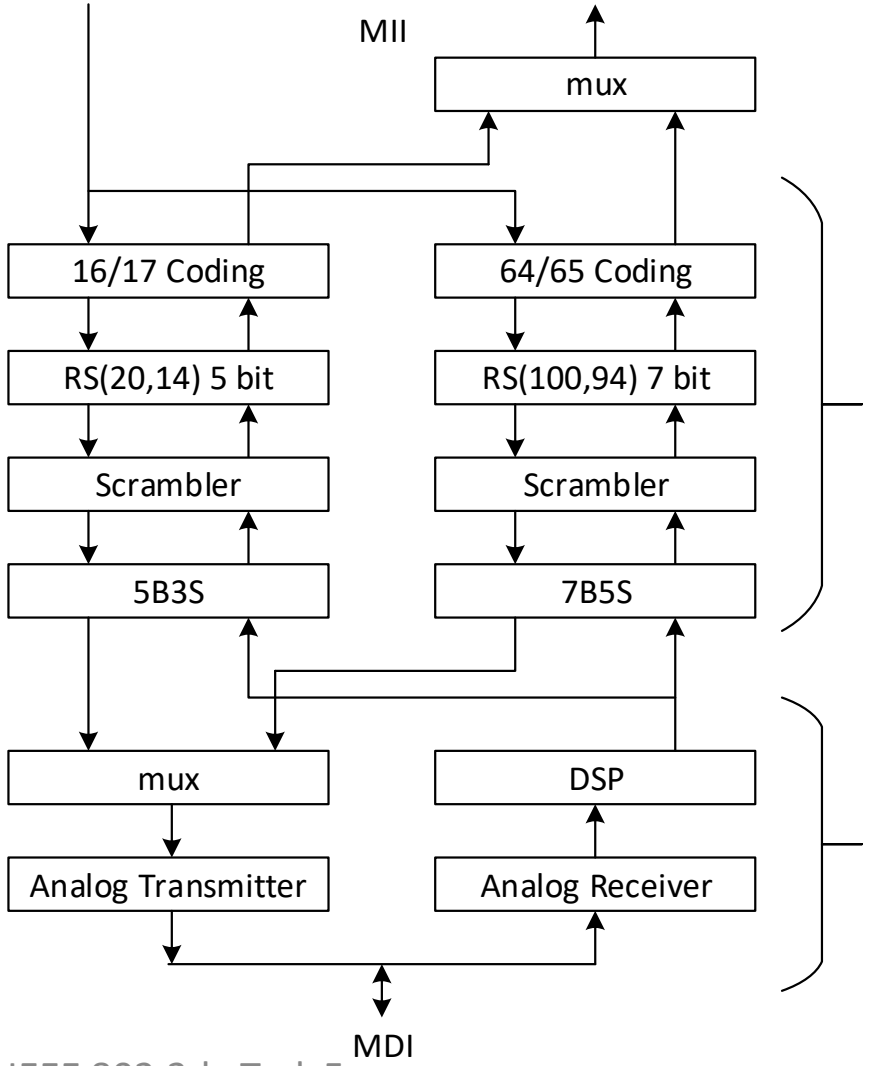
All values in ns	Low Latency No bounded disparity	Low Latency Bounded disparity	Long Reach	Low Latency Bounded disparity FEC turned off
Encoder	160	160	640	160
RS Encoder Underflow	192	192	384	192
PAM4 mapping	12.8	32	64	32
PAM4 unmap	12.8	32	64	32
Syndrome Calc	640	640	6400	0
Decoder	0	0	0	64
Total Algorithm Latency	1017.6	1056	7552	480
Margin for implementation	482.4	444		

PCS Parameter Summary

- Are we ok with different PHYs for low latency vs long reach?
- Low latency uses more bandwidth for short codes
- Long reach keeps baud rate low to reduce signal attenuation over long cables

	Low Latency No bounded disparity	Low Latency Bounded disparity	Long Reach
Encoder	16/17	16/17	64/65
Reed Solomon	RS(20, 14) 5-bits	RS(20, 14) 5-bits	RS(100, 94) 7-bits
Encoder blocks per RS Frame	4	4	10
OAM Bits per RS Frame	2	2	8
Encoder + RS Overhead	56.25%	56.25%	9.38%
Scrambler	33-bits	33-bits	33-bits
PAM4 mapping	gray coding	5-bit to 3-symbol	7-bit to 4-symbol
Max burst protection	96ns	96ns	192ns
Baud Rate	78.125 MBaud	93.75 MBaud	62.5 MBaud
Total Algorithm Latency (ns)	1017.6	1056	7552
Margin for implementation (ns)	482.4	444	

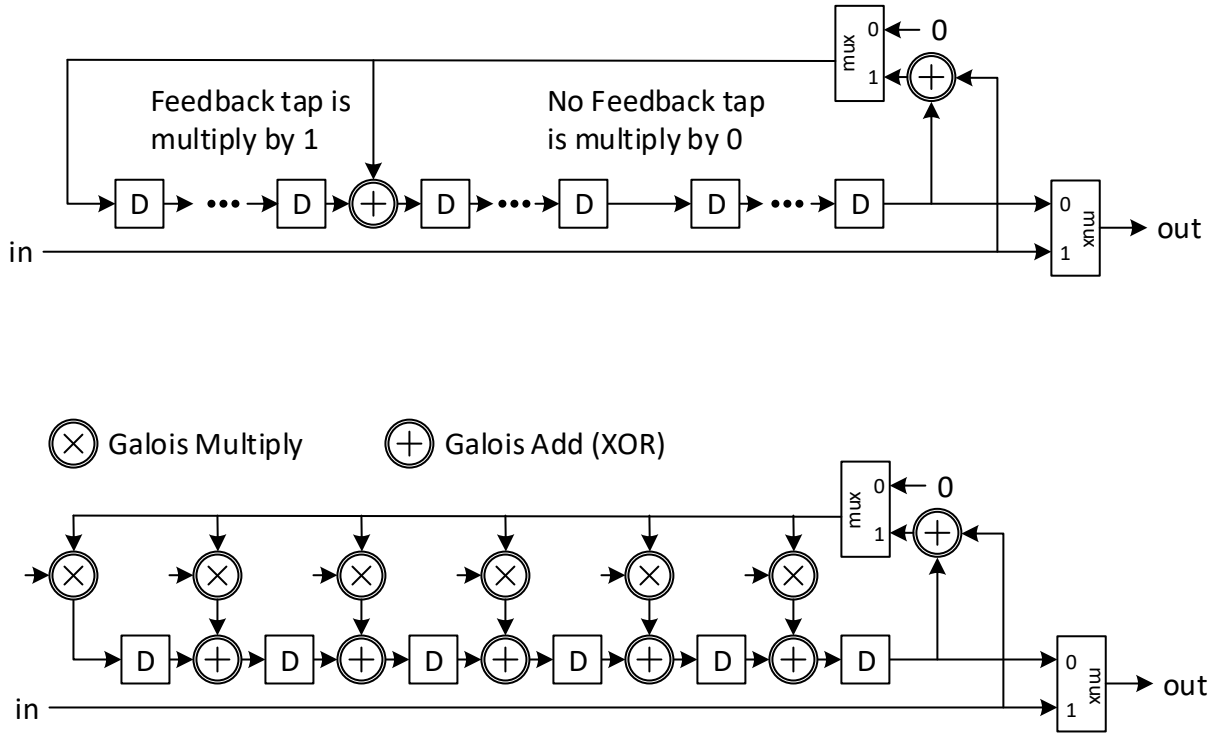
Implementation Considerations for Dual Mode



- Relatively inexpensive to implement
- Possible logic sharing
  - 8N/8N+1 parameterize the N for 2 and 8.
  - RS Frame FIFO
  - RS datapath (both has 6 parity symbols)
  - Same scrambler

- Expensive to implement
  - Keep PAM 4 for both
  - Low latency 93.75 MBaud is 1.5x of Long Reach of 62.5 MBaud
  - Simply run 1.5 times faster for Low Latency
  - 3.33 times echo taps for long reach

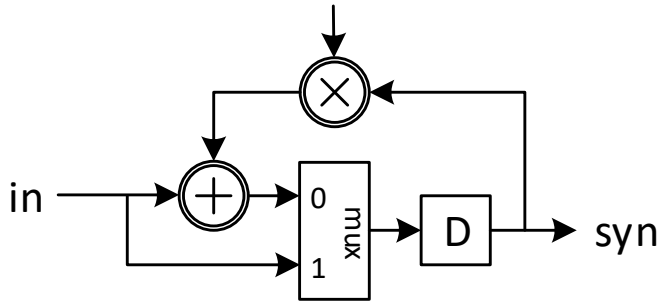
Reed Solomon Costs - Encoder



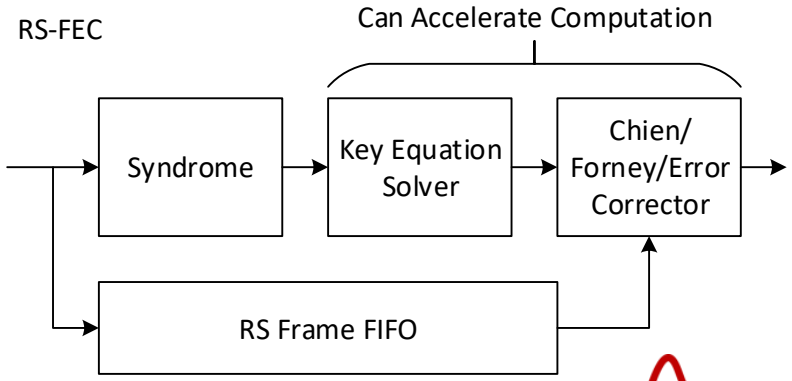
- CRC32 – 32 flip flops, XOR in feedback position
- RS(20, 14) 5 bits
  - Similar structure to CRC except 5-bit Galois operation.
  - 6 x 5 = 30 flip flops
  - Constant Galois multipliers
  - Approx 2x size of CRC32
- RS(100, 94) 7 bits
  - 6 x 7 = 42 flip flops
  - Approx 3x size of CRC32

Reed Solomon Costs - Decoder

- RS(20, 14) 5 bits Syndrome
  - 6 parity x 5-bits = 30 flip flops
  - Constant Galois multipliers
  - Approx 2x size of CRC32
- RS(100, 94) 7 bits
  - 6 x 7 = 42 flip flops
  - Approx 3x size of CRC32



- Other RS-FEC blocks generally have similar structure to syndrome block
  - Scales with number of parity symbols x bits/symbol.
  - Choose your favorite algorithm
- RS-FEC size not counting the FIFO
  - 6 parity 5-bit size ~10x to 15x CRC32
  - 6 parity 7-bit size ~15x to 20x CRC32
- Parallelization or run faster clock to lower latency on accelerable portions.



Friendly PLL Ratios

- 62.5 Mbaud, 78.125 Mbaud, and 93.75 Mbaud were matched with RS parameter selection and bounded disparity overhead
- 78.125 = 1.25 x 62.5
- 93.75 = 1.5 x 62.5

- Using 25MHz clock (Common inexpensive component) PLL ratios:
- 62.5MHz = 20/8 x 25MHz
- 78.125MHz = 25/8 x 25MHz
- 93.75MHz = 30/8 x 25MHz

- 62.5 Mbaud is about as low as we can go with required overhead for RS-FEC and bounded disparity

Summary

- Two PHY proposals tailored to reach low latency and long reach goals respectively
- Low gate count RS-FEC in both cases with similar 6 parity symbol structure
- Implementation efficient 5B3S and 7B4S codes – Bounded disparity guaranteed
- Friendly clock ratios for PLLs
- Common frontend for dual mode – simply scale the clock speed
- Minimal incremental cost for dual mode PHYs

- Next Steps:
  - Make decision whether we have 2 different PHYs
  - Do we want to baseline some portion of this proposal to put a stake in the ground?
  - What additional evaluation is required?

THANK YOU