



Canova Tech

The Art of Silicon Sculpting

PIERGIORGIO BERUTO
ANTONIO ORZELLI

IEEE802.3cg TF

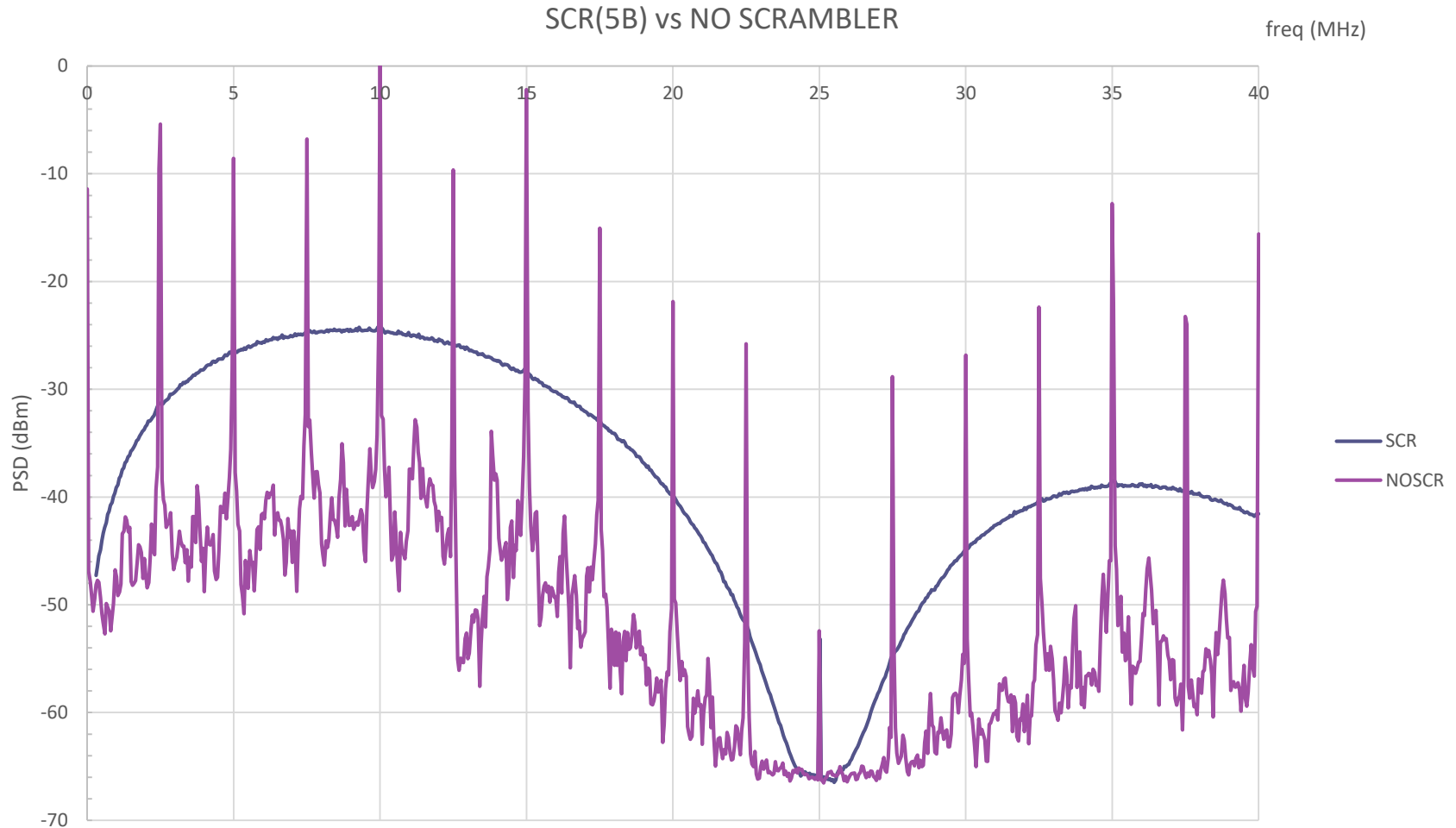
TIS scrambler proposal

March 28th, 2018

- **Martin Miller (Microchips)**
- **Steffen Graber (Pepperl+Fuchs)**

- Several contributions in the group to add scrambling in 10BASE-TIS
 - <http://www.ieee802.org/3/cg/public/adhoc/AlternateScrambleScheme20180228.pdf>
 - http://www.ieee802.org/3/cg/public/adhoc/tazebay_8023cg_further_thoughts_on_scrambling.pdf
 - http://www.ieee802.org/3/cg/public/Jan2018/tazebay_3cg_01b_0118.pdf
- There's general consensus in 802.3cg to define a scrambler for the TIS to improve EMI performance in case of repetitive patterns in packets payload
 - Not needed for BLW compensation as DME is intrinsically balanced

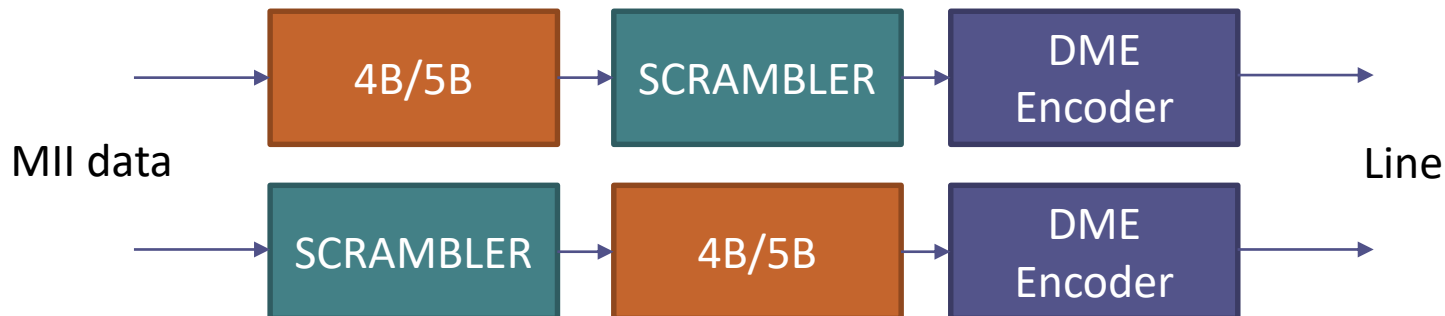
Effect of scrambler $x^{15} + x^4 + 1$ after 4B/5B enc.



- Continuous stream of 1530 bytes packets
- RBW = 10 KHz, VBW = 100 Hz
- 15 bit, fixed seed scrambler vs NO scrambler

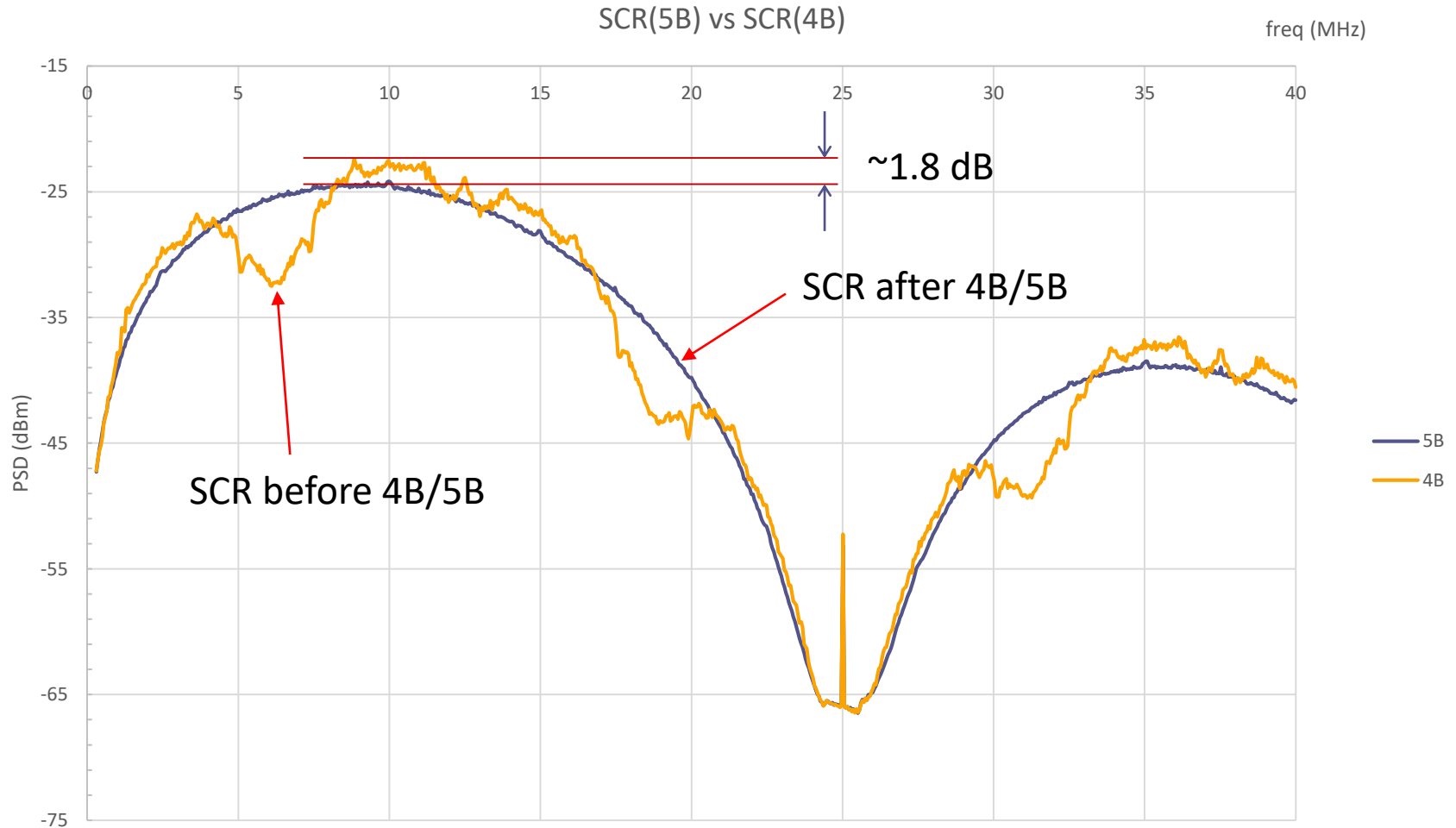
- Payload of all 0x00, all 0xFF or all 0x55
- More than 25 dB difference in peak PSD

Where to scramble?



- In principle scrambling after 4B/5B gives the best EMI performance (full randomization)
- However 4B/5B coding has properties:
 - 5B symbols that don't map on 4B data can be used for intra-PHY signaling
 - T1S PHY uses special symbol 'J', 'K', 'T', 'R' for SSD/ ESD
 - PLCA also uses 'N' for BEACON
 - JJ, JK, NN sequences are unique, allowing for synchronization and proper alignment to 5B boundary
- Scrambling the 5B symbols would kill these properties, adding complexity to the PHY
 - Having a mix of scrambled and unscrambled bits on the line (preamble / data) also looks bad from a system definition perspective
- Scrambling before 4B/5B encoding would preserve current architecture
 - **What about EMI performance?**

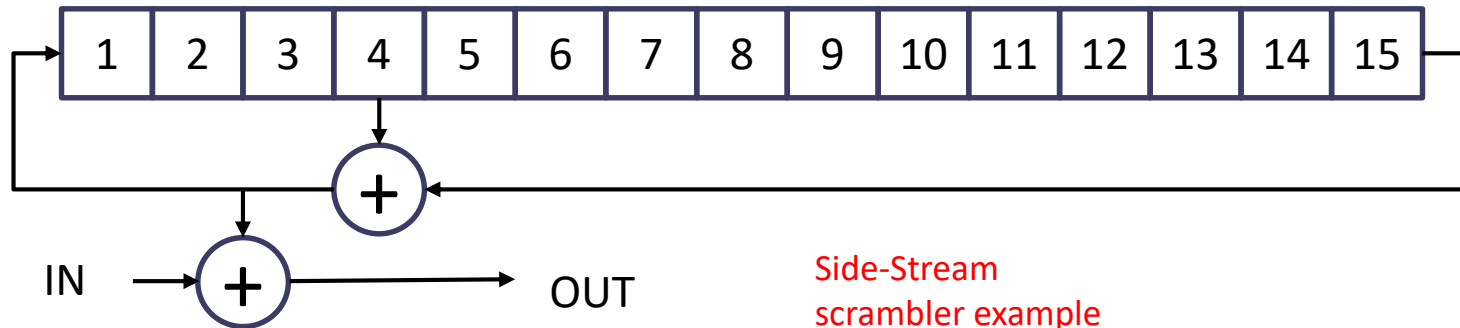
Side stream scrambler $x^{15} + x^4 + 1$



- Continuous stream of 1530 bytes packets
- RBW = 10 KHz, VBW = 100 Hz
- 15 bit, fixed seed scrambler
- Payload of all 0x00, all 0xFF or all 0x55
- Less than 2dB peak difference in PSD measure

What scrambler?

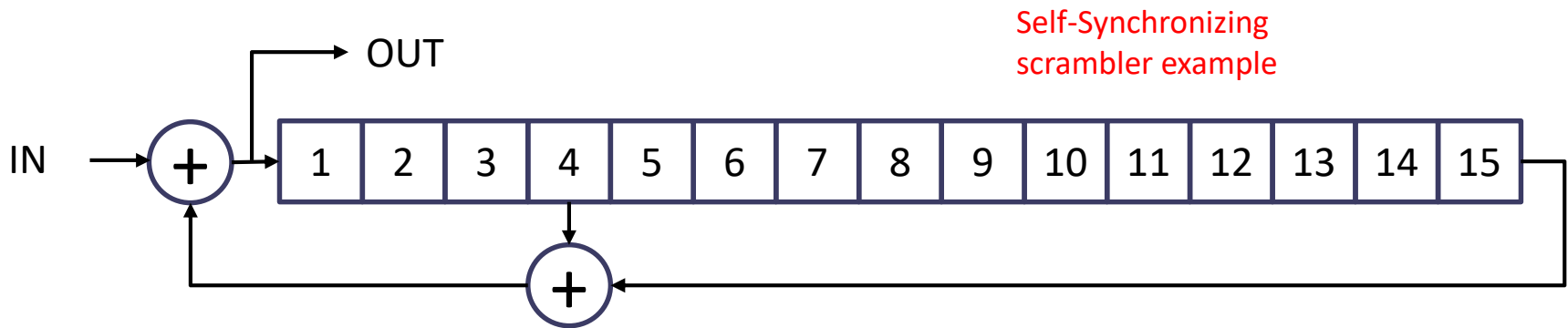
- Side-Stream (additive) scrambler



- Does not propagate errors
- Requires a seed
 - Fixed seed: simplest solution but...
 - Killer packet problem: it is possible to manufacture a packet that “resonates” with the scrambler resulting in a sequence of repetitive data
 - Not a problem for the receiver as DME is inherently balanced
 - Problem from EMI perspective (same as not having the scrambler at all)
 - Random seed: solves the killer packet problem but...
 - Adds complexity to the PHY: how to sync the scrambler?
 - TIS has no training sequence, no continuous idle (would not be possible on the mixing segment anyway)
 - Sending the seed in the preamble looks like a possibility, however...
 - Seed must be sent at least twice to allow error detection: **not enough bytes in the preamble! Would preclude 802.3br**
 - Adds complexity to both receiver and transmitter


What scrambler? (cont.)

- Self-Synchronizing (multiplicative) scrambler

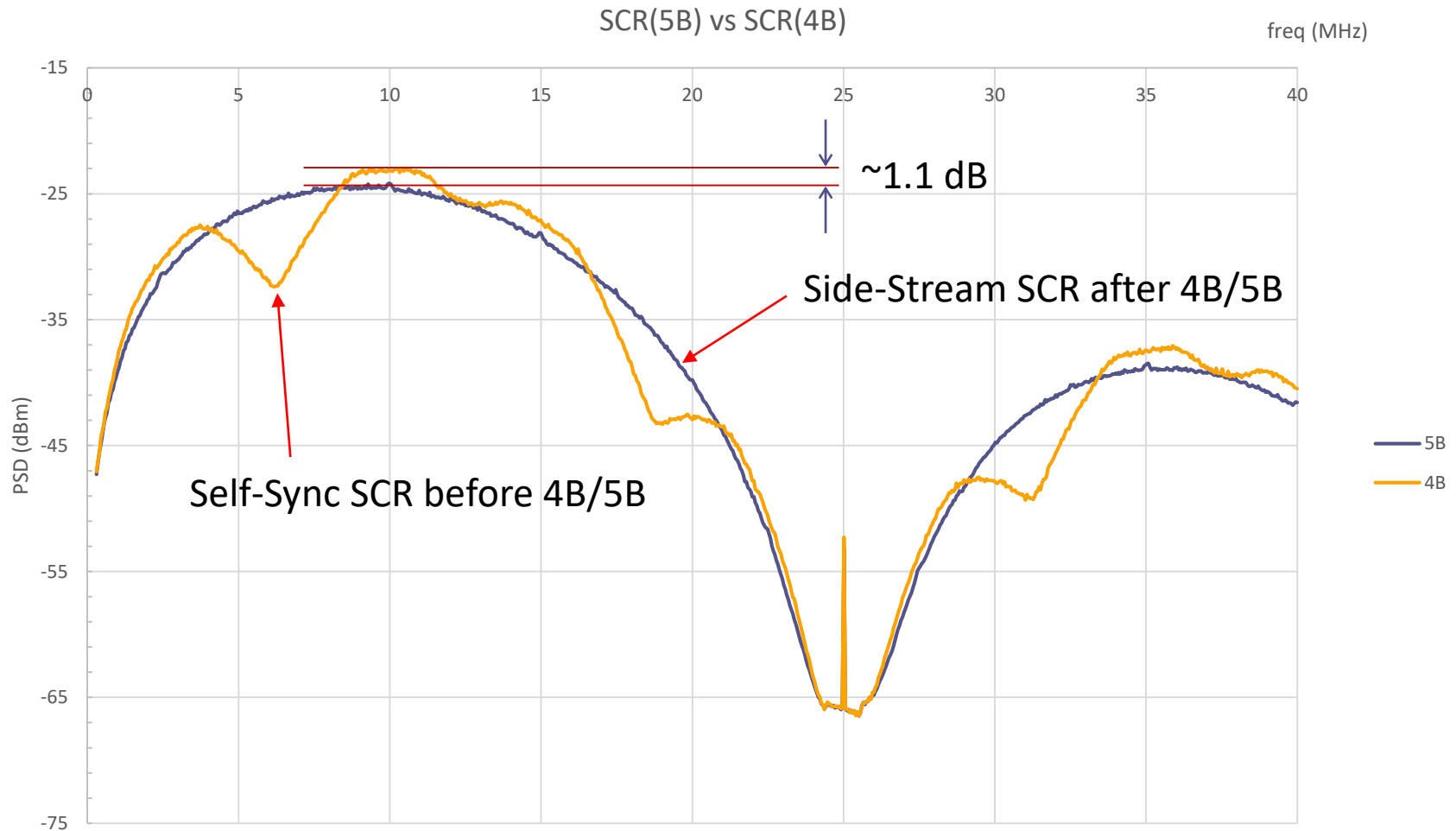


- Does not require a seed (self-synchronizing)
 - Very easy to implement
 - Requires exactly a number of bits equal to its degree to achieve lock (only 2 bytes for a 15 bit scrambler)
- Propagates errors
 - A two tap descrambler multiplies input errors by three
 - In principle, error detection could be compromised
 - Is it really a problem for 10BASE-T1S?
- Same EMI performance of side-stream scrambler
 - Measured PSD depends on polynomial, not on scrambler type

Error propagation

- Ethernet FCS can detect up to 4 errors with 100% chance for packets up to 91607 bits
 - Can also detect with 100% chance one 32-bit burst error or two 9-bit burst errors in a single data unit
 - Multiplying the number of errors by three sounds like FCS strength would be compromised
 - **Propagated errors, however, are not independent from each other (they're generated by the scrambler polynomial)**
- Let's see
 - $G(x) = 802.3 \text{ CRC32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
 - Not reducible in GF(2)
 - $D(x) = \text{packet payload including FCS}$ $S(x) = \text{scrambler polynomial}$ $E(x) = \text{error polynomial}$
 - Transmitted data unit (multiplicative scrambler):
 $S(x) * D(x)$
 - At receiver:
 $S(x) * D(x) + E(x)$
 - After de-scrambling:
 $S(x) * (S(x) * D(x) + E(x)) = D(x) + S(x) * E(x)$
 - FCS at receiver
 $\{ D(x) + S(x) * E(x) \} / G(x) = D(x) / G(x) + S(x) * E(x) / G(x)$
 - Reminder of $D(x) / G(x)$ is the original FCS (without errors)
 - FCS collision (undetected error) if-f reminder of $S(x)*E(x) / G(x) = 0$ 
 - If $E(x)$ is divisible by $G(x)$ we have a collision
 - but doesn't depend on the scrambler \rightarrow ignore this case
 - Else if $S(x)$ has at least one prime factor in common with $G(x)$, remaining factors may cancel up with $E(x)$ thus causing a collision
 - That would indeed weaken the FCS
 - However $G(x)$ is non reducible \rightarrow as long as $S(x)$ order is less than $G(x)$ order, reminder will always be $\neq 0$
 - **FCS strength not altered by choosing a scrambler polynomial of degree less than 32**

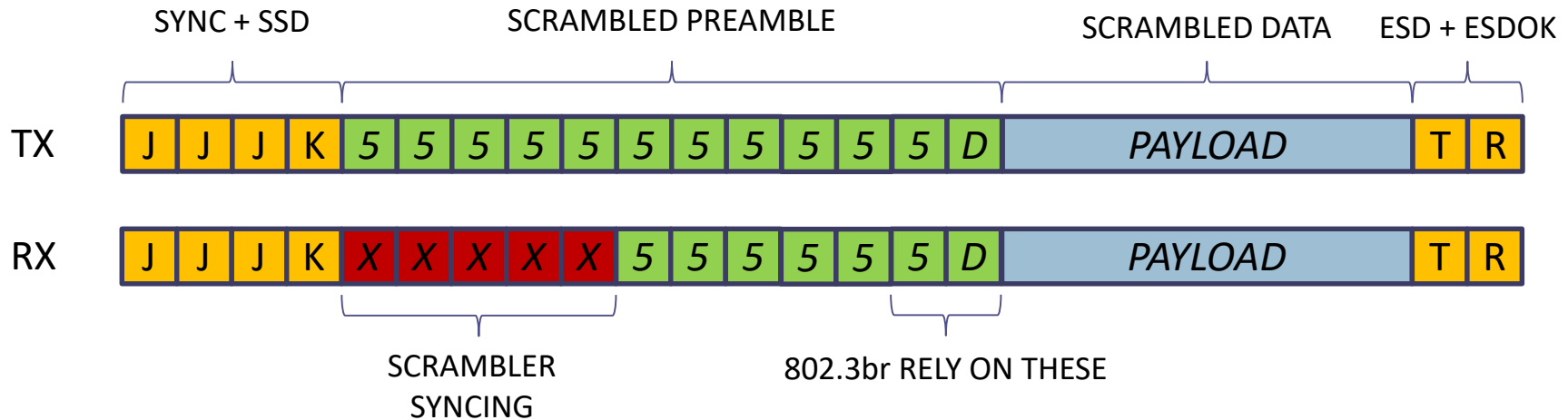
Self-Synchronizing scrambler $x^{17}+x^{14}+1$



- Continuous stream of 1530 bytes packets
- RBW = 10 KHz, VBW = 100 Hz
- 17 bit, multiplicative scrambler

- Payload of all 0x00, all 0xFF or all 0x55
- ~1.1 dB peak difference in PSD measure

Scrambling proposal



- Scrambler polynomial: $x^{17} + x^{14} + 1$
- Start scrambling MII data (including preamble) after JJJK
- SYNC, SSD, ESD, ESDOK, ESDERR and BEACON are **NOT** scrambled
 - 4B/5B coding properties preserved
 - JJJK is good for synchronizing DME (00011 00011 00011 10001)
- Minimal changes to clause 147, no changes to clause 148

- **147.1.2 Operation of 10BASE-T1S**
 - The 10BASE-T1S PHY utilizes two level Differential Manchester Encoding (DME) modulation transmitted at a 12.5 MBd rate (\pm TBD). **A 17-bit self-synchronizing scrambler is used to improve the EMC performance.** 4B/5B encoding is used to further improve EMC performance and to perform out-of-band signaling among the connected PHYs. [. . .]
 - The 4B/5B mapping **is and the scrambler are** contained in the PCS (see 147.2) while the DME encoder/decoder is contained in the PMA (see 147.4).

- **147.2.2.3 Functions**

- ENCODE

In the PCS transmit process, this function takes as its arguments ~~the pcs_txd input~~ one data nibble, scrambles it into $Sd_n[3:0]$ as defined in 147.2.2.5 and returns the corresponding 5B symbol as defined in Table 147–1.

Proposed text changes (cont.)

- **147.2.2.5 Self-synchronizing scrambler**

The PCS Transmit function shall implement multiplicative scrambling using the following generator polynomial:

$$g(x) = 1 + x^{14} + x^{17}$$

An implementation of self-synchronizing scrambler by linear-feedback shift register is shown in figure TBD#1. The bits stored in the shift register delay line at time n are denoted by $Scr_n[16:0]$. At every MII clock cycle, for each bit of $TXD[3:0]$ the scrambler is advanced by one bit, and the output bit $Sd_n[i]$ represented by the exclusive OR of $Scr_n[13]$, $Scr_n[16]$ and $TXD[i]$ is shifted in as $Scr_{n+1}[0]$, with i ranging from 0 to 3 (i.e. LSB first). The scrambler is reset upon execution of the PCS Reset function. If PCS Reset is executed, all bits of the 17-bit vector representing the self-synchronizing scrambler state are arbitrarily set. The initialization of the scrambler state is left to the implementer. In no case shall the scrambler state be initialized to all zeros.

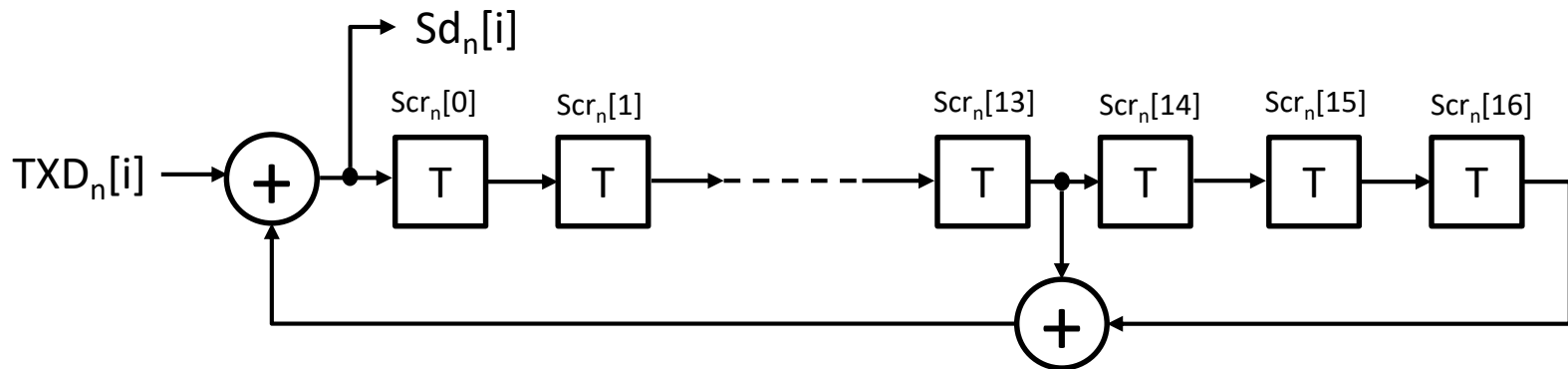


Figure TBD#1

Proposed text changes (cont.)

- **147.2.3.2 Functions**

- **DECODE**

In the PCS Receive process, this function takes as its arguments ~~the sym_rx input data from PMA~~ one 5B symbol, decodes the corresponding nibble as defined in Table 147-1 and returns the descrambled result as defined in 147.2.3.4. ~~and returns the corresponding 4B MII data as defined in Table 147-1.~~ If a violation of the encoding rules is detected, PCS Receive asserts the signal RX_ER for at least one symbol period.

- **147.2.3.1 Variables**

receiving [...]

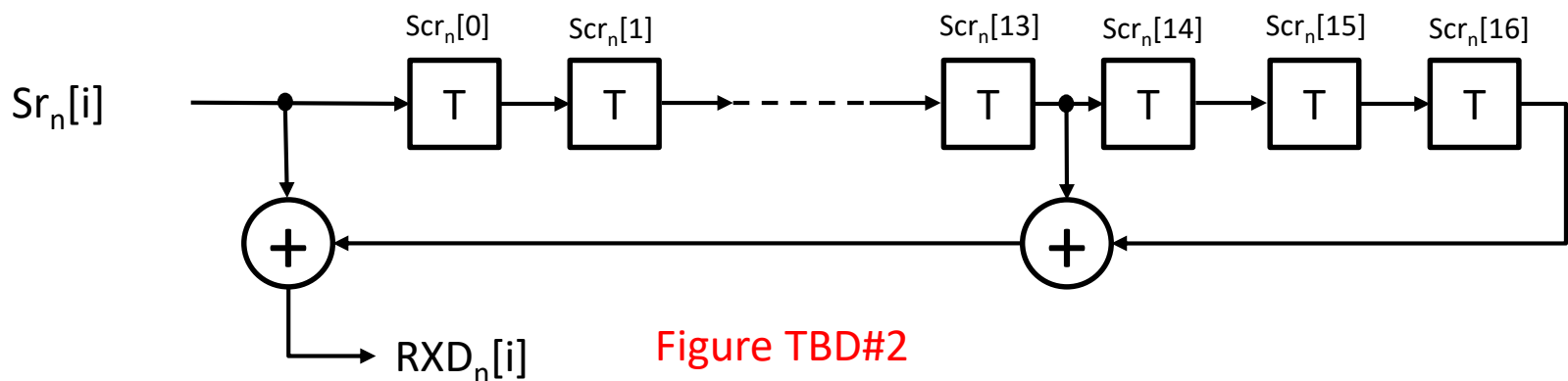
precnt counter for preamble regeneration

pcs_rxdv [...]

Proposed text changes (cont.)

- **147.2.3.4 Self-synchronizing descrambler**

The PCS Receive function shall descramble the 5B4B decoded data stream and return the proper nibble for generation of $RXD[3:0]$ to the MII. The descrambler shall employ the polynomial defined in 147.2.2.5. An implementation of self-synchronizing descrambler by linear-feedback shift register is shown in figure TBD#2. The bits stored in the shift register delay line at time n are denoted by $Dcr_n[16:0]$. At every MII clock cycle, each bit of $Sr_n[3:0]$ is shifted in as new $Scr_n[0]$ and the descrambler is advanced by one bit. The output bit $RXD[i]$ represented by the exclusive OR of $Dcr_n[13]$, $Dcr_n[16]$ and $Dr_n[i]$ is generated, with i ranging from 0 to 3 (i.e. LSB first). The descrambler is reset upon execution of the PCS Reset function. If PCS Reset is executed, all bits of the 17-bit vector representing the self-synchronizing descrambler state are arbitrarily set. The initialization of the descrambler state is left to the implementer.



Proposed text changes (cont.)

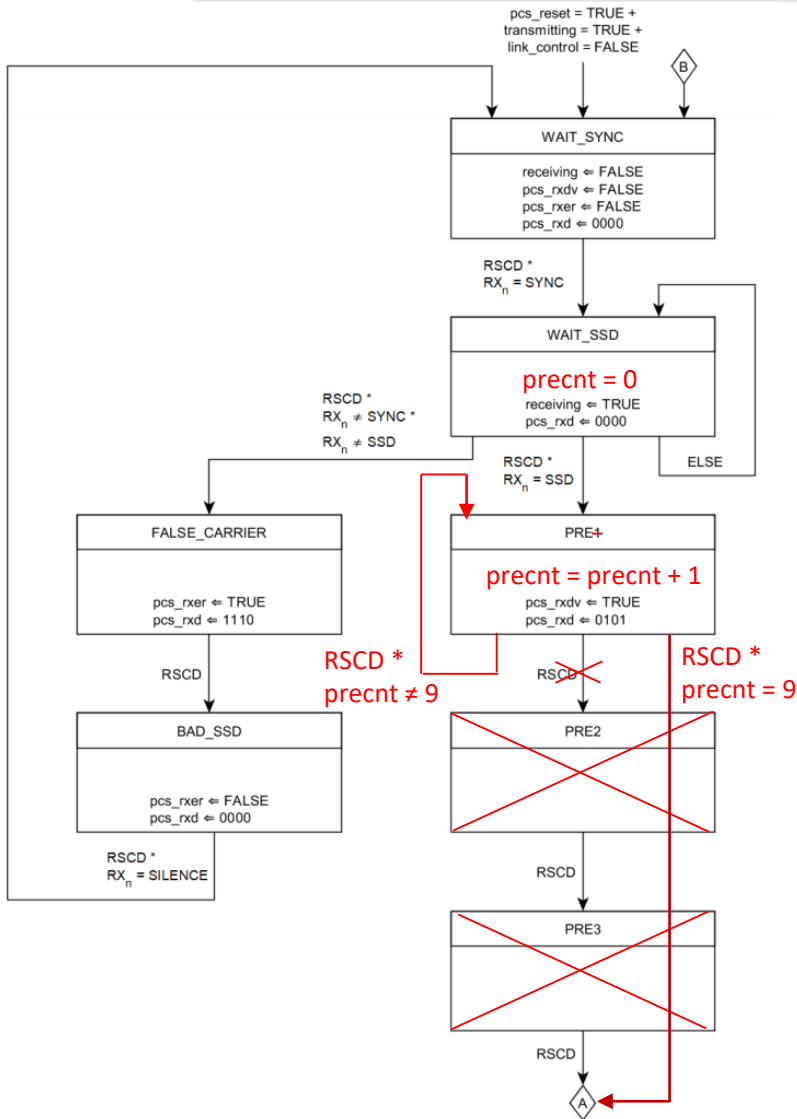


Figure 147-5—PCS Receive state diagram (1 of 2)

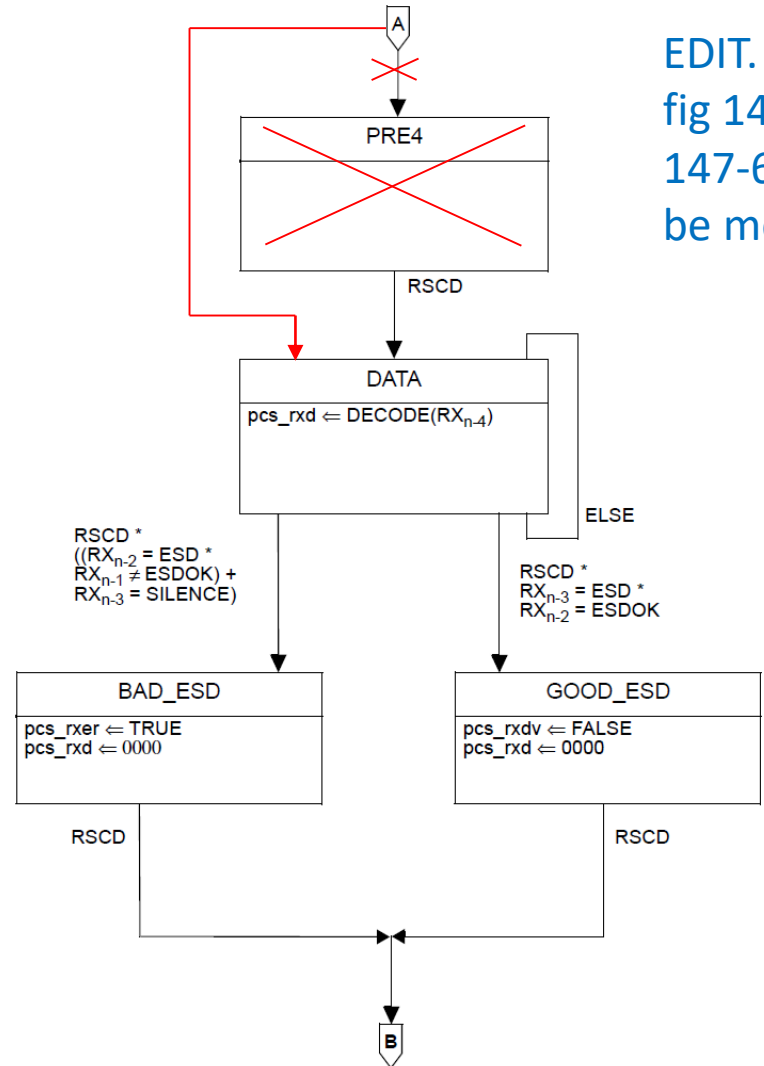


Figure 147-6—PCS Receive state diagram (2 of 2)

EDIT. NOTE:
fig 147-5 and
147-6 could
be merged

- A scrambler improves EMI performance
 - Peak reduction of at least 25 dB in emission PSD
- Scrambling before 4B/5B encoder gives ~ 1.1 dB less margin than scrambling after 4B/5B on the peak PSD but
 - Keeps 4B/5B features, preserving current architecture
 - Simpler solution, lower PHY complexity
 - Minimizes spec changes
- Self-Synchronizing (multiplicative) scrambler looks better
 - Much simpler solution
 - Does not break 802.3br (aka Ethernet preemption)
 - Error propagation is not a concern
- Propose to adopt text changes as in slide #12 and following