ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Proposals to Adopt for the PCS for 100BASE-T1L - Sequence Ordered Sets

Brian Murray
Jacobo Riesco
Philip Curran

Proposal for PCS for 100BASE-T1L PHY: 8N/(8N+1) Encoding

- Defined by the following pseudo-code where N is the number of octets encoded in a block
  - N = 8 when the Reed-Solomon FEC is used and N = 2 when it is not used
  - Octets within a block are numbered using and increasing index n, from 0 to N-1, with n = 0 being the first octet of the block presented on the MII interface.

```
TC[n]          : 0 if octet n is a data octet, i.e., if the octet n contains two MII data nibbles, TXD[2n][0:3] and  TXD[2n+1][0:3]; 1 otherwise
TC[-1]         : 1 by definition
TD[n][0:7]     : MII octet n (TD[n][0:3] = TXD[2n][0:3], TD[n][4:7] = TXD[2n+1][0:3]) if TC[n] = 0
C[n][0:2]      : MII control code n, corresponding to MII data nibbles 2n, 2n+1 as per the encoding table
B[0:8N]        :  8N+1 block. Bit 0 transmitted first
OR(n)          : Bitwise OR of TC[n:N-1]
OR(N)          : 0 by definition
NEXT(n)[0:2]   : bit position of lowest bit in TC[n:N-1] that is a 1. Bit 3 is MSB
M(n)[0:1]      : MII mode n, corresponding to MII data nibbles 2n, 2n+1
                 M[n][0] = 1 if encoded symbol is CD; else 0
                 M[n][1] = TXD2n[0] if encoded symbol is CD; else OR(n+1)

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

Proposed Constant Latency MII to 8N/8N+1 Encoding

- We propose the following for the constant latency MII to 8N/8N+1 encoding
  - May or may not include sequence ordered sets in the PCS 8N/8N+1 encoding
    - May or may not preclude end to end communication of sequence ordered sets
    - Can always use sequence ordered sets between the PHY and the RS
  - Do not group/ungroup 3 nibbles at the start and do not expand/compress one idle at the end of the packet
    - This avoids an increase of the transmit latency by one MII clock cycle (40 ns)
    - This results in constant **lower** latency for packets
    - If not supporting end to end sequence ordered sets can use state-less encoding which is simpler
  - The Codes for sequence ordered sets starting on an odd cycle (Co) are not needed
  - Do not have to deal with unexpected behaviour that may occur in case of errors
  - Encoder and decoder do not need to remain synchronized
- Simpler and more robust state-less encoding
  - Always decodes an octet into two nibbles
  - For the case of not supporting end to end sequence ordered sets it is state-less and dealing with the case of errors is simpler

Proposal with Sequence Ordered Sets NOT Supported

- The following slides describe the control codes and example cases for Constant Latency MII to 8N/8N+1 Encoding where sequence ordered sets are not included in the PCS 8N/8N+1 encoding
  - This precludes end to end communication of sequence ordered sets

Control Codes – Sequence Ordered Sets Not Supported

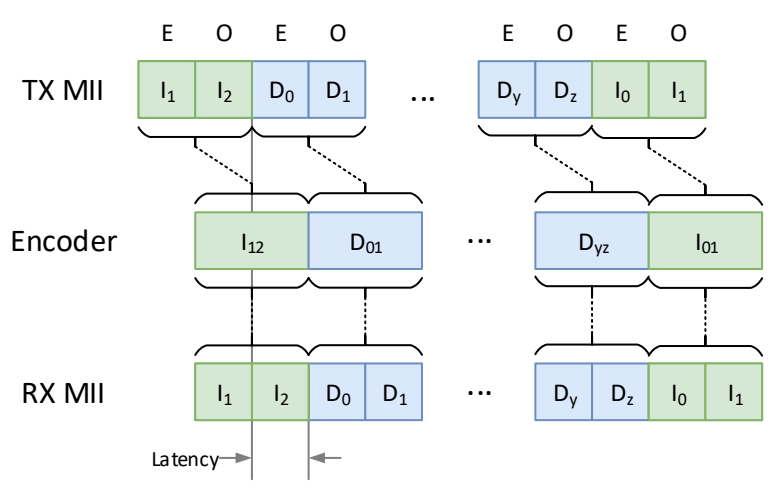
3	4	5	6	7		
Mode		**Control Code**			**Symbol**	**Definition**
0	0	0	0	0		No more control codes, Reserved
0	0	0	0	1	E	No more control codes, Transmit Error Propagation
0	0	0	1	0	I	No more control codes, Normal Inter-Frame (Idle)
0	0	0	1	1	Cs	No more control codes, Start of Frame with leading Idle
0	0	1	0	0		No more control codes, Reserved
0	0	1	0	1	L	No more control codes, Assert Low Power Idle
0	0	1	1	0		No more control codes, Reserved
0	0	1	1	1		No more control codes, Reserved
0	1	0	0	0		More control codes, Reserved
0	1	0	0	1	E	More control codes, Transmit Error Propagation
0	1	0	1	0	I	More control codes, Normal Inter-Frame (Idle)
0	1	0	1	1	Cs	More control codes, Start of Frame with leading Idle
0	1	1	0	0		More control codes, Reserved
0	1	1	0	1	L	More control codes, Assert Low Power Idle
0	1	1	1	0		More control codes, Reserved
0	1	1	1	1		More control codes, Reserved
1	0	0	0	0	CD0	Dribble Nibble, Data = 0x0
1	0	0	0	1	CD8	Dribble Nibble, Data = 0x8
1	0	0	1	0	CD4	Dribble Nibble, Data = 0x4
1	0	0	1	1	CDC	Dribble Nibble, Data = 0xC
1	0	1	0	0	CD2	Dribble Nibble, Data = 0x2
1	0	1	0	1	CDA	Dribble Nibble, Data = 0xA
1	0	1	1	0	CD6	Dribble Nibble, Data = 0x6
1	0	1	1	1	CDE	Dribble Nibble, Data = 0xE
1	1	0	0	0	CD1	Dribble Nibble, Data = 0x1
1	1	0	0	1	CD9	Dribble Nibble, Data = 0x9
1	1	0	1	0	CD5	Dribble Nibble, Data = 0x5
1	1	0	1	1	CDD	Dribble Nibble, Data = 0xD
1	1	1	0	0	CD3	Dribble Nibble, Data = 0x3
1	1	1	0	1	CDB	Dribble Nibble, Data = 0xB
1	1	1	1	0	CD7	Dribble Nibble, Data = 0x7
1	1	1	1	1	CDF	Dribble Nibble, Data = 0xF

Control Codes Encoding Table (with sequence ordered sets not supported)

Start on Even Cycle, Packet Even Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
- Decoding: split octet to nibbles

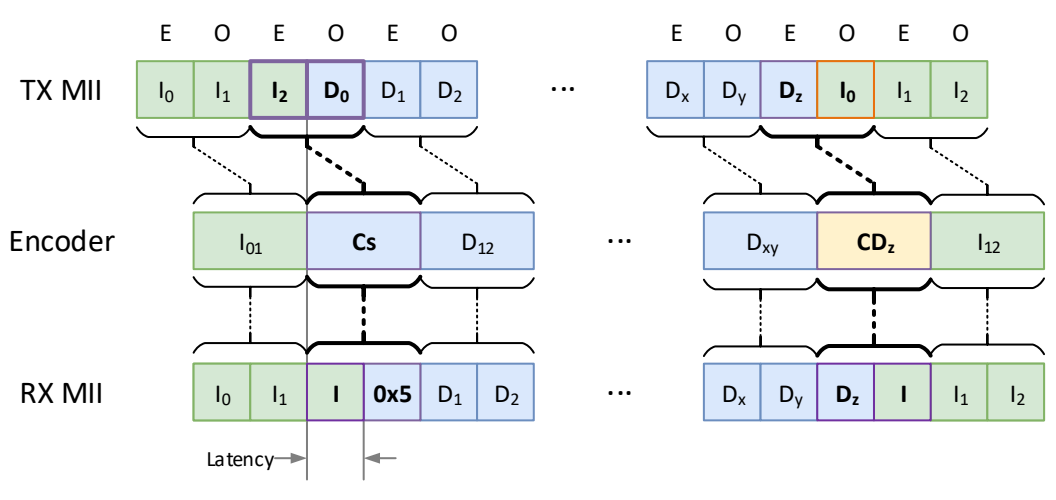
New proposal



Start on Odd Cycle, Packet Even Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Idle nibble followed by first preamble data nibble in octet, encoded to Cs
  - Data nibble (D_z) followed by idle nibble in octet, encoded to CD_z
- Decoding: Split octet to nibbles
  - Cs decoded to idle nibble followed by 0x5 data nibble
  - CD_z decoded to D_z data nibble followed by idle nibble

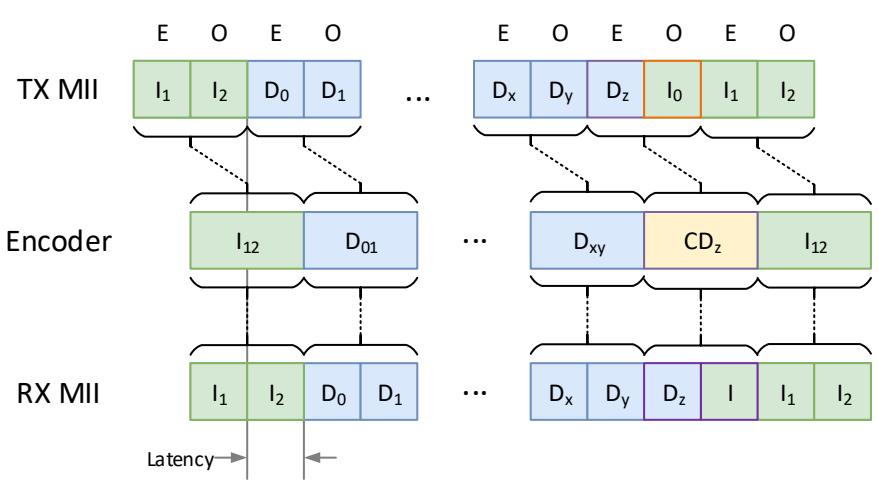
New proposal



Start on Even Cycle, Packet Odd Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Data nibble (D_z) followed by Idle nibble in octet, encoded to CD_z
- Decoding: Split octet to nibbles
  - CD_z decoded to D_z data nibble followed by Idle nibble

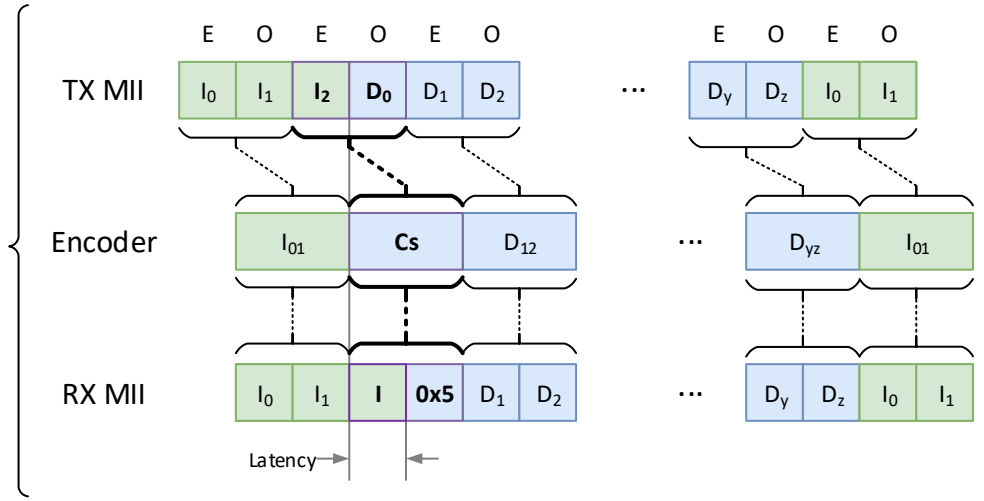
New proposal



Start on Odd Cycle, Packet Odd Number of Nibbles

- Encoding: combine even + odd nibbles into octet presented to encoder
  - Idle nibble followed by first preamble data nibble in octet, encoded to Cs
- Decoding: Split octet to nibbles
  - Cs decoded to idle nibble followed by 0x5 data nibble

New proposal



Proposal with Sequence Ordered Sets Supported

- The following slides describe the control codes and example cases for Constant Latency MII to 8N/8N+1 Encoding where sequence ordered sets are supported in the PCS 8N/8N+1 encoding
  - This supports end to end communication of sequence ordered sets
  - This may be a sequence ordered set initiated in the RS to be transmitted to the remote RS
  - If a new MII uses sequence ordered sets to communicate between the PHY and the MAC, this may be a sequence ordered set transparently carried PCS to PCS

Rules for Constant Latency MII Encoding Onto 8N/8N+1

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Rules summary with Sequence Ordered Sets Supported

TX MII	Data at encoder input	Data at encoder output	RX MII
Start of packet at odd cycle	Group idle and data to Cs control symbol.	Cs	Output 2 nibbles: idle, 0x5
End of packet octet is data, idle	Group data and idle to one of the CD control symbols	CD	Output 2 nibbles: data, idle
[1] Start of sequence at odd cycle. First nibble is idle, lpi or data	Group idle and sequence to I control symbol	I	Output 2 nibbles: idle, idle
	Group LPI and sequence to LI control symbol	L	Output 2 nibbles: LPI, LPI
	Group data and sequence to CD control symbol	CD	Output 2 nibbles: data, idle
	Convert following sequence to octet sequence control symbol	O	Output 2 nibbles: Sequence, Sequence
[1] End of sequence that started on odd cycle	Group idle and the two preceding nibbles to data	D	Output 2 nibbles: data, data
Else	Group even and odd nibbles to octet	Else	Output 2 nibbles

[1] Only required to encode sequence ordered sets for transmission between link-partners

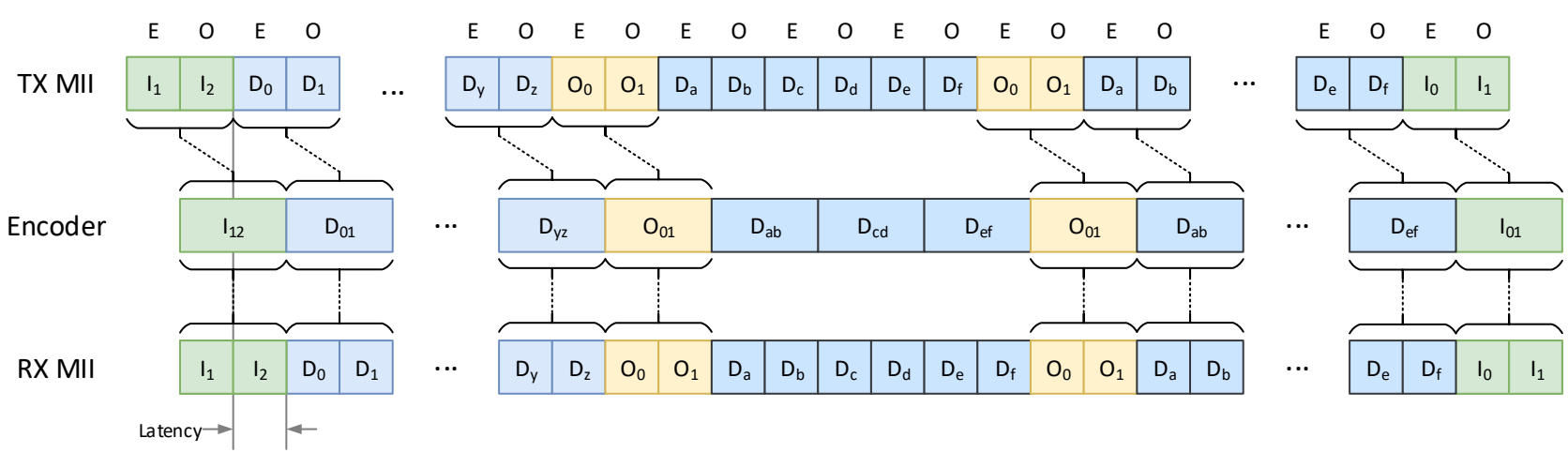
Control Codes – Sequence Ordered Sets Supported

3	4	5	6	7		
Mode		**Control Code**			**Symbol**	**Definition**
0	0	0	0	0	O	No more control codes, Sequence
0	0	0	0	1	E	No more control codes, Transmit Error Propagation
0	0	0	1	0	I	No more control codes, Normal Inter-Frame (Idle)
0	0	0	1	1	Cs	No more control codes, Start of Frame with leading Idle
0	0	1	0	0		No more control codes, Reserved
0	0	1	0	1	L	No more control codes, Assert Low Power Idle
0	0	1	1	0		No more control codes, Reserved
0	0	1	1	1		No more control codes, Reserved
0	1	0	0	0	O	More control codes, Sequence
0	1	0	0	1	E	More control codes, Transmit Error Propagation
0	1	0	1	0	I	More control codes, Normal Inter-Frame (Idle)
0	1	0	1	1	Cs	More control codes, Start of Frame with leading Idle
0	1	1	0	0		More control codes, Reserved
0	1	1	0	1	L	More control codes, Assert Low Power Idle
0	1	1	1	0		More control codes, Reserved
0	1	1	1	1		More control codes, Reserved
1	0	0	0	0	CD0	Dribble Nibble, Data = 0x0
1	0	0	0	1	CD8	Dribble Nibble, Data = 0x8
1	0	0	1	0	CD4	Dribble Nibble, Data = 0x4
1	0	0	1	1	CDC	Dribble Nibble, Data = 0xC
1	0	1	0	0	CD2	Dribble Nibble, Data = 0x2
1	0	1	0	1	CDA	Dribble Nibble, Data = 0xA
1	0	1	1	0	CD6	Dribble Nibble, Data = 0x6
1	0	1	1	1	CDE	Dribble Nibble, Data = 0xE
1	1	0	0	0	CD1	Dribble Nibble, Data = 0x1
1	1	0	0	1	CD9	Dribble Nibble, Data = 0x9
1	1	0	1	0	CD5	Dribble Nibble, Data = 0x5
1	1	0	1	1	CDD	Dribble Nibble, Data = 0xD
1	1	1	0	0	CD3	Dribble Nibble, Data = 0x3
1	1	1	0	1	CDB	Dribble Nibble, Data = 0xB
1	1	1	1	0	CD7	Dribble Nibble, Data = 0x7
1	1	1	1	1	CDF	Dribble Nibble, Data = 0xF

Control Codes Encoding Table (with sequence ordered sets supported)

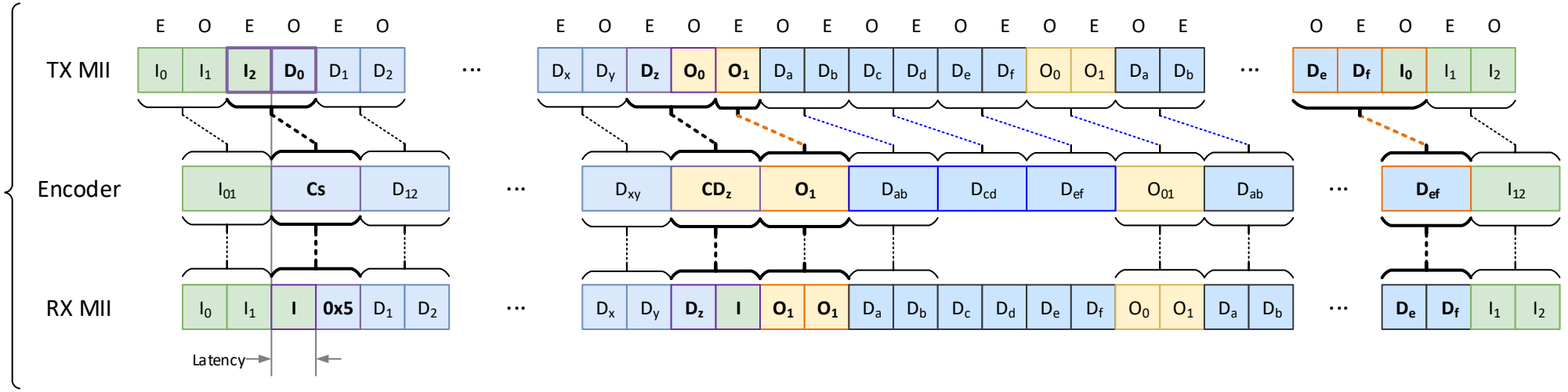
Start on Even Cycle, Packet Even Number of Nibbles

New proposal



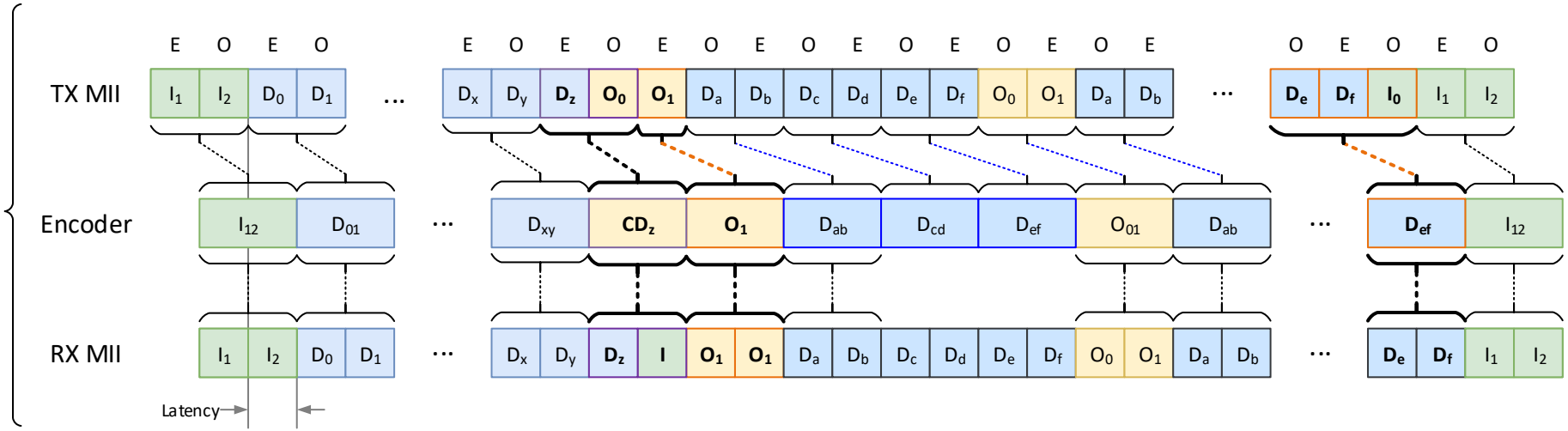
Start on Odd Cycle, Packet Even Number of Nibbles

New proposal



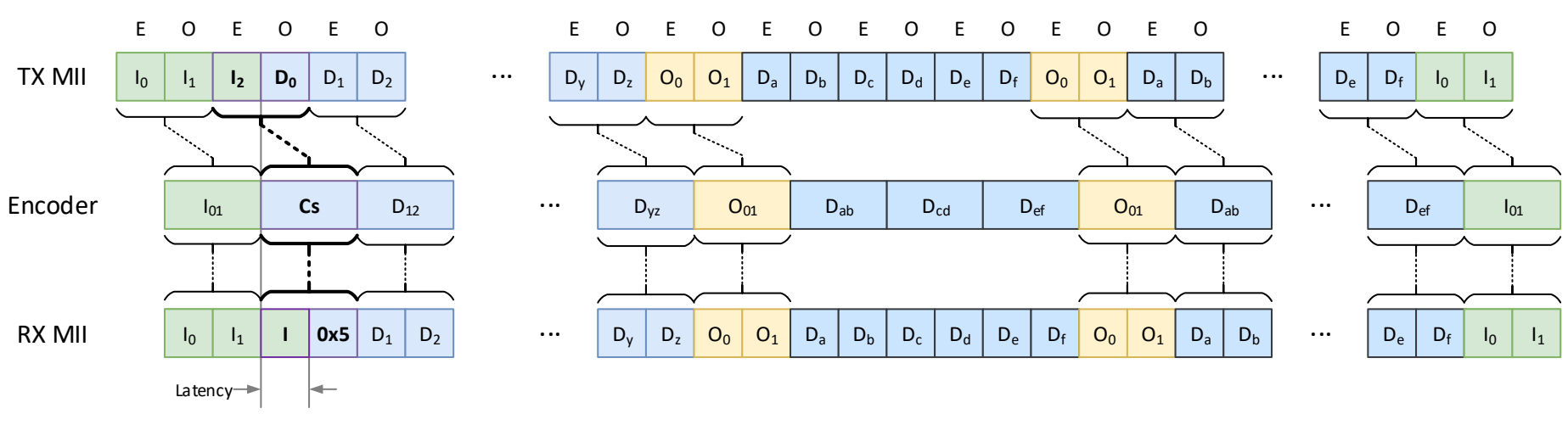
Start on Even Cycle, Packet Odd Number of Nibbles

New proposal



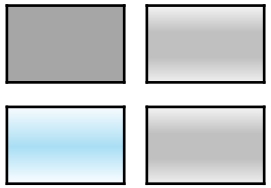
Start on Odd Cycle, Packet Odd Number of Nibbles

New proposal



Constant Latency MII to 8N/8N+1 Encoding/Decoding

Nibble		2n+1				
		D_y	IDL	LPI	SEQ	ERR
2n	D_x	D_x, D_y	CD_x	**CD_x**	**CD_x**	E
	IDL	Cs	I	I	**I**	E
	LPI	*E*	LI	LI	LI	*E*
	SEQ	*E*	***E***	***E***	O	*E*
	ERR	E	E	E	E	E



Only to encode sequence ordered sets

Should not appear at encoder