64b/66b PCS

updated 6/30/2000

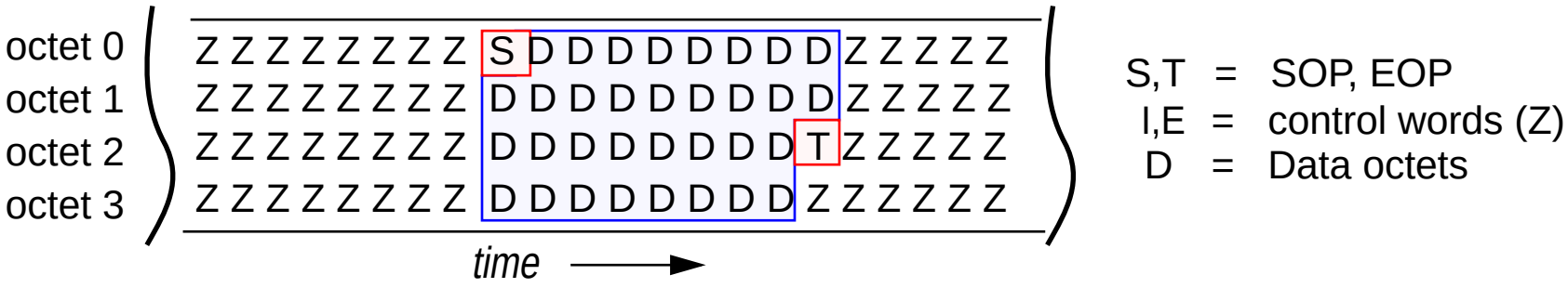
state machines modified 7/17/2000

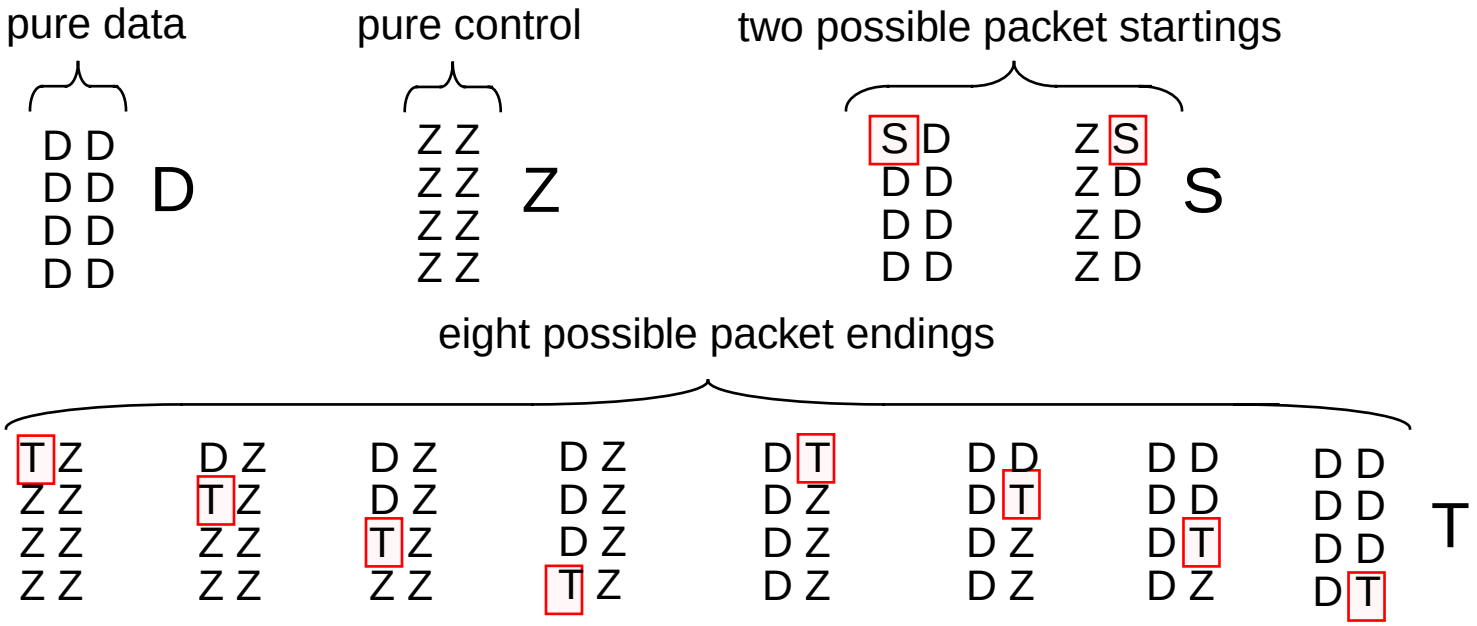
Rick Walker	Agilent	Howard Frazier	Cisco
Richard Dugan	Agilent	Paul Bottorff	Nortel
Birdy Amrutur	Agilent	Shimon Mueller	Sun
Rich Taborek	nSerial	Brad Booth	Intel
Don Alderrou	nSerial	Kevin Daines	World Wide Packets
John Ewen	IBM	Osamu Ishida	NTT
Mark Ritter	IBM	Jason Yorks	Cielo
Al Bezoni	Lucent	Henning Lysdal	Giga/Intel
Drew Plant	Agilent	Justin Chang	Quake

Topics

- Code review and update
- Test vectors
- Bit ordering sequence
- Frame sync algorithm and state machine
- TX,RX error detection state machines
- Optional code features
- Summary

Building frames from 10GbE RS symbols





Code Overview

Data Codewords have “01” sync preamble

Mixed Data/Control frames are identified with a “10” sync preamble. Both the coded 56-bit payload and TYPE field are scrambled

00,11 preambles are considered code errors and cause the packet to be invalidated by forcing an error (E) symbol on coder output

Code Summary

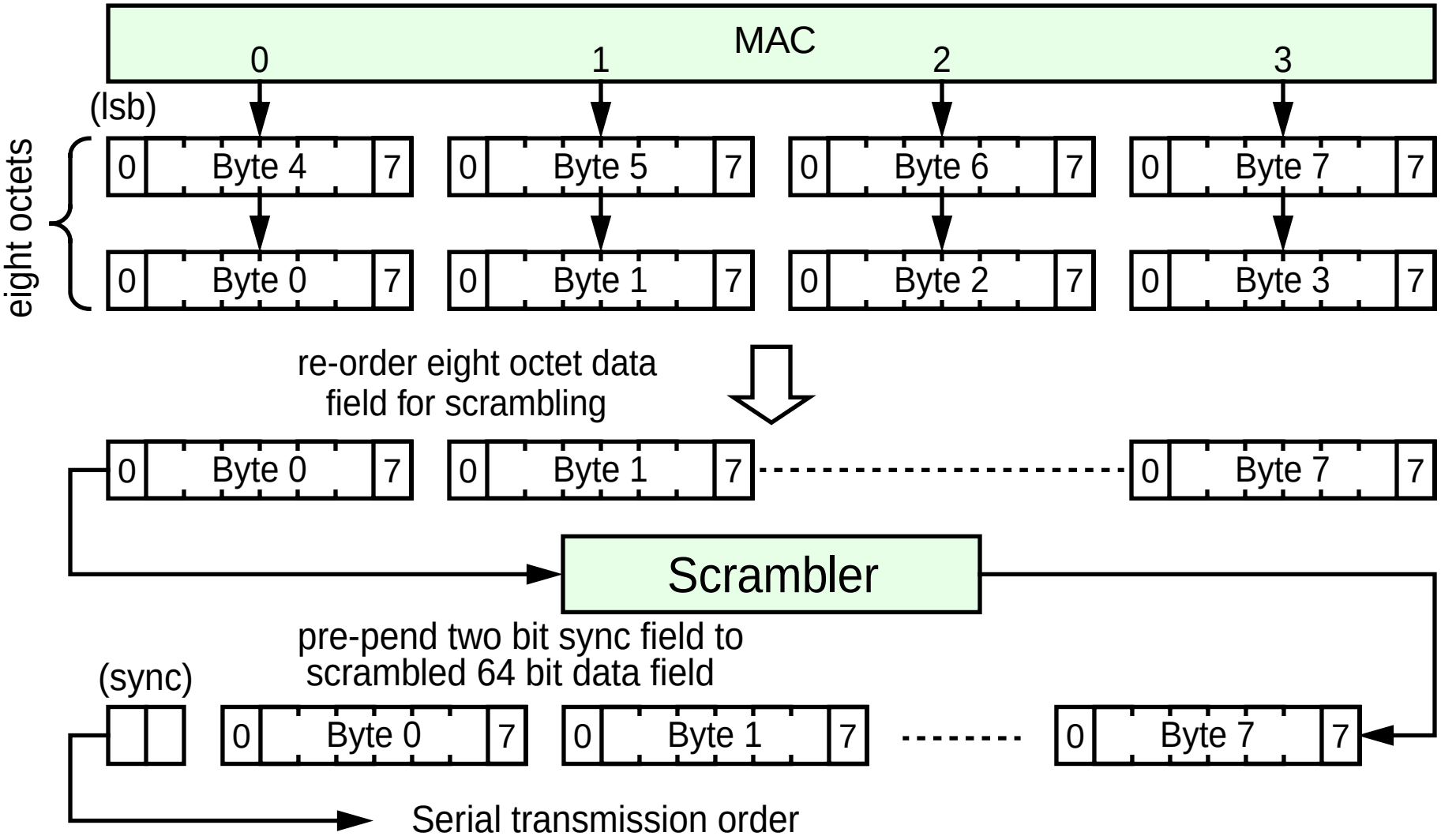
Input Data (first RS transfer / second RS transfer)	Sync [0]	Sync [1]	Bit fields [2]							[65]
$D_0D_1D_2D_3/D_4D_5D_6D_7$	0	1	D_0 [0] [7]	D_1 [0] [7]	D_2 [0] [7]	D_3 [0] [7]	D_4 [0] [7]	D_5 [0] [7]	D_6 [0] [7]	D_7 [0] [7]
$Z_0Z_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x1e "01111000"	C_0 [0] [6]	C_1 [0] [6]	C_2 [0] [6]	C_3 [0] [6]	C_4 [0] [6]	C_5 [0] [6]	C_6 [0] [6]	C_7 [0] [6]
$Z_0Z_1Z_2Z_3/S_4D_5D_6D_7$	1	0	0x33	C_0	C_1	C_2	C_3		D_5	D_6	D_7
$S_0D_1D_2D_3/D_4D_5D_6D_7$	1	0	0x78	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
$T_0Z_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x87		C_1	C_2	C_3	C_4	C_5	C_6	C_7
$D_0T_1Z_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0x99	D_0		C_2	C_3	C_4	C_5	C_6	C_7
$D_0D_1T_2Z_3/Z_4Z_5Z_6Z_7$	1	0	0xaa	D_0	D_1		C_3	C_4	C_5	C_6	C_7
$D_0D_1D_2T_3/Z_4Z_5Z_6Z_7$	1	0	0xb4	D_0	D_1	D_2		C_4	C_5	C_6	C_7
$D_0D_1D_2D_3/T_4Z_5Z_6Z_7$	1	0	0xcc	D_0	D_1	D_2	D_3		C_5	C_6	C_7
$D_0D_1D_2D_3/D_4T_5Z_6Z_7$	1	0	0xd2	D_0	D_1	D_2	D_3	D_4		C_6	C_7
$D_0D_1D_2D_3/D_4D_5T_6Z_7$	1	0	0xe1	D_0	D_1	D_2	D_3	D_4	D_5		C_7
$D_0D_1D_2D_3/D_4D_5D_6T_7$	1	0	0xff	D_0	D_1	D_2	D_3	D_4	D_5	D_6	

- all undefined bit fields (in yellow) are set to zero for 10GbE

RS “Z” code to 7 bit “C” field mapping

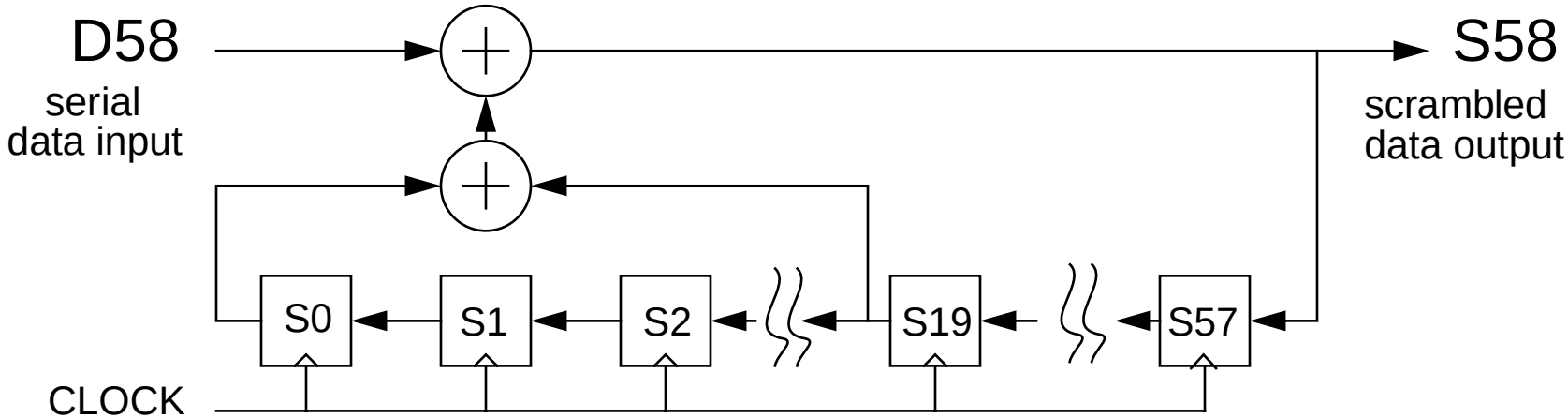
RS Z value	name	shorthand	7-bit C field line code
0x07,1	idle	[I]	0x00
0xfb,1	start	[S]	encoded by TYPE byte
0xfd,1	terminate	[T]	encoded by TYPE byte
0xfe,1	error	[E]	0x1e
0x1c,1	reserved0	-	0x2d
0x3c,1	reserved1	-	0x33
0x7c,1	reserved2	-	0x4b
0xbc,1	reserved3	-	0x55
0xdc,1	reserved4	-	0x66
0xf7,1	reserved5	-	0x78

Bit ordering sequence



Scrambler definition

Serial form of the Scrambler:



The serial form of the scrambler is shown here for bit ordering purposes. Parallel implementations could also be used. For details see:

http://grouper.ieee.org/groups/802/3/ae/public/mar00/walker_1_0300.pdf

Sample 64b/66b Test Vector

- Start with a minimum length (64 byte) Ethernet packet with preamble and CRC

```
55 55 55 55 55 55 d5 08 00 20 77 05 38 0e 8b 00 00 00 00 08 00 45 00 00 28 1c 66 00 00 1b 06 9e
d7 00 00 59 4d 00 00 68 d1 39 28 4a eb 00 00 30 77 00 00 7a 0c 50 12 1e d2 62 84 00 00 00 00 00
                                  00 00 00 93 eb f7 79
```

- Add SOP, EOP, Idles and convert to RS indications

```
07,1 07,1 07,1 07,1 07,1 07,1 07,1 07,1  fb,1 55,0 55,0 55,0 55,0 55,0 55,0 d5,0
08,0 00,0 20,0 77,0 05,0 38,0 0e,0 8b,0  00,0 00,0 00,0 00,0 08,0 00,0 45,0 00,0
00,0 28,0 1c,0 66,0 00,0 00,0 1b,0 06,0  9e,0 d7,0 00,0 00,0 59,0 4d,0 00,0 00,0
68,0 d1,0 39,0 28,0 4a,0 eb,0 00,0 00,0  30,0 77,0 00,0 00,0 7a,0 0c,0 50,0 12,0
1e,0 d2,0 62,0 84,0 00,0 00,0 00,0 00,0  00,0 00,0 00,0 00,0 93,0 eb,0 f7,0 79,0
                    fd,1 07,1 07,1 07,1 07,1 07,1 07,1 07,1
```

- Arrange bytes into frames with type indicators and sync bits

```
"10" 1e 00 00 00 00 00 00 00 "10" 78 55 55 55 55 55 55 d5 "01" 08 00 20 77 05 38 0e 8b
"01" 00 00 00 00 08 00 45 00 "01" 00 28 1c 66 00 00 1b 06 "01" 9e d7 00 00 59 4d 00 00
"01" 68 d1 39 28 4a eb 00 00 "01" 30 77 00 00 7a 0c 50 12 "01" 1e d2 62 84 00 00 00 00
             "01" 00 00 00 00 93 eb f7 79 "10" 87 00 00 00 00 00 00 00
```

- Scramble and transmit left-to-right, lsb first, (scrambler initial state is set to all ones)

```
"10" 1e 00 00 00 80 f0 ff 7b "10" 78 15 ad aa aa 16 30 62
"01" 08 e1 81 c5 6e 7c 76 6a "01" e6 30 28 80 cc aa f4 8d
"01" 83 ee 49 ae 6d 93 db 2c "01" f3 46 70 db 82 5a 90 74
"01" 1e 51 79 6b 1a 25 7a c5 "01" 41 1f bf d4 0c 44 ca 4a
"01" 09 28 12 d2 b5 2d 3f 2c "01" 49 92 de c8 b3 33 0e 32
             "10" 2a a3 3a c8 d7 ad 99 b5
```

Frame alignment algorithm

Look for presence of “01” or “10” sync patterns every 66 bits

This can be done either in parallel, by looking at all possible locations, or in serial by looking at only one potential location.

In either case, a frame sync detector is used to statistically qualify a valid sync alignment.

In the parallel case, a barrel shifter can immediately make the phase shift adjustment. In the serial case, a sync error is used to cycle-slip the demultiplexor to hunt for a valid sync phase.

So what algorithm should be used for reliable and rapid frame sync detection?

Frame sync criteria

If misaligned, then sync error rate will be 50%. We must quickly assert loss of sync and “slip” our alignment to another candidate location

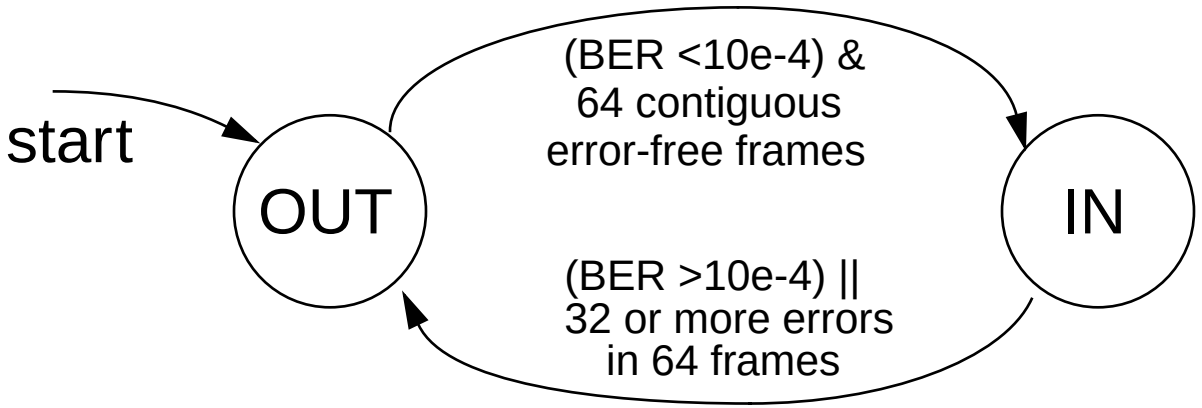
If already aligned with good BER (<10e-9), then we want to stay in sync with very high reliability

If BER is worse than10e-4 we should suppress sync, to avoid likelyhood of False Packet Acceptance due to CRC failures

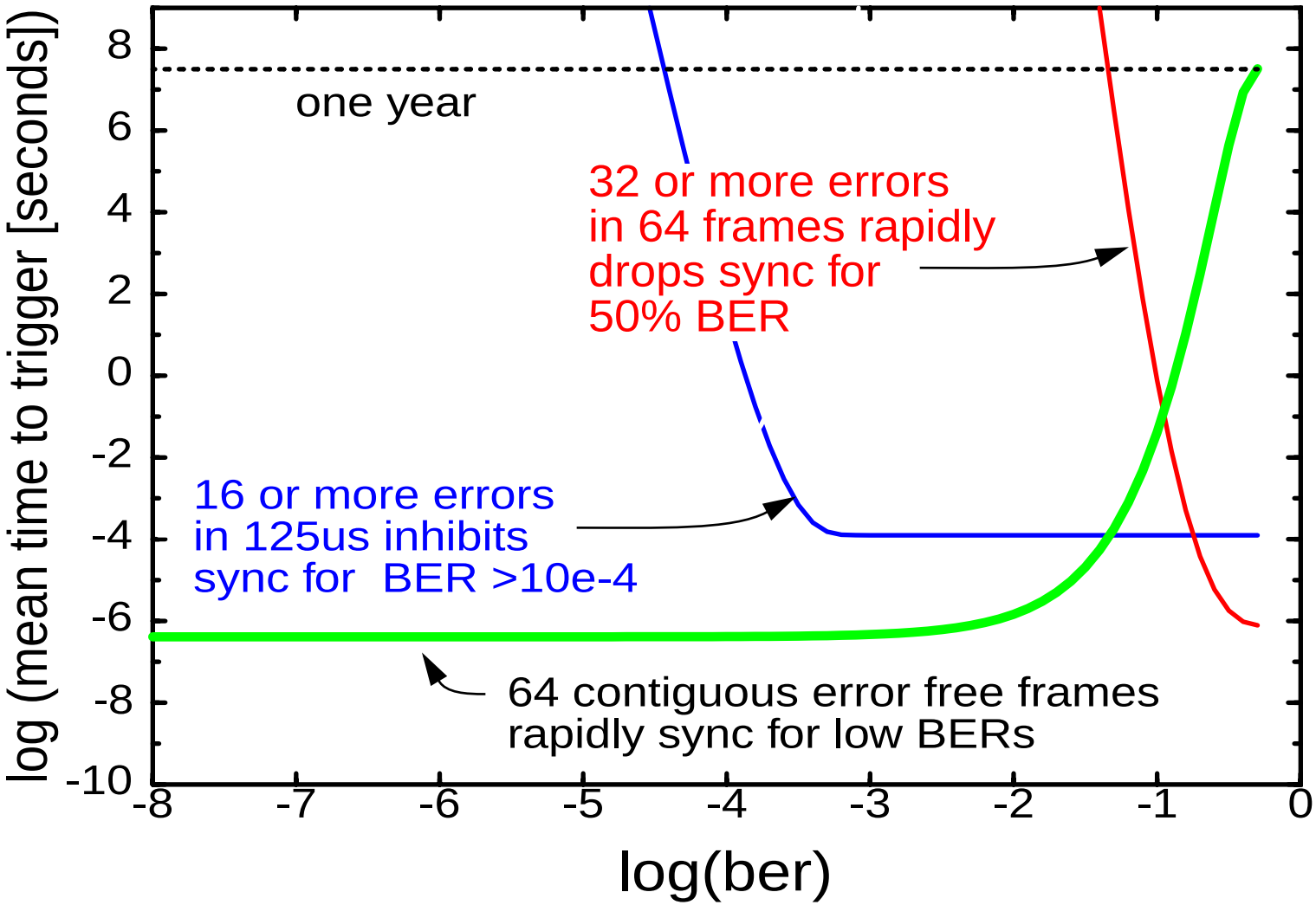
BER	current sync state	next sync state	notes
~50%	in	out	should be fast
>10e-4	in	out	prevents MTTFPA events, can be relatively slow to trigger
<10e-9	out	in	should be fast

Frame sync algorithm

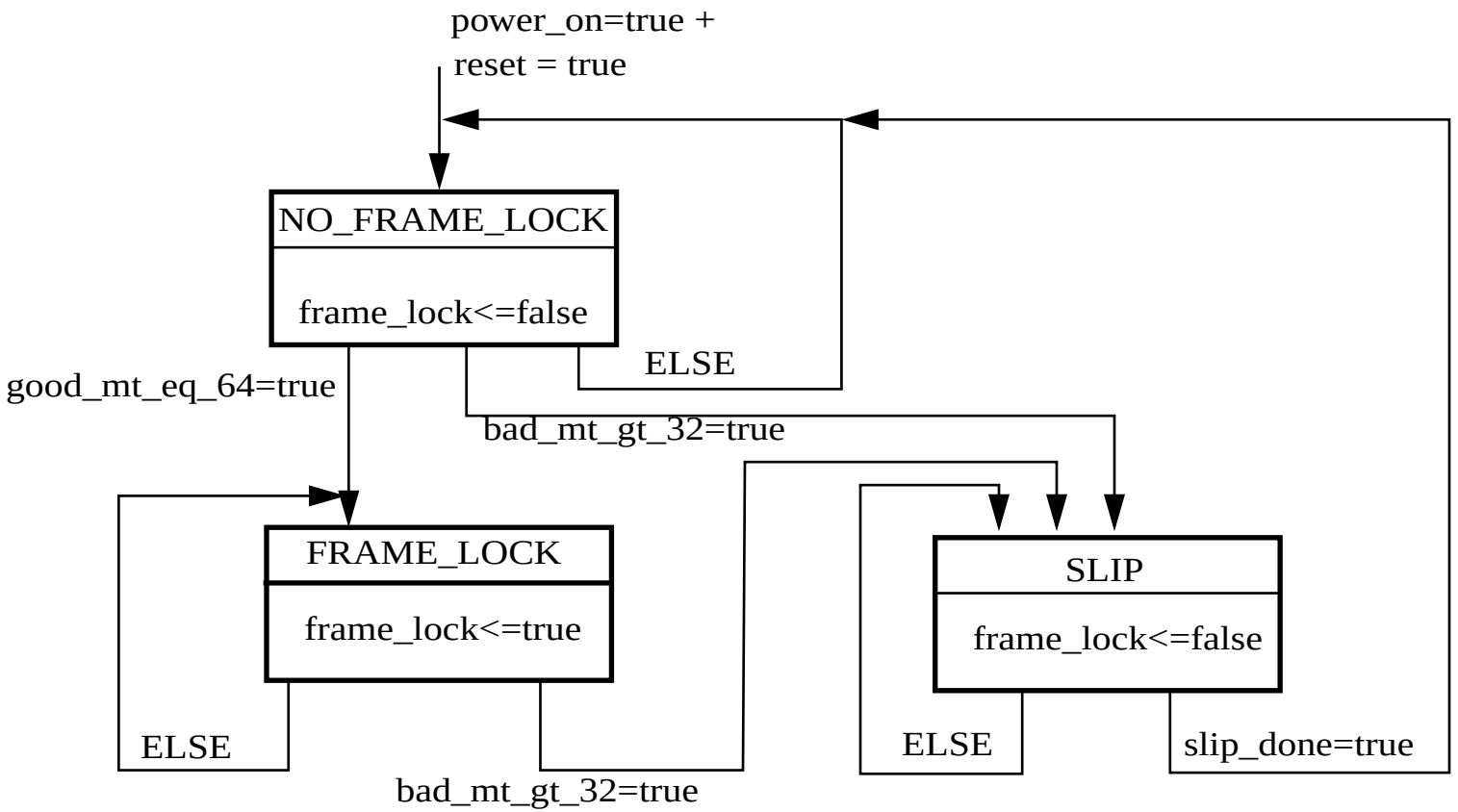
- frame sync is acquired after 64 contiguous frames have been received with valid “01” or “10” sync headers
- frame sync is declared lost after 32 “11” or “00” sync patterns have been declared in any block of 64 frames
- In addition, if there are 16 or more errors within any 125us time interval (~10e-4 BER), then frame sync is inhibited



64/66 frame sync performance

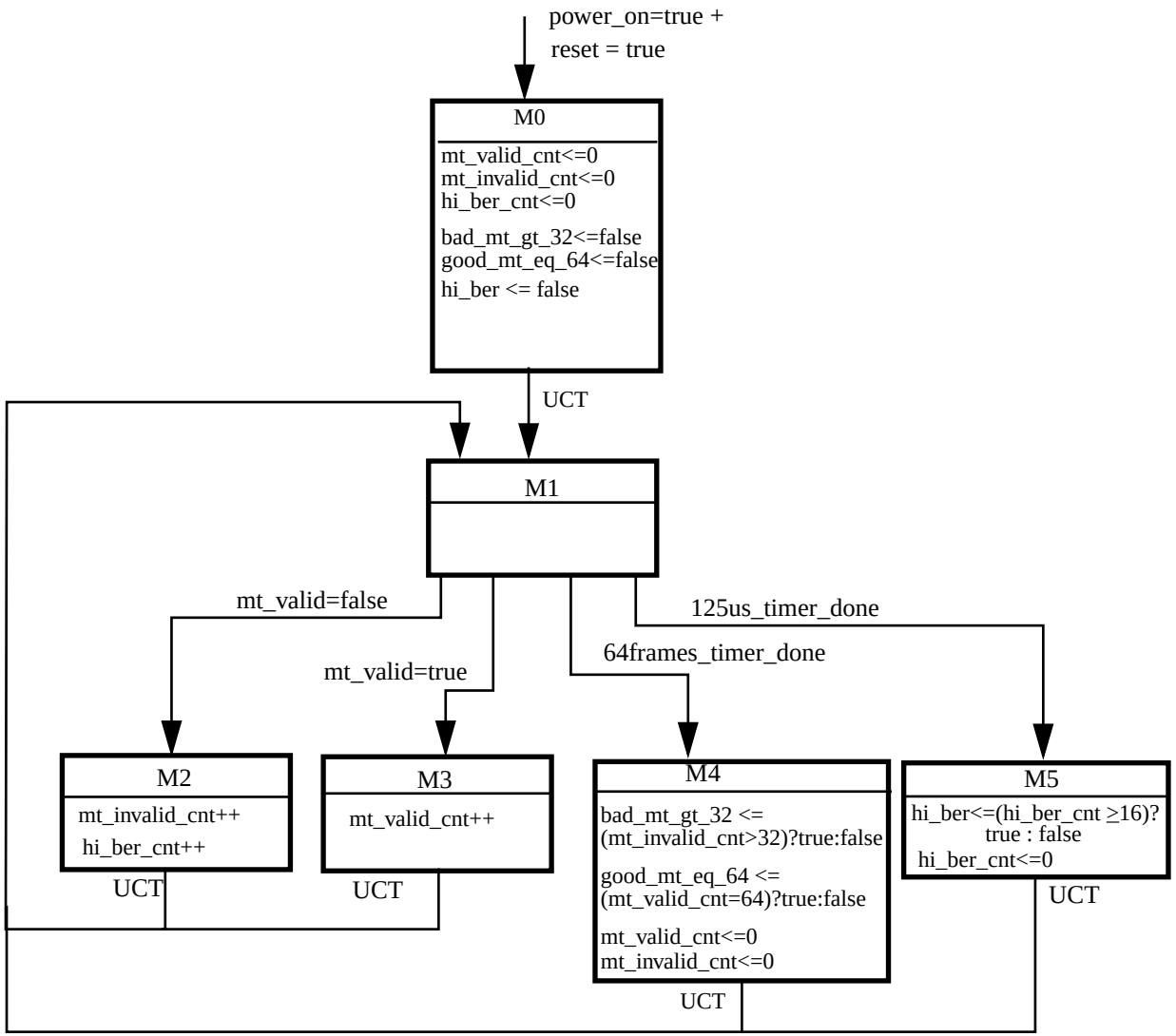


Frame lock process



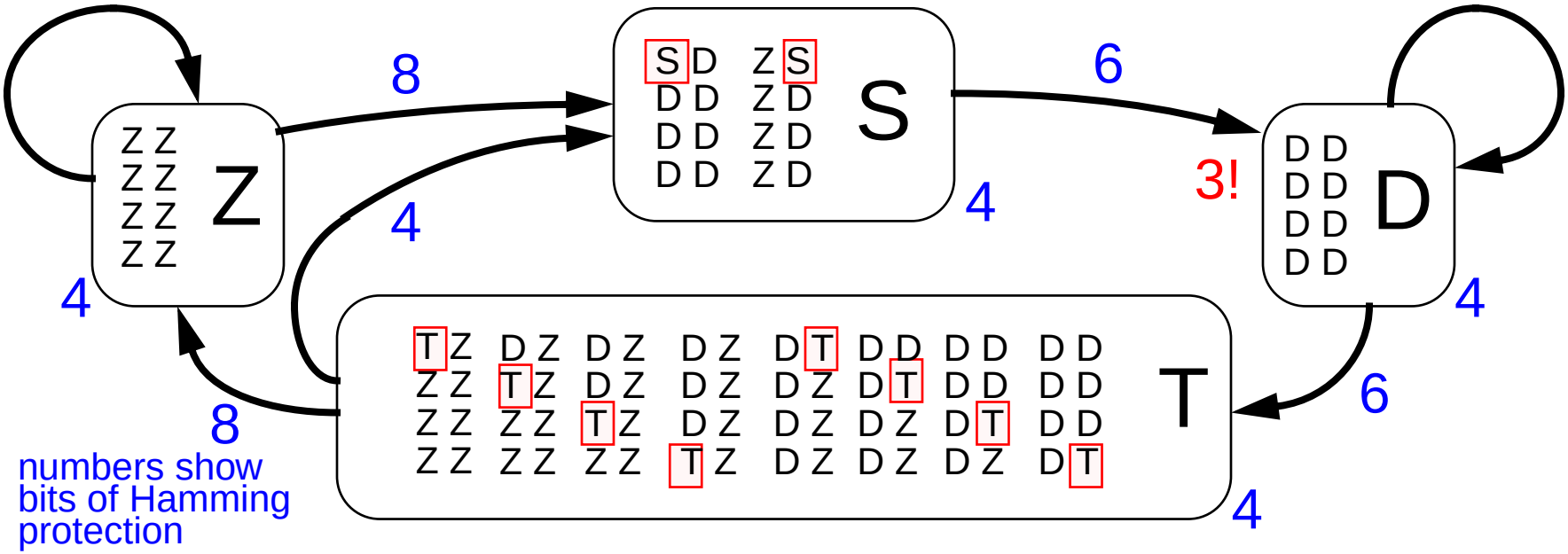
Receiver Synchronization condition
sync_done <= frame_lock=true * hi_ber=false

BER monitor process

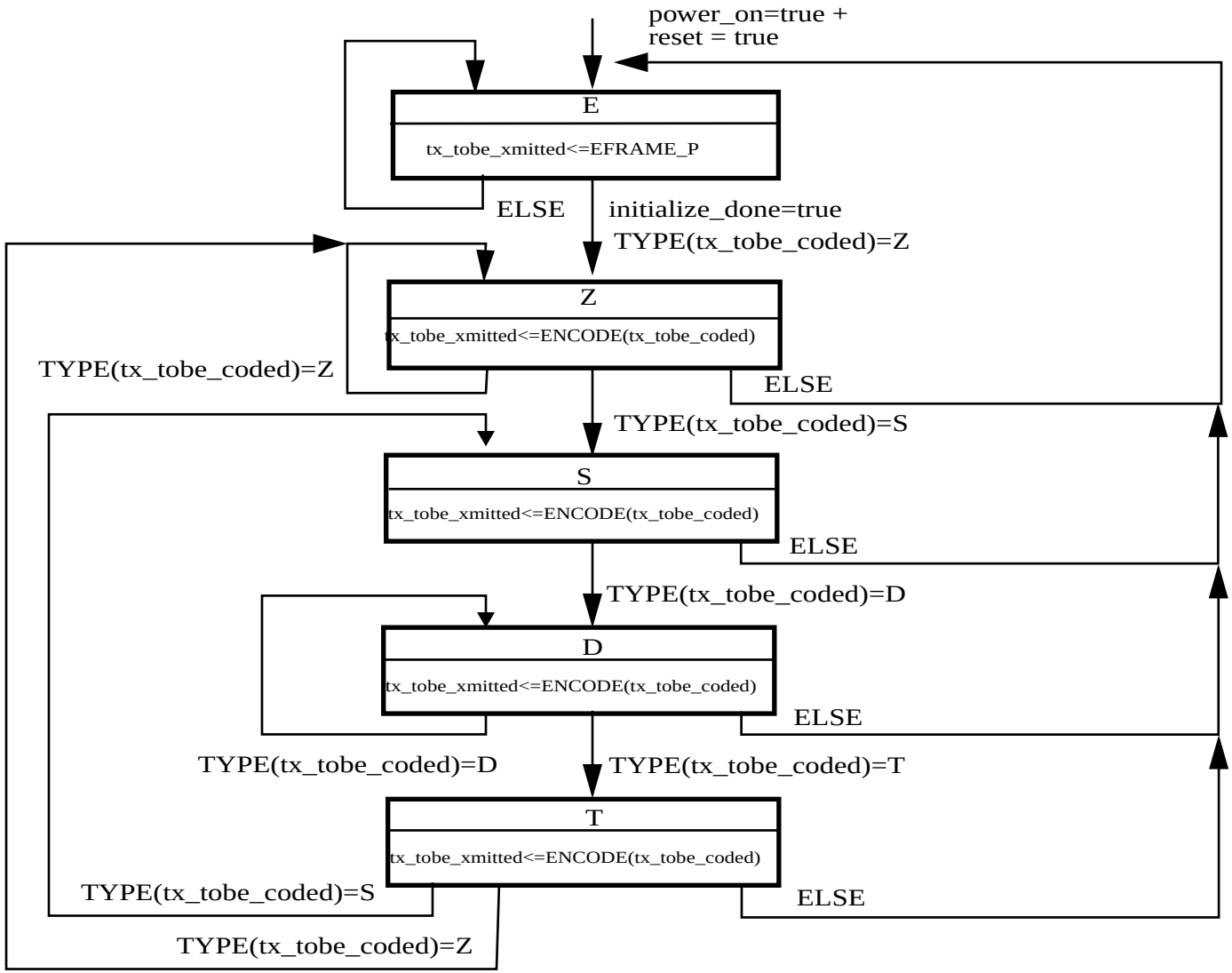


Packet boundary protection

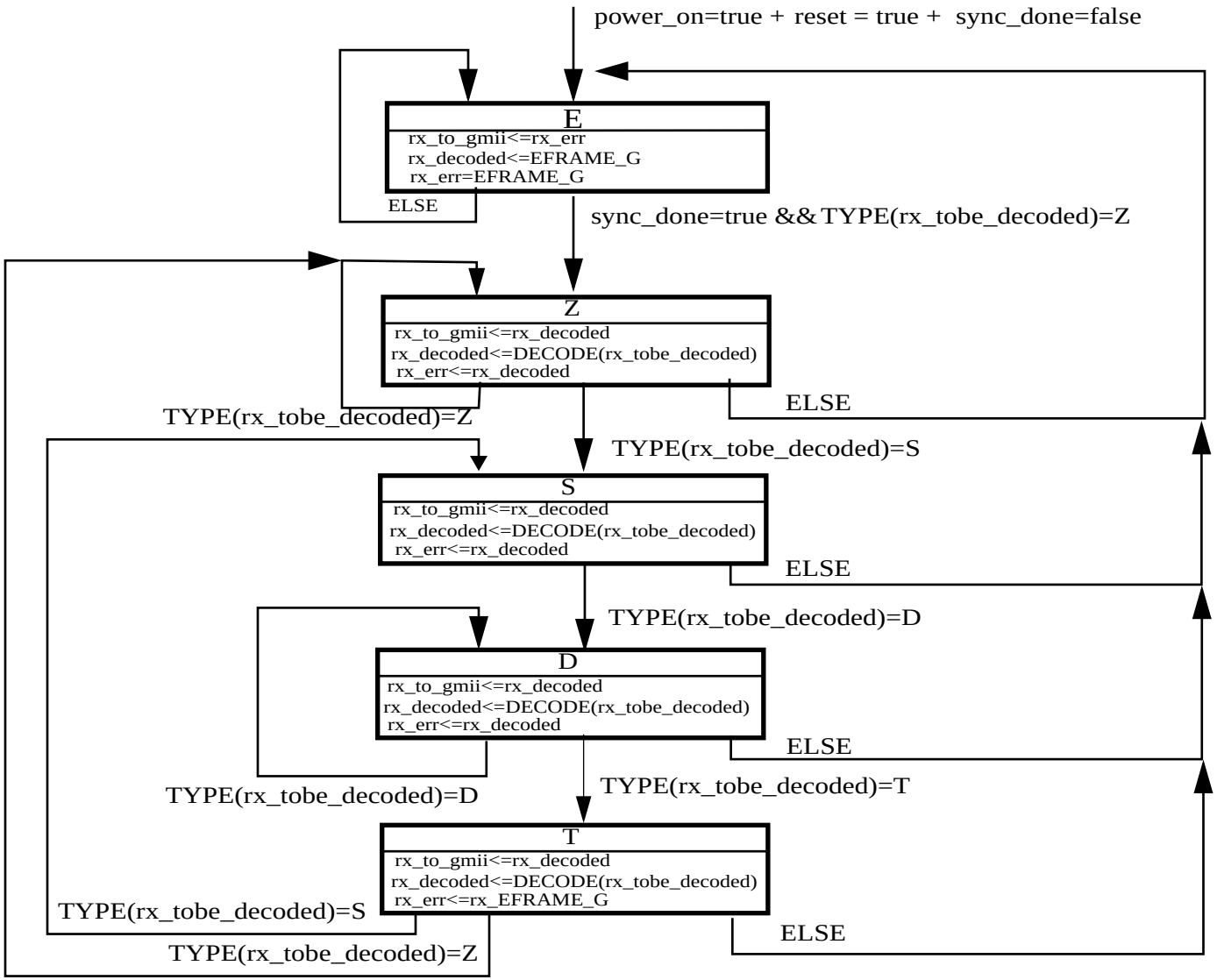
- A 2 bit error in the sync preamble can convert a packet boundary (S,T) into a Data frame (D) and vice-versa. However, all such errors violate frame sequencing rules unless another 4 errors recreate a false S,T packet (a total of six errors). Frame sequence errors invalidate the packet by forcing an (E) on the coder output.



TX process


power_on=true +
reset = true
E
tx_tobe_xmitted<=EFRAME_P
ELSE
initialize_done=true
TYPE(tx_tobe_coded)=Z
Z
tx_tobe_xmitted<=ENCODE(tx_tobe_coded)
TYPE(tx_tobe_coded)=Z
ELSE
TYPE(tx_tobe_coded)=S
S
tx_tobe_xmitted<=ENCODE(tx_tobe_coded)
ELSE
TYPE(tx_tobe_coded)=D
D
tx_tobe_xmitted<=ENCODE(tx_tobe_coded)
ELSE
TYPE(tx_tobe_coded)=D
TYPE(tx_tobe_coded)=T
T
tx_tobe_xmitted<=ENCODE(tx_tobe_coded)
TYPE(tx_tobe_coded)=S
ELSE
TYPE(tx_tobe_coded)=Z

RX process


power_on=true + reset = true + sync_done=false
E
rx_to_gmii<=rx_err
rx_decoded<=EFRAME_G
rx_err=EFRAME_G
ELSE
sync_done=true && TYPE(rx_tobe_decoded)=Z
Z
rx_to_gmii<=rx_decoded
rx_decoded<=DECODE(rx_tobe_decoded)
rx_err<=rx_decoded
TYPE(rx_tobe_decoded)=Z
ELSE
TYPE(rx_tobe_decoded)=S
S
rx_to_gmii<=rx_decoded
rx_decoded<=DECODE(rx_tobe_decoded)
rx_err<=rx_decoded
ELSE
TYPE(rx_tobe_decoded)=D
D
rx_to_gmii<=rx_decoded
rx_decoded<=DECODE(rx_tobe_decoded)
rx_err<=rx_decoded
ELSE
TYPE(rx_tobe_decoded)=D
TYPE(rx_tobe_decoded)=T
T
rx_to_gmii<=rx_decoded
rx_decoded<=DECODE(rx_tobe_decoded)
rx_err<=rx_EFRAME_G
TYPE(rx_tobe_decoded)=S
ELSE
TYPE(rx_tobe_decoded)=Z

Optional Code Features

- Special frames are reserved to support ordered sets for both Fiber Channel and 10GbE Link Signalling Sublayer (LSS)
- x,y ordered-set IDs are “1111” for FC and “0000” for 10GbE LSS

XGMII Pattern	Sync		Bit fields 0-63									
ZZZZ/ODDD	1	0	0x2d	Z0	Z1	Z2	Z3	y	D5	D6	D7	
ODDD/ZZZZ	1	0	0x4b	D1	D2	D3	x	Z4	Z5	Z6	Z7	
ODDD/ODDD	1	0	0x55	D1	D2	D3	x	y	D5	D6	D7	
ODDD/SDDD	1	0	0x66	D1	D2	D3	x	y	D5	D6	D7	
SDDD/DDDD	1	0	0x78	D1	D2	D3	D4		D5	D6	D7	
undefined	1	0	0x00	reserved for future expansion								

rs value	name	shorthand	7-bit line code
0x5c,1	FC ordered-set	[Of]	encoded by TYPE byte
0x9c,1	10 GbE Link Signalling	[LS]	encoded by TYPE byte

Summary

- We’ve shown a simple and reliable algorithm for 64b/66b frame sync detection
- Bit ordering has been clarified to be compatible with Ethernet CRC definition
- The TX and RX error control state machines have been presented
- A simple test vector has been produced to help to verify new implementations
- Optional 64b/66b extensions exist to support FC ordered sets and LS signalling

Supplementary slides

State machine notation conventions

Variables

TXD<35:0>	TXD signal of GMII
RXD<35:0>	RXD signal of GMII
tx_tobe_coded<71:0>	72 bit vector which is to be encoded by the PCS before transmission to the PMA.It is formed by concatenation of two consecutive TXD vectors. With the most recently received TXD word in the 35 : 0 bit locations.
tx_tobe_xmitted<65:0>	A 66 bit vector which is the result of a PCS ENCODE operation and is to be transmitted to the PMA.
rx_tobe_decoded<65:0>	A 66 bit vector containing the most recently received code word from the PMA.
rx_decoded<71:0>	72 bit vector which is the result of the PCS DECODE operation on the received bit vector, rx_tobe_decoded
rx_to_gmii<71:0>	72 bit vector which is a pipelined delayed copy of rx_decoded. This is sent to GMII in two steps of 36 bits each. Bits 71:36 are sent first to RXD, followed by bits 35:0.
rx_err<71:0>	This holds either a pipeline delayed copy of rx_decoded or the error frame EFRAME_G
state	Holds the current state of the transmit or the receive process.
sync_done	Boolean variable is set true when receiver is synchronized and set to false when receiver looses frame lock.
frame_lock	boolean variable is set true when receiver acquires frame delineation
mt_valid	boolean variable is set true if received frame rx_tobe_decoded has valid frame prefix bits. I.e, mt_valid = rx_tobe_decoded[65] ^ rx_tobe_decoded[64]
mt_valid_cnt	Holds the number of frames within a window of 64 frames, with valid prefix bits
mt_invalid_cnt	Holds the number of frames within a window of 64 frames with invalid prefix bits
good_mt_eq_64	Boolean variable is set true when there are 64 contiguous valid prefix bits
bad_mt_gt_32	Boolean variable is set true when there are at least 32 invalid prefix bits within a block of 64
hi_ber_cnt	Holds the number of with invalid prefix bits, within a 125us period
hi_ber	Boolean is asserted true when the hi_ber_cnt exceeds 16 indicating a bit error rate $>=10^{-4}$
slip_done	Boolean variable is set true when the hi_ber_cnt exceeds 16 indicating a bit error rate $>=10^{-4}$

State machine notation conventions

Constants

const enum FRAME_TYPE = { Z, S, T, D} ..... Each 72 bit vector, tx_tobe_coded and the 66 bit vector, rx_tobe_decoded, can be classified to belong to one of the four types depending on its contents. The frame types Z,S, T, D are defined in TBD.

EFRAME_G<71:0> .....72 bit vector to be sent to the GMII interface and represents a error octet in all the eight octet locations

EFRAME_P<65:0> .....66 bit vector to be sent to the PMA and represents a error octet in all the eight octet locations,

Functions

ENCODE(tx_tobe_coded<71:0>) ..... Encodes the 72 bit vector into a 66 bit vector to be transmitted to the PMA

DECODE(rx_tobe_decoded<65:0>) ..... Decodes the 66 bit vector into a 72 bit vector to be sent to the GMII

TYPE(tx_tobe_coded<71:0>)

TYPE(rx_tobe_decoded<65:0>) ..... Decodes the FRAME_TYPE of the tx_tobe_coded<71:0> bit vector or the rx_tobe_decoded<65:0>

Timers

64frames_timer_done .....Timer which is triggered once every 64 of the 66-bit frames in the receive process

125us_timer_done .....Timer which is triggered once every 125us (is approximately 2^{14} 66-bit frames in the receive process).