64B/66B overview
10 Gb/s Phy for EPON SG

Pat Thaler

July 2006

BROADCOM.
Connecting everything®

Introduction

- Adapted from Rick Walker's slide deck:
  - Walker_1_0700
- This is a high level overview – for details use IEEE 802.3 Clause 49.

BROADCOM
Connecting everything®

Title slide from walker_1_0700

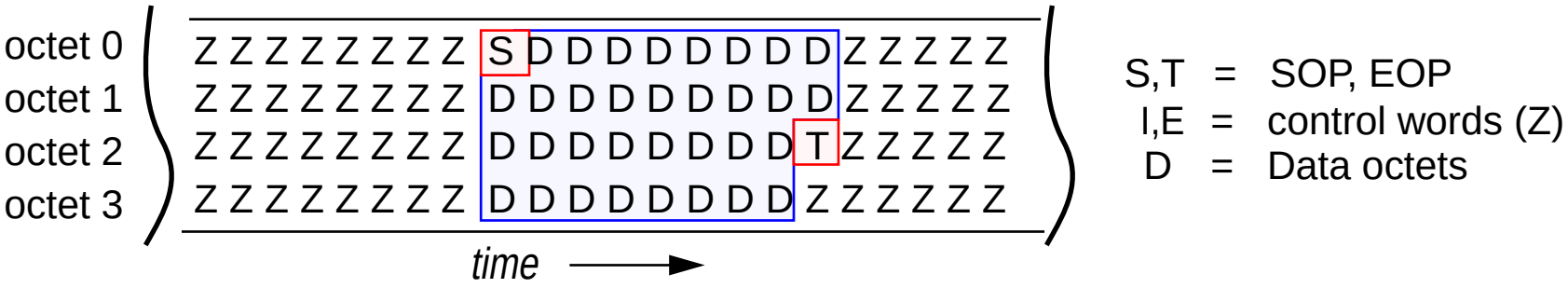
64b/66b PCS

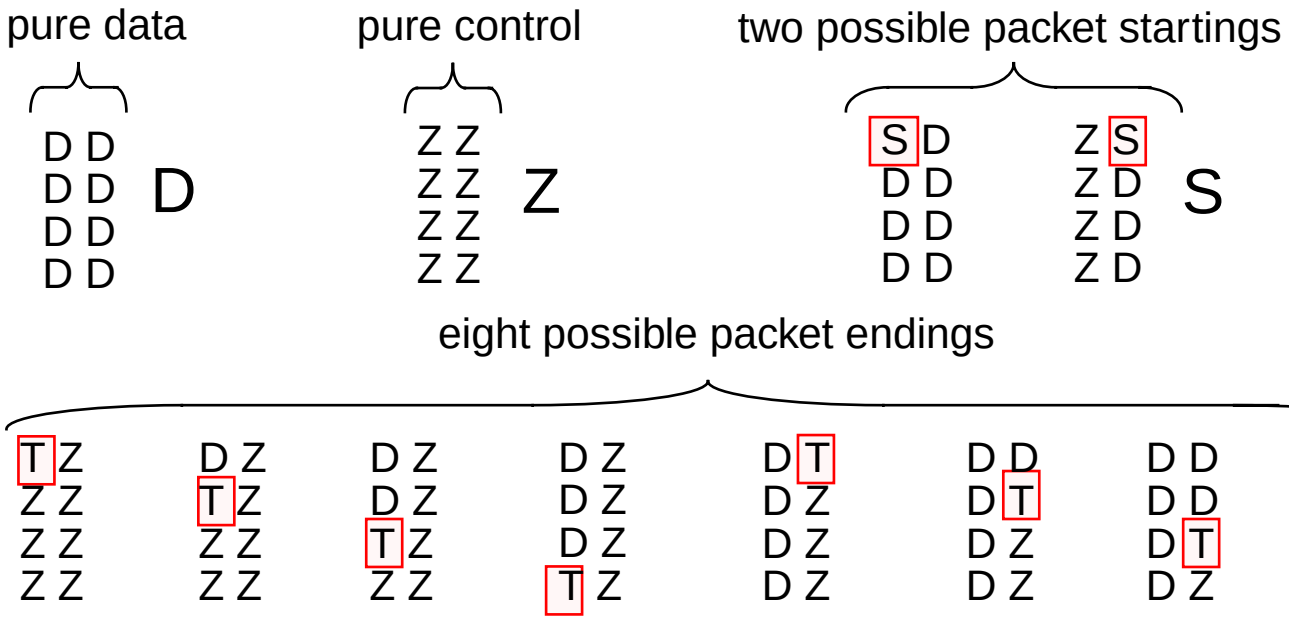
updated 6/30/2000

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

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

Ordered Set Support

- 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							

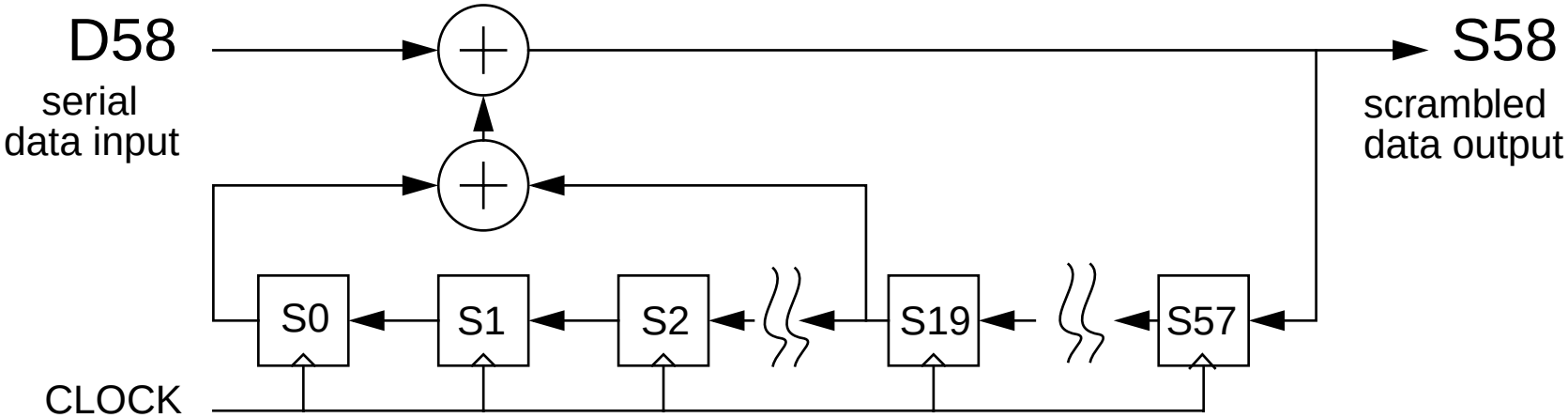
Signal ordered set - not deleted for clock skew

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

Sequence ordered set - may be deleted for clock skew if two consecutive sets occur

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

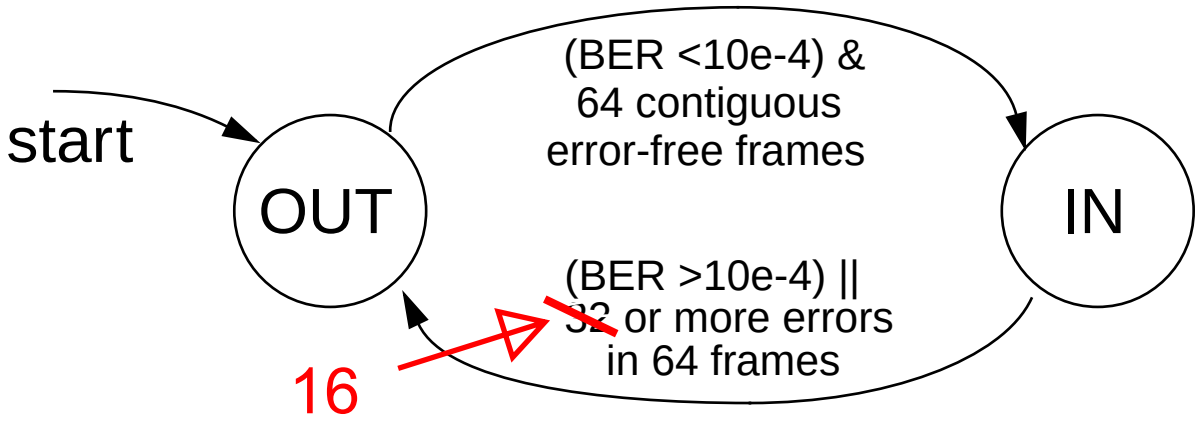
If already aligned with good BER (<10e-9), then we want to stay in sync with very high reliability

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

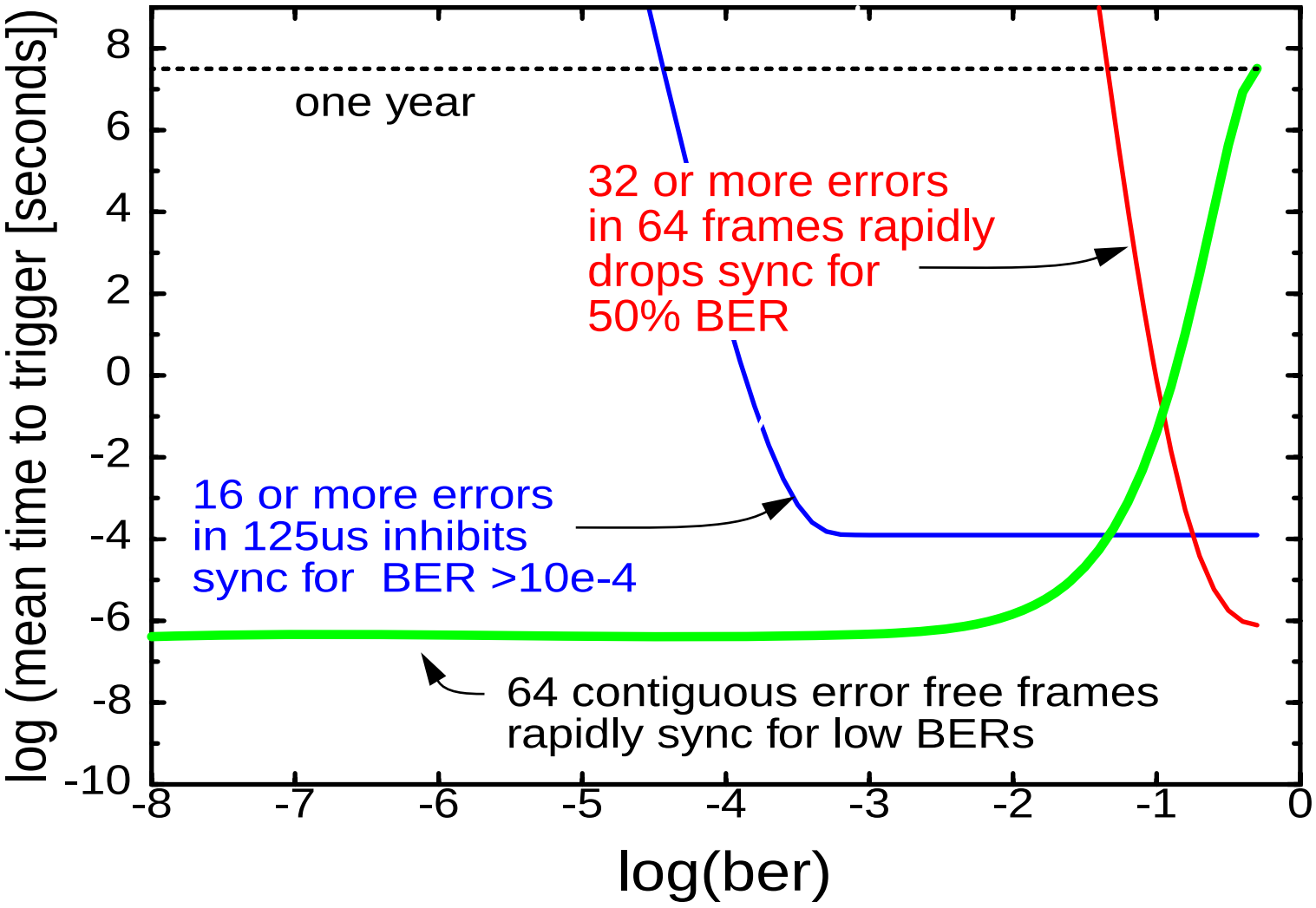
BER	current sync state	next sync state	notes
~50%	in	out	should be fast
>10e-4	in	out	prevents MTTFPA events, can be relatively slow to trigger
<10e-9	out	in	should be fast

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~~ 16 “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

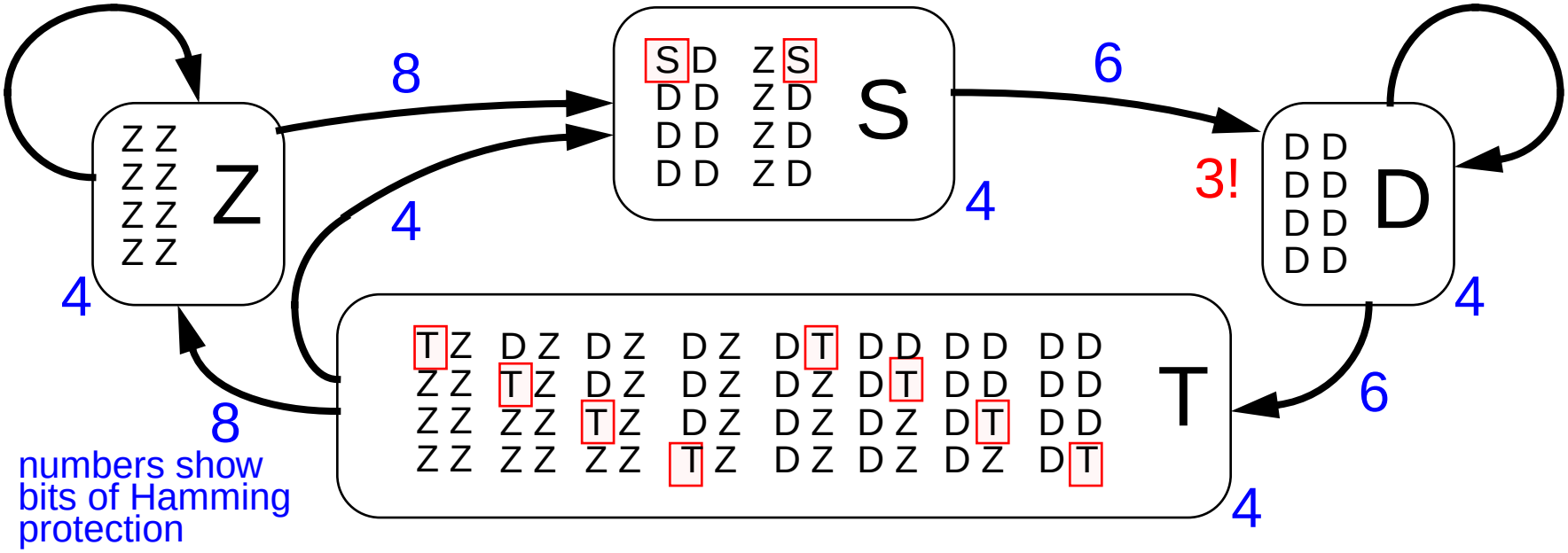


64/66 frame sync performance



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.



Receive State Machine

- Shows details of packet boundary protection.

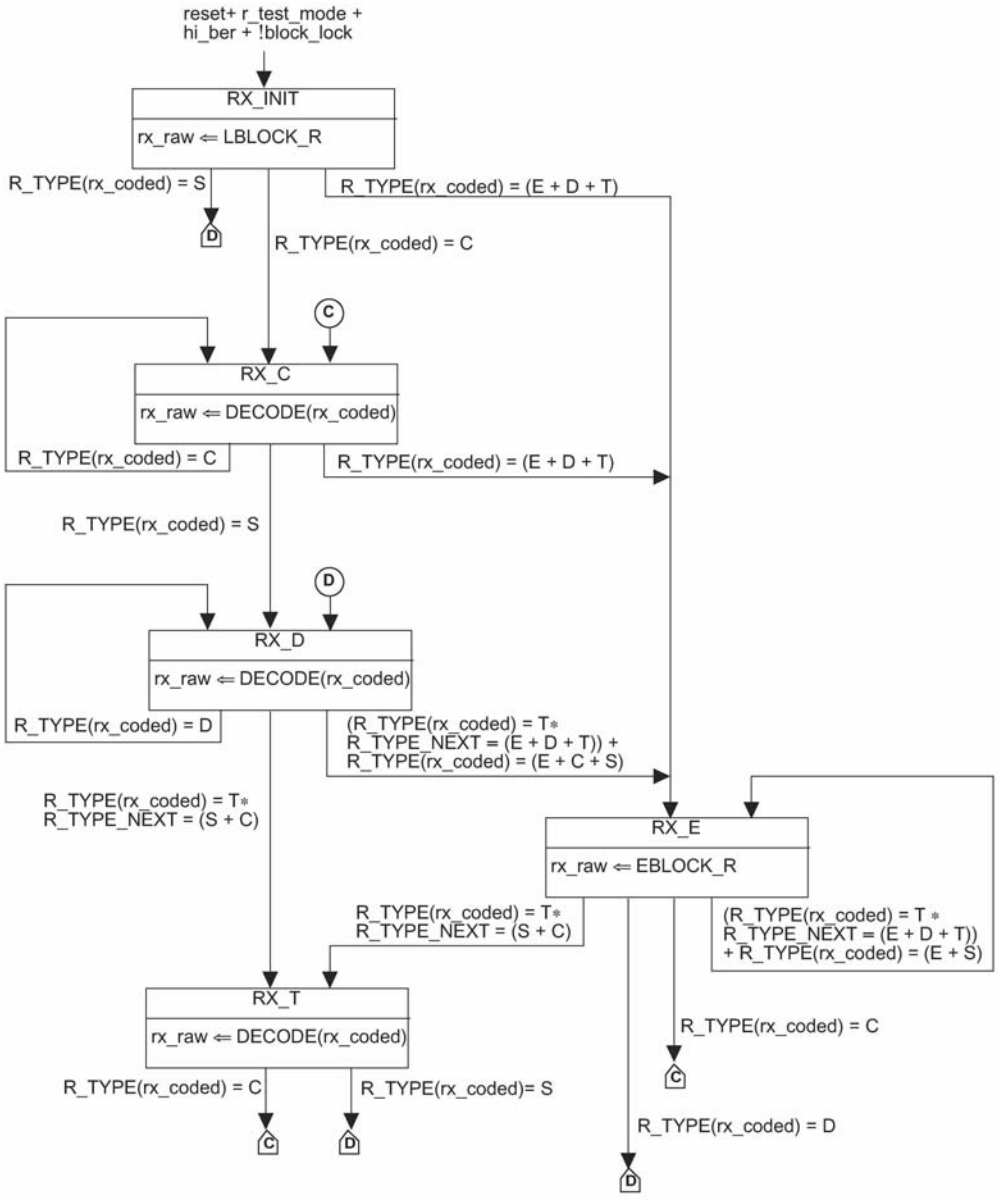


Figure 49–15—Receive state machine

everything®