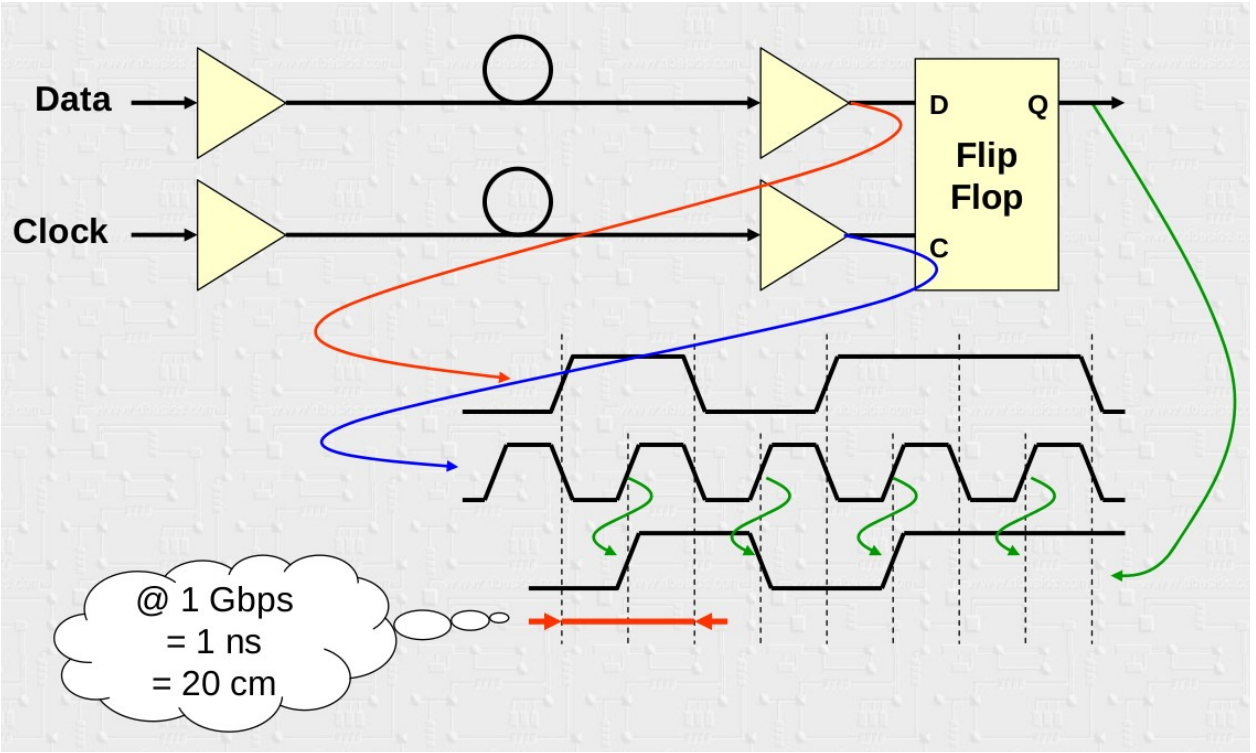
Understanding the Aurora protocol

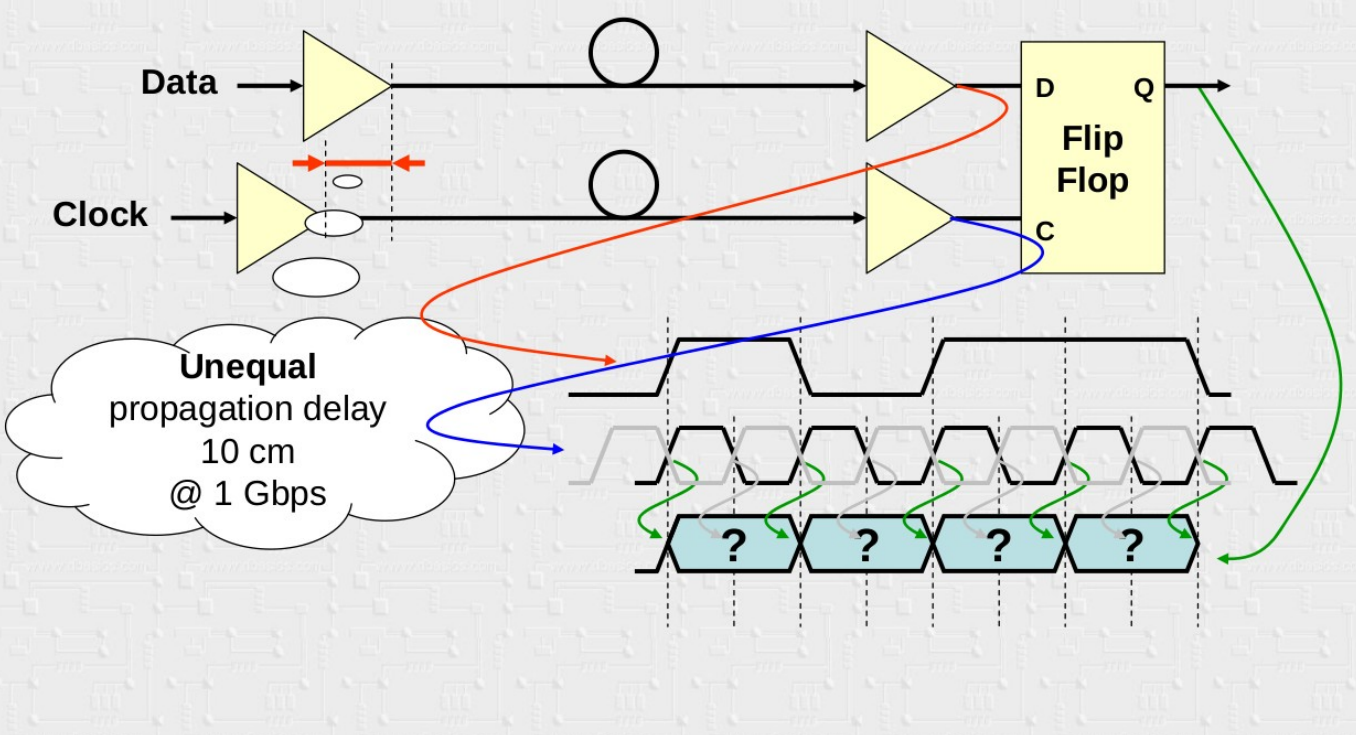
E.J. Schioppa
Lecce, April 2020

Source: Xilinx, Aurora 64B/66B Protocol Specification, SP011 (v1.3) October 1, 2014

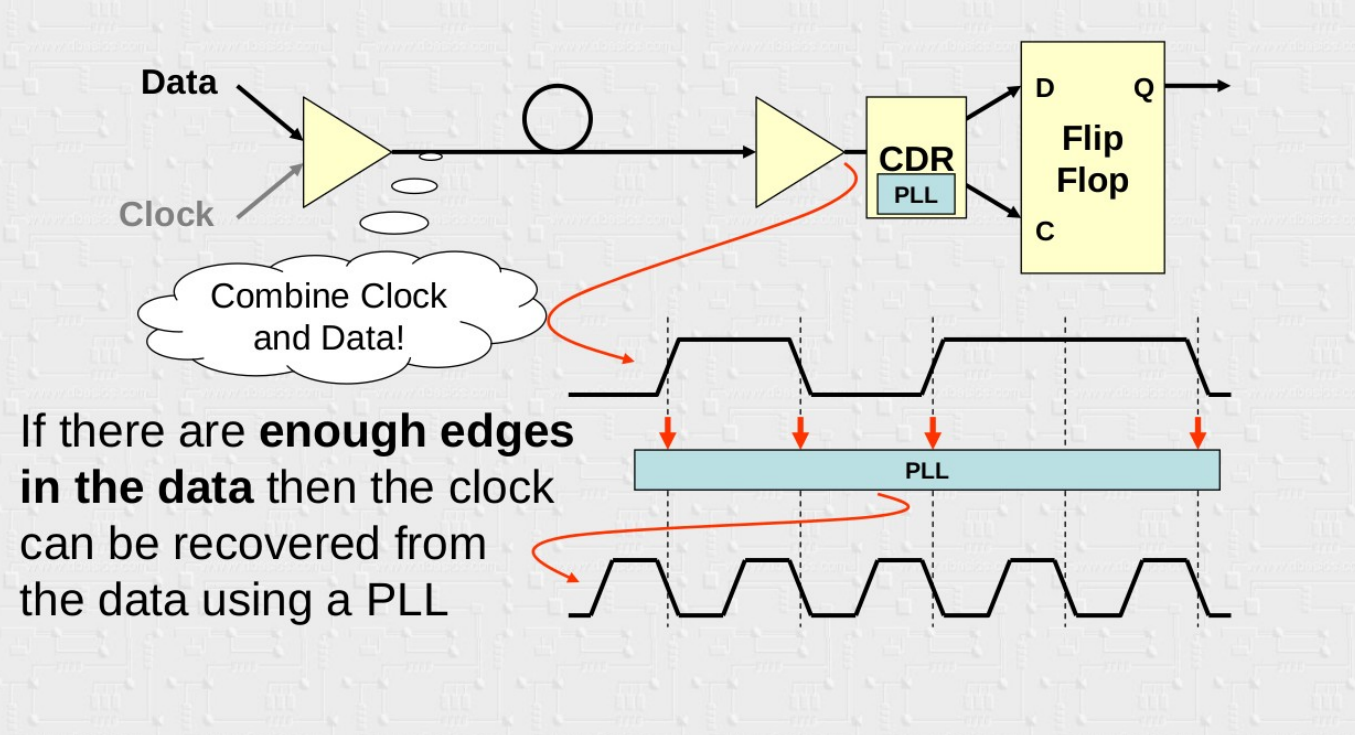
Data transmission



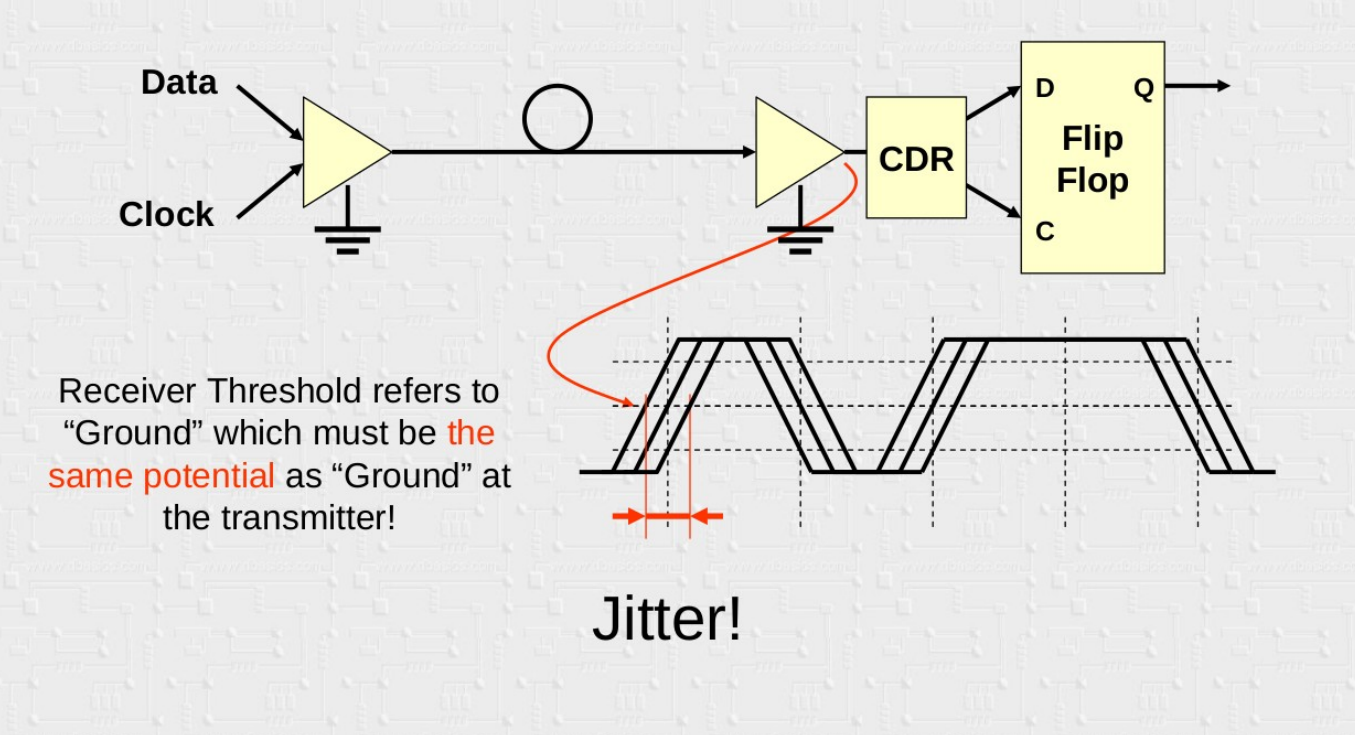
Data transmission



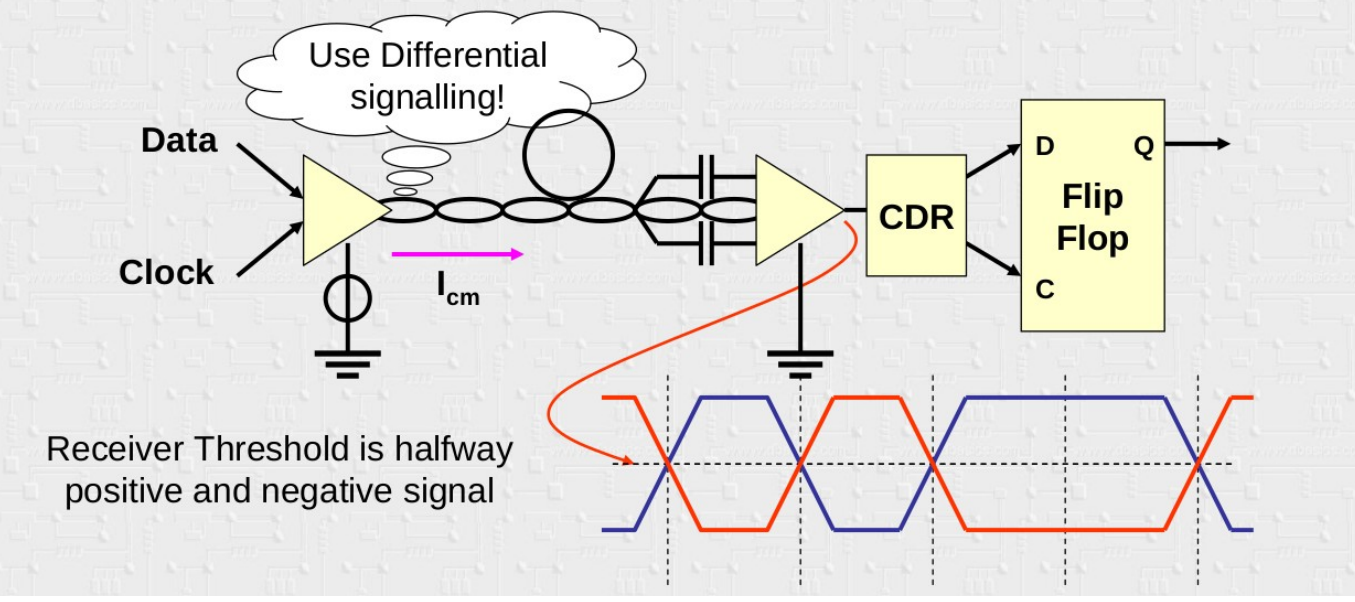
Data transmission



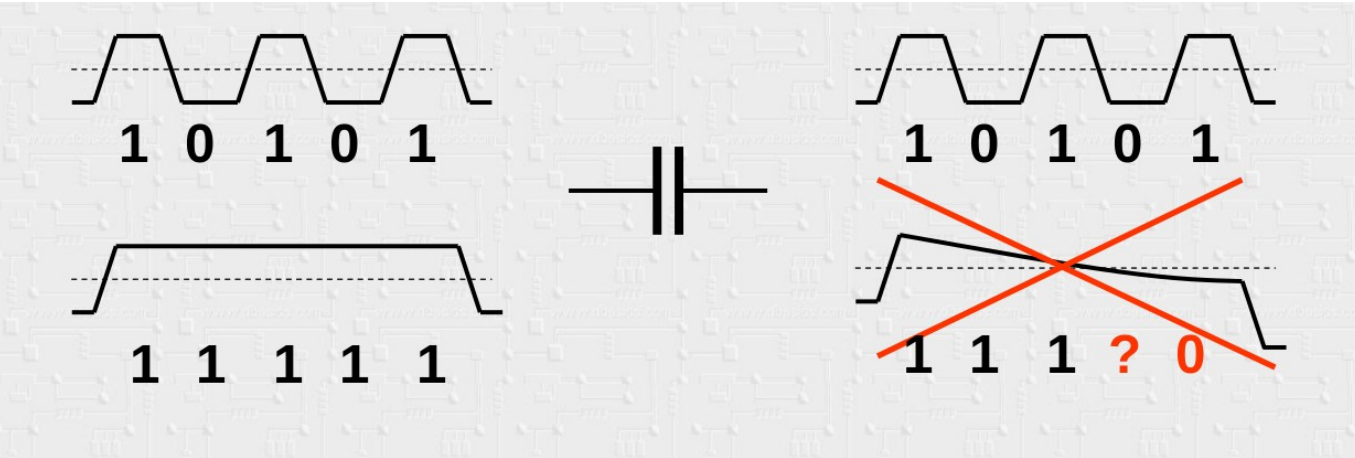
Data transmission



Data transmission



Data transmission



Running Disparity (#1-#0)
+
Maximum run length
=
DC balance

→ *8B/10B encoding*

8B/10B encoding

8 bit = 256 values | **10 bit = 1024 values**

Table C-1: **Valid Data Characters**

Data Byte Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
D0.0	000 00000	100111 0100	011000 1011
D1.0	000 00001	011101 0100	100010 1011
D2.0	000 00010	101101 0100	010010 1011
D3.0	000 00011	110001 1011	110001 0100
D4.0	000 00100	110101 0100	001010 1011
D5.0	000 00101	101001 1011	101001 0100
D6.0	000 00110	011001 1011	011001 0100
D7.0	000 00111	111000 1011	000111 0100
D8.0	000 01000	111001 0100	000110 1011
D9.0	000 01001	100101 1011	100101 0100
⋮	⋮	⋮	⋮
D31.7	111 11111	101011 0001	010100 1110

Most of the 8 bit values are mapped onto two 10 bit values: RD+ and RD-

- This operation is called **scrambling**
- Max. run length = 5
- Uses 512 out of 1024 available values
  → left over values can be assigned special functions
- (In DX.Y, D stands for Data)

8B/10B encoding

The 12 special K characters

Comma Characters

"The only patterns that have 5 consecutive '1's or '0's

Table C-2: **Valid Control K Characters**

Special Code Name	Bits HGF EDCBA	Current RD – abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7(1)	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

Common "error propagate"

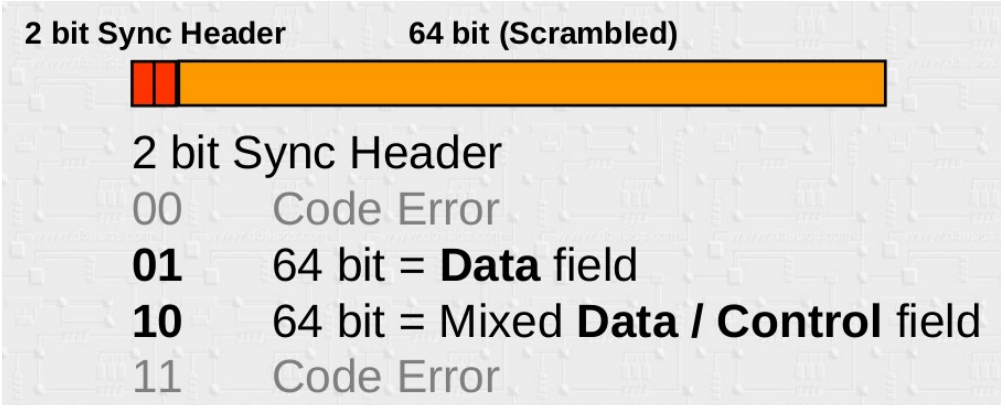
Comma characters are used for alignment. Example:

0101010011011001	1100000101	0110111010	0101001001	1101001100	1001110100	111001
??.?	K28.5	D16.2	D31.3	D11.3	D0.0	

64B/66B encoding

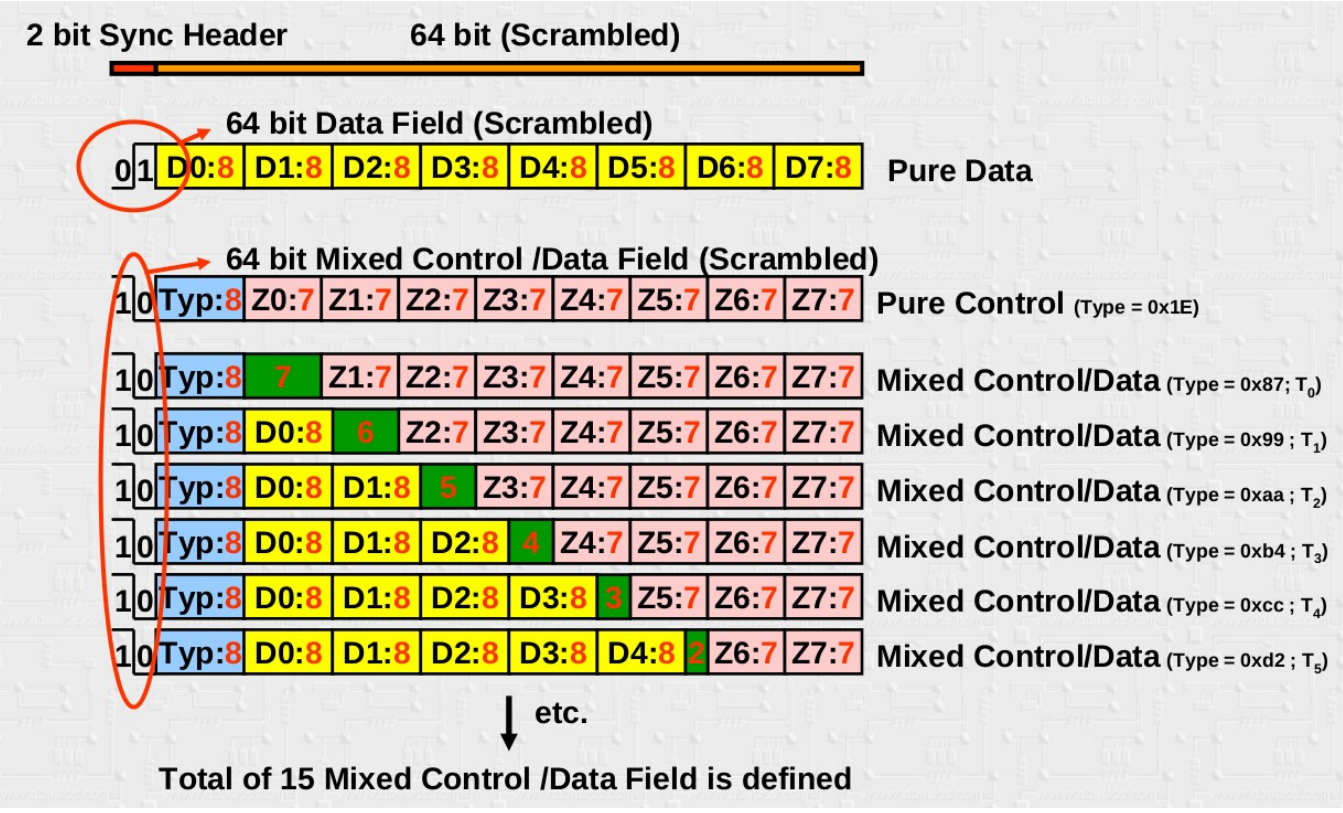
- 8B/10B overhead = 2 out of 10 → 20%
- 64B/66B overhead = 2 out of 66 → 3%

- **Sync Header**: first two bits of the 66 stream must be 01 or 01



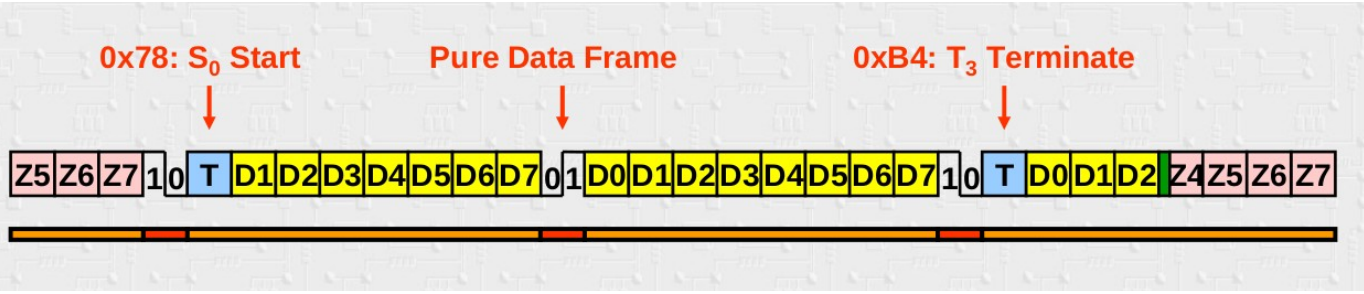
Notice: in fact, this is a 64B → 64B map!
DC balance can only be guaranteed statistically (see later)

64B/66B encoding



64B/66B encoding

Example: transmitting 18 bytes of data:



To the **gearbox**: chopping 66B words into octets / hexplets

64B/66B encoding

Compare:

- 8B/10B (pure N → N+2 map)
- 64B/66B (actual N → N map)

	8B/10B	64B/66B
Run Length	5	Relies on Scrambler
DC Balance	Excellent	Not guaranteed Demanding for receiver
Bit Synchronization Clock Recovery	Excellent	Relies on Scrambler, but al least one transition per 66 bits
Word Synchronization	"Comma" K-Characters	Sync-Header
Control Characters	K-Characters	Control-Codes

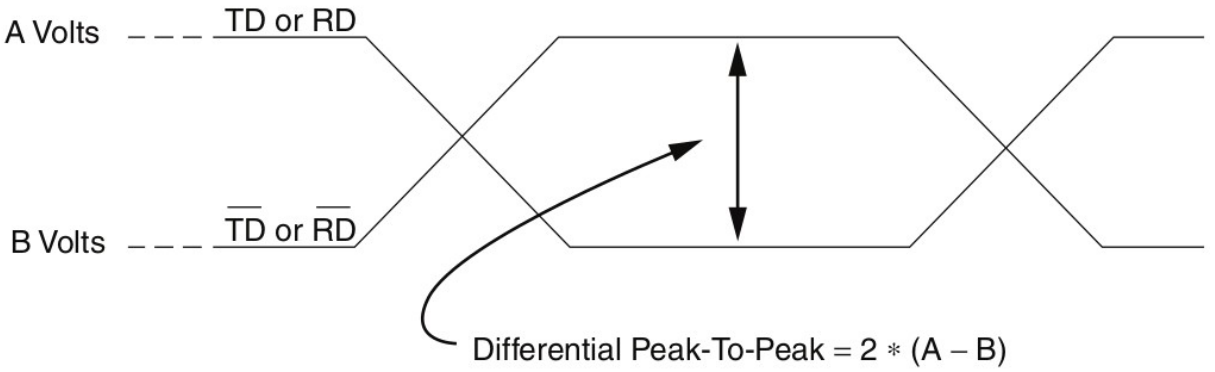
- Most clock recovery circuits can cope with run lengths up to 80 bit → guaranteed
- Probability of run length 65 (from random patterns) is

$$\binom{66}{65}\frac{1}{2^{64}}$$

  at 10Gbps → 1900 years
- Scramblers are designed to mimic random patterns
- C=1nF, R=100Ohm → DC shift more than 2.5% every 10^{22} bits (31700 years @ 10Gbps)

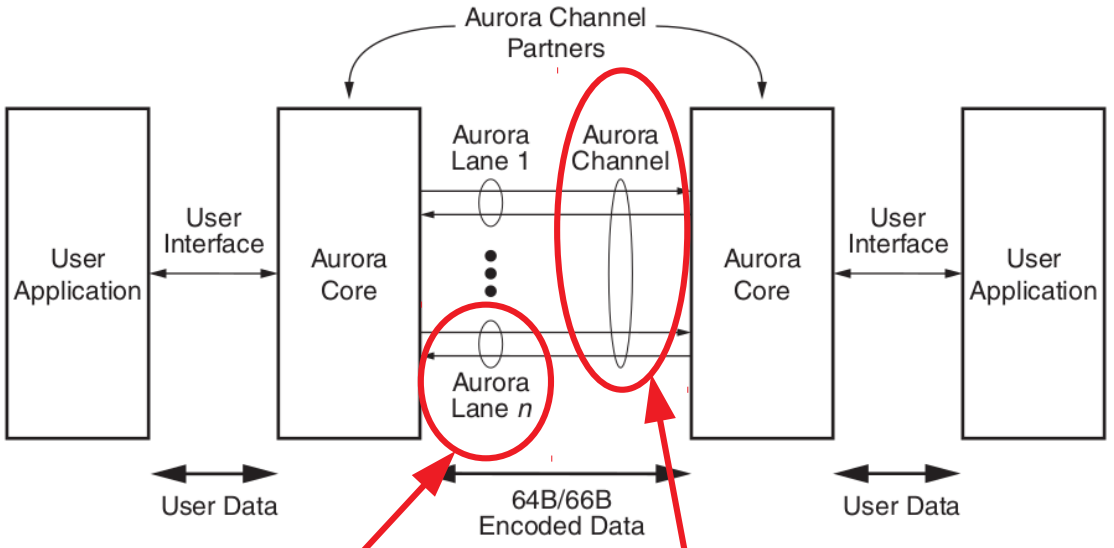
Electrical protocol

Non-return to zero (NRZ)



AC coupling at receiver

Overview

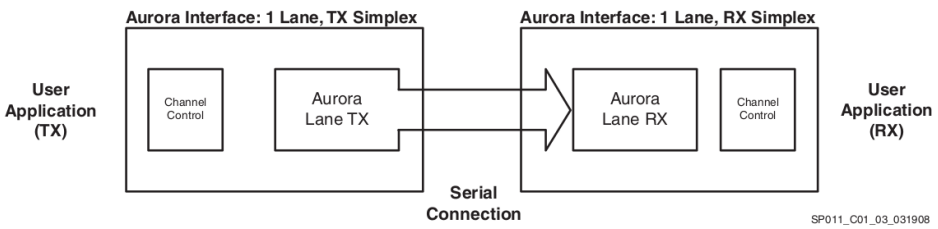


Serial lane (simplex or full duplex)

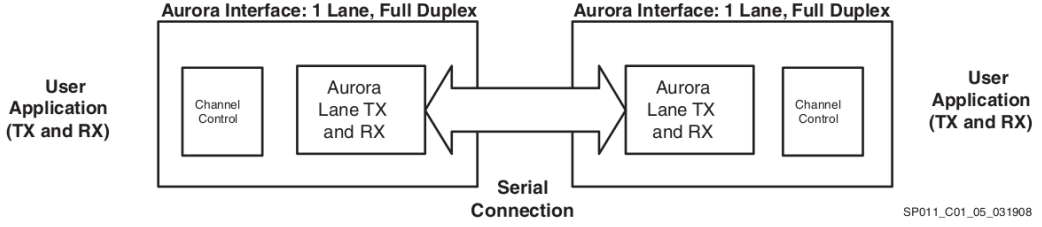
Same channel for data frames and control info (control messages, clock compensation, idles)

Configuration of Aurora interfaces

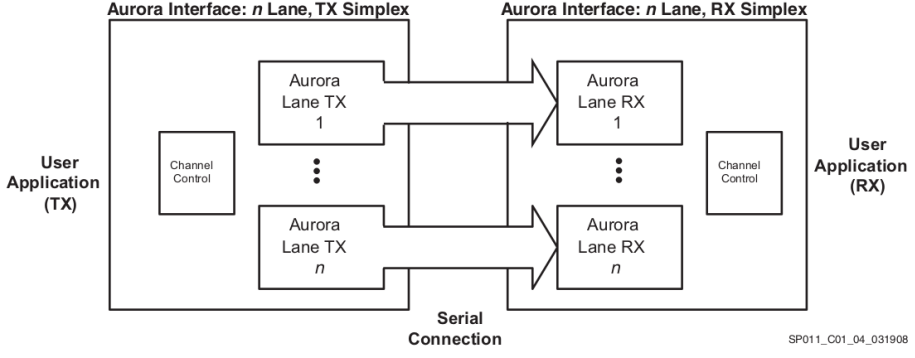
Single lane simplex



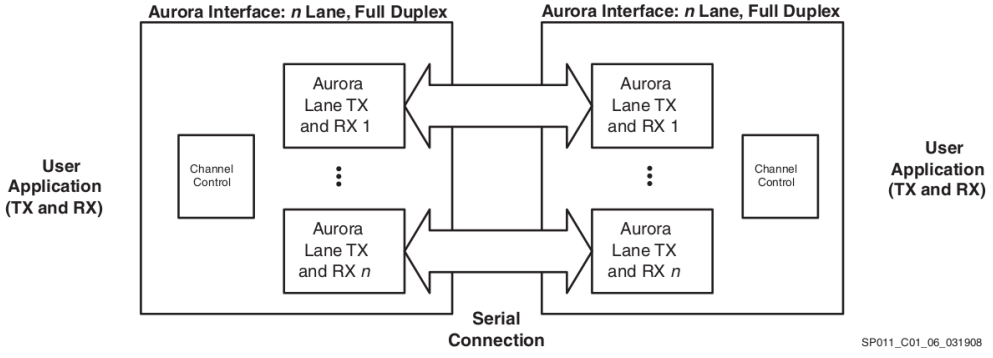
Single lane full duplex



Multi lane simplex



Multi lane full duplex



Frames and blocks

Frames

- Data format
- Any length, any format
- Only frame boundary is specified by the protocol
- Frames can be interrupted by control messages or idles
- No gap is needed between frames

Blocks = 64-bit words

- Clock compensation → for asynchronous channels (at least every 10k blocks)
- Not ready → for full duplex channels
- Channel bonding → for multi lane, to correct for inter-lanes delays
- Native flow control → see later
- User flow control → see later
- User K-blocks → 9 blocks outside the 64B/66B decoding, for user controls
- Data → 8 octets of data
- Separator → end of current frame (0 to 6 octects, see later)
- Separator-7 → separator with 7 valid octets
- Idle → sent when nothing else is sent

Block priority

Normal operation

Block Name	Priority
Clock Compensation	1 (Highest)
Not Ready	2
Channel Bonding	3
Native Flow Control	4
User Flow Control, Data blocks carrying UFC message	5
User K-Blocks	6
Data, Separator, Separator-7	7
Idle	8

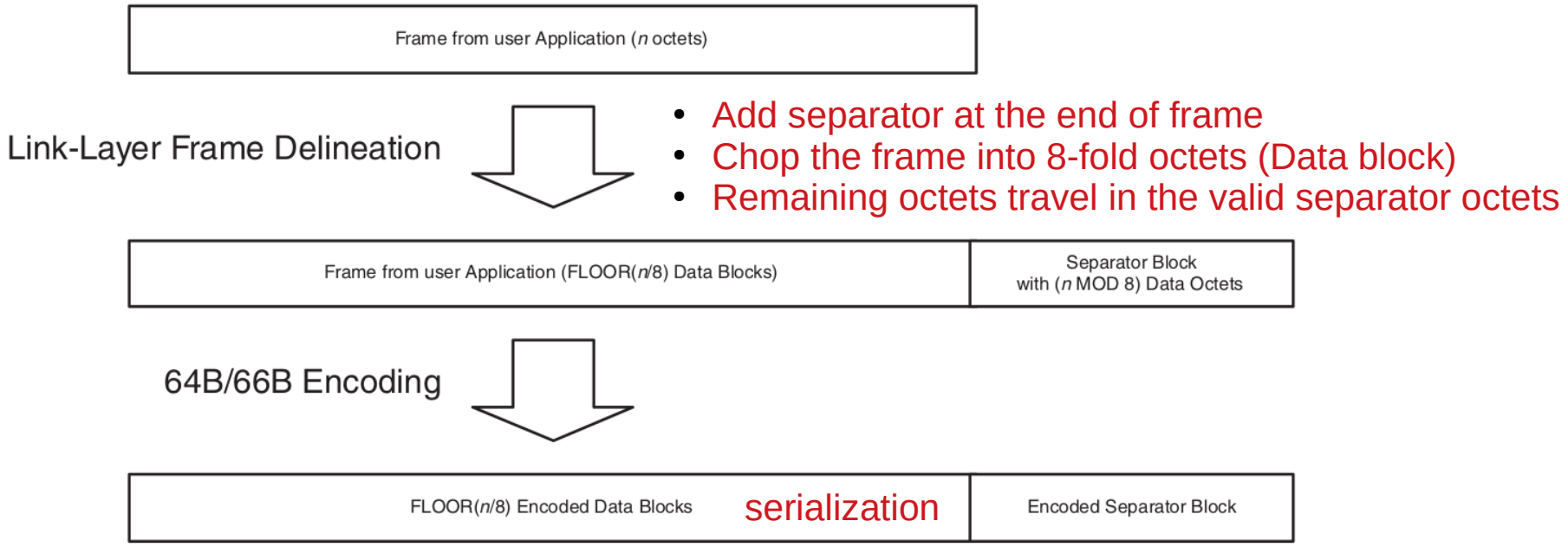
During flow control countdown

Block Name	Priority
Clock Compensation	1 (Highest)
Not Ready	2
Channel Bonding	3
Native Flow Control	4
User Flow Control, Data blocks carrying UFC message	5
User K-Blocks	6
Idle	7
Data, Separator, Separator-7	8

Block type fields (BTF)

Control Block Name	Block Type Field (BTF) Value: Block Code[2:9]
Idle/NotReady/Clock Compensation/ Channel Bonding	0x78
Native Flow Control	0xaa
User Flow Control	0x2d
Separator	0x1e
Separator-7	0xe1
User K-Block 0	0xd2
User K-Block 1	0x99
User K-Block 2	0x55
User K-Block 3	0xb4
User K-Block 4	0xcc
User K-Block 5	0x66
User K-Block 6	0x33
User K-Block 7	0x4b
User K-Block 8	0x87
Reserved	0xff

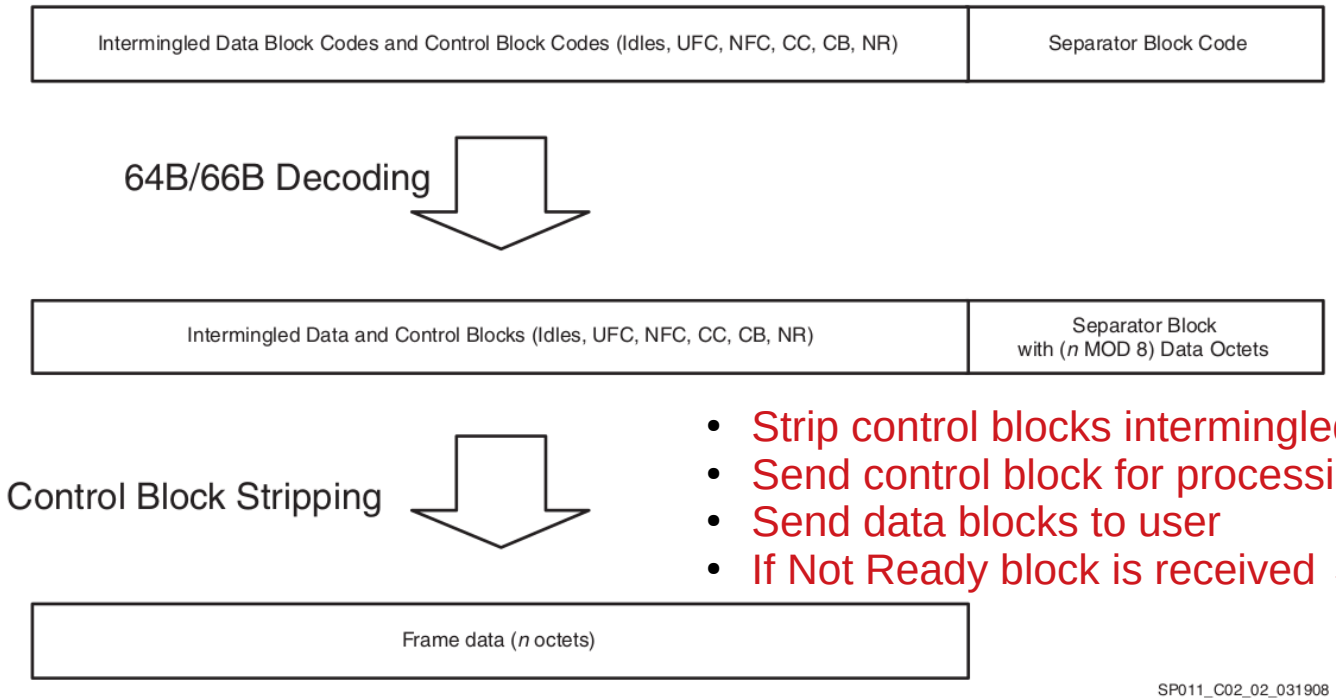
Frame transmission



In multi lane transmission, Data blocks are distributed evenly across lanes

Notice: a higher priority block can be sent at any time, even "inside" a Data block

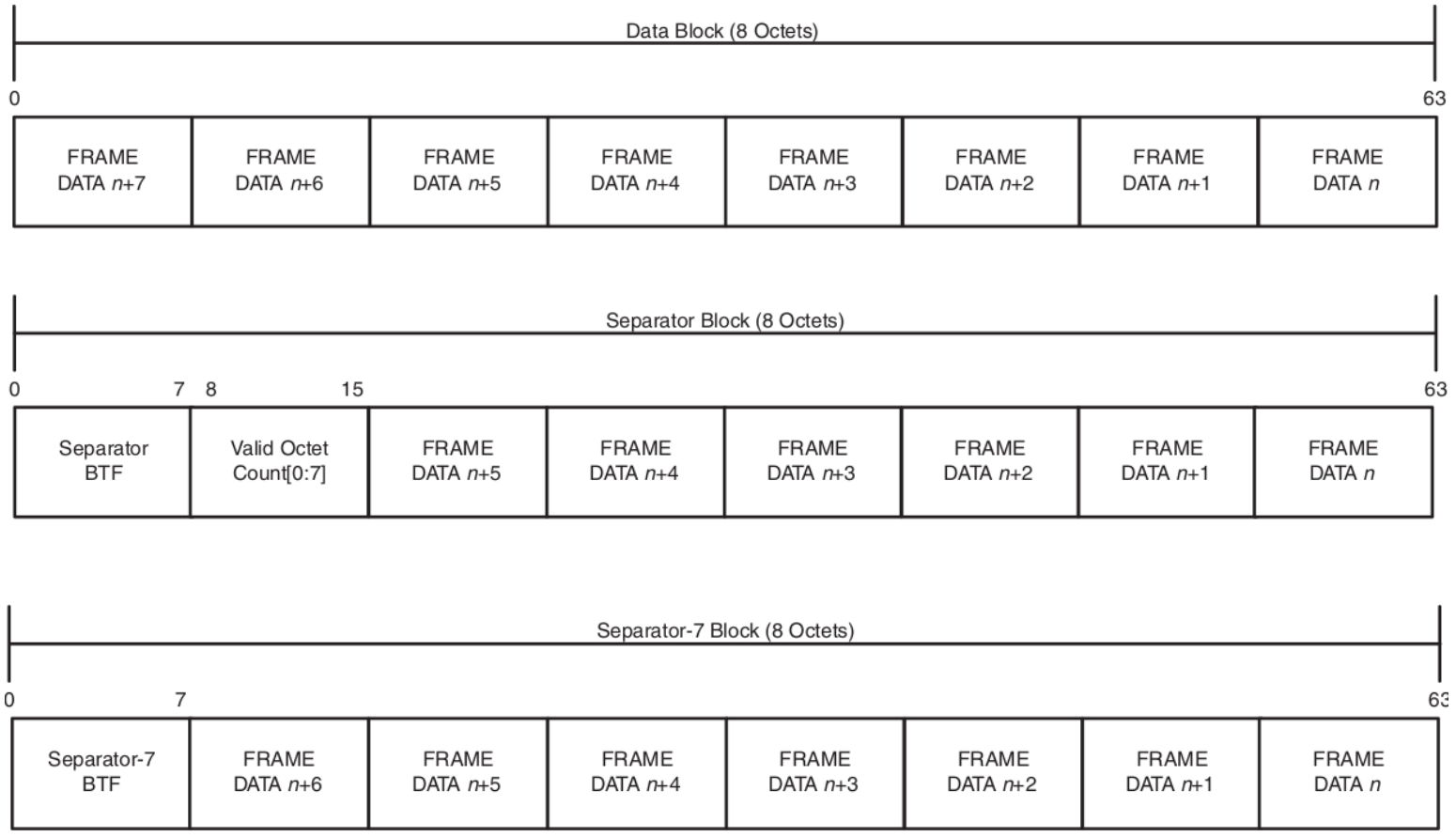
Frame reception



deserialization

- Strip control blocks intermingled with data blocks
- Send control block for processing
- Send data blocks to user
- If Not Ready block is received → reset, frame is lost

Data frames


Data Block (8 Octets)
0
63
FRAME DATA n+7
FRAME DATA n+6
FRAME DATA n+5
FRAME DATA n+4
FRAME DATA n+3
FRAME DATA n+2
FRAME DATA n+1
FRAME DATA n
Separator Block (8 Octets)
0
7
8
15
63
Separator BTF
Valid Octet Count[0:7]
FRAME DATA n+5
FRAME DATA n+4
FRAME DATA n+3
FRAME DATA n+2
FRAME DATA n+1
FRAME DATA n
Separator-7 Block (8 Octets)
0
7
63
Separator-7 BTF
FRAME DATA n+6
FRAME DATA n+5
FRAME DATA n+4
FRAME DATA n+3
FRAME DATA n+2
FRAME DATA n+1
FRAME DATA n

Example of data frame transfer

Single lane

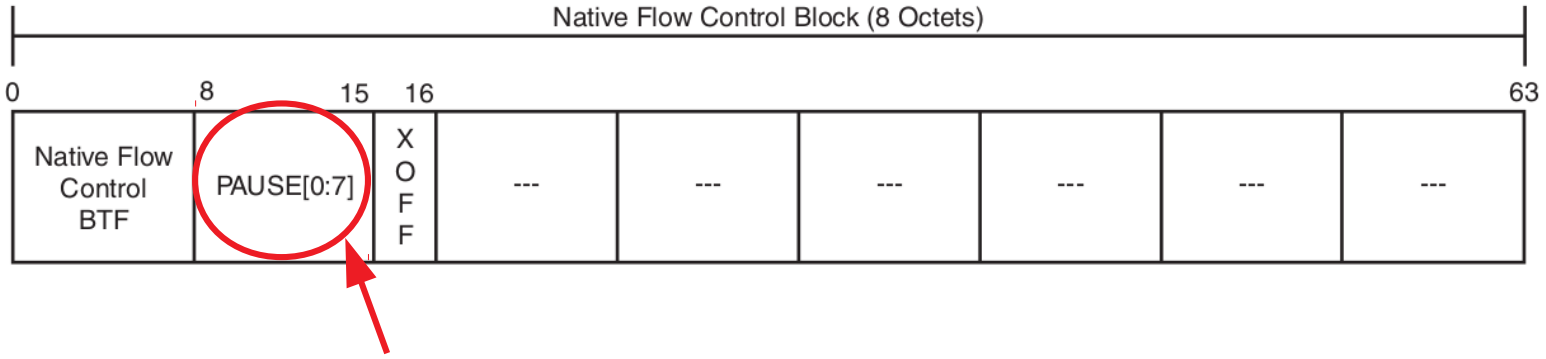
Lane 0
IDLE
07,06,05,04,03,02,01,00
IDLE
SEP,2,–,–,--,--,09,08
IDLE
17,16,15,14,13,12,11,10
1F,1E,1D,1C,1B,1A,19,18
SEP,0,--,--,–,--,--,--
SEP-7,25,24,23,22,21,20
SEP,1,--,–,--,--,--,30

Multi lane

Lane 0	Lane 1
IDLE	IDLE
07,06,05,04,03,02,01,00	SEP,2,–,--,--,--,09,08
IDLE	IDLE
17,16,15,14,13,12,11,10	1F,1E,1D,1C,1B,1A,19,18
SEP,0,--,–,--,--,--,–	SEP-7,25,24,23,22,21,20
SEP,1,–,--,--,--,--,30	IDLE

Native flow control

= request to transmit idles instead of data



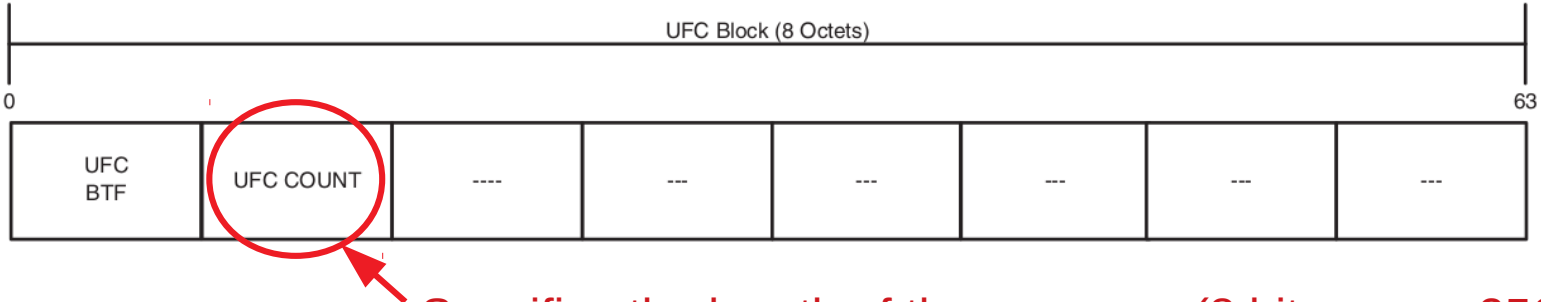
- When this is received, idles are set to higher priority than data
- The binary value indicates the length of the NFC countdown (max 256 cycles)

- If XOFF=1, idles stay high priority until NFC XOFF=0 or Not Ready block
- 2 operation modes: immediate vs completion
- Non-cumulative

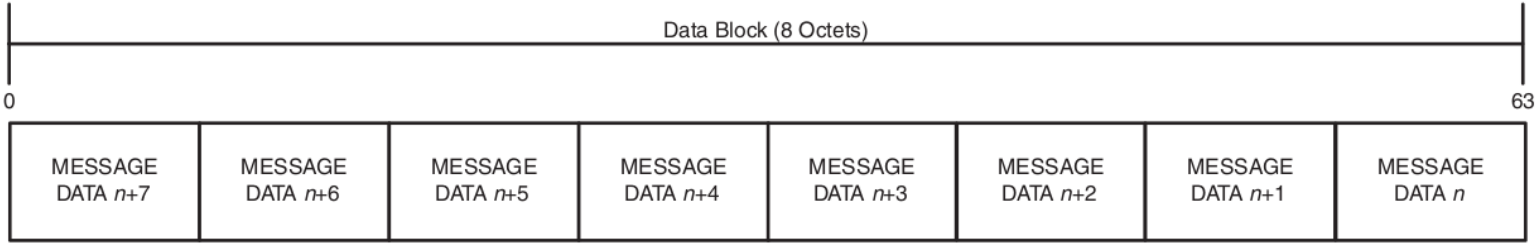
User flow control

To send short, high priority control messages

Header



Message as data frame



Example of UFC transfer

Single lane

Lane 0
IDLE
UFC(10)
07,06,05,04,03,02,01,00
--,--,--,–,--,--,09,08
IDLE
FRAME DATA
UFC(3)
IDLE
--,--,–,–,--,02,01,00
IDLE

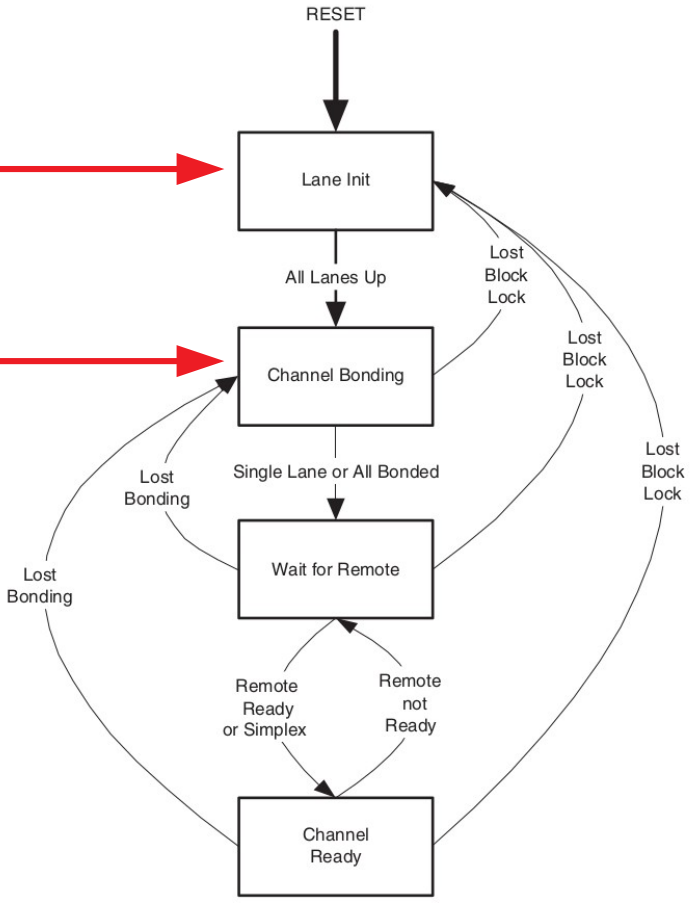
Multi lane

Lane 0	Lane 1
IDLE	UFC(10)
07,06,05,04,03,02,01,00	–,--,--,--,–,--,09,08
IDLE	FRAME DATA
UFC(3)	IDLE
--,--,--,–,--,02,01,00	IDLE

Initialization

Align block boundaries

Remove skew between lanes



Notice: this state machine is always active

Error handling

Errors are of two types:

- **Hard** (e.g. channel disconnection, buffer overflow, hardware failure)
  → Aurora can detect them and **reset**

- **Soft** (illegal values such as 00 and 11, illegal block)
  → Aurora can detect them and inform the user

Sync header