Space Technology Centre
University of Dundee

SpaceFibre

Steve Parkes, Chris McClements,
Martin Dunstan, Peter Mendham

Space Technology Centre
University of Dundee

ESA Study Manager: Martin Suess

Agenda

- SpaceFibre requirements
- Lessons learnt from SpaceWire
- Analysis of high speed serial interfaces
- SpaceFibre CODEC architecture
- SpaceFibre/SpaceWire Router
- SpaceFibre demonstration system
- SpaceFibre demonstration

Space Technology Centre
University of Dundee

SpaceFibre CODEC

Space Technology Centre
University of Dundee

SpaceFibre Requirements

- Faster than SpaceWire (> 1 Gbit/s)
- Further than SpaceWire (100 m)
- Lighter than SpaceWire
- As simple as SpaceWire
- Backwards compatible with SpaceWire
- Galvanically isolated

Lessons Learnt from SpaceWire

- Cable Mass
  - 87 g/m approximately
  - Bi-directional
  - Data strobe signalling
    - No need for PLL
  - Differentially encoded
    - Good EMC performance
  - 8 signal wires
  - Individual screens on pairs
  - Overall screen
  - High cable mass
- To reduce cable mass need fewer wires

Space Technology Centre
University of Dundee

Lessons Learnt from SpaceWire

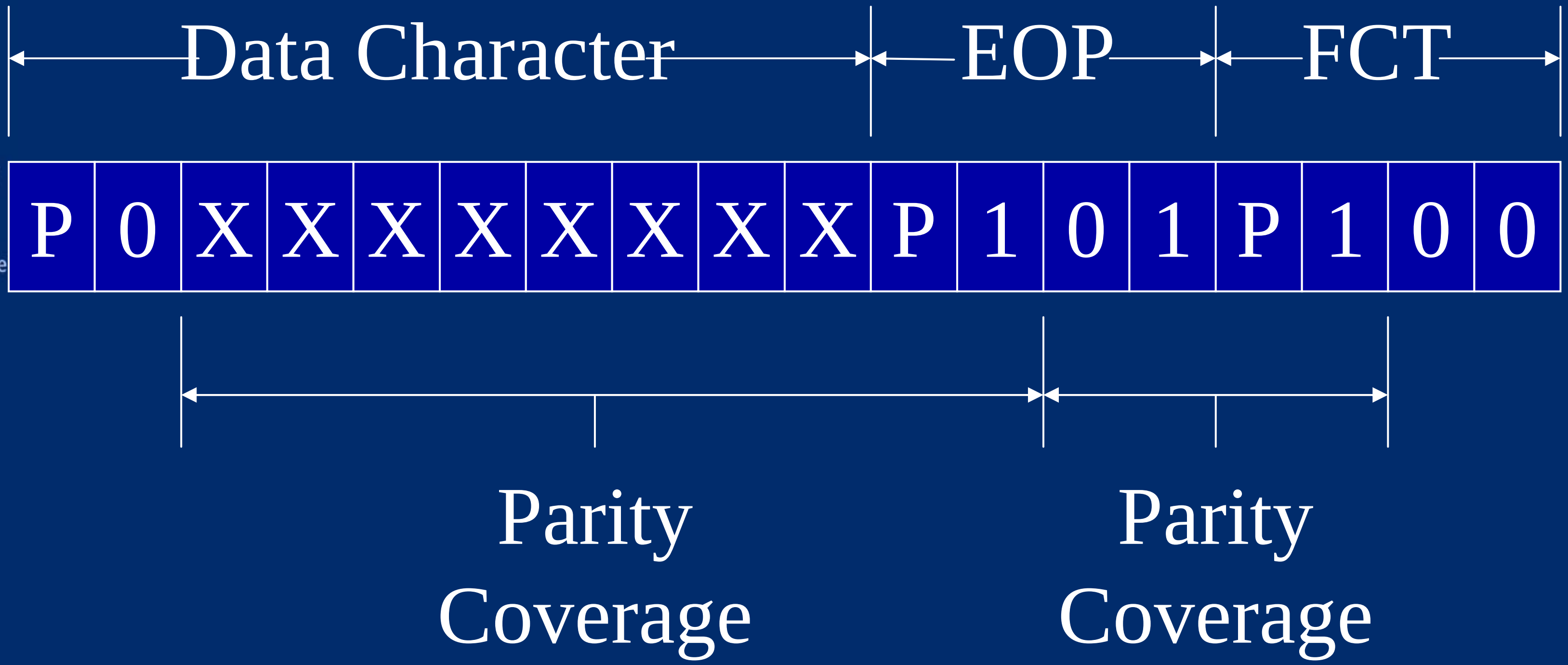
- Data Rate
  - Limited by
    - Cable attenuation
    - Skew between data and strobe signals
  - Longer cables
    - Exacerbate these problems
  - SpaceWire practically limited to
    - 200 Mbits/s
    - Up to 10m length
- Faster links require different signalling scheme

Lessons Learnt from SpaceWire

- Character Sizes
  - FCT, EOP, EEP 4-bits
  - Null 8-bits
  - Data character 10-bits
  - Time-code 14-bits
- Variable character size
  - Makes CODEC design more difficult
- Keep characters all the same size

Lessons Learnt from SpaceWire

- Parity coverage



- Parity bit to cover just one character

Lessons Learnt from SpaceWire

- Transmitted DC Component
  - SpaceWire characters
    - All possible bit patterns used
    - Coding not DC balanced
    - Degrades transmission characteristics
    - Makes them broadband
    - Prevents AC coupling
- Use DC balanced signalling scheme

Space Technology Centre
University of Dundee

Lessons Learnt from SpaceWire

- Galvanic Isolation
  - – No method of galvanic isolation provided in SpaceWire
  - – Various techniques possible
- Support galvanic isolation

Lessons Learnt from SpaceWire

- Matched Impedance Connectors
  - 9-pin MDM connector
  - Not impedance matched
  - Becomes a problem for high data rates
  - Alternative impedance matched connectors
  - Have been developed
- Use impedance matched connector

Lessons Learnt from SpaceWire

- Initialisation Protocol
  - SpaceWire does not use full handshake
    - Part timing
    - Part handshake
  - Allowed backwards compatibility with IEEE1355 devices
  - Can lead to false initialisation due to noise
  - And erroneous flow of data
  - External bias resistors help with this
- A full handshake protocol is preferable

Space Technology Centre
University of Dundee

Requirements

- High data rate
  - 2.5 G bits/s plus
  - Over fibre and copper
- Fibre optic communications
  - 100 m plus
- Copper
  - Short length (1m)
- Galvanically isolated
- Light weight cables
- Low power per Gbit/s
- Radiation tolerant
- Rugged
- Able to integrate with SpaceWire network

Space Technology Centre
University of Dundee

Analysis of Existing Protocols

- Gigabit Ethernet
- Fibre Channel
- Serial ATA
- PCI Express
- Infiniband
- HyperTransport

SpaceFibre Approach

- Use the lower level of Fibre Channel as the basis for SpaceFibre
  - Bit and word synchronisation
  - 8B/10B encoding
  - Ordered Sets
- Scrambling of data should be included for EM emission reduction
- Frame concept used in Fibre Channel, PCI Express and Serial ATA should be adopted
- Fine grained power management of the link interfaces should be supported
- Virtual channel and traffic class concepts of PCI Express should be adopted

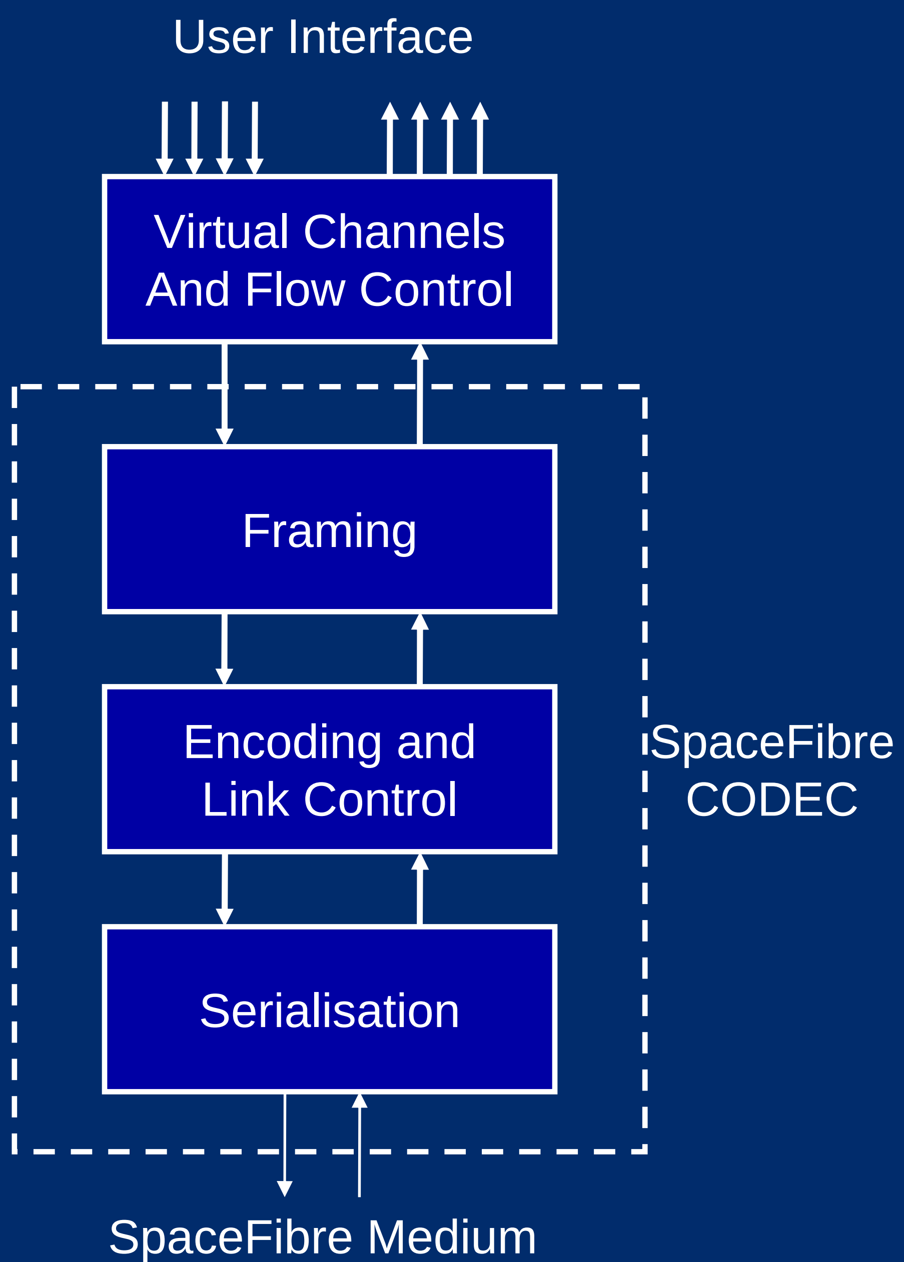
Space
Technology
Centre
University of Dundee

Compatibility with SpaceWire

Space Technology Centre
University of Dundee

- Compatibility with SpaceWire is a key issue
- Wanted to take advantage of existing SpaceWire equipment
- Must be able to integrate with SpaceWire system
- Various options considered
- Decided on:
  - Compatibility at SpaceWire packet level
  - Use a bridge between SpaceWire and SpaceFibre
  - Multiplexes many SpaceWire links over SpaceFibre

Architecture Overview

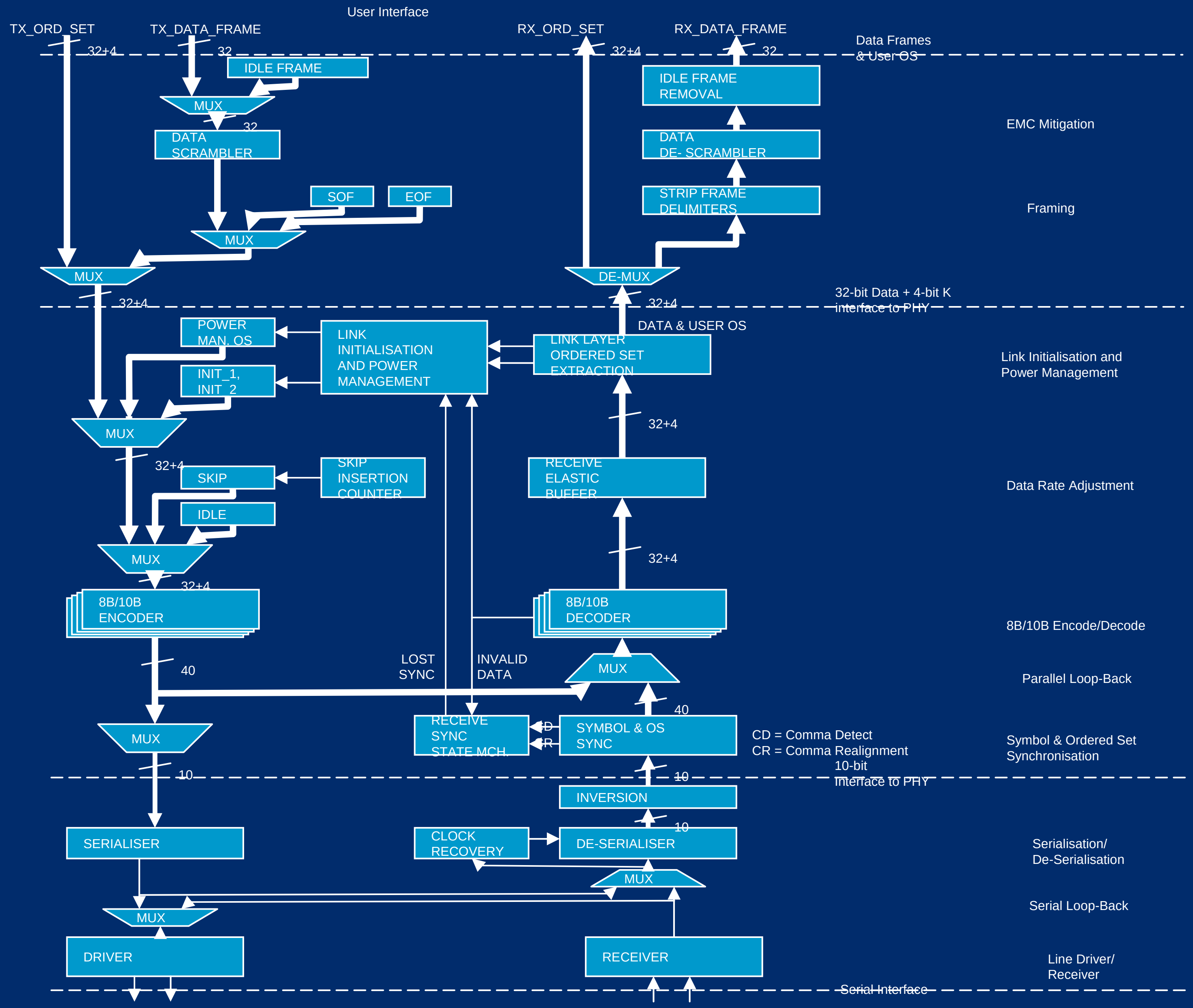


Architectural Overview

- Virtual Channel and Flow Control:
  – Quality of service channels
  – Flow control over SpaceFibre link
- Framing:
  – Framing data
  – inserting user ordered sets
- Encoding and Link Control:
  – Encoding/decoding data into symbols
  – Link initialisation
- Serialisation:
  – Serialising and de-serialising SpaceFibre symbols

Space Technology Centre
University of Dundee

Overall Architecture

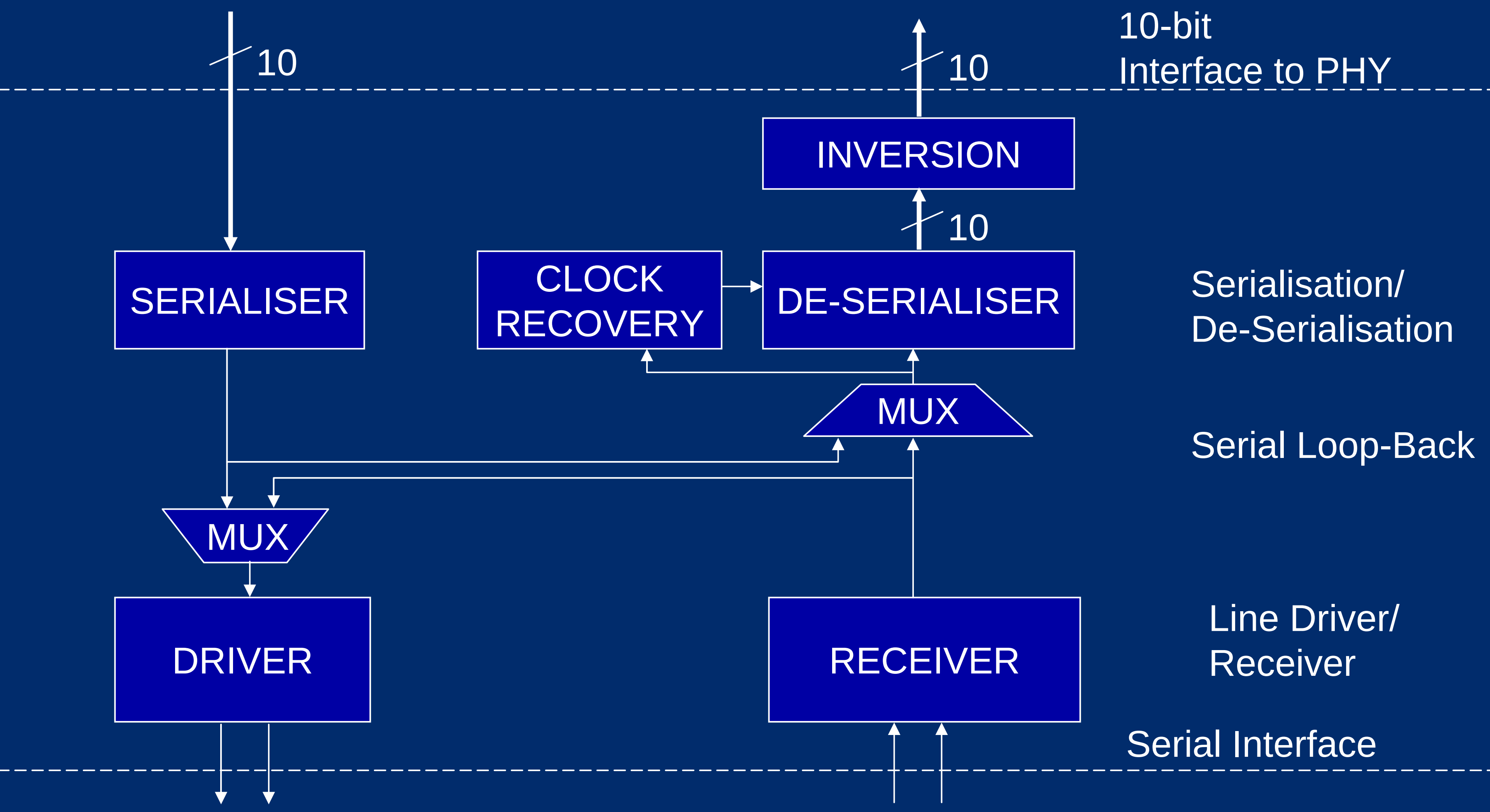


Space
Technology
Centre
University of Dundee

SpaceFibre CODEC Layers

- User interface
- EMC mitigation
- Framing
- Link initialisation and power management
- Data rate adjustment
- 8B/10B encoding and decoding
- Parallel loop-back
- Symbol and ordered set synchronisation
- Serialisation and de-serialisation
- Serial loop-back
- Line driver and receiver

Serialiser / De-serialiser

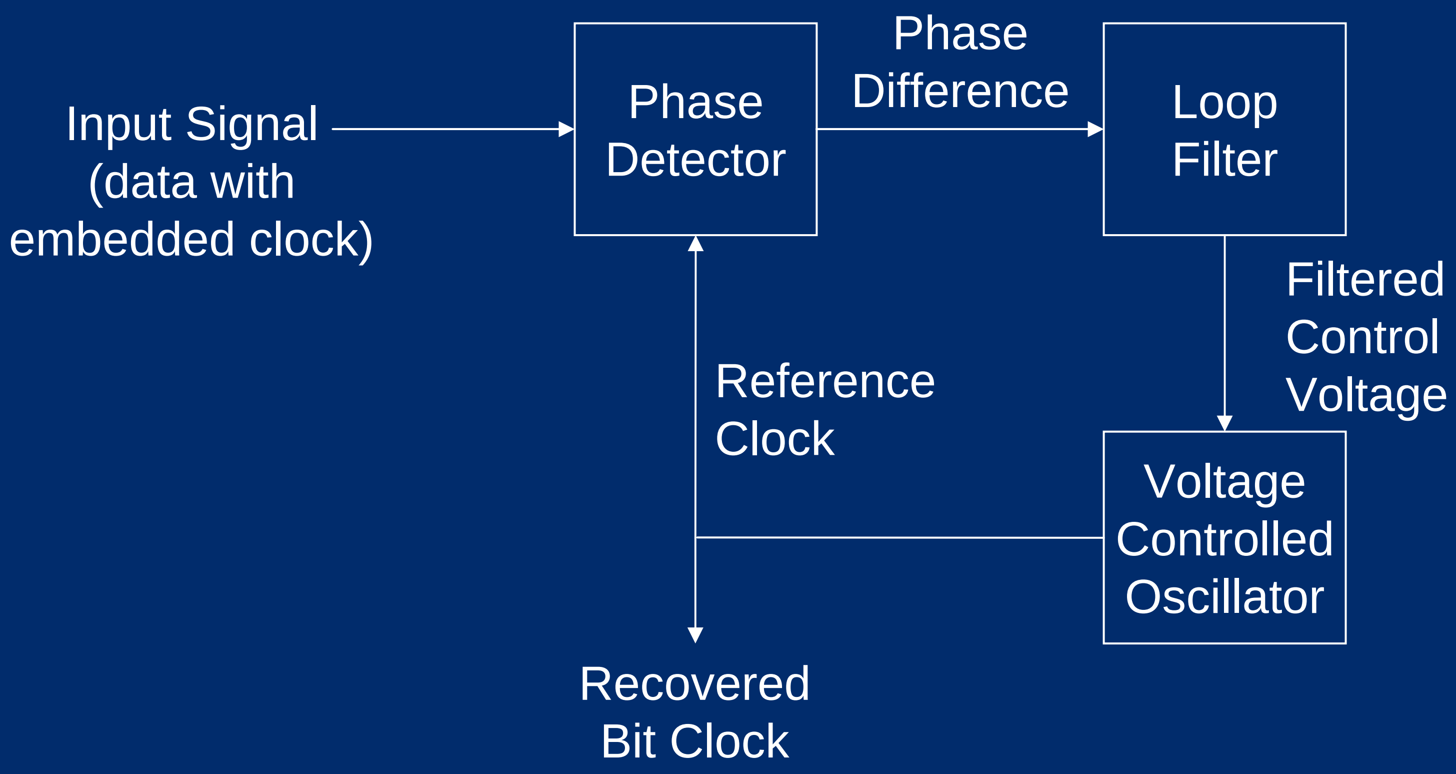


Line Drivers & Receivers

- CML
  - Current mode logic
  - Differential
    - Low EM emissions
  - High speed

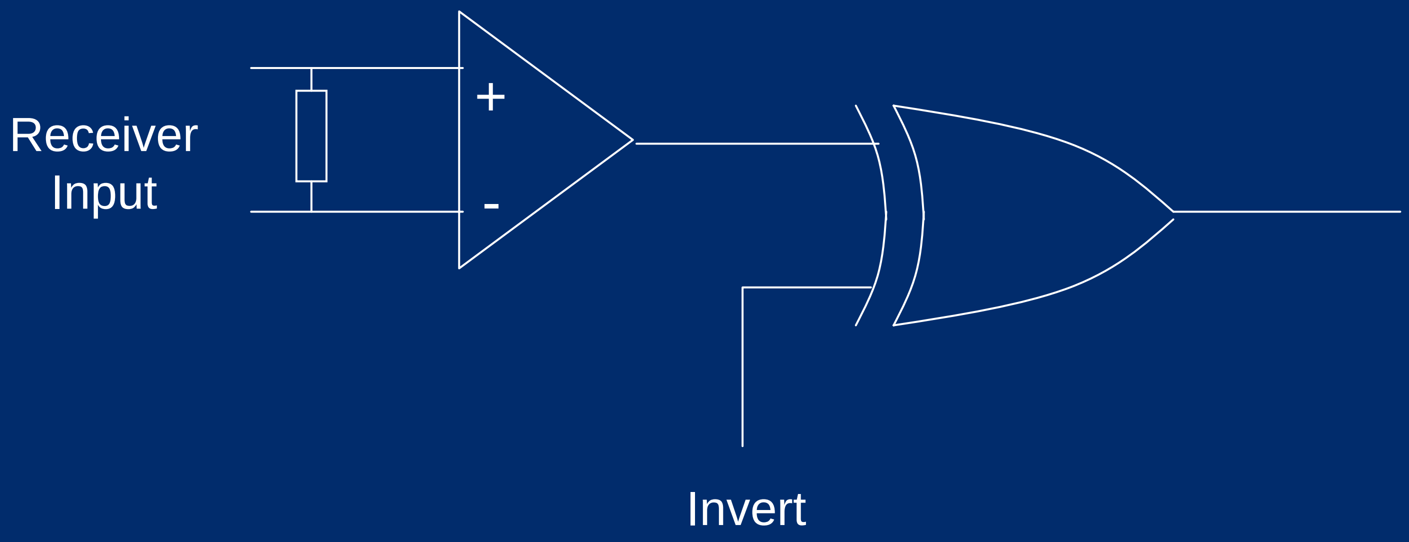
Receive Clock Recovery

- Bit synchronisation



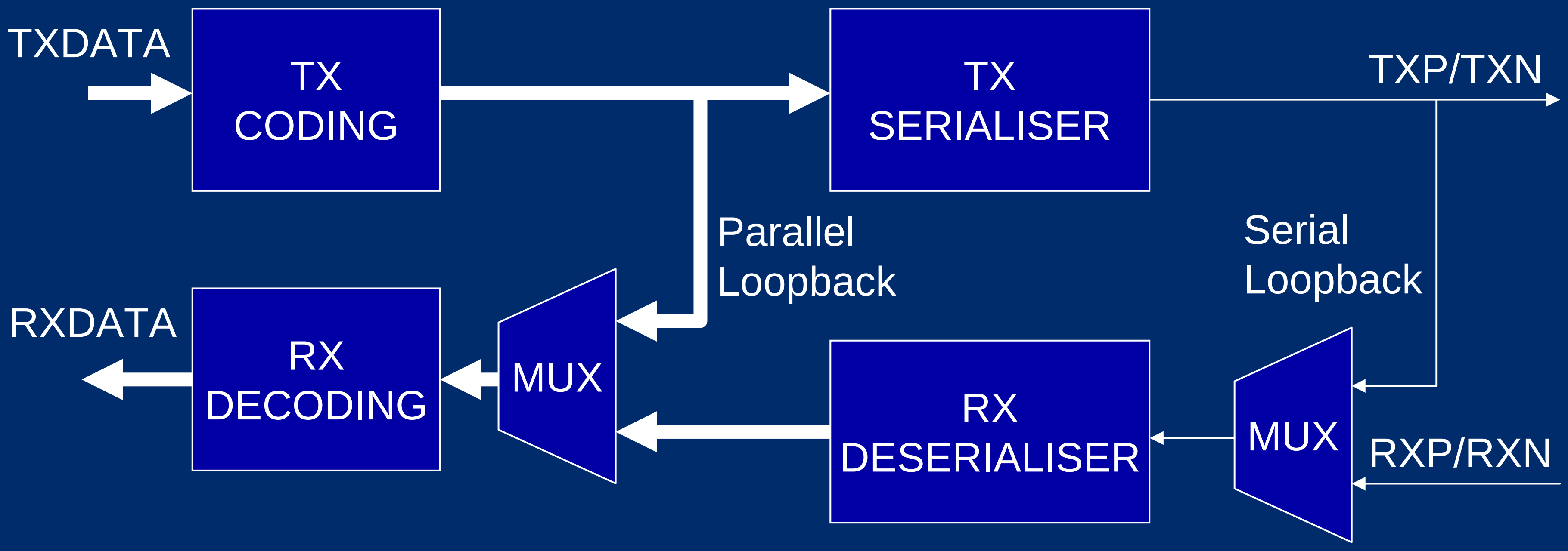
Changing Polarity

- To simplify PCB layout
- Include change of polarity on receiver input
- Detect polarity
- Swap polarity if required

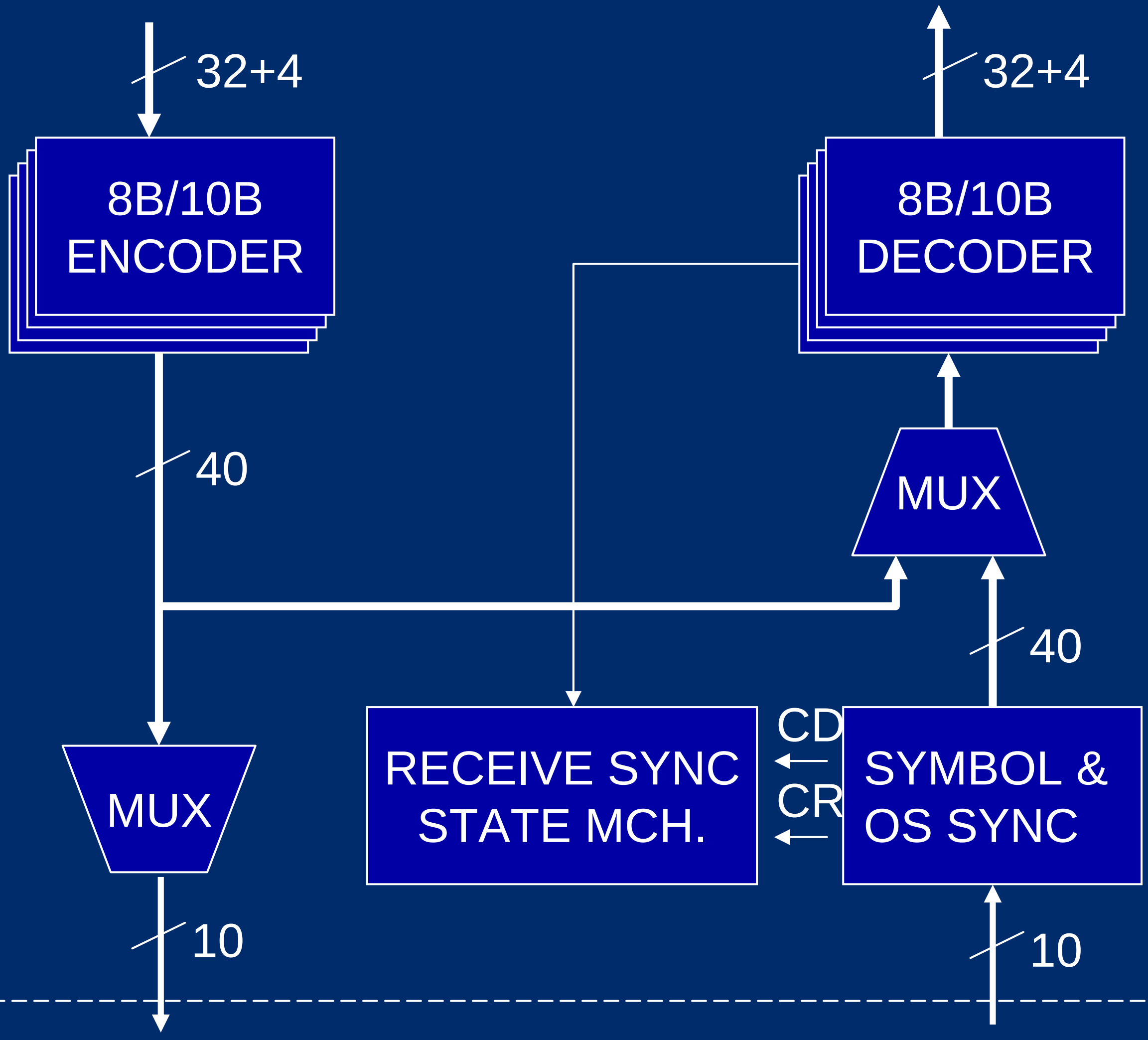


Loopback

- Operating Modes:
  – Normal operation
  – Internal parallel loopback
  – Internal serial loopback



Encoder / Decoder



8B/10B Encode/Decode

Parallel Loop-Back

Symbol & Ordered Set Synchronisation

10-bit
Interface to PHY

CD = Comma Detect
CR = Comma Realignment

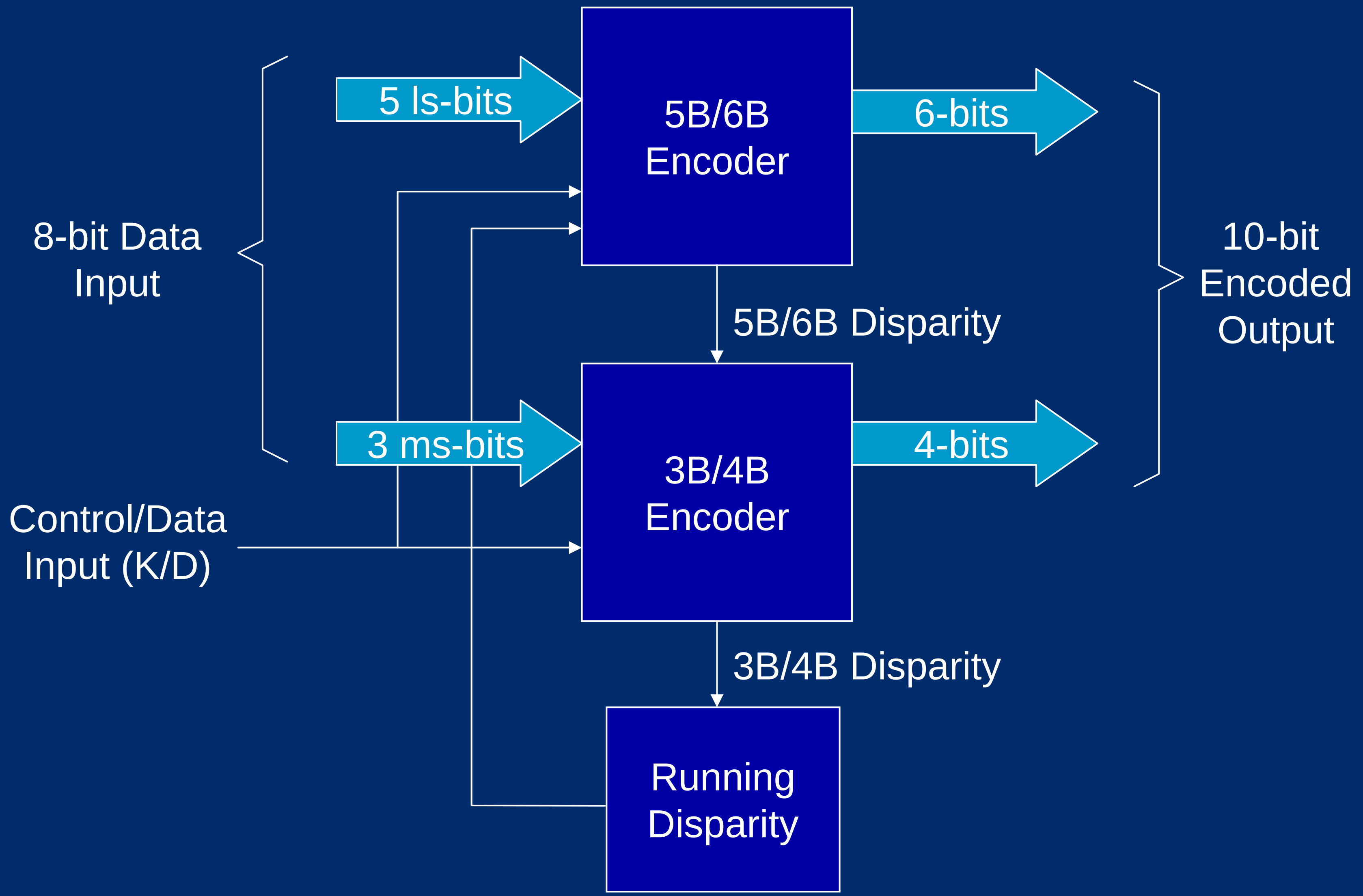
Benefits of 8B/10B Coding

- Zero DC bias: same number of ones and zeros
- 8-bit data codes + some control codes
  - Only codes with
    - 5 ones and 5 zeros
    - 4 ones and 6 zeros
    - 6 ones and 4 zeros
  - Are used
  - Characters with uneven ones and zeros have two possible codings to preserve DC bias

Benefits of 8B/10B Coding

- Ensures sufficient bit transitions for clock recovery
  - No more than 5 consecutive ones or zeros
- All characters encoded with 10-bits giving constant bit and character rates, simplifying transmitter and receiver
- Unused codes can be used to detect link errors

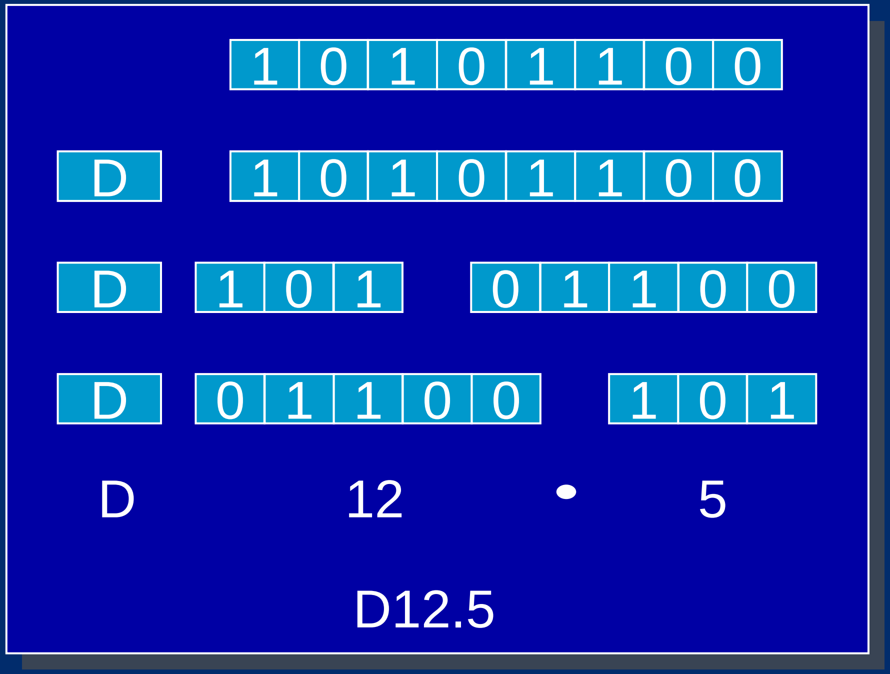
8B/10B Encoder

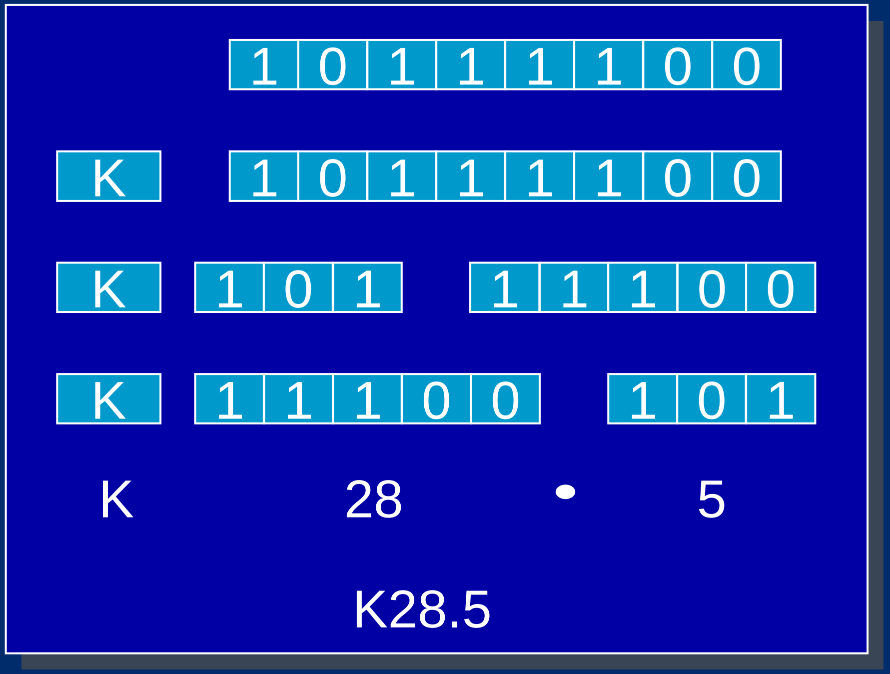


8B/10B Notation

	Data/ Control	ms							ls
ASCII Character		7	6	5	4	3	2	1	0
8B/10B Notation	D/K	H	G	F	E	D	C	B	A
Sub-Blocks	D/K	H	G	F		E	D	C	B	A
Swap Sub-Blocks	D/K	E	D	C	B	A		H	G	F
Decimal values	D/K	XX			•			Y	
Notation	D/KXX.Y								

8B/10B Notation Examples





Part of 5B/6B Encoding Table

Input		Output	
Data Input	**Data bits 43210 (EDCBA)**	**Current Running Disparity -ve abcdei**	**Current Running Disparity +ve abcdei**
D00.y	00000	100111	011000
D01.y	00001	011101	100010
D02.y	00010	101101	010010
D03.y	00011	110001	
D04.y	00100	110101	001010
D05.y	00101	101001	
D06.y	00110	011001	
D07.y	00111	111000	000111
D08.y	01000	111001	000110
D09.y	01001	100101	
D10.y	01010	010101	

3B/4B Encoding

Input		Output	
Data Input	**Data bits 765 (HGF)**	**3B/4B Disparity -ve fghj**	**3B/4B Disparity +ve fghj**
D/Kxx.0	000	1011	0100
Dxx.1	001	1001	
Kxx.1	001	0110	1001
Dxx.2	010	0101	
Kxx.2	010	1010	0101
D/Kxx.3	011	1100	0011
D/Kxx.4	100	1101	0010
Dxx.5	101	1010	
Kxx.5	101	0101	1010
Dxx.6	110	0110	
Kxx.6	110	1001	0110
Dxx.7	111	1110/0111	0001/1000
Kxx.7	111	0111	1000

8B/10B Control (K) Codes

Input	Output	
Special Character Name	**Current Running Disparity -ve**	**Current Running Disparity +ve**
K28.0	001111 0100	110000 1011
K28.1	001111 1001	110000 0110
K28.2	001111 0101	110000 1010
K28.3	001111 0011	110000 1100
K28.4	001111 0010	110000 1101
K28.5	001111 1010	110000 0101
K28.6	001111 0110	110000 1001
K28.7	001111 1000	110000 0111
K23.7	111010 1000	000101 0111
K27.7	110110 1000	001001 0111
K29.7	101110 1000	010001 0111
K30.7	011110 1000	100001 0111

8B/10B Comma Pattern

- Three control codes contain a unique 7-bit pattern
- 0011111 or 1100000
- Does not occur in data codes
- Cannot be produced by combining any data code or other control code
- Pattern is known as the comma pattern
- Widely used for character synchronisation

Commas and character alignment

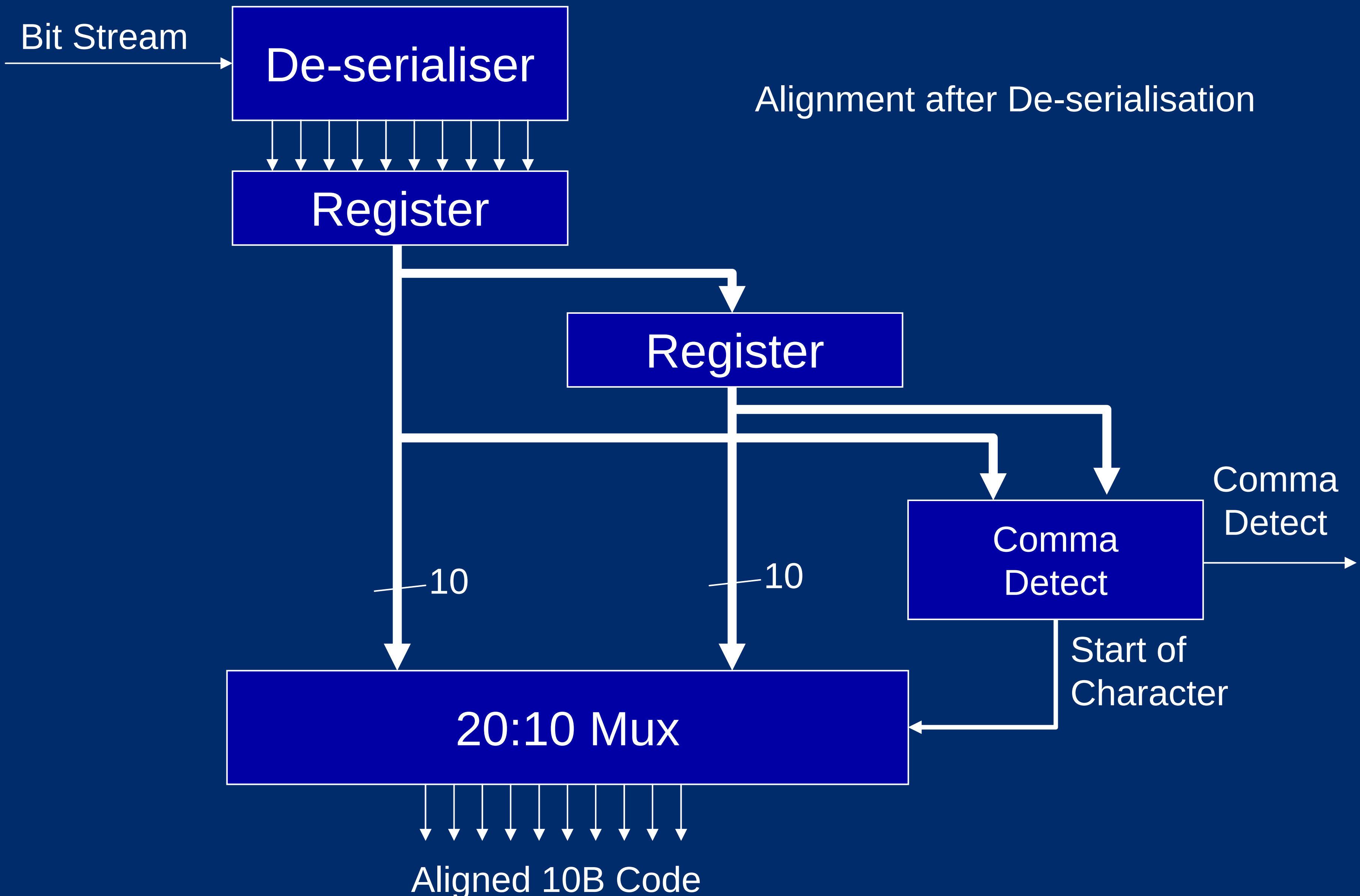
- Commas are used to detect the character boundaries in the serial bit stream
- Comma sequences are unique seven bit sequences
- Plus Comma
  - 0011111
- Negative Comma
  - 1100000
- Example

010011 0011111 001 0101011100

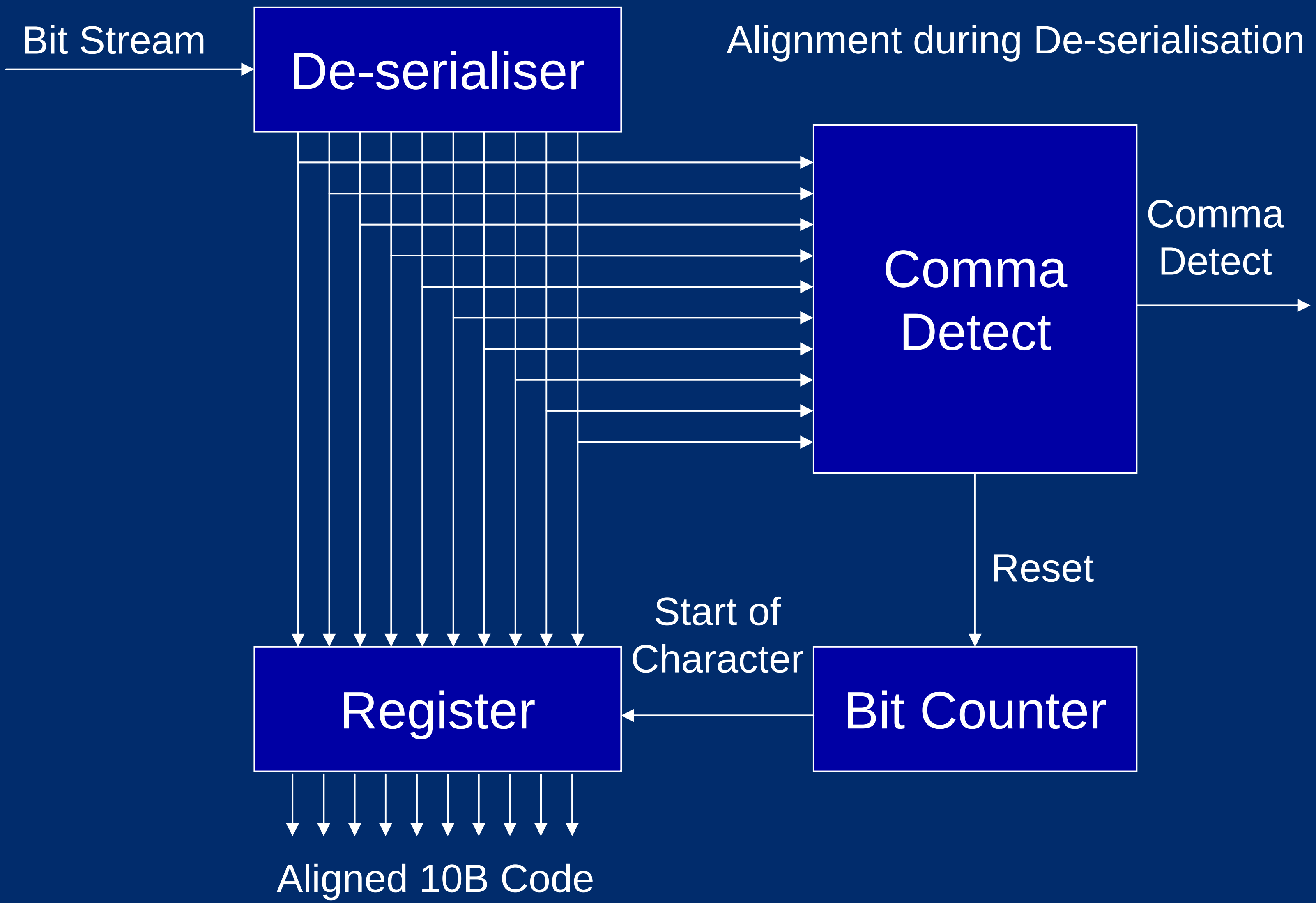
Comma

Start of Character

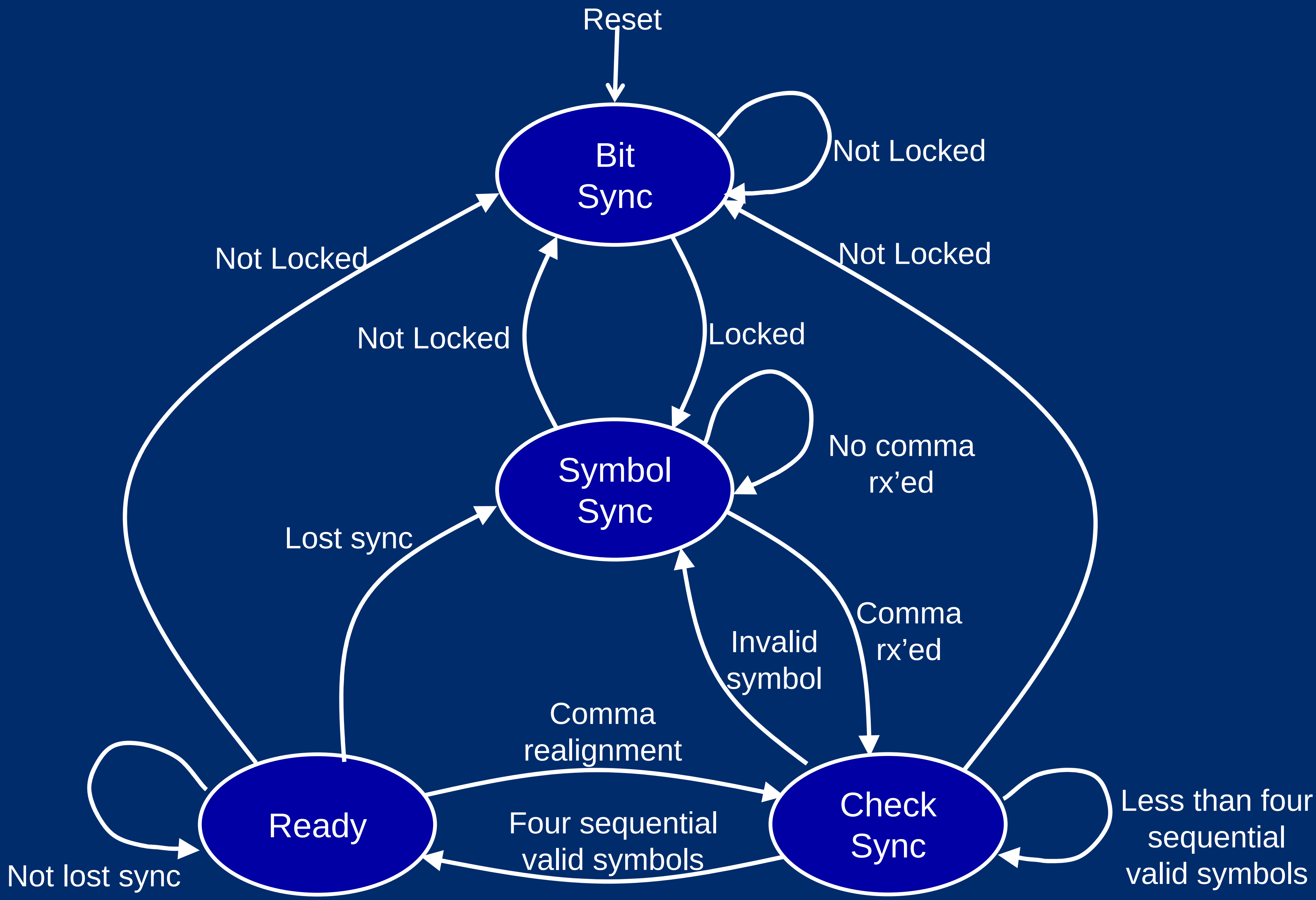
De-serialiser and Character Alignment



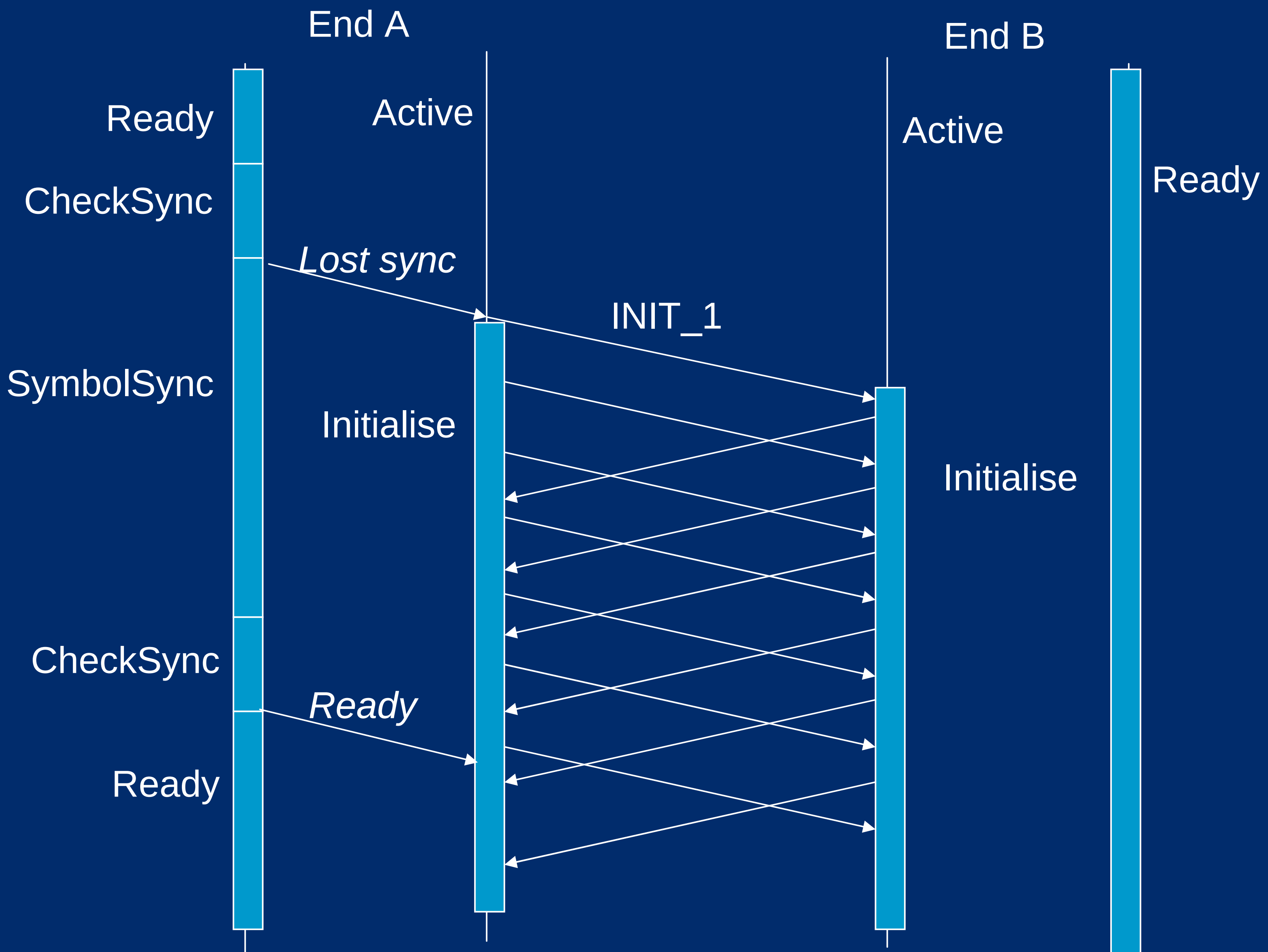
De-serialiser and Character Alignment



RX Synchronisation State Machine



Re-Initialisation after Loss of Sync



Ordered Sets

- A group of four characters
  - Starting with a comma (K28.5)
- Several types of ordered set:
  - Link-level ordered sets
  - Power management ordered sets
  - Reset ordered sets
  - Framing ordered sets
  - Flow control ordered sets
  - User ordered sets

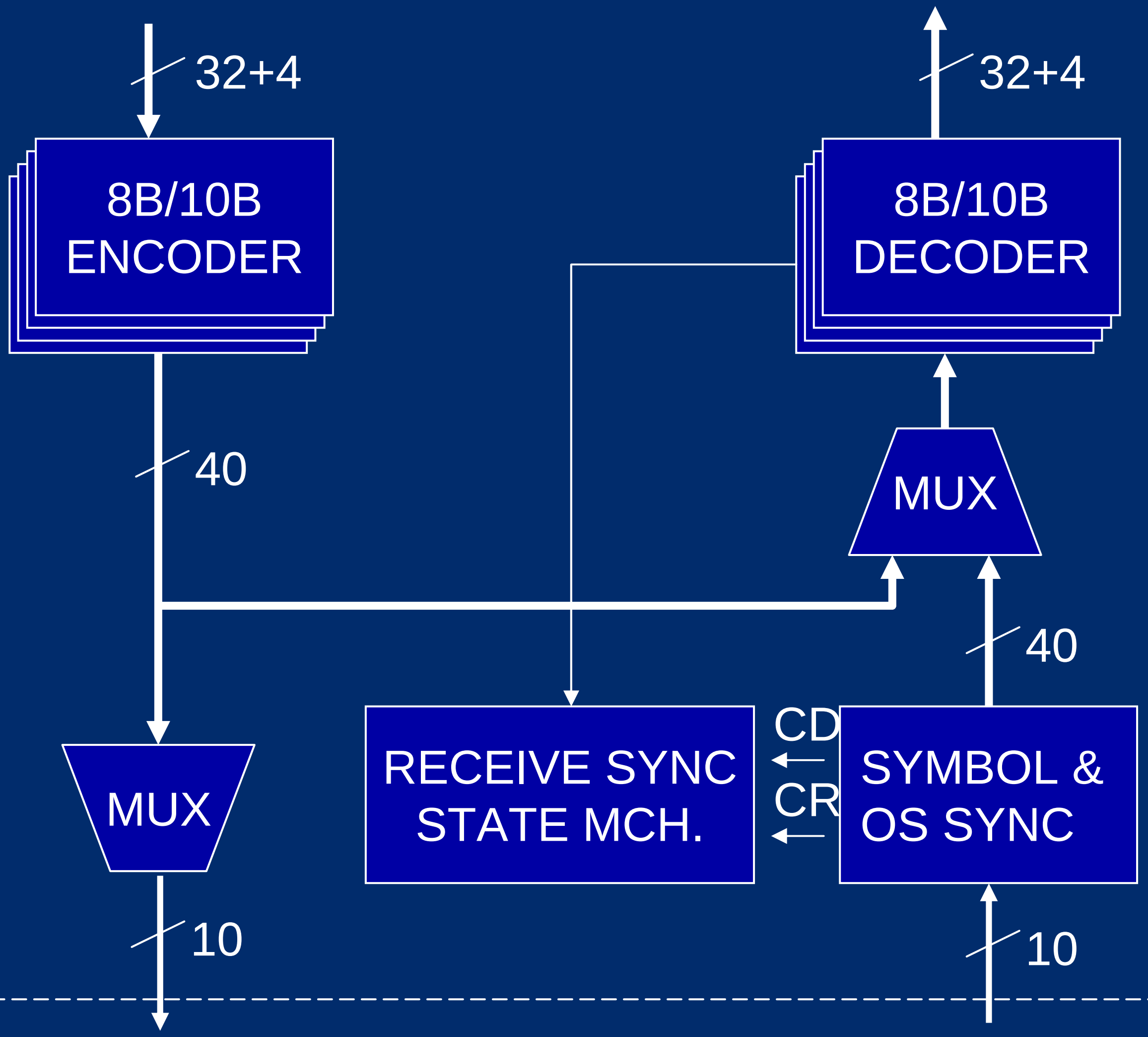
Space Technology Centre
University of Dundee

Link Level Ordered Sets

Link Layer Ordered Sets

Name	**Ordered Set**	**Function**
SKIP	Comma, SKIP, Count MS, Count LS K28.5, D0.0, cnt_ms, cnt_ls	Send every N ordered sets or data words to support receiver elastic buffer operation. N must be less than or equal to 5000.
IDLE	Comma, IDLE, 0, 0 K28.5, D0.1, D0.0, D0.0	Sent whenever there is no data frame, idle frame or other ordered set to send. It keeps the link active.
INIT_1	Comma, INIT, 1, Speed K28.5, D10.2, D0.1, speed	Send as part of the initialisation handshake. If received at any other time causes a re-initialisation.
INIT_2	Comma, INIT, 2, Speed K28.5, D10.2, D0.2, speed	Send as part of the initialisation handshake.

Encoder / Decoder



8B/10B Encode/Decode

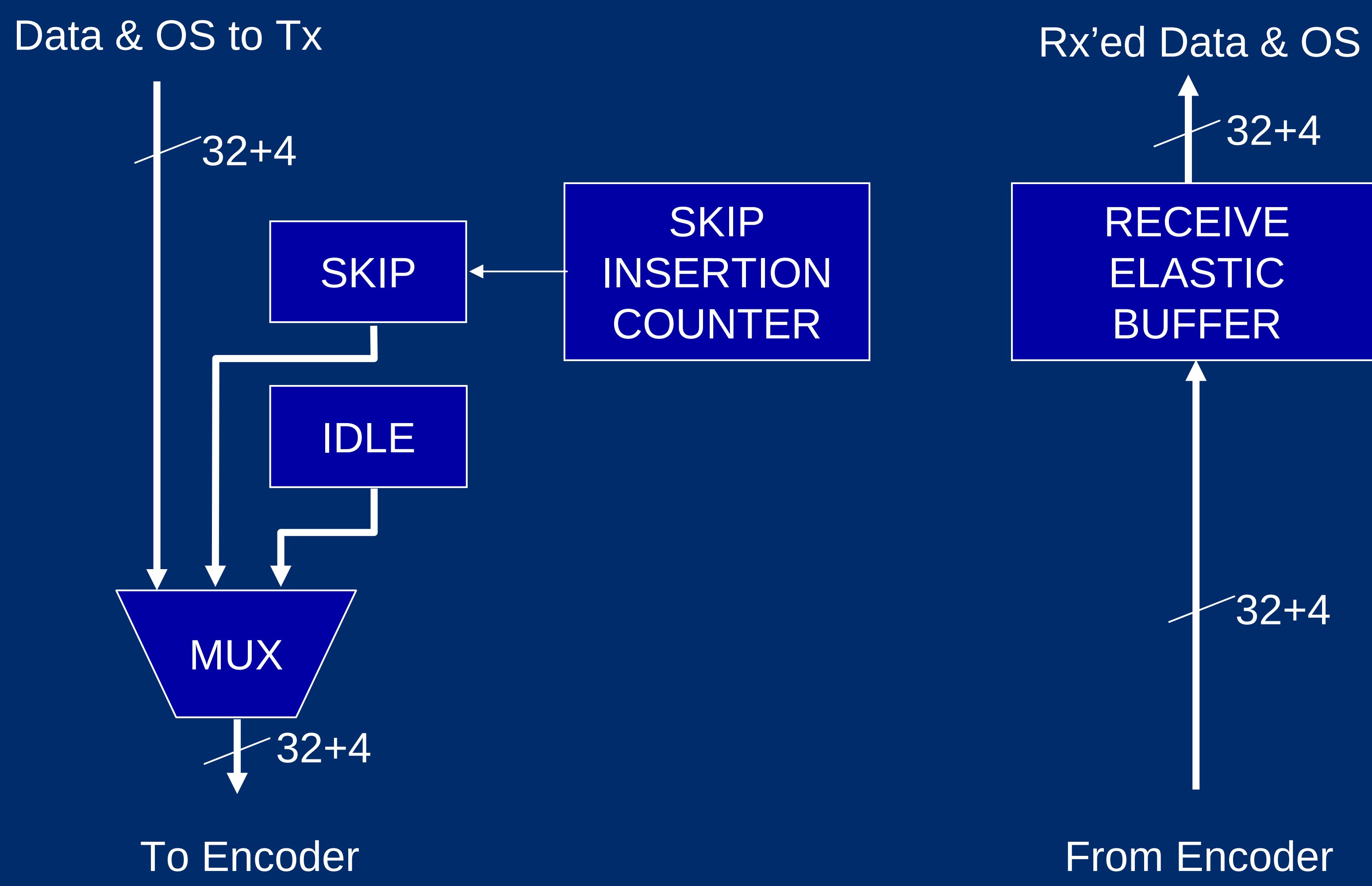
Parallel Loop-Back

Symbol & Ordered Set Synchronisation

10-bit Interface to PHY

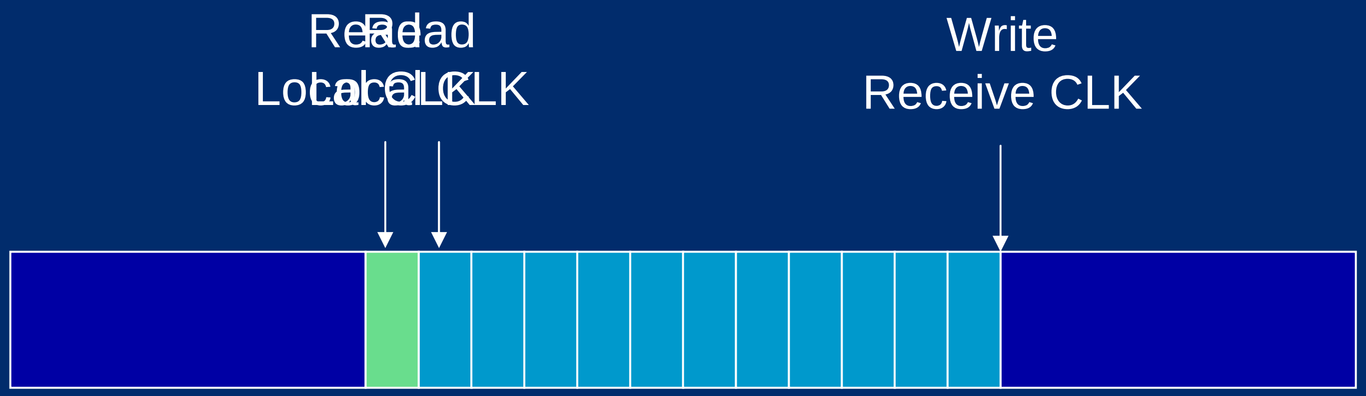
CD = Comma Detect
CR = Comma Realignment

Data Rate Adjustment



Receive Elastic Buffer

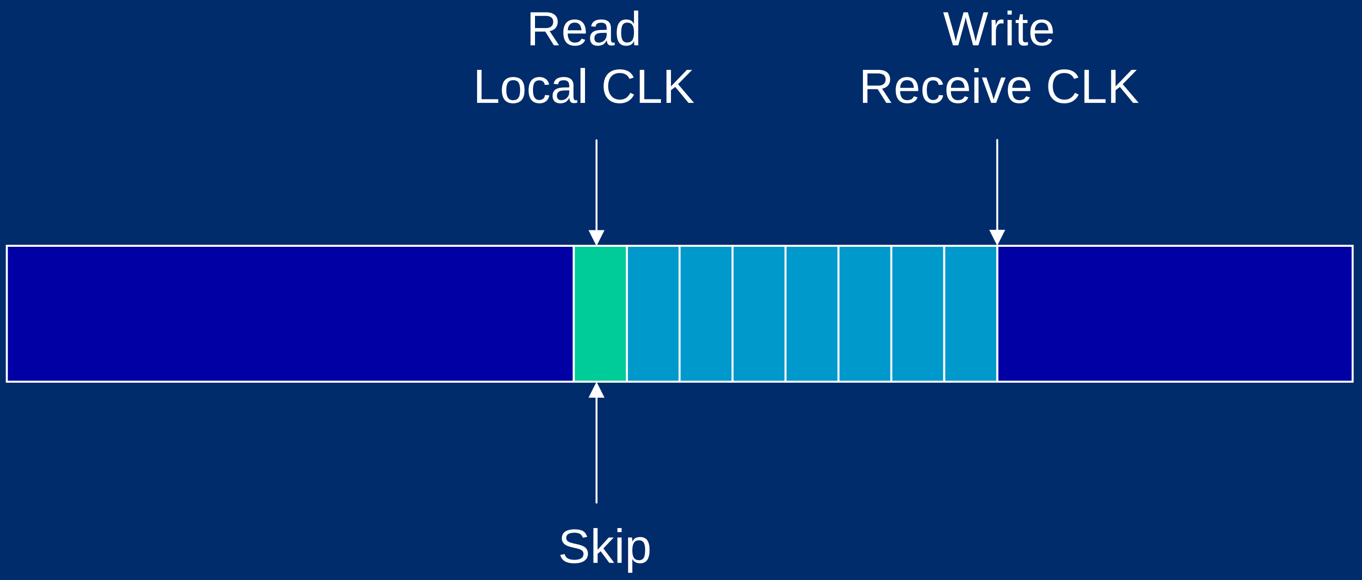
- Receive clock and system clock will be at slightly different frequencies
- Receive elastic buffer makes up for these differences



Nominal condition buffer half-full

Receive Elastic Buffer

- When buffer less than half full
- Local CLK is faster than Receive CLK
- Skip characters are read but read pointer not incremented (once only)
- Effect is to add Skips to the buffer



Buffer less than half full (emptying)

Receive Elastic Buffer

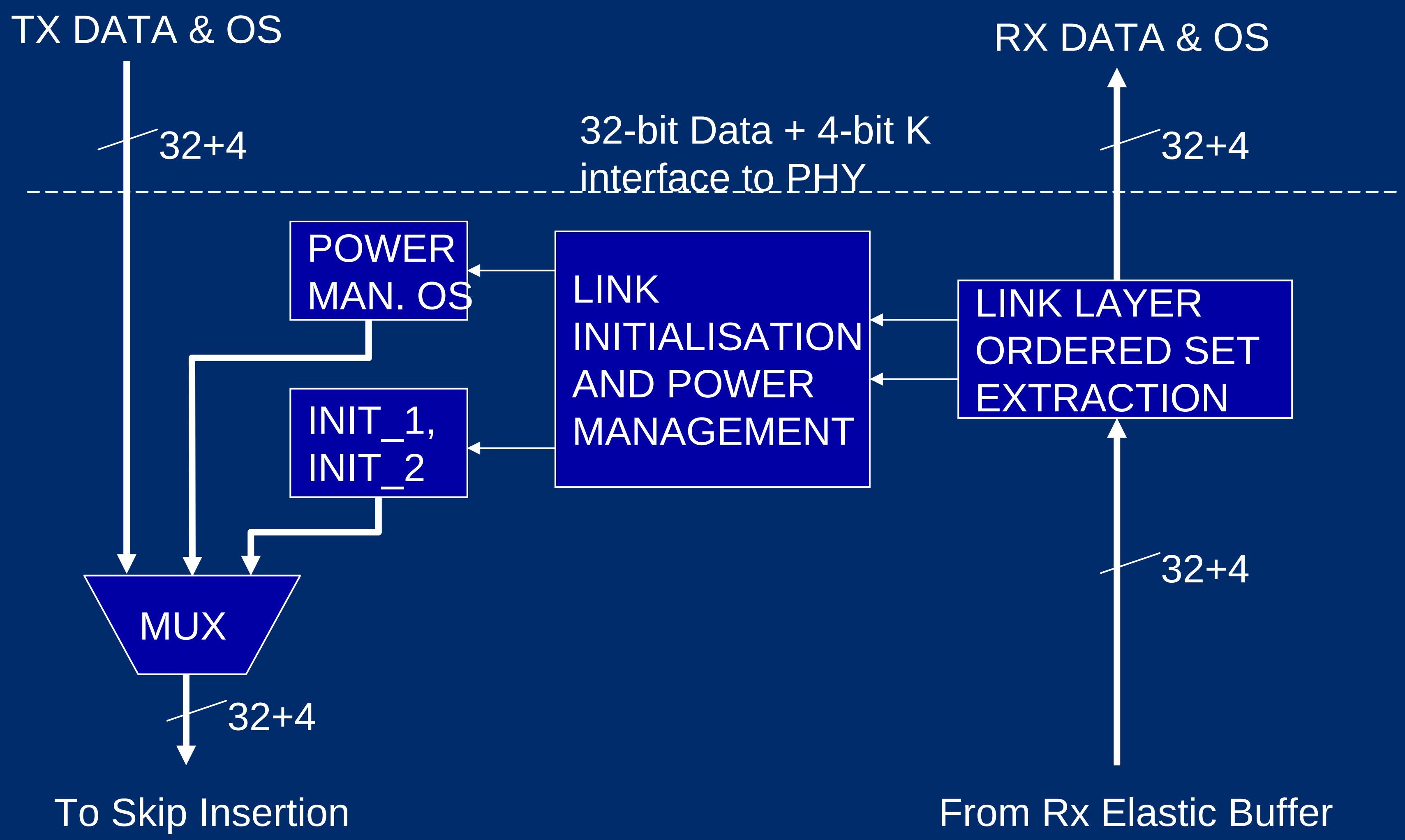
- When buffer more than half full
- Local CLK is slower than Receive CLK
- Skip characters are skipped: read pointer incremented past them
- Effect is to remove skips from buffer

Buffer more than half full (filling up)

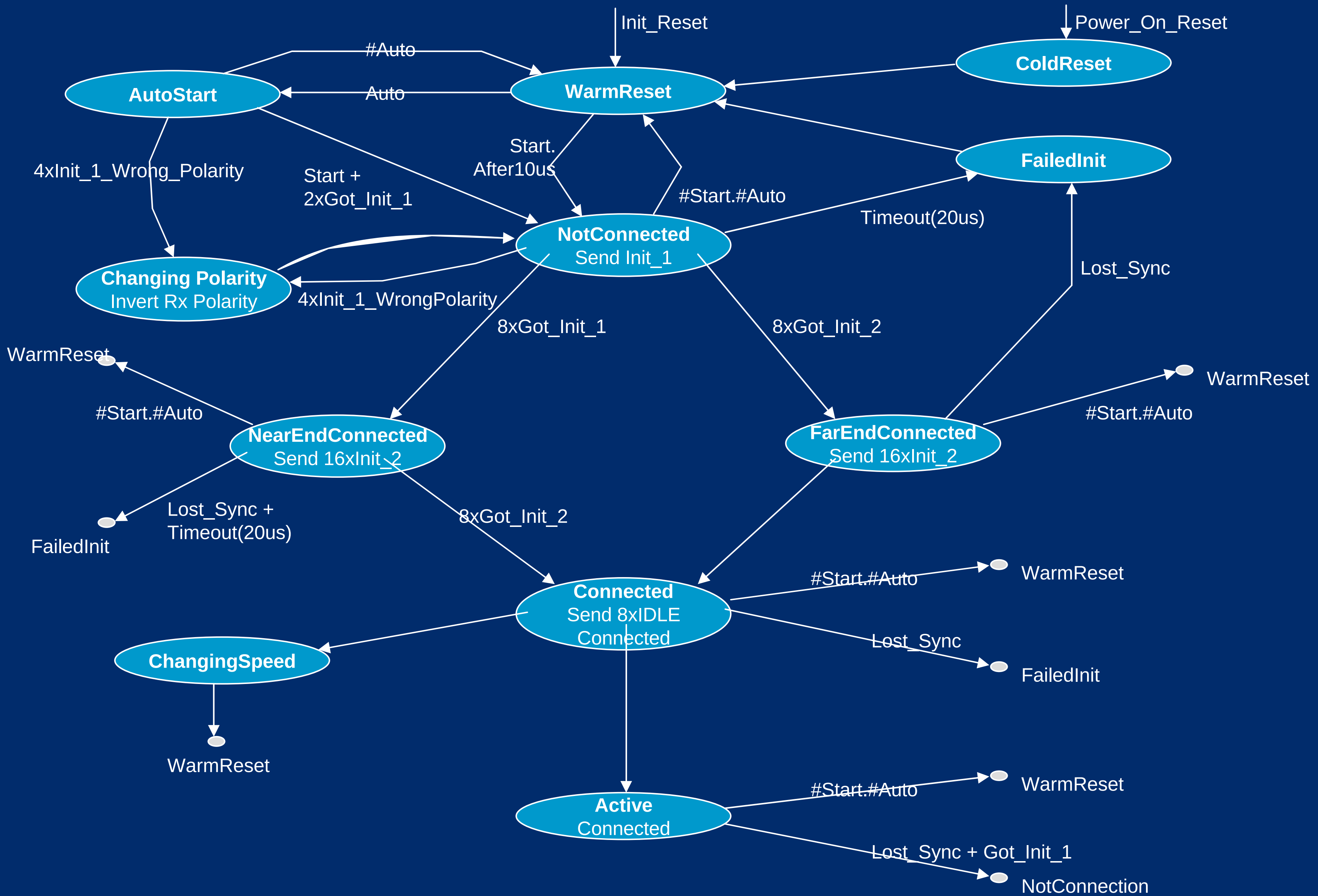
Receive Elastic Buffer

- Must ensure that there are sufficient Skips in the data stream
- So that they can be removed if necessary
- Frequency of Skips depends on:
  - Size of elastic buffer
  - Maximum frequency difference between
    - Local CLK: System clock at this end of link
    - Receive CLK: System clock at other end of link
    - One skip every 20000 symbols

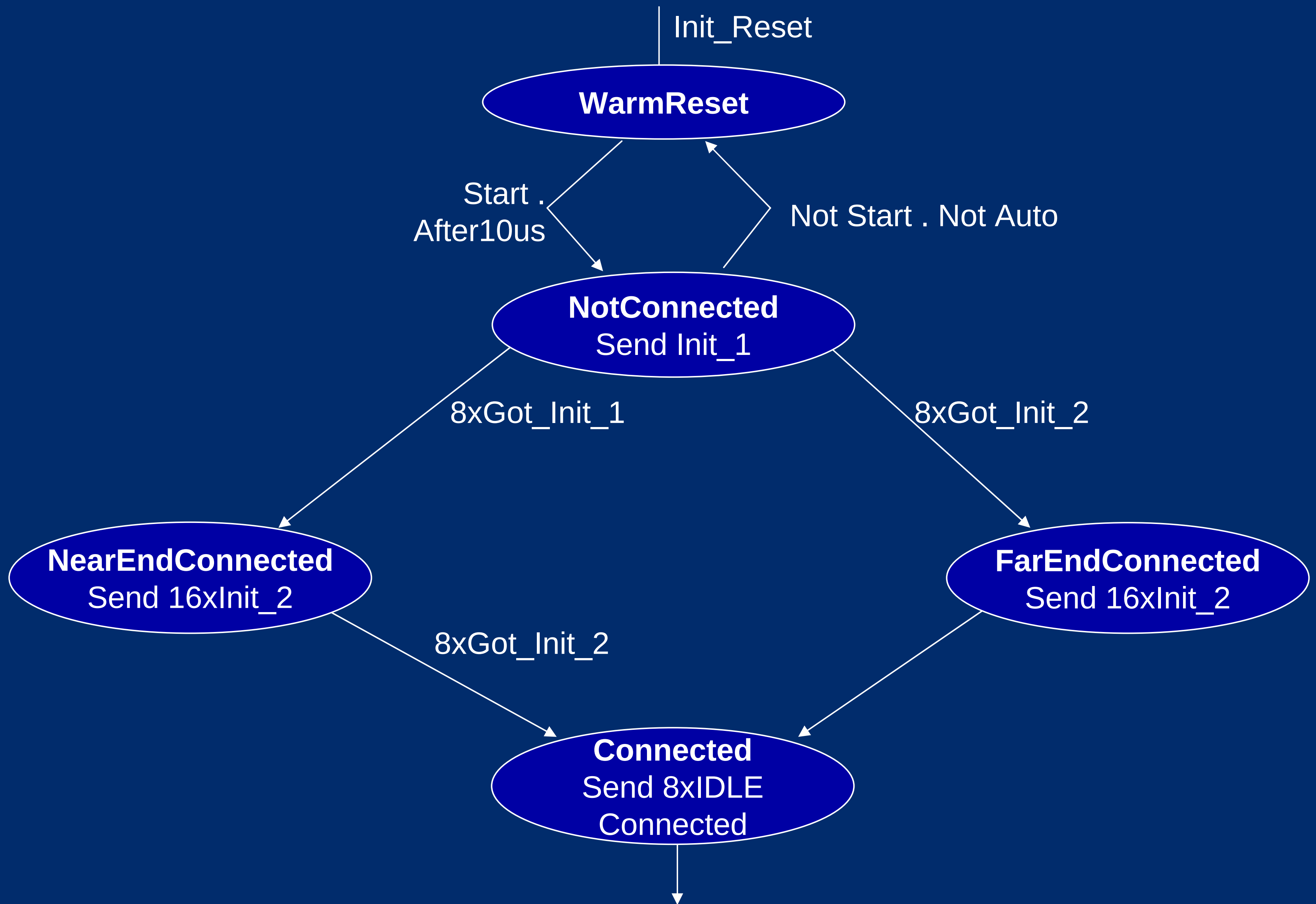
Link Initialisation and Power Management



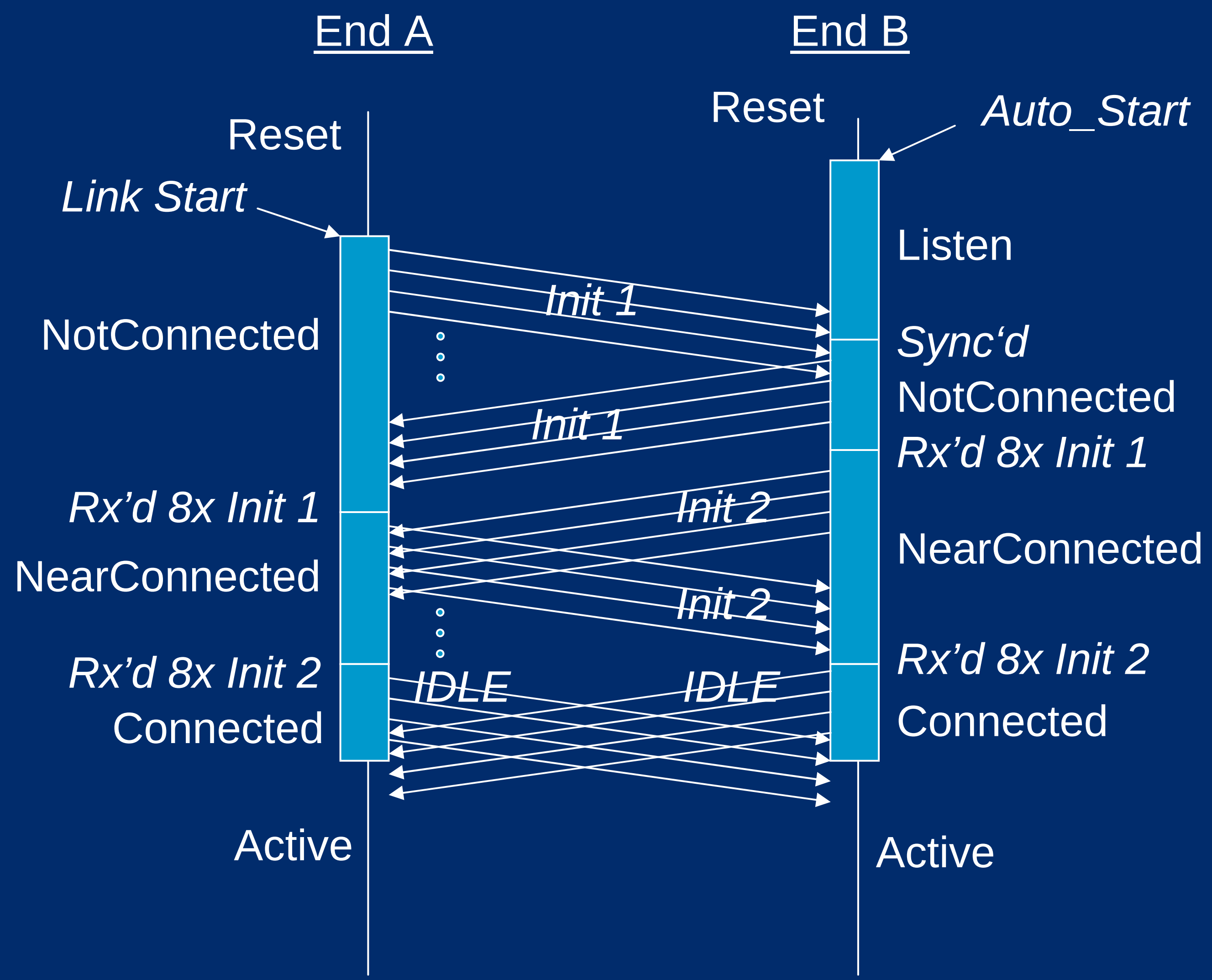
Link Initialisation State Machine



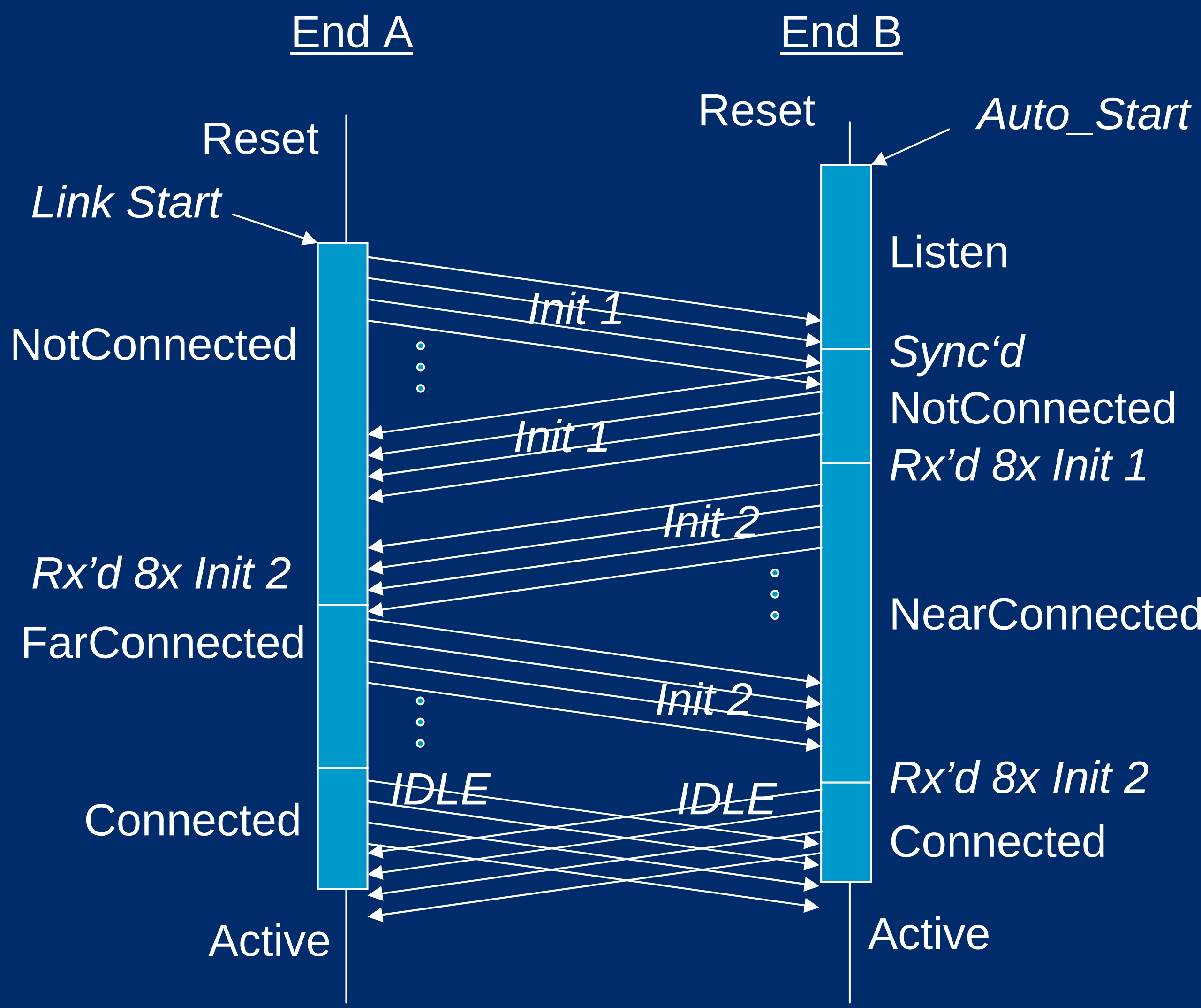
Link Initialisation State Machine



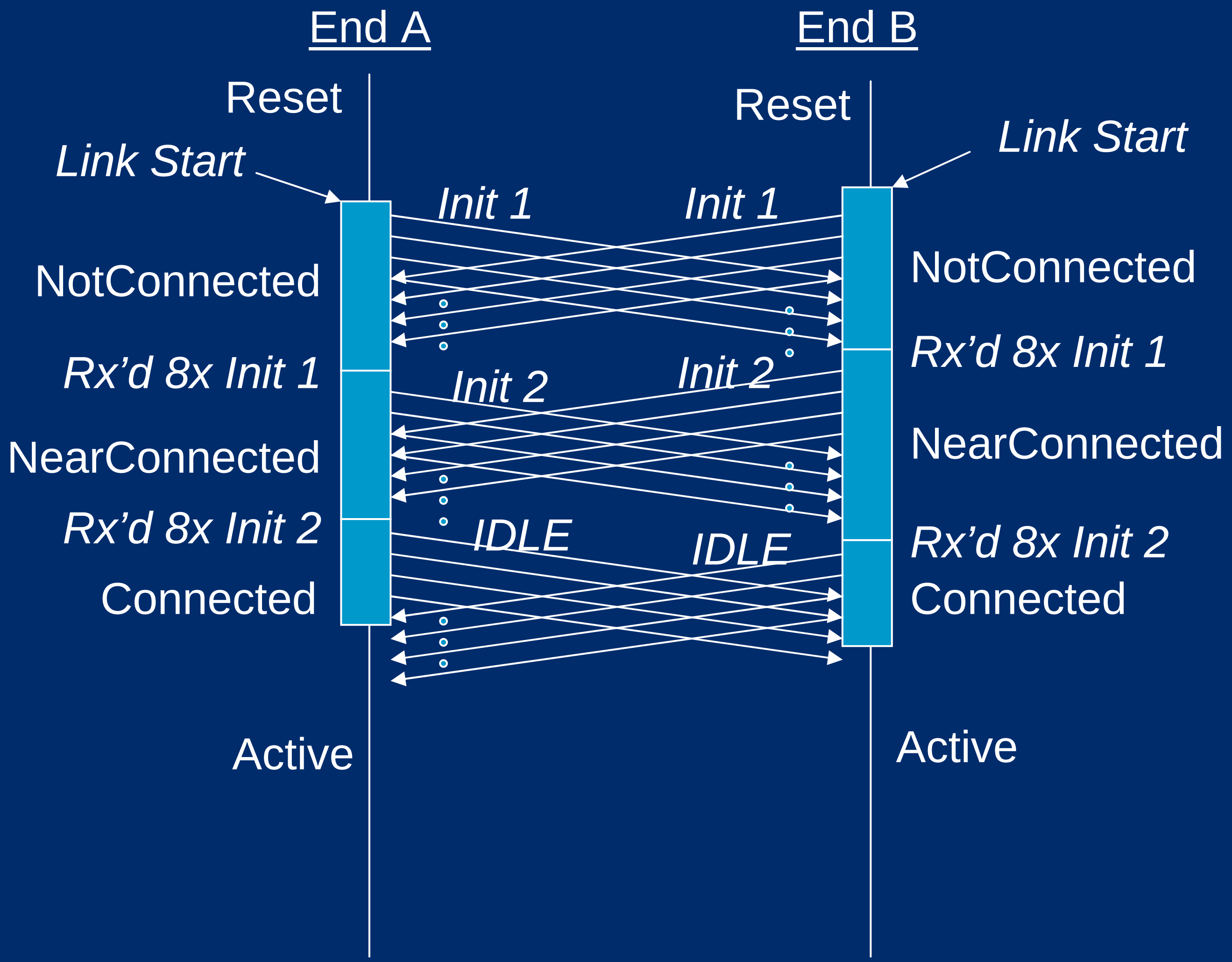
Initialisation from Auto-Start



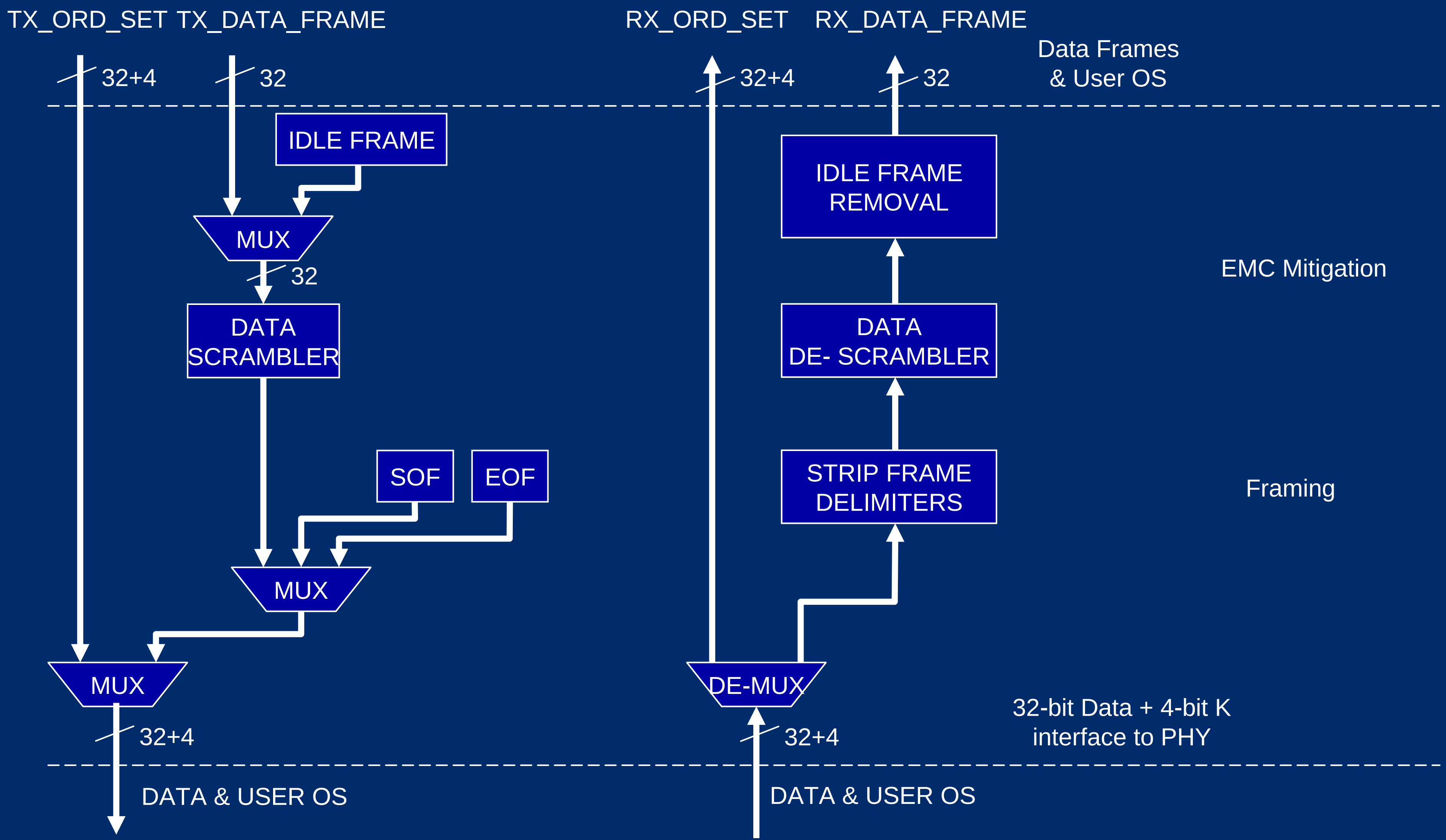
Initialisation through FarConnected



Initialisation when both ends Link Start



EMC Mitigation and Data Framing



Frames

- Data sent in frames
- Frame defined by
  - Start of Frame
  - Data
  - CRC
  - End of Frame
- When no data to send idle frame sent

Data Framing Ordered Sets

Data Framing Ordered Sets		
Name	**Ordered Set**	**Function**
SDF	Comma, SDF, VC, Word Count K28.5, D0.2, VC, Len	Start of Data Frame. Contains type of frame, virtual channel, number and length of frame .
SIF	Comma, SIF, VC, Word Count K28.5, D0.3, VC , Len	Start of Idle Frame. Contains type of frame, virtual channel number and length of frame.
EOF	Comma, EOF, MS, LS K28.5, D0.4, crc_ms, crc_ls	End of Frame. Contains for frame.
EEF	Comma, EEF, MS, LS K28.5, D0.5, crc_ms, crc_ls	Error End of Frame. Indicates that the frame was terminated early for some reason.

Data Frame Format

31 — 24	23 — 16	15 — 8	7 — 0
COMMA	SDF	VC	Word Count
DATA 1 MS	DATA 1	DATA 1	DATA 1 LS
DATA 2 MS	DATA 2	DATA 2	DATA 2 LS
...	...	...	...
DATA	DATA N	DATA N	DATA N LS
COMMA	EOF	CRC_MS	CRC_LS

Idle Frame Format

31 ... 24	23 ... 16	15 ... 8	7 ... 0
COMMA	SIF	0	255
IDLE 1 MS	IDLE 1	IDLE 1	IDLE 1 LS
IDLE 2 MS	IDLE 2	IDLE 2	IDLE 2 LS
...	...	...	...
IDLE	IDLE N	IDLE N	IDLE N LS
COMMA	EOF	CRC_MS	CRC_LS

Idle word is 0x00000000

Space Technology Centre
University of Dundee

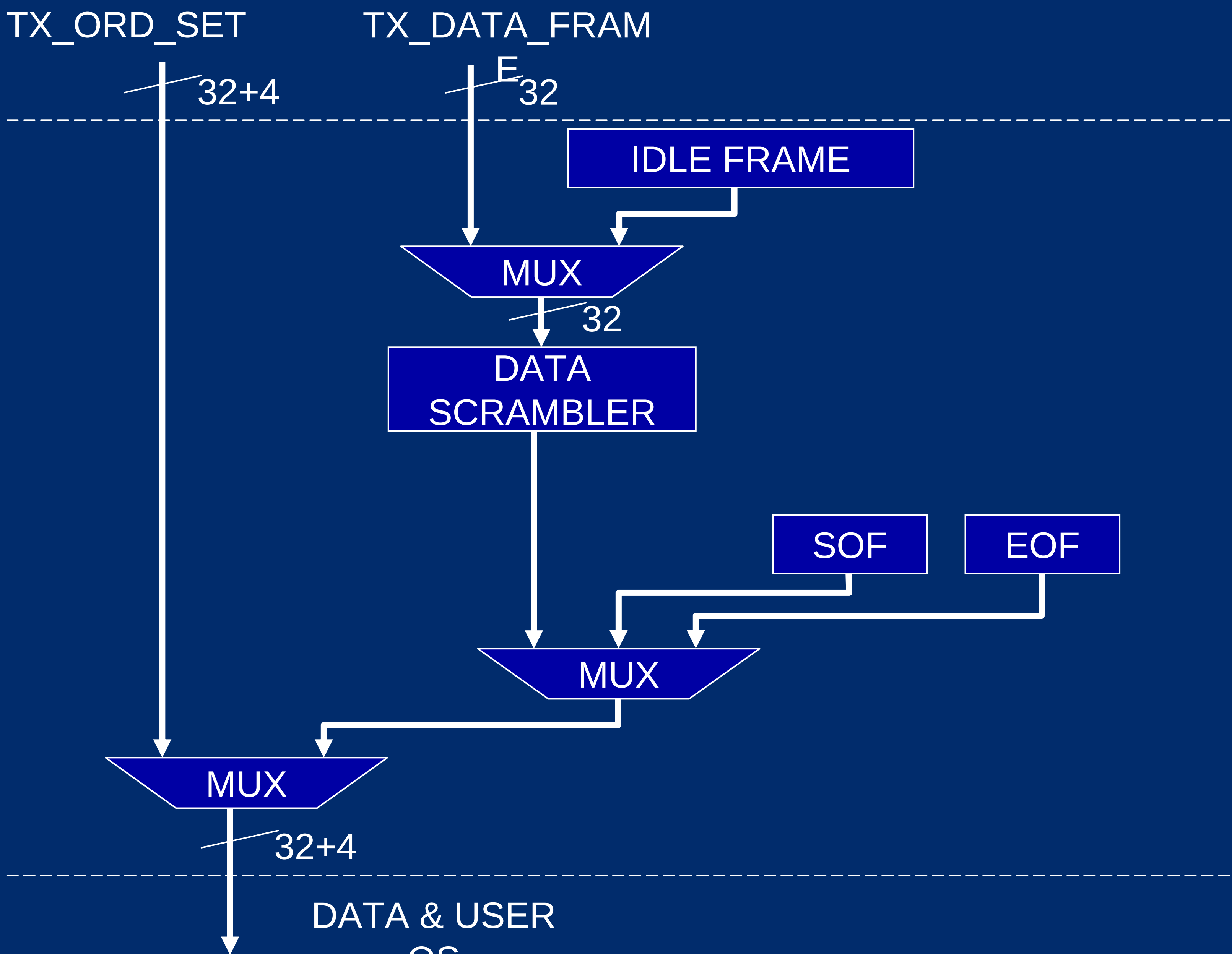
Frames

- Up to 255 32-bit data words in a data frame
- Up to 255 32-bit idle words in an idle frame
- Idle frame sent when there is no data to send
- Idle frame terminated as soon as there is more data to send

CRC

- 16-bit CRC
- Transmitter
  - CRC unit generates CRC
  - Recognises SOF, EOF
  - Computes 2-byte CRC on data between SOF and EOF
  - Inserts CRC in EOF
  - Modifies EOF to generate correct running disparity
- Receiver
  - Checks CRC

EMC Mitigation and Data Framing



Scrambler

- Spread the spectrum of data / idle frames
- By convolution
  - with a broad spectrum signal
  - i.e. a noise or random number source
- Convolution in frequency domain is multiplication in time domain.
- Multiplication of a bit sequence can be done by XOR

Scrambler

- Bit wise multiplication (XOR) of data with a sequence of random numbers produced from a scrambling polynomial
- The scrambling polynomial is

  $G(x) = X^{16} + X^{5} + X^{4} + X^{3} + 1$
- Seed for scrambler is 0xffff
- Re-seeded at the start of every new data or idle frame
- Data field of data frames and idle frames scrambled prior to transmission

Space Technology Centre
University of Dundee

De-Scrambling

- De-convolve known "noise" from received signal
- De-convolution in frequency domain is division in time domain
- Multiplication and division give the same result in bit-wise boolean algebra

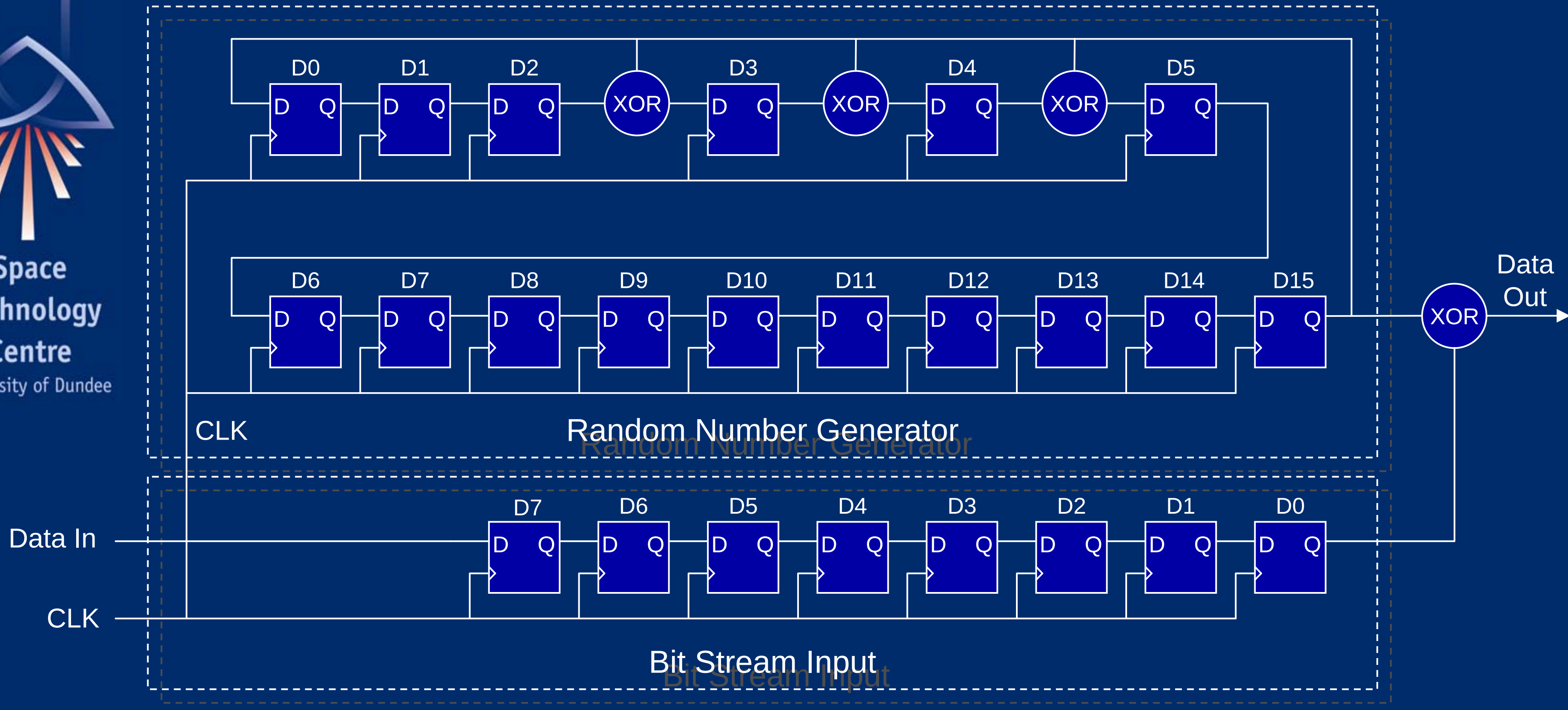
0 represents -1, 1 represents +1

-1 x -1 = +1	-1 / -1 = +1	0 XOR 0 = 0	INV = 1
-1 x +1 = -1	-1 / +1 = -1	0 XOR 1 = 1	INV = 0
+1 x -1 = -1	+1 / -1 = -1	0 XOR 1 = 1	INV =0
+1 x +1 = +1	+1 / +1 = +1	1 XOR 1 = 0	INV = 1

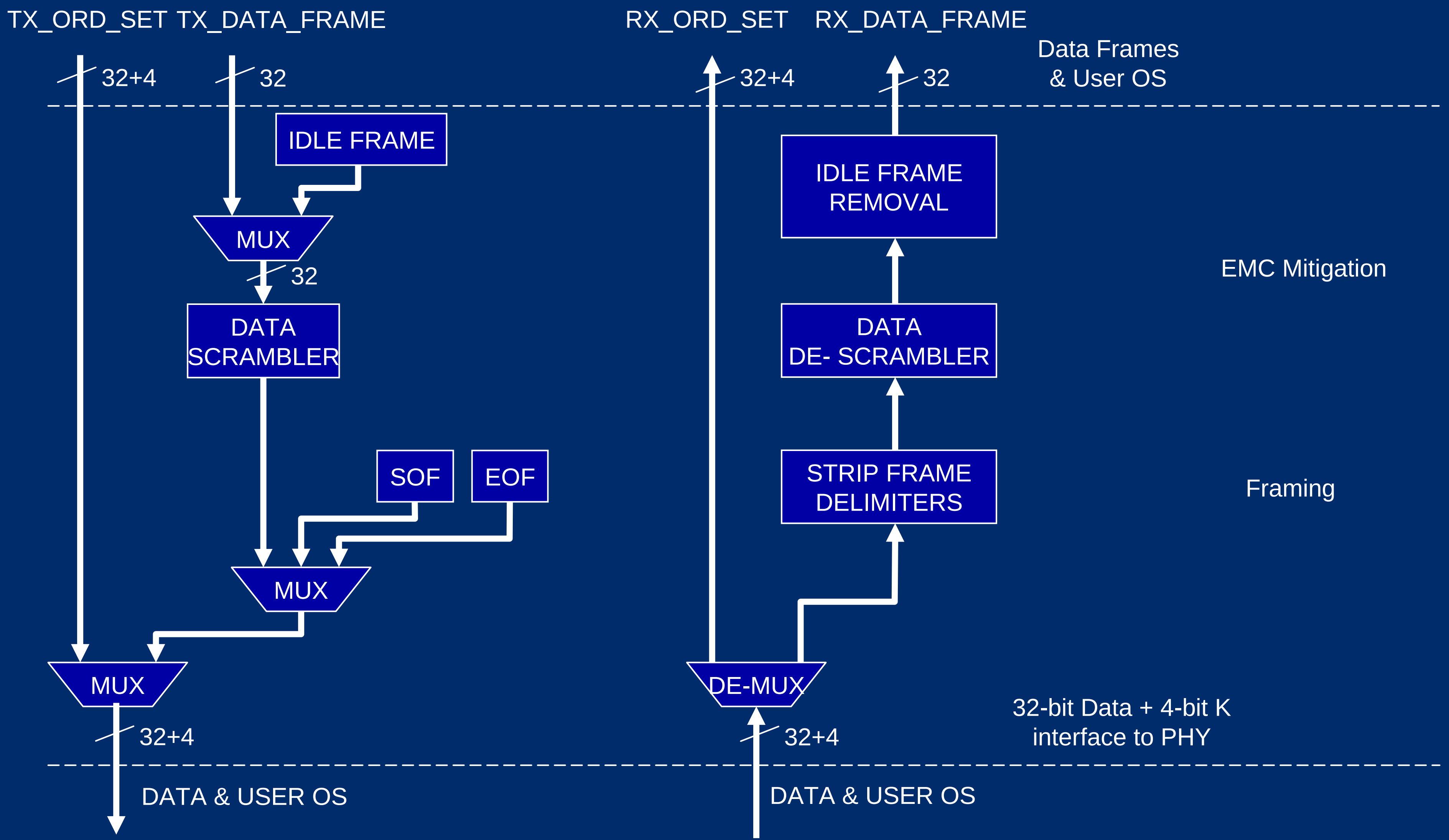
- Therefore XOR the incoming bit stream with the scrambling polynomial to recover data

Space Technology Centre
University of Dundee

Scrambler/De-Scrambler



EMC Mitigation and Data Framing



Transmit Data Frame Interface

Transmit Data Frame Interface		
Signal	**Dir**	**Function**
User_Txdata(31:0)	In	User frame data.
User_Txdata_Rdy	In	When asserted a complete frame is ready to transmit. The lowest order byte of the first User_Txdata word holds the frame length and forms part of the start of frame ordered set.
User_Txdata_Read	Out	Read user frame data into the SpaceFibre CODEC for transmission.

Space Technology Centre
University of Dundee

Transmit Ordered Set Interface

Transmit Ordered Set Interface		
Signal	**Dir**	**Function**
User_Tx_Ord_Set(31:0)	In	User ordered set data. Bits 31:24 should be set to K28.5 as the ordered set must have a comma code as the most significant byte.
User_Tx_Ord_Set_Rdy	In	When User_Tx_Ord_Set_Rdy is asserted then the User_Tx_Ord_Set data is valid. User ordered sets are transmitter by the SpaceFibre CODEC when the link initialisation state machine has connected
User_Tx_Ord_Set_Read	Out	Asserted when an ordered set has been read into the SpaceFibre CODEC for transmission.

Space Technology Centre
University of Dundee

Receive Data Frame Interface

Receive Data Frame Interface		
Signal	**Dir**	**Function**
User_Rxdata(31:0)	Out	Received frame data.
User_Rxdata_SOF	Out	When asserted the frame data on User_Rxdata is a start of frame
User_Rxdata_EOF	Out	When asserted the frame data on User_Rxdata is an end of frame
User_Rxdata_Valid	Out	When asserted the User_Rxdata, User_Rxdata_SOF and User_Rxdata_EOF outputs are valid.
User_Rx_Out_Of_Frame_Error	Out	Asserted when a character is received when not expected.
User_Frame_Length_Error	Out	Asserted when a frame is terminated with an end of frame ordered set before the complete frame length has been received.

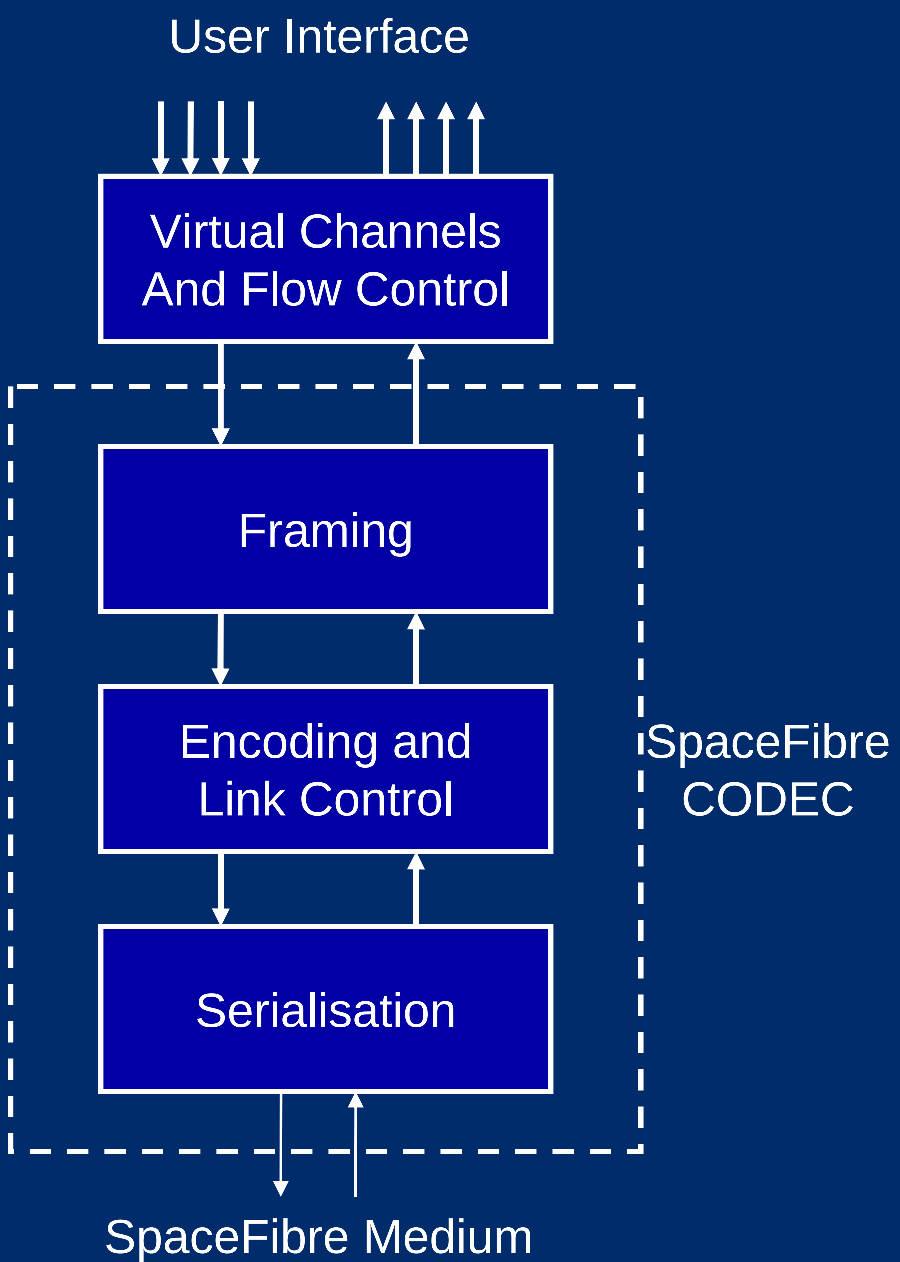
Space Technology Centre
University of Dundee

Receive Ordered Set Interface

Receive Ordered Set Interface		
Signal	**Dir**	**Function**
User_Rx_Ord_Set(31:0)	Out	Received ordered set data.
User_Rx_Ord_Set_Valid	Out	When asserted the User_Rx_Ord_Set output is valid.

Space Technology Centre
University of Dundee

Architecture Overview



Virtual Channels

- Virtual channel
  - Unidirectional data connection
  - Between pair of buffers
    - Source buffer
    - Destination buffer
- VC source buffer sends data to VC destination buffer
  - With same VC number
- Up to 256 virtual channels supported
- Used for:
  - Multiplexing data over a link
  - Quality of Service

Space Technology Centre
University of Dundee

Flow Control

- Flow control
  - From source VC buffer
  - To destination VC buffer
- Only send data frame
  - When destination VC buffer has room
- Destination VC buffer sends Flow Control OS
  - When room from one more frame
  - FCOS reserves space in destination VC buffer
- Flow Control OS contains
  - VC number
  - Sequence number

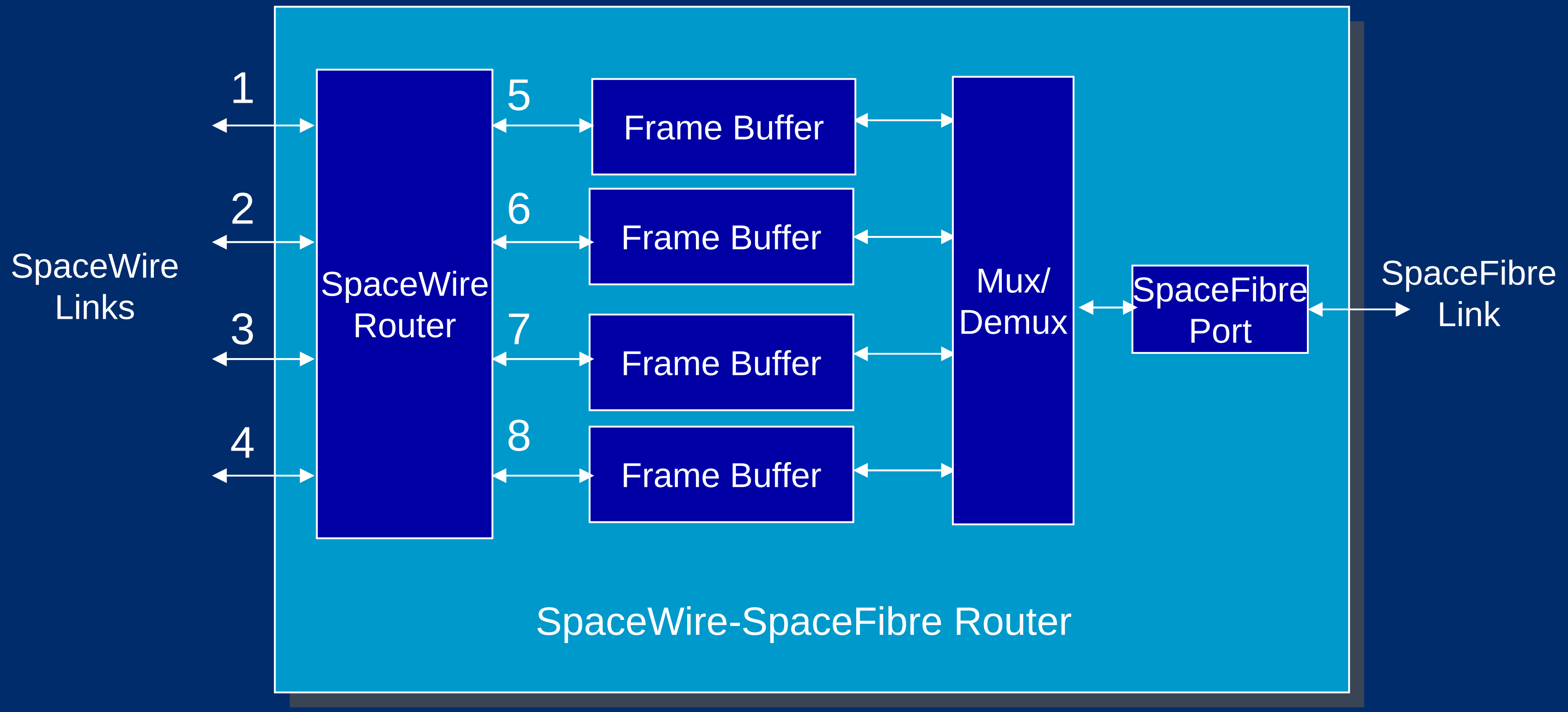
Space Technology Centre
University of Dundee

Flow Control Ordered Set

Flow Control Ordered Set		
Name	**Ordered Set**	**Function**
FCT	Comma, FCT, Sequence No., Channel No. K28.5, D0.6, Seq, Ch	Flow Control Token Indicates that the receive buffer for a specific virtual channel has room for another complete data frame. Sequence number increments for each new FCT sent related to a specific channel. Channel number identifies the virtual channel which this FCT is for.

SpaceFibre/SpaceWire Router

SpaceWire-SpaceFibre Router



SpaceWire Packet Mapping

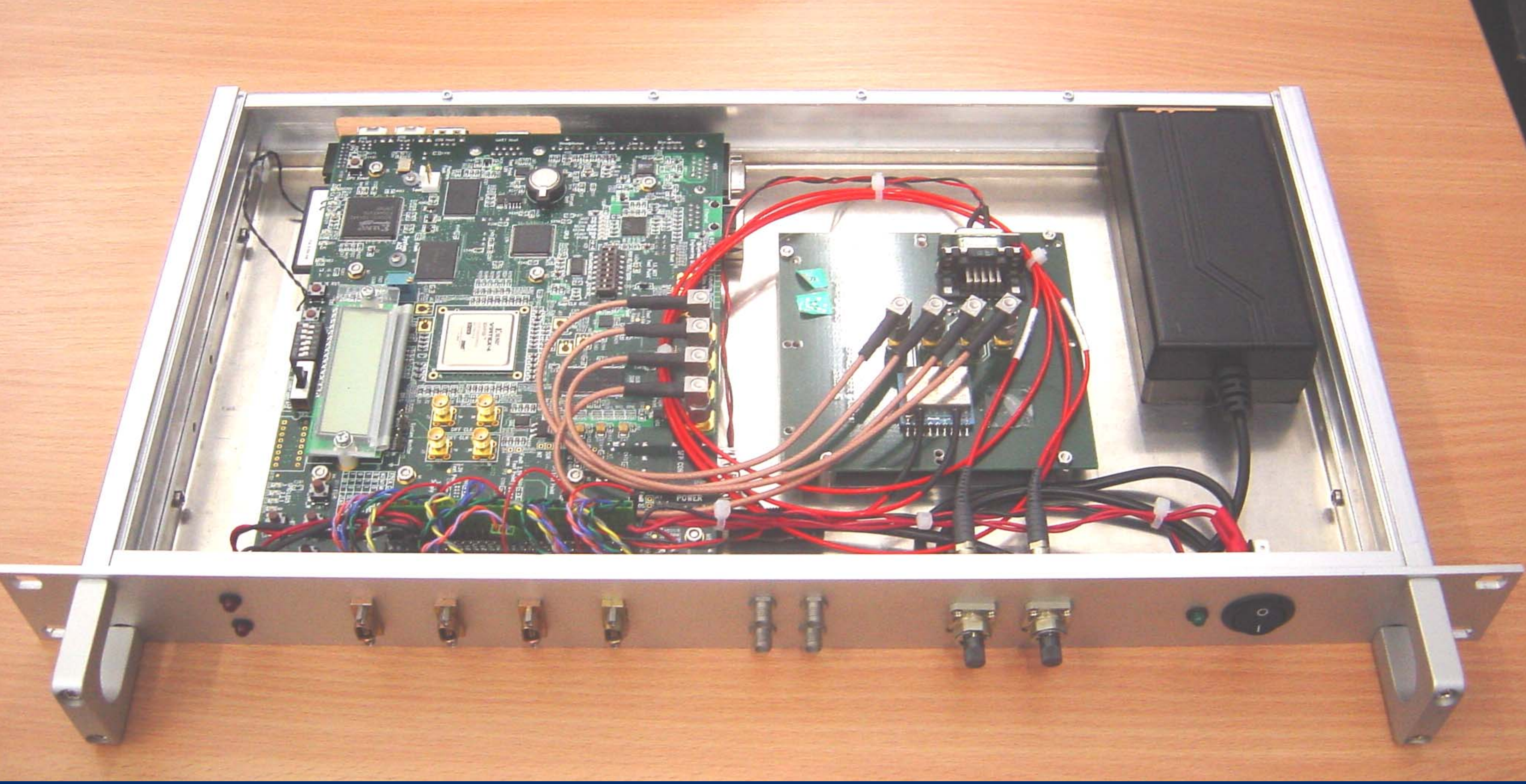
31 24	23 16	15 8	7 0
COMMA	SDF	VC	Word Count
Not Used	Not Used	Not Used FLG	Packet Count
DATA 1 MS	DATA 1	DATA 1	DATA 1 LS
...	...	...	...
DATA	DATA N	DATA N	DATA N LS
COMMA	EOF	CRC_MS	CRC_LS

- FLG
  - Start of SpW packet
  - Continuation of SpW packet
  - End of SpW packet and type
- Packet count
  - Number of SpW data characters in frame

Space Technology Centre
University of Dundee

SpaceFibre/SpaceWire Router

- Implemented using Xilinx Virtex-4
- Uses Rocket IO
  - SerDes
  - Bit Synchronisation
  - 8B/10B Encoding
  - Symbol Synchronisation
  - Receive Elastic Buffer
- Custom configuration
- Everything else built in FPGA fabric


POWER

Space
Technology
Centre
University of Dundee

Results

- Operates at 2 Gbit/s
- Link Initialisation
- Virtual Channels
- Flow Control
- Fibre optics
- SpaceWire over SpaceFibre
  - Four SpaceWire links multiplexed over SpaceFibre
  - In both directions
  - Virtual channels used to differentiate SpaceWire links
  - Transparent transfer of SpaceWire packets over SpaceFibre

SpaceFibre Requirements

- Faster than SpaceWire (> 1 Gbit/s)
  - Yes
- Further than SpaceWire (100 m)
  - Yes
- Lighter than SpaceWire
  - Yes
- As simple as SpaceWire
  - Almost
- Backwards compatible with SpaceWire
  - Yes
- Galvanically isolated
  - Yes

Space Technology Centre
University of Dundee

Conclusions

- Appropriate SpaceFibre CODEC designed
  - Meets SpaceFibre requirements
- Demonstration system built and tested
- Initial draft of SpaceFibre standard written
- Future work
  - Virtual channels and flow control
  - Quality of service
  - Power management
  - Speed switching
  - Consolidation of standard
  - VHDL implementation and test

Space Technology Centre
University of Dundee