



mipi[®]
DEVCON

MIPI C-PHYSM: Introduction

*From Basic Theory to
Practical Implementation*

*Mohamed Hafed
Introspect Technology*

Original Spark: Three Phase Encoding!

1 Unit Interval
of Data ↔ 2.285 Bits of
Information

US008472551B2

(12) **United States Patent**
Wiley

(10) **Patent No.:** US 8,472,551 B2
(45) **Date of Patent:** Jun. 25, 2013

(54) **THREE PHASE AND POLARITY ENCODED SERIAL INTERFACE**

(75) **Inventor:** George A Wiley, San Diego, CA (US)

(73) **Assignee:** QUALCOMM Incorporated, San Diego, CA (US)

(*) **Notice:** Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.
This patent is subject to a terminal disclaimer.

(21) **Appl. No.:** 13/301,454
(22) **Filed:** Nov. 21, 2011

(65) **Prior Publication Data**
US 2012/0155565 A1 Jun. 21, 2012

Related U.S. Application Data
(63) Continuation of application No. 11/712,941, filed on Mar. 2, 2007, now Pat. No. 8,064,535.

(51) **Int. Cl.** (2006.01)
H04L 25/34 (2006.01)
H04L 25/49 (2006.01)

(52) **U.S. Cl.**
USPC 375/288; 375/289; 375/292; 341/58; 341/69; 341/70

(58) **Field of Classification Search**
USPC 375/259-264, 284-288, 290-293; 341/50, 341/58, 68-70
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS
4,201,958 A 5/1980 Ahamed
5,259,002 A 11/1993 Carlstedt
5,359,595 A 10/1994 Weidie et al.

(Continued)

FOREIGN PATENT DOCUMENTS
CN 1871635 A 11/2006
JP 2002199032 A 7/2002
WO 2005094164 5/2005

OTHER PUBLICATIONS
International Search Report and Written Opinion—PCT/US2008/055566, International Search Authority—European Patent Office—Nov. 24, 2008.
Sevanto, J., "Multimedia messaging service for GPRS and UMTS", IEEE on WCNC, Sep. 1999, pp. 1422-1426, vol. 3.
Haseh, "Crosstalk Suppression Technique for Multi-Wire High-Speed F.O.I. Links," Post-Quest Dissertations and Theses, 2010; UCL A, 189 pages.
John Position, et al., "Multivire Differential Signaling," UNC-CH Department of Computer Science, Version 1.1, Aug. 6, 2003.

(Continued)

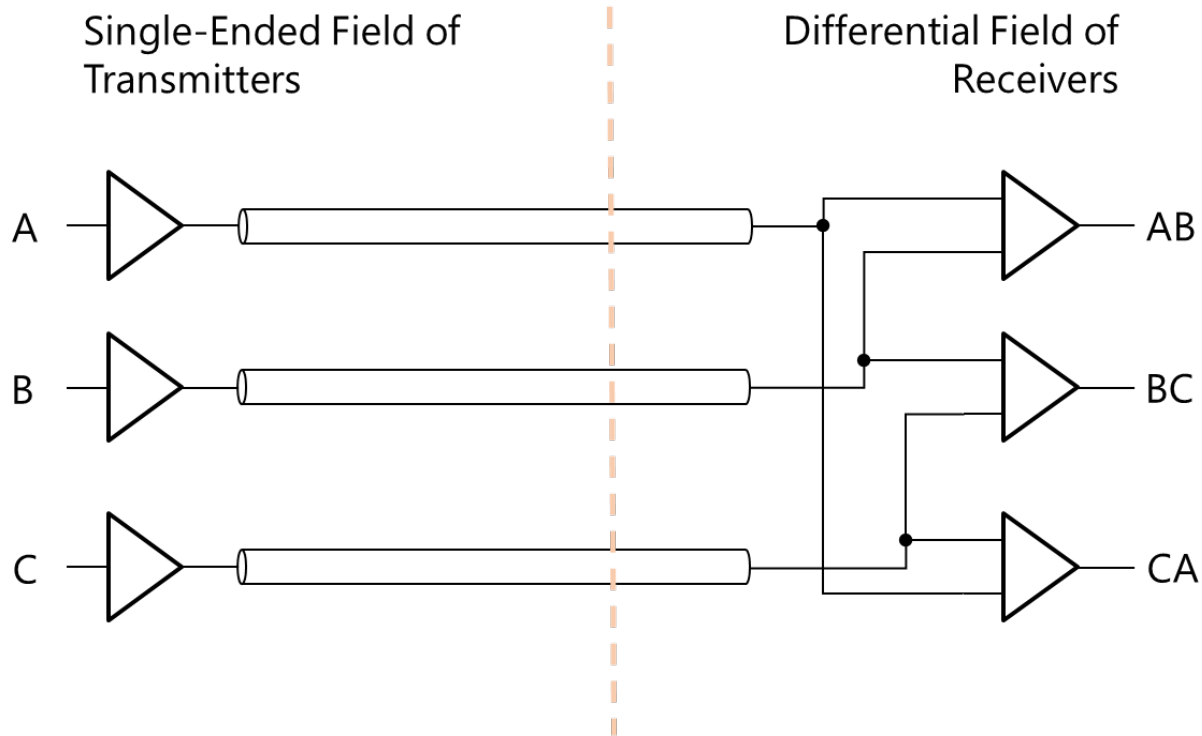
Primary Examiner — Shuwang Liu
Assistant Examiner — James M Perez
(74) **Attorney, Agent, or Firm** — Kevin T. Cheatham; Raphael Erevirih.

ABSTRACT
(57) A high speed serial interface is provided. In one aspect, the high speed serial interface uses three phase modulation for jointly encoding data and clock information. Accordingly, the need for de-skewing circuitry at the receiving end of the interface is eliminated, resulting in reduced link start-up time and improved link efficiency and power consumption. In one embodiment, the high speed serial interface uses fewer signal conductors than conventional systems having separate conductors for data and clock information. In another embodiment, the serial interface allows for data to be transmitted at any speed without the receiving end having prior knowledge of the transmission data rate. In another aspect, the high speed serial interface uses polarity encoded three phase modulation for jointly encoding data and clock information. This further increases the link capacity of the serial interface by allowing for more than one bit to be transmitted in any single baud interval.

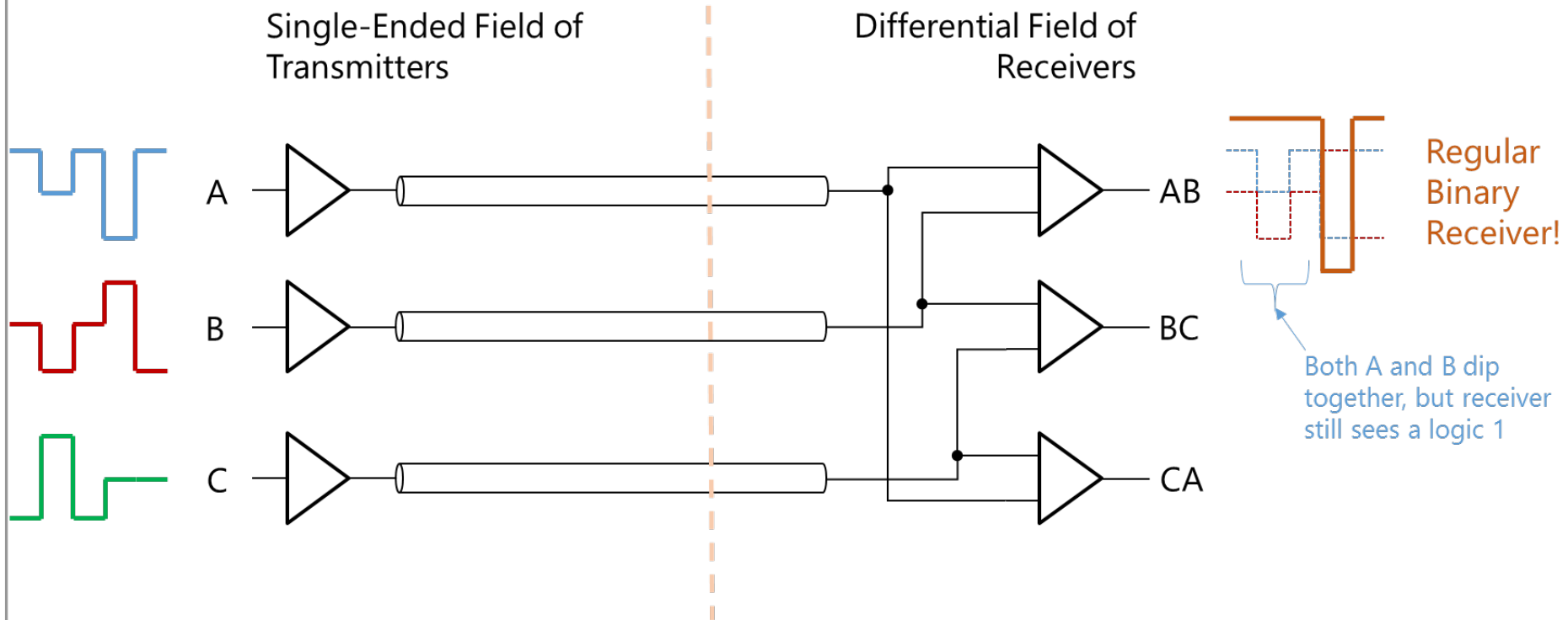
39 Claims, 15 Drawing Sheets

George Wiley, Qualcomm

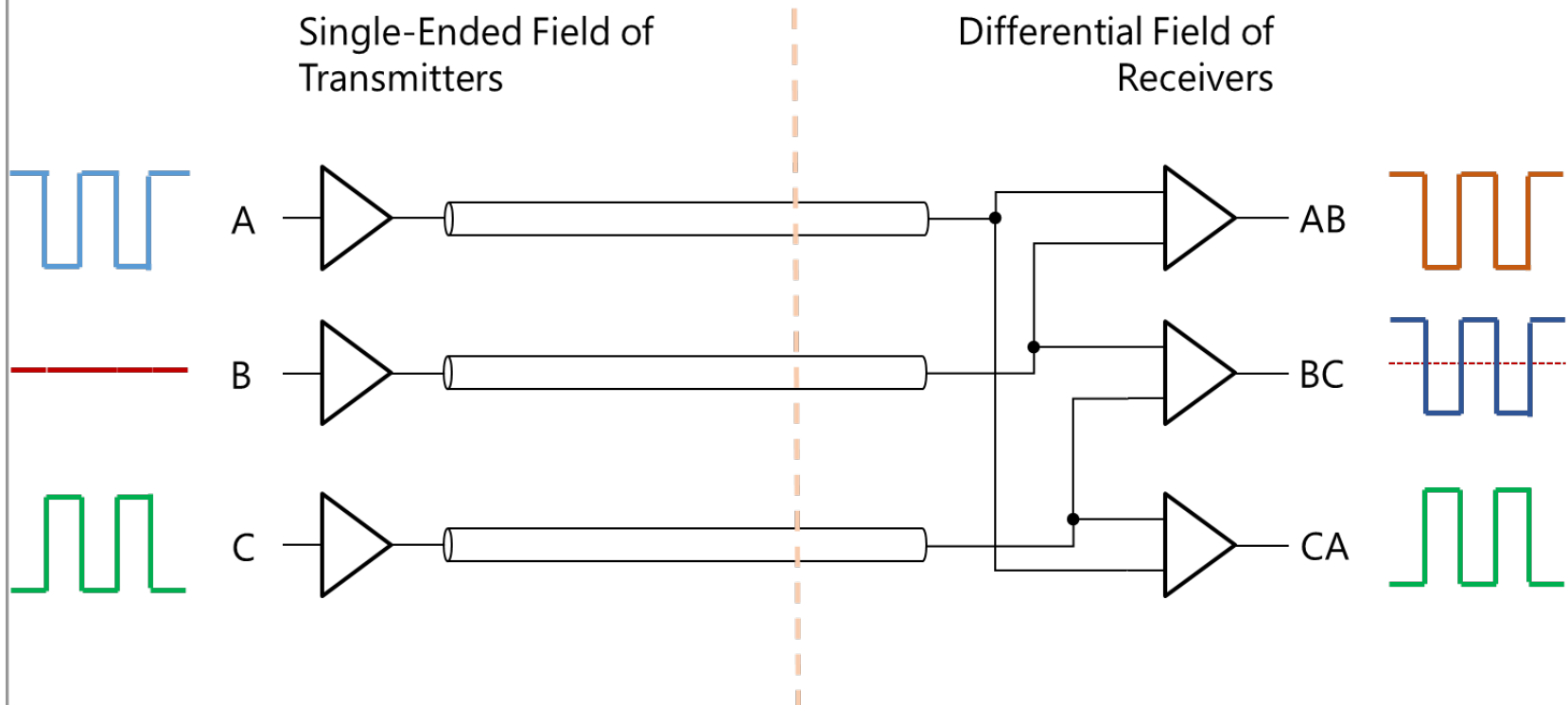
Basic Concept of Three Phase Encoding



Three Voltage Levels Per Wire Ensure Proper Differential Reception



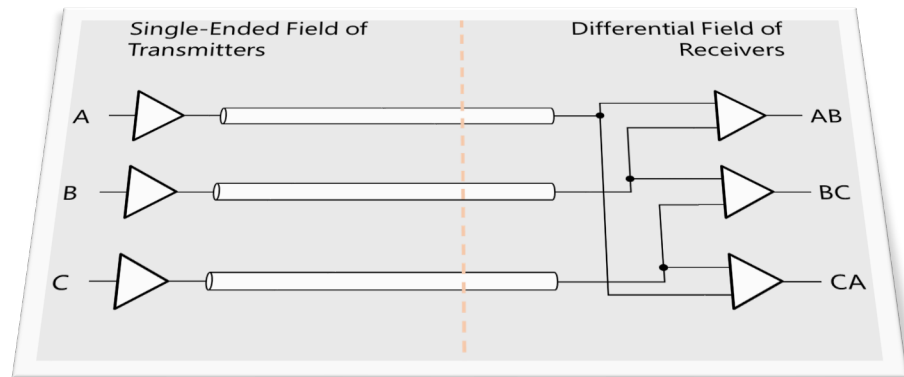
Always-Toggle Design Allows for Simple Clock Recovery (100% Transition Density)



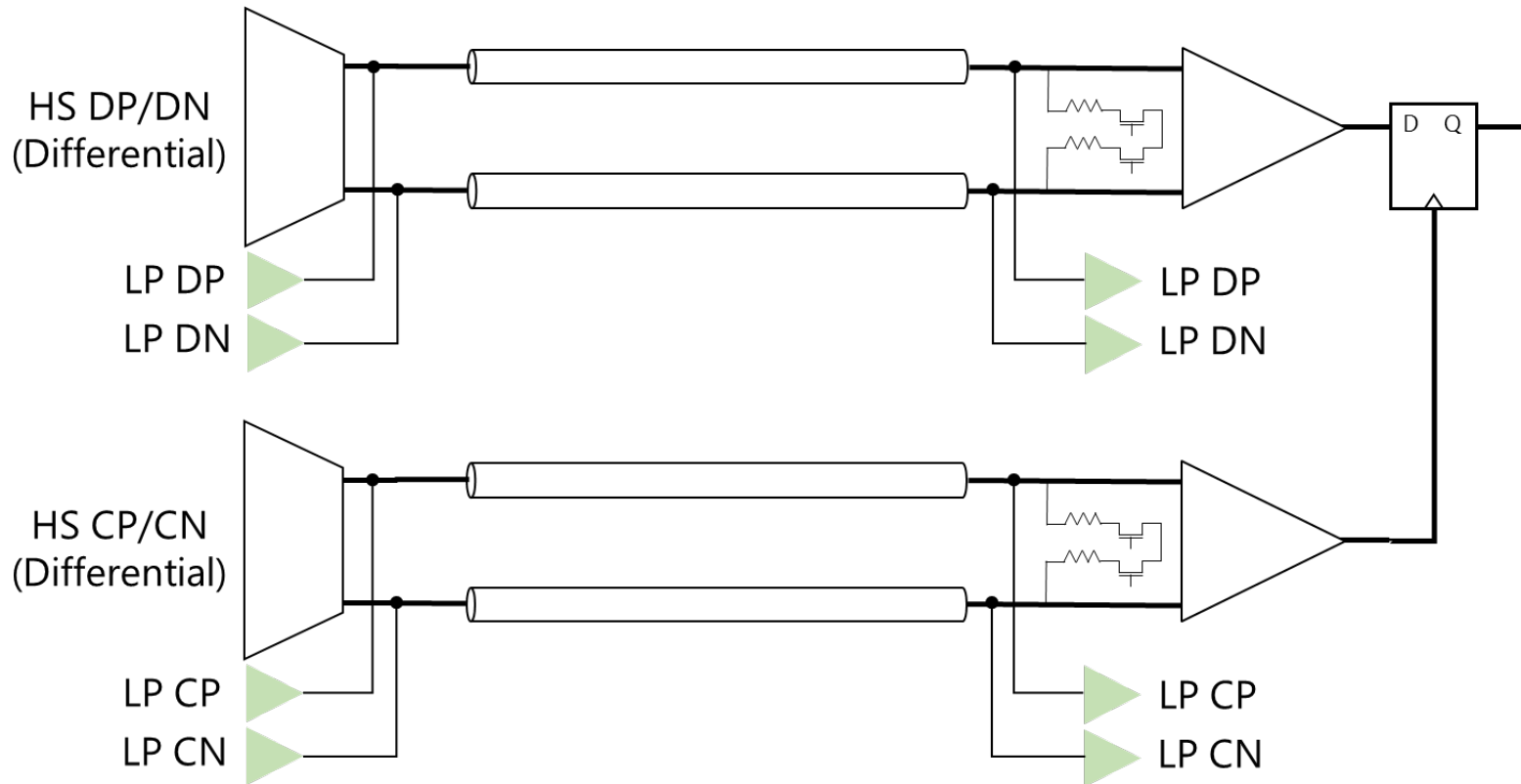
Key Takeaways

Three-level single-ended signaling

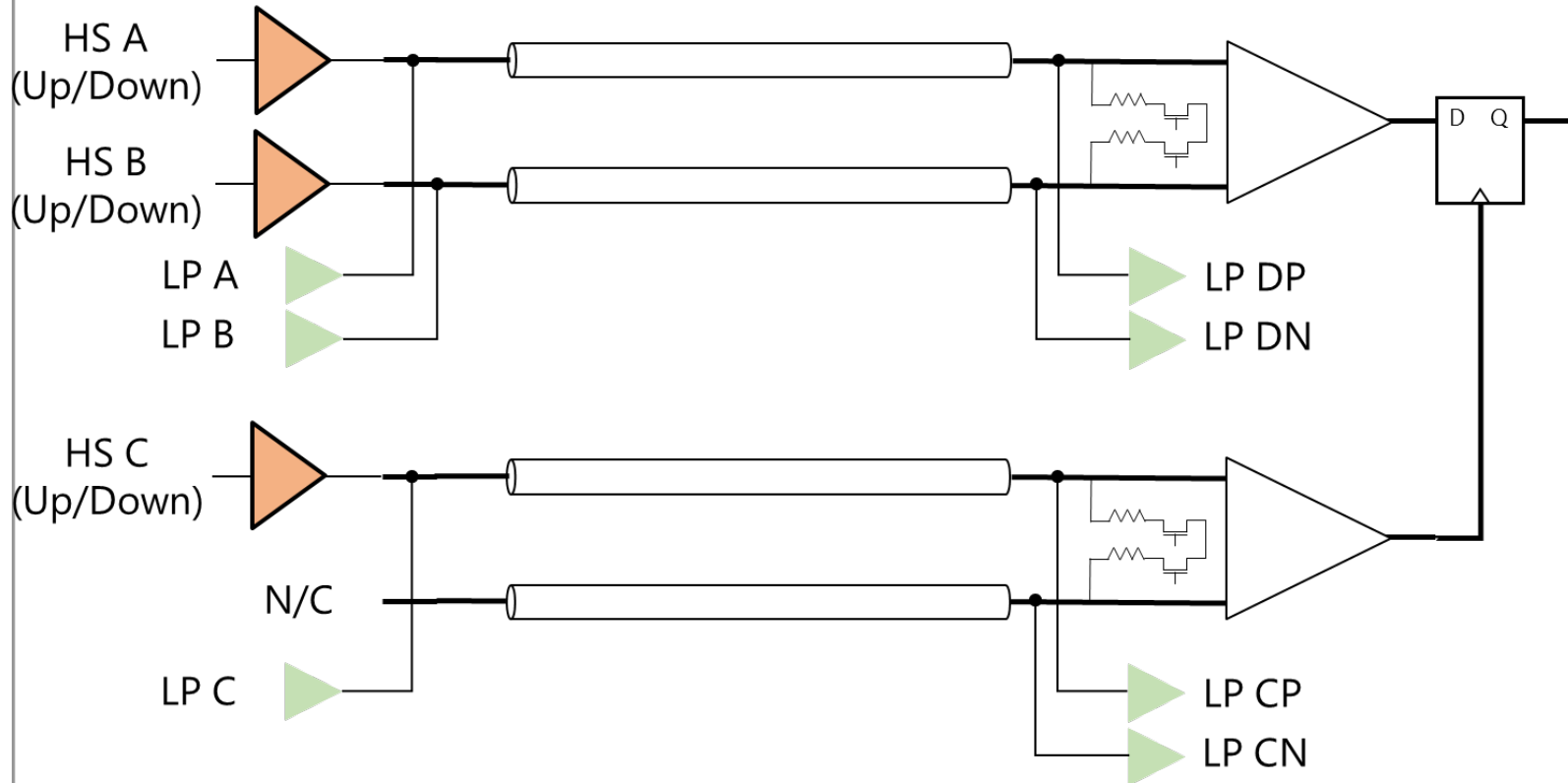
Non-deterministic transitions based on self-clocked mapping and encoding algorithm



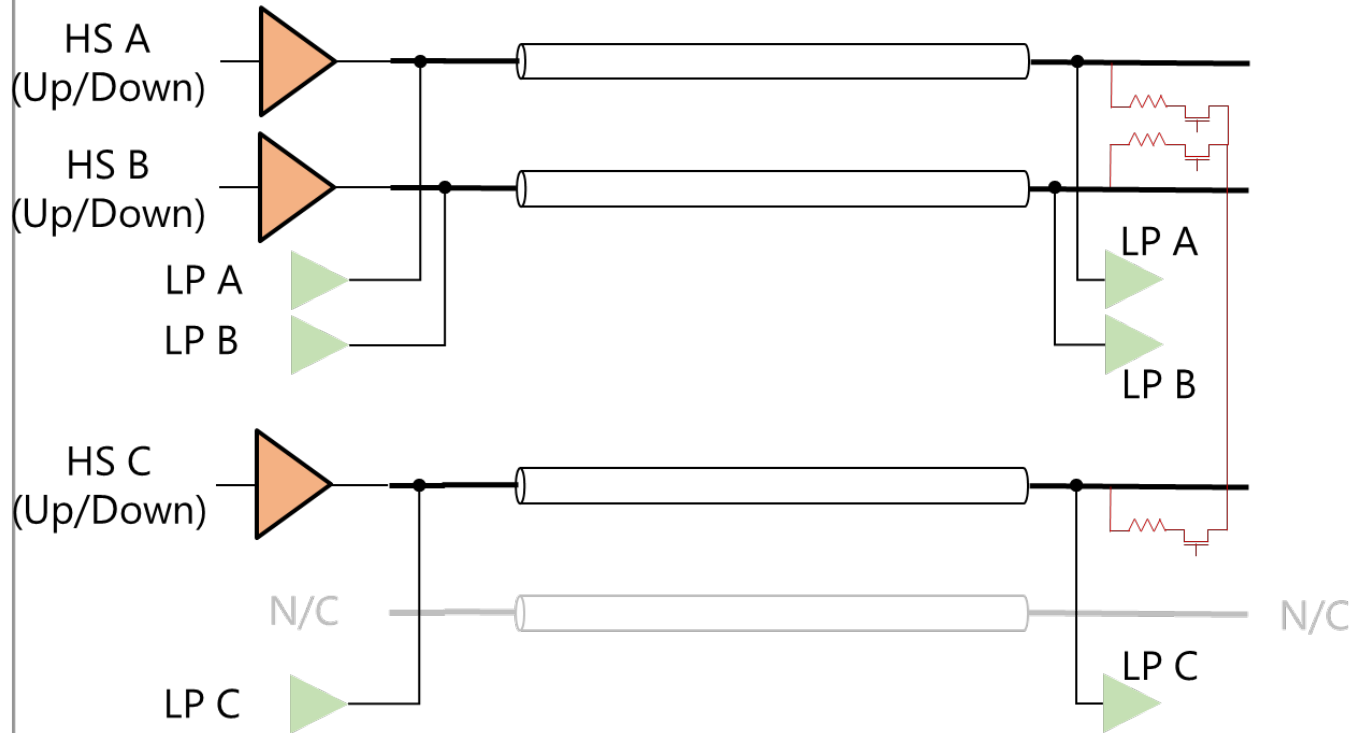
Evolution from D-PHY (1 Lane, 4 Wires)



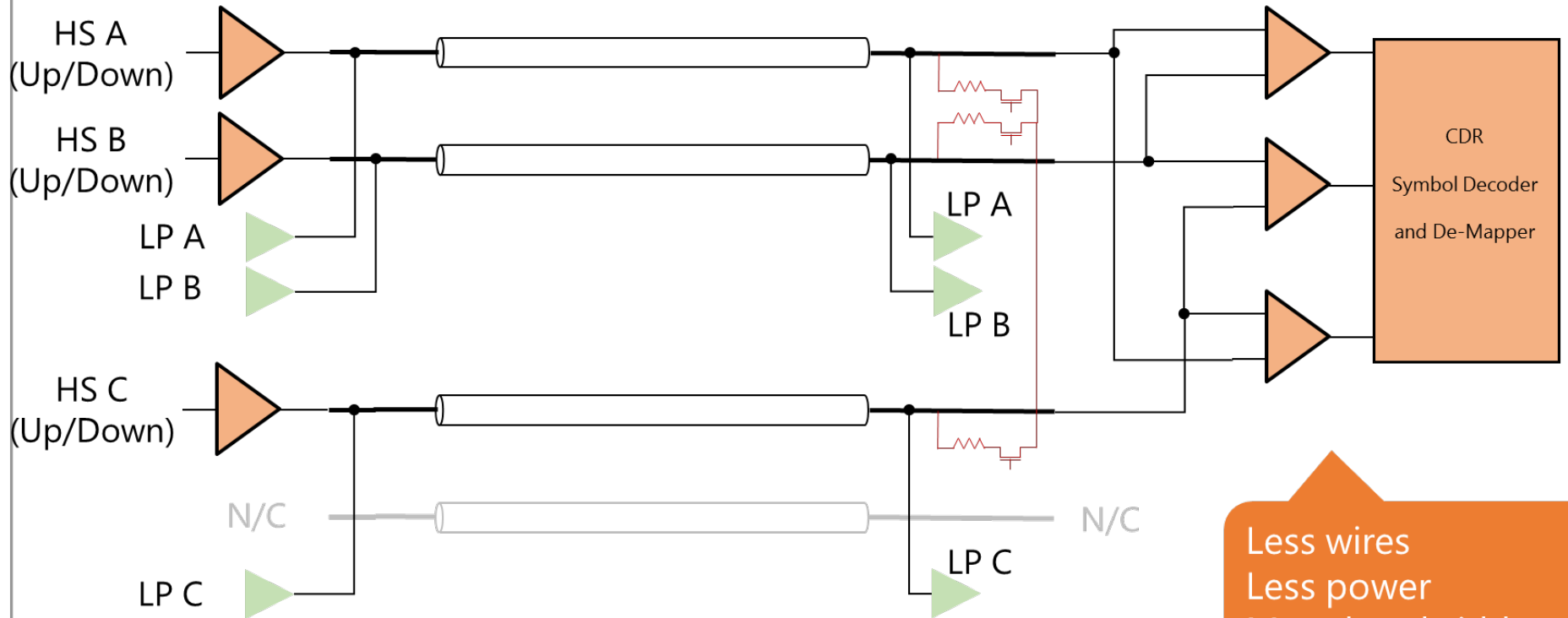
Evolution from D-PHY (1 Lane, 4 Wires)



Evolution from D-PHY (1 Lane, 4 Wires)



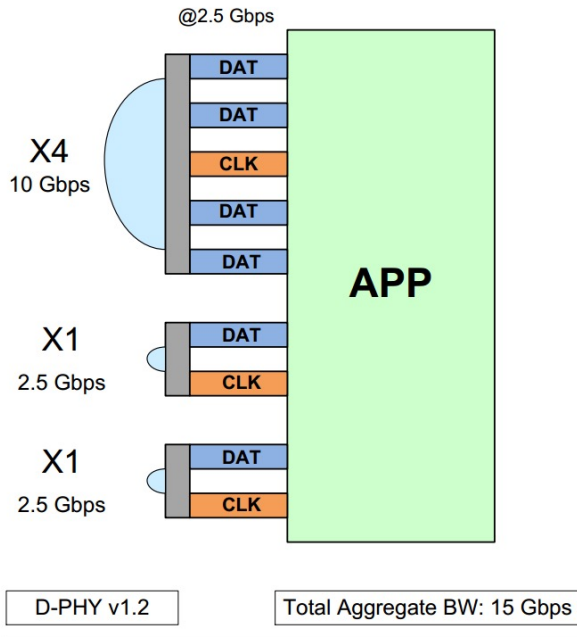
Evolution from D-PHY (1 Lane, 4 Wires)



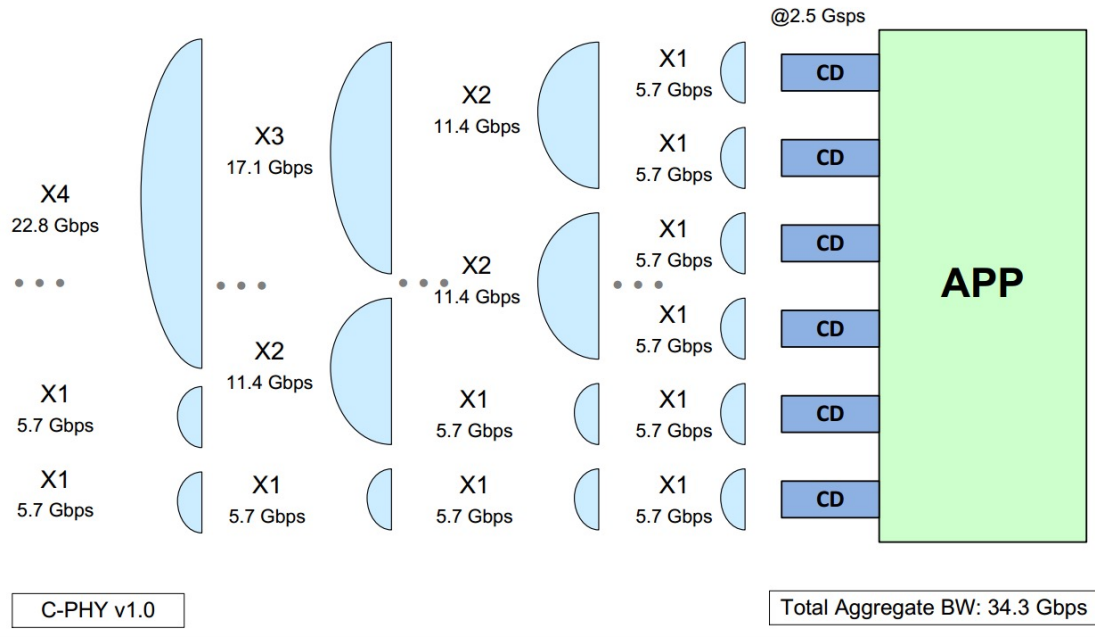
Less wires
Less power
More bandwidth

Architecturally Flexible

18 pin Forwarded Sync Clock SoC
Limited Fixed Configurations



18 pin Embedded Clock and Data SoC
All Configurations Supported



Source: MIPI Alliance



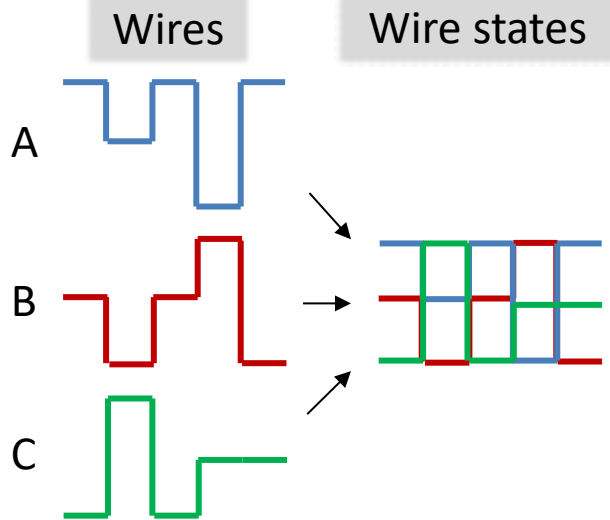
mipi[®]
DEVCON

Mapping and Encoding

C-PHY Data Types

ANALOG

DIGITAL



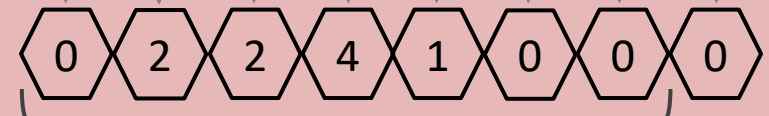
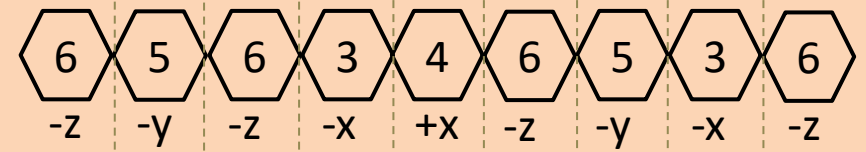
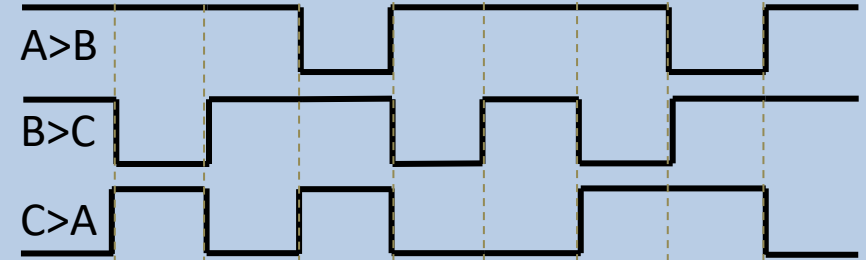
- 3 wires per lane
- 3-level wires (LOW, MID, HIGH)
- Every unit interval must contain LOW, MID, and HIGH wires
- No two consecutive identical wire states

Wire differential

Wire States (3 bits)

Symbols (3 bits)

Integers (16 bits)



7-symbol to 16-bit mapping

0x7290

Wire States

- A wire state is the collection of A, B, and C
- 6 possible wire states

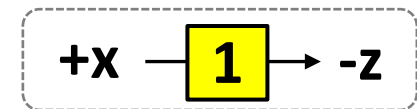
ANALOG			DIGITAL (3 bits)			Wire state name
A	B	C	A>B	B>C	C>A	
HIGH	LOW	MID	1	0	0	+x
LOW	HIGH	MID	0	1	1	-x
MID	HIGH	LOW	0	1	0	+y
MID	LOW	HIGH	1	0	1	-y
LOW	MID	HIGH	0	0	1	+z
HIGH	MID	LOW	1	1	0	-z

Symbols: Now We're Transmitting!

- A symbol represents a transition between two wire states
- 5 possible symbols

	Symbol (3 bits)		
	Flip	Rotate	Polarity
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	DC	DC

Example:



Flip	
0	-
1	Same letter, toggle sign.

Rotate	
0	Decr. letter
1	Incr. letter

Polarity	
0	-
1	Toggle sign

Mapping 7 Symbols \longleftrightarrow 16-bit Integers

- C-PHY defines a mapping between 7-symbol words and 16-bit integers

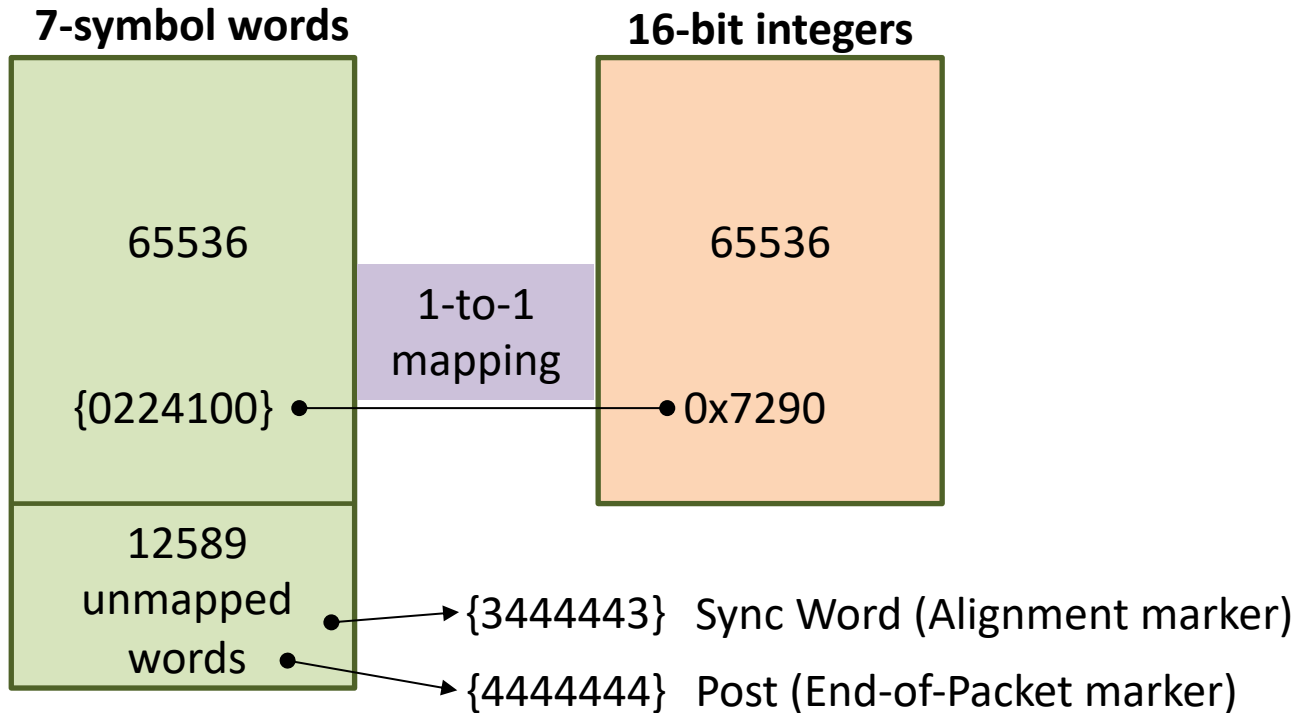
Number of **7-symbol words**:

$$5^7 = 78125$$



Number of **16-bit integers**:

$$2^{16} = 65536$$



Well Defined Algorithms from MIPI Alliance

Integer range	Flip bits	Integer	Rotate and Polarity bits
0xFC00-0xFFFF:	Flip=[1,0,1,0,0,0,0]	Integer=[1, 1, 1,	1, 1, 1, ro5, po5, ro3, po3, ro2, po2, ro1, po1, ro0, po0]
0xF800-0xFBFF:	Flip=[0,1,1,0,0,0,0]	Integer=[1, 1, 1,	1, 1, 0, ro6, po6, ro3, po3, ro2, po2, ro1, po1, ro0, po0]
0xF400-0xF7FF:	Flip=[1,0,0,1,0,0,0]	Integer=[1, 1, 1,	1, 0, 1, ro5, po5, ro4, po4, ro2, po2, ro1, po1, ro0, po0]
0xF000-0xF3FF:	Flip=[0,1,0,1,0,0,0]	Integer=[1, 1, 1,	1, 0, 0, ro6, po6, ro4, po4, ro2, po2, ro1, po1, ro0, po0]
0xEC00-0xEFFF:	Flip=[0,0,1,1,0,0,0]	Integer=[1, 1, 1,	0, 1, 1, ro6, po6, ro5, po5, ro2, po2, ro1, po1, ro0, po0]
0xE800-0xEBFF:	Flip=[1,0,0,0,1,0,0]	Integer=[1, 1, 1,	0, 1, 0, ro5, po5, ro4, po4, ro3, po3, ro1, po1, ro0, po0]
0xE400-0xE7FF:	Flip=[0,1,0,0,1,0,0]	Integer=[1, 1, 1,	0, 0, 1, ro6, po6, ro4, po4, ro3, po3, ro1, po1, ro0, po0]
0xE000-0xE3FF:	Flip=[0,0,1,0,1,0,0]	Integer=[1, 1, 1,	0, 0, 0, ro6, po6, ro5, po5, ro3, po3, ro1, po1, ro0, po0]
0xDC00-0xDFFF:	Flip=[0,0,0,1,1,0,0]	Integer=[1, 1, 0,	1, 1, 1, ro6, po6, ro5, po5, ro4, po4, ro1, po1, ro0, po0]
0xD800-0xDBFF:	Flip=[1,0,0,0,0,1,0]	Integer=[1, 1, 0,	1, 1, 0, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro0, po0]
0xD400-0xD7FF:	Flip=[0,1,0,0,0,1,0]	Integer=[1, 1, 0,	1, 0, 1, ro6, po6, ro4, po4, ro3, po3, ro2, po2, ro0, po0]
0xD000-0xD3FF:	Flip=[0,0,1,0,0,1,0]	Integer=[1, 1, 0,	1, 0, 0, ro6, po6, ro5, po5, ro3, po3, ro2, po2, ro0, po0]
0xCC00-0xCFFF:	Flip=[0,0,0,1,0,1,0]	Integer=[1, 1, 0,	0, 1, 1, ro6, po6, ro5, po5, ro4, po4, ro2, po2, ro0, po0]
0xC800-0xCBFF:	Flip=[0,0,0,0,1,1,0]	Integer=[1, 1, 0,	0, 1, 0, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro1, po1]
0xC400-0xC7FF:	Flip=[1,0,0,0,0,0,1]	Integer=[1, 1, 0,	0, 0, 1, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro1, po1]
0xC000-0xC3FF:	Flip=[0,1,0,0,0,0,1]	Integer=[1, 1, 0,	0, 0, 0, ro6, po6, ro4, po4, ro3, po3, ro2, po2, ro1, po1]
0xBC00-0xBFFF:	Flip=[0,0,1,0,0,0,1]	Integer=[1, 0, 1,	1, 1, 1, ro6, po6, ro5, po5, ro3, po3, ro2, po2, ro1, po1]
0xB800-0xBBFF:	Flip=[0,0,0,1,0,0,1]	Integer=[1, 0, 1,	1, 1, 0, ro6, po6, ro5, po5, ro4, po4, ro2, po2, ro1, po1]
0xB400-0xB7FF:	Flip=[0,0,0,0,1,0,1]	Integer=[1, 0, 1,	1, 0, 1, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro1, po1]
0xB000-0xB3FF:	Flip=[0,0,0,0,0,1,1]	Integer=[1, 0, 1,	1, 0, 0, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro2, po2]
0xA000-0xAFFF:	Flip=[1,0,0,0,0,0,0]	Integer=[1, 0, 1,	0, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro1, po1, ro0, po0]
0x9000-0x9FFF:	Flip=[0,1,0,0,0,0,0]	Integer=[1, 0, 0,	1, ro6, po6, ro4, po4, ro3, po3, ro2, po2, ro1, po1, ro0, po0]
0x8000-0x8FFF:	Flip=[0,0,1,0,0,0,0]	Integer=[1, 0, 0,	0, ro6, po6, ro5, po5, ro3, po3, ro2, po2, ro1, po1, ro0, po0]
0x7000-0x7FFF:	Flip=[0,0,0,1,0,0,0]	Integer=[0, 1, 1,	1, ro6, po6, ro5, po5, ro4, po4, ro2, po2, ro1, po1, ro0, po0]
0x6000-0x6FFF:	Flip=[0,0,0,0,1,0,0]	Integer=[0, 1, 1,	0, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro1, po1, ro0, po0]
0x5000-0x5FFF:	Flip=[0,0,0,0,0,1,0]	Integer=[0, 1, 0,	1, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro0, po0]
0x4000-0x4FFF:	Flip=[0,0,0,0,0,0,1]	Integer=[0, 1, 0,	0, ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro1, po1]
0x0000-0x3FFF:	Flip=[0,0,0,0,0,0,0]	Integer=[0, 0,	ro6, po6, ro5, po5, ro4, po4, ro3, po3, ro2, po2, ro1, po1, ro0, po0]

“Don’t
Even Worry
About It”



Eco-System Is Developed for Tools

Three-Phase
Signals



```

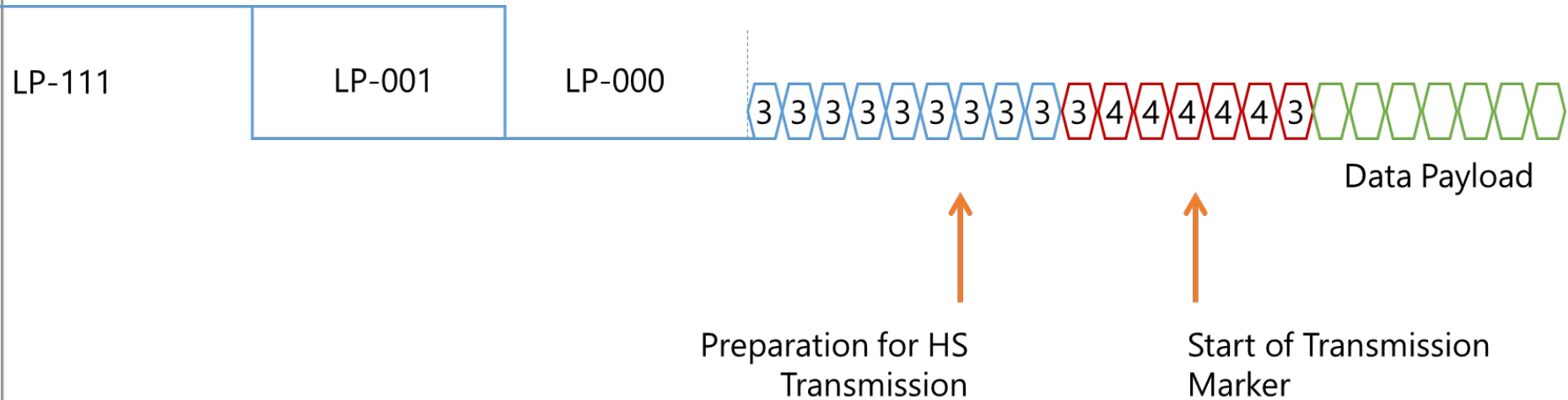
MipiCphyDataCaptureViewer
Lane 1  Lane 2  Lane 3  Lane 4 
Bits: 3612 : 3701 <<< < > >>>
Captured: Wires
wireAB: 000000000111000111000111000111000111001010110101011001010110001010000100101100100000
wireBC: 000000001100011100011100011100011100011100010101001110110100100011010111011001010111010110
wireCA: 0000000000011100011100011100011100011100101011011101010001101100101101110001010111011
-----
wireState: 000000002645132645132645132645132645132645134343453635365214241563124363123512425632531231
symbol: 8888888888333333333333333333333333333333333333333333333333333333334444430022000422200122100402101230024223401011
data (dec):
data (hex):

```



Decoded Data

Anatomy of a Packet Transmission

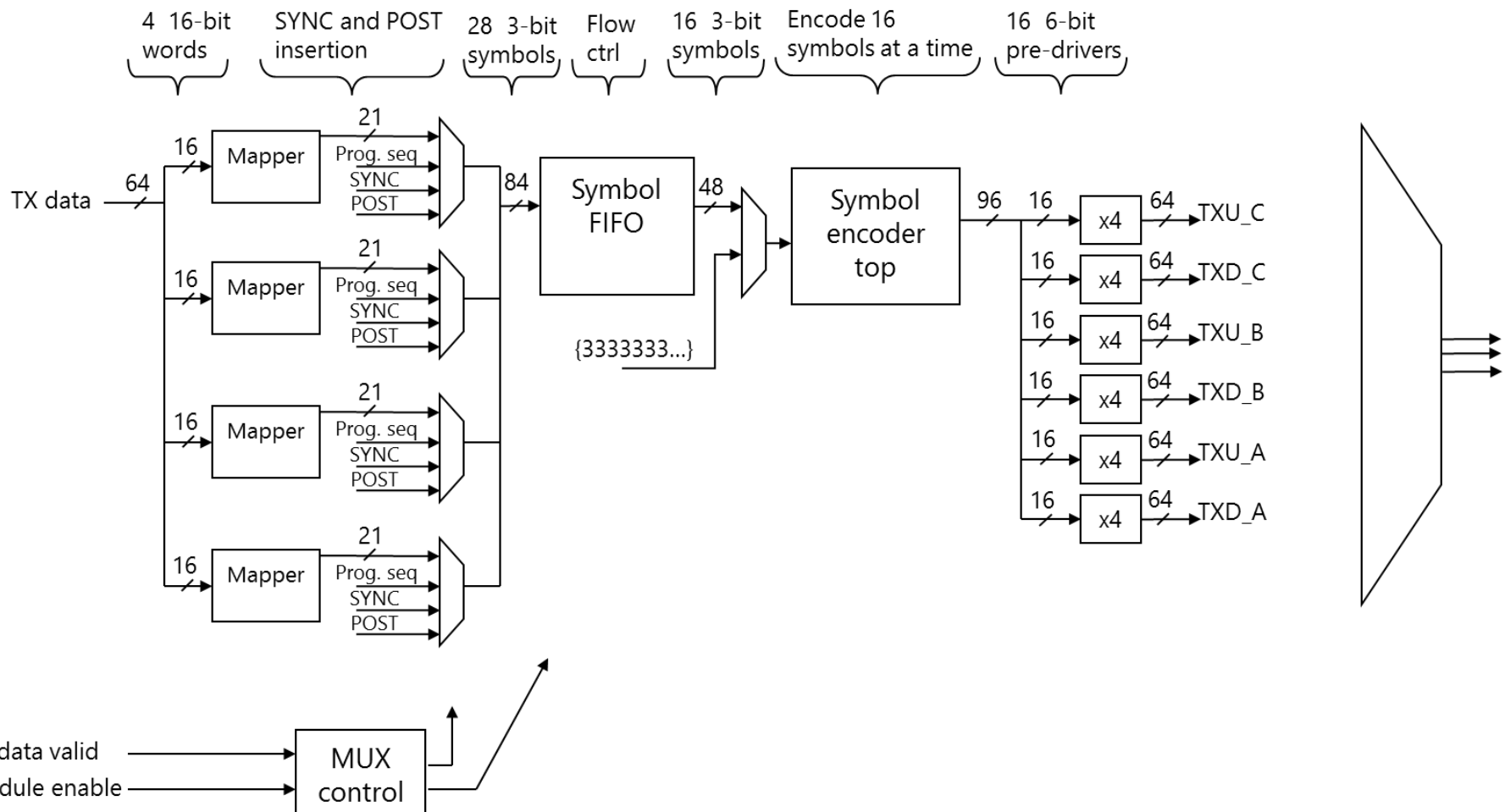




mipi[®]
DEVCON

**Practical
Experiences**

Tx: Both Mapping and Encoding Before Serializer

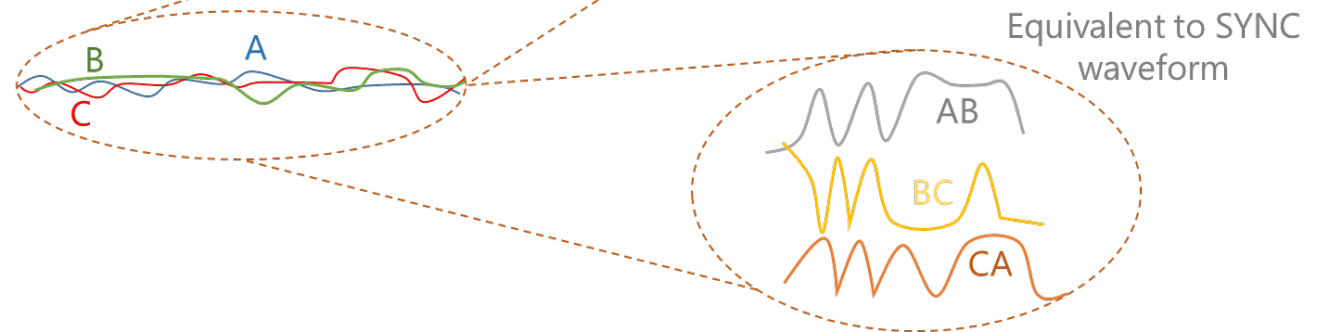
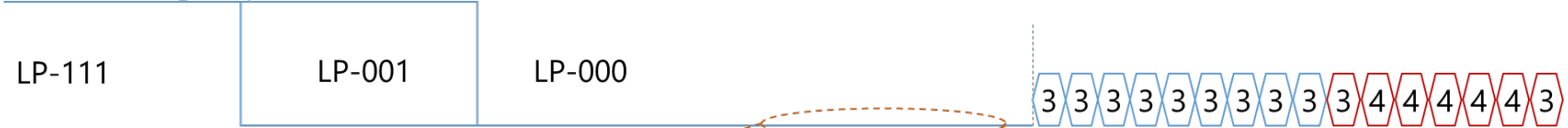


Rx: Avoiding False Sync Detection (Problem Statement)

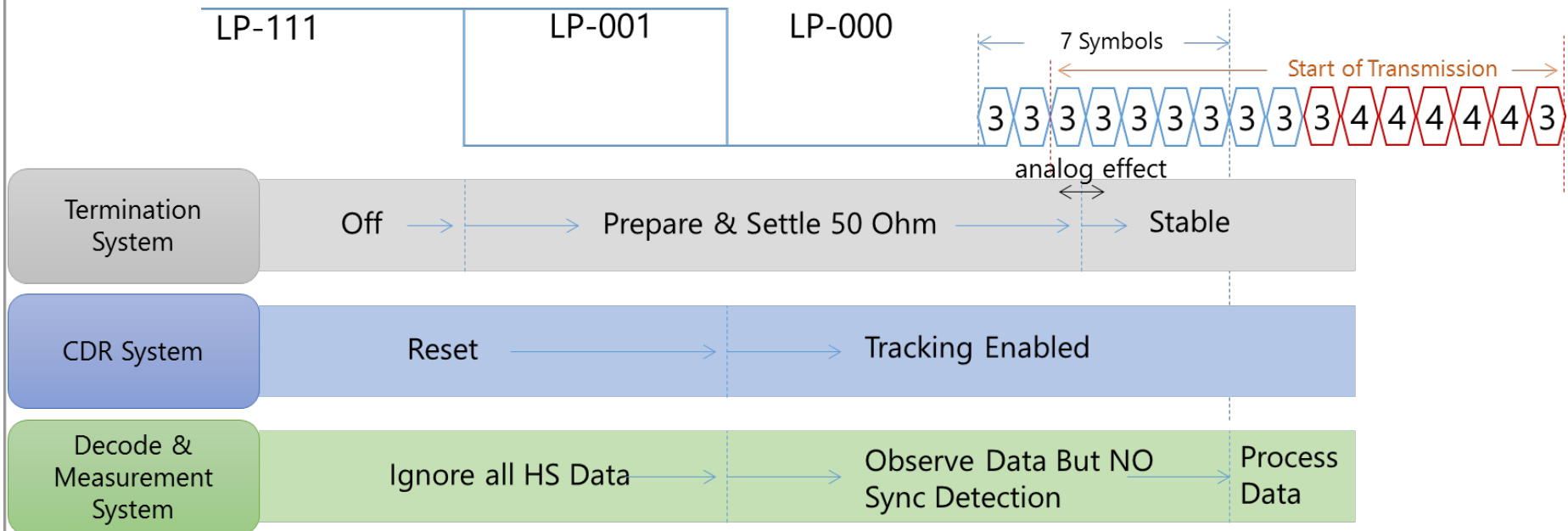
Tx With Short Prepare



Tx With Long Prepare



Rx: Avoiding False Sync Detection (Solution)

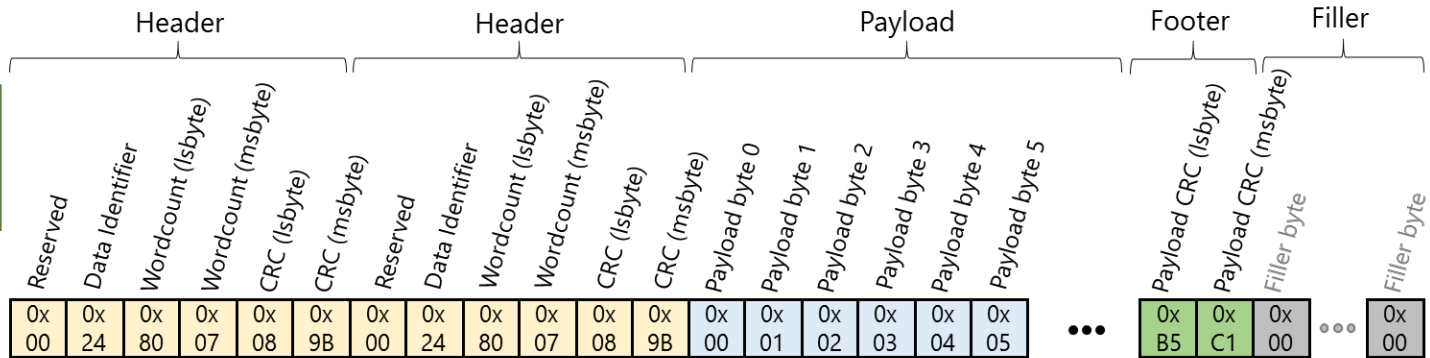


Detect SYNC with Pre-End as Marker for Start of Transmission

CSI-2 Long Packets in C-PHY

Build packet as list of bytes, similar to DPHY

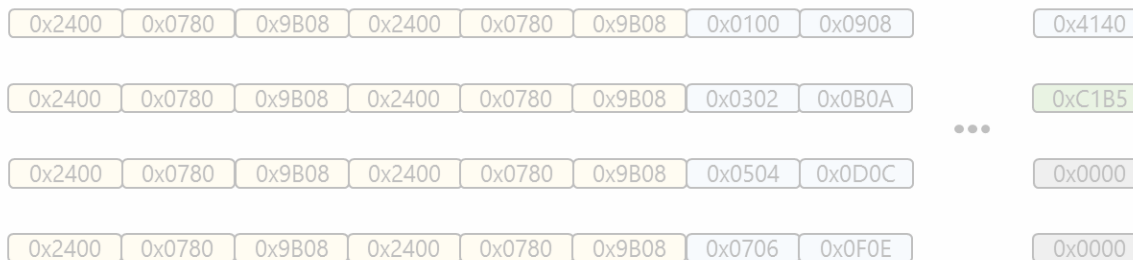
Bytes



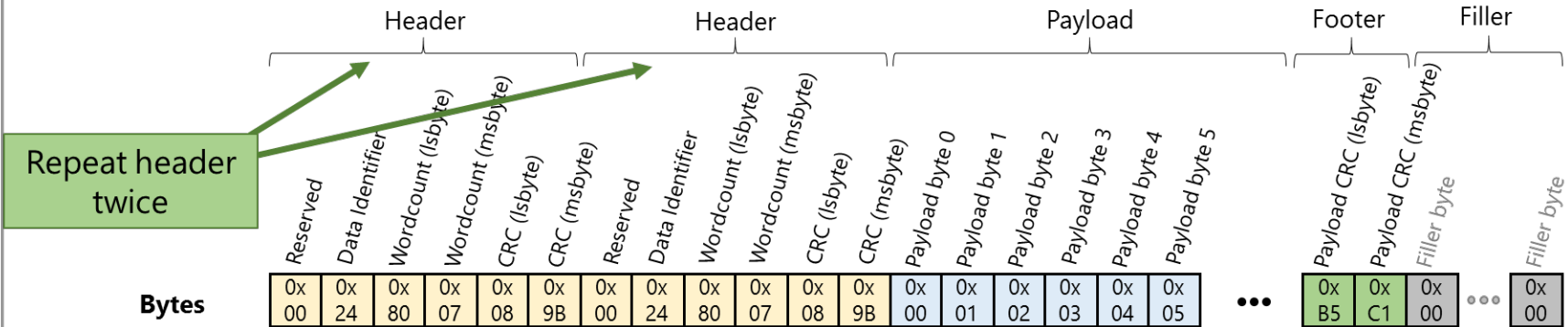
Integers - 1 lane



Integers - 4 lanes distributed



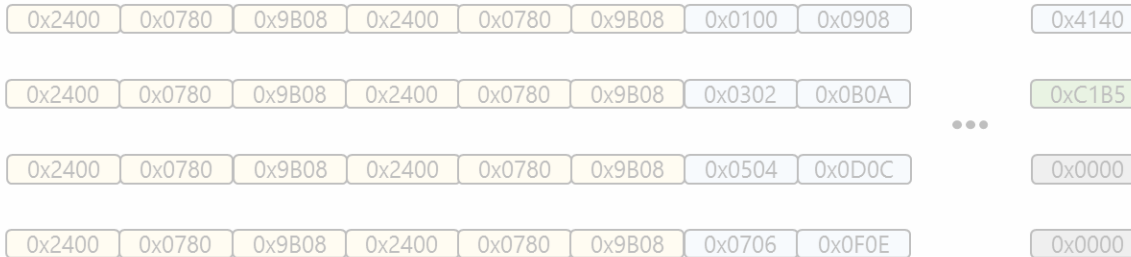
CSI-2 Long Packets in C-PHY



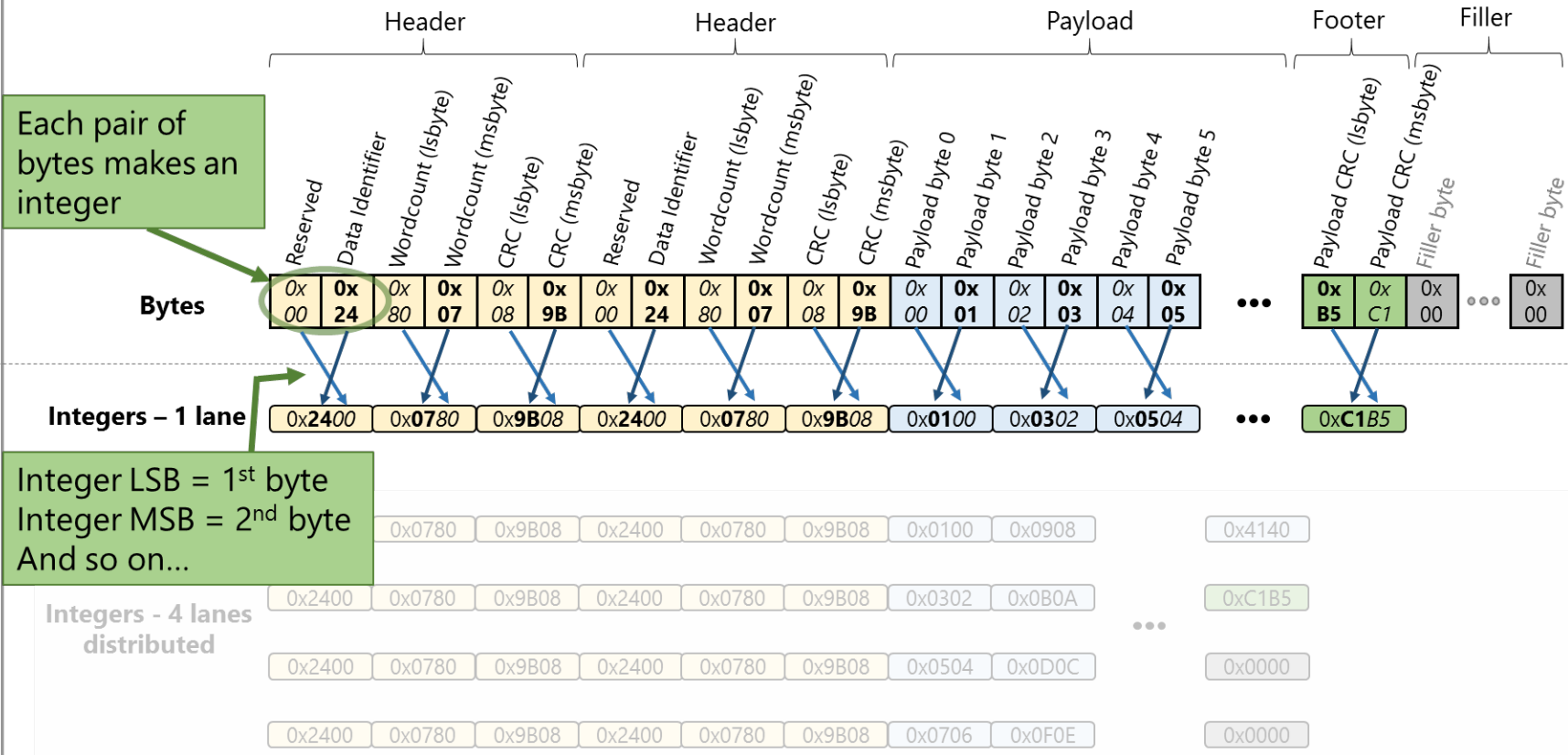
Integers – 1 lane



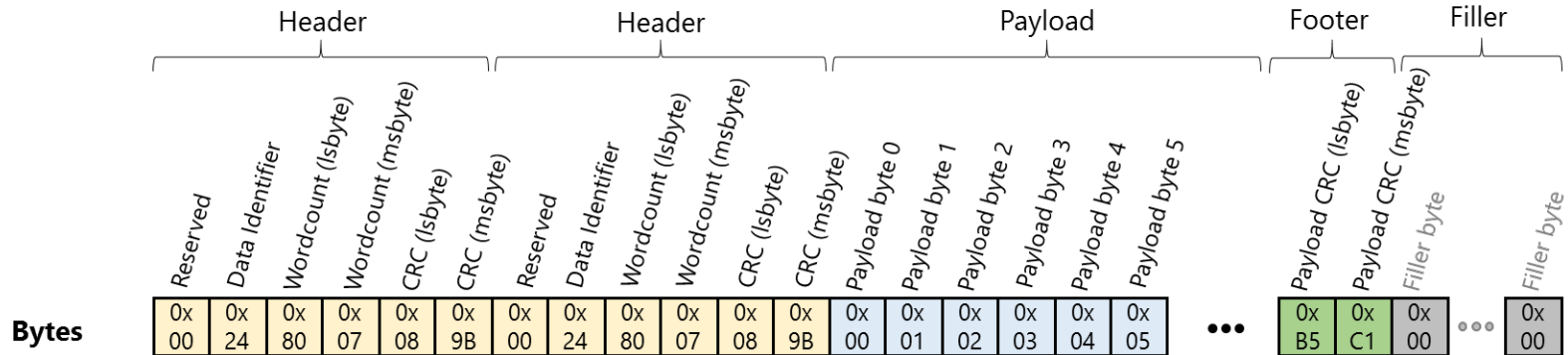
Integers - 4 lanes distributed



CSI-2 Long Packets in C-PHY



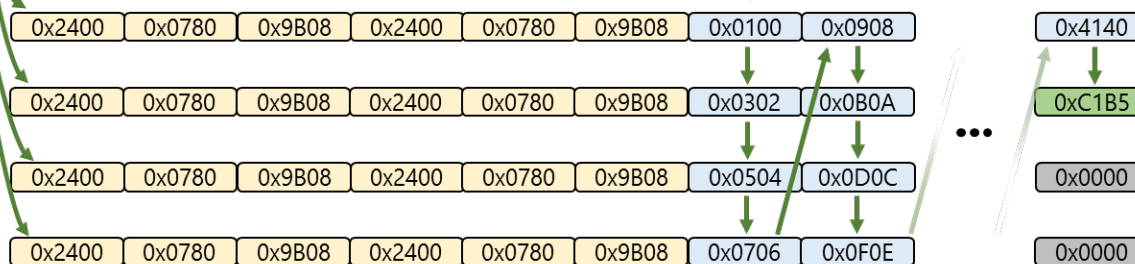
CSI-2 Long Packets in C-PHY



Header is not distributed but COPIED on every lane

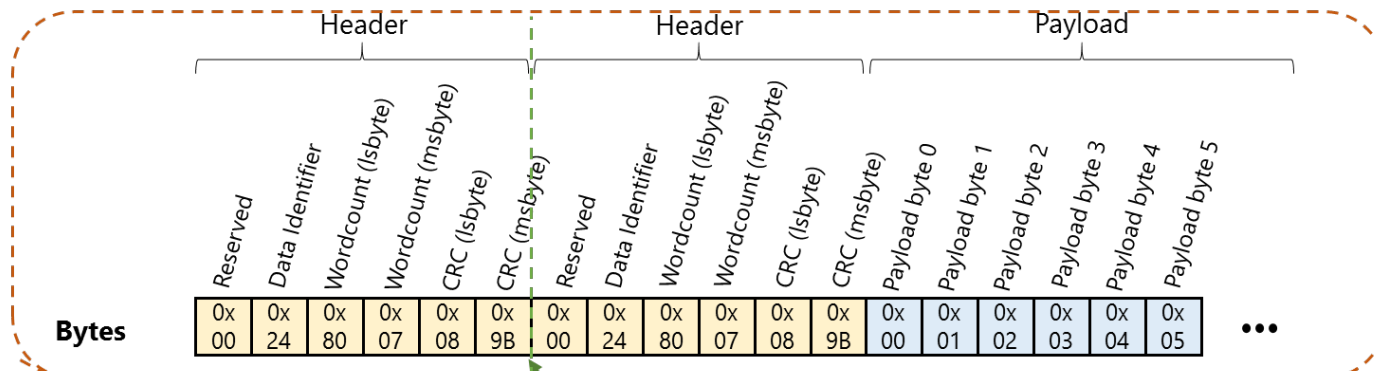
Payload and Footer are DISTRIBUTED across lanes

Integers - 4 lanes distributed

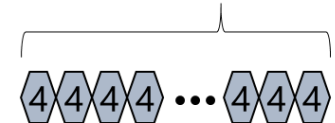
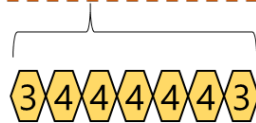
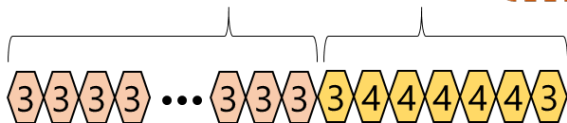
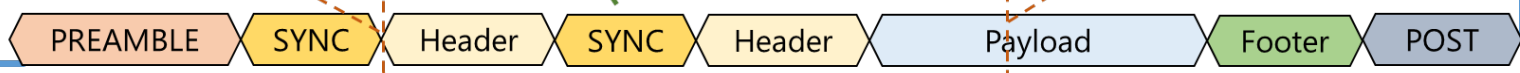


Add filler bytes to make all lanes the same length

CSI-2 Long Packets in C-PHY: The Invisible SYNC

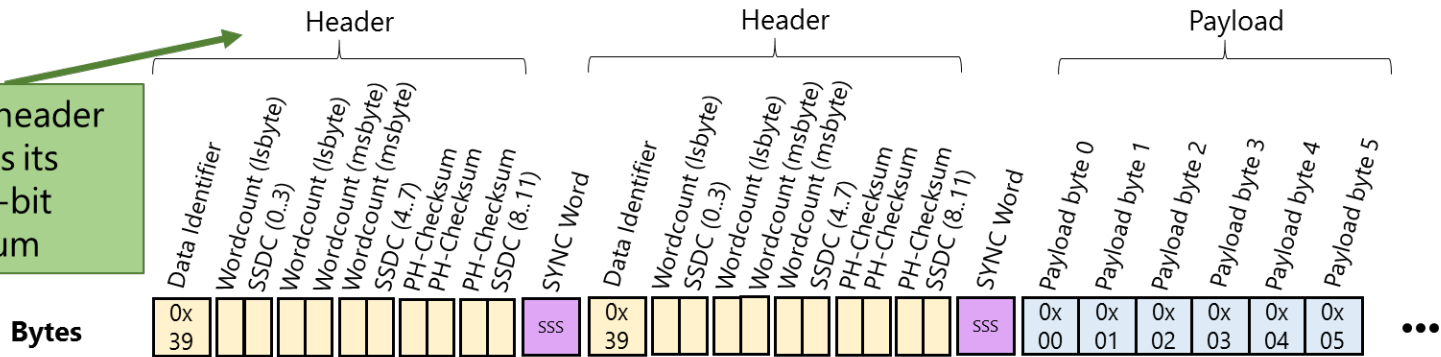


Invisible in Integer Domain, But Transmitted in Symbols!



DSI-2 Long Packets in C-PHY

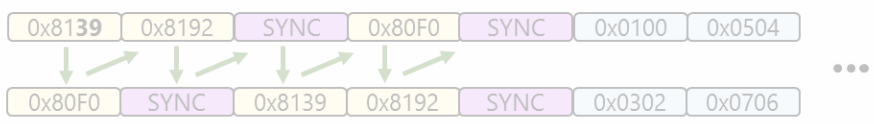
Packet header contains its own 12-bit checksum



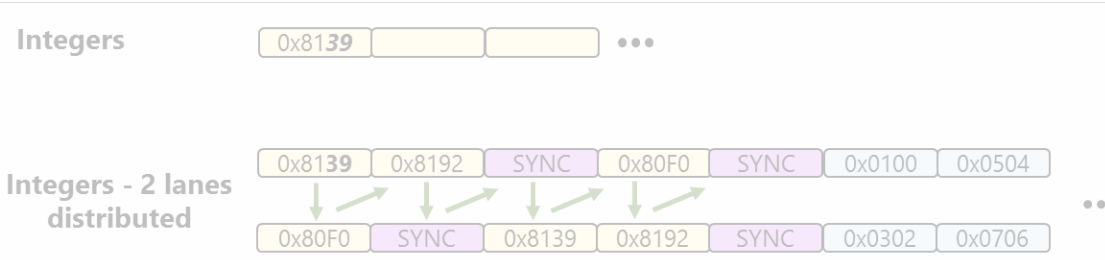
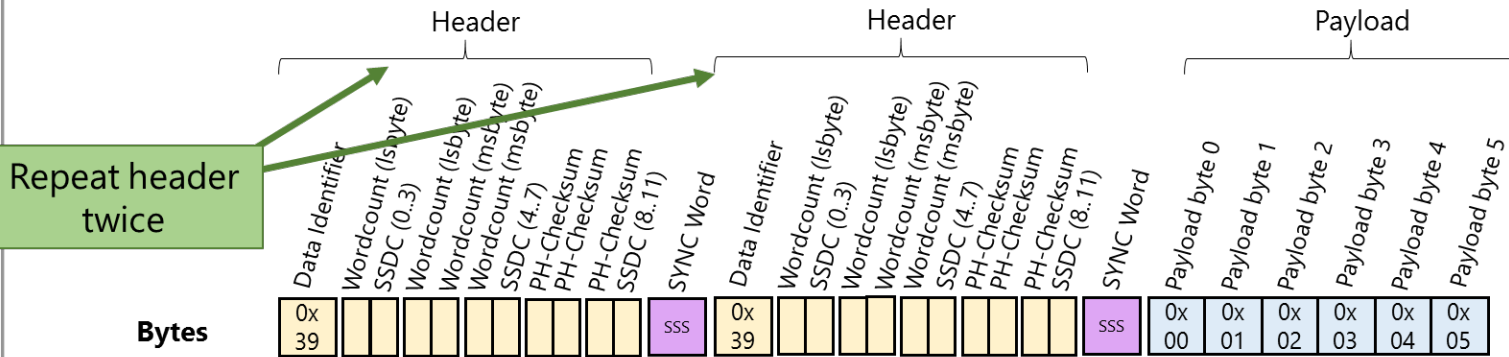
Integers



Integers - 2 lanes distributed

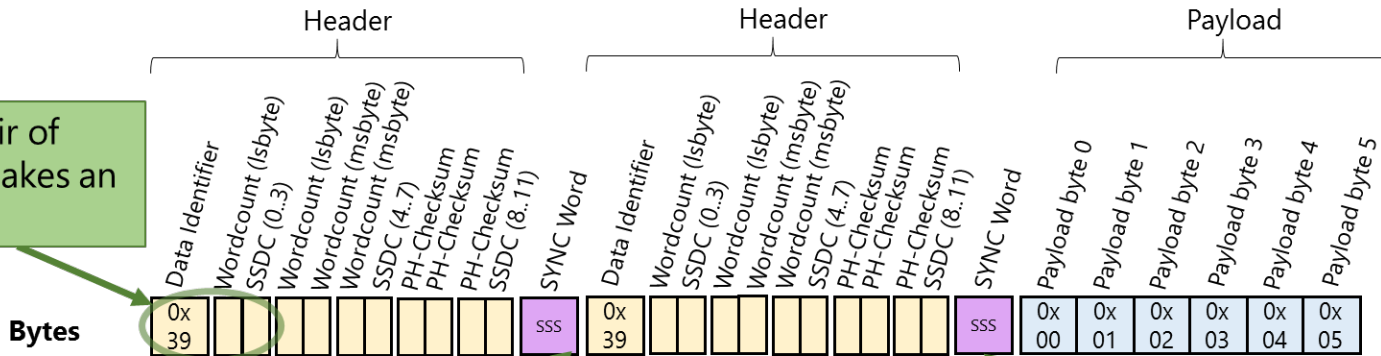


DSI-2 Long Packets in C-PHY



DSI-2 Long Packets in C-PHY

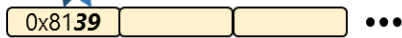
Each pair of bytes makes an integer



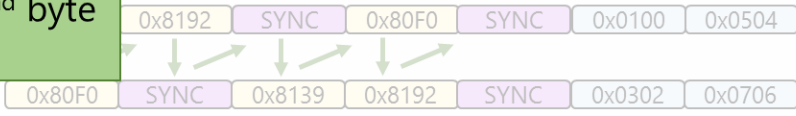
Bytes



Integers

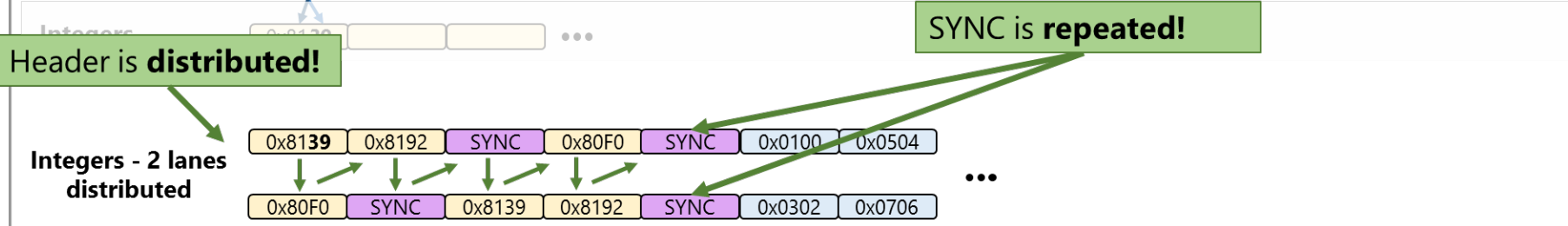
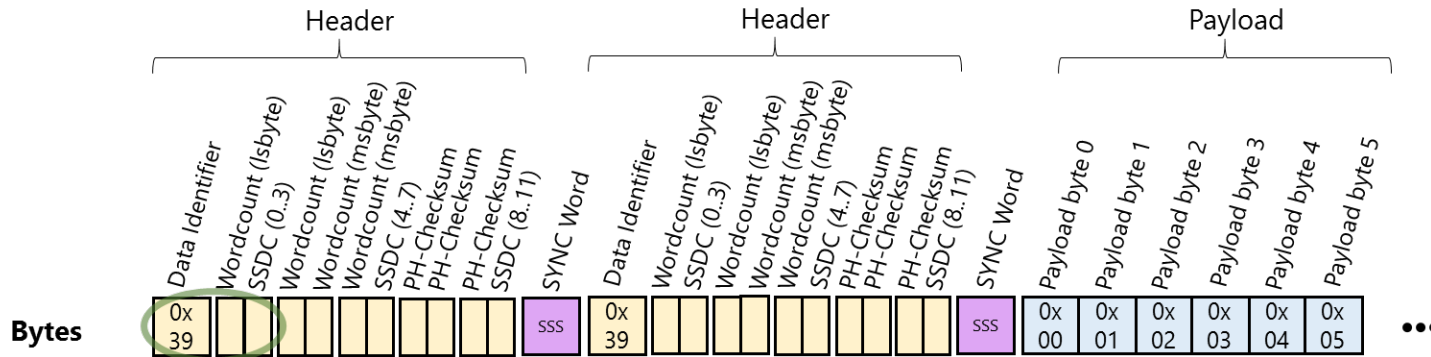


Integer LSB = 1st byte
Integer MSB = 2nd byte
And so on...



SYNC is visible in the protocol specification

DSI-2 Long Packets in C-PHY



DSI-2 Sample Protocol Analyzer Trace

CPHY DataCapture: Run_2016-08-05_1129_2lane / dsiDataCapture1

HS_immediate
 lane1 lane2 lane3 lane4 Go To: Packet#...

HS Bursts DSI Packets

Burst	VC	DT	DT name	Header CRC	WC	Shot	Payload CRC
0	0	0x39	DcsLongWrite	0x0192	3841		0x95F9
0	0	0x39	DcsLongWrite	0x0192	3841		0x6414
0	0	0x39	DcsLongWrite	0x0192	3841		0x63D5
0	0	0x39	DcsLongWrite	0x0192	3841		0xA823
0	0	0x39	DcsLongWrite	0x0192	3841		0xD8F4

Packet 0 Detail

0 |<< << < > >> >>|

	0	1	2	3	4	5	6	7	8
lane1 data:	8139 '1230410'	8192 '2012410'	DSYNC '3444443'	80F0 '0033400'	DSYNC '3444443'	012C '0320100'	1207 '3100201'	000B '3200000'	0002 '2000000'
lane2 data:	80F0 '0033400'	DSYNC '3444443'	8139 '1230410'	8192 '2012410'	DSYNC '3444443'	000F '3300000'	0104 '0100100'	0C00 '0000030'	0007 '3100000'
bytes:	3981F080	9281	3981	F0809281		2C010F00	07120401	0B00000C	02000700

Distributed integers/symbols as seen on the lanes

Transmitted bytes

DSI-2 Sample Protocol Analyzer Trace

CPHY DataCapture: Run_2016-08-05_1129_2lane / dsiDataCapture1

HS_immediate
 lane1 lane2 lane3 lane4 Go To: Burst#...

Burst ID	NumData	PreBegin	ProgSeq	PreEnd	Post	NumBits	SyncOffset	PostOffset	DSI Packets
0	20306	97	14	7	63	144320	223	142372	21
1	122808	97	14	7	60	861824	213	859876	121
2	122808	97	14	7	59	861760	198	859861	127
3	122808	97	14	7	64	861824	249	859912	127
4	122808	97	14	7	61	861824	235	859898	118
5	122808	97	14	7	57	861824	224	859887	108

Burst 0 Detail
 offset: 223 <<< << < > >> >>> SYNC POST PKT

```

wireAB: 101010001101111110110010101101110100101011000001101010101101111011010100001010001101100011010000001001010001101101011
wireBC: 101010100010101001000101010010010010101001101110001111111011010110111001110000111011010101100101101001010011101101101
wireCA: 01010111011001010010110101011001001101010110101011101000101101011011000011101100100001010110110000
wireStates: 616161314534656542541343434536452413434345321236515363626365365653653624123241412365365312462413123124124241236536536246
symbols: 344444312304102012410344444300334003444443032010031002013200000200000010001300200100000010011003101100000200010000000101
data (dec): 33081 33170 DSYNC 33008 DSYNC 300 4615 11 2 3329 264 256 1797 5 258 1024
data (hex): 8139 8192 80F0 012C 1207 000B 0002 0D01 0108 0100 0705 0005 0102 0400
  
```

lane1 data:	lane2 data:	bytes:
8139 '1230410'	8192 '2012410'	DSYNC '3444443'
80F0 '0033400'	DSYNC '3444443'	8139 '1230410'
3981F080	9281	3981 F0809281

Time domain view illustrates C-PHY byte ordering

Key Takeaways

Tx mapping and encoding in parallel domain

Rx false sync avoidance required pre-begin monitoring

Packet header definition required careful design of SYNC manipulation (both Tx and Rx)

CSI-2 & DSI-2 treat SYNC insertion differently





mipi[®]
DEVCON

Thank You!