



-
-
-
-
-

P1394b Simulation Taskforce

February '99 Report

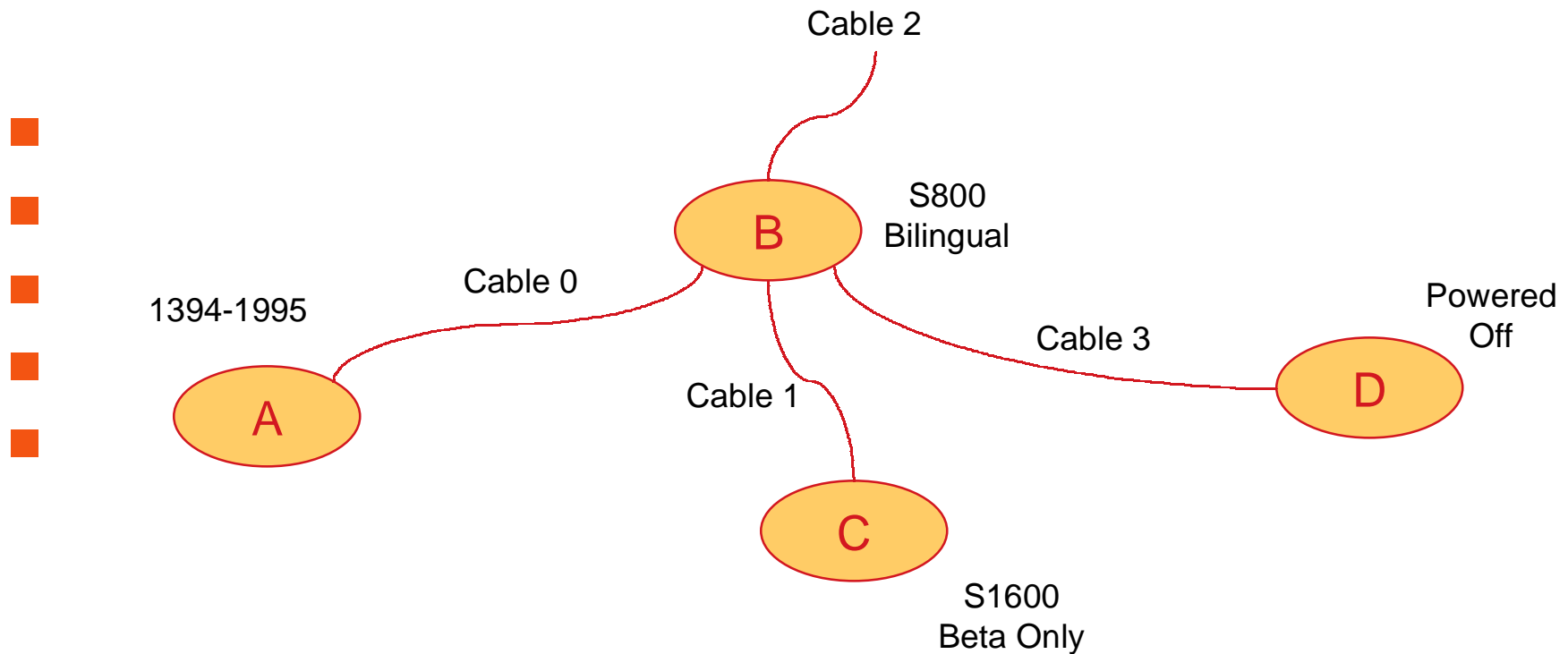
Jerry Hauck



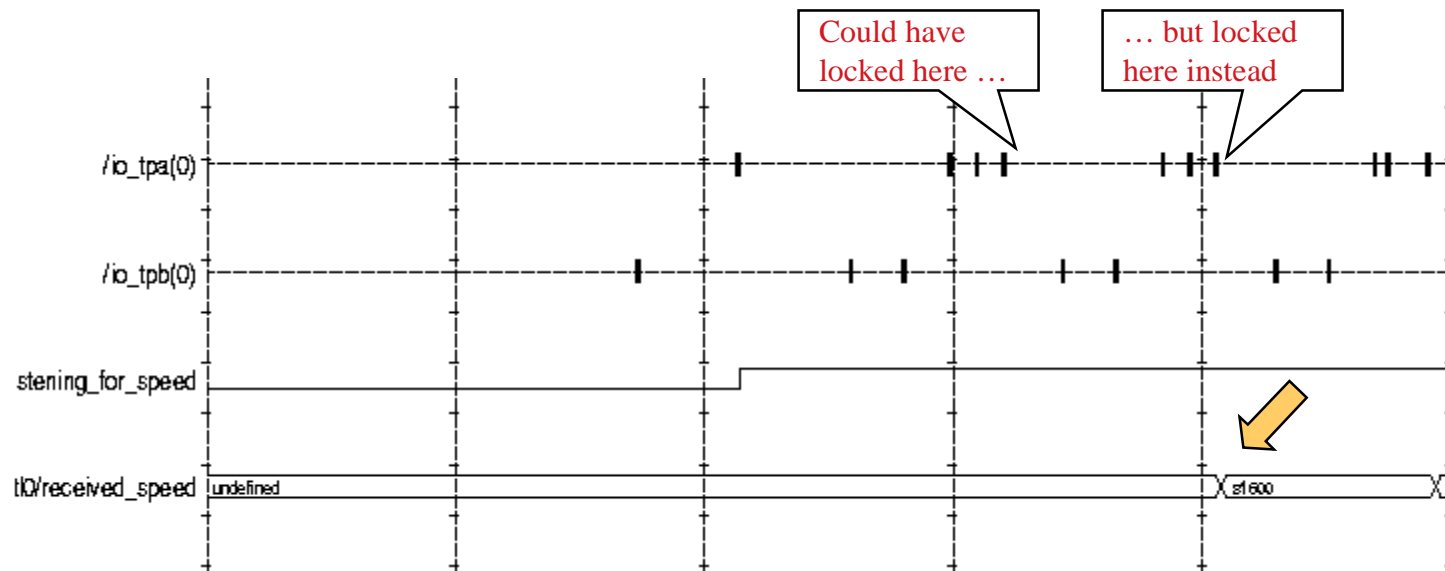
-
-
-
-
-

“Upstarts” Review Clause 11, Draft 0.15

Test Configuration for Examples

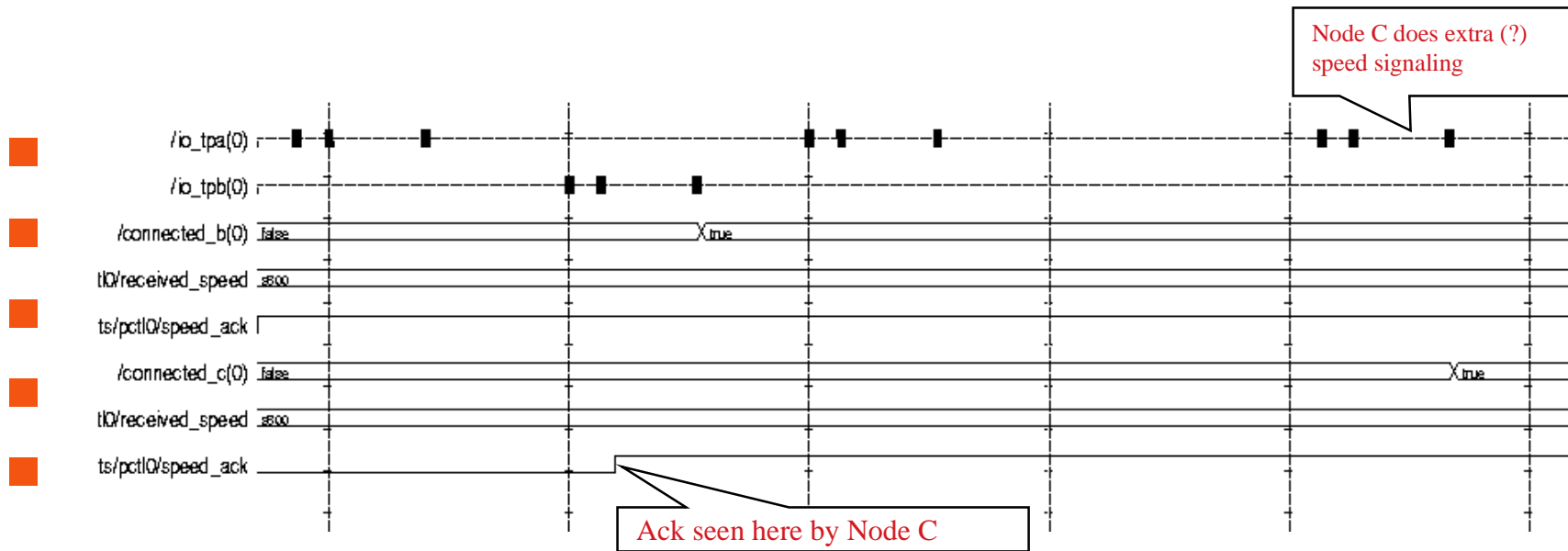


Improved Gap Detection

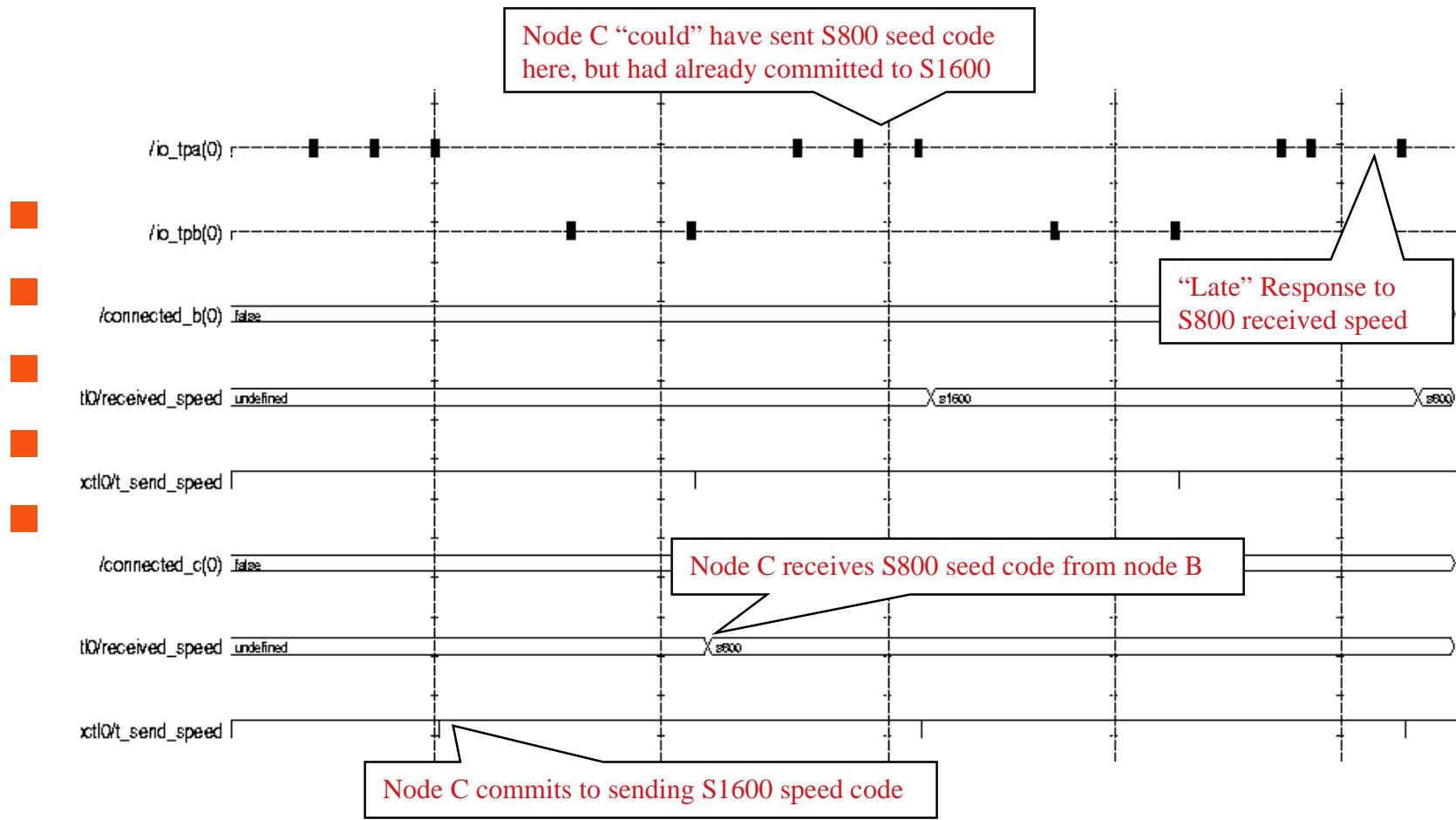


- Maximum idle run not part of a gap = 27 TONE_DURATIONS
- Maximum idle run between start bits = 63 TONE_DURATIONS
- Draft 0.15 code detected gap after 24 TONE_DURATIONS and expected a start bit within 59 TONE_DURATIONS from the last non-idle bit.
- Draft 0.16 detects a gap after 28 TONE_DURATIONS and expects a start bit within 64 TONE_DURATIONS after the last non-idle bit.

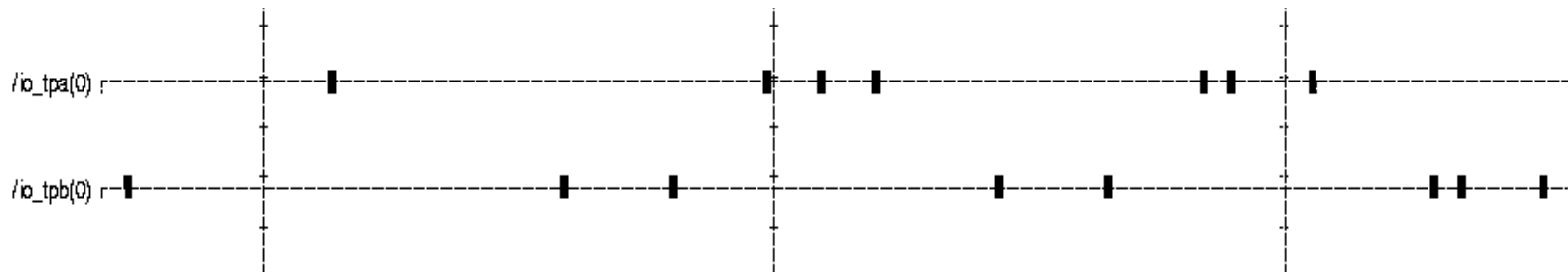
Extra Speed Signal



Delayed Speed Recognition



Draft 0.16 Improvements



- Speed transmitter and receiver are now independent and autonomous
- Set in motion and resulting events monitored by `set_beta()`



-
-
-
-
-

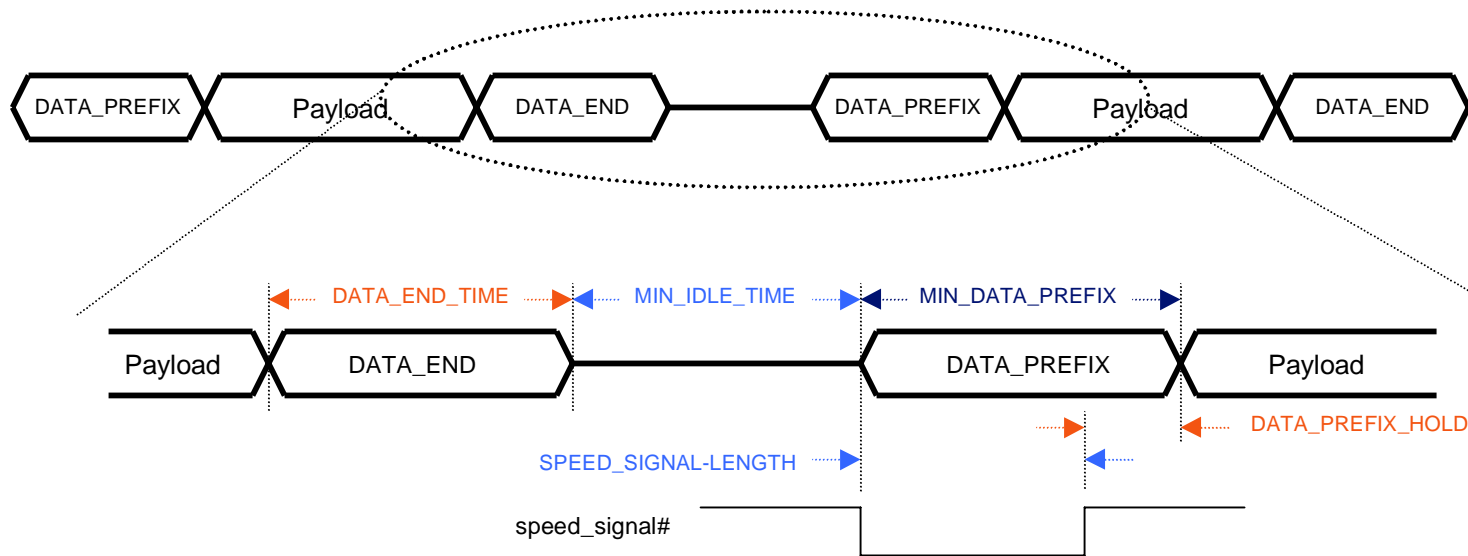
“Bport” Review Clause 11, Draft 0.16

Clause 10 Technical Issues

#	Reference	Description/Comment
1.		If DS port originates a packet with speed signal and DP coincidental with all min timings, how will beta mode PHY be able to repeat and have time to put a symbol of data prefix before the speed signal? Consider topology in which root P1394a PHY is attached to P1394b phy. As the root generates consecutive (non-concatenated) packets, there may not be any DATA_PREFIX before the speed signal. Seems like for each S200 packet, 40 ns of additional delay is inserted! Would it be better to distinguish between legacy and non-legacy packets in a different fashion?



“Sardine” Packets



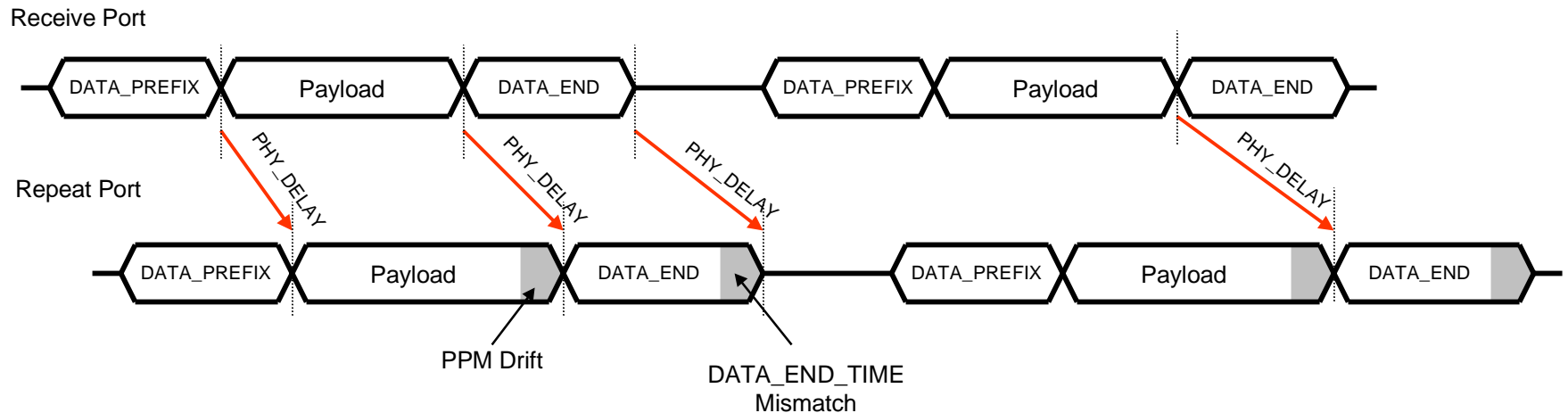
-
-
-
-
-

Minimum payload spacing allowed by P1394a:

$$\min(\text{DATA_END_TIME} + \text{MIN_IDLE_TIME} + \text{MIN_DATA_PREFIX})$$

Nothing is deletable!! Payload is not elastic and minimum timings are specified for originating and repeating ports.

Min/Max Mismatches

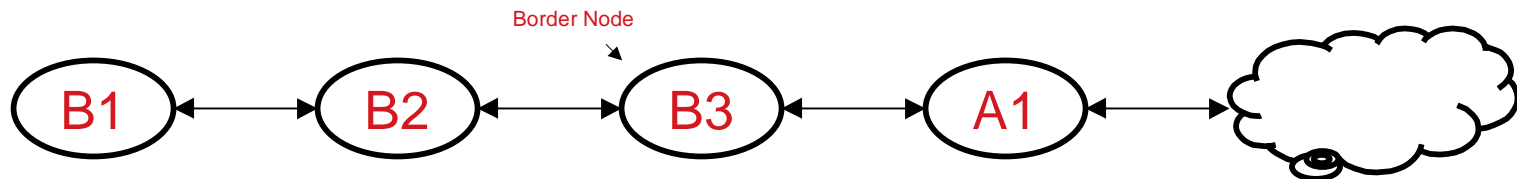


- With no deletable symbols or timings, the repeat delay through a PHY (PHY_DELAY) can steadily increase due to min/max timing mismatches.
 - PPM drift during payload (> 17 ns for a maximum sized packet)
 - 260 ns max vs 240 ns min DATA_END_TIME
 - Second order mismatches from MIN_DATA_PREFIX and MIN_IDLE_TIME
- Maximum FIFO depths and gap count calculations both affected by unbounded or out of spec PHY_DELAYS.

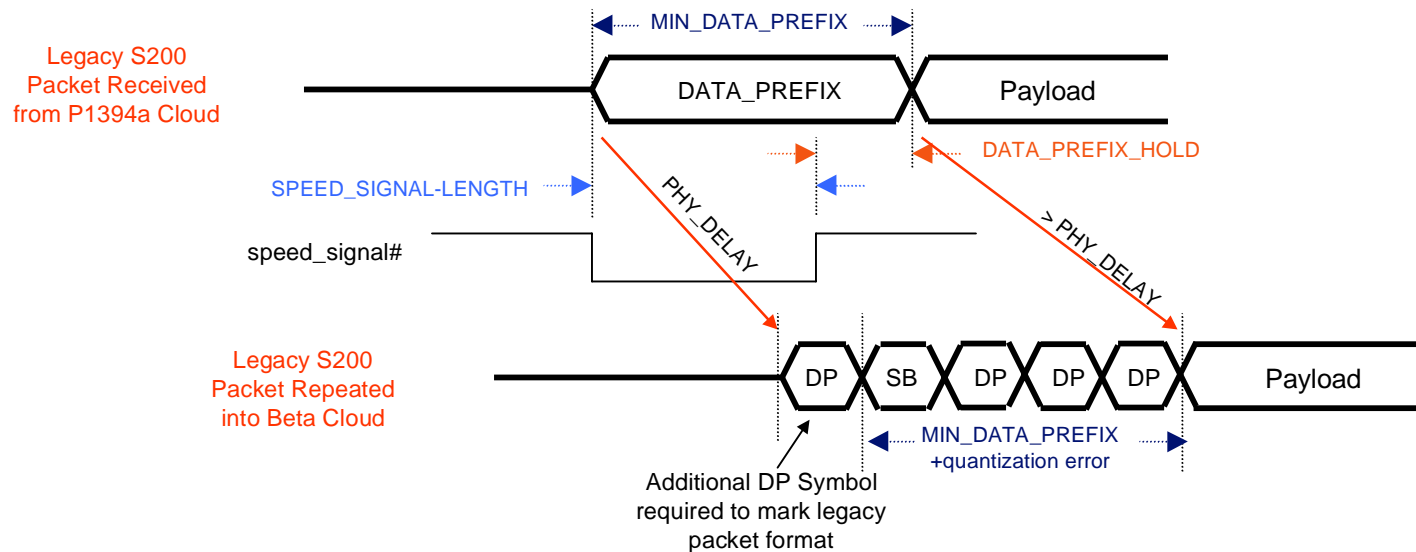
“Problem” Topologies

Can a realistic topology sustain minimally-spaced packets?

- P1394a root issuing non-concatenated isochronous packets
 - No arbitration between packets and subactions are unacknowledged
 - Sustainable for a full isochronous period.
- P1394b PHY's behind a border node
 - BOSS arbitration is overlapped, allowing any node to begin transmission immediately after a previous packet (as if legacy root).
 - Now applies to isochronous and asynchronous packets, indefinitely sustained
 - Example: B1 sends request, B2 acks, B2 sends request, B3 acks, etc. Node A1 sees and must repeat minimally-spaced packets



Border Node Worsens Problem



- Although speed signal and DATA_PREFIX can begin concurrently on a DS port, beta port signaling of a legacy packet requires an initial DP token
- The SB and subsequent DP tokens must consume at least MIN_DATA_PREFIX (140 ns). With 40 ns S200 symbol times, a 20 ns quantization error is inserted.

Don't Forget Concatenated Sequences

Concatenated sequences face similar issues

- Repeaters have minimum timings to enforce between packets
- Originators meet minimum timings plus 20-40 ns of elastic timings.
- Concatenated sequences are anticipated to be significantly more common
- Border node quantization error (particularly at S100 speeds) can easily "eat" any elasticity introduced by a P1394a originator

Summary

P1394a has theoretical problem

- PHY_DELAY can increase with each received packet
- - Only sustained for an isochronous period and likely to be “rare”
- - Concatenated sequences more common and some elasticity is provided

P1394b border nodes exasperate issue

- - More than 60 ns added to PHY_DELAY with each received packet
- - Number of potential problem topologies increased relative to P1394a
- Sustainable for longer periods of both asynch and isoch traffic.

Recommended Actions

- Consider alternative way to distinguish legacy and beta-only packet formats (don't require insertion of extra symbol).
- Rules for meeting MIN_DATA_PREFIX on each individual legacy packet format must be relaxed to account for quantization error
 - As an average, timings must be met to bound FIFO depths at border nodes.
 - Instantaneously, allow +/- one symbol (?) to maintain average.
- All packets originated by a P1394b node should include an appropriate number of elastic symbols to allow for an unlimited number of minimally-spaced packets without violation of PHY_DELAY.

Potential Clause 10 Bugs

#	Reference	Description/Comment
1.	Draft 0.16 10.4.4 p. 112	If S100 legacy packet is sent, how does pkt_speed get set to S100 as required for the data padding loop?
2.	Draft 0.16 10.4.4 p. 112	If pkt_speed > port_speed, bport_transmit_actions () doesn't replace speed code and data with data prefix properly. For example, for loop on 14 prevents tx_character from being called when pkt_speed > port_speed. Without tx_character being called for a symbol time, what is sent on the cable? Also, when bportT.tag == DATA, tx_character is called without regard to speed.
3.	Draft 0.16 10.6.1.1 p. 110	<p>Clause 10 appears to be written assuming an "semaphore" style of operation for signaled events. Otherwise, the top-level while loops in bport_transmit_actions(), tx_sync_lost_actions(), etc. could miss one-shot indications of their respective exit conditions.</p> <p>Given such, the first transition into PTX2 after the most recent activation of bport_active will be immediately followed by a transition back to PTX2:PTX1. This occurs since the event SYNC_LOST was signaled while in PTX1 (corresponding receive state machine performed the signal) and never tested until PTX2. Consequently, a stale indication is always present for SYNC_LOST the first time through. A dummy test of SYNC_LOST could be performed upon entry to PTX2 to clear any stale indication; however, the recommended approach is to use variables to provide the semaphore signaling.</p> <p>Also, both the Tx and Rx machines have transitions based on SYNC_OK. If semaphore operation is assumed, the first machine to test the semaphore will clear it, leaving the other machine without the ability to test the indication.</p> <p>Signals SYNC_LOST, RX_SYNC_DONE, and SYNC_OK should be converted to the corresponding variables sync_lost, rx_sync_done, and sync_ok. All three can be cleared in PRX0. PRX1 should clear sync_ok and rx_sync_error before setting sync_lost.</p> <p>rx_sync_done gets set in PRX3, but not before sync_lost is cleared (which could be done in PRX1, PRX2, or PRX3).</p> <p>sync_ok gets set in PRX3 and is used by both the Rx and Tx machines to move into the main receive and transmit states.</p>

Interpretation of signal(EVENT)

P1394a and P1394b both make use of the “invented” signal(EVENT) call to communicate between processes. Two plausible interpretations:

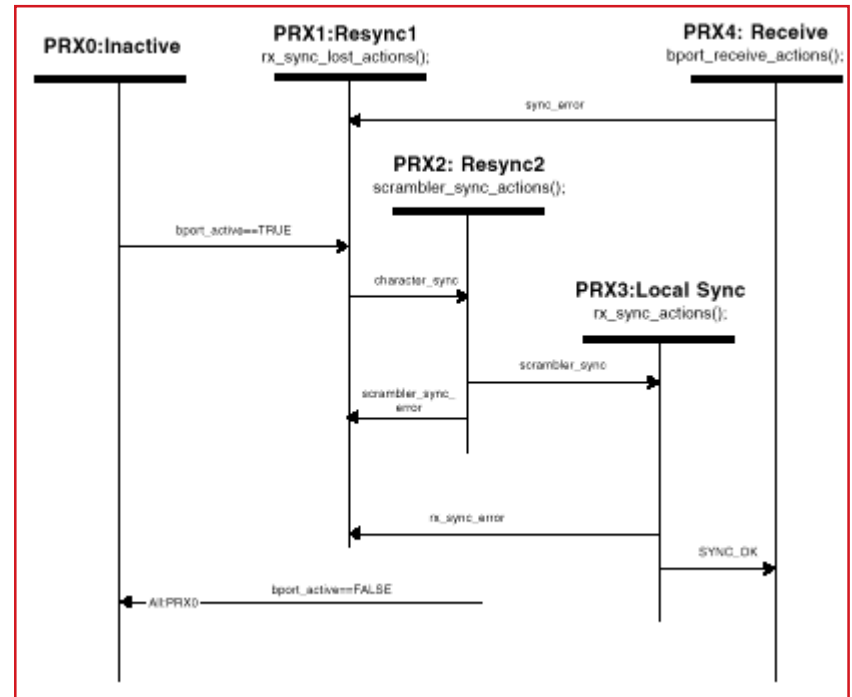
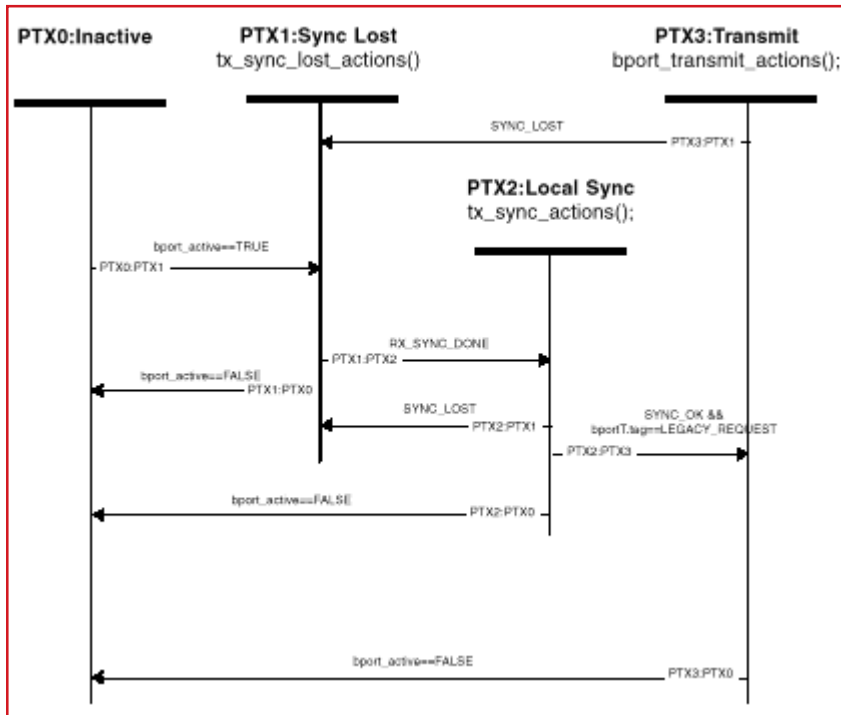
- “Semaphore” operation

- signal(EVENT) causes EVENT to be set
- EVENT remains set until tested by a test_event or wait_event call
- Example from P1394a: test_event(SPEED_SIGNAL_RECEIVED) arguably determines if a speed signal was previously detected and latched

- “Indication” operation

- signal(EVENT) causes a “one-shot” on EVENT
- Only processes waiting with wait_event() recognize the one-shot indication
- Example from P1394a/b: wait_event(PH_CLOCK.indication()) is intended to wait until the next rising edge of the bit clock.

Bport State Machines



Proposal

For consistency and clarity,

- signal(EVENT) and wait(EVENT) follow the “one-shot” interpretation, allowing one process to suspend until the next indication signaled by a second process.
- For code which needs “semaphore” operation, shared variable should be used with explicit set and clear statements. This has the added benefit of allowing multiple process to test a given semaphore.

Clause 10 C Code Syntax and Nits

#	Reference	Description/Comment
1.	Draft 0.16 10.4.4 p. 111 l. 27	variable "port_speed" has already been declared on line 22.
2.	Draft 0.16 10.4.4 p. 112 l. 30	The brace '{' at the end of line 30 is extra.
3.	Draft 0.16 10.4.4 p. 114	The use of variable "i" on line 45 overwrites the loop variable "i" on line 41.

Clause 10 C Code Syntax and Nits (continued)

#	Reference	Description/Comment
4.	Draft 0.16 10.4.4 p. 112	<p>Speed encoding is not consistent across specification or within bport_transmit_actions. port_speed is provide by clause 11 using the following encoding:</p> <pre>enum speedCode {Undefined, S100, S200, S400, S800, S1600, S3200}; // speed codes are encoded 001 010 011 100 101 110</pre> <p>bport_transmit_actions calles convertspeed to create pkt_speed which returns intergers in {100, 200, 400, 800, 1600, 3200 }. Subsequently, port_speed and pkt_speed are directly compared (i.e., port_speed isn't converted).</p> <p>Proposal: The standard should use a consistent encoding matching the enumerated speedCode type above. Speed codes can than be freely referenced as integers and will have values in the range 001 – 110 for defined speed codes. This proposal requires an edit to Table 14-1 on page 182 to correct the speedCode enumerated type shown there.</p> <p>port_speed declared on line 22 (or 27) should be declared as an int for consistency with Clause 7 declaration on p. 135, l. 6?</p> <p>min_bus_speed on p. 111, l. 29 should be set to 1 or S100.</p> <p>bport_transmit_actions on page 122 become:</p> <pre>void bport_transmit_actions() { int pkt_speed; //speed of each packet. int speed_ratio; //number of symbols per byte for given pkt_speed and port_speed int i,j; pkt_speed=port_speed;</pre>

Clause 10 C Code Syntax and Nits (continued)

#	Reference	Description/Comment
		<pre> while(~SYNC_LOST && bport_active) { if(bportT.tag==SPEED) { //currently will send a speed signal even for S100 - needs arb. //st. machine to not send speed signal for //S100 if non-beta nodes on bus i.e. long packet format. pkt_speed=bportT.speed; localT.tag=CTRL; speed_ratio = 1<<(port_speed-pkt_speed); j=port_speed-pkt_speed; number of intervening speeds for(i=0;i<speed_ratio;i++) { if(i=j) localT.ctrl=SPEEDb; else localT.ctrl=SPEEDa; tx_character(localT); } } else if(bportT.tag==DATA) { tx_character(bportT); localT.tag=CTRL; localT.ctrl=SPEEDa; speed_ratio = 1<<(port_speed-pkt_speed); for(i=1;i<speed_ratio;i++) tx_character(localT); //send padding if required } else if(bportT.tag==REQUEST) { tx_character(bportT); pkt_speed=port_speed; //occurence of a request indicates end of all packet components, //so set pkt_speed to default } } </pre>

Clause 10 C Code Syntax and Nits (continued)

#	Reference	Description/Comment
		<pre> else if(bportT.tag==CTRL) { if (bportT.ctrl==ARB_RESET bportT.ctrl==ARB_RESET_GRANT) { //stretch control to length of a S100 byte speed_ratio = 1<<(port_speed-min_bus_speed); for (i=0;i<speed_ratio;i++) tx_character(bportT); } else { //stretch control to length of a padded byte at packet speed speed_ratio = 1<<(port_speed-pkt_speed); for (i=1;i<speed_ratio;i++) tx_character(bportT); } else tx_character(bport T); //send legacy requests as and when instructed } } </pre> <p>The routine convert_speed on page 113 should be removed.</p> <p>Other places not yet identified ... presumably in the receive actions.</p>
5.	Draft 0.16 10.4.4 p. 114	The use of variable "i" on line 45 overwrites the loop variable "i" on line 41.

Clause 10 Editorial Comments

#	Reference	Description/Comment
1.	Draft 0.16 10.3.7.1 p. 91 l. 18-19	8B/10B characters should reference scrambled (rather than unscrambled) bytes xx = the 5 bit input value, base 10, for bits ABCDE y = the 3 bit input value, base 10, for bits FGH should become xx = the 5 bit input value, base 10, for bits A'B'C'D'E' y = the 3 bit input value, base 10, for bits F'G'H'
2.	Draft 0.16 10.3.7.1.2 pp. 91-93	Tables 10-5 and 10-6 do not include any references to the traditional "K" (control) bit used in 8B/10B encoding nor is the 'K' bit defined for P1394b. Consequently, the reference to "the state of the control bit" (p. 91, l. 54) in 10.3.7.1.2 and the reference to "(K=0)" in the title for Table 10-7 should be removed.
3.	Draft 0.16 10.4.4 p. 111 l. 7	The port transmit and receive state machines are independent with independent states. A single portState type can not represent both the transmit state and the receive state simultaneously. Perhaps the enumerated portState type should be split into portTxState for the transmit states and portRxState for the receive states. The rx_state variable on line 26 should then be of type portRxState. Declarations for portRxState and rx_state may be better suited in the receive C code beginning on p.118.
4.	Draft 0.16 10.6.1.1 & 10.4.5 pp. 110 & 117	For completeness, add power reset transitions into the state machines. All: PTX0a and All: PRX0a triggered by "power reset"
5.	Draft 0.16 10.6.1.1 p. 110	For consistency with the receive state machine, combine PTX1:PTX0, PTX2:PTX0, and PTX3:PTX0 into a single All:PTXb transition.