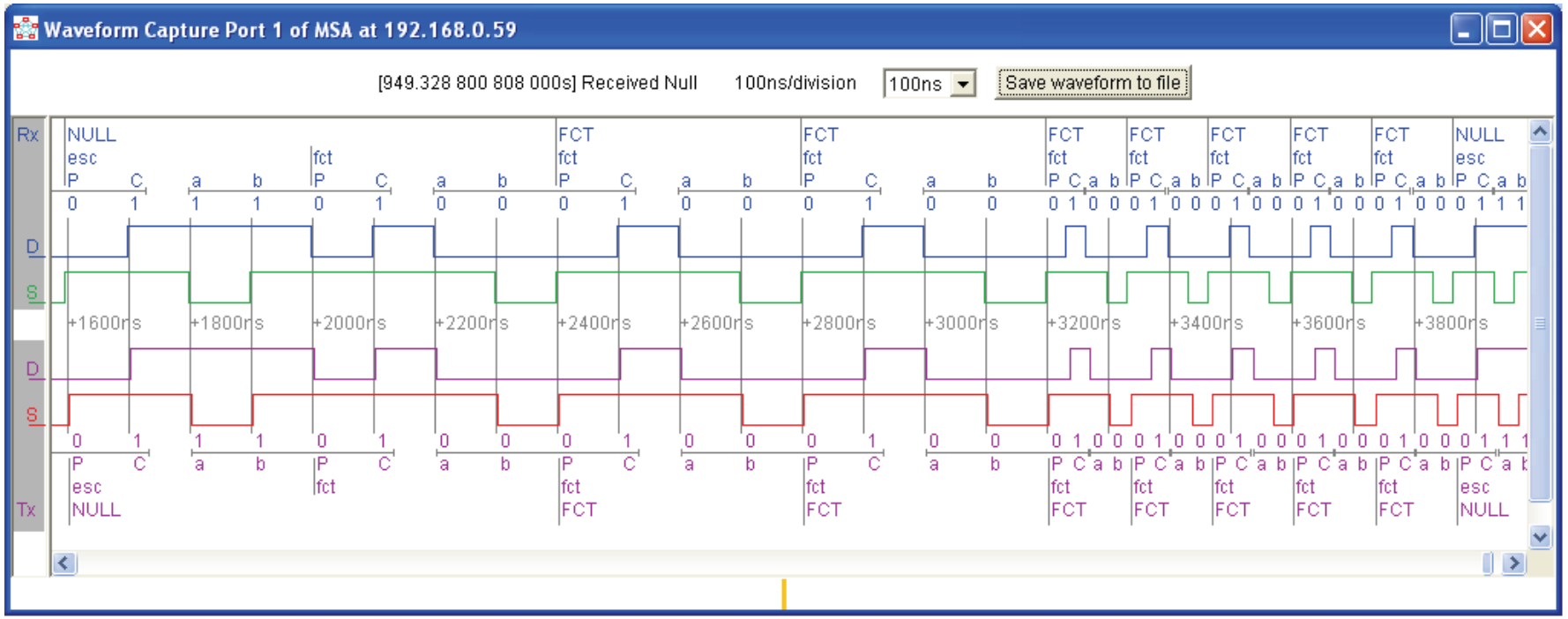
Measuring Time and Time-related Aspects of SpaceWire

Barry M Cook, C Paul H Walker
4Links Limited

International SpaceWire Conference
17-19 September 2007

SpaceWire signals

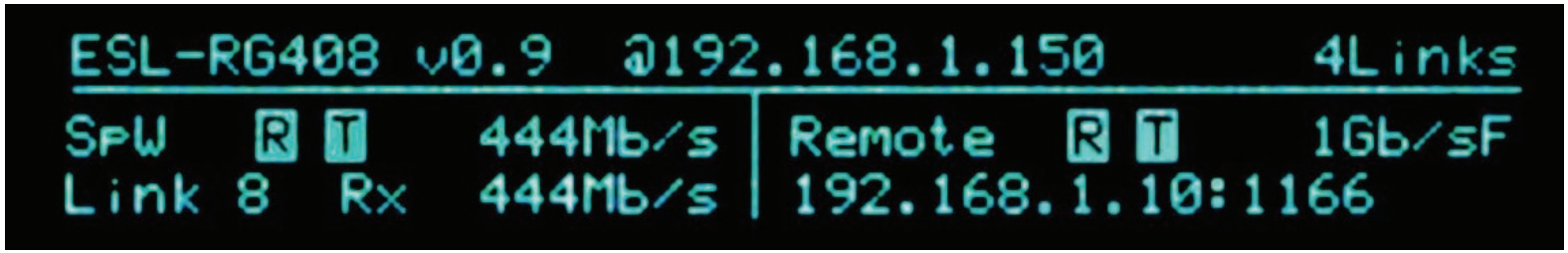


- **Initialization sequence, 10Mbits/s to 30Mbits/s**
- **Adjacent bits can be different lengths**
- **SpaceWire data reception is Asynchronous**

Asynchronous design

- SpaceWire's Data/Strobe makes clock recovery easy
- But
- The recovered clock is Asynchronous
- Tools for Asynchronous design are much less well developed than for synchronous design
- Asynchronous design presents traps, even for experienced designers
- So SpaceWire developers need to measure time
- And they need to test for time

Measuring Time 1



- **Front panel display shows not only transmit speed but also the receive speed**

- **Measured by counting the number of Rx clock cycles in so many cycles of a reference (system) clock**

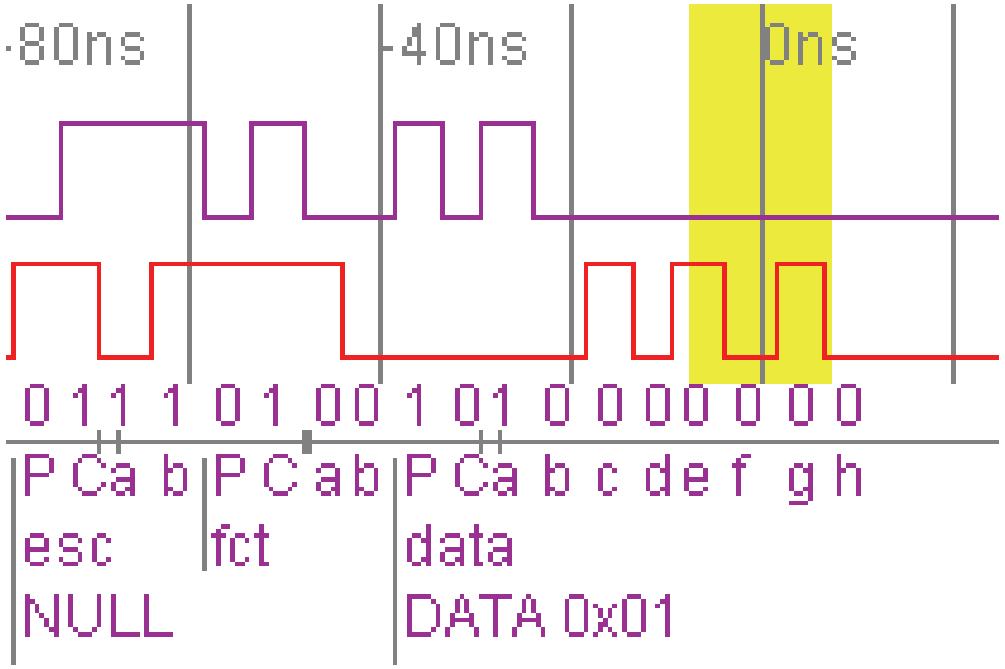
Measuring Time 2

- **Initialization sequence, 10Mbits/s to 100Mbits/s**
- **Time Tags set to record FCTs (after first FCT)**

```
Rx:@1 /*664.606 343 013 3s*/ FCT
      /*664.606 343 413 3s*/ FCT // 400ns
      /*664.606 343 813 3s*/ FCT // 400ns
      /*664.606 343 853 3s*/ FCT // 40ns
      /*664.606 343 893 3s*/ FCT // 40ns
      /*664.606 343 933 3s*/ FCT // 40ns
```

How to measure time

- Signals are sampled at close to 1GSample/s
- State of wires is recorded on every sample
- Interesting Events trigger Waveform Capture and/or Time Tag
- Time Tag has resolution of sampling interval

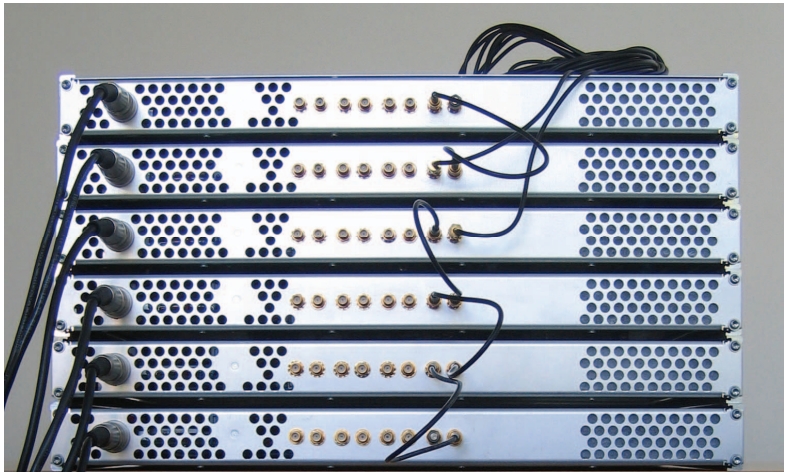


Unifying time across a (large) system

4Links®

// Test 1
// Check alignment of received timetags (Specification is within +/- 3ns)
// Generate synchronized outputs from C, receive on A, B & C

Tx:@C@1 +
Tx:@C@2 +
Tx:@C@3 +



Tx:@C@1 [| #01 #01 #C1 #A1 EOP]
Tx:@C@2 [| #01 #01 #C2 #B1 EOP]
Tx:@C@3 [| #01 #01 #C3 #C4 EOP]
// @C Barrier lifted at T=6705.304 943 739 9s
Rx:@A@1 /*6705.304 943 738 9s*/ #01 #01 #C1 #A1 EOP
Rx:@B@1 /*6705.304 943 741 1s*/ #01 #01 #C2 #B1 EOP
Rx:@C@4 /*6705.304 943 740 0s*/ #01 #01 #C3 #C4 EOP

Total spread 2.2ns, well within +/- 3ns

Testing time-related behaviour

How to do time-related tests?

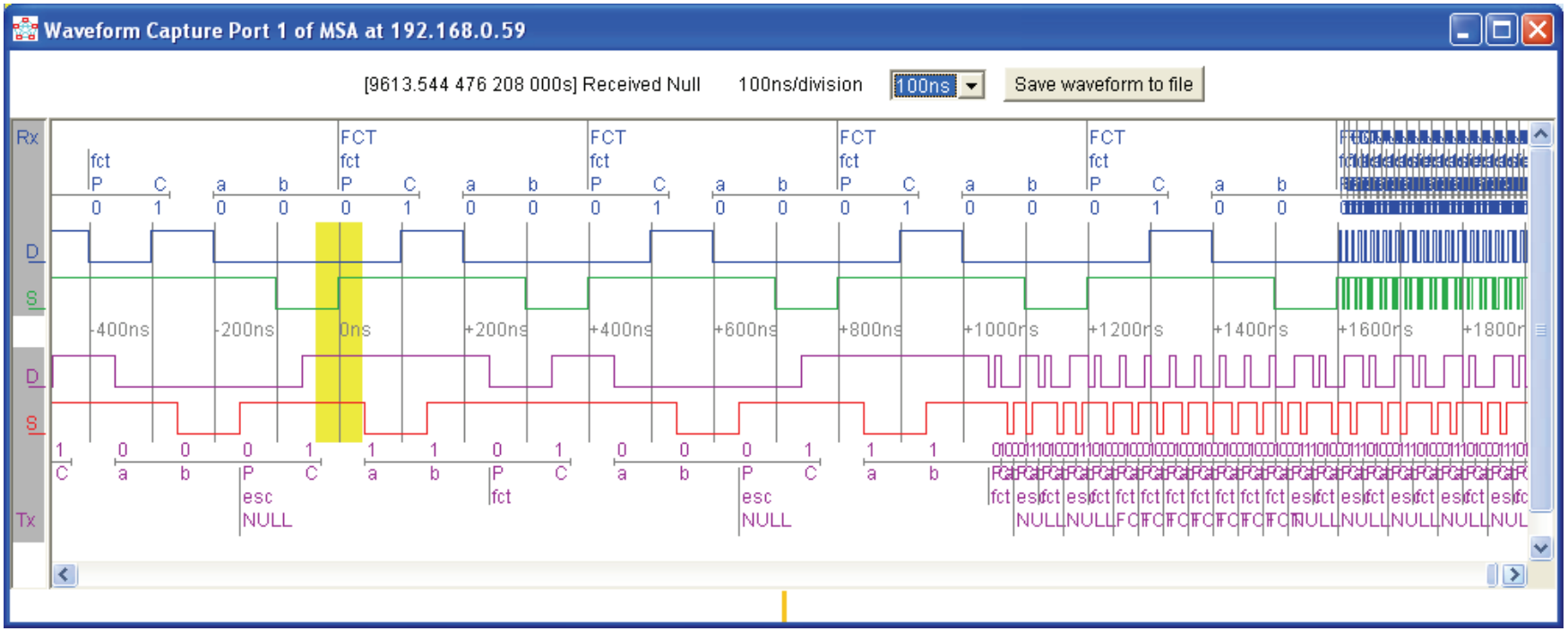
- Control the transmit speed over a wide range (1Mbit/s to 400Mbit/s) in small increments (1Mbit/s or less)
- Insert arbitrary gaps between D and S transitions
- Synchronize outputs, both within a box and between boxes
- Use these testing tools together with measurement and comprehensive diagnostics to determine behaviour

Testing time-related behaviour

- Two examples shown, many more possible:

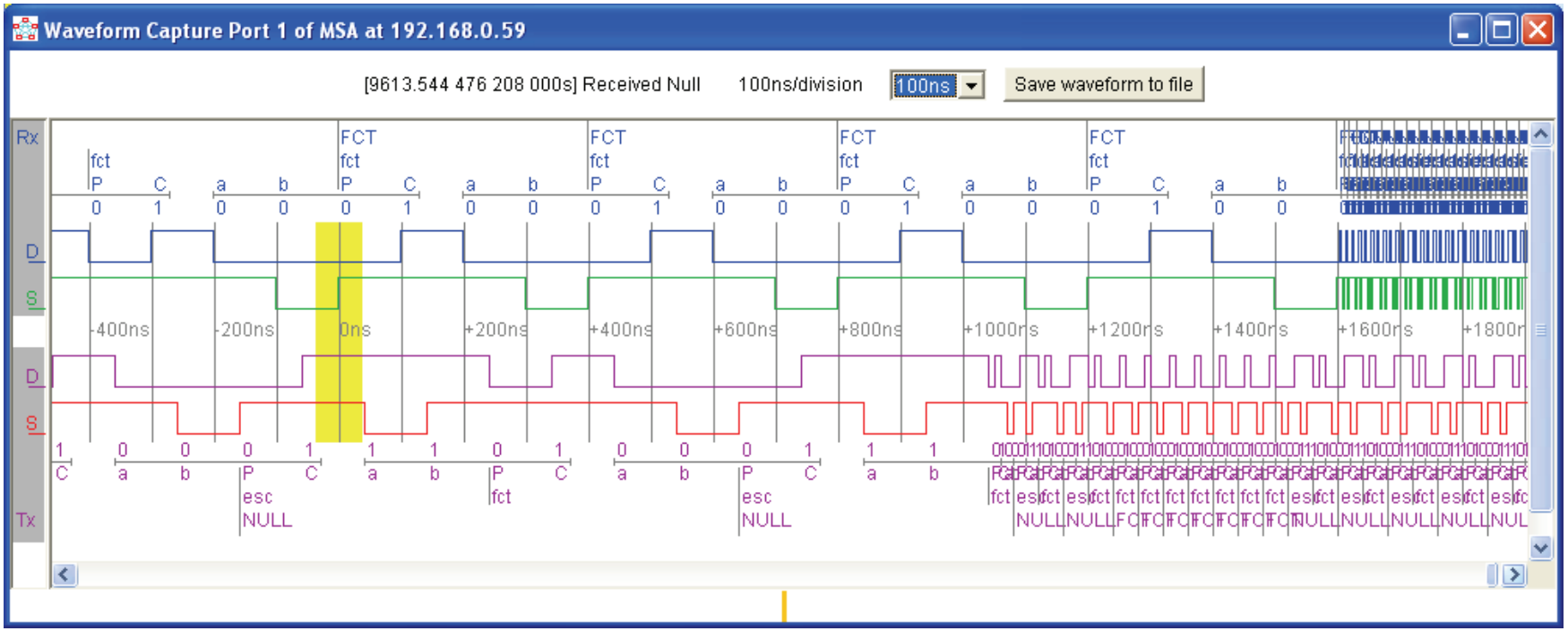
- Simulating noise
- Measuring Disconnection Timeouts

Simulating Noise: Fast and slow SpaceWire



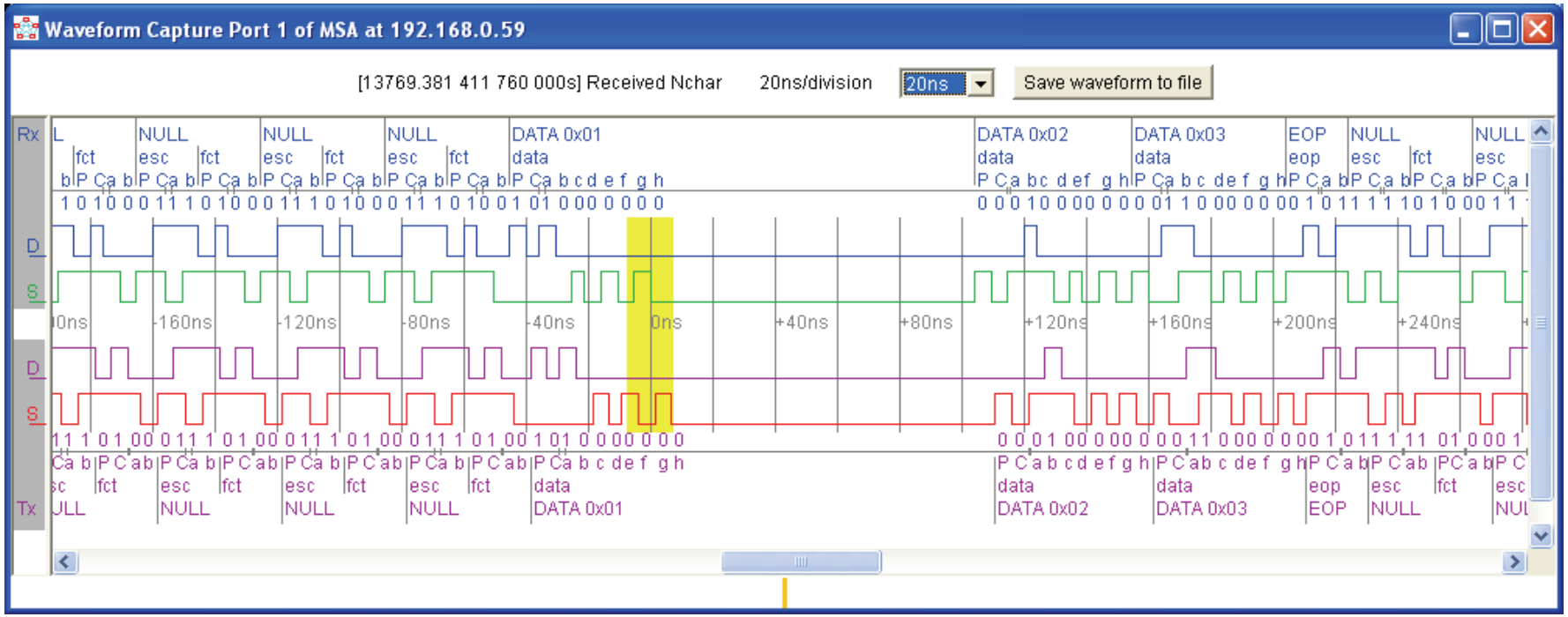
- One end starts and changes to 100Mbits/s
- The other end starts and changes to 400Mbits/s
- If the first end can't receive at 400Mbits/s: What then?

Simulating noise: Fast and slow SpaceWire



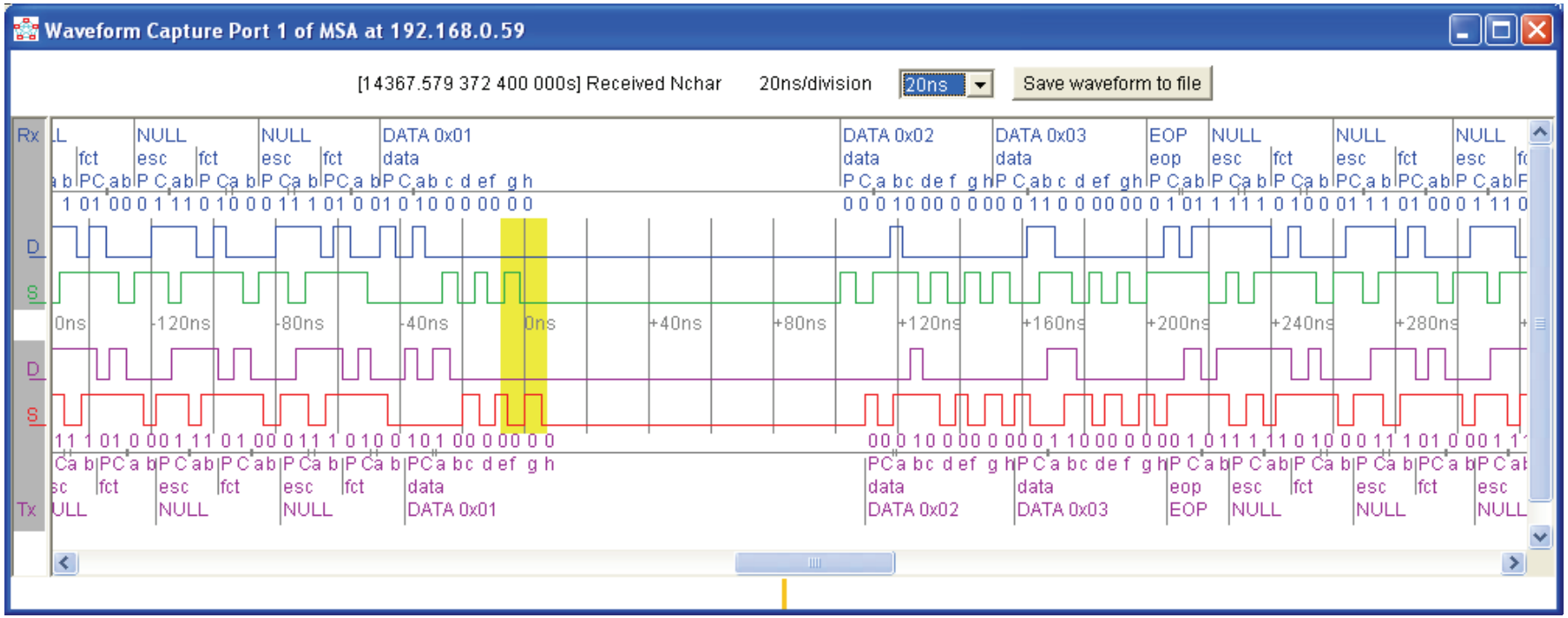
- If the first end can't receive at 400Mbits/s: What then?
- Two out of two designs we have tested like this have locked up. Disconnection does not remove the lock-up
- Both need a full reset to recover

Inserting a gap between transitions



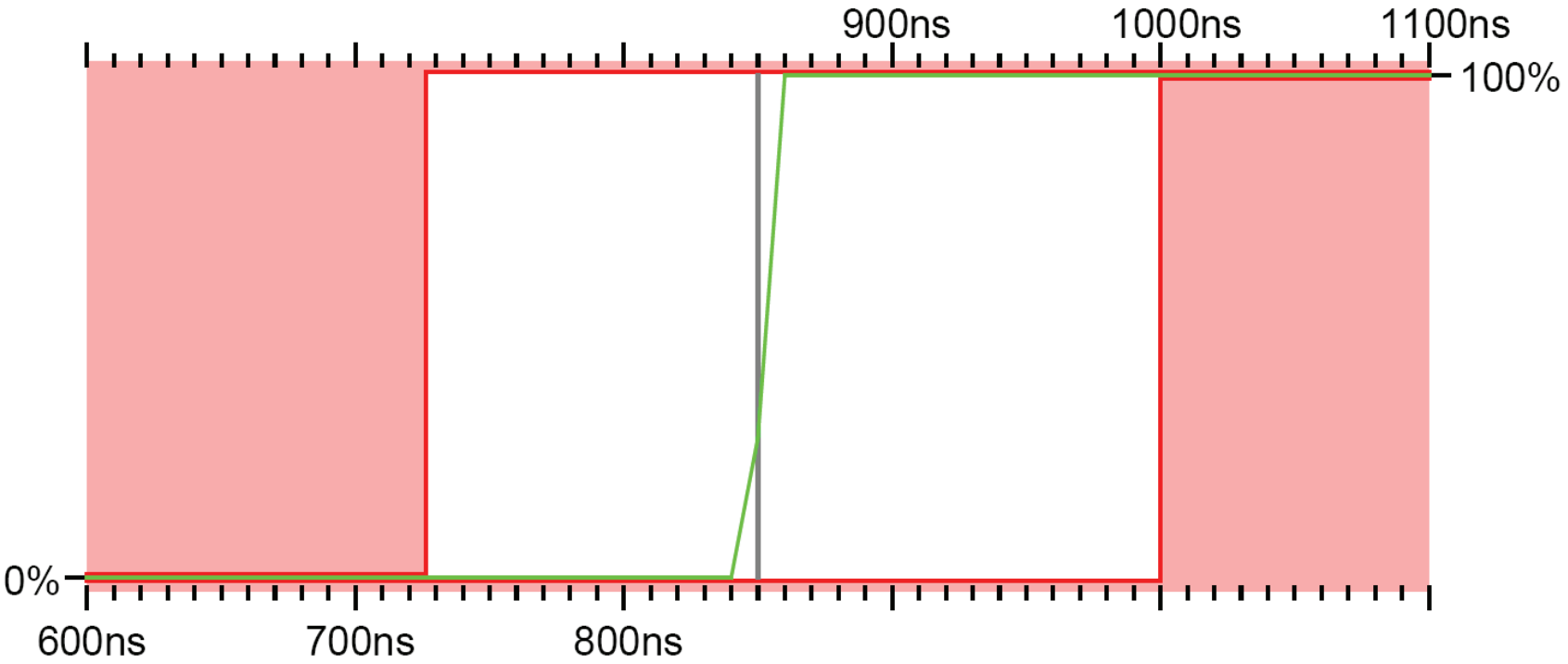
- For test purposes, gaps can be inserted between bits
- Gap here is 21 bit-times at 200Mbits/s, or 105ns

Fine tuning Tx Link speed



- Links speed set to 203Mbits/s (for example)
- Useful for testing operating margins of receiver
- 21 bit-time gap at 203 Mbits/s is 103.4ns
- Can fine tune gap to measure Disconnection Timeout

Disconnection Timeout SpaceWire design A



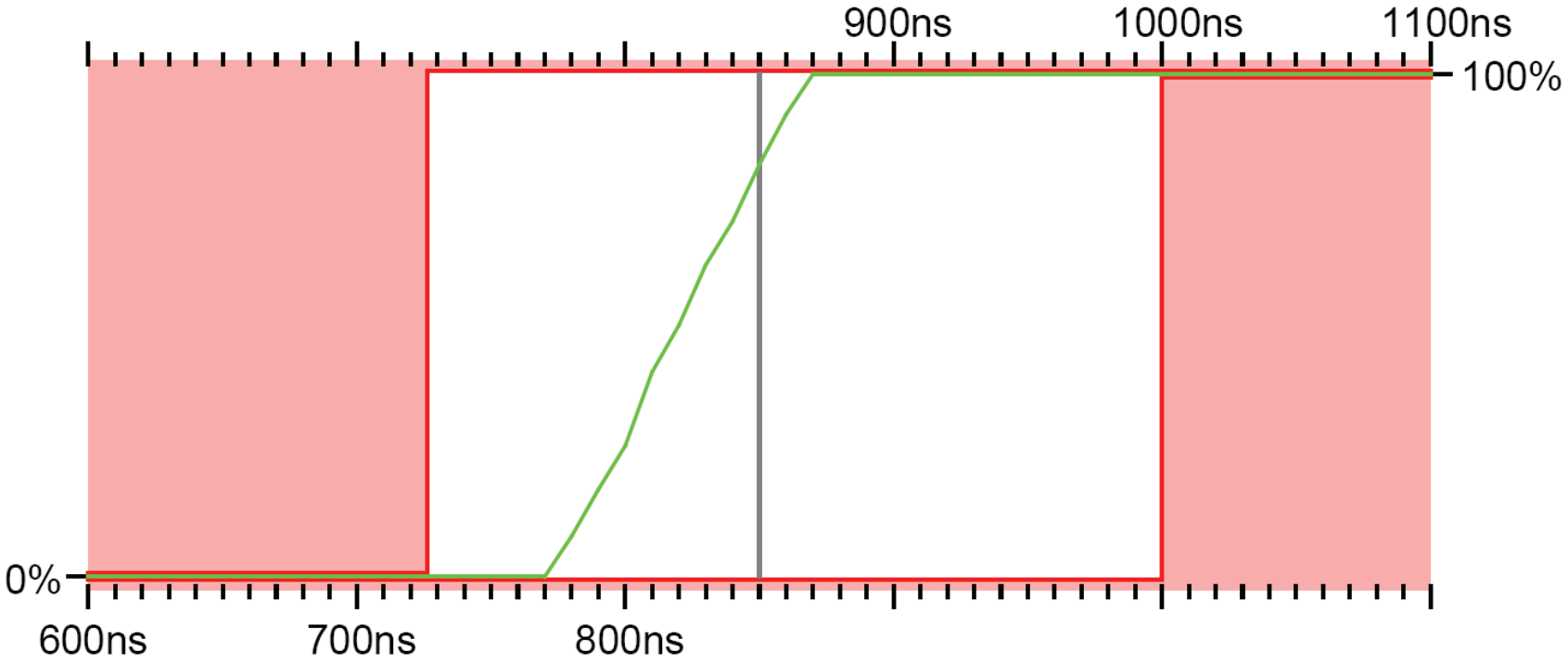
The ECSS Standard specifies nominal timeout of 850ns:

727ns must never disconnect

1000ns must always disconnect

Design A never disconnects at 840ns, always at 860ns, OK

Disconnection Timeout SpaceWire design B

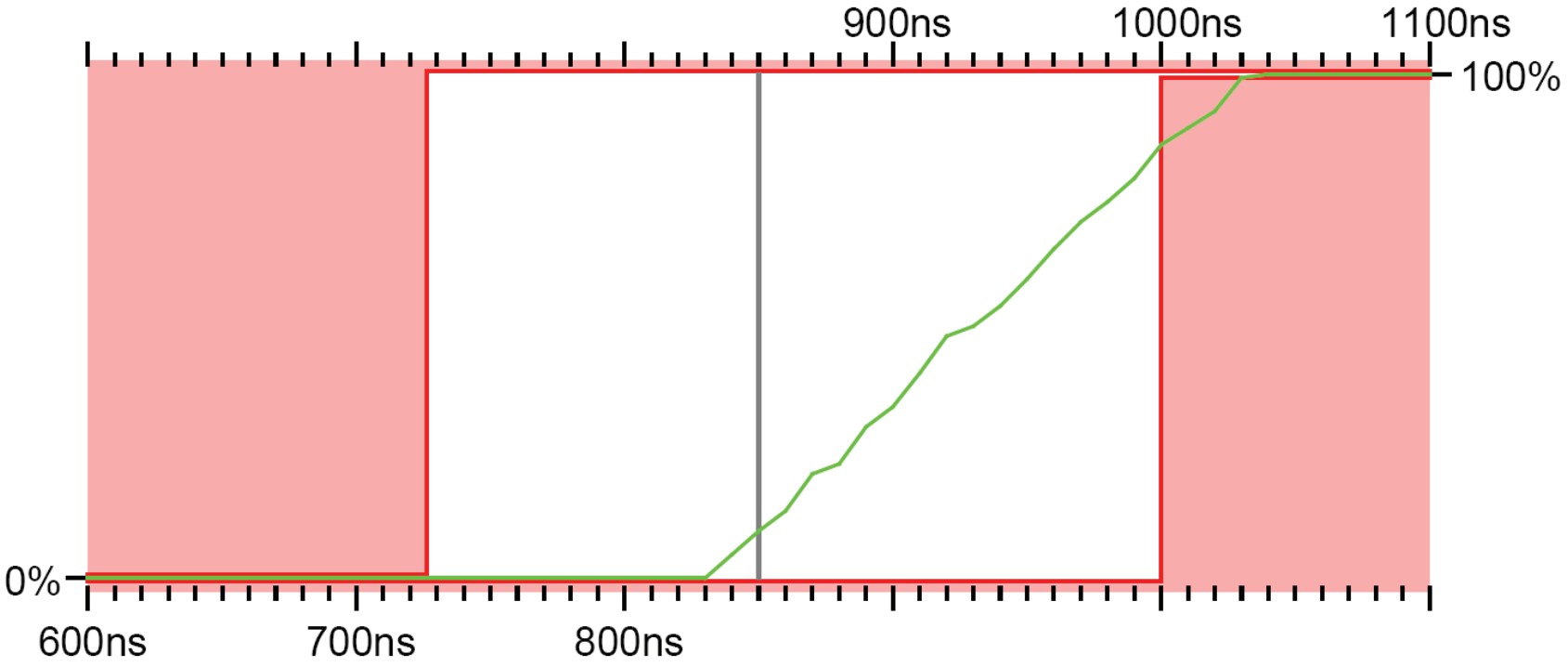


727ns must never disconnect

1000ns must always disconnect

Design B never disconnects at 770ns, always at 870ns, OK

Disconnection Timeout SpaceWire design C



Design C never disconnects at 830ns,
Does not always disconnect at 1000ns,

Fails to comply with ECSS standard

Conclusions

- Asynchronous design is not easy
- ‘Mature’ designs have been found to contain bugs in their time-related behaviour
- If these tests have not been completed successfully, can you honestly say that

1. Your design works ?
2. Your design complies with the ECSS standard ?

- Equipment is available to perform these tests, not only on individual devices but on large systems and with extraordinary consistency and resolution