FEC Striping Options for 100 Gb/s Backplane and Copper Task Force

IEEE P802.3bj Task Force

Chicago September 2011

Mark Gustlin – Cisco Systems

Supporters and Contributors

- Pete Anslow – Ciena
- Sudeep Bhoja – Broadcom
- Frank Chang - Vitesse
- Arthur Marris - Cadence

How to Stripe FEC Across PCS Lanes

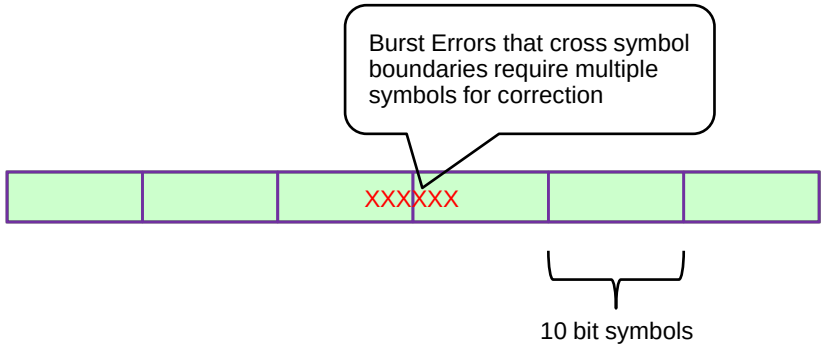
- If FEC is part of the 100 Gb/s backplane and copper solution, and if we decide to stripe the FEC blocks across physical lanes in order to reduce the latency penalty of FEC, then we need a mechanism to stripe the data and realign the data before decoding the FEC blocks
- A mechanism to deskew the physical lanes is needed first before the FEC blocks are decoded
- A couple of options to deskew the lanes using Alignment Markers were shown in: **gustlin_02a_0511.pdf**
- That won't be repeated in this presentation, but burst error spreading will be looked at

Error Spreading Concerns

- When taking a FEC block and distributing it across multiple physical lanes, you need to worry about burst error spreading (a single burst of errors being broken up into two or more bursts, and spreading out in the FEC block)
- This can impact a FEC's correction capability; when using a Reed-Solomon code for instance, it can require additional corrections when compared to the original error. When using a Firecode, it can defeat the correction capability.
- What is the best way to distribute a FEC block to multiple physical lanes and prevent/limit error spreading impacts?
- The key questions are tied to how we distribute data to the multiple physical lanes…typically we would distribute data based on the natural word width of a protocol (66b, 65b etc)
- With an RS FEC, it makes more sense to distribute based on the RS symbol size (m, for example 8 or 10 bits). Either distribute at the native symbol size, or any multiple of the symbol size.

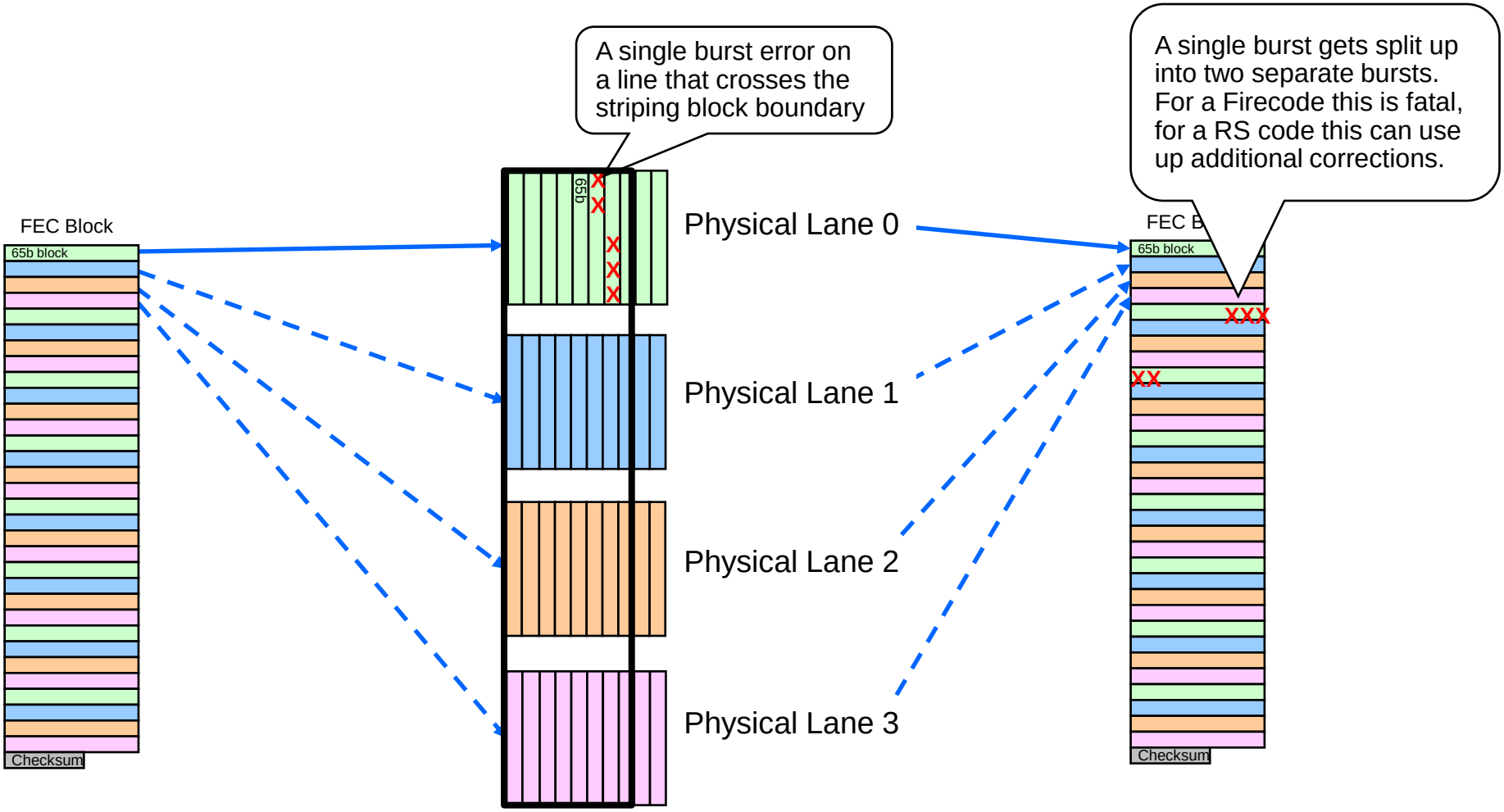
Reed Solomon Code Properties

- By adding *t* check symbols to the data, the RS code can detect *t* erroneous symbols and correct *t/2* bursts
- Some examples of RS codes have already been discussed in the study group, such as RS(276, 260, t=8), m=10 in wang_01_0511.pdf
  - In the best case, this code can correct up to 8 bursts, each up to 10 bits in length
  - In the worst case, if the burst errors cross the symbol boundaries, then only 4 bursts can be corrected



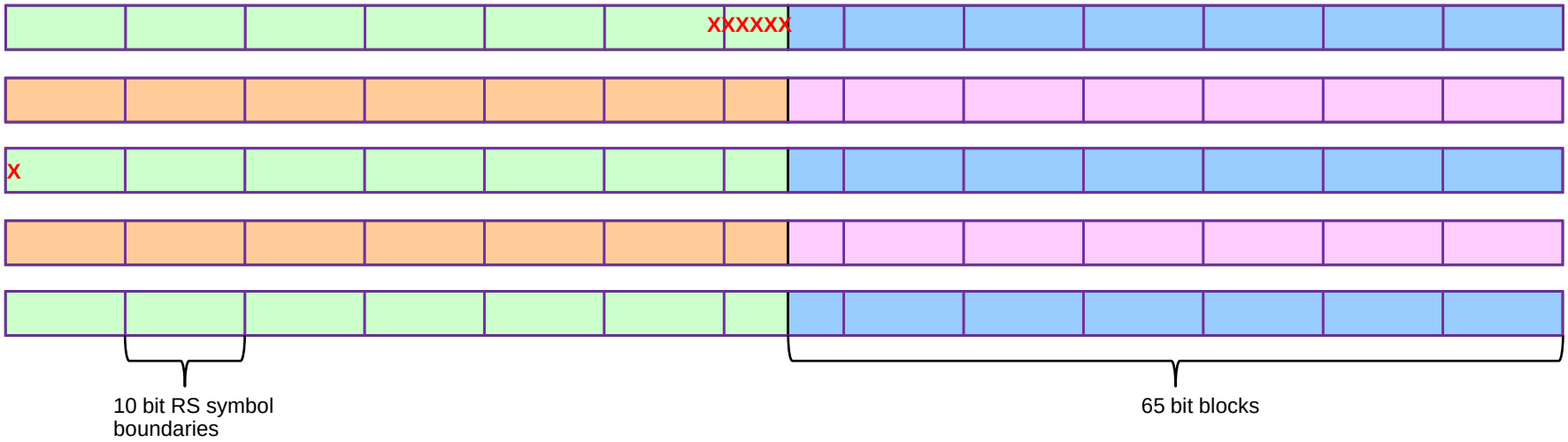
FEC Across Lanes Spreads Errors

- Burst errors are problematic for FEC blocks that are distributed across lanes
- This example shows blocks being distributed at 65b boundaries



FEC Across Lanes Spreads Errors

- Below shows a 10 bit RS codes with 65 bit striping
- A single burst error of 7 bits can require 3 symbols for correction



- Note that each color represents a different physical lane

FEC In the PCS

- When presenting gustlin_02a_0511.pdf, it was asked why not integrate the FEC into the PCS, before striping, and then correct after de-striping?
  - Assuming that we want the FEC striped across all physical lanes to reduce the latency
- This has benefits:

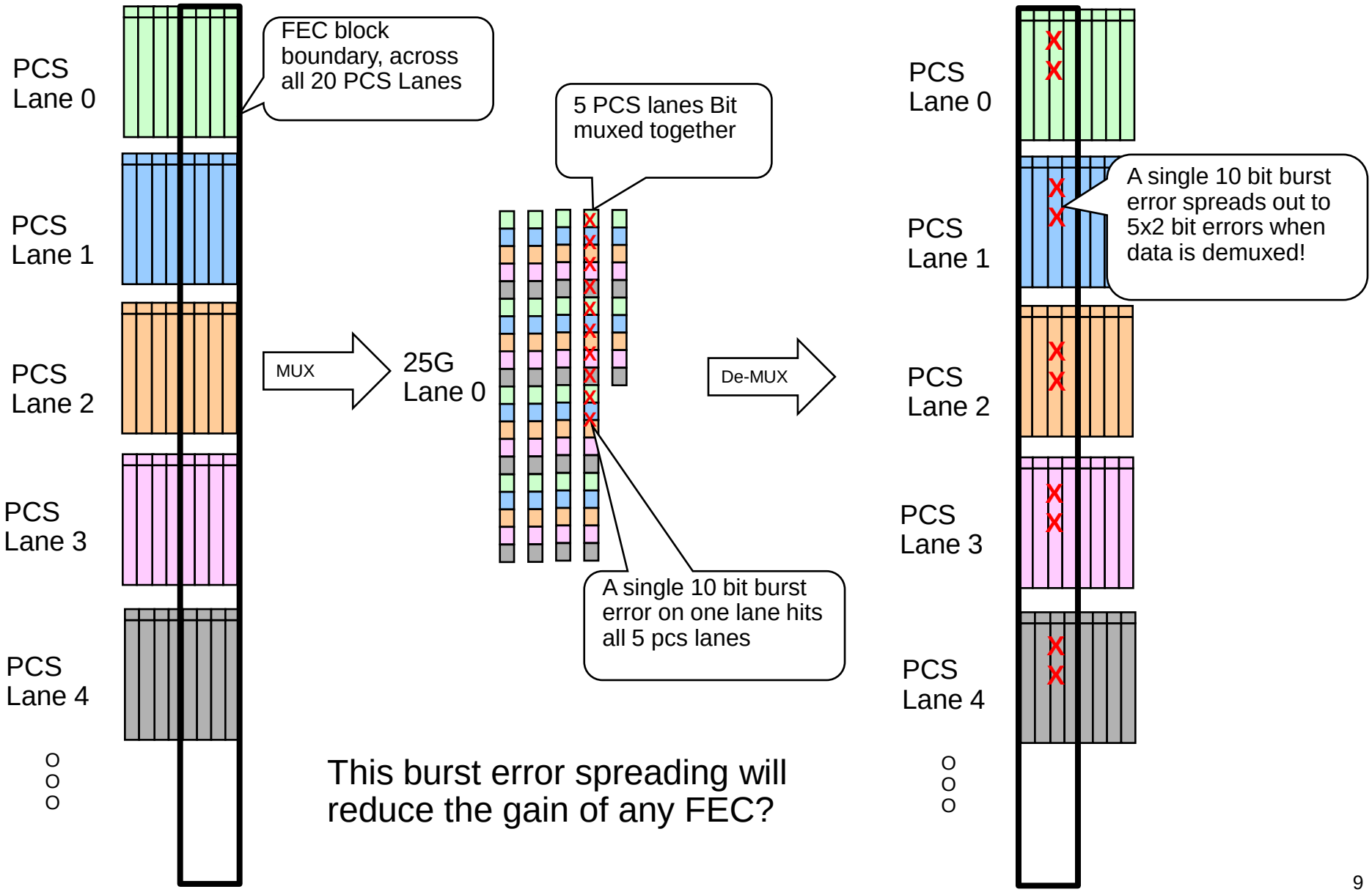
  Low latency is achieved

  Less logic overall (no repeat of any PCS functions)
- But it does have a big negative:

  A single burst error is split into 5 separate bursts (we expect burst errors, so this hurts the FEC burst immunity)

  The next slide depicts the error spreading

FEC Striping Within the PCS



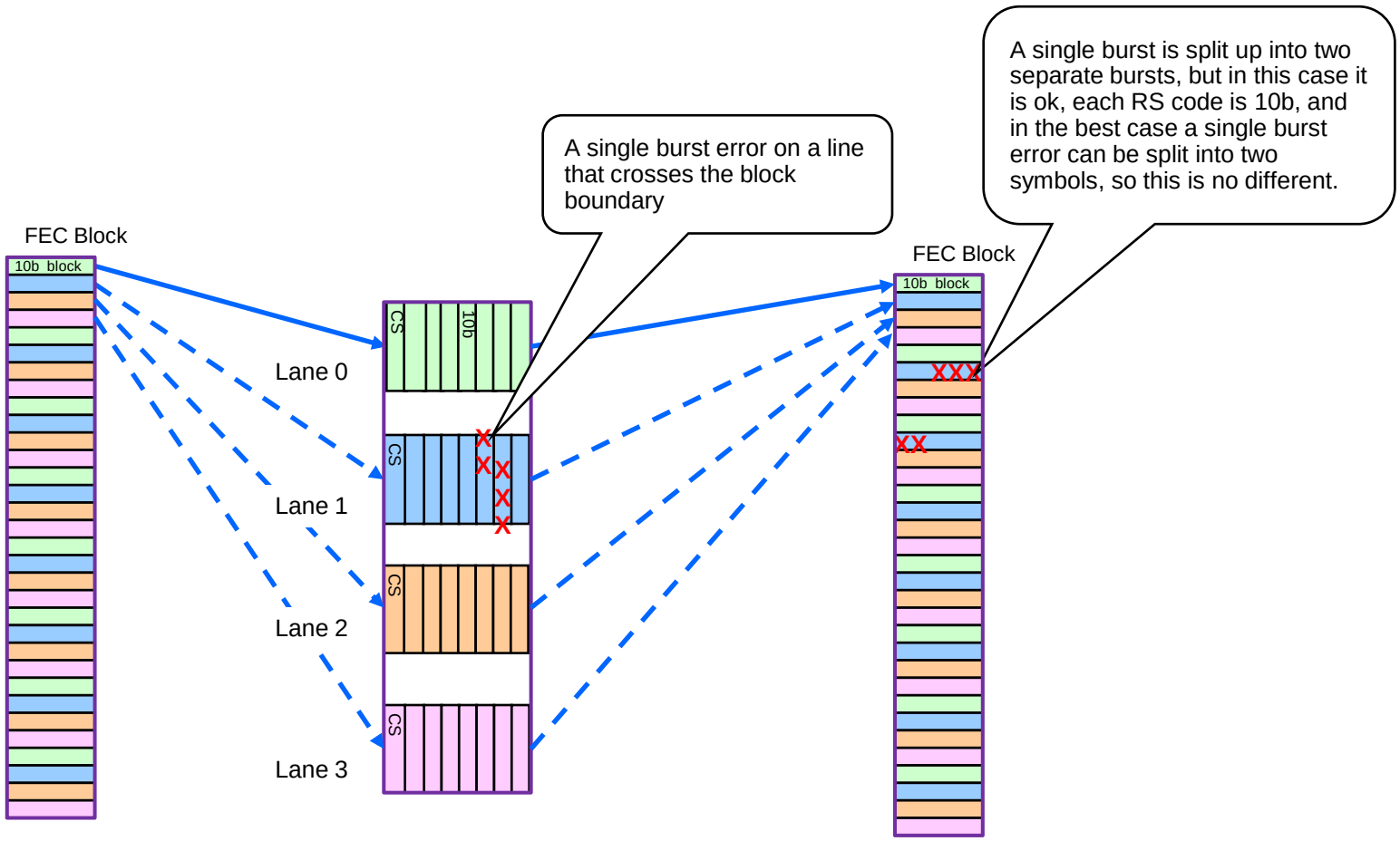
This burst error spreading will reduce the gain of any FEC?

Thoughts?

- Striping of a FEC block across multiple physical lanes causes burst error spreading (bursts are split up)
- This is fatal for a block FEC like the KR Firecode
  - KR FEC can correct a single burst error up to 11 bits, if that burst is split up then it cannot correct the errors
- This can greatly impact the correction capability of a Reed Solomon code, depending on how the FEC blocks are striped
- Creating FEC blocks within the PCS, and distributing the FEC blocks to all 20 PCS lanes, and then allowing the normal 100 Gb/s bit multiplexing seems fatal to the FEC burst error correction capability
- What will work well with a FEC code is distributed to multiple lanes?
  - It looks like Reed Solomon codes are good candidates for this, as long as striping is done multiples of the symbol size (m)

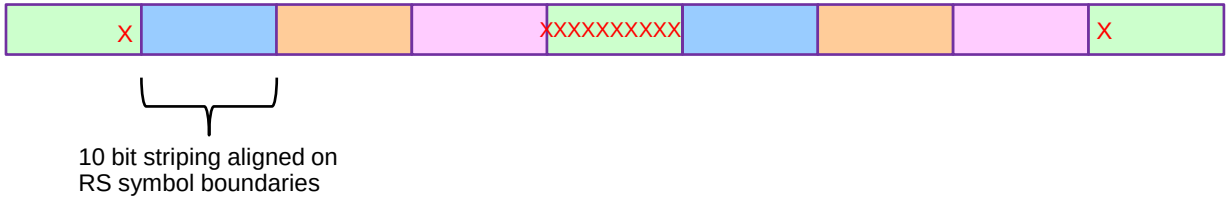
FEC Across Lanes Spreads Errors

- Distribute data at 10 bits for a 10 bit RS code, does that help? – Yes



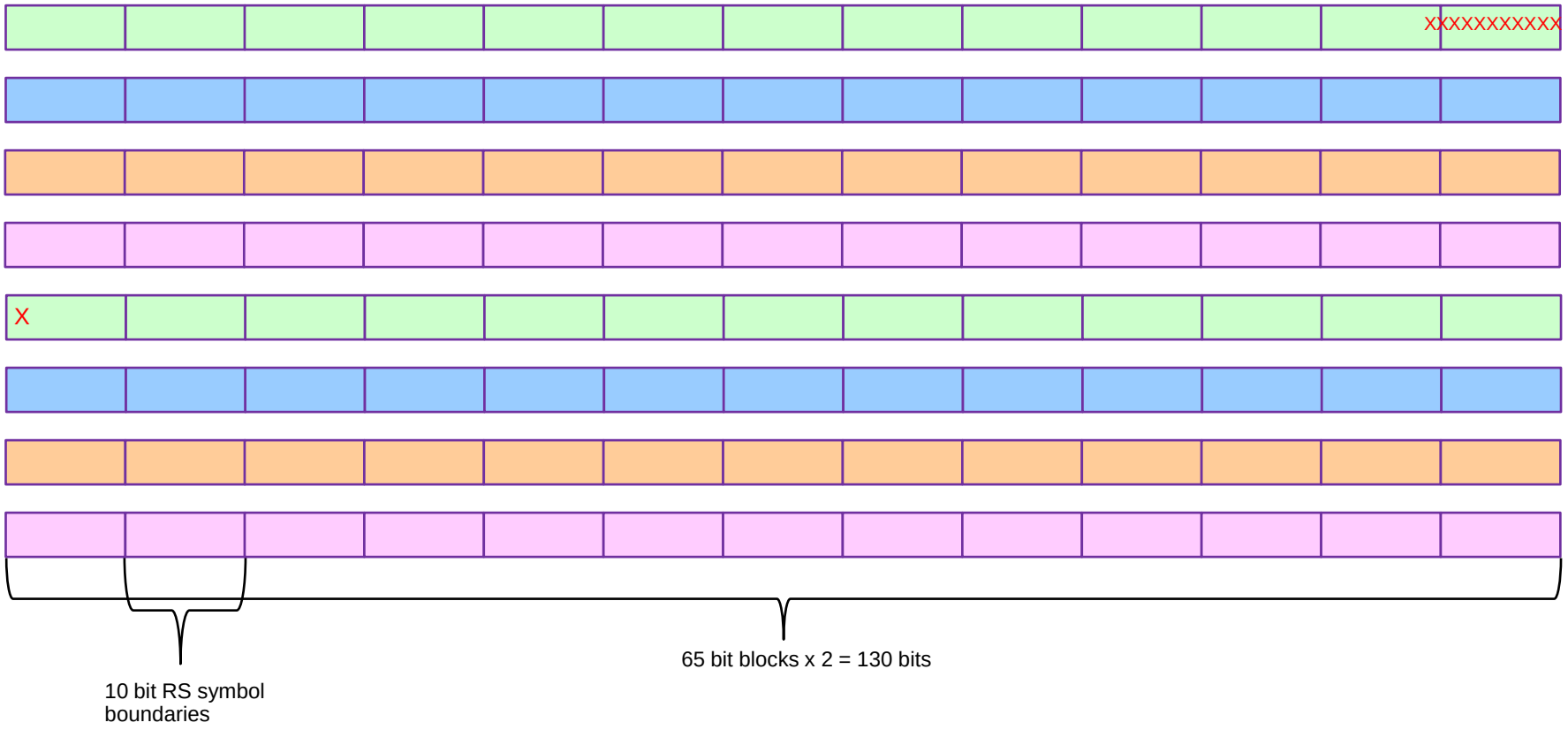
FEC Across Lanes Spreads Errors

- Below shows a 10 bit RS code with 10 bit striping
- A single burst error of 11 bits or less requires at most 2 symbols for correction
- A single burst error of 12 bits or more may require 3 symbols or more for correction
- This is the same even if the FEC block is not striped across multiple lanes



FEC Across Lanes Spreads Errors

- Below shows a 10 bit RS codes with 130 bit striping (13 x m striping)
- A single burst error of 11 bits or less requires at most 2 symbols for correction
- A single burst error of 12 bits or more may require 3 symbols or more for correction
- These properties are identical to the 10 bit striping case

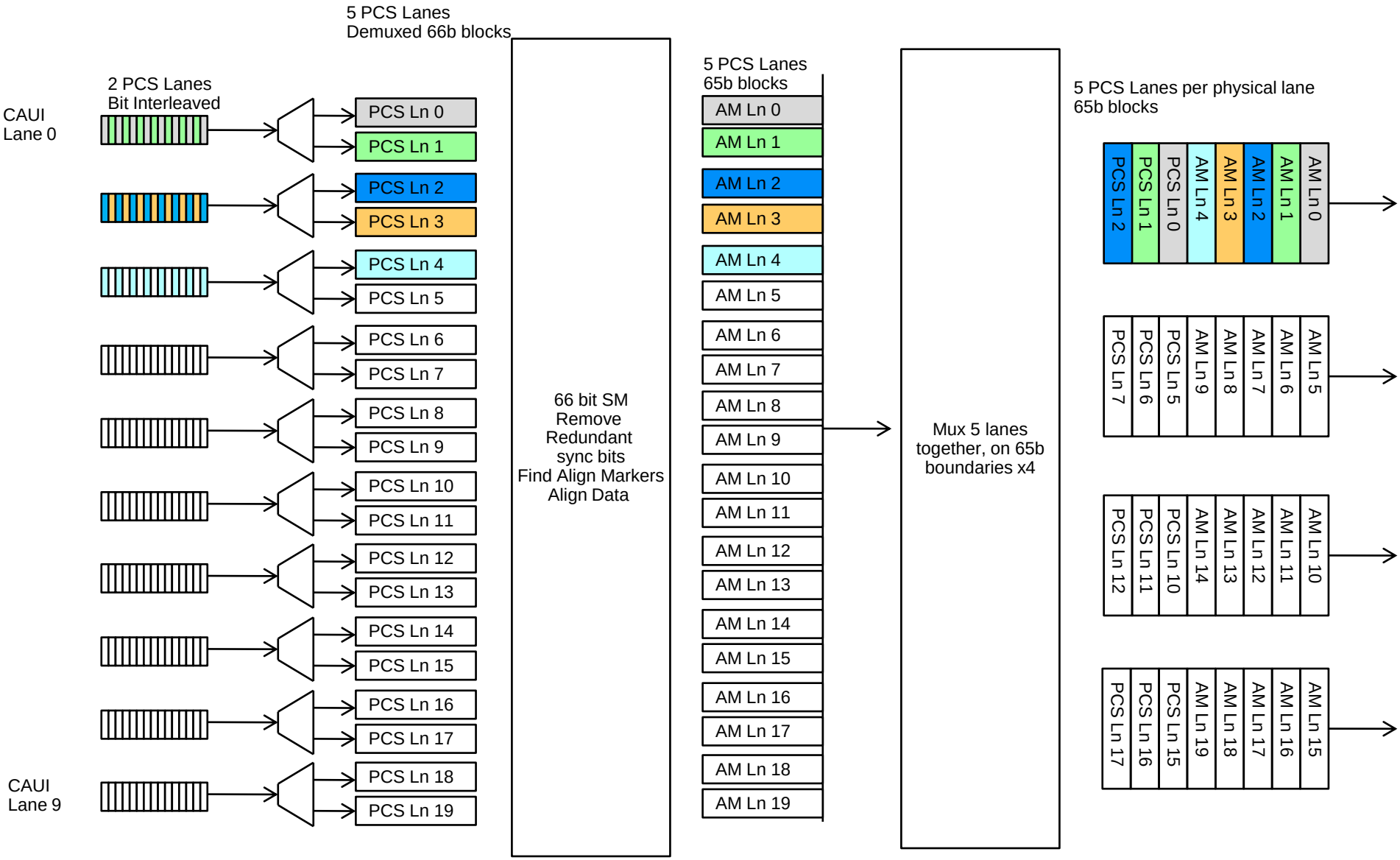


Generic Rules for Correction Capability

- Assuming that you use an RS FEC code, and you stripe at some multiple of the symbol code size (m), then these rules apply:
- This is the worst case, depending on how the errors align to the code boundaries, less symbols can be required

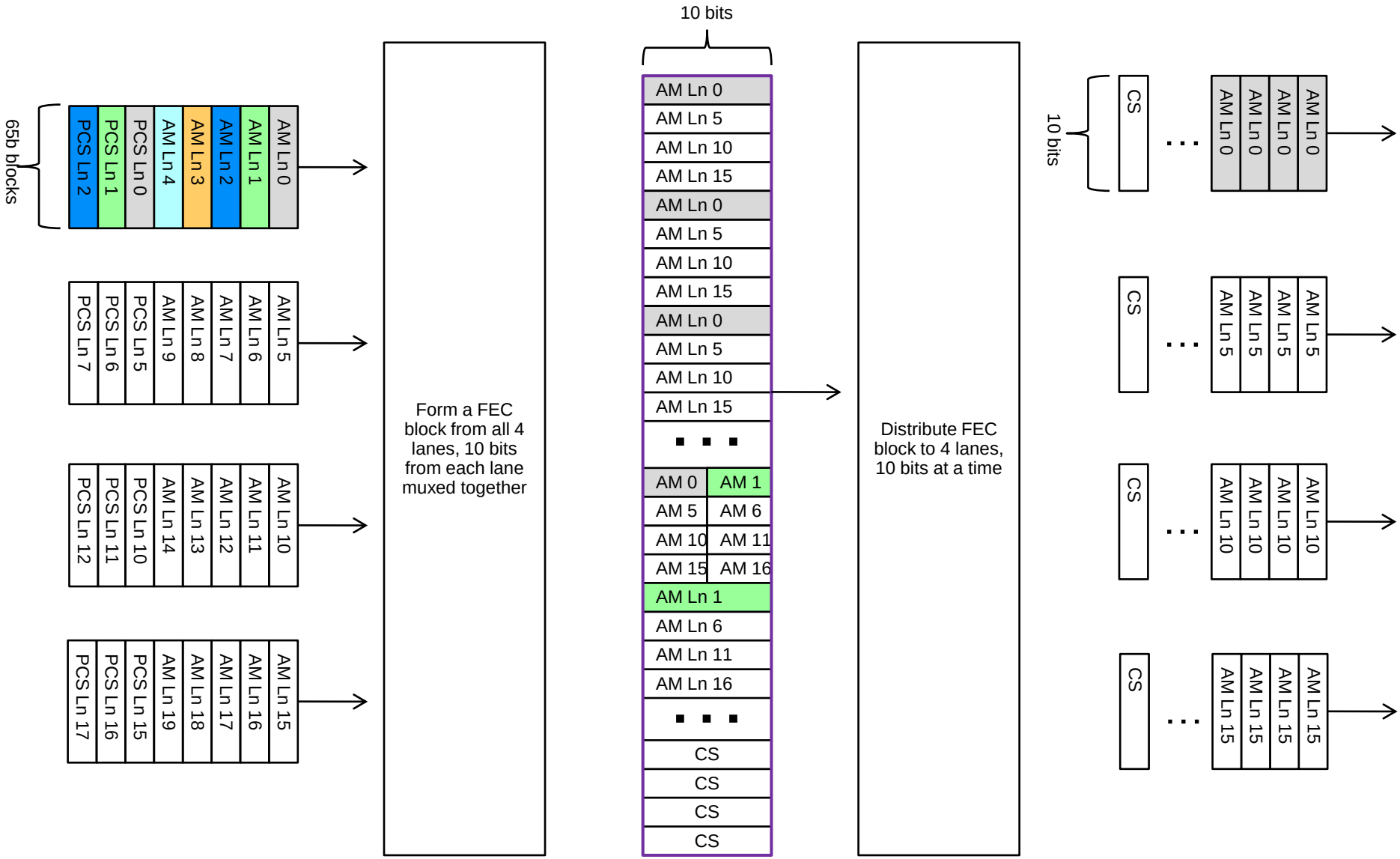
Burst Error Size (bits)	Symbols Required for Correction
1	1
2 to m+1	2
m+2 to 2*m+1	3

Data Processing, 10b Distribution Part 1

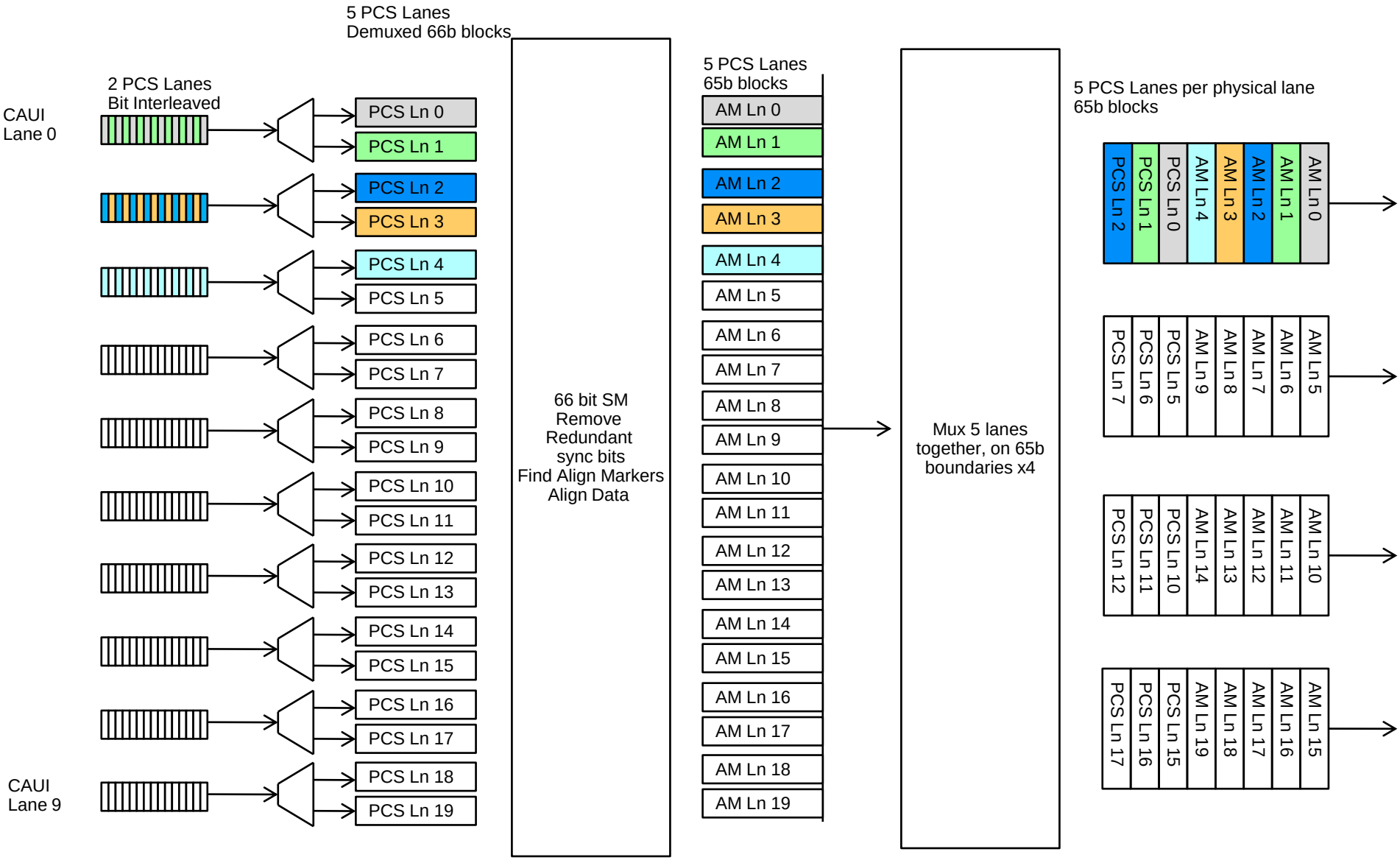


Note: Assumes a 10 bit RS code

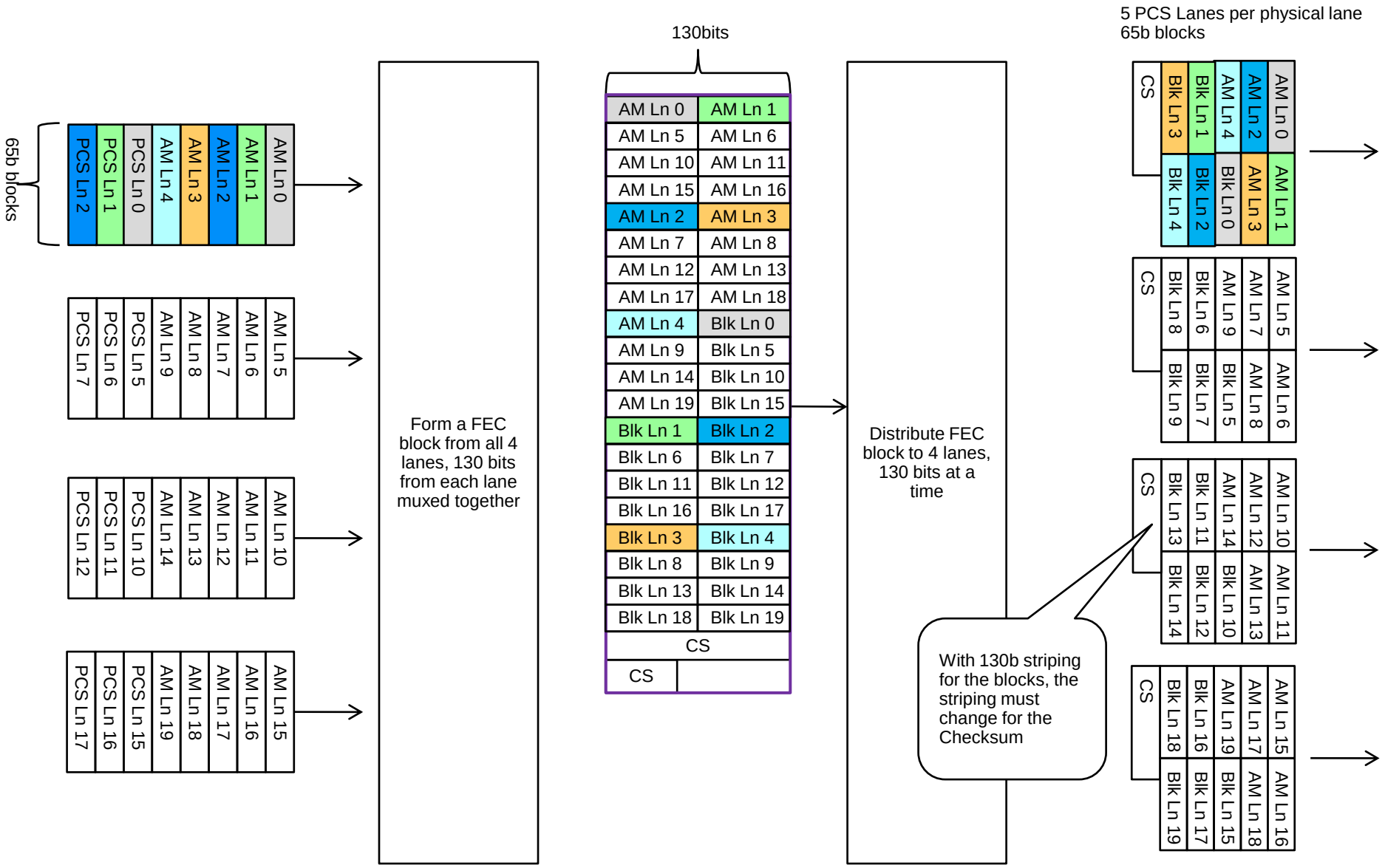
Data Processing, 10b Distribution Part 2



Data Processing, 130b Distribution Part 1



Data Processing, 130b Distribution Part 2



Note: Assumes a 10 bit RS code

Conclusions

- FEC blocks striped across physical lanes can work well for Reed Solomon codes as long as the striping is based on some multiple of the symbol size (m)
  - No degradation of the correction capability due to burst errors being split up
- 10 bit striping will work well (for a m=10 bit code), but Alignment Markers are split up
- 130 bit striping will work well (for a m=10 bit code), and keeps Alignment Markers recognizable, but the striping must change for the checksum (to ¼ of the checksum)
- An alternative is to stripe at ¼ of the checksum
  - For example in a RS(340, 320) m =10 code, the checksum is 200 bits, so you can stripe at 50 bits per lane, then the striping size stays constant

Thanks!