Interrupt-driven Maximum Likelihood Sequence Detection for High Speed Links

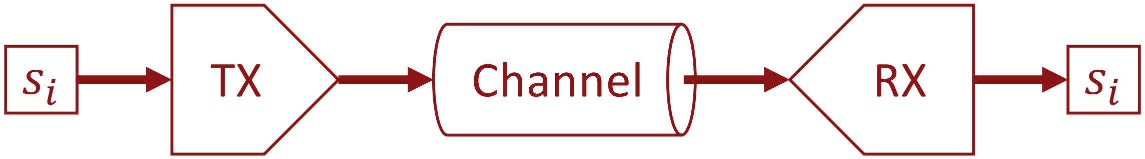
Zachary Myers, Stanford VLSI Group

August 31, 2023

Stanford University

Stanford University

High Speed Links



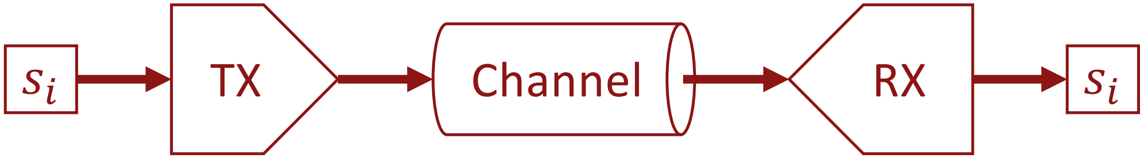
Historically, high speed links ‘leveraged’ the wideband bandwidth and low noise of wires to deliver high data rates at low latency

Each transmitted symbol is detected – ‘symbol-by-symbol’ to minimize latency and hardware complexity

No free lunch – symbol-by-symbol detection trades ‘decision’ SNR for simplicity

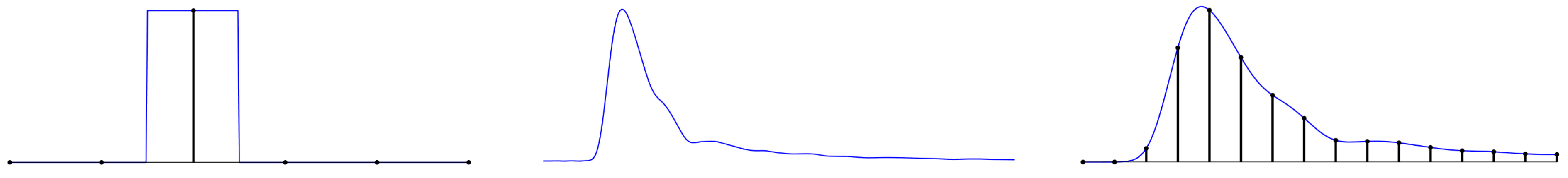
The key metric for High Speed Links is $\boxed{\text{Symbol Error Rate}}$

Nature always ‘smooths’ things out!



s_i → TX → Channel → RX → s_i

As data rates increase, the channel (wire) starts to ‘smooth’ out the transmitted symbol pulse, causing symbols to interfere with their neighboring symbols

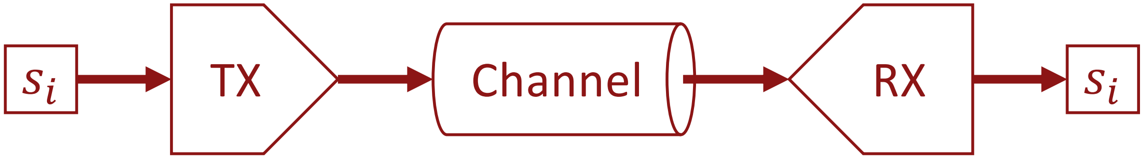


Pulse In

Channel Response

Pulse Out

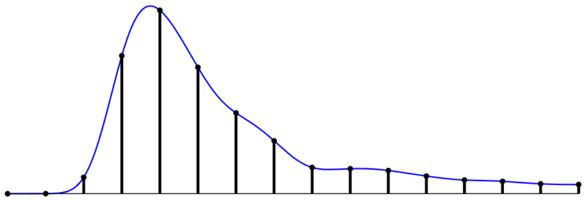
Nature always ‘smooths’ things out!



As data rates increase, the channel (wire) starts to ‘smooth’ out the transmitted symbol pulse, causing symbols to interfere with their neighboring symbols

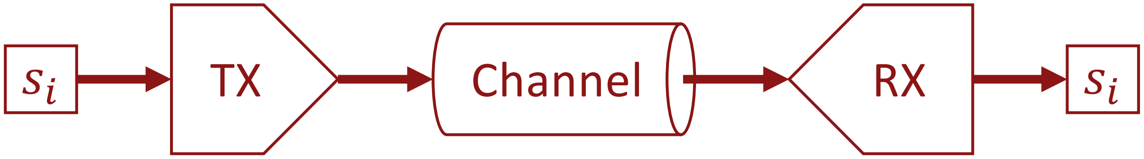
This effect is called ‘Inter-Symbol Interference’ (ISI)

Removing ISI is called ‘Equalization’

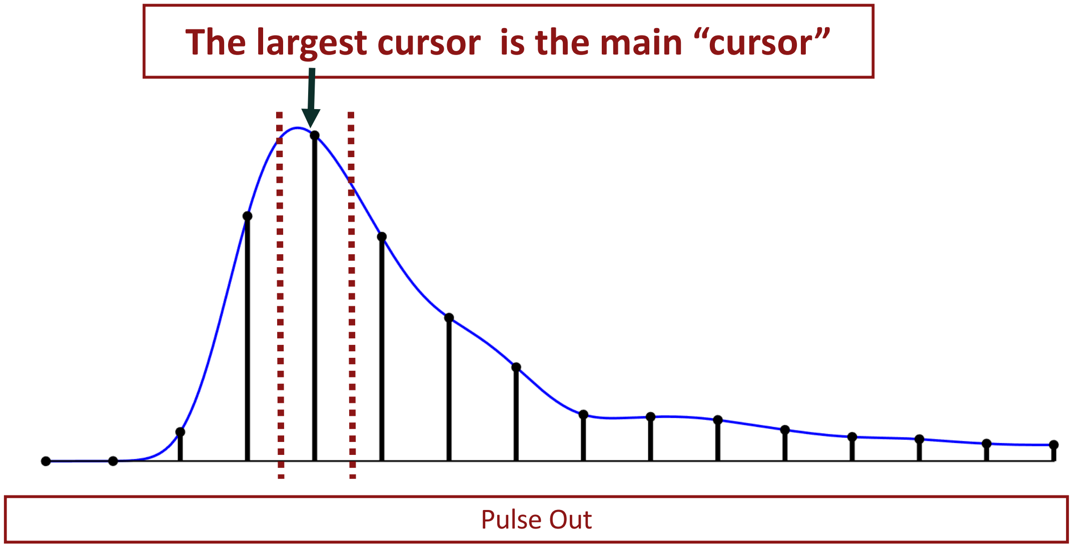


Pulse Out

Nature always ‘smooths’ things out!



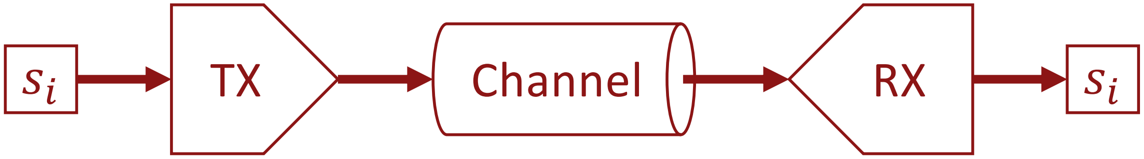
s_i → TX → Channel → RX → s_i

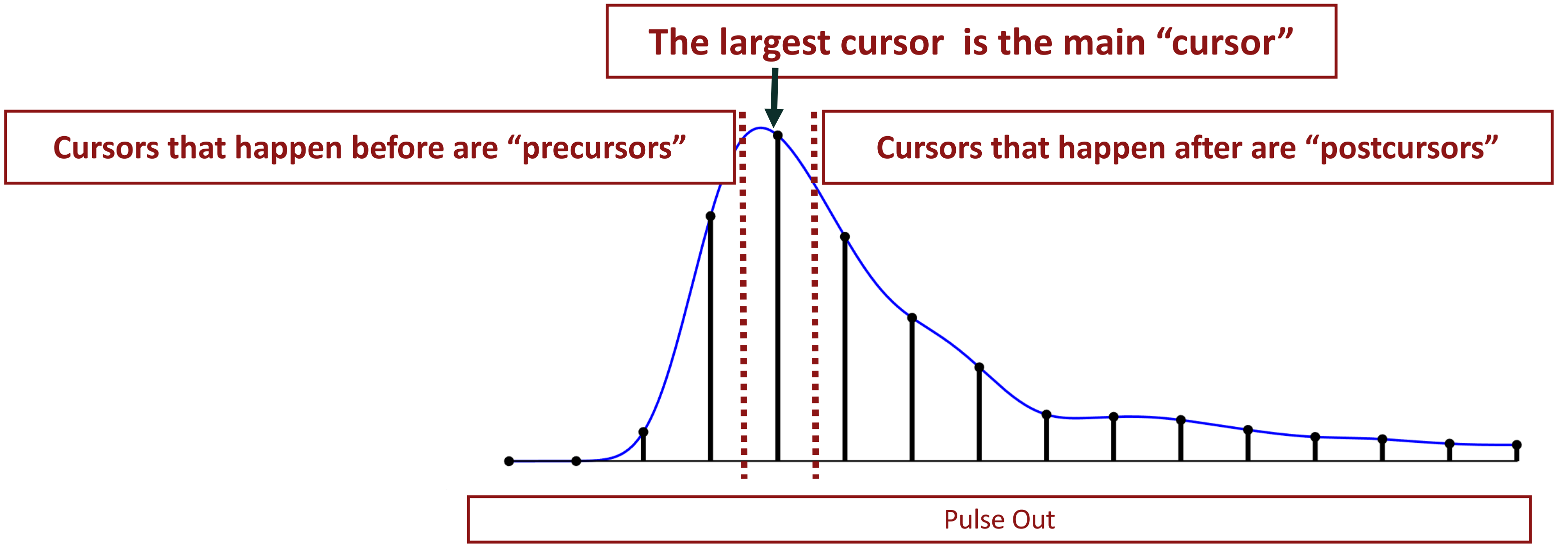


The largest cursor is the main “cursor”

Pulse Out

Nature always ‘smooths’ things out!




The largest cursor is the main “cursor”

Cursors that happen before are “precursors”

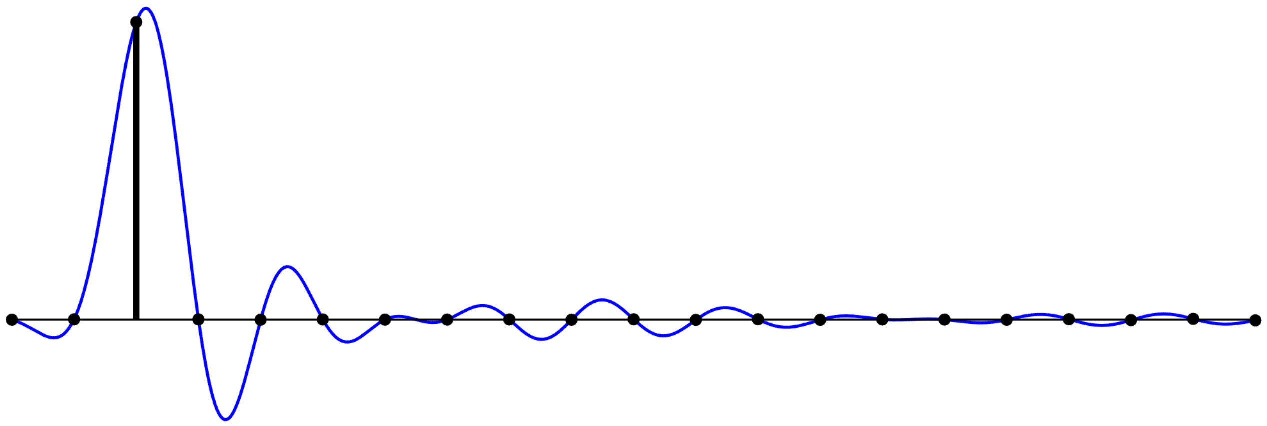
Cursors that happen after are “postcursors”

Pulse Out

Linear Equalization (Feed-Forward Equalization)

‘Sharpens the pulse back together’

Sharpen

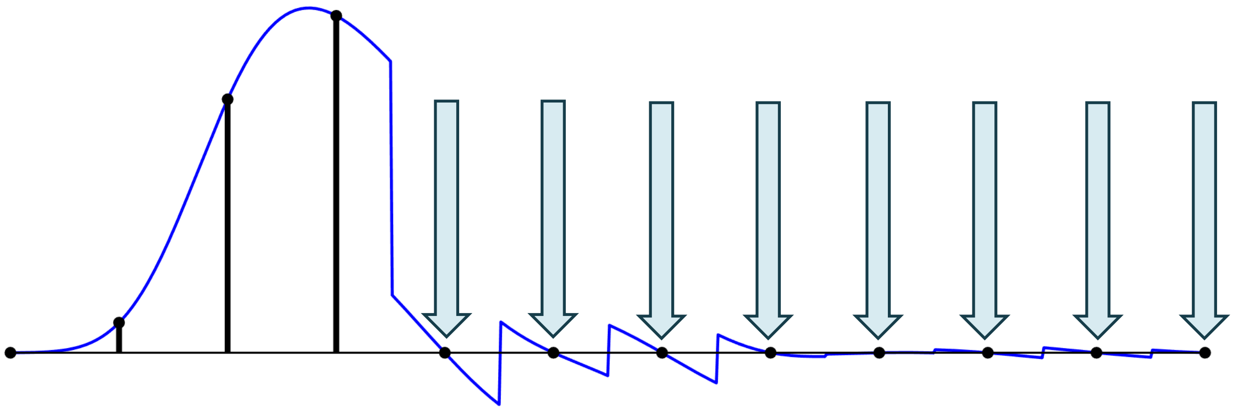


Linear Equalization

Nonlinear Equalization (Decision Feedback Equalization)

'Subtract out the effects of previous pulses'

Subtract

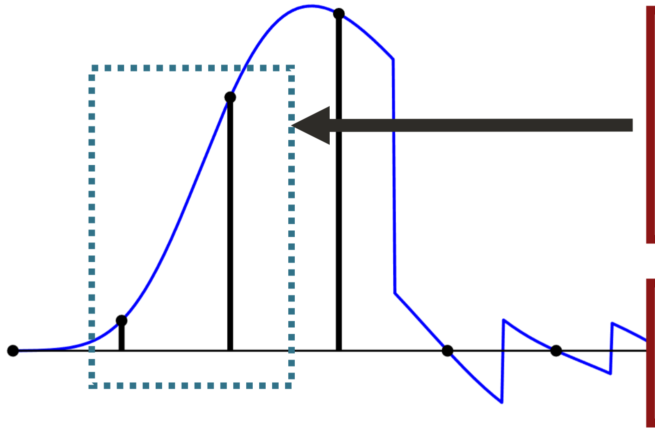


Nonlinear Equalization

Nonlinear Equalization (Decision Feedback Equalization)

‘Subtract out the effects of previous pulses’

Subtract

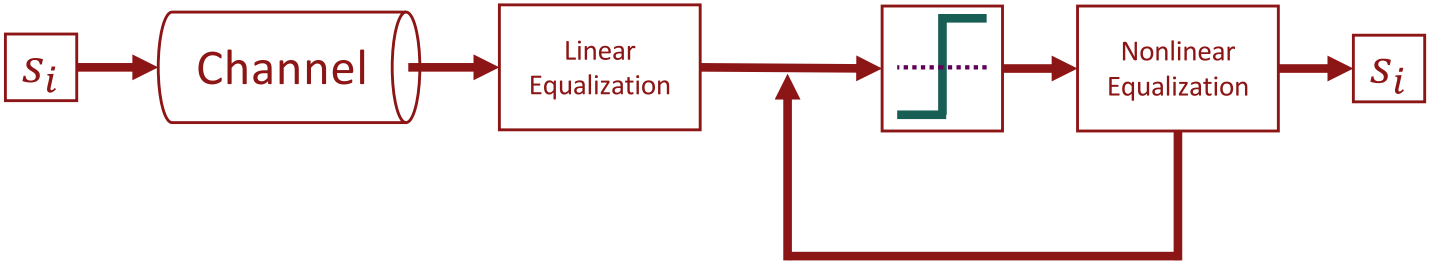


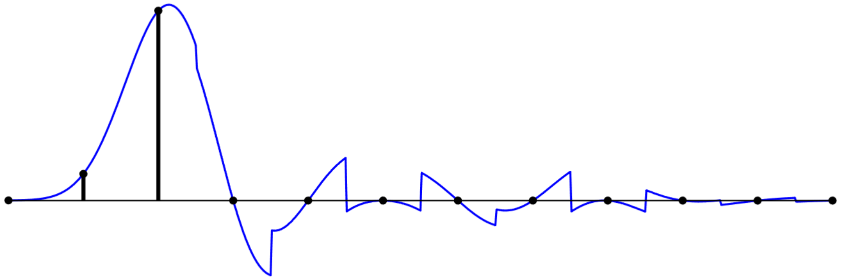
Note: Nonlinear Equalization cannot remove ISI from future symbols

Both FFEs and DFEs are required

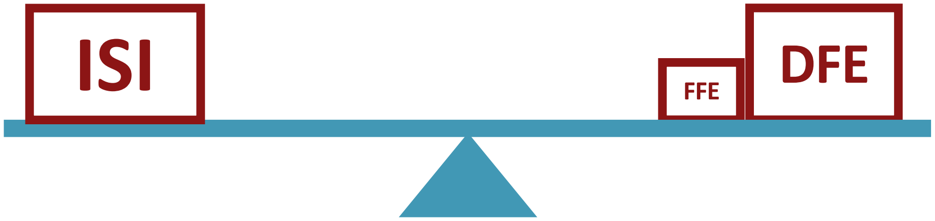
Nonlinear Equalization

The Balance of the Canonical High Speed Link



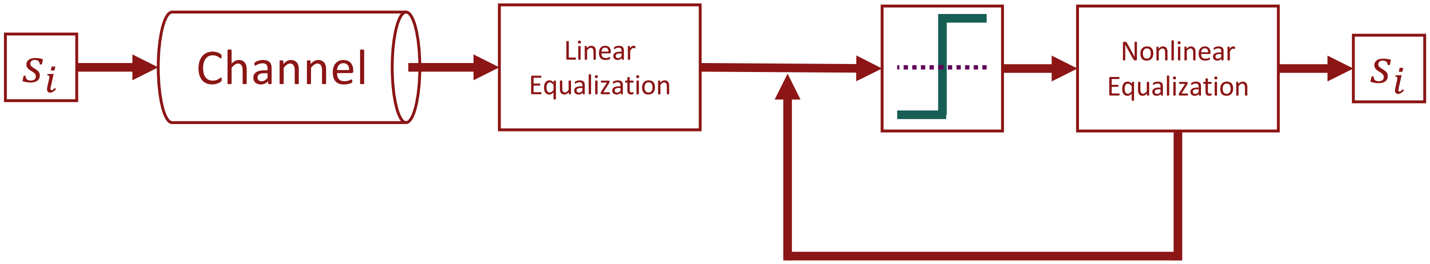


Canonical Equalization



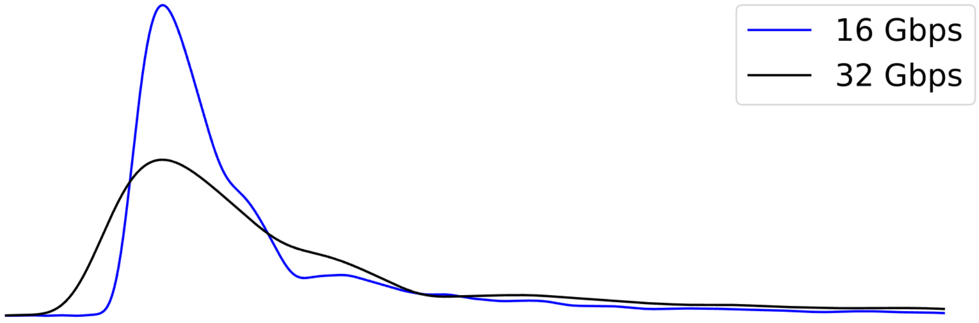
A bit of FFE to reduce the precursor…
A lot of DFE to remove postcursors

The Problem: There is never enough bandwidth!



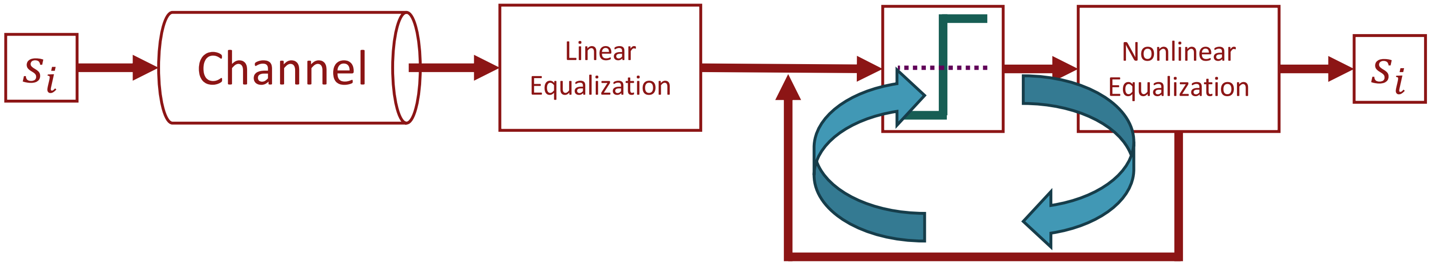
Machine Learning drives enormous demand for increased ‘package’ to ‘package’ bandwidth

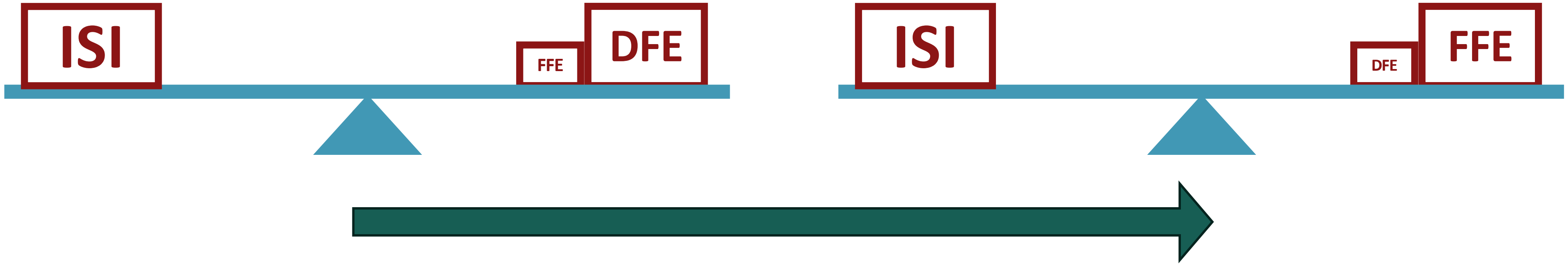
As the links run faster, the smoothing of the channel increases, and the current equalization strategy fails



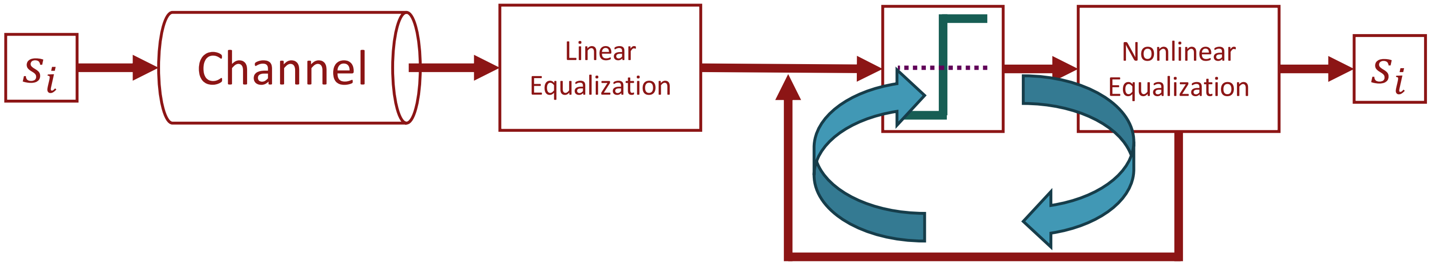
Channel vs Gb/s

And it’s no longer free ☹





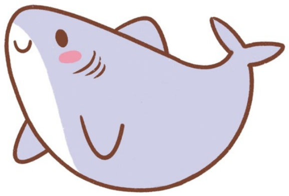
And it’s no longer free ☹



Current State Of Electrical Links

Symbol Rate ↑ → Symbol Error Rate ↑

ISI’s are friends, not food

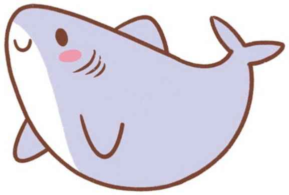


Symbol-by-Symbol Detection ‘fights’ the channel…

But the channel (ISI) is ‘**signal**’ – it still encodes information about what is transmitted

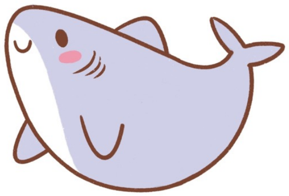
By moving from ‘symbol-by-symbol’ detection to ‘sequence’ detection, you embrace the channel’s ISI.

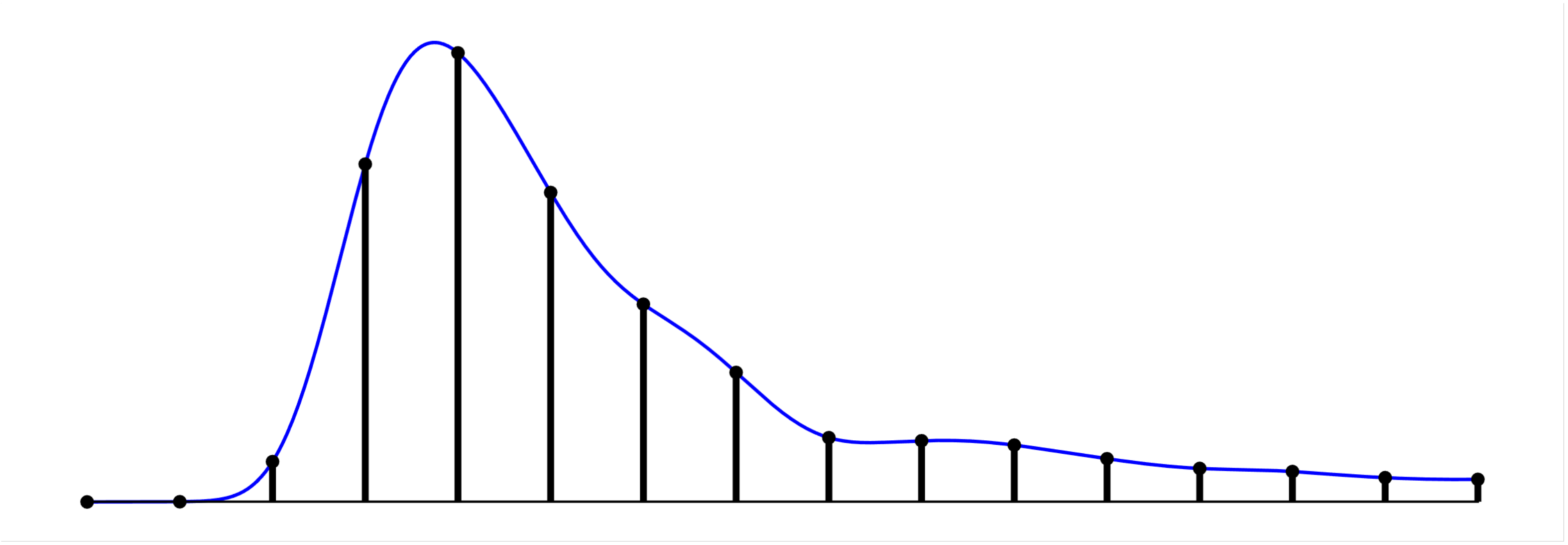
ISI’s are friends, not food



Pulse Response

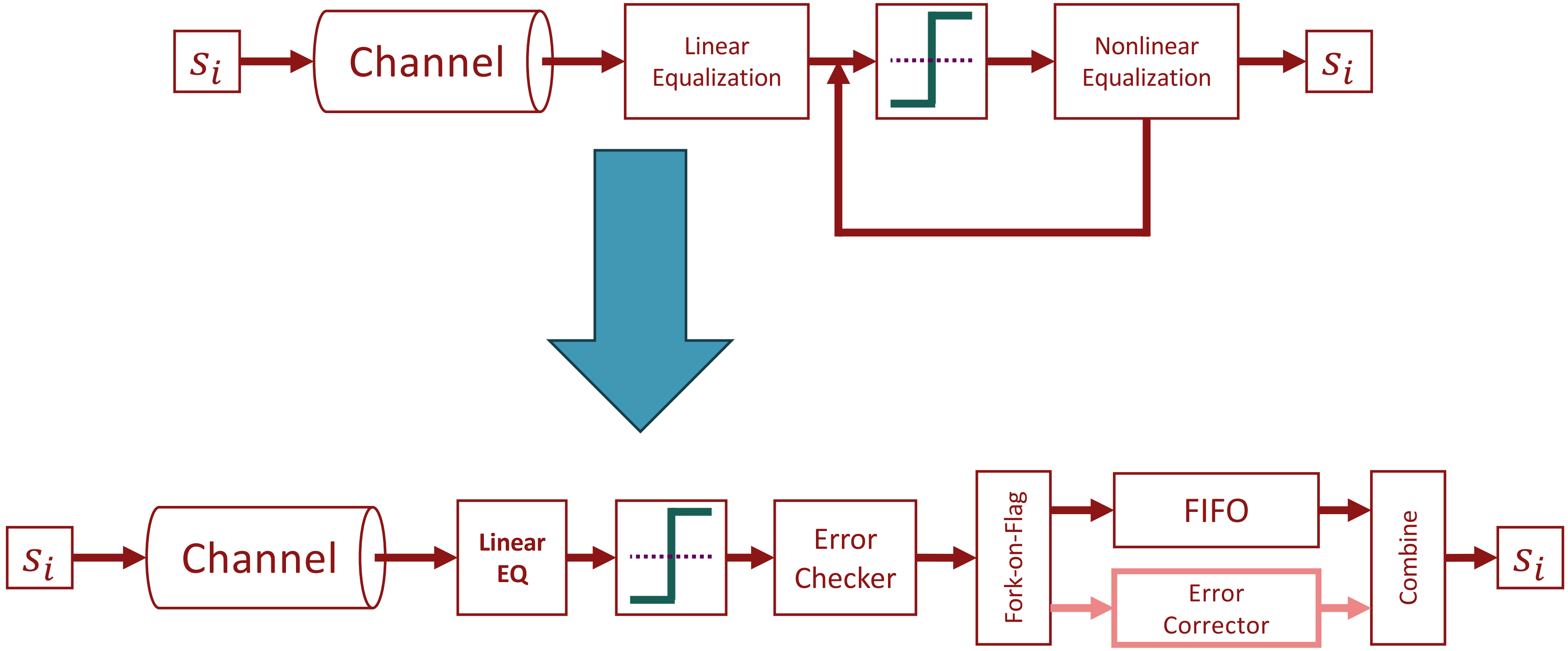
ISI’s are friends, not food



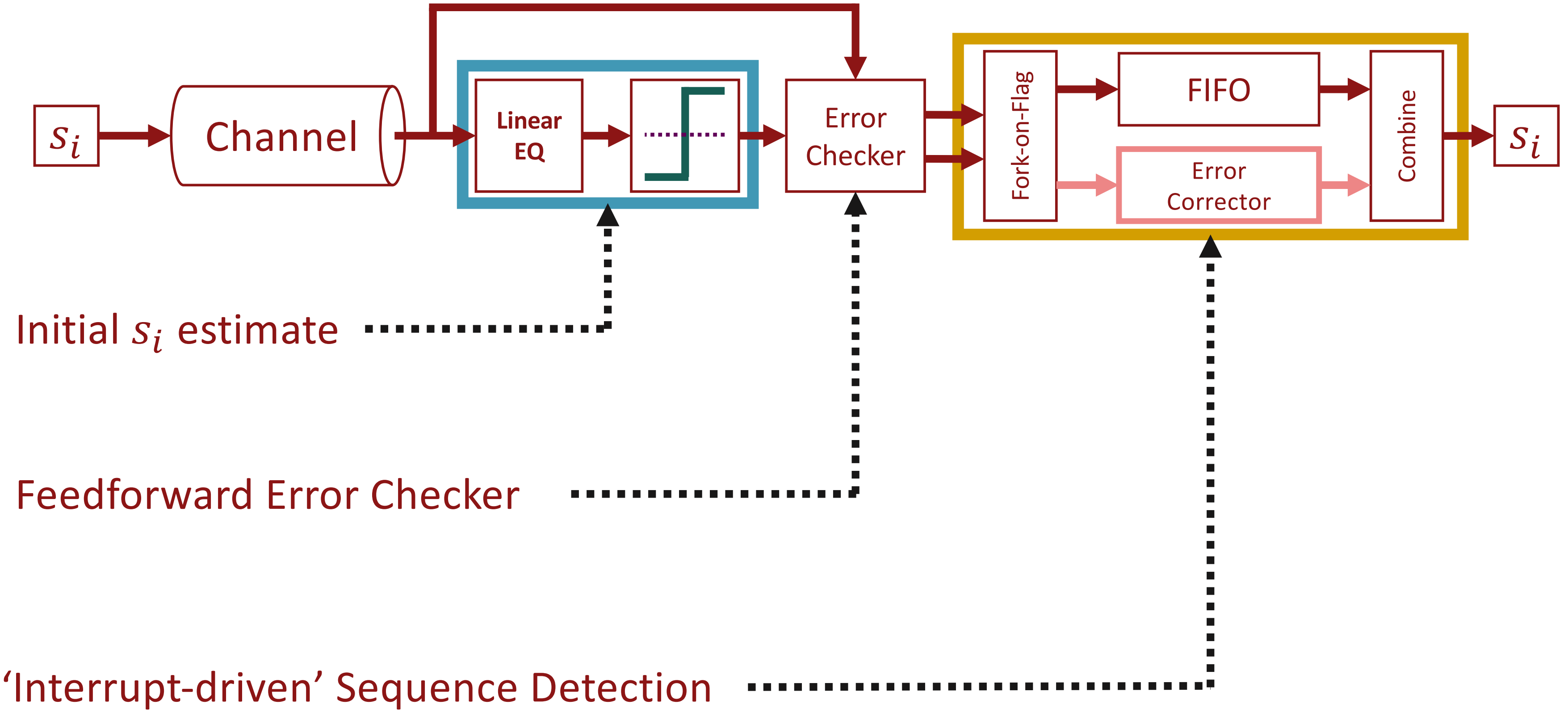


Pulse Response

Interrupt-driven MLSD-based Links



Interrupt-driven MLSD-based Links

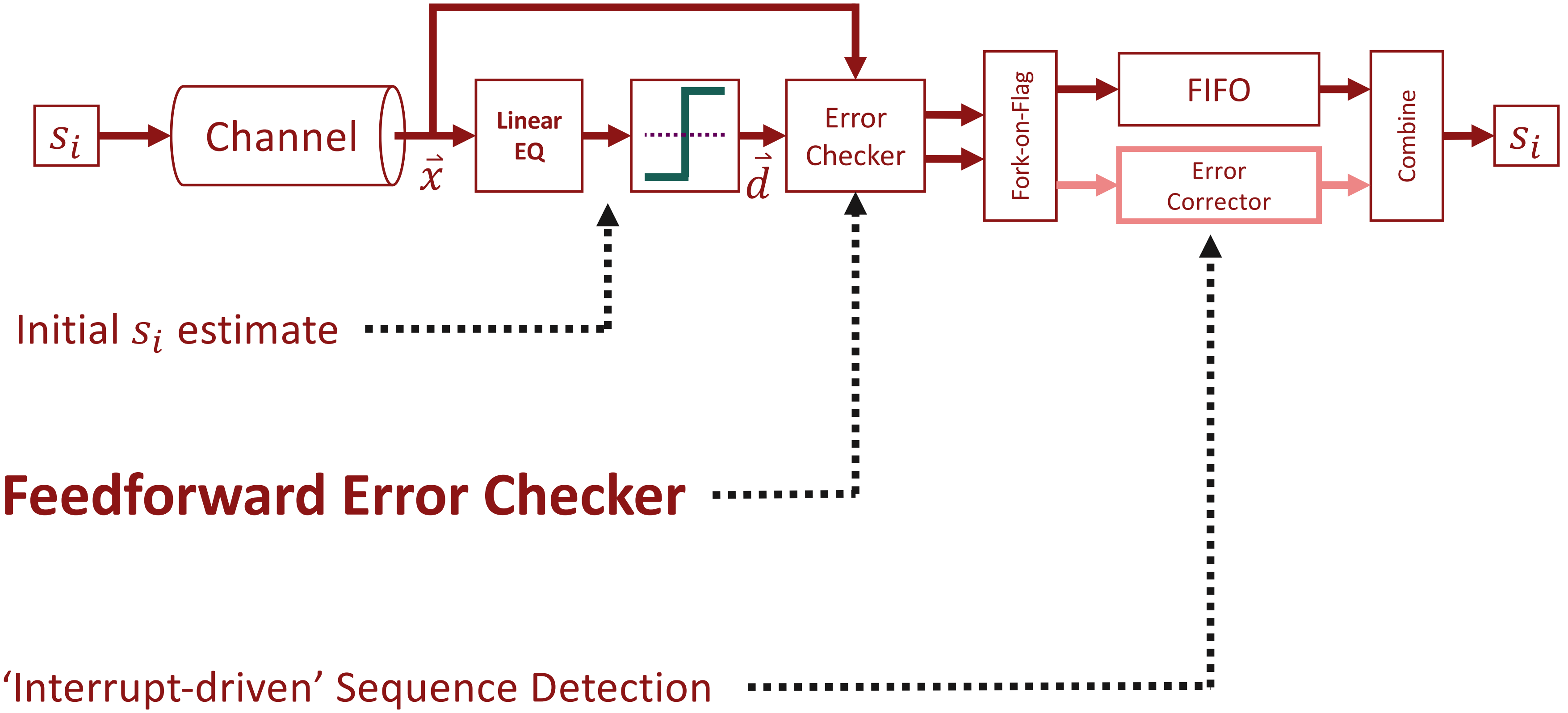


Initial s_i estimate

Feedforward Error Checker

‘Interrupt-driven’ Sequence Detection

Interrupt-driven MLSD-based Links

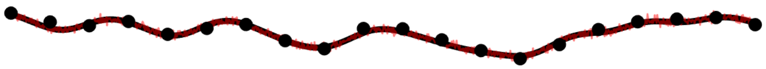


Initial s_i estimate

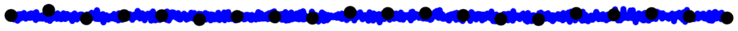
Feedforward Error Checker

‘Interrupt-driven’ Sequence Detection

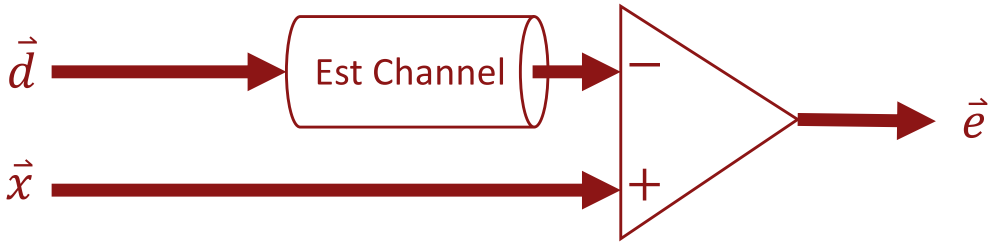
Error Free Residual Error Signal



Raw Input and Recreation



Residual Error Signal



$\vec{d}$

Est Channel

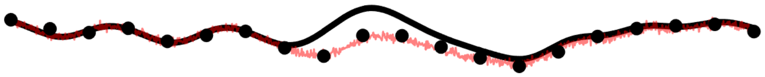
$-$

$+$

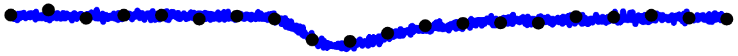
$\vec{x}$

$\vec{e}$

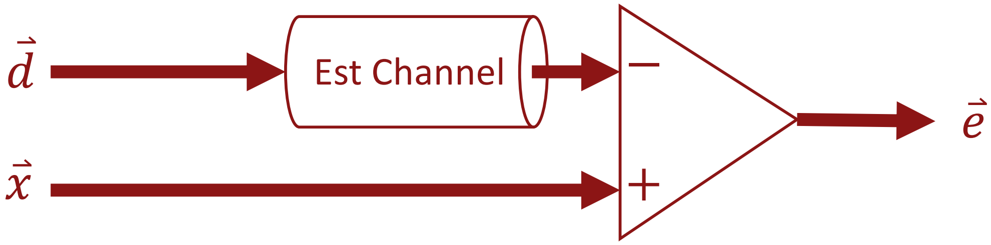
Residual Error Signal with a Single Error



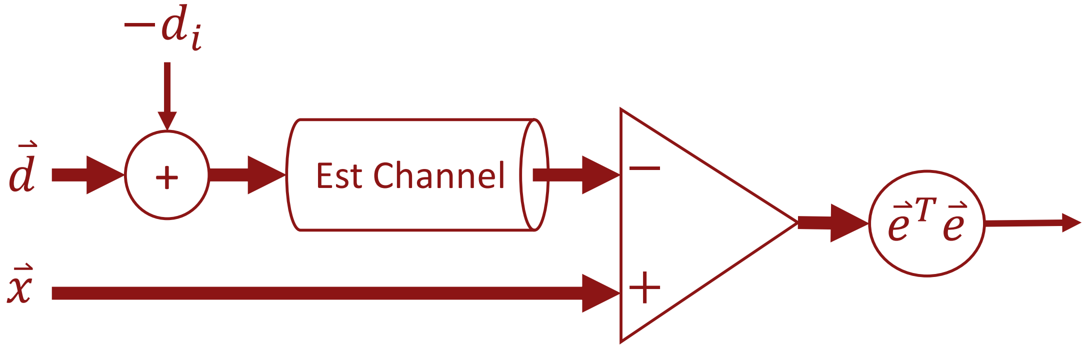
Raw Input and Recreation



Residual Error Signal



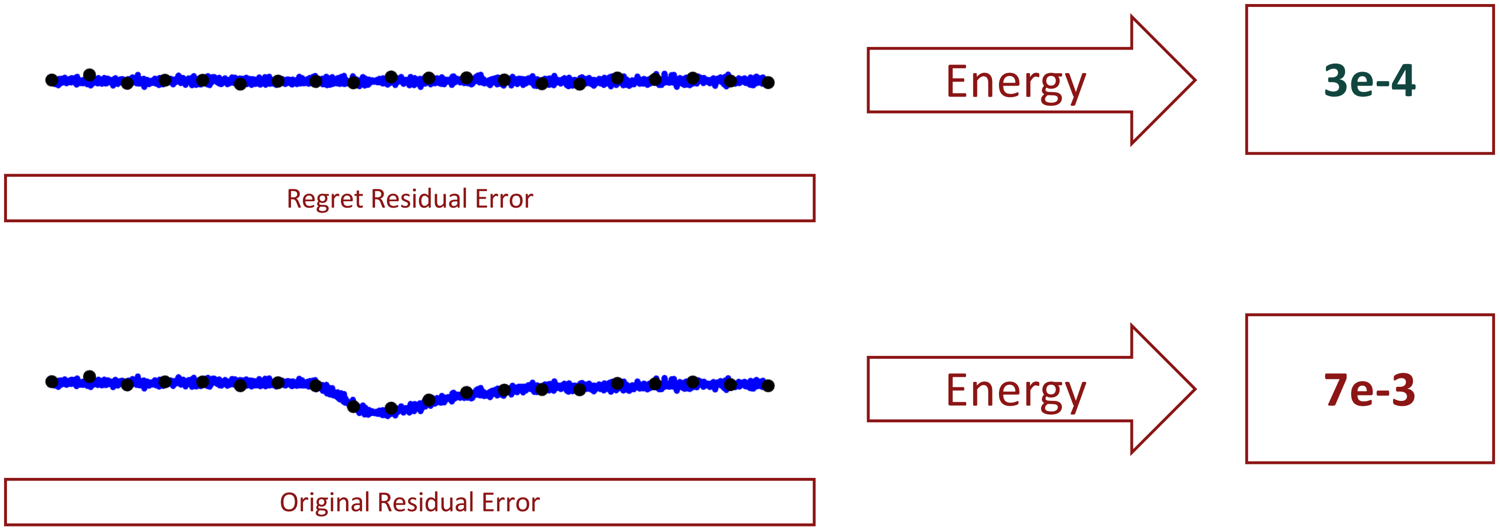
Feed Forward Error Checking



Here is a simplified form of our detection scheme, this version, we call the ‘regret’-based detector.

For each symbol, you check whether the other decision would’ve led to a smaller residual error. Basically, do you regret the choice you’ve made?

Feed Forward Error Checking



Energy

3e-4

Regret Residual Error

Energy

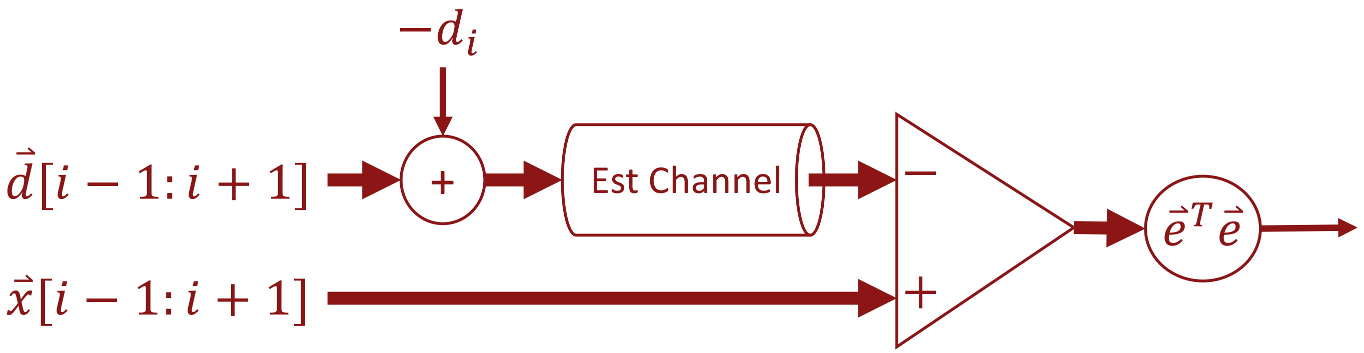
7e-3

Original Residual Error

We compare between the ‘regret’ residual error and the original residual error by calculating their energy.

If the altered residual error has a lower energy than the original, then the detector raises a flag.

Feed Forward Error Checking

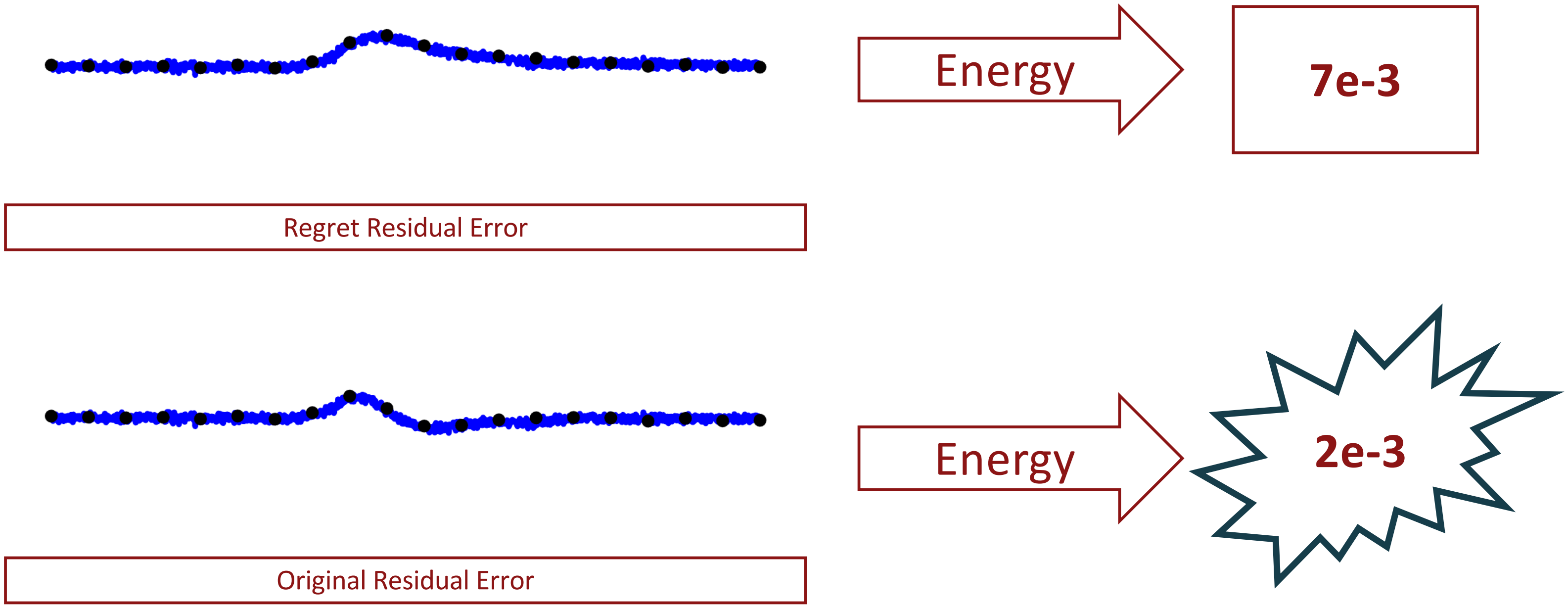


In practice, the detector only compares the energy over a small number of symbol times… acceptable given most of the energy is in the first precursor, the main cursor and the first postcursor.

BUT there is no free lunch… Errors also have ISI ☹.

SO we extended this type of detector to cover cases with bursts of errors. **Ask us me if you want to know how!**

Residual Error with Multiple Errors



Energy

7e-3

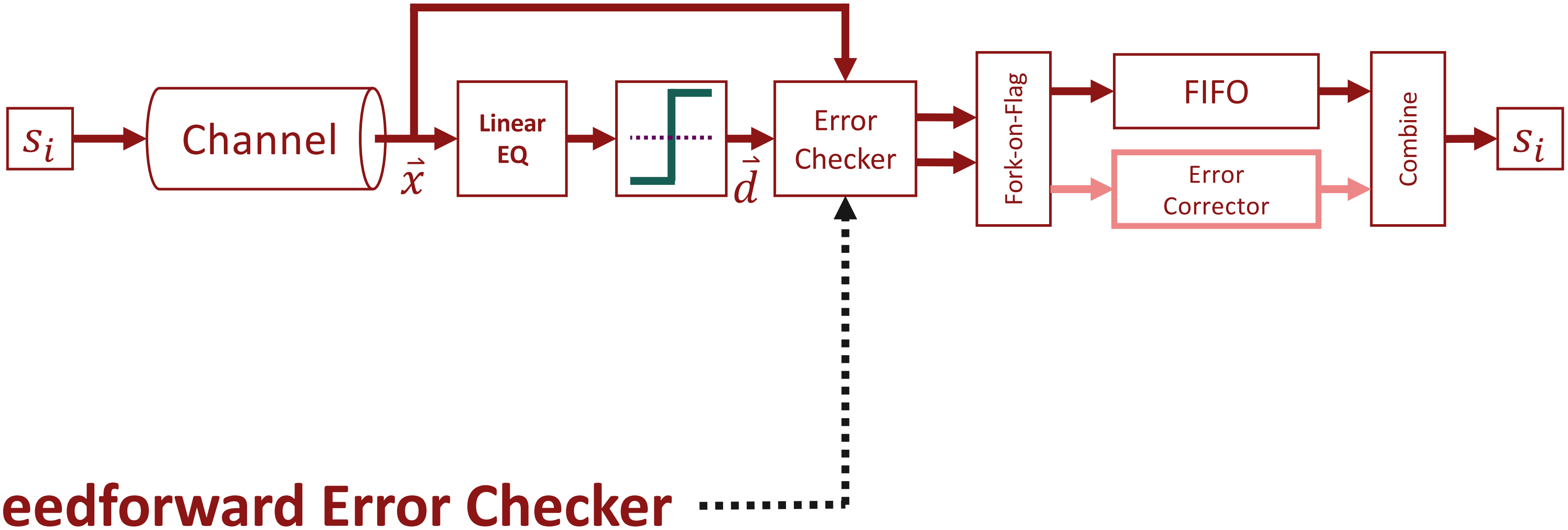
Regret Residual Error

Energy

2e-3

Original Residual Error

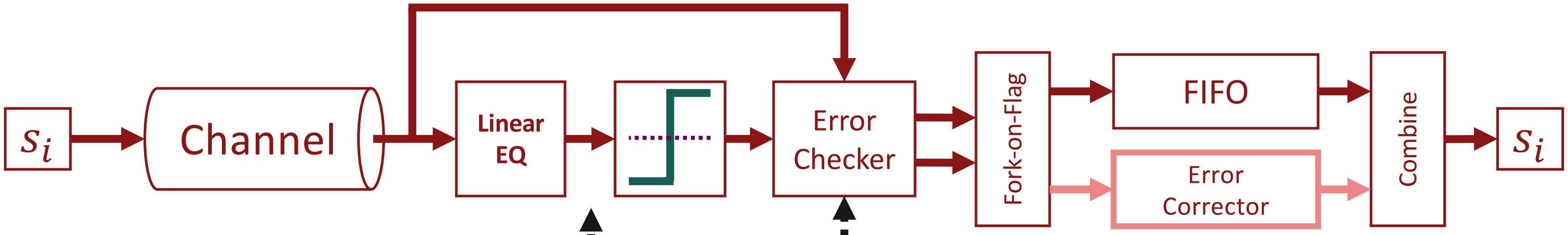
Interrupt-driven MLSD-based Links



Feedforward Error Checker

We have an internal whitepaper that Mark and I wrote on residual error-based checkers if you are interested!

Interrupt-driven MLSD-based Links

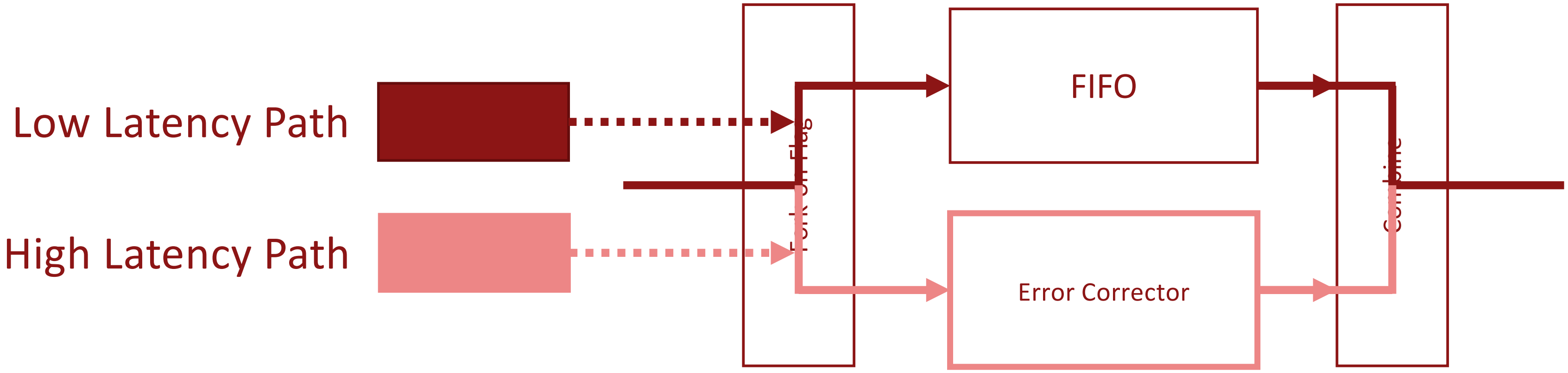


Initial s_i estimate

Feedforward Error Checker

‘Interrupt-driven’ Sequence Detection

Interrupt-driven MLSD-based Links



Thank You