Fundación FULGOR

UNIC-CASS Project

Adaptive Fractionally Spaced Equalizer (FSE)

CAS
IEEE CIRCUITS AND SYSTEMS SOCIETY

Project Design

- Design Title:

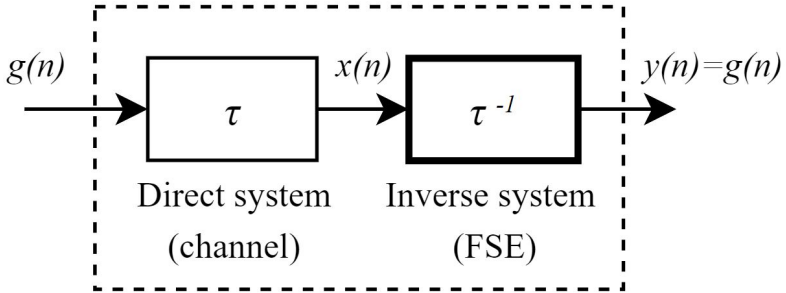
  Fractionally Spaced Equalizer (FSE) with LMS Algorithm in Time Domain

- Team Composition:

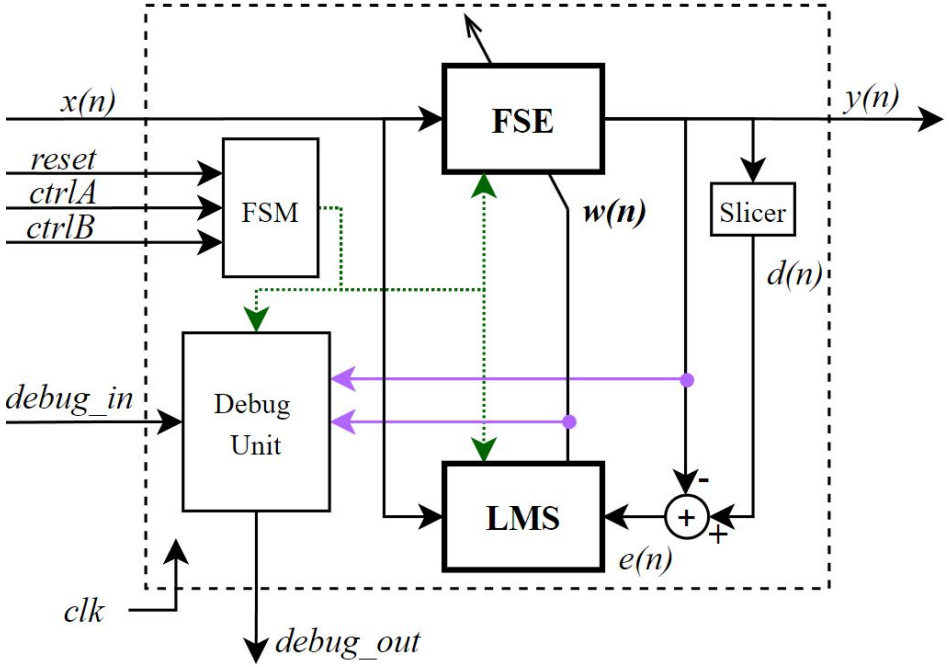
  - Ramiro Ferreyra - Master Student - Universidad Nacional del Sur
  - Baccino Octavio - Bachelor Student - Universidad Nacional de Córdoba
  - Guerra Jorge - Master Student - Universidad Nacional del Sur
  - Cavestri Ezequiel - Bachelor Student - Universidad Tecnológica Nacional
  - Santiago Garaventta - Graduate Student - Universidad Nacional de La Plata

Description of the idea

In digital communication systems, the transmission channel introduces distortions that often result in intersymbol interference (ISI). To mitigate this effect, we propose the use of a Fractionally Spaced Equalizer (FSE) combined with the Least Mean Squares (LMS) algorithm, enabling adaptive compensation. The design, described at the register-transfer level (RTL), consists of an FIR filter and an LMS block, where the coefficients are iteratively updated to minimize error, thus recovering the original pulses with reduced ISI and controlled complexity.



Block Diagram



$$w_{n+1} = w_n + \mu \cdot e_n \cdot x_n$$

LMS Algorithm

Number of pins

- VCC, GND power pins
- x(n) input 8 bits.
- Clock and reset
- FSM control pins (2 bits)
- Debug input pins (8 bits)
- y(n) output 8 bits
- Debug output pins (8 bits)

Input: 20 pins

Output: 16 pins

Description of each block

FSE: Fractionally Spaced Equalizer, is an adaptive FIR filter with two samples per symbol. It reduces ISI errors, providing the equalized output *y(n)*.

Slicer: It is a decision block, It is responsible for making a decision *d(n)* on the output of the equalizer, that is, it gives a final output depending on whether the signal is above or below the decision threshold in the corresponding modulation. The decision is used to calculate the error *e(n)=d(n)-y(n)*.

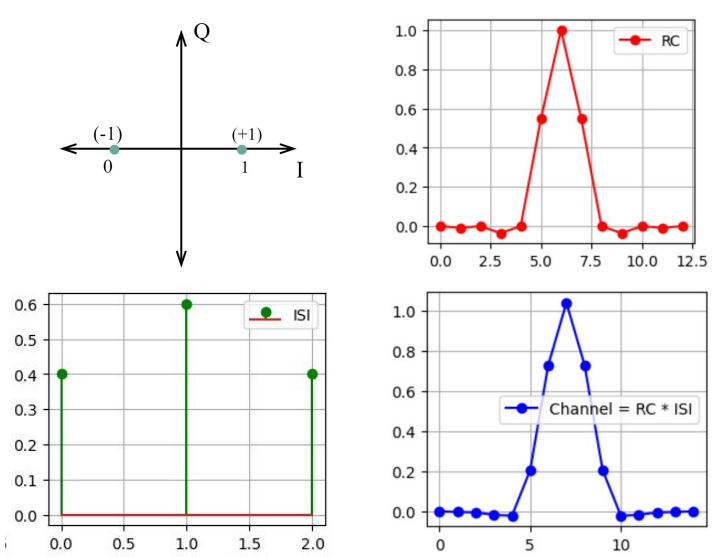
LMS: Least Means Square, is an adaptive algorithm that updates the equalizer coefficients *w(n)* using the error *e(n)*, minimizing the mean squared error.

FSM: Finite State Machine for general control of system. This FSM sends enable/valid/reset signals to each block to achieve a global control and synchronization, ensuring the correct execution of the algorithm phases.

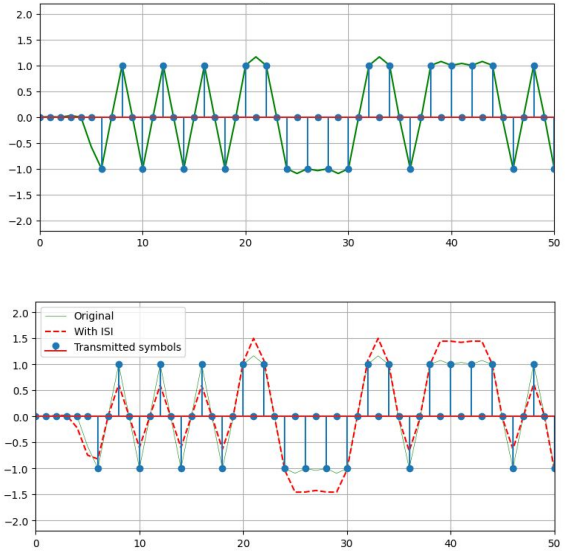
Debug Unit: It provides a hardware interface for real-time system verification and analysis. It allows observation of key internal algorithm signals, such as the filter coefficients w(n) and the error signal e(n).

Simulation - Input Signal

We take BPSK pulses and do pulse shaping with a Raised Cosine (RC) filter. Then we add ISI to simulate the distortion, and that makes our channel model. In the implementation, this would be the input signal coming from the ADC.

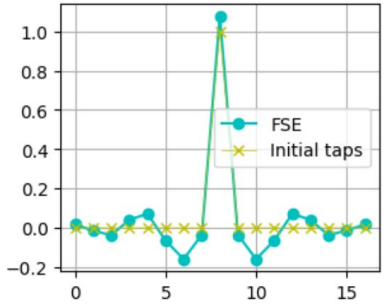


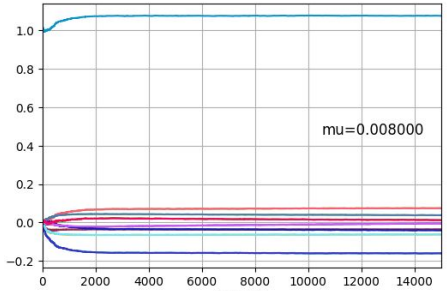
*Impulse responses of RC, ISI and total channel (RC * ISI)*



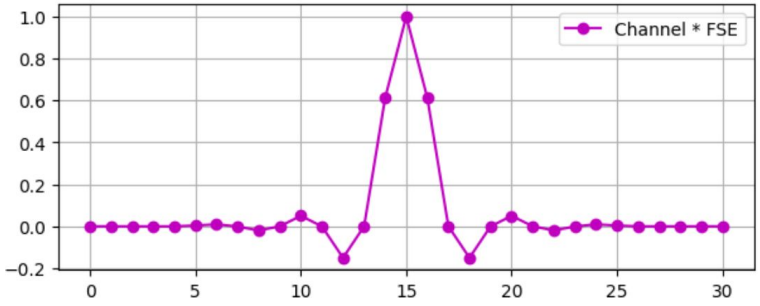
Input signal without ISI and with ISI

Simulation - Equalization

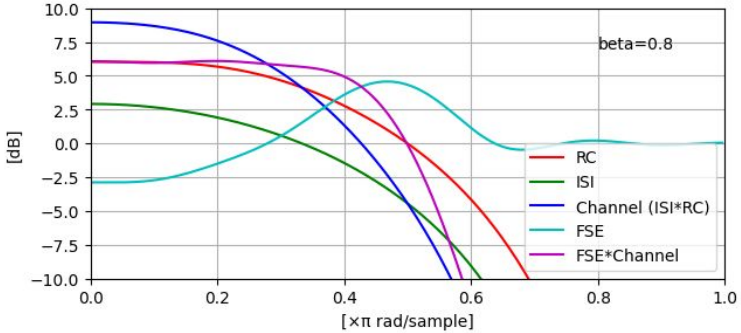




Evolution of the coefficients in time

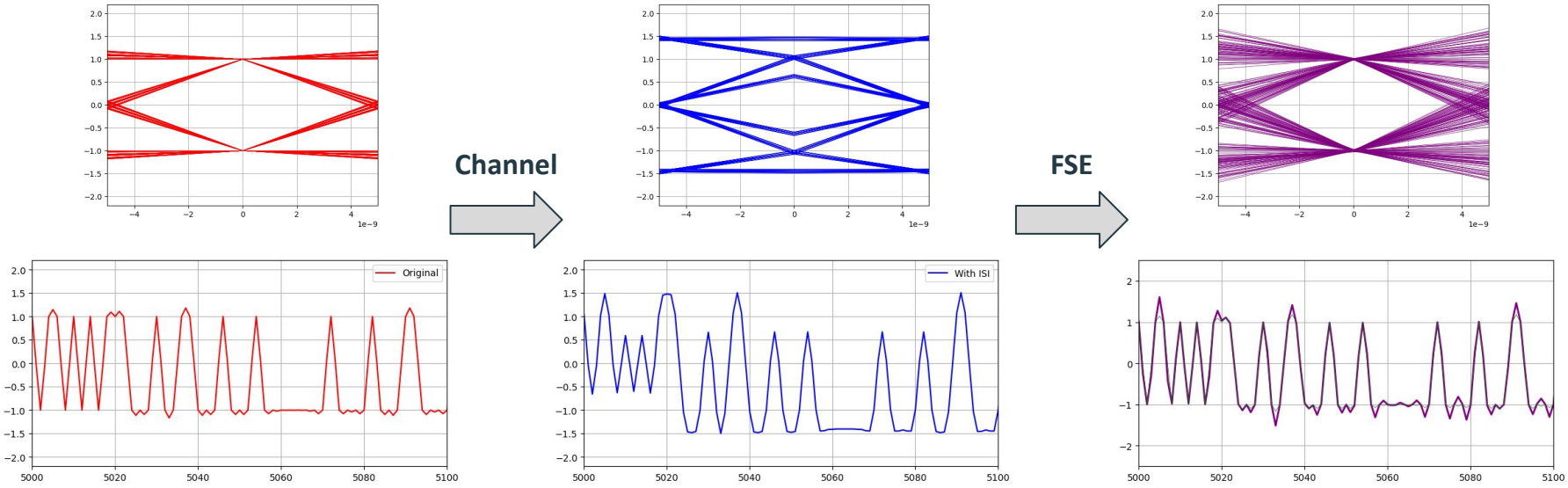


*Impulse responses of FSE and the total system (channel * FSE)*



Frequency Response in different stages

Simulation - Equalization

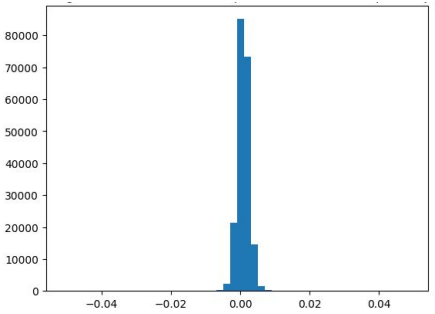


Signal before the channel, after the channel and equalized.

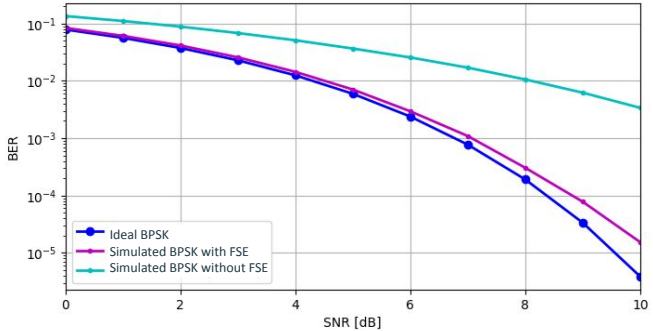
We recover the original sampling points

Simulation - Equalization

Up to this point we have been working with **floating point**, now we pass that simulation to **fixed point** to use it as a reference. We choose the representation *s(8;5)* for input and output, with 8 total bits and 5 fractional bits.

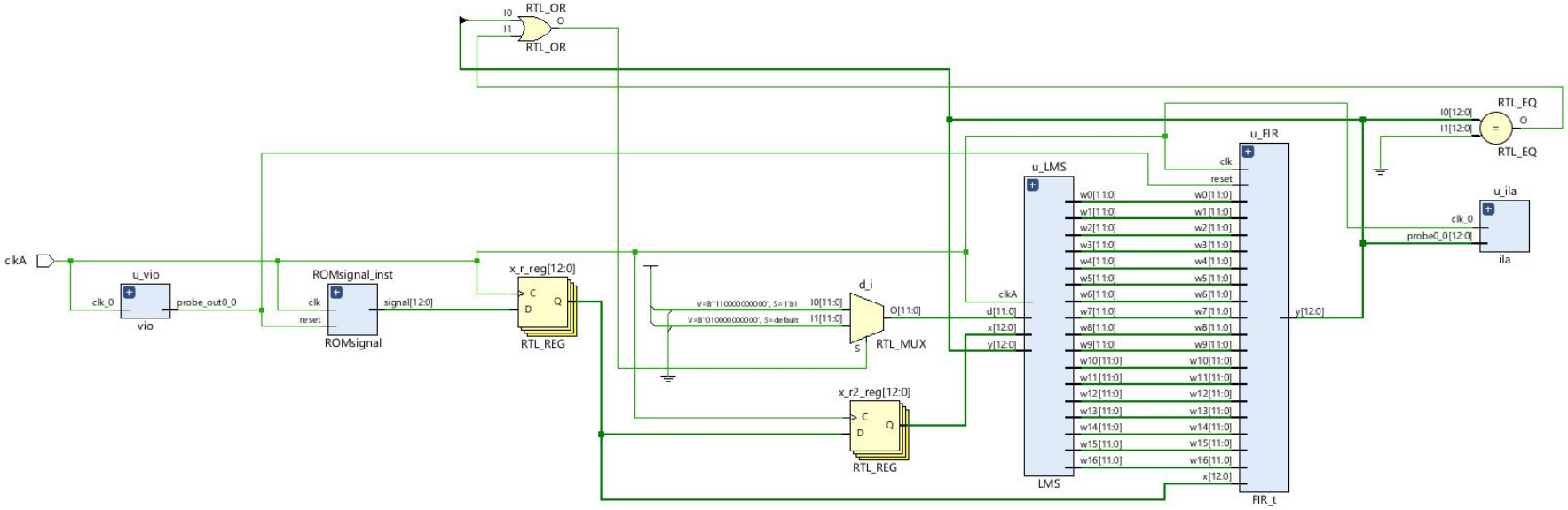


Error histogram between floating-point and fixed-point signals



BER vs. SNR curves

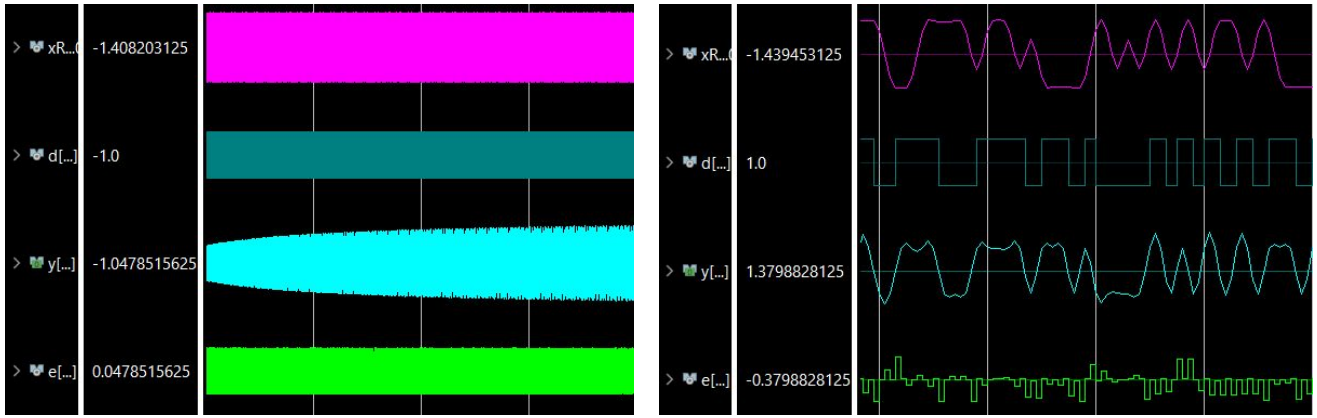
RTL Design



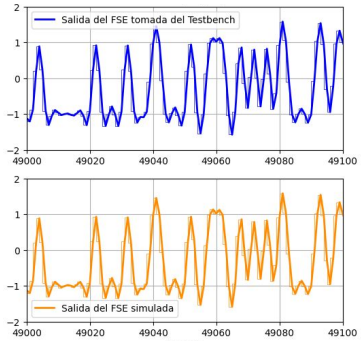
Pre-synthesis Schematic

RTL Design - Testbench

To have a point of reference we use the same input in the Testbench as in the simulation. The output is saved to a file so we can do a Vector Matching (comparison) between the high-level simulation and the RTL description.



Testbench waves



Comparison between Testbench and simulation

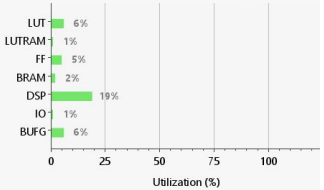
RTL Design - Synthesis and Implementation

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	LUT	FF	DSP	BRAM	URAM
synth_4 (active)	constrs_1	Synthesis Out-of-date									1764	1003	52	0.5	0
impl_4 (active)	constrs_1	Implementation Out-of-date	2.944	0.000	0.038	0.000	9.140	0.000	0.080	0	1227	2108	17	1	0

Worst Negative Slack (WNS):	2.944 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	5268

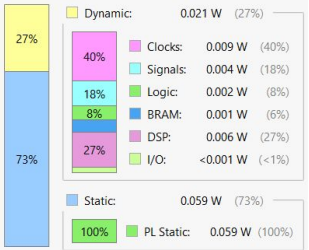
Timing report

Resource	Utilization	Available	Utilization %
LUT	1227	20800	5.90
LUTRAM	96	9600	1.00
FF	2108	41600	5.07
BRAM	1	50	2.00
DSP	17	90	18.89
IO	1	210	0.48
BUFG	2	32	6.25

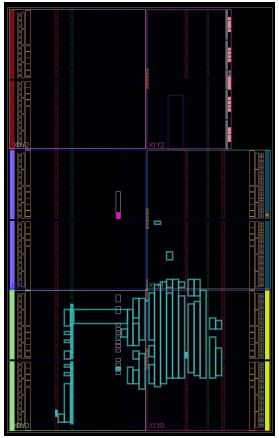


Area report

Total On-Chip Power:	**0.08 W**
Junction Temperature:	**25.4 °C**
Thermal Margin:	74.6 °C (15.5 W)
Effective θJA:	4.8 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium



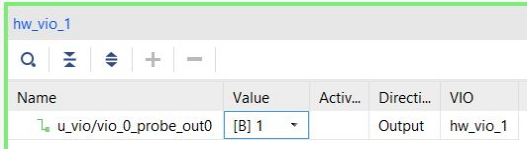
Power Dissipation report



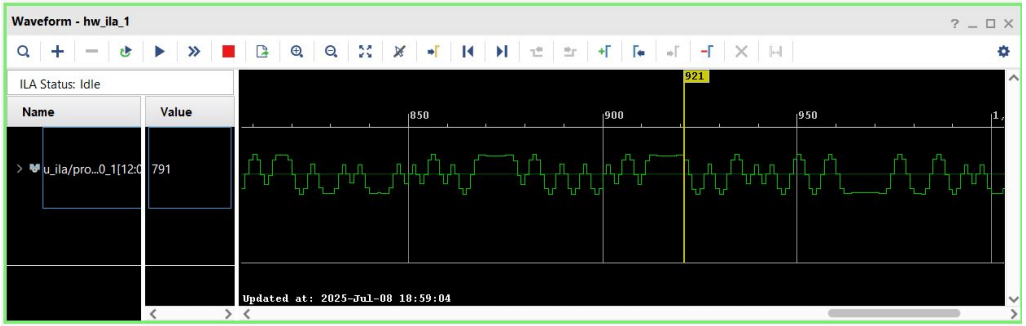
Post-synthesis device

RTL Design - Verification in FPGA

We connected remotely to an "*Arty A7-100T*" FPGA using **VIO** (Virtual Input/Output) and **ILA** (Integrated Logic Analyzer) blocks to validate the design on the board.



VIO to control the reset



ILA to monitor the output