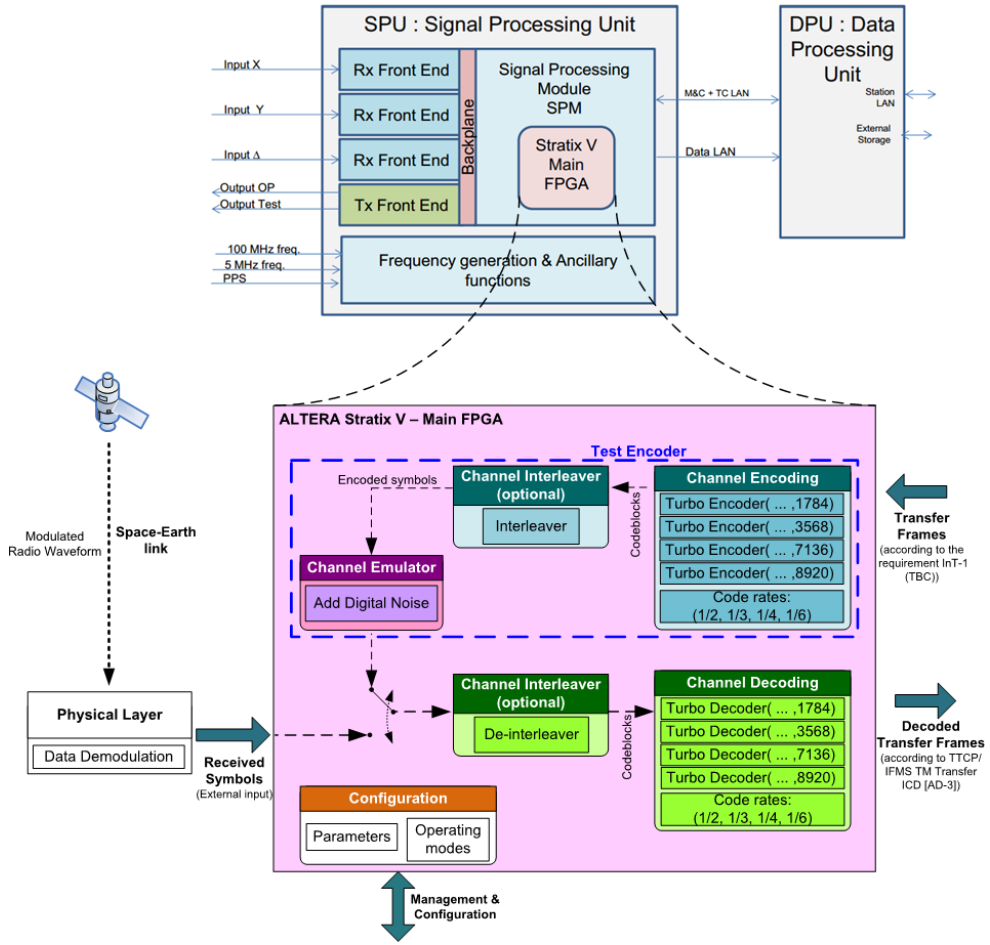
VIRTUDE

Very High Rate Turbo Decoder with Interleaver in the TTCP



Final Presentation

25th November 2022 – ESOC

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

15:45 – Q&A

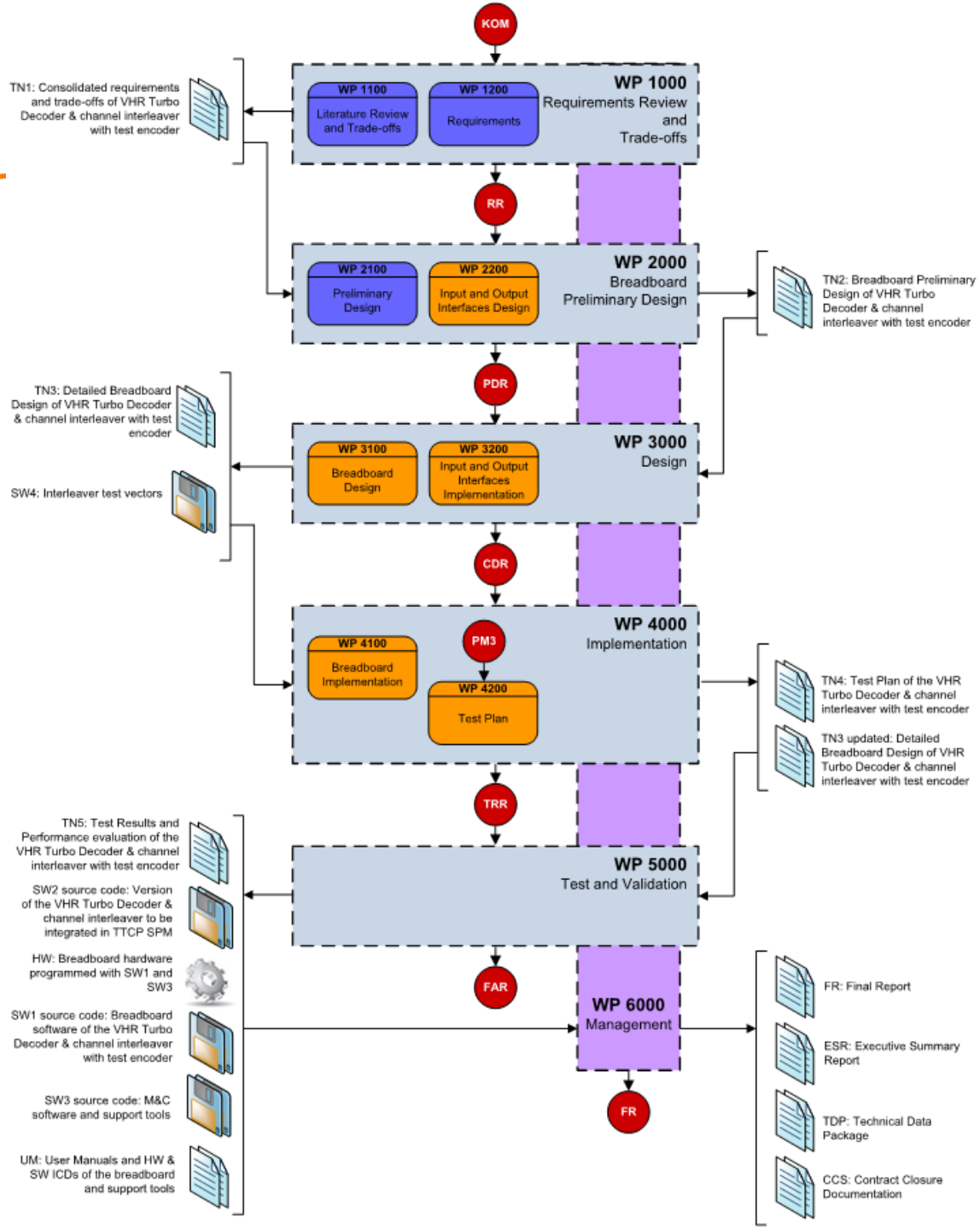
16:00 – Meeting closure

Introduction

- **Project overall objectives**
  - Aiming **Envision mission** to transmit data from a **SAR (Synthetic Aperture Radar)**
  - Analyse the possible architectures to implement a **Very High Rate (VHR) Turbo Decoder** and respective **test Turbo Encoder** for Turbo rates 1/2, 1/3, 1/4 and 1/6 (information block lengths of 1784, 3568, 7136 and 8920 bits) in order to work in **real-time up to 80 Mbps** (75 Mbps + 5 Mbps of margin).
  - Design and Implementation of the **Channel Interleaver**
  - **Implementation** the transmitter and receiver chains in the TTCP platform fulfilling the required performance of **80 Mbps** and **using only the current resources available at TTCP**
  - **Validation of the transmitter and receiver chains in the TTCP platform,** all modules running in the single master processing FPGA (**ALTERA Stratix V**) of TTCP.

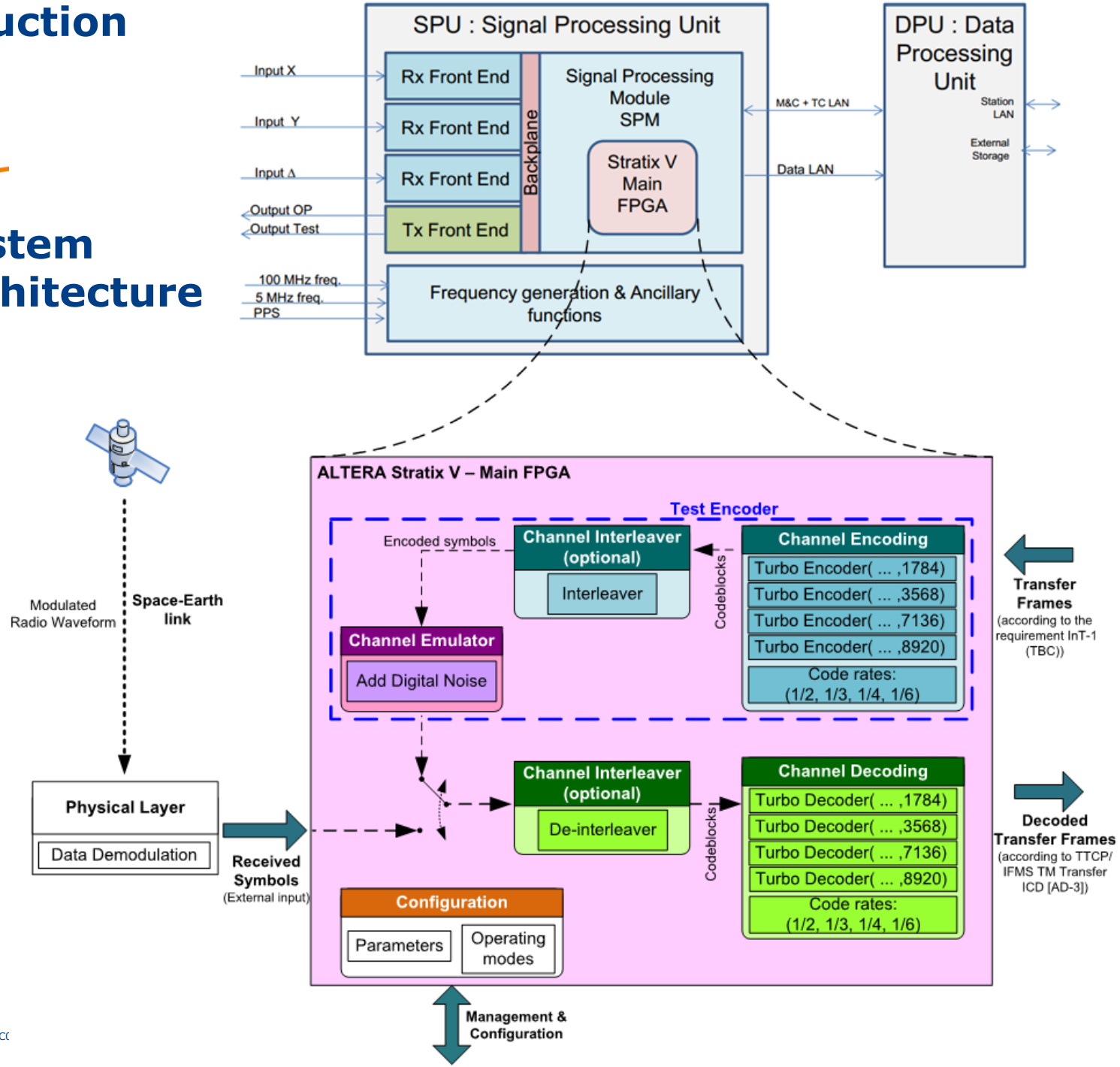
Introduction

- **Study logic**



Introduction

- **System architecture**

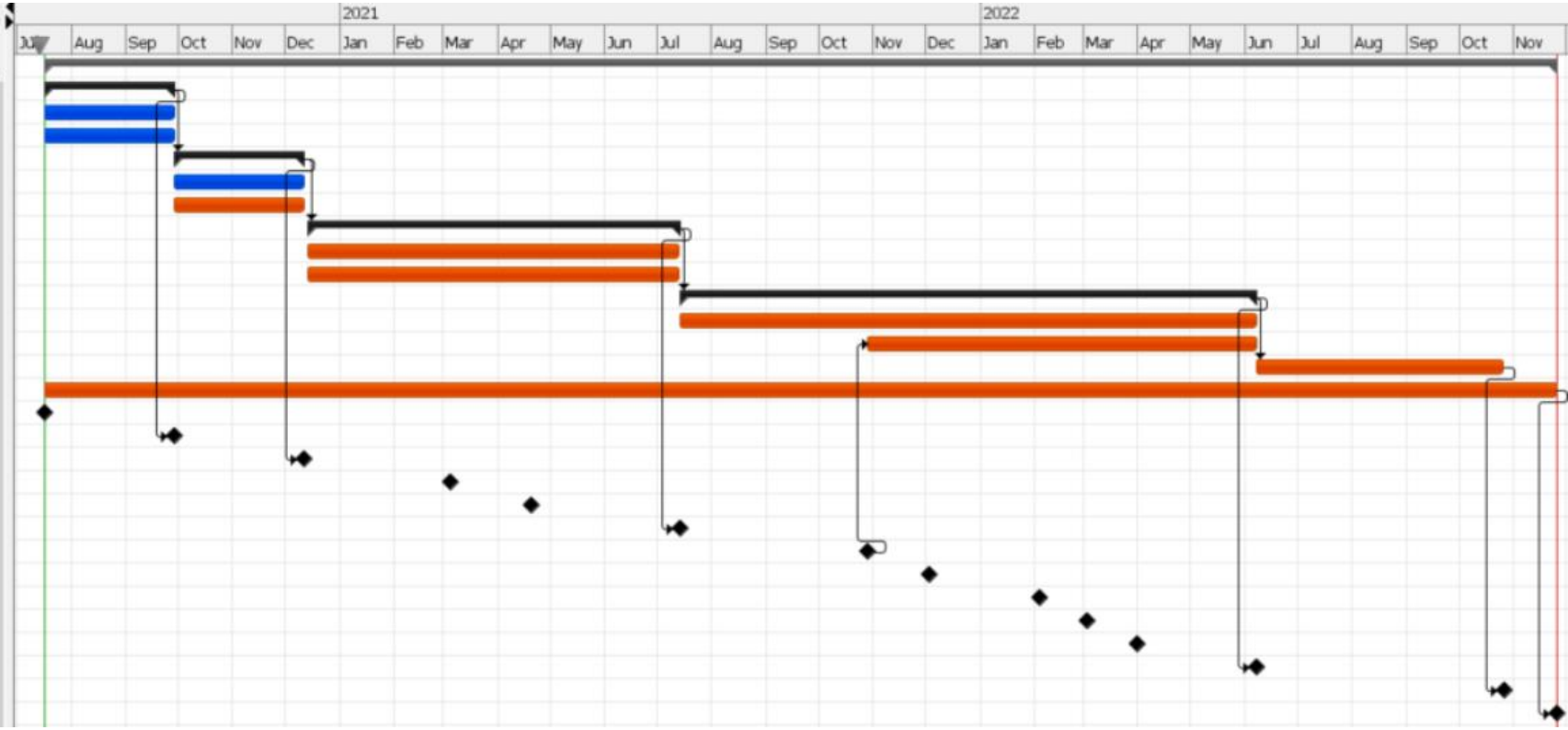


- **FPGA** ALTERA Stratix V
- Useful data rate: **80 Mbps**
- Symbol rate: **320 MSps**
- Interfaces:
  - Transfer Frames
  - Received Symbols
  - Decoded Transfer Frames according to TTCP-ICD-TM
  - Monitoring and configuration
- **Only with the current resources** available at TTCP

Introduction

- **Actual schedule**

Task Name	Start	End
Very High Rate Turbo Decoder with Interleaver in the TTCP	Jul 17, 2020	Nov 25, 2022
WP 1000: Requirements Review and Trade-offs	**Jul 17, 2020**	**Sep 28, 2020**
WP 1100: Literature Review and Trade-offs	Jul 17, 2020	Sep 28, 2020
WP 1200: Requirements	Jul 17, 2020	Sep 28, 2020
WP 2000: Breadboard Preliminary Design	**Sep 29, 2020**	**Dec 11, 2020**
WP 2100: Preliminary Design	Sep 29, 2020	Dec 11, 2020
WP 2200: Input and Output interfaces Design	Sep 29, 2020	Dec 11, 2020
WP 3000: Design	**Dec 14, 2020**	**Jul 13, 2021**
WP 3100: Breadboard Design	Dec 14, 2020	Jul 13, 2021
WP 3200: Input and Output interfaces Implementation	Dec 14, 2020	Jul 13, 2021
WP 4000: Implementation	**Jul 14, 2021**	**Jun 7, 2022**
WP 4100: Breadboard Implementation	Jul 14, 2021	Jun 7, 2022
WP 4200: Test Plan	Oct 29, 2021	Jun 7, 2022
WP 5000: Test and Validation	Jun 8, 2022	Oct 26, 2022
WP 6000: Management	Jul 17, 2020	Nov 25, 2022
KOM	Jul 17, 2020	Jul 17, 2020
Requirements Review (RR)	Sep 28, 2020	Sep 28, 2020
M1 - Preliminary Design Review (PDR)	Dec 11, 2020	Dec 11, 2020
PM1 - Progress Meeting 1	Mar 5, 2021	Mar 5, 2021
PM2 - Progress Meeting 2	Apr 20, 2021	Apr 20, 2021
M2 – Critical Design Review (CDR)	Jul 13, 2021	Jul 13, 2021
PM3 - Progress Meeting 3	Oct 29, 2021	Oct 29, 2021
PM4 - Progress Meeting 4	Dec 3, 2021	Dec 3, 2021
PM5 - Progress Meeting 5	Feb 4, 2022	Feb 4, 2022
PM6 - Progress Meeting 6	Mar 3, 2022	Mar 3, 2022
PM7 - Progress Meeting 7	Apr 1, 2022	Apr 1, 2022
M3 – Test Readiness Review (TRR)	Jun 7, 2022	Jun 7, 2022
M4 – Factory Acceptance Review (FAR)	Oct 26, 2022	Oct 26, 2022
M5 - Final Presentation and final delivery	Nov 25, 2022	Nov 25, 2022



- KO date: 17th July 2020

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

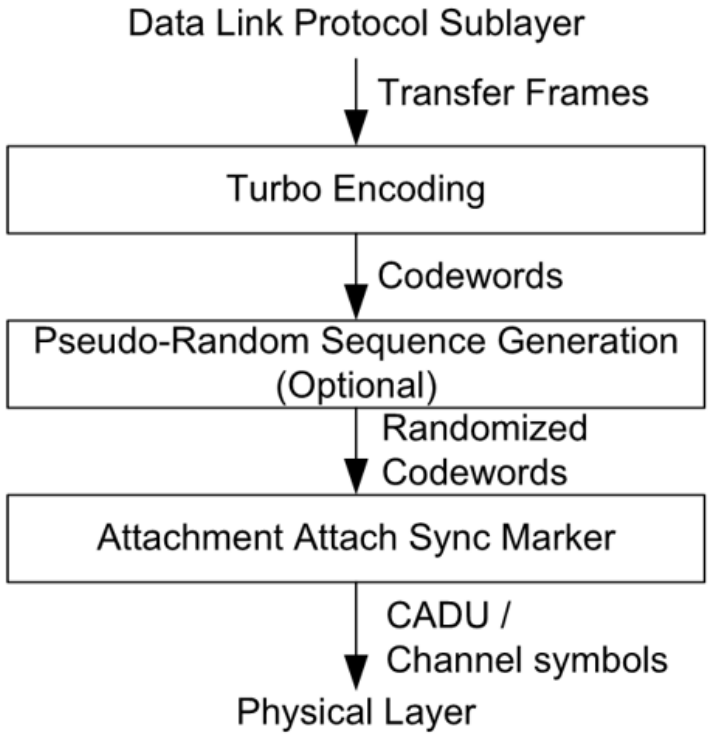
15:25 – Results Overview

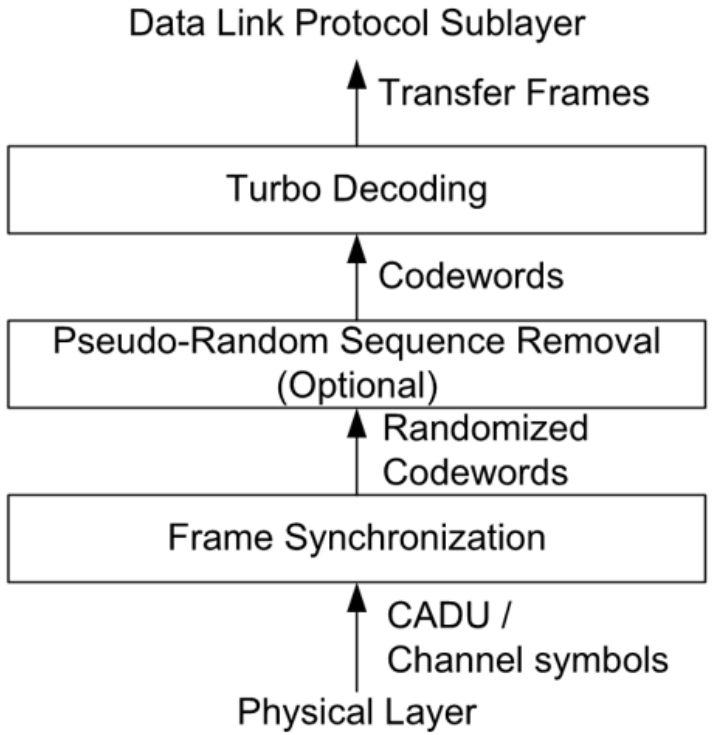
15:45 – Q&A

16:00 – Meeting closure

Synchronization and Channel Sublayer (1/2)

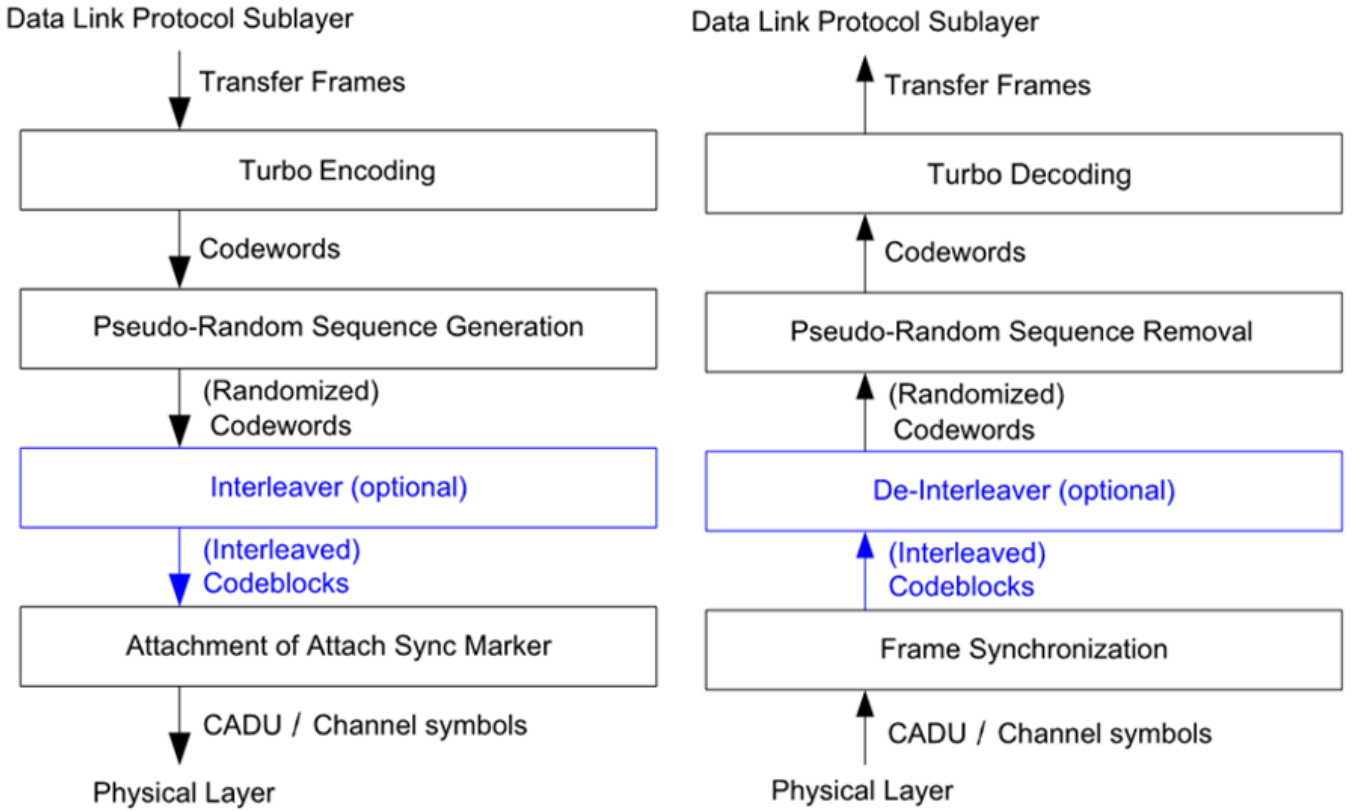
- VIRTUDE has implemented the TM Synchronization and Channel Coding (S&CC) sub-layer for the Turbo code case
  - Transmitter Chain on left and Receiver Chain on right





Synchronization and Channel Sublayer (2/2)

- VIRTUDE includes the Channel Interleaver module (at blue color) which is placed after the Randomizer in order to overcome the burst vulnerability of the Turbo codes
  - The CCSDS Blue Book may be updated soon with this updated organization

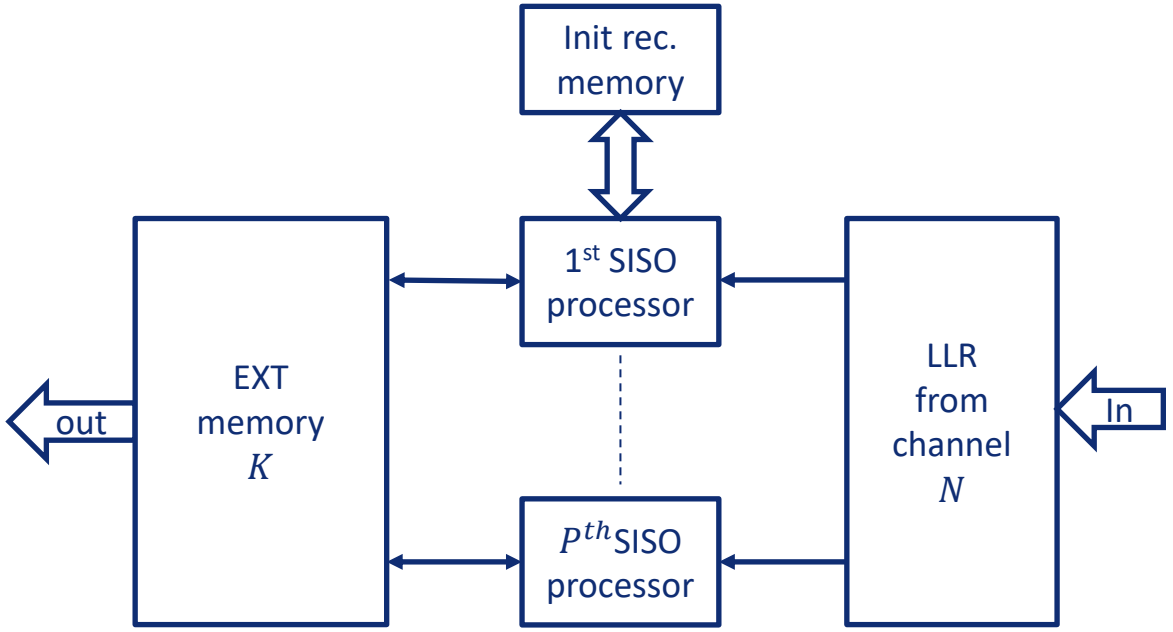


Encoder and decoder design methodology

- Hierarchical approach:
  - the overall VHDL models are structured as multiple levels of description
  - the highest level reflects the component architecture presented in TN1,
  - the lower levels provide a detailed description of the internal units
- Each single unit has been individually verified, by means of functional simulations and comparisons of input and output patterns against C or Matlab behavioural models
- Integration to verify the correct description of all internal interfaces
- Synthesis:
  - Individual synthesis of the key internal units
  - Overall synthesis to estimate clock period and occupied resources

Reference turbo decoder architecture

- The architecture is built around *P* parallel SISO decoders that process separate trellis windows of the constituent convolutional encoders of size W.
- The processors read the LLR values from the memory storing the channel LLR and update (read/write) the extrinsic information stored in memory EXT.
- Throughput (approximation)
  - N_D is the number of full turbo decoders working on separate frames,
  - f_c is the clock speed of each SISO processor, (trellis steps per second),
  - I is the required number of iterations per received packet.



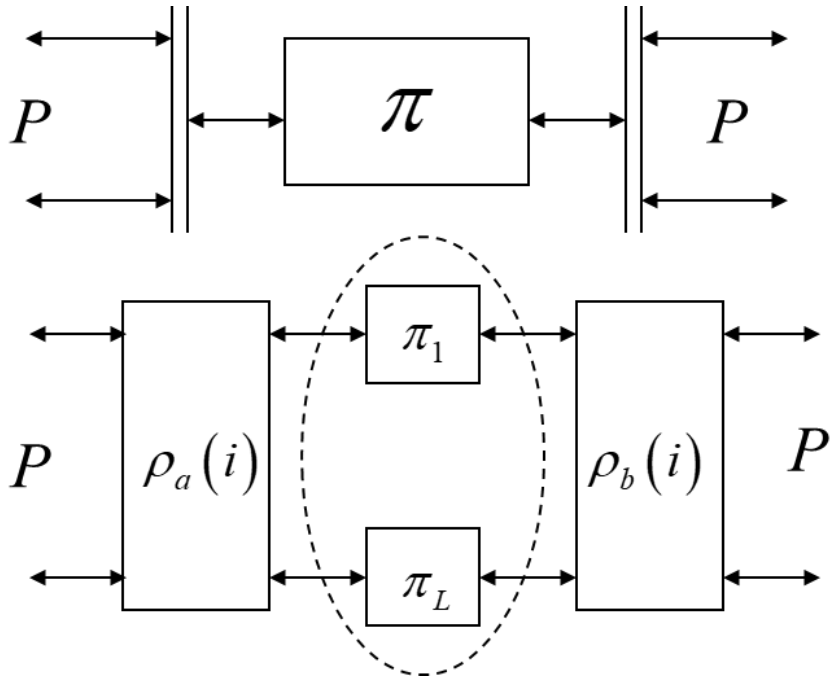
$$T = N_D P \frac{f_c}{2I}$$

Increase the parallelism (*P*)

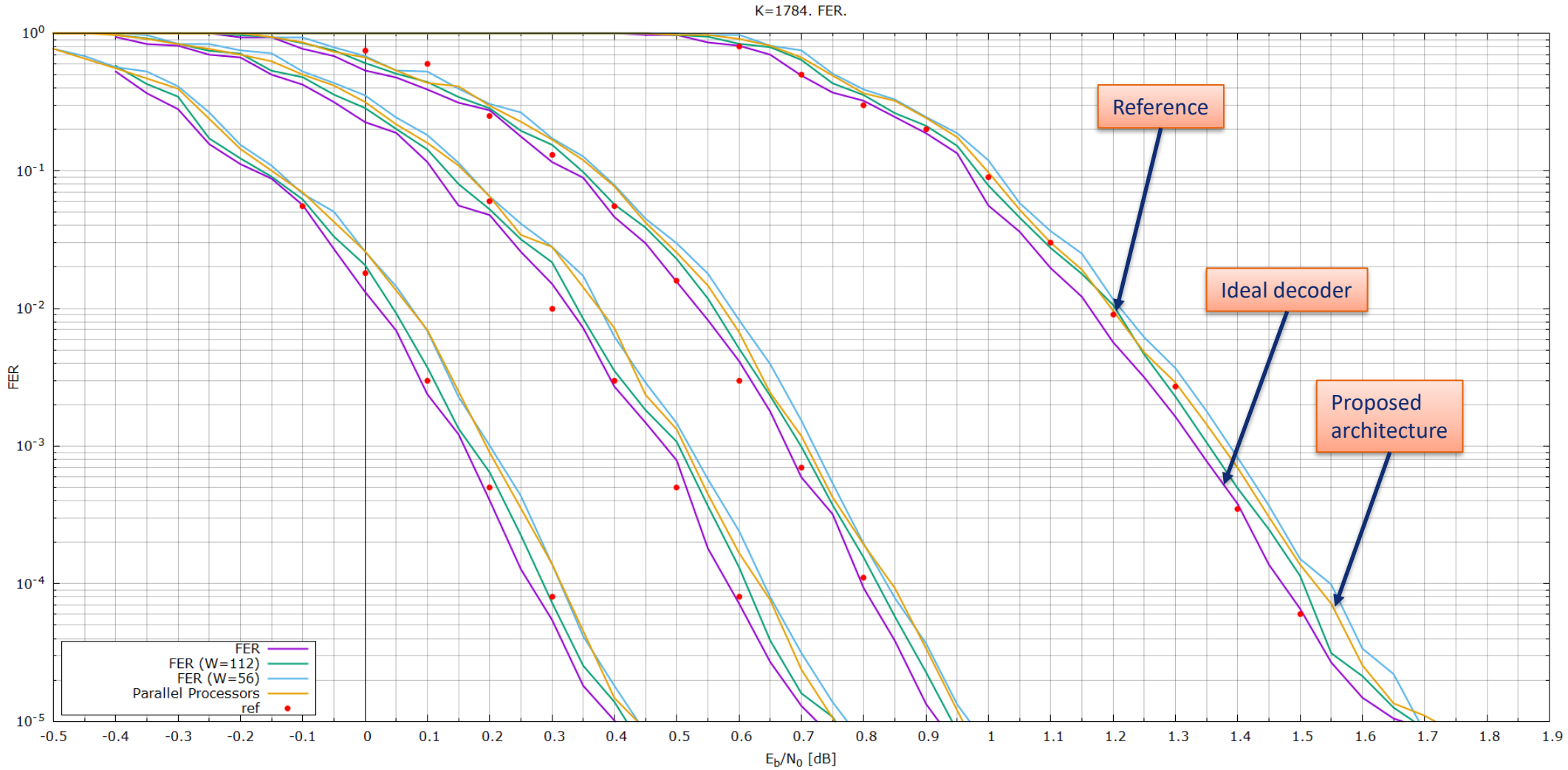
- Increase the number of SISO processors *(P)*
- Advantages
  - Memory requirements for EXT and LLR do not increase with P
  - Decoding latency does not increase with P
- Disadvantages
  - memory collision and routing problem.
    - The P SISO processors must access a single memory in two different ways (natural and permuted order) with P parallel data at frequency f_C

Permutation decomposition

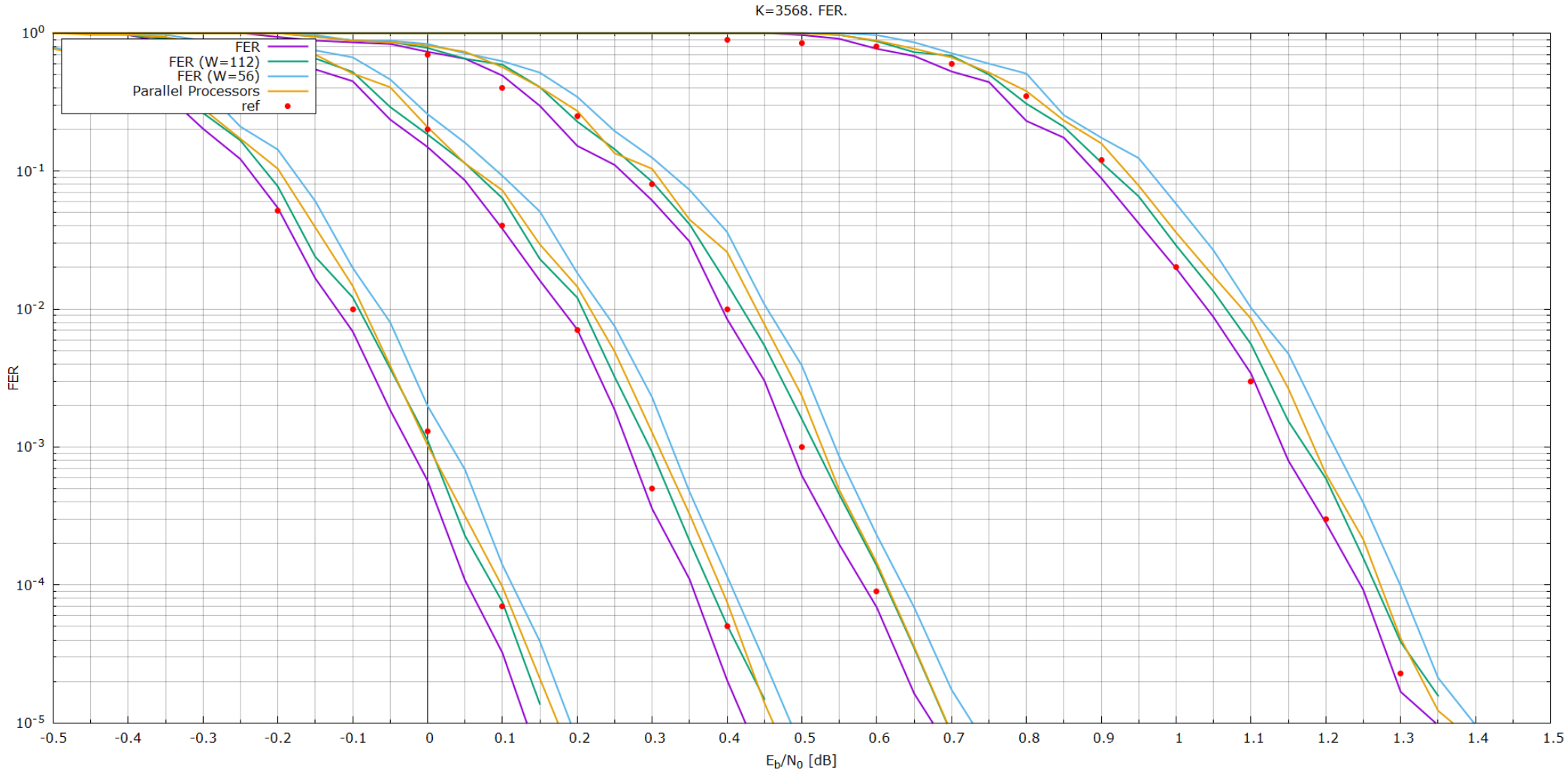
- In [Tarable] it has been shown that, for any desired parallelism P, and any desired interleaving law $\boldsymbol{\pi}$ it is possible to find a collision-free mapping
- In the decomposed architecture a set of P memory banks is accessed in parallel using a set of smaller permutations of size $\frac{K}{P}$ through the time varying permutations $\rho(i)$ of size P on the input parallel data
- The block denoted by $\rho_a(i)$ are then programmable crossbars with size P routing the data from the SISO processors to the P banks of memory and viceversa



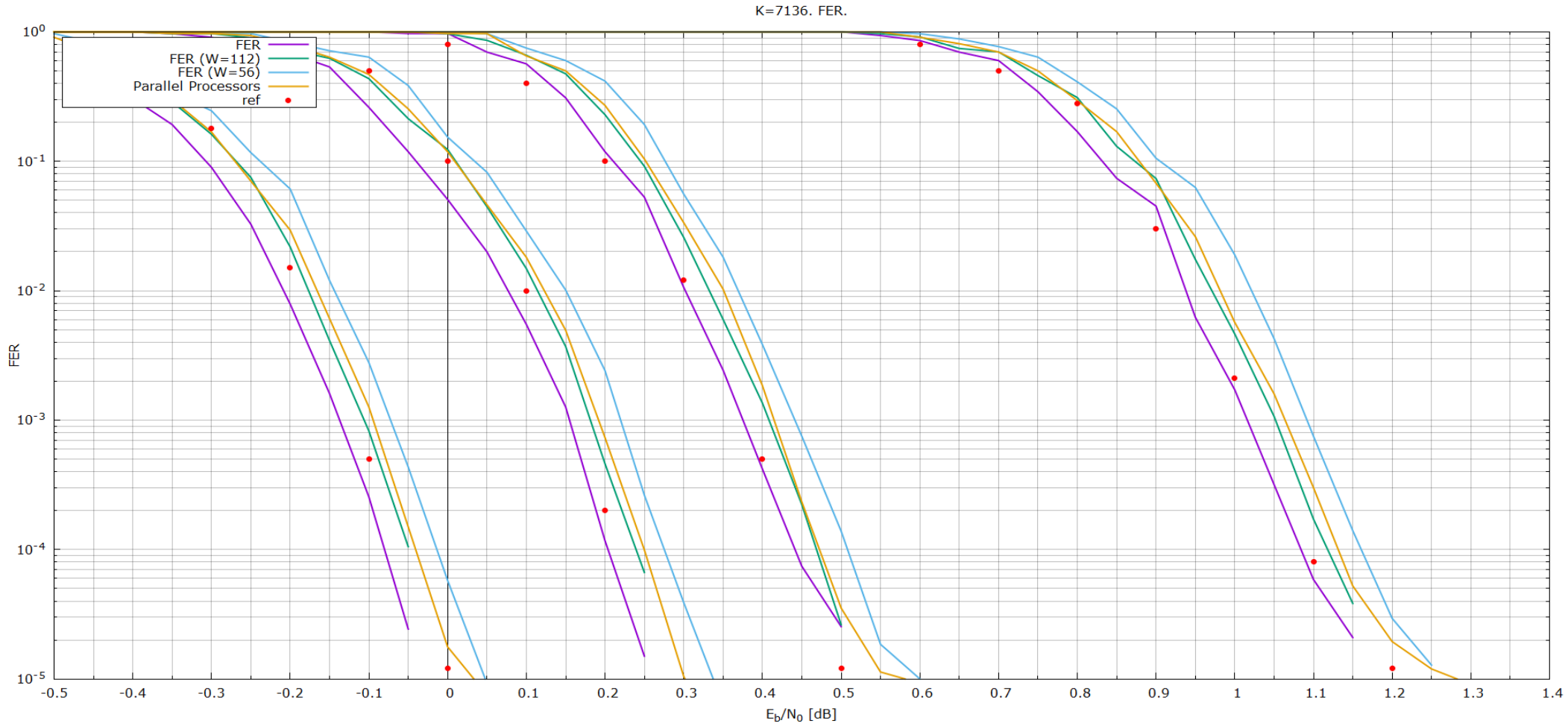
FER performances of proposed architecture



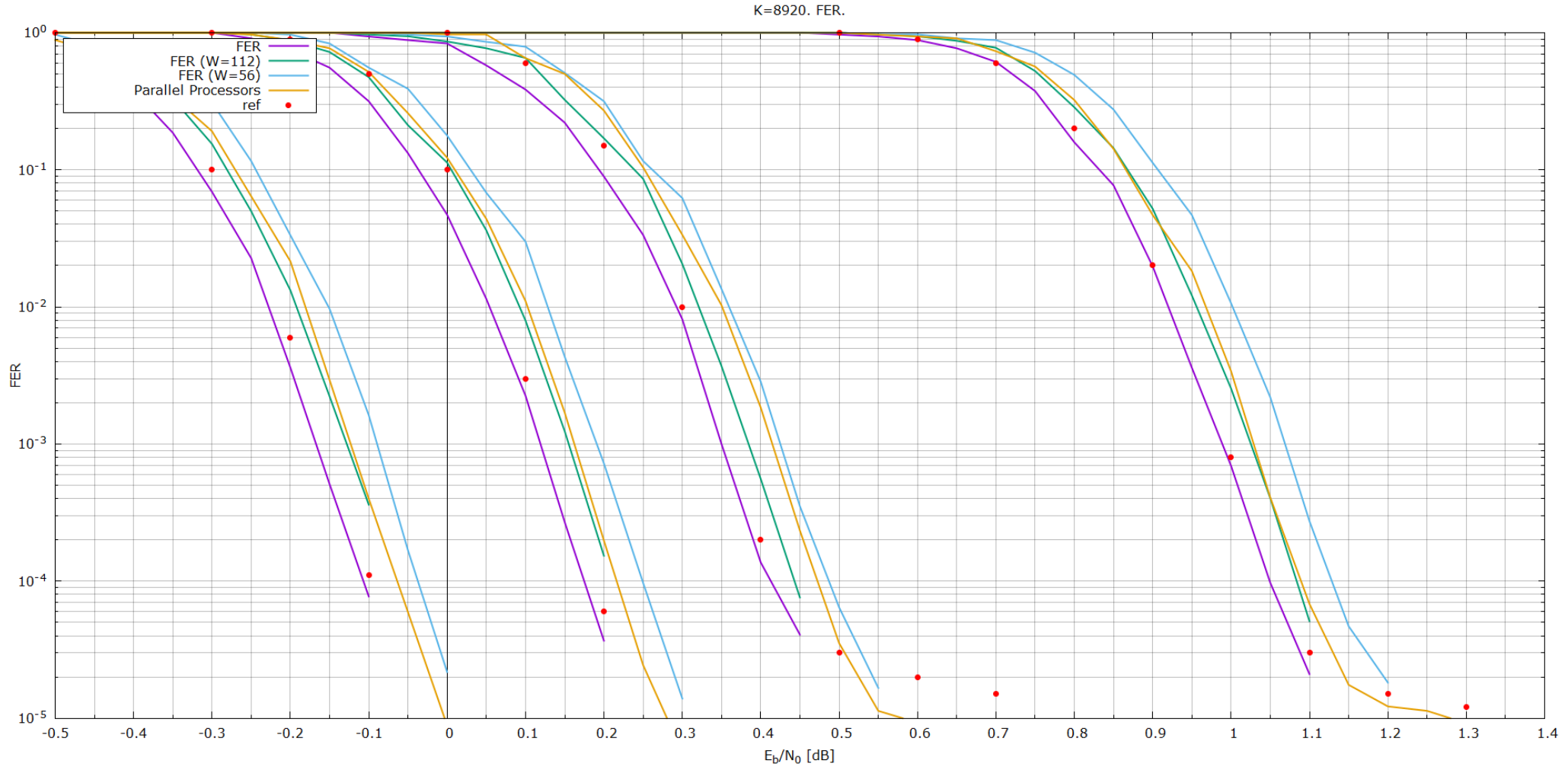
FER performances K=3568



FER performances K=7136



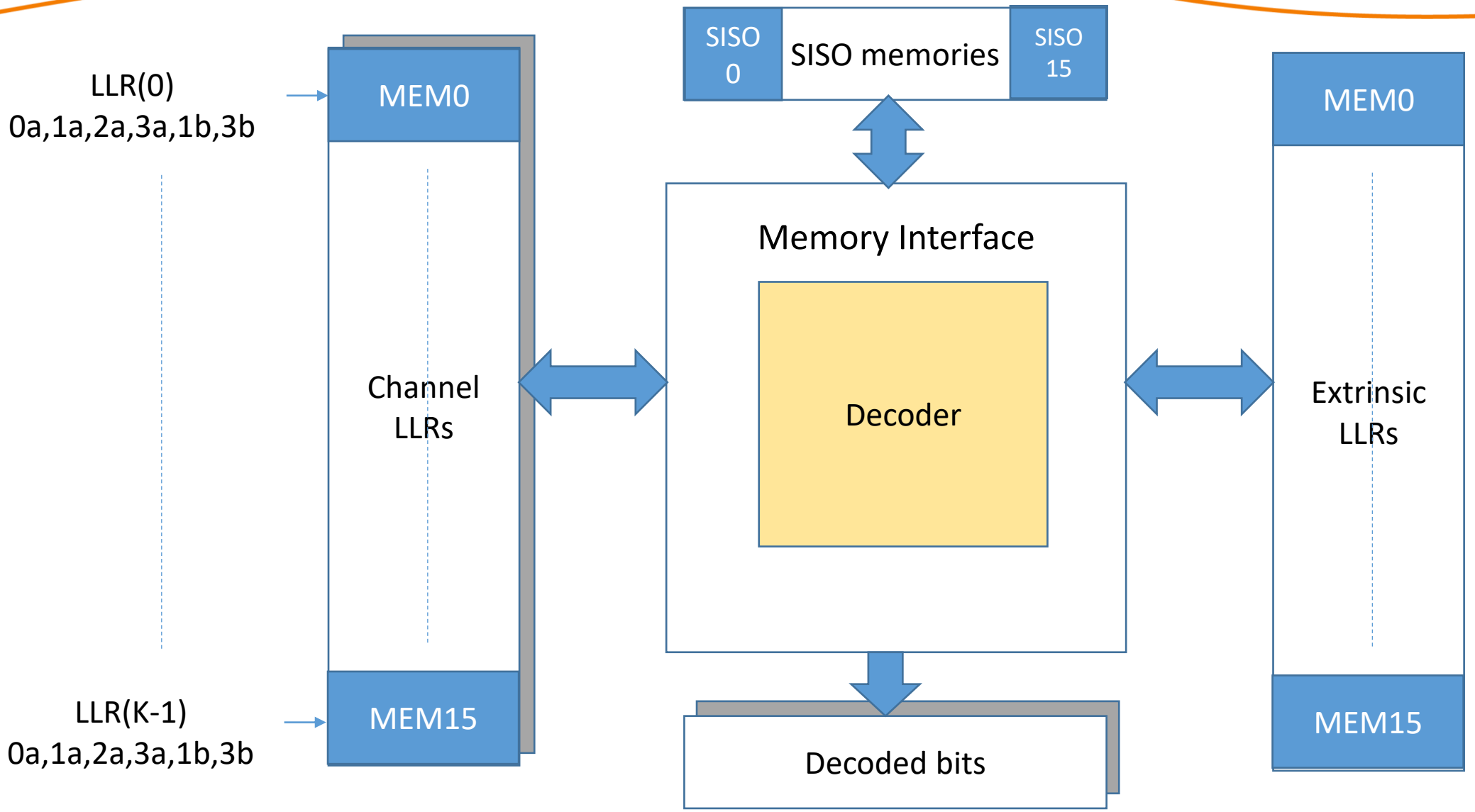
FER performances K=8920



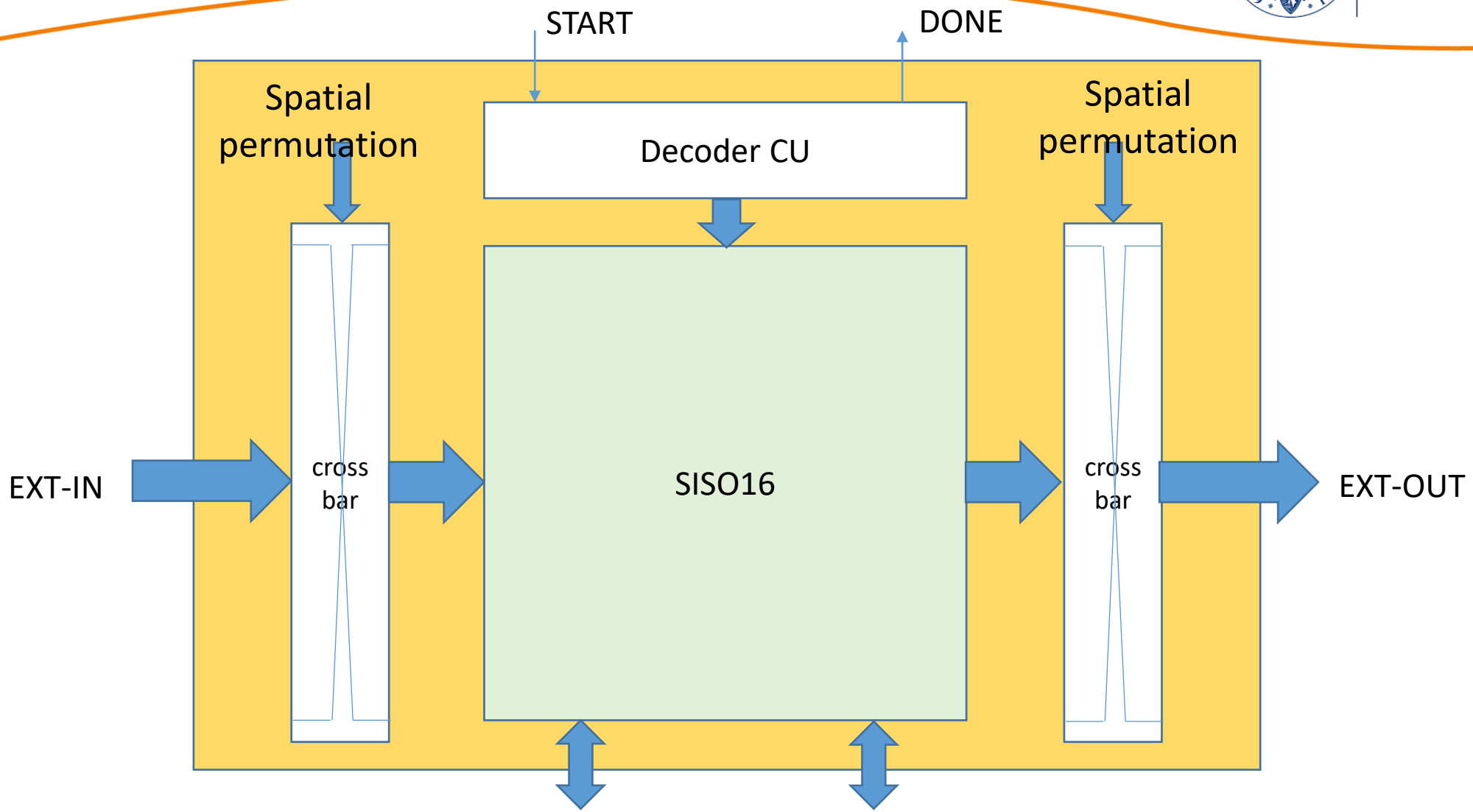
Proposed architectural solution:

- 16 parallel SISO processors, Window size 56 trellis steps
- Delayed initialization of forward and backward recursion
- 2 fully programmable 16 crossbars
- Per block parallelism:
  - N_d=4 and $P = 4$ for $K = 1784$,
  - N_d=2 and $P = 8$ for $K = 3568$,
  - N_d=1 and $P = 16$ for $K = 7136, 8920$.
- Fixed point representation of quantization bits for LLR: 5 bits (3.2)
- Fixed point representation of quantization bits for EXT: 7 bits (5.2)

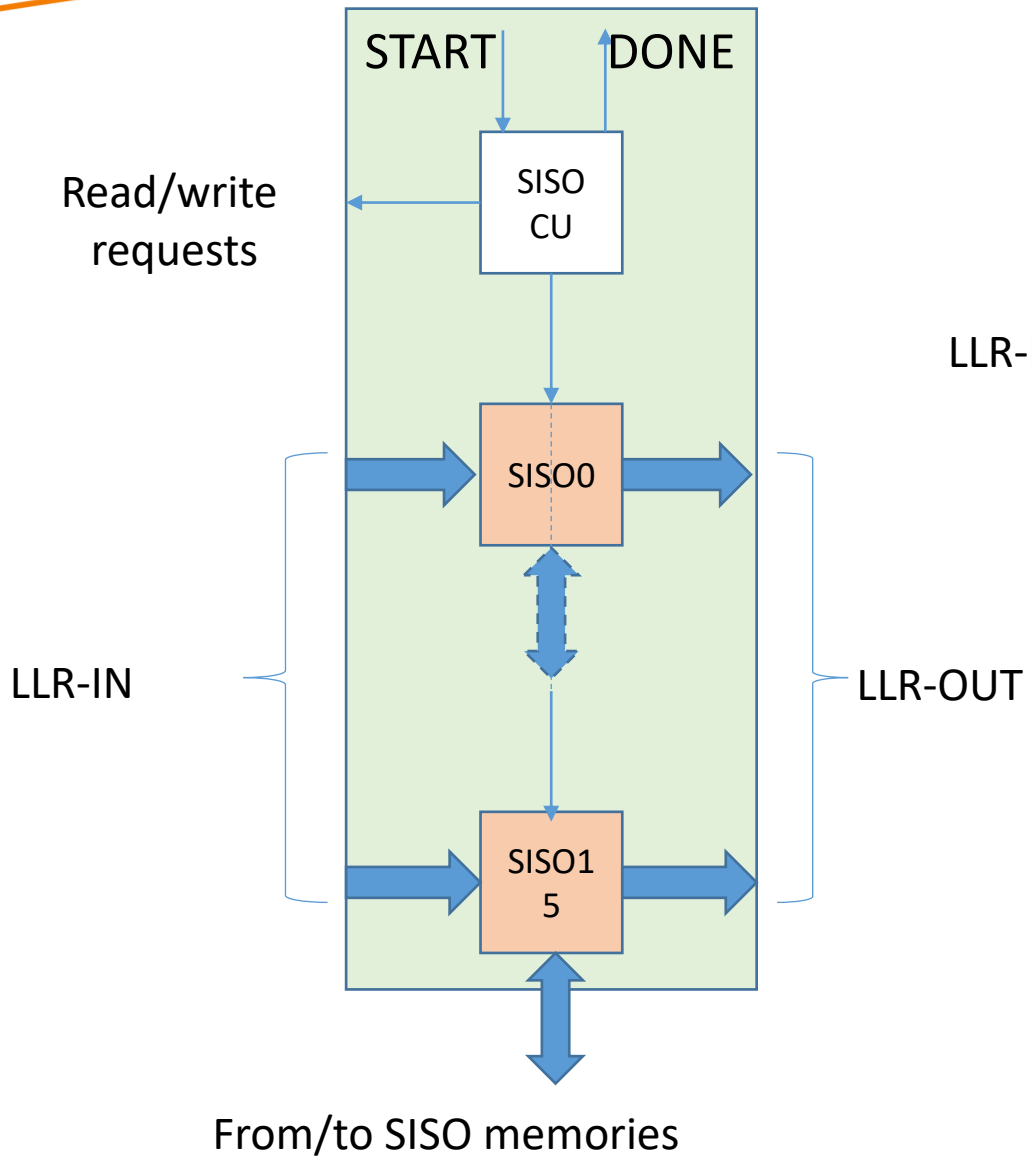
High-level view of the decoder

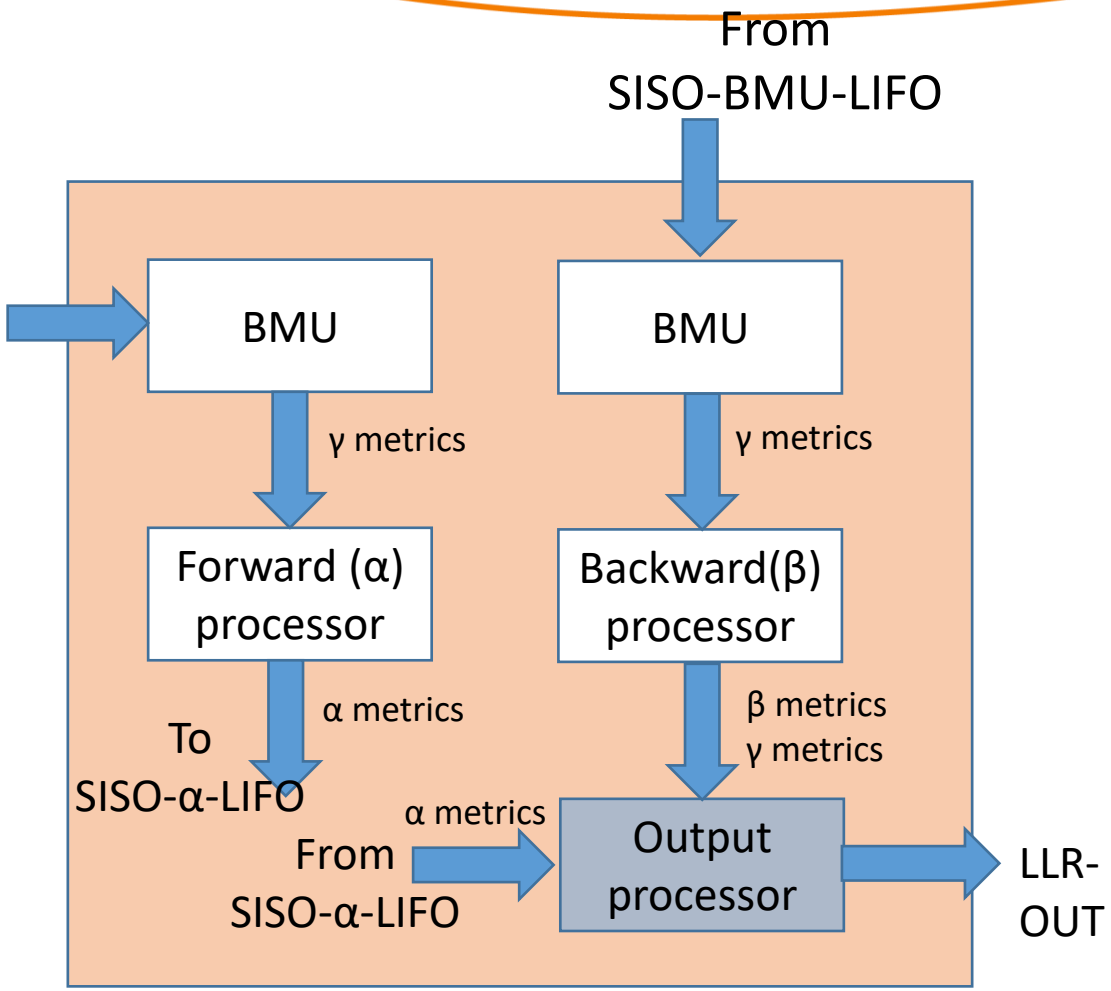


Decoder architecture

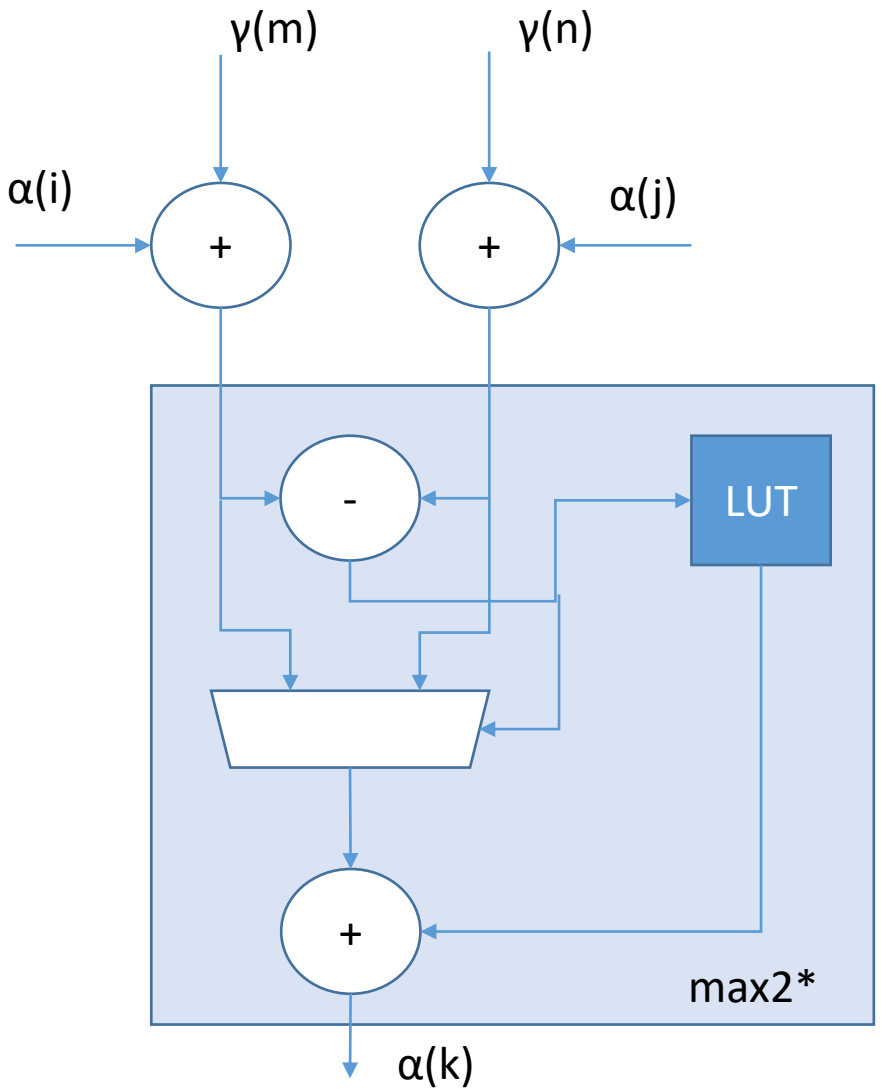


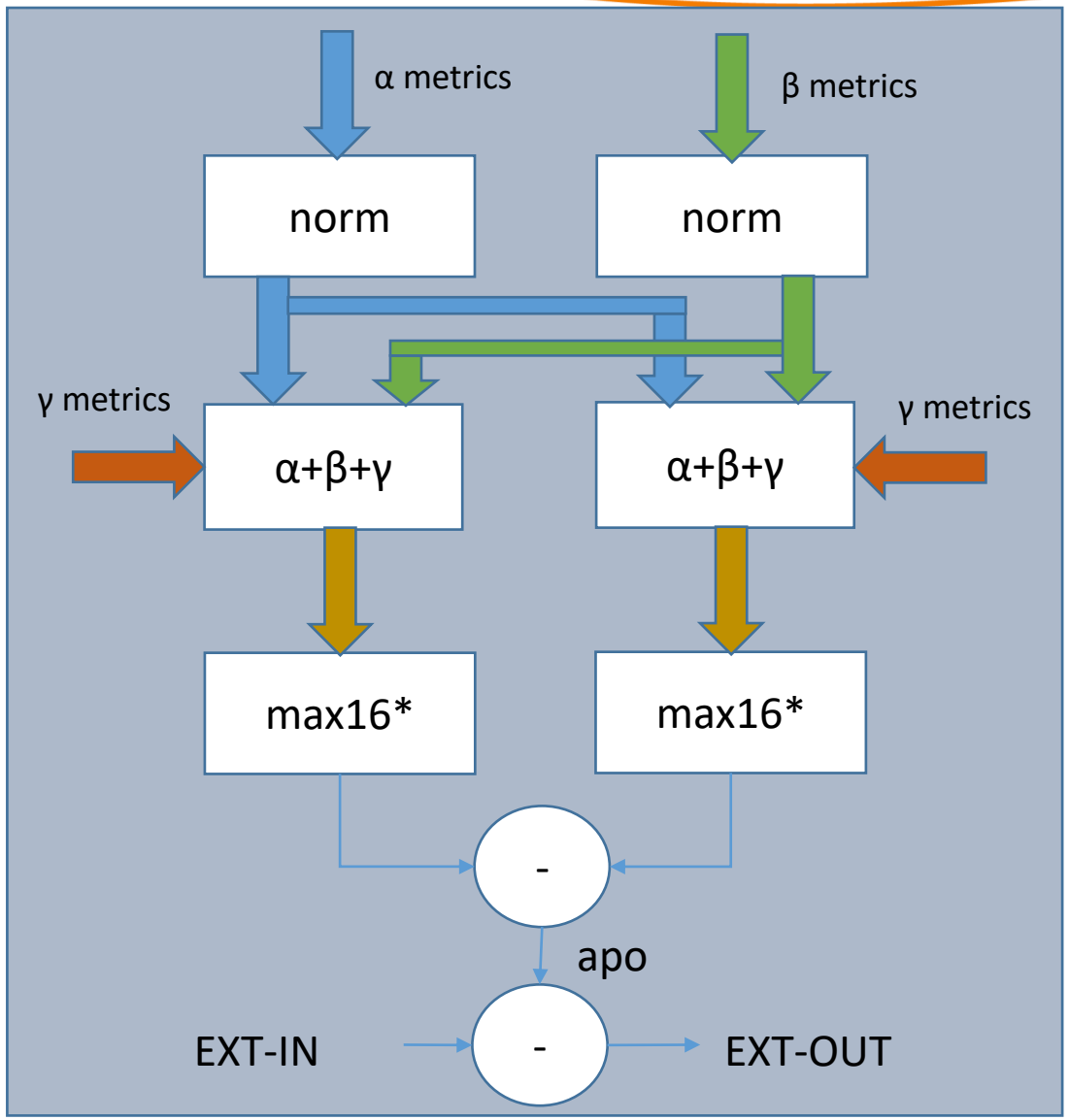
SISO architecture





Processor architectures





Encoder Architecture

- Given the specified throughput of 80 Mbit/s and the target clock frequency of 153 MHz, the encoding unit can proceed by encoding one information bit per cycle.
- No need of double input buffer as module receives 8 unencoded bits per cycle (see next slide)
- Key components:
  - A single input buffer (one M20K, less logic complexity)
  - The encoding unit, which includes the convolution encoder, the additional logic required to handle the termination bits and the puncture
  - A programmable counter to generate the write addresses for storing the received bytes in the buffer.
  - A second programmable counter to generate the in-order read addresses, which are used by the encoding unit to read the frame bits in the natural order
  - A permutation unit that implements the specified interleaving rules and generates the scrambled read addresses

Encoder Input

Encoder input buffer:

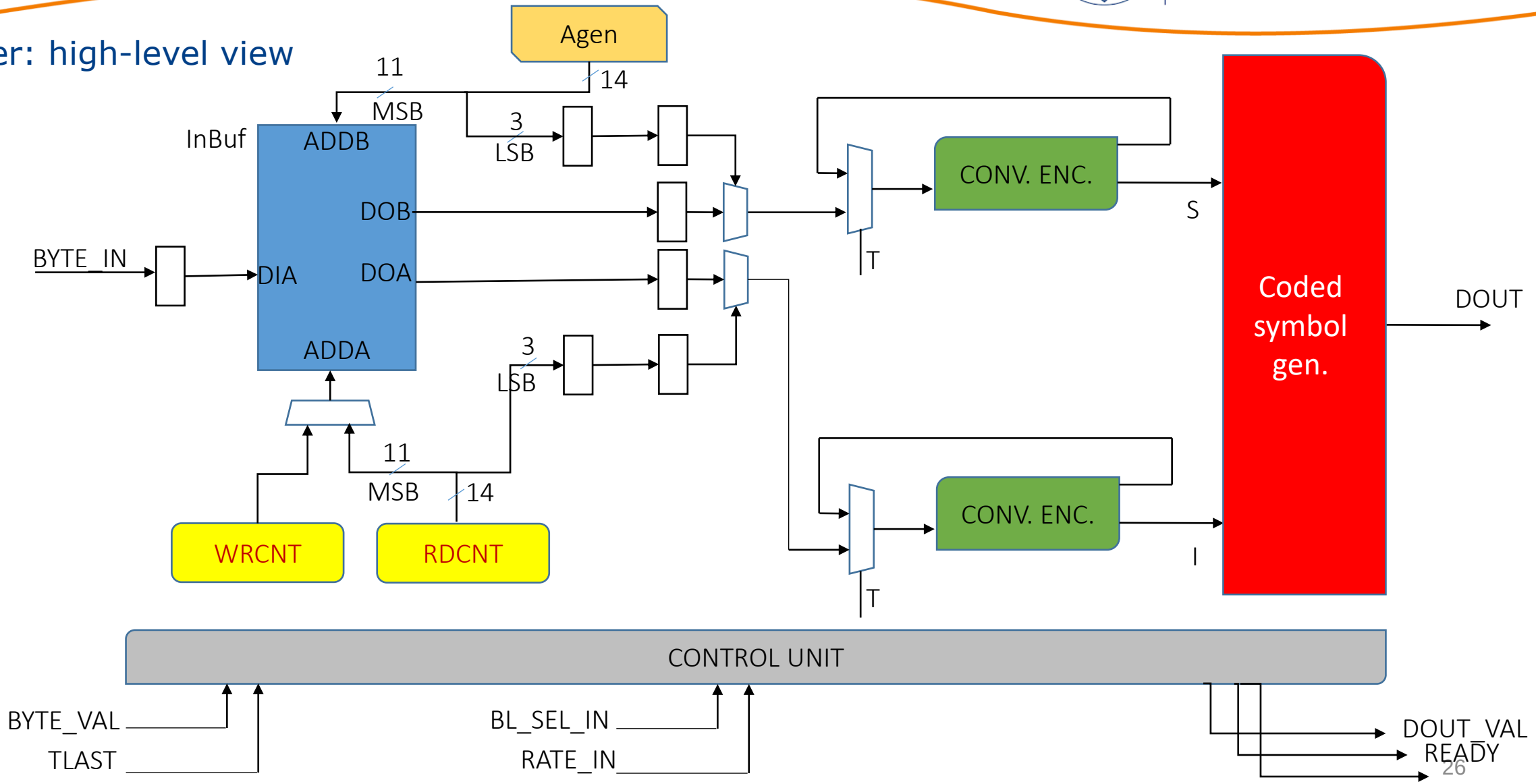
- With the byte parallel input interface, a frame is loaded in K/8 cycles.
- The encoder generates one encoded bit per cycle
- By using a single input buffer and running sequentially both the buffer loading and the frame encoding, the achievable throughput is

$$\frac{K}{K\left(1+\frac{1}{8}\right)T_{CK}} = \frac{8}{9}f_{CK}$$

- which is compliant with the throughput specification if the clock frequency is equal to 153.85 MHz

High-level view of the encoder

Encoder: high-level view



Encoder Output

- The two convolutional encoders (green blocks) receive the in-order and scrambled sequences from the buffer and generate the four bit coded outputs. After completing the sequence of k symbols, additional four steps are required two terminate the trellis (multiplexers)
- The Coded symbol generation unit (red block) provides the final coded symbols in a packetized way, based on the code rate. The output signal, BYTE_OUT, is an eight bit value for all supported code rates; however, the meaningful bits are aligned with the least significand positions of the byte

RATE	7	6	5	4	3	2	1	0
1/2	0	0	-	-	-	-	1a/1b	0a
1/3	0	1	-	-	-	1b	1a	0a
1/4	1	0	-	-	1b	3a	2a	0a
1/6	1	1	3b	1b	3a	2a	1a	0a

CCSDS Turbo codes Channel interleaver: Design and Performance results with tumbling spacecraft and solar scintillation channel models

Guido Montorsi, POLITO

CCSDS meeting, 28 May 2021

POLITECNICO
DI TORINO

DET
Department of Electronics and Telecommunications

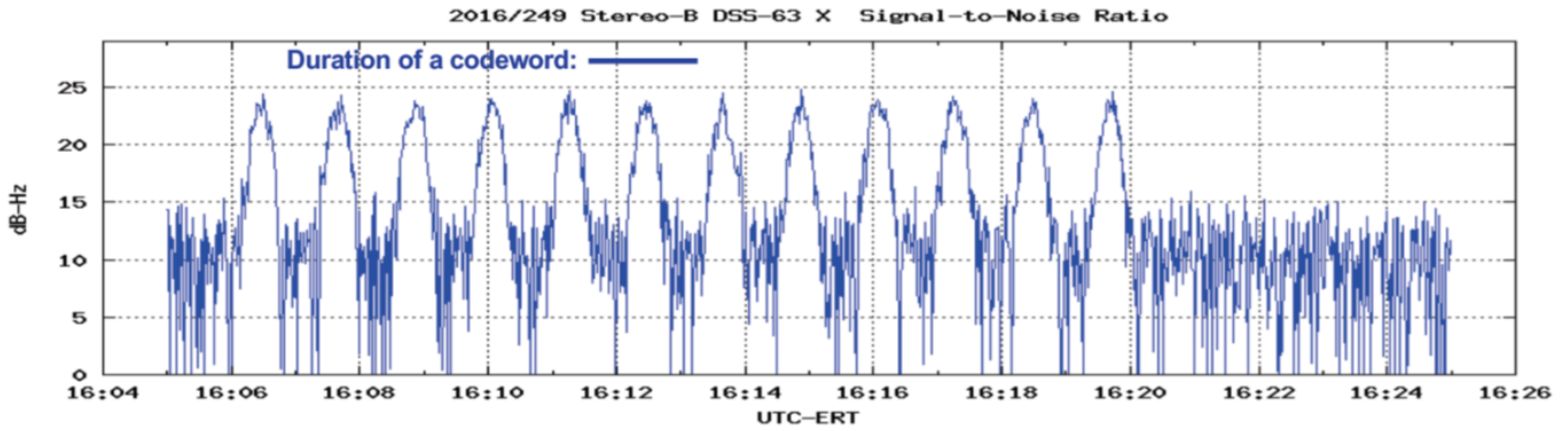
Background and contributions

- In CCSDS fall meeting 2017, JPL presented the Stereo-B anomaly, which is a practical example of a “tumbling spacecraft”
- This anomaly put in evidence that the Turbo decoders inherit a vulnerability to burst errors from their constituent convolutional codes
- In a previous contribution we proposed another channel model for considering other physical phenomena affecting the transmission like the fading induced by the solar scintillation
- A well know potential solution to such impairments, is to place a channel interleaver immediately following the turbo encoder
- In this contribution we provide performance results and design conclusions for the row-column interleaver and golden angle interleaver on the two considered channel models

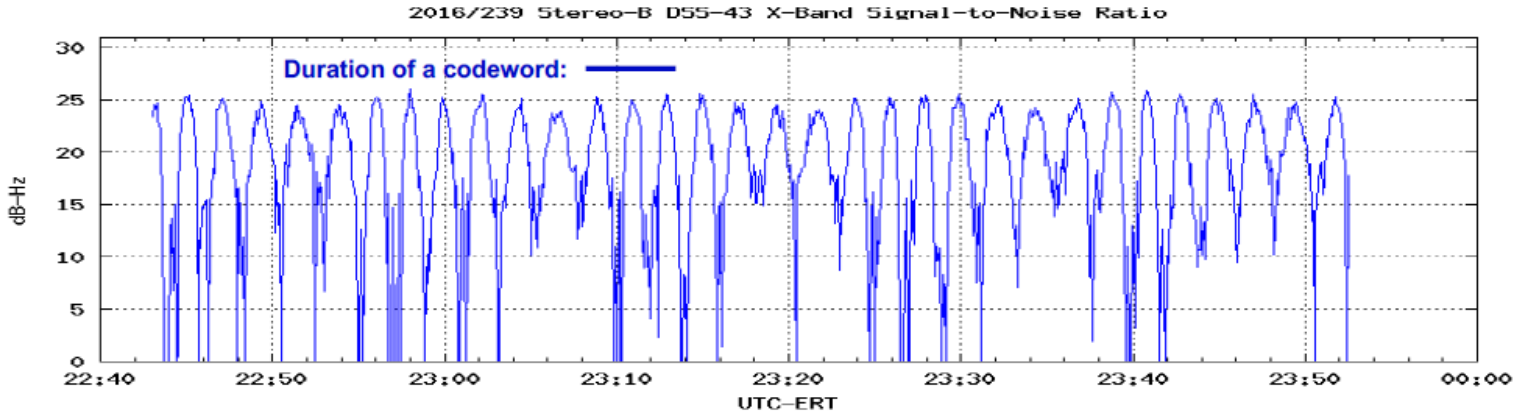
TUMBLING SPACECRAFT

Tumbling spacecraft

First example of oscillations of SNR with tumbling spacecraft



Second example of oscillations of SNR with tumbling spacecraft



Tumbling spacecraft model

- The two examples show an approximately sinusoidal and deterministic behavior of the SNR (in dB).
- A general model for such behavior is:

$$SNR(t) = SNR_0 + A \sin\left(\phi + 2\pi t \frac{\alpha}{T_f}\right) [dB],$$

where:

  - A is the SNR oscillations amplitude in dB,
  - α is the frequency of the SNR oscillations, normalized to the codeword rate ($1/T_f$)
  - T_f is the codeword duration, and ϕ is an arbitrary phase offset.
- As an example, for first figure, $A \approx 7$ dB and $\alpha \approx 1$, whereas for second, $A \approx 5$ dB and $\alpha \approx 3$.

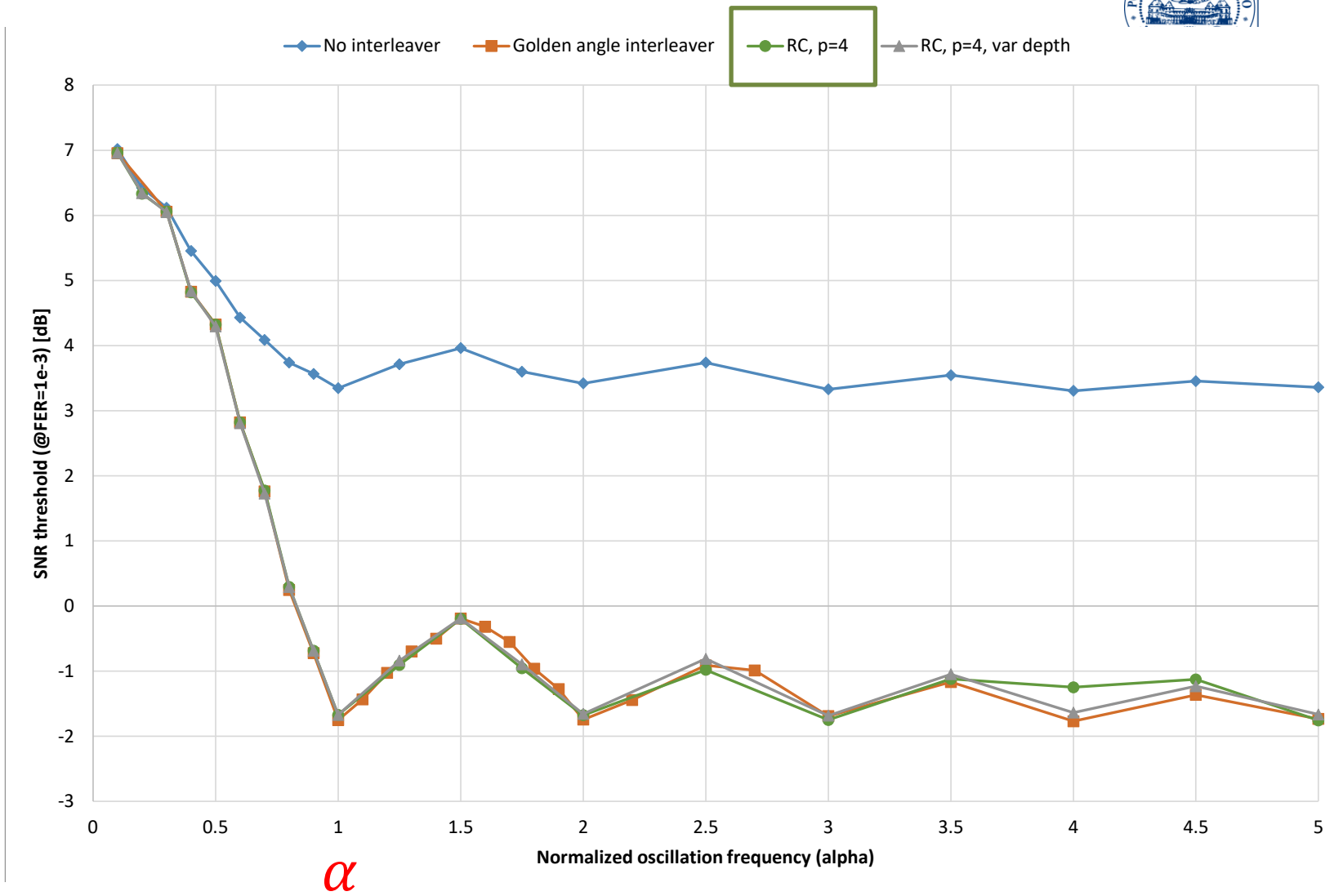
Tumbling spacecraft

- The value α in a general scenario can vary in a range that depends on the relative values of the baud-rate and the oscillation frequency of spacecraft.
- We report the performances as a function of α for four considered interleaver choices:
  - no interleaver,
  - golden angle interleaver,
  - Two row-column interleavers with properly tuned depth.

Tumbling spacecraft: performances

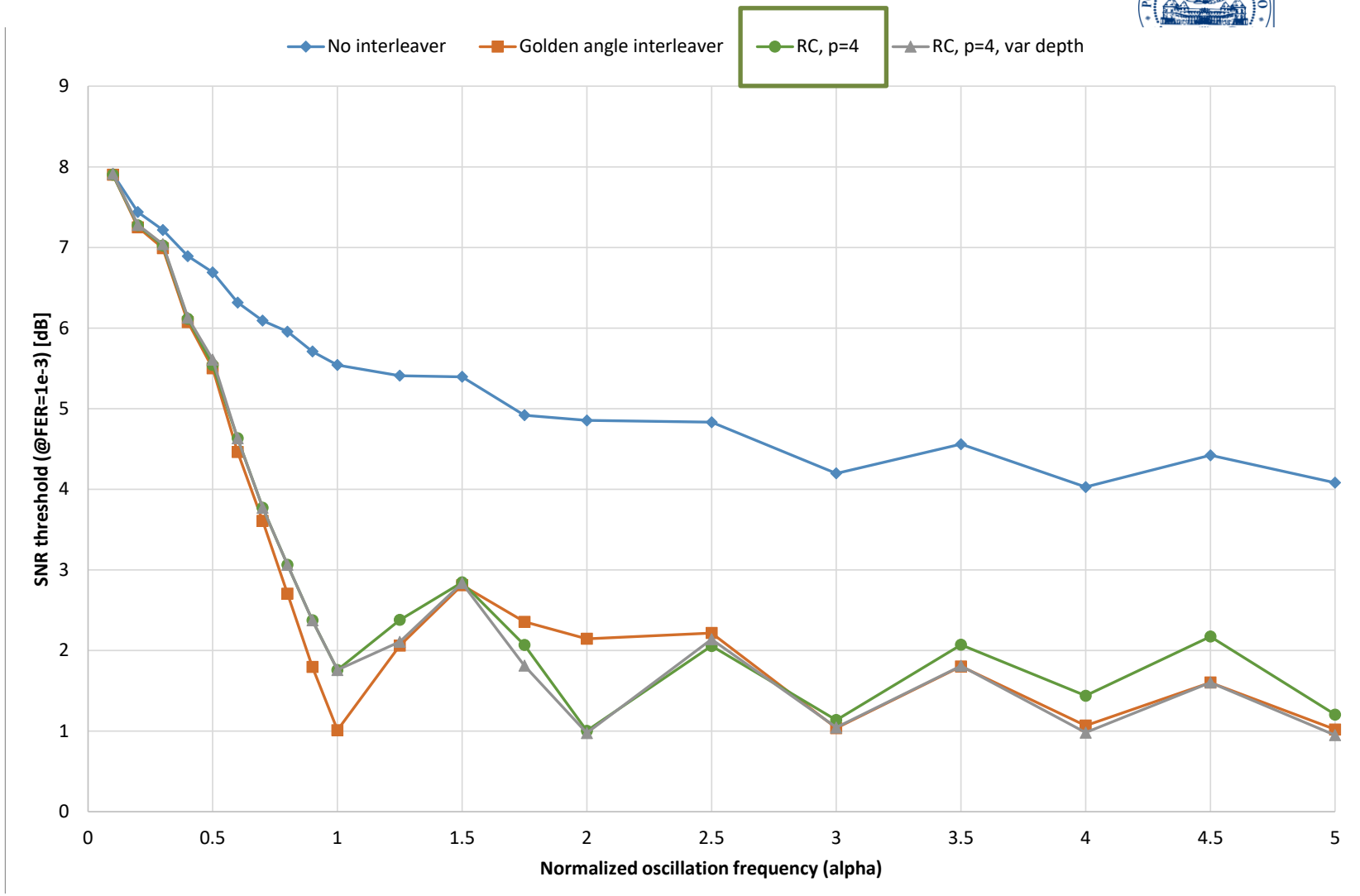
SNR thresholds ($\boldsymbol{SNR_0}$) vs relative periodicity of tumbling spacecraft.

$\boldsymbol{R_c}$=1/6, short codewords, .



Tumbling spacecraft: performances

SNR thresholds ($\boldsymbol{SNR_0}$) vs relative periodicity of tumbling spacecraft.
$R_c = 1/2$, long codewords, A=7.



Inter-frame interleaver

SOLAR SCINTILLATION MODEL

Solar Scintillation

- We proposed a non-deterministic fading channel for assessing the interleaver performance in the presence of solar scintillation phenomena.

$$r_k = a_k s_k + n_k,$$

- a_k is a Gaussian correlated stationary process with given spectrum and possibly nonzero mean values, yielding a Rician first order statistic,characterized by the rician factor K.

$$S_a(f) \propto \frac{1}{1+\left(\frac{f}{f_0}\right)^2} = \frac{1}{1+(T_c R_s f)^2}$$

Normalized coherence time

Comments

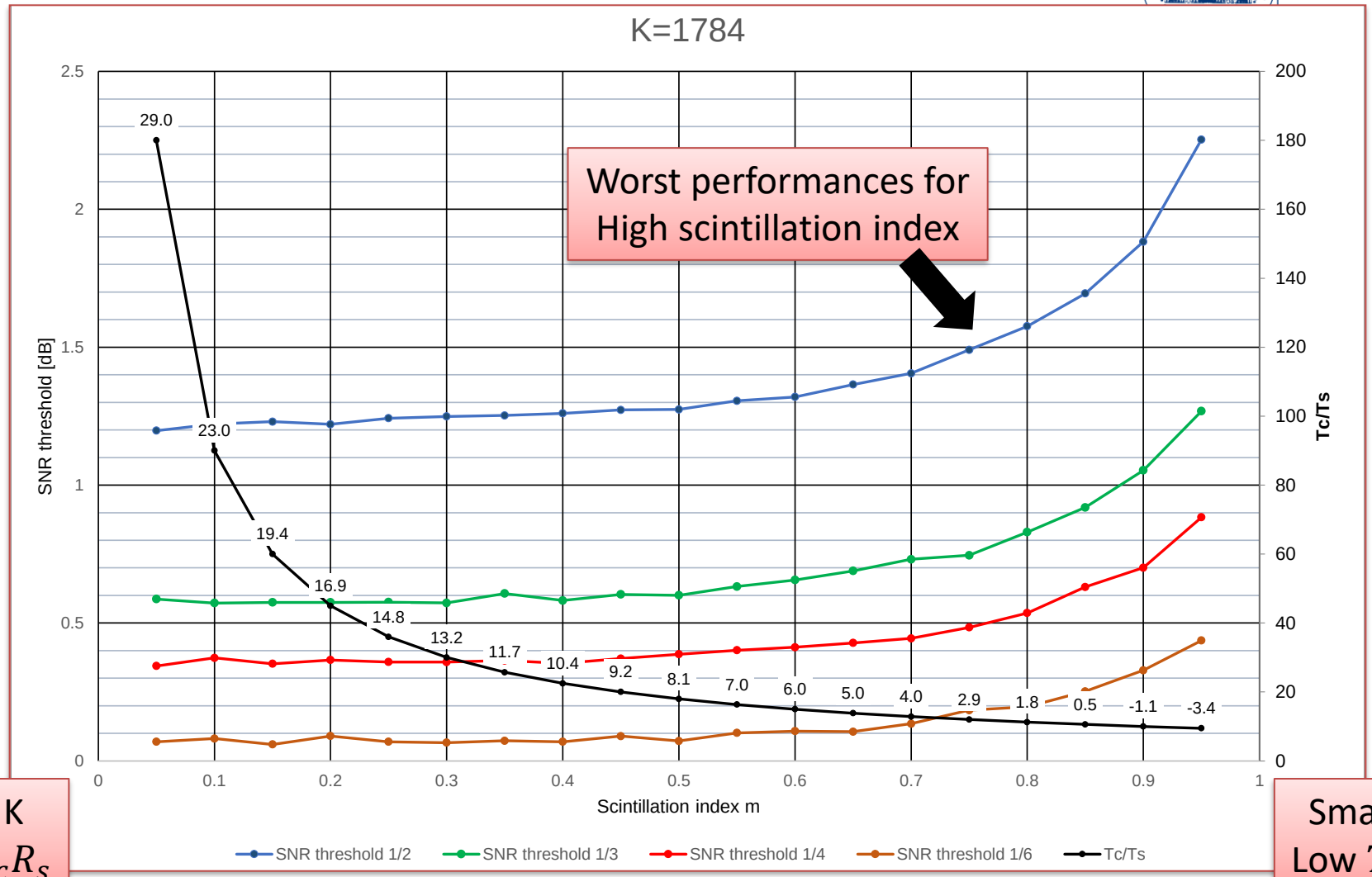
- In the solar scintillation model, both coherence time and Rician factor are related to the scintillation index *m*. The measured relationship with the coherence time is reported in [RD-2], while the Rician factor can be obtained as:

$$K = \frac{\sqrt{1-m^2}}{1-\sqrt{1-m^2}}$$

- Thus, when increasing the scintillation index both the coherence time and the Rician factor decrease, yielding opposite effect on the performance.
- To jointly consider these effects in the next figure we report the SNR thresholds versus the *scintillation index*.
- The black curve reports the values of normalized coherence time T_c/T_s (right vertical axis). The corresponding Rician factor is indicated in the label. The considered symbol rate in this case is $R_s = 10$ kbaud.

SNR thresholds

- Both Rician factor and coherence time varies as function of the scintillation index *m*
- Black curve reports values of coherence time (right vertical axis), while the corresponding Rician factor is reported in the label.
- The baud rate is Rs=10 kbaud



Large K
Large $T_c R_s$

Small K
Low $T_c R_s$

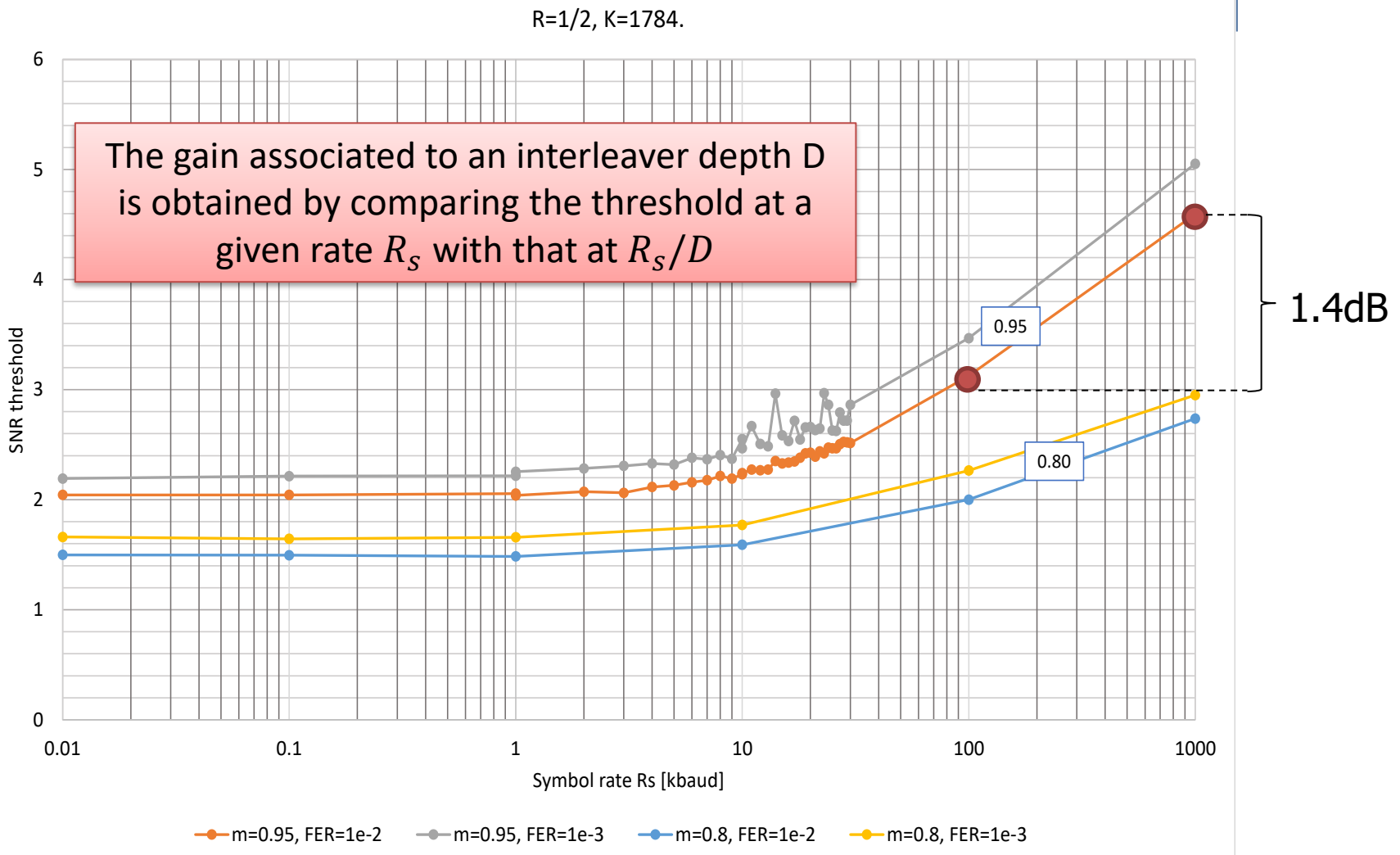
Coments

- The performance degradation increases for larger scintillation indexes, that is for smaller values of the coherence time and smaller Rician factors.
- As an example, with m=0.95, the Rician factor is K=-3.4 dB, and the coherence time around 10. The SNR threshold degradation is 0.4 dB for rate 1/6, 0.6 dB for rate 1/4, 0.7 dB for rate 1/3 and 1.0 dB for rate ½.
- The degradation is due to the transition from the performance associated to the AWGN channel (high values of K) and that associated to the Rayleigh Fading channel (low values of K).
- In order to assess the impact on performances of the insertion of an inter-frame interleaver, we focus on the case with highest loss, with R=1/2, K=1784.
- In next figure we show the SNR thresholds vs the baud rate for two high values of the scintillation index, corresponding to low Rician factors

SNR thresholds vs baud rate

for m=0.80 and m=0.95.

$R_c = 1/2, K = 1784$



Comments

- Increasing the baud rate increases the normalized coherence time $R_s T_c$ and consequently reduces the effective codelength.
- The reduction of the effective codelength in turns degrades the slope of the FER performance curve and consequently the threshold.
- The figure shows that performance degradation occurs only when $R_s > 10$ kbaud .
- The interleaver depth D has the effect of reducing the normalized coherence time
  - the plot allows to quantify the threshold gain introduced by the insertion of an inter-frame interleaver.
  - The gain is obtained by comparing the threshold at a given rate R_s with that at R_s/D
- For example, an interleaver depth D=10, with scintillation index 0.95, yields a 1.4 dB gain at baud rate 1Mbaud, 0.9 dB at 100 kbaud, 0.2 dB at 10 kbaud and negligible gain at symbol rate below 1kbaud. These gains would increase considering smaller FER values for the SNR threshold definition.
- Considering that the typical mission baud rates are below 10kbaud and that the considered case is the worst case (small block size, large code rate and large scintillation index), the adoption of an inter-frame interleaver is then not reccommended

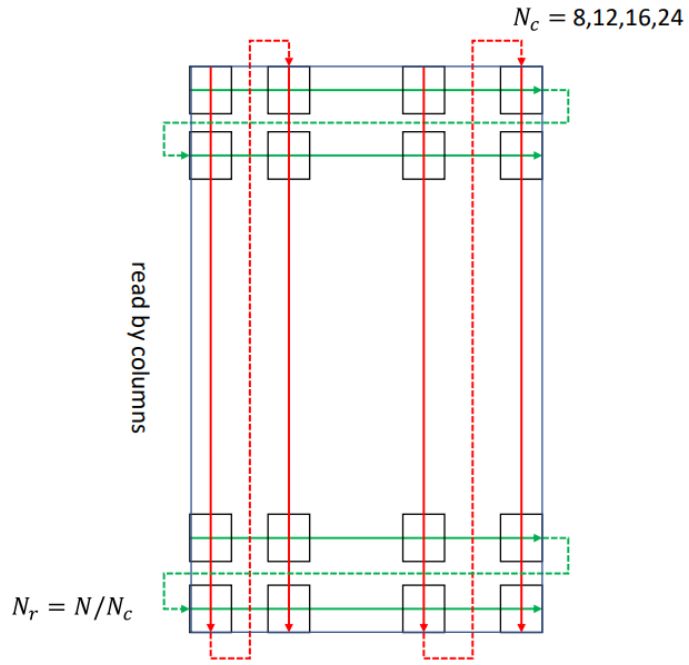
Summary of conclusions

- To summarize, the recommendation of this contribution are the following
- **The adoption of an intra-frame interleaver is strongly recommended**.
  - It has no impact of memory and latency of the system and provides vary large performance improvements in some realistic scenario.
- **Row-column interleaver is mildly recommended**.
  - It provides similar performances as the “golden angle” with a slight improvement in the permutation representation and memory access for the implementation.
- **The adoption of an inter-frame interleaver is not recommended.**
  - It has impact of memory and latency of the system and provides performance improvements in scenarios corresponding to large baud rates and/or low SNR oscillation frequencies, which are not considered relevant at the moment with the current assumptions

Interleaver specifications (proposal of standard modification)

- At the encoder side, the bits produced by the turbo encoder of rate $Rc = 1/n$, with $n = 2, 3, 4, 6$ are written on the interleaver rows of size $Nc = 4n = 8, 12, 16, 24$. Each row thus contains four consecutive outputs of the two constituent encoders of the turbo encoder.
- The bits are then read out from the interleaver column by column. The column length Nr depends on the codeword length as $Nr = N/Nc$.

R_c	1/2		1/3		1/4		1/6	
N_c	8		12		16		24	
K	N	Nr	N	Nr	N	Nr	N	Nr
1784	3576	447	5364	447	7152	447	10728	447
3568	7144	893	10716	893	14288	893	21432	893
7136	14280	1785	21420	1785	28560	1785	42840	1785
8920	17848	2231	26772	2231	35696	2231	53544	2231



- **Rationale**: to have p =4 consecutive trellis steps of the constituent encoders affected by fading amplitudes that are spread in time "as much as possible". This ensures maximum time diversity on the short error events for both constituent encoders. The choice of 4 is well matched to the numerology of the standard as 4 is a divisor for all values of N n = (1788, 3572, 7140, 8924).
- Values larger than 4 for p were also considered at the design stage but were found to provide negligible gains wrt this simpler solution.

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

SW4 Description

Functionality

The delivered "**SW4" delivered on 28th July 2021** contains the self-contained software for the simulation and the text vector generation of the full VIRTUDE system

TX

- Turbo encoding
- Intra and inter frame interleaving
- Randomizer
- Modulation

Channel

- AWGN or two models of fading channel

RX

- Demodulation
- Frame synchronization
- Derandomizer
- Deinterleaving
- Turbo decoding

A target FER can be set. An SNR loop in steps of 0.1 dB ends when the measured FER falls below the target FER

If desired, by uncommenting a section of code in the main program, the user can write test vectors taken from encoder sections in an output file

The project CCSDS_Sync can be used to check the performance of frame synchronizer

SW4 Description

File and Folder structure

CCSDS_Full: Main files and input text file for the simulation and BER statistic of the full VIRTUDE system

- The MSVC++ solution "CCSDS_Full.sln" can be used to build all executables
- "test_CCSDS_PCCC.cpp" is the main file
- "input_PCCC.txt" is the input file for configuring the simulation parameters

CCSDS_Sync: Main file and input text file for the simulation and performance analysis of the frame synchronizer

The project can be accessed from the previous solution CCSDS_Full.sln

Classes: All TOPCOM++ classes (declarations and definitions) used by the programs

Output results are stored in the "Data" subfolders

Input file (input_PCCC.txt)

```
Name output file
Data/temp
Number of blocks
1000000
Eb/N0 gap [db]
0
Target FER
1e-2
Size (0-3)
3
Rate
16

Fading?
0
Thumb. Spacecraft period?
1
Thumb. Spacecraft PP gain [dB]
7
Rice factor [dB]
10
Normalized Coherence Time  (Tc/Ts)
10

Number of iterations decoder
10
Factor
4
Q LLR
5
Q EXT
7
```

Prefix output file

Starting Eb/N0 as gap from capacity

Set the target FER for simulations

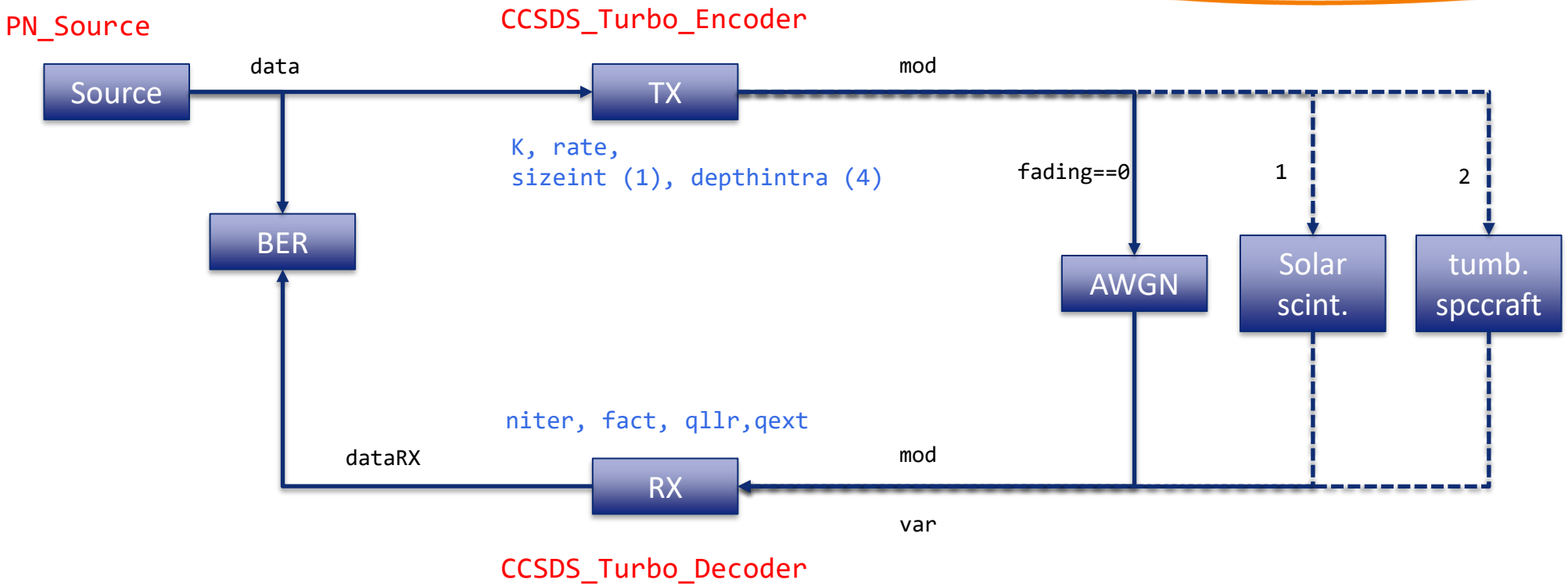
Code parameters

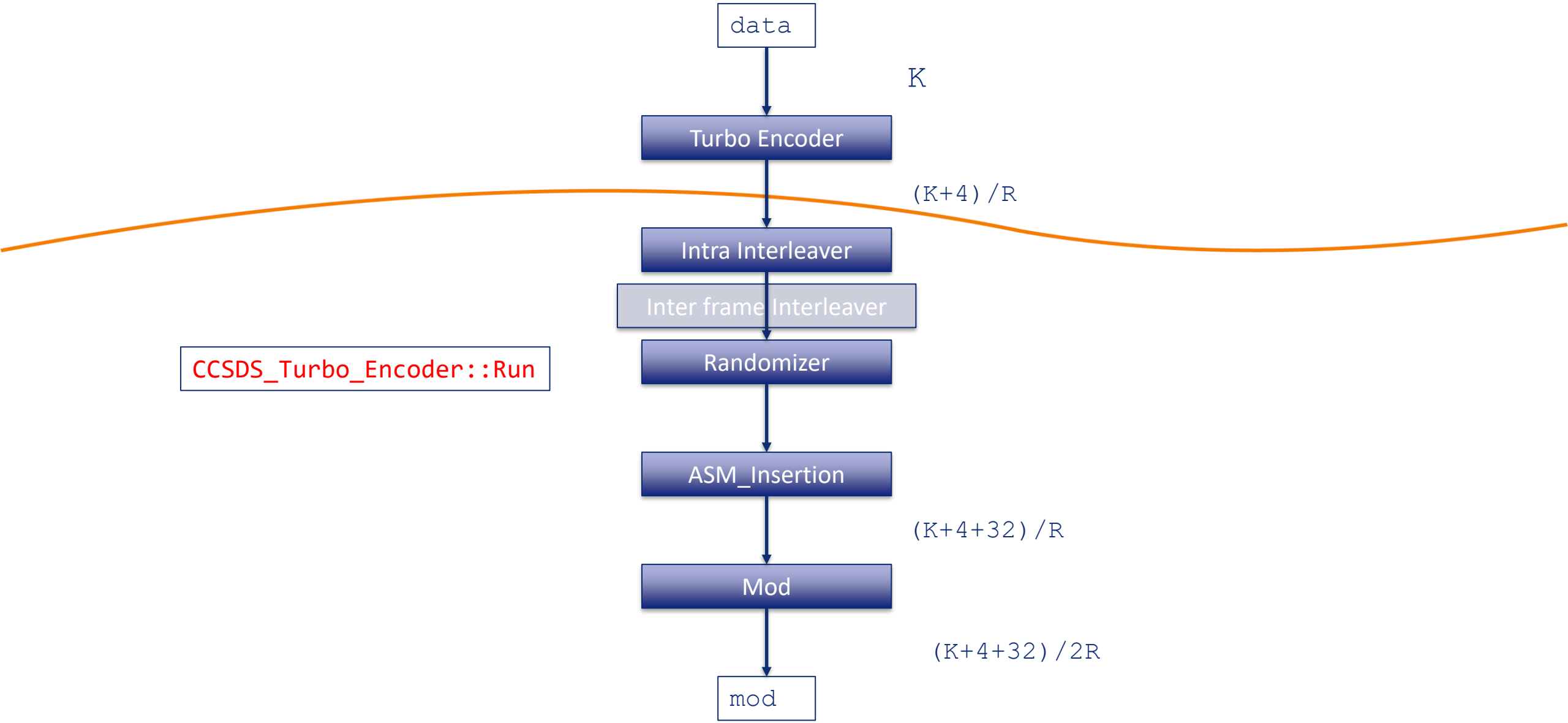
Channel type (1: Solar sc., 2: Tumb. Spacecraft, 0: AWGN only)

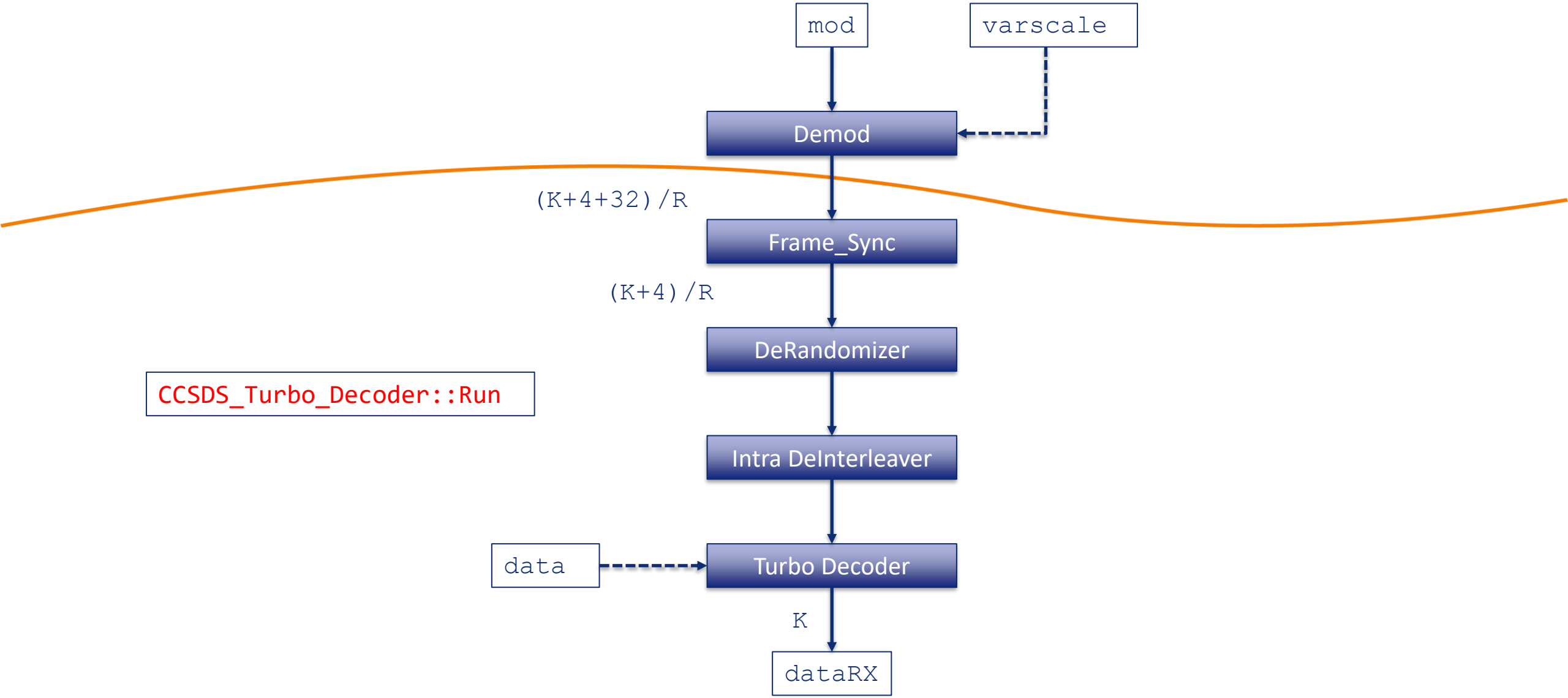
Parameters for fading channels

Decoder parameters

Structure of main simulation loop







Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

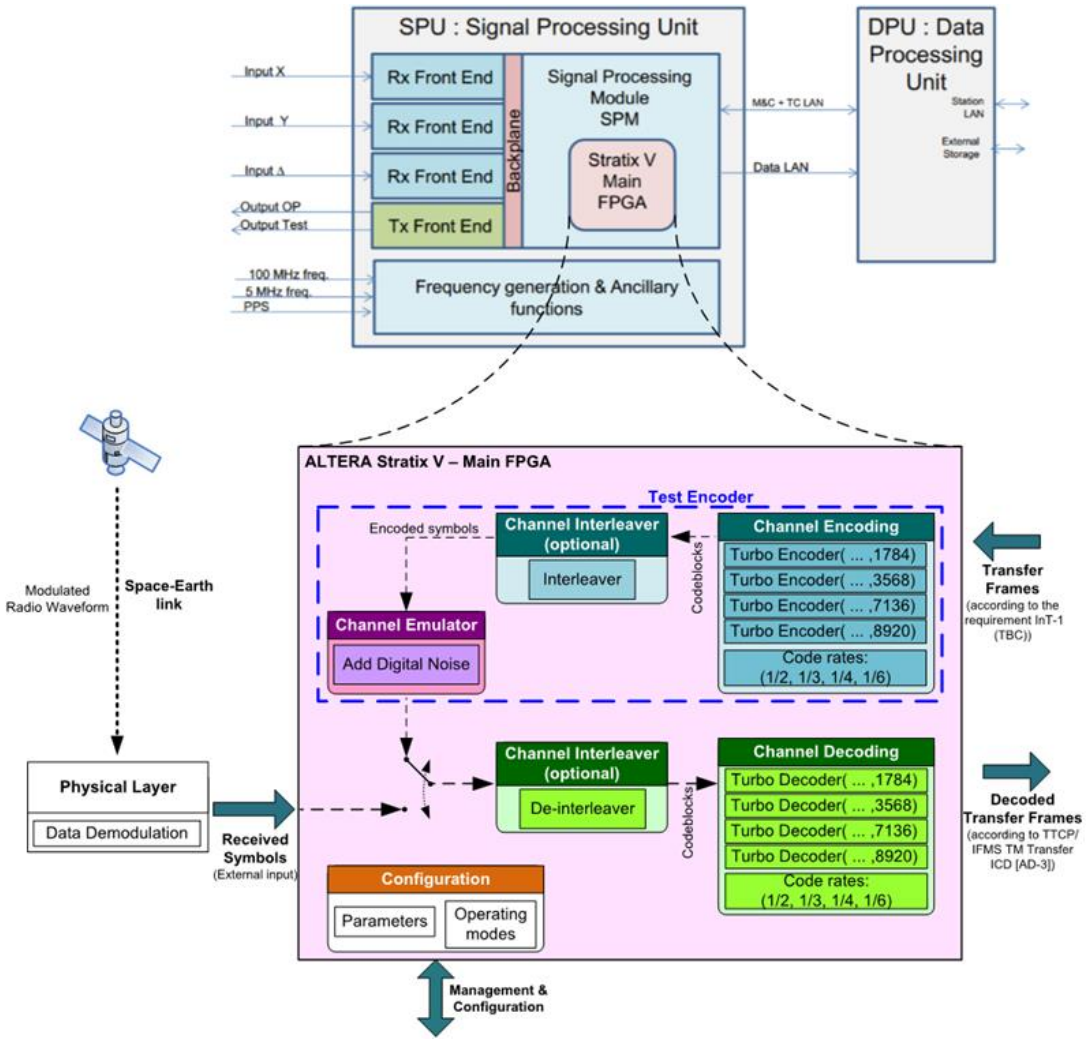
15:45 – Q&A

16:00 – Meeting closure

TTCP

- Platforms available at the TTCP:
  - CPU
  - ARM based FPGA
  - FPGA
- Integrated in two types of units/devices:
  - Data Processing Unit (DPU)
  - Signal Processing Modules (SPM)
- VIRTUDE modules will be integrated in the ALTERA Stratix V 5SGXAB FPGA within the SPM of the SPU with constrained resources:

Features	Stratix 5SGXEABN2F45C2LN [%]
ALMs	56.9 %
M20Ks	71.6 %
DSPs	21.6 %



Interface Requirements

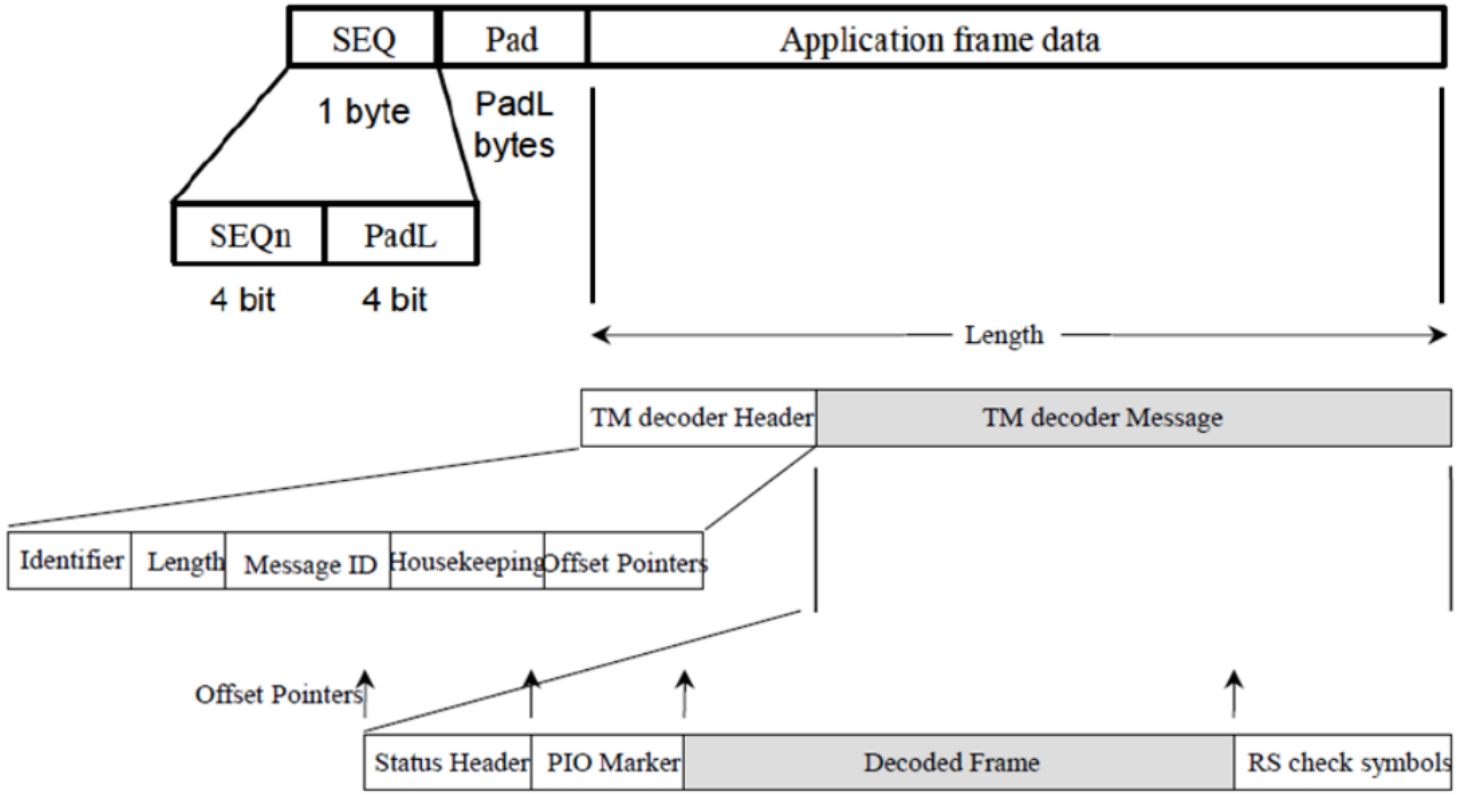
- The breadboard shall receive soft-symbols at the maximum symbol rate of 320 MSym/s:
  - Corresponds to a useful data rate of 80 Mbps at rate of 1/4 or 53 Mbps at rate of 1/6
  - Thus, the input interface needs to support a bit rate of 2.56 Gbps (each symbol is 8 bits)
  - Hence, the 10 Gb Ethernet interface is a suitable interface for the **input interface**!
- Typical Ethernet interfaces could be used either for the **output interface** or for the **monitoring and configuration interface**

Req. Id.	Requirement Description	Prop. Verif.	Comp. with SOW	Comments
[Int-1]	- The input and output data must access the breadboard through a LAN interface (TBC).	T, RD	Y	As described in section 7.3.1.
[Int-2]	- The output data must be delivered to the LAN interface with the format defined in [AD-3].	T, RD	Y	As described in section 7.2.1.
[Int-3]	- The input data to the decoder, when coming from the external input, must have a format TBD, but in the form of serial 8-bit soft decisions, a strobe and a timecode.	T, RD	Y	As described in section 7.2.2.
[Int-4]	- The monitoring and control of the breadboard must be performed through a LAN Interface.	T, RD	Y	As described in section 7.3.1.

Req. Id.	Requirement Description	Prop. Verif.	Comp. with SOW	Comments
[PeR-1]	- The breadboard modules shall support data rates up to 80 Mbps in the TTCP SPM main FPGA.	T, RD	Y	

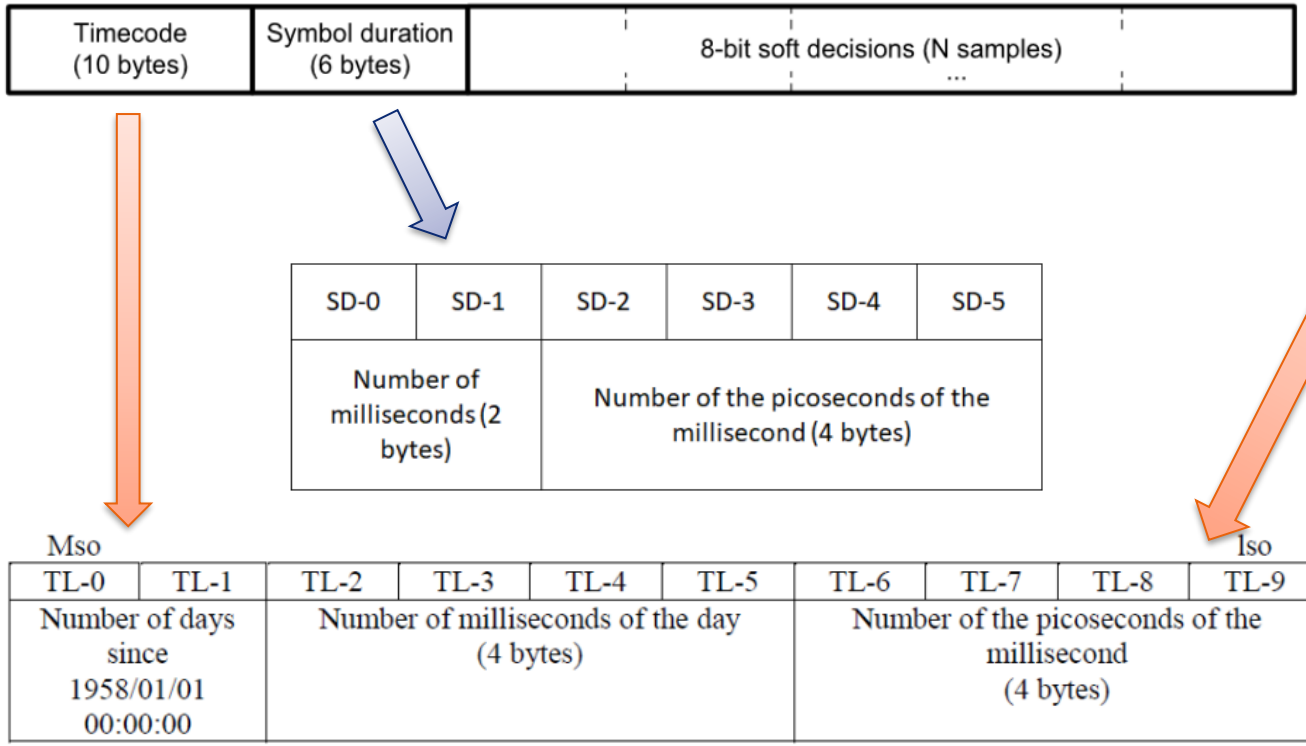
Output interface – TTCP-ICD-TM

- The application frame has 2 possible formats for:
  - **Data Message**: includes the decoded frame alongside other header fields;
  - **Heartbeat Message:** sent periodically when no data is outputted to keep the connection active.

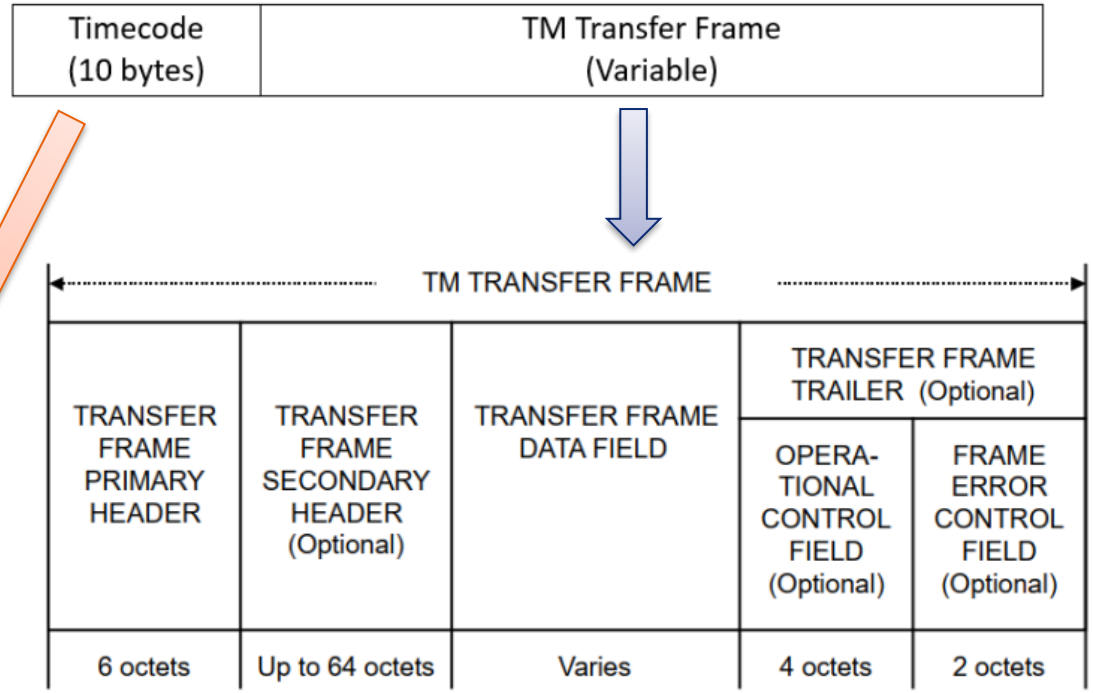


Input Inteface – Protocols

- **Soft-Symbols**
  - **Timecode**: reception time of first sample in UDP frame
  - The length of the soft-symbol frame is not fixed!

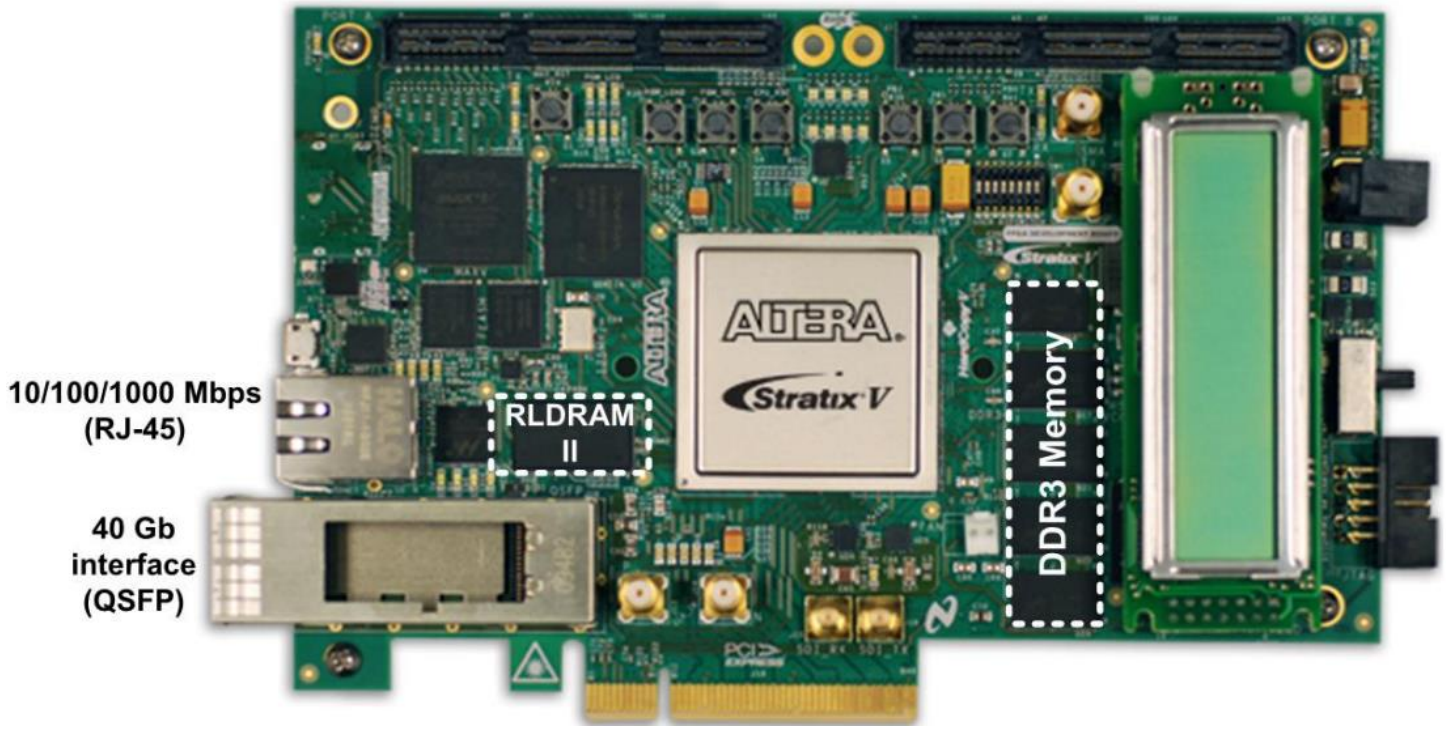


- **Transfer Frames**
  - For VIRTUDE, the FECF field is mandatory!
  - TM Transfer Frame: 223, 446, 892 or 1115 bytes



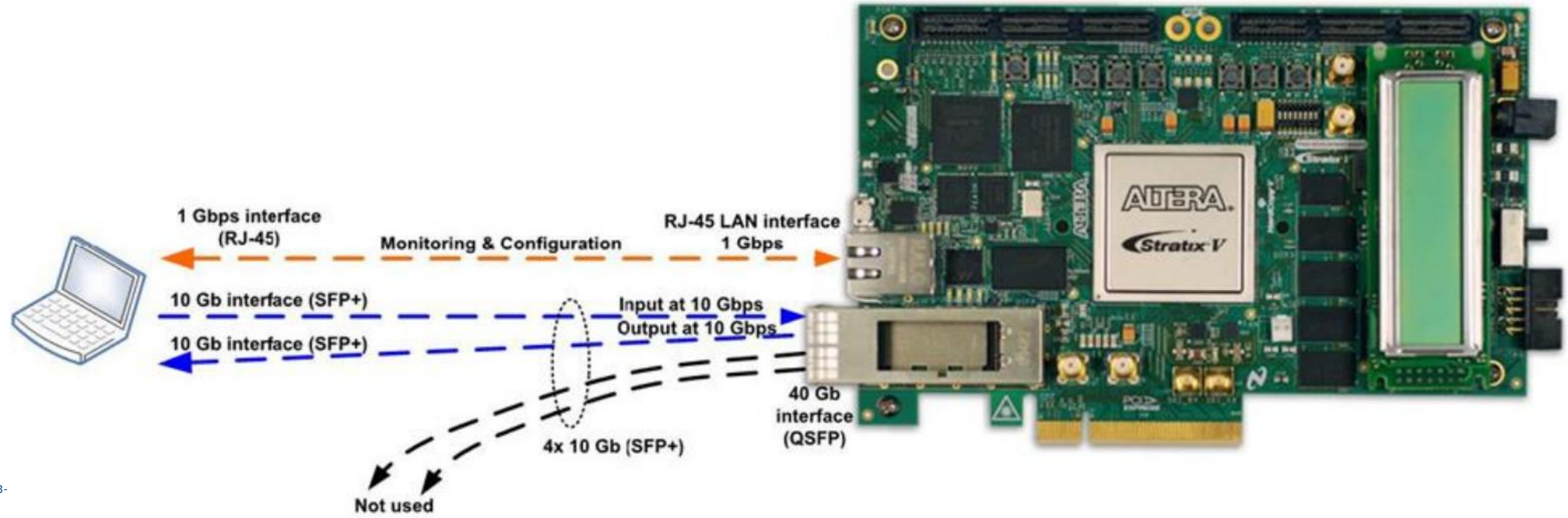
Selected COTS for the Project (1/2)

- The “Stratix V GX FPGA Development Kit” was the COTS chosen as it includes external memory
  - Foreseen in case there was a lack of M20K modules for the Interleaver but ended up not being required!
- This COTS has a 40 Gbps (QSFP+) and a 10/100/1000 Mbps (RJ45) Ethernet interfaces
  - The 40 Gb interface is split in 4x 10 Gbps Ethernet interfaces using an adapter cable from QSFP+ to 4x SFP+



Selected COTS for the Project (2/2)

- The 3 physical interfaces between the selected COTS and the external computer then are:
  - 10 Gbps – Input interface (either to receive Transfer Frames or Receive Symbols);
  - 10 Gbps – Output interface (Decoded Transfer Frames or Heartbeat Messages);
  - 10/100/1000 Mbps – Monitoring and Configuration (M&C) interface.
- The M&C interface was implemented at 1 Gbps due to a limitation of the IP core used in the project
  - The output interface did not require 10 Gbps but the RJ-45 was already selected for the M&C interface



Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

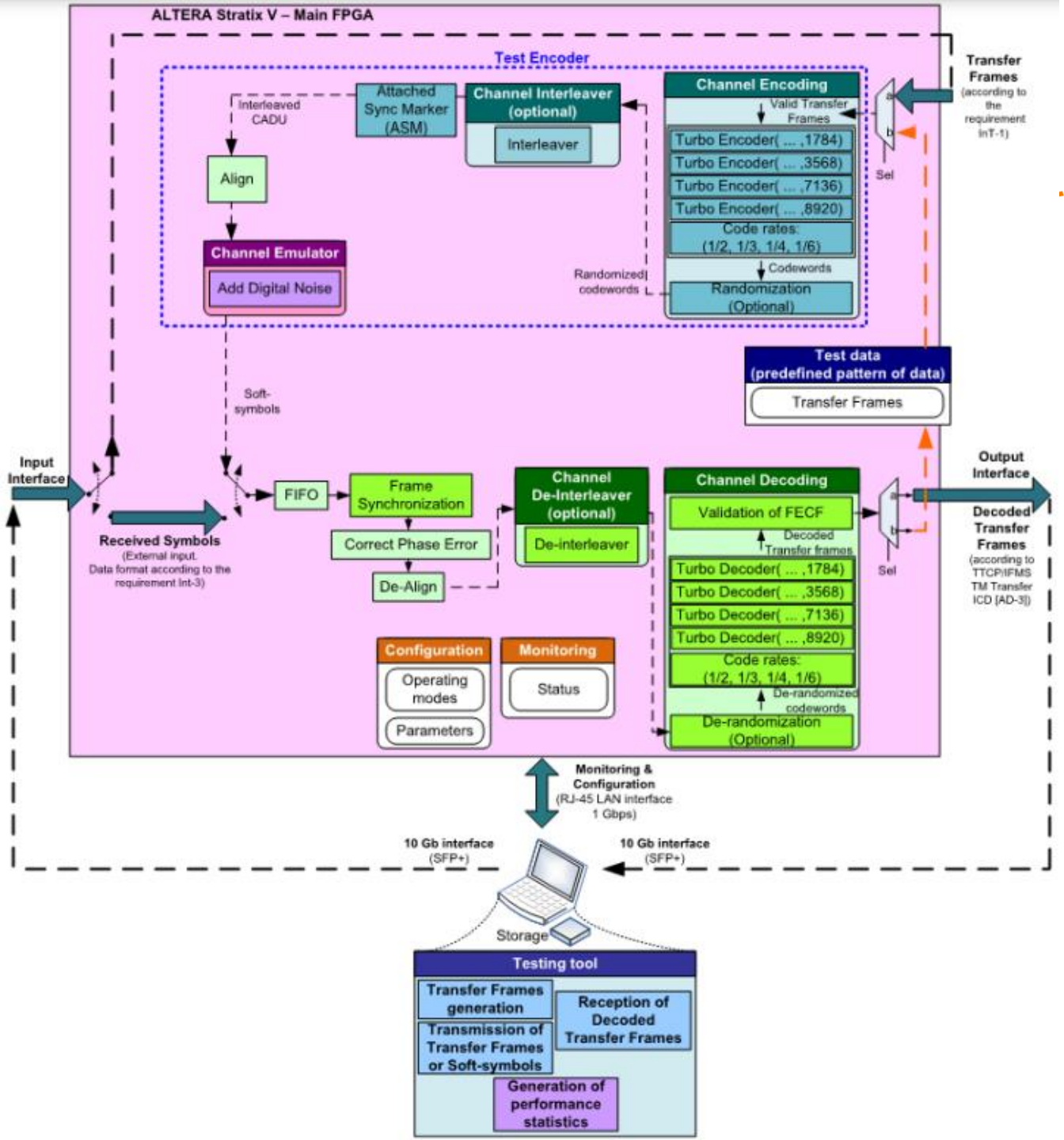
15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

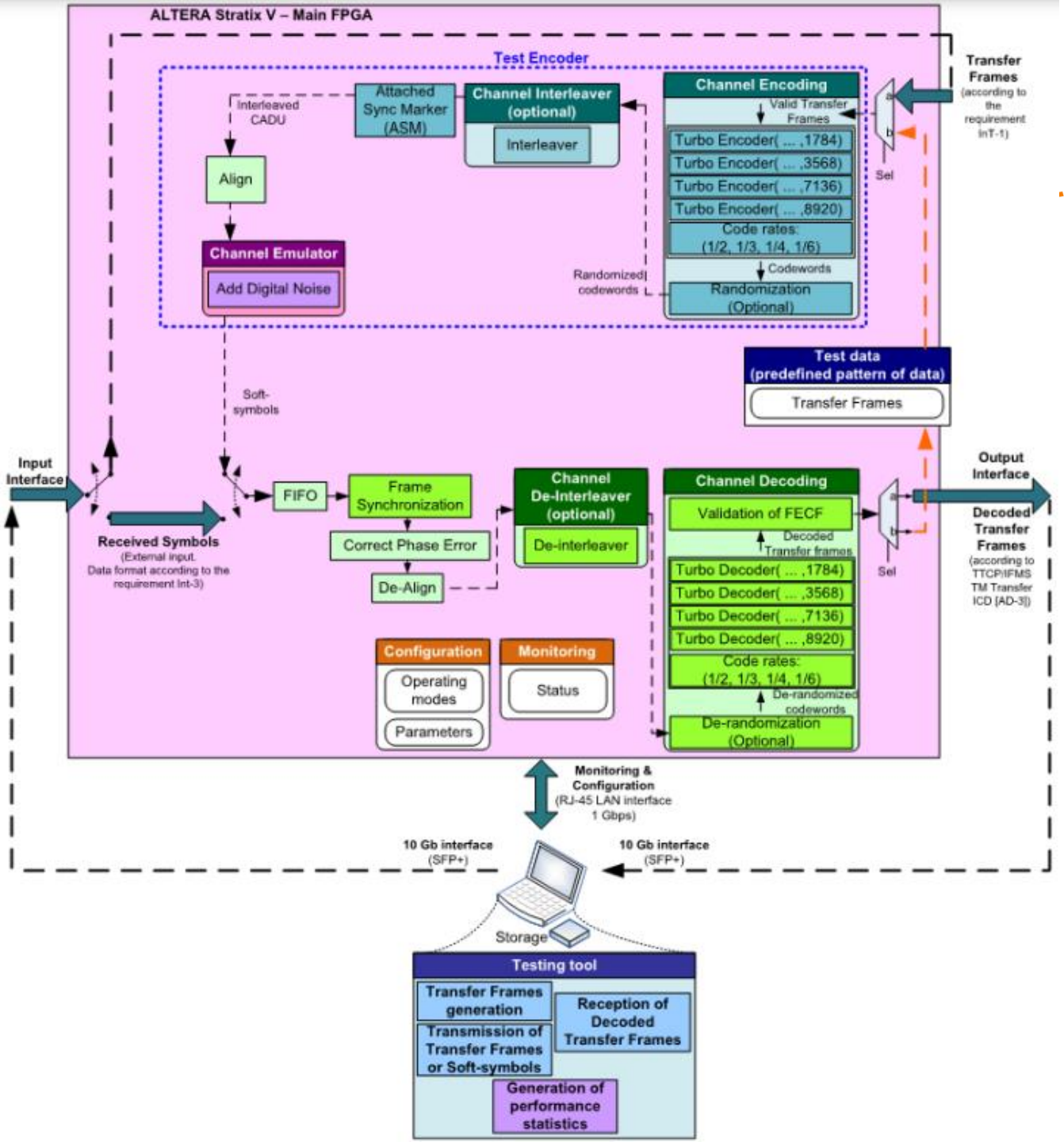
System Decomposition

- **Breadboard:**
  - External Interface
  - Transmitter Chain
  - Receiver Chain
  - Auxiliary Modules
- **Testing Tool**
  - Runs in external computer
  - Injects Transfer Frames or Soft-Symbols
  - Receives Decoded Transfer Frames
  - Measures BER/FER and throughput



Data Flow and Throughput

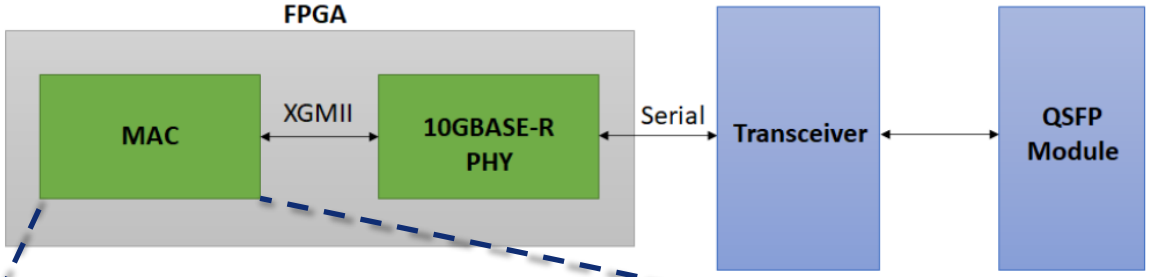
- Target bit rate of 80 Mbps
  - 320 MSps for the code rate of 1/4
  - 318 MSps for the code rate of 1/6
- Keep up with 136.75 Mbps from Encoder
  - 6 XOR in parallel in Randomizer
  - Write 2/3/4/6 bits in parallel in Interleaver
- Keep up x8 parallelism for Turbo Decoder
  - Read 8 bits from Interleaver concurrently
  - Add noise to 8 symbols in Channel Emulator
  - Write/read 8 symbols in parallel in De-Interleaver
  - Apply Boolean logic to 8 symbols in De-Randomizer
- Decoder outputs 1 byte per clock cycle
  - Validate FECF byte by byte



External Interfaces – IP Cores Evaluation

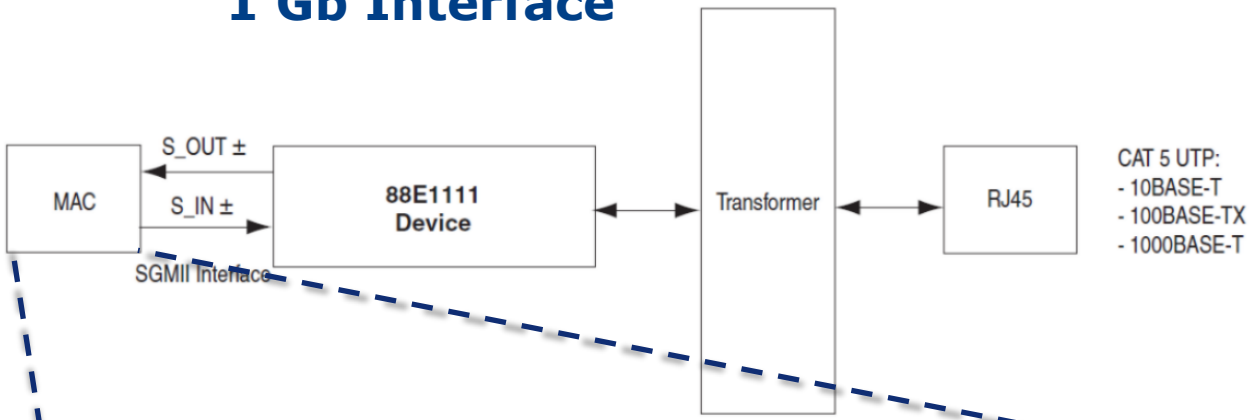
- Evaluated open-source MAC IP cores for both 10G and 1G Ethernet Interfaces
  - The one selected comes from the same source due to its permissive license and includes an UDP stack
- The MAC IP core chosen for the M&C interface only supports 1Gb connection

10 Gb Interface



Name	License	Language	Data interface
10G Ethernet MAC [OpenCores/10GEthMAC]		Verilog	Native
Ethernet 10GE MAC [OpenCores/Eth10GMAC]	LGPL	Verilog	Wishbone
Ethernet 10GE Low Latency MAC [OpenCores/LowLatMAC]	LGPL	Verilog	Native
Verilog Ethernet Components [GitHub/VerilogEth]	MIT	Verilog	AXI-Stream
Open Source 10GbE MAC for FPGAs [GitHub/nfmac10G]		Verilog	AXI-Stream

1 Gb Interface

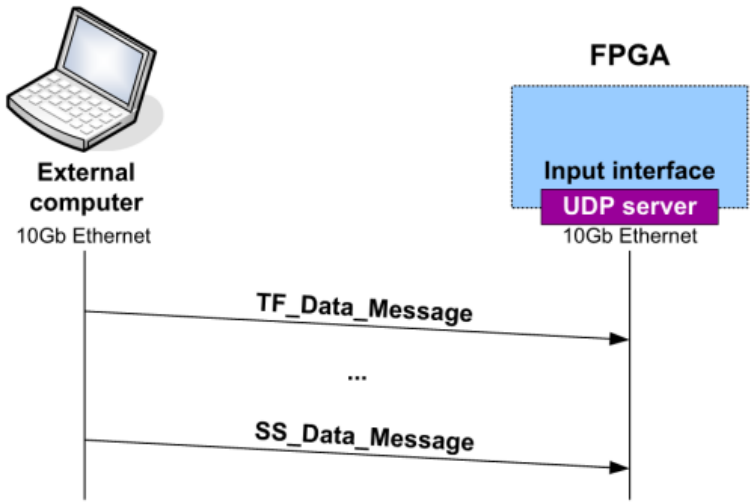


Name	License	Language	Data interface
10_100_1000 Mbps tri-mode ethernet MAC [OpenCores/TEthMAC]	LGPL	Verilog	Native
Ethernet 100/1000 Mbps [OpenCores/Eth100_1000]	LGPL	VHDL	Native
10/100/1000 VHDL Ethernet MAC [GitHub/EthMAC]		VHDL	Wishbone
Verilog Ethernet Components [GitHub/VerilogEth]	MIT	Verilog	AXI-Stream

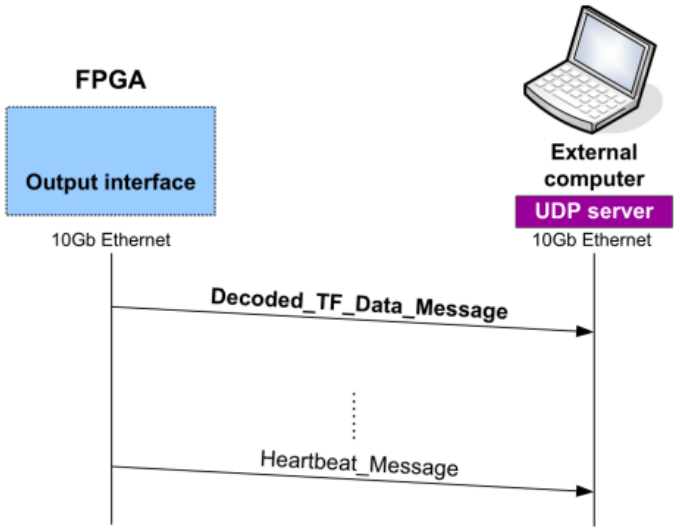
External Interfaces – Message Flow

- The 3 interfaces use the UDP protocol
  - The Input and Output Interfaces have no flow control
  - The M&C Interface adopted user-defined acknowledgment messages to ensure the configurations are properly set and to receive the results of the Unit Tests

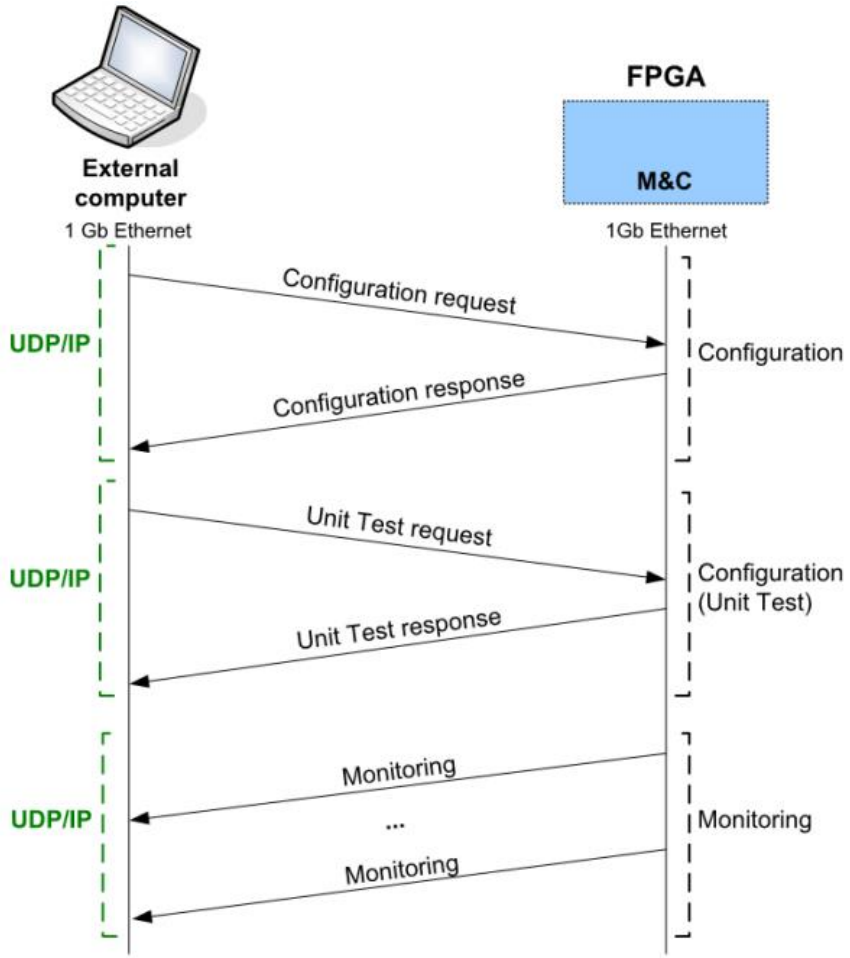
Input Interface



Output Interface



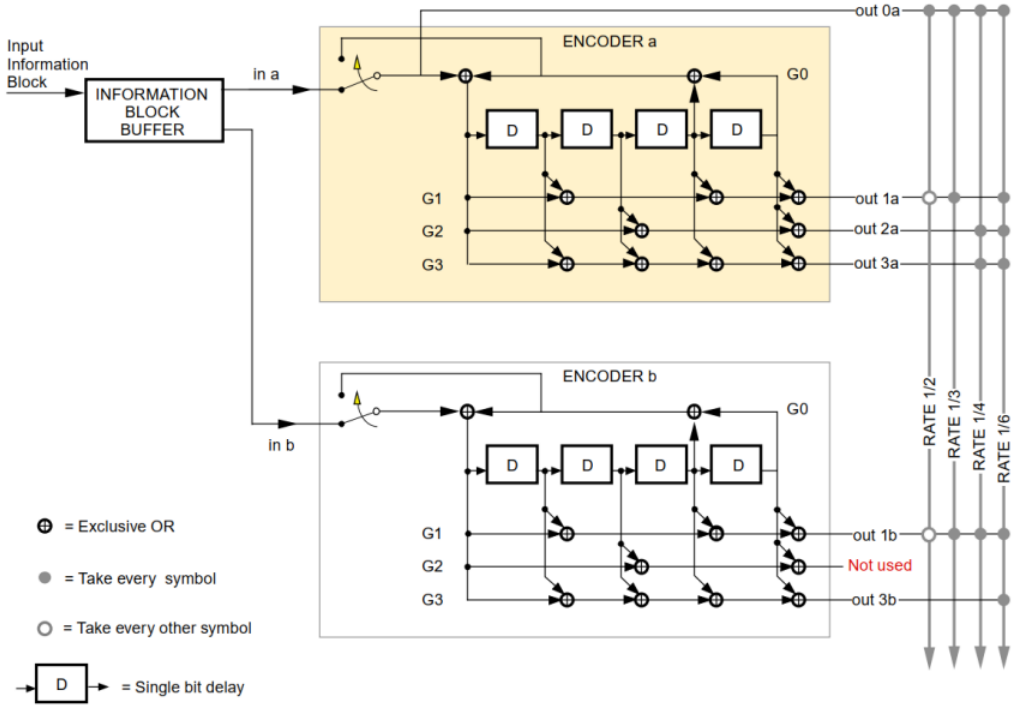
M&C Interface



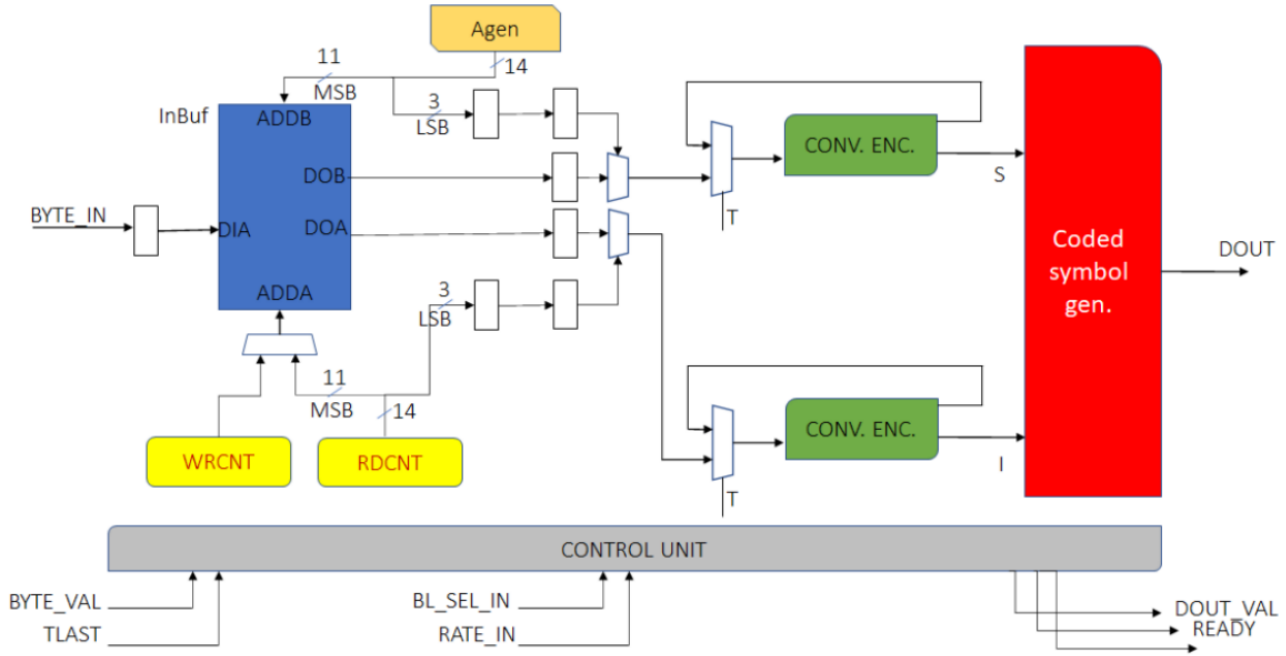
Transmitter Chain – Turbo Encoder

- First module of the transmitter chain based on a combination of two convolutional encoders
- For each received information bit, it generates the parity symbols outputting 2, 3, 4 or 6 symbols per clock cycle for the code rates of 1/2, 1/3, 1/4 and 1/6

Block Diagram



Actual Implementation

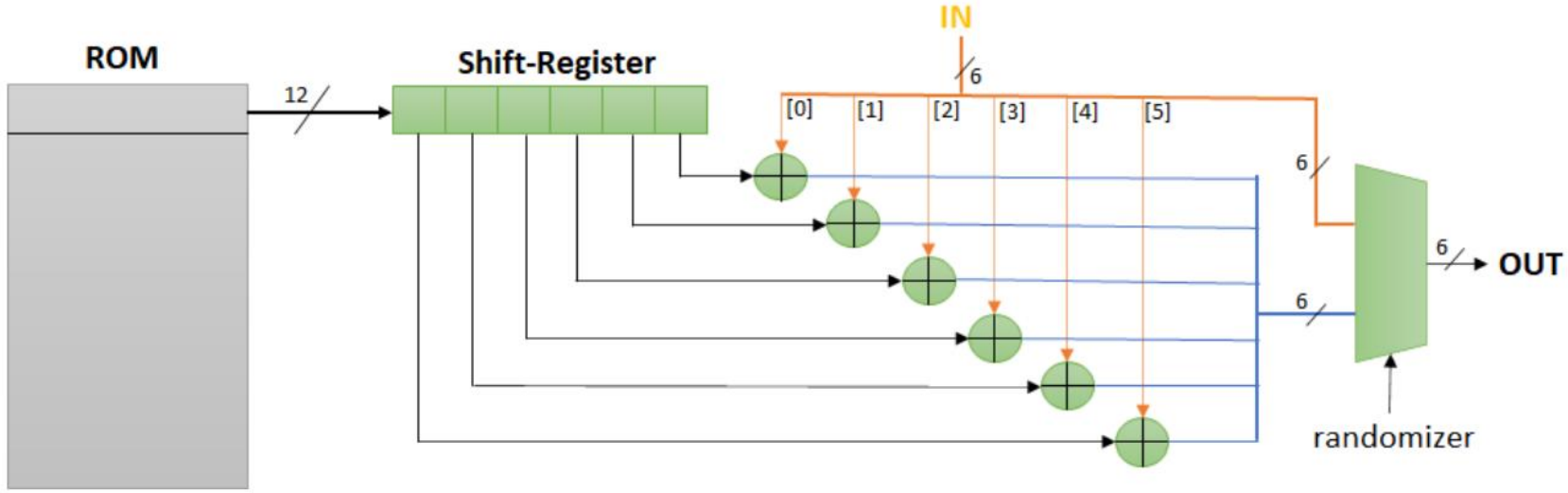


Transmitter Chain – Randomizer

- Ensures sufficient bit transitions to keep receiver symbol synchronizers in lock by applying an exclusive-OR to each bit of the codeword with a standard pseudo-random sequence
  - This sequence is fixed, repeats each 255 bits and is generated by the following polynomial:

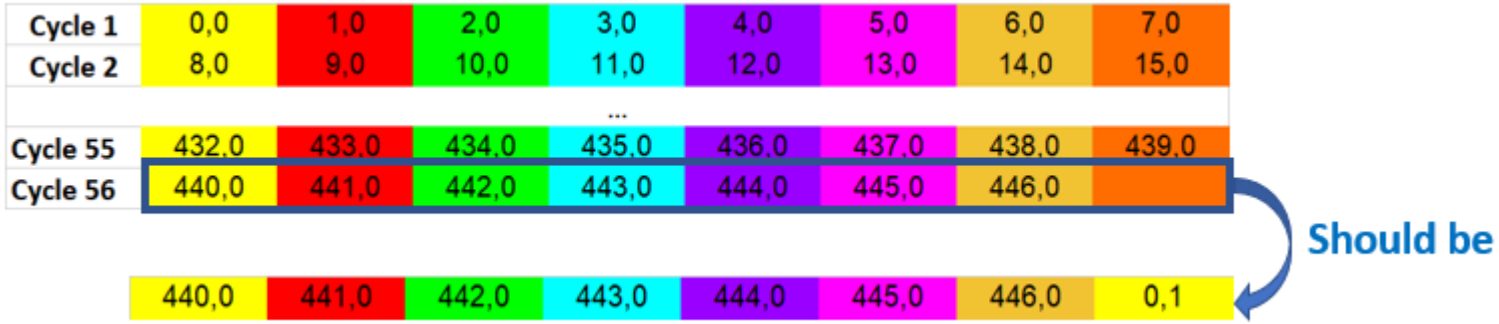
$$h(x) = x^8 + x^7 + x^5 + x^3 + 1$$

- The implementation is parallelized to allow the computation of all input bits concurrently
  - 6 XORs to apply the operation to all 6 bits, when required
  - The ROM stores the pseudo-random sequence and the shift register aligns its data based on the code rate
  - The multiplexer allows to bypass the module if the randomization is not desired!



Transmitter Chain – Channel Interleaver

- Writes in rows and reads in columns.
- **Approach**: Store each Interleaver row in a different memory position
  - 1 symbol from 8 consecutive rows are read concurrently –> Data must be stored in 8 memories!
- The **writing process** is simple: data is joined for 4 cycles in a register before being actually written in the correspondent memory
- The **reading process** is mostly straightforward: the same address position and bit within that position of all 8 memories is being read at the same clock cycle
  - However, there is an alignment issue at the end of each column as the number of rows is never divisible by 8 (447, 893, 1785 or 2231)
  - **Solution**: Use a valid signal for each of the 8 output bits (i.e., use a 8-bit valid signal) which is then handled by the Align Module



Transmitter Chain: ASM

- Developed jointly with the Interleaver module
- State machine for reading part has initial state for first outputting ASM
- Number of clock cycles to read data (ASM + encoded bits) must be equal or less than for writing. This is guaranteed as:
  - Parallelism of writing is 2/3/4/6
  - Parallelism of reading is always 8

Block Length	Code Rate	Number of clock cycles			
		Write	Read Bits	Read ASM	Read (Total)
1784	1/2	1788	448	8	**456**
	1/3		672	12	**684**
	1/4		896	16	**912**
	1/6		1344	24	**1368**
3568	1/2	3572	896	8	**904**
	1/3		1344	12	**1356**
	1/4		1792	16	**1808**
	1/6		2688	24	**2712**
7136	1/2	7140	1792	8	**1800**
	1/3		2688	12	**2700**
	1/4		3584	16	**3600**
	1/6		5376	24	**5400**
8920	1/2	8924	2232	8	**2240**
	1/3		3348	12	**3360**
	1/4		4464	16	**4480**
	1/6		6696	24	**6720**

Transmitter Chain: Align Module

- Solves the alignment issue when reading data from interleaver
- Ensures all or none outputted bits are valid in a given clock cycle:
  - If all 8-bits are valid, the valid output signal is asserted (‘1’)
  - Otherwise, it deasserts the valid output signal (‘0’) and stores the valid bits in a register
  - On the next clock cycle, the previous valid bits are joined with the current valid bits until forming again 8 valid bits
  - The spare valid bits are then joined with the ones on the following clock cycle and so on

		Input							
Cycle 1	Data	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
	Valid	1	1	1	1	1	1	1	1
Cycle 2	Data	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
	Valid	1	1	1	1	1	1	1	1
		...							
Cycle 55	Data	432,0	433,0	434,0	435,0	436,0	437,0	438,0	439,0
	Valid	1	1	1	1	1	1	1	1
Cycle 56	Data	440,0	441,0	442,0	443,0	444,0	445,0	446,0	
	Valid	1	1	1	1	1	1	1	0
Cycle 57	Data	0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1
	Valid	1	1	1	1	1	1	1	1
Cycle 58	Data	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1
	Valid	1	1	1	1	1	1	1	1

	FIFO Register															
Output valid	Output data															
1	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0								
1	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0								
1	432,0	433,0	434,0	435,0	436,0	437,0	438,0	439,0								
0	440,0	441,0	442,0	443,0	444,0	445,0	446,0									
1	440,0	441,0	442,0	443,0	444,0	445,0	446,0	0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	
1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1	

Transmitter Chain: Channel Emulator

- Adds digital noise to the incoming bits and outputs 8-bit soft symbols
- The noise (N) corresponds to an AWGN dependent on the chosen Es/No

$$N = AWGN * \sigma$$

$$\text{And, } \sigma = \sqrt{\frac{1}{2*10^{\frac{Es/No}{10}}}}$$

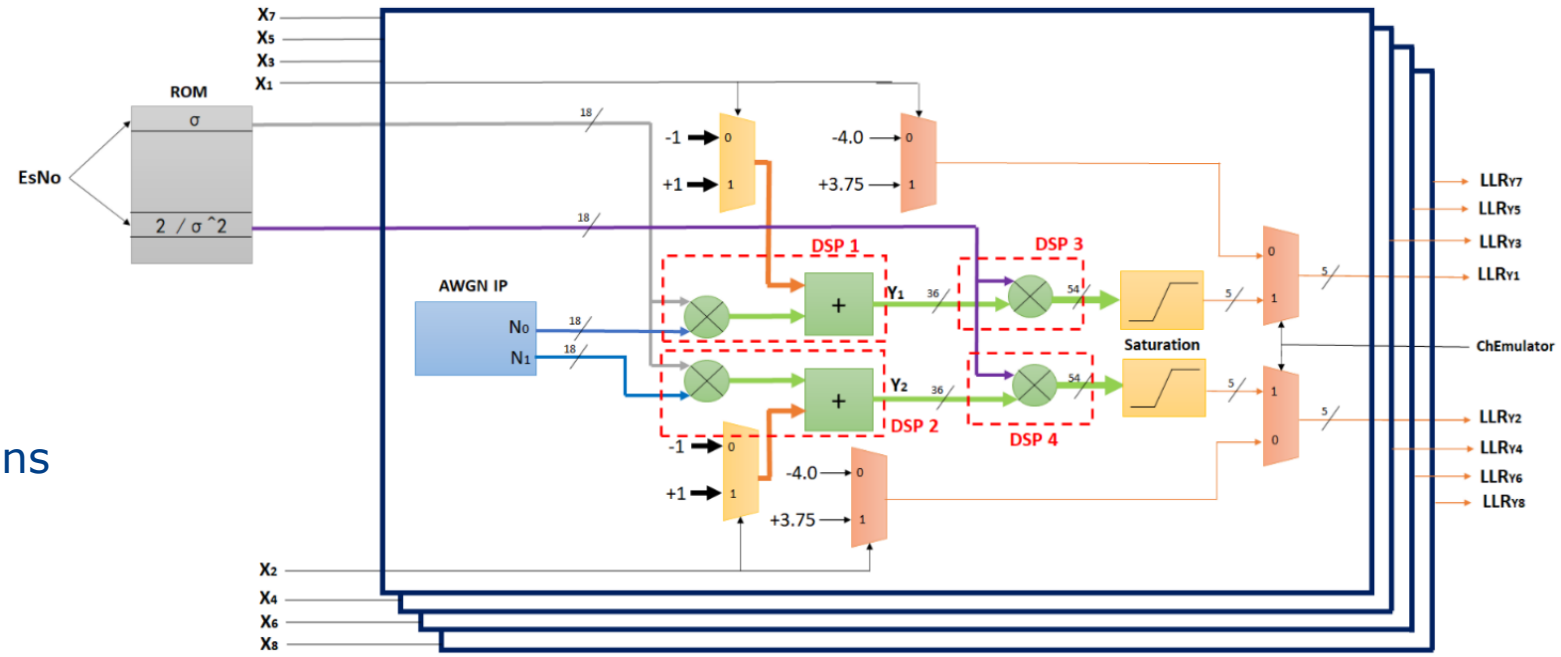
- Being X is the incoming bit, the output Y is:

$$Y = (2 * X - 1) + N = \begin{cases} -1 + N & if\ X = 0 \\ 1 + N & if\ X = 1 \end{cases}$$

- The receiver chain expects the LLR value of Y:

$$LLR_Y = Y * \frac{2}{\sigma^2}$$

- For each LLR value:
  - 2 multiplications and 1 addition
- The AWGN IP core generates two independent AWGN signals
  - LLRs are then calculated in pairs
- Parallelism level of 8 is required
  - Each pair is instantiated 4 times!
- The second term of both multiplications are pre-processed and stored in the same ROM



Proposed Frame Synchronizer Algorithm

- [AD-2] specifies different possible metrics for the Frame Synchronizer algorithm.
- The proposed Frame Synchronizer algorithm uses the approximated optimal Massey metric providing the best complexity-performance trade off (see [AD-2] for details).
- During the **sync acquisition phase**, the metrics of all candidate positions are accumulated at each frame. Acquisition is declared when the difference between the highest metric and second one exceeds a threshold computed as

$$T_h = S\sqrt{n} \cdot 2\sqrt{\frac{M}{\sigma^2}}\,\mathrm{erfc}^{-1}\left(\frac{2 \cdot 10^{-1}}{F}\right)$$

- Where n is the number of accumulated metrics (received frames), M the ASM length F the total frame length and σ^2 the noise variance.
- During the **sync tracking phase**, the running average of the last P metrics is updated for the best position (estimated sync) and the two adjacent ones.
- A sync loss is declared if the metric relative to the estimated sync is worse than one of the other two.
- Parameters P and S of the adopted procedure are designed to guarantee a desired maximal False Lock Probability (FLP) and False Sync Loss Probability (FSLP) and a high True Sync Loss Probability (TSLP) when insertion or deletions occurs.
- Conclusions on the design of optimal parameters P and S of the adopted algorithm are provided in the next slide.

Design of parameters *S* and *P*

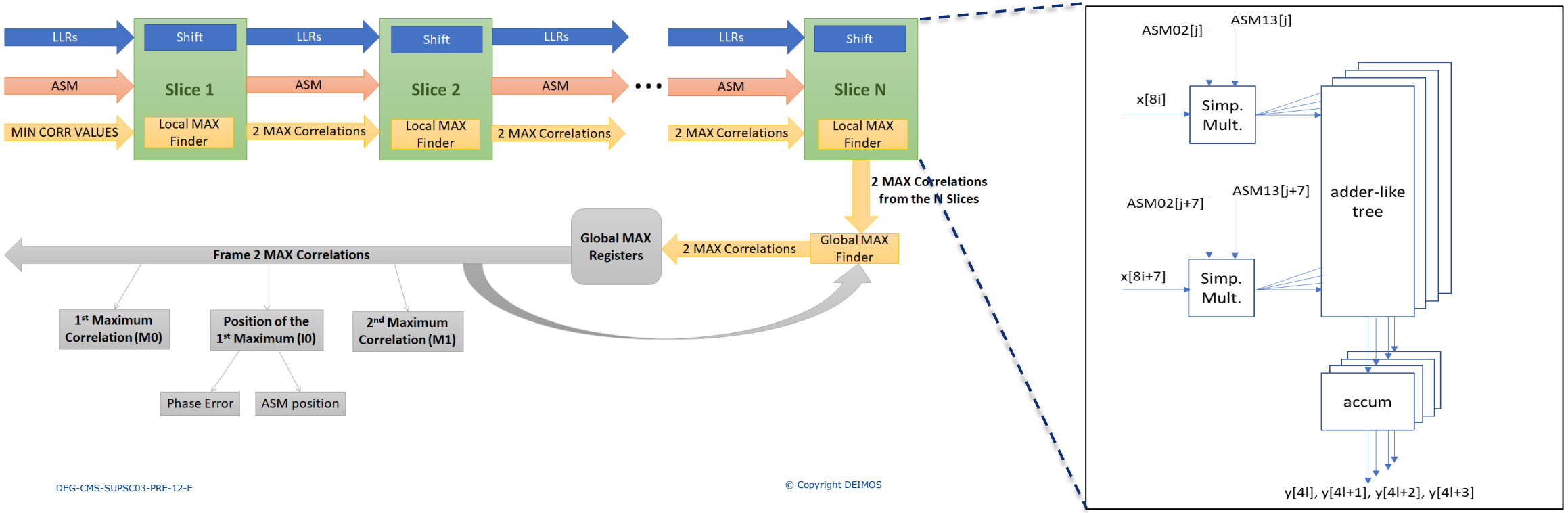
- We performed extensive simulation campaign to design the optimal values for the parameters *P* and S
  - Results reported in TN3
- In the following we report the conclusions
- **S=0.4**, guarantees a false lock probability below 1e-8, and an average lock time below one frame at the threshold of the code
  - While in principle the correspondent threshold T_h depends on rate, code length and SNR, we propose to use a different value for each rate and size, while fixing the SNR to that corresponding to the code thresholds. This yields a table of 16 entries for T_h (x.3 fixed point representation of LLR)

	1784	3568	7136	8920
1/2	12.89	13.40	13.90	14.05
1/3	10.11	10.50	10.87	10.99
1/4	8.63	8.94	9.25	9.35
1/6	6.98	7.23	7.47	7.55

- **P=2** guarantees an average false sync loss probability below 1e-8 at the SNR corresponding to the code threshold and an average of two frame for reacting to a true sync loss

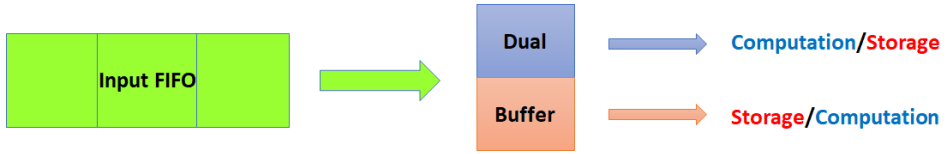
Frame Synchronizer: Acquisition

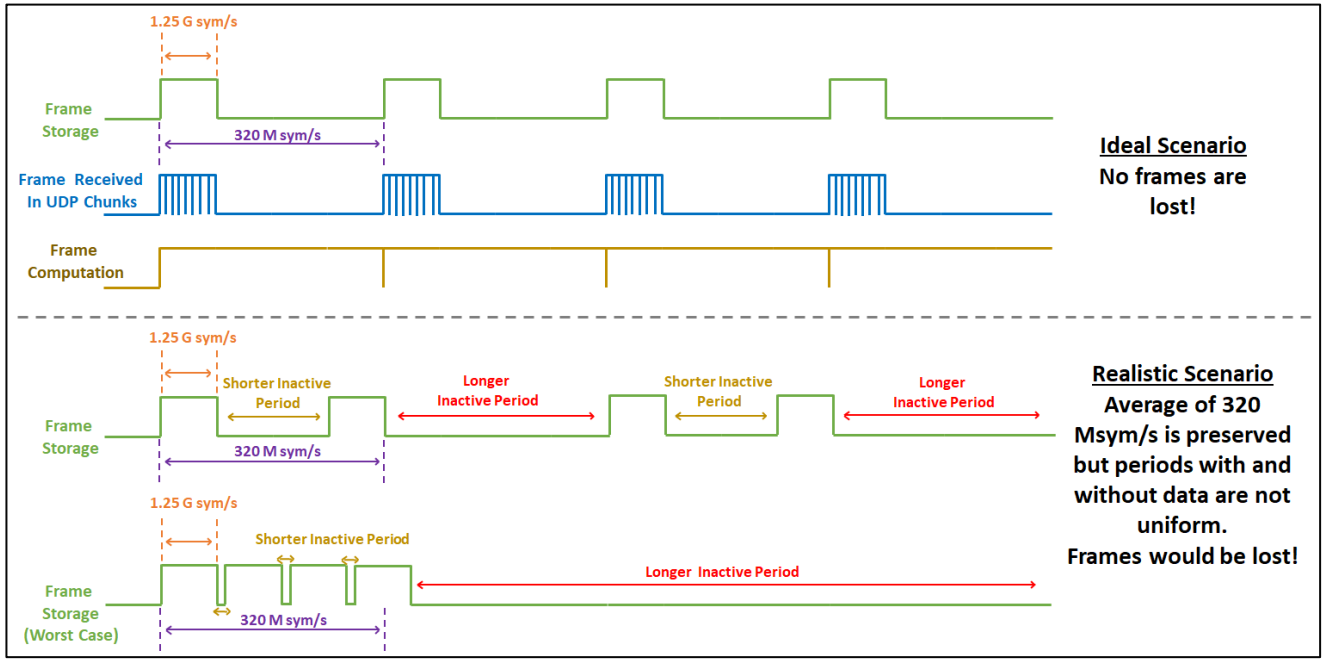
- The frame synchronization architecture is divided in two parts: acquisition and tracking
  - The goal of the acquisition phase is to obtain the two maximum correlation results (out of the total number of correlation results of a given frame) and to check if their difference is higher than the threshold
  - 4 correlations windows (one for each ASM pattern) are computed for each position in the symbol frame
  - Many slices are cascaded in a pipeline fashion so that several correlations can be performed in parallel



Frame Synchronizer: Input FIFO

- To match the storage and computation time (for acquisition) of the frame synchronizer:
  - 192 slices would be needed, however, there are enough resources to allocate that many
- For the acquisition phase, the correlations results need to be accumulated
  - Decided to use 64 slices (good trade-off between the memory requirements and the acceleration of the computation of the frame acquisition phase at the expense of consumption of ALMs)
- To avoid loosing frames, an input FIFO is required:
  - Stores a maximum of 3 of the biggest symbol frames

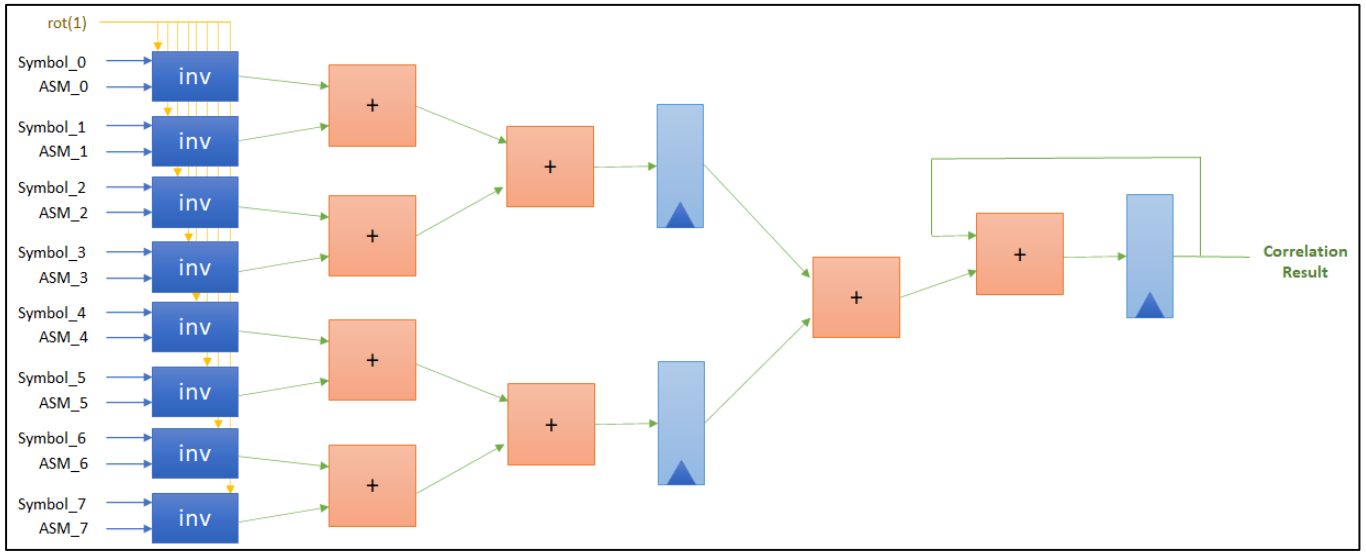


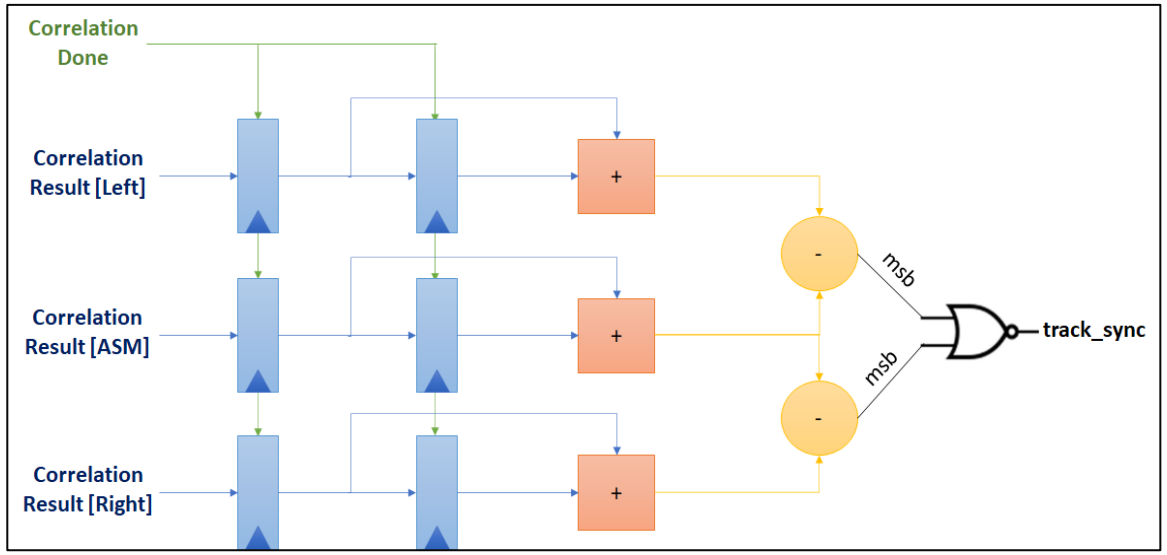


# Slices (N)	M20Ks per slice	Total M20Ks	# Slices (N)	M20Ks per slice	Total M20Ks
8	14	112	104	2	208
16	7	112	112	2	224
24	5	120	120	2	240
32	4	128	128	2	256
40	3	120	136	2	272
48	3	144	144	2	288
56	3	168	152	2	304
64	2	128	160	2	320
72	2	144	168	2	336
80	2	160	176	2	352
88	2	176	184	2	368
96	2	192	192	1	192

Frame Synchronizer: Tracking

- In the tracking phase, 3 correlations are calculated (for the ASM position and its right and left adjacent)
  - Only the correlations results from the last 2 frames need to be accumulated
- New slice architecture to calculate correlations for tracking:
  - Keep level of parallelism of 8 and 2 pipeline stages
  - Each slice requires only about 55 ALMs => 3 slices are required in total!
- The synchronization is kept as long as the 2-frame accumulated correlations for the ASM position is greater than both adjacent ones

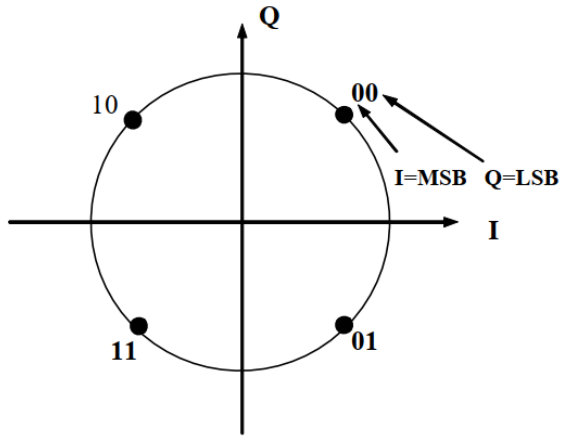


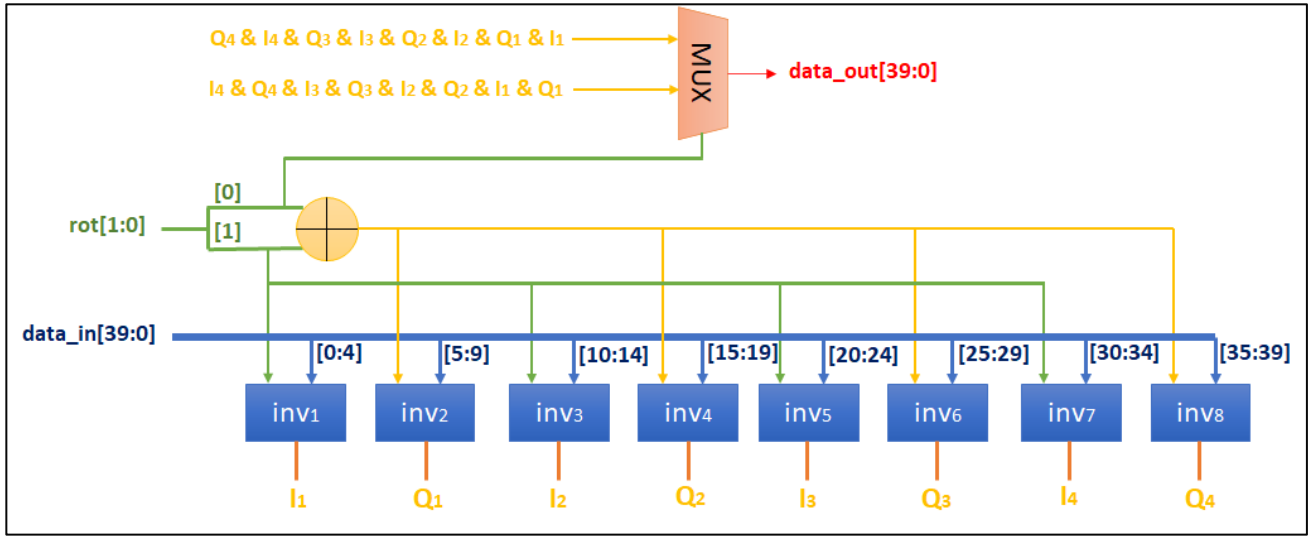


Receiver Chain: Phase Error Correction

- Corrects the phase errors detected by the frame synchronizer in the input symbols
  - Receives 2-bit "rot" signal from Synchronizer
- The constellation uses 2 bits per symbol:
  - If phase error is 0º -> I and Q in phase
  - If phase error is 90º -> $I_R = -Q_T$ and $Q_R = I_T$
- Implementation design includes:
  - 8 signal inverters to compute 8 symbols in parallel
  - Multiplexer to choose order of output "IQ" samples
  - Inverter and MUX are controlled by "rot" signal

Carrier phase error [º]	Received data	
	I_R	Q_R
0	I_T	Q_T
90	$-Q_T$	I_T
180	$-I_T$	$-Q_T$
270	Q_T	$-I_T$





Receiver Chain: De-Align Module

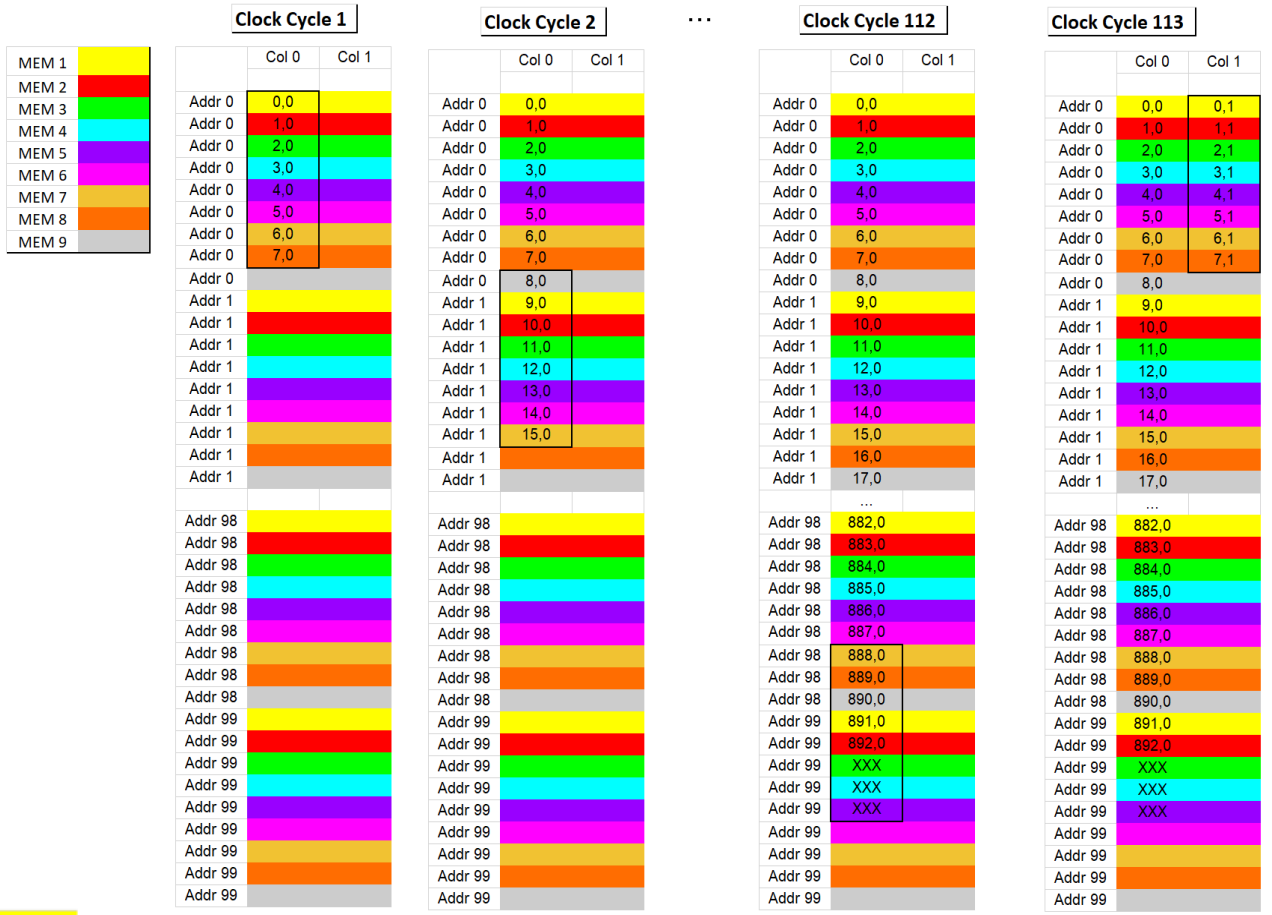
- Goal: data to be stored in the same order as it is stored in the Interleaver;
- The last data of one column can be received alongside the initial data of the next column
  - Jeopardizes the integrity of the order of the data stored in the de-interleaver memories;
  - Solution: last data from one column is sent in the current clock cycle, storing the remaining data in a FIFO alike register.
  - In the next clock cycle, the previous stored data is joined with the data now being received until forming again an 8-symbol bus;
  - The remaining symbols are kept in the register to be sent in the next clock cycle and so on.

	Input data							
Cycle 1	0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0
Cycle 2	8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0
				...				
Cycle 55	432,0	433,0	434,0	435,0	436,0	437,0	438,0	439,0
Cycle 56	440,0	441,0	442,0	443,0	444,0	445,0	446,0	0,1
Cycle 57	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1
Cycle 58	9,1	10,1	11,1	12,1	13,1	14,1	15,1	16,1

FIFO Register								
Output data								
0,0	1,0	2,0	3,0	4,0	5,0	6,0	7,0	
8,0	9,0	10,0	11,0	12,0	13,0	14,0	15,0	
				...				
432,0	433,0	434,0	435,0	436,0	437,0	438,0	439,0	
440,0	441,0	442,0	443,0	444,0	445,0	446,0	0,1	
0,1	1,1	2,1	3,1	4,1	5,1	6,1	7,1	8,1
8,1	9,1	10,1	11,1	12,1	13,1	14,1	15,1	16,1

Receiver Chain: De-Interleaver

- Works in the opposite way of the Interleaver, i.e., writes columns and reads rows
- **Same Approach**: Store each De-Interleaver row in a different memory position
  - **Advantage:** data comes already well-aligned
  - At least 8 memories are needed to write in 8 different rows in the same clock cycle –> 9 memories used to save M20Ks
- The **writing process** is more complex as there are 9 memories: there is one shift register than controls the write enable of each memory as well as the address counter of each memory
- The **reading process** is simple for the code rates of 1/2, 1/4 and 1/6 where only one memory is read per clock cycle. For 1/3, from time to time 2 memories are read on the same clock cycle:



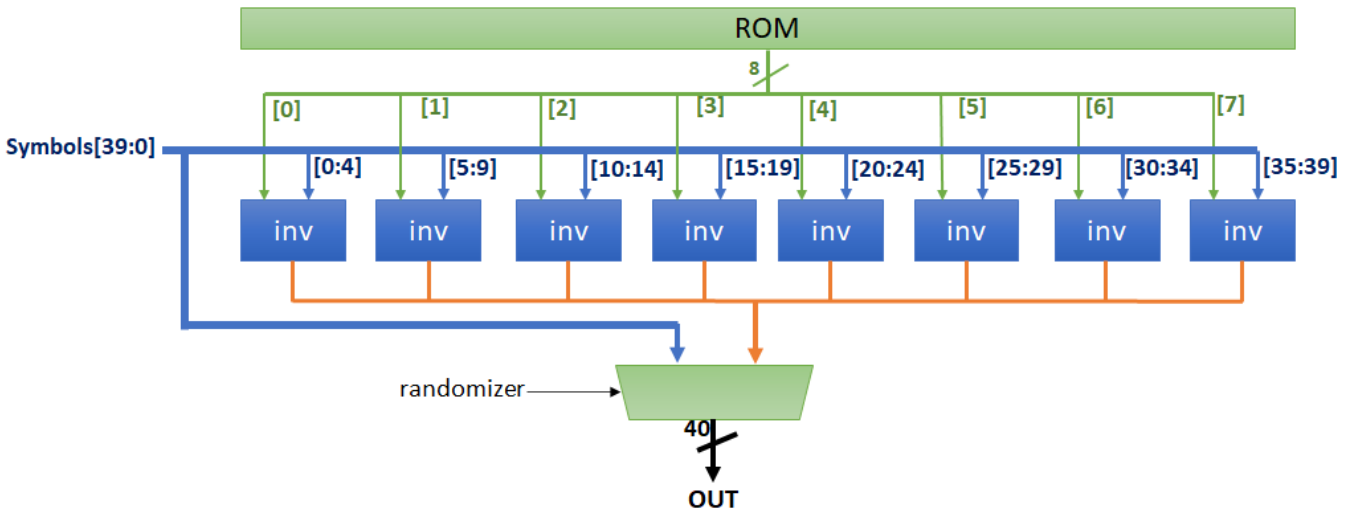
MEM 1	Cycle 1	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7
MEM 2	Cycle 2	0,8	0,9	0,10	0,11	1,0	1,1	1,2	1,3
MEM 3	Cycle 3	1,4	1,5	1,6	1,7	1,8	1,9	1,10	1,11
MEM 4	Cycle 4	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7
	Cycle 5	2,8	2,9	2,10	2,11	3,0	3,1	3,2	3,3
	Cycle 6	3,4	3,5	3,6	3,7	3,8	3,9	3,10	3,11

Clock Cycle	Col cnt	Row cnt	Left Shift register	Addr counter								
				MEM1	MEM2	MEM3	MEM4	MEM5	MEM6	MEM7	MEM8	MEM9
1	0	0	111111110	0	0	0	0	0	0	0	0	0
2	0	1	111111101	1	1	1	1	1	1	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...
112	0	111	111110111	99	99	99	99	99	99	98	98	98
113	1	0	111111110	0	0	0	0	0	0	0	0	0

Receiver Chain: De-Randomizer

- Uses the same 255-bit pseudo-random sequence as the Randomizer. Two main differences:
  - Single level of parallelism of 8 -> ROM is configured as 255x8 (1 M20K)
  - Performs signal inversion operation rather than XOR operation
- The implementation is quite straightforward:
  - 8 signal inversion modules to ensure the proper parallelism
  - Output multiplexer that allows to bypass this module if randomization is not desired

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Addr 0
...								
X_{248}	X_{249}	X_{250}	X_{251}	X_{252}	X_{253}	X_{254}	X_0	Addr 31
...								
X_{249}	X_{250}	X_{251}	X_{252}	X_{253}	X_{254}	X_0	X_1	Addr 63
...								
X_{250}	X_{251}	X_{252}	X_{253}	X_{254}	X_0	X_1	X_2	Addr 95
...								
X_{251}	X_{252}	X_{253}	X_{254}	X_0	X_1	X_2	X_3	Addr 127
...								
X_{252}	X_{253}	X_{254}	X_0	X_1	X_2	X_3	X_4	Addr 159
...								
X_{253}	X_{254}	X_0	X_1	X_2	X_3	X_4	X_5	Addr 191
...								
X_{254}	X_0	X_1	X_2	X_3	X_4	X_5	X_6	Addr 223
...								
X_{247}	X_{248}	X_{249}	X_{250}	X_{251}	X_{252}	X_{253}	X_{254}	Addr 254



Receiver Chain: Turbo Decoder

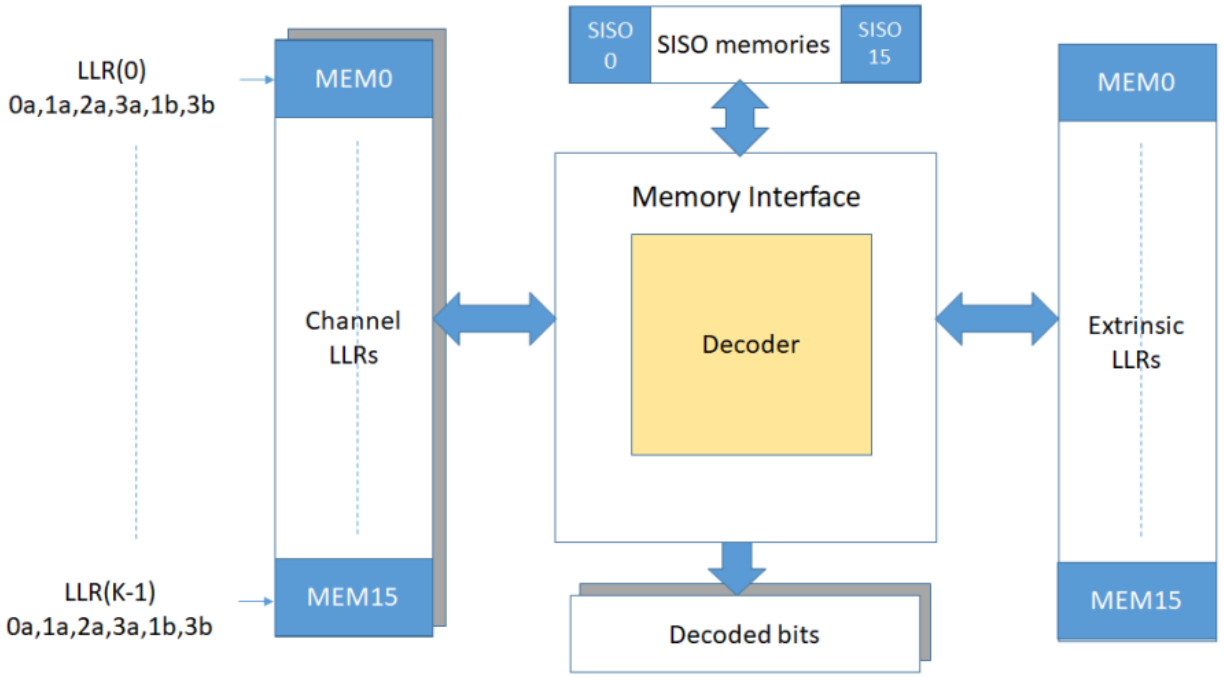
- Most complex module of the design
  - Receives the codewords and outputs the Decoded Transfer Frames
- Achievable throughputs:

$$T_{1784} = 4 \cdot \frac{1784 \cdot 156.25 \cdot 10^6}{21 \cdot (56 \cdot 8 + 56 + 22) + 23} = 100.73\ Mb/s$$

$$T_{3568} = 2 \cdot \frac{3568 \cdot 156.25 \cdot 10^6}{21 \cdot (56 \cdot 8 + 56 + 22) + 23} = 100.73\ Mb/s$$

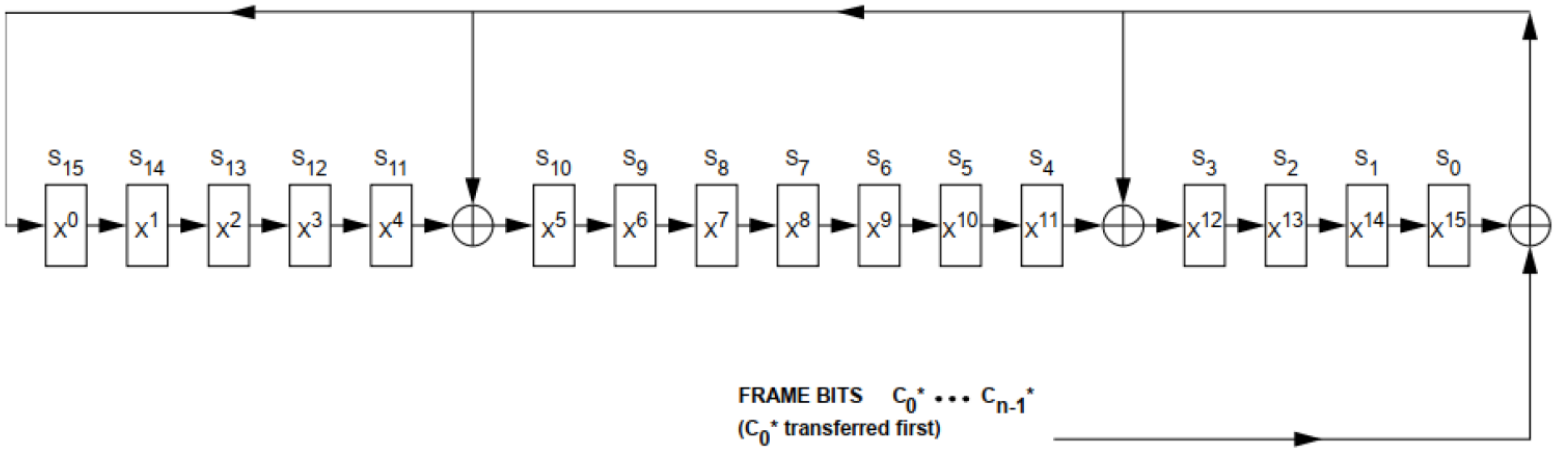
$$T_{7136} = 1 \cdot \frac{7136 \cdot 156.25 \cdot 10^6}{21 \cdot (56 \cdot 8 + 56 + 22) + 23} = 100.73\ Mb/s$$

$$T_{8920} = 1 \cdot \frac{8920 \cdot 156.25 \cdot 10^6}{21 \cdot (56 \cdot 10 + 56 + 22) + 23} = 103.85\ Mb/s$$



Receiver Chain: FECF Validation

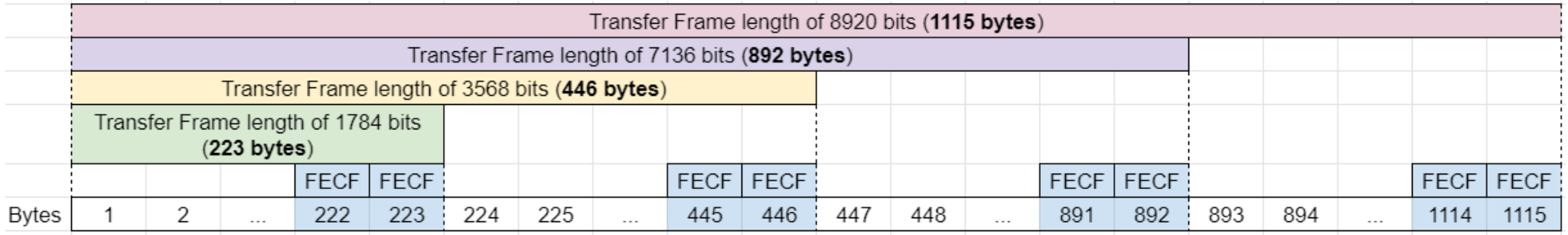
- The FECF validation is performed at the end of the Receiver Chain
  - The registers (S_{15} to S_0) are updated for each bit injected from the Decoded Transfer Frame
  - After injecting the last bit, the value of the registers will be zero if no errors are detected.
- By default, the diagram receives 1 bit and performs 3 XOR operations per clock cycle
  - Design was parallelized to receive 1 byte and perform 24 XOR operations per clock cycle!



$XOR_1 = C_0 \oplus S_0$	$XOR_5 = XOR_1 \oplus S_4$	$XOR_9 = XOR_1 \oplus S_{11}$
$XOR_2 = C_1 \oplus S_1$	$XOR_6 = XOR_2 \oplus S_5$	$XOR_{10} = XOR_2 \oplus S_{12}$
$XOR_3 = C_2 \oplus S_2$	$XOR_7 = XOR_3 \oplus S_6$	$XOR_{11} = XOR_3 \oplus S_{13}$
$XOR_4 = C_3 \oplus S_3$	$XOR_8 = XOR_4 \oplus S_7$	$XOR_{12} = XOR_4 \oplus S_{14}$
$XOR_{13} = C_4 \oplus XOR_5$	$XOR_{17} = XOR_{13} \oplus S_8$	$XOR_{21} = XOR_{13} \oplus S_{15}$
$XOR_{14} = C_5 \oplus XOR_6$	$XOR_{18} = XOR_{14} \oplus S_9$	$XOR_{22} = XOR_{14} \oplus XOR_1$
$XOR_{15} = C_6 \oplus XOR_7$	$XOR_{19} = XOR_{15} \oplus S_{10}$	$XOR_{23} = XOR_{15} \oplus XOR_2$
$XOR_{16} = C_7 \oplus XOR_8$	$XOR_{20} = XOR_{16} \oplus XOR_9$	$XOR_{24} = XOR_{16} \oplus XOR_3$

Auxiliary Modules

- **Configuration module:** receives the parameters from the Testing Tool and stores them in registers which are accessed by all the other modules in the design
- **Monitoring module:** receives the status parameters from the other modules in the FPGA and provides periodically the status of the breadboard to the external computer via the M&C interface
- **Test data:** allows to test internally both Transmitter and Receiver Chains of the breadboard by making use of predefined sequences of data stored in local memory without using the external 10Gb interfaces. At the end of the Receiver Chain is verified if the decoded sequence is equal to the transmitted data
  - The same data pattern can be sent several times to the transmission chain, which then inserts randomness via the AWGN added in the channel emulator
  - Decided to use only one 1 M20K to save resources -> two different test vectors are stored



FPGA Final Resources

- Despite the available resources at ALTERA Stratix V for the VIRTUDE project are limited, the final FPGA implementation shows that less than 80% of the available ALMs and M20Ks are used:
  - About 80% for the ALMs;
  - About 79% for the M20Ks;
  - About 7% for the DSPs.
- All breadboard modules will be implemented using only FPGA resources:
  - Previous possibility of using external memory for the Interleaver/De-Interleaver to release M20K blocks was not needed!

Module	ALMs	M20Ks	DSPs
Transmitter chain			
Turbo Encoder	121	1	0
Randomizer	64	0	0
Interleaver (with ASM)	402	16	0
Align Module	36	0	0
Channel Emulator	4 864	9	48
Receiver chain			
Input FIFO	1 694	40	0
Frame Synchronizer	43 236	169	0
Correct Phase Error	62	0	0
De-Align Module	188	0	0
De-Interleaver	1 423	27	0
De-Randomizer	47	1	0
Turbo Decoder	60 832	273	0
Computation of FECF	82	0	0
Auxiliary modules			
Configuration	133	0	0
Monitoring	185	0	0
Test Data	209	1	0
External Interfaces			
10 Gb input interface	3 178	21	0
10 Gb output interface	3 252	15	0
Output protocol fields	832	3	4
1 Gb M&C interface	2 004	12	0
External Ips			
10 Gb PHY	27	0	0
1Gb PHY	439	0	0
Total resources	**123 310**	**588**	**52**
Available for VIRTUDE	**154 815**	**749**	**828**
% of used resources / available resources.	**79.65**	**78.50**	**6.28**

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

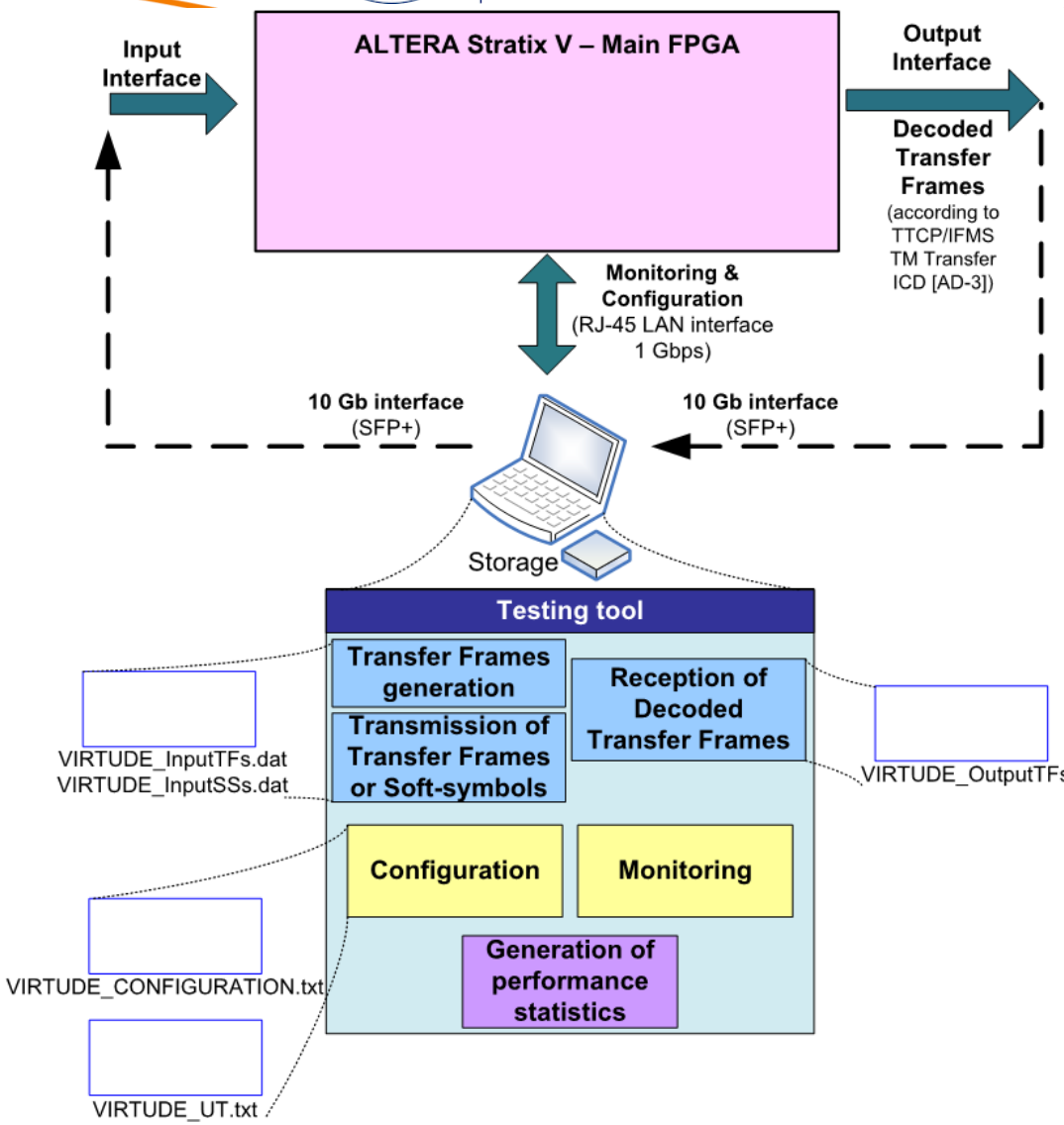
15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

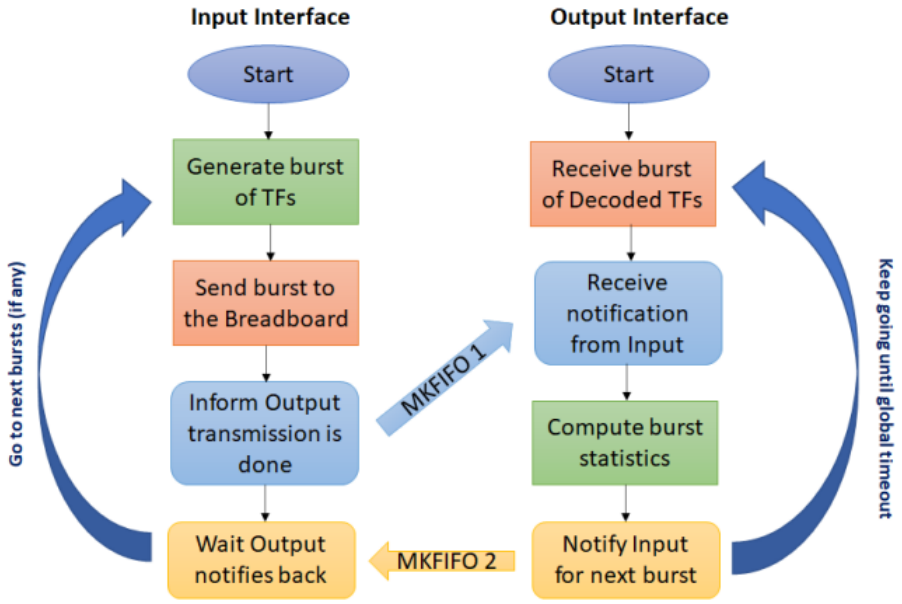
Testing Tool: Features

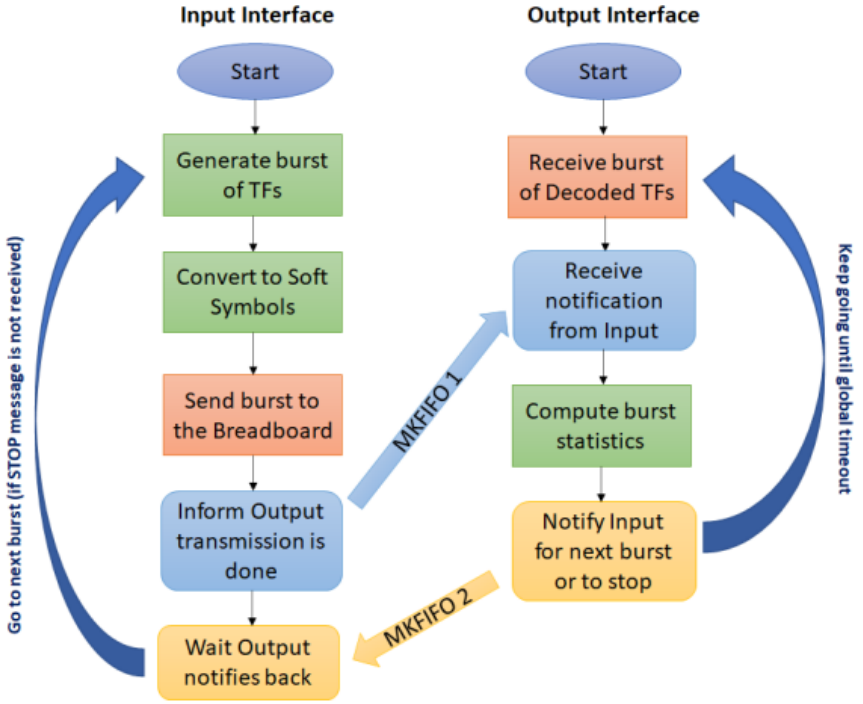
- This tool has been developed in C++ language without a Graphical User Interface (GUI):
  - 1 executable file is used for all functionalities of the app (different arguments run different functionalities);
  - Command line provides results/info in real-time;
  - Text files are used to provide the configurations for the breadboard and also the configuration for the Unit Test when intended.
- Configures the breadboard;
- Unit tests the breadboard using its internal test vectors;
- Generates random transfer frames and soft-symbols and injects them to the transmitter chain or receiver chain, respectively;
- Receives the decoded transfer frames sent by the Turbo Decoder;
- Saves generated and received frame data on data files;
- Evaluate the all-breadboard performance: throughput, BER and FER



Testing Tool: Software Pre-requisites

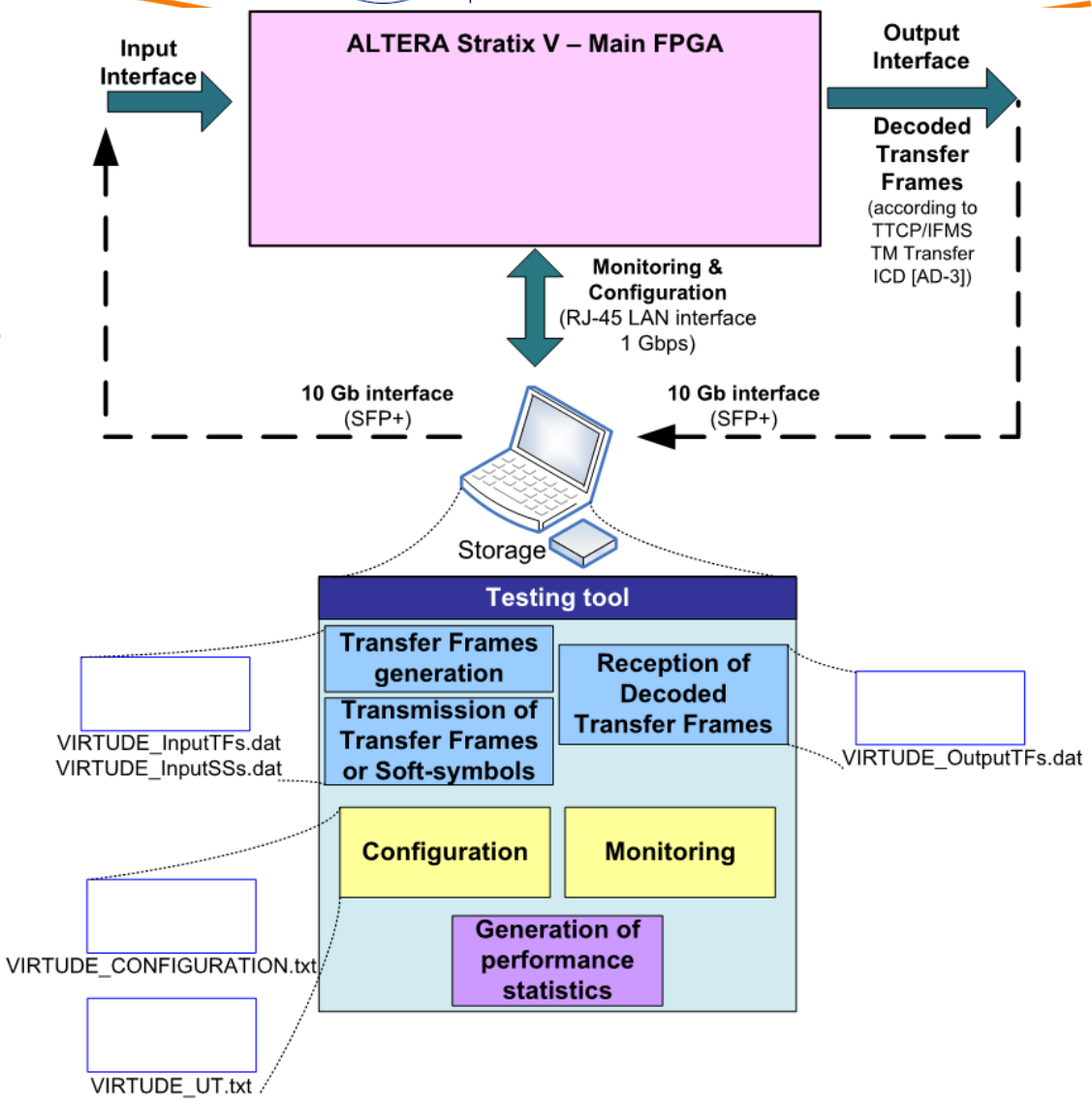
- The Testing Tool was developed and tested in a Debian GNU/Linux 9 (Stretch) machine
  - Other Linux distributions such as Ubuntu should be compatible
- The generation of Transfer Frames and Soft-Symbols is slow in the Testing Tool
  - To overcome this limitation, data is injected at the target bit rate in bursts of 1000 frames
  - Synchronization is required between the Input and Output Interfaces via ***mkfifos***
- The **wondershaper** Linux tool is used to limit the speed of input interface to achieve the configured bit rate
  - Otherwise, frames are injected at 10 Gbps and frames are lost!



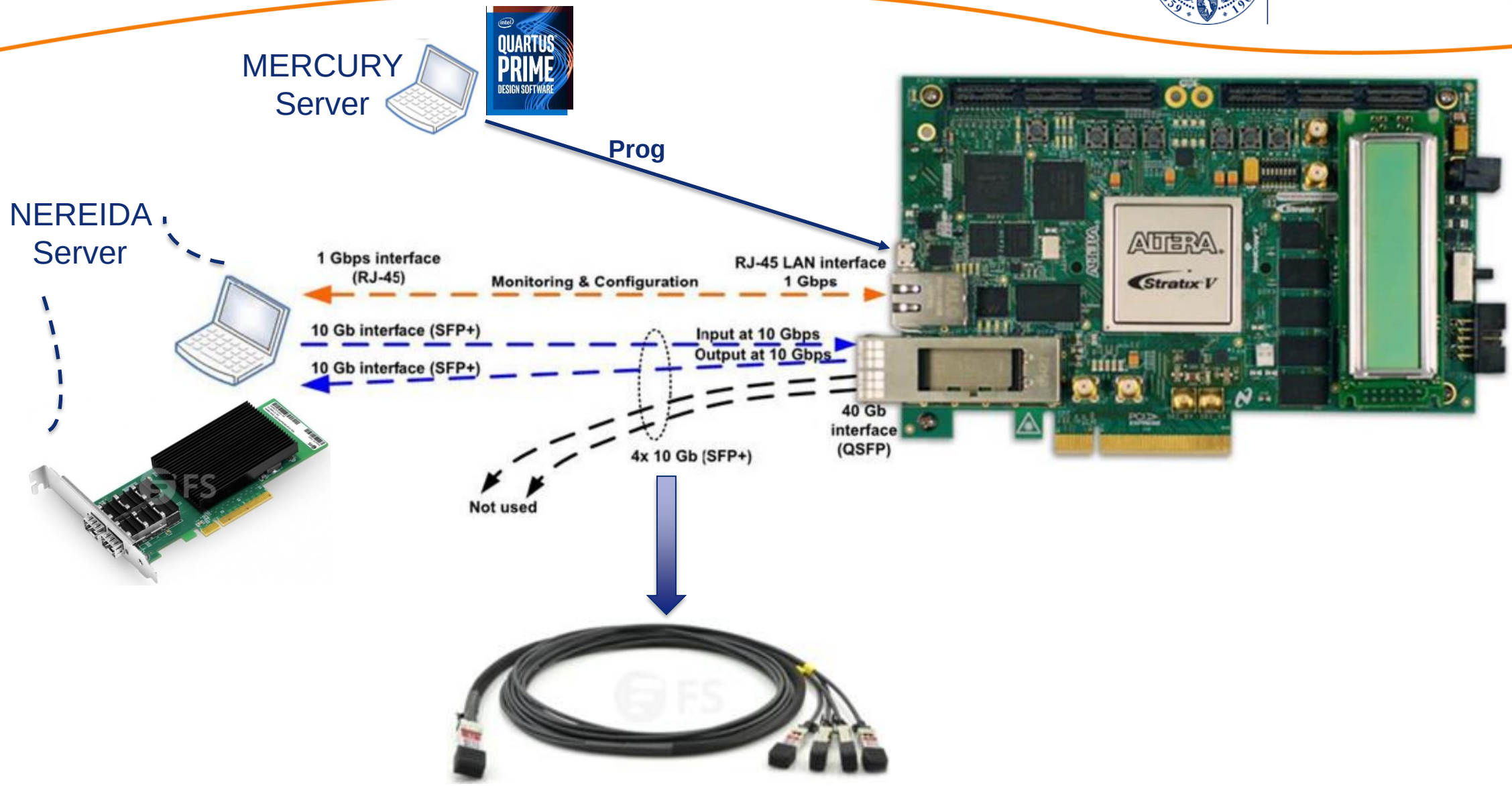


Testing Tool: Application

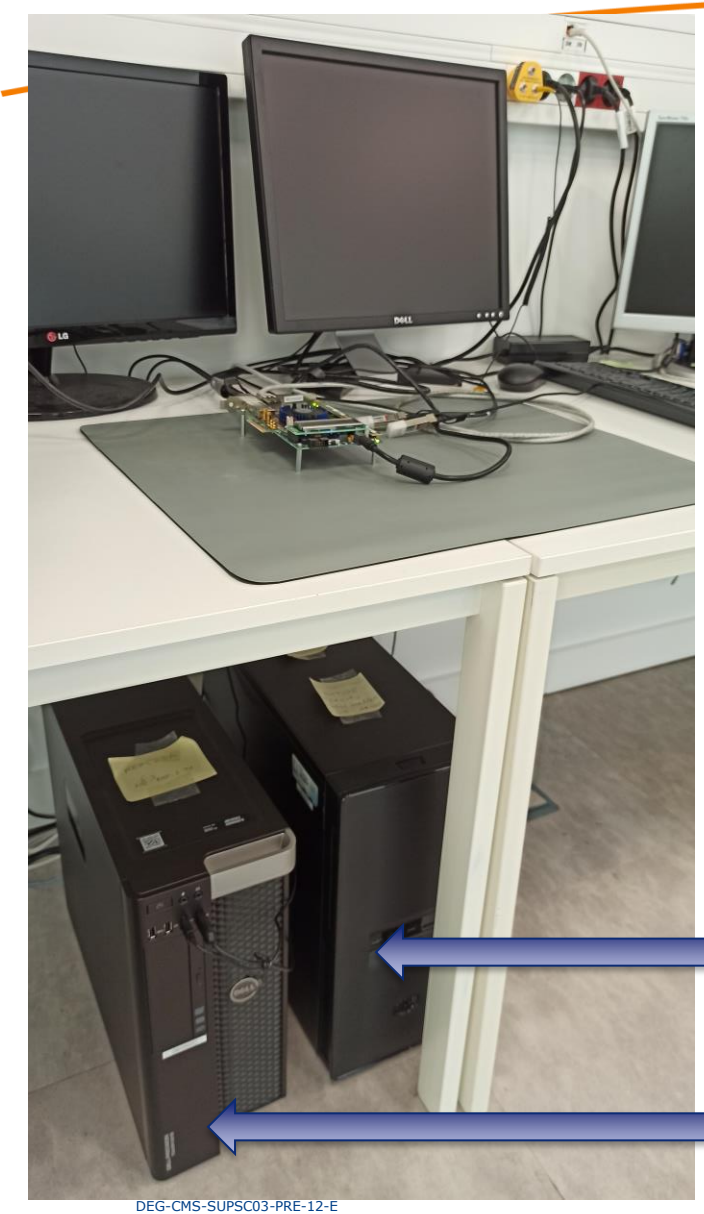
- The application can be launched through the following executable:
  - *VIRTUDE_TT.exe* -> requires an argument to select which functionality to use
- Each of the arguments launches a software with one of the following threads:
  - Interact with the 1 Gb M&C interface to handle the configuration messages (*conf*);
  - Interact with the 1 Gb M&C interface to handle the monitoring messages (*mon*);
  - Interact with the 1 Gb M&C interface to handle the unit test messages (*ut*);
  - Interact with the 10 Gb Input interface to inject either Transfer Frames or Soft Symbols (*in*);
  - Interact with the 10 Gb Output interface to receive either Decoded Transfer Frames or Heartbeat Messages (*out*).



VIRTUDE Setup (1/2)



VIRTUDE Setup (2/2)



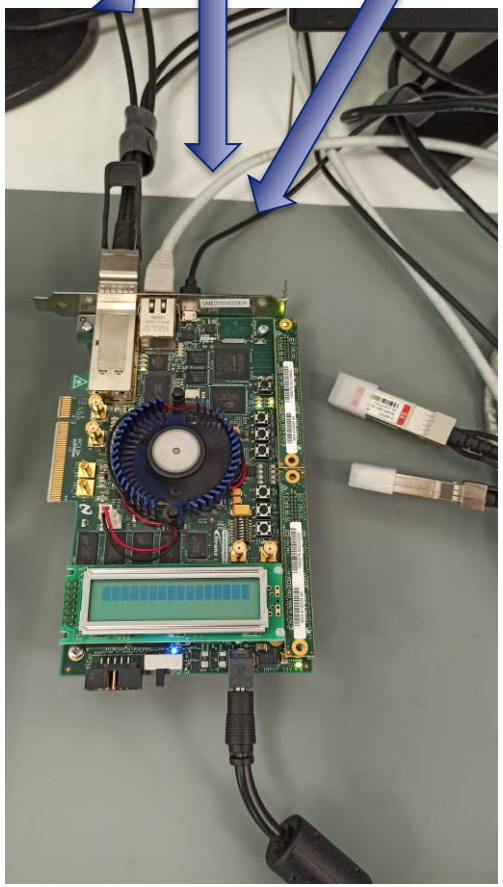
MERCURY Server

NEREIDA Server

10 Gb Input
10 Gb Output

1 Gb M&C

Prog



1 Gb M&C

10 Gb Input
10 Gb Output

Testing Tool: Procedure (1/2)

1) Edit the configuration file "*VIRTUDE_Configurations.txt*" with the desired configurations

The Tool includes example configurations files for testing Transfer Frames, Soft-Symbols, Heartbeat Messages, BPSK modulation, etc.

Any of this file can be renamed to "*VIRTUDE_Configurations.txt*" to be use in the Testing Tool

2) Run "*Virtude_TT conf*" to send a new configuration to the Breadboard

In this example, it is configured to send Soft-Symbols generated with the maximum Es/No

The monitoring messages are also enabled

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT conf
Arguments obtained:
conf


Client will try to reach server on IP address 192.168.1.13 and port number 21006
Configurations selected:
Message length: 58
Message type: 1
Operating Mode: 1
Input Type: 1
Bit Rate: 80000000
Code Rate: 2
TF length: 0
Randomizer: 1
interleaver: 1
emulator: 0
Es/No: 140
Monitor period: 1000
Heartbeat period: 0
Input Remote IP: 192.168.2.1
Input Local IP: 192.168.2.100
Input Local Port Number: 21001
Output Local IP: 192.168.3.101
Output Local Port Number: 21003
Output Remote IP: 192.168.3.2
Output Remote Port Number: 21004
Monitor Local IP: 192.168.1.13
Monitor Local Port Number: 21006
Monitor Remote IP: 192.168.1.102
Monitor Remote Port Number: 21005
FPGA MAC: 0:7:237:29:20:69
Modulation Type: 0

Created server on IP address 192.168.1.102 and port number 21005

* Configuration message sent *

Response received: 0x3a 0x11 0x01 0x01 0x04 0xc4 0xb4 0x00 0x02 0x00 0x01 0x01 0
x00 0x8c 0x03 0xe8 0x00 0x00 0xc0 0xa8 0x02 0x01 0xc0 0xa8 0x02 0x64 0x52 0x09 0
xc0 0xa8 0x03 0x65 0x52 0x0b 0xc0 0xa8 0x03 0x02 0x52 0x0c 0xc0 0xa8 0x01 0x0d 0
x52 0x0e 0xc0 0xa8 0x01 0x66 0x52 0x0d 0x00 0x07 0xed 0x1d 0x14 0x45 0x00

* Breadboard confirms new configurations *

Configuration Interface closing...
dcpp@nereida:~/virtude/TestingTools/bin$
```

Testing Tool: Procedure (2/2)

3) Run "*VIRTUDE_TT mon*" to open the monitoring interface (in case it is enabled). The status parameters of the number of UDP frames received and of decoded Transfer Frame increase after the input interface is executed

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT mon
Arguments obtained:
mon


Reading configurations from file...
Created server on IP address 192.168.1.102 and port number 21005
-------------------------------------------------
Timestamp since last configuration: 1000 ms
Number of received frames with wrong CRC: 0
Number of received frames with wrong length: 0
Latency of the turbo decoder: 70841 ns
Number of Transfer Frames received since last configuration: 0
Number of UDP frames received since last configuration: 0
Number of decoded Transfer Frames since last configuration: 0
Number of decoded Transfer Frames with errors since last configuration: 0
-------------------------------------------------
```

⋮

```
-------------------------------------------------
Timestamp since last configuration: 6000 ms
Number of received frames with wrong CRC: 0
Number of received frames with wrong length: 0
Latency of the turbo decoder: 70841 ns
Number of Transfer Frames received since last configuration: 0
Number of UDP frames received since last configuration: 891
Number of decoded Transfer Frames since last configuration: 1000
Number of decoded Transfer Frames with errors since last configuration: 0
-------------------------------------------------
```

4) Run "*VIRTUDE_TT out*" to open the output interface. Data starts being received after input interface is run

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT out
Arguments obtained:
out


Save received decoded Transfer Frames: Yes
Reading configurations from file...
Server created on IP 192.168.3.2 and port number 21004
Waiting for incoming messages...
```

⋮

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT out
Arguments obtained:
out


Save received decoded Transfer Frames: Yes
Reading configurations from file...
Server created on IP 192.168.3.2 and port number 21004
Waiting for incoming messages...
--> First Frame after acquisition! (lost 0 frames before)
Received 1 notification from Input Interface
Saving 303000 bytes on the file, which correspond to 1000 TFs.
Getting input TF from file...
***************
* BER: 0        *
* FER: 0        *
***************
cummulative average BER: 0
cummulative average FER: 0

time between first decoded Transfer Frame and last received: 22139.7us
throughput (useful output bit rate) calculated (223000 bytes): 80.5791 Mbps
cummulative average throughput (useful output bit rate): 80.5791 Mbps
DLC value of last decoded Transfer Frame: 0

Notified Input Interface for the 1 time
-------------------

*** Final Stats ***
cummulative average BER: 0
cummulative average FER: 0
cummulative average throughput (useful output bit rate): 80.5791 Mbps

Average lock time (number of frames lost per acquisition): 0
Total number of frames received: 1000

Output Interface closing...
dcpp@nereida:~/virtude/TestingTools/bin$
```

5) Run "*VIRTUDE_TT in –n 1000*" to send 1000 Soft-Symbols via the input interface

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT in -n 1000
Arguments obtained:
in
-n
1000


nTFs: 1000
delay: 0
rotation: 0
tumbling_freq: 0
amplitude: 7
nConfigs: 1
Clearing Input data files...
Reading configurations from file...
mkfifos created
MKFIFO_1 configured
MKFIFO_2 configured
Soft-Symbol generation selected.
Generating Transfer Frames...
Generating Soft Symbols from TFs...
Data (7294256 bytes/elements) was saved on ../resources/VIRTUDE_InputSSs.dat
1000 Soft-Symbol frames were generated

time between first UDP frame (carrying SSs) and last sent: 22204.1us
throughput calculated (7280000 bytes): 2622.93 Mbps
useful input bit rate: 81.9667 Mbps
Notified Output Interface for the 1 time
Received 1 notification from Output Interface
dcpp@nereida:~/virtude/TestingTools/bin$
```

Testing Tool: Unit Test

1) Edit the configuration file "*VIRTUDE_UT.txt*" with the desired unit test configurations

2) Run "*Virtude_TT ut*" to send a UT request and to wait until receiving the UT response

When there are no decoding errors (high Es/No), the Tool shows the message "PASSED"

When there are decoding errors (low Es/No), the Tool show the message "FAILED" and reports the BER/FER metrics

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT ut
Arguments obtained:
ut


Configurations selected:
Message length: 18
Message type: 2
Operating Mode: 2
Memory Index: 0
Bit Rate: 80000000
Code Rate: 1
TF length: 2
Randomizer: 1
Interleaver: 1
Emulator: 1
Es/No: 140
Modulation Type: 0
Number of Frames: 1000
message has a length of 19 bytes and its size should be 19 bytes.
Unit Test message sent

Waiting for UT Response with a timeout of 1 seconds...
Response received: 0x0b 0x12 0x02 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
Unit Test performed without issues.
------------------
 Test results

 __   __   __   __   __   __
|__) /\  (__  (__  |__  |  \
|   /~~\ .__) .__) |__  |__/

dcpp@nereida:~/virtude/TestingTools/bin$
```

```
dcpp@nereida:~/virtude/TestingTools/bin$ ./VIRTUDE_TT ut
Arguments obtained:
ut


Configurations selected:
Message length: 18
Message type: 2
Operating Mode: 2
Memory Index: 0
Bit Rate: 80000000
Code Rate: 1
TF length: 2
Randomizer: 1
Interleaver: 1
Emulator: 1
Es/No: 0
Modulation Type: 0
Number of Frames: 1000
message has a length of 19 bytes and its size should be 19 bytes.
Unit Test message sent

Waiting for UT Response with a timeout of 1 seconds...
Response received: 0x0b 0x12 0x02 0x01 0x00 0x28 0x08 0xd0 0x00 0x00 0x03 0xe8
Unit Test performed without issues.
-----------------
 Test results

 __        ___  __
|__  /\  | |   |__  |  \
|   /~~\ | |__ |__  |__/

Number of wrong decoded frames: 1000
Number of bits errors: 2623696
BER: 0.36767
FER: 1
dcpp@nereida:~/virtude/TestingTools/bin$
```

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

Breadboard Unit Tests

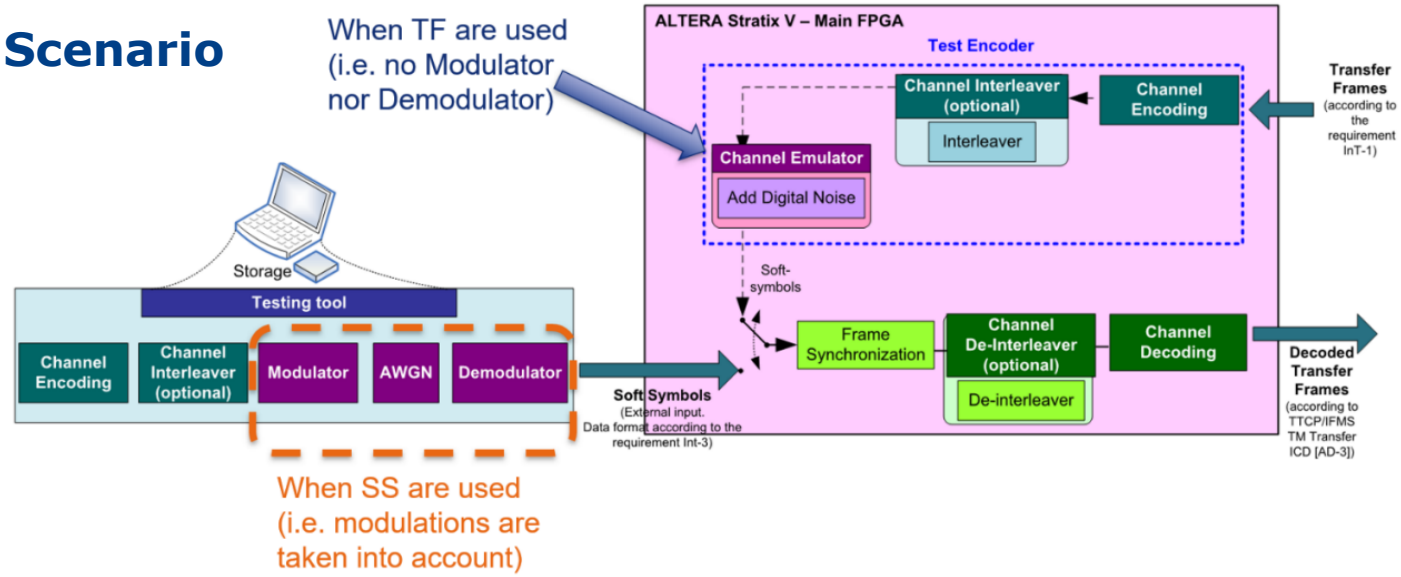
Component	Test ID	Objective	Test Results
External Interface <EI>			
	UT_EI_INTF	To confirm that the TFs are sent to the breadboard according to the requirement InT-1.	Validated using [Wireshark].
	UT_EI_INTSS	To confirm that the Soft-Symbols (SS) are sent to the breadboard according to the requirement InT-3.	Validated using [Wireshark].
	UT_EI_OUT	To verify that the breadboard is sending the Heartbeat messages according to TTCP-ICD-TM [AD-3].	Validated using [Wireshark].
	UT_EI_MON	To verify that the breadboard is transmitting its status according to the *Monitoring Message* format [TNO03].	Validated.
BreadBoard <BB>			
	UT_BB_ALL_1784	To confirm that all internal breadboard modules are working properly when TFs of 1784 bits are received.	Validated.
	UT_BB_ALL_3568	To confirm that all internal breadboard modules are working properly when TFs of 3568 bits are received.	Validated.
	UT_BB_ALL_7136	To confirm that all internal breadboard modules are working properly when TFs of 7136 bits are received.	Validated.
	UT_BB_ALL_8920	To confirm that all internal breadboard modules are working properly when TFs of 8920 bits are received.	Validated.

Breadboard Performance Tests

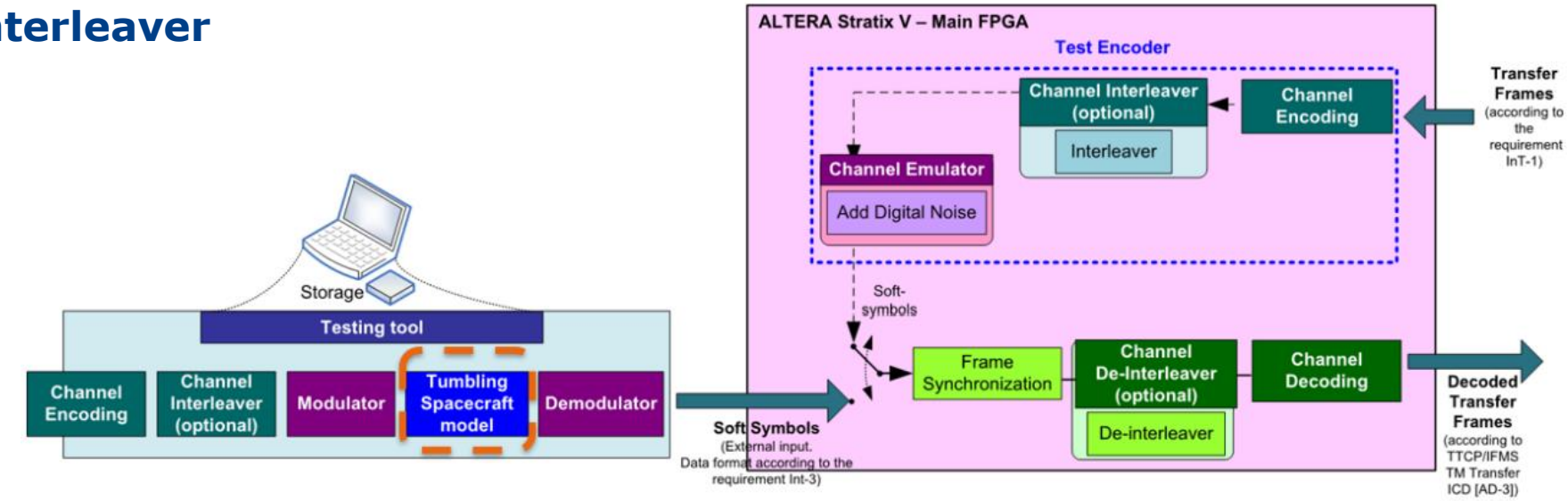
Component	Test ID	Objective	Test Results
Breadboard <BB>			
	PT_BB_ALL_1784	To validate the BB throughput of all modules for the information block length of 1784 bits with all optional modules enabled.	Validated.
	PT_BB_ALL_3568	To validate the BB throughput of all modules for the information block length of 3568 bits with all optional modules enabled.	Validated.
	PT_BB_ALL_7136	To validate the BB throughput of all modules for the information block length of 7136 bits with all optional modules enabled.	Validated.
	PT_BB_ALL_8920	To validate the BB throughput of all modules for the information block length of 8920 bits with all optional modules enabled.	Validated.
Receiver <RX>			
	PT_RX_TD_1784	To validate the Receiver Chain throughput and the Turbo Decoder performance for the information block length of 1784 bits.	Validated.
	PT_RX_TD_3568	To validate the Receiver Chain throughput and the Turbo Decoder performance for the information block length of 3568 bits.	Validated.
	PT_RX_TD_7136	To validate the Receiver Chain throughput and the Turbo Decoder performance for the information block length of 7136 bits.	Validated.
	PT_RX_TD_8920	To validate the Receiver Chain throughput and the Turbo Decoder performance for the information block length of 8920 bits.	Validated.
	PT_RX_FS_R12	To validate the Frame Synchronizer performance for the code rate of 1/2.	Validated.
	PT_RX_FS_R13	To validate the Frame Synchronizer performance for the code rate of 1/3.	Validated.
	PT_RX_FS_R14	To validate the Frame Synchronizer performance for the code rate of 1/4.	Validated.
	PT_RX_FS_R16	To validate the Frame Synchronizer performance for the code rate of 1/6.	Validated.
	PT_RX_CI_R12	To validate the Channel Interleaver for the Tumbling Spacecraft scenario using a code rate of 1/2.	Validated.
	PT_RX_CI_R13	To validate the Channel Interleaver for the Tumbling Spacecraft scenario using a code rate of 13.	Validated.
	PT_RX_CI_R14	To validate the Channel Interleaver for the Tumbling Spacecraft scenario using a code rate of 1/4.	Validated.
	PT_RX_CI_R16	To validate the Channel Interleaver for the Tumbling Spacecraft scenario using a code rate of 1/6.	Validated.

Performance Tests Setup

- **Normal Scenario**

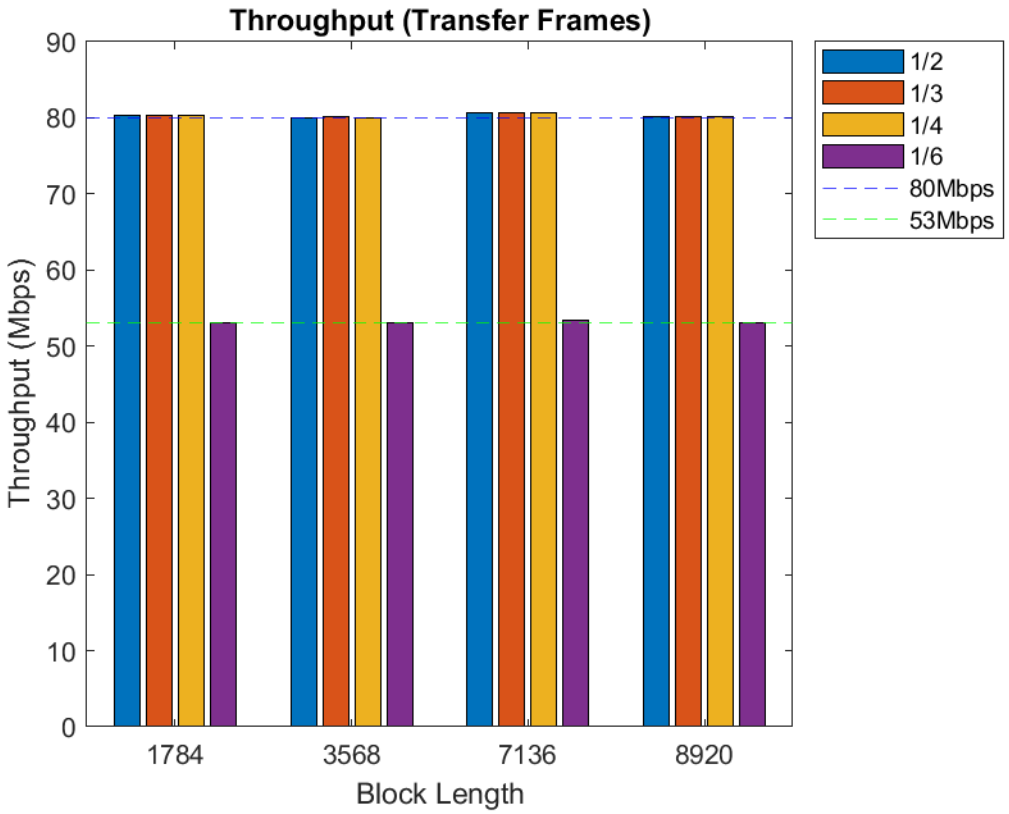


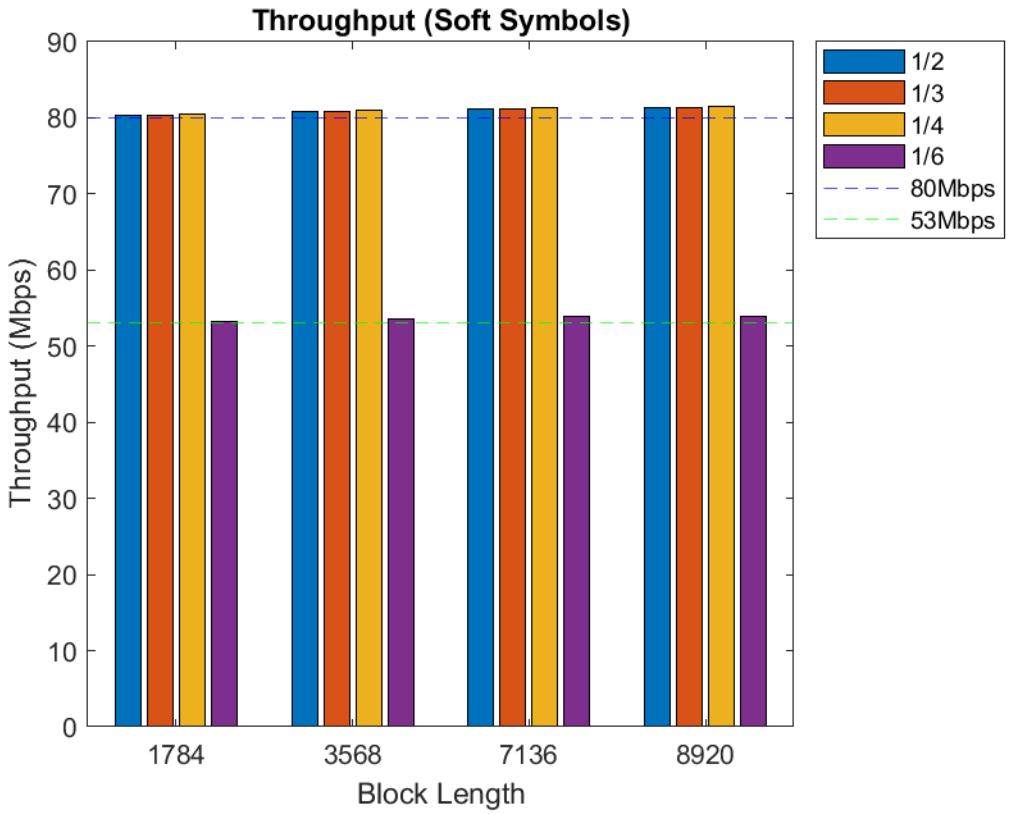
- **For Channel Interleaver**



Throughput

PeR-1: The Breadboard modules shall support data rates up to 80 Mbps in the TTCP SPM main FPGA

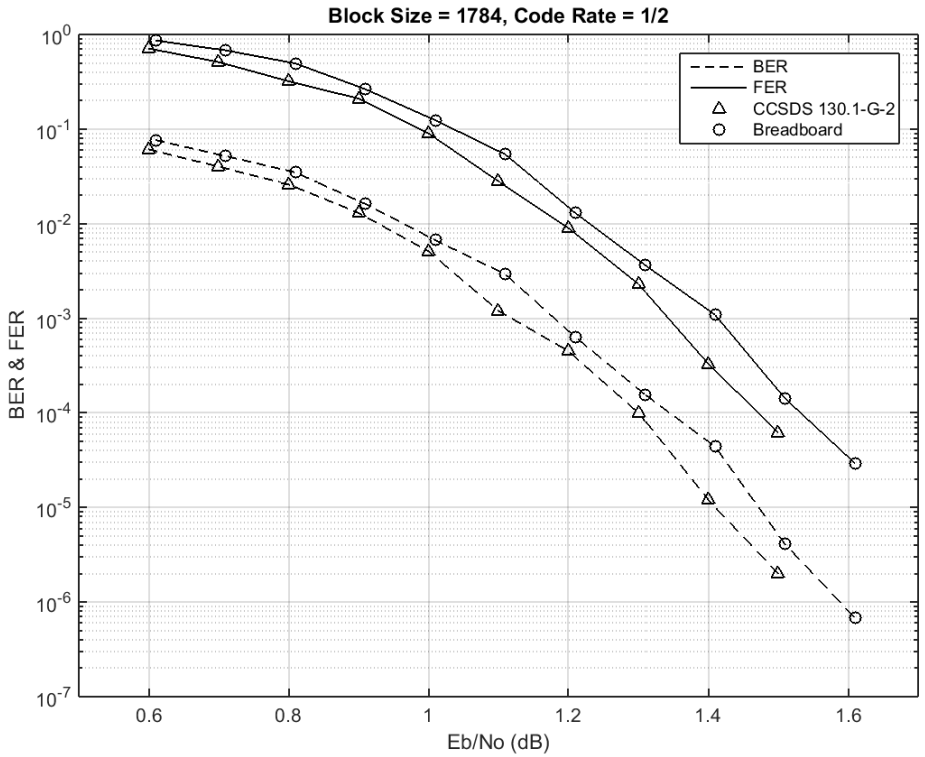


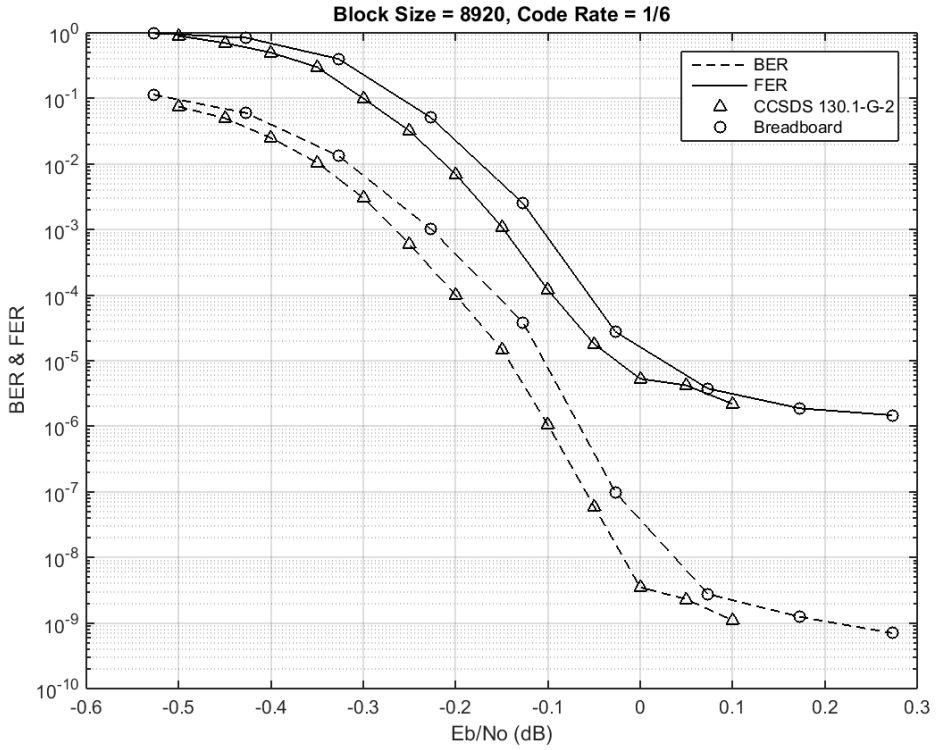


Performance: Turbo Decoder

- **PeR-2:** The technical loss of the Turbo decoder shall be less than 0.16 dB with respect to the reference figures of the Turbo decoders in [AD-2]

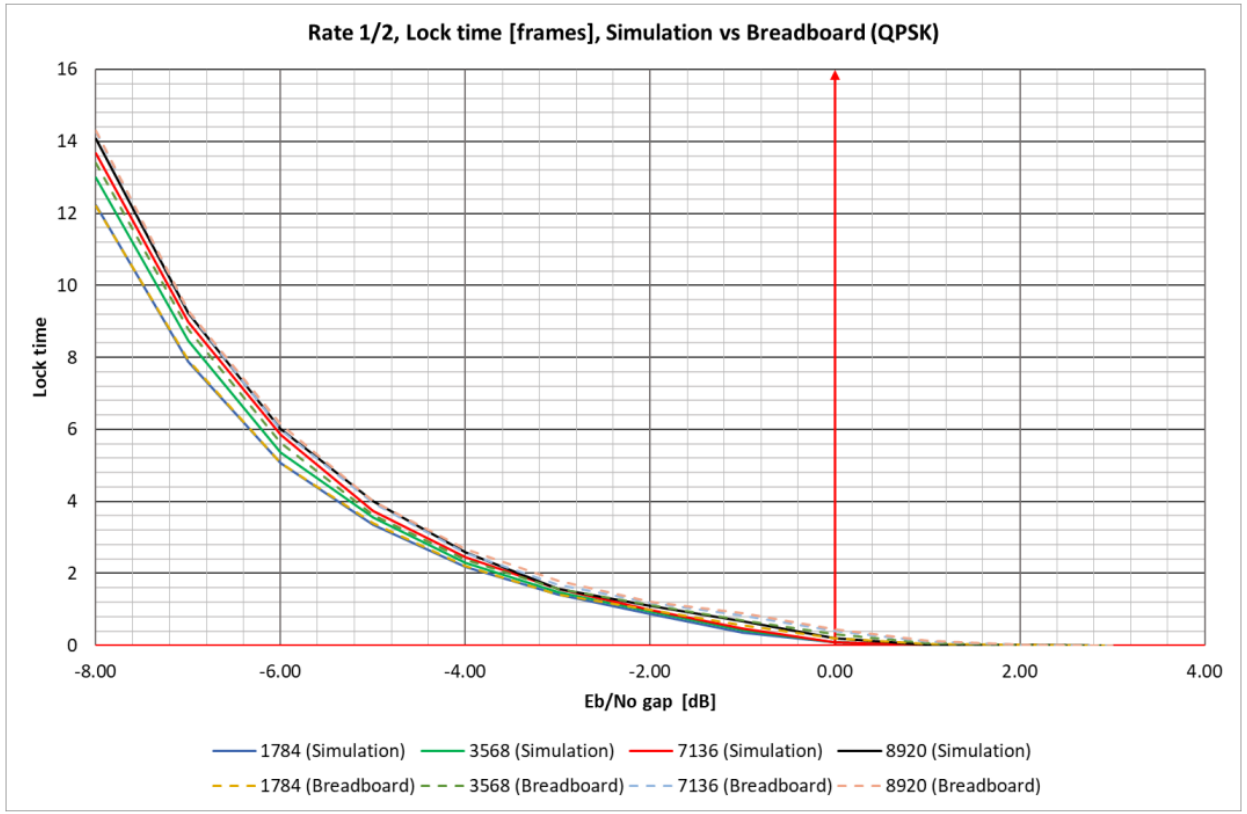
	BER				FER			
Block Length / Code Rate	**1/2**	**1/3**	**1/4**	**1/6**	**1/2**	**1/3**	**1/4**	**1/6**
1784	0.07	0.07	0.02	0.02	0.11	0.09	0.04	0.03
3568	0.08	0.06	0.04	0.03	0.09	0.07	0.04	0.04
7136	0.06	0.05	0.03	0.01	0.13	0.06	0.05	0.04
8920	0.15	0.05	0.11	0.08	0.13	0.07	0.04	0.05

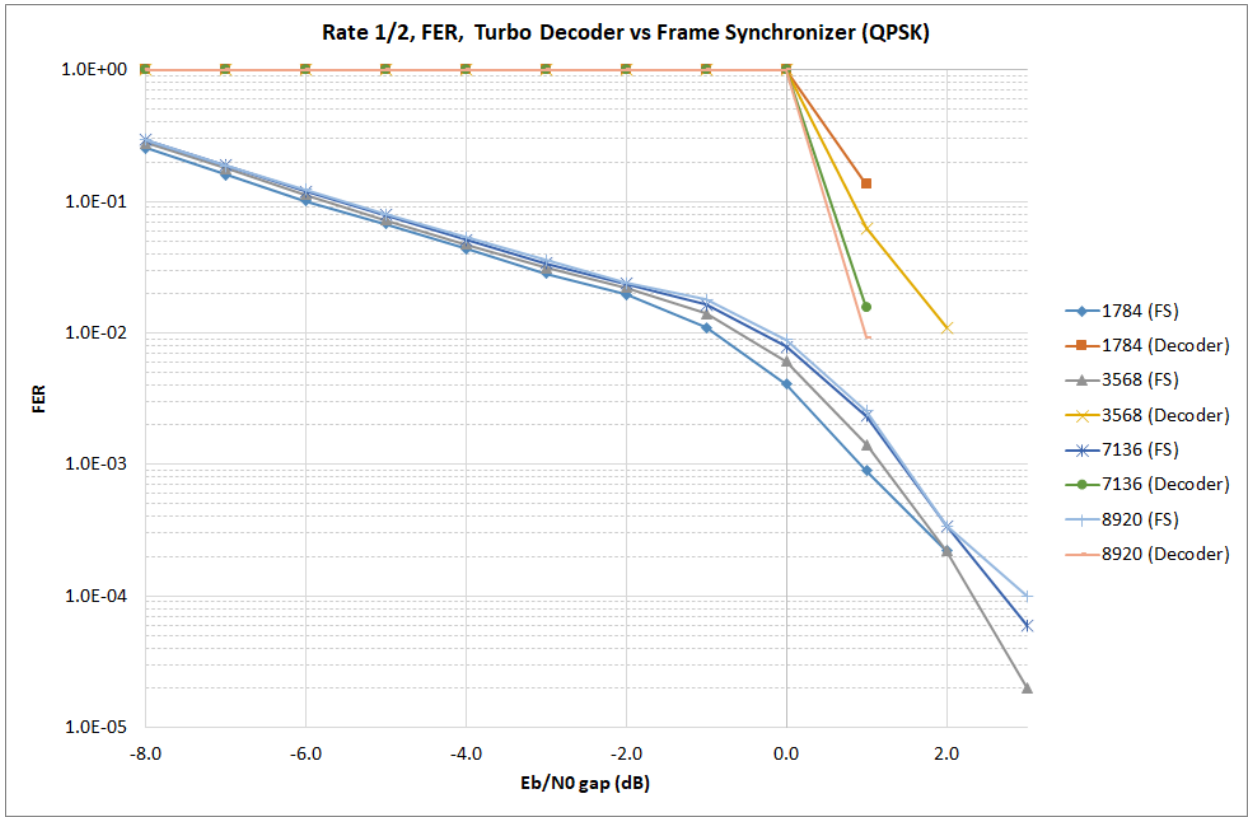




Performance: Frame Synchronizer (QPSK)

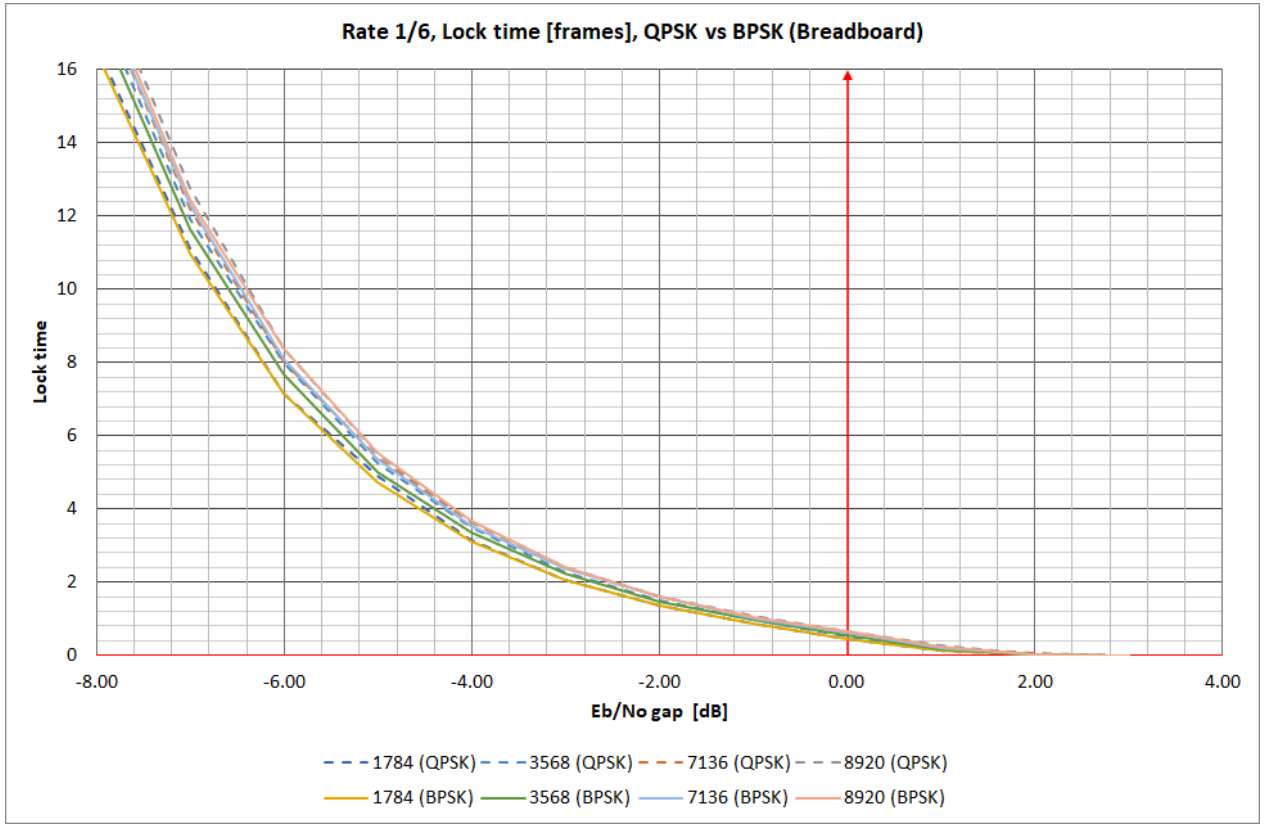
- Average number of frames lost during acquisition -> compared with simulation results
- Proved that the performance of Frame Synchronizer is superior than the Decoder

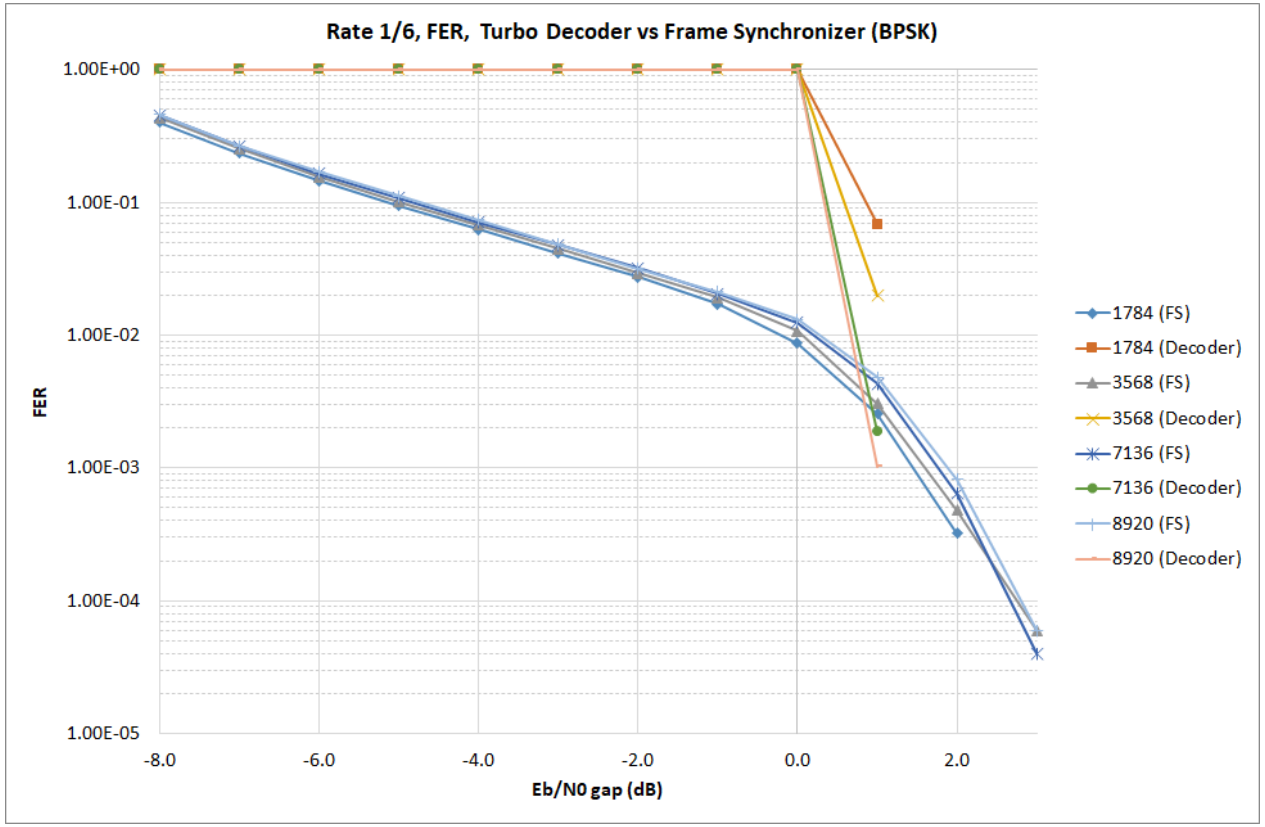




Performance: Frame Synchronizer (BPSK)

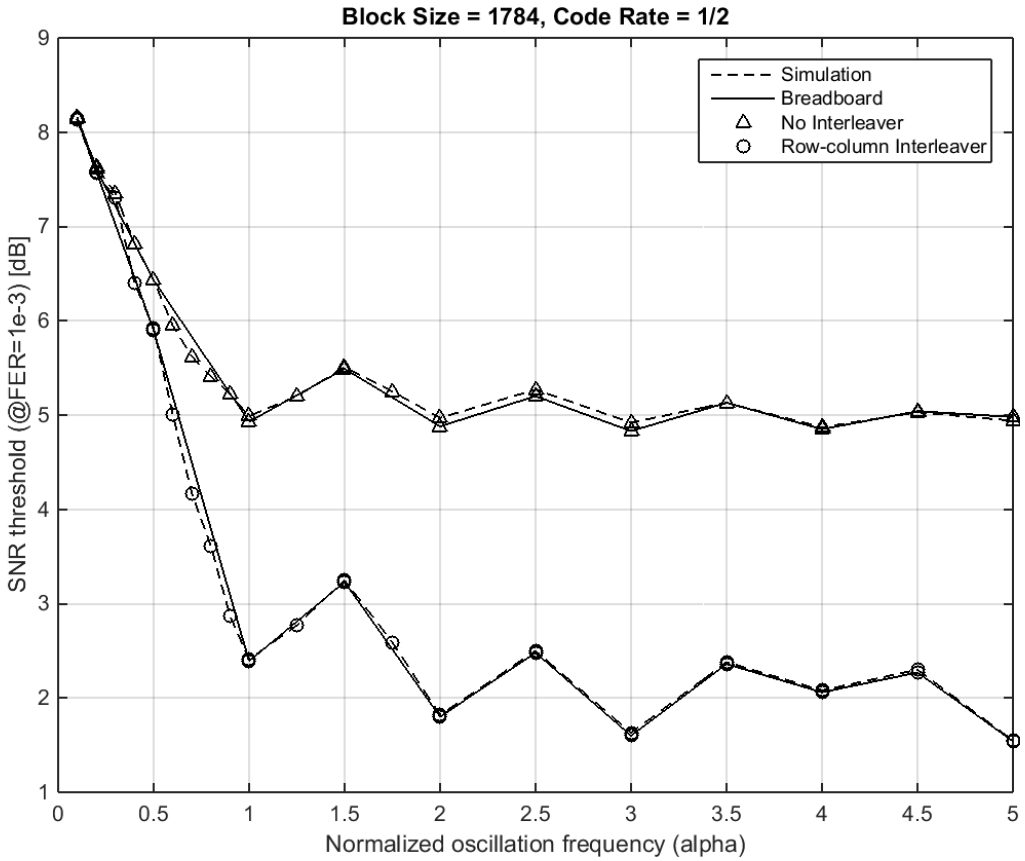
- Average number of frames lost during acquisition -> compared with QPSK results
- Proved that the performance of Frame Synchronizer is superior than the Decoder

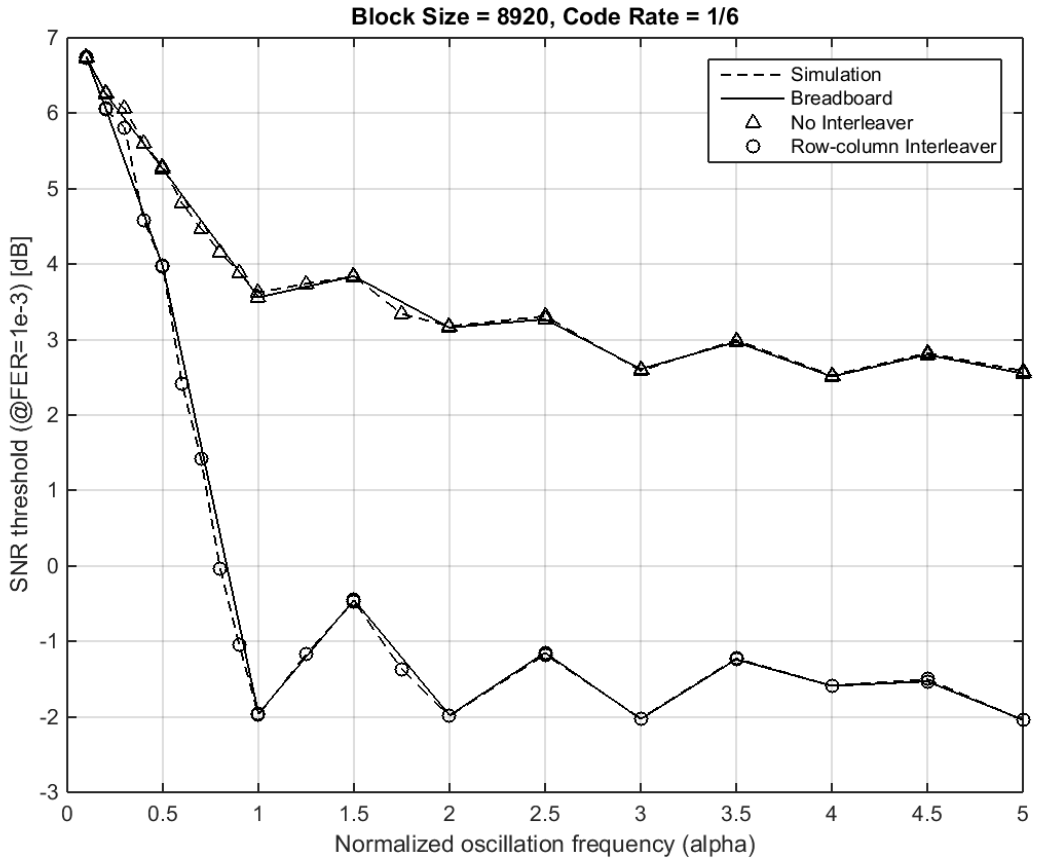




Performance: Channel Interleaver

- **Gain between 2 and 5 dB** on the SNR threshold for the FER of 1e-3 when using the Channel Interleaver on the Tumbling Spacecraft scenario for normalized oscillation frequencies above 1!





Conclusions

- All breadboard implementation only uses the FPGA (i.e., without using external memories or any external co-processor, as had been identified at the proposal stage)
- The channel interleaver was assessed and designed considering “tumbling spacecraft” and “solar scintillation” channel models
  - The row-column interleaver was successfully implemented and validated
- The Testing Tool (external tool) running on an external computer was developed to configure, monitor and evaluate the breadboard performance
- Main results:
  - The complete breadboard implementation supports the useful data rate of **80 Mbps**
  - Turbo Decoder complies with the maximum allowable implementation loss of **0.16 dB**
  - Channel Interleaver results are in line with the simulations allowing a gain between **2 and 5 dB** for the Tumbling Spacecraft model
- The project is ready to be integrated at TTCP

Agenda

14:00 – Introduction

14:10 – Synchronization and Channel Coding (S&CC) Sublayer

14:35 – Software of the CCSDS S&CC Sublayer (SW4)

14:45 – COTS Platform

14:55 – Breadboard Design

15:20 – Testing Tool

15:25 – Results Overview

15:45 – Q&A

16:00 – Meeting closure

Thank you

www.elecnor-deimos.com