Evaluation of SmartNIC devices for use in Trigger and Data Acquisition Systems

Adam Abed Abud, Andreas Klavenes Berg, Enrico Gamberini, Roland Sipos

24th IEEE Real Time Conference - ICISE, Quy Nhon, Vietnam
22-26 April 2024

Overview

- Introduction
  - SmartNICs
  - Motivation
- Use-case
  - Description
  - Hardware devices
  - Applications and implementation
- Results
  - Different scenarios
  - Conclusions
- Summary
  - Outlook and future work
  - Impressions

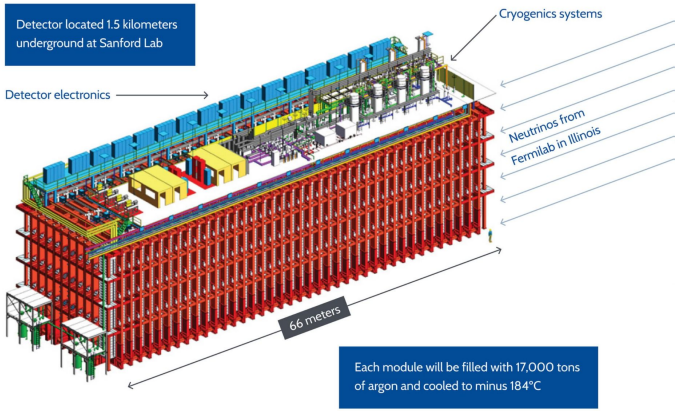
SmartNICs

Multi port/bypass “smart” adapters with programmable extensions (CPUs/FPGAs) to boost performance

- Offload processing from CPU
  - Network processing
  - Infrastructure plane
- Notable applications in data centers
  - Cyber security
  - Network monitoring and analytics
  - Traffic management
  - High-frequency trading
- For Trigger and DAQ system?

Pictures from: https://www.silicom-usa.com/

Picture from: https://lbnf-dune.fnal.gov/how-it-works/detectors-and-computing/



Motivation

Potential use-cases for TDAQ systems

- High data rate is common
  - Orders of TB/s are produced in large detectors
    E.g. on this conference: The Ethernet readout of the DUNE DAQ system
- Quasi-real time data reduction of incoming data streams
- Pattern recognition and stream manipulation (e.g.: calibration path)
- This contribution aims to explore both advantages and weaknesses of a SmartNICs for a particular use-case

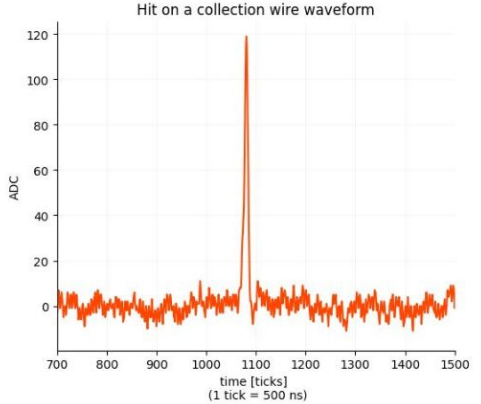
~10Tbps / module

x75 / module, each with 200Gbps

Case study

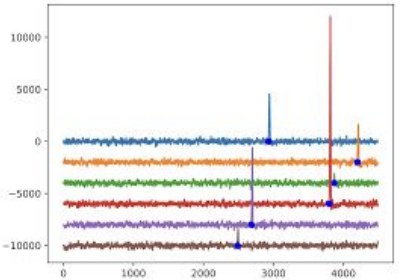
Description and constraints

- Based on the DUNE DAQ readout system
- Demonstrate how a SmartNIC may be used for reception of, and feature extraction from data generated by a detector.
- Features are extracted by processing the full complement of data and identifying time windows and electronics channels exhibiting activity not compatible with electronics noise:
  Known as Hit Finding



Trigger Primitive Generation (TPG)

Process that does hit finding and results in the forming of trigger primitive information

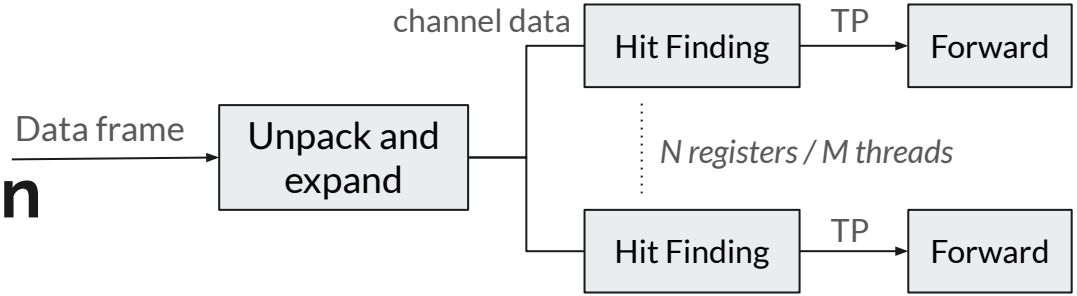


- Trigger Primitives (TPs) are generated whenever hits are detected
- Typically at ~0.005% of the incoming data frequency
- In parallel to TP generation, frames are buffered on host for several seconds waiting for the trigger to form decisions by the combined analysis of TPs
  - This long time is not driven by processing constraints but the sparse arrival of the signals that must be evaluated together in order to detect events
- With the design with an SmartNIC, the data buffer might be on the host machine, and the TPs are generated from the processor on the NIC and sent from it to the host when ready

Requirements

- Targeting to process 6144 channels, each producing 14 bit values at a rate of ~2MHz
- Overall data throughput is ~172 Gbps
  - Concentration strategy is into either 2 x 100 Gbps or 1 x 200 Gbps network interfaces
- Specific Front-End data frame format that requires “unpacking” of individual channels
  - Processing resource intensive operation
- Expected TP rate per channel is ~100 Hz
- TPs should be sent to downstream TDAQ components (e.g.: trigger, data selection)

TPG implementation



Single instruction Multiple Data (SiMD) parallelized implementation with the use of Advanced Vector Extensions (AVX) to the x86 instruction set.
Use of AVX2 registers with 256 bits. The TPG chain consist of:

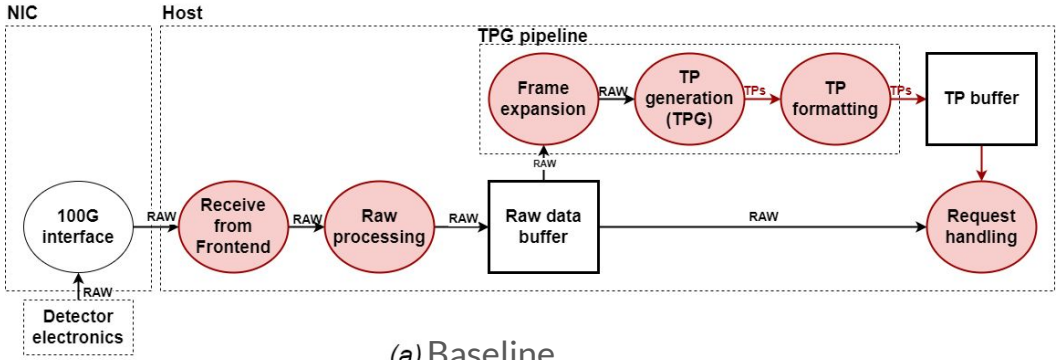
1. Data expansion: unpack channel from data frames and expand ADC data from 14 bit to 16 bit to ease interface with rest of computing infrastructure (I/O intensive task)
2. Initialize pedestal values
3. Execute the hit finding algorithm(s) (CPU intensive task)
4. TP extraction: parse output from hit finding and forward it to the following processing component

AVX2 code sample

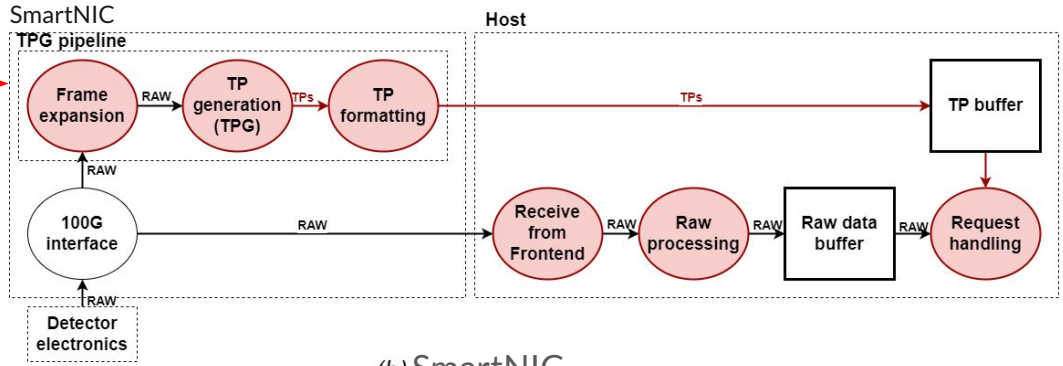
```
// if the sample is greater than the median, add one to the accumulator
// if the sample is less than the median, subtract one from the accumulator.

__m256i is_gt = _mm256_cmpgt_epi16(s, median);
__m256i is_eq = _mm256_cmpeq_epi16(s, median);

__m256i to_add = _mm256_set1_epi16(-1);

to_add = _mm256_blendv_epi8(to_add, _mm256_set1_epi16(1), is_gt);
to_add = _mm256_blendv_epi8(to_add, _mm256_set1_epi16(0), is_eq);

// Don't add anything to the channels which are masked out
to_add = _mm256_and_si256(to_add, mask);

accum = _mm256_add_epi16(accum, to_add);

is_gt = _mm256_cmpgt_epi16(accum, _mm256_set1_epi16(acclimit));
__m256i is_lt =
  _mm256_cmpgt_epi16(_mm256_sign_epi16(accum, _mm256_set1_epi16(-1 * acclimit)),
                     _mm256_set1_epi16(acclimit));
```

Offloading topology

Simplified schematic of the readout data-path in the baseline solution, compared with a modified design where the TPG processing chain has been relocated to a SmartNIC's on-board processing unit, and the raw data stream is duplicated towards the host.



(a) Baseline



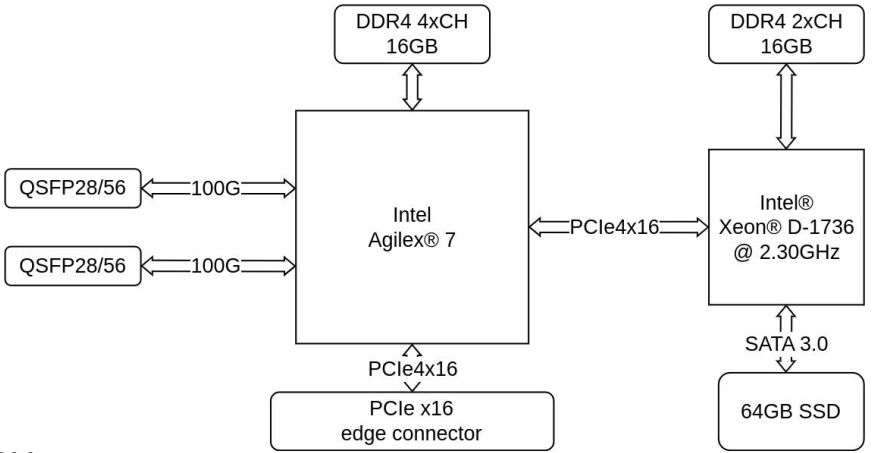
(b) SmartNIC

Picture from: https://intel.com

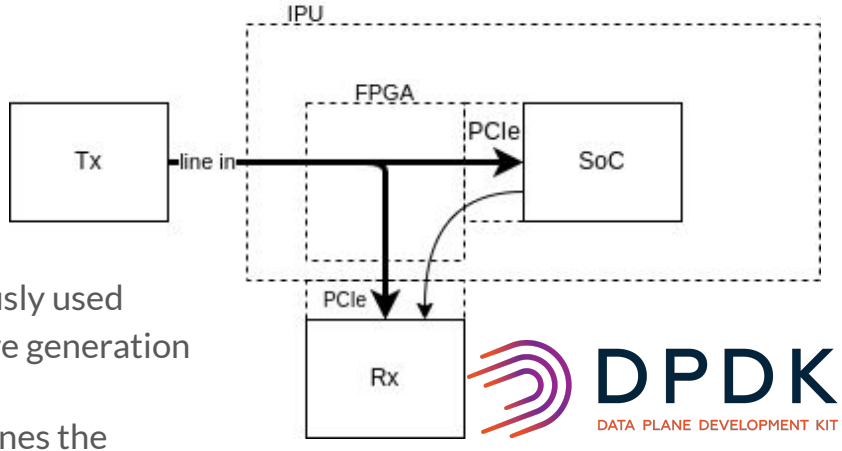
Intel IPU F2000X-PL

Our target platform to test with has the following specifications:

- Built with Altera Agilex 7 F-series FPGA and Intel Xeon-D-1736 processor (8 cores)
- 2 x 100 Gb interfaces
- PCIe 4.0 x16 interfaces from FPGA to both host CPU and Xeon-D processor
- Programmable through development kits:
  - Infrastructure Programmer Development Kit (IPDK)
  - Data Plane Development Kit (DPDK)
  - Storage Performance Development Kit (SPDK)
- Full Linux environment running on the on-board CPU



Test applications



Implemented based on DPDK, with modifications on previously used software components for data reception and trigger primitive generation

- The receiver (RX) application for the IPU's CPU combines the data reception software with the format specific TPG AVX2 implementation for the ProtoDUNE Single Phase detector
  - Uses an older detector data format
  - With the possibility of different stages of the TPG to be enabled or disabled

- Transmitter (TX) applications are running on dedicated servers connecting to the IPU
  - With ratelimiter to control rate of frame arrival
  - Can send realistic data with replay mode

Results

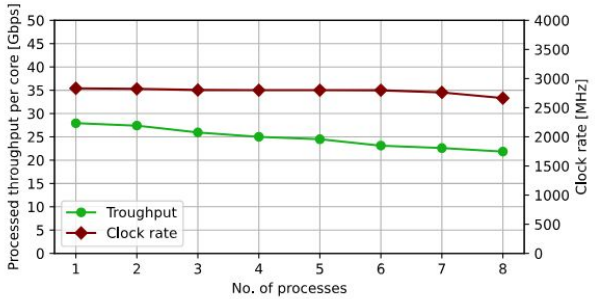
Measurements and Performance

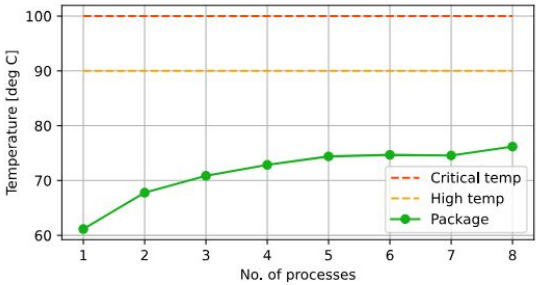
- Tested many different scenarios via incrementally enabling the stages of the workload
- Pure processing tests explore theoretical limits of the hardware resources
- Unpacking stage overhead measurement
- Realistic workload with receiving from the network, and sending data towards host

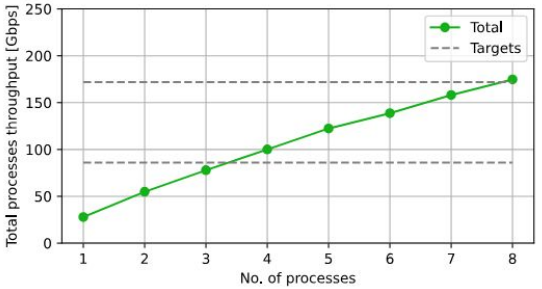
Pure processing

Stress test with scaling up the number of TPG processes, without sending the results downstream

- Without receiving any data (no TX)
- Each process is using a small static dataset, which should be ideal for the CPU cache lines
  - The expected TP rate is for each channel to produce a hit at a frequency of 100 Hz:
    Only 0.005% of the frames will produce a hit
    (See backup slide for TP rate scan!)
- Results give an upper cap to the range of performance when adding more stages



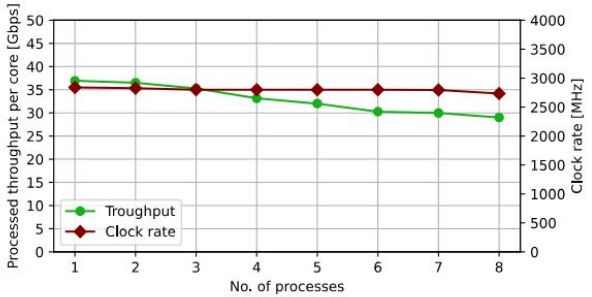


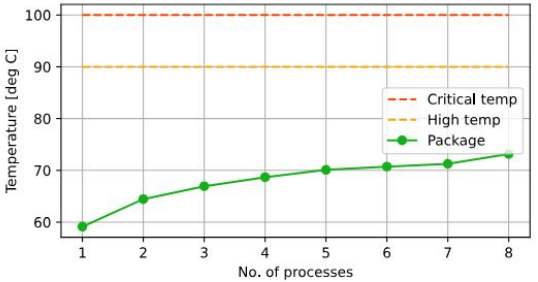


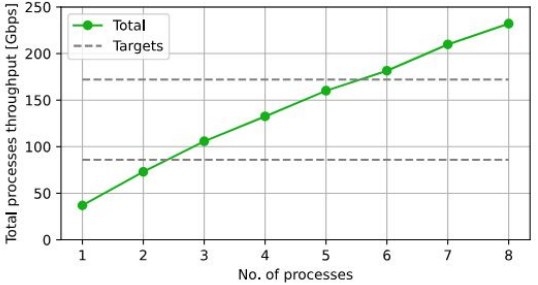
Unpacking overhead

We explored the resource utilization overhead of the frame unpacking and expansion stage of the TPG

- The unpacking stage does explicit copies for rearranging the data
- To see the difference compared to the results in the previous slide, we processed already unpacked data



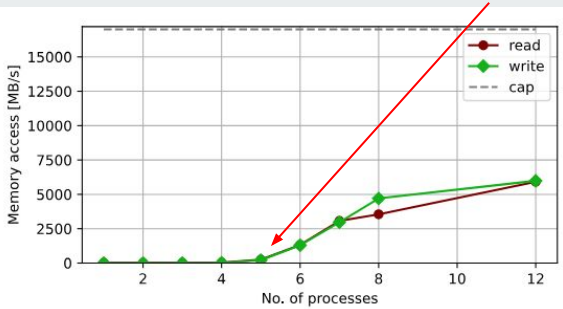


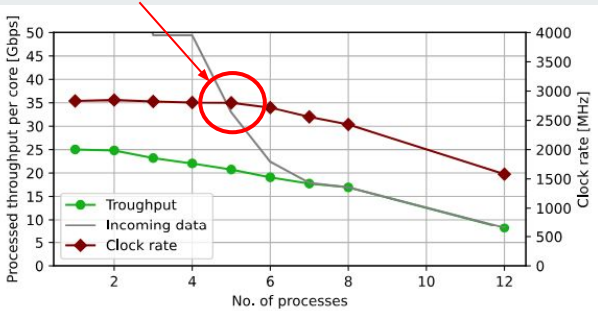


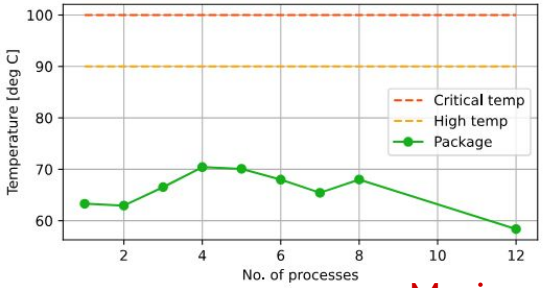
Realistic workload

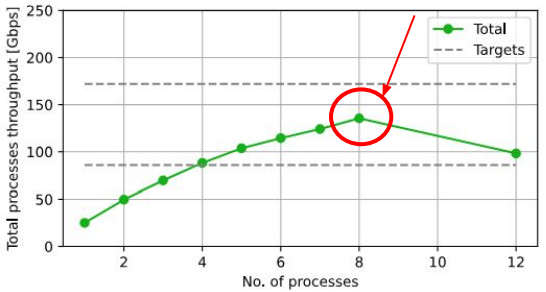
Receive data over the network, produce results in form of Trigger Primitives, and send it towards the host

- Incoming data saturates the link, packets are distributed over a number of active cores
- Packets are processed directly on the DMA buffers, without explicit copies
- In case of backpressure, unprocessed frames are overwritten
- Also requires extra memory I/O for data reception and TP formatting

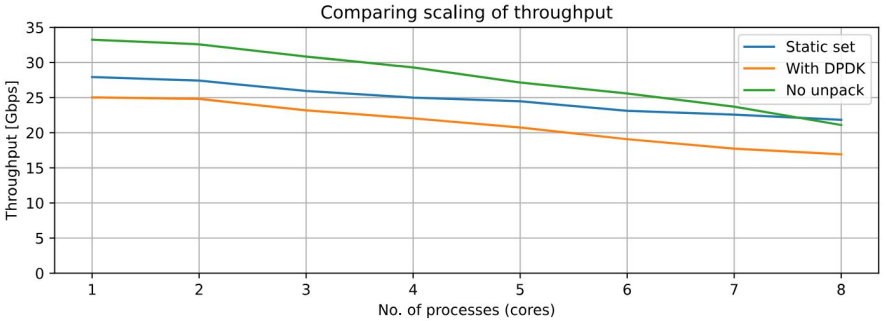








Conclusion on results



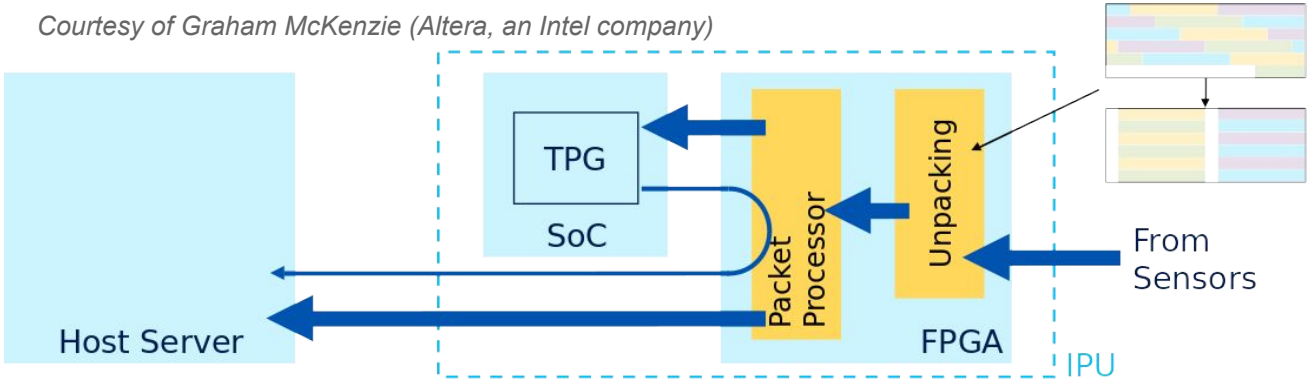
- We compared different scenarios for exploring the resource utilization overhead of different stages of the data reception and trigger primitive generation stages
- The test application configured with the realistic test achieved 130 Gbps throughput (70% of the required)
- Identified a major hot-spot: data unpacking carries substantial overhead on memory bandwidth and CPU utilization
- Available on-board CPU resources are limited for this high-throughput use-case
  - Despite software optimizations and heavy tuning of the hardware

Summary

- Ongoing and future work
- Impressions

Outlook

Courtesy of Graham McKenzie (Altera, an Intel company)



Ongoing work on software

- Adapt new data format of the DAQ
  - more efficient for network traffic
- Use the recently available emulators for increasing hit rate and impact on throughput
- Test other new algorithms

Implementation of frame expansion and reordering in the FPGA

- Introduce user block in the FPGA for manipulation of packets
- Frame expansion and channel extraction
  - Bit/byte shuffling is heavy on CPUs and memory
- Unpacking is a fixed-function, trivial offload for the FPGA

Impressions

- The use-case might not be a perfect match for a SmartNIC, but provided an excellent performance evaluation opportunity and see how far we can go
- In-depth knowledge on the IPU and its configuration was needed
  - The firmware was the baseline and a basic configuration was provided by Silicom experts
- Tuning and optimization was required on almost all components, including hardware (power DIP switches), SoC and Host BIOS (power settings) and extensive software optimizations to fit in the resource envelope

End

Thank you for your attention!

Thanks to our colleagues at Silicom, Intel, and Altera for providing the platform, and their contributions to this research!

Backup: Effects on TP rate

We explored the performance effects as the hit rate of the TPG is increased beyond the expected values.

- Incoming data rate was set slightly above the nominal rate of 2 MHz
- Expected TP rate of 100 Hz per channel
- TP rate that could be sustained at nominal data rate: at 90 kHz (4%, x900 expected rate)