Implementation of nBLM algorithms

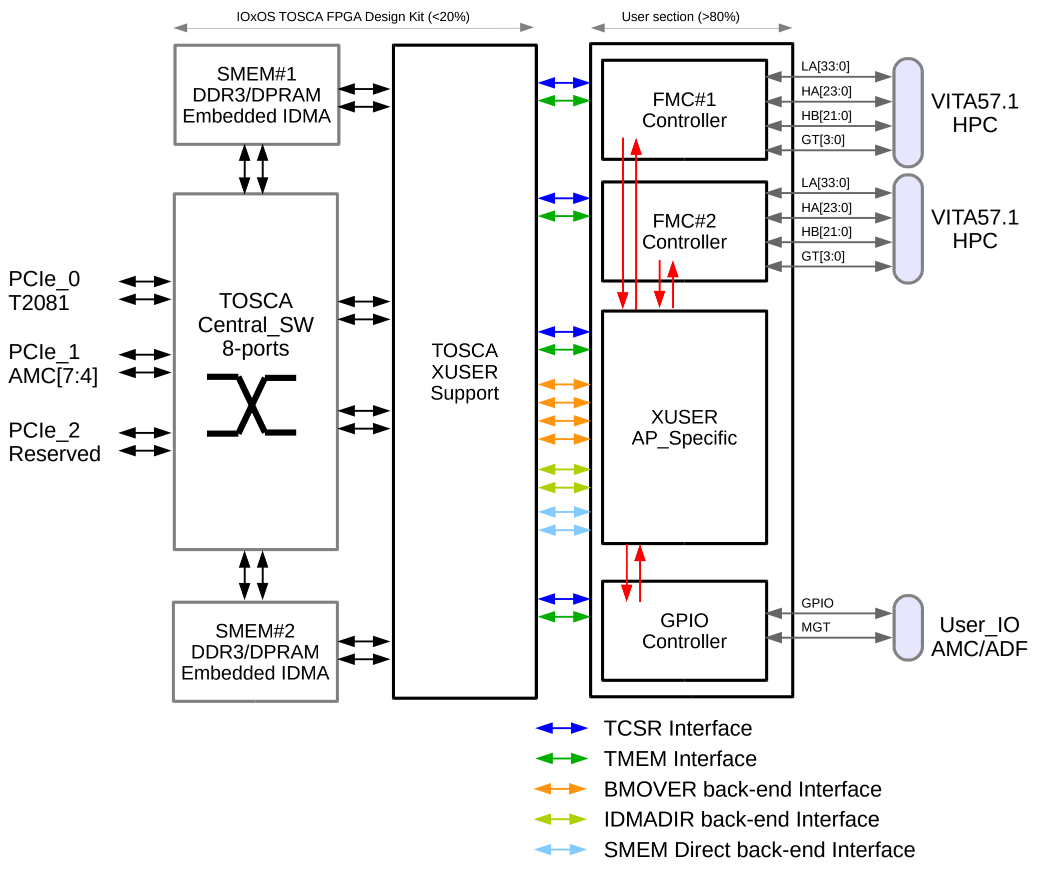
dr inż. Grzegorz Jabłoński,
dr inż. Wojciech Jalmużna,
dr inż. Rafał Kiełbik

21.11.2018

Department of Microelectronics and Computer Science, Lodz University of Technology
http://www.dmcs.p.lodz.pl

nBLM BEE hardware platform IOxOS IFC_1410

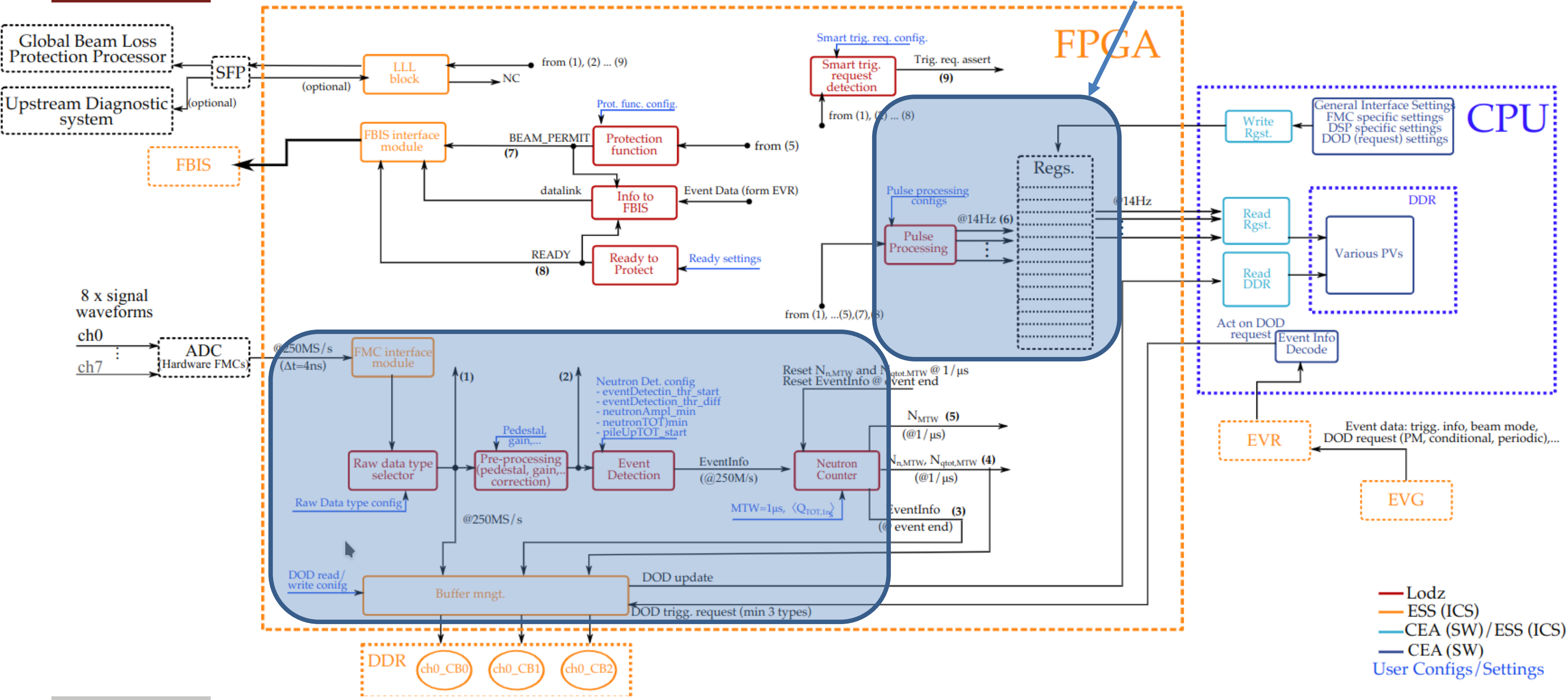
- 2 banks of 512 MB DDR3 memory
- T2081 POWER processor
- XCKU040 FPGA
- PCIe interface
- 2 FMC Slots



Department of Microelectronics and Computer Science, Lodz University of Technology
http://www.dmcs.p.lodz.pl

Implementation status

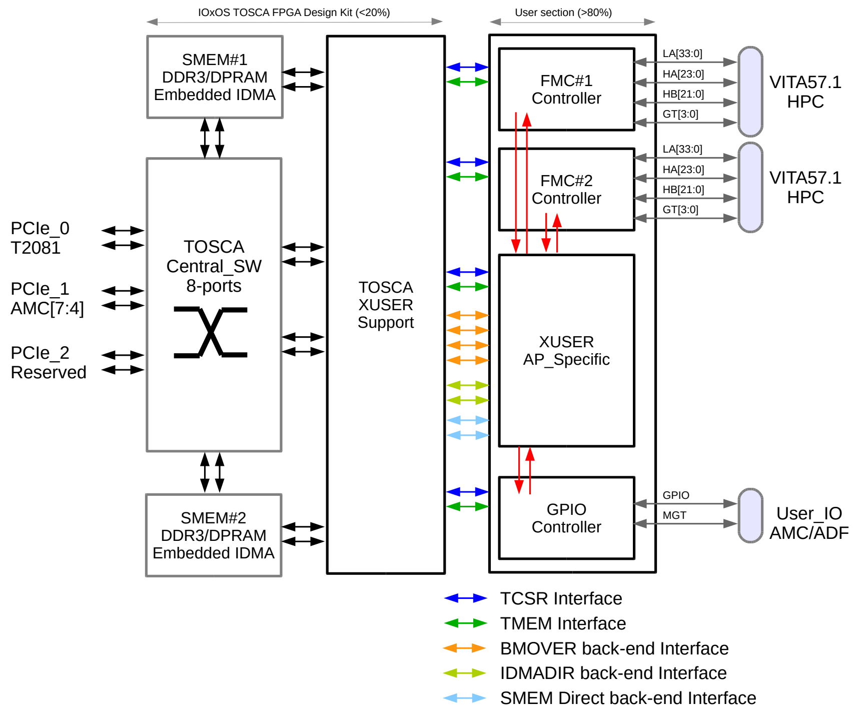
Periodic data is available via DDR



Department of Microelectronics and Computer Science, Lodz University of Technology
http://www.dmcs.p.lodz.pl

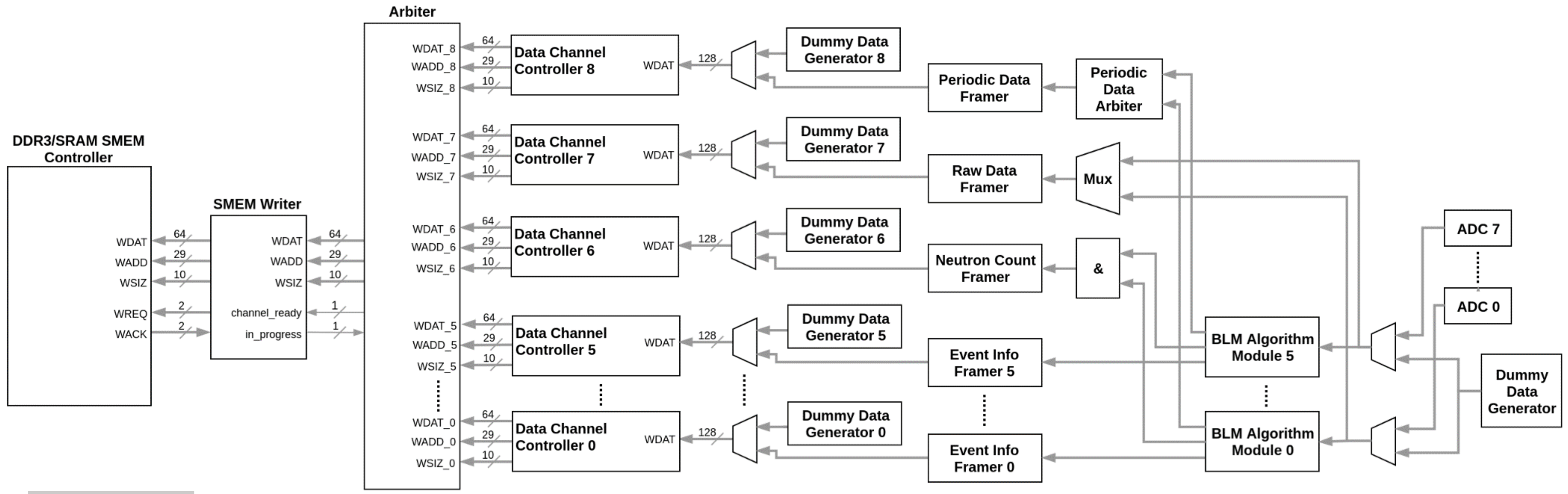
Hardware-software interface

- Using circular buffers in DRAM to stream data to CPU
- Using TSCR interface for control and algorithm parameters
- Single DDR bank (currently) via SMEMDIR interface, 1700 MB/s bandwidth
- Not needing Block RAMs overlapping initial part of DDR



Data transmission protocol layers

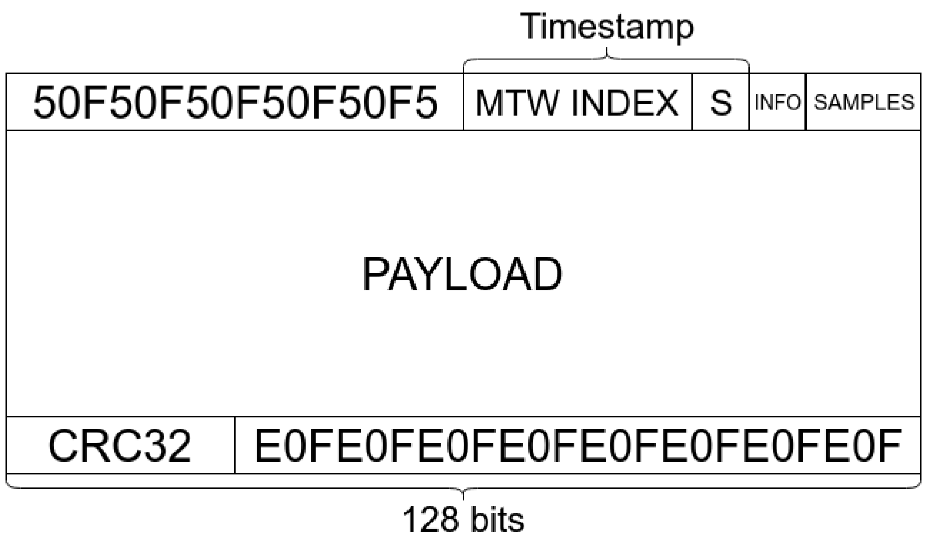
- Circular buffers – stream of unstructured data
- Data frames – timestamping and integrity check
- Periodic data – content-specific header



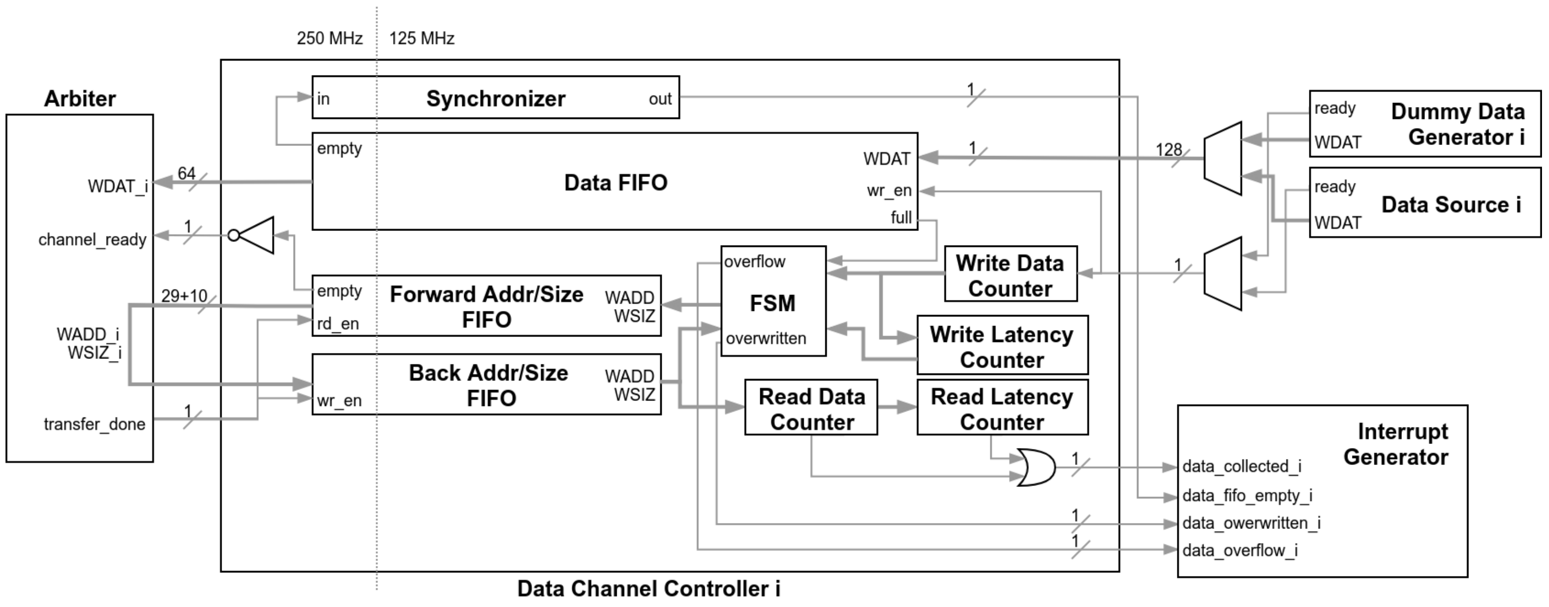
Department of Microelectronics and Computer Science, Lodz University of Technology
http://www.dmcs.p.lodz.pl

Frame layout

- Start-of-frame pattern
- Timestamp
  - Serial number of the 1-microsecond window
  - Sample index within the window.
- 1 generic information byte
- 16-bit number of samples in the frame
- Payload packed back-to-back on the bit level without any additional padding
- 32-bit CRC of all the previous words in the frame followed by the End-of-frame pattern.



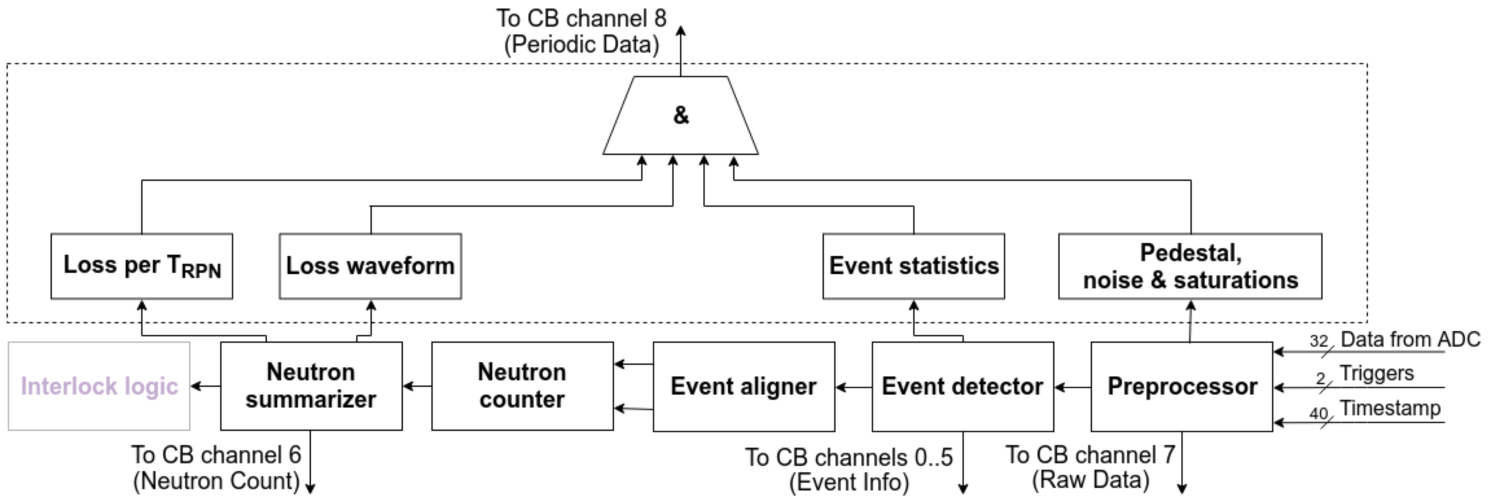
Circular Buffer Implementation



- Read pointer, write pointer for each channel
- Overflow – not able to write data to DDR at given input data rate
- Overwrite – data readout by DMA too slow
  - Read pointer at the moment of overwrite recorded

nBLM algorithms block diagram

- Main flow – 5 pipelined processing blocks
- Implemented in C++ with High Level Synthesis
- Data in CB channels 0-7 timestamped by MTW number and sample number within MTW (unique within more than 1 hour, cycle-accurate)
- Periodic data – timestamped in the same way, but not cycle-accurate



Event detection algorithm

```
void
detect (hls::stream<preprocessedData>& A, hls::stream<eventInfo>& E, hls::stream<eventInfo>& E2,
hls::stream<pedestalComputationData>& PC, hls::stream<eventInfoForArchiving>& event_stream,
uint16_t neutronTOT_min_indx, uint16_t pileUpTOT_start_indx)
{
#pragma HLS LATENCY min=1 max=1
#pragma HLS PIPELINE II=1
#pragma HLS INTERFACE axis off port=A
#pragma HLS DATA_PACK variable=A
// ….
for (int i = 0; i < 2; ++i)
//…
      if (data.belowThr1 || (data.belowThr2 && ended_by_frame))
        {
          //start an event
          if (!bEve)
            {
              MTWindx = data.frame_index;
              bEve = true;
              TOTstartTime = data.sample_index;
              peakValue = data.adjusted_sample;
              peakTime = 0;
              peakValid = false;
              TOTvalid = false;
              pileUp = false;
              TOTlimitReached = false;
              event_isPart2 = ended_by_frame;
              TOT = -1;
              Q_TOT = 0;
              peakCounter = 0;
```

Problems with Tosca framework

- Problems with timing closure
  - Example project does not compile cleanly
  - Constraining location one of BUFGs helped
- Problems during firmware/board startup
  - Requires several reboots to work properly
- Three different drivers
  - IOxOS: Tsc, factory test driver, not actively developed
  - PSI: Tosca, DMA 560 MB/s, userspace interrupt supported, convenient interface, cannot be used on Concurrent CPU, not actively developed
  - ESS: Modified Tsc, DMA 950 MB/s, no userspace interrupt support, less convenient interface, actively developed

Current status

- Fully implemented main data processing chain for 6 ADC channels (selectable from 8 inputs) with 8 data streams
  - Neutron, saturations and background event count every microsecond from all channels
  - Raw data from any of the 6 channels
  - Raw events from 6 channels
- Periodic data partially implemented
  - Pedestal, noise and saturations
  - Loss waveforms in 4 windows
  - Loss every T_{RPN}
  - Event statistics
- Tosca and Tsc drivers supported on POWER CPU
  - Possible to record 2 s of raw data from 1 channel with Tsc driver
- Ongoing algorithm evaluation by CEA and BI-ESS teams.

Thank you for your attention