FireDAQ for FLAME – architecture and hardware considerations

Jakub Moroń

AGH University of Science and Technology, Krakow, Poland

Outline

- New readout scheme
  - New Trenz Electronic module with Zynq UltraScale+
- FPGA firmware details
- System bandwidth (maximal event rate)
- Work to be done (and by whom...)

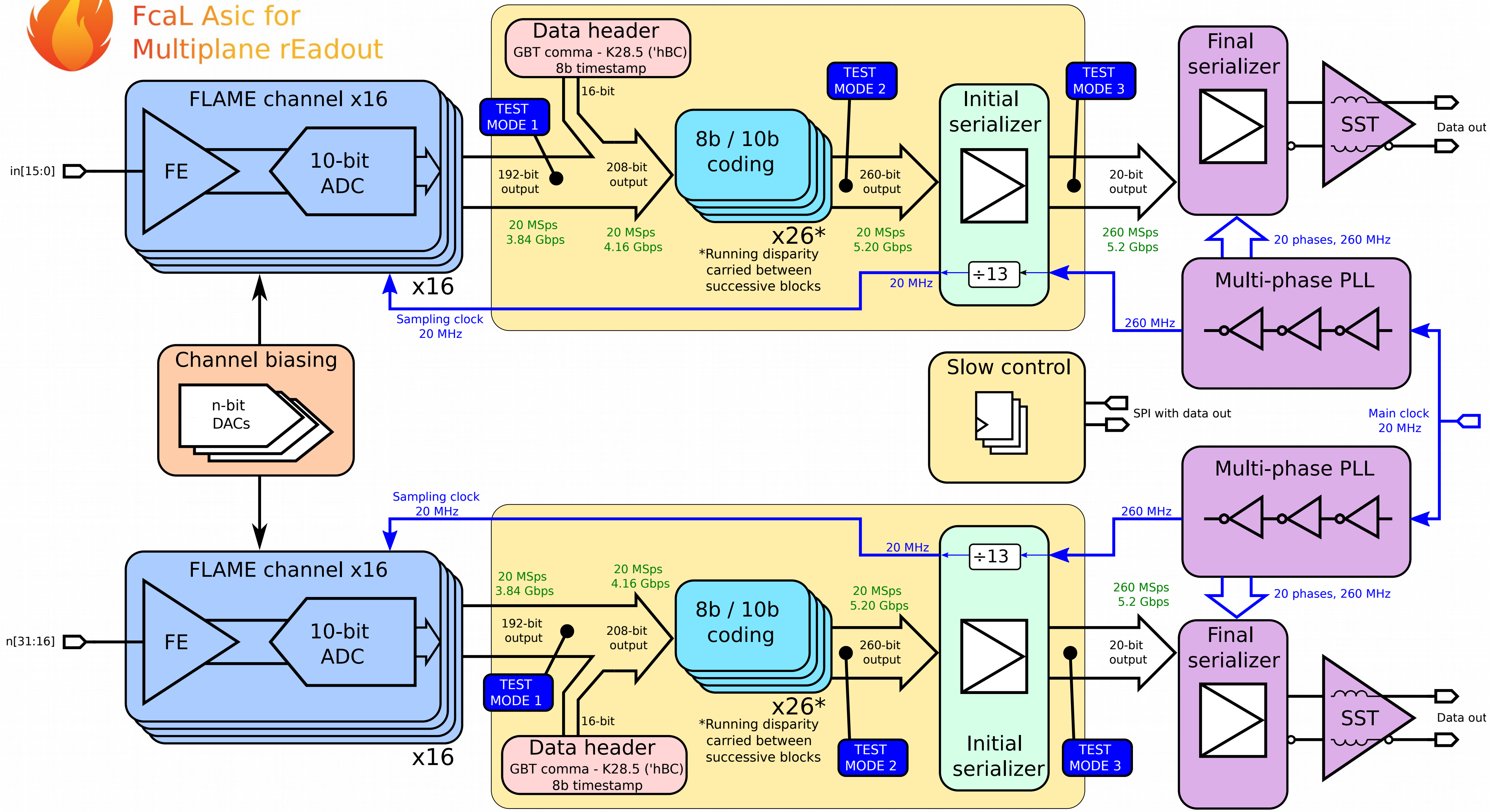
FLAME ASIC architecture

AGH

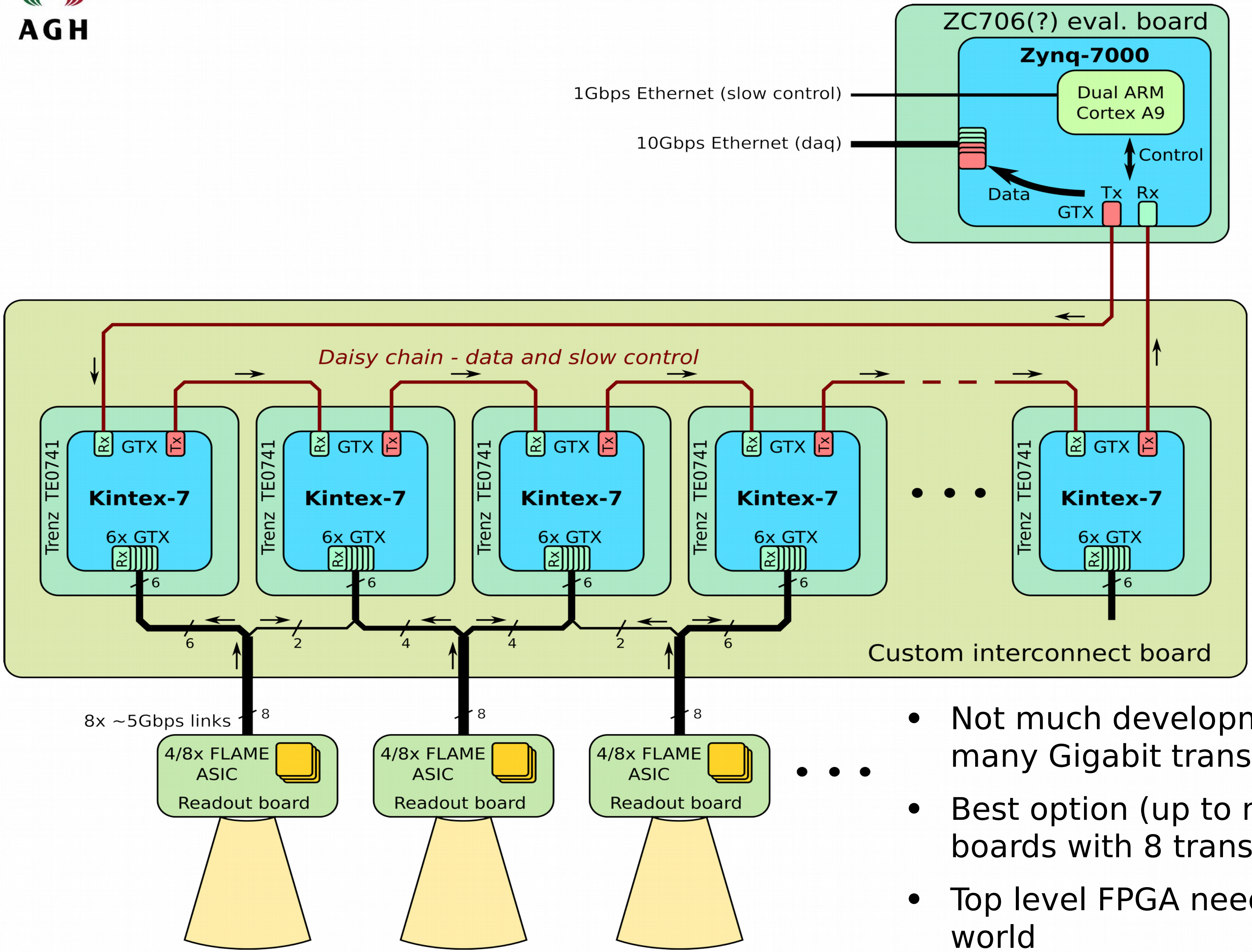
FLAME

FcaL Asic for Multiplane rEadout

- Two 16-channel, fully functional blocks → two “ASICs” in one padring to save the PCB area and maximize the instrumented sensor area

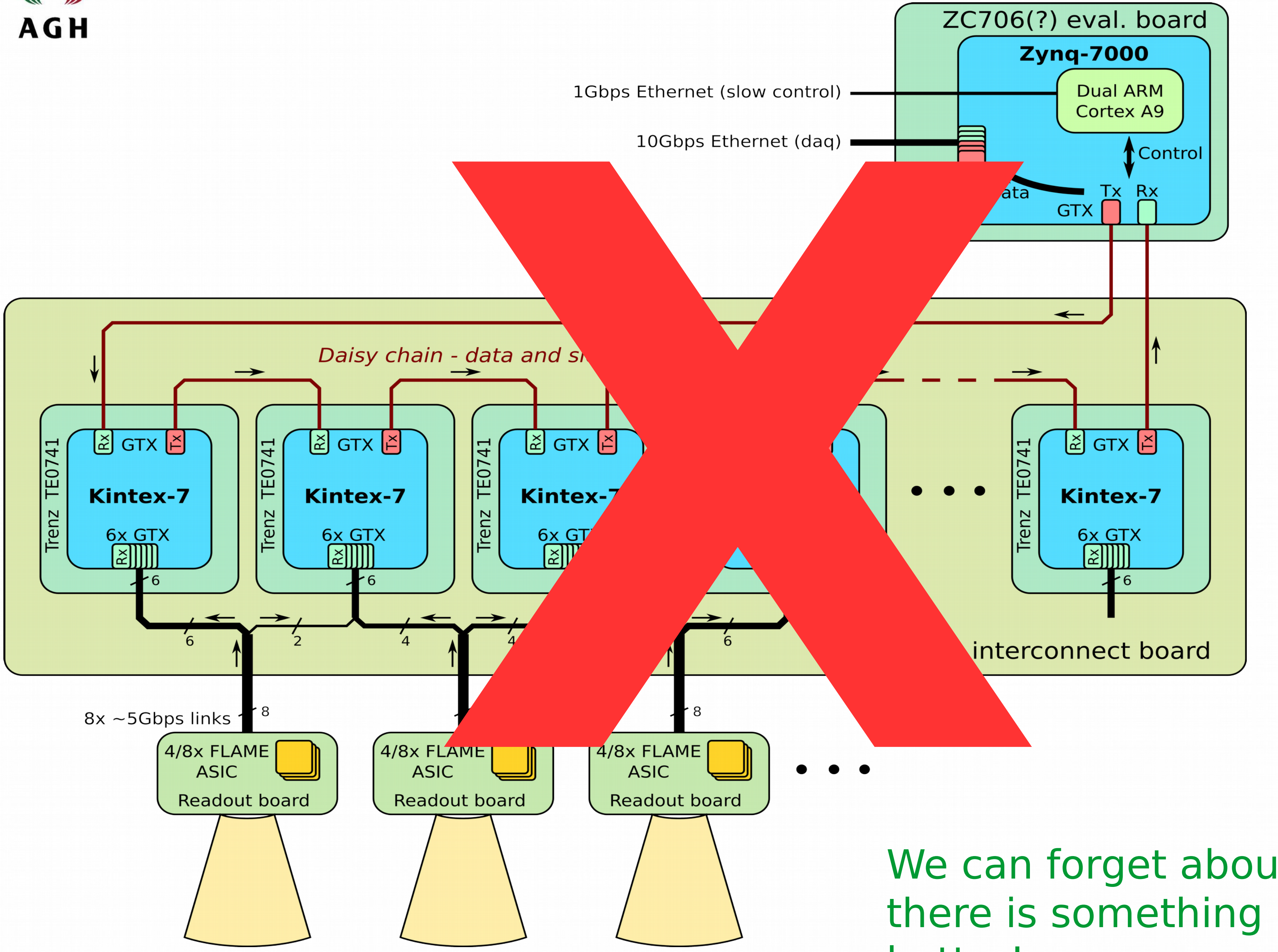


Old DAQ architecture



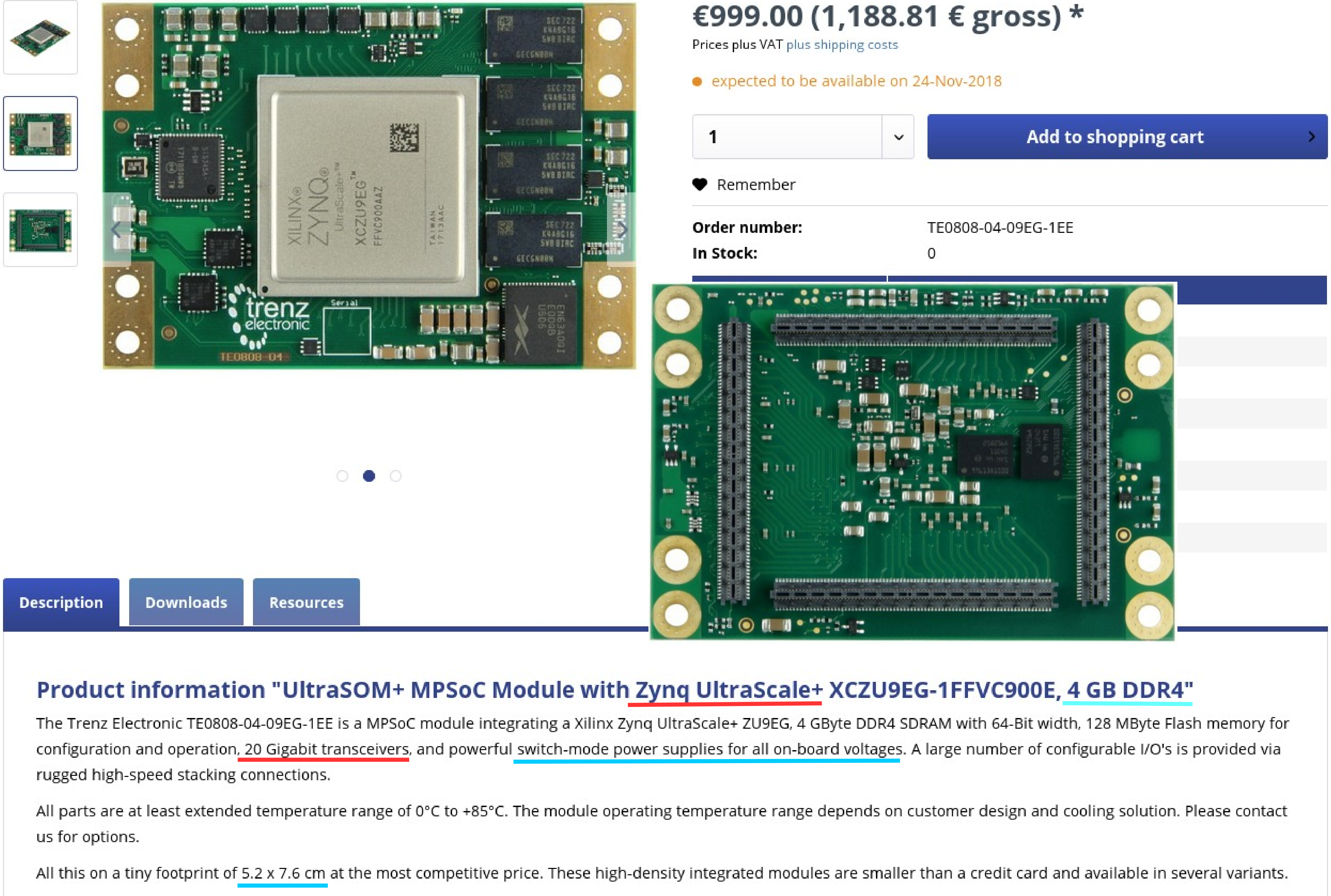
- Not much development boards with many Gigabit transceivers available...
- Best option (up to now) – Trenz Electronic boards with 8 transceivers
- Top level FPGA needed as port to outside world
- Very complicated scheme...

Old DAQ architecture



We can forget about this scheme, there is something definitely better!

Trenz Electronic TE0808 modules



€999.00 (1,188.81 € gross) *

Prices plus VAT plus shipping costs

● expected to be available on 24-Nov-2018

1 | Add to shopping cart ›

Remember

Order number:	TE0808-04-09EG-1EE
In Stock:	0

Description | Downloads | Resources

Product information "UltraSOM+ MPSoC Module with Zynq UltraScale+ XCZU9EG-1FFVC900E, 4 GB DDR4"

The Trenz Electronic TE0808-04-09EG-1EE is a MPSoC module integrating a Xilinx Zynq UltraScale+ ZU9EG, 4 GByte DDR4 SDRAM with 64-Bit width, 128 MByte Flash memory for configuration and operation, 20 Gigabit transceivers, and powerful switch-mode power supplies for all on-board voltages. A large number of configurable I/O's is provided via rugged high-speed stacking connections.

All parts are at least extended temperature range of 0°C to +85°C. The module operating temperature range depends on customer design and cooling solution. Please contact us for options.

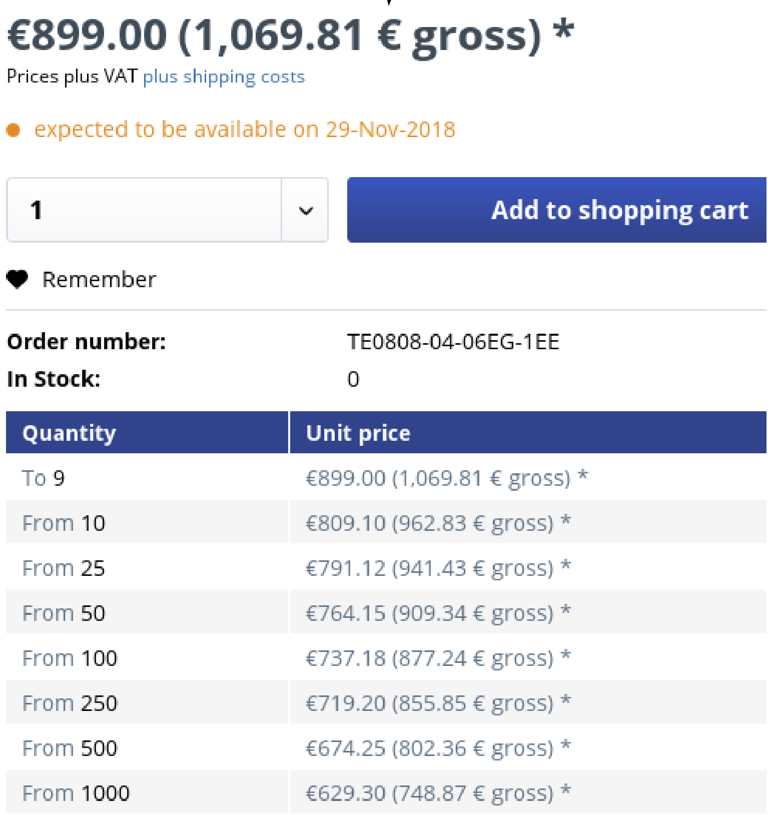
All this on a tiny footprint of 5.2 x 7.6 cm at the most competitive price. These high-density integrated modules are smaller than a credit card and available in several variants.

Slightly shorter than the credit card

Trenz Electronic TE0808 modules

We are trying to buy two TE0808-04-09EG modules just now

Trenz board	FPGA	Price (EUR with tax)	System cells	CLB flip-flops	CLK LUTs	DSP slices	DDR4 RAM
TE0808-04-06EG-1EE	XCZU6EG-1FFVC900E	1070	469k	429k	215k	1973	4 GB
TE0808-04-09EG-1EE	XCZU9EG-1FFVC900E	1189	600k	548k	274k	2520	4 GB
TE0808-04-15EG-1EE	XCZU15EG-1FFVC900E	1605	747k	682k	341k	3528	2 GB



€899.00 (1,069.81 € gross) *

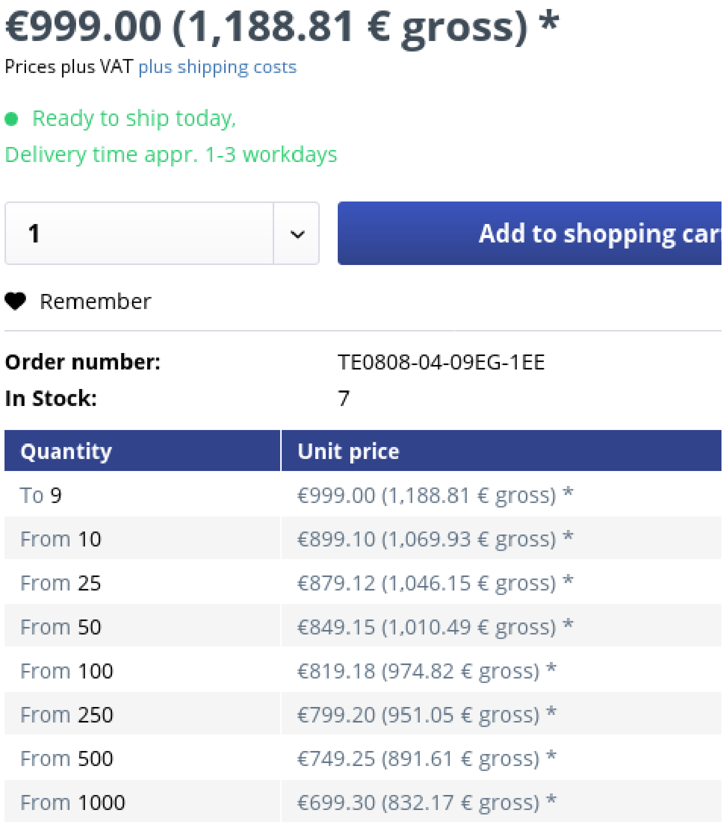
Prices plus VAT plus shipping costs

● expected to be available on 29-Nov-2018

Order number: TE0808-04-06EG-1EE

In Stock: 0

Quantity	Unit price
To 9	€899.00 (1,069.81 € gross) *
From 10	€809.10 (962.83 € gross) *
From 25	€791.12 (941.43 € gross) *
From 50	€764.15 (909.34 € gross) *
From 100	€737.18 (877.24 € gross) *
From 250	€719.20 (855.85 € gross) *
From 500	€674.25 (802.36 € gross) *
From 1000	€629.30 (748.87 € gross) *



€999.00 (1,188.81 € gross) *

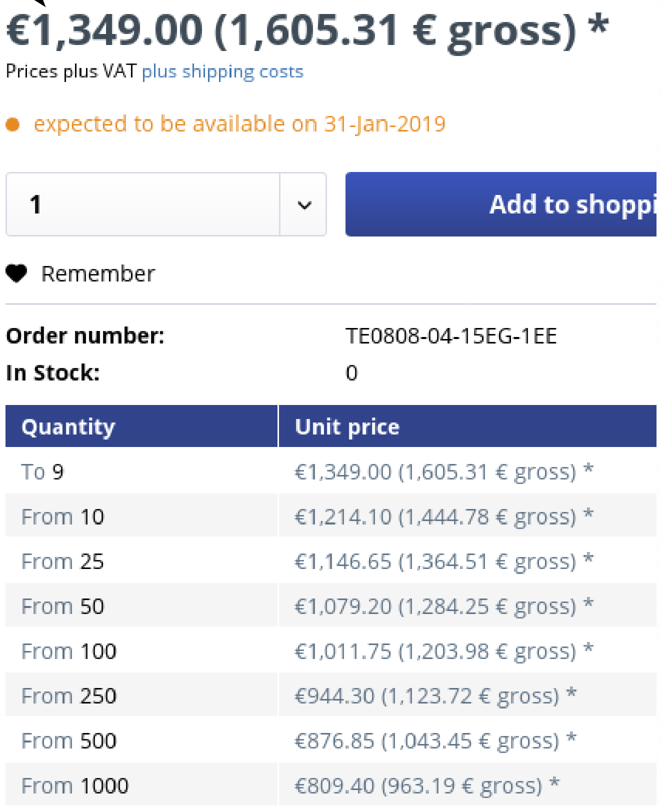
Prices plus VAT plus shipping costs

● Ready to ship today,
Delivery time appr. 1-3 workdays

Order number: TE0808-04-09EG-1EE

In Stock: 7

Quantity	Unit price
To 9	€999.00 (1,188.81 € gross) *
From 10	€899.10 (1,069.93 € gross) *
From 25	€879.12 (1,046.15 € gross) *
From 50	€849.15 (1,010.49 € gross) *
From 100	€819.18 (974.82 € gross) *
From 250	€799.20 (951.05 € gross) *
From 500	€749.25 (891.61 € gross) *
From 1000	€699.30 (832.17 € gross) *



€1,349.00 (1,605.31 € gross) *

Prices plus VAT plus shipping costs

● expected to be available on 31-Jan-2019

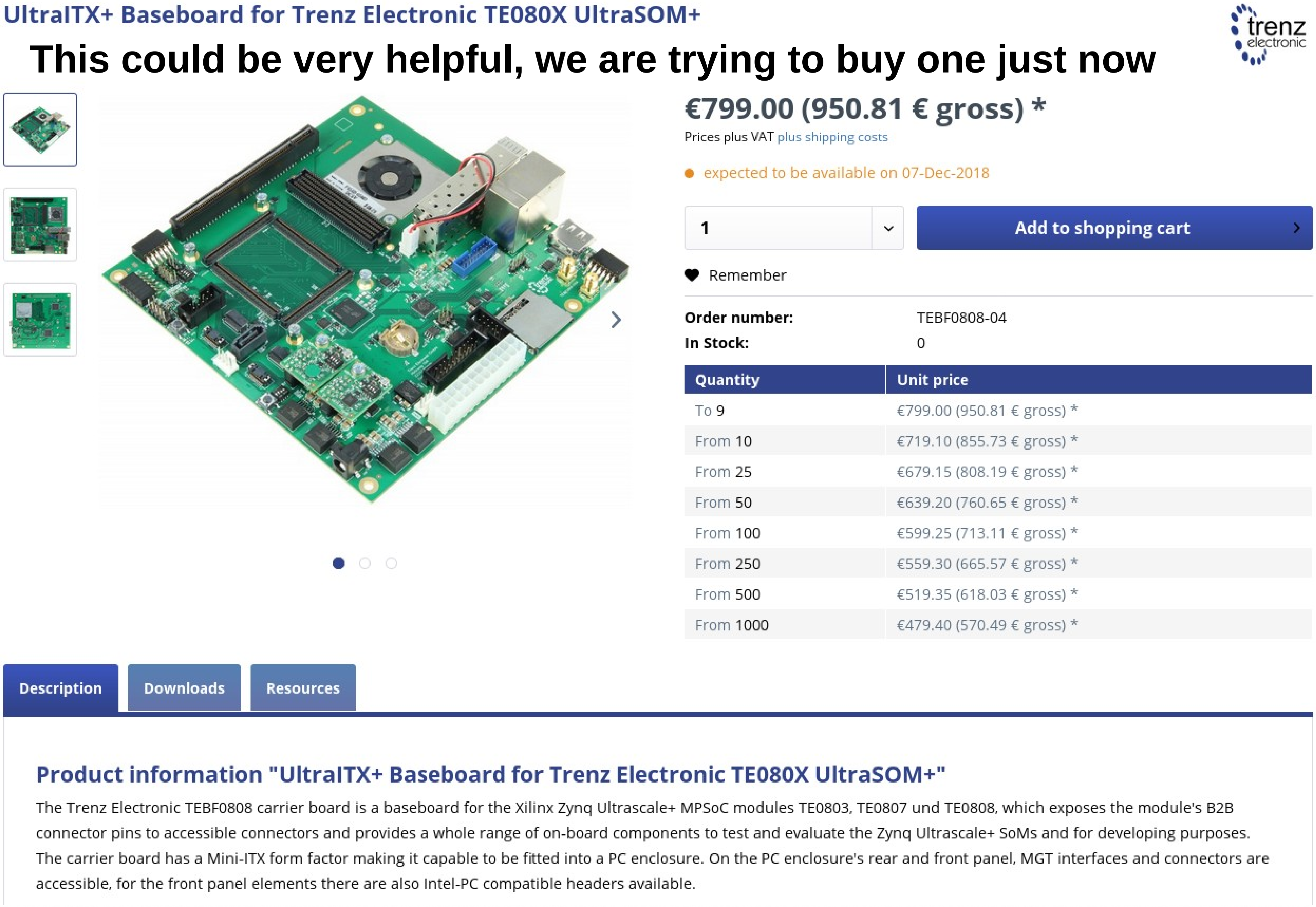
Order number: TE0808-04-15EG-1EE

In Stock: 0

Quantity	Unit price
To 9	€1,349.00 (1,605.31 € gross) *
From 10	€1,214.10 (1,444.78 € gross) *
From 25	€1,146.65 (1,364.51 € gross) *
From 50	€1,079.20 (1,284.25 € gross) *
From 100	€1,011.75 (1,203.98 € gross) *
From 250	€944.30 (1,123.72 € gross) *
From 500	€876.85 (1,043.45 € gross) *
From 1000	€809.40 (963.19 € gross) *

Trenz TE080X baseboard for developement

This could be very helpful, we are trying to buy one just now



UltraITX+ Baseboard for Trenz Electronic TE080X UltraSOM+

€799.00 (950.81 € gross) *

Prices plus VAT plus shipping costs

● expected to be available on 07-Dec-2018

Order number: TEBF0808-04

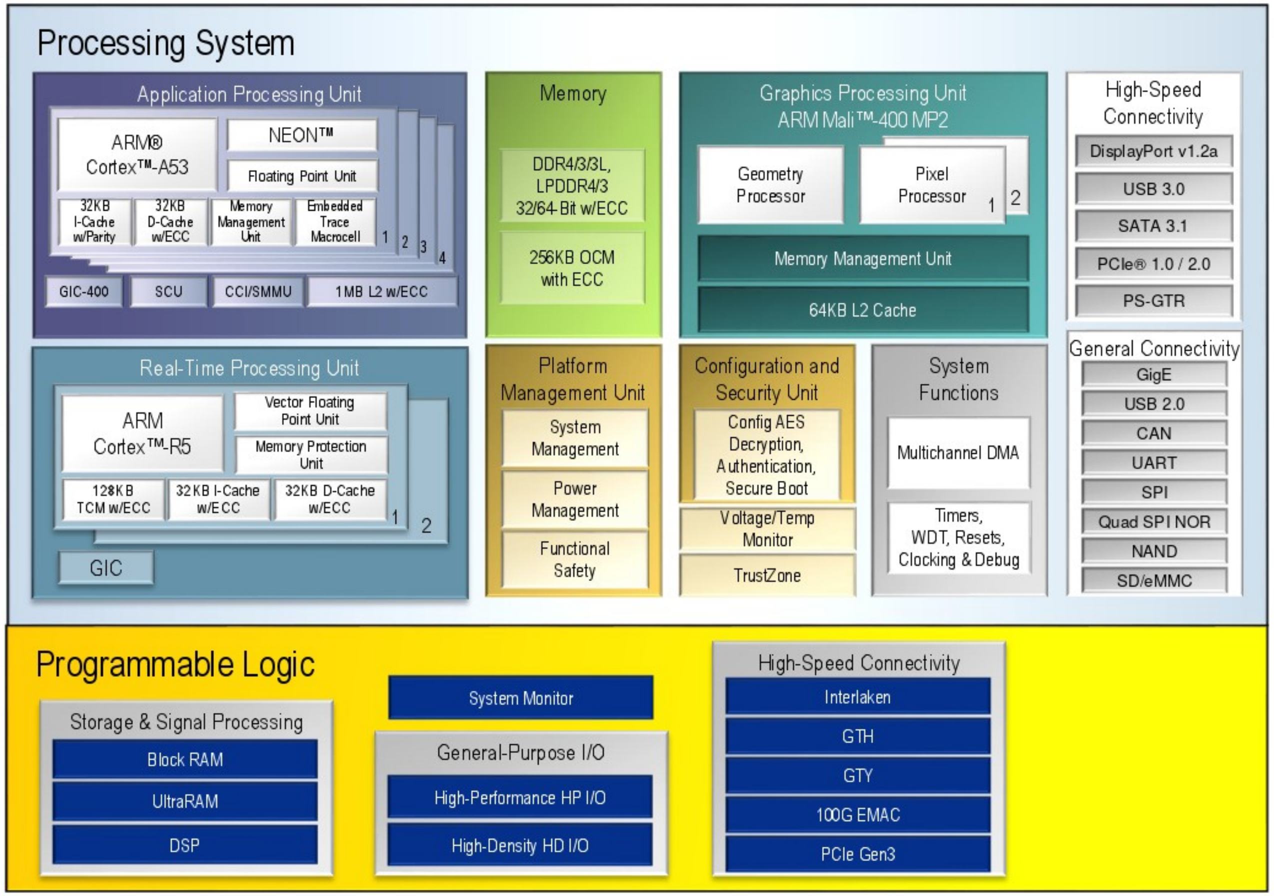
In Stock: 0

Quantity	Unit price
To 9	€799.00 (950.81 € gross) *
From 10	€719.10 (855.73 € gross) *
From 25	€679.15 (808.19 € gross) *
From 50	€639.20 (760.65 € gross) *
From 100	€599.25 (713.11 € gross) *
From 250	€559.30 (665.57 € gross) *
From 500	€519.35 (618.03 € gross) *
From 1000	€479.40 (570.49 € gross) *

Product information "UltraITX+ Baseboard for Trenz Electronic TE080X UltraSOM+"

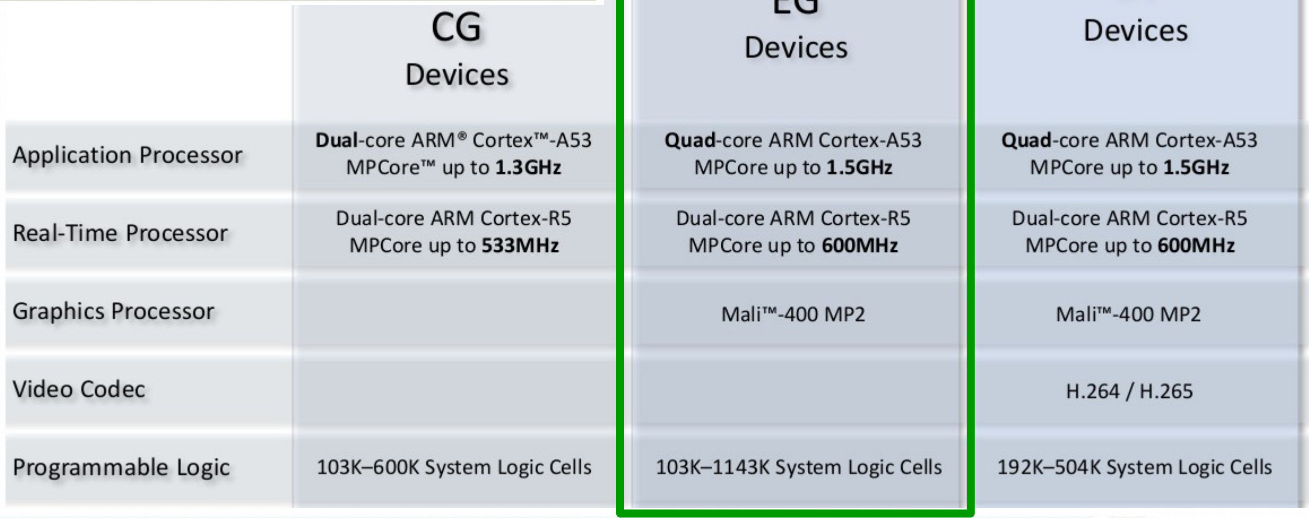
The Trenz Electronic TEBF0808 carrier board is a baseboard for the Xilinx Zynq Ultrascale+ MPSoC modules TE0803, TE0807 und TE0808, which exposes the module's B2B connector pins to accessible connectors and provides a whole range of on-board components to test and evaluate the Zynq Ultrascale+ SoMs and for developing purposes. The carrier board has a Mini-ITX form factor making it capable to be fitted into a PC enclosure. On the PC enclosure's rear and front panel, MGT interfaces and connectors are accessible, for the front panel elements there are also Intel-PC compatible headers available.

Zynq UltraScale+ Family



XILINX

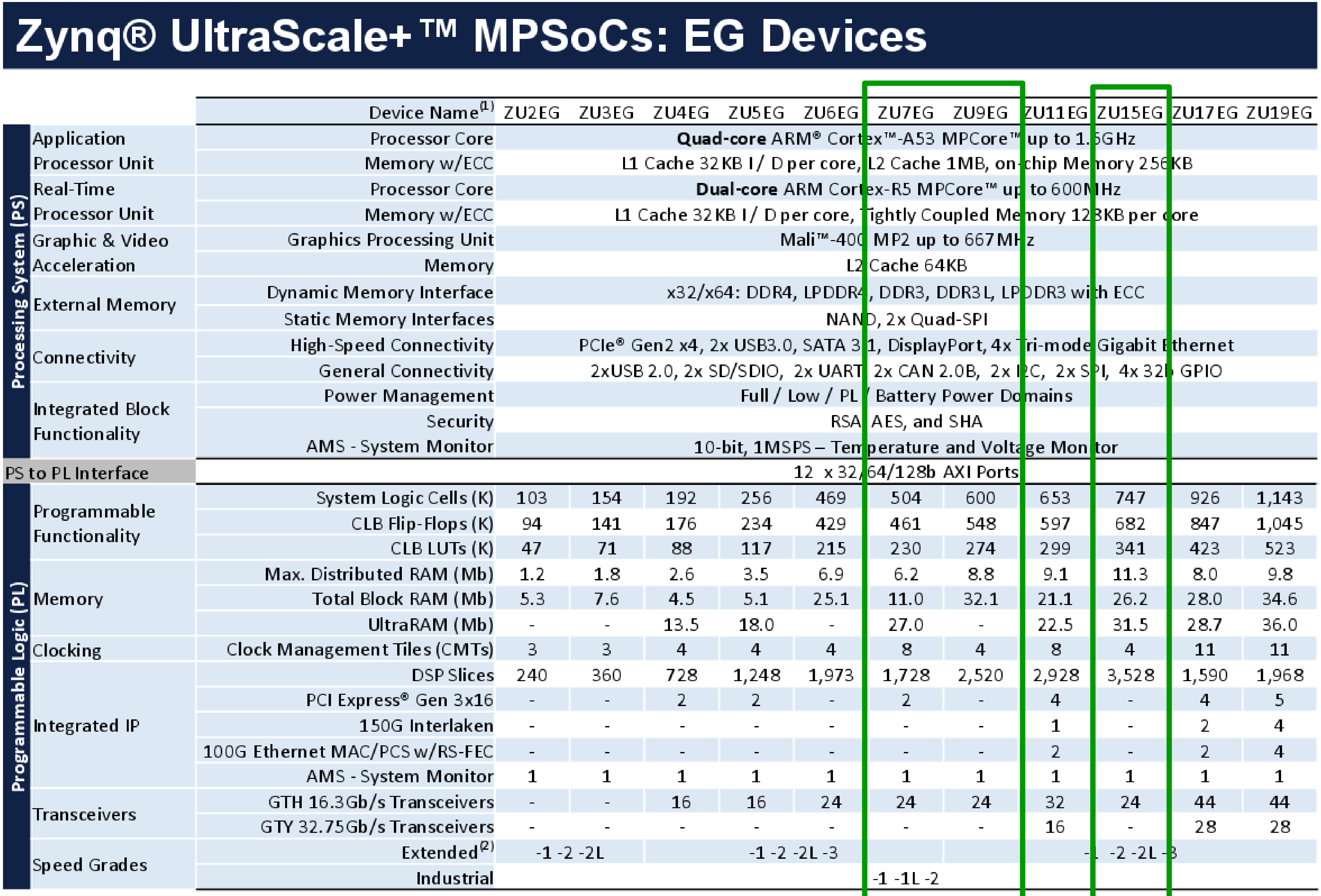
ZYNQ UltraSCALE+



	CG Devices	EG Devices	EV Devices
Application Processor	**Dual**-core ARM® Cortex™-A53 MPCore™ up to **1.3GHz**	**Quad**-core ARM Cortex-A53 MPCore up to **1.5GHz**	**Quad**-core ARM Cortex-A53 MPCore up to **1.5GHz**
Real-Time Processor	Dual-core ARM Cortex-R5 MPCore up to **533MHz**	Dual-core ARM Cortex-R5 MPCore up to **600MHz**	Dual-core ARM Cortex-R5 MPCore up to **600MHz**
Graphics Processor		Mali™-400 MP2	Mali™-400 MP2
Video Codec			H.264 / H.265
Programmable Logic	103K–600K System Logic Cells	103K–1143K System Logic Cells	192K–504K System Logic Cells

Zynq UltraScale+ MPSoC
Product Tables and Product Selection Guide

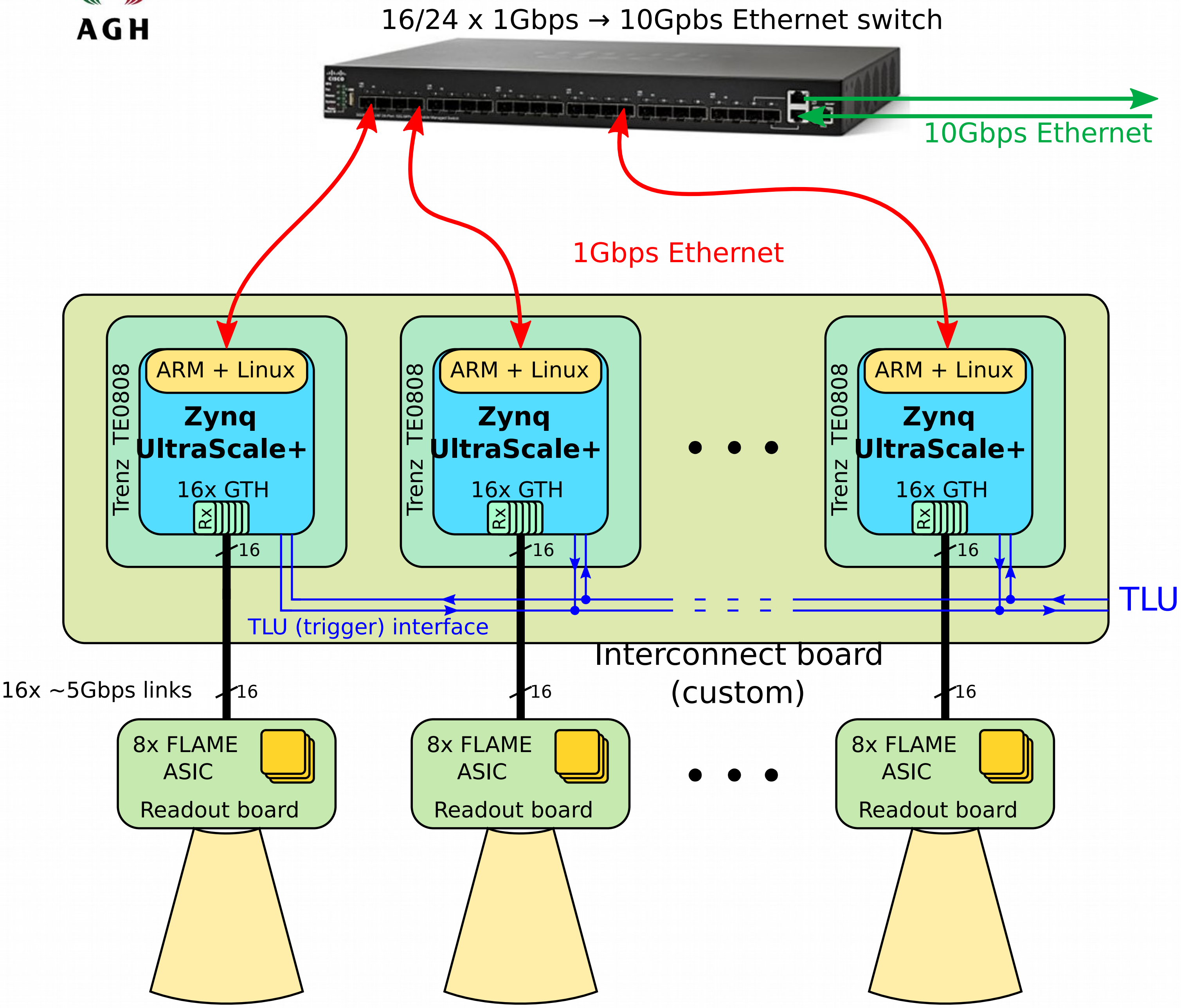
Zynq UltraScale+ Family



Zynq® UltraScale+ ™ MPSoCs: EG Devices

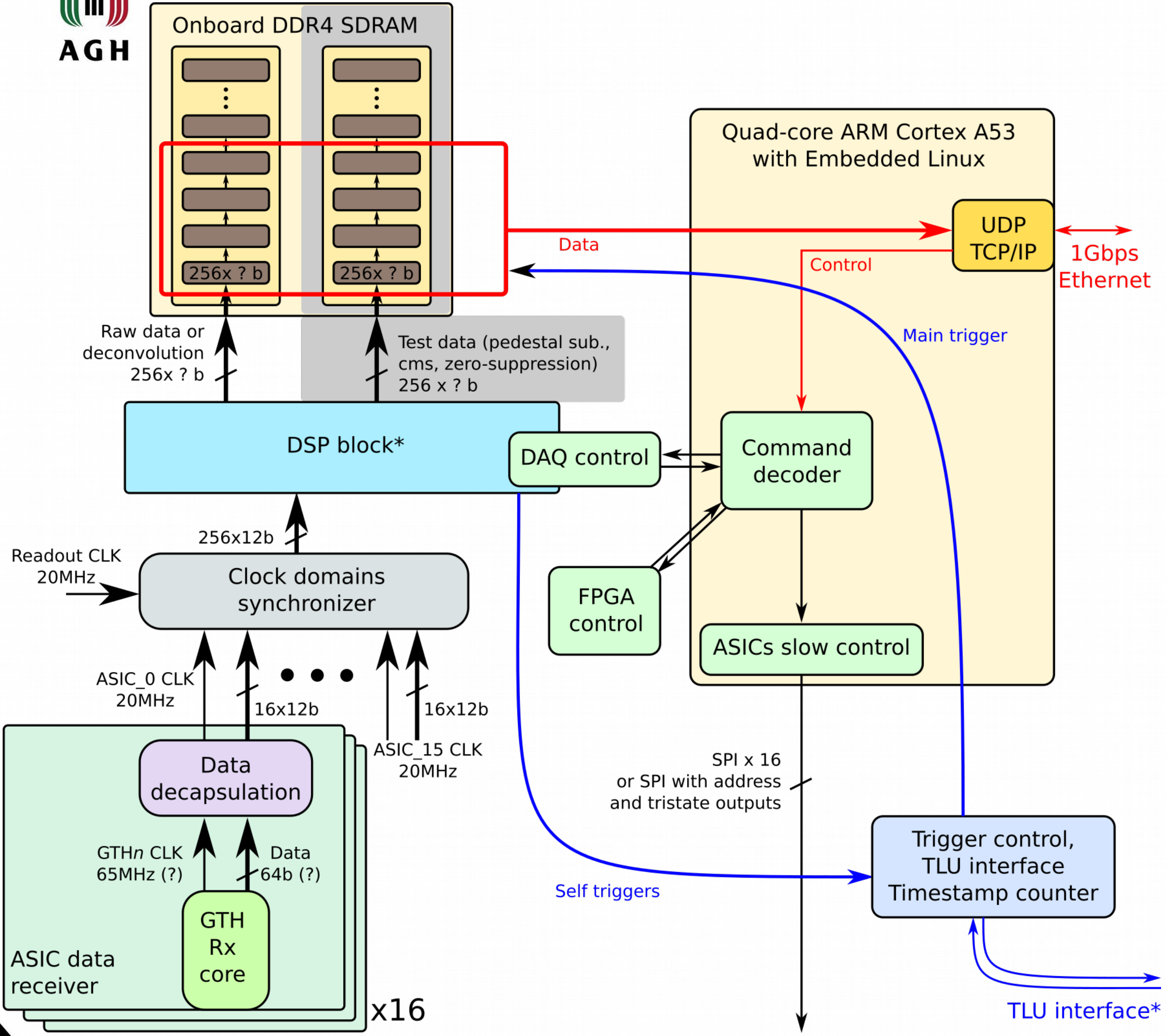
		Device Name[1]	ZU2EG	ZU3EG	ZU4EG	ZU5EG	ZU6EG	ZU7EG	ZU9EG	ZU11EG	ZU15EG	ZU17EG	ZU19EG
Processing System (PS)	Application Processor Unit	Processor Core	**Quad-core** ARM® Cortex™-A53 MPCore™ up to 1.5GHz										
		Memory w/ECC	L1 Cache 32KB I / D per core, L2 Cache 1MB, on-chip Memory 256KB										
	Real-Time Processor Unit	Processor Core	**Dual-core** ARM Cortex-R5 MPCore™ up to 600MHz										
		Memory w/ECC	L1 Cache 32KB I / D per core, Tightly Coupled Memory 128KB per core										
	Graphic & Video Acceleration	Graphics Processing Unit	Mali™-400 MP2 up to 667MHz										
		Memory	L2 Cache 64KB										
	External Memory	Dynamic Memory Interface	x32/x64: DDR4, LPDDR4, DDR3, DDR3L, LPDDR3 with ECC										
		Static Memory Interfaces	NAND, 2x Quad-SPI										
	Connectivity	High-Speed Connectivity	PCIe® Gen2 x4, 2x USB3.0, SATA 3.1, DisplayPort, 4x Tri-mode Gigabit Ethernet										
		General Connectivity	2xUSB 2.0, 2x SD/SDIO, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO										
	Integrated Block Functionality	Power Management	Full / Low / PL / Battery Power Domains										
		Security	RSA, AES, and SHA										
		AMS - System Monitor	10-bit, 1MSPS – Temperature and Voltage Monitor										
PS to PL Interface			12 x 32/64/128b AXI Ports										
Programmable Logic (PL)	Programmable Functionality	System Logic Cells (K)	103	154	192	256	469	504	600	653	747	926	1,143
		CLB Flip-Flops (K)	94	141	176	234	429	461	548	597	682	847	1,045
		CLB LUTs (K)	47	71	88	117	215	230	274	299	341	423	523
	Memory	Max. Distributed RAM (Mb)	1.2	1.8	2.6	3.5	6.9	6.2	8.8	9.1	11.3	8.0	9.8
		Total Block RAM (Mb)	5.3	7.6	4.5	5.1	25.1	11.0	32.1	21.1	26.2	28.0	34.6
		UltraRAM (Mb)	-	-	13.5	18.0	-	27.0	-	22.5	31.5	28.7	36.0
	Clocking	Clock Management Tiles (CMTs)	3	3	4	4	4	8	4	8	4	11	11
	Integrated IP	DSP Slices	240	360	728	1,248	1,973	1,728	2,520	2,928	3,528	1,590	1,968
		PCI Express® Gen 3x16	-	-	2	2	-	2	-	4	-	4	5
		150G Interlaken	-	-	-	-	-	-	-	1	-	2	4
		100G Ethernet MAC/PCS w/RS-FEC	-	-	-	-	-	-	-	2	-	2	4
		AMS - System Monitor	1	1	1	1	1	1	1	1	1	1	1
	Transceivers	GTH 16.3Gb/s Transceivers	-	-	16	16	24	24	24	32	24	44	44
		GTY 32.75Gb/s Transceivers	-	-	-	-	-	-	-	16	-	28	28
	Speed Grades	Extended[2]	-1 -2 -2L		-1 -2 -2L -3					-1 -2 -2L -3			
		Industrial	-1 -1L -2										
								1070	**1189**		**1605**	**EUR**	

Proposed DAQ architecture



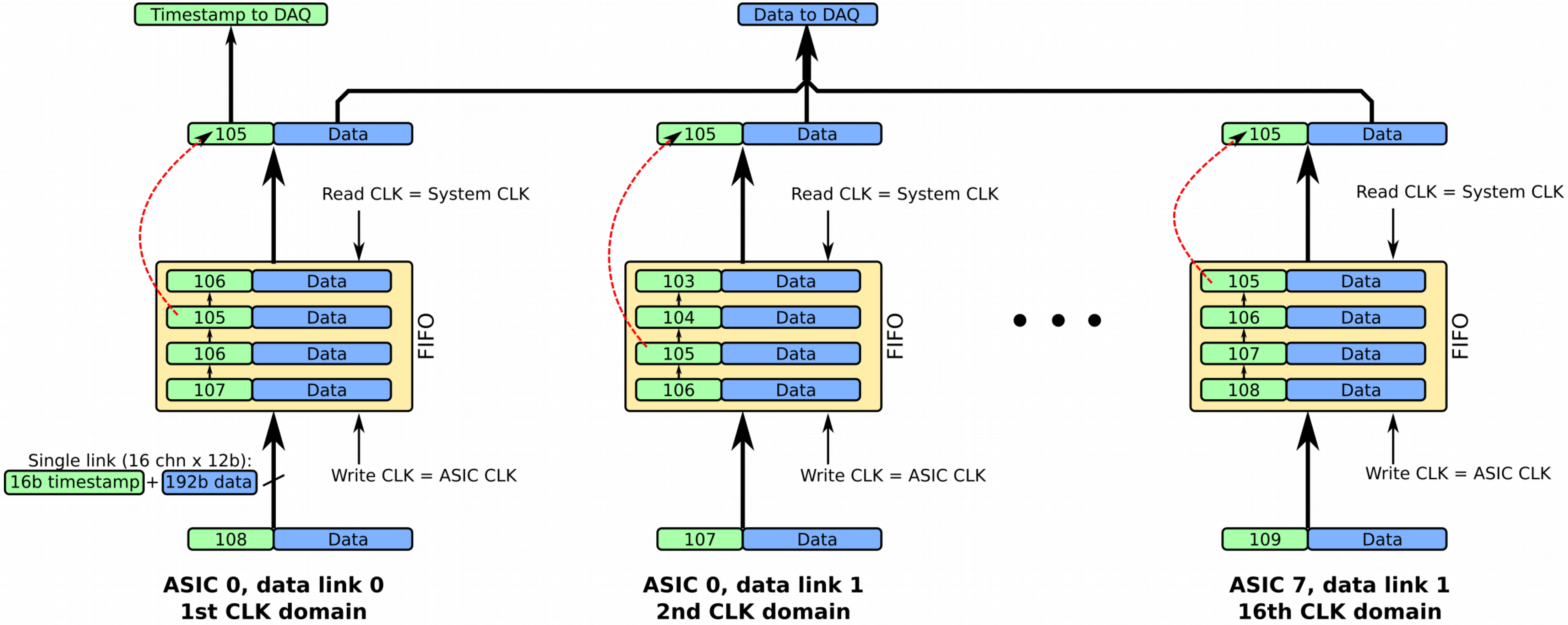
- 8 FLAME ASICs / plane = 256 channels = 16 data links (2 links per ASIC)
- New Trenz Electronic modules with Zynq UltraScale+ FPGAs available from the end of this year.
- 16 GTH transceivers / FPGA = 1 FPGA / plane
- Integrated ARM + embedded linux = 1Gbps Ethernet “for free”
- Simple Ethernet switch used as data concentrator
- One drawback – TLU (trigger) interface and timestamp synchronization not so straightforward…

Proposed DAQ architecture



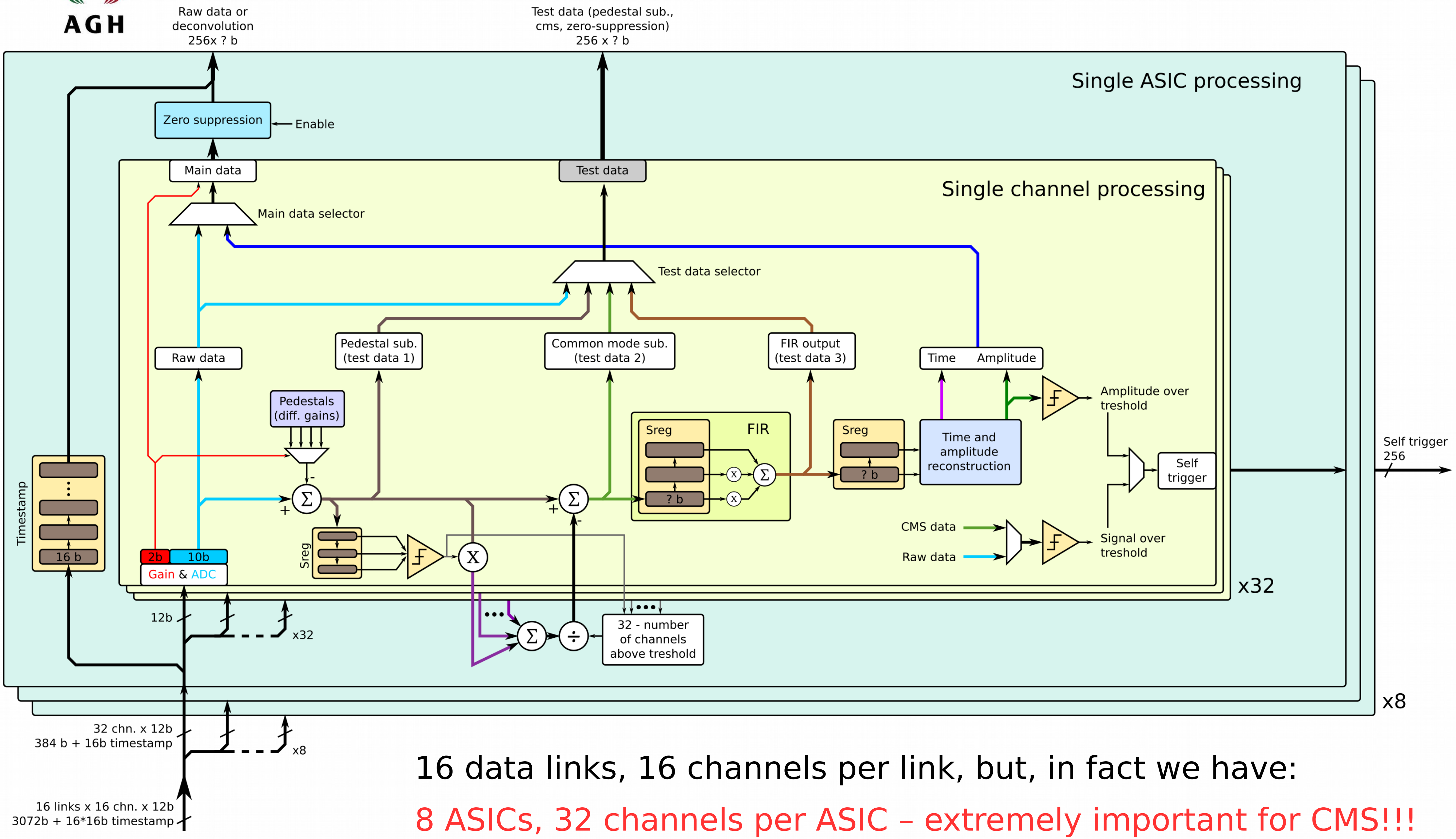
- Data from 8 ASICs (16 links) received by GTH transceivers and decapsulated
- Clock domains (16 receivers = 16 domains) synchronized with main CLK (see next)
- DSP (pedestal, cm subtraction, FIR, deconvolution, ZS)
- Data feed by FPGA logic into onboard RAM
- On trigger data read out by ARM and send out through 1 Gbps ethernet
- DAQ and ASICs slow control – by software on ARM (linux)

Clock domains synchronizer



- Clock domains synchronizer combines samples with the same timestamp and synchronizes clock phases
- If one ASIC / data link is dead, the synchronizer should build incomplete sample and inform DAQ that one data channel is missing and should not be processed, especially in cms procedure

DSP scheme



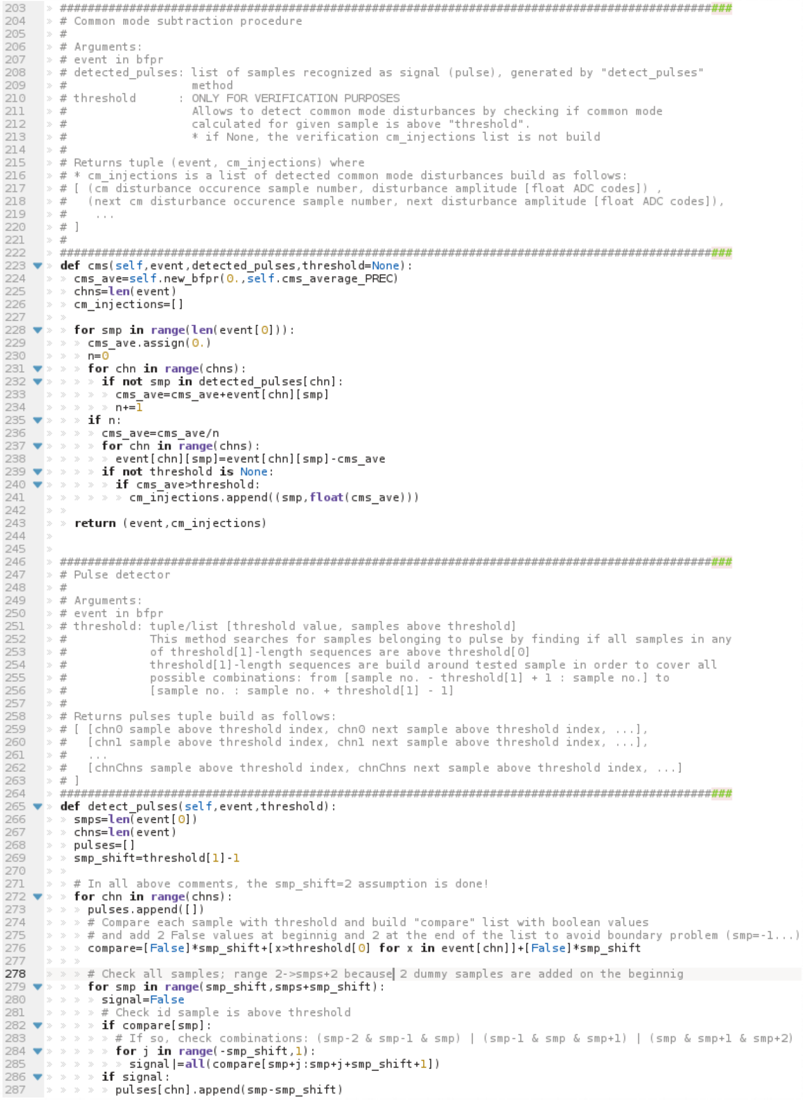
16 data links, 16 channels per link, but, in fact we have:

8 ASICs, 32 channels per ASIC – extremely important for CMS!!!

DSP simulations software

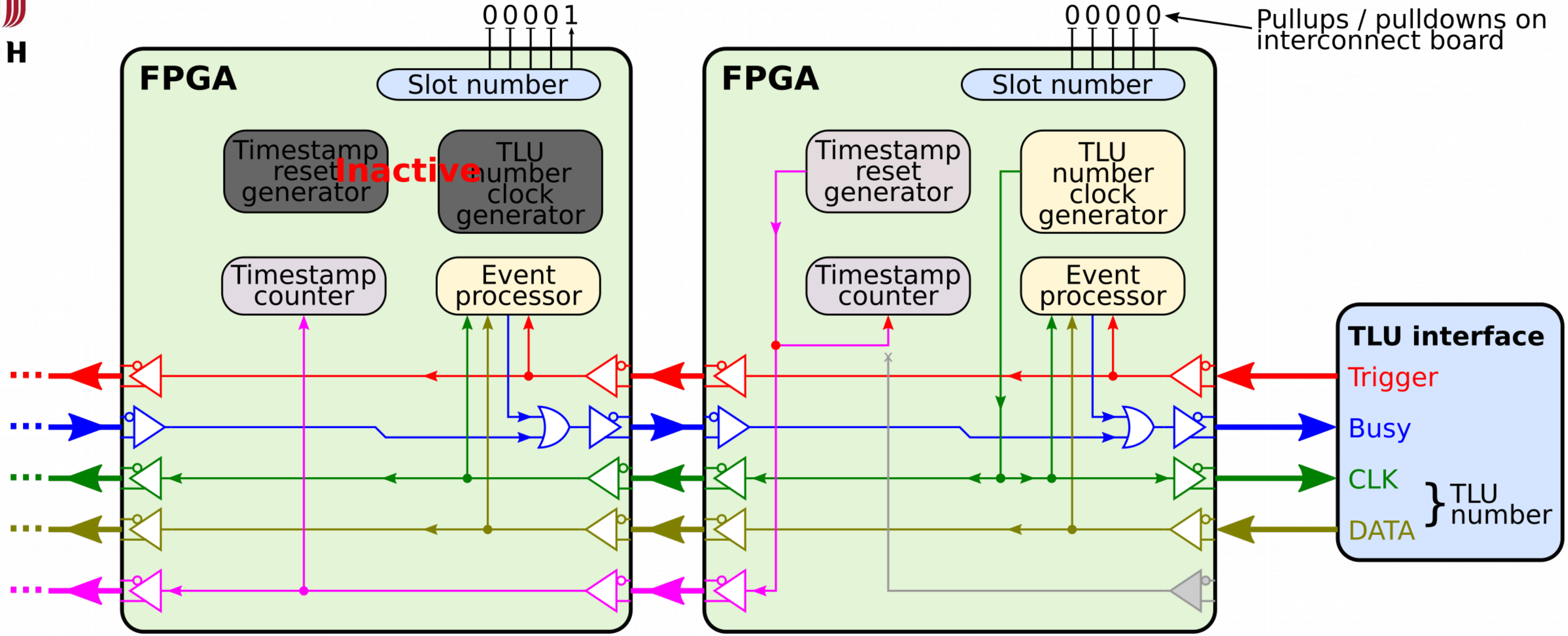
Python-based software simulating whole ASIC and DAQ chain is almost done:

- FLAME data generator is done
    - Generates “real” data – pedestal with noise, randomly distributed CM disturbances and randomly generated CR-RC pulses
- DAQ based on real binary fixed point arithmetic (with overflow supervision) is done
    - Pedestal subtraction
    - Signal detection for CMS (this is a little tricky…)
    - Common mode subtraction (CMS)
    - FIR (deconvolution filter)
    - Signal detection in FIR output samples for amplitude reconstruction (gives also zero suppression)
    - Amplitude and time reconstruction
- Verification class – work in progress
- Simulations of complete chain to determinate required fixed point resolution – not done
- We should schedule additional meeting for DSP details…



```
#################################################################################################
# Common mode subtraction procedure
#
# Arguments:
# event in bfpr
# detected_pulses: list of samples recognized as signal (pulse), generated by "detect_pulses"
#                  method
# threshold      : ONLY FOR VERIFICATION PURPOSES
#                  Allows to detect common mode disturbances by checking if common mode
#                  calculated for given sample is above "threshold".
#                  * if None, the verification cm_injections list is not build
#
# Returns tuple (event, cm_injections) where
# * cm_injections is a list of detected common mode disturbances build as follows:
# [ (cm disturbance occurence sample number, disturbance amplitude [float ADC codes]) ,
#   (next cm disturbance occurence sample number, next disturbance amplitude [float ADC codes]),
#    ...
# ]
#
#################################################################################################
def cms(self,event,detected_pulses,threshold=None):
    cms_ave=self.new_bfpr(0.,self.cms_average_PREC)
    chns=len(event)
    cm_injections=[]

    for smp in range(len(event[0])):
        cms_ave.assign(0.)
        n=0
        for chn in range(chns):
            if not smp in detected_pulses[chn]:
                cms_ave=cms_ave+event[chn][smp]
                n+=1
        if n:
            cms_ave=cms_ave/n
            for chn in range(chns):
                event[chn][smp]=event[chn][smp]-cms_ave
            if not threshold is None:
                if cms_ave>threshold:
                    cm_injections.append((smp,float(cms_ave)))

    return (event,cm_injections)


#################################################################################################
# Pulse detector
#
# Arguments:
# event in bfpr
# threshold: tuple/list [threshold value, samples above threshold]
#            This method searches for samples belonging to pulse by finding if all samples in any
#            of threshold[1]-length sequences are above threshold[0]
#            threshold[1]-length sequences are build around tested sample in order to cover all
#            possible combinations: from [sample no. - threshold[1] + 1 : sample no.] to
#            [sample no. : sample no. + threshold[1] - 1]
#
# Returns pulses tuple build as follows:
# [ [chn0 sample above threshold index, chn0 next sample above threshold index, ...],
#   [chn1 sample above threshold index, chn1 next sample above threshold index, ...],
#   ...
#   [chnChns sample above threshold index, chnChns next sample above threshold index, ...]
# ]
#################################################################################################
def detect_pulses(self,event,threshold):
    smps=len(event[0])
    chns=len(event)
    pulses=[]
    smp_shift=threshold[1]-1

    # In all above comments, the smp_shift=2 assumption is done!
    for chn in range(chns):
        pulses.append([])
        # Compare each sample with threshold and build "compare" list with boolean values
        # and add 2 False values at beginnig and 2 at the end of the list to avoid boundary problem (smp=-1...)
        compare=[False]*smp_shift+[x>threshold[0] for x in event[chn]]+[False]*smp_shift

        # Check all samples; range 2->smps+2 because 2 dummy samples are added on the beginnig
        for smp in range(smp_shift,smps+smp_shift):
            signal=False
            # Check id sample is above threshold
            if compare[smp]:
                # If so, check combinations: (smp-2 & smp-1 & smp) | (smp-1 & smp & smp+1) | (smp & smp+1 & smp+2)
                for j in range(-smp_shift,1):
                    signal|=all(compare[smp+j:smp+j+smp_shift+1])
            if signal:
                pulses[chn].append(smp-smp_shift)
```

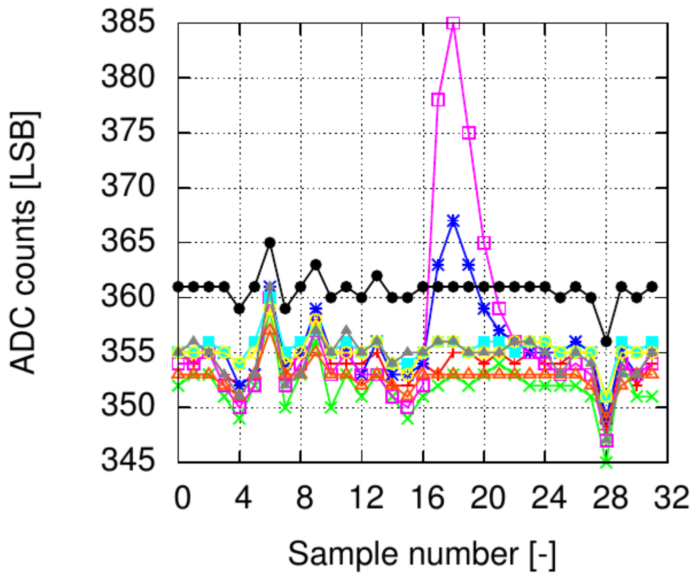
TLU interface idea



- Since LVDS/CML standard is used for TLU signals, FPGA inputs cannot be connected in parallel. Solution: pass (wire) through each FPGA with conversion CML → single ended (inside FPGA) → CML
- Trigger and TLU number data passed to all FPGAs; busy ORed inside each FPGA with busy signal from previous
- FPGA in slot **0** responsible for generating the TLU number read clock → distributed to TLU and all other FPGAs
- FPGA in slot **0** responsible for generating timestamp reset signal (on demand from DAQ control software)
- Timestamp reset and TLU number clock generators inactive in all FPGAs with slot number **!= 0**
- Slots numbered by combination of pullups / pulldowns on interconnect board
- Exactly the same firmware in each FPGA – behavior determined only by slot number → miniboards with FPGA can be replaced / exchanged between slots without any firmware change

System bandwidth

- Old DAQ scheme – 16/32 raw ADC samples (10b) per channel per each trigger (event)
- New DAQ (with amplitude reconstruction) – two 16b(?) values (time, amplitude) per channels per event. *Even less with ZS, dependently on occupancy*



- Assuming 32 raw samples / event, 12b data (10b ADC + 2b gain):
  - 32 × 12b × 256 channels = 12 kB / event / plane for raw data
  - 1 × 32b × 256 channels = 1 kB / event /plane for reconstruction data
- Assuming 400 Mbps as reasonable bandwidth for 1Gbps ethernet, we have
  - ~4k events per second or <49k events per second for singe plane
- Assuming 16 planes and 4 Gbps bandwidth for 10Gbps link:
  - ~2.5k events per second or ~30.5k events per second

Ethernet is not a bottleneck for DAQ

System bandwidth

The storage is a main bottleneck for DAQ

- Assuming 16 planes and 4 Gbps bandwidth for 10Gbps link:
  - ~2.5k events per second or ~30.5k events per second
- Single HDD (WD Red) – average write bandwidth ~100 MBps:
  - ~500 events per second or ~6.2k events per second – 5 times slower!
- RAID 5 (3x WD Red) – average write bandwidth ~300 Mbps:
  - ~1.5k events per second or <19k events per second – still 1.7 times slower!

But there is one more limit...

- Assuming one week long testbeam, data collected only for 70% of the time
  - For raw data **@1k events/s** we will need **81 TB!!!** of storage capacity
  - For reconstructed amplitudes **@1k events/s** only **7 TB** is needed…

We cannot work with raw data!

Open questions – system size...

- DAQ will be fully and easily scalable:
  - the interconnect board can be modular: e.g. PCB for 4 FPGA modules that can connected to another interconnect board maintaining the TLU interface and synchronization chain
  - We don not have to fill all FPGA slots, the system will work even with one FPGA
  - The 16-port ethernet switch seems to have the best price to capabilities ratio; if more than 16 detector layers will be used, a second switch would be the best choise

Questions

1) Are we going to use this DAQ only for AIDA2020 (one testbeam), or also for future developments?
2) How many layers are we going to use in testbeam next year?
   a) Especially – how many FPGA modules should we bought now?
3) How many layers are we going to use in the future – 16 / 30 / 40?

Work status and responsibilities

Firmware

- FLAME data receiver (*IFJ Krakow → ???*)
  - Done, but on different FPGA – have to be ported to UltraScale+ Zynq
- Clock domains synchroniser (*JINR Dubna*)
  - Probably done
- DSP (*JINR Dubna*)
  - Not started yet, waiting for software DSP model (*by me*)
- Control, TLU interface, timestamp synchronization, etc. (*???*)
  - Not started yet
- ARM linux, ethernet & software (*???*)
  - Not started yet

Hardware (PCBs)

- FLAME testboard, readout (detector plane) PCB (*AGH Kraków*)
  - Not started yet, waiting for final FLAME padring
- FPGA interconnection board PCB (*AGH Kraków*)
  - Not started yet, waiting for tests on Trenz Electronic module and some decisions...

Summary and questions

- New DAQ scheme proposed based on Zynq UltraScale+ modules

- Some firmware details still have to be fixed

- We should decide how to share the work on firmware

  1) **Who can help – Krakow / Dubna / somebody else?**

- We should buy more TE0808 modules on the beginning of next year

  2) **How many layers are we going to use in testbeam next year and how many in the future?**

  3) **Can anyone buy a few more modules?**