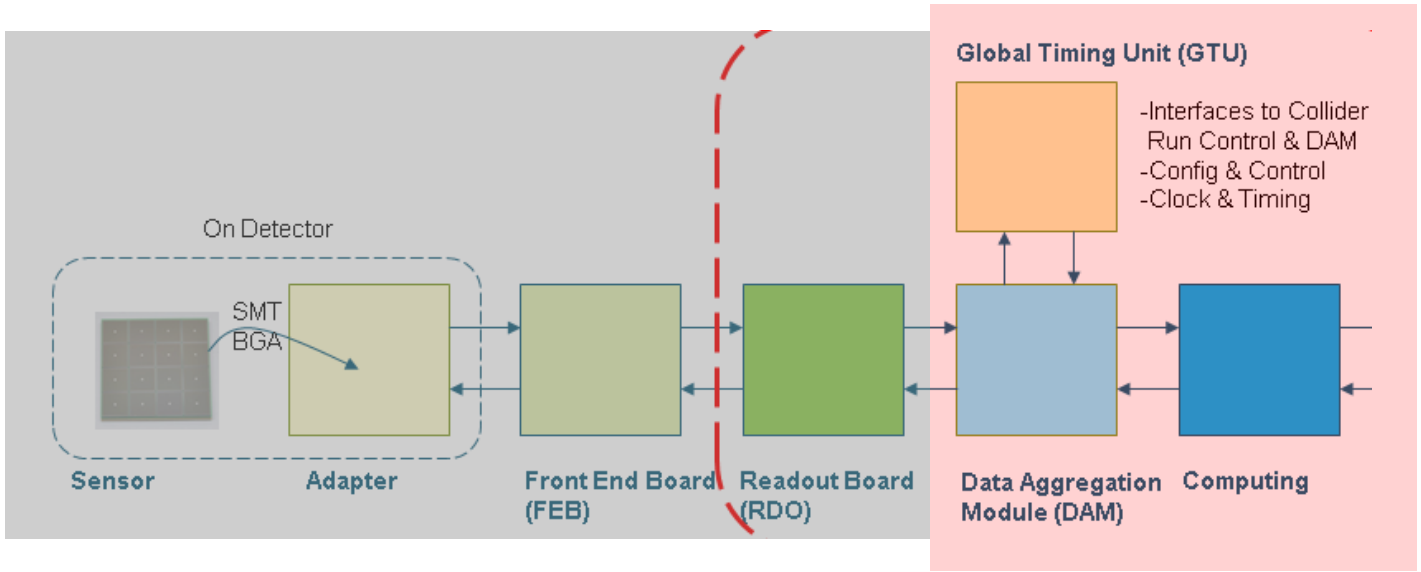
GTU updates

1. ePIC DAQ introduction
2. ePIC GTU design
3. The ePIC DAQ setup
4. Some test results
5. Works to do



1. ePIC DAQ introduction - general structure

System Clock: 19.7 MHz

GTU: Global Timing Unit

control encoding, status decoding.
The interface for machine clock source,
Data acquisition control and monitor etc.

DAM: Data Aggregation Module: (FLX155)

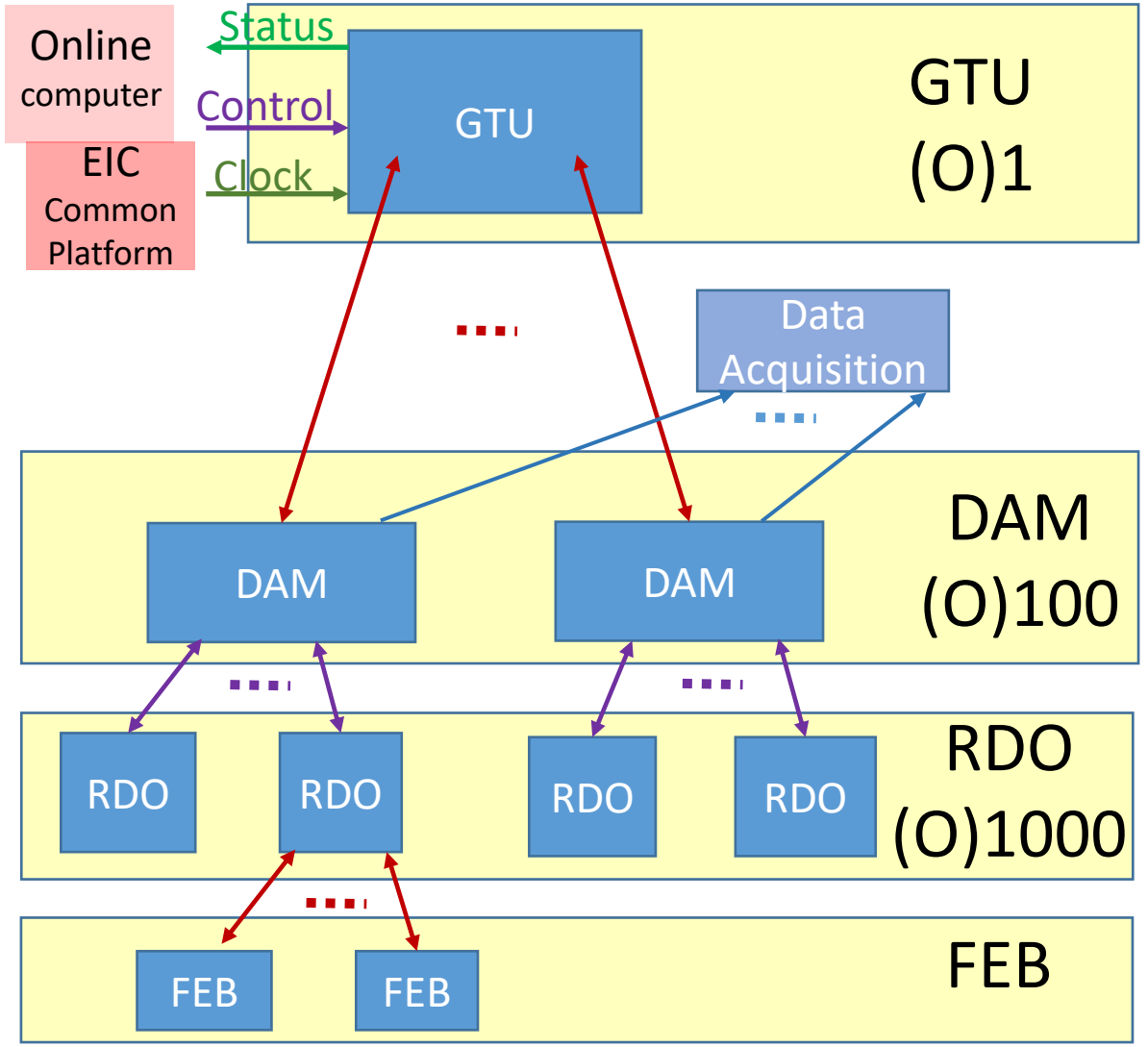
The clock/control fan out, and status/busy accumulation.
local control over the RDO boards connected, though the main function of the DAM is for Data Acquisition.

RDO: optical interface of detector ReaDOut

control decoding, status encoding.
The clock/control interface to the frontend electronics.
Data collection from Frontend boards, and data transmission to the DAM

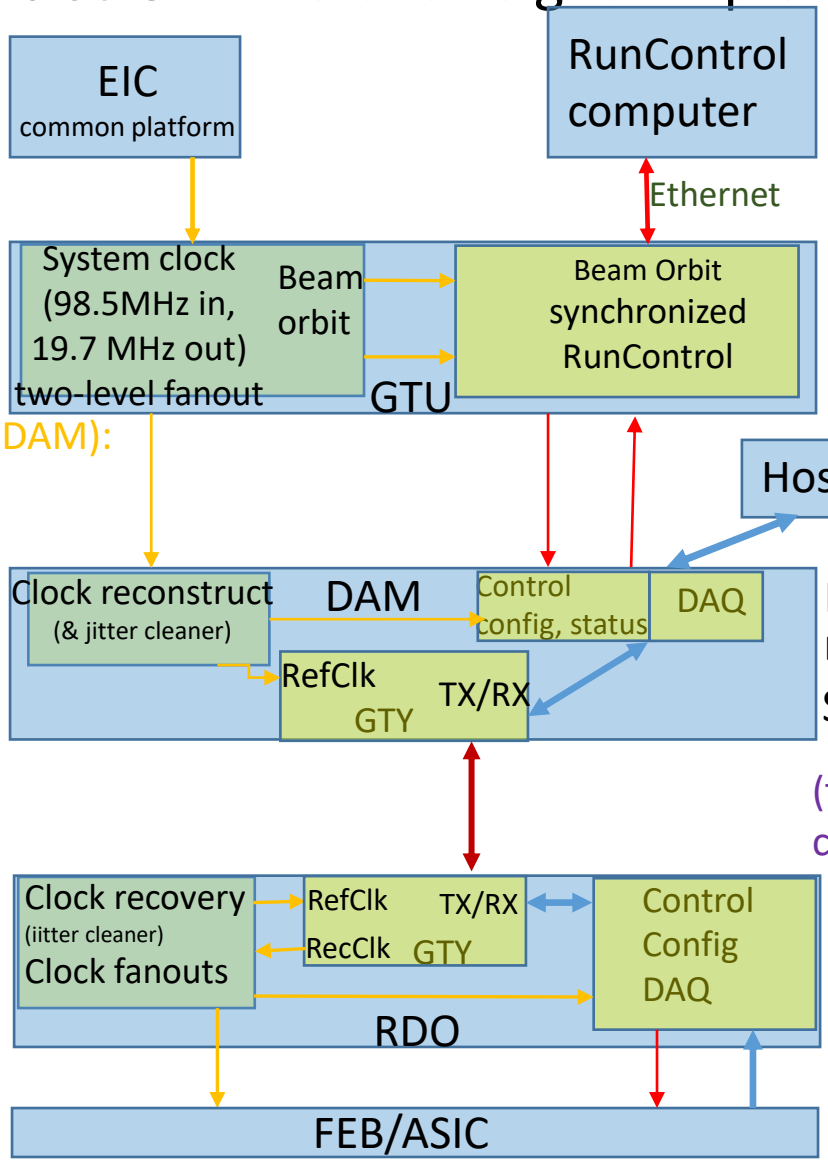
FEB:

Detector dependent Front End / ASIC carrier boards.



1. ePIC DAQ Introduction - Detailed signal implementations

1.1 Clock Distribution



PCB and fanouts:
Additive Jitter <1ps
Phase: fixed

Fiber cable (one per DAM):
Additive Jitter <1ps
Phase: fixed

PCB and fanouts:
Additive Jitter <1ps
Phase: fixed

Optic fiber cable:
Additive Jitter <1ps
Phase: fixed

GTY REC Clock:
Or LpGBT rec clock

Beam Crossing phase aligned

1.2 RunControl and DAQ

RunControl: Run_Start/Stop, Run_type, config, Reset, Synchronization, SRO time_start/stop, Throttling…
Status: Buffer busy, out of sync …

(two fibers per DAM):
control/Downlink, Data_Status/uplink
DAM to PC: 2xPCIe5x8, DAM to switch: 100 GBE

RunControl: forward or 'hijack' the GTU control;
DAM, RDO, FEB/ASIC configuration, readout backpressure
Status: Buffer, sync, etc.

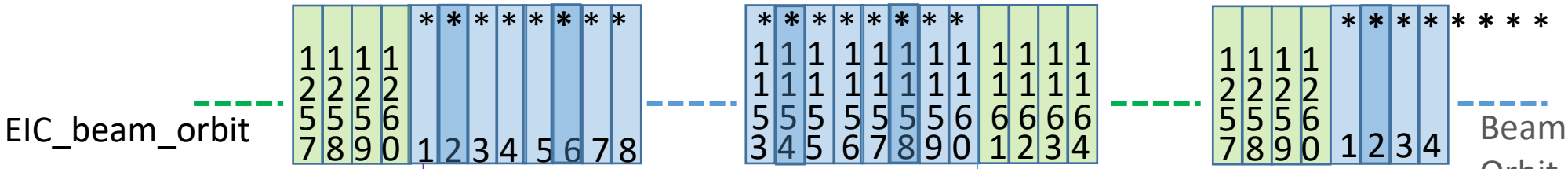
(two fibers per RDO):
control/Downlink, Data_Status/uplink

Config synchronized with beam orbit, FEB/ASIC readout and control
Status: Buffer, sync, etc.

DAQ time slice/window aligned

2. ePIC GTU Design - The ePIC clocks:

Single clock source, distributed to the whole system



EIC_BX_Clock, 98.5 MHz

GTU→DAM, 19.7 MHz
(Frequency divided by 5)
252 clock_cycles per Orbit

ePIC system clock

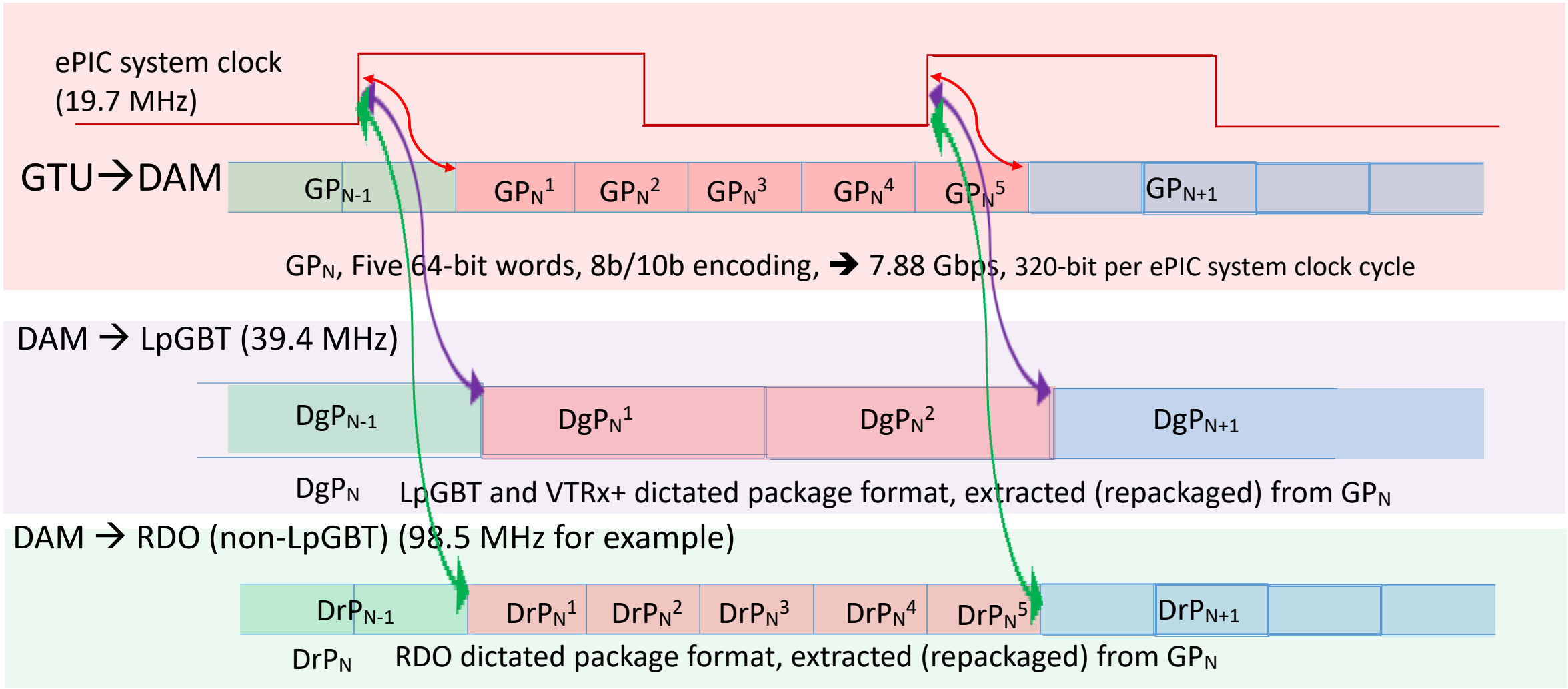
Fixed delay (divider, cable, PCB delays)

The phase (divider) is determined by the orbit information

DAM rec. Clock
39.4 MHz
98.5 MHz
others
n*252 clock_cycles per Orbit

All the reconstructed clocks are synchronized (same frequency, fixed phase)

2. ePIC GTU Design - run-control (downlink) and Data/status (uplink)



2. ePIC GTU Design - DAQ partition (multiple run-control support)

Hardware: Each MGT transceiver corresponds to one DAM. It can be 'controlled' independently.
Firmware: supports 32 run-control processes, (ID#0-31), with ID#0 and #31 reserved for global run.
Software: to be developed

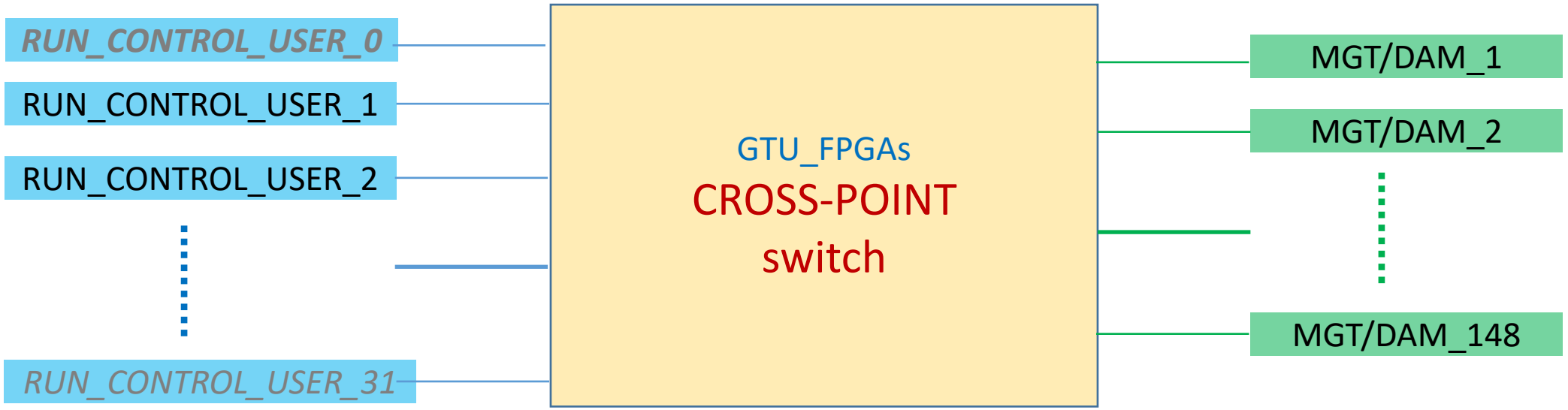
1) Run-control-N requests for a new system ID (1 to 30) by reading the sub-system (bookkeeping) register (32-bit, offset 0x200) from GTU, and set the bit#ID to '1' (means that subsystem#ID is being used);
2) Run-control loads the corresponding DAM/MGT with the sub-system ID. (the DAM/MGT with the same ID numbers belong to the same sub-system, and controlled by the run-control-N, N and ID can be different);
3) The ID will be propagated from the GTU_MGT to the corresponding DAM via the GTU_DAM downlink;
4) The DAM remembers its sub-system ID, and direct the data to run-control-N;
5) At the end of run-control-N, release the DAM/MGT by unloading the sub-system ID (reverse the step#2 above), release the subsystem ID (set bit#ID to '0' in sub-system bookkeeping register, reverse the step#1 above).

- Maximum sub-system can be running simultaneously: 30
- Number of DAMs per sub-system: min: 1, max: 148 (not to overlap with other sub-systems)

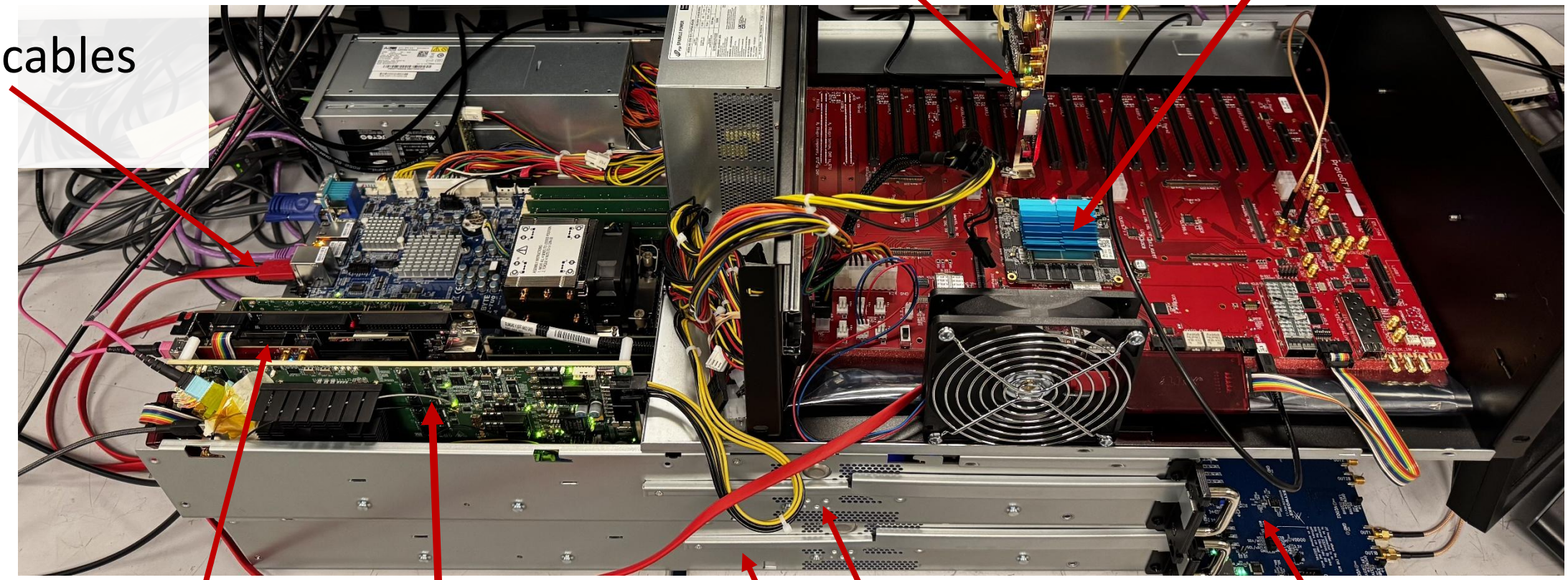
2. ePIC GTU Design - DAQ partition (multiple run-control support)

Firmware implementation:

- One 32-bit sub-system (bookkeeping) register (offset 0x200) in GTU;
- One set of Beam-crossing counters (64-bit, up to 4000 years), beam-crossing number within the orbit (11-bit, from 0 to 1159), and the first beam-crossing signal (BX0) in the orbit;
- 32 (sub system ID#0~31) sets of: orbit counters (32-bit, up to 14 hours), streaming time intervals (or windows, 14-bit at 19.7 MHz, up to 800 μs), streaming START/STOP/PAUSE etc.
- Sub-system ID tagged run control command (8-bit code, run-start, run-stop, various resets etc.), which affects the DAM with the specific ID only.



3. The ePIC DAQ setup - at BNL (510_1-231)



3. The ePIC DAQ setup - at BNL (510_1-231)

AlmaLinux 10

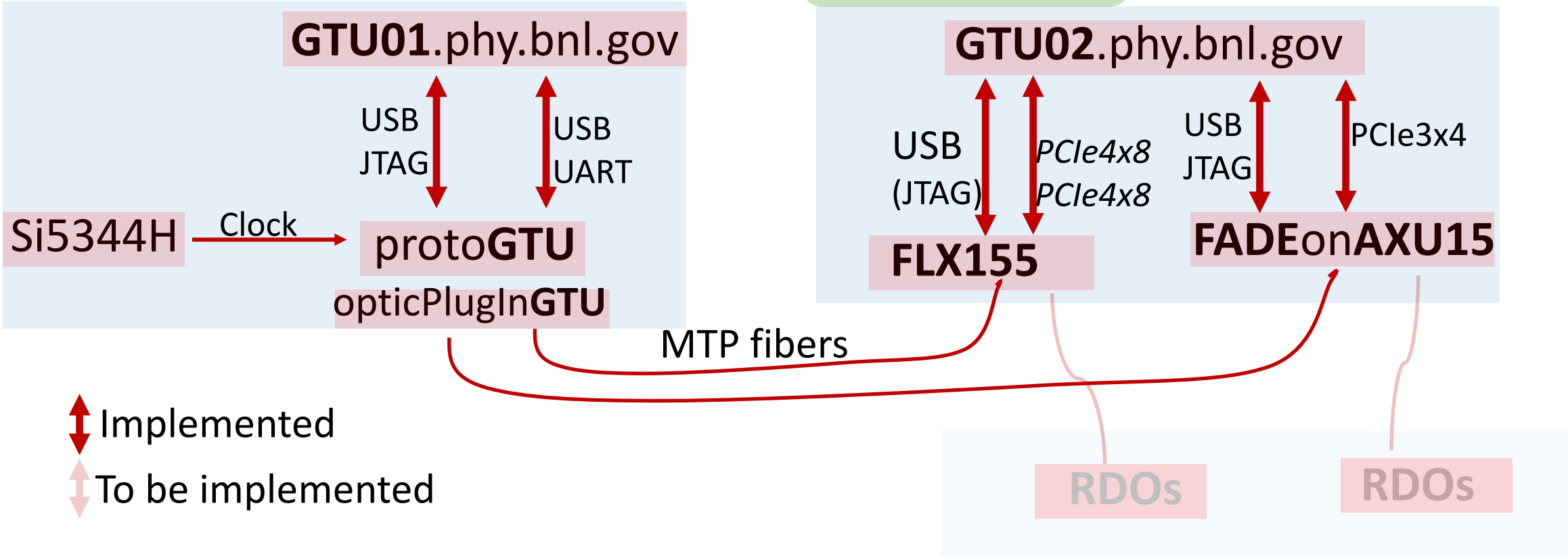
/tools/Xilinx/Vivado & Vitis **2024.1** license @maia.phy.bnl.gov

/data/PCIeControl/Axu15DAM on GTU02 xdma software/driver

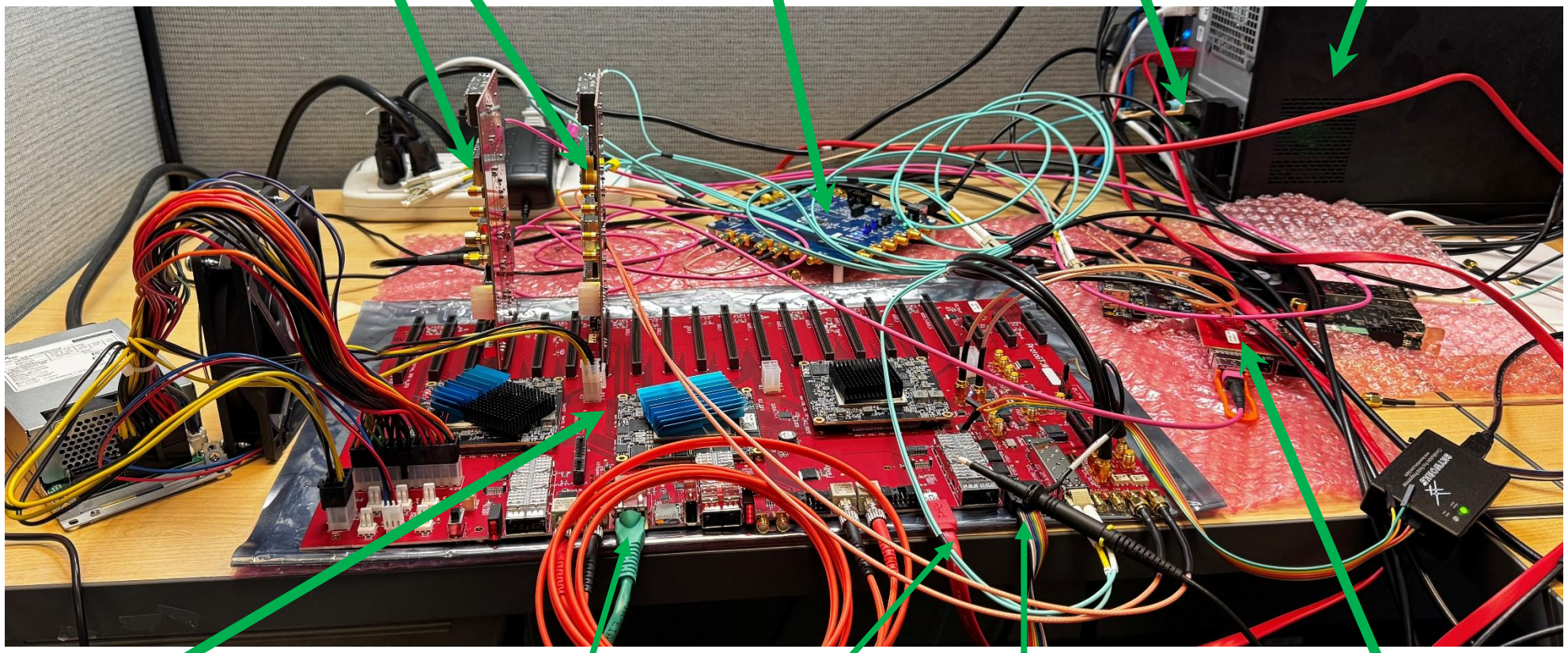
/data/fpga_v2024/GTUzu19 on GTU01 xczu19EG firmware

Axu15DAM on GTU02 xcau15p firmware

Flx155DAM on GTU02 xcvp1552 firmware



3. The ePIC DAQ setup - at JLAB (to be moved to CC/F109)



4. Some test results - Clock distribution

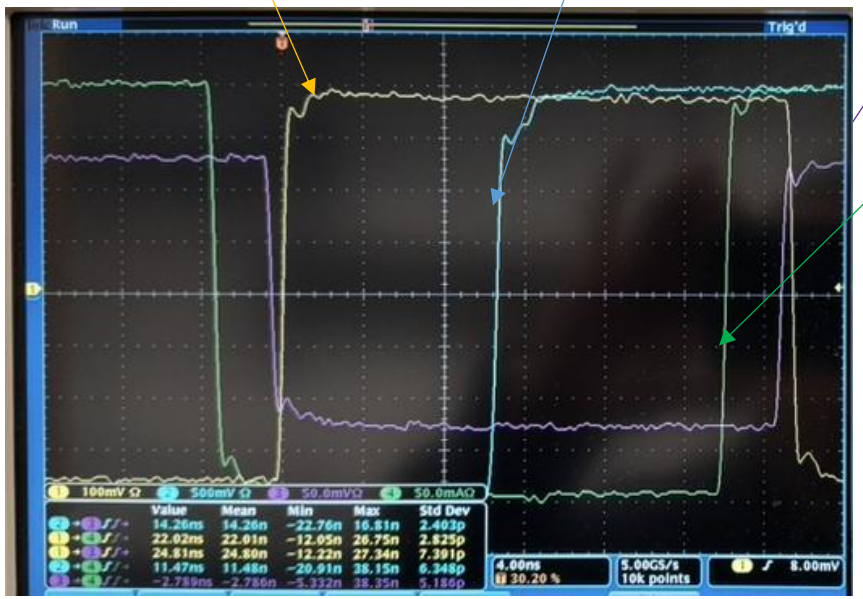
Measurement @ Mar. 27, BNL
Oscilloscope 5 GS/s ➔ 200 ps sampling

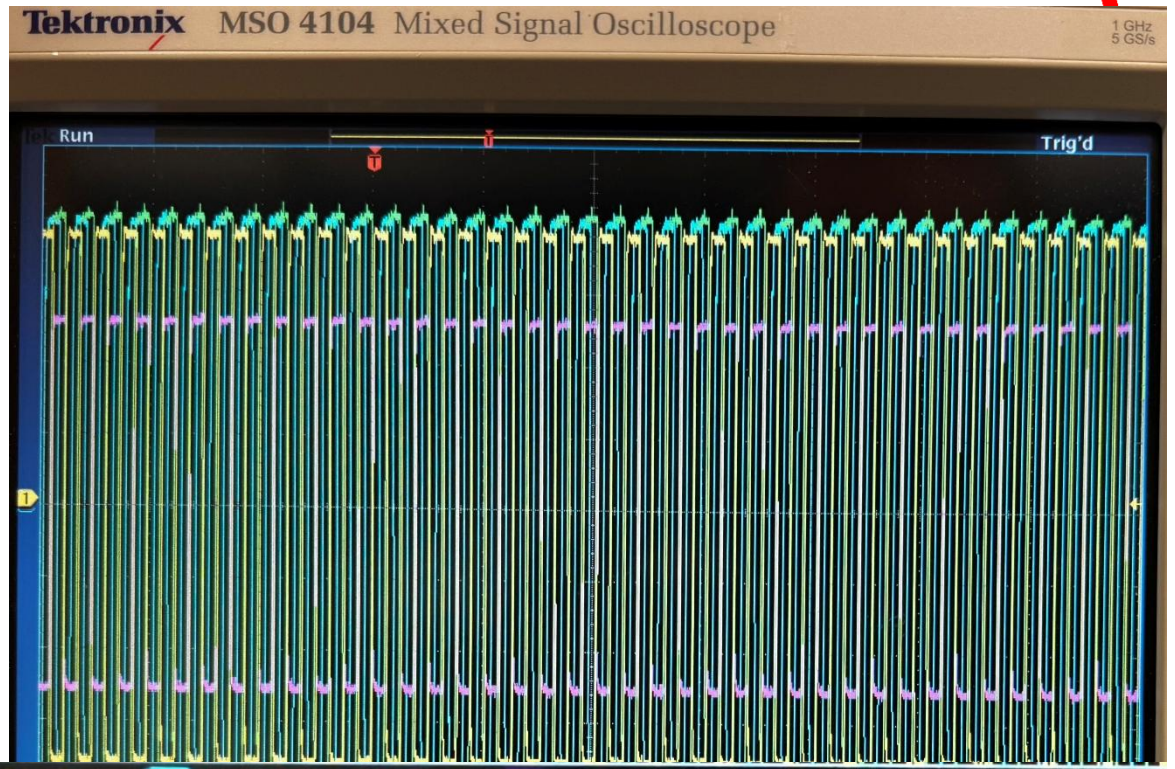
EIC clock
SI5344H
CH#1

ePIC Clock
GTU
CH#2

DAM clock
FLX155a
CH#3

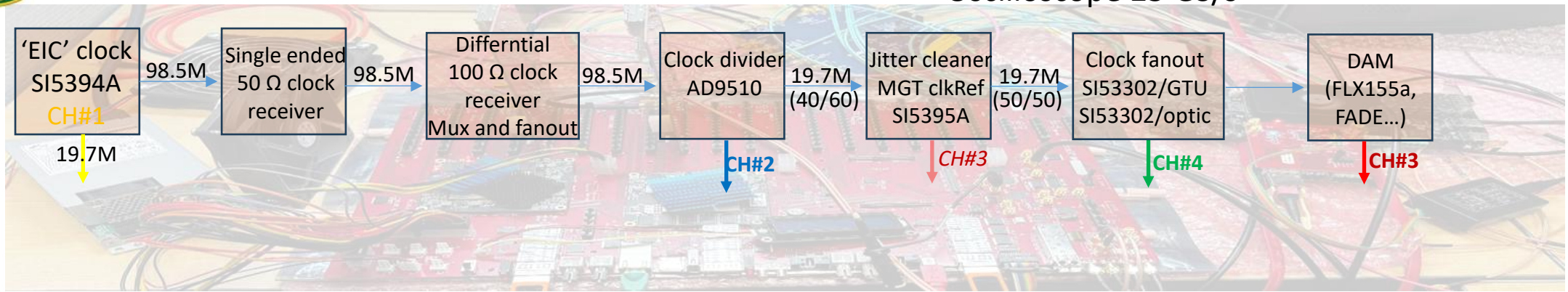
DAM clock
FADE
CH#4





4. Some test results - Clock distribution

Measurement @ July 15th, Jlab
Oscilloscope 25 GS/s



ePIC system clock (19.7 MHz) from GTU (to DAM):

- CH#4: TIE: σ = 1.9 ps, FWHM: 4.5 ps
- Latency of CH#4 vs CH#1: Phase stability:
  45.328ns ± 2.1 ps, FWHM: ~6 ps

ePIC system clock (19.7 MHz) recovered on (FADE) DAM:

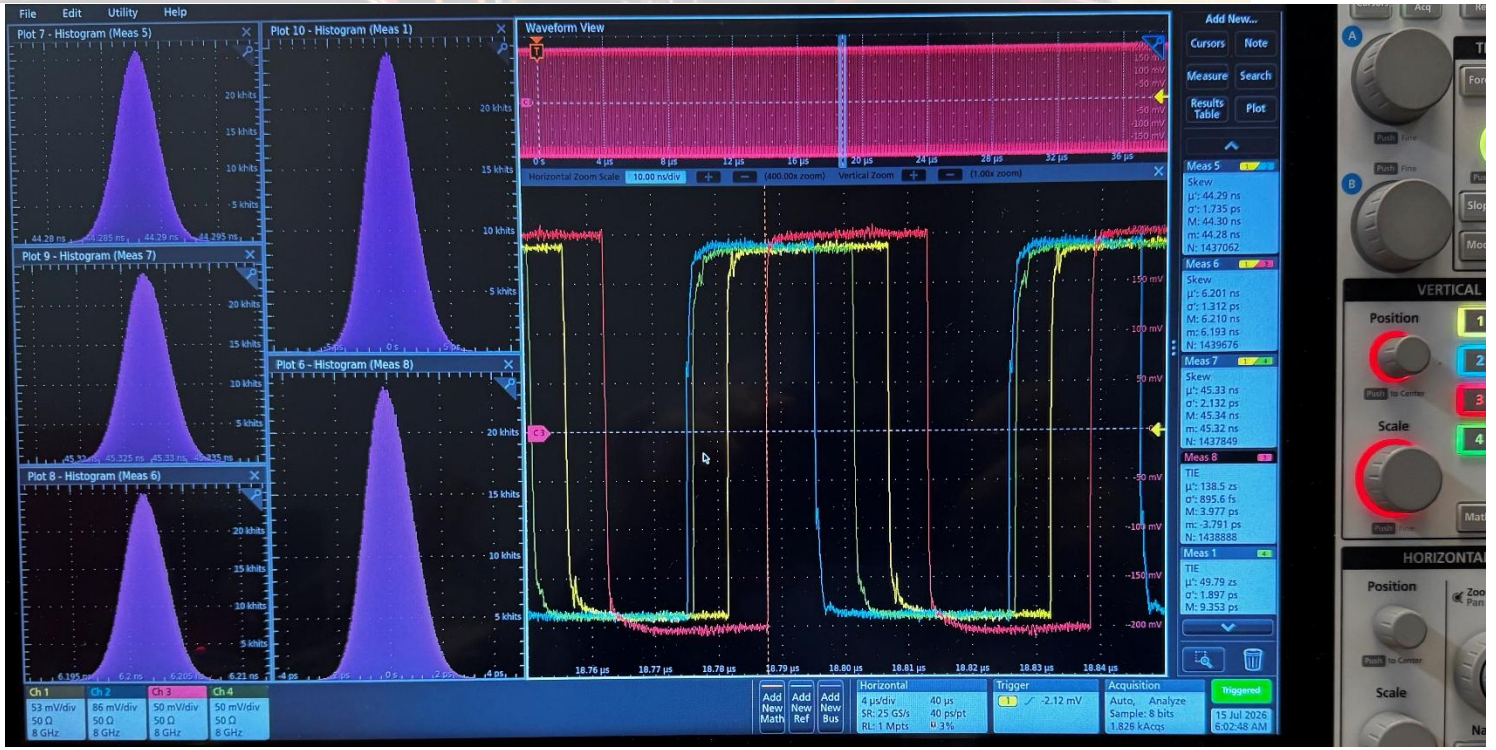
- CH#3: TIE: σ = 0.9 ps, FWHM: ~2 ps
- Latency of CH#3 vs CH#1: Phase stability:
  6.2015ns ± 1.3 ps, FWHM: ~4.5 ps

Clock divider AD9510 output:

CH#2: 44.288 ns ± 1.7 ps, FWHM: ~4.5 ps

GTU Jitter cleaner and clock synthesizer SI5395 output:

CH#3: 21.0162 ns ± 1.36 ps, FWHM: ~4 ps



The system clock divider (98.5 MHz/5) works

4. Some test results - GTU → DAM communication (Downlink)

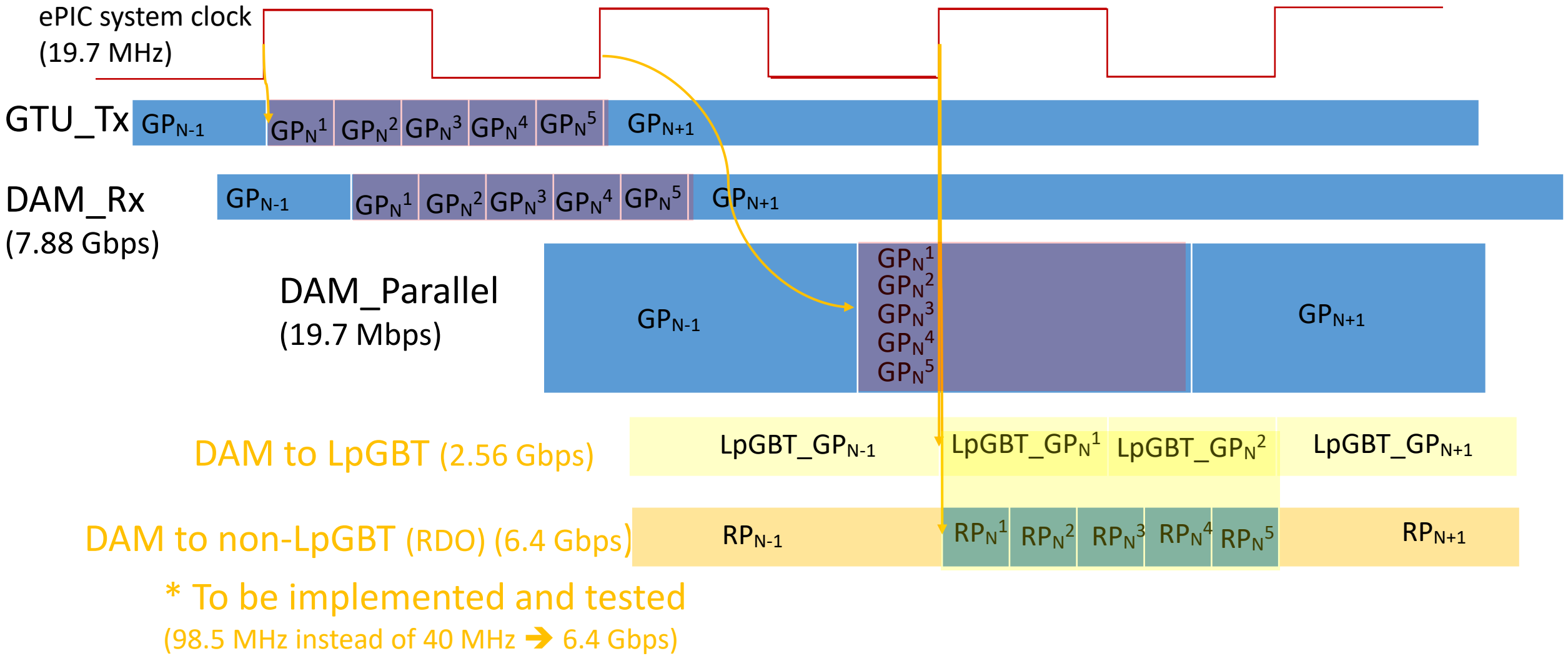
packet	One packet at 19.7 MHz, (or five 64-bit words at 98.5 MHz)					
Word#	1	2	3	4	5	five words per package
Bit(7:0)	K28.1_3C 0011111001	K28.2_5C 0011110101	K28.3_7C 0011110011	K28.4_9C 0011110010	K28.5_BC 0011111010	Bit alignment, word alignment, and packet alignment
Bit(15:8)	GTU_RC	GTU_RC	GTU_RC	GTU_RC	GTU_RC	Synchrnous Run_control codes
Bit(31:16)	BX(15:0)	OrbitC(31:0)		SRO_T(13:0)	BX(63:16), Orbit(47:16), etc. Global and subsystem counters	
Bit(36:32)	subSysID	subSysID	subSysID	subSysID	subSysID	Partition this DAM to subsystem (ID) run-control
Bit(63:37)	USER	USER	USER	USER	USER	Extras

GTU_RC: 8-bit codes for BX_0, Data Acquisition START, STOP, Sync_RESET, SRO_TIME etc.

The data are 8b/10b encoded, which results in 7.88 Gbps downlink,

This is tested with the fiber loopbacks : port_A_Tx → port_A_Rx;
port_A_Tx → port_B_Rx, port_B_Tx → port_A_Rx;

4. Some test results - GTU → DAM communication (Downlink)



* To be implemented and tested

(98.5 MHz instead of 40 MHz → 6.4 Gbps)

4. Some test results – GTU, zynqMP

ARM-cortex-A53, quad core with 4/5 GB DDR4 memory

- Gigabit Ethernet: DHCP to Jlab's DAQ network;
- UART: serial port terminal emulator;
- JTAG: ZU19EG, KU15P firmware loading, logic analyzer;
- USB2.0 with type A connector;
- 32 GB EMMC;
- SD (TF) card reader;
- PS-QSPI, PS-SPI (clock divider AD9510 setting)
- PS_I^2C-0, PS_I^2C-1 (GTU thermistor readout, QSFP status readout, EEPROM R/W…)
- Character display with I^2C control

All these features are tested, and working

5. Works to do

5.1: to test/debug the chip-to-chip communication between the zynqMP (zu19eg) and the KU15p FPGAs;

5.2: to implement the GTU downlink receiving firmware in DAM (flx155a, flx182, FADE, etc.);

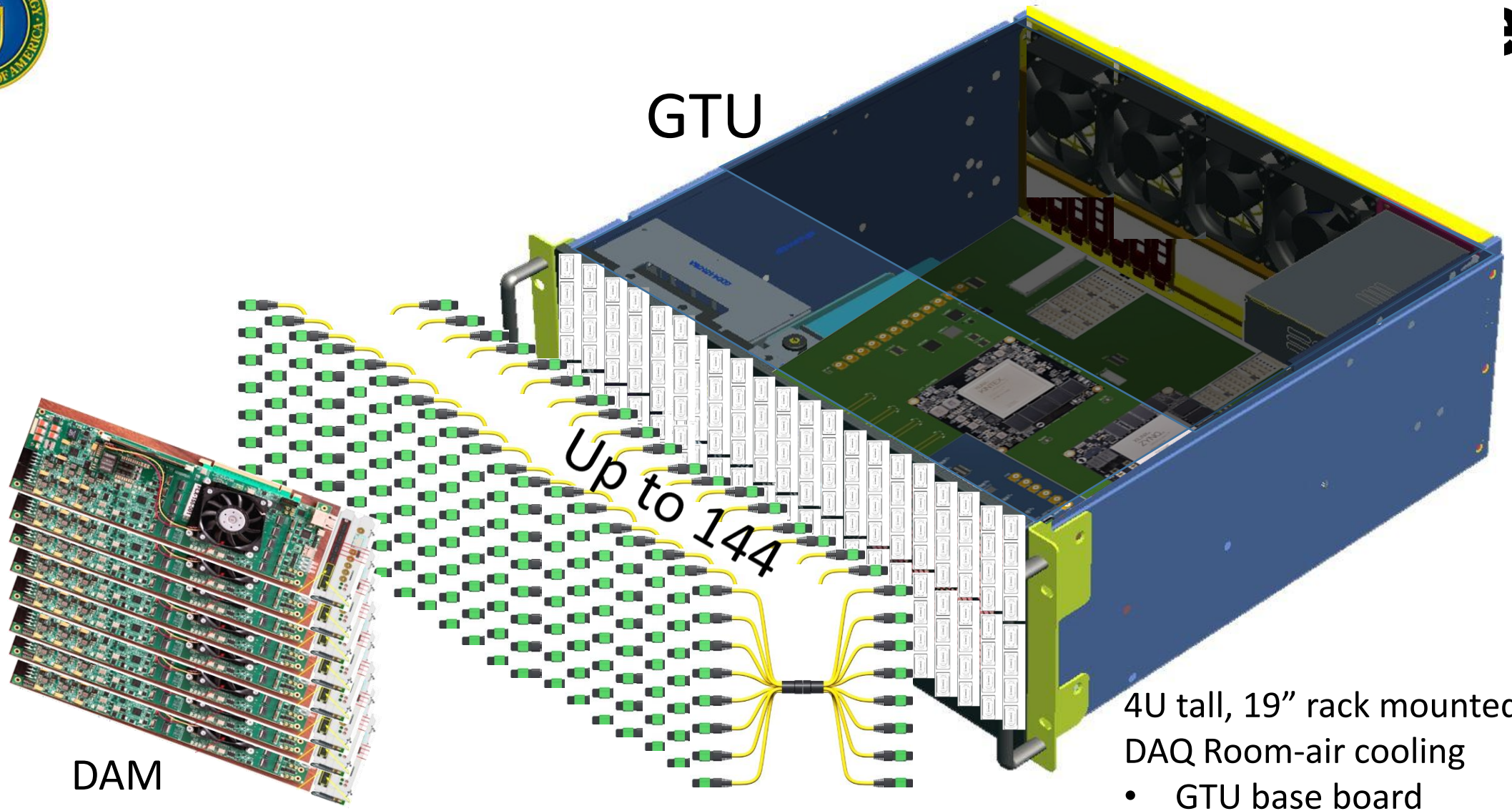
5.3: DAQ software development (on the two hardware setups);

5.4: sub-system run-control test;

5.5: to connect with real RDO (with or without LpGBT chip);

5.6: to make a clock phase monitoring board.

Thank you

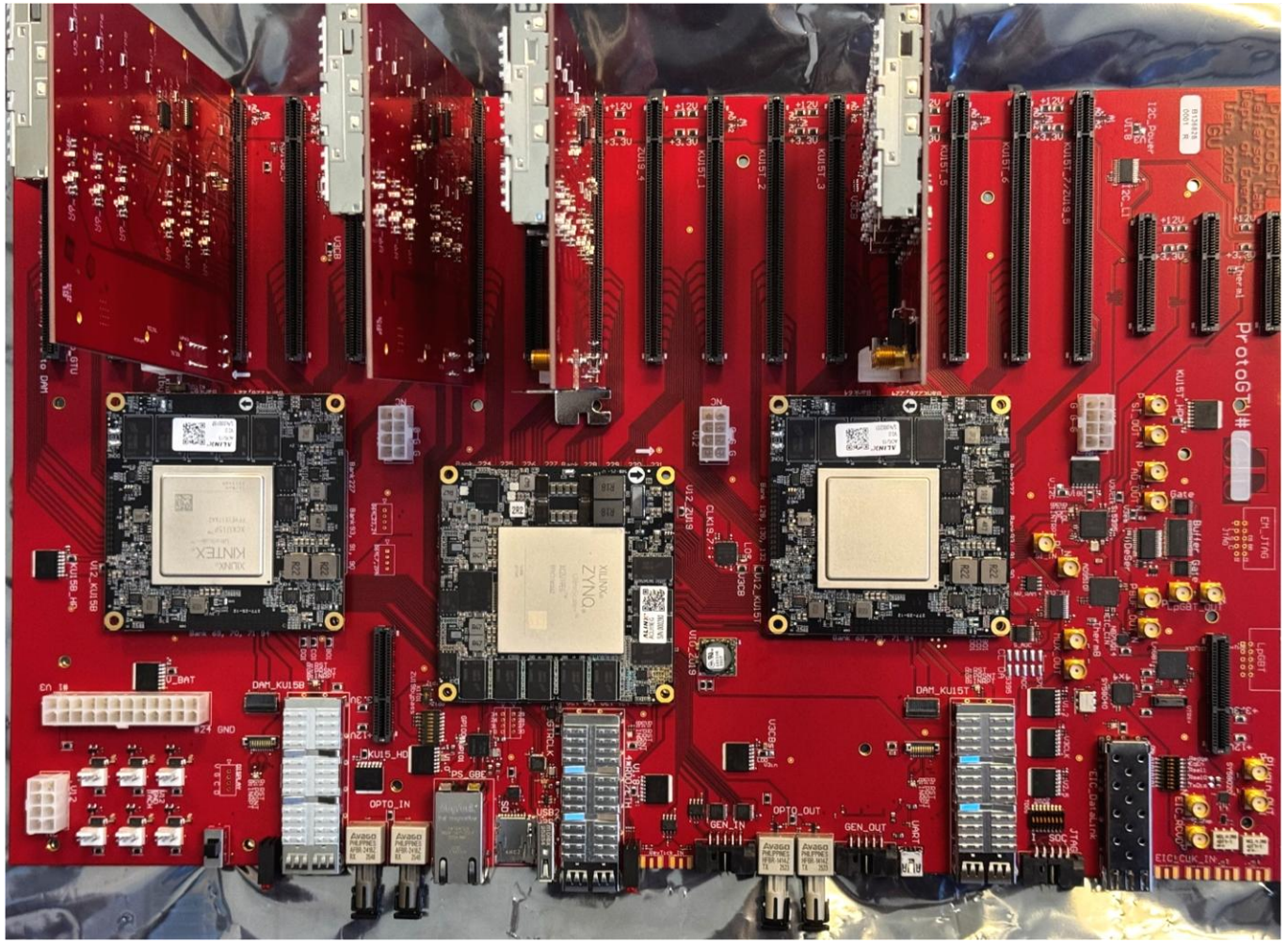


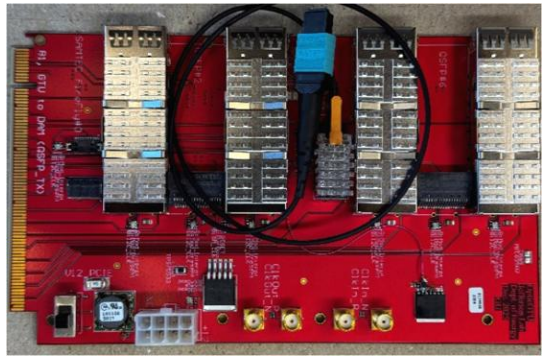
4U tall, 19” rack mounted

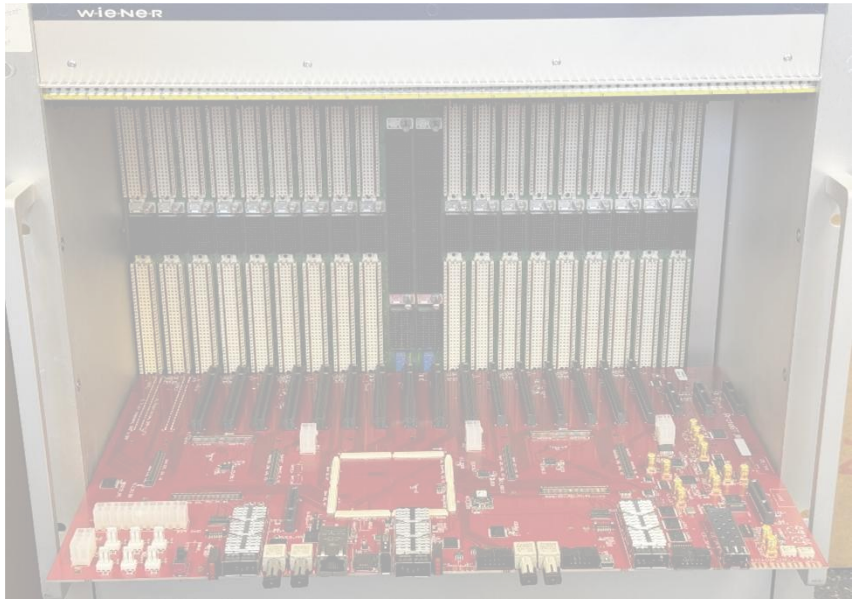
DAQ Room-air cooling

- GTU base board
- Optic transceiver adaptor boards
- Clock monitor board

ProtoGTU


ProtoGTU
DAM_KU15T
DAM_KU15B
OPTO_IN
OPTO_OUT
GEN_IN
GEN_OUT




wiener

GTU functional requirements

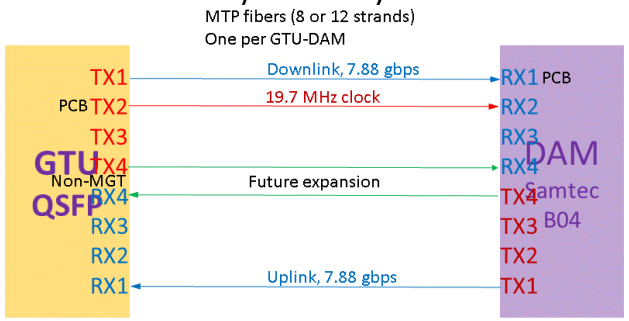
GTU required:

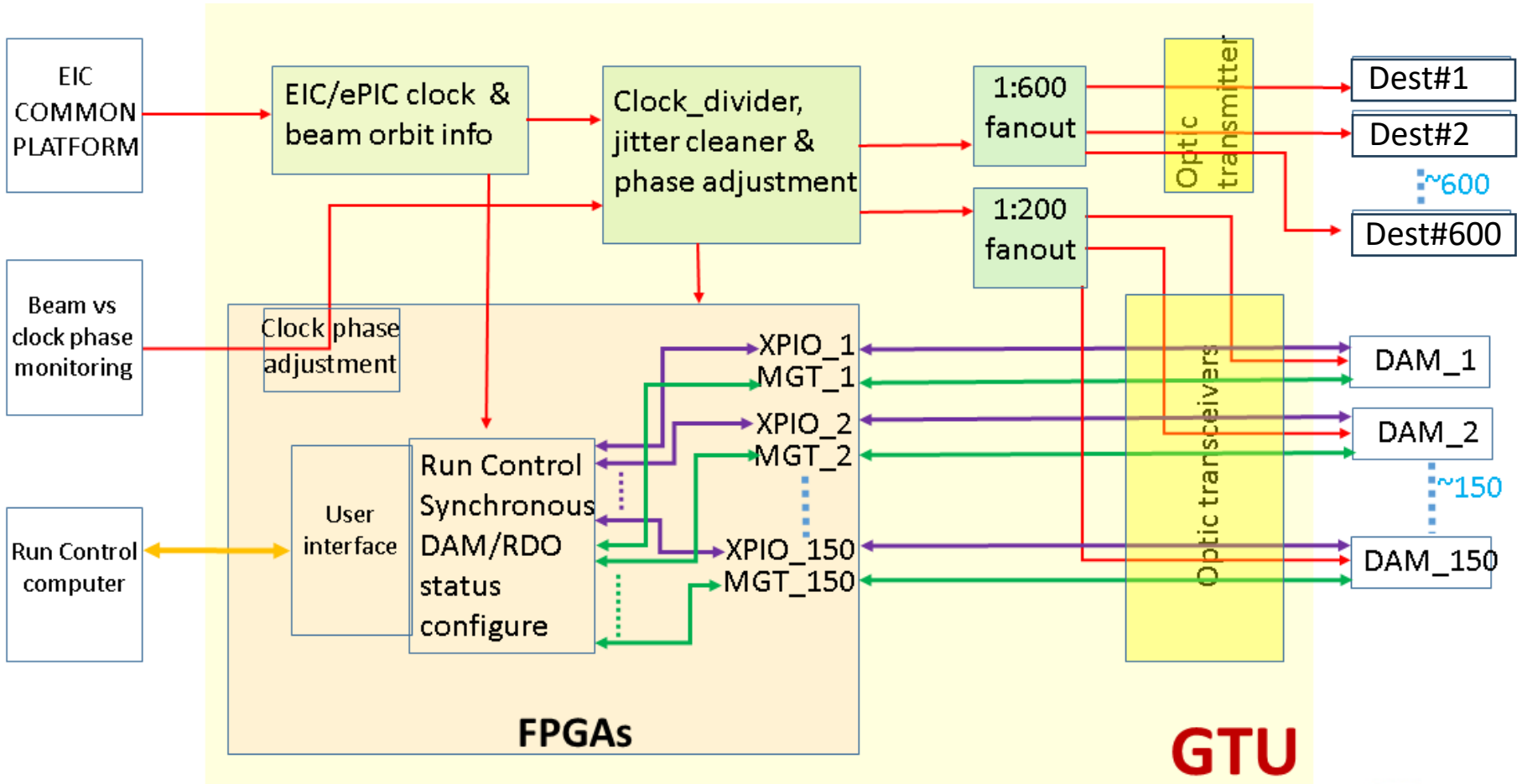
- EIC common platform interface;
- Run Control;
- ~150 DAM interface

Risk mitigation:

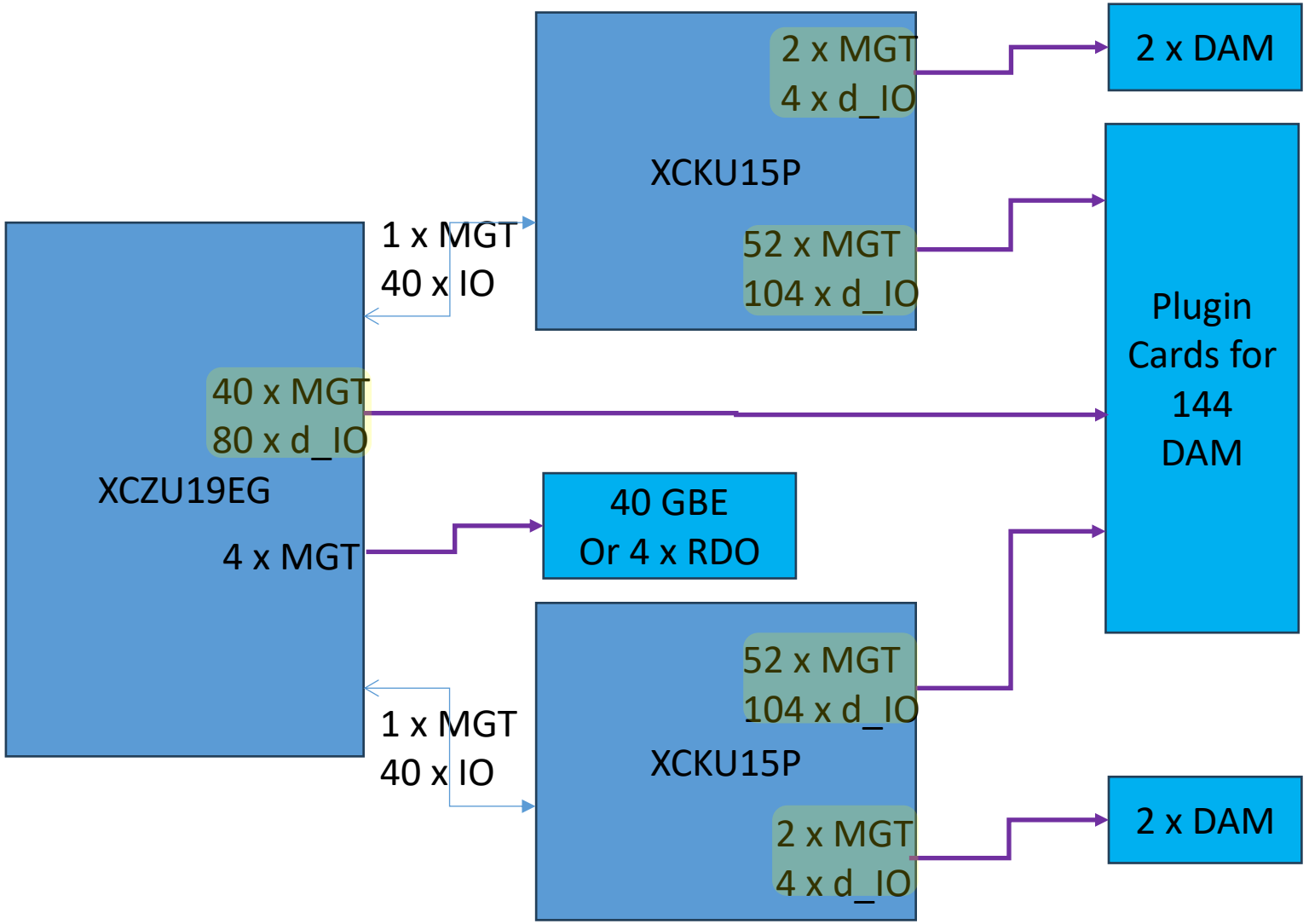
- Beam vs clock phase monitoring,
- Dedicated clock fanout

Between GTU and DAM (Quad LTI) Same-length Multimode fibers: 4 Tx/4 Rx (inexpensive, standard, commercially available)





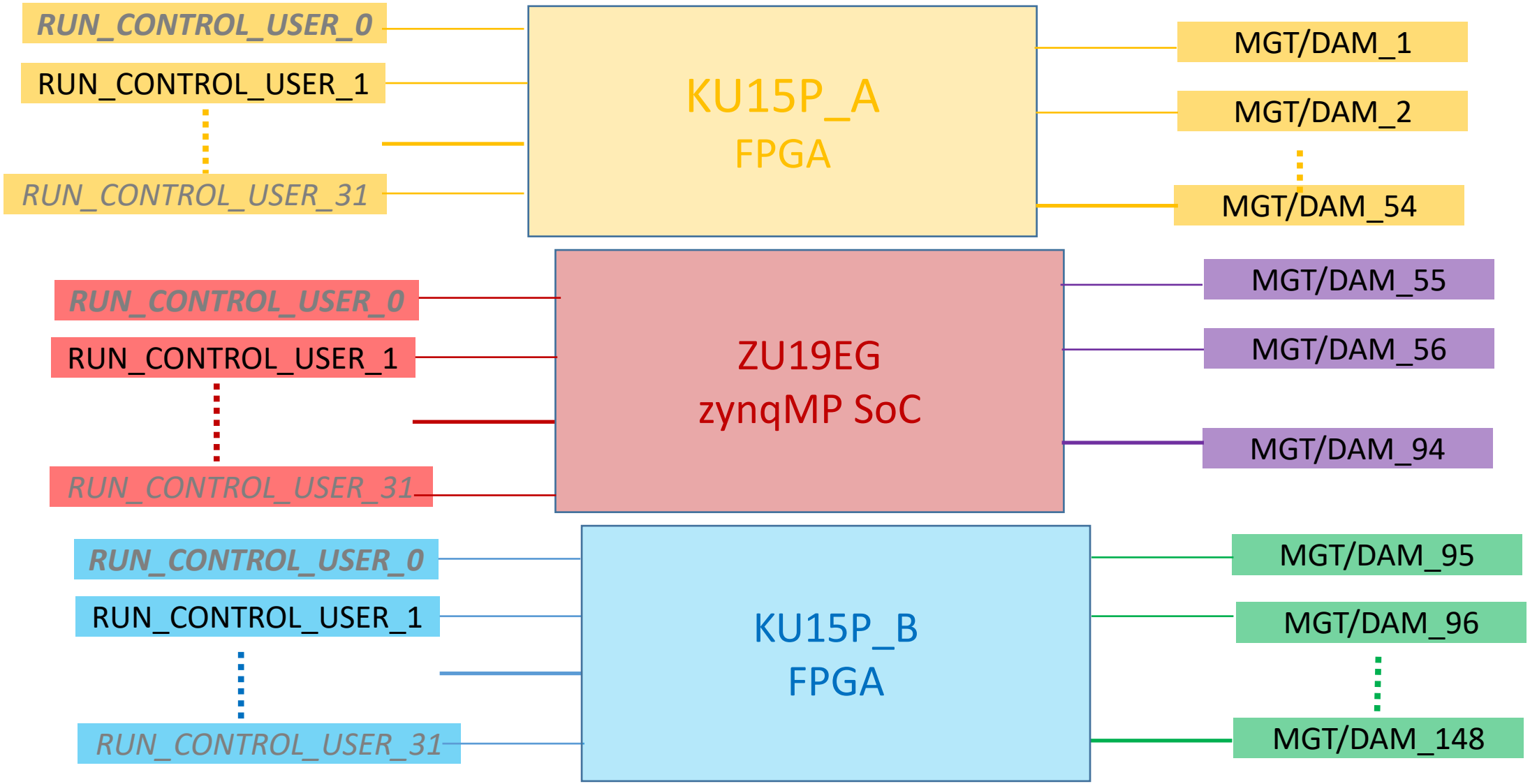
DAM control link (downlink/uplink)



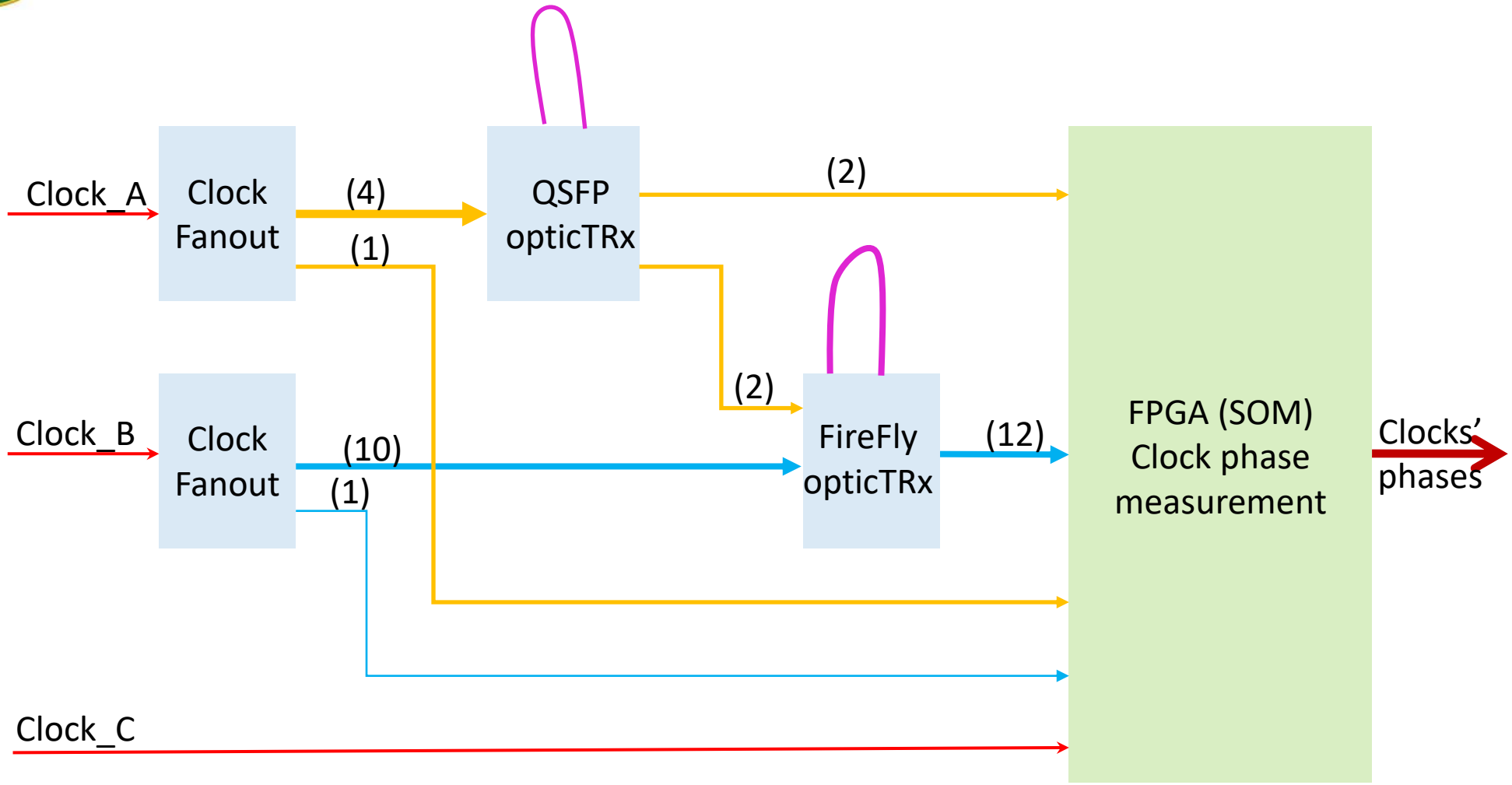
Supports:

- 148 DAM
- 4 RDO (or 40GBE)

The actual implementation of DAQ partition (multiple subsystem run-control)



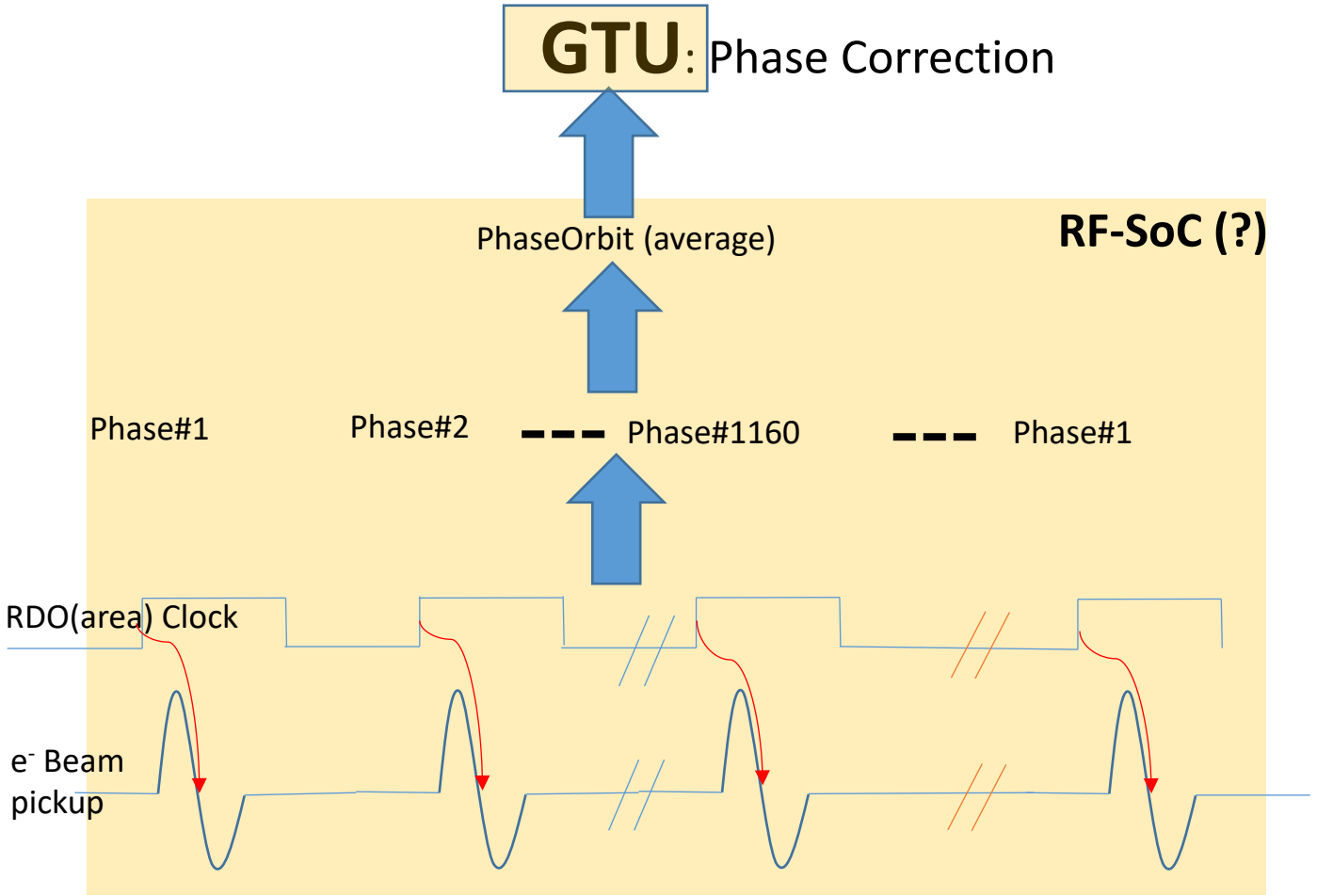
Clock monitoring (another GTU plugin module)



Fiber loopback, either the spares between GTU and DAM, or the spares between DAM and RDO

Clocks from the GTU base board, and the phase measurement results to the GTU base board

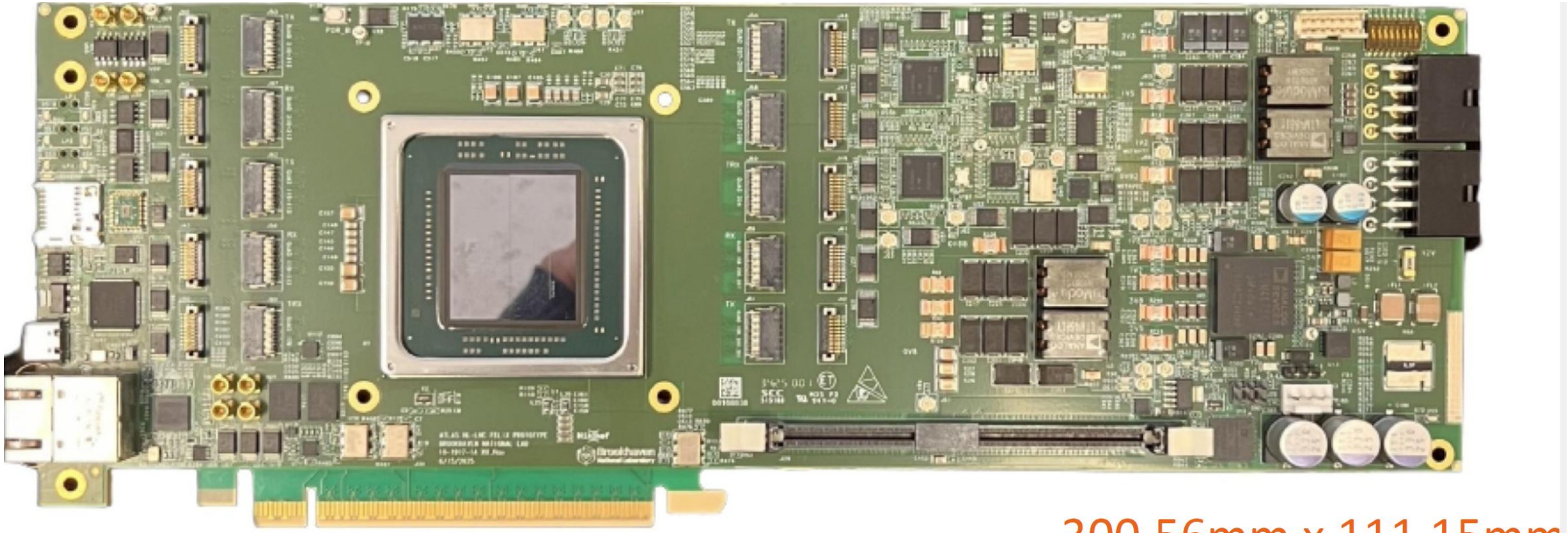
Beam pickup signal: stand alone PCB design



Phase adjustment:

- Si5395 (I2C) : ~70 ps steps
- AD9510 (SPI) : ~25 ps steps

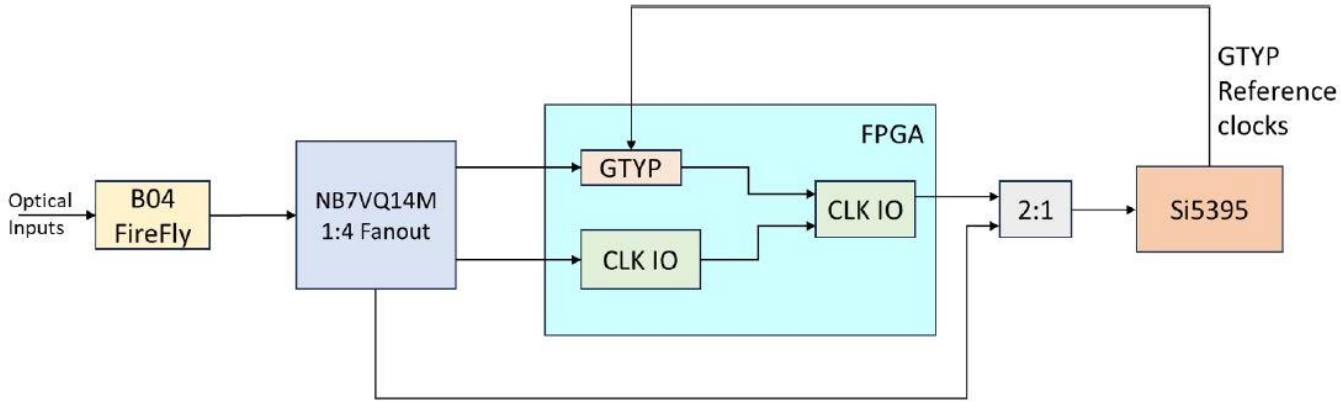
FLX155a – without the Firefly modules and front panel



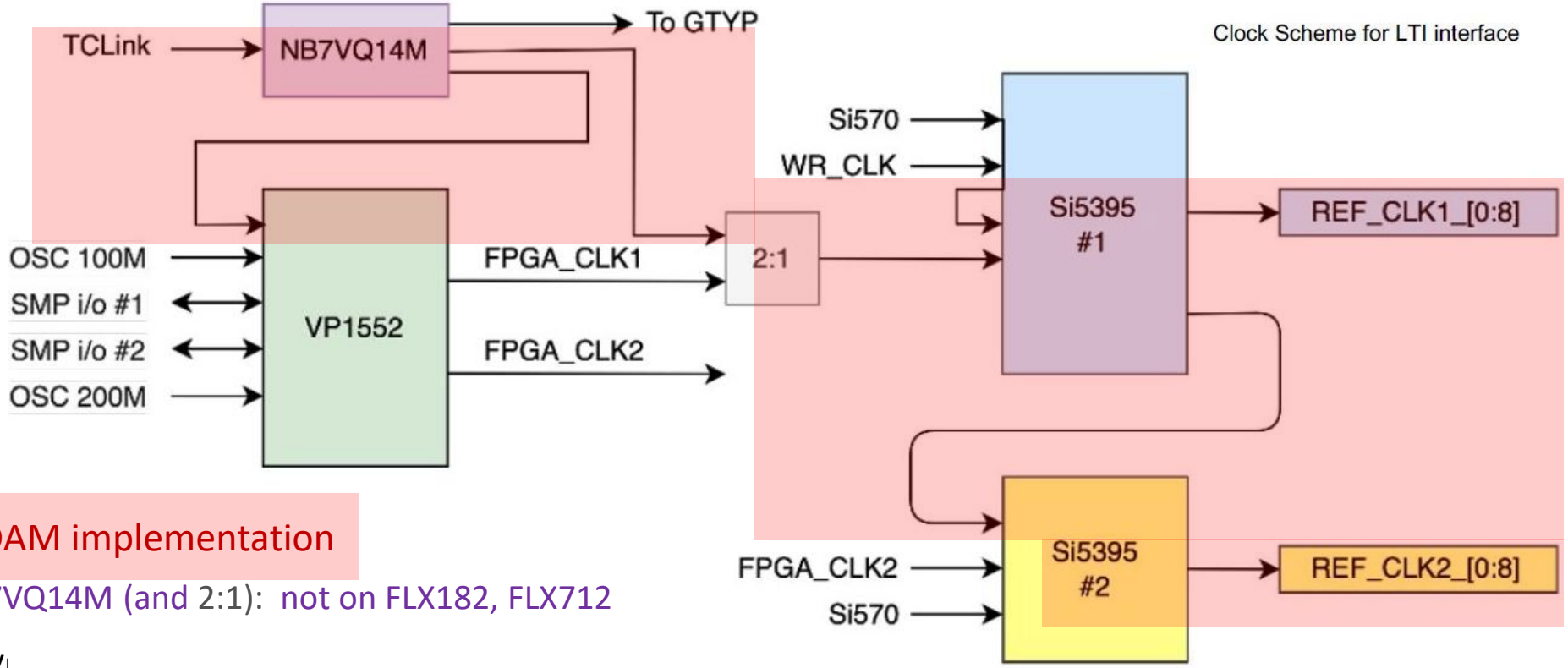
300.56mm x 111.15mm

DAM (FLX155) clock interface

- Dedicated Clock input,
- up to three MGT lanes (Tx/Rx)
- MTP multimode 850µm optical



Clock Scheme for LTI interface

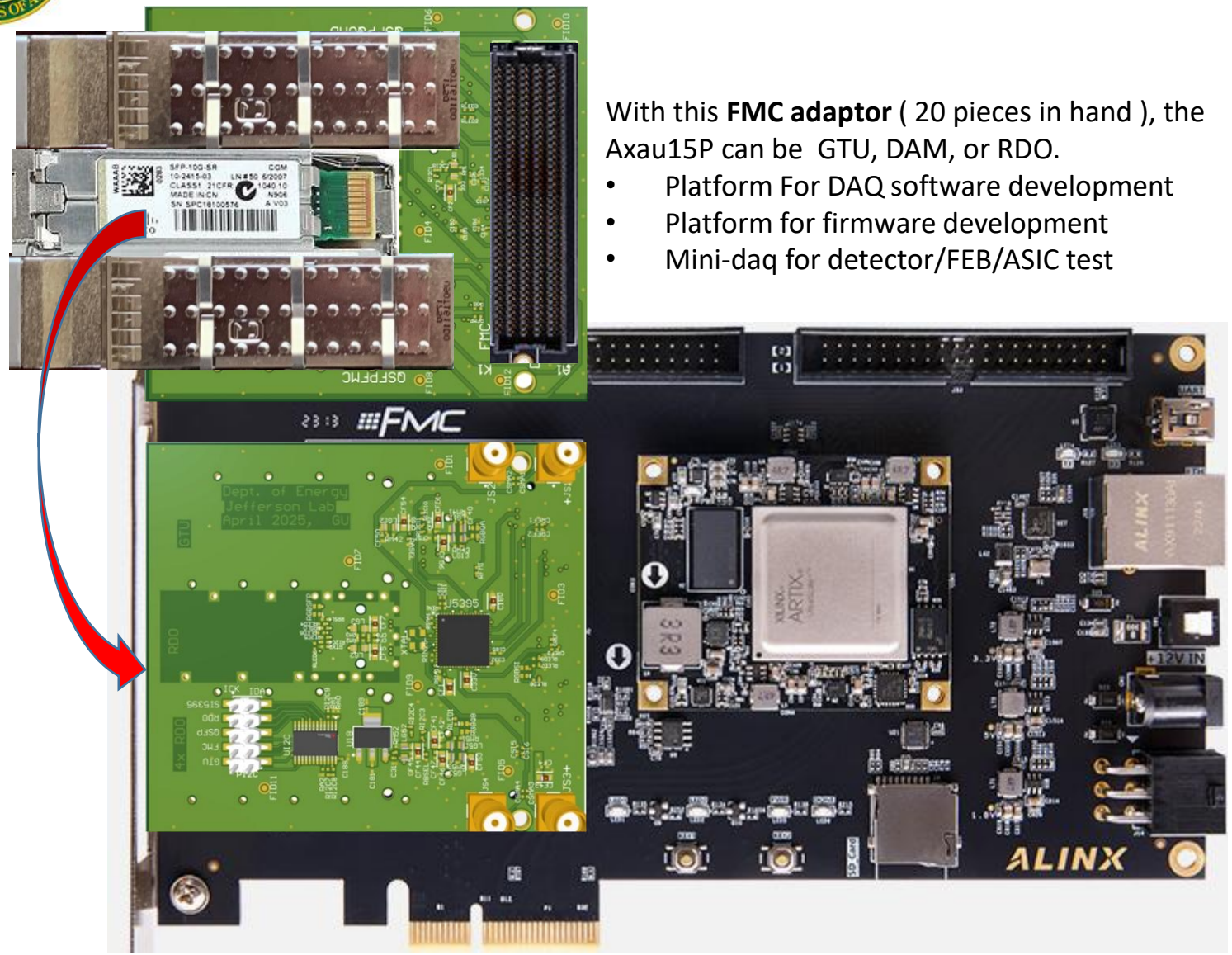


* DAM implementation

NB7VQ14M (and 2:1): not on FLX182, FLX712

* These two diagrams are from FLX155 presentations

FADE on Axau15p



With this **FMC adaptor** (20 pieces in hand), the Axau15P can be GTU, DAM, or RDO.

- Platform For DAQ software development
- Platform for firmware development
- Mini-daq for detector/FEB/ASIC test