Status of L1 concentrator

Aleksandr Boikov

on behalf of DAQ group

E-mail: Boikov@jinr.ru

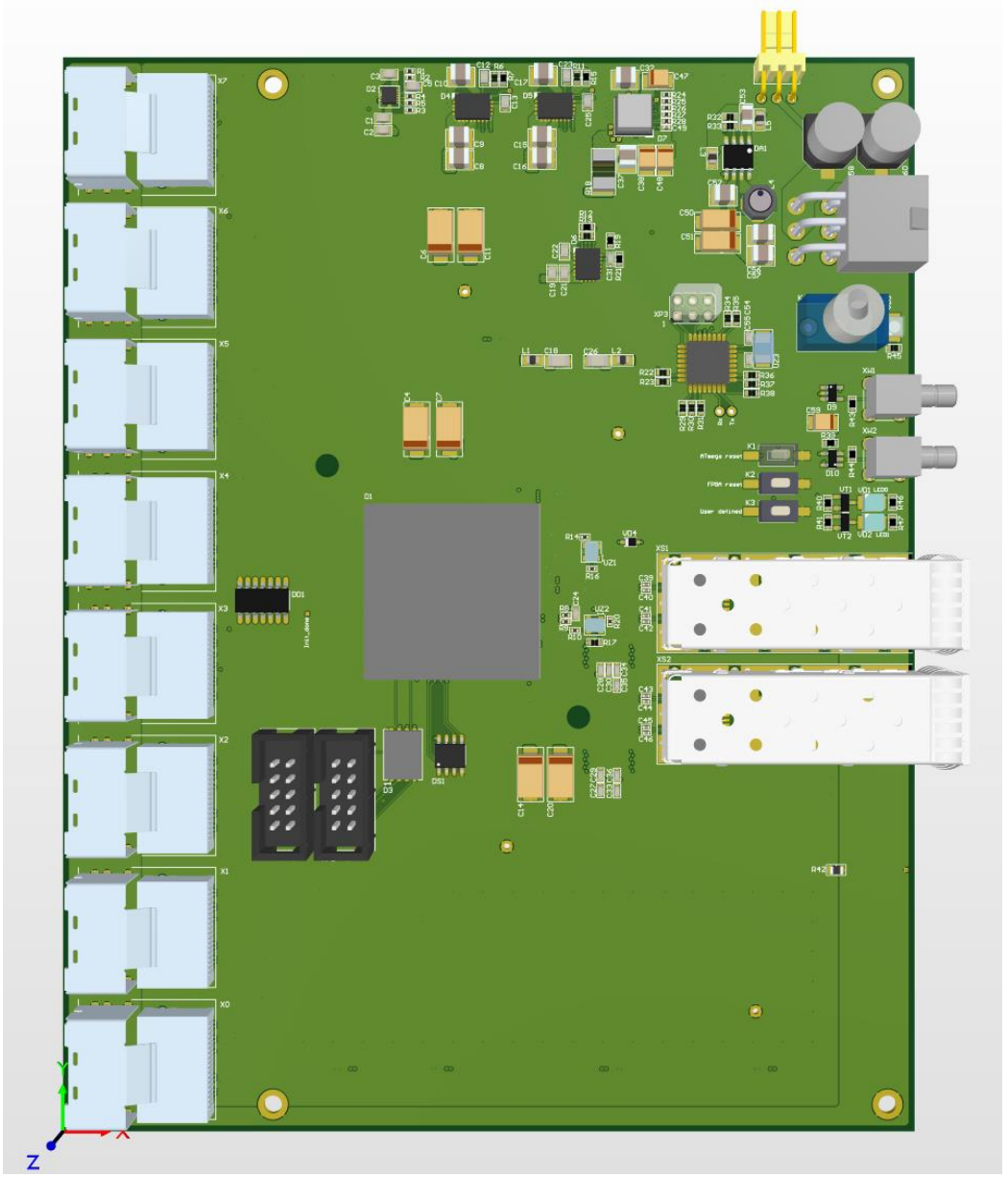
5.11.24

PCB of L1 concentrator (Current prototype)

- Cyclone 10GX (105YF780E6G)
- 8x Links for connect front-end boards (miniSAS connectors) 8 diff pairs per connector
- SFP+ 10Gb transceiver for data transmission to L2
- SFP+ transceiver for TSS (White Rabbit)

Concentrator tasks:

- Collecting data from the front-end boards
- Distribution of clock and commands from TSS
- Data integrity and timestamp control
- Slow control for FEB
- Reconfiguring front-end boards? (firmware)



PANGOMICRO (option for L1)

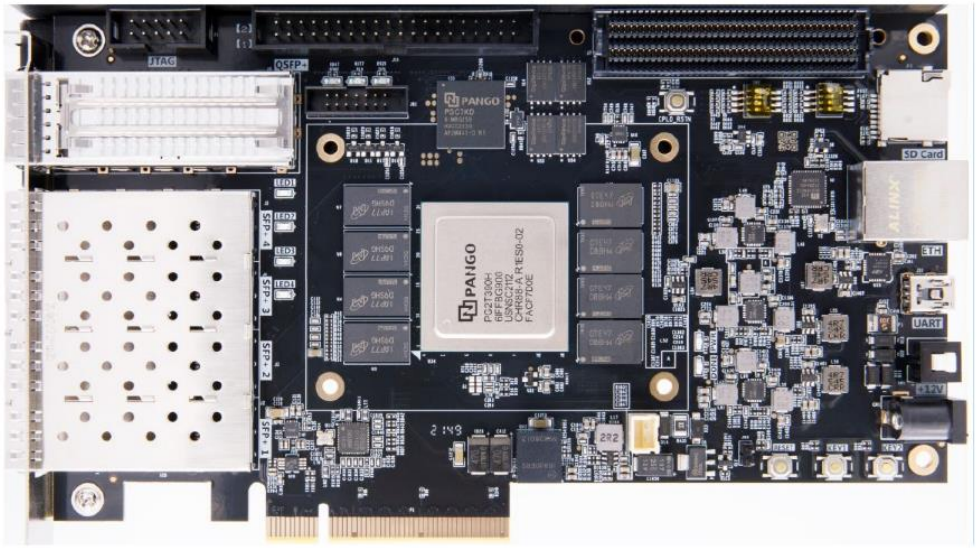
Pangomicro TITAN 2 (*Chinese FPGA*)

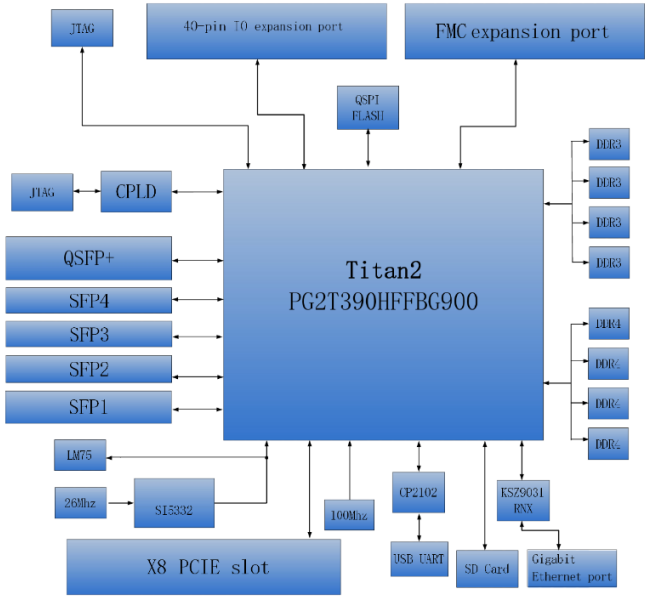
- Available for us *(without sanctions)* cost 220$ per chip
- Pin 2 pin compatible Xilinx Kintex 7 (*differences in architecture*)
- More resources compared to Cyclone 10GX
- Not detailed documentation
- There are a lot of problems with the IDE *(at the moment)*

Resource name		PG2T390H
CLM	LUT6	243600
	logical unit	389760
	FF	487200
	Distributed ram (Kb)	4712
DRM (36Kbits/ pc)		480
APM(units)		840
PLLs	GPLLs	10
	PPLLs	10
ADC (dual core)	Dedicated analog channel (differential input pair)	1
	Multiplexed analog channels (differential input pair)	11
SERDES LANE [1]		16
PCIE GEN2×8 CORE		1

ALINX APX390 *(Current available development board for TITAN 2)*

- Good documentation and reference designs for work
- Today: Ready 10 G link
- Need test SERDES and integration of White Rabbit.
- For CDR (clock data recovery) mode need own development





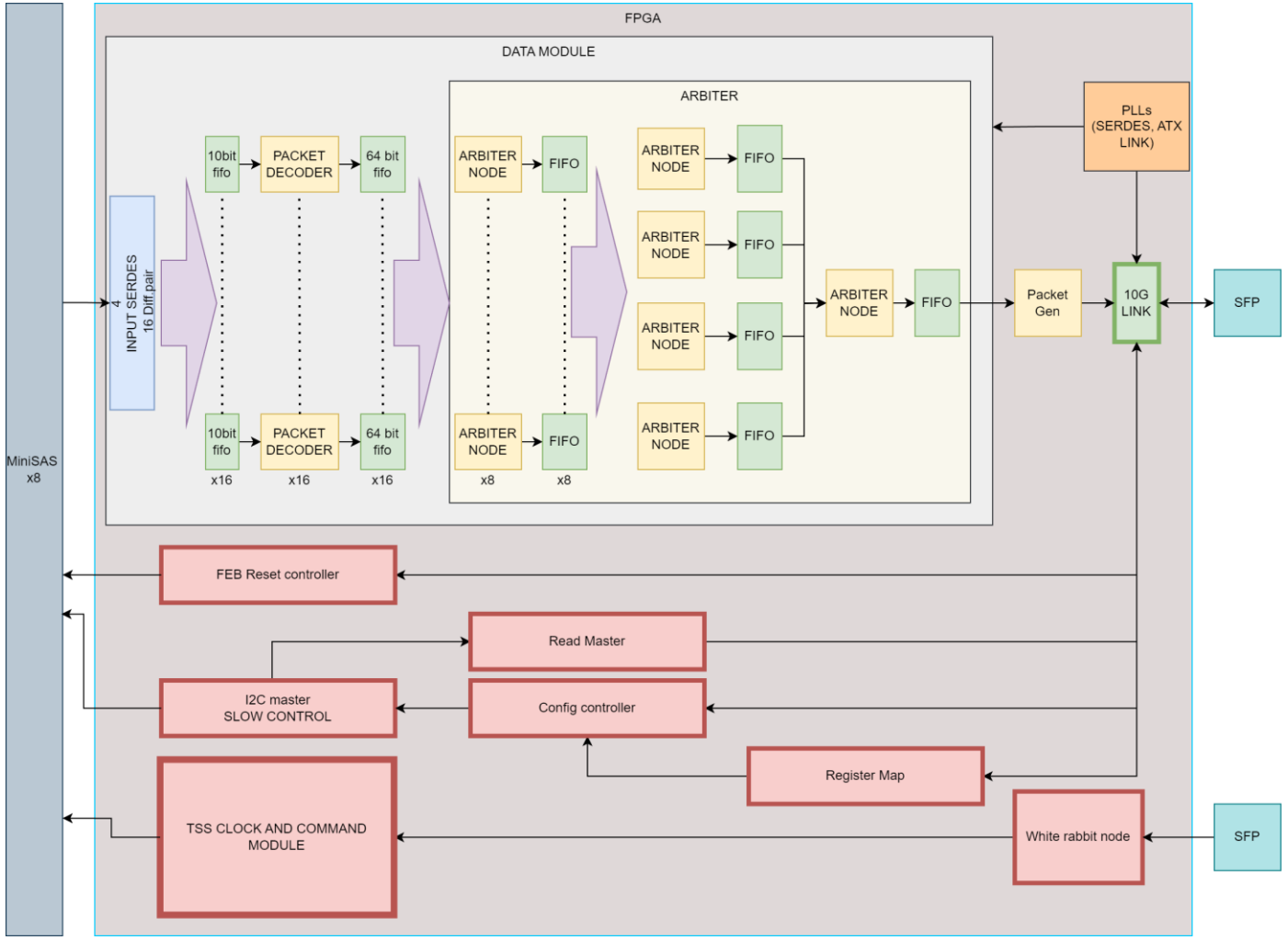
Changed Firmware architecture

Last 5 month:

1. Changed internal data bus (AvalonST/MM to AXIS/AXIL)
2. Minimized use of IP-cores from the manufacturer of FPGA. (PLLs, SERDES and Transceivers is remained)
3. All control registers can be generated by using SystemRDL tool. (HDL, C-header, Docs)
   thanks V. Borch for help with this task
4. Completed support for UDP, ARP and ICMP for standalone operation without L2
5. Added support *raw* Ethernet II frames for communication between L1 and L2

Plans :

1. Integrating and test TSS command generator (TSS group provided source code)
2. Integration of modules for control L1 via L2 (L2 group provide source code)
3. Preparing an example IP-core for communication between FEB and L1
4. Preparing an example design of a TSS command receiver for FEB.



Plans of Firmware verification

Now:

Using partially manual verification with only HDL tools.

Plans:

Use special Python based Open-Source framework Cocotb for verification.

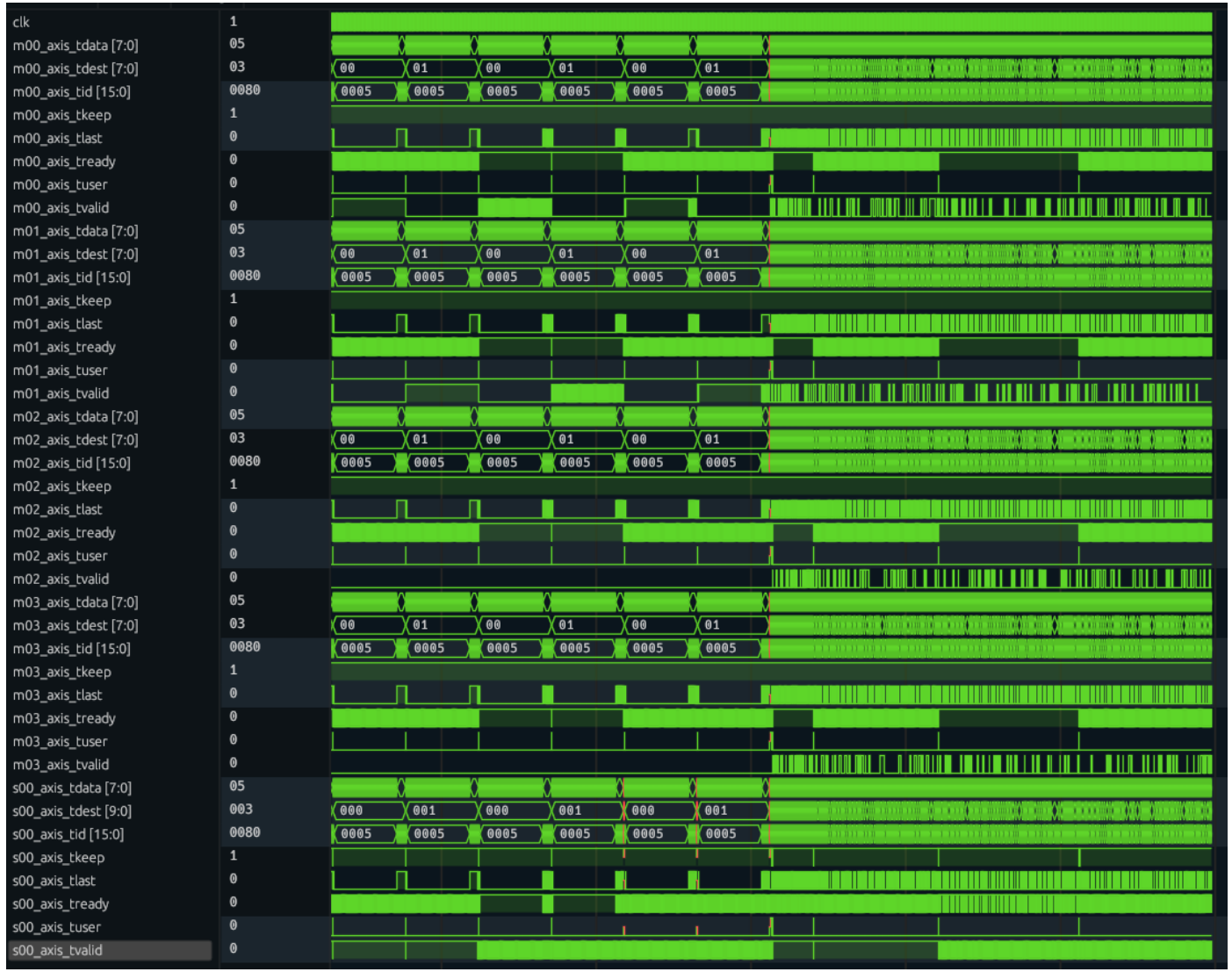
Optional:

CI/CD for build firmware and long test

Cocotb Example

```
run_stress_test_004 passed
**************************************************************************************************
** TEST                                          STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
**************************************************************************************************
** test_axis_switch.run_test_001                 PASS        7330.00           0.10      74458.06  **
** test_axis_switch.run_test_002                 PASS        7340.00           0.09      84677.96  **
** test_axis_switch.run_test_003                 PASS       23530.00           0.26      91518.48  **
** test_axis_switch.run_test_004                 PASS       23530.00           0.28      84762.75  **
** test_axis_switch.run_test_005                 PASS       23550.00           0.21     109535.18  **
** test_axis_switch.run_test_006                 PASS       23550.00           0.21     112090.92  **
** test_axis_switch.run_test_007                 PASS       23570.00           0.30      79542.59  **
** test_axis_switch.run_test_008                 PASS       23570.00           0.31      75783.28  **
** test_axis_switch.run_test_tuser_assert_001    PASS         440.00           0.01      39235.86  **
** test_axis_switch.run_test_tuser_assert_002    PASS         440.00           0.01      56601.69  **
** test_axis_switch.run_stress_test_001          PASS       13130.00           0.17      78828.12  **
** test_axis_switch.run_stress_test_002          PASS       40530.00           0.48      84978.20  **
** test_axis_switch.run_stress_test_003          PASS       45430.00           0.42     108203.20  **
** test_axis_switch.run_stress_test_004          PASS       43130.00           0.57      75306.52  **
**************************************************************************************************
** TESTS=14 PASS=14 FAIL=0 SKIP=0                          299070.01           3.97      75294.06  **
**************************************************************************************************
```

Cocotb Example



E-Link *port*

Dedicated line for:

- Start of slice
- Start of frame
- Set Next Frame
- Global clock from L1 to Front-End

- Clock for data. From Front-End to L1
- Data lines (Max 1250 Mbps)
- 8 Unipolar lines for slow interface and different signals

SPI OPTIONS

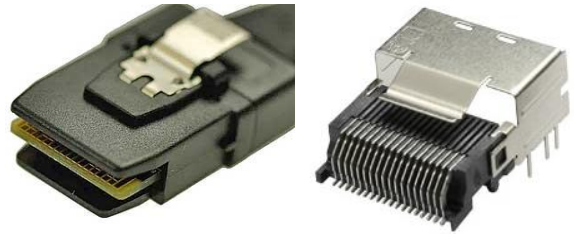
Line on FEB	Signal description	Line on L1	Comments
TX0	Data line from FEB	RX0	
TX1	Data line from FEB	RX1	
TX2	Data line (RECONFIG?)	RX2	
TX3	Clock line from FEB	RX3	Synchronization Clock for data line
RX0	Clock from L1	TX0	Global cklock from TSS
RX1	Start of Slice (SNF)	TX1	
RX2	Start of Frame (SOS)	TX2	
RX3	Set Next Frame (SOF)	TX3	Frame number (32 bit data)
Unipolar 0	Reset (HARD)	Unipolar 7	Hard reset of FEB (active hight)
Unipolar 1	SCIK	Unipolar 3	Clock for SPI (5-10 MHz)
Unipolar 2	CS_1	Unipolar 4	CS for slow control
Unipolar 6	CS_2 (optional)	Unipolar 5	CS for reconfig (optional)
Unipolar 7	MOSI (L1)	Unipolar 0	Data from L1
Unipolar 3	MISO (L1)	Unipolar 1	Data from FEB
Unipolar 4	Ready (From FEB)	Unipolar 2	FEB initialized (after RESET)
Unipolar 5	Ready (From L1)	Unipolar 6	Data line synchronized

VS

I2C OPTIONS

Line on FEB	Signal description	Line on L1	Comments
TX0	Data line from FEB	RX0	
TX1	Data line from FEB	RX1	
TX2	Data line (RECONFIG?)	RX2	
TX3	Clock line from FEB	RX3	Synchronization Clock for data line
RX0	Clock from L1	TX0	Global cklock from TSS
RX1	Start of Slice (SNF)	TX1	
RX2	Start of Frame (SOS)	TX2	
RX3	Set Next Frame (SOF)	TX3	Frame number (32 bit data)
Unipolar 0	Reset (HARD)	Unipolar 7	Hard reset of FEB (active hight)
Unipolar 1	SCL	Unipolar 3	Clock for I2C
Unipolar 2	SDA	Unipolar 4	Data line I2C
Unipolar 6	?	Unipolar 5	
Unipolar 7	?	Unipolar 0	
Unipolar 3	?	Unipolar 1	
Unipolar 4	Ready (From FEB)	Unipolar 2	FEB initialized (after RESET)
Unipolar 5	Ready (From L1)	Unipolar 6	Data line synchronized

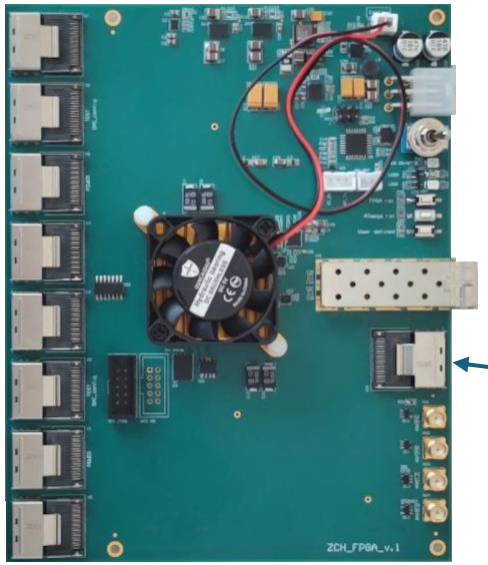
MiniSAS



Pin function	Side B Pin	Side A Pin	Pin function
GND	B1	A1	GND
Tx0+	B2	A2	Rx0+
Tx0-	B3	A3	Rx0-
GND	B4	A4	GND
Tx1+	B5	A5	Rx1+
Tx1-	B6	A6	Rx1-
GND	B7	A7	GND
Unipolar 0	B8	A8	Unipolar 7
Unipolar 1	B9	A9	Unipolar 3
Unipolar 2	B10	A10	Unipolar 4
Unipolar 6	B11	A11	Unipolar 5
GND	B12	A12	GND
Tx2+	B13	A13	Rx2+
Tx2-	B14	A14	Rx2-
GND	B15	A15	GND
Tx3+	B16	A16	Rx3+
Tx3-	B17	A17	Rx3-
GND	B18	A18	GND

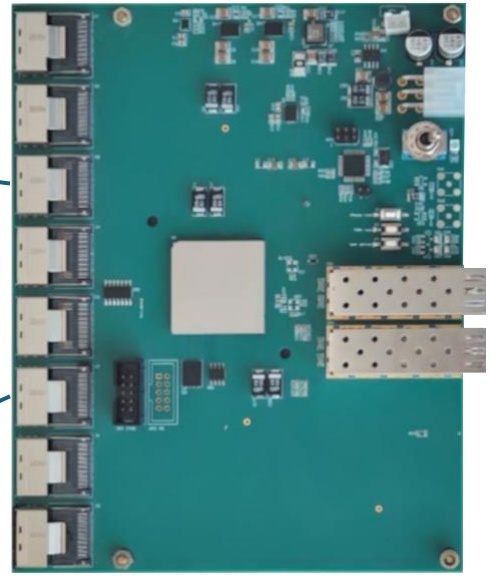
Hardware *Plans*

Our *FEB* test board

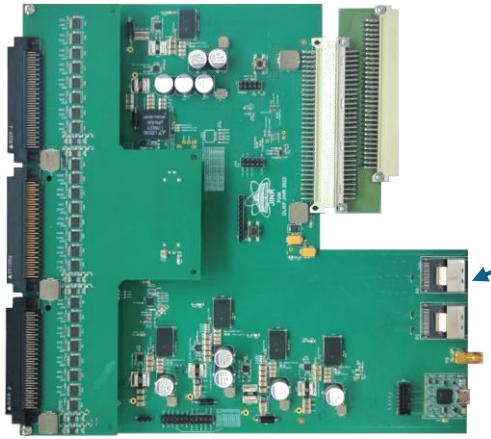


Test:
1. TSS command generation
2. Slow control

L1 prototype



RS FEB



Test: (*After connection is established*)
1. Communication between FEB and L1
2. Data transfer (standalone mode without L2)

Thanks for your attention

E-mail: Boikov@jinr.ru