Control of ASICs via GBTx

Wojciech M. Zabołotny

Institute of Electronic Systems, Warsaw University of Technology

in cooperation with Robert Szczygieł and Krzysztof Kasiński from AGH university of Science and Technology

GBTx and its functions

- Project website:
  https://espace.cern.ch/GBT-Project/default.aspx
  (requires CERN account)
- GBTx manual:
  https://espace.cern.ch/GBT-Project/GBTX/Manuals/gbtxManual.pdf

Most important (for us) functions

- GBTx allows to transfer certain sequence of bits (frame) via 4,8Gbps link (40 Mframes/s) in different modes:
  - GBT frame mode – frame contains 80 user bits (+2 bits IC and 2 bits EC), but is FEC protected – we use it for downlink transmission
  - Wide frame mode – frame contains 112 user bits long (+2 bits IC and 2 bits EC), we use it for uplink transmission
- In each direction we may use different number of e-links, depending on the clock speed. For 80-bit frame and 160 MHz clock we may have 20 channels per GBTx. For 112-bit frame and 320 MHz clock we may have 14 channels per GBTx.
- As we need higher bandwidth in uplink direction to transmit the measured data, than in downlink direction, we can use 2 "slave" GBTx with simplex transmitters connected to 1 "master" GBTx with duplex transceiver (more details in description of STS Readout Boards)

Current STS Readout Concept

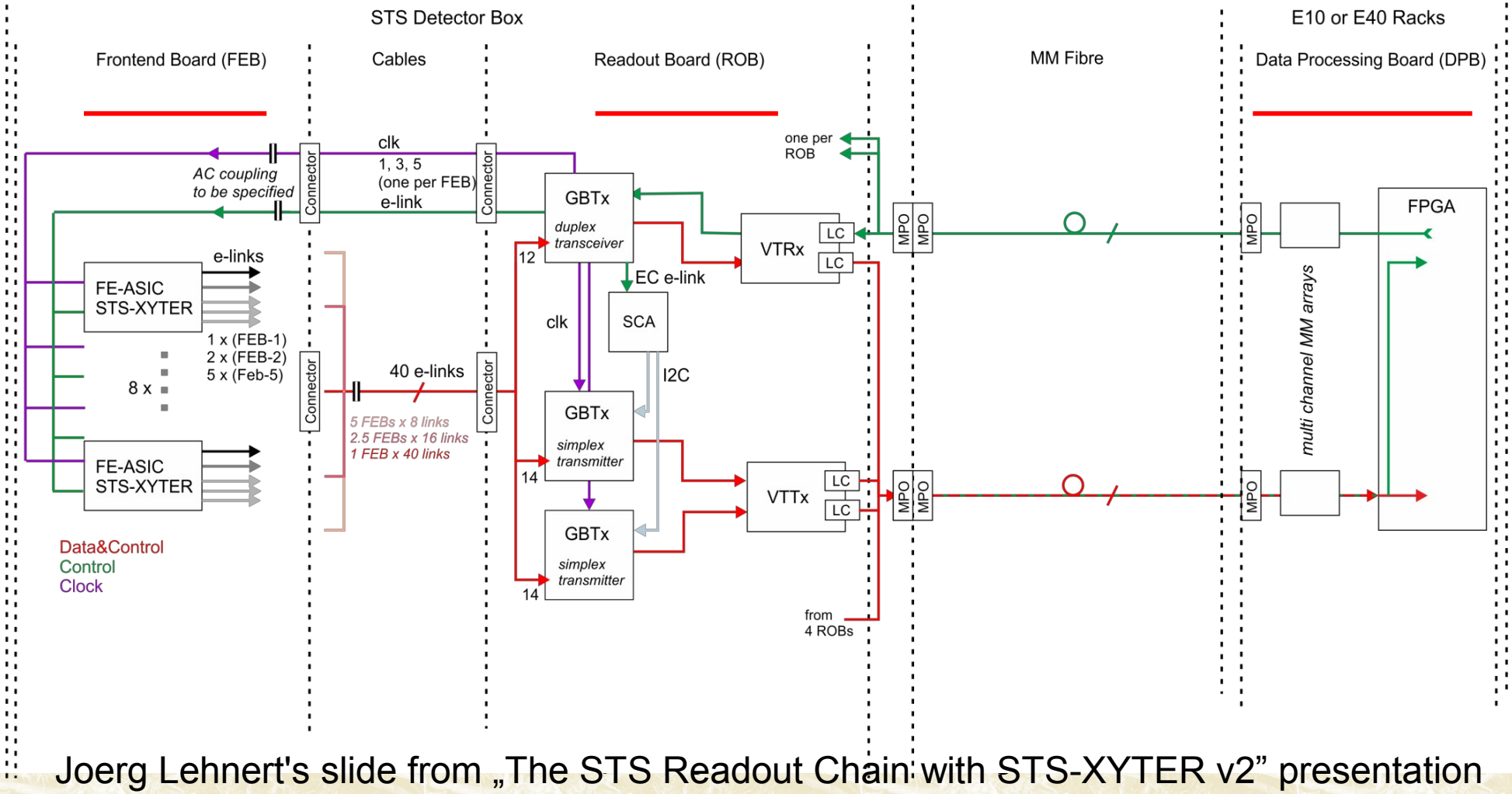
FEB
8 STSXYTER

el. interf.
SLVS
1022 pairs/FEB

ROB
GBTx / VL

optical interface
4 MM fibers /ROB

DPB



Joerg Lehnert's slide from „The STS Readout Chain with STS-XYTER v2" presentation

STS Readout Board (ROB)

1 master GBTx connected to 1 VTRx transceiver
- Slow control, clock distribution
- Data readout

2* slave GBTx connected to 1 VTTx (twin transmitter)
- Data readout

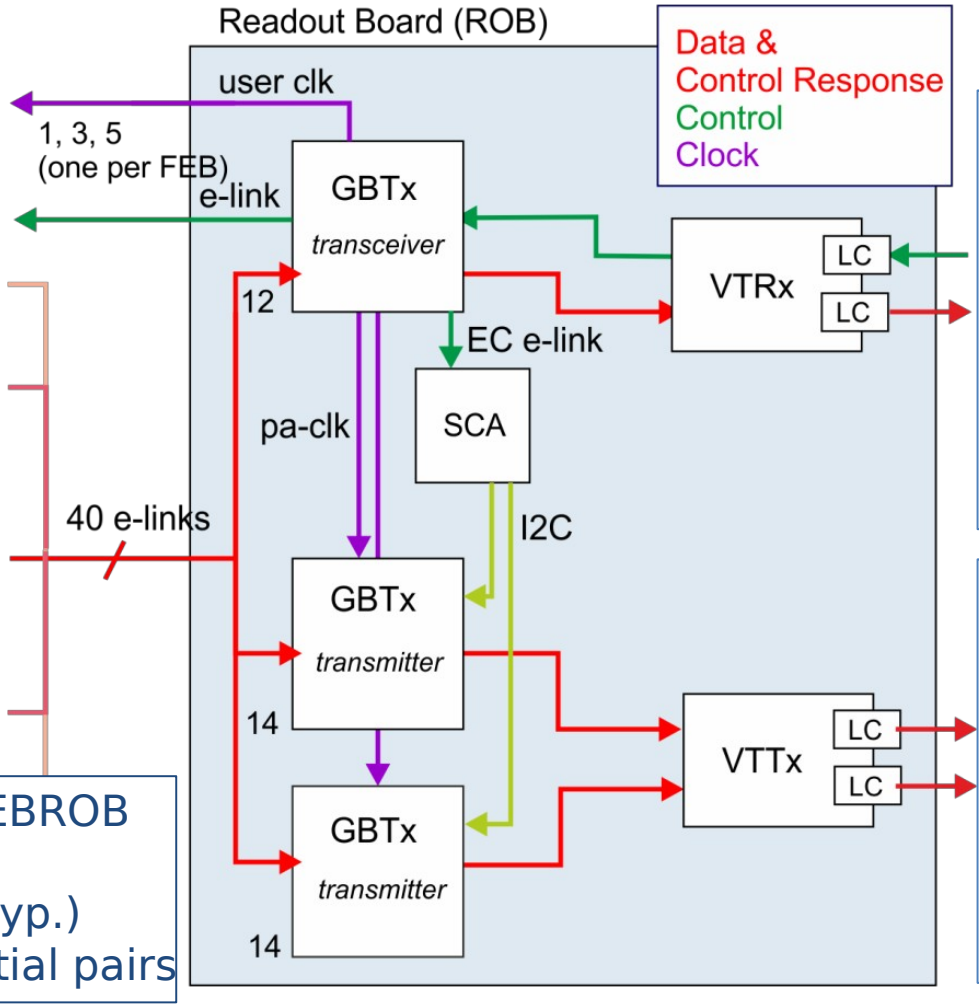
1 SCA as slow control interface from master to slaves

Frontend Side

Clock and Slow Control:
- single clock and slow control downlink to FEB
- multidrop to 8 ASICs

Input from FEBs:
- 3x14 = 42 e links@320Mbps
- GBTx wide bus mode
- input bandwidth 13.44 Gbps

Minimized number of FEBROB connections:
on FEB side from 8+2 (typ.)
to 40+2 (max.) differential pairs



Backend Side

Output to DPBs:
- 1 VTRx, 1 VTTx
- 1 downlink
- 3 uplinks
- user bandwidth 3x4.48 = 13.44 Gbps

SCA:
- I2C masters for slave GBTx control
- Use additional slow control features?

Joerg Lehnert's slide from „The STS Readout Chain with STS-XYTER v2” presentation

Control of the GBTx itself

- To control the GBTx itself and to control other peripherals or "slave" GBTx connected via standard GBT-SCA chip, it is necessary to send HDLC encoded bitstream in IC or EC channels
- The appropriate controller is not ready yet There is a chance that one group from CERN will provide a non-supportes example design of such controller in April
- Maybe our own solution is needed?

GBTx as FE ASIC controller

- We use GBTx to provide the 160 MHz clock to the FE ASIC, the SDR data stream in the link to FE ASIC, and to receive the DDR data stream in each link from the FE ASIC.
- Because the FE ASICS must be connected via AC coupled links, it is necessary to use DC balanced encoding (please note that GBTx 8b/10b encoded mode doesn't help there!)
  - Different options exist (Manchester, 7b/8b, 8b/10b)
  - We have used 8b/10b encoding (moderate overhead, easy synchronization)

Downlink frames

- Downlink frames are transmitted as 60-bit frames (after 10b/8b encoding – 6 bytes before encoding). This allows to send special 20-bit sequences (described later) without corrupting of the frame sequence.
- Each frame contains:
  - K28.5 synchronization comma character
  - Frame sequence number and chip address
  - Command and optional payload
  - 16-bit CRC
- Commands include: read from given address, set address for write command, write data
- Each command should be confirmed. 4-bit sequence numbers allow to handle acknowledgements properly when multiple commands in flight are used.

BYTE<0> frame_bits<47:40> bits_8b10b<59:50>	BYTE<1> frame_bits<39:32> bits_8b10b<49:40>	BYTE<2> frame_bits<31:24> bits_8b10b<39:30>	BYTE<3> frame_bits<23:16> bits_8b10b<29:20>	BYTE<4> frame_bits<15:8> bits_8b10b<19:10>	BYTE<5> frame_ bits<7:0> bits_8b10b<9:0>
Comma character	Frame ID & Chip Address	Request type / Payload	Payload	CRC	CRC
K28.5 001111 1001 110000 0110	<7:4> chip address (0..7, 15) <3:0> sequence number (0..15)	<7:6> Request type (0..3) <5:0> Payload <13:8>	<7:0> Payload <7:0>	<7:0> CRC <15:8>	<7:0> CRC <7:0>

Uplink frames

- Uplink frames are optimized for throughput therefore Huffman encoding is used to describe type of the frame
  - Frame starting with 0 bit is the frame with readout data (for those frames throughput is crucial)
  - Frame starting with 11 is the timestamp
  - Frame starting with 101 contains data read from register
  - Frame starting with 100 is the command acknowledgement

  (more details in the STS-XYTER v2 protocol description)
- Uplink frames are 3 bytes long (30 bits after 8b/10b encoding)
- To ensure link synchronization, special synchronization frame consisting of three K28.5 comma characters is transmitted periodically

Table. 3. Structure of the uplink frames.

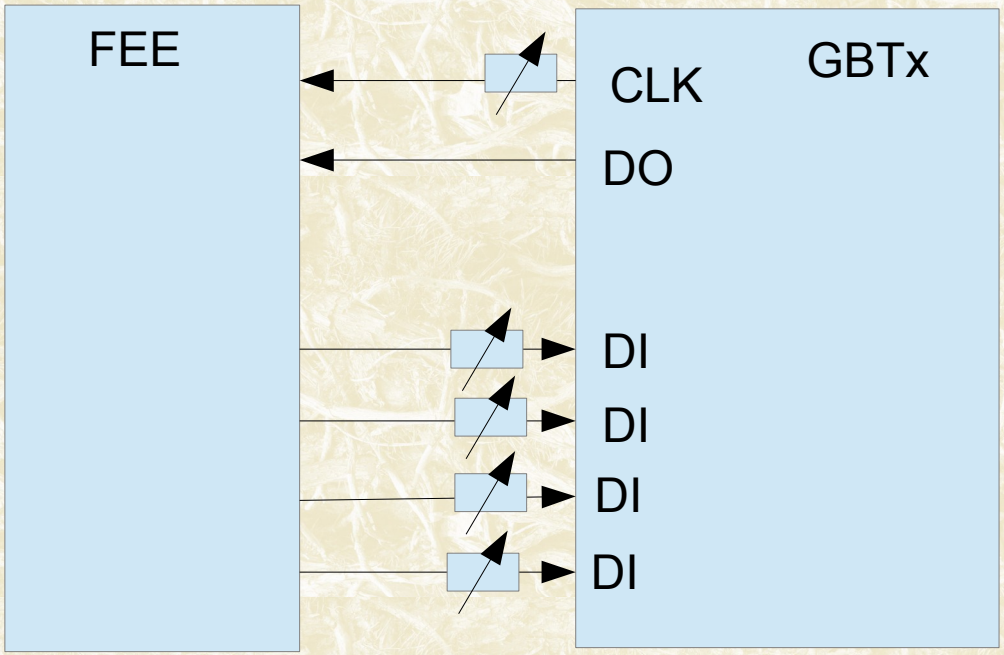
Structure of the uplink frames (before 8b/10b encoding)																									
	BYTE<0> frame_bits<23:16> bits_8b10b<29:20>								BYTE<1> frame_bits<15:8> bits_8b10b<19:10>								BYTE<2> frame_bits<7:0> bits_8b10b<9:0>							
Type	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Dummy Hit	0	7-bit channel address = 0x00							5-bit ADC = 0x00					0x0		0x0 (TODO: Timestamp<13:6> (actual state of counter))								0
Hit	0	7-bit channel address							5-bit ADC > 0x00					TS<9:8> (overlap)		Timestamp<7:0>								EM
TS_MSB	1	1	Timestamp<13:8>						Timestamp<13:8>					Timestamp<13:8>							4-bit CRC poly 0x9 = (x^4)+x+1				
RDdata_ack	1	0	1	14-bit register content													3-bit sequence number (LSB)				4-bit CRC poly 0x9 = (x^4)+x+1				
Ack	1	0	0	ACK	4-bit sequence number			CP	4-bit status value					0x00 or Timestamp<7:2> depending on CONFIG<1> register setting							4-bit CRC poly 0x9 = (x^4)+x+1				

Byte order on the link: **BYTE<0>** first (then 1 and 2)
Bit order on the link: **bits_8b10b<29>** first (then 28, 27, ..., 1, 0);

Structure of the uplink frames (represented only in 8b/10b encoding			
Type	10-bit frame (8b/10b)	10-bit frame (8b/10b)	10-bit frame (8b/10b)
Sync	K28.5 comma character	K28.5 comma character	K28.5 comma character
POST_Reset	K28.5 comma character		

Link synchronization

- When system starts, skew between the clock line and data lines is unknown.
- GBTx offers possibility to adjust the clock phase and delay on data inputs
- Special procedure must be used to set proper delays



Link synchronization 1

- We must be able to switch FE ASIC to the synchronization mode
- Special sequence must be transmitted which is reliably detected by the ASIC even when link is out of sync
- The right sequence is "00000_00000_11111_11111" (SOS - Start of Synchronization)
- Due to misalignment of clock and data lines the number of 0s or 1s may be changed by 1, but anyway the sequence may be reliably detected and distinguished from other words transmitted through the 8b/10b encoded link
- So in the first step of synchronization GBTx starts to transmit SOS, and waits until on all input lines SOS is received

Link synchronization 2

- We must align output clock so, that the FE ASIC correctly receives data
- To achieve that, we transmit K28.1 comma character
- If the FE ASIC sees correct K28.1 character, it responds with K28.1. Otherwise it still transmits SOS
- We can't reliably receive K28.1, as the delay in input lines may be still incorrect, but we can state if we receive SOS or something else.
- We find the range of clock phases in which the FE ASIC receives correctly the K28.1 character and set the clock phase in the center of this range (if there are multiple ranges, we select the widest one).

Link synchronization 3

- We know, that FE ASIC sends K28.1, so now we can adjust delays in input lines.
- We scan all possible values of delays and for each line we find the delay range(s) in which the GBTx correctly receives K28.1
- We set the input line delay in each line to the center of the widest area of correct reception.

Link synchronization 4

- To leave the synchronization mode we use another non-standard sequence, easily distinguishable from other words transmitted via 8b/10b encoded link
- The sequence is "1100_1111_1100_0000_1100" (EOS-End of Synchronization)
- When the FE ASIC receives this sequence, it leaves the synchronization mode and responds with the same sequence
- When GBTx receives EOS in each input lines, we stop sending EOS sequences and start sending normal frames

Protocol implementation in the FPGA IP core

- Transmission of dowlink commands and reception of acknowledgments and responses is handled in hardware
- Synchronization of the link is handled in software. Only detectors of sequences used during the synchronization are implemented in FPGA. Transmitter used for downlink commands is also reused for transmission of special sequences
- Reception of readout data is handled in hardware

Thank you for your attention!