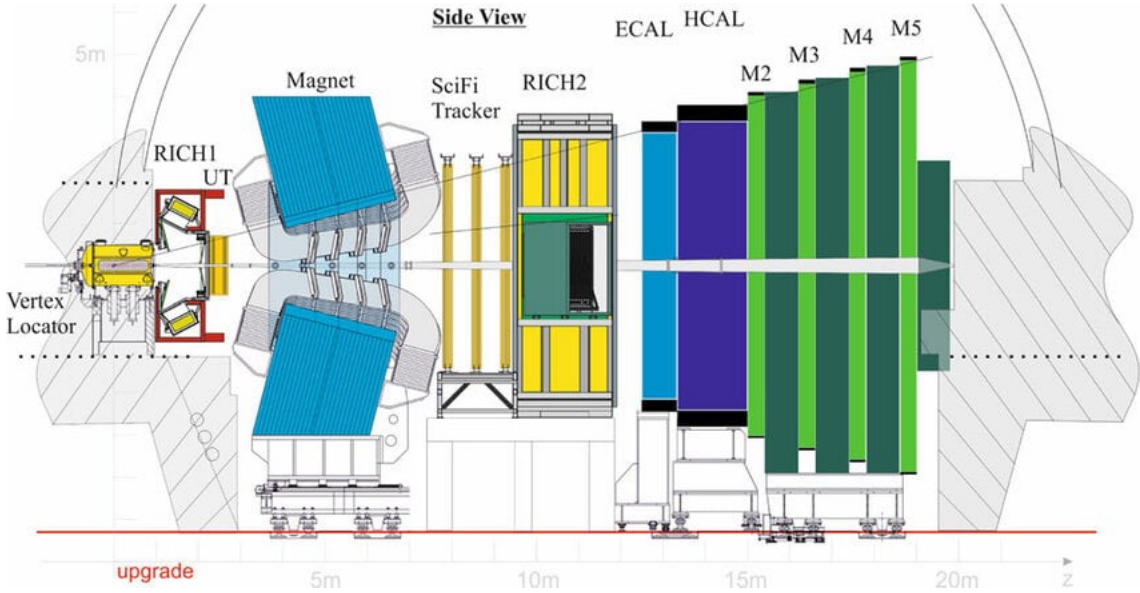
Data acquisition for HV-CMOS sensors

Hussain Al Saleh, Jon Webster, Pratik Gheewalla, Daniel Foulds-Holt, Miles Armour.

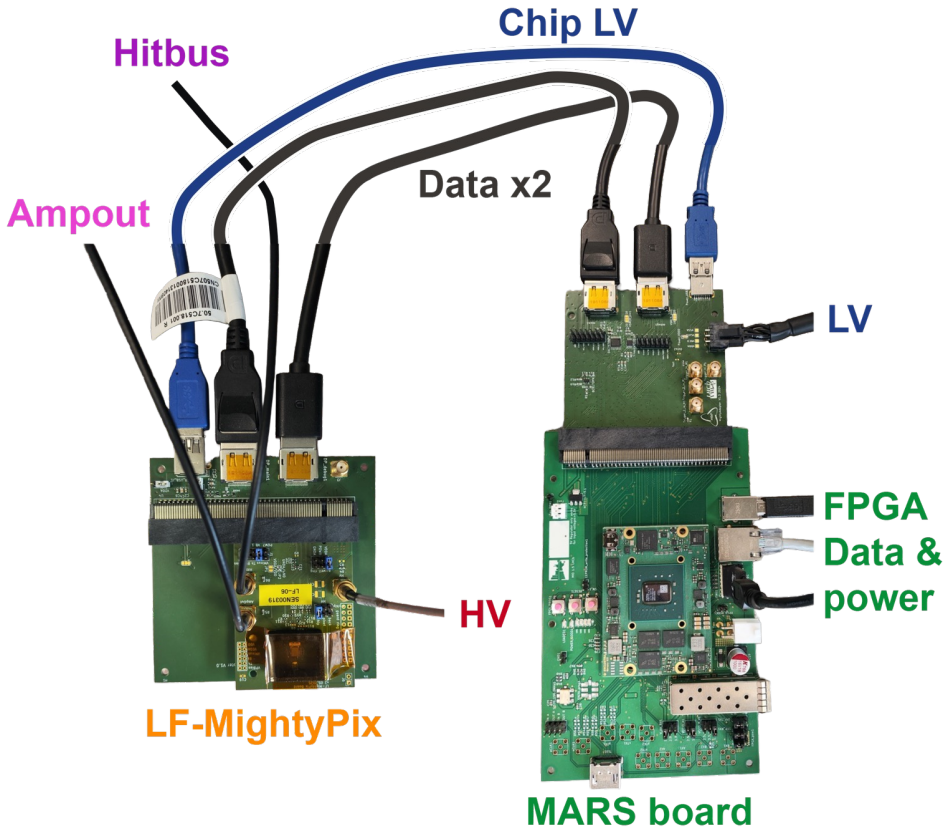
LF-MightyPix

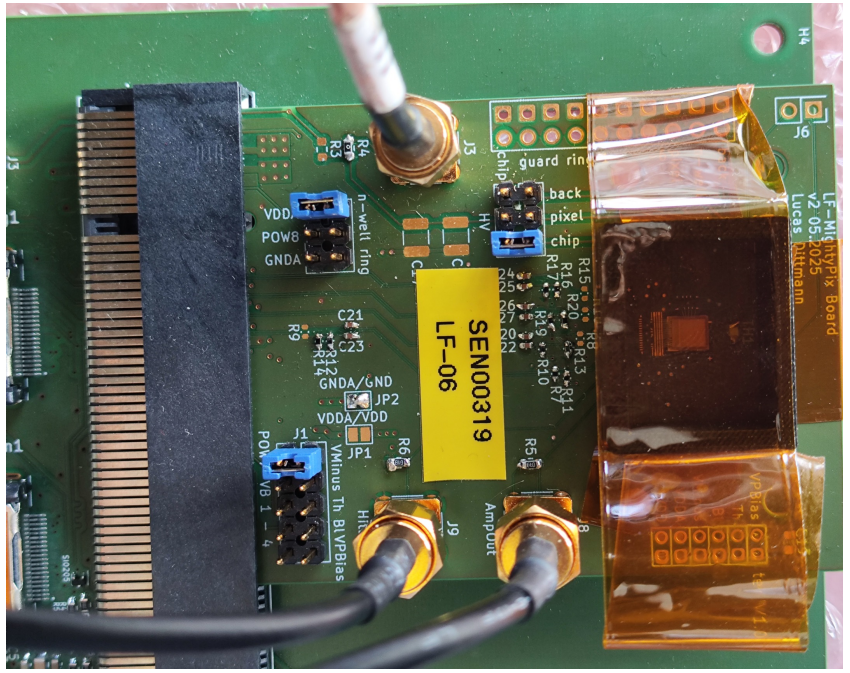
- HV-CMOS replacing the inner part of SciFi (Scintillating Fibre) Tracker.
- Few prototypes: we have LF-MightyPix.
- Chip size: 3.5 mm ×4 mm.
- Matrix Size: 28×23 (columns×rows)

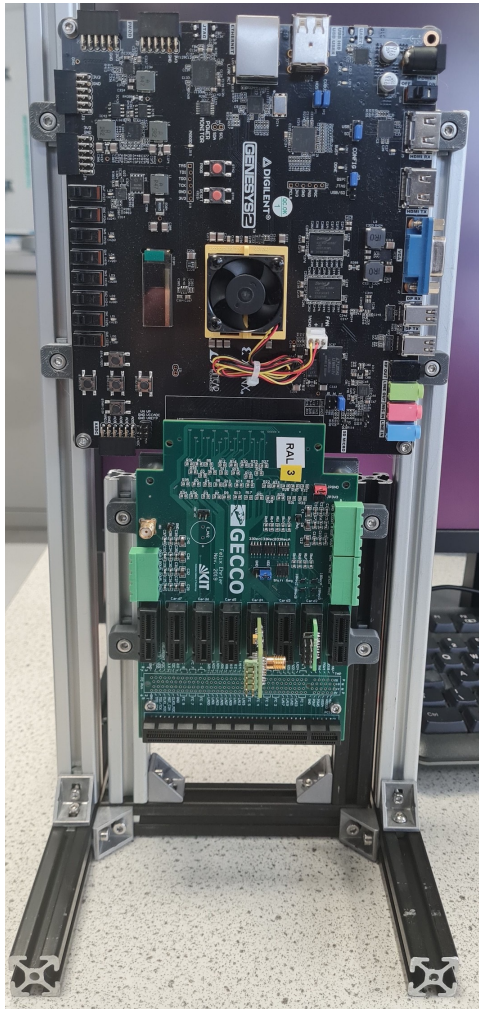


DAQ systems

- Two system are developed/being developed for DAQ:
  - GECCO: Genesys2 Development board + C++ interface to PC.
  - MARS: Mercury KX1 FPGA + Python via BASIL package.







What the DAQ Do?

1) Configure the chip.
2) Send control commands.
3) Read data out of the sensor.



Chip to FPGA

No.	Name	Type	Description
B1	vdda	Power	VDD of Pixel Matrix and Global DACs (1.8V)
B2	sc1	Digital In/Output	I2C clock in/output
B3	sda	Digital In/Output	I2C data in/output
B4	so	Digital Output	Output of the Shift-Registers
B5	rb	Digital Input	Read-Only Flag of Shift-Registers
B6	clk40p	Digital Input(LVDS)	Bunch-crossing clock, terminated in chip
B7	clk40n	Digital Input(LVDS)	Bunch-crossing clock, terminated in chip
B8	tfcp	Digital Input(LVDS)	TFC command input, terminated in chip
B9	tfcn	Digital Input(LVDS)	TFC command input, terminated in chip
B10	resn	Digital Input	Reset chip (active LOW)
B11	i2cmux	Digital Input	Switch for the I2C interface
B12	sin	Digital Input	Shift-Registers data input
B13	ld	Digital Input	Shift-Registers 1st clock
B14	ck1	Digital Input	Shift-Registers 1st clock
B15	ck2	Digital Input	Shift-Registers load signal
B16	ckextp	Digital Input(LVDS)	Data Out clock, terminated in chip
B17	ckextn	Digital Input(LVDS)	Data Out clock, terminated in chip
B18	doutp	Digital Output(LVDS)	Data Out
B19	doutn	Digital Output(LVDS)	Data Out
B20	gnd	Power	Ground of Hit-buffer and digital periphery
B21	vdd	Power	VDD of Hit-buffer and digital periphery
B22	hb	Digital Output	Hit Bus output
B23	inj	Analog Input	Test pulse input
B24	ao	Analog Output	Output monitor of CSA
B25	VMinusPix	Analog Input	VMinus of Pixel Matrix (typ 1V-500mV)
B26	BL	Analog Output	Monitor for BL
B27	Vth	Analog Output	Monitor for Th
B28	VPBias	Analog Output	global DACs monitor. Add a decoupling C to VDDA on the PCB. Draw current to overwrite
B29	gnda	Power	Ground of Pixel Matrix and Global DACs (0V)
B30	vdda	Power	VDD of Pixel Matrix and Global DACs (1.8V)

Configuration and Control

- The chip can be configured in two ways:
  - Direct write to the shift registers.
  - Using I2C to propagate the configuration.
- The control can have two forms:
  - DAQ dependant: Driver for MARS through Ethernet and JTAG (USB) for GECCO.
  - DAQ independent: TFC.
- The TFC is a custom LHCb communication protocol:
  - Send Sync pattern
  - Fast reset of the chip readout.
  - Bunch-crossing reset
  - Etc.

Data Path

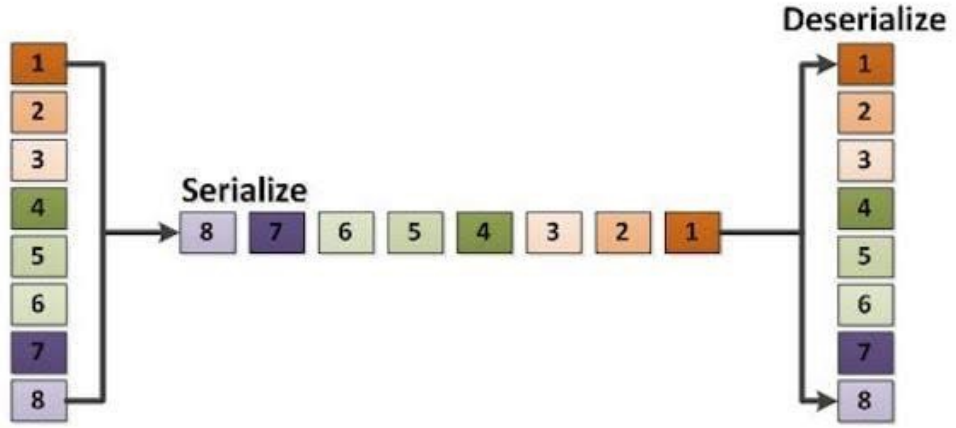
Data going through the FPGA from the chip passes through:

1) Deserialiser
2) Alignment + Descrambler
3) Readout FSM

After all of that, Data goes to the PC.

•Deserialiser

- The chip sends 32 bits word serially:
  1. At 1.28Gbps
  2. In Double Data Rate (DDR) mode
  3. Width of 1 bit.
- We need to reconstruct these 32 bits words => Deserialiser.



Alignment

- Data is continuously being received from the sensor

... 1011 0110 1011 0000 1000 1100 0111 1010 ...



Which bit is first of the 32 bits word?

- The alignment is required to confirm that the word is read correctly.
- How to do it? Compare to a known pattern! For 4 bits alignment take 1011.
- Shift the word to that pattern.
- Do it multiple times.

Descrambler

- It an optional module to maintain signal quality.
- How:
  - It makes the data looks more randomised.
  - Avoid long sequences of identical bits.
- The chip scramble the data with a known pattern.
- The receiver use the same pattern to return back to the original data.
- Eg. Chip: 1100 XOR 1011 = 0111 (1011: Scramble Pattern) FPGA: 0111 XOR 1011 = 1100 (1011: Scramble Pattern)
- The LF-MightyPix uses the same pattern as VeloPix.

Readout FSM

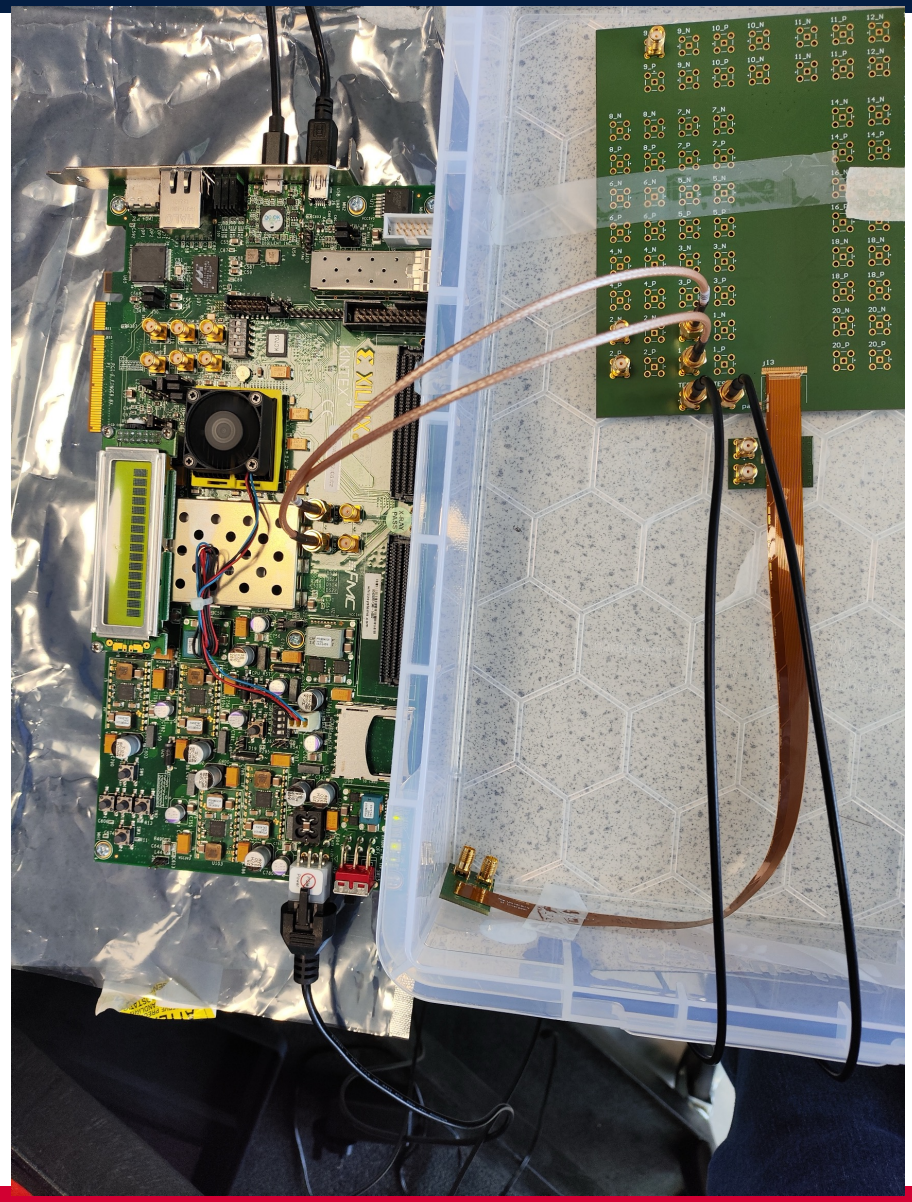
- Data aligned => valid data.
- valid data can be:
  - Ideal/Sync patterns
  - Single hit data.
  - Multi-hit data.
- The FSM takes the raw data and do the initial decoding.
- Decoded data is then sent to the PC to be decoded further and to be analised.

Another Project: Data Flux

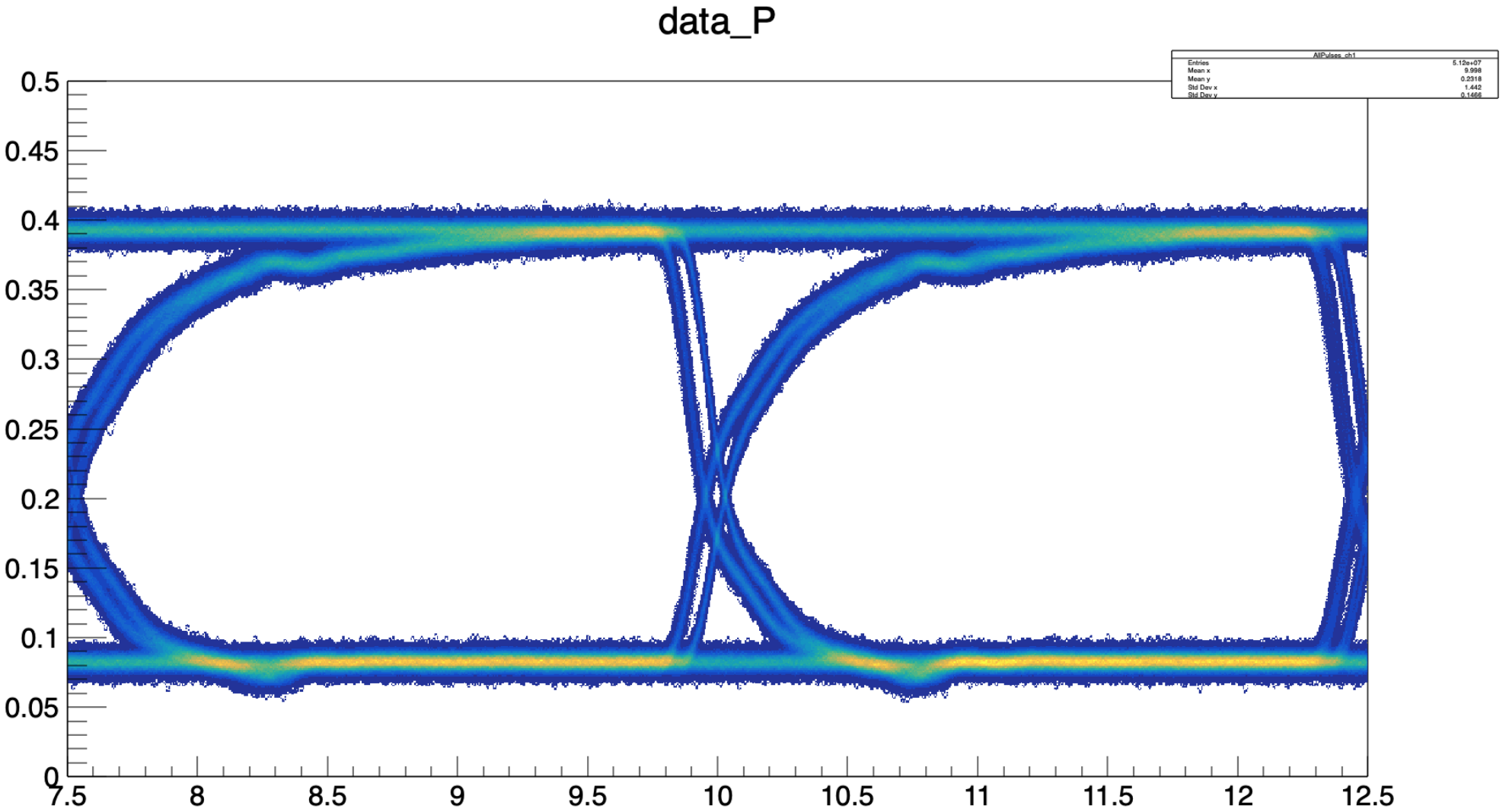
- A breakout board has been designed (Signal in to the cable).
- Output boards are designed (to carry the signal out of the cable).
- All board have been wire-bonded (Bonds are fragile handle with care).
- Random data have been sent at 400Mbps to check the bonds.
- The signal have been seen on the output boards.

Setup

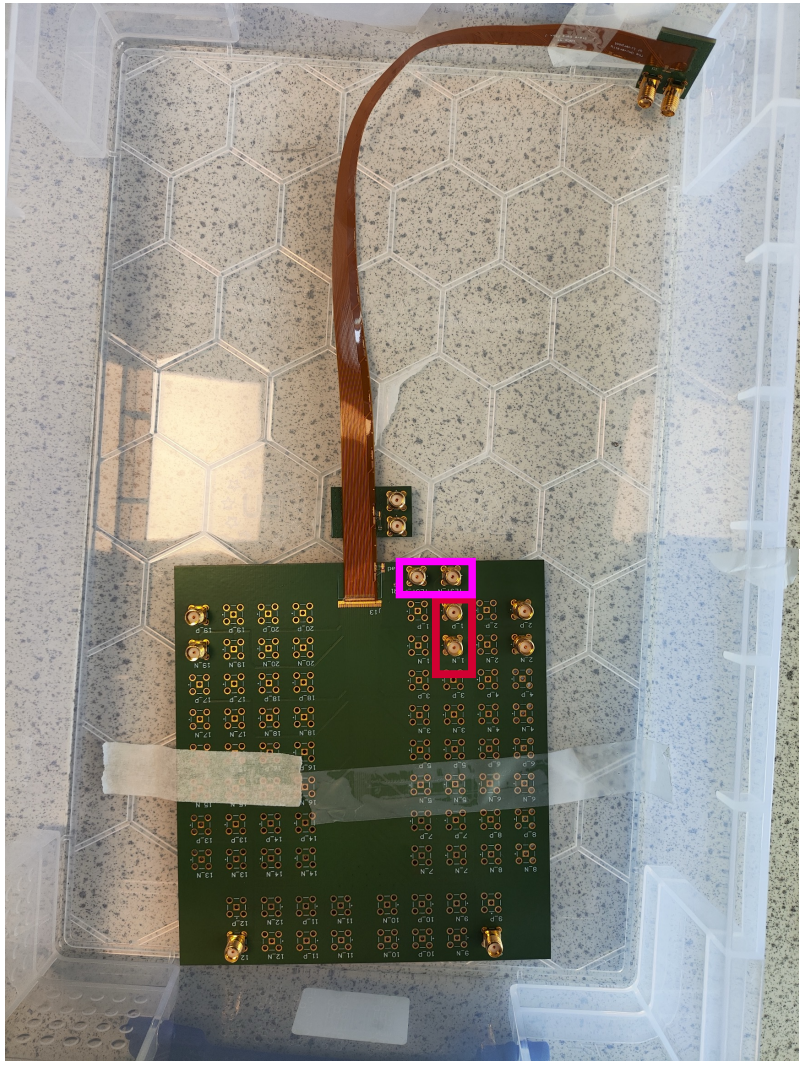
- Breakout Board
- FPGA -> 400Mbps PRNG32
- 4 SMA -> 2 FPGA signal + 2 Cable Signal
- Oscilloscope -> Signal Shape.



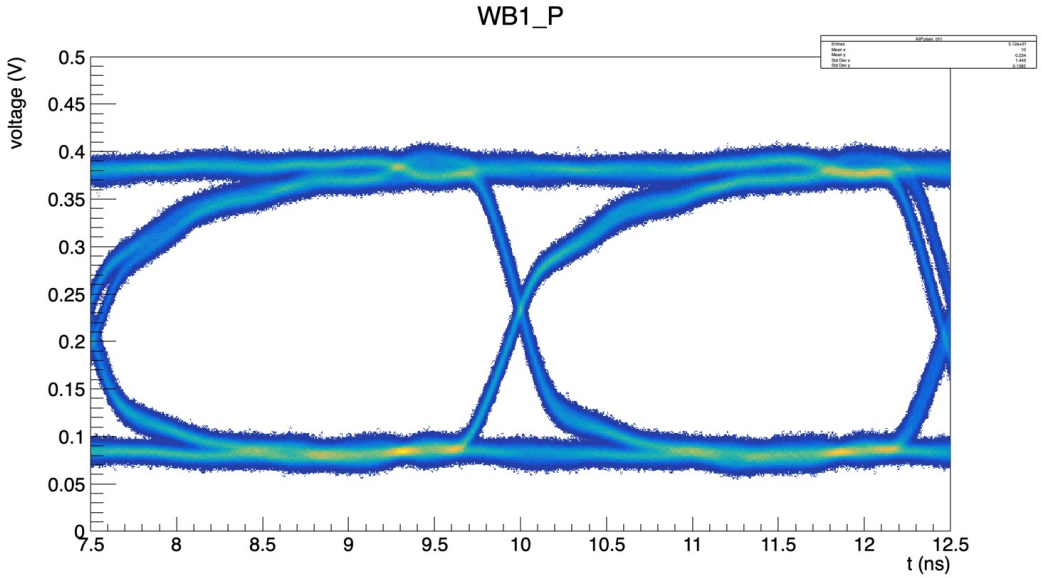
Ref Shape (FPGA to Oscilloscope Directly)

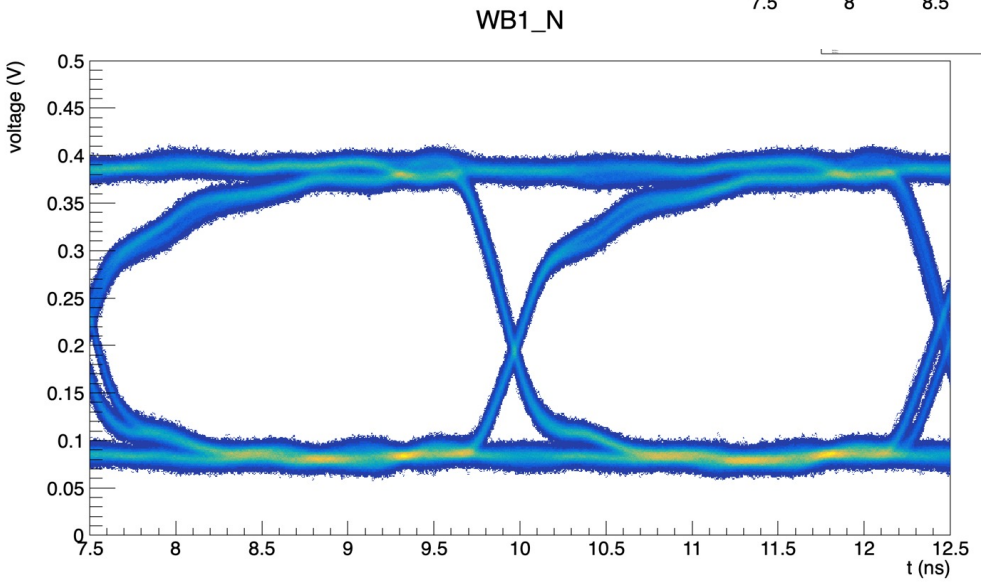


Bond 1

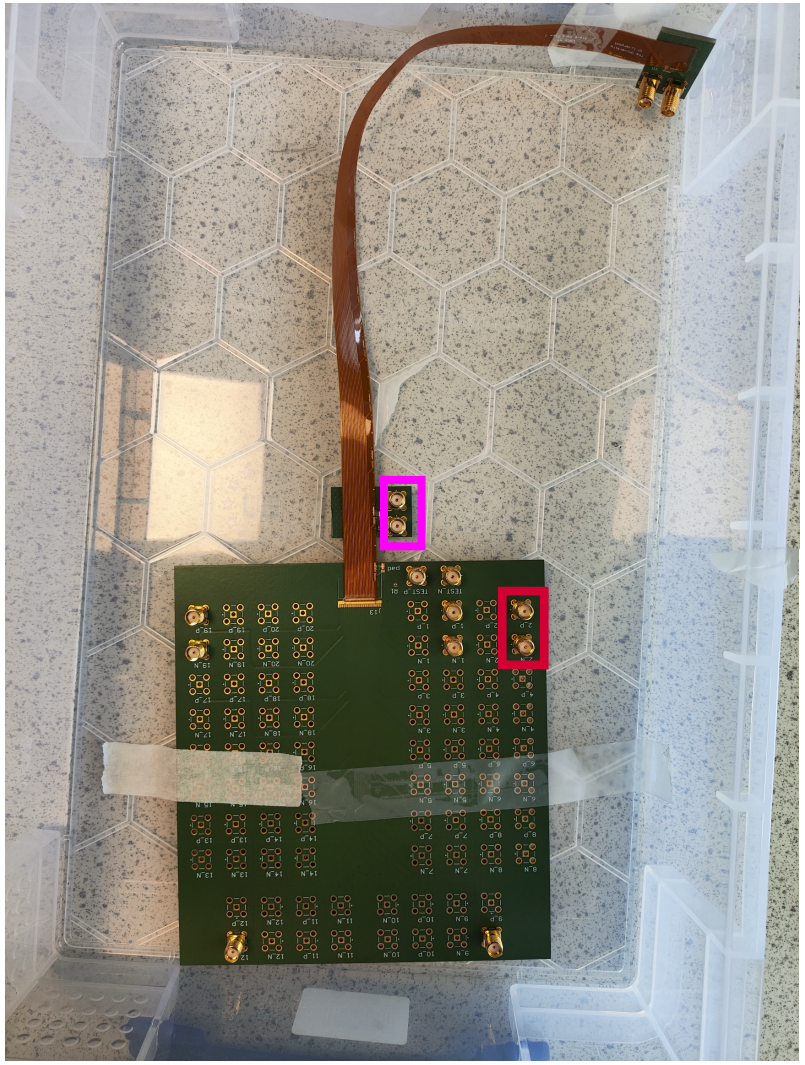


Red : FPGA
Pink: Oscilloscope

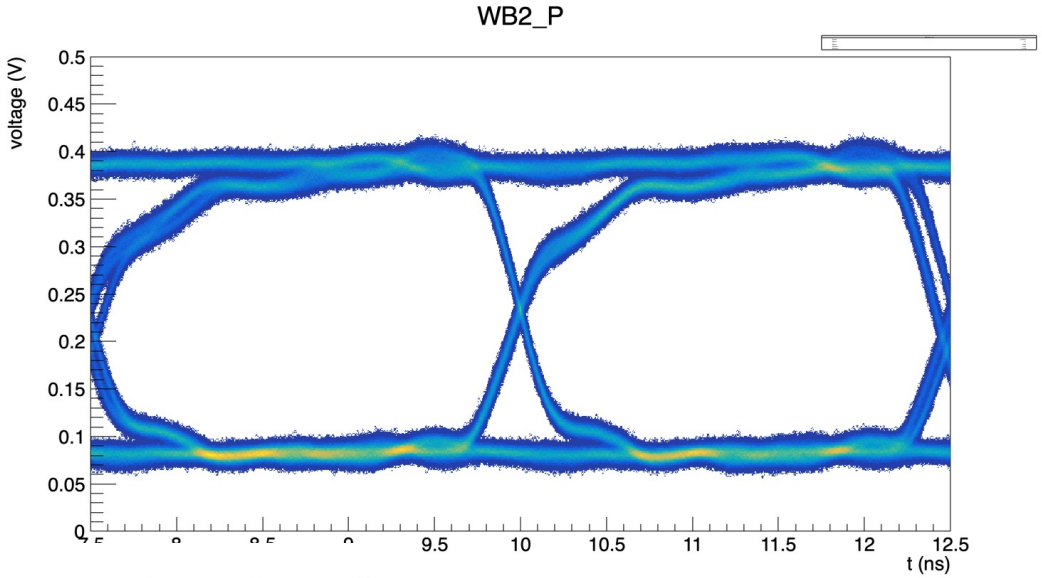


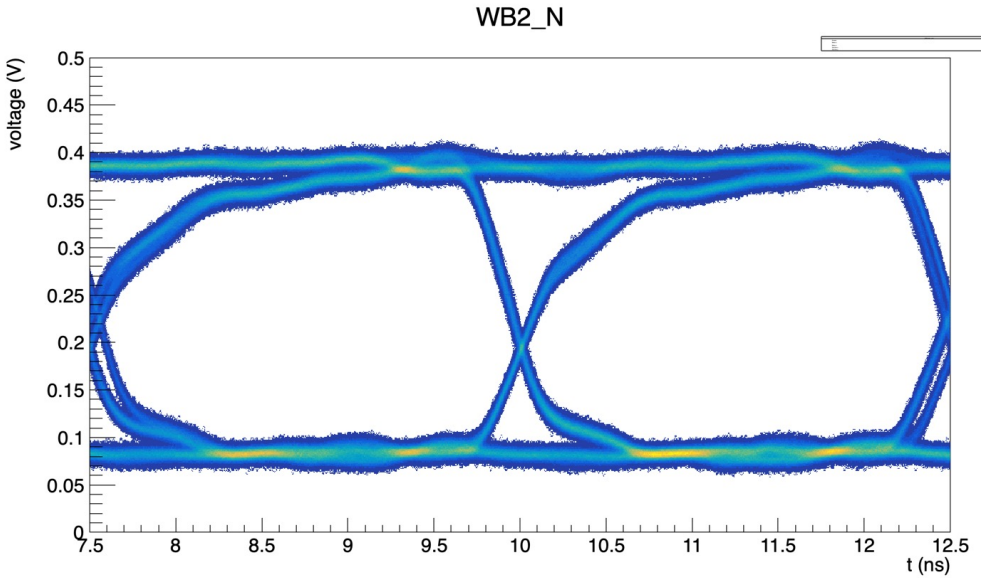


Bond 2

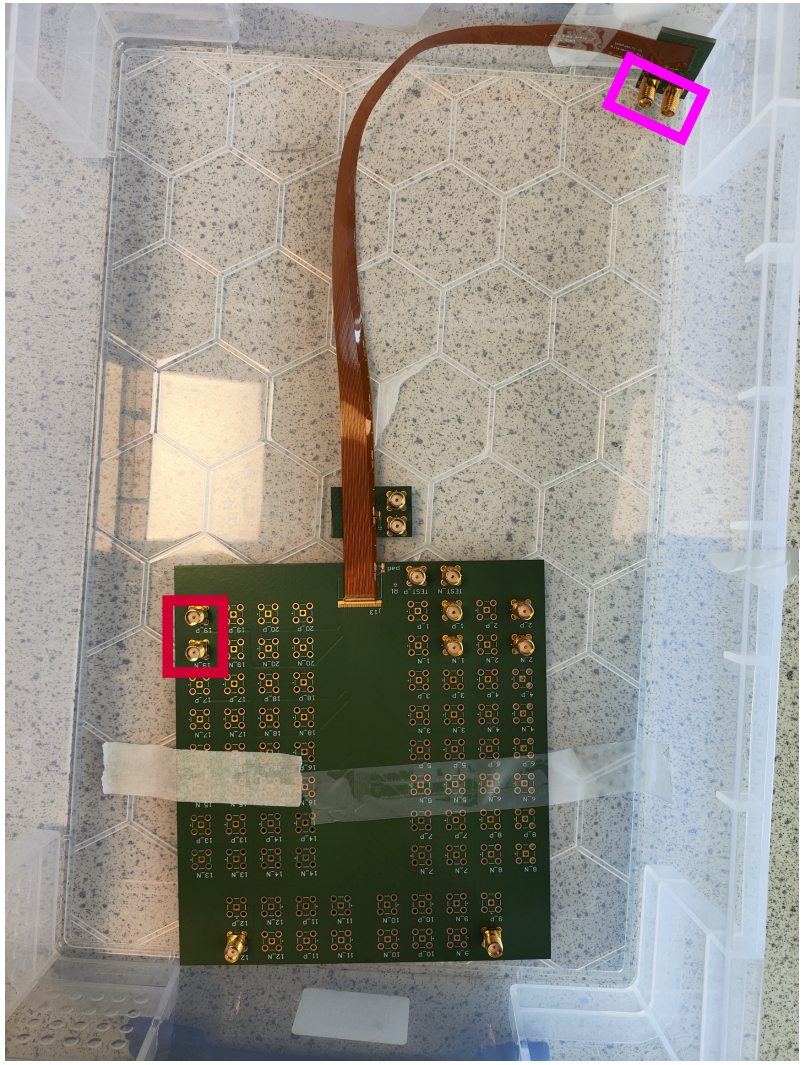


Red : FPGA
Pink: Oscilloscope

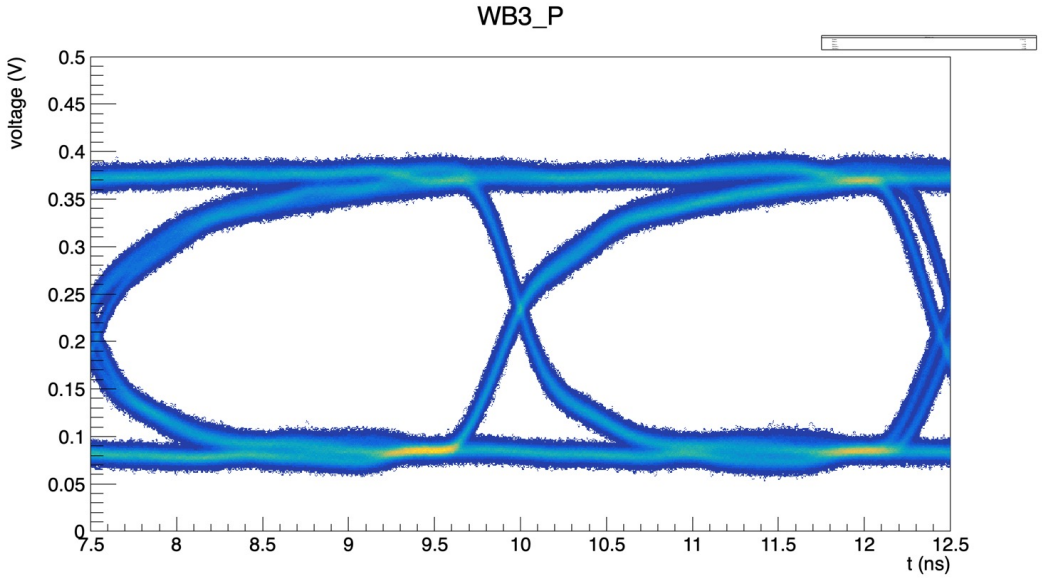


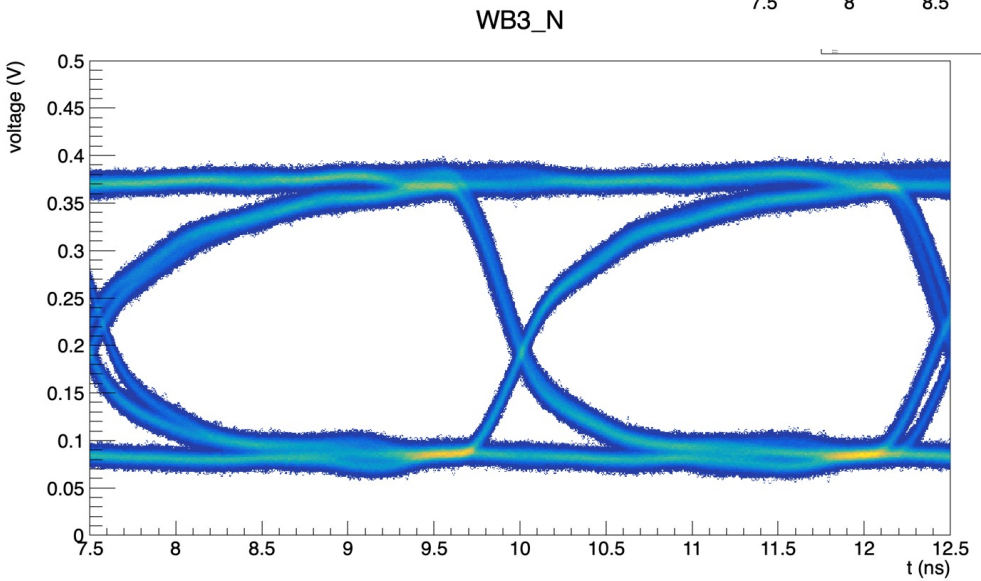


Bond 3

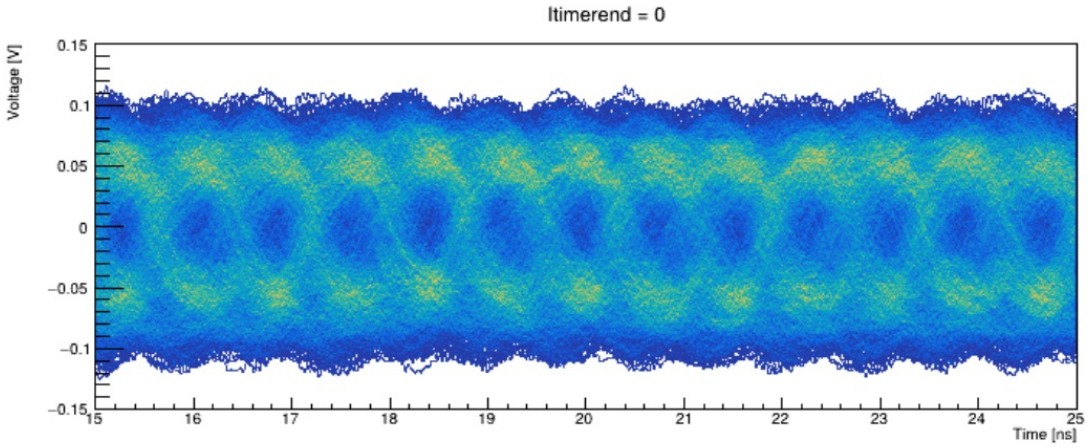


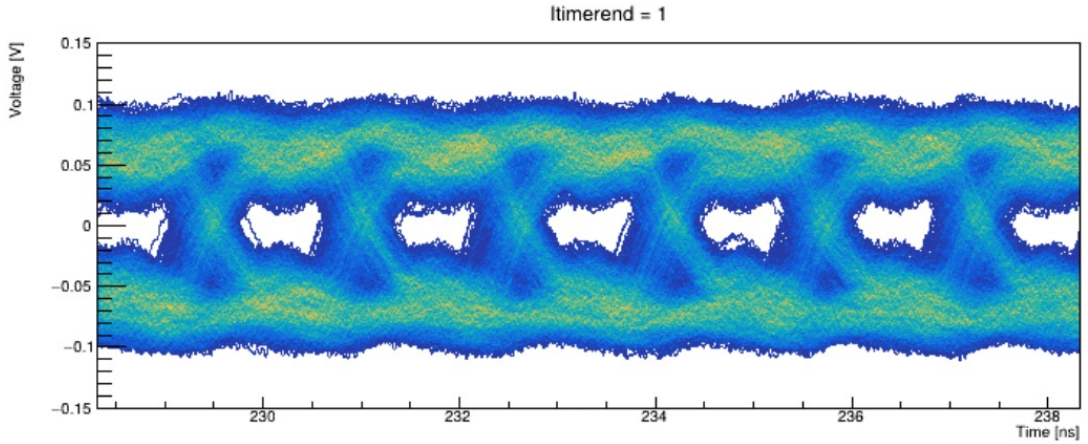
Red : FPGA
Pink: Oscilloscope

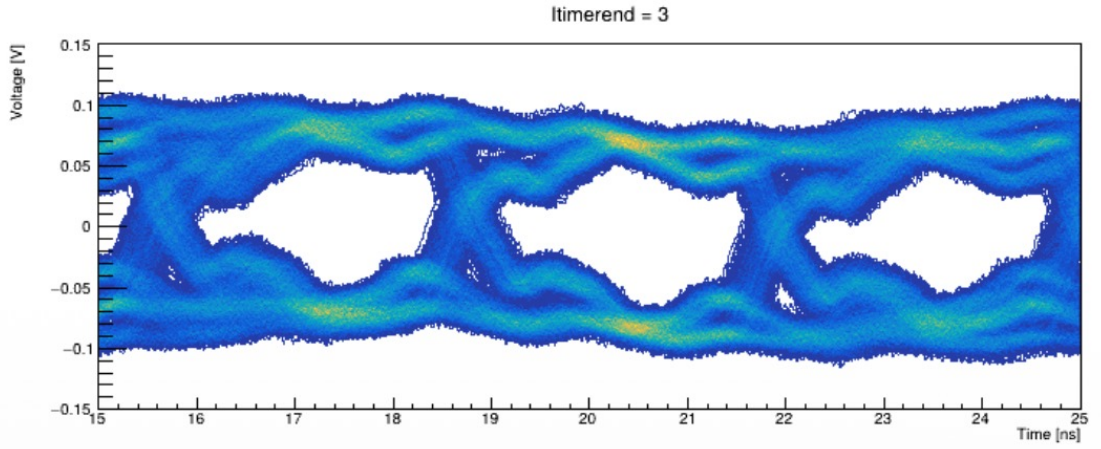




MARS Signal Integrity







Conclusion and Outlook

- The LF-MightyPix firmware is complex.
- We are building expertise in firmware development and in general for HV-CMOS.
- We are aiming to reach levels at which we will be able to develop the readout system for particle telescope using HVCOMS like LF and its successor/s.

Backup

Scramble Sequence

```
assign data_out[29] = data_in[29] ^ data_prev[29] ^ data_prev[0] ^ data_prev[5] ^ data_prev[3];
assign data_out[28] = data_in[28] ^ data_prev[28] ^ data_in[29] ^ data_prev[4] ^ data_prev[2];
assign data_out[27] = data_in[27] ^ data_prev[27] ^ data_in[28] ^ data_prev[3] ^ data_prev[1];
assign data_out[26] = data_in[26] ^ data_prev[26] ^ data_in[27] ^ data_prev[2] ^ data_prev[0];
assign data_out[25] = data_in[25] ^ data_prev[25] ^ data_in[26] ^ data_prev[1] ^ data_in[29];
assign data_out[24] = data_in[24] ^ data_prev[24] ^ data_in[25] ^ data_prev[0] ^ data_in[28];
assign data_out[23] = data_in[23] ^ data_prev[23] ^ data_in[24] ^ data_in[29] ^ data_in[27];
assign data_out[22] = data_in[22] ^ data_prev[22] ^ data_in[23] ^ data_in[28] ^ data_in[26];
assign data_out[21] = data_in[21] ^ data_prev[21] ^ data_in[22] ^ data_in[27] ^ data_in[25];
assign data_out[20] = data_in[20] ^ data_prev[20] ^ data_in[21] ^ data_in[26] ^ data_in[24];
assign data_out[19] = data_in[19] ^ data_prev[19] ^ data_in[20] ^ data_in[25] ^ data_in[23];
assign data_out[18] = data_in[18] ^ data_prev[18] ^ data_in[19] ^ data_in[24] ^ data_in[22];
assign data_out[17] = data_in[17] ^ data_prev[17] ^ data_in[18] ^ data_in[23] ^ data_in[21];
assign data_out[16] = data_in[16] ^ data_prev[16] ^ data_in[17] ^ data_in[22] ^ data_in[20];
assign data_out[15] = data_in[15] ^ data_prev[15] ^ data_in[16] ^ data_in[21] ^ data_in[19];
assign data_out[14] = data_in[14] ^ data_prev[14] ^ data_in[15] ^ data_in[20] ^ data_in[18];
assign data_out[13] = data_in[13] ^ data_prev[13] ^ data_in[14] ^ data_in[19] ^ data_in[17];
assign data_out[12] = data_in[12] ^ data_prev[12] ^ data_in[13] ^ data_in[18] ^ data_in[16];
assign data_out[11] = data_in[11] ^ data_prev[11] ^ data_in[12] ^ data_in[17] ^ data_in[15];
assign data_out[10] = data_in[10] ^ data_prev[10] ^ data_in[11] ^ data_in[16] ^ data_in[14];
assign data_out[9] = data_in[9] ^ data_prev[9] ^ data_in[10] ^ data_in[15] ^ data_in[13];
assign data_out[8] = data_in[8] ^ data_prev[8] ^ data_in[9] ^ data_in[14] ^ data_in[12];
assign data_out[7] = data_in[7] ^ data_prev[7] ^ data_in[8] ^ data_in[13] ^ data_in[11];
assign data_out[6] = data_in[6] ^ data_prev[6] ^ data_in[7] ^ data_in[12] ^ data_in[10];
assign data_out[5] = data_in[5] ^ data_prev[5] ^ data_in[6] ^ data_in[11] ^ data_in[9];
assign data_out[4] = data_in[4] ^ data_prev[4] ^ data_in[5] ^ data_in[10] ^ data_in[8];
assign data_out[3] = data_in[3] ^ data_prev[3] ^ data_in[4] ^ data_in[9] ^ data_in[7];
assign data_out[2] = data_in[2] ^ data_prev[2] ^ data_in[3] ^ data_in[8] ^ data_in[6];
assign data_out[1] = data_in[1] ^ data_prev[1] ^ data_in[2] ^ data_in[7] ^ data_in[5];
assign data_out[0] = data_in[0] ^ data_prev[0] ^ data_in[1] ^ data_in[6] ^ data_in[4];
assign data_out[31:30] = data_in[31:30];
```

Encoded Data

Idle word

Word	Bit	Name	Value	Size
0	31:30	Header	b10	2
0	29:28	Data ID	b11	2
0	27:24	Fill	0	4
0	23:0	Bin Counter	Counts	24

Sync word

Word	Bit	Name	Value	Size
0	31:30	Header	b10	2
0	29:0	Sync pattern	0	30

Single hit

Word	Bit	Name	Value	Size
0	31:30	Header	b10	2
0	29:28	Data ID	b00	2
0	27:0	Hit	Hit [40:13]	28
1	31:30	Header	b10	2
1	29:28	Data ID	b01	2
1	27	Has 2nd hit	0	1
1	26:13	Fill	0	14
1	12:0	Hit	Hit [12:0]	13

Encoded Data + Decoded Data

Multiple hits

Word	Bit	Name	Value	Size
0	31:30	Header	b10	2
0	29:28	Data ID	b00	2
0	27:0	Hit 1	Hit 1 [40:13]	28
1	31:30	Header	b10	2
1	29:28	Data ID	b01	2
1	27	Has 2nd hit	1	1
1	26:14	Hit 1	Hit 1 [12:0]	13
1	13:0	Hit 2	Hit 2 [40:27]	14
2	31:30	Header	b10	2
2	29:28	Data ID	b10	2
2	27	Fill	0	1
2	26:0	Hit 2	Hit 2 [26:0]	27

Decoded Data

Word	Bit	Name	Value	Size
0	63:56	Header	0x1	8
0	55:53	Fill	0	3
0	52:41	TS1	Timestamp	12
0	40	TS1N	LSB TS1	1
0	39:0	System TS	Timestamp	40
1	63:56	Header	0x2	8
1	55:49	Col	Column Address	7
1	48:40	Row	Row Address	9
1	39:0	Trigger TS	Timestamp	40
2	63:56	Header	0x3	8
2	55:30	Fill	0	26
2	29:24	TS2	Timestamp	6
2	23:0	Bin Counter	Counts	24