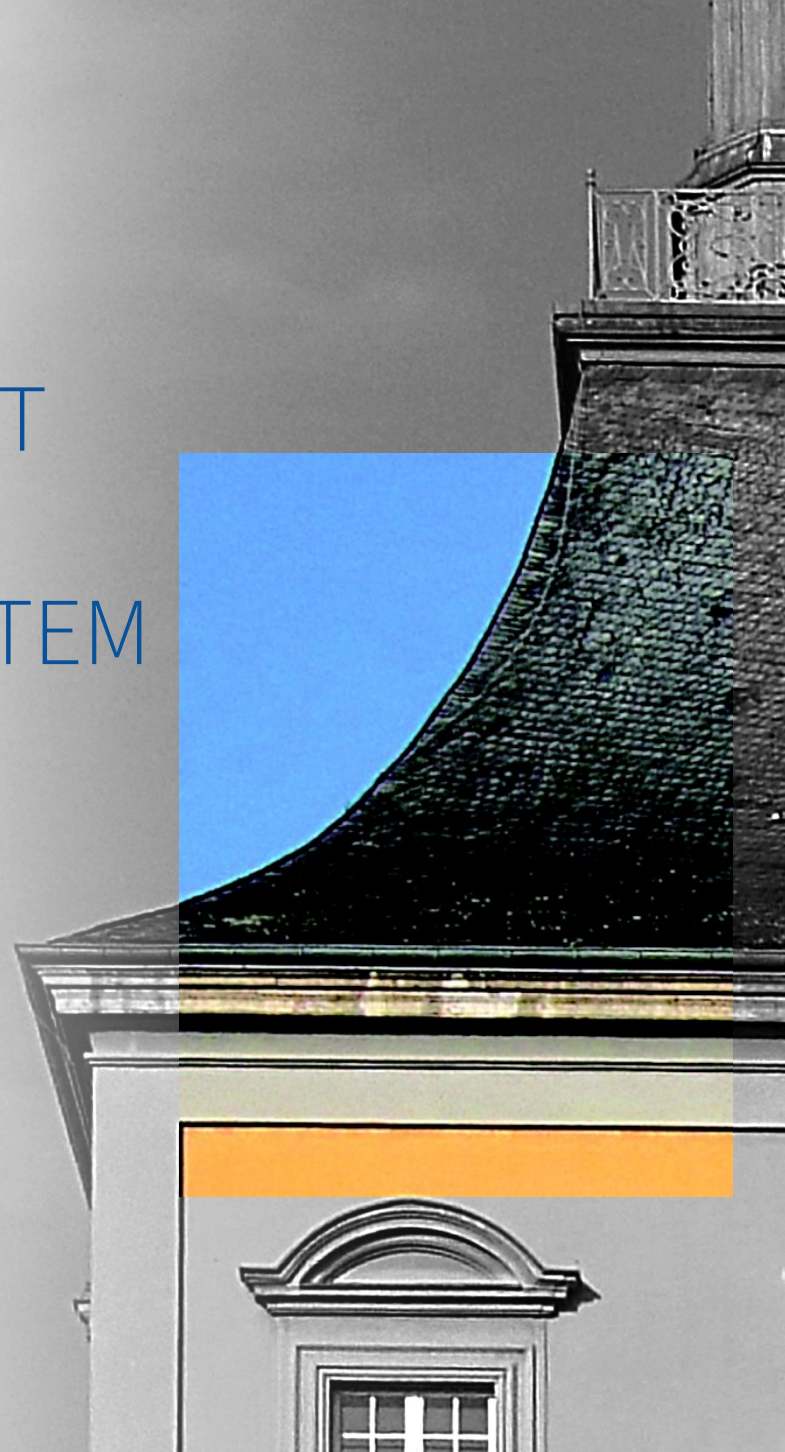
UNIVERSITÄT BONN

Physikalisches Institut

FIRMWARE DEVELOPMENT FOR THE SCALABLE READOUT SYSTEM (SRS) WITH VMM3A

Patrick Schwäbig

RD51 Collaboration Meeting June 2020



UNIVERSITÄT BONN

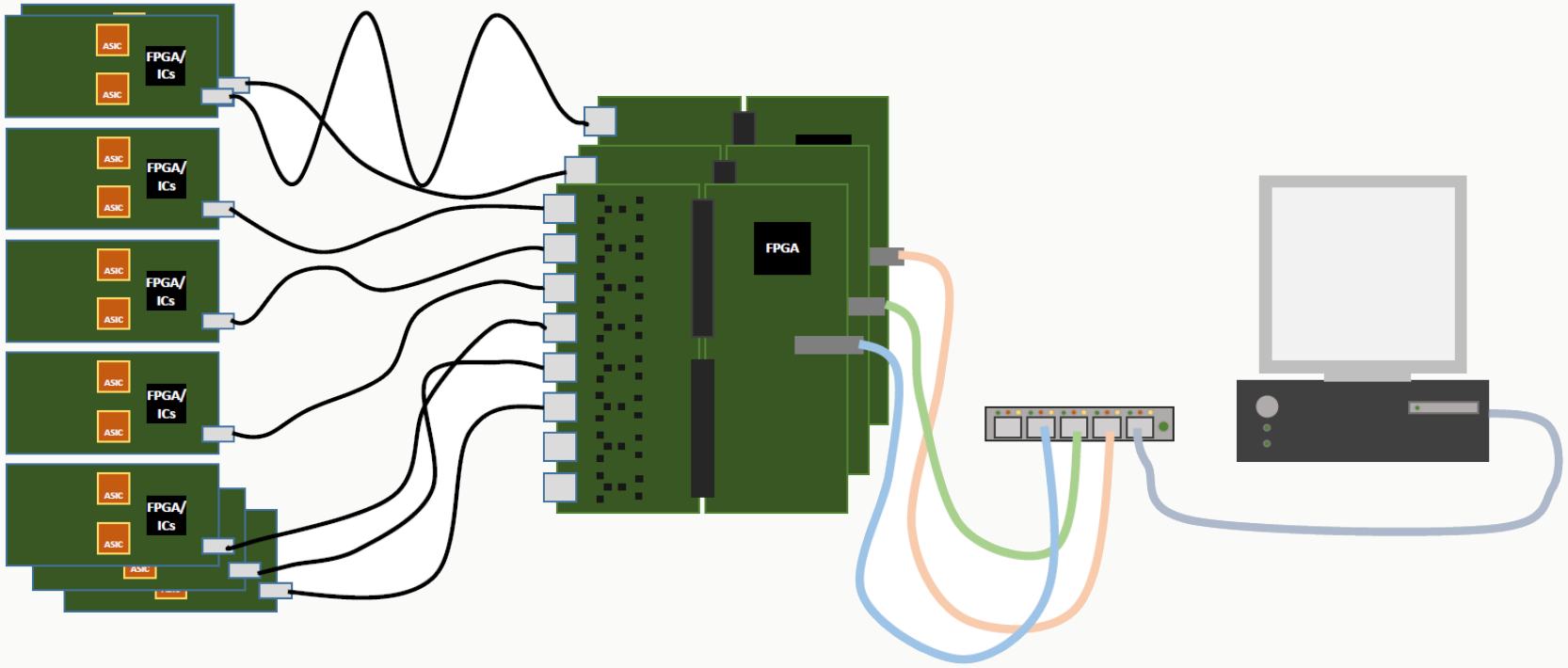
OUTLINE

- The Scalable Readout System
- The VMM
- Supported transfer rates
- Speedup of data transfer between VMM and FPGA
- Speedup of ADC conversion times
- Ongoing improvements
- Storing VMM configuration in BRAM
- Summary

UNIVERSITÄT BONN

THE SCALABLE READOUT SYSTEM (SRS)

- Versatile read-out system, invented by H. Müller
- Scalable from a few dozen to many thousand channels
- Compatible with different front-end ASICs
- Implemented: APV25 and VMM (since 2018, DOI: 10.1016/j.nima.2018.06.046)

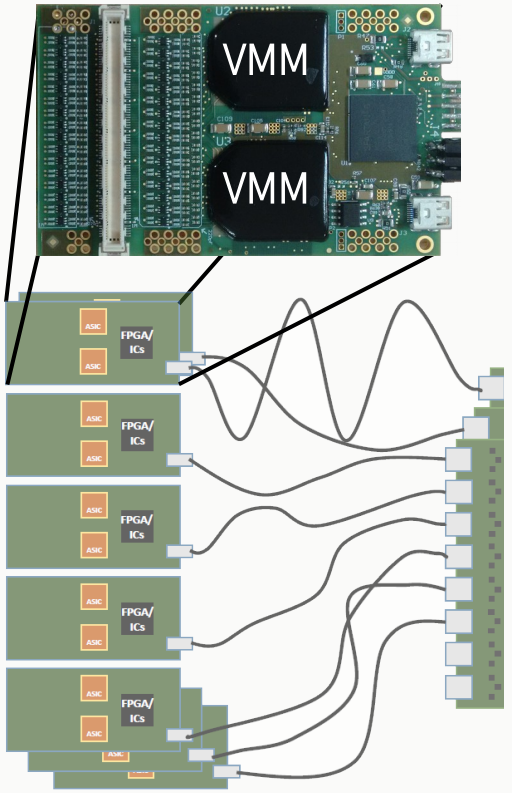


Hybrid → HDMI cable → Adapter card + FEC → Ethernet → Switch →Ethernet → PC

UNIVERSITÄT BONN

THE VMM

- Front-end ASIC for tracking detectors (DOI:10.1109/NSSMIC.2012.6551184)
- Current version: VMM3a (ATL-MUON-PROC-2019-010)
- 130 nm CMOS technology
- Highly flexible, large range of configuration parameters (e.g. adjustable gain, adjustable shaping time)
- 64 channels per VMM → 128 channels per Hybrid
- Each channel can handle up to 4 MHits/s
- Continuous readout at high rates, low electronic noise
- Used for Micromegas and sTGC detectors of the ATLAS New Small Wheel (NSW)
- Will also be used at various other experiments e.g. for the NMX instrument at ESS in Lund, Sweden

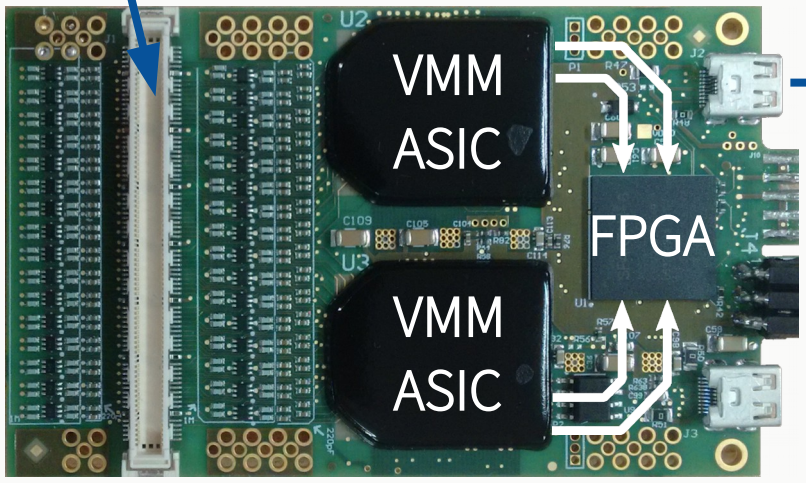


UNIVERSITÄT BONN

DATA TRANSFER BETWEEN VMM AND FPGA

- VMM reads directly from detector
- Generates 38 bits of data per hit (timing, amplitude etc.)
- Two data lanes per VMM to FPGA
- VMM: Up to 200 MHz DDR (Double Data Rate) supported
- FPGA: Currently 160 MHz DDR supported
- 200 MHz DDR would allow 800 Mb/s data transfer
- But: Was unstable in previous tests
  → Check maximum supported frequency

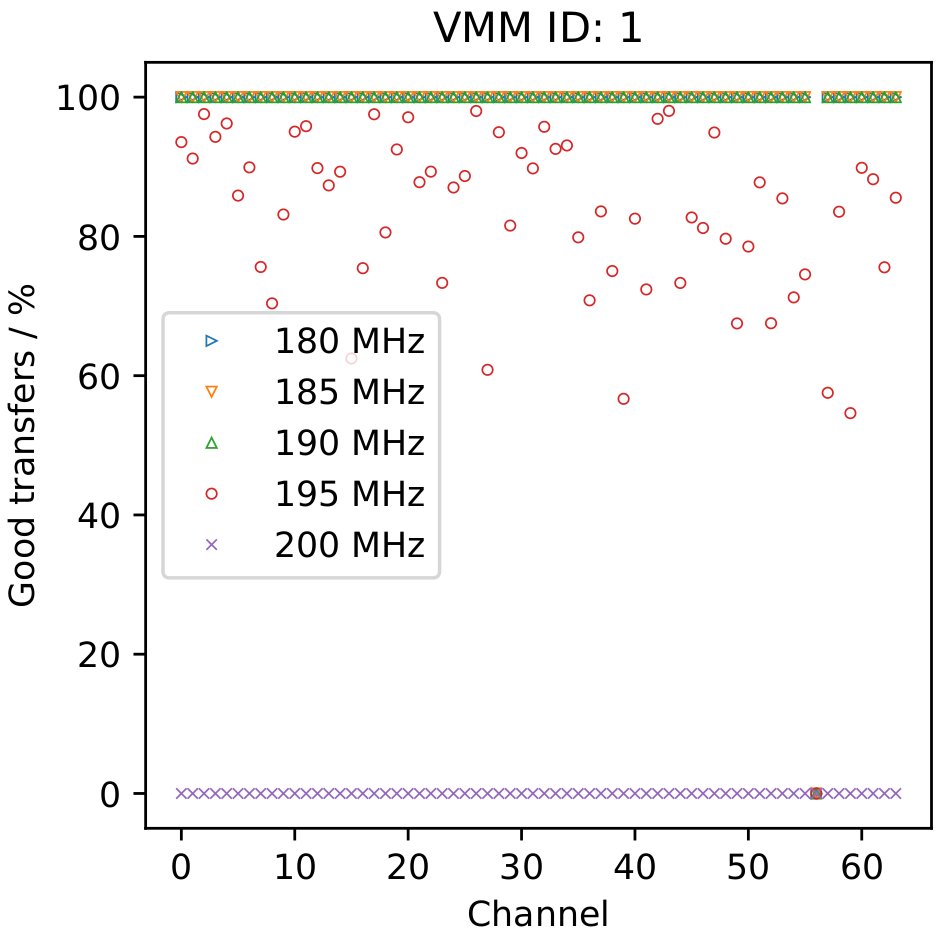
Plugged onto detector

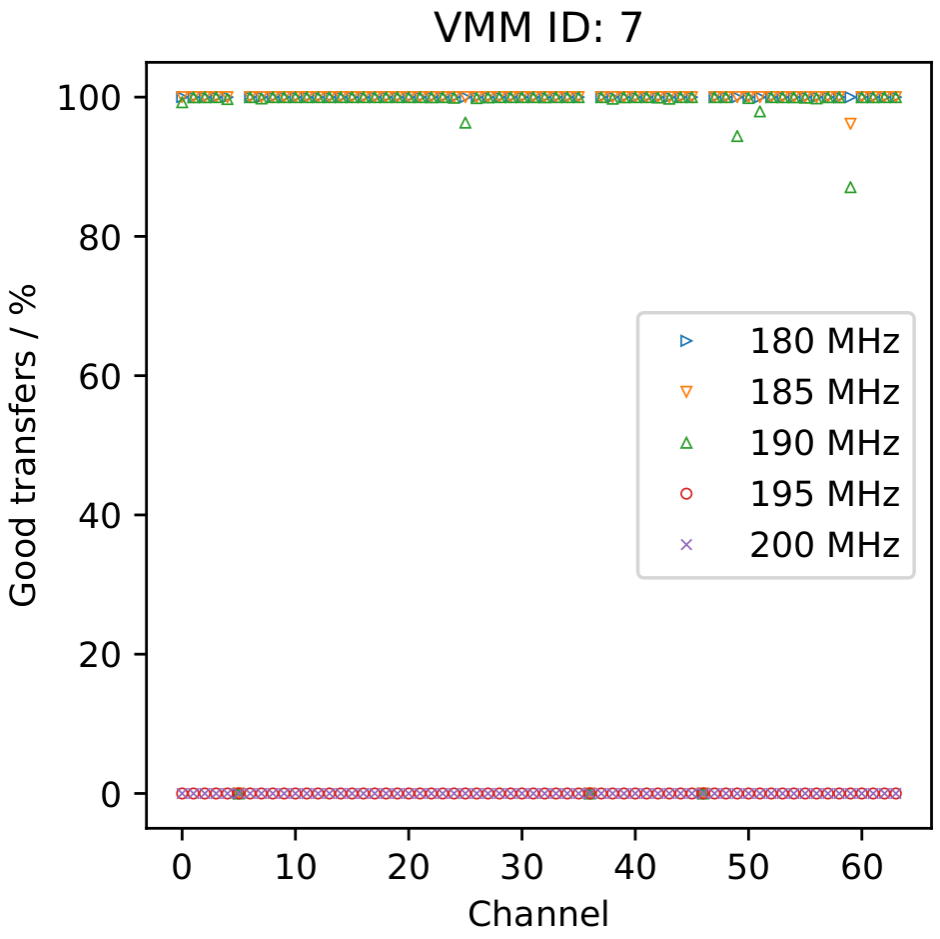


HDMI to Adapter card

UNIVERSITÄT BONN

DIFFERENCES IN SUPPORTED DATA RATES



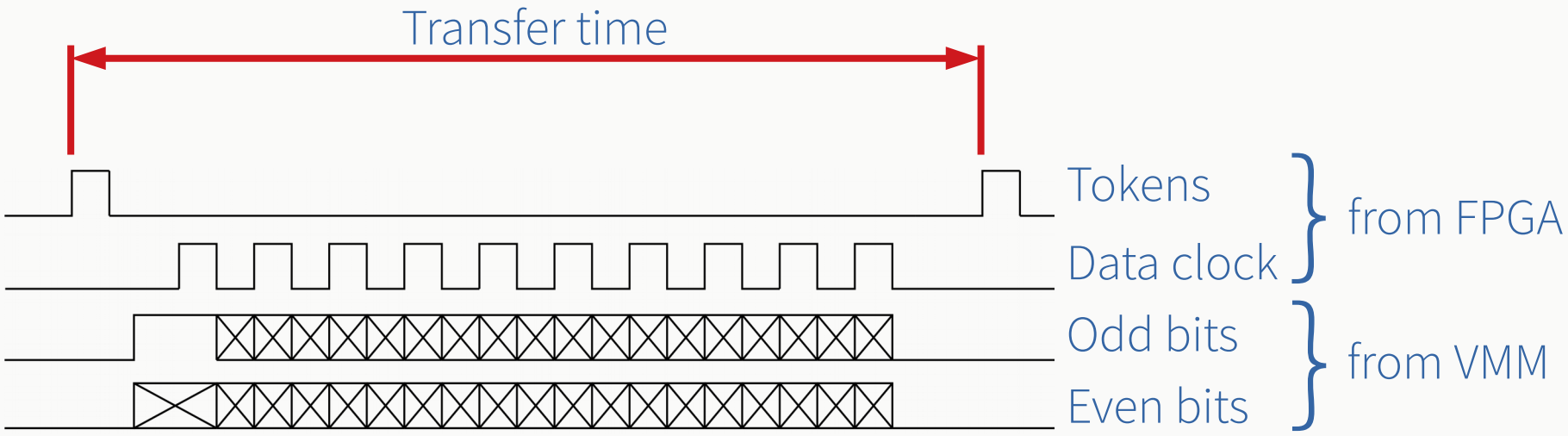


(All at double data rate)

UNIVERSITÄT BONN

SPEEDING UP DATA TRANSFER BETWEEN VMM AND FPGA

→ 180 MHz DDR works!

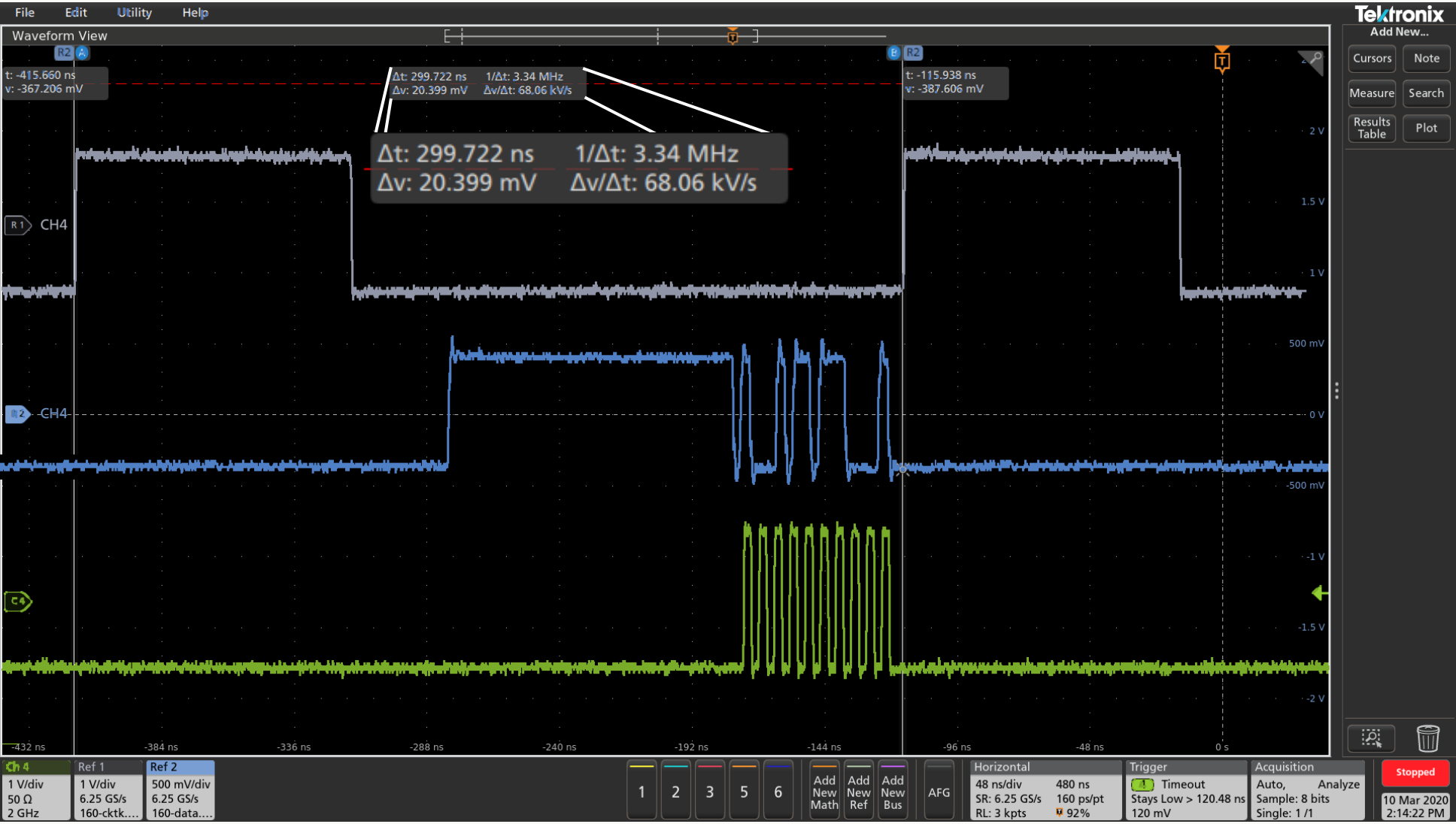


Aim: Stable 180 MHz transfer while reducing transfer time as far as possible

- Rewrote VHDL firmware block responsible for data transfers
- Using the IOB deserializer instead of a SDR register (allows higher data rate)
- Using Spartan-6 Dynamic Reprogramming Port (DRP) of PLL (changeable data clock)

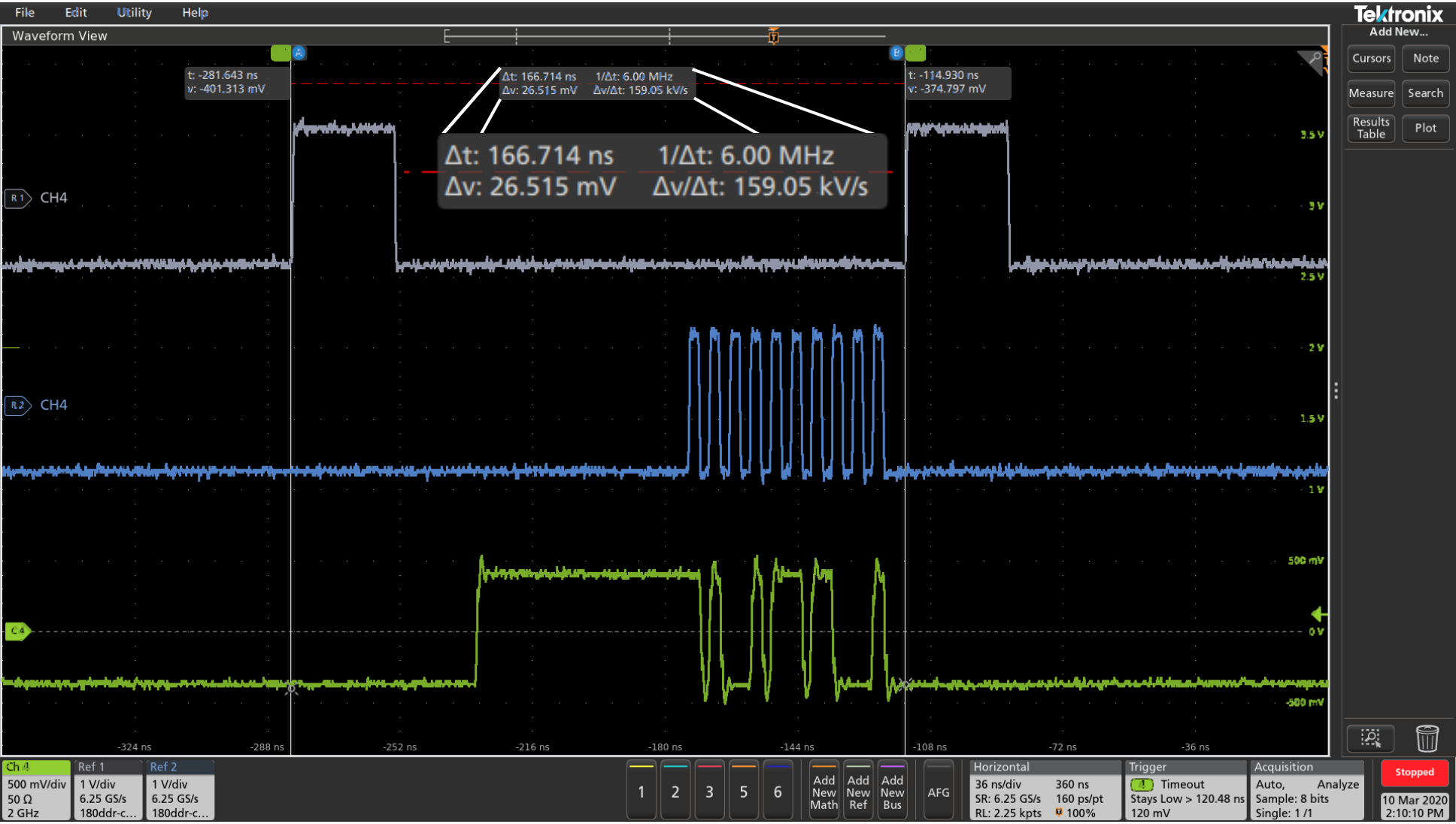
UNIVERSITÄT BONN

OLD FIRMWARE: 160 MHZ



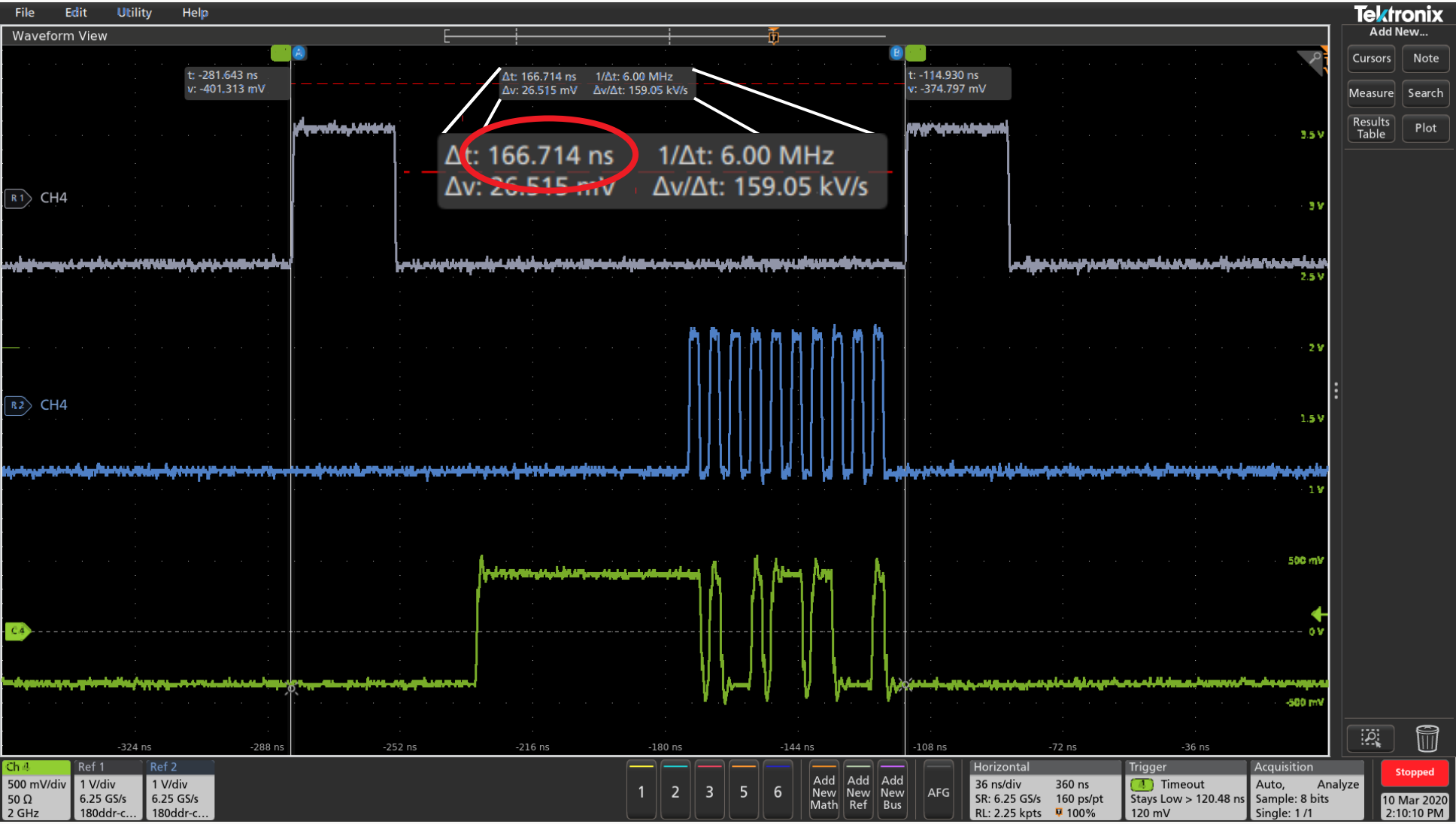
UNIVERSITÄT BONN

NEW FIRMWARE: 180 MHZ



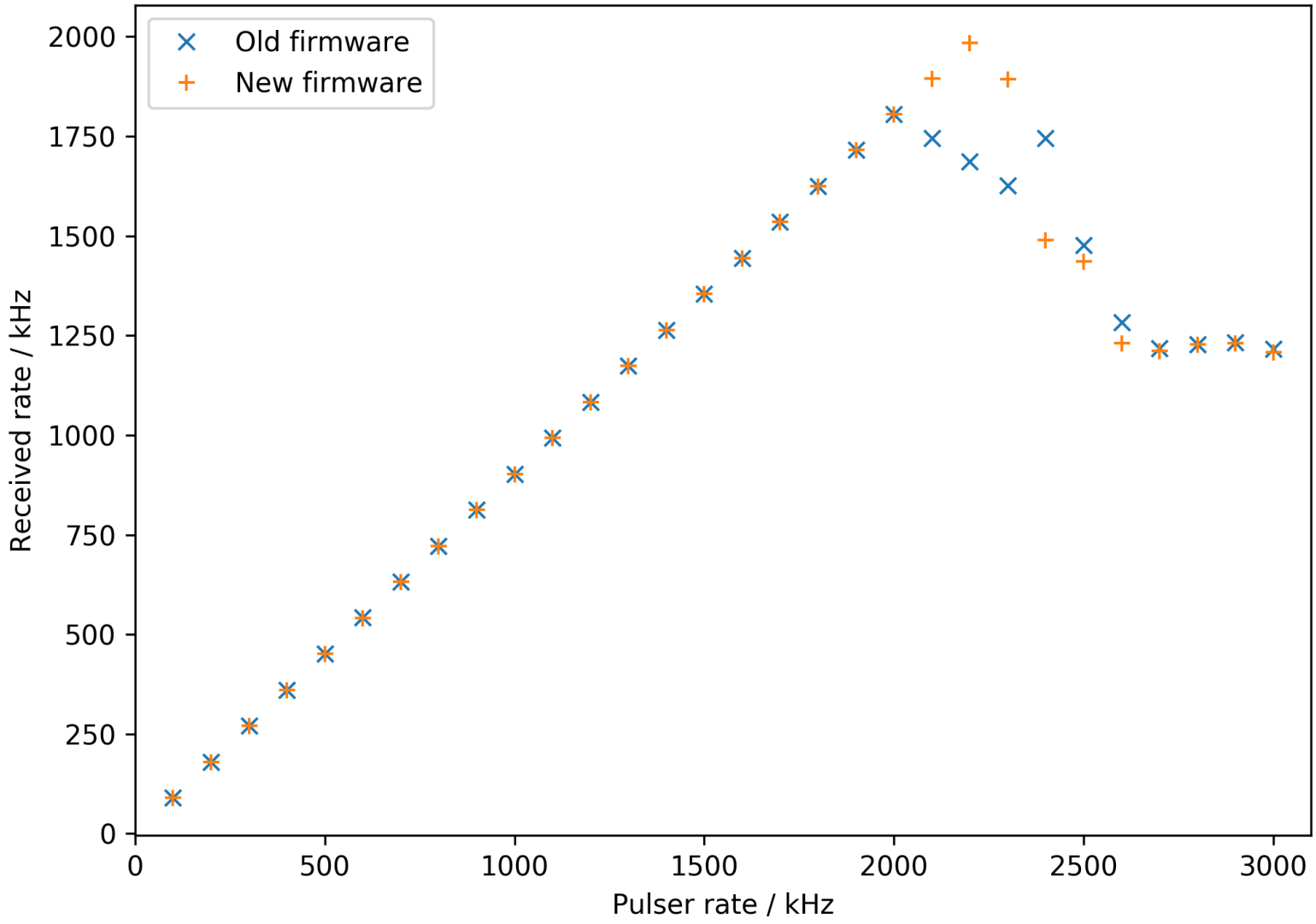
UNIVERSITÄT BONN

NEW FIRMWARE: 180 MHZ



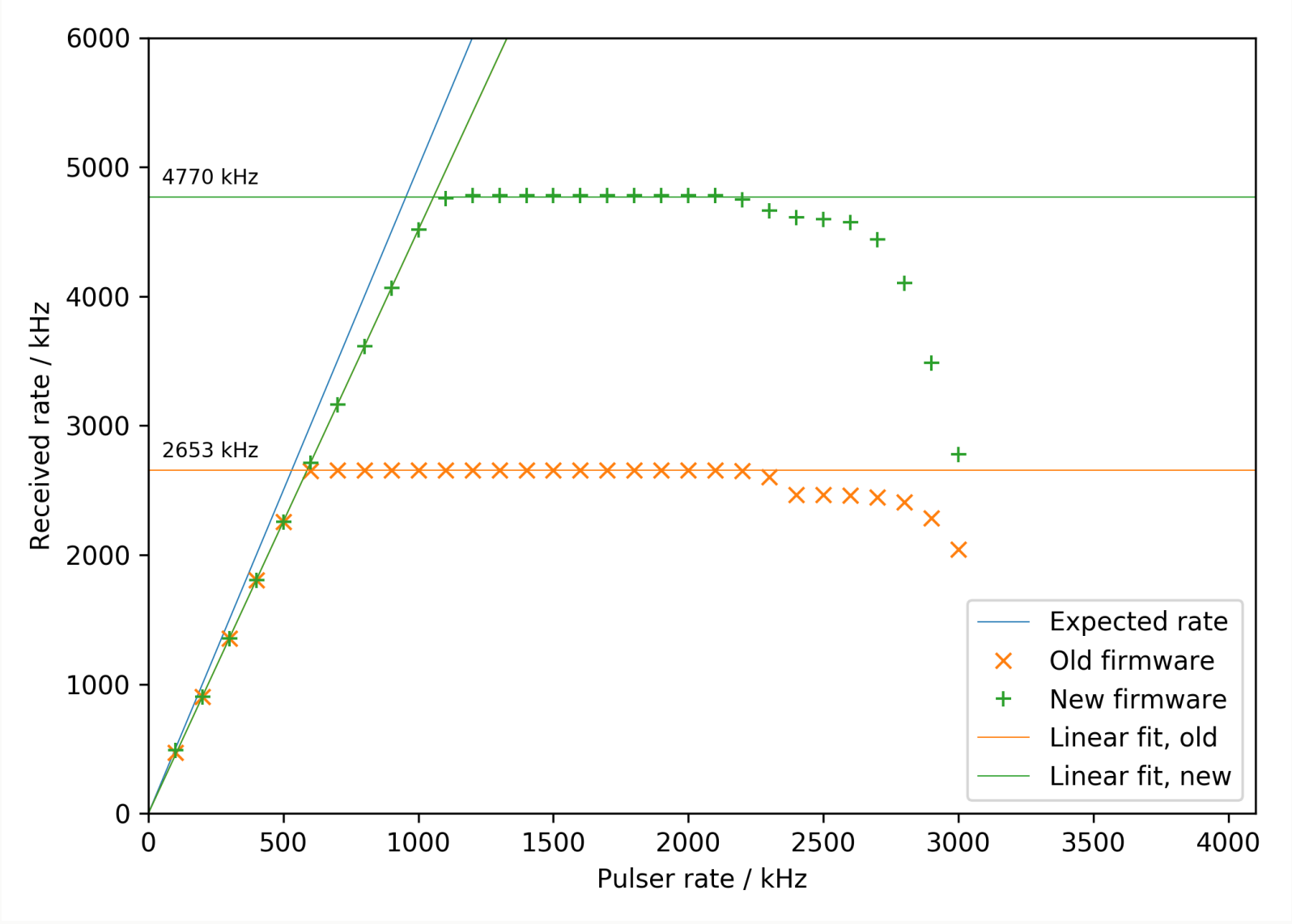
UNIVERSITÄT BONN

PULSING ONE CHANNEL



UNIVERSITÄT BONN

PULSING FIVE CHANNELS



UNIVERSITÄT BONN

REDUCING ADC CONVERSION TIMES

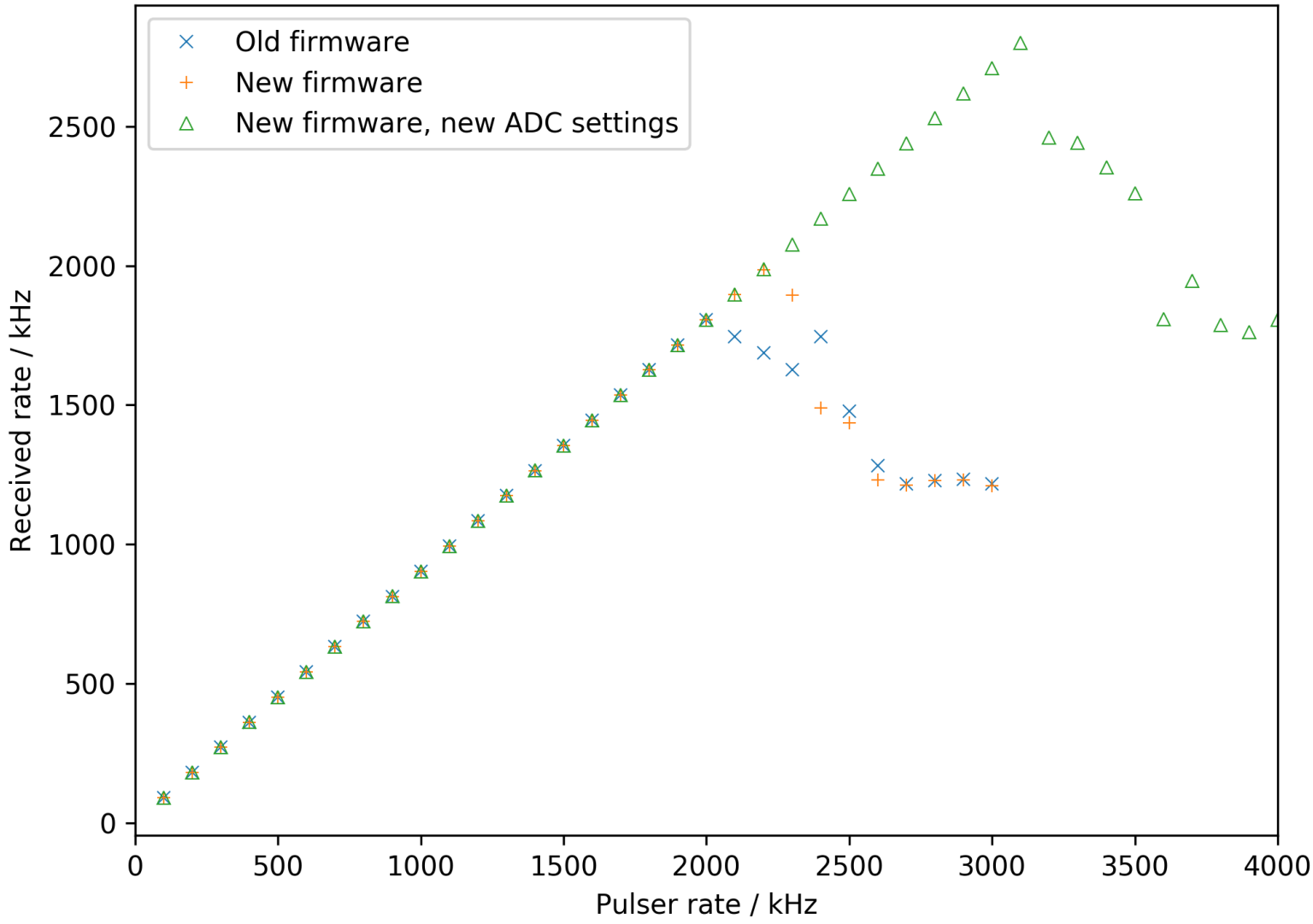
- Multi-channel transfer: Speedup by ~80% corresponds to time reduction from 300 ns to 167 ns seen on oscilloscope/in simulation
- Drop off for single channel between 2 MHz and 2.5 MHz due to incomplete implementation of the ADC conversion time bits in Slow Control:

stc1,stc0 [00 01 10 11]	TAC slope adjustment (60, 100, 350, 650 ns)
sdt9-sdt0 [0:0 through 1:1]	coarse threshold DAC
sdp9-sdp0 [0:0 through 1:1]	test pulse DAC
sc010b,sc110b	10-bit ADC conversion time
sc08b,sc18b	8-bit ADC conversion time
sc06b, sc16b, sc26b	6-bit ADC conversion time
s8b	8-bit ADC conversion mode
s6b	enables 6-bit ADC (requires sttt enabled)
s10b	enables high resolution ADCs (10/8 bit ADC enable)

- New settings implemented (by Michael Lupberger) in Slow Control software
- Large increase in data rates, impact on conversion quality still under investigation

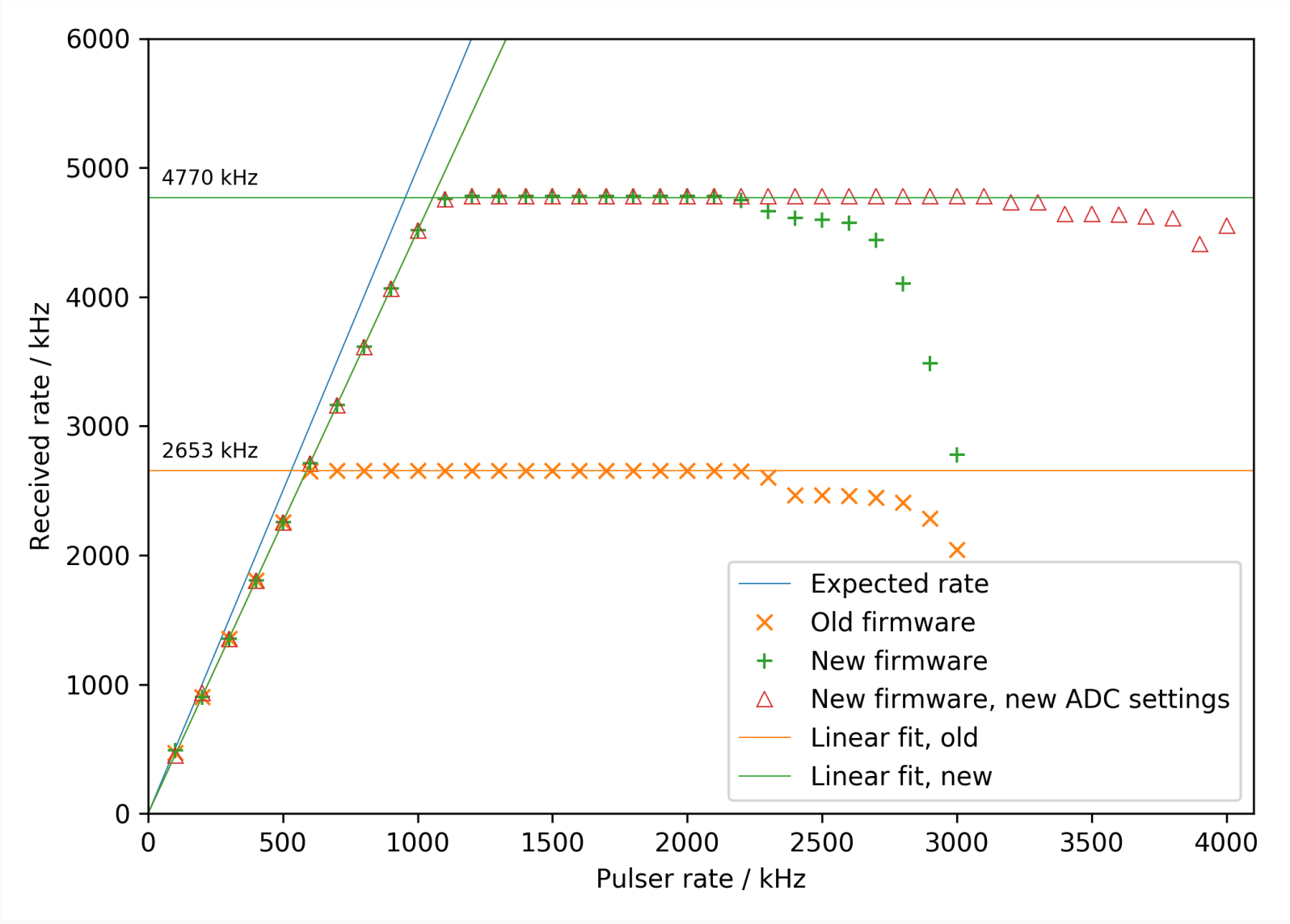
UNIVERSITÄT BONN

PULSING ONE CHANNEL



UNIVERSITÄT BONN

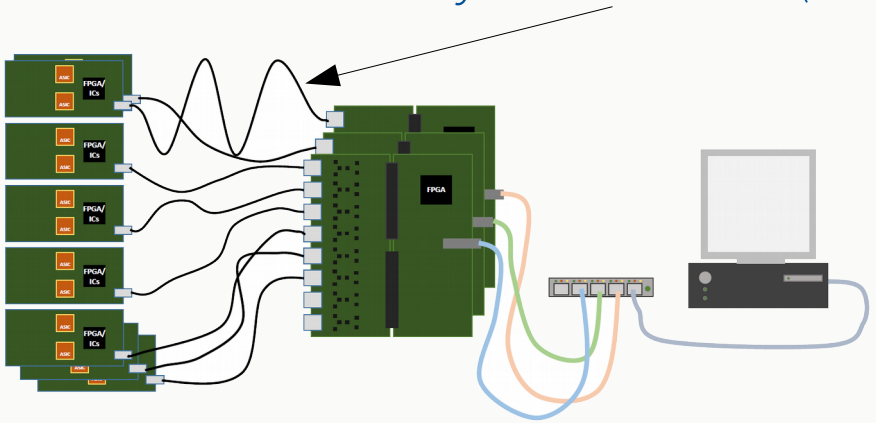
PULSING FIVE CHANNELS



UNIVERSITÄT BONN

ONGOING IMPROVEMENTS

- More efficient transfer of data between hybrid and FEC (HDMI)

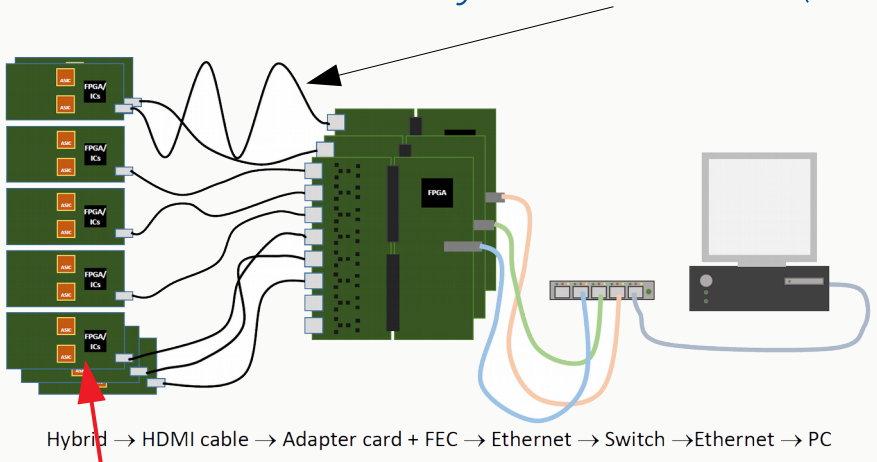


- Previously: 150 ns per hit, now 125 ns per hit
  → Further speedup of transfer between VMM and FPGA possible
  → 125 ns transfers between VMM and FPGA are currently being tested
- Ongoing tests of VMM behaviour in different situations:
  - Varying time between tokens
  - Readout times below 100 ns per hit (buffered on FPGA)
  - Maximum possible data rates and impact of different VMM settings (shaping time, gain)

UNIVERSITÄT BONN

ONGOING IMPROVEMENTS

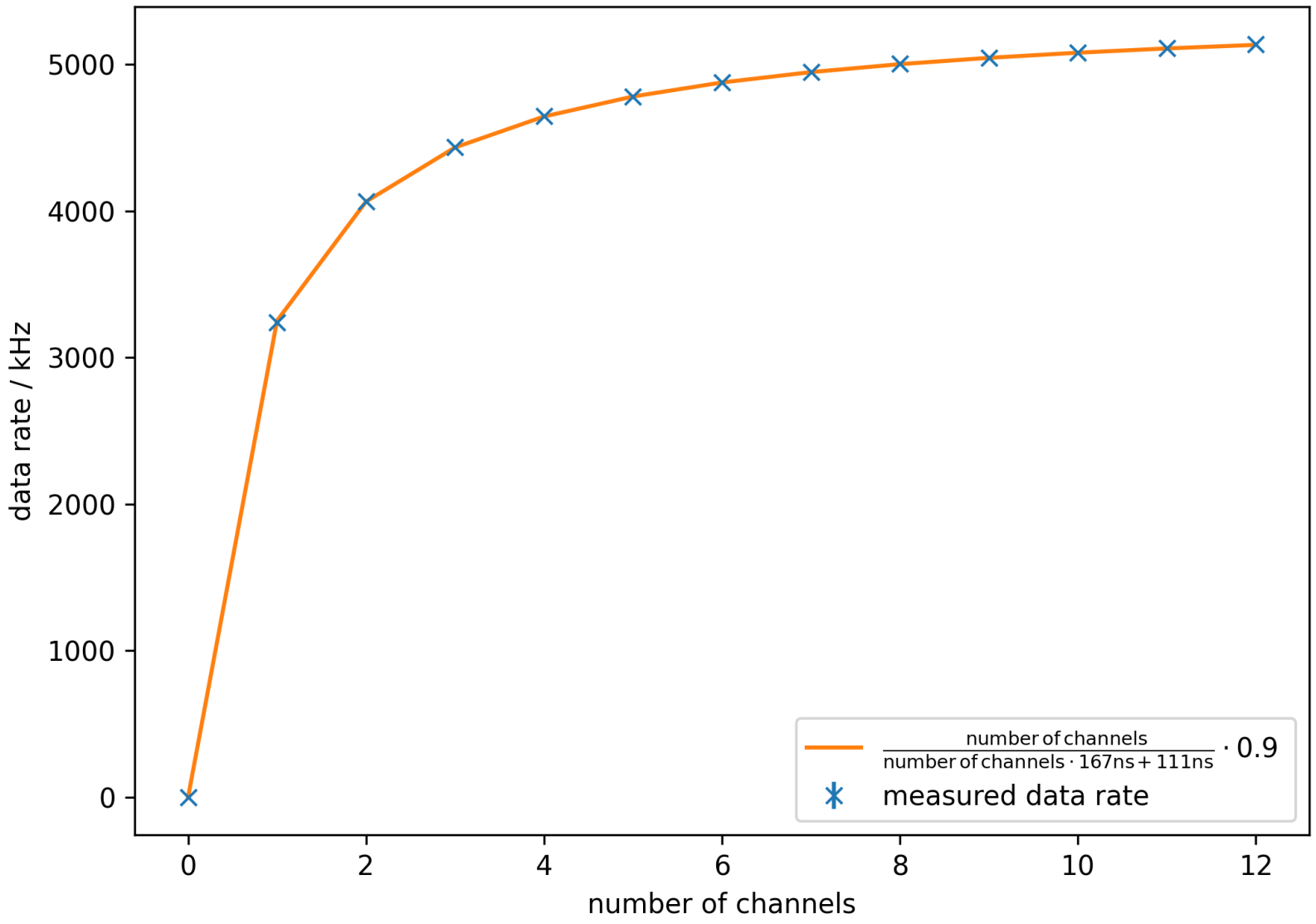
- More efficient transfer of data between hybrid and FEC (HDMI)



- Previously: 150 ns per hit, now 125 ns per hit
  → Further speedup of transfer between VMM and FPGA possible
  → 125 ns transfers between VMM and FPGA are currently being tested
- Ongoing tests of VMM behaviour in different situations:
  - Varying time between tokens
  - Readout times below 100 ns per hit (buffered on FPGA)
  - Maximum possible data rates
  - Impact of different VMM settings (shaping time, gain)

UNIVERSITÄT BONN

THEORETICAL AND PRACTICAL MAXIMUM DATA RATE



UNIVERSITÄT BONN

STORING VMM CONFIGURATION IN BRAM

- Configuration bits for the VMM (about 1.7K each) sent from FEC to FPGA
- FPGA forwards configuration to VMM
- Old firmware: stores configuration in FPGA logic (2 vectors, 1.7Kbit each)
- New firmware: stores configuration in Block RAM

	Old firmware	New firmware
Number of used Slice Registers	4,547 (24%)	1,205 (6%)
Number of used Slice LUTs	9,112 (42%)	1,574 (17%)

UNIVERSITÄT BONN

SUMMARY

- Maximum transfer rate seems to be 180 MHz DDR
- Rewrote FPGA firmware block responsible for receiving data
  - Uses IOB deserializer instead of SDR register
  - Implemented PLL DRP such that the transfer clock can be changed
  - Future changes in firmware might shift data during readout, calibration process implemented in firmware to realign data easily
  - In case the VMM is updated to support 200 MHz, the firmware is already designed for these data rates
- Up to 5.33 MHit/s readout rate (single VMM, 64 active channels), 167 ns data transfer per hit
- Different VMMs show large fluctuations of the maximum single channel rates (2.6 MHz – 3.1 MHz)
- Implemented more efficient data transfer between hybrid and FEC
  - Faster transfers between VMM and FPGA now possible and under investigation

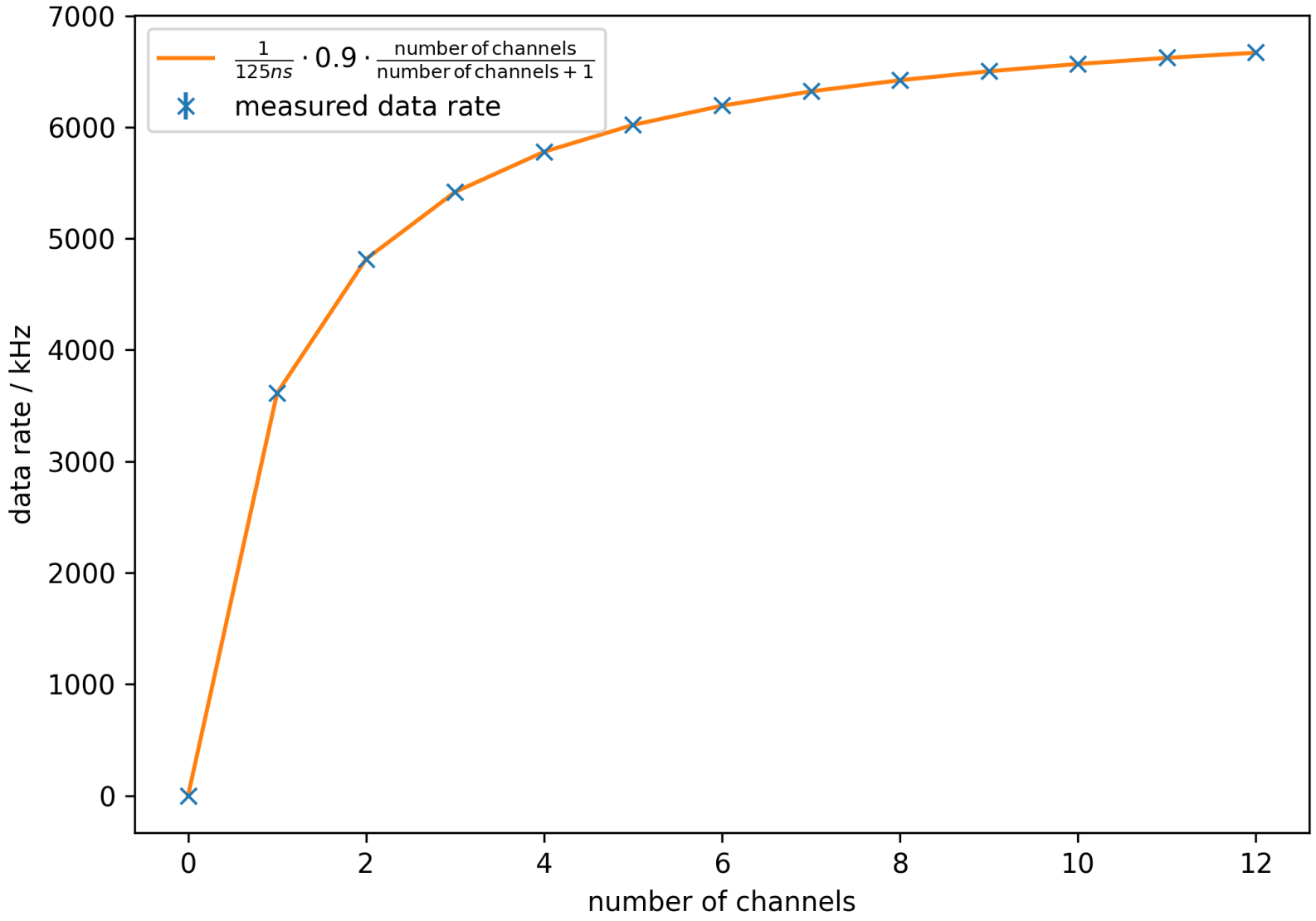
UNIVERSITÄT BONN

Thanks for your attention!

Questions?

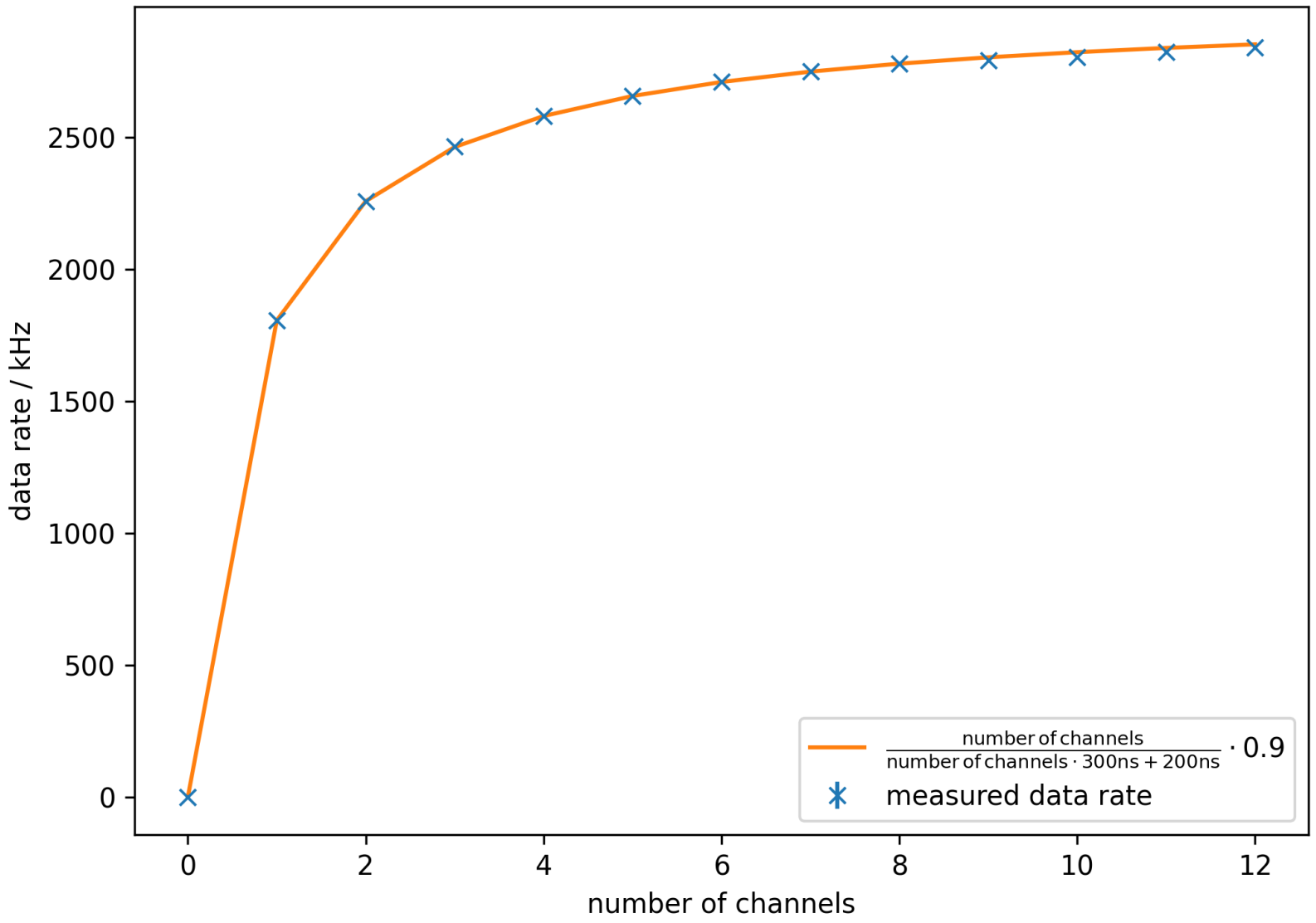
UNIVERSITÄT BONN

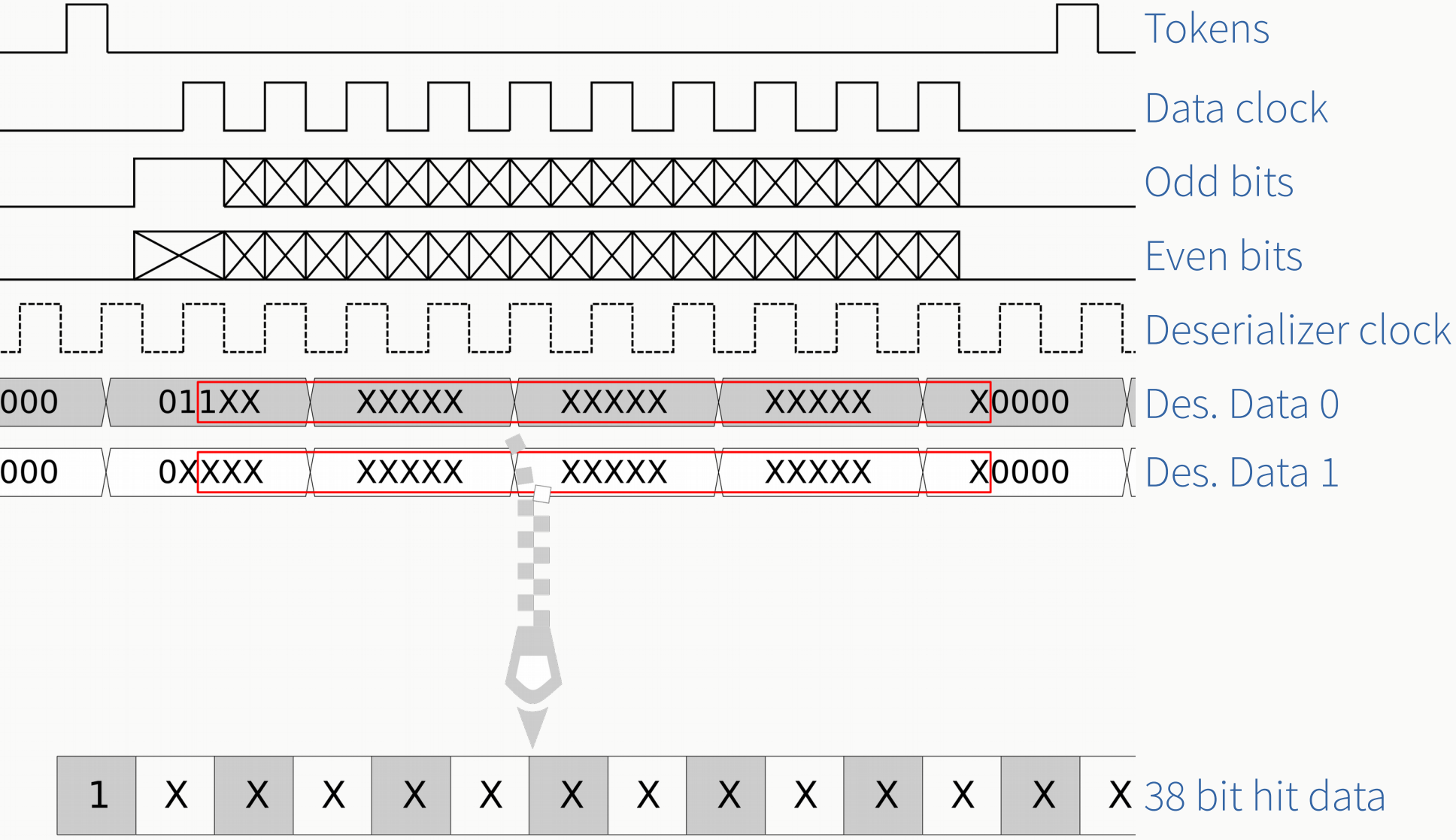
THEORETICAL AND PRACTICAL MAXIMUM DATA RATE



UNIVERSITÄT BONN

THEORETICAL AND PRACTICAL MAXIMUM DATA RATE




Tokens
Data clock
Odd bits
Even bits
Deserializer clock
000
011XX
XXXXX
XXXXX
XXXXX
X0000
Des. Data 0
000
0XXXX
XXXXX
XXXXX
XXXXX
X0000
Des. Data 1
1
X
X
X
X
X
X
X
X
X
X
X
X
X
38 bit hit data

UNIVERSITÄT BONN

PYVMM LIBRARY

- Python 3 library based on PyShark (TShark)
- Receives and decodes data from FEC
- Different example programs e.g. displaying all data from hits,
  visual data rate per channel (using OpenGL) etc.
- Usage:
  - Register self-written callback function
  - Will be called for each received marker and hit
  - Measures hit rates of single or multiple channels using the markers
  - Data taking can be stopped from within callback function using a stop event

UNIVERSITÄT BONN

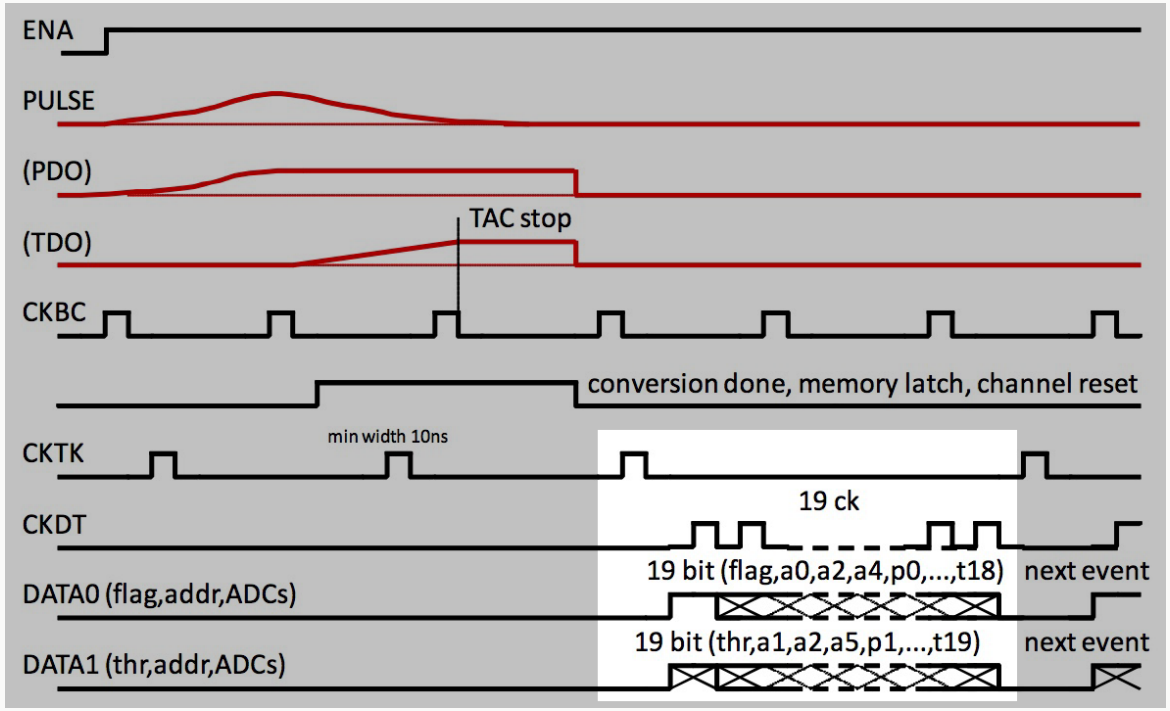
Table 25: **Interface Performances**

Description	I/O Resource	Clock Buffer	Data Width	Speed Grade				Units
				-3	-3N	-2	-1L	
Networking Applications[1]								
SDR LVDS transmitter or receiver	IOB SDR register	BUFG	–	400	400	375	250	Mb/s
DDR LVDS transmitter or receiver	ODDR2/IDDR2 register	2 BUFGs	–	800	800	750	500	Mb/s
SDR LVDS transmitter	OSERDES2	BUFPLL	2	500	500	500	250	Mb/s
			3	750	750	750	375	Mb/s
			4-8	1080	1050	950	500	Mb/s
DDR LVDS transmitter	OSERDES2	2 BUFIO2s	2	500	500	500	250	Mb/s
			3	750	750	750	375	Mb/s
			4-8	1080	1050	950	500	Mb/s
SDR LVDS receiver	ISERDES2 in RETIMED mode	BUFPLL	2	500	500	500	—	Mb/s
			3	750	750	750	—	Mb/s
			4-8	1080	1050	950	—	Mb/s

Table 25: **Interface Performances**

Description	I/O Resource	Clock Buffer	Data Width	Speed Grade				Units
				-3	-3N	-2	-1L	
Networking Applications[1]								
SDR LVDS transmitter or receiver	IOB SDR register	BUFG	–	400	400	375	250	Mb/s
DDR LVDS transmitter or receiver	ODDR2/IDDR2 register	2 BUFGs	–	800	800	750	500	Mb/s
SDR LVDS transmitter	OSERDES2	BUFPLL	2	500	500	500	250	Mb/s
			3	750	750	750	375	Mb/s
			4-8	1080	1050	950	500	Mb/s
DDR LVDS transmitter	OSERDES2	2 BUFIO2s	2	500	500	500	250	Mb/s
			3	750	750	750	375	Mb/s
			4-8	1080	1050	950	500	Mb/s
SDR LVDS receiver	ISERDES2 in RETIMED mode	BUFPLL	2	500	500	500	—	Mb/s
			3	750	750	750	—	Mb/s
			4-8	1080	1050	950	—	Mb/s

DATA TRANSFER BETWEEN VMM AND FPGA (CONTINUOUS MODE)



(from Spartan-6)

(from Spartan-6)

(from VMM, even data bits)

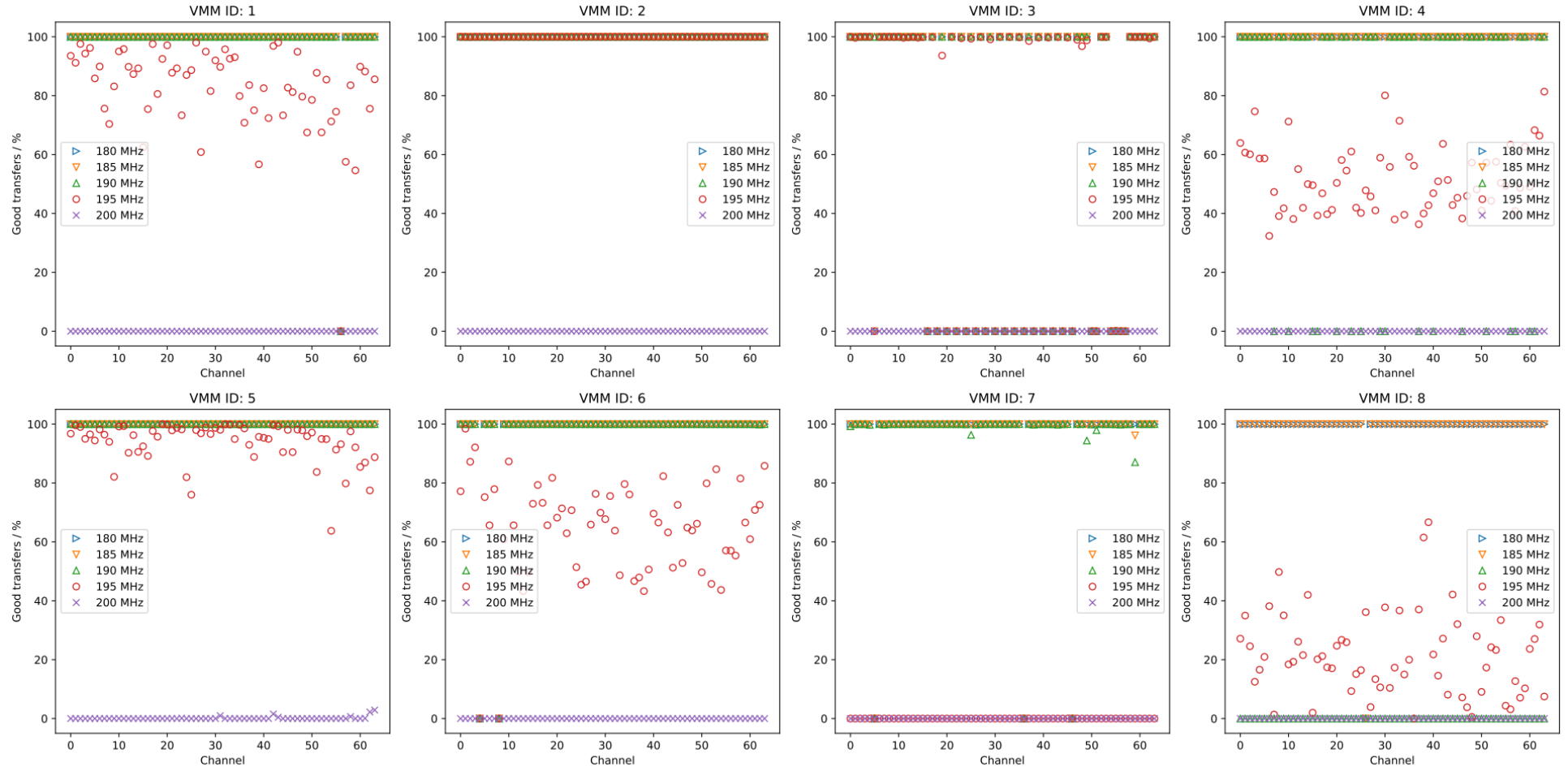
(from VMM, odd data bits)

Data transfer:

- FPGA sends tokens on **CKTK**
- If VMM has data, raises flag after token on **DATA0**
- FPGA starts data clock on **CKDT**
- VMM sends data on **DATA0** and **DATA1**, 19 bits each

UNIVERSITÄT BONN

MAXIMUM RATE FOR ALL VMMS



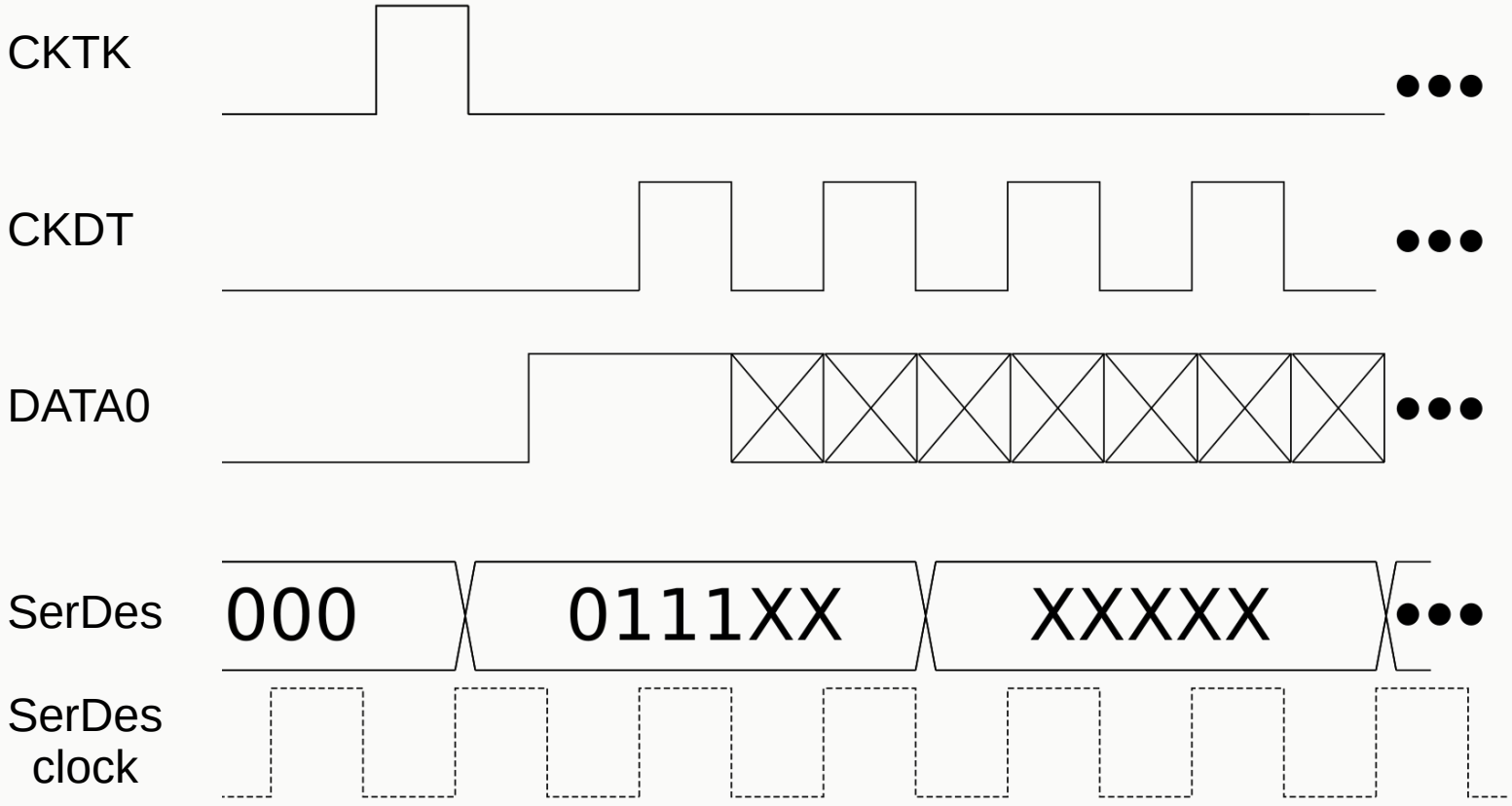
UNIVERSITÄT BONN

MEASURING THE PERFORMANCE IN PRACTICE

- Pulsing externally
- Adapter from Ketzer group
- Function generator:
  internal arbitrary function generator (AFG) of MSO56 oscilloscope
- Procedure:
  - Python script controls AFG via USB using the usbtmc library
    (script runs on Debian in VirtualBox on Windows Host, a miracle it's working)
  - Goes through 100 – 4000 kHz frequency range in 100 kHz steps
  - Measures received rate using my PyVMM library

UNIVERSITÄT BONN

AUTO-CALIBRATION

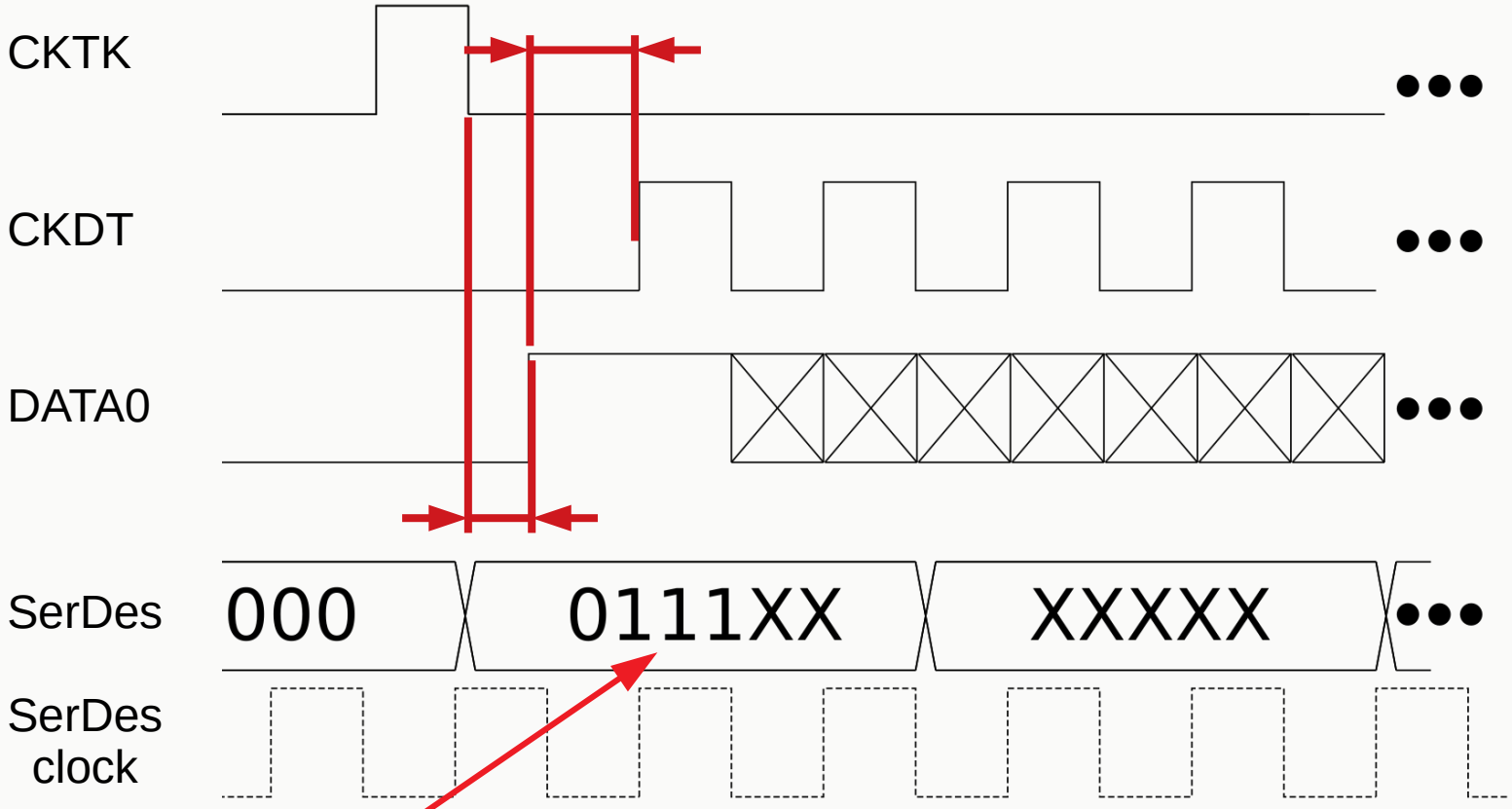


Multiple leading ‘1’, shifted depending on phase between SerDes and data

- affected by delay between token and flag
- time between recognizing flag and start of CKDT

UNIVERSITÄT BONN

AUTO-CALIBRATION



Multiple leading ‘1’, shifted depending on phase between SerDes and data

- affected by delay between token and flag
- time between recognizing flag and start of CKDT

AUTO-CALIBRATION (WORK IN PROGRESS)

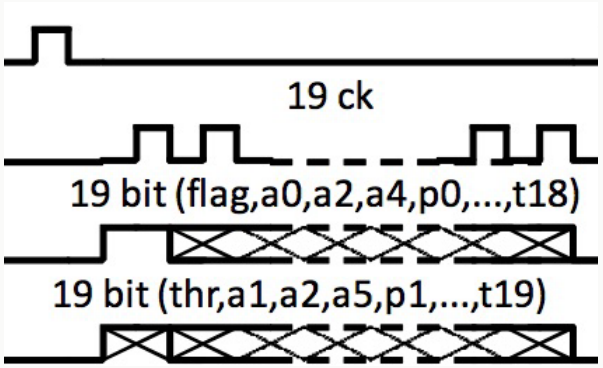
Where does the data really start?

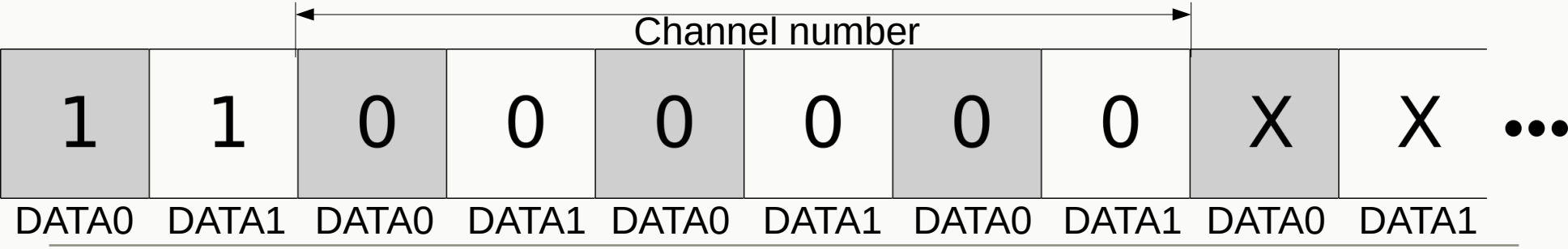
Old firmware:

- counters with different frequencies
- for each CKDT frequency setting a set of conditions
  which counter must have which value

New firmware:

- send test pulse on a single channel, e.g. channel 0
- where does the first '0' occur?
- infer data shift from this information
- apply this shift to all incoming data





AUTO-CALIBRATION (WORK IN PROGRESS)

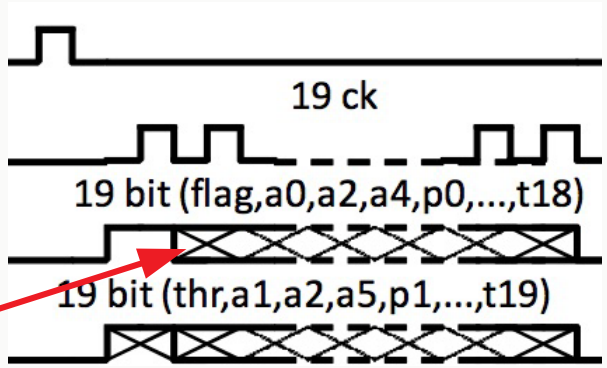
Where does the data really start?

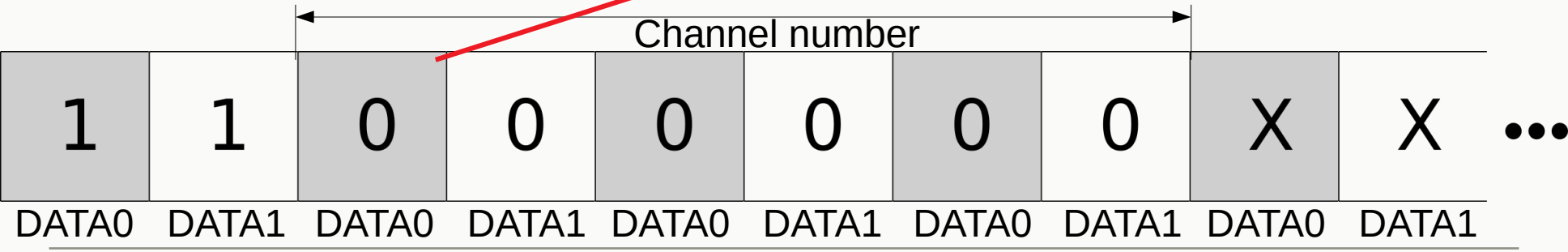
Old firmware:

- counters with different frequencies
- for each CKDT frequency setting a set of conditions
  which counter must have which value

New firmware:

- send test pulse on a single channel, e.g. channel 0
- where does the first '0' occur?
- infer data shift from this information
- apply this shift to all incoming data





UNIVERSITÄT BONN

200 MHZ TRANSFER