1.28 Gbps firmware

(or: Maria tries VHDL Pt 2)

Maria Mironova, Timon Heim

Weekly student instrumentation meeting

28/04/2023

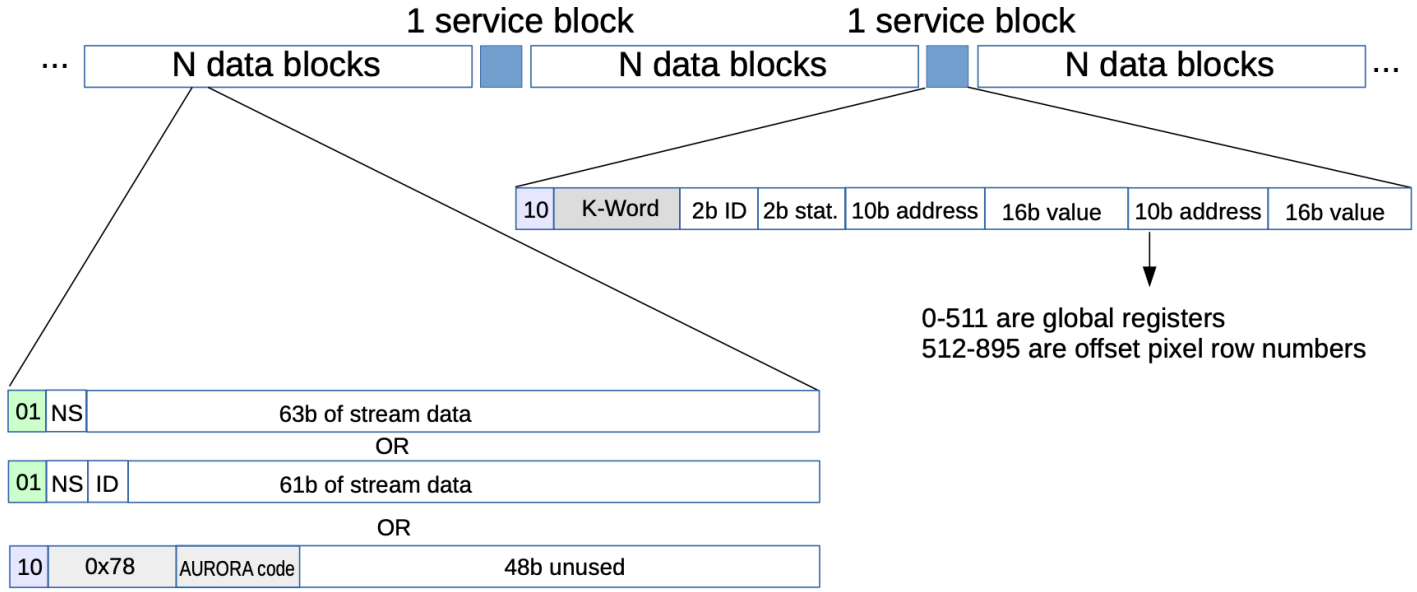
BERKELEY LAB

ATLAS EXPERIMENT

Motivation

- ITk requires readout link speed of 1.28 Gbits/s to cope with the high data rates, especially in the innermost regions
- We need to be able to test data transmission for QC
- Currently most of the testing in YARR is done at 640 Mbps, and readout at 1.28 Gbps was not possible (reliably)
- Goals:
  - Make 1.28 Gbps firmware work and understand better the behaviour of the data transmission
  - Make a basic scan for testing data transmission for module QC
  - Implement some mechanism to automatically updated the spec configuration to the optimal settings

Introduction



- RD53B uses Aurora64b66b protocol for data transmission
- Each Aurora block consists of 66 bits, with a 2-bit sync header (01 for data, 10 for service), plus 64 bits of scrambled data (scrambling means that the data stream is balanced, i.e. consists of an equal number of 0 and 1)
- Sync header block allows for frame alignment, i.e. identifying where each 66-bit block starts

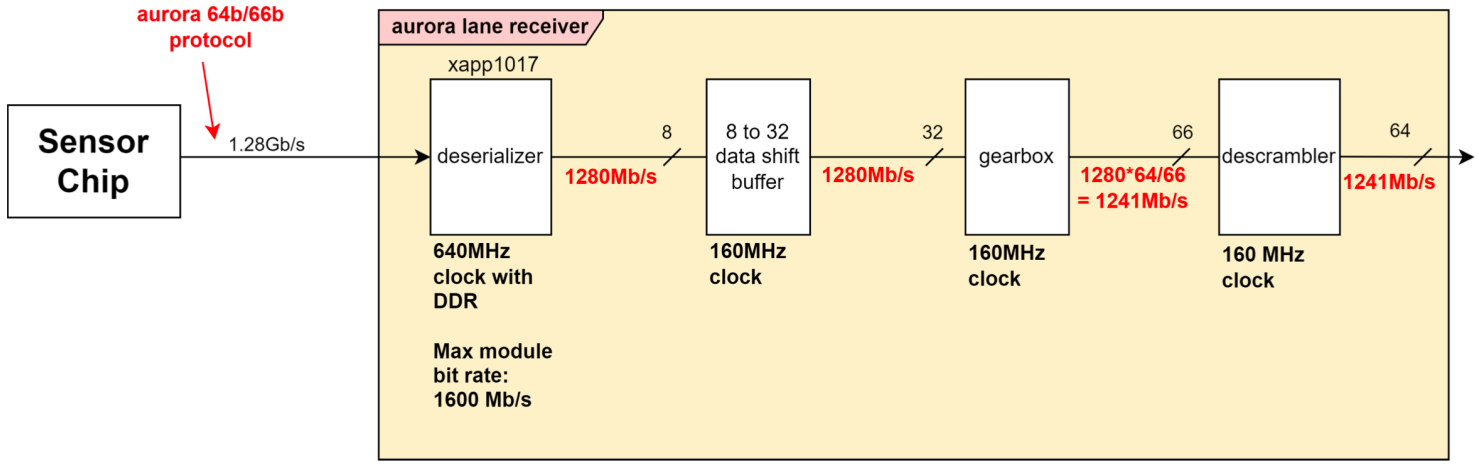
Data processing in YARR

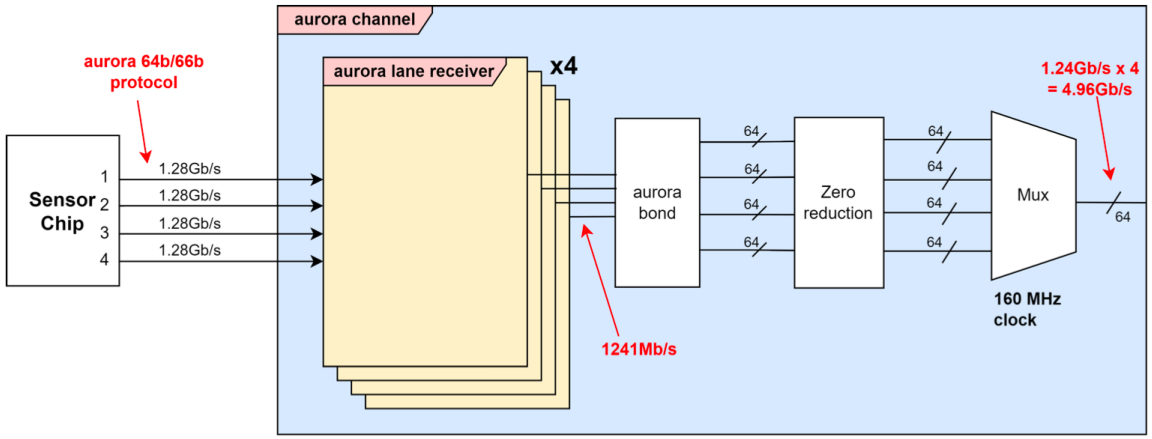
- Data transmission works via Aurora lanes:

1. Aurora lane receives aurora data blocks, which go into a deserializer
   - → Makes single bit signal into 8-bit signal
   - → **Previously deserializer was specific Xilinx application, which was trying to figure out sampling delay in a smart way**
   - → **With the new firmware, sampling delay is configurable from software**
2. Data goes into a 8 to 32 data shift buffer
3. 32 data goes into an internal buffer, which the gearbox process shifts through (in steps of 2) until it finds the sync blocks
4. Descrambler then extracts the 64-bits of data

- Several Aurora lanes can make up one Aurora channel, depending on what configuration we want, e.g.:
  - 4x4: 4 channels with 4 lanes
  - 16x1: 16 channels with 1 lane

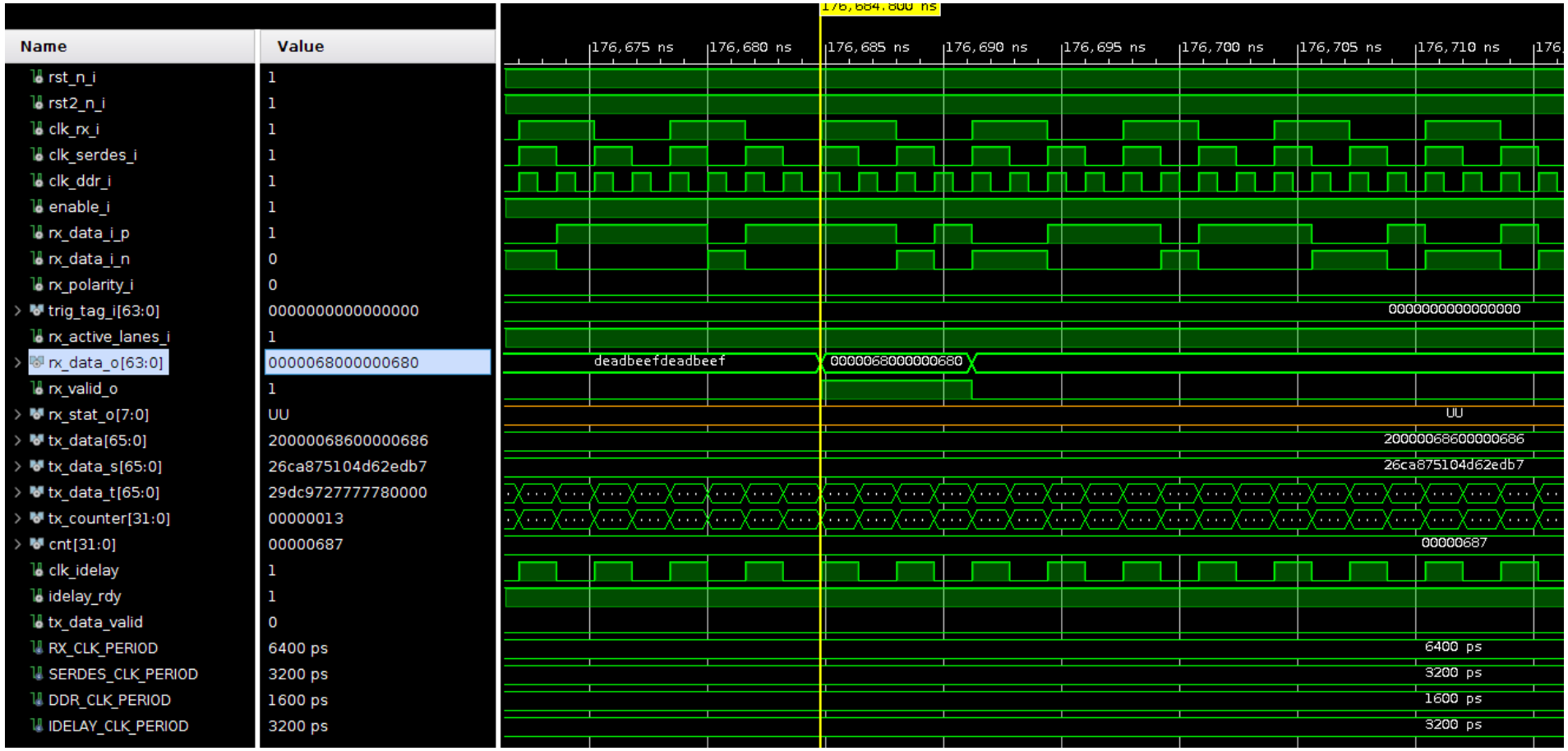
Plots from Loic's thesis





Firmware simulation

- Instead of testing firmware directly on an FPGA, it is possible to run simulation
- Simulation shows all of the signals etc inside of the FPGA, in this e.g. in this case for one RX channel
- So, as a first step I spent some time making the YARR simulation work to use it for testing the new deserialiser



New deserialiser

- For the new deserialiser, we want to be able to set the delay setting from software, to be able to vary it, make scans and find the optimal settings
- In practice this means implementing a new component in the FW, which basically just a delay + deserialiser
- Tested hardcoding various delay settings in the simulation, to validate the implementation works

Delay (configurable from software)

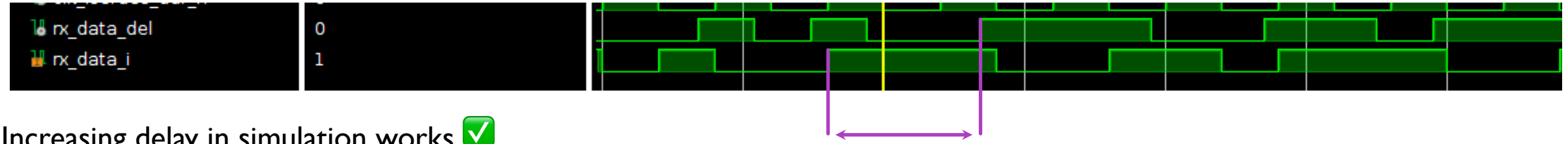
Input data

Deserialised output data

```
serdes_idelay_rdy <= rst_n_i;
serdes_cmp: serdes_1_to_8_idelay_ddr
port map (
    rst_n_i => rst_n_i,
    sys_clk_i => sys_clk_i,
    delay_value_i => rx_delay_i,
    clk_iserdes_ddr_i => clk_serdes_i,
    rx_data_i => rx_data_i_p,
        clk_rx_i => clk_rx_i,
    --datain_n(0) => rx_data_i_n,
    --bitslip => '0',
    rx_data_o(7) => serdes_data8(7),
    rx_data_o(6) => serdes_data8(6),
    rx_data_o(5) => serdes_data8(5),
    rx_data_o(4) => serdes_data8(4),
    rx_data_o(3) => serdes_data8(3),
    rx_data_o(2) => serdes_data8(2),
    rx_data_o(1) => serdes_data8(1),
    rx_data_o(0) => serdes_data8(0)
    --debug => xapp_debug
);
```

Delay 16

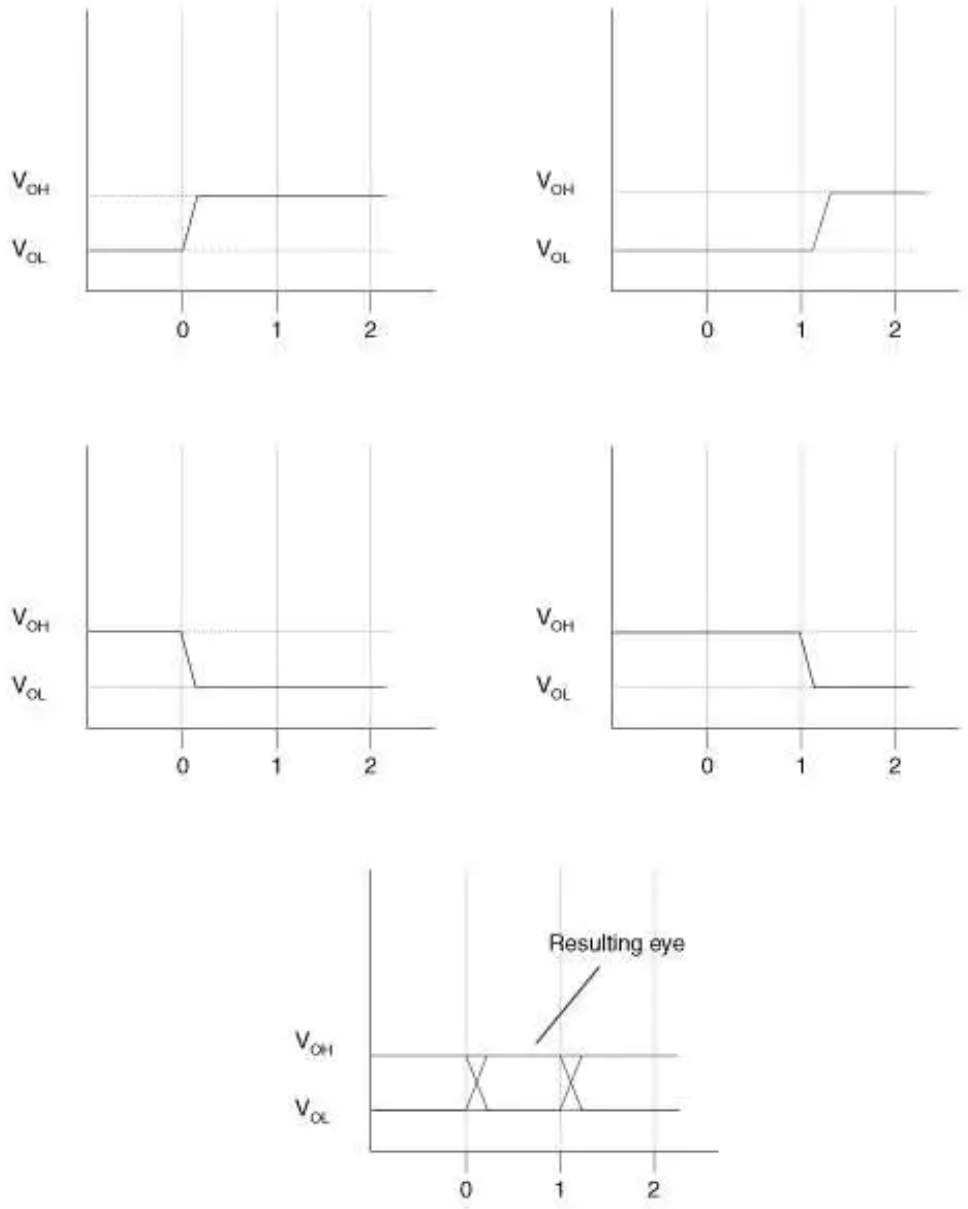
Delay 31

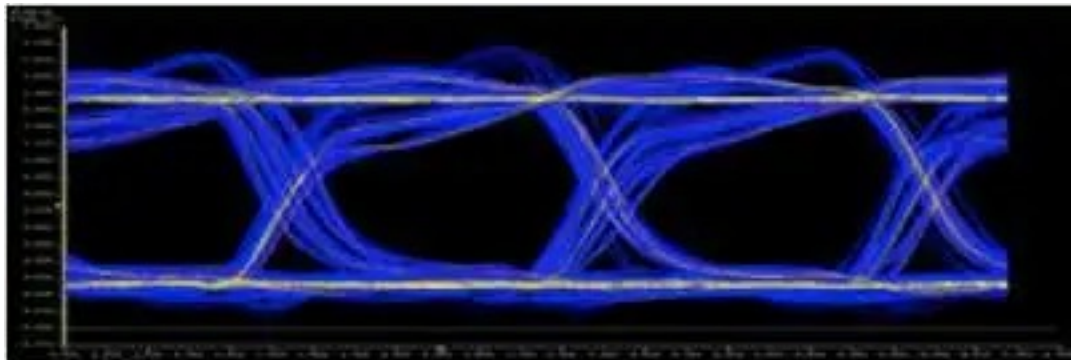


→ Increasing delay in simulation works ✅

Aside: Eye diagrams

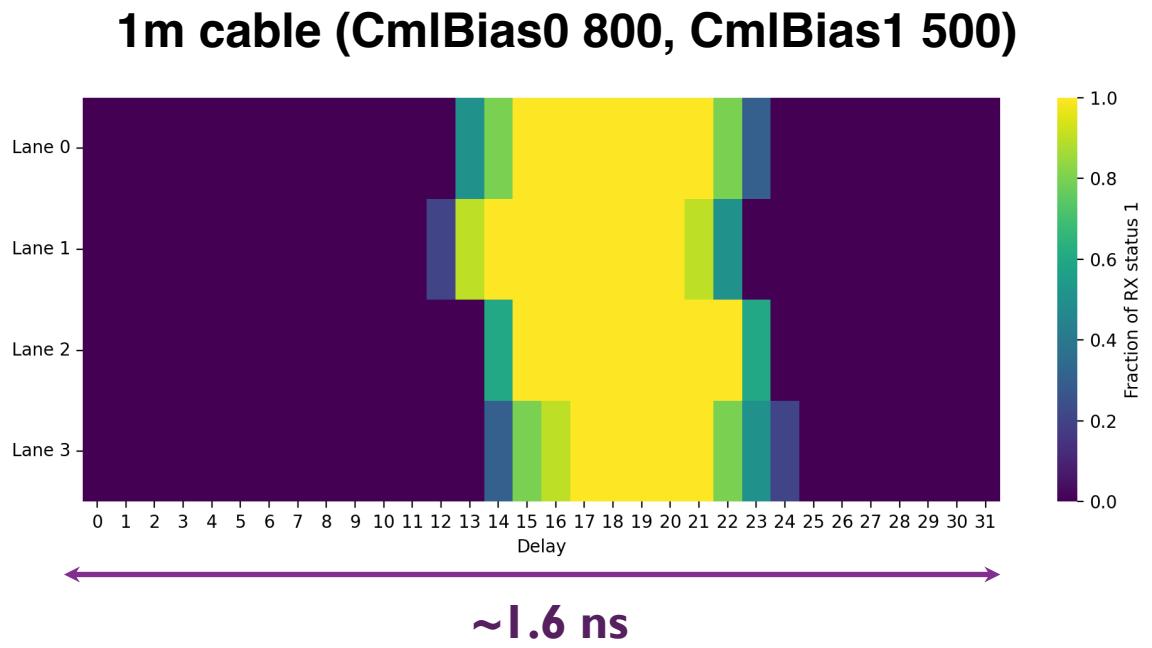
- Eye diagrams are generally used for testing data transmission and signal integrity
- Generally measured by overlaying several digital signals
- In an ideal world the transitions would be rectangular, but in the real word you have a slope and form an eye-shape
- Additionally, measuring several times allows to see some jitter
- For good data transmission you want to be somewhere in the flat section/middle of the eye

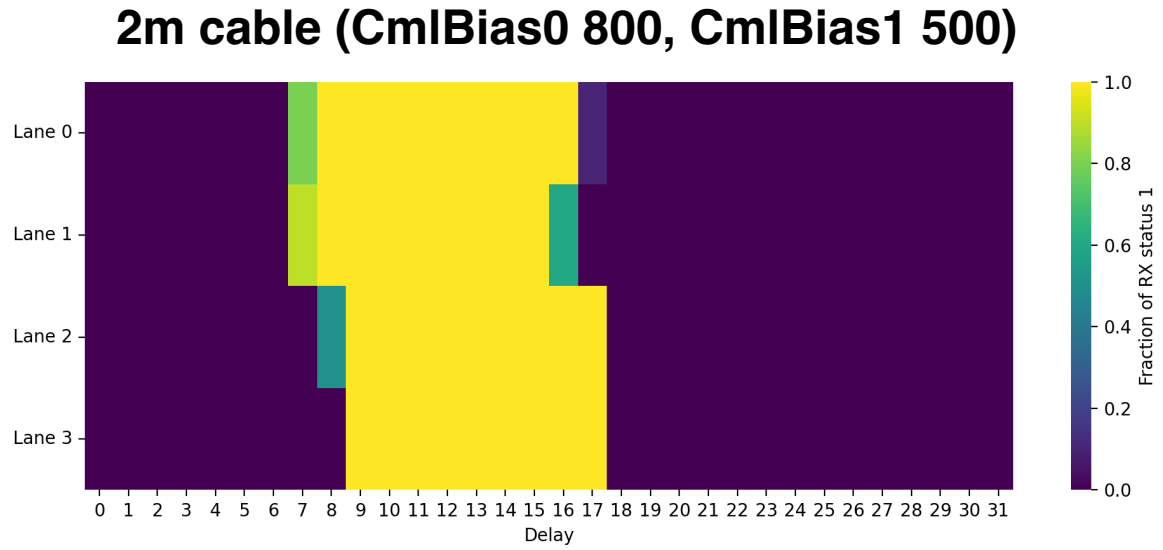




Eye diagram scans

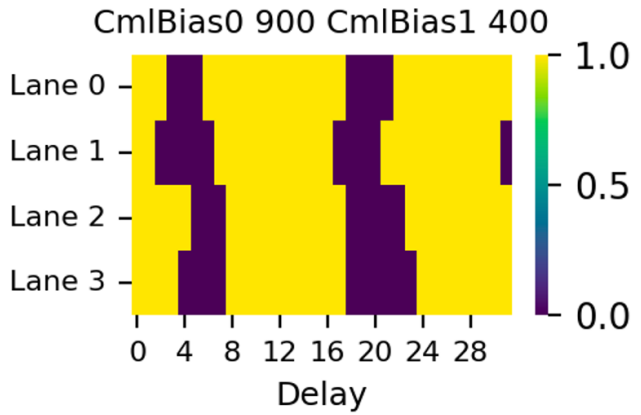
- By varying the deserialiser delay value we can make a (kind of) eye diagram
- To do so:
  - Loop over all delay values
  - Write delay value from software
  - Check RX sync status
  - Repeat 10 times
- Can see eye opening and jitter
- Also different behaviour for different cables (would generally expect shorter cables to be better, but may be just a bad cable)
- Problem: over the course of 1.6 ns we'd expect more than one eye and the "not good" regions seem too long





Gearbox bug fix

- Found an issue in the gearbox implementation, which caused the data stream to only lock onto every second eye
- Long story short, gearbox used to slip the data in steps of 2 bits, which made it possible to miss the "01" and "10" sync headers
- Fixed by changing gearbox slipping in steps of 1 and increasing the buffer size → with this we're now able to see two eyes in the eye diagrams → nice! ✅

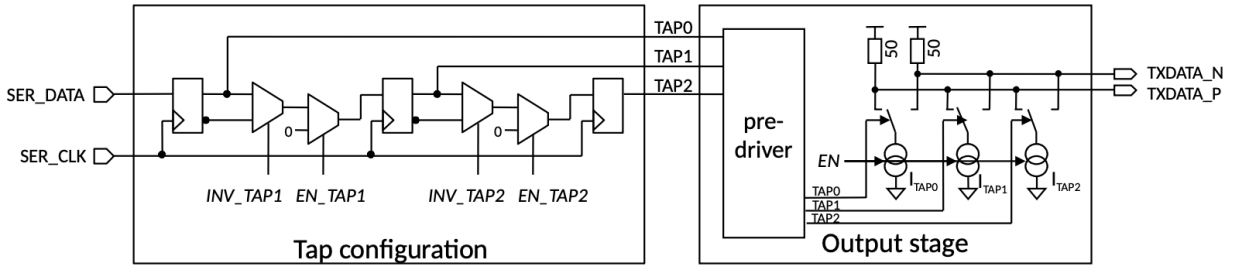



```
67          data66_buf2_read <= '0';
68          if (data32_valid_i = '1') then
69              shift_cnt <= not shift_cnt;
70  -           buffer128(127 downto 0) <= buffer128(95 downto 0) &
    data32_i;
71  -           data66_t <= buffer128(128-(to_integer(gearbox_cnt(4 downto
    0))*2)-1 downto 62-(to_integer(gearbox_cnt(4 downto 0))*2));
72              if (shift_cnt = '1') then
73                  if (slip_i = '1') then
74  -                   gearbox_cnt <= gearbox_cnt;
75                      data66_t_valid <= '1';
76  -               elsif (gearbox_cnt = 32) then
77                      gearbox_cnt <= (others => '0');
78                      data66_t_valid <= '0';
79                  else
80  -                   gearbox_cnt <= gearbox_cnt + 1;
81                      data66_t_valid <= '1';
82              end if;
83          else
```

```
67          data66_buf2_read <= '0';
68          if (data32_valid_i = '1') then
69              shift_cnt <= not shift_cnt;
70  +           buffer160(159 downto 0) <= buffer160(127 downto 0) &
    data32_i;
71  +           data66_t <= buffer160(160-(to_integer(gearbox_cnt(5 downto
    0)))-1 downto 94-(to_integer(gearbox_cnt(5 downto 0))));
72              if (shift_cnt = '1') then
73                  if (slip_i = '1') then
74  +                   gearbox_cnt <= gearbox_cnt + 1;
75                      data66_t_valid <= '1';
76  +               elsif (gearbox_cnt = 64) then
77                      gearbox_cnt <= (others => '0');
78                      data66_t_valid <= '0';
79  +               elsif (gearbox_cnt = 65) then
80  +                   gearbox_cnt <= x"01";
81  +                   data66_t_valid <= '0';
82                  else
83  +                   gearbox_cnt <= gearbox_cnt + 2;
84                      data66_t_valid <= '1';
85              end if;
86          else
```

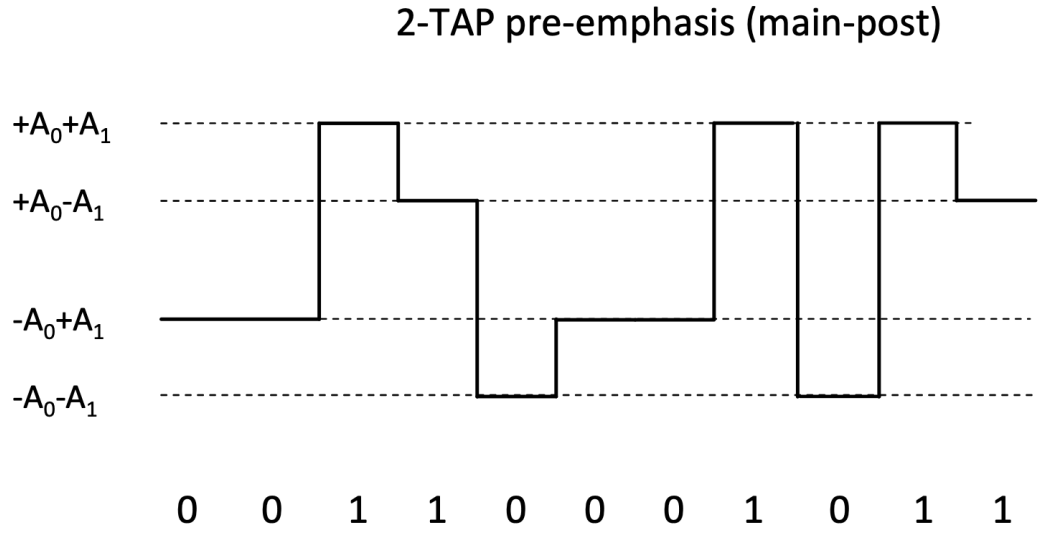
Cml bias scan

- With firmware now available for manually adjusting delays it is possible to characterise data transmission
- In addition to just measuring the eye diagrams, ITkPix has parameters for signal shaping → e.g. CmlBias1 for adding pre-emphasis
- Good settings should decrease the time it takes to transition in the eye diagram and increase eye opening width
- Scan over CmlBias0 and CmlBias1 to understand what good settings would be
- Start by testing this with 4x4 firmware on a single chip card (i.e. with changing delays individually for each lane)



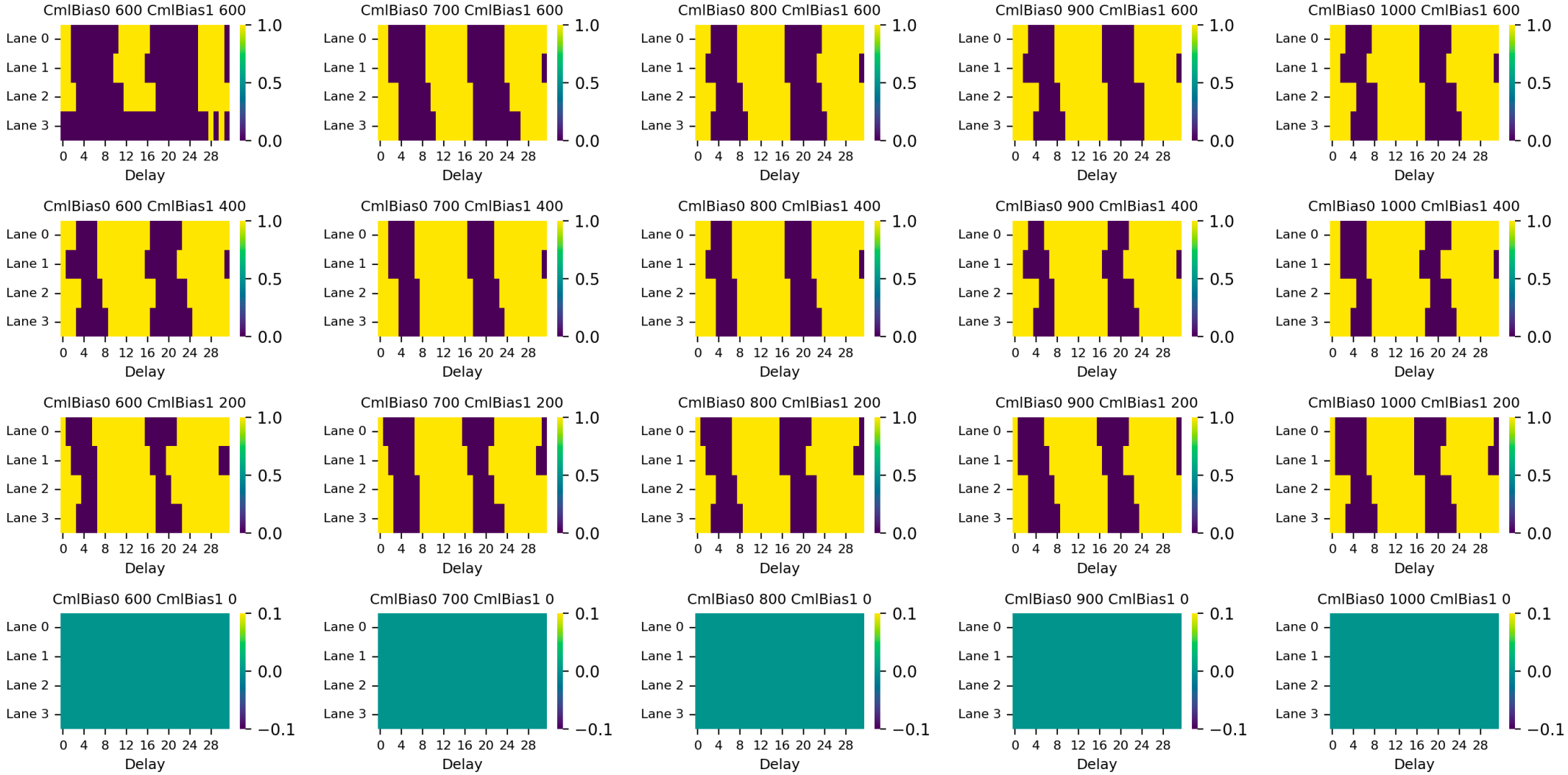
TAP configuration
- *INV_TAP[2:1]*
- *EN_TAP[2:1]*

CML output configuration
- *EN*
- *TAP0_BIAS[9:0]*
- *TAP1_BIAS[9:0]*
- *TAP2_BIAS[9:0]*



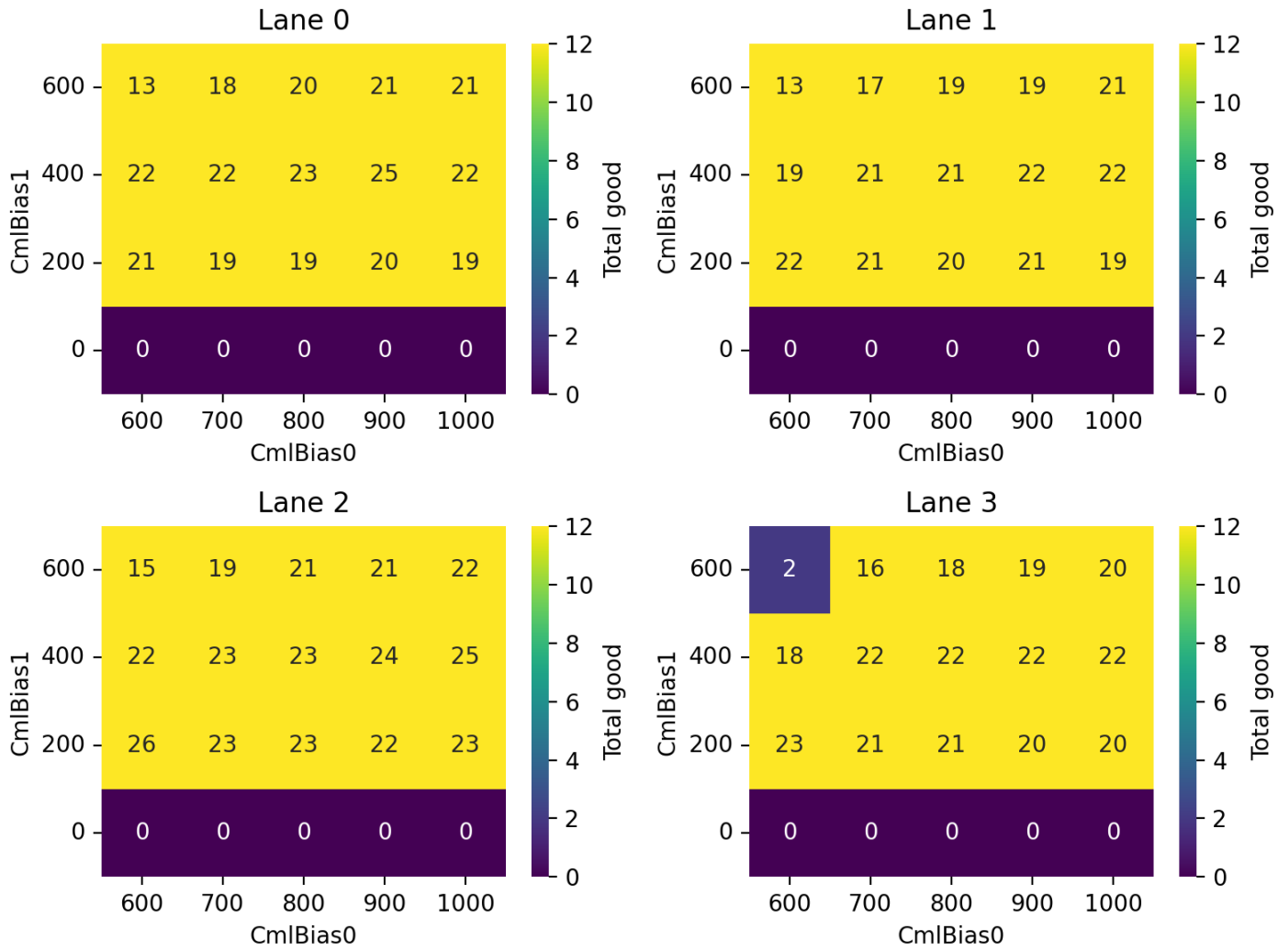
Cml bias scan on SCC

→ All lanes look very similar, expected for a single chip



Cml bias scan on SCC

- Can also plot ”total good” delay values as a function of CmlBias (i.e. number of points where it was possible to establish communication)
- Definitely some pre-emphasis (CmlBias1), otherwise communication does not work at all
- Pretty consistent behaviour between lanes, as expected for a single chip
- CmlBias0 800, CmlBias1 400 seems like a good setting for an SCC

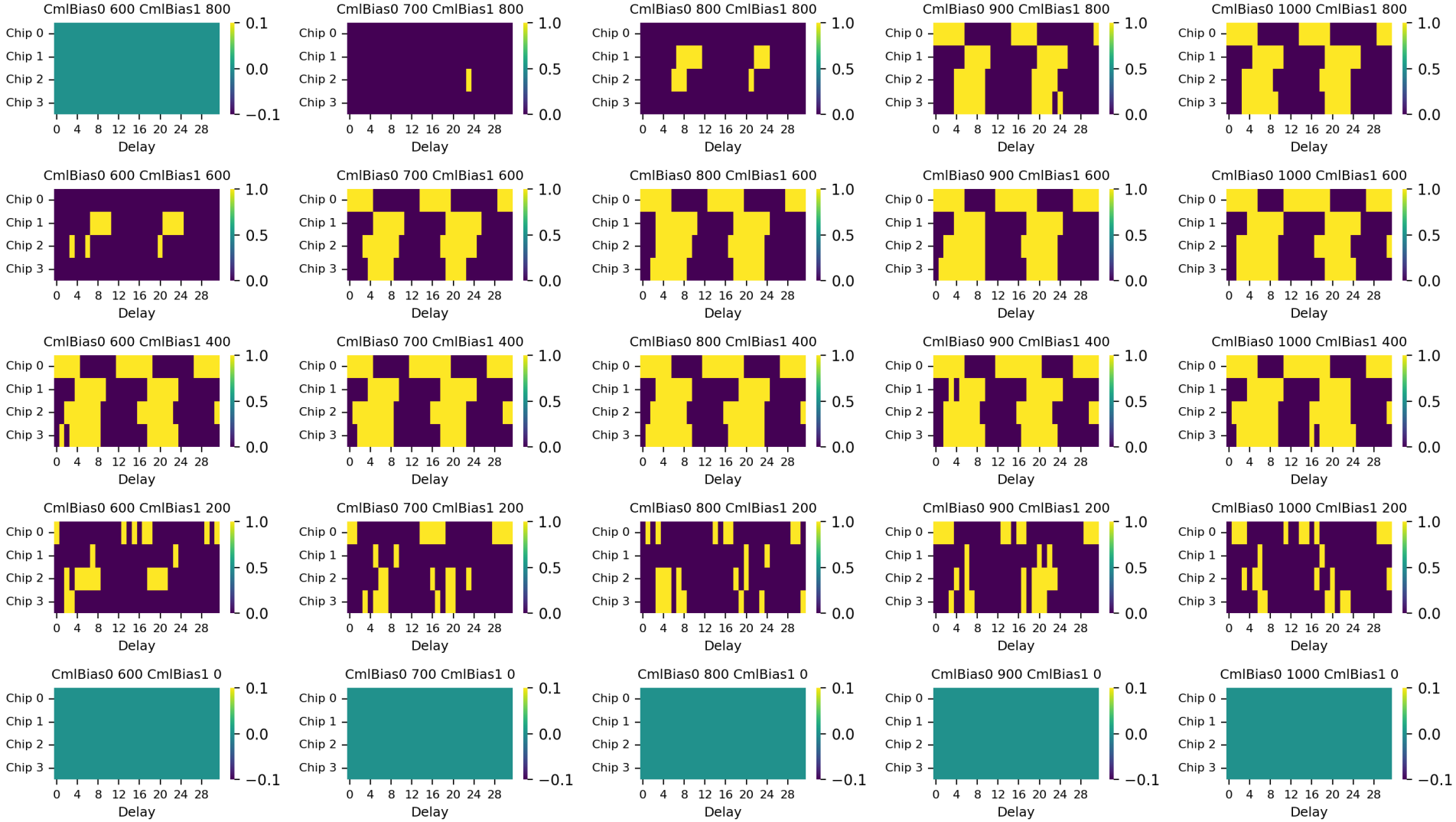


Quad cml bias scan

- Next, go to a quad for testing
- On a quad, we read each chip with one lane

→ Now scanning over different chips rather than lanes

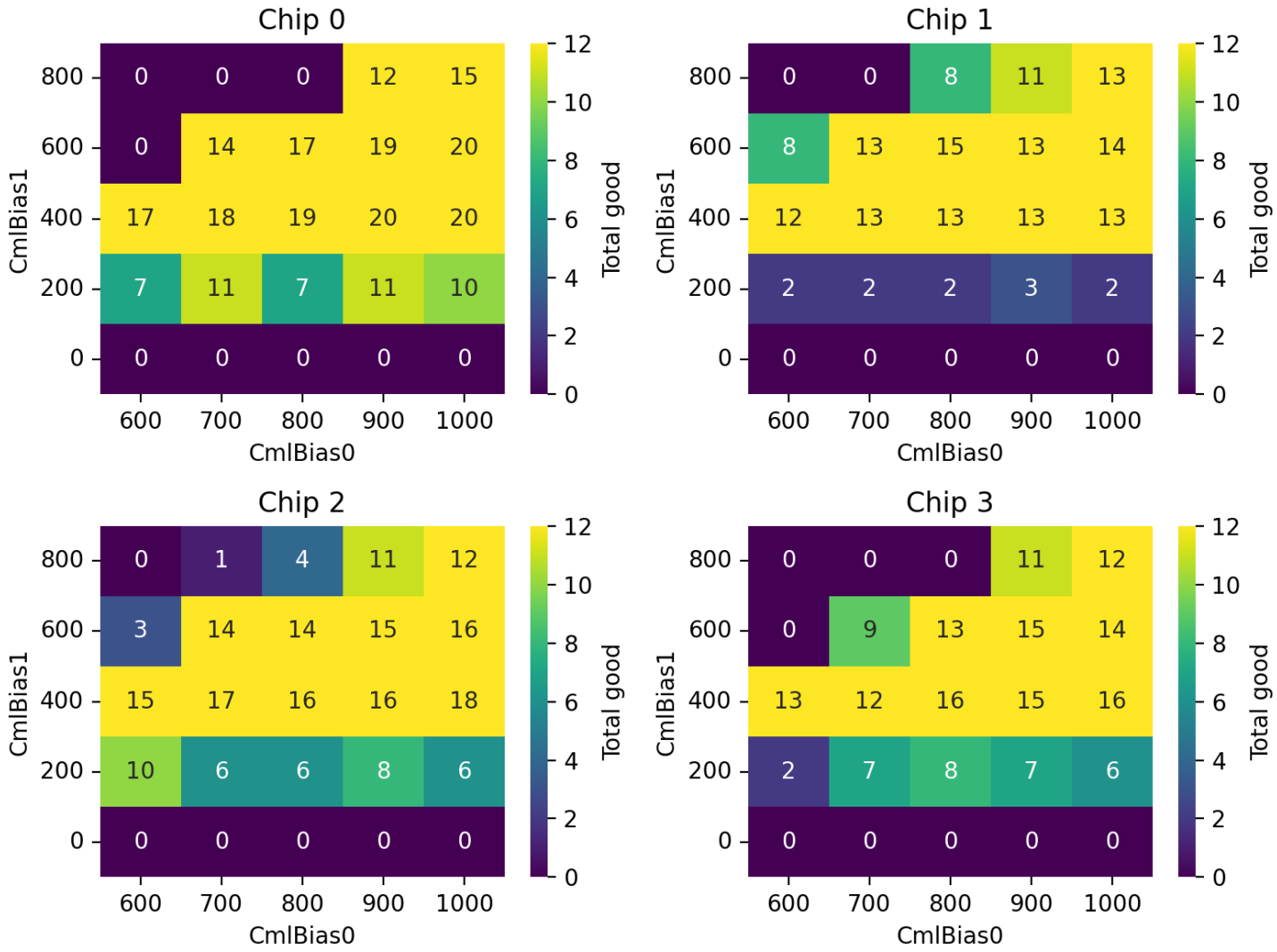
- Generally, data transmission worse on a quad
- Also, the different chips are not synchronised, which is expected



Quad cml bias scan

- Also consider ”total” good as a function of CmlBias
- Much more variety in data transmission quality for different chips and CmlBias settings
- Especially Chip 1 seems to perform worse
- Again, we clearly need pre-emphasis
- Best settings seem to consistently around CmlBias0 1000, CmlBias1 600

- Summary: **We can find delay and CmlBias settings at which data transmission works at 1.28 Gbps**



Automatic delay adjustment

- Can also try to make the delay value selection automatic by making it vary with counter from the gearbox process

→ While the gearbox loops over the different bits in the data stream, it will also vary delay settings

- Will lock onto the datastream when sees the sync headers and the deserialiser is at a good delay value

→ Works well for selecting a reasonable value

```
[11:01:00:506][  info  ][  ScanConsole  ][3776]: Sent configuration to all FEs in 1431 ms!
[11:01:00:507][  info  ][  ScanConsole  ][3776]: Checking com 0x1549a
[11:01:00:507][  info  ][    SpecRx     ][3776]: Active Rx channels: 0x4
[11:01:00:507][  info  ][    SpecRx     ][3776]: Active Rx lanes: 0x1
[11:01:00:507][  info  ][    SpecRx     ][3776]: Rx Status 0xf
[11:01:00:507][  info  ][    SpecRx     ][3776]: Number of lanes: 1
[11:01:00:507][  info  ][    SpecRx     ][3776]: Channel 2 Lane 0 synchronized with delay 19!
[11:01:00:539][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data (raw): 0x11549a08
[11:01:00:540][ error  ][     Rd53b     ][3776]: Chip serial number decoded from e-fuse data (0x154ba) does not appear in Chip "
r chip with ChipId = 12
[11:01:00:540][  info  ][     Rd53b     ][3776]: Chip serial number decoded with old format from e-fuse data: 0x1549a
[11:01:00:540][  info  ][  ScanConsole  ][3776]: ... success!
[11:01:00:540][  info  ][  ScanConsole  ][3776]: Checking com 0x15caa
[11:01:00:540][  info  ][    SpecRx     ][3776]: Active Rx channels: 0x2
[11:01:00:540][  info  ][    SpecRx     ][3776]: Active Rx lanes: 0x1
[11:01:00:540][  info  ][    SpecRx     ][3776]: Rx Status 0xf
[11:01:00:540][  info  ][    SpecRx     ][3776]: Number of lanes: 1
[11:01:00:540][  info  ][    SpecRx     ][3776]: Channel 1 Lane 0 synchronized with delay 23!
[11:01:00:571][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data (raw): 0x1154aa10
[11:01:00:571][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data: 0x15caa
[11:01:00:571][  info  ][  ScanConsole  ][3776]: ... success!
[11:01:00:571][  info  ][  ScanConsole  ][3776]: Checking com 0x154ba
[11:01:00:571][  info  ][    SpecRx     ][3776]: Active Rx channels: 0x1
[11:01:00:571][  info  ][    SpecRx     ][3776]: Active Rx lanes: 0x1
[11:01:00:571][  info  ][    SpecRx     ][3776]: Rx Status 0xf
[11:01:00:571][  info  ][    SpecRx     ][3776]: Number of lanes: 1
[11:01:00:571][  info  ][    SpecRx     ][3776]: Channel 0 Lane 0 synchronized with delay 29!
[11:01:00:603][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data (raw): 0x1154ba02
[11:01:00:603][ error  ][     Rd53b     ][3776]: Chip serial number decoded from e-fuse data (0x1549a) does not appear in Chip "
r chip with ChipId = 14
[11:01:00:603][  info  ][     Rd53b     ][3776]: Chip serial number decoded with old format from e-fuse data: 0x154ba
[11:01:00:603][  info  ][  ScanConsole  ][3776]: ... success!
[11:01:00:603][  info  ][  ScanConsole  ][3776]: Checking com 0x154ca
[11:01:00:603][  info  ][    SpecRx     ][3776]: Active Rx channels: 0x8
[11:01:00:603][  info  ][    SpecRx     ][3776]: Active Rx lanes: 0x1
[11:01:00:603][  info  ][    SpecRx     ][3776]: Rx Status 0xf
[11:01:00:603][  info  ][    SpecRx     ][3776]: Number of lanes: 1
[11:01:00:603][  info  ][    SpecRx     ][3776]: Channel 3 Lane 0 synchronized with delay 22!
[11:01:00:634][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data (raw): 0x1154ca00
[11:01:00:634][  info  ][     Rd53b     ][3776]: Chip serial number obtained from e-fuse data: 0x154ca
[11:01:00:634][  info  ][  ScanConsole  ][3776]: ... success!
```

Automatic delay selection

- Only caveat: this method always selects a value at the rising edge of the eye diagram (i.e. as soon as there's communication)
- This is not the best place to be, as jitter may make communication unstable

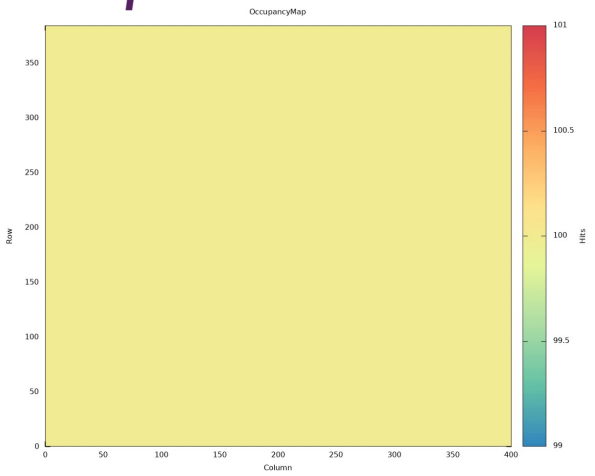
```
[17:39:21:779][ info ][    SpecRx    ][27997]: Active Rx lanes: 0x1
[17:39:21:779][ info ][    SpecRx    ][27997]: Rx Status 0x4
[17:39:21:779][ info ][    SpecRx    ][27997]: Number of lanes: 1
[17:39:21:779][ info ][    SpecRx    ][27997]: Locked at delay 18 with Rx Status 4
[17:39:22:280][ info ][    SpecRx    ][27997]: delay 0 with Rx Status 0, relevant bit is 0
[17:39:22:780][ info ][    SpecRx    ][27997]: delay 1 with Rx Status 0, relevant bit is 0
[17:39:23:280][ info ][    SpecRx    ][27997]: delay 2 with Rx Status 0, relevant bit is 0
[17:39:23:780][ info ][    SpecRx    ][27997]: delay 3 with Rx Status 100, relevant bit is 0
[17:39:24:280][ info ][    SpecRx    ][27997]: delay 4 with Rx Status 100, relevant bit is 0
[17:39:24:780][ info ][    SpecRx    ][27997]: delay 5 with Rx Status 100, relevant bit is 0
[17:39:25:281][ info ][    SpecRx    ][27997]: delay 6 with Rx Status 100, relevant bit is 0
[17:39:25:781][ info ][    SpecRx    ][27997]: delay 7 with Rx Status 100, relevant bit is 0
[17:39:26:281][ info ][    SpecRx    ][27997]: delay 8 with Rx Status 100, relevant bit is 0
[17:39:26:781][ info ][    SpecRx    ][27997]: delay 9 with Rx Status 100, relevant bit is 0
[17:39:27:281][ info ][    SpecRx    ][27997]: delay 10 with Rx Status 0, relevant bit is 0
[17:39:27:781][ info ][    SpecRx    ][27997]: delay 11 with Rx Status 0, relevant bit is 0
[17:39:28:282][ info ][    SpecRx    ][27997]: delay 12 with Rx Status 0, relevant bit is 0
[17:39:28:782][ info ][    SpecRx    ][27997]: delay 13 with Rx Status 0, relevant bit is 0
[17:39:29:282][ info ][    SpecRx    ][27997]: delay 14 with Rx Status 0, relevant bit is 0
[17:39:29:782][ info ][    SpecRx    ][27997]: delay 15 with Rx Status 0, relevant bit is 0
[17:39:30:282][ info ][    SpecRx    ][27997]: delay 16 with Rx Status 0, relevant bit is 0
[17:39:30:782][ info ][    SpecRx    ][27997]: delay 17 with Rx Status 100, relevant bit is 0
[17:39:31:283][ info ][    SpecRx    ][27997]: delay 18 with Rx Status 100, relevant bit is 0
[17:39:31:783][ info ][    SpecRx    ][27997]: delay 19 with Rx Status 100, relevant bit is 0
[17:39:32:283][ info ][    SpecRx    ][27997]: delay 20 with Rx Status 100, relevant bit is 0
[17:39:32:783][ info ][    SpecRx    ][27997]: delay 21 with Rx Status 100, relevant bit is 0
[17:39:33:283][ info ][    SpecRx    ][27997]: delay 22 with Rx Status 100, relevant bit is 0
[17:39:33:783][ info ][    SpecRx    ][27997]: delay 23 with Rx Status 100, relevant bit is 0
[17:39:34:284][ info ][    SpecRx    ][27997]: delay 24 with Rx Status 0, relevant bit is 0
[17:39:34:784][ info ][    SpecRx    ][27997]: delay 25 with Rx Status 0, relevant bit is 0
[17:39:35:284][ info ][    SpecRx    ][27997]: delay 26 with Rx Status 0, relevant bit is 0
[17:39:35:784][ info ][    SpecRx    ][27997]: delay 27 with Rx Status 0, relevant bit is 0
[17:39:36:284][ info ][    SpecRx    ][27997]: delay 28 with Rx Status 0, relevant bit is 0
[17:39:36:784][ info ][    SpecRx    ][27997]: delay 29 with Rx Status 0, relevant bit is 0
[17:39:37:285][ info ][    SpecRx    ][27997]: delay 30 with Rx Status 0, relevant bit is 0
[17:39:37:785][ info ][    SpecRx    ][27997]: delay 31 with Rx Status 0, relevant bit is 0
```

Digital scan

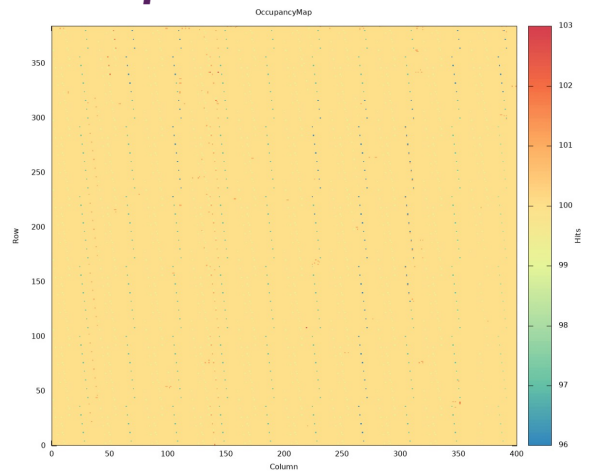
- With the automatic delay adjustments, can run digital scan on a quad at 1.28 Gbps
- Results look mostly good, apart from Chip 1
- Chip 1 had the worst eye diagram, so this is probably caused by being in the beginning of the eye

→ Plan to implement in the software and automatic adjustment, which shifts the delay value slightly higher, to be in a more stable region

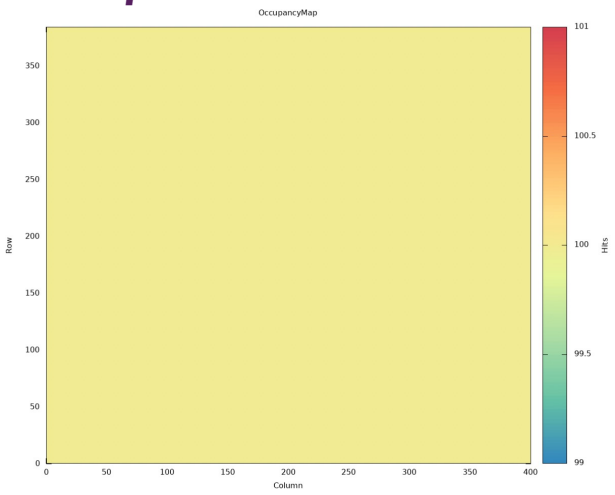
Chip 0



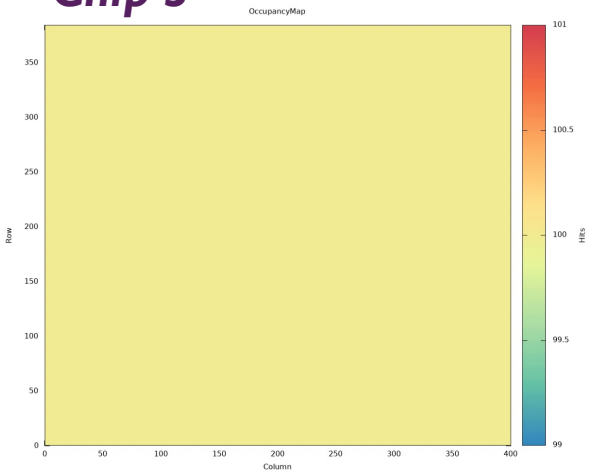
Chip 1



Chip 2



Chip 3



Summary

- Implemented a new deserialiser in the YARR firmware, which allows to manually vary deserialiser delay settings
- Can now run data transmission scans
- Overall: **1.28 Gbps readout is possible with YARR**
- Still to do (for this afternoon):
  - Make automatic increase for delay when running scans
  - Clean up firmware implementation
  - Clean up eye diagram scan implementation so it can be integrated into module QC

Thank you!

Questions?