Belle2Link implementation and data-error check

Yun-Tsung Lai

KEK IPNS

ytlai@post.kek.jp

1

Belle II Trigger/DAQ workshop 2022 @ Nara Women's University

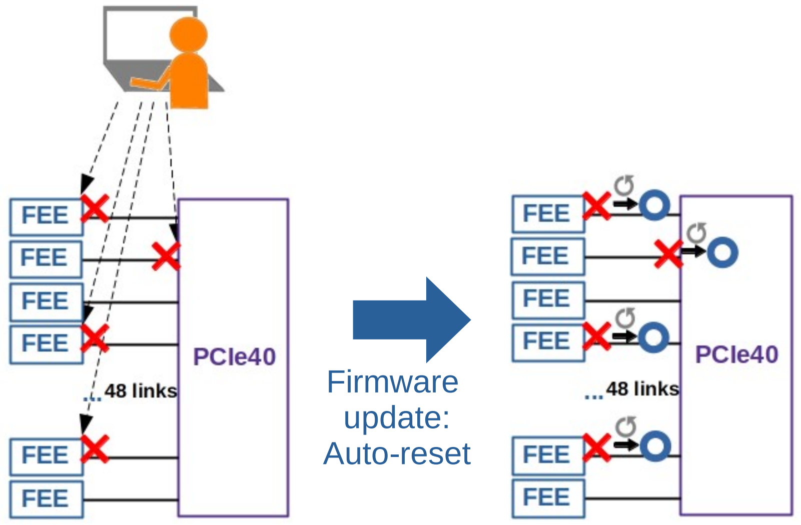
30th Nov., 2022

Outline

- Data link auto-reset function in firmwares
- Data-error check
- Summary

Auto-reset function

- In the beginning of PCIe40 development work, we observed that:
  - A data link may be turned down while one of the ends (FEE or PCIe40) is re-programmed.
  - If a PCIe40 is re-programmed, 50% of all of the links could be down.
  - Then, we need to re-program those devices to recover the links, or repeating is needed.
- An effective and convenient way to recover a link is needed.
  - Issue proper reset signal to the transceiver IPcore.
  - Do it automatically and properly.
- The design of the auto-reset function:
  - Define a link up flag to check the link's hardware status.
  - Issue reset while observing any instability.
  - Repeat it until link is stable.
- Result:
  - Tested to have ~100% readiness while any device is touched.
  - Work for all detectors.
  - No manual approach is needed.

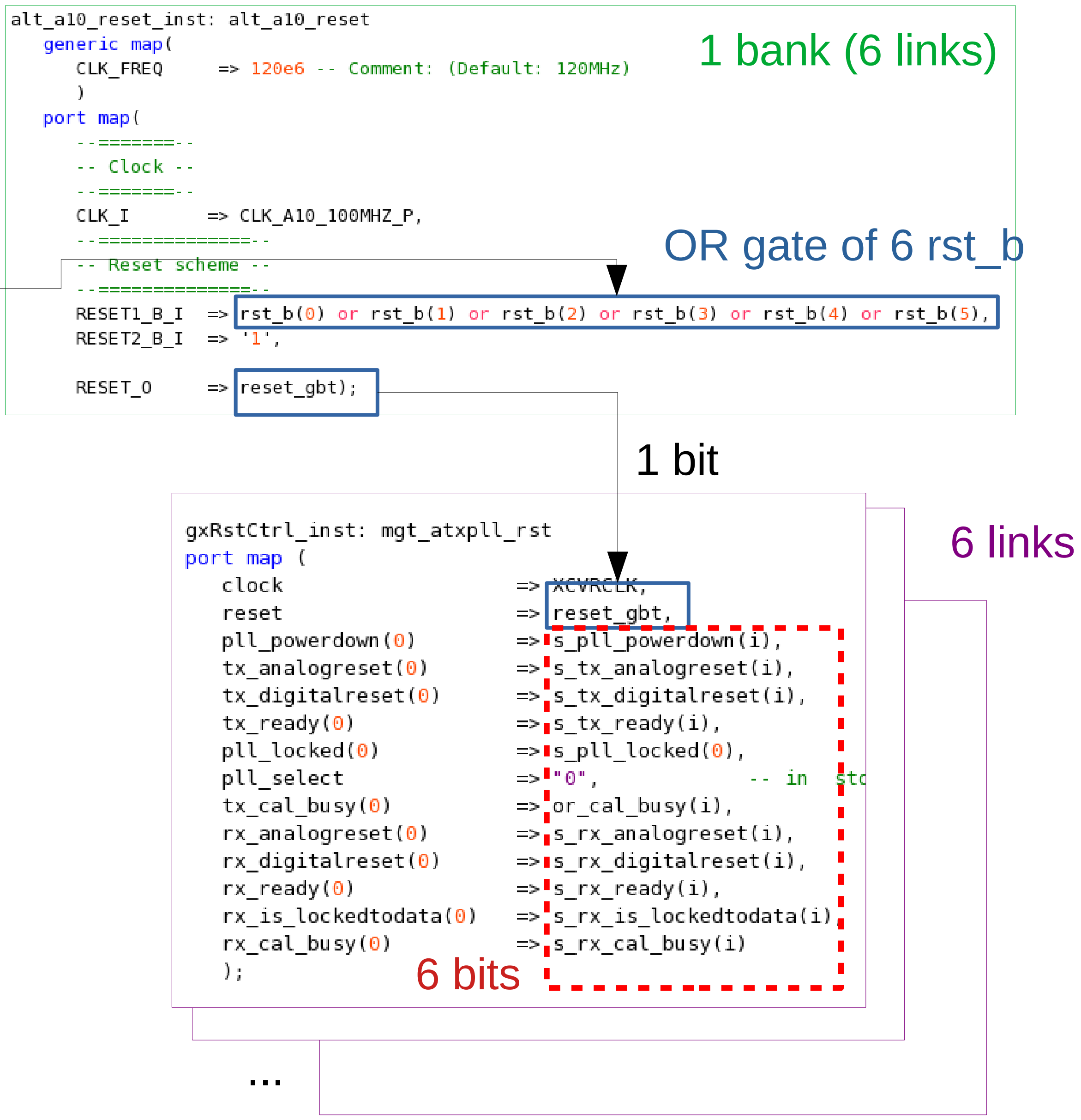


Link up flag

- The definition of the link up flag (lane_up) is based on the FPGA transceiver IPcore's interface.
- Basic definition:
  - gtp_ready/gtx_ready
  - notinstable (decoding error)
  - disparity error
  - PLL locked
  - rxvalid
  - Wrong position of K character
- It depends on the difference of transceiver, so adaption is needed for different FPGA.
- For the b2l and b2tt modules in firmware, they were using gtp_ready/gtx_ready to confirm the link's hardware status in the original design.
  - Now, it is modified for them to use this lane_up signal as flag.

Auto-reset function in PCIe40 firmware

- By slow control pio_cm3:
  - Trigger a **restart** signal to **TX** state-machine: generate **rst_b** signal.
- Inst of alt_a10_reset:
  - For 6 links.
- But rst_b is active low:
  - OR gate:
    All rst_b have to be fired
    at the same clock
    to trigger reset_gbt.



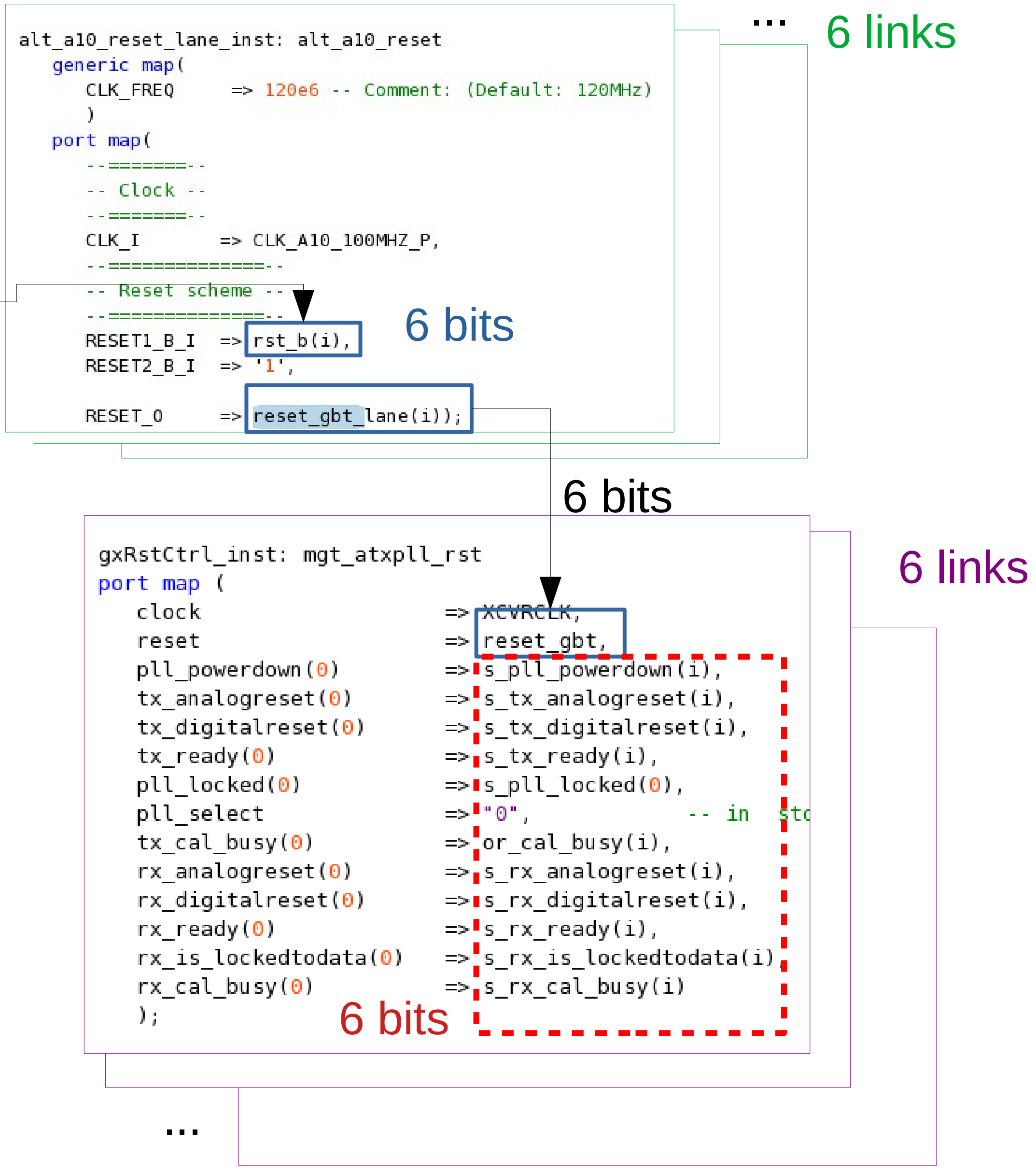
```
alt_a10_reset_inst: alt_a10_reset
   generic map(
      CLK_FREQ     => 120e6 -- Comment: (Default: 120MHz)
      )
   port map(
      --========--
      -- Clock --
      --========--
      CLK_I       => CLK_A10_100MHZ_P,
      --===============--
      -- Reset scheme --
      --===============--
      RESET1_B_I  => rst_b(0) or rst_b(1) or rst_b(2) or rst_b(3) or rst_b(4) or rst_b(5),
      RESET2_B_I  => '1',

      RESET_O     => reset_gbt);
```

```
gxRstCtrl_inst: mgt_atxpll_rst
port map (
   clock                    => XCVRCLK,
   reset                    => reset_gbt,
   pll_powerdown(0)         => s_pll_powerdown(i),
   tx_analogreset(0)        => s_tx_analogreset(i),
   tx_digitalreset(0)       => s_tx_digitalreset(i),
   tx_ready(0)              => s_tx_ready(i),
   pll_locked(0)            => s_pll_locked(0),
   pll_select               => "0",           -- in  sto
   tx_cal_busy(0)           => or_cal_busy(i),
   rx_analogreset(0)        => s_rx_analogreset(i),
   rx_digitalreset(0)       => s_rx_digitalreset(i),
   rx_ready(0)              => s_rx_ready(i),
   rx_is_lockedtodata(0)    => s_rx_is_lockedtodata(i),
   rx_cal_busy(0)           => s_rx_cal_busy(i)
   );
```

Auto-reset function in PCIe40 firmware (cont'd)

- By slow control pio_cm3:
  - Trigger a **restart** signal to **TX** state-machine: generate **rst_b** signal.
- New design:
  - Inst alt_a10_reset is made for each link.
  - So the reset signal to each link can be issued individually.

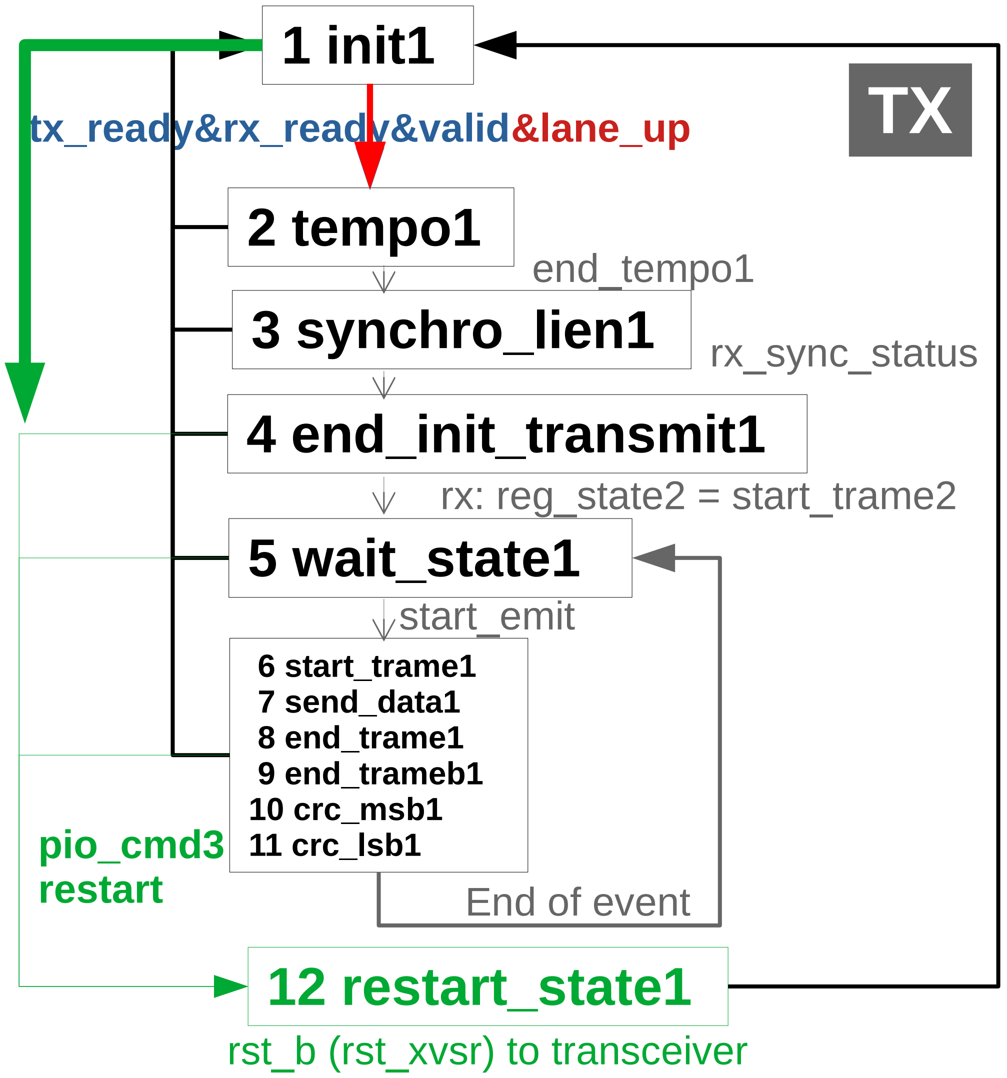


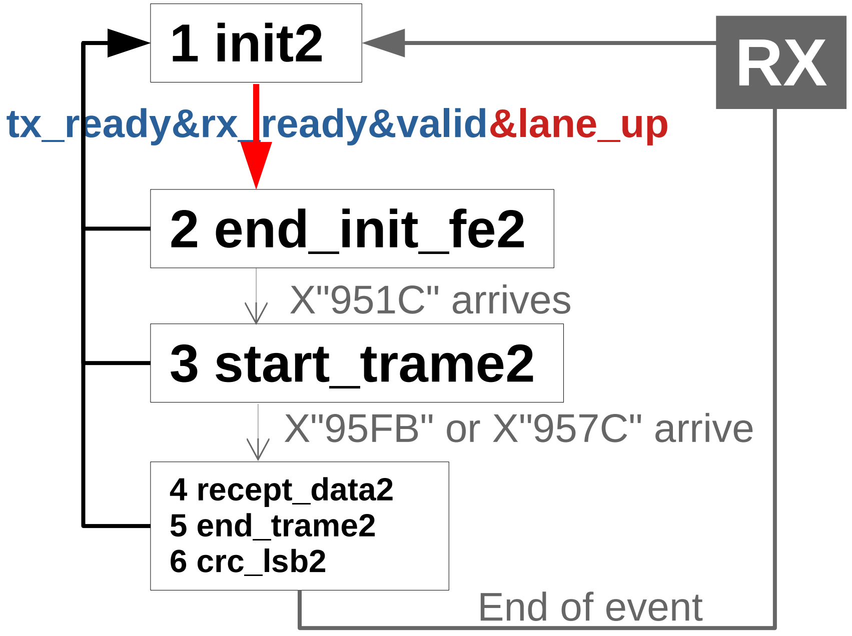
```
alt_a10_reset_lane_inst: alt_a10_reset
   generic map(
      CLK_FREQ      => 120e6 -- Comment: (Default: 120MHz)
      )
   port map(
      --=======--
      -- Clock --
      --=======--
      CLK_I        => CLK_A10_100MHZ_P,
      --==============--
      -- Reset scheme --
      --==============--
      RESET1_B_I   => rst_b(i),
      RESET2_B_I   => '1',

      RESET_O      => reset_gbt_lane(i));
```

```
gxRstCtrl_inst: mgt_atxpll_rst
port map (
   clock                  => XCVRCLK,
   reset                  => reset_gbt,
   pll_powerdown(0)       => s_pll_powerdown(i),
   tx_analogreset(0)      => s_tx_analogreset(i),
   tx_digitalreset(0)     => s_tx_digitalreset(i),
   tx_ready(0)            => s_tx_ready(i),
   pll_locked(0)          => s_pll_locked(0),
   pll_select             => "0",          -- in  st
   tx_cal_busy(0)         => or_cal_busy(i),
   rx_analogreset(0)      => s_rx_analogreset(i),
   rx_digitalreset(0)     => s_rx_digitalreset(i),
   rx_ready(0)            => s_rx_ready(i),
   rx_is_lockedtodata(0)  => s_rx_is_lockedtodata(i),
   rx_cal_busy(0)         => s_rx_cal_busy(i)
   );
```

Auto-reset function in PCIe40 firmware (cont'd)





- Modification in Belle2Link state machine:
  - Include the lane_up flag in the condition for state 1.
  - Other states go to state 1 while lane_up = '0'.
  - Allow state 1 go to state 12 to make the reset fully effective.

Auto-reset function in FEE firmware

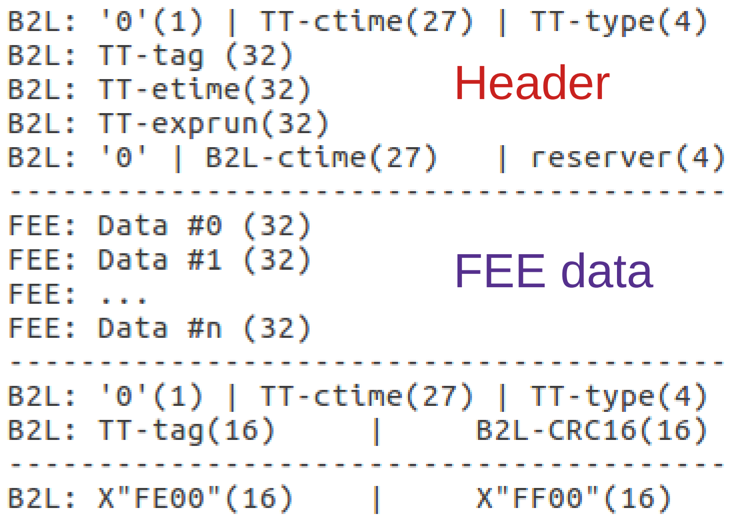
- Adaption is needed for different FPGA transceiver IPcore.
- TRG UT3 GTX:
  - The first one implemented with this function in 2018~2019.
  - The example design is originated from this.
- Finally finished in 2022 summer.
- Some difficulties:
  - TRG UT3 GTH: Re-compilation of 3D tracker firmware took a few months. It required improvement in the firmware's timing condition.
  - ECL collector: Firmware version problem and improvement in the firmware's timing condition.

Detector FEE	Transceiver
SVD	Spartan-6 GTP
CDC	Virtex-5 GTP
TOP	Kintex-7 GTX
ARICH	Virtex-5 GTP
ECL	Spartan-6 GTP
KLM	Virtex-6 GTX
TRG	UT3: Virtex-6 GTX, GTH UT4: UltraScale GTH, GTY

Data error check

- Process the RX data from Belle2Link state machine.

- Types of error check:
  - **crc16**: Checked at the end of event.
  - **Event tag incrementation**:
    Checked at the beginning of event.
    Ignored for the first event of a run.
  - **Exprun same as one in last event**:
    Checked at the beginning of event.
    Ignored for the first event of a run.
  - **tt ctime and type between header and trailer**:
    Checked at the end of event.
  - **Event tag between header and trailer (LSB 16 bit)**:
    Checked at the end of event.



```
B2L: '0'(1) | TT-ctime(27) | TT-type(4)
B2L: TT-tag (32)
B2L: TT-etime(32)
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)   | reserver(4)
-----------------------------------------
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE: ...
FEE: Data #n (32)
-----------------------------------------
B2L: '0'(1) | TT-ctime(27) | TT-type(4)
B2L: TT-tag(16)     |     B2L-CRC16(16)
-----------------------------------------
B2L: X"FE00"(16)    |     X"FF00"(16)
```

Header

FEE data

Trailer

Data error check at the beginning of an event

- **Event tag incrementation**:
- **Exprun same as one in last event**:

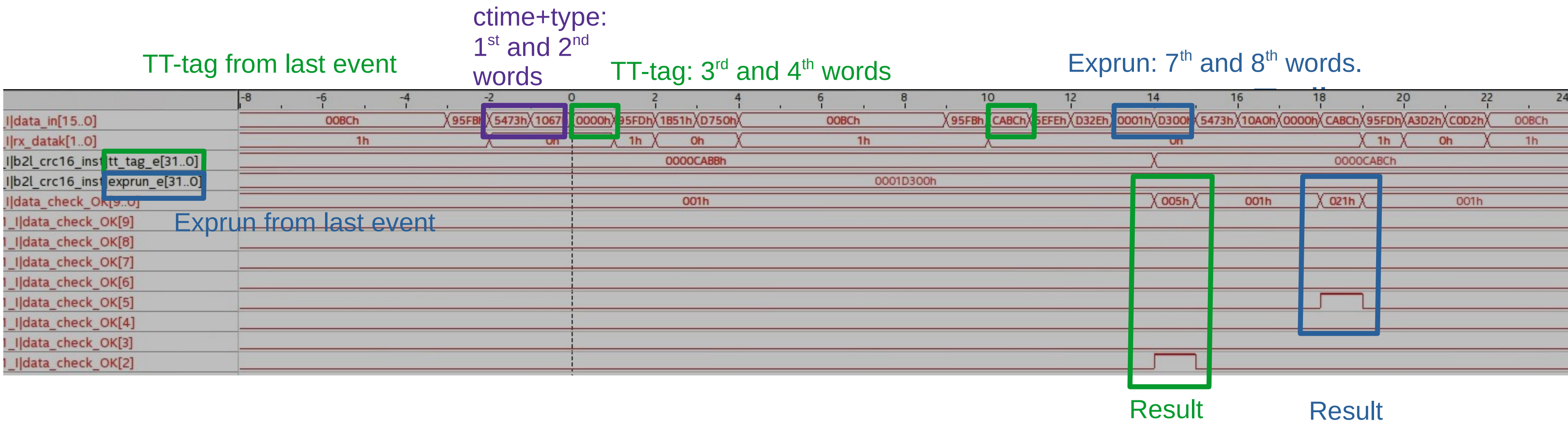
- Use runreset signal
  to identify the first event of a run.

```
B2L: '0'(1) | TT-ctime(27) | TT-type(4)
B2L: TT-tag (32)
B2L: TT-etime(32)
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)    | reserver(4)
------------------------------------------
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE: ...
FEE: Data #n (32)
------------------------------------------
B2L: '0'(1) | TT-ctime(27) | TT-type(4)
B2L: TT-tag(16)      |      B2L-CRC16(16)
------------------------------------------
B2L: X"FE00"(16)     |      X"FF00"(16)
```

Header

FEE data

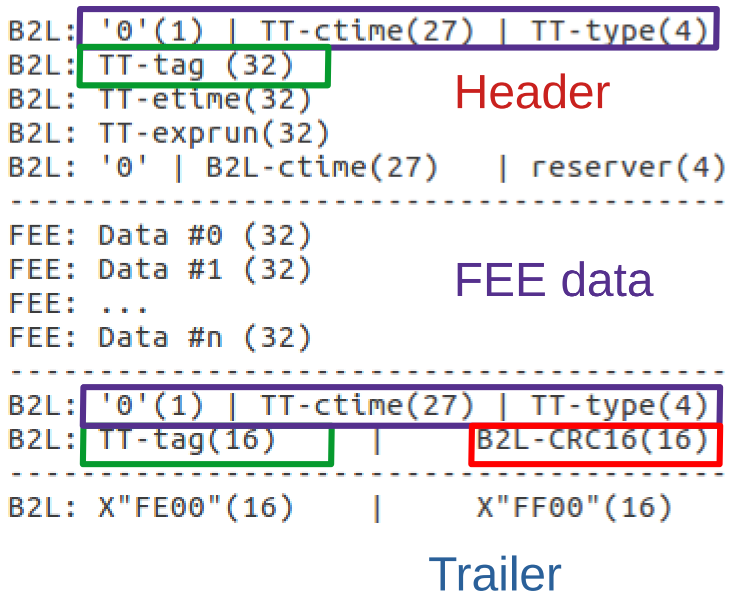
Trailer



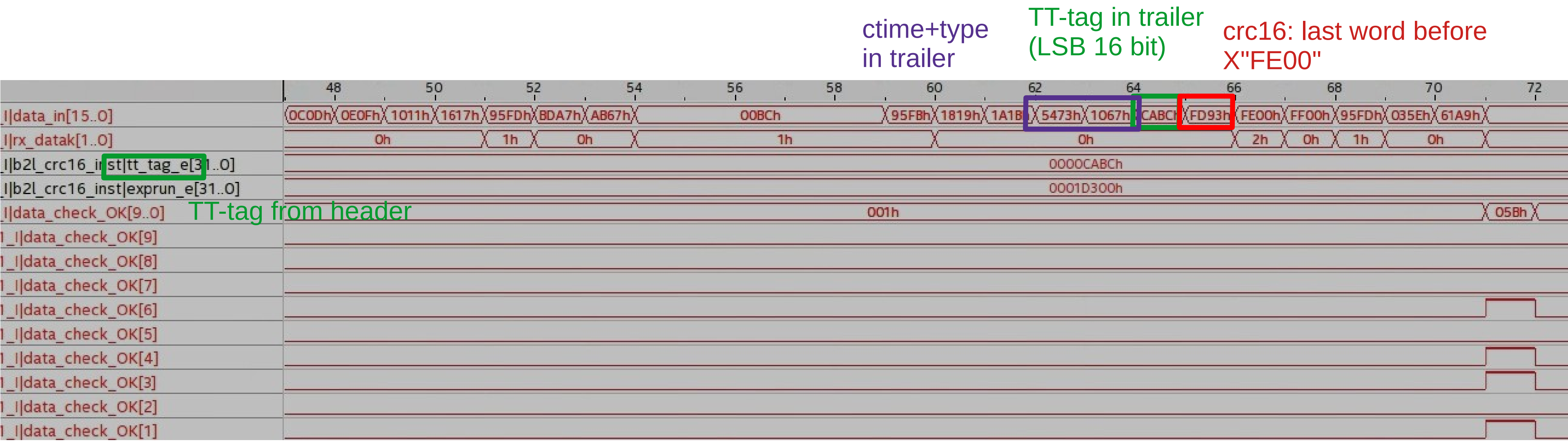
Data error check at the end of an event

- **crc16.**
- **tt ctime and type between header and trailer**.
- **Event tag between header and trailer (LSB 16 bit)**.

- Use wr_en_length_info from SM1_V2 to identify the end of an event.



```
B2L:  '0'(1) | TT-ctime(27) | TT-type(4)
B2L:  TT-tag (32)
B2L:  TT-etime(32)
B2L:  TT-exprun(32)
B2L:  '0'  | B2L-ctime(27)   | reserver(4)
----------------------------------------
FEE:  Data #0 (32)
FEE:  Data #1 (32)
FEE:  ...
FEE:  Data #n (32)
----------------------------------------
B2L:  '0'(1) | TT-ctime(27) | TT-type(4)
B2L:  TT-tag(16)     |        B2L-CRC16(16)
----------------------------------------
B2L:  X"FE00"(16)    |        X"FF00"(16)
```



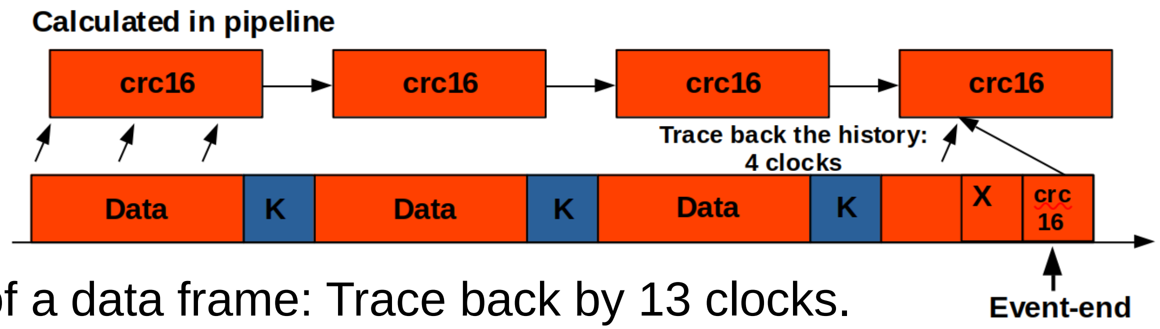
Results

Fake error problem in Sep. 2022

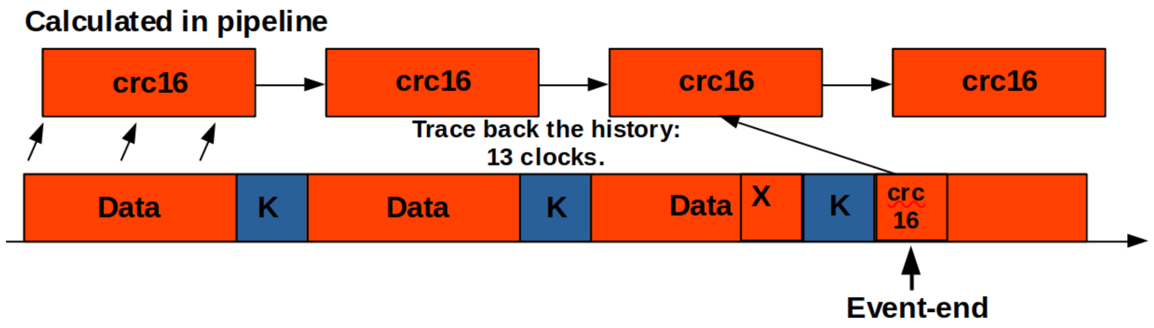
- We sometimes observed fake error raised by the module:
  - Data error check detected something, but software didn't.

- One of the reason was found in Sep. 2022 and has been fixed.
  - Only fake crc16 error.
  - Reason: Position of event-end flag.

- Flag is at the end of a data frame: Trace back by 4 clocks.



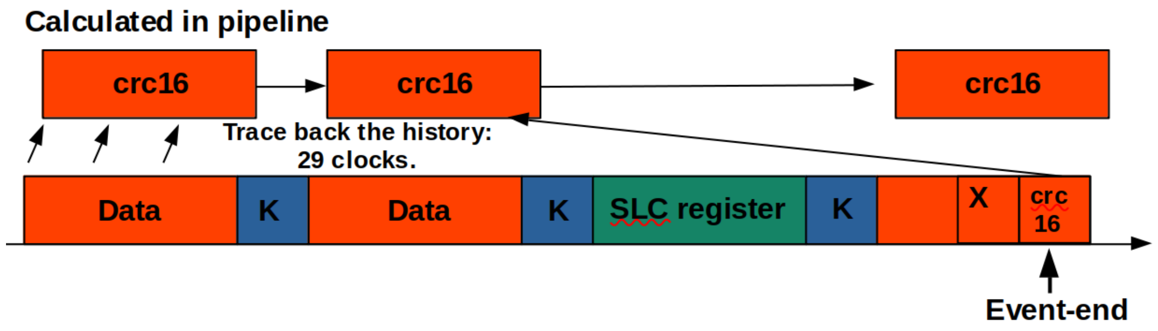
- Flag is at the beginning of a data frame: Trace back by 13 clocks.



Fake error problem in Sep. 2022 (cont'd)

- In the case that a slow control frame arrives right before the last data frame of an event: Need to trace back by 29 clocks.
  - We didn't notice it while developing the module using test bench.
    In a real detector system, slow control daemon software is always running, so there is such a daemon process of SLC in Belle2Link.

- The problem was fixed by updating the code.
  - Keep the valid crc16 pattern in shift register in a pipe-line, such that we don't need to trace back by so many clocks.



- For now, there is still fake error from time to time, but the frequency is very low (~1/month).
  - Need to run signaltap and keep it triggering for a long time during operation.

Summary

- Data link auto-reset function:
  - Implemented in all FEE firmwares and PCIe40 firmware.
  - ~100% readiness in system initialization.

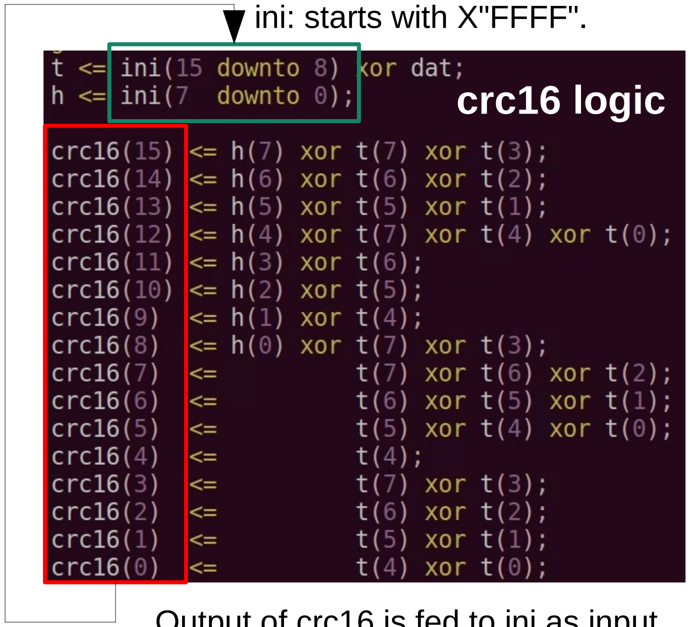
- Data error check:
  - Process the RX data from Belle2Link state machine to find data error.
  - For now, there is still fake error with a very low frequency.
    Still need to keep an eye on it.

Backup

crc16 check

- In b2l_transmitter, crc16 is calcuated for 8-bit data before going to the data FIFO. At receiver side, data arrives in 16-bit.
  - We can process the data in 8-bit but latency will be doubled.

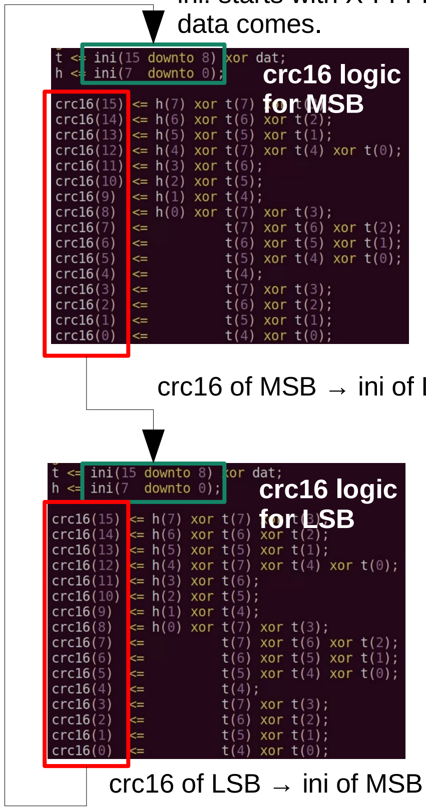
- Calculate crc16 of the MSB 8 bit first (w/o clock), and feed the output to the LSB 8 bit (1 clock), and feed the output of LSB to the MSB part in the next clock.
  - The crc of both MSB abd LSB 8 bit can be obtained in 1 clock.



```
t <= ini(15 downto 8) xor dat;
h <= ini(7  downto 0);

crc16(15) <= h(7) xor t(7) xor t(3);
crc16(14) <= h(6) xor t(6) xor t(2);
crc16(13) <= h(5) xor t(5) xor t(1);
crc16(12) <= h(4) xor t(7) xor t(4) xor t(0);
crc16(11) <= h(3) xor t(6);
crc16(10) <= h(2) xor t(5);
crc16(9)  <= h(1) xor t(4);
crc16(8)  <= h(0) xor t(7) xor t(3);
crc16(7)  <=          t(7) xor t(6) xor t(2);
crc16(6)  <=          t(6) xor t(5) xor t(1);
crc16(5)  <=          t(5) xor t(4) xor t(0);
crc16(4)  <=          t(4);
crc16(3)  <=          t(7) xor t(3);
crc16(2)  <=          t(6) xor t(2);
crc16(1)  <=          t(5) xor t(1);
crc16(0)  <=          t(4) xor t(0);
```

Output of crc16 is fed to ini as input of the next clock.


ini: starts with X"FFFF" until data comes.
crc16 logic for MSB
crc16 of MSB → ini of LSB part
crc16 logic for LSB
crc16 of LSB → ini of MSB part next clock

```
t <= ini(15 downto 8) xor dat;
h <= ini(7  downto 0);

crc16(15) <= h(7) xor t(7) xor t(3);
crc16(14) <= h(6) xor t(6) xor t(2);
crc16(13) <= h(5) xor t(5) xor t(1);
crc16(12) <= h(4) xor t(7) xor t(4) xor t(0);
crc16(11) <= h(3) xor t(6);
crc16(10) <= h(2) xor t(5);
crc16(9)  <= h(1) xor t(4);
crc16(8)  <= h(0) xor t(7) xor t(3);
crc16(7)  <=          t(7) xor t(6) xor t(2);
crc16(6)  <=          t(6) xor t(5) xor t(1);
crc16(5)  <=          t(5) xor t(4) xor t(0);
crc16(4)  <=          t(4);
crc16(3)  <=          t(7) xor t(3);
crc16(2)  <=          t(6) xor t(2);
crc16(1)  <=          t(5) xor t(1);
crc16(0)  <=          t(4) xor t(0);
```

```
t <= ini(15 downto 8) xor dat;
h <= ini(7  downto 0);

crc16(15) <= h(7) xor t(7) xor t(3);
crc16(14) <= h(6) xor t(6) xor t(2);
crc16(13) <= h(5) xor t(5) xor t(1);
crc16(12) <= h(4) xor t(7) xor t(4) xor t(0);
crc16(11) <= h(3) xor t(6);
crc16(10) <= h(2) xor t(5);
crc16(9)  <= h(1) xor t(4);
crc16(8)  <= h(0) xor t(7) xor t(3);
crc16(7)  <=          t(7) xor t(6) xor t(2);
crc16(6)  <=          t(6) xor t(5) xor t(1);
crc16(5)  <=          t(5) xor t(4) xor t(0);
crc16(4)  <=          t(4);
crc16(3)  <=          t(7) xor t(3);
crc16(2)  <=          t(6) xor t(2);
crc16(1)  <=          t(5) xor t(1);
crc16(0)  <=          t(4) xor t(0);
```

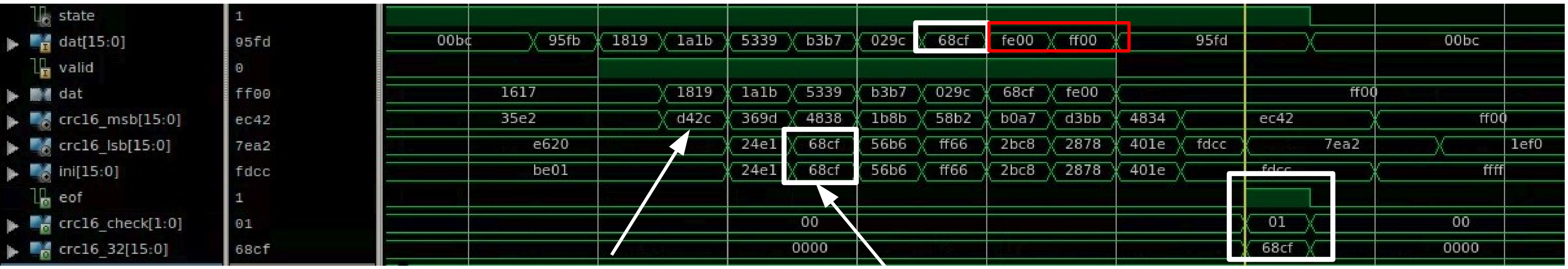
Logic gates. No clock.

1-clock.

An example of crc16 check

- Event-end is indicated by wr_en_length_info from SM1_V2.

1-clock before FE00: crc16 from data



state 1
dat[15:0] 95fd
valid 0
dat ff00
crc16_msb[15:0] ec42
crc16_lsb[15:0] 7ea2
ini[15:0] fdcc
eof 1
crc16_check[1:0] 01
crc16_32[15:0] 68cf
00bc 95fb 1819 1a1b 5339 b3b7 029c 68cf fe00 ff00 95fd 00bc
1617 1819 1a1b 5339 b3b7 029c 68cf fe00 ff00
35e2 d42c 369d 4838 1b8b 58b2 b0a7 d3bb 4834 ec42 ff00
e620 24e1 68cf 56b6 ff66 2bc8 2878 401e fdcc 7ea2 1ef0
be01 24e1 68cf 56b6 ff66 2bc8 2878 401e fdcc ffff
00 01 00
0000 68cf 0000

In each clock, the crc16 is calculated for both MSB and LSB 8 bit.

Calculated crc16 with rx data.

Check result at the end of an event.

crc16 of LSB → ini of MSB part next clock