Lecture: Data Acquisition

Mikihiko Nakao (KEK IPNS / SOKENDAI)
mikihiko.nakao@kek.jp

2025.10.22
Belle II Trigger/DAQ Workshop 2025

DAQ in general

Target
- Collision event of particles

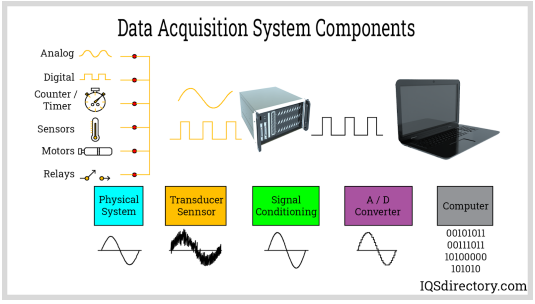
Physical System
- Drift chamber: ionization → avalanche
- Calorimeter: EM shower → visible light

Sensors
- Drift chamber: sense wires
- Calorimeter: photon detectors

Analog processing (Signal Conditioning)
- Pre-amplifier
- Pulse Shaping (analog circuit)

Digitization (A-D conversion)
- Pulse-height to a digital value (ADC)
- Time difference to a digital value (TDC)

Collect to a computer



At Belle II:
- **Sub-detector's task up to digitization**
- **DAQ group's task is to collect and store digitized data into computers**

Mission of Belle II DAQ

Collect digitized data from a huge number of detector channels

- **Digitized data** from 8 million pixels from PXD, 14,336 wires from CDC, …
- Data suppression: data only from channels with meaningful signal to be kept
- Data of the **same event** to pack into a single event file, in multiple steps

At the timing when interesting collision occurs

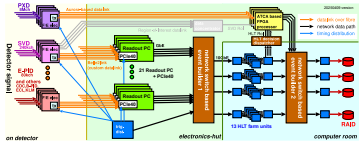
- Trigger decision given by the trigger group

At high rate

- 30 kHz with small dead-time fraction

With smooth operation and minimum downtime

- Run control, High voltage control including GUI
- System immune to various errors
- Quick restart in case of an error



With correct detector conditions

- Environment monitors, Configuration database, Data quality monitor, …

Data and Format

Mission of DAQ rephrased: collect bits and pack into a data file

Bit, Byte, Word

- Digital circuit handles 1-bit data '0' or '1' (electronics part of DAQ)
- Measured quantity is usually expressed in an n-bit integer
- Packing 8-bits (byte) or 16-/32-bits (word) for computers (computer part of DAQ)

Data format

- Measured quantities are stored in a memory space, and eventually to a file
- How to store quantities into memory has to be pre-defined

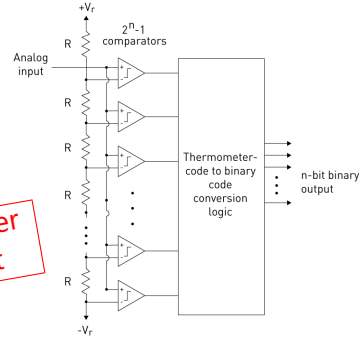
```
HSL: 0xFFAA(16) -- B2L header  |   HSLB-tag(16)
B2L: '0'(1) | TT-ctime(27)          | TT-type(4)
B2L: TT-tag(32)
B2L: TT-utime(32)
B2L: TT-exprun(32)
B2L: '0' | B2L-ctime(27)          |  reserved(4)
FEE: Data #0 (32)
FEE: Data #1 (32)
FEE:    ....
FEE: Data #n-1 (32)
B2L: '0'(1) | TT-ctime(27)          | TT-type(4)
B2L: TT-tag(16)              |       B2L-CRC16(16)
HSL: 0xFF55(16) | link-error(1) | CRC-error(15)
```

Belle2link example
from XWiki Belle2link page

ROOT header example
https://root.cern.ch/doc/master/header.html

Byte Range	Record Name	Description
0...3	"root"	Identifies this file as a ROOT file
4...7	Version	File format version
8...11	BEGIN	Byte offset of first data record (100)
12...15 [12...19]	END	Pointer to first free word at the EOF
16...19 [20...27]	SeekFree	Byte offset of FreeSegments record
20...23 [24...31]	NbytesFree	Number of bytes in FreeSegments record
24...27 [32...35]	nFree	Number of free data records
28...31 [36...39]	NbytesName	Number of bytes in TKey+TNamed for TFile at creation
32...32 [40...40]	Units	Number of bytes for file pointers (4)
33...36 [41...44]	Compress	Zip compression level (i.e. 0-9)
37...40 [45...52]	SeekInfo	Byte offset of StreamerInfo record
41...44 [53...56]	NbytesInfo	Number of bytes in StreamerInfo record
45...46 [57...58]	UUID vers	TUUID class version identifier
47...62 [59...74]	UUID	Universally Unique Identifier
63...99 [75...99]		Extra space to allow END, SeekFree, or SeekInfo to become 64 bit without moving this header

Digitization example



High speed Analog-to-Digital Conversion (ADC)

ADC for CDC can handle up to 40 MHz, and used at 32 MHz

Flash-ADC: brute force with many comparators

1023 comparators for 10-bit data

Commercial products

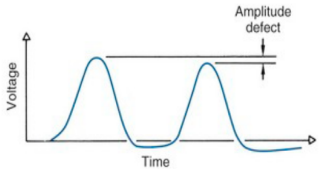
1,200 yen/channel for CDC

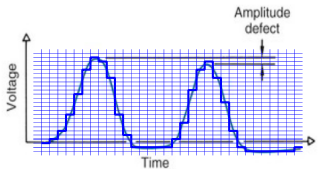
Optimal frequency and number of bit is detector dependent

AD9212ABCPZ-40 IC ADC 10BIT PIPELINED 64LFCSP Analog Devices Inc.	18 In Stock	1 : ¥9,676.00000 Tray	-	Tray	Active	10	40M	8	Differential, Single Ended	LVDS - Serial

Waveform can be recorded

O(100ns) CDC signal digitized in 32 ns unit



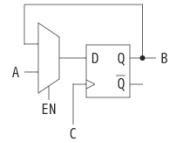


More correct signal height by finding the baseline shift

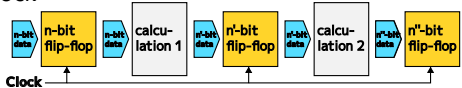
Time-sliced digital signal to be recorded by DAQ

Digital signal processing

Clock and flip-flop



- Digital value '0'/'1' can be stored in a **flip-flop** circuit at every **clock**
- **Registor** — n-bit data in n flip-flops
- **Calculation** can be made on register data (AND, OR, NOT, +, −, ...)
- **Synchronous circuit** — everything driven by a single clock
- Each calculation step has to finish within one clock cycle and can be divided into two steps if it takes longer
- Typical calculation in O(ns), typical clock at O(100 MHz)



Examples

- **Counter** — increment an n-bit value (under some condition)
- **Comparator** — '1' when an n-bit value is larger than other value, else '0'
- **Multiplexer** — switch the input signal (under some condition)

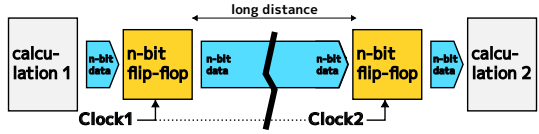
Memory

- Modify (**write**) or retrieve (**read**) ***m***'th value of 2^{n}-bit register
- The value ***m*** can be stored in a ***n***-bit register (**address**)
- 2^{n} byte memory is a 8 set of 2^{n} bit memories with a common address register

Data transfer

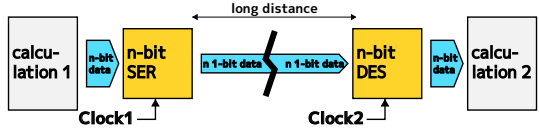
Parallel data transfer

- Data transfer as a synchronous circuit, but then Clock has to be also synchronous to data
- Digital signal will be degraded over a long line and difficult to control even at O(100MHz)



Serial data transfer

- **Serializer (SER)** to convert n-bit data at Clock1 frequency to 1-bit data at n×Clock1 frequency
- **Deserializer (DES)** to convert n 1-bit data at **Clock1** frequency to n-bit data at Clock1 frequency
- Clock1 can be different from Clock2 (next page)
- Faster clock is required but >GHz is easier to handle if no source-destination synchronization required



Serial data transfer

Principle

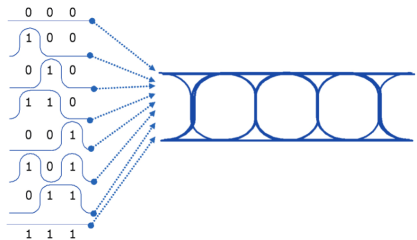
- n-bit data is serially transmitted bit-by-bit n-times faster
- Receiver has to find the **n-bit data boundary** (simple case: add start and stop bit)

8b10b encode

- Map **8-bit** data into **10-bit** (256 → 1024 patterns), using patterns with good mix of '0' and '1'
- 12 more non-data special patterns — e.g., to find the data boundary
  Example: 1001110100**1100000101**0111010100 — **decoded** as 0x00 comma 0x01,
  comma is a uniquely detected pattern because no other pattern has 00000
- Easy to implement: e.g.,
  `https://gitlab.desy.de/belle2/daq/ftsw/-/blob/master/b2tt/b2tt/b2tt_8b10b.vhd`

Clock data recovery (CDR)

- Clock frequency can be restored from data edges
  (if frequently switched between '0' and '1')
- A reference clock is required in the receiver
  with similar frequency (no need to be identical)
- Phase difference detected by the data edge timing



Protocol

Predefined rules to transfer meaningful data

- Both sender and receiver have to follow the same rules
- Data frequency, start and end of data
- 8b10b is a simple protocol to transfer 8-bit data
- **Our goal is to transfer the detector data in our format**

Idle time of data transfer

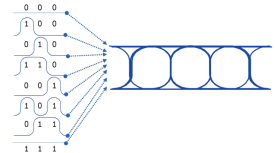
- Digital circuit cannot stop, it has to keep sending something
- But we have to send data only where there is an event data
- Certain data pattern can be used as idle data which can be thrown away by the receiver

Protocol design and implementation

- Simple protocol can be designed by ourselves (b2tt, belle2link)
- Industry standard protocols are more complex (Ethernet, USB, HDMI, etc)

Signal line and bandwidth

ideal serial signal



Single-ended

- One signal trace with a common ground (voltage reference)
- Easy, low-cost ⇔ less immune to external noise
- Slow signal <**O(10MHz)**, short distance (within circuit board)

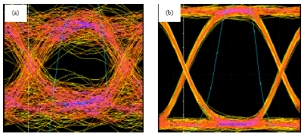
Differential

- Two lines per signal, positive swing and negative swing to suppress common mode noise
- Faster signal **O(100MHz)**, board-to-board connection **(O(10m))**

AC coupling

- DC-balanced fast signal can be decoupled by capacitors
- Ground level can be decoupled **to avoid ground loop**
  (Long distance ground and signal connection makes a loop antenna)

poor signal / good signal
(eye diagram)



poor signal can cause data error

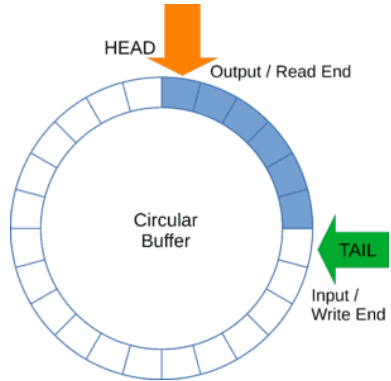
Optical transmission

- Convert electric signal to optical (transceiver device)
- More cost, very little attenuation and no electric noise
- FPGA-based serial line — **O(1 GHz)** to **O(10 GHz)** range, **O(100m)** to more

Memory and buffer

Memory as buffer

- Empty buffer: read address = write address
- Write and increment write address,
  Read and increment read address
- Read only when buffer is not empty
- Usual (single port) memory — can read only when not writing
  Dual port memory — can write and read at the same time



Ring buffer / FIFO / pipeline

- Address goes back to zero after the last address **(ring buffer)**
- First written in data is first read out **(FIFO)**
- Ring buffer can be used as a **pipeline** (or **delay**)
  if read/write address difference is constant

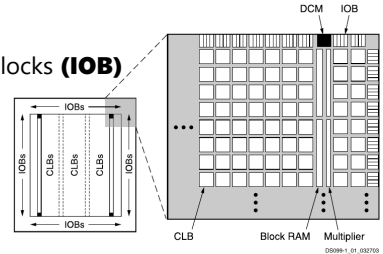
Buffer usage

- Buffer absorbs the timing difference of write and read
  (Dual port memory can also absorb the clock difference of write and read)
- **Example:** ADC data written to buffer at the sampling clock (e.g. 32 MHz),
  and data read out at data transfer clock (e.g. 127 MHz) when trigger is received

FPGA and firmware

FPGA basic structure

- FPGA is a programmable digital circuit device
- FPGA contains many Configurable Logic Blocks **(CLB)** and I/O Blocks **(IOB)**
- CLB contains a combination logic, multiplexer, and flip-flop
- Combination logic is a **look-up table** (a kind of memory)
- **Multiplexer** is used choose the signal route (also a memory)
- Therefore, FPGA logic is based on **configuration memory**
- Configuration memory file (bit file) is called **firmware**

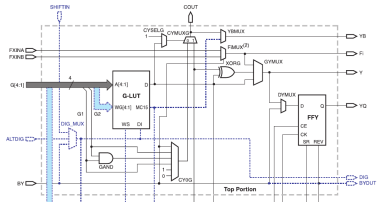


More functions included in FPGA

- Larger memory **(block RAM)**, in addition to flip-flop in CLB
- Clock circuit **(DCM)**, high speed serial line, etc

Firmware description language

- Digital circuit descrbed by a language (VHDL, Verilog)
- Like software language, one has to compile
- Every CLB (source file line) file is processed in parallel
- Complex software logic just runs slowly,
  but complex firmware logic will not work if it does not finish within the clock cycle



Look-Up Table (LUT)

Address as Input / Value as Output

- Very fast operation (O(ns)) for multi-100 MHz clock

Example 1: 2-bit comparator (4-bit LUT, 16-bit memory)

- Value '1' if input AB = CD
  (Value '1' for address 0000 0101 1010 1111, else value '0')

Example 2: 2-bit adder (4-bit LUT, 3 × 16-bit memory)

- Input AB + CD as 3-bit output
- Address 0000 → Value 0
- Address 0001 0100 → value 1
- Address 0010 1000 0101 → value 2
- Address 0011 1100 0110 1001 → value 3…

Any complex logic can be realized by LUT(s)

- Number of single LUT bit is limited (6-bit LUT for typical Xilinx FPGA)
- Multi-step LUT can make any complex logic (but becomes slower)

addr	comp	add
0000	1	000
0001	0	001
0010	0	010
0011	0	011
0100	0	001
0101	1	010
0110	0	011
0111	0	100
1000	0	010
1001	0	011
1010	1	100
1011	0	101
1100	0	011
1101	0	100
1110	0	101
1111	1	110

Trigger

Event rate is rather low

- Only up to several 100 Hz of interesting physis process at 5×10^{34} $cm^{-2}s^{-1}$ (including 60 Hz $B\bar{B}$)
- Only 10 times more even at the target SuperKEKB luminosity

Record only when there is interesting signal

- Decision to select interesting event from signal of the entire detector — **Trigger**
- Decision has to be made before recording the data

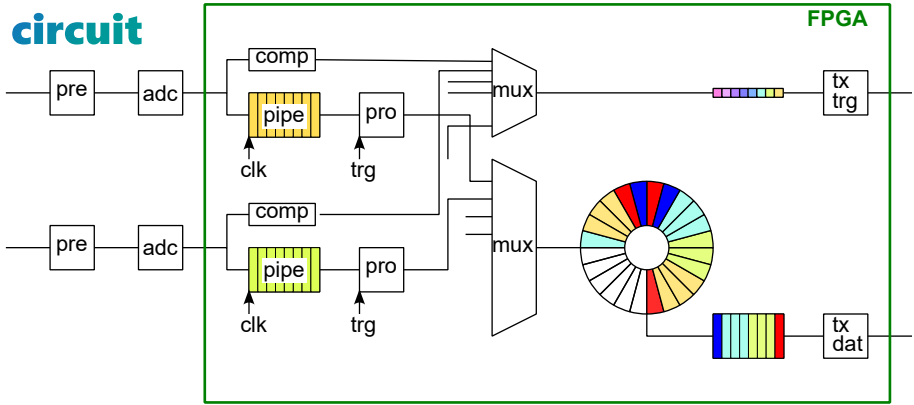
Data for the trigger decision

- **Trigger** — **priority in speed** than precision to decide within short time
- **DAQ** — **priority in precision** to be processed slowly later

Background events

- Charged particles and photons induced by beam can easily fulfill the trigger decision condition
- Priority in not to lose interesting events — backgrounds also recorded and suppressed later
- **Belle II is designed to handle up to 30 kHz triggers**

Readout circuit



- FPGA logic after ADC
- First part is driven by the samplng clock
- Pipeline buffer is to keep data for a given period
- Latter part is driven by the trigger
- Ring buffer size is limited, and too many triggers will overflow the buffer

Readout firmware

```
begin
  gen48: for i in 0 to N - 1 generate
    process (clock)
    begin
      if rising_edge(clock) then
        pipe_dat(i)(pipewr_ptr) <= adc_dat(i);
      end if;
    end process;
  end generate;

  process (clock)
  begin
    if rising_edge(clock) then
      if pipewr_ptr = PIPEWR_PTR_MAX - 1 then
        pipewr_ptr <= x"0000";
      else
        pipewr_ptr <= pipewr_ptr + 1;
      end if;
    end if;
  end process;
end;
```

- FPGA logic described in VHDL
- Left: Fill buffer at every clock (N-times parallel processing)
- Right: read 3 data from buffer at every trg

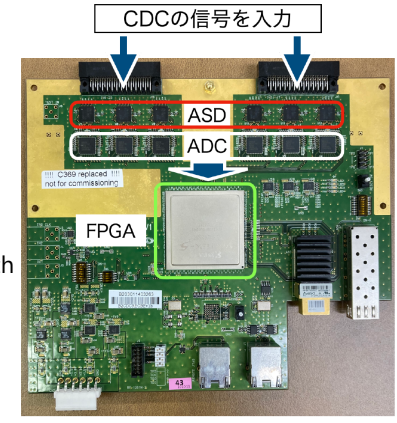
```
begin
  process (clock)
  begin
    trg1 <= trg;
    if trg = '1' and trg1 = '0' then
      trg2 <= '1';
      trg3 <= '1';
    elsif trg2 = '1' then
      trg2 <= '0';
    else
      trg3 <= '0';
    end if;
    if (trg = '1' and trg1 = '0') or trg3 = '1' then
      if piperd_ptr = PIPEWR_PTR_MAX - 1 then
        piperd_ptr <= x"0000";
      else
        piperd_ptr <= piperd_ptr + 1;
      end if;
    end if;
  end process;

  pro48: for i in 0 to N - 1 generate
    process (clock)
    begin
      if trg = '1' and trg1 = '0' then
        pro_dat(i)(0) <= pipe_dat(piperd_ptr);
      elsif trg2 = '1' then
        pro_dat(i)(1) <= pipe_dat(piperd_ptr);
      elsif trg3 = '1' then
        pro_dat(i)(2) <= pipe_dat(piperd_ptr);
      end if;
    end process;
  end generate;
end;
```

Readout board

Example: CDC readout board

- Analog and digital signal process for 48 wires
- 6 ASIC (ASD), 6 flash ADC,1 large FPGA
- Trigger data output, readout data output
- Trigger timing receiver port



Location

- Directly attached to the CDC to minimize the analog signal path
- High radiation area, Single Event Upset due to neutrons occur

Other subdetectors

- Signal type and amount are subdetector dependent
- Dedicated readout board is developed for each subdetector
- DAQ group provides unified interface
  — trigger distribution **(b2tt)** and data transfer **(belle2link)**
- Exception: PXD data is too large to be handled by belle2link

b2tt

Belle II original trigger distribution protocol

- Distribute and collect b2tt data using FTSW modules
- b2tt data transfer using LAN cable (4 differential lines)
- Clock, trigger, and many other fast control signals

b2tt mechanism

- 254 Mbps serial signal
  (Using only low-cost general LVDS input/output)
- 8b10b encoding, 2560-bit frame format,
  synchronized to the beam revolution
- Clock (127 MHz) is separately distributed
  (Clock Data Recovery is not used)
- Unified firmware component to decode and provide
  trigger, time stamp, reset signal, etc

b2tt implementation

- LAN (RJ-45) port on the readout board
- 4 differential signal to FPGA's general I/O pins
- Unified component to be included in the firmware

```
map_b2tt: entity work.b2tt
  port map (
    -- RJ-45
    clkp     => clk_p,
    clkn     => clk_n,
    trgp     => trg_p,
    trgn     => trg_n,
    rsvp     => rsv_p,  -- out
    rsvn     => rsv_n,  -- out
    ackp     => ack_p,  -- out
    ackn     => ack_n,  -- out

    -- link status
    b2clkup  => sig_clkup, -- out
    b2ttup   => sig_ttup,  -- out

    -- system clock and time
    sysclk   => clk_127m,  -- out
    utime    => sta_utime, -- out
    ctime    => sta_ctime, -- out

    -- run reset
    exprun   => buf_exprun,    -- exp(10)|run(14)|sub(8)
    runreset => sig_runreset,
    feereset => sig_feereset,
    b2lreset => sig_b2lrstin, -- reset belle2link state
    gtpreset => set_gtpreset, -- reset gtp

    -- trigger
    trgout   => sig_trgin,
    trgtyp   => sig_trgtyp,
    trgtag   => cnt_trg,
```

Belle2link

Belle II custom data transfer protocol

- Frontend and backend are connected via a pair of fibers
- FPGA's high-speed serial data transfer is used (2.54 Gbps)
- Backend was COPPER module, now replaced with PCIe40

belle2link mechanism

- Data is sent as one event, with header and trailer
- Header includes run/event number and time stamp
- Cyclic redundancy check (CRC) is added in the trailer to ensure the data is not broken
- Idle pattern is sent between event and event
- Frontend can be controlled from backend using the bidirectional link to read/write registers in frontend

belle2link implementation

- SFP optical transceiver on the frontend board
- Bidirectional signal connected to FPGA's high speed link
- Unified belle2link component to be included in firmware
- Write data into this component when trigger is received

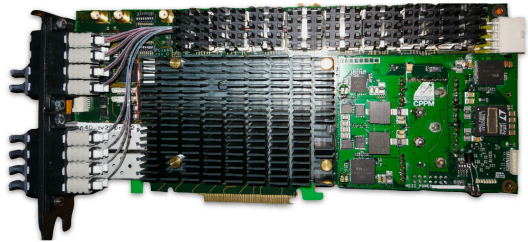
```
map_b2l: entity work.belle2link
  port map (
    -- Belle2link data
    b2lclk        => clk_127m,      -- in
    b2lwe         => b2l_we,        -- in
    b2ldata       => b2l_data,      -- in

    -- from/to GTP
    gtpclk        => txusrclk2,     -- in
    gtpready      => gtp_ready,     -- in
    rxdata        => rx_data,       -- in
    rxcharisk     => rx_charisk,    -- in
    txdata        => tx_data,       -- in
    txcharisk     => tx_charisk,    -- in
    rxcdrreset    => rx_cdrreset,   -- out
    rxreset       => rx_reset,      -- out
    gtpreset      => b2l_gtpreset,  -- out

    -- from/to B2TT
    runreset      => sig_runreset,  -- in
    trgtag        => cnt_trg,       -- in
    ttdata        => buf_ttdata,    -- in
    exprun        => buf_exprun,    -- in
    ctime         => sta_ctime,     -- in
    ttnext        => sig_ttnext,    -- out
    busy          => b2l_busy,      -- out

    -- registers (A7D8 space and A16D32 space)
    en32          => Para32En,      -- out
    wr32          => Para32Wr,      -- out
    rd32          => Para32Rd,      -- out
    data32i       => Para32De2Re,   -- in
    data32o       => Para32Re2De,   -- out
    addr32        => Para32Addr,    -- out
    ack           => ParaRdAck,     -- in
```

PCIe40



PCI-Express card

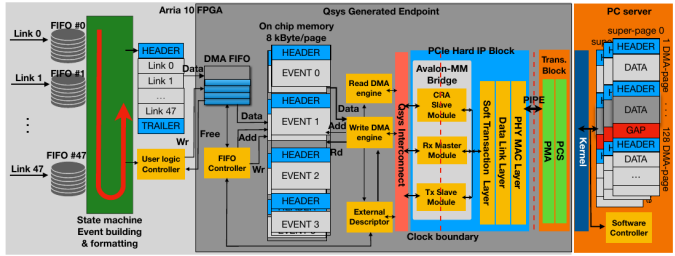
- Attach to a high-spec PC server
- 48 belle2link receivers in one card
- Intel Arria 10 FPGA (not Xilinx)
- 48 belle2link data packed and sent to PC via PCI-express

Developed for LHCb and ALICE

- Using only industrial standard technologies
- Many firmware components for LHC are reused at Belle II

Firmware

- Original part (belle2link, b2tt) are ported to PCIe40
- Basic function is to receive data into buffer and transfer to the next step (PC-server)



PC server

Device driver

- Software to run in the kernel space to access PCIe40 hardware
- Data transfer to the memory space where user software can access

basf2

- belle2link data converted to rawdata class format
- ROOT format is used when data is saved to a file

Data process on PC

- Removal of redundant info (no need to keep 48 copies)
- Error check of the data (e.g., CRC)
- Data transfer to the next (network programming)

Process and speed

- Unnecessary data copy slows down the performance
- Faster algorithm for CRC calculation
- When data cannot be accepted, trigger has to be stopped
  (busy from PCIe40 to the trigger distribution system using b2tt)

Data transfer over network

Ethernet

- 1Gbps / 10Gbps data transfer between PC is easy and low cost than any other custom made solutions
- Long distance data transfer can be easily done with optical Ethernet

IP / port

- Every network interface has an IP address
- Port number is assigned for each purpose (SSH 22, HTTPS 443, Postgres 5432, or any unused port)

UDP and TCP

- **UDP:** Sending only protocol without confirmation of reception (data may be lost)
- **TCP:** Make a connection, and wait until data is received, and data transmission is guaranteed

System call

- (TCP) receiver: bind、accept、read / sender: connect、write
- Data transfer is a combination of read and write, same as file read and write
- In reallity, one has to handle various kind of network errors (e.g. nsmd2 is solely based on system calls)
- In many cases, easier library is available (e.g., NSM2 library or daq_slc, ZeroMQ)

Event building and HLT

Event building - collect data from all PCIe40

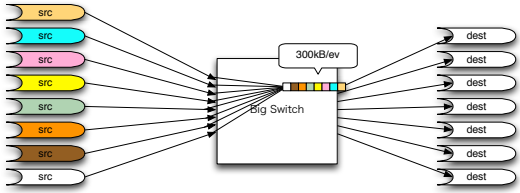
- Concept is the same as data collection inside readout board or PCIe40
- Put all input data into order
- Sender is a process inside PC of PCIe40, Receiver is a receiver process of HLT
- And a large network switch in between

Process the events with many PCs

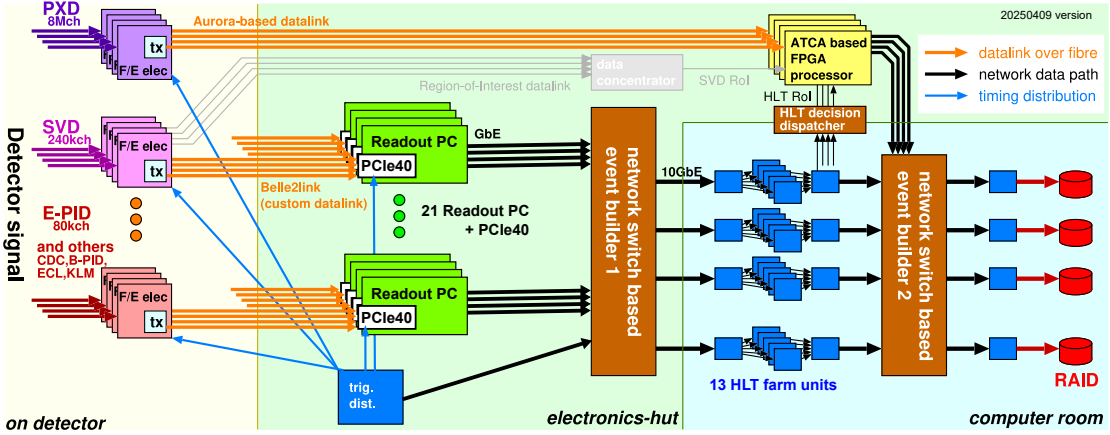
- Fully combined event at the full trigger rate is too much for a PC to receive
  → divide into multiple HLT unit (division by detector → division by event)
- Cyclic ordering, easy to handle but not necessarily efficient

HLT process

- Every event is sent to one of many worker PCs
- Worker PC runs basf2 for event reconstruction
- Time to process the event is not uniform
- Inside HLT, event is sent to a waiting worker PC
- Event ordering is not preserved



Belle II DAQ



- This is our current design — readout entire Belle II at 30 kHz trigger rate
- Since PXD data is too large, it is read out separately and merged later

To the next stage

Beyond the 30 kHz limit

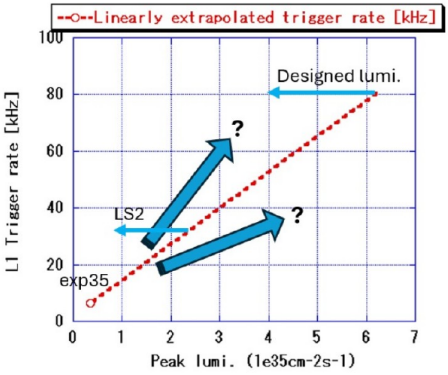
- Current limit is due to the SVD readout design, which uses the APV25 readout device designed for LHC
- New VTX will be able to handle 100 kHz

Many other limitations

- The other detector readout also designed for 30 kHz
- It should be easier to design and produce faster readout if funding/human resources are available

DAQ limitation

- 30 kHz limit is not only in the detector, also in DAQ



This workshp is to understand the current problem and think about future!