Trigger Timing Distribution

Mikihiko Nakao (KEK IPNS)

April 22, 2010

PXD-DAQ workshop im Gasthof zum Boarn, Rain am Lech

The Mission

- Distribute L1 trigger to everywhere needed, without losing
  (need some mechanism to avoid that)
- Minimize the deadtime
  (need careful design)
- Do various fast controls
  (because this is the only route)

TODAY: find out any possible inconsistency in the PXD timing signal handling

Concept

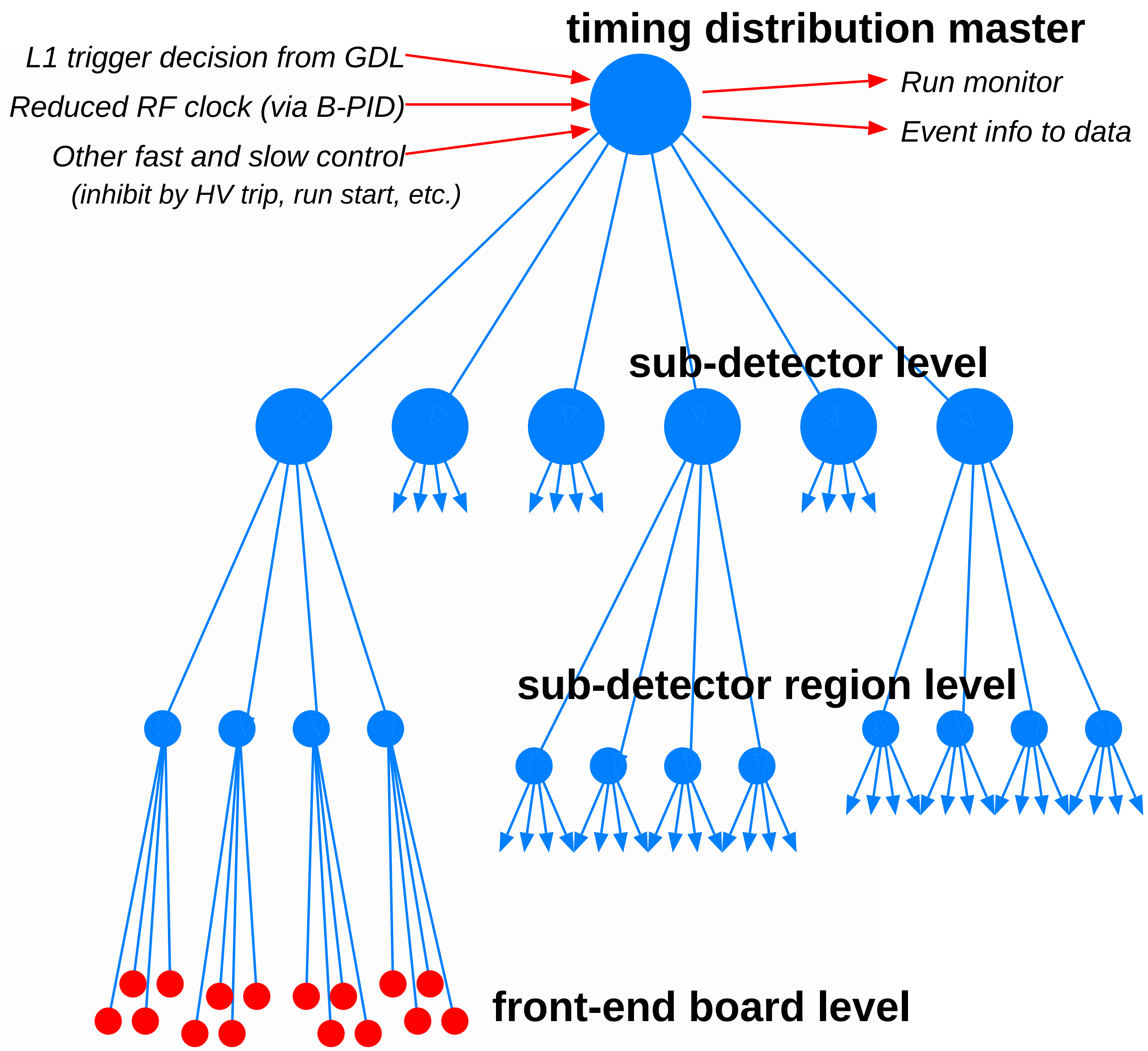
Trigger Timing Distribution (TTD) system is a tree-like network between the central DAQ and your front-end readout boards (and COPPERs), based on unified components and protocols:

To distribute

- Clock
- L1 Trigger
- Fast control signals

and to collect

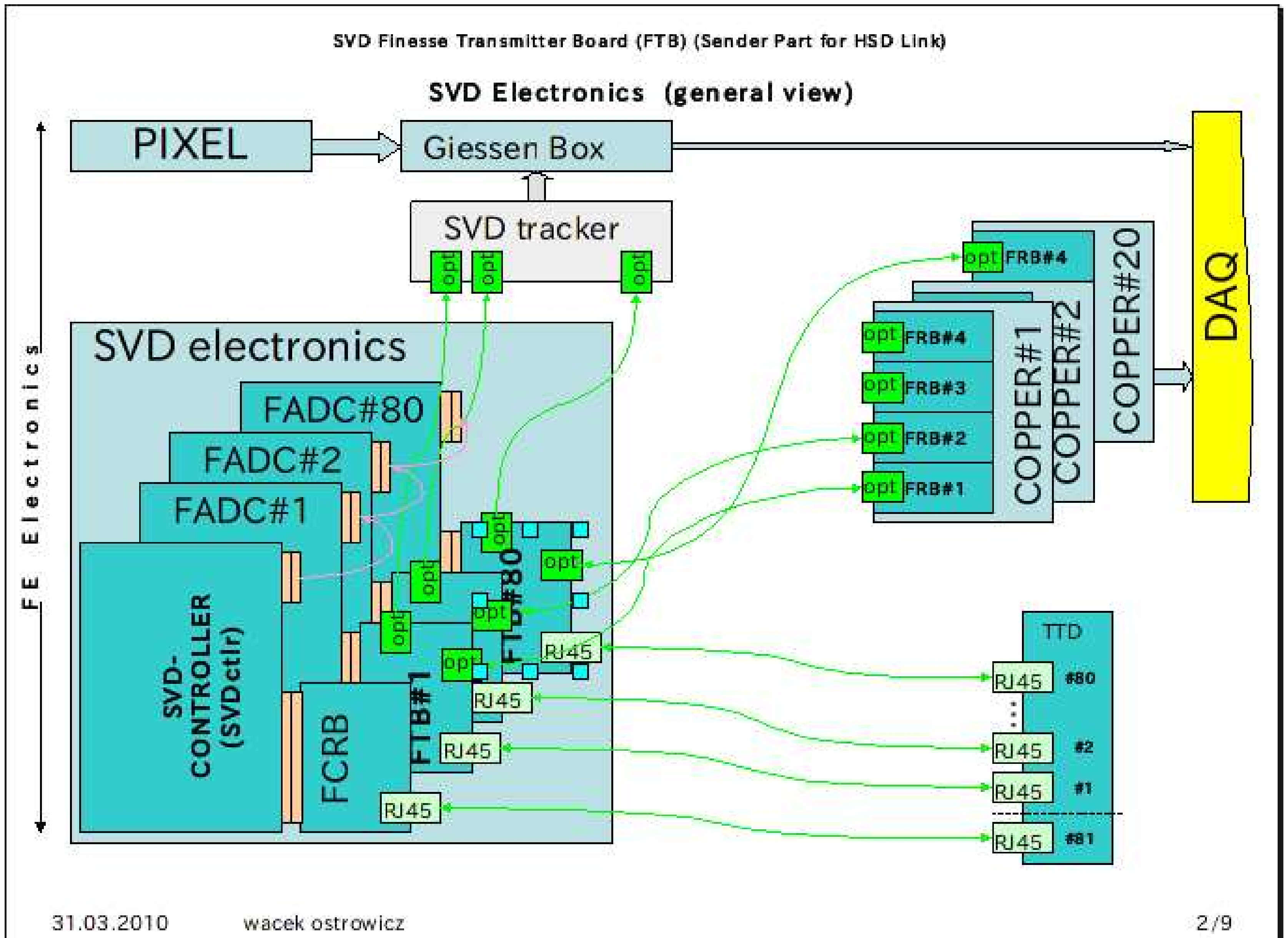
- Fast status information


timing distribution master
L1 trigger decision from GDL
Reduced RF clock (via B-PID)
Other fast and slow control
(inhibit by HV trip, run start, etc.)
Run monitor
Event info to data
sub-detector level
sub-detector region level
front-end board level

Unified Timing Distribution

- TTD for the door-to-door delivery of necessary signals, no additional in-house distribution system (please)
- Based on bad experience at Belle: SEQ, SVD (TTM) and TTD (for COPPER), all had (different) problems

SVD agreed to receive timing signals at each board



Hardware

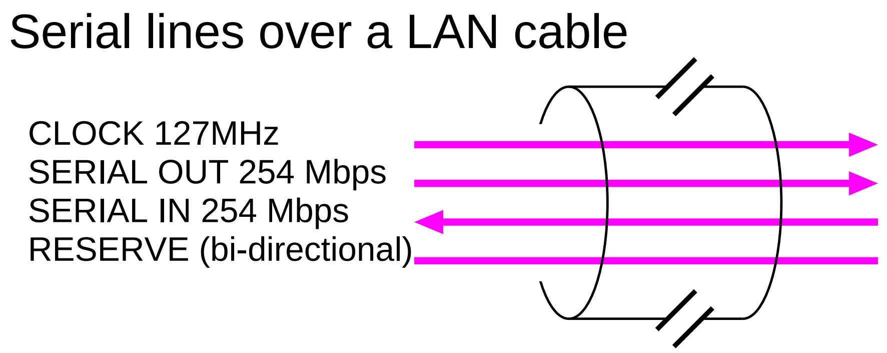
Unified components

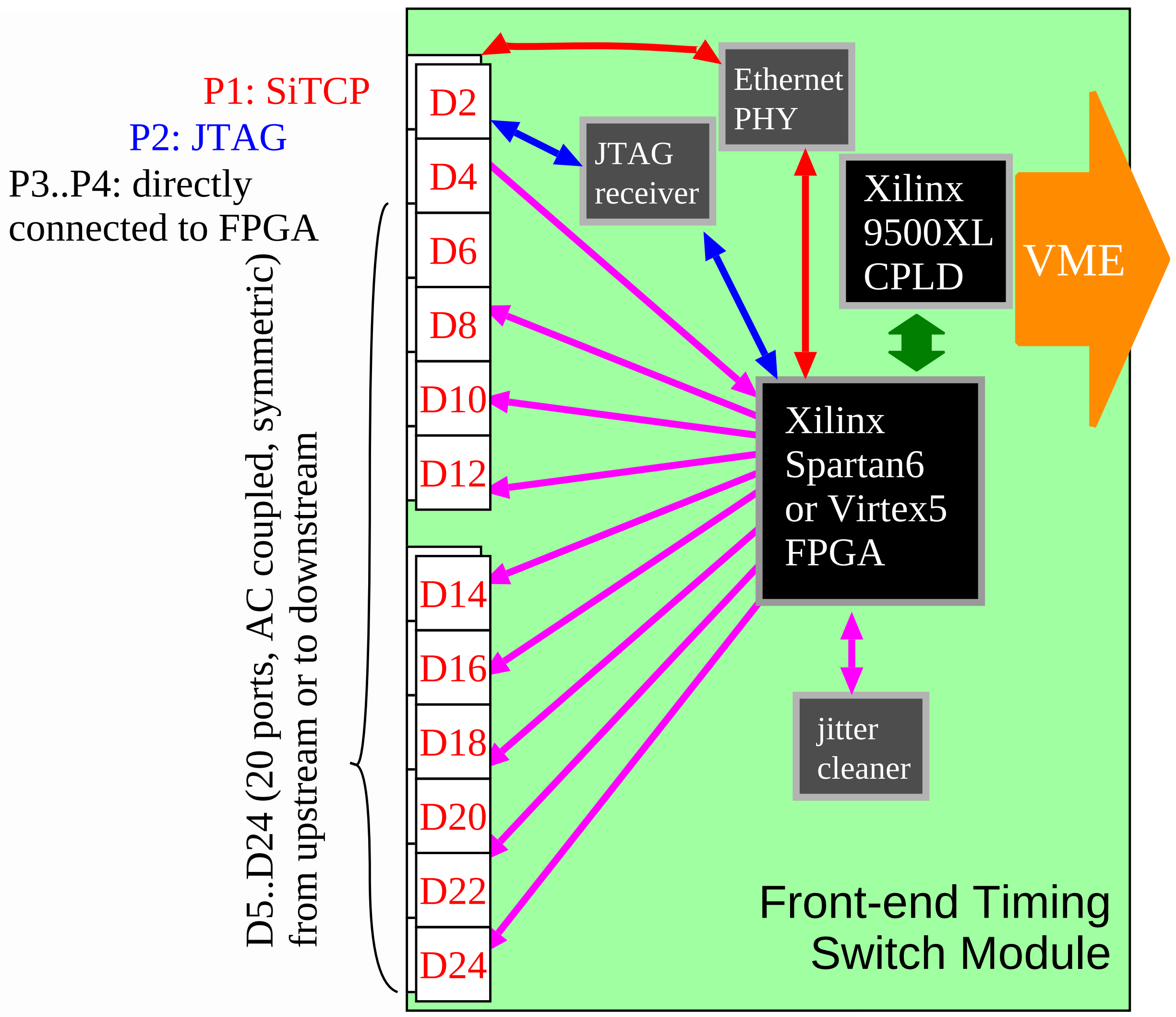
- **FTSW**: 1-to-20 front-end timing-signal switch module (single type module to cover everything)
- Off-the-shelf **LAN cables** to carry serial-link LVDS signals (up to 30 m length where front-end boards located)
- Xilinx **Virtex5 LXT** or **Spartan6 LXT** FPGAs at the frontend with requested peripherals (connectors, etc) (Other FPGA are not in consideration now. Any request?)
- Unified **timing receiver logic** in the frontend FPGA
- **COPPER** (and **TT-RX**) for most of the sub-detectors for fast signals at the backend entry point

These are designed to work with the unified datalink (Belle2link) for most of the detectors, but it should also apply to PXD except for COPPER related items.

FTSW (Front-end Timing SWitch) module

- 24-port RJ-45 on double-width 6U VME
  - 1x SiTCP
  - 1x JTAG-in-LVDS
  - 1x General I/O
  - 1x Rx port
  - 20x Tx ports
- Not necessarily be in VME, control from SiTCP (on-detector) or JTAG (inside-detector)
- Single 5V power for VME
  Single 3.3V power if not VME




P1: SiTCP
P2: JTAG
P3..P4: directly connected to FPGA
D5..D24 (20 ports, AC coupled, symmetric) from upstream or to downstream
D2
D4
D6
D8
D10
D12
D14
D16
D18
D20
D22
D24
Ethernet PHY
JTAG receiver
Xilinx 9500XL CPLD
VME
Xilinx Spartan6 or Virtex5 FPGA
jitter cleaner
Front-end Timing Switch Module

Prototype production this summer is under preparation

Cable

Since location of modules are in E-hut, on-detector or inside-detector, 20–30m cable length is needed (15m may be OK in some cases)

- Standard LAN cable pairing
  - 1–2 pair for clock out
  - 3–6 pair for serial data out
  - 7–8 pair for serial data in
  - 5–4 pair for reserve (direction not decided)
- Off-the-shelf LAN cable is desireable
  - CAT 7 shielded twisted cable (thick and heavy. . .)
  - Flat CAT 7 shielded twisted cable would be nice for inside-detector cabling

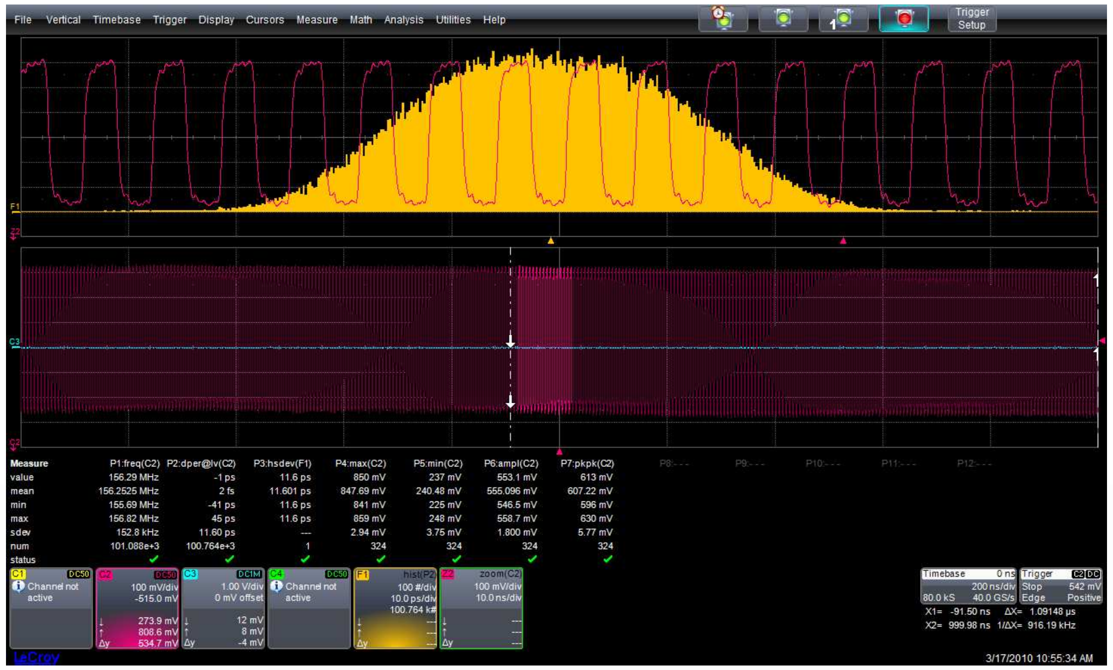
Clock

dummy horizontal text dummy horizontal text dummy h
text dummy horizontal text dummy horizontal text dum
zontal text

Clock

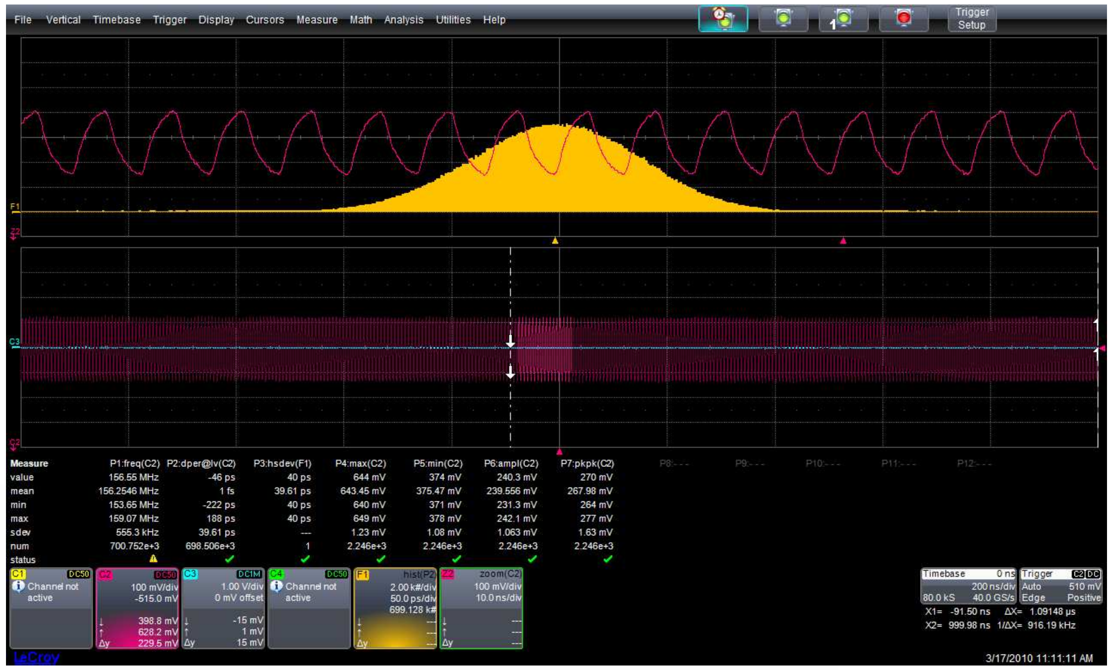
- **509 MHz** (508.87 MHz) and beam-revolution **(REVO)** signal from KEKB at a ultra-stable frequency (O (a few ppm))
- 127 MHz low-jitter clock is generated by B-PID (div by 4)
- **127 MHz** (always) and **REVO** signal (not always) are distributed to the front-end FPGA via TTD network
- Clock frequencies are synthesized in the front-end FPGA
  - 127 MHz — B-PID, datalink? (, GDL)
  - 101 MHz — PXD-DHH (can't run at 127 MHz?)
  - 63 MHz — ?
  - 42 MHz — ECL(, COPPER)
  - 32 MHz — SVD, CDC, B-PID
  - 2 MHz — ECL (. . . not necessarily the complete list)
- These clocks are jitter cleaned to O(a few 10ps) and synchronized to the REVO signal

Clock transmission test



```
Measure  P1:freq(C2)   P2:dper@lv(C2)  P3:hsdev(F1)  P4:max(C2)  P5:min(C2)  P6:ampl(C2)  P7:pkpk(C2)
value    156.29 MHz    -1 ps           11.6 ps       850 mV      237 mV      553.1 mV     613 mV
mean     156.2525 MHz  2 fs            11.601 ps     847.69 mV   240.48 mV   555.096 mV   607.22 mV
min      155.69 MHz    -41 ps          11.6 ps       841 mV      225 mV      546.5 mV     596 mV
max      156.82 MHz    45 ps           11.6 ps       859 mV      248 mV      558.7 mV     630 mV
sdev     152.8 kHz     11.60 ps        ---           2.94 mV     3.75 mV     1.800 mV     5.77 mV
num      101.088e+3    100.764e+3      1             324         324         324          324
status   ✓             ✓               ✓             ✓           ✓           ✓            ✓
```

3/17/2010 10:55:34 AM



```
Measure  P1:freq(C2)   P2:dper@lv(C2)  P3:hsdev(F1)  P4:max(C2)  P5:min(C2)  P6:ampl(C2)  P7:pkpk(C2)
value    156.55 MHz    -46 ps          40 ps         644 mV      374 mV      240.3 mV     270 mV
mean     156.2546 MHz  1 fs            39.61 ps      643.45 mV   375.47 mV   239.556 mV   267.98 mV
min      153.65 MHz    -222 ps         40 ps         640 mV      371 mV      231.3 mV     264 mV
max      159.07 MHz    188 ps          40 ps         649 mV      378 mV      242.1 mV     277 mV
sdev     555.3 kHz     39.61 ps        ---           1.23 mV     1.08 mV     1.063 mV     1.63 mV
num      700.752e+3    698.506e+3      1             2.246e+3    2.246e+3    2.246e+3     2.246e+3
status   ⚠             ✓               ✓             ✓           ✓           ✓            ✓
```

3/17/2010 11:11:11 AM

Cable attenuation/jitter (attached with a CAT6 relay adaptor)

cable	jitter	swing	cable	jitter	swing
reference	11.6ps	555mV	CAT6e UTP 15m flat	75.0ps	132mV
CAT5e STP 1m	15.0ps	490mV	CAT6e UTP 20m flat	97.5ps	93mV
CAT5e UTP 10m	27.8ps	339mV	CAT7 STP 10m flat	24.9ps	321mV
CAT5e UTP 15m	32.4ps	314mV	CAT7 STP 15m flat	39.6ps	240mV
CAT5e UTP 50m	65.8ps	152mV	CAT7 STP 20m	24.0ps	376mV
CAT6e UTP 10m flat	41.3ps	258mV	CAT7 STP 30m	30.2ps	286mV

Cable jitter recovery by FPGA PLL

- PLL in Virtex5 can be used as a jitter cleaner
- Test 1: jitter cleaning of good clock (1m cable connection) 16.9ps ⇒ 14.7ps
- Test 2: jitter cleaning of bad clock (50m cable connection) 43.2ps ⇒ 15.3ps
- These jitters are not good enough for Barrel PID, but maybe OK for most of other subdetectors
- Dedicated jitter cleaner chip (e.g., TI CDCE62002) should do a better job (planned at every SW module, but not at the subdetector readout board).

Trigger over Serial-Link

dummy horizontal text dummy horizontal text dummy h
text dummy horizontal text dummy horizontal text dum
zontal text

Trigger Timing and Serial Link

- Trigger timing is determined at GDL with a few ns resolution (with B-PID trigger, or otherwise ~20ns from ECL)
- Will be quantized to the 127 MHz clock
- Instead of directly sending this signal, it is encoded in a serial-data word (24 or 32 bit) that is distributed every ~200 ns or less, including the flag for the L1 decision (5-bit fine timing info within ~200 ns coarse timing)
- This leads to a line rate of 127 to 254 Mbps
- Trigger timing is recovered at F/E (~60 ns long pulse)
- The same serial data word to carry various other info
  - Revo signal timing
  - Trigger tag and timing source info (B-PID, ECL or other)
  - Various control signal such as RUN START
- Design based on Belle COPPER/TTD experiences (10-bit over 508 Mbps for control, fine trigger timing was not encoded)

SERDES driven by FPGA

- Direct connection of serial lines to FPGA
  (external SERDES requires too many traces for parallel lines)
- Serialization and deserialization in FPGA
  - 254 Mbps may be already slow enough to be directly programmed in FPGA.
  - Dedicated SERDES will have better timing performance, ISERDES/OSERDES (Virtex5) or ISERDES2/OSERDES2 (Spartan6)
  - Delay adjustment from the serial data is needed IODELAY in Virtex5/Spartan6
- Other advantages
  - No clock recovery from serial data, smaller latency
  - Many ISERDES/OSERDES port in one FPGA
    (e.g., much more than RocketIO ports)

ISERDES and OSERDES

- Up to 6 bits (Vertex5) or 5 bits (Spartan6)
  - 4-bit × 8-clock-cycle (63 MHz) to make a 32-bit word
  - Rotation inside 8-clock-cycle word with 63 MHz logic
  - Bit-rotation inside 4-bit by ISERDES
- Two clocks (254MHz and 63MHz) generated by single PLL

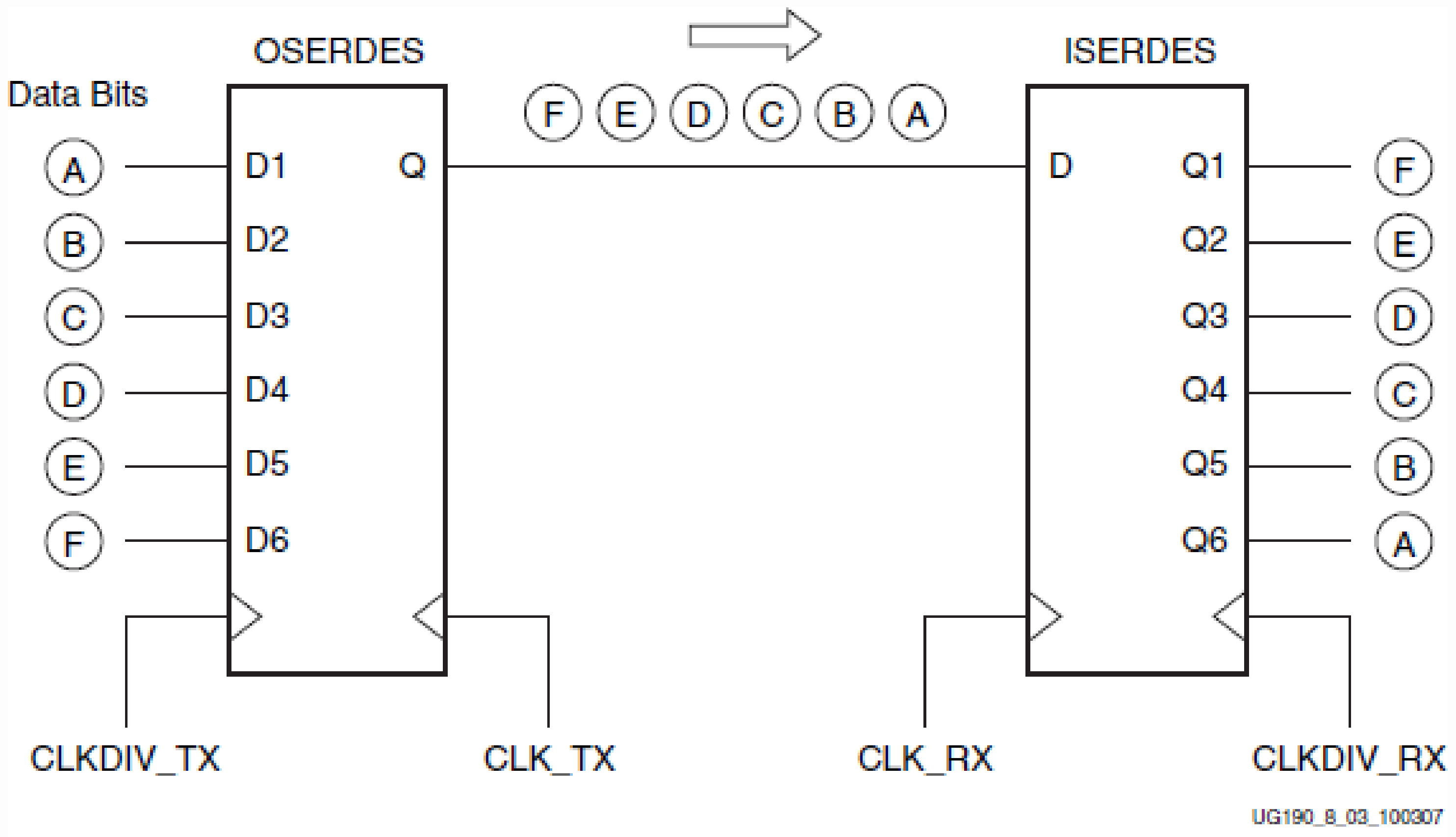
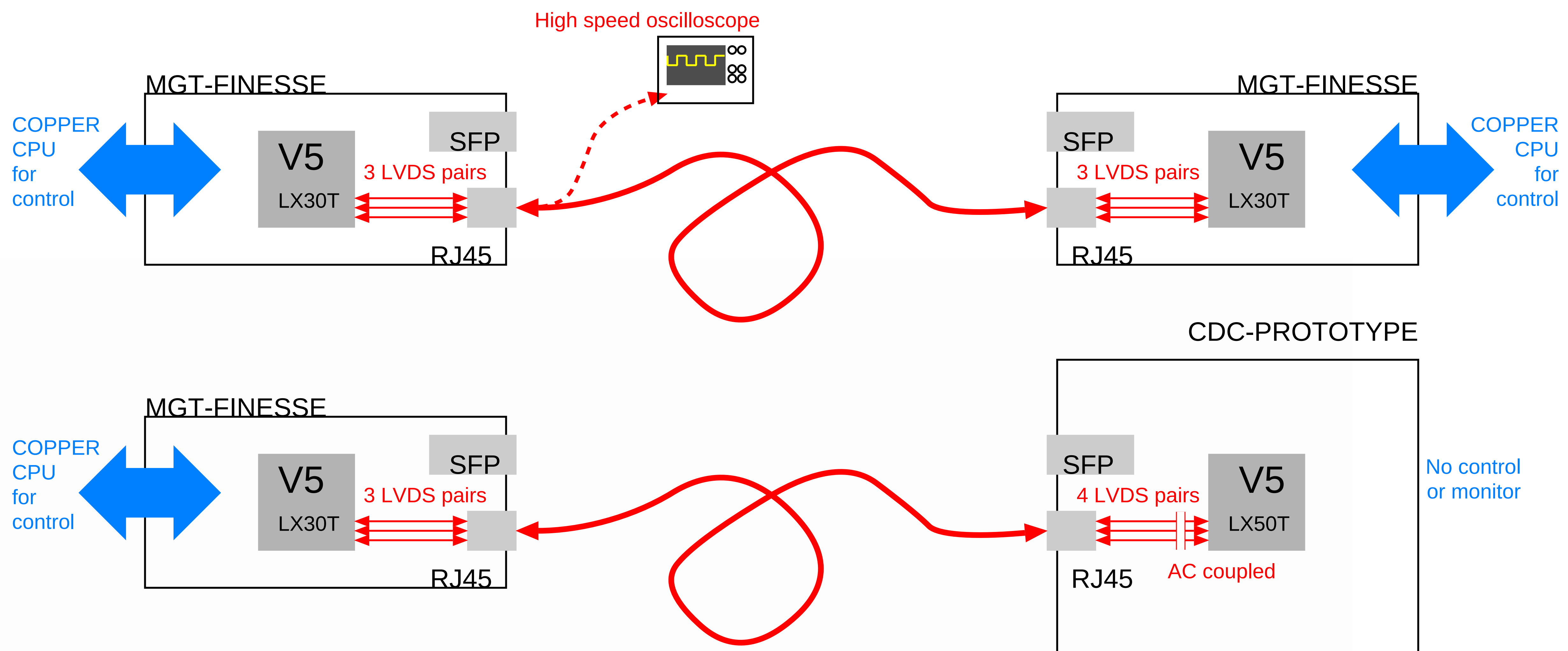


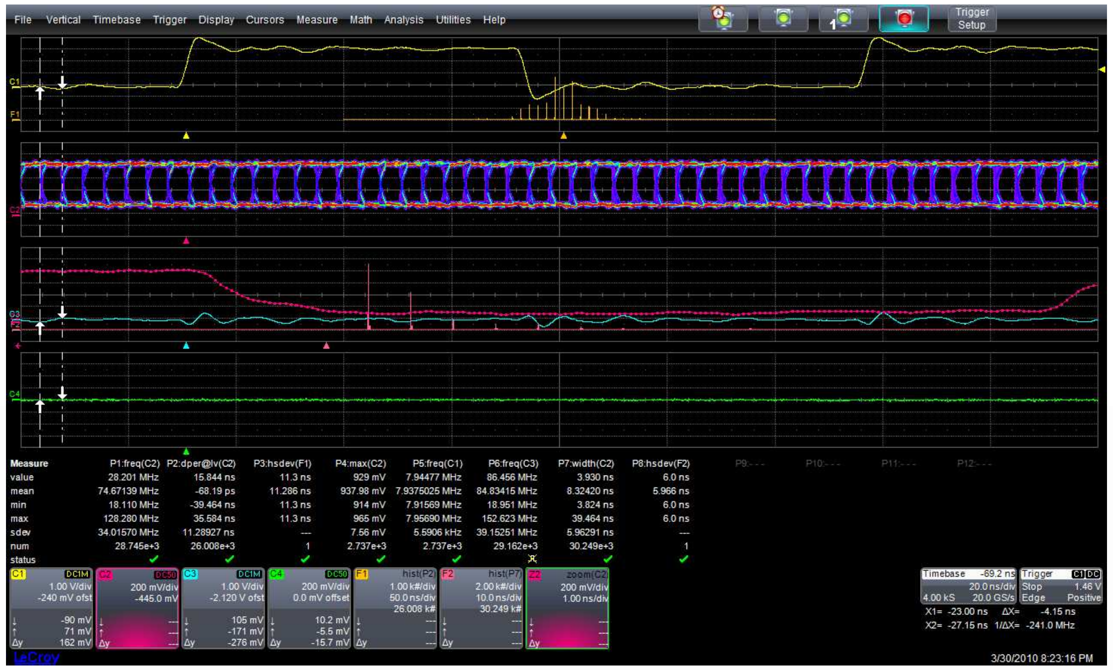
Figure 8-3: **Bit Ordering on Q1–Q6 Outputs of ISERDES_NODELAY Ports**

Test setup



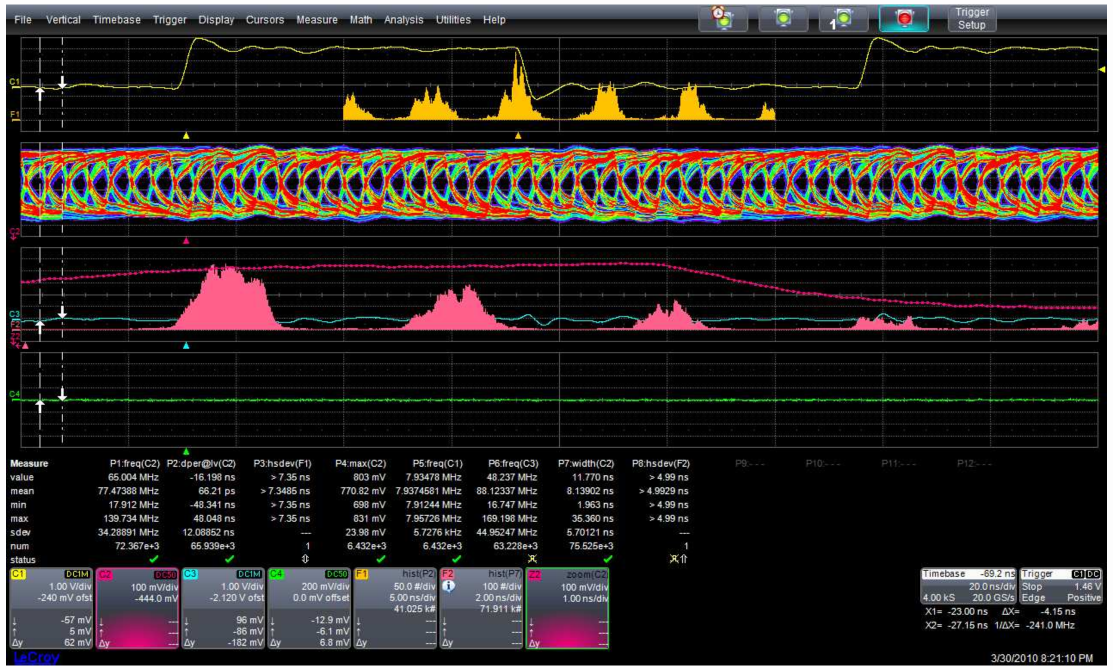
- Full control of a Virtex5 FINESSE card from CPU
- 2.5 Gbps oscilloscope with jitter measurement capability
- Single direction test and round-trip test

OSERDES output and cable attenuation



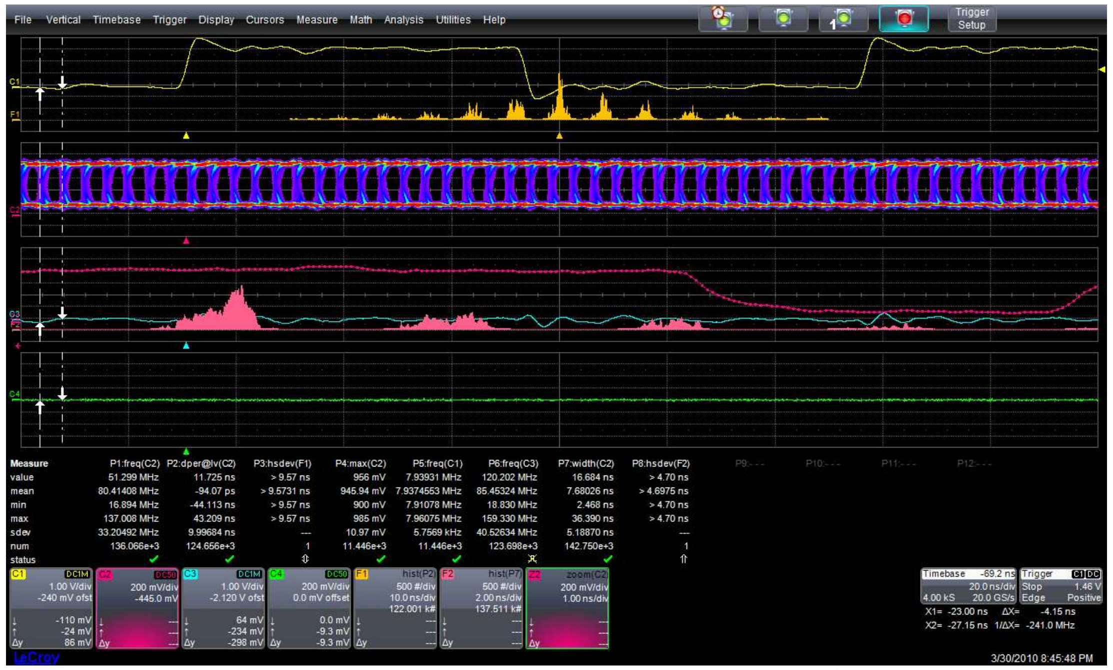
```
Measure  P1:freq(C2)    P2:dper@lv(C2)  P3:hsdev(F1)  P4:max(C2)  P5:freq(C1)    P6:freq(C3)     P7:width(C2)  P8:hsdev(F2)
value    28.201 MHz     15.844 ns       11.3 ns       929 mV      7.94477 MHz    86.456 MHz      3.930 ns      6.0 ns
mean     74.67139 MHz   -68.19 ps       11.286 ns     937.98 mV   7.9375025 MHz  84.83415 MHz    8.32420 ns    5.966 ns
min      18.110 MHz     -39.464 ns      11.3 ns       914 mV      7.91569 MHz    18.951 MHz      3.824 ns      6.0 ns
max      128.280 MHz    35.584 ns       11.3 ns       965 mV      7.95690 MHz    152.623 MHz     39.464 ns     6.0 ns
sdev     34.01570 MHz   11.28927 ns     ---           7.56 mV     5.5906 kHz     39.15251 MHz    5.96291 ns    ---
num      28.745e+3      26.008e+3       1             2.737e+3    2.737e+3       29.162e+3       30.249e+3     1
```

3/30/2010 8:23:16 PM



```
Measure  P1:freq(C2)    P2:dper@lv(C2)  P3:hsdev(F1)  P4:max(C2)  P5:freq(C1)    P6:freq(C3)     P7:width(C2)  P8:hsdev(F2)
value    65.004 MHz     -16.198 ns      > 7.35 ns     803 mV      7.93478 MHz    48.237 MHz      11.770 ns     > 4.99 ns
mean     77.47388 MHz   66.21 ps        > 7.3485 ns   770.82 mV   7.9374581 MHz  88.12337 MHz    8.13902 ns    > 4.9929 ns
min      17.912 MHz     -48.341 ns      > 7.35 ns     698 mV      7.91244 MHz    16.747 MHz      1.963 ns      > 4.99 ns
max      139.734 MHz    48.048 ns       > 7.35 ns     831 mV      7.95726 MHz    169.198 MHz     35.360 ns     > 4.99 ns
sdev     34.28891 MHz   12.08852 ns     ---           23.98 mV    5.7276 kHz     44.95247 MHz    5.70121 ns    ---
num      72.367e+3      65.939e+3       1             6.432e+3    6.432e+3       63.228e+3       75.525e+3     1
```

3/30/2010 8:21:10 PM



```
Measure  P1:freq(C2)    P2:dper@lv(C2)  P3:hsdev(F1)  P4:max(C2)  P5:freq(C1)    P6:freq(C3)     P7:width(C2)  P8:hsdev(F2)
value    51.299 MHz     11.725 ns       > 9.57 ns     956 mV      7.93931 MHz    120.202 MHz     16.684 ns     > 4.70 ns
mean     80.41408 MHz   -94.07 ps       > 9.5731 ns   945.94 mV   7.9374553 MHz  85.45324 MHz    7.68026 ns    > 4.6975 ns
min      16.894 MHz     -44.113 ns      > 9.57 ns     900 mV      7.91078 MHz    18.830 MHz      2.468 ns      > 4.70 ns
max      137.008 MHz    43.209 ns       > 9.57 ns     985 mV      7.96075 MHz    159.330 MHz     36.390 ns     > 4.70 ns
sdev     33.20492 MHz   9.99684 ns      ---           10.97 mV    5.7569 kHz     40.52634 MHz    5.18870 ns    ---
num      136.066e+3     124.656e+3      1             11.446e+3   11.446e+3      123.698e+3      142.750e+3    1
```

3/30/2010 8:45:48 PM

- 254Mbps signal (left-top) after 15m cable (left-bottom) after FPGA buffer (right-bottom) seems marginally OK
- Clock phase to be tuned

SERDES test

- Test with MGT-FINESSE $\rightarrow$ MGT-FINESSE
  - No automatic bit-rotation and quartet rotation yet, hand adjustment by seeing data
  - Every single bit transmission was checked
  - Round-trip test of O(10^4) random 32-bit pattern was successfully transmitted
  - Long-term test yet to be done
- Test with MGT-FINESSE $\rightarrow$ CDC-PROTOTYPE
  - AC-coupled, single-bit in 32-bit cannot be transmitted
  - Random data test: stopped at 2188th pattern with 12-bit sequence of high (expected)
  - Should be OK with proper encoding

Deadtime

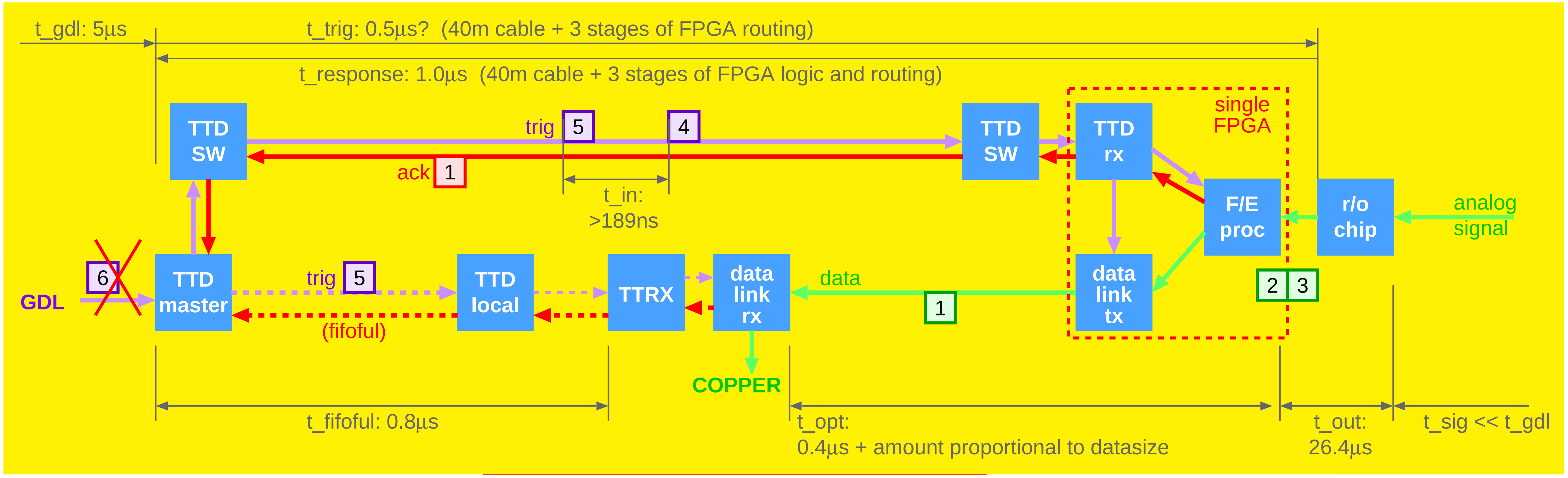
Deadtime and handshake

- Trigger rate: max 30 kHz
- Deadtime fraction ~ 3.4% at 30 kHz (already slightly violated)
- Key parameters
  - Depth of the ring buffer (N_{buf})
  - Minimum time interval between two triggers (t_{in})
  - Latency of processing the trigger (t_{out})

    Example (SVD), should be the worst case:

    f_{RCLK} = 32 MHz, N_{buf} = 5, t_{in} = 189 ns, (6clk) t_{out} = 26.4 μs (6×140clk)
- Main points
  - Processing multiple (=5) trigger at the same time
  - No back pressure at most of the steps (careful buffering)
  - Handshake at the very front-end for the conditions above
  - Handshake at the very back-end (of datalink) for buffers

Deadtime and handshake

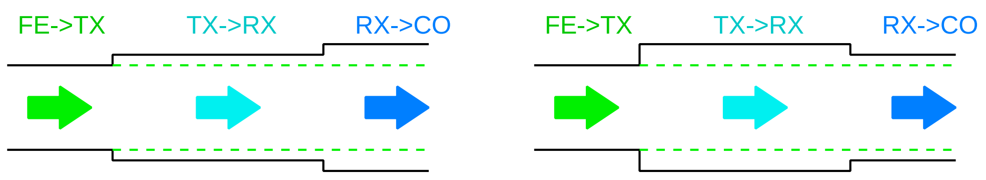


f_{RCLK} = 32 MHz, N_{buf} = 5, t_{in} = 189 ns, (6clk) t_{out} = 26.4 μs (6×140clk)

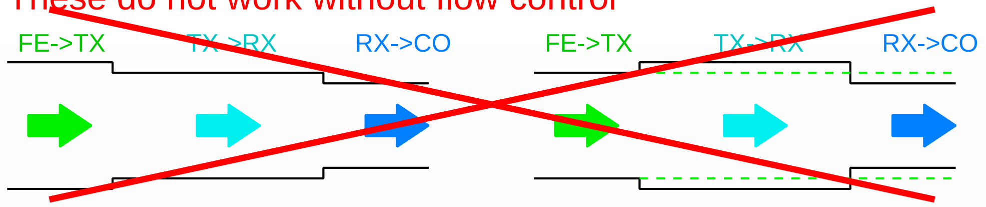
- Main points
  - Processing multiple (=5) trigger at the same time
  - No back pressure at most of the steps (careful buffering)
  - Handshake at the very front-end for the conditions above
  - Handshake at the very back-end (of datalink) for buffers

How to avoid buffer full in the datalink

These do not need flow control unless CO blocks



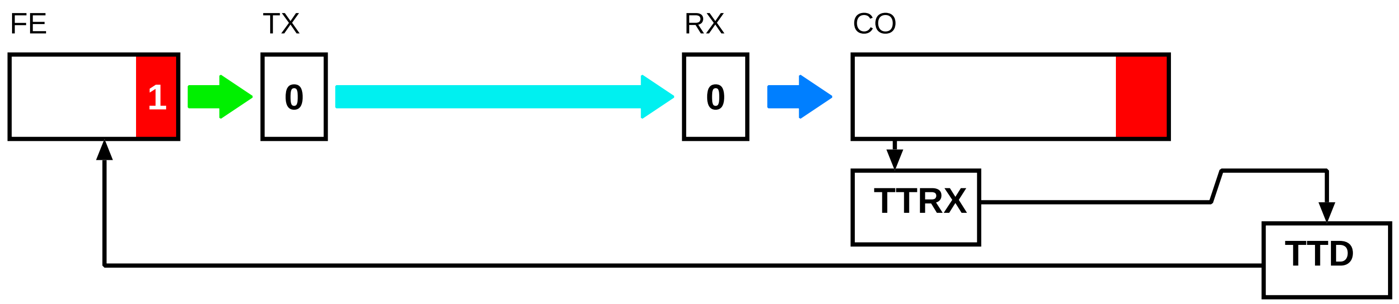
These do not work without flow control



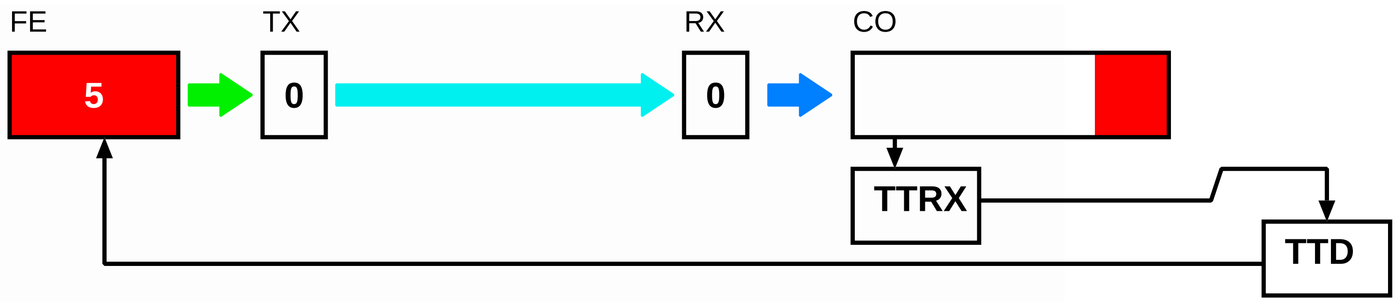
$$R_{FE} < R_{CO}$$

and $R_{FE} < R_{LK}$

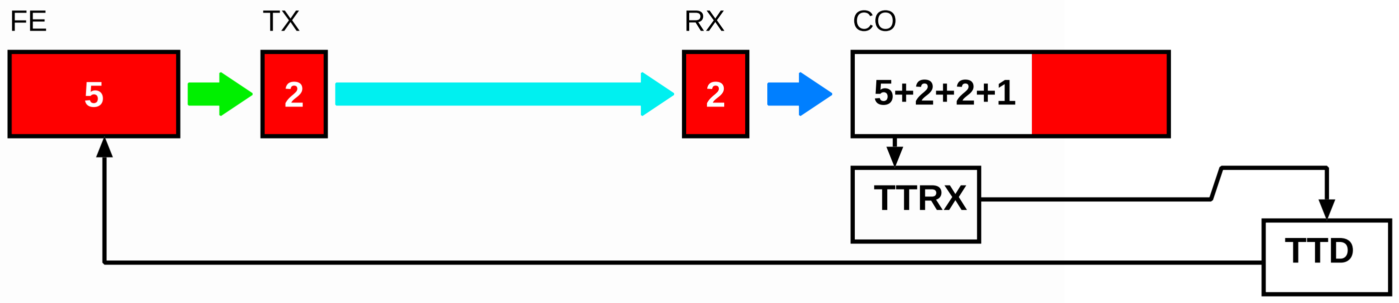
Normal case (1 event in FE, some events in CO, TX/RX are empty)



Trigger block at FE (5 events in FE, some events in CO, TX/RX are empty)



Trigger block at CO (5 events in FE, 5+2+2+1 events free in CO, TX/RX are full)



- COPPER: 42 MHz × 32bit ⇒ Frontend: 31 MHz × 32bit?
- RocketIO can control the data bandwidth by inserting IDLE words (up to 3.125 Gbps)
- Or $R_{LK} = R_{FE}$ with RocketIO driven by 127 MHz (RF clock/4)

Deadtimeless Trigger Flow Control

- How to prevent the 6th trigger to be issued, when 5 triggers are already being processed?
  - TTD counts number of issued trigger
  - TTD has to count number of processed trigger, too
  - Therefore, frontend (timing) firmware needs to receive a **response signal** when processing the trigger is finished.
- When the 6th trigger will be issued?
  - When all response signals for the 1st trigger are collected
  - It should happen in principle after $\sim$30 μs after 1st trigger
- COPPER FIFO
  - Always space for 5+alpha events needed, otherwise busy to TTD (alpha: discussed later)
  - No busy response is needed (no trigger is needed?)

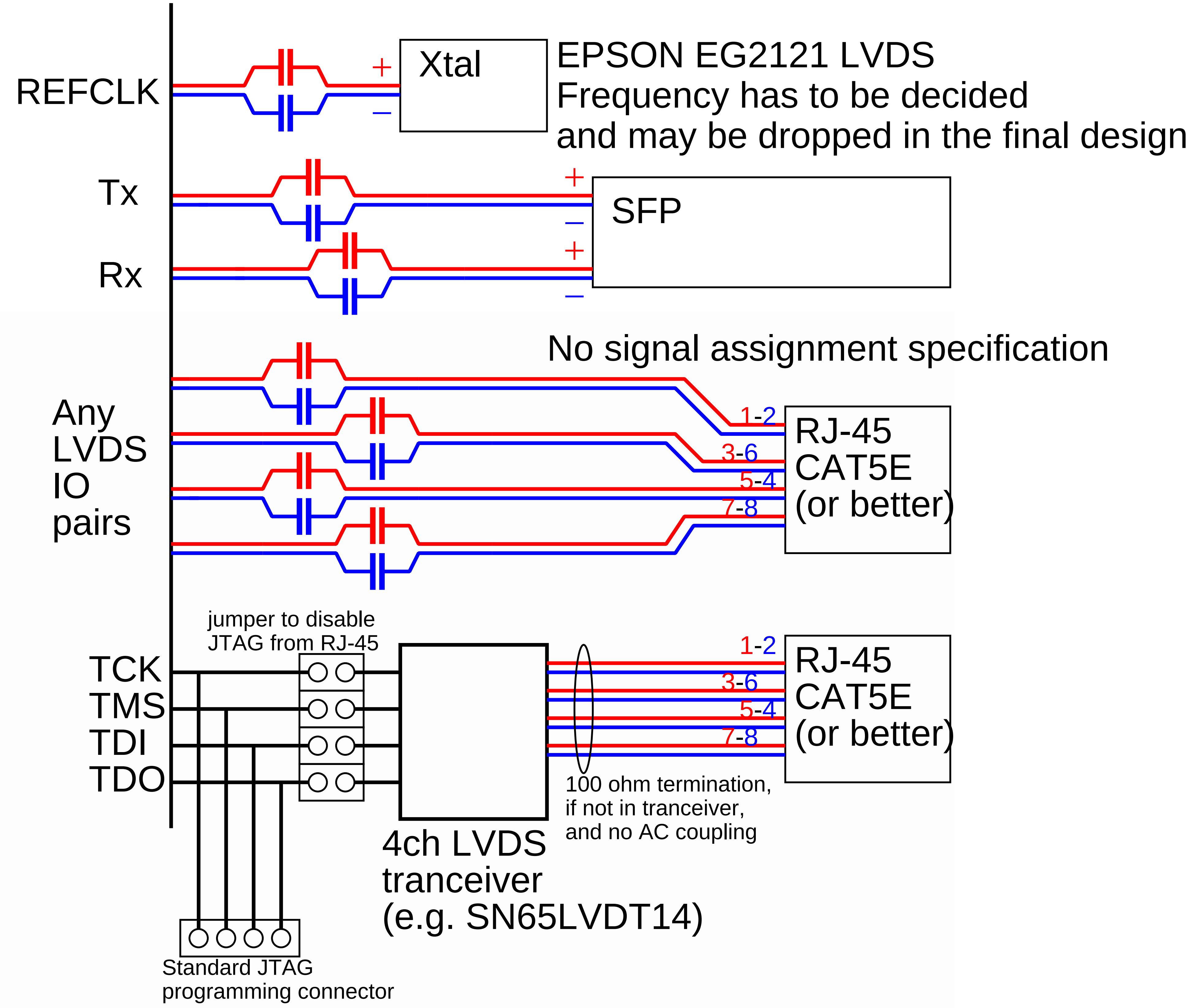
Interface

dummy horizontal text dummy horizontal text dummy h
text dummy horizontal text dummy horizontal text dum
zontal text

Hardware requirements in the frontend (proposal)

20100407 version

Virtex5 LXT
or
Spartan6 LXT



Summary of Timing Distribution interface

In addtion to the RocketIO interface (SFP slot, Xtal) . . .

- RJ-45 connector 1
  - 1–2 pair (1: CLK+, 2: CLK–)
  - 3–6 pair (3: SIN+, 6: SIN–)
  - 5–4 pair (5: RSV+, 4: RSV–)
  - 7–8 pair (7: SOUT+, 8: SOUT–)
  - All lines are AC coupled (0.1 μF or 1 μF)
    (if DC coupling is needed in future, replace it with 0Ω register)
- FPGA resources
  - one PLL, (and maybe one DCM)
  - IODELAY, IDELAYCTRL
  - (ISERDES, OSERDES are automatically available at the IOPAD)

TTD signals inside FPGA

- Signals to the frontend readout logic
  - Clock1,2 two out of 127, 101, 63, 42, 32, 2 MHz
    clocks are derived from 127 MHz → 508 MHz div by n
  - REVO 100 kHz, 63 ns long pulse
  - Trigger timing synchronized to 127 MHz clock, 63ns wide
  - Trigger id and type available at the clock timing
  - RESET to start a run, synchronized to REVO signal, 1 REVO cycle long
  - Any other signal?
- Signals from the frontend logic
  - Response signal for completion of processing for every trigger
  - Trigger id for the last processed trigger
  - READY signal kept HIGH during the run
    asserted to LOW for a short time
    asserting to LOW during the run in case of fatal error

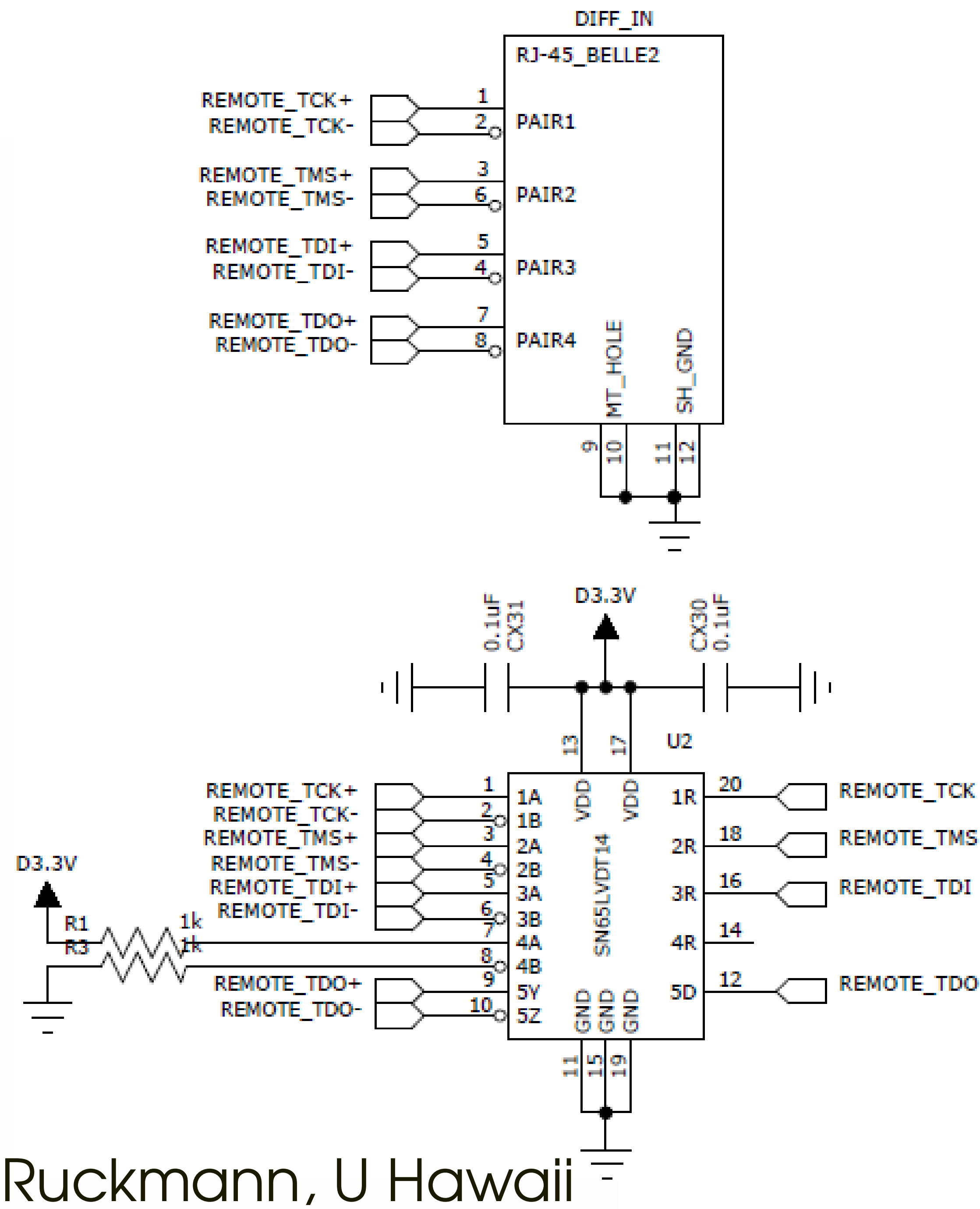
Initialization of the front-end board

- Initialization through the TTD serial link
- RESET signal generated for a fixed period
  - One REVO cycle long, synchronized to the REVO signal
  - Wait for READY signal goes (H $\rightarrow$) L $\rightarrow$ H
  - No trigger will be issued for one more REVO cycle
- Issued to start a run, but also issued just to check the status
- Time stamp counter and trigger-id counter are reset to 0
  - Time stamp is incremented at every serial-link clock (64-bit)
  - Trigger-idis incremented and checked at every trigger (32-bit)
  - Both are attached through datalink, to uniquely identify the event fragment and to test data synchronization
- Is there any request for initialization that takes $>10\ \mu s$?
  (besides downloading FPGA via JTAG and constants via datalink)

Remote JTAG programming

After Radiation test by Gary, we propose to avoid using configuration memories (and CPLDs) on the boards in the radiation harsh area.

- FPGA has to be programmed remotely
- PIN assignment (+ for odd / – for even pins)
  1–2 pair: TCK (input)
  3–6 pair: TMS (input)
  5–4 pair: TDI (input)
  7–8 pair: TDO (output)
- Both directions in one chip (TI SN65LVDT14)
- The same FTSW module to distribute the JTAG signals (DC coupled version)



L. Ruckmann, U Hawaii

Data from TTD at Belle2

Recently I was asked how to access event related information in some data. Now I think following data should be stored in MDST at Belle2 (about 16 words or so). Some of the information may not be necessarity related to TTD, but saving in the TTD datastream guarantees that the information is saved.

- Experiment number, Run number, Event number (= L1 TAG)
- Event time in system clock (64-bit), event time in UNIX time
- Bunch-id (0..5119)
- Time since last HER injection, time since last LER injection
- Time since last trigger inhibit, up to two sources (current limitter, HV trip)
- Beam currents (LER, HER), instantaneous luminosity
- Trigger type if available (i.e., timing source)

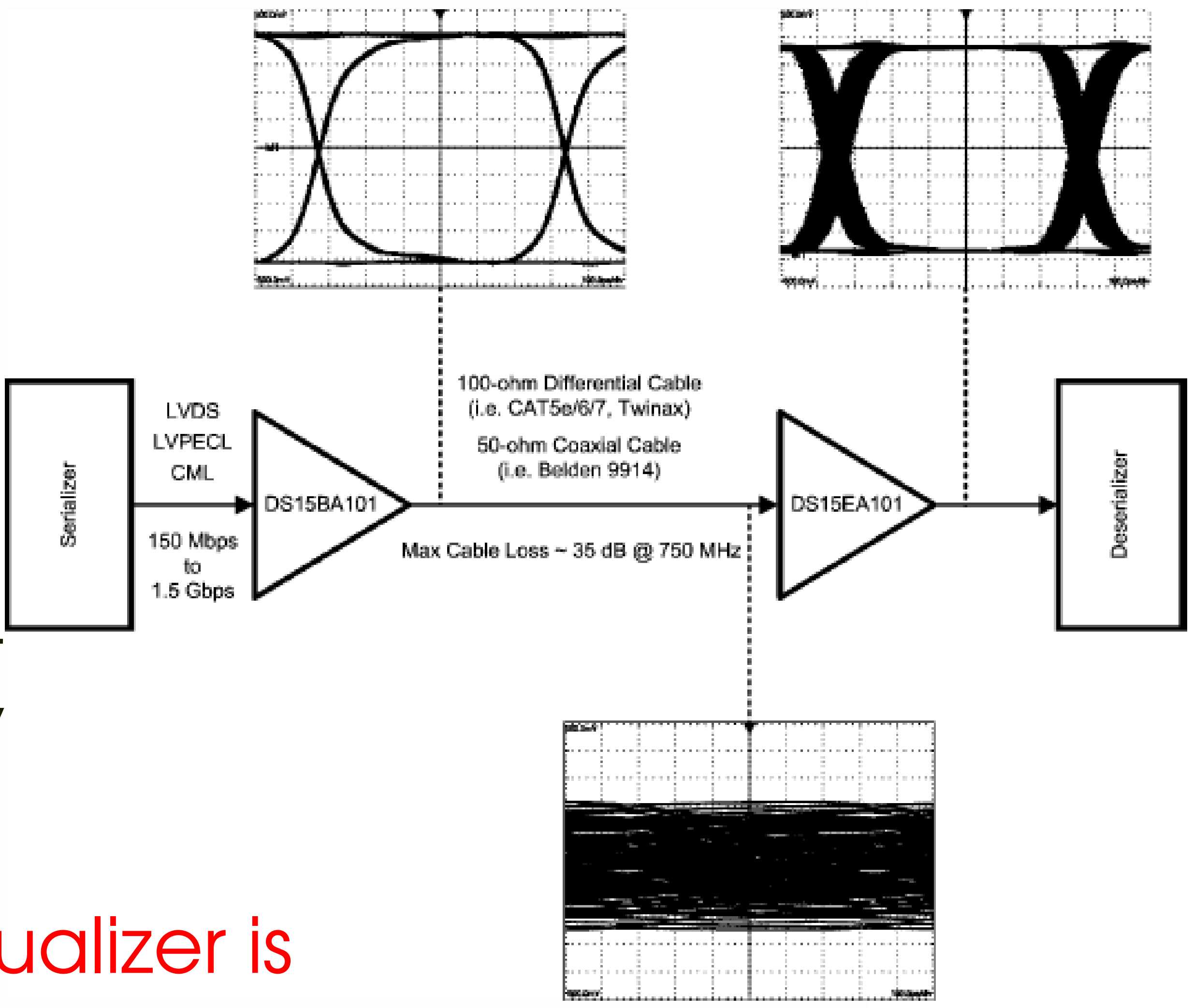
Time is in system clock (127 MHz) unit, for 32-bit, unless otherwise specified. This list is kept in Belle2 Wiki.

Summary

- Most if not all current TTD considerations are discussed (even w.r.t. TDRv1.0)
  - To be converted into a technical note
- Short-term plans
  - Radiation test preparation (FPGA, RocketIO, TTD and JTAG) Power and JTAG interface were added to the MGT-FINESSE board
  - Many small tests
  - FTSW board design — part selection mostly done, need to decide whether Spartan6 or Virtex5 and logic size

Equalizer?

- Insertion of a driver and receiver chip to "equalize" the frequency response will allow for a better and stable SERDES. (this functionality is in RocketIO, but not in ISERDES/OSERDES).
- This introduces a change to what I previously said to subdetector readout boards
- This will fix the direction of the LVDS pair
- Several chips are available in the market, survey needed



Serializer
LVDS
LVPECL
CML
150 Mbps to 1.5 Gbps
DS15BA101
100-ohm Differential Cable (i.e. CAT5e/6/7, Twinax)
50-ohm Coaxial Cable (i.e. Belden 9914)
Max Cable Loss ~ 35 dB @ 750 MHz
DS15EA101
Deserializer

Considering the drawbacks, equalizer is not adopted.

Danke schön!