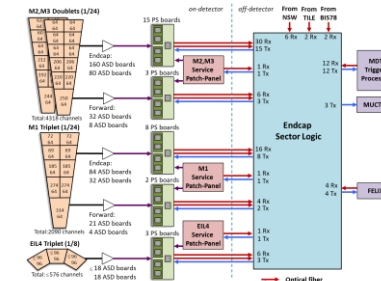


# Building an FPGA Design to Meet Timing Challenges in HEP Experiments

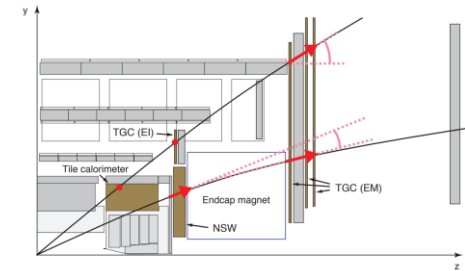
Lessons and techniques learned from experience with ATLAS L0 Muon

# Intro

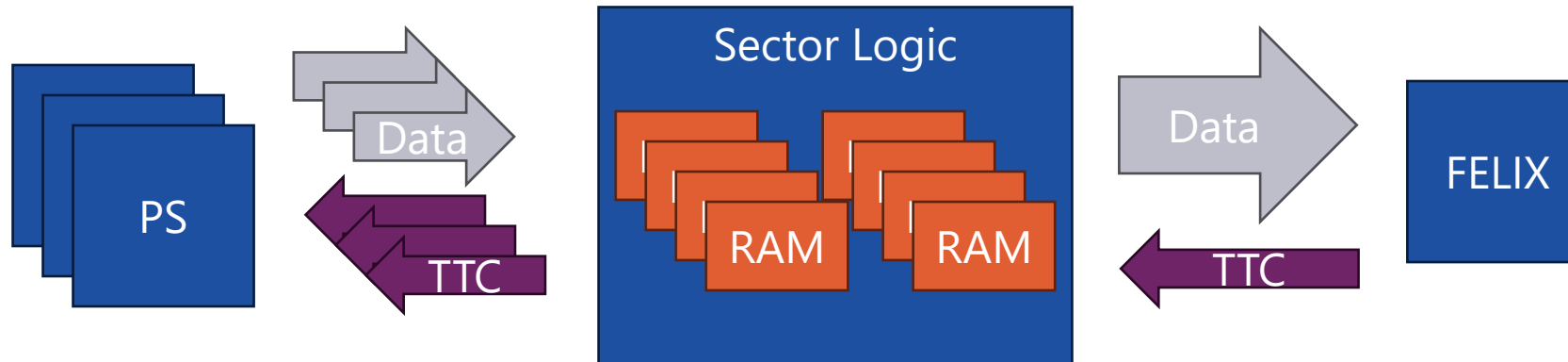
- ATLAS Muon TDAQ electronics get major upgrade in the upcoming Phase-II
- Brand new **Endcap Sector Logic (SL)** board for “Level-0” trigger system
  - Off-detector
  - Gather data mainly from the Big Wheel TGC
  - Basic track reconstruction and data processing for trigger decision
- TL;DR version for this talk:
  - Uplink: gather, process, and send out data
    - The “process” needs many memory devices: LUTRAM, Block RAM (BRAM), Ultra RAM (URAM)
  - Downlink: receive and distribute the **LHC Bunch Clock (BC)** and **Trigger and Timing Control (TTC)** signals
  - **FELIX** in the uplink, **PS** in the downlink direction



<https://cds.cern.ch/record/2285580>



<https://cds.cern.ch/record/2285584>





# Taming Timing (Violation)

Avoiding common mistakes in FPGA design

# Timing Violation

- Signal arrives at the destination Filp-Flop(FF)/register outside the allowed window
- Design with timing violations can fail in hardware
  - A ticking time bomb even if it works
- Vivado's timing summary report is not enough. Violations can be hidden by:
  - **Timing exceptions** such as set\_false\_path or set\_clock\_groups
  - **Unsafe timed paths** between clocks that have no defined relationship
    - Appear as CRITICAL WARNING in Methodology and red box in Vivado's clock interaction report

## Design Timing Summary

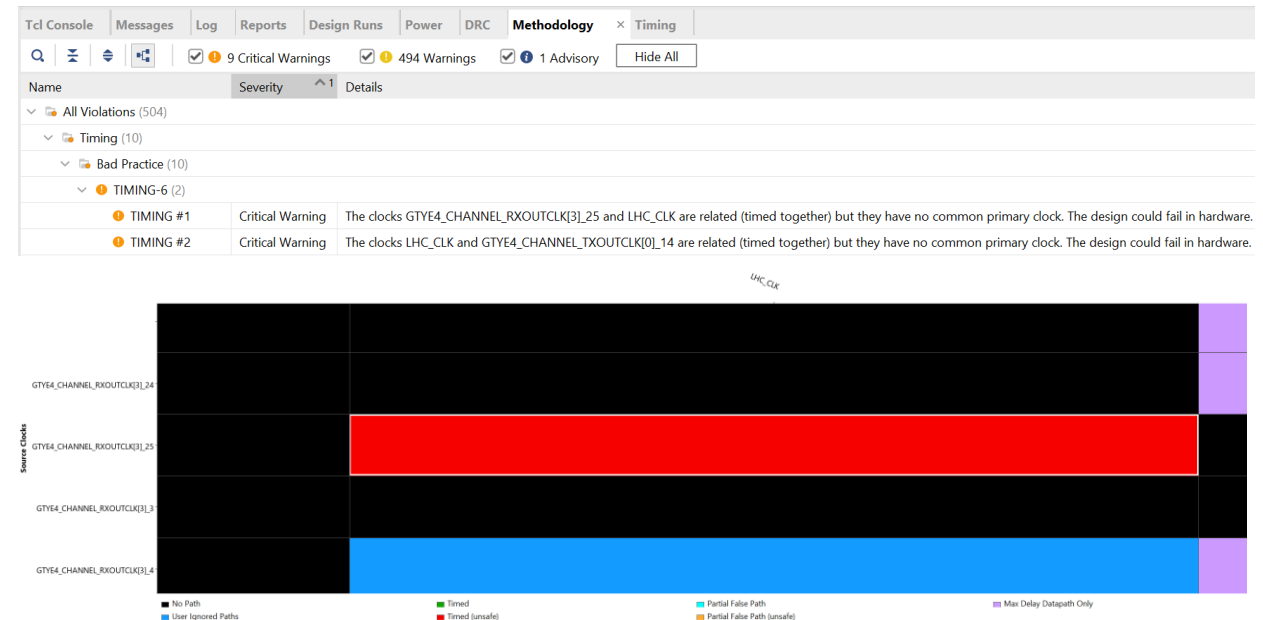
### Setup

Worst Negative Slack (WNS): **0.013 ns**  
Total Negative Slack (TNS): **0.000 ns**  
Number of Failing Endpoints: **0**  
Total Number of Endpoints: **1837395**

**Timing constraints are not met.**

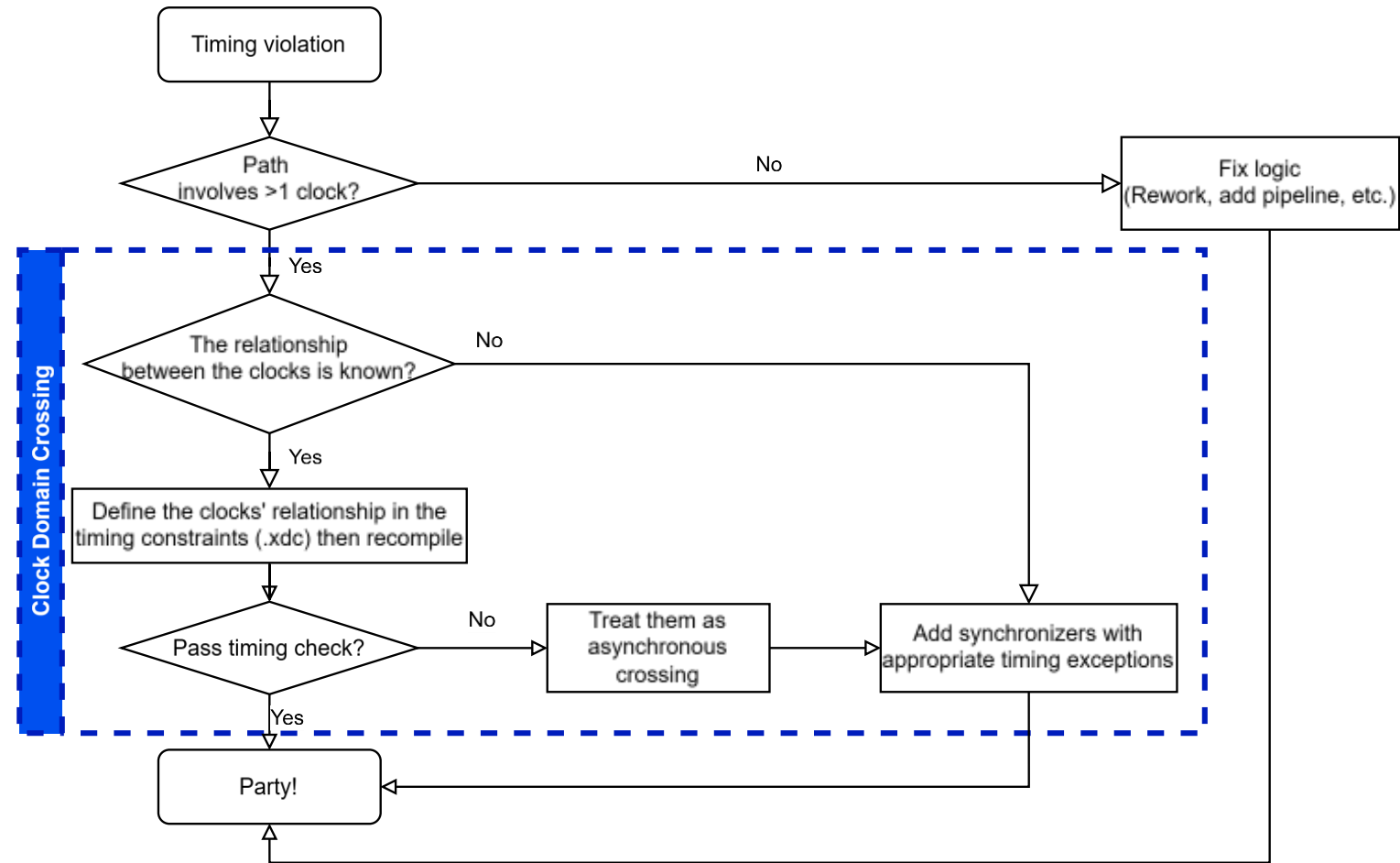
### Hold

Worst Hold Slack (WHS): **-0.001 ns**  
Total Hold Slack (THS): **-0.001 ns**  
Number of Failing Endpoints: **1**  
Total Number of Endpoints: **1826116**



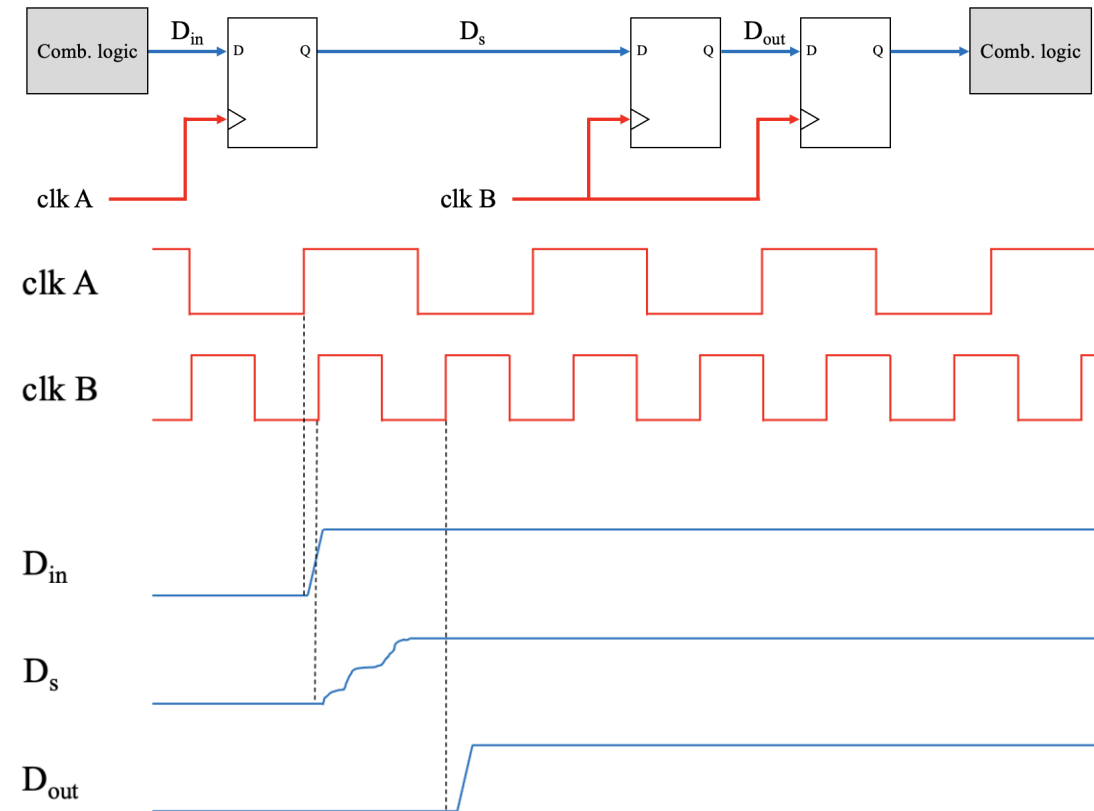
# CDC Path Violation

- Violations in paths between two clocks usually means the “Clock Domain Crossing (CDC)” is not properly handled
- If known, put clock relationship in the timing constraints (.xdc) file
  - “Synchronous CDC”
  - No extra headaches (details upcoming)
- If not (or violation persist), it needs an appropriate “Clock Domain Crossing Circuit (CDCC)”



# CDC Circuit

- 2-FF (or more) bit synchronizer
  - Two closely placed FFs acts as a [metastability guard](#)
  - "Cut" the CDC path ( $D_s$ ) with [timing exception](#)
- Important points to remember:
  - Timing exceptions (e.g., `set_false_paths`)...
    - Tell the tool to ignore the paths
    - Do not solve timing violations
  - The 2-FF must be fed from a FF in the clkA domain
  - Depending on the clocks' relationship...
    - Some transitions may be missed
    - [The latency can change](#) during runtime
      - 1-3 cycles for 2-FF, [Explanation](#)
  - Does not work for multi-bit (data bus)
    - An array of 2-FF synchronizers is not a CDCC for data bus

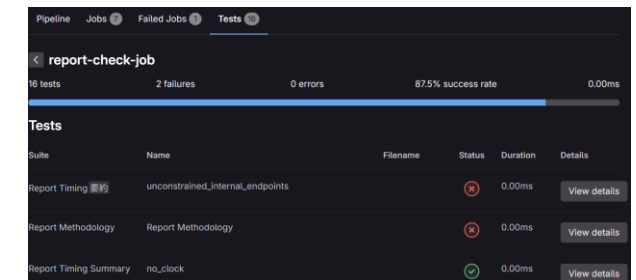


# CDCC Best Practices

- **Never** use clock-to-clock exception
  - set\_clock\_groups -asynchronous includes
  - Accidentally cut other paths
  - Instead, specifies the cells (reg-to-reg rule)
- Avoid reinventing the wheel. Use:
  - Vendor-provided packages (e.g., Xilinx's [XPM](#))
  - Well-known open-source packages
- Use "scoping XDC" with custom-made CDCC
- Consider adding an impossible maximum delay for all clock pairs
  - E.g., set\_max\_delay -from clkA -to clkB 0.0
  - Catch any unhandled CDC paths
- Regularly check Methodology and Clock Interaction report (Vivado)
  - We automated this in GitLab-CI

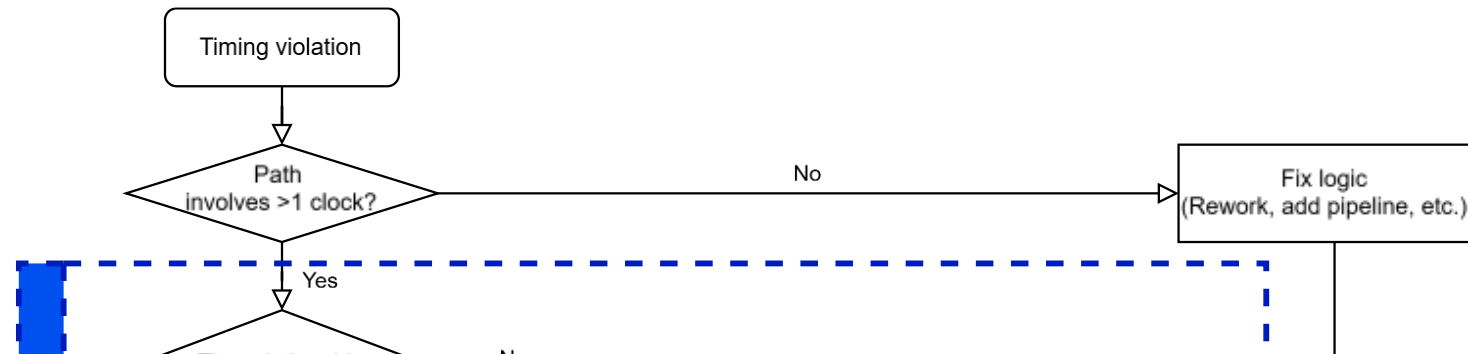
## Xilinx Parameterized Macros

- [XPM\\_CDC\\_ARRAY\\_SINGLE](#)
- [XPM\\_CDC\\_ASYNC\\_RST](#)
- [XPM\\_CDC\\_GRAY](#)
- [XPM\\_CDC\\_HANDSHAKE](#)
- [XPM\\_CDC\\_PULSE](#)
- [XPM\\_CDC\\_SINGLE](#)
- [XPM\\_CDC\\_SYNC\\_RST](#)
- [XPM\\_FIFO\\_ASYNC](#)

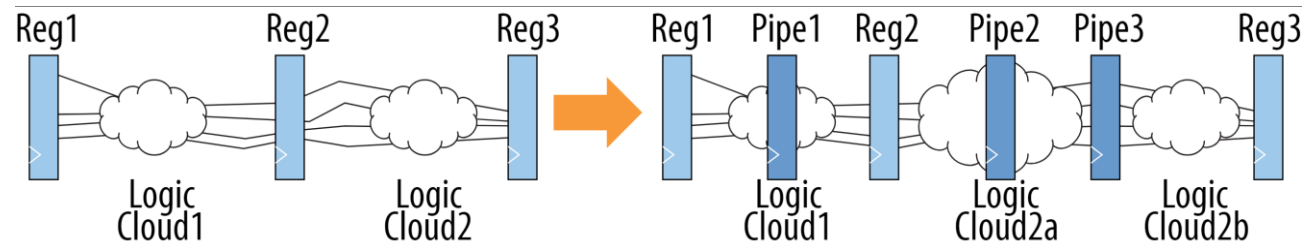


Suite	Name	Filename	Status	Duration	Details
Report Timing	unconstrained_internal_endpoints		✗	0.00ms	<a href="#">View details</a>
Report Methodology	Report Methodology		✗	0.00ms	<a href="#">View details</a>
Report Timing Summary	no_clock		✓	0.00ms	<a href="#">View details</a>

# Non-CDC Path Violation



- Non-CDC path violations imply design problems and/or lack of resources
- Pipelining—adding register stages to ease timing—is a very common fix



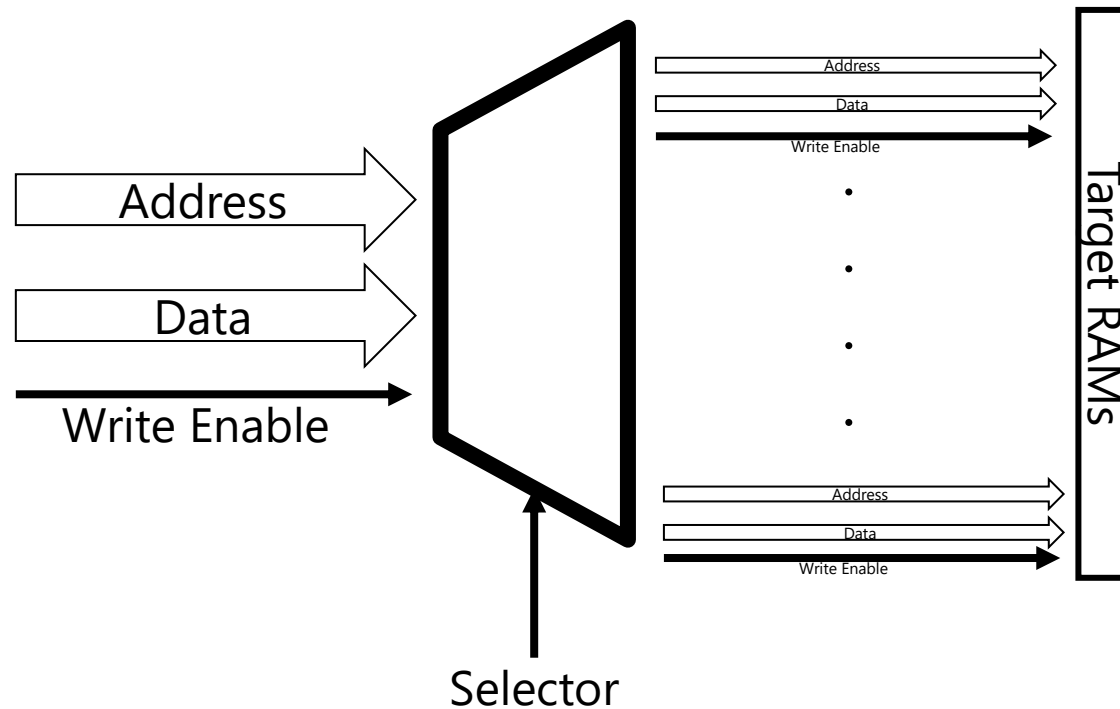
[2.3.1. Conventional Versus Hyper-Pipelining \(intel.com\)](#)

- However, pipelining only fixes a specific set of problems
  - Large logic cloud
  - Long physical distance
- No all-purpose solution



# RAMController—A Case Study

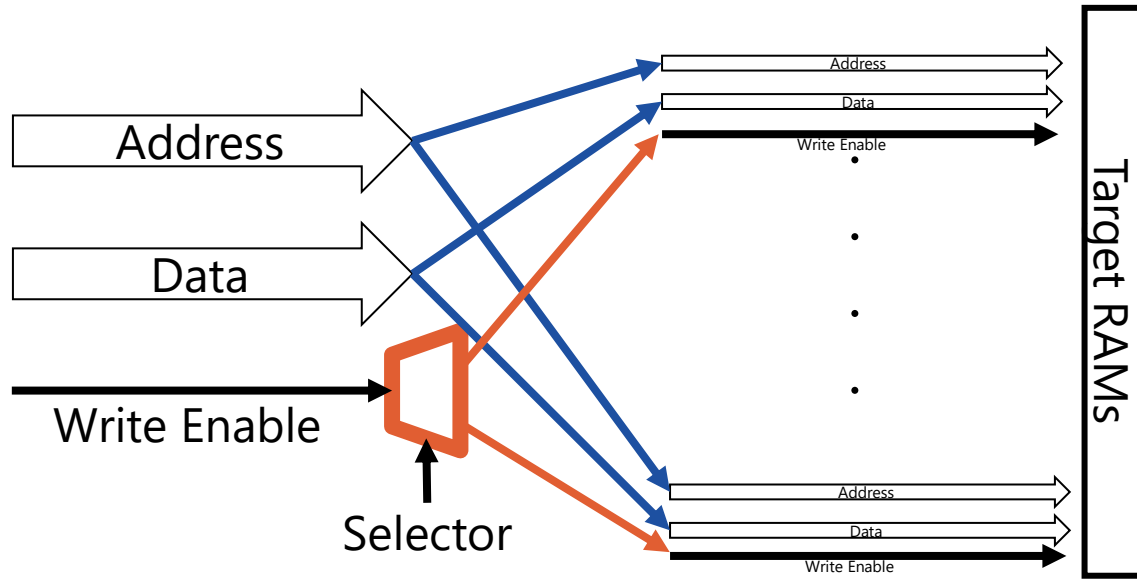
- Take write command and direct it to the target RAMs
- Critical paths have almost no combinational logic



- The Address, Data, and WE are directed to the intended RAM device
- This is a very large demultiplexer (DEMUX) that must be built with a huge network of LUT primitives

# Reduced Demultiplexer

- But the demux is only needed for WE



- Address and data are duplicated
- Only write enable signal is demultiplex'ed
- Massive reduction of the resource usage

RAMController

## Primitive Statistics

Primitive type	Count
CLB/CARRY	2
CLB/LUT	30960
CLB/MUXF	1
REGISTER/SDR	345

## Net Boundary Statistics

Boundary-crossing Nets
21350

## Clock Report

Domain (Module)	Resource	Instances
LHC_CLK_out( SLR1 )	Global	345

RAMController

## Primitive Statistics

Primitive type	Count
CLB/LUT	981
REGISTER/SDR	173

## Net Boundary Statistics

Boundary-crossing Nets
900

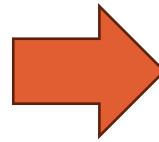
## Clock Report

Domain (Module)	Resource	Instances
LHC_CLK_out( SLR1 )	Global	173

# Data and Control

- An example of mixing **data** and **control** part
- Another common mistake is resetting the entire **data bus** instead of just the **control signal**
- For example, consider the code below:

```
always @ ( posedge CLK_240M ) begin
    if( reset ) begin
        output_valid <= 1'b0;
        reg_TGCdata_in_BC0 <= 16'b0;
    end
    else begin
        if( |reg_TGCdata_in_BC0 ) begin
            output_valid <= 1'b1;
            reg_TGCdata_out <= reg_TGCdata_in_BC0;
        end
    end
end
```



```
always @ ( posedge CLK_240M ) begin
    reg_TGCdata_out <= reg_TGCdata_in_BC0;
    if( reset ) begin
        output_valid <= 1'b0;
    end
    else begin
        output_valid <= 1'b1;
    end
end
```

- **output\_valid** tell the validity of the data. **reg\_TGCdata\_out** need no reset
- Goal: avoid operations on **data** and use a few **control** bits instead (e.g., WE)

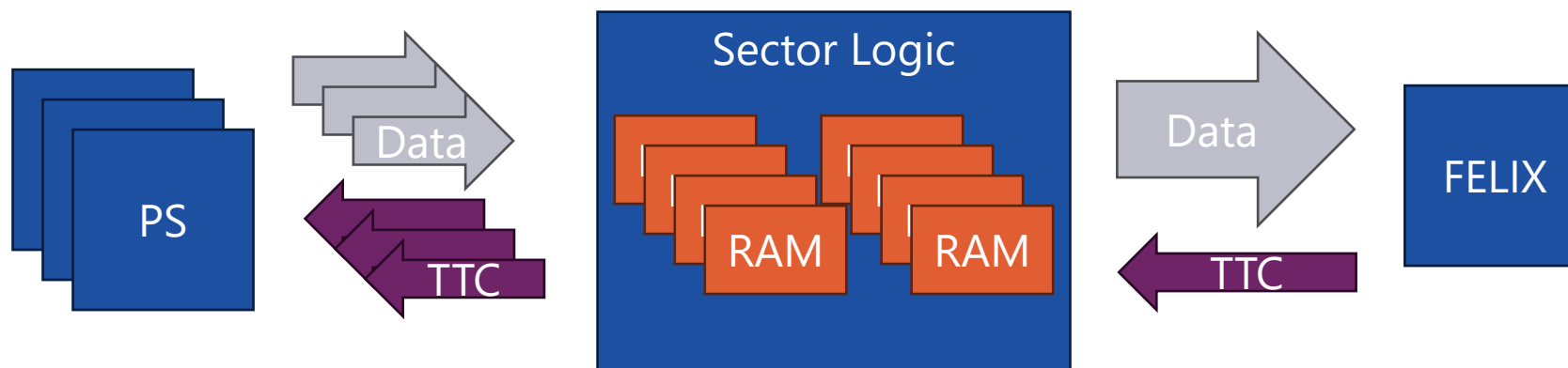
# Locking Latency

Building latency-fixed data links



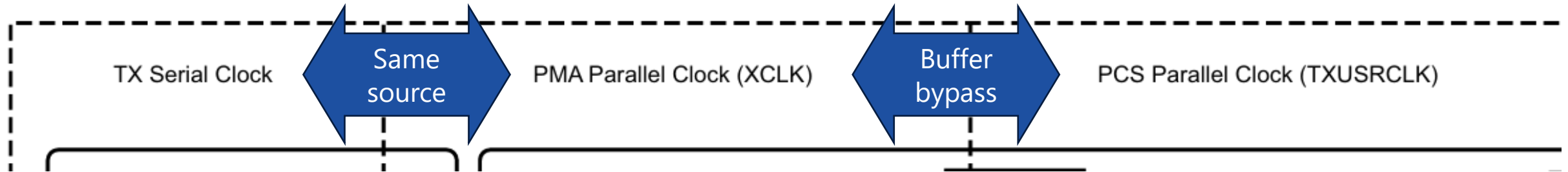
# The Latency Problem

- CDC needs synchronizer. Synchronizer adds latency variation
- What if that is not acceptable?
- Latency-fixed data link is a common requirement for HEP applications
- For SL, Trigger and Timing Control (TTC) signals need one



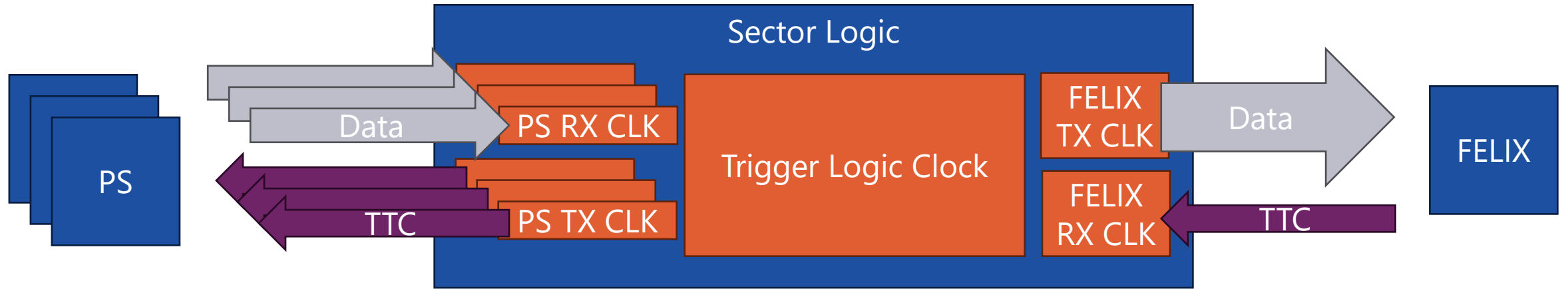
# MGT Clocks and Latency

- Multi-Gigabit Transceiver (MGT) needs multiple clock domains
- MGT interface with the rest of the FPGA via the **user clock (USRCLK)**
- USRCLK and MGT's transmission clocks are different
- We typically use "buffer bypass" mode for this internal CDC
  - Complicated procedures, but for us it is just an option to pick from



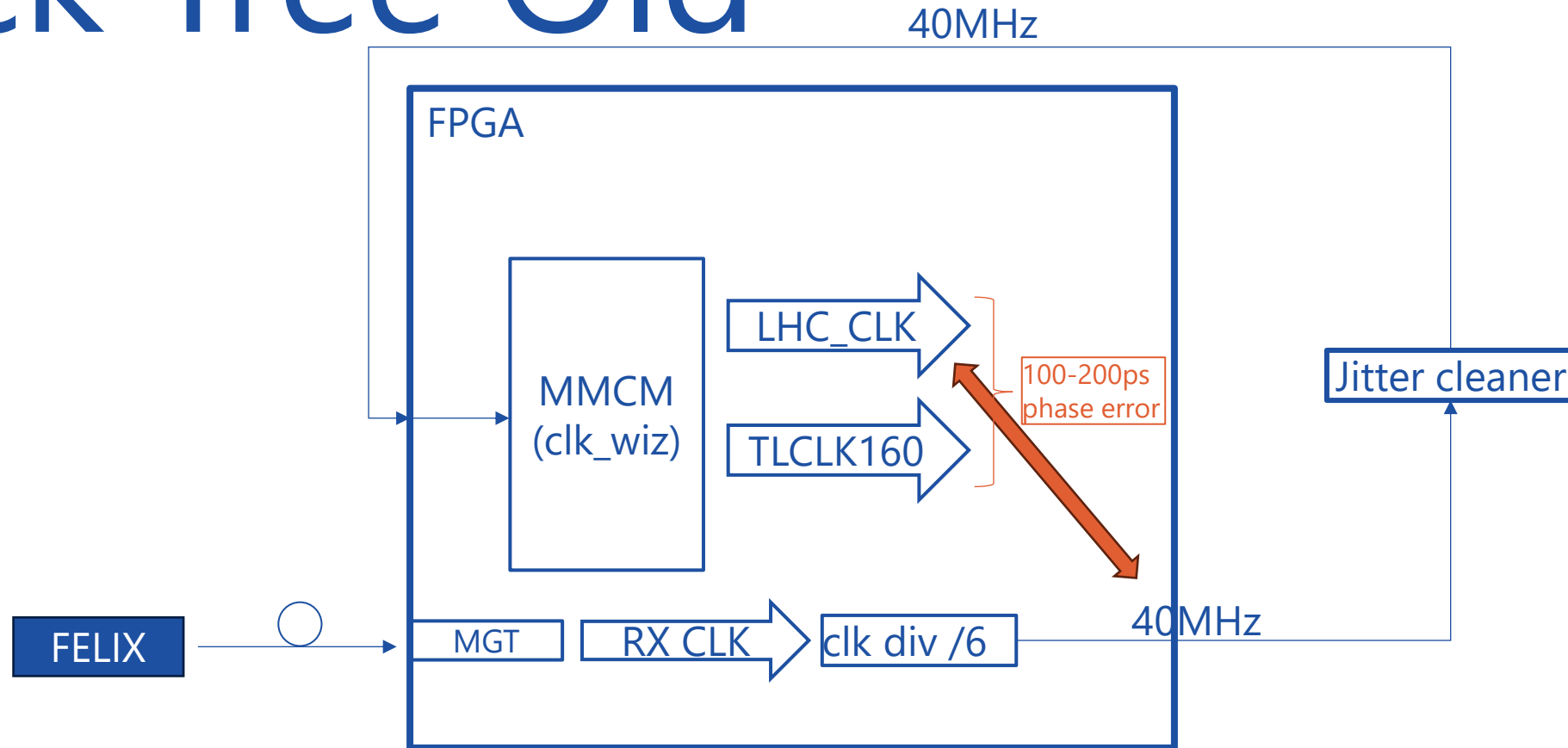
- Latency is fixed, if we do not need another CDC
- Spoiler alert: we do

# SL Clocks Domains



- FELIX RX CLK is the “master” clock
- PS TX CLKs send out TTC
- PS RX CLKs receive hit data
- FELIX TX CLK send out data
- Trigger Logic CLock @ 160MHz (TLCLK160) interacts with all the above
- Data are “timestamped”, so uplink can self-align and will be omitted from this discussion
  - A lot of work went into this but mostly unrelated to the upcoming discussion

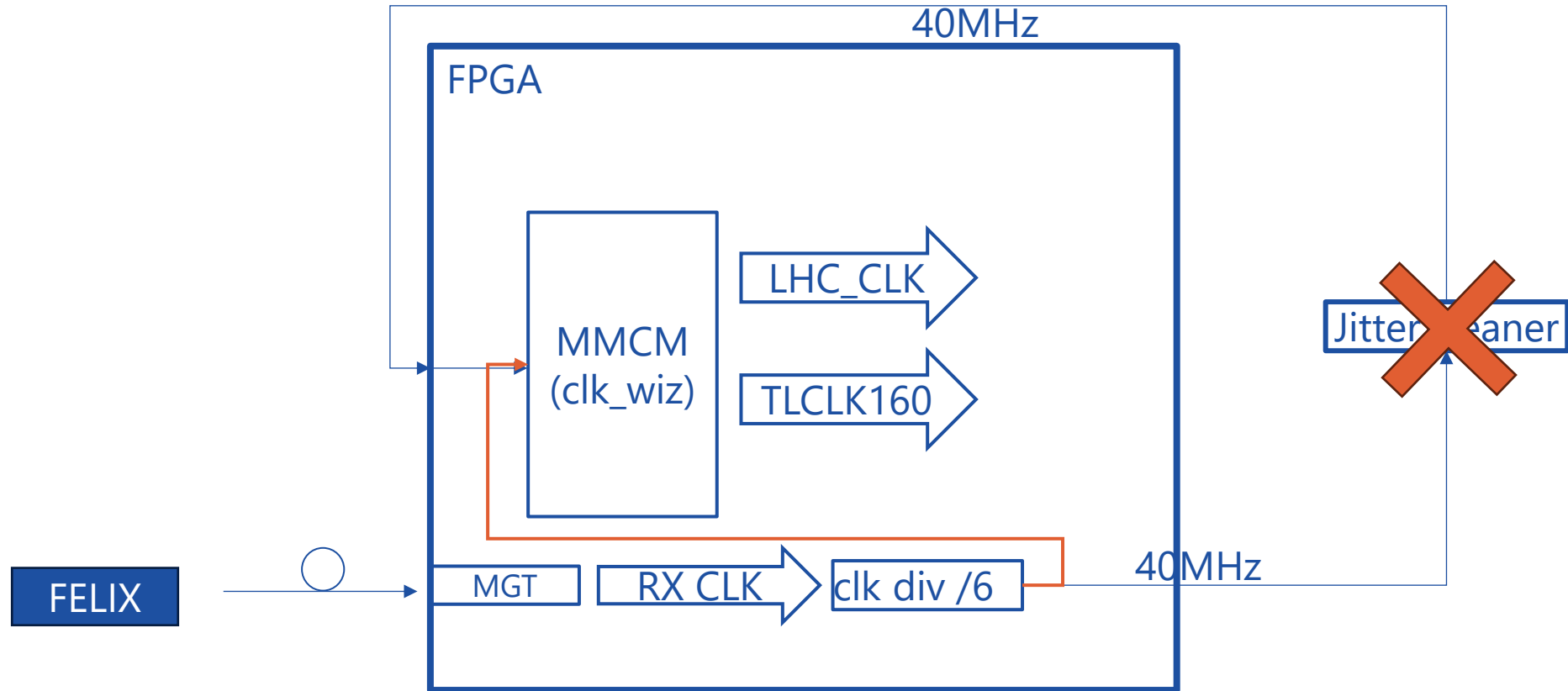
# Clock Tree Old



- 240MHz RX CLK → 40MHz → external jitter cleaner → MMCM
- MMCM generates 160MHz trigger clock (TLCLK160) and 40MHz BC (LHC\_CLK)
  - ~200ps phase error between outputs, which contributed to timing violations
- **Phase** between the FELIX's 40MHz and LHC\_CLK is **unknown, but fixed**

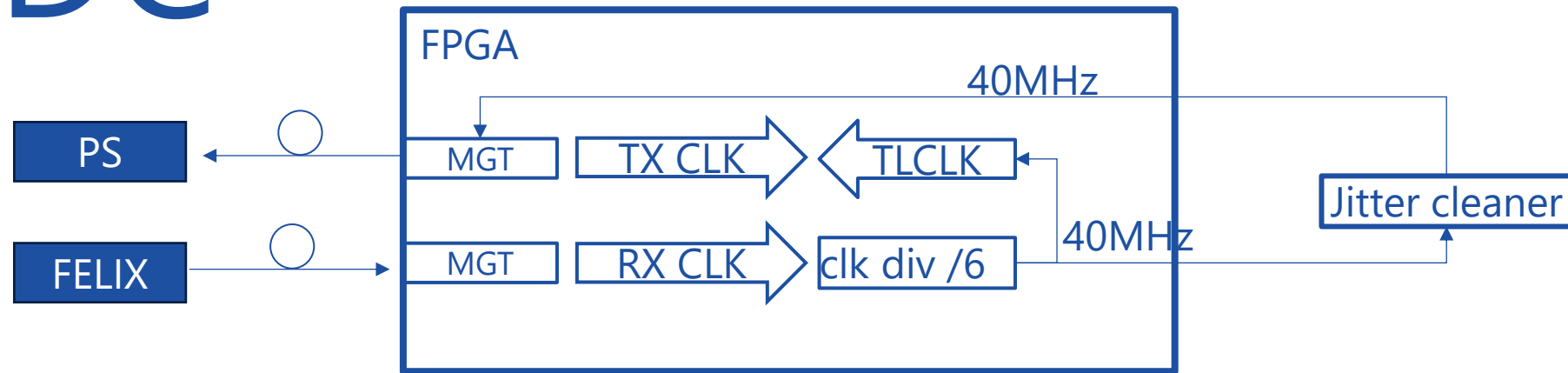


# Clock Tree New



- More design optimization to compensate for MMCM phase error
- Directly feed FELIX's 40MHz to the MMCM
- FELIX RX CLK → TLCLK160/LHC\_CLK becomes synchronous CDC
  - External routing delay was why the **phase** was **unknown**

# TX CDC



- One CDC left: TLCLK → TX CLK
  - MGT needs jitter cleaner
- Relative phase is **unknown, but it is fixed** (assuming the jitter cleaner works properly)
- We develop an “auto-calibration” approach
  - A short auto-calibration period following each reset to stabilize the latency at cycle-level
  - Our frontend ASIC can compensate for sub-cycle variation
- For more stringent requirements, check out CERN’s TCLink
  - O(ps) phase stable
  - Need user intervention for the first reset after compilation

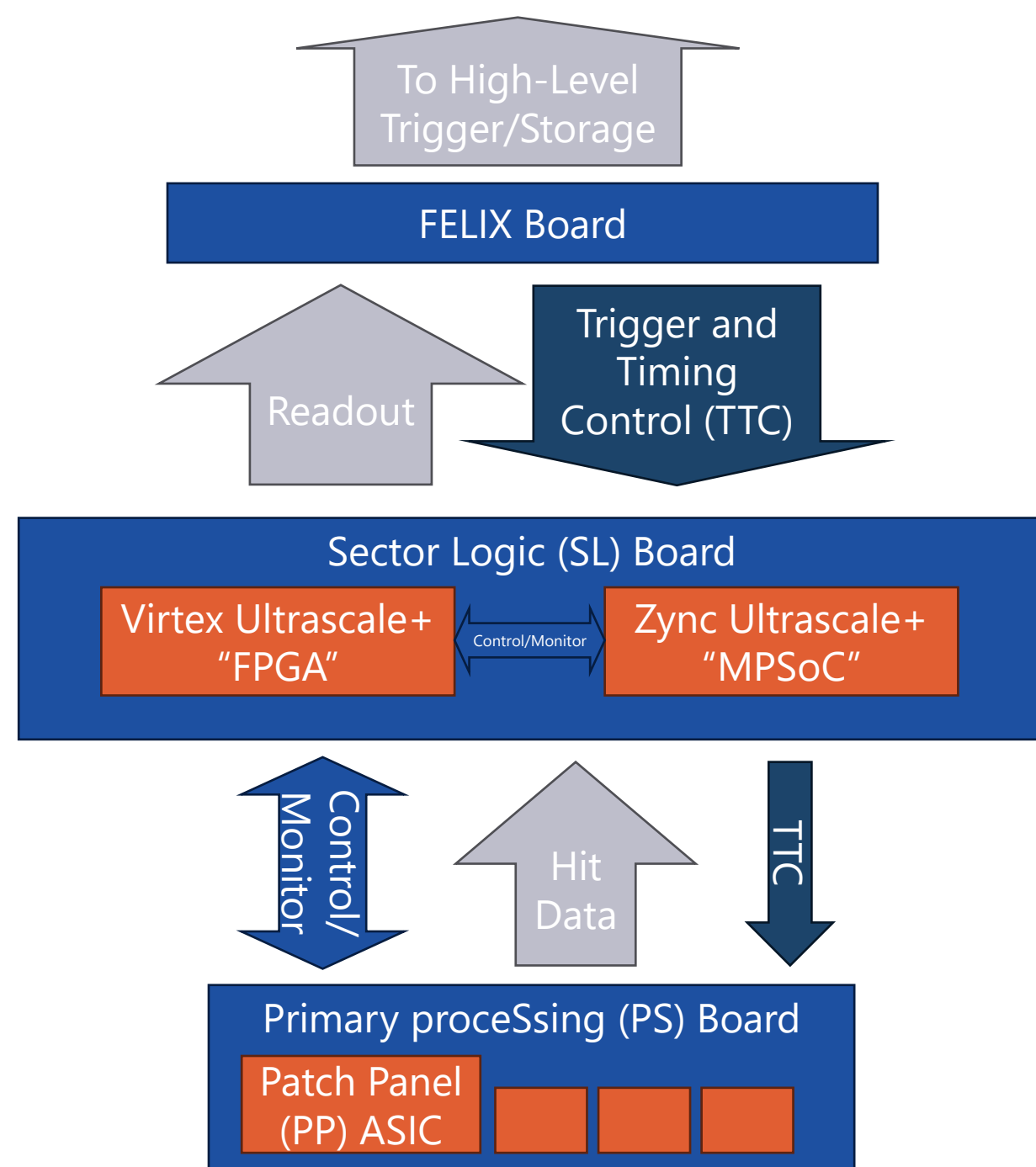
# Summary

- There are many common timing pitfalls that can ruined your FPGA design
  - Timing exception rules with clock-to-clock path spec
  - Timing exception without CDCC
- Non-CDC paths' timing failure usually indicate an inefficient design
  - Pipelining is frequently present as "the fix"
  - There is no one-stop solution
- Latency-fixed data link needs specialized design
  - Not common outside HEP application
- Same as any digital electronics, clock design in FPGA is crucial
  - Early planning is strongly recommended

# Backup

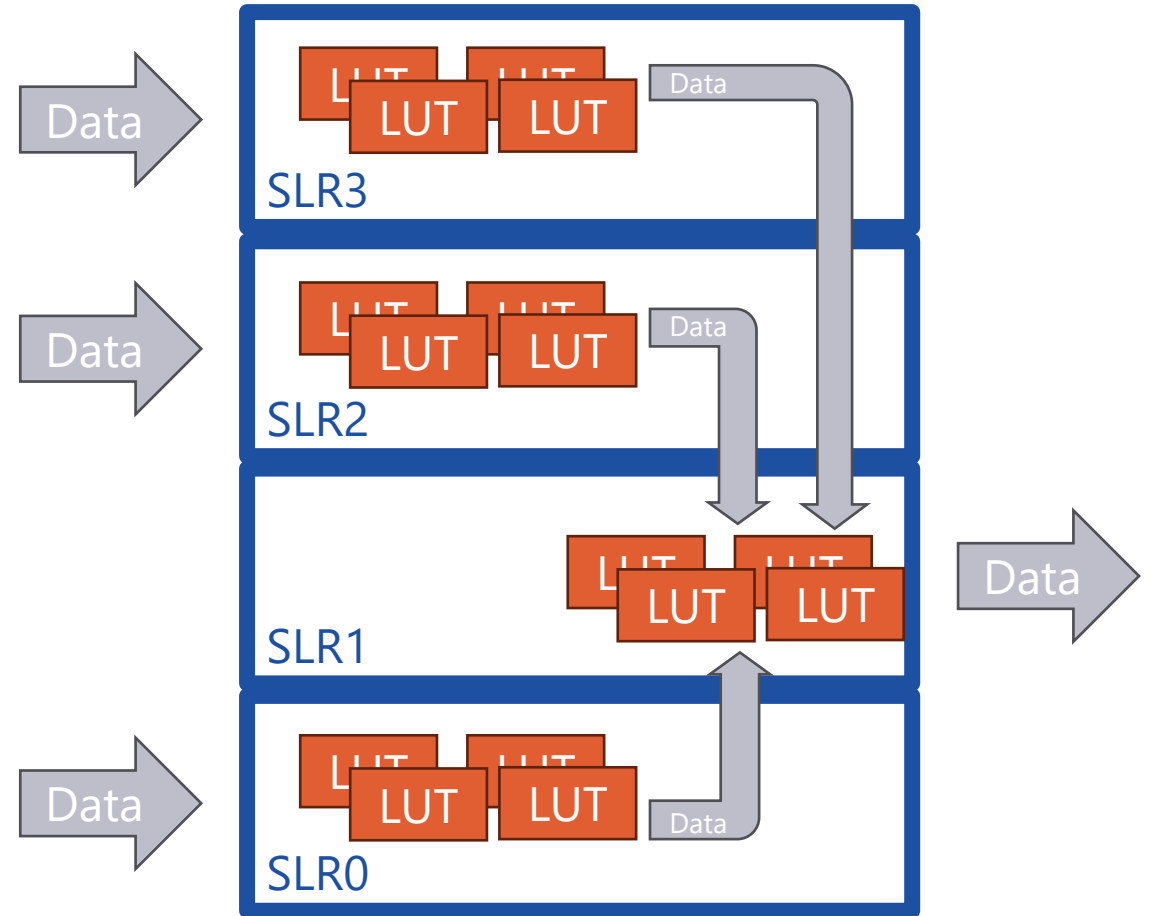
# Intro

- For this talk, I will focus on the firmware for the “FPGA” on the SL board
- What it does (very simplified)
  - Downstream (PS board)
    - Receive hit data
    - Distribute clocks and Trigger and Timing Control signals (TTC)
    - Send and receive control and monitoring info
  - Upstream (FELIX board)
    - Send trigger and data readout
    - Receive TTC signals
  - MDT Trigger Processor and Muon Central Processor are mostly omitted for the purpose of this talk



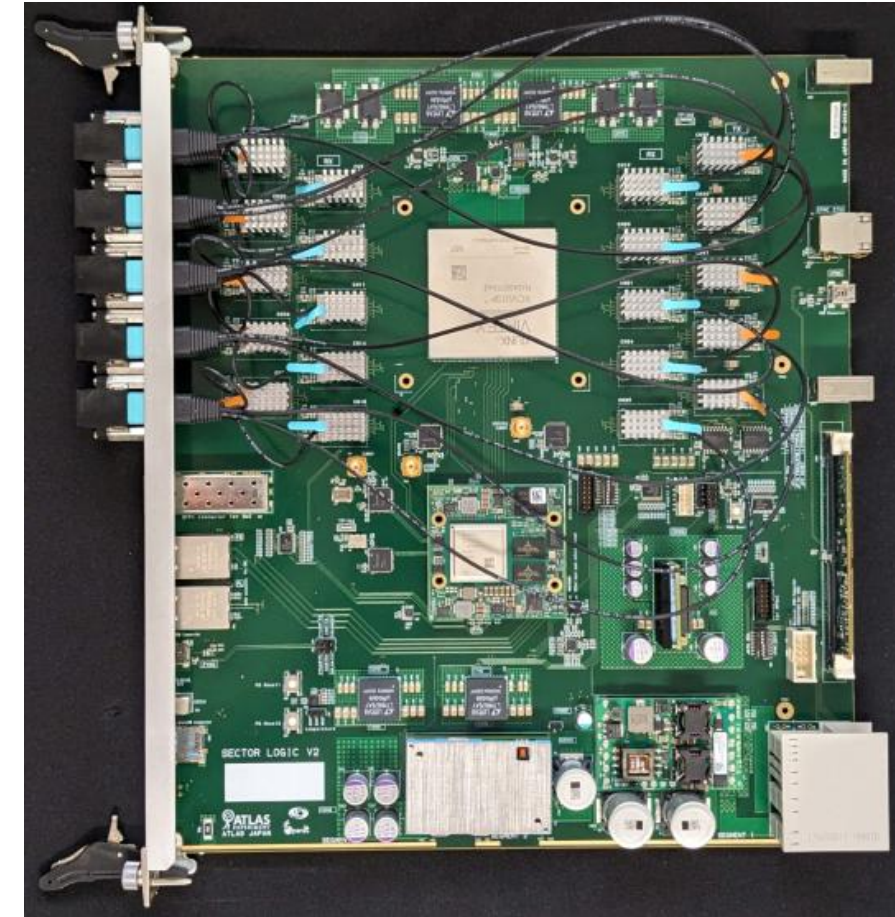
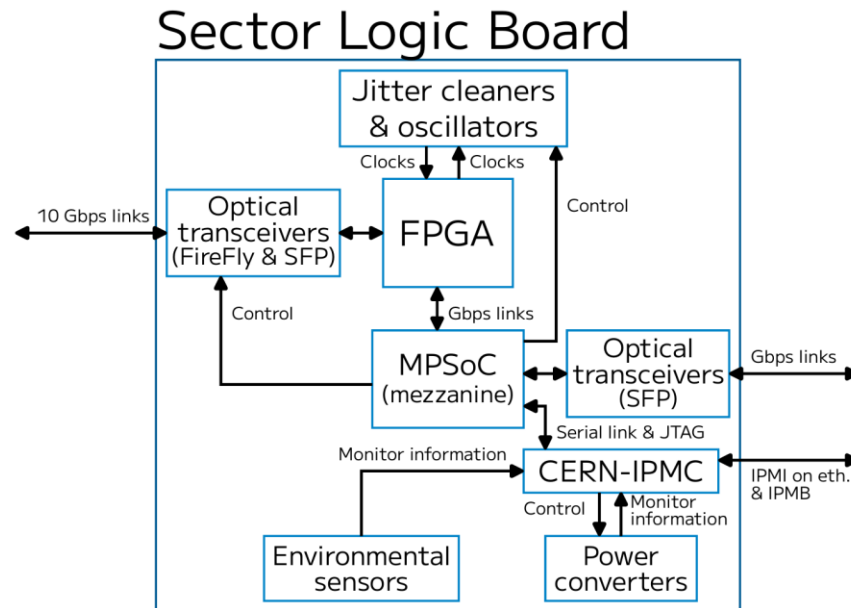
# Intro

- SL trigger logic design (also very simplified)
  - Data arrive in SLR 0/2/3 via gigabit transceivers (GT)
  - Utilized many LookUp Tables (LUTs) to form track candidates
    - Use LUTRAM, Block RAM (BRAM), or UltraRAM (URAM) depending on the size of the LUT
    - Not to be confused with FPGA LUT primitives
  - Candidates are gathered in SLR1 for checking coincidence with other systems (NSW, Tile, etc.) before being sent out to upstream FELIX
- SL readout logic has a similar concept
  - No track reconstruction logic so mostly no LUT
  - RAMs are utilized for buffer and packet forming
- TL;DR:
  - SL takes data from multiple GTs
  - process them in multiple logics that use RAM
  - Sent out data using another GT



# SL Hardware

- Major components:
  - Virtex Ultrascale+ FPGA (xcvu13p) [Readout and trigger logic]
  - Zync Ultrascale+ MPSoC [Slow control/monitoring]
  - CERN-IPMC [Power management and hardware monitor]
  - 10G Optical Transceiver (FireFly and SPF+)
  - Jitter Cleaner (Si5395) [PLL for GT clocks]
- Single-slot ATCA blade form factor

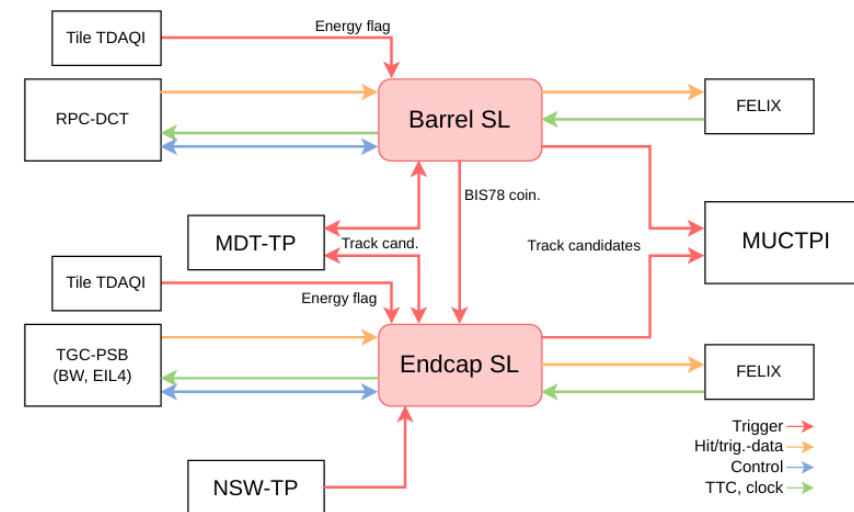


Prototype-2 SL board

# Integrations

- Status:

System	status (link)	status(protocol)
RPC-DCT	planned	planned
TGC-PSB	done	demonstrated
Tile TDAQi	done	planned
NSW-TP	planned	planned
FELIX	verified	demonstrated
MUCTPI	verified	demonstrated
MDT-TP	done	planned
SL	done	planned



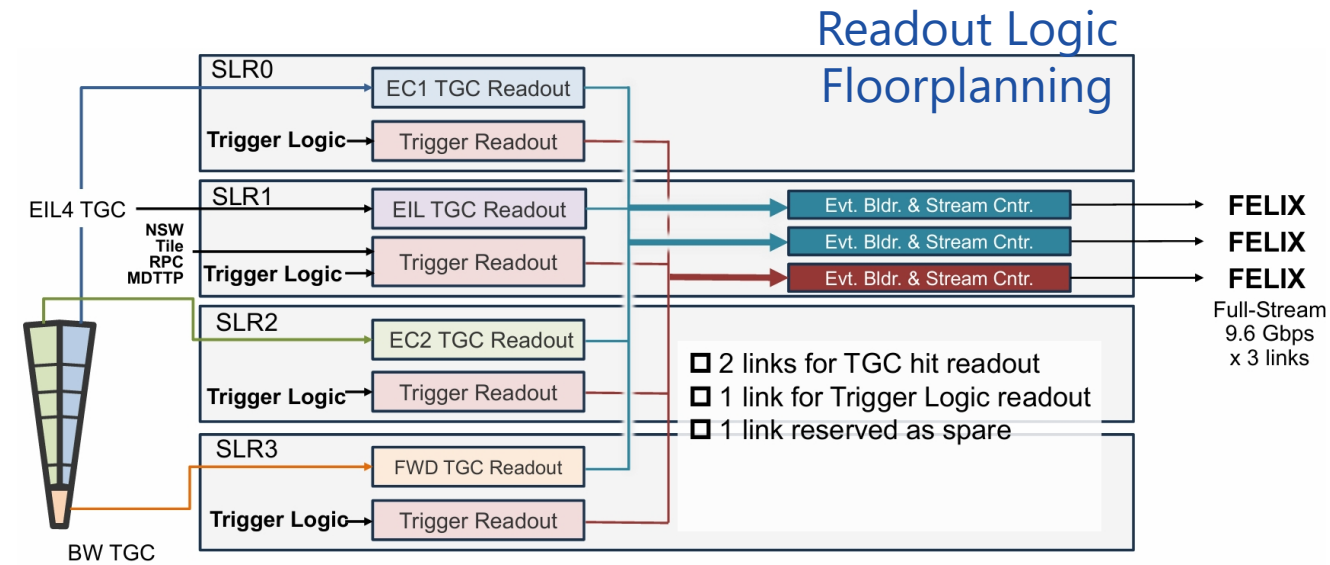
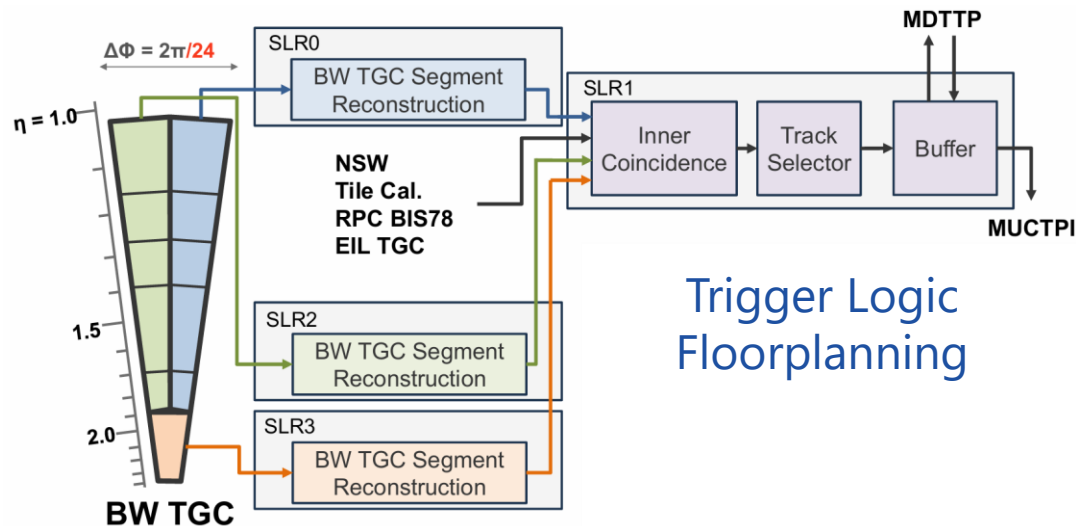
Links Diagram for  
Barrel and Endcap SL

- SL <-> MUCTPI system test has been done
  - ALTI/LTI acting as FELIX for TTC distribution, but with the new protocol
  - BER checked out ( $<10^{-15}$ )

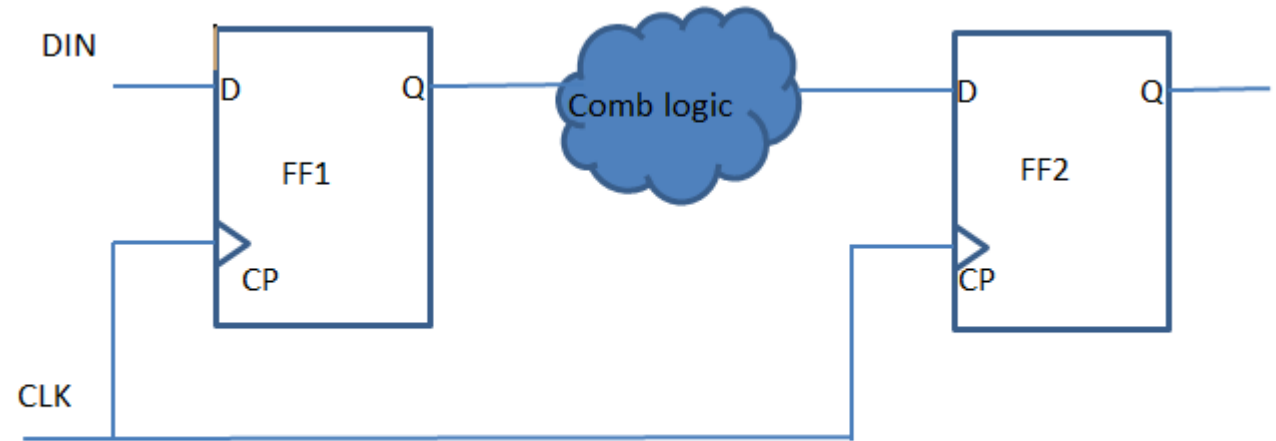
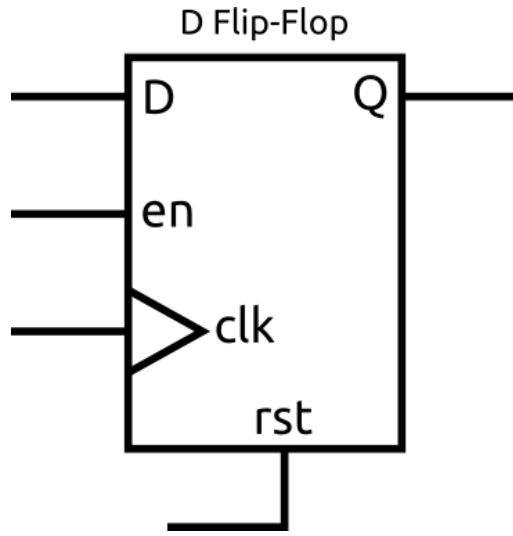


# Overview

- Endcap SL gathers hits data from TGC PS boards (10G with custom protocol)
- One SL board covers one  $\Delta\Phi=2\pi/24$  "blade"
  - For trigger logic, SLR0/2 process data from half of the "endcap" sector, while SLR3 handles the "forward" sector. All data are forwarded to SLR1, which perform coincidence logic with information from inner detectors and select the set of final track candidates
  - For data and trigger readout, the logic are implemented where the data arrive, then forward to SLR1 where the GT links to FELIX reside



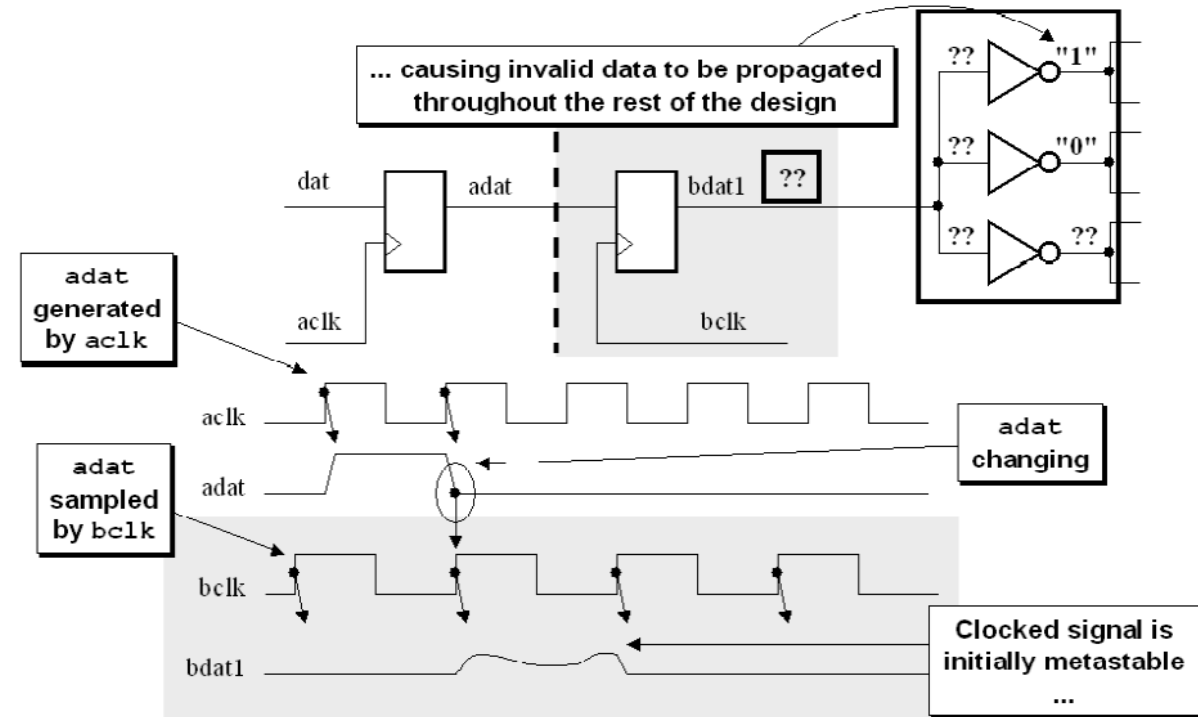
# Flip-Flop (FF)



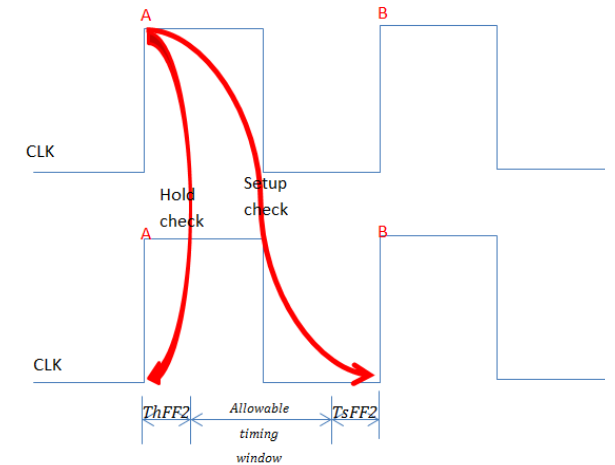
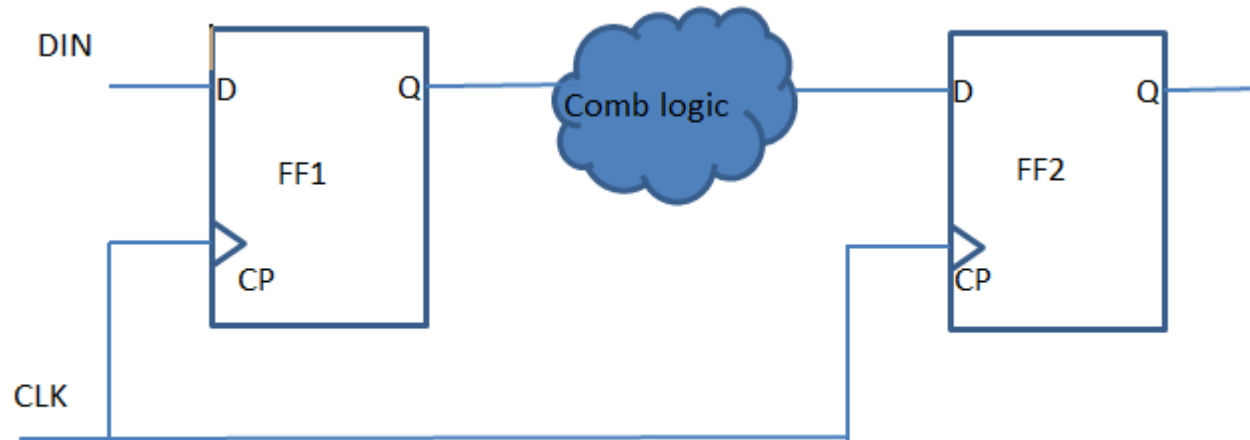
- Flip-flop is a fundamental piece of sequential logic
- It holds 1-bit signal, which gets updated at the clock (clk) edge using input at the (D) pin and outputs it to the output (Q) pin
  - Optional reset (rst) and enable (en) pin if we want it to update on some clock edge
- Reminder: Static Timing Analysis (STA) looks at the timing path between two flip-flops and checks if the signal can propagate correctly
  - Neither too late (Setup check) nor too early (Hold check)

# Metastability

- Timing violation is obviously bad because the wrong signal can be captured by the destination FF
- But it is worse than that because it can make FF goes into metastable state
  - Signal is neither 0 nor 1
- Why is metastability bad? [Quote](#):
  - If the unstable data is fed to several other places in the design, it may lead to a high current flow and even chip burnout in the worst case.
  - Different fan-out cones may read different values of the signal, and may cause the design to enter into an unknown functional state, leading to functional issues in the design.
  - The destination domain output may settle down to the new value or may return to the old value. However, the propagation delay could be high leading to timing issues.



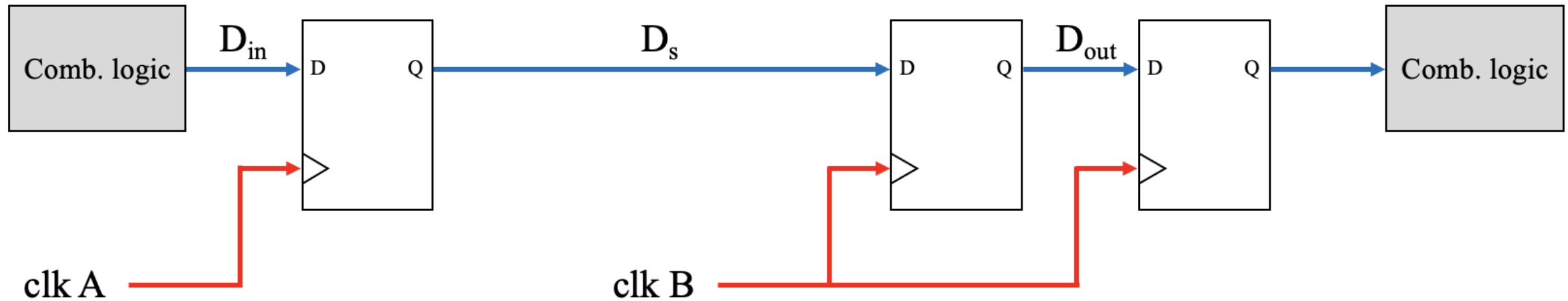
# Timing Analysis



- Static timing analysis (STA) tool checks paths between all pair of registers/Filp-Flop (FF)
- Determine the allowed timing window and try to place every components accordingly
- Timing “slack” is the margin left before going out of the timing window
- If any path fails, we have “timing violation”
  - Timing slack  $< 0$
  - Data arrive too early (hold time violation) or too late (setup time violation)

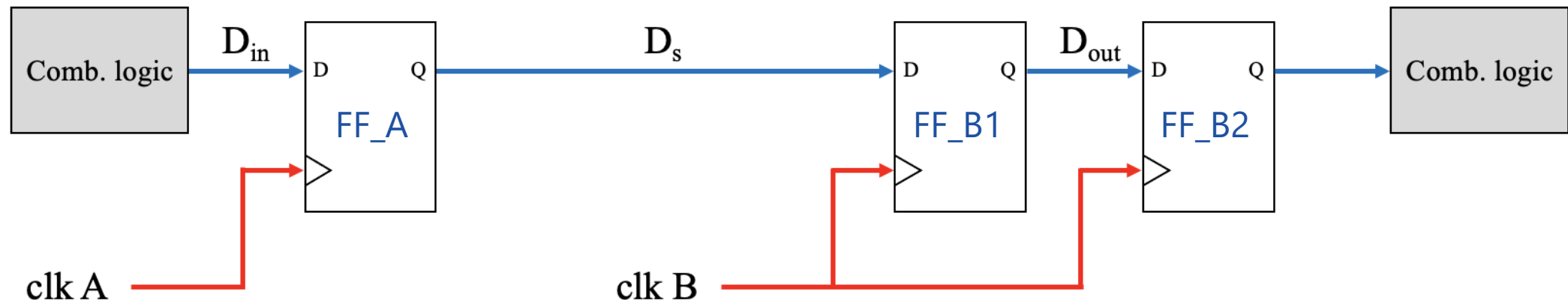
# Timing for 2-FF Synchronizer

- With metastability guard in place, now we can add timing info to help the STA tool
- The two clkB FF's should be close together to allow maximum time for metastability to settle
  - Xilinx specific: attribute `ASYNC_REG = true`
  - Platform independent: use `set_max_delay` with value  $< (\text{clkB period} - \text{margins})$ . Preferably as small as possible
    - Do not use `-datapath_only` here (more on this later)
- Add exception to the CDC path (`D_s`)



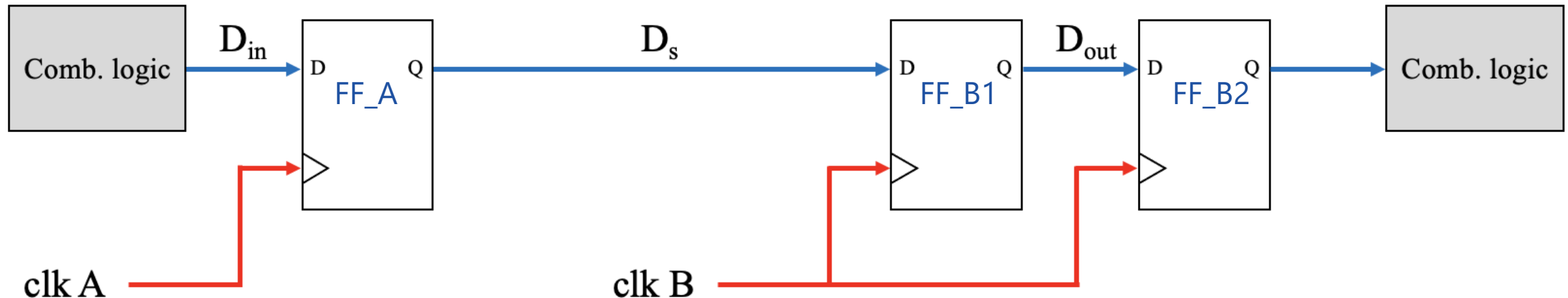
# CDCC Timing Exception

- Timing exceptions such as `set_false_paths` tell the tool to ignore the paths
  - They do not solve timing violations
- With a **metastability guard** in place, we can “cut” the CDC path ( $D_s$ )
- **Never** use clock-to-clock rule
  - Accidentally cut other paths
  - This includes `set_clock_groups -asynchronous`
- Instead, specifies the pins (reg-to-reg rule)
  - E.g. `FF_A/C` pin to `FF_B1/D`

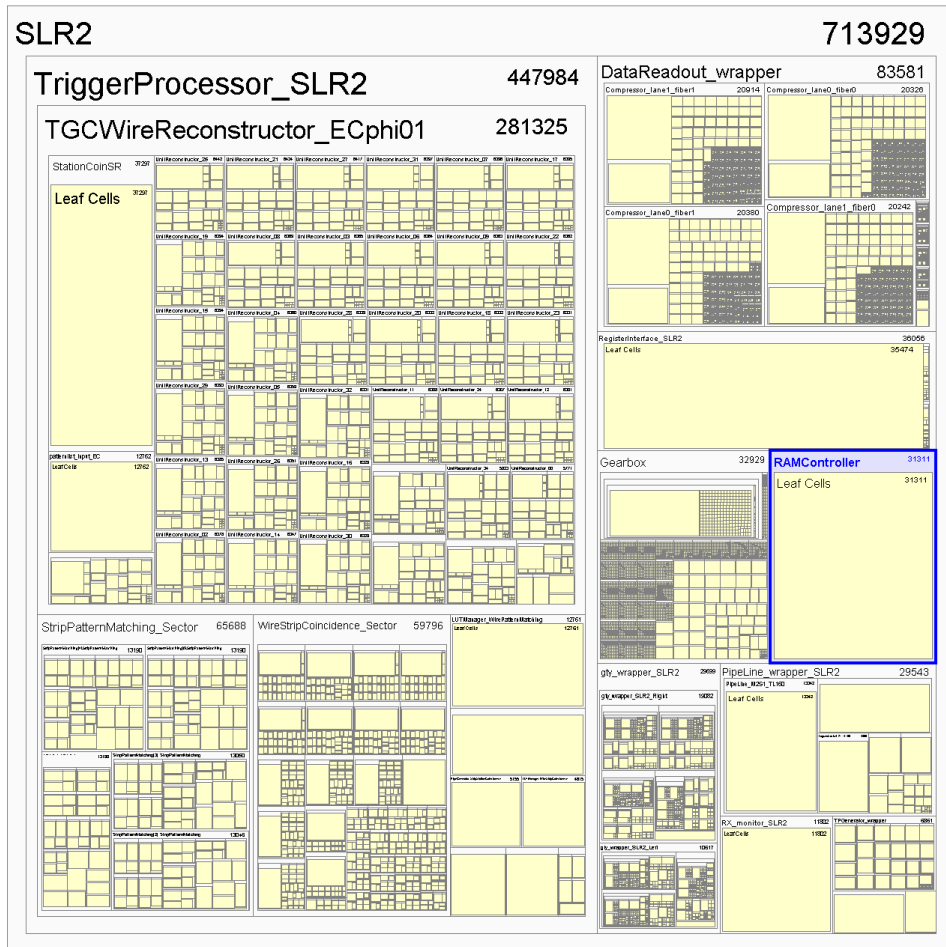


# CDC Path Exception

- “Traditional” but wrong way to do it: use `set_false_path` between `clkA` and `clkB`
  - Path between `FF_A` and `FF_B1` will be totally ignore and the tool is allowed to put any distance between them
    - Could be 1ns. Could be 9999ns (unlikely, but allowed)
  - If one insist to do it this way (read: don't), then `set_false_path` on this path (`FF_A`'s `Q` to `FF_B1`'s `D`) only
- The correct way: use `set_max_delay -datapath_only  $t_{delay}$` 
  - This tell the tools “ignore the clock relationship here but make sure the delay is no more than  $t_{delay}$ ”
  - $t_{delay}$  can be arbitrary large. But [ $t_{delay}$ =period of the faster clock] is a good starting choice

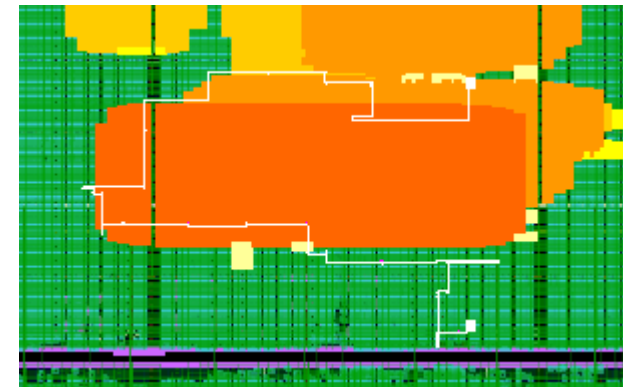


# RAMController—A Case Study



Hierarchy View representing the relative size of resources usage with boxes

- This module is very large for what it does
  - Write values to the LUTs during the initial boot up
- Critical paths have almost no logic
- Part of the "congested" areas



Color represent congestion.  
Darker orange is more congested

- Pipelining does not help



# Reduced Demultiplexer



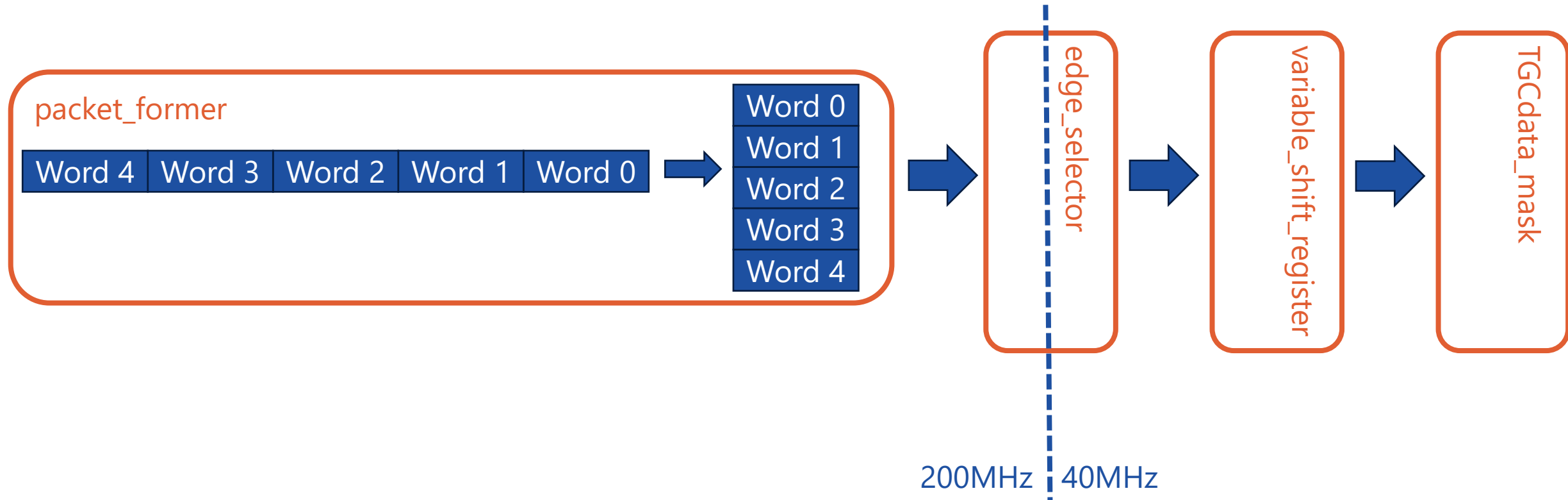
# reset\_sync\_wrapper

- A chain of reset\_synchronizer ([5FF reset bridge](#)) and bit\_synchronizer ([5FF synchronizer](#))
- Way overkill for CDC
- ~~Have no purpose~~ Apparently, there was
  - It was to get a reset signal synchronize into some clock domain while “lining up” with the raising edge of the LHC clock
  - This double reset “method” is also used without this module
  - But synchronizer does not do that
- Will be removed
- Instead, we need proper gearboxes and CDCC

```
for (gi = 0; gi < SYNC_WIDTH; gi = gi +  
1) begin : reset_sync  
    reset_synchronizer reset_sync (  
        .clk_in (clk),  
        .rst_in (rst_in[gi]),  
        .rst_out (rst_int[gi])  
    );  
    bit_synchronizer bit_sync (  
        .clk_in (clk),  
        .i_in (rst_int[gi]),  
        .o_out (rst_out[gi])  
    );  
end
```

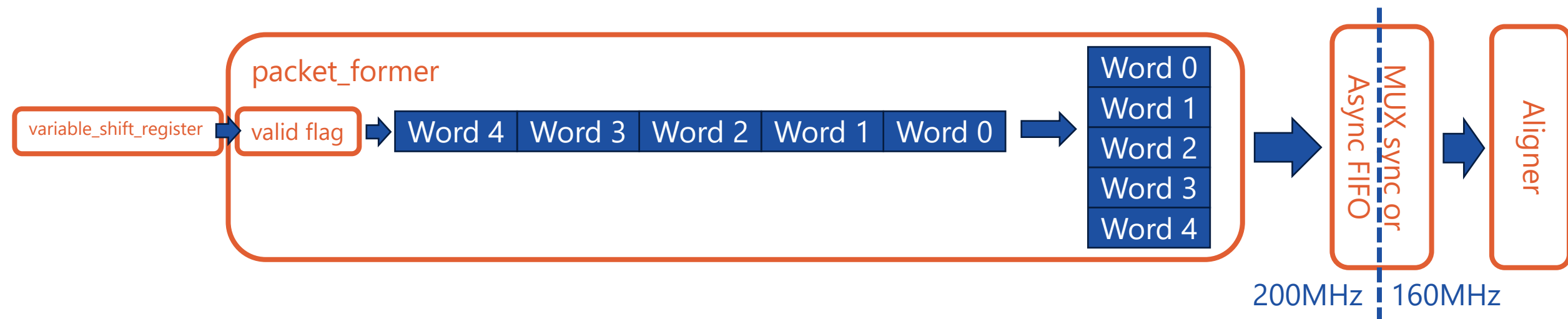
# Original (RX) Gearbox

- Originally, the (RX) gearbox consists of, in order:
  - packet\_former: handle the data bus conversion i.e. the 32:160 gearbox
  - edge\_selector: avoiding metastability
  - variable\_shift\_register: add delay to BCID-match
- There is also a "TGCdata\_mask" to flag invalid PSB links post-gearbox



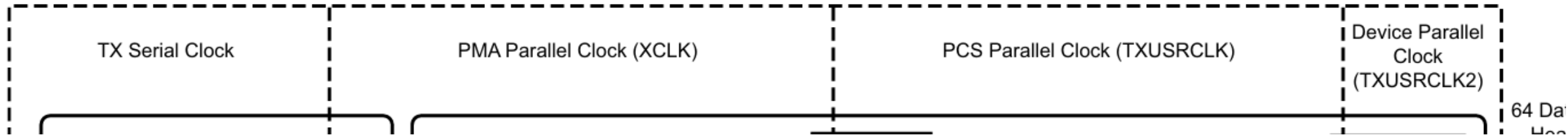
# New (Purposed) RX Gearbox

- packet\_former: still a 32:160 gearbox
  - Add options for MDTTP/TTC format and big-endian (Sue) and handle invalid (masking) flag (Max)
- variable\_shift\_register: move to the front of packet former but use 40MHz delay step
- CDC with MUX sync or Async FIFO
  - Use 160MHz to match trigger logic and ensure enough edges to capture data
  - Same no margin requirement
- Aligner: align all PSB link to the same 160MHz phase
  - Not aligning BCID. That is still handled manually by variable\_shift\_register



# MGT Clock Domains

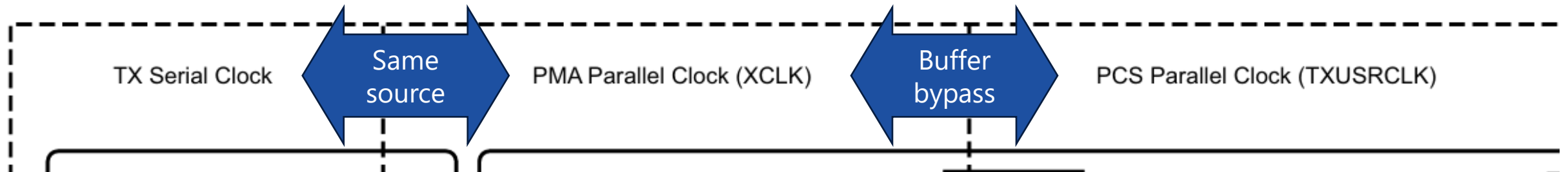
- A quick primer on Multi-Gigabit Transceiver (MGT) clock domains:



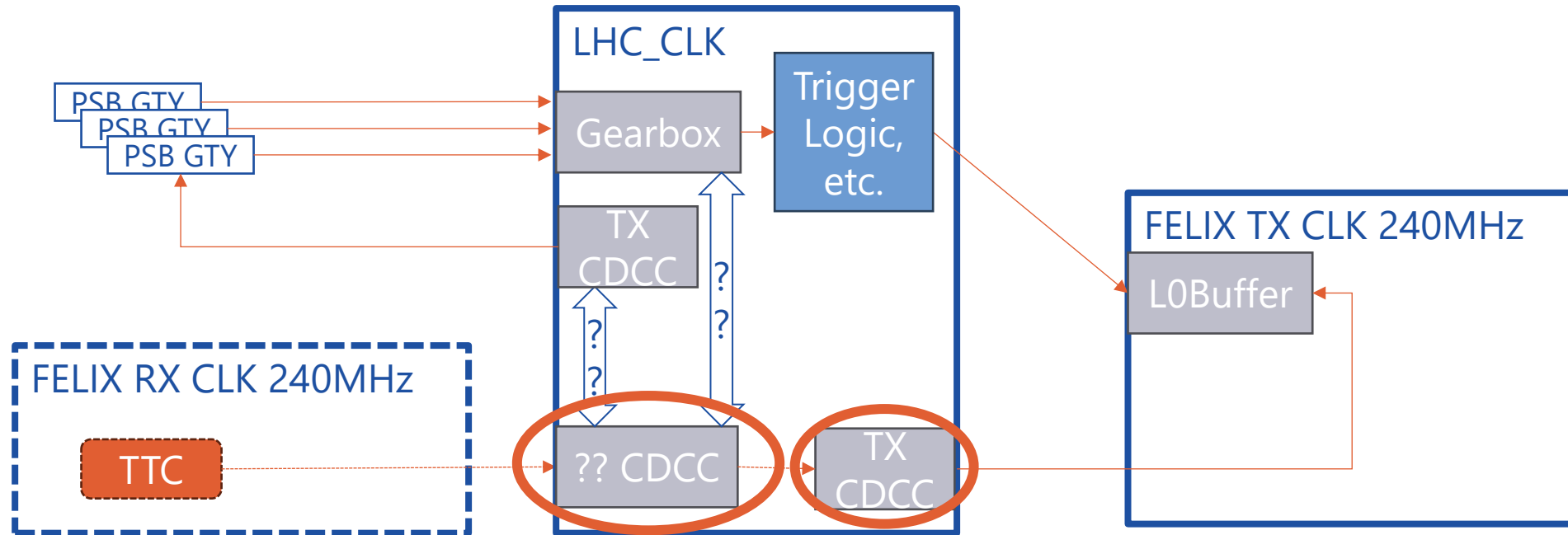
- Terminologies:
  - Frame: complete set of data. Typically, one frame per LHC bunch clock (BC) in our design
  - Word: subset of a frame. This is the parallel chunk of data MGT works with internally
  - Serial stream: stream of bits in transmission
- Clocks:
  - Serial clock is the clock that the actual transmission runs on. For MGT, this is O(GHz)
  - XCLK is a clock for word processing in MGT. For us, this is usually 200 or 240 MHz
  - TX/RXUSRCLK is also a clock for word, but live in the fabric i.e. the rest of our design instead
- For reasons not explain here, XLCK and TX/RXUSRCLK have the same frequency but cannot be the same clock

# MGT Buffer

- A CDC between XCLK and TX/RXUSRCLK is needed
- In a standard setup, a (asynchronous) FIFO is used
  - Simple, but not latency-fixed, as observed by our own testing (thanks Okumura-san)
- In “buffer bypass” mode, TX/RXUSRCLK is continuously adjusted to align with XCLK
  - Complicated procedures. Not only it has to compensate for fixed delay source such as routing but also voltage and temperature variation. Collectively refers to as Process, Voltage, and Temperature (PVT)
  - Thankfully, those are done by Xilinx and for us, it is just a “switch” to choose from
  - The latency between TX/RXUSRCLK and XCLK is fixed after alignment

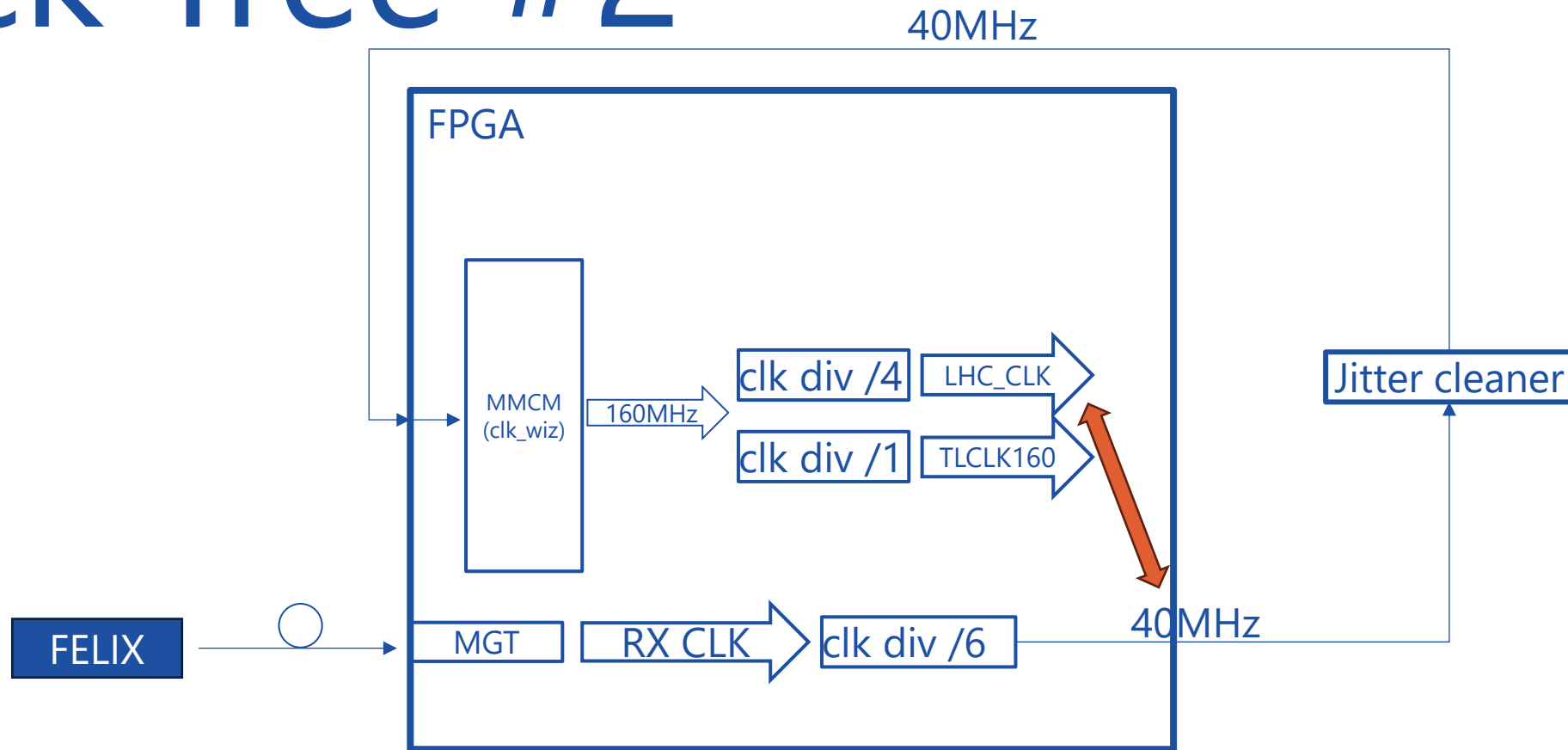


# Problem #2



- Problem #2: How to do clock domain crossing (CDC) for TTC signals?
  - They arrive in FELIX RX clock domain
  - Interact with trigger logic in LHC\_CLK domain
  - Control the readout data in FELIX TX clock domain
  - Each CDC adds at least one cycle uncertainty to the latency
    - L0Buffer module absorbs the uncertainty for readout data, but itself is controlled by TTC signals

# Clock Tree #2



- We switched to single 160MHz MMCM output, then use dividers to get 40MHz
  - No phase error help fix timing violations
- Problem: there are four possible relative phases between the original 40MHz and LHC\_CLK
- Phase between the FELIX's 40MHz and LHC\_CLK is **unknown, and not fixed**



# RX CDC

[https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink\\_timing\\_link.pdf](https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink_timing_link.pdf)

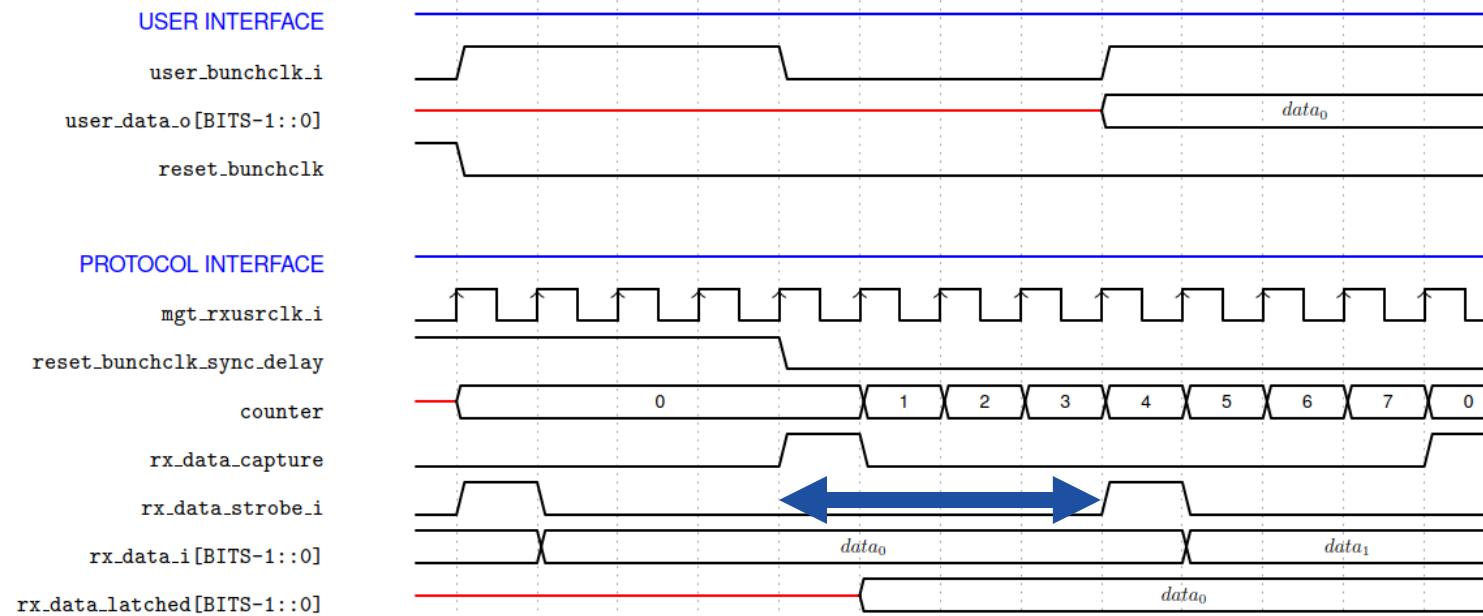


Figure 11: Rx clock-domain-crossing. Example for N=8.

- The trick for this CDR is to use reset signal from LHC\_CLK to start a word counter in the RXUSRCLK, this counter is then use to generate a pulse to capture the data within the RXUSRCLK domain.
  - May need extra delay to ensure the data is stable on the capture (posedge) of LHC\_CLK
- The reset signal can have variation per reset, so the position of data\_strobe respect to the counter can be save after the first reset to get a fixed latency for subsequent resets
- TX side needs to also has latency-fixed for this to hold

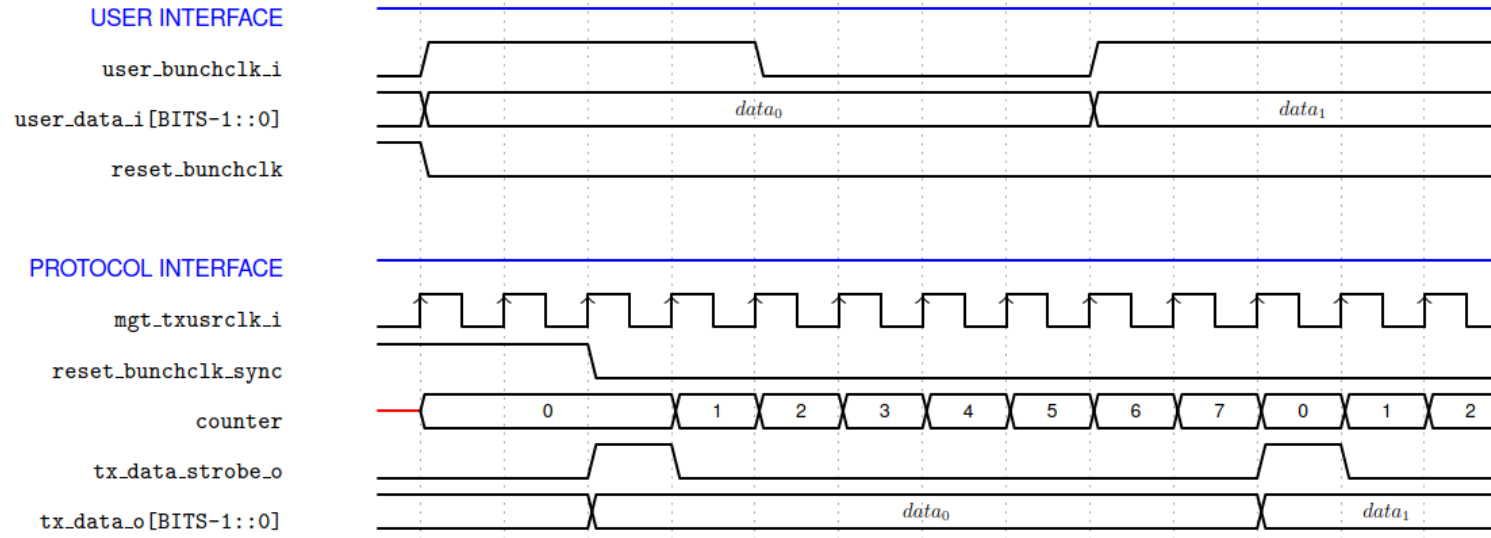


Figure 9: Tx clock-domain-crossing. Example for N=8.

- A reset signal from LHC\_CLK is sent to txusrclk through a 2-FF synchronizer
- When the reset signal arrived in the txusrclk domain, it acts as the start signal of N-cycle counter
  - This counter generate a data\_strobe signal that goes high once every N-cycles
- The data are transferred when data\_strobe is high
  - data\_strobe is guaranteed to be far away from the transition region of the source, thus safe from metastability
  - Delay does not change during the operation

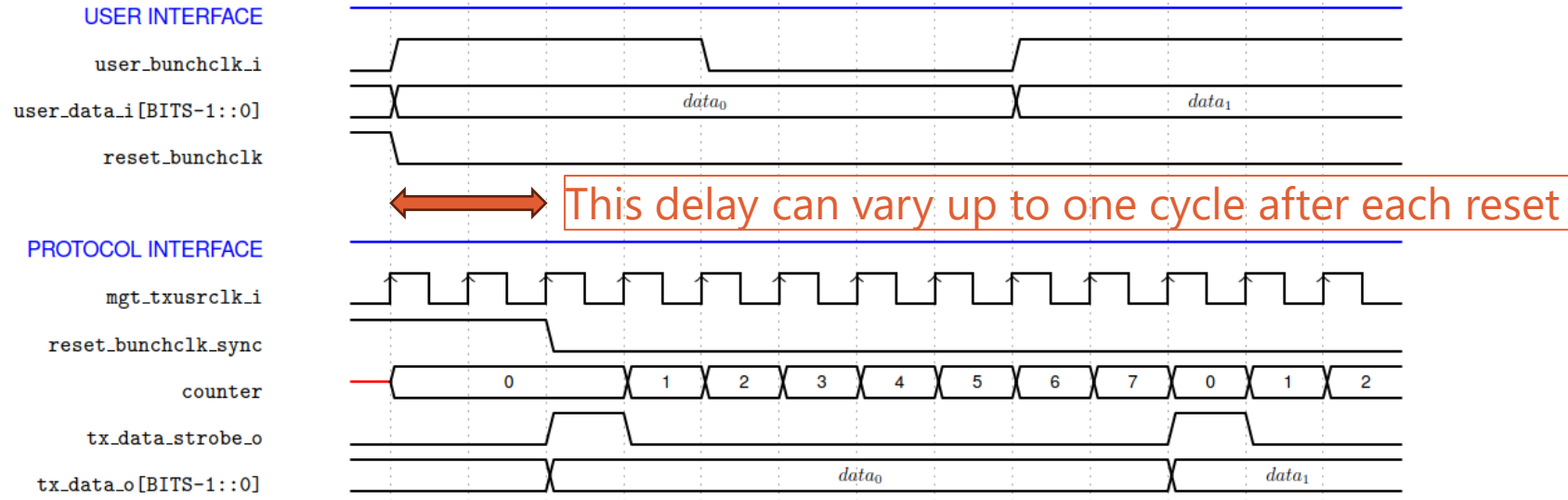


Figure 9: Tx clock-domain-crossing. Example for N=8.

- However, 2-FF has an uncertainty of about one cycle for when the data get transferred
  - The destination clock edge can be in the transition region of the reset signal
    - For most of the compilations, this will not be case, and therefore the delay is fixed
    - But when stars (mis)aligned, we roll the dice for the number of delays after every reset
- To avoid this issue, we will introduce an auto-calibration procedure after each reset

# Auto-calibration

[https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink\\_timing\\_link.pdf](https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink_timing_link.pdf)

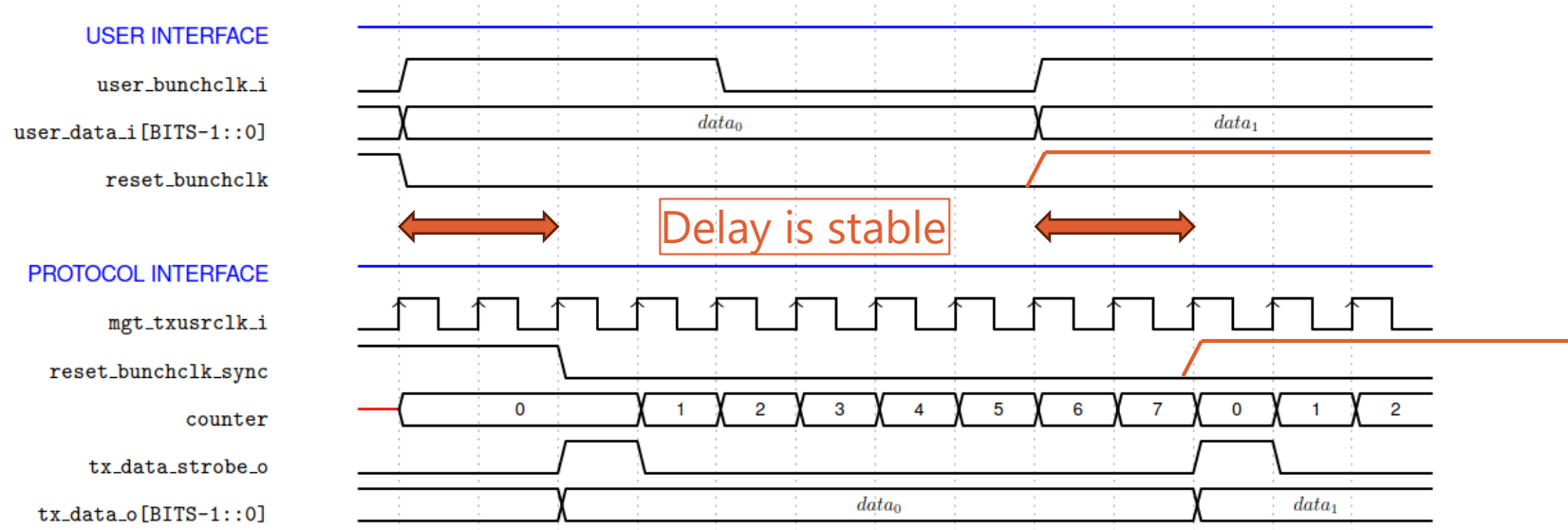


Figure 9: Tx clock-domain-crossing. Example for N=8.

- After all clocks are stable, the TX CDC enters the **calibration phase**
  - During this, the "reset\_bunchclk" in the picture keep **changing every BC cycle**
  - "reset\_bunchclk\_sync" (i.e. txusrclk side) should reflect that and flip at the BC interval as well
- Like before, the reset signal acts as the start signal of a counter in the txusrclk domain
- Look at the counter number when the **"reset\_bunchclk\_sync" (txusrclk side) flips**
  - If it is always "correct" (7 in this picture, 5 for us), then there is no metastability. We are done

# Auto-calibration

[https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink\\_timing\\_link.pdf](https://gitlab.cern.ch/HPTD/tclink/-/blob/master/tclink_timing_link.pdf)

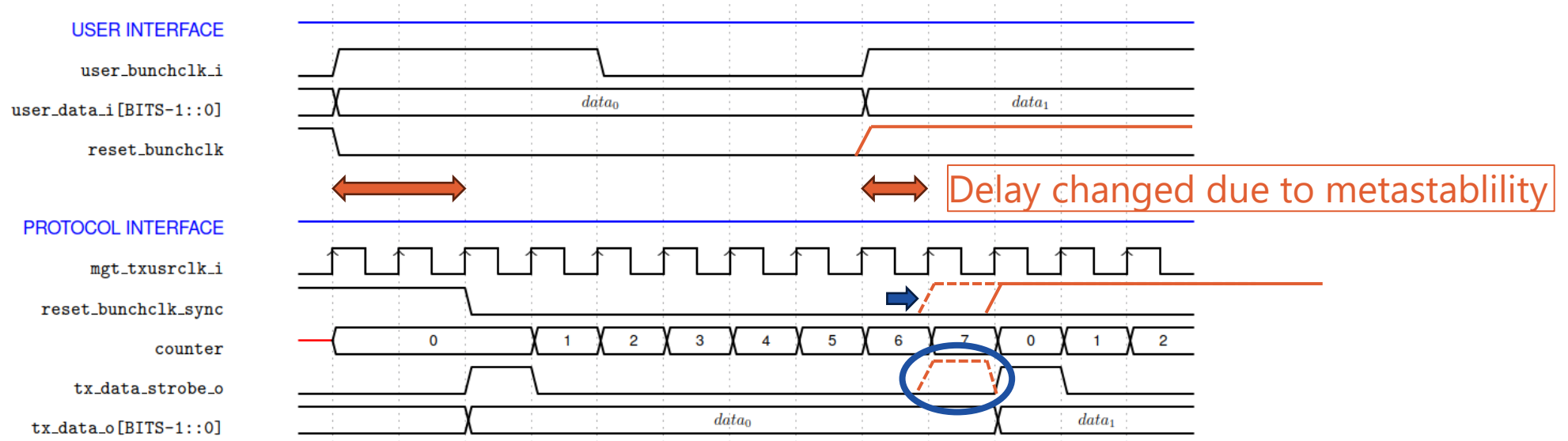


Figure 9: Tx clock-domain-crossing. Example for N=8.

- However, if we see an "incorrect" number (6/8 in this picture, 4/6 for us), then we hit metastability
- For example, if the delay got "shorten":
  - We will see the `reset_bunchclk_sync` flips at counter N=4 (instead of 5)
  - Then the next flip will be at counter N=6
- We (the code) will pick one choice, let's say the "6-edge", to be the strobe signal edge

# Auto-calibration

## Procedures

- Define how long the calibration phase is
  - The longer we wait, the more certain that we will hit an incorrect number at least once
- If we hit the chosen “incorrect” number (6), exit the calibration phase
  - Make the reset signal (on both domains) goes low and stay low
  - Reset the counter to zero
  - Changes the counter to normal mode (makes data\_strobe goes high once every  $N=6$  cycles)
- If an incorrect number is never seen for the entire duration of the calibration phase
  - Make the reset signal (on both domains) goes low and stay low
  - Change the counter to normal mode (makes data\_strobe goes high once every  $N=6$  cycles)