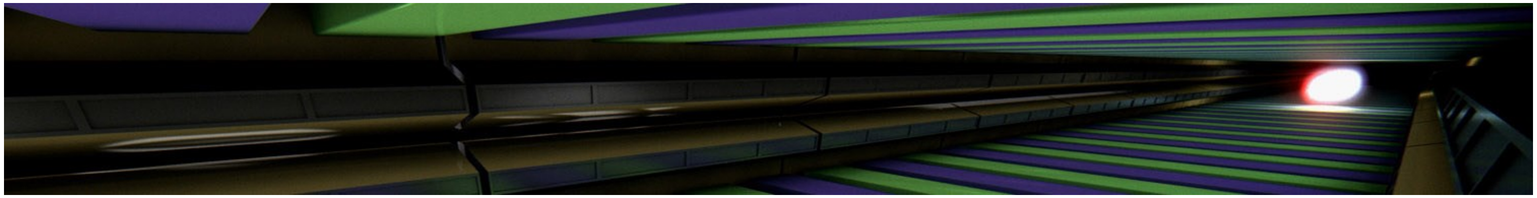


Front end electronics and FPGA developments

Patrick Gessler

for the

Joint Electronics Group

Detector Developments (WP75) + Data Acquisition (WP76)

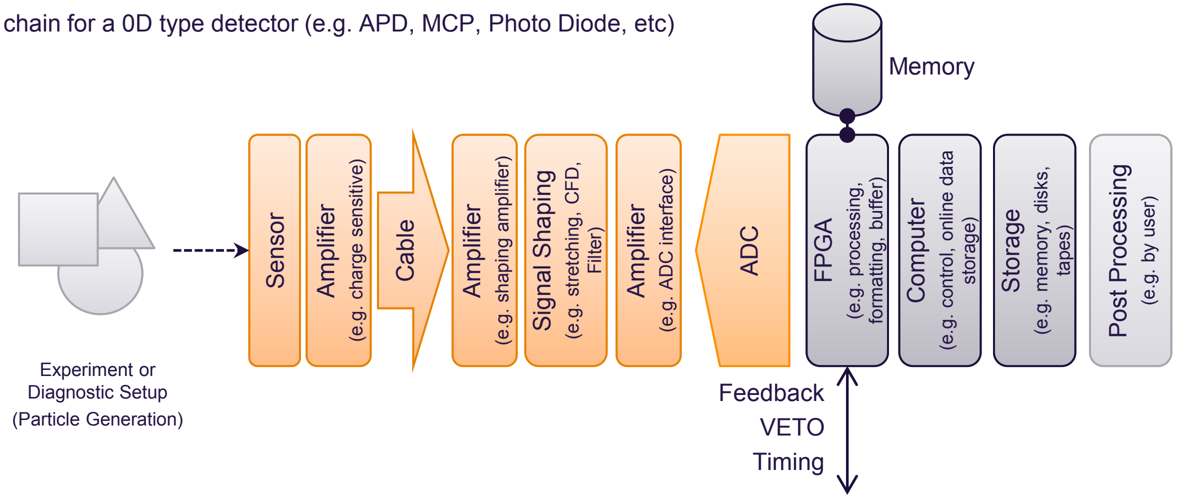
European XFEL GmbH

Overview

- Front end electronics
  - ➔ Overview of typical Detector-Data-Acquisition Chains
  - ➔ Usual tasks and pitfalls depending on detector and detection principle
- Introduction into MicroTCA as data acquisition platform
  - ➔ Motivation
  - ➔ Overview
- FPGA developments
  - ➔ Brief introduction into FPGAs
  - ➔ Main tasks and required features
  - ➔ Programming principles followed
  - ➔ High-Level Algorithm Programming Framework with Simulink
  - ➔ High-Speed and Low-Latency Serial Interfaces
  - ➔ High resolution trigger
  - ➔ Pulse energy detection
- Conclusion

Overview of typical Detector-Data-Acquisition Chains

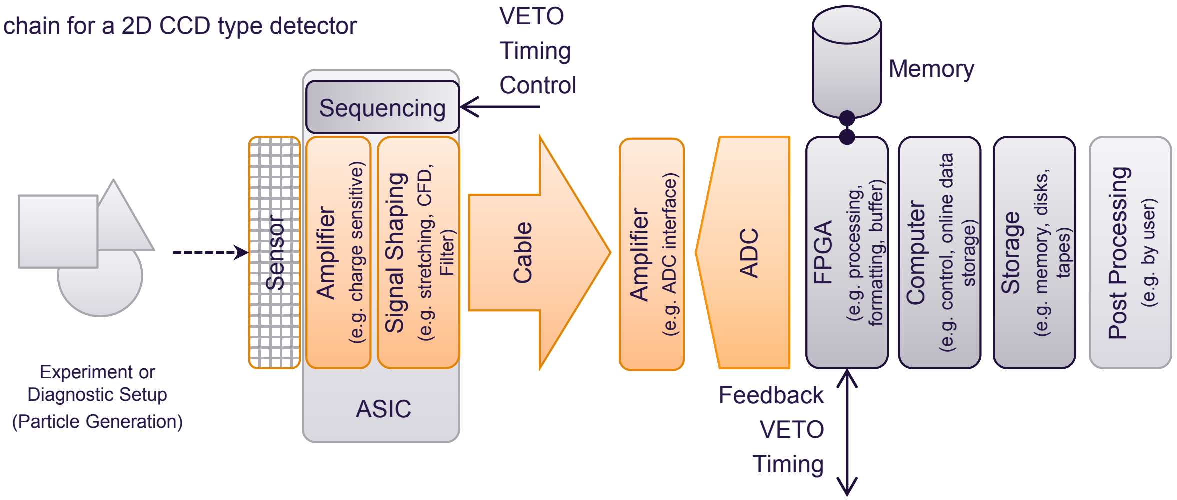
Typical chain for a 0D type detector (e.g. APD, MCP, Photo Diode, etc)



🟧 Analog

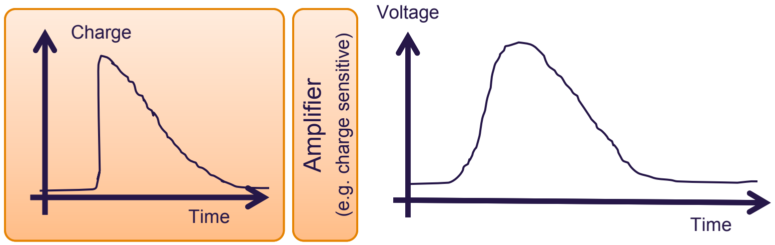
⬜ Digital

Typical chain for a 2D CCD type detector



Front end electronics: Typical tasks depending on detector and detection principle

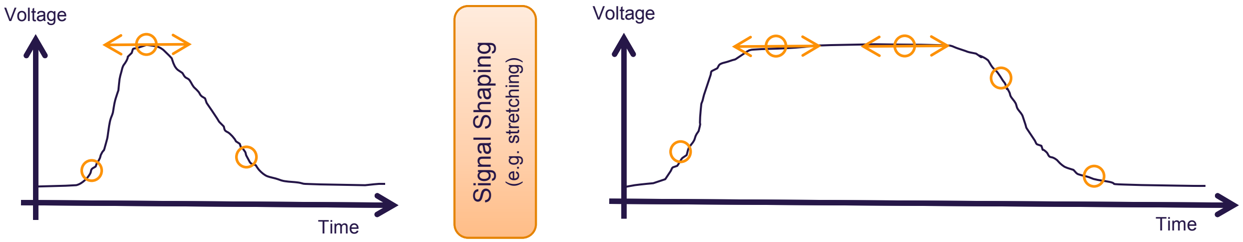
- Convert collected charge into voltage or current



  - The bandwidth of the amplifier could be lower than the one of the sensor (like in the example)
  - Charges are collected and formed in a current
  - The amplifier produces a proportional output voltage

Front end electronics: Typical tasks depending on detector and detection principle

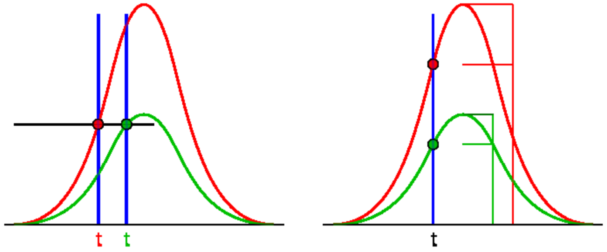
- Pulse stretching
  - To sample all important properties with limited sampling speed
  - To reduce noise by oversampling and later averaging
  - To reduce jitter based effects (sampling time fluctuations)



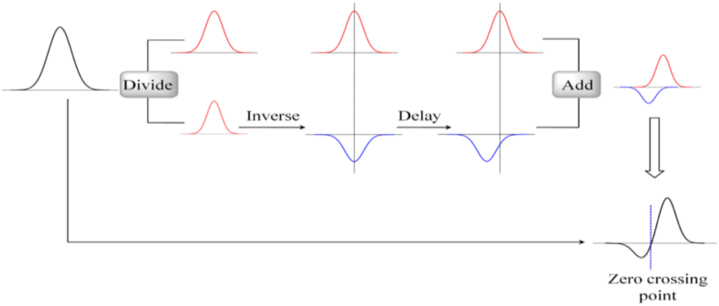
  - Negative:
    - ➔ Fast signal properties are lost
    - ➔ Time between pulses has to be longer

Front end electronics: Typical tasks depending on detector and detection principle

- Constant Fraction Discrimination (e.g. for ToF experiments)
  - Using a threshold for time determination provides an error



  - Amplitude independent time detection by constant fraction



  - Divider and time delay are usually user parameters!

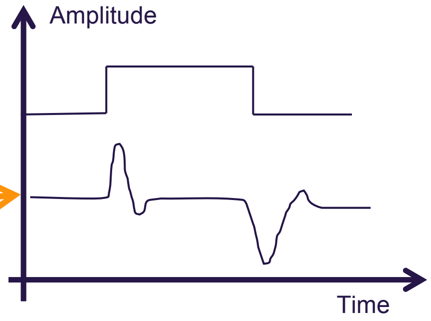
Pictures from wikipedia

Front end electronics:
Typical tasks depending on detector and detection principle

- Low pass filtering
  - Def.: All frequency components UP TO the cut-off frequency will pass
  - The cut-off is the 3dB line → this means higher frequencies are only attenuated!
  - The order of the filter defines the “sharpness” of the cut-off
  - Applications
    - ➔ Reduce high frequency noise
    - ➔ Match signal bandwidth to ADC sampling rate or input bandwidth
      - To avoid signal reflections and imaging effects in the data

Front end electronics: Typical tasks depending on detector and detection principle

- High pass filtering
  - Def.: All frequency components ABOVE the cut-off frequency will pass
  - The cut-off is the 3dB line → this means lower frequencies are only attenuated!
  - The order of the filter defines the "sharpness" of the cut-off
  - Applications
    - → Reduce low frequency noise
    - → Block DC voltages (remove constant offset)
    - → Allow edge detection



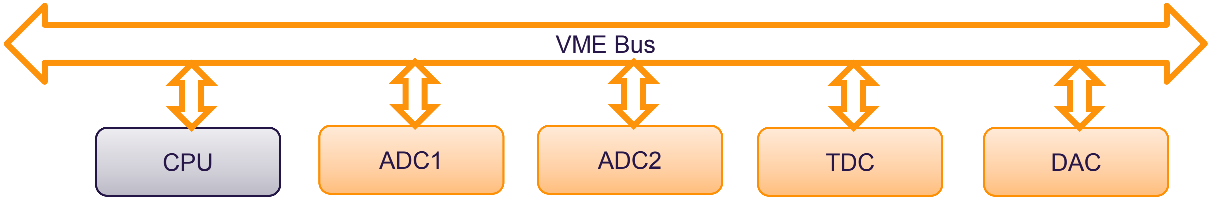
- Band pass filtering
  - Combines low and high pass and provide two cut-off frequencies
- Notch filter
  - Is the inverse of the band pass filter

Front end electronics:
Typical tasks depending on detector and detection principle

- AC and DC input coupling to ADCs
  - AC: Alternating Current
    - → Used for alternating (usually periodic) signals
    - → Includes high-pass filter to block DC voltages!
    - → Be aware of high-pass filter effects like fluctuating base line
    - → Positive: Usually only passive connection (capacitor)
  - DC: Direct Current
    - → Used for arbitrary signals
    - → Usually includes an amplifier (increases noise and limits bandwidth)
    - → Due to the amplifier mostly low-pass filter behavior
    - → Positive: constant base line (no drifts)

Introduction into MicroTCA – Motivation

- Limitations on existing VME based solutions
  - Bandwidth: only one parallel bus



  - No module replacement at run time (Hotswap)
  - Remote control
  - Health management
  - Redundancy
  - No in-crate timing distribution

Introduction into MicroTCA – ATCA and MTCA

- Advanced Telecommunication Computing Architecture (ATCA)
- Advanced Mezzanine Card (AMC)
- Micro Telecommunication Computing Architecture (µTCA)

Advanced Mezzanine Card (AMC)

µTCA Crate

ATCA Blade / Module

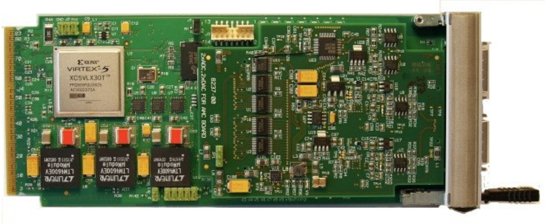
ATCA Crate

Introduction into MicroTCA – MicroTCA.4

- xTCA not designed for physics applications (e.g. Timing, Interfacing, ADCs)
- xTCA for Physics Group at PCI Industrial Computing Manufacturers Group (PICMG)
  - Participants from labs and industries
  - Defined extensions of the standard for our applications
    - → Timing interface on the backplane
    - → High-speed module interconnections
    - → More space → double size modules
    - → Modular interfacing and signal shaping → Rear Transition Modules (RTMs)

UCL Clock & Control RTM

DESY DAMC2

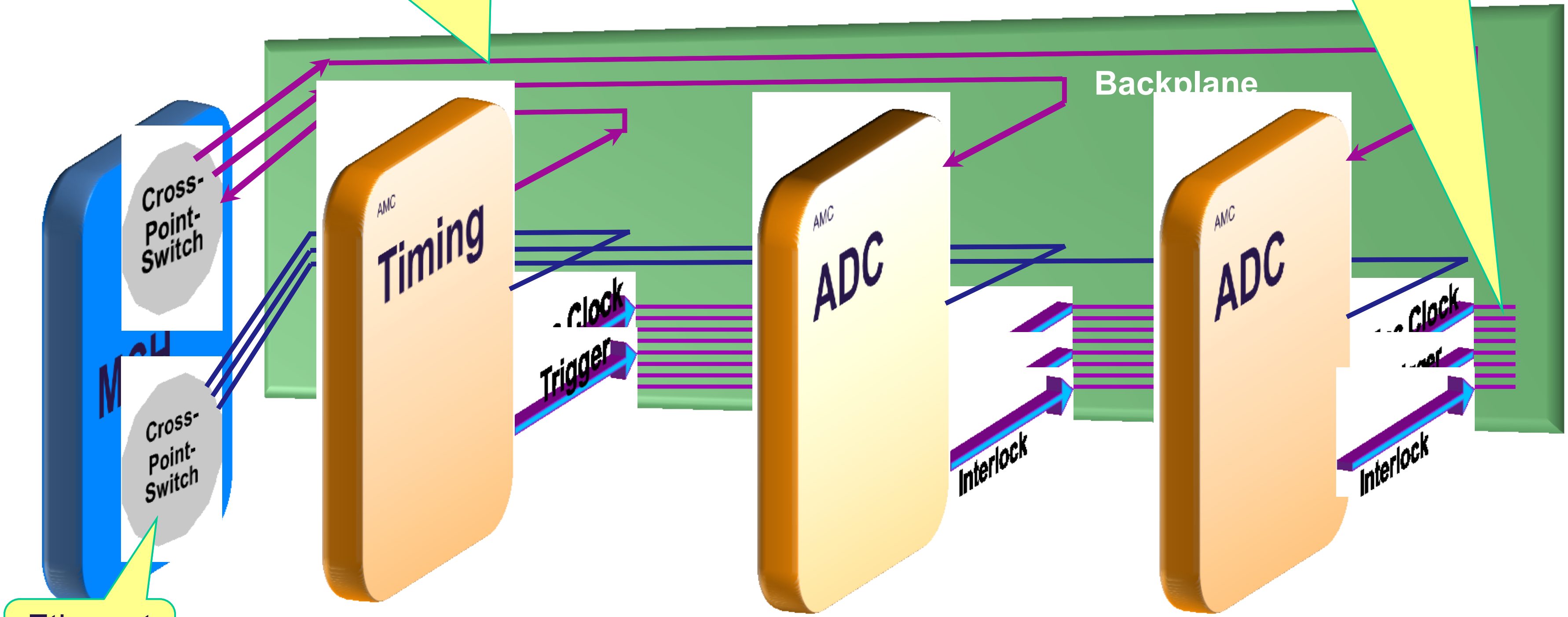


DESY DAMC1

Introduction into MicroTCA – In-crate Timing

2 radial clocks per AMC,
Low jitter, configurable direction

8 bussed M-LVDS lines,
For triggers, clocks and interlocks

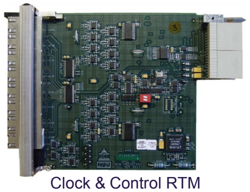


Ethernet
PCIe

Curtsey: Kay Rehlich (DESY)

Introduction into MicroTCA – as main platform

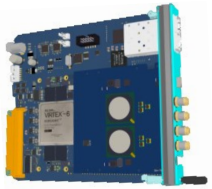
- MicroTCA.4 allows
  - High-bandwidth communication between
    - Boards and CPU via PCIe
    - Boards via point-to-point connections
  - Synchronization via Timing Receiver
    - Trigger
    - Clocks
    - Machine parameters
    - Bunch structure
  - Remote control and monitoring
  - Module changes during operation (Hotswap)
  - Functional extension via RTMs



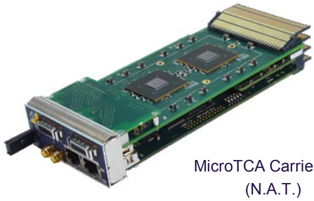
Clock & Control RTM
(University Collage London)

DAMC2 – Universal
digital AMC
(DESY)

Timing System
(Stockholm University
/ DESY)



High-Speed Digitizer Family
(Signal Processing Devices Sweden AB)



MicroTCA Carrier Hub
(N.A.T.)

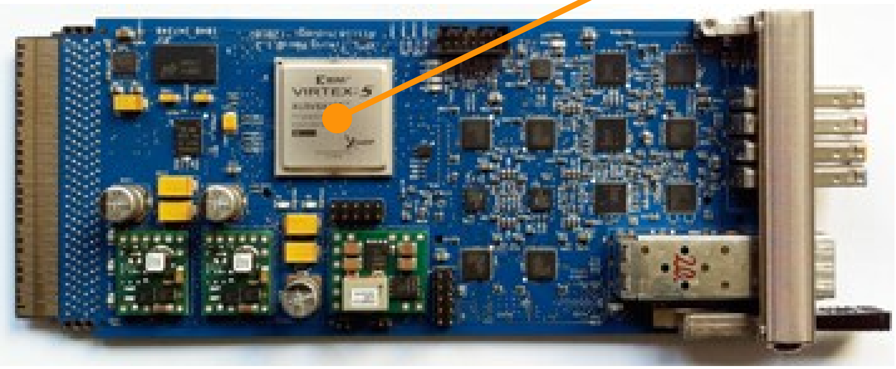
SIS8300 – ADC AMC
(Struck Innovative Systems)

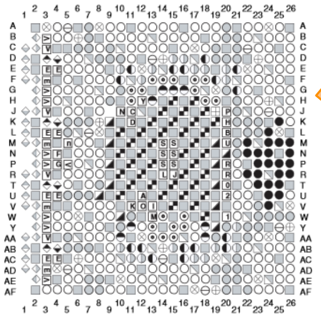
High-performance
DSP and FPGA board
(DMCS/DESY)

If you are interested in this topic, please register for the MicroTCA Workshop in December @ DESY: http://mtcaws.desy.de

Brief introduction into FPGAs

- Field Programmable Gate Array (FPGA)
  - Allow hundreds of thousand logic functions
  - All in parallel (if required)
  - Reprogrammable by the user





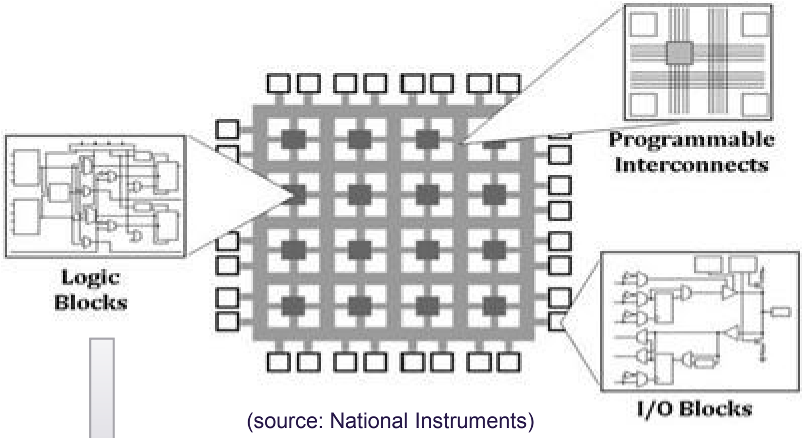
“Only” 665 pins.
Largest one has 1760 pins

(source: Xilinx)

Simplified Configurable Logic Block (CLB)

Brief introduction into FPGAs

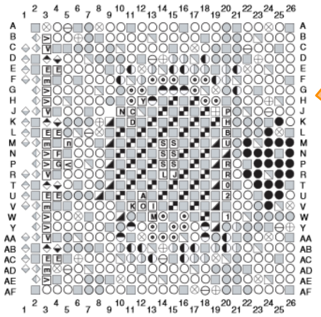
- Field Programmable Gate Array (FPGA)
  - Flexible inner structure



(source: National Instruments)

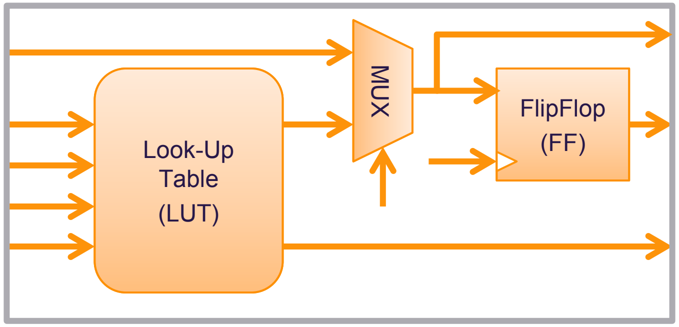
Not shown here:

- RAM
- Clock Modules
- Gigabit Transceivers for
  - PCIe
  - 10GbE
- DDR interfaces
- CPUs
- DSPs



“Only” 665 pins.
Largest one has 1760 pins

(source: Xilinx)

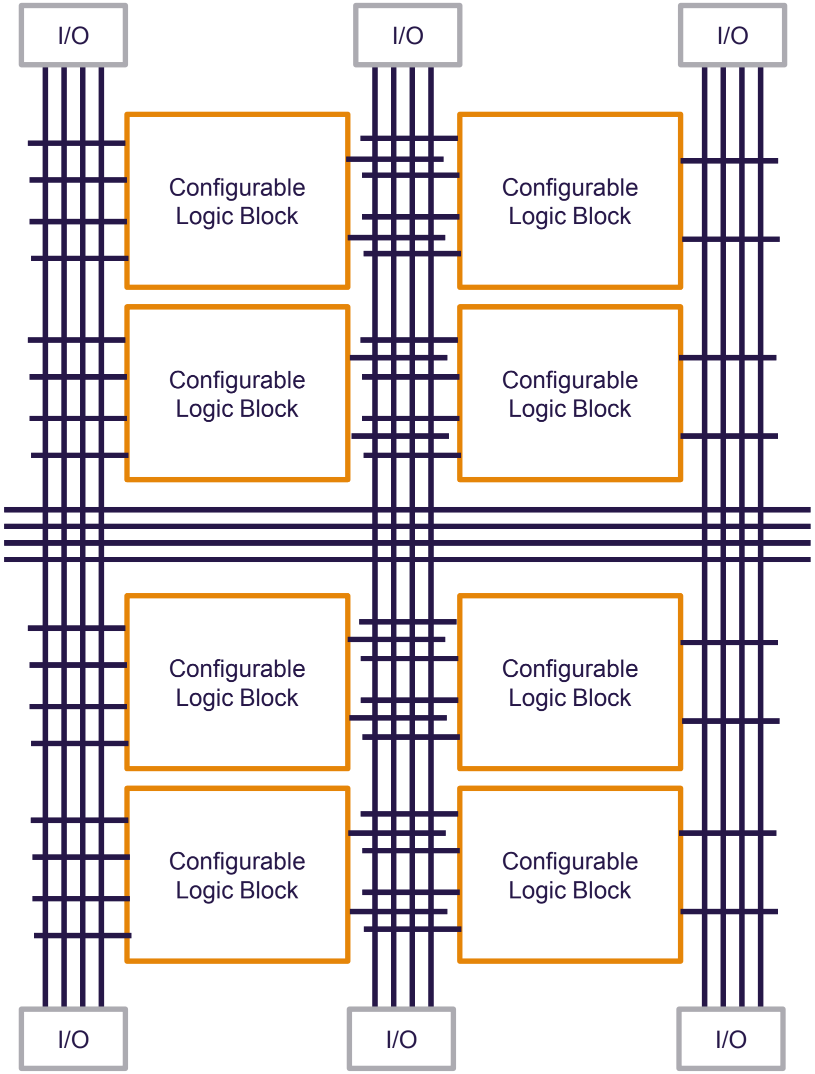


Simplified Configurable Logic Block (CLB)

A	B	AND	OR	NAND	XOR
0	0	0	0	1	0
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0

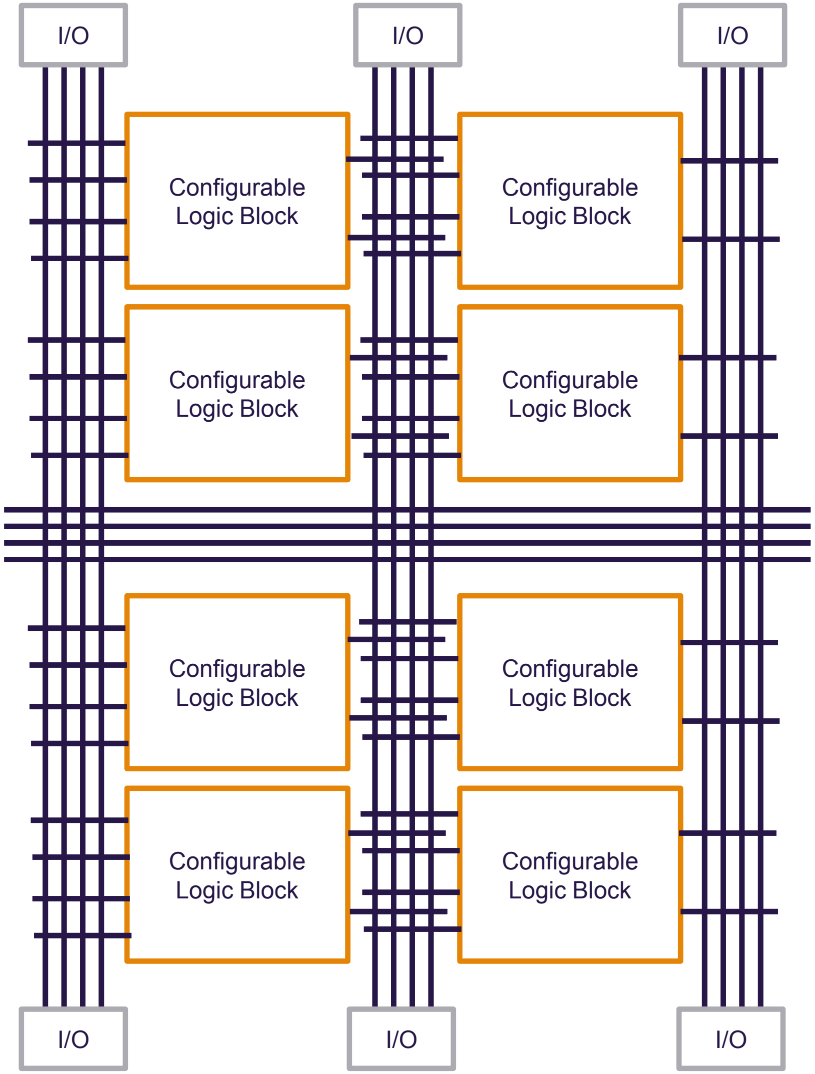
Look-Up Table (LUT) examples with 2 inputs only

Brief Introduction into FPGAs - Example

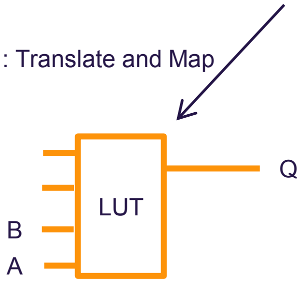


Defined Program: **Q = A and B**

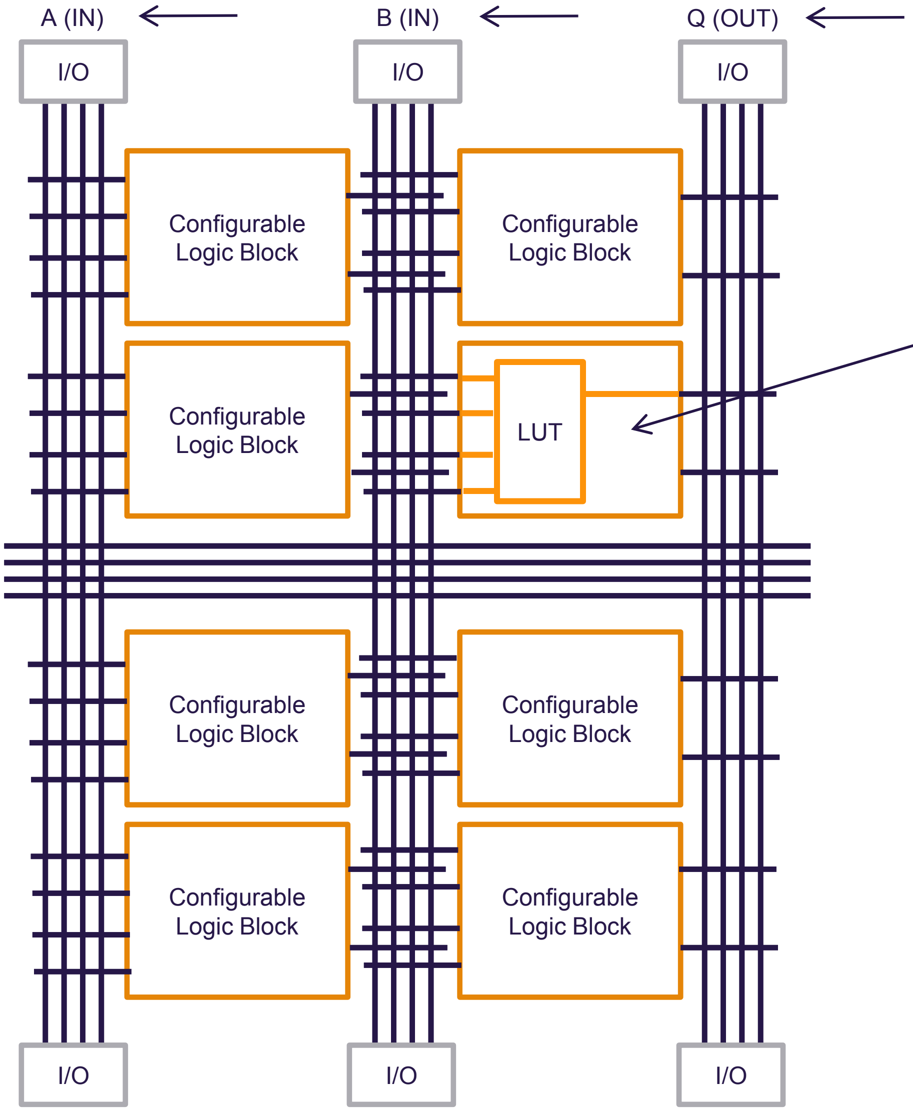
Brief Introduction into FPGAs - Example



Defined Program: **Q = A and B**

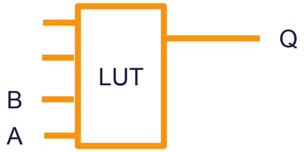


Brief Introduction into FPGAs - Example



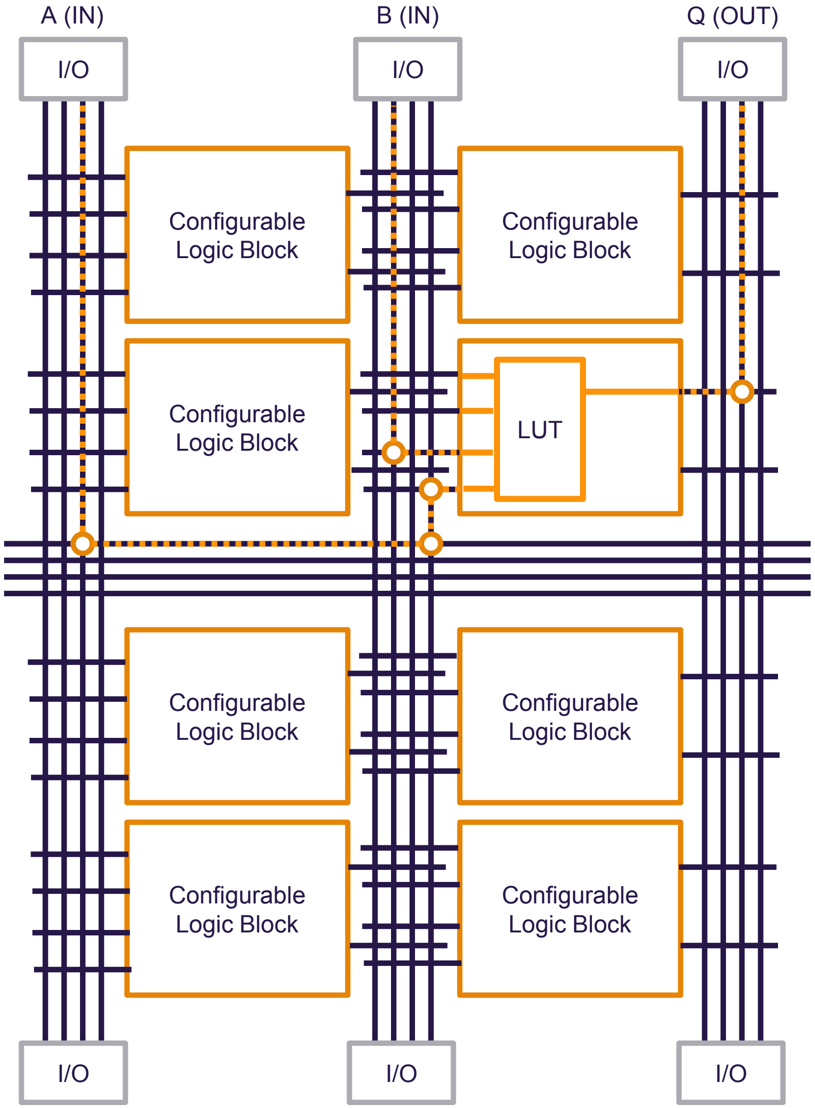
Defined Program: **Q = A and B**

Step 1: Translate and Map



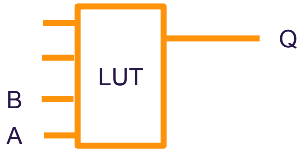
Step 2: Place

Brief Introduction into FPGAs - Example



Defined Program: **Q = A and B**

Step 1: Translate and Map



Step 2: Place

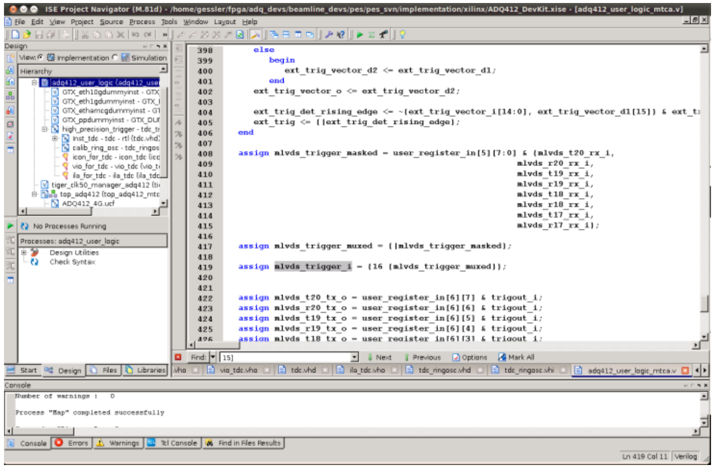
Step 3: Route

Main tasks and required features of FPGAs

- Receive data from
  - Integrated circuits (e.g. ADCs)
  - Other modules via communication interfaces
- Synchronize data flow (Triggers, clocks)
- (Online) Processing of data (with run-time parameters)
  - like integration, peak detection, TDC, FFT, etc
- (Re-)Formatting and coding of data
  - like 10Gb Ethernet
- Transmission of data (e.g. high-bandwidth, low-latency)
- Monitoring, debugging and diagnostics

VHDL Programming basic example

```
entity myAND is
        port( A: in std_logic;
              B: in std_logic;
              Q: out std_logic
        );
end myAND;


architecture RTL of myAND is
begin
        Q <= A and B;
end;
```

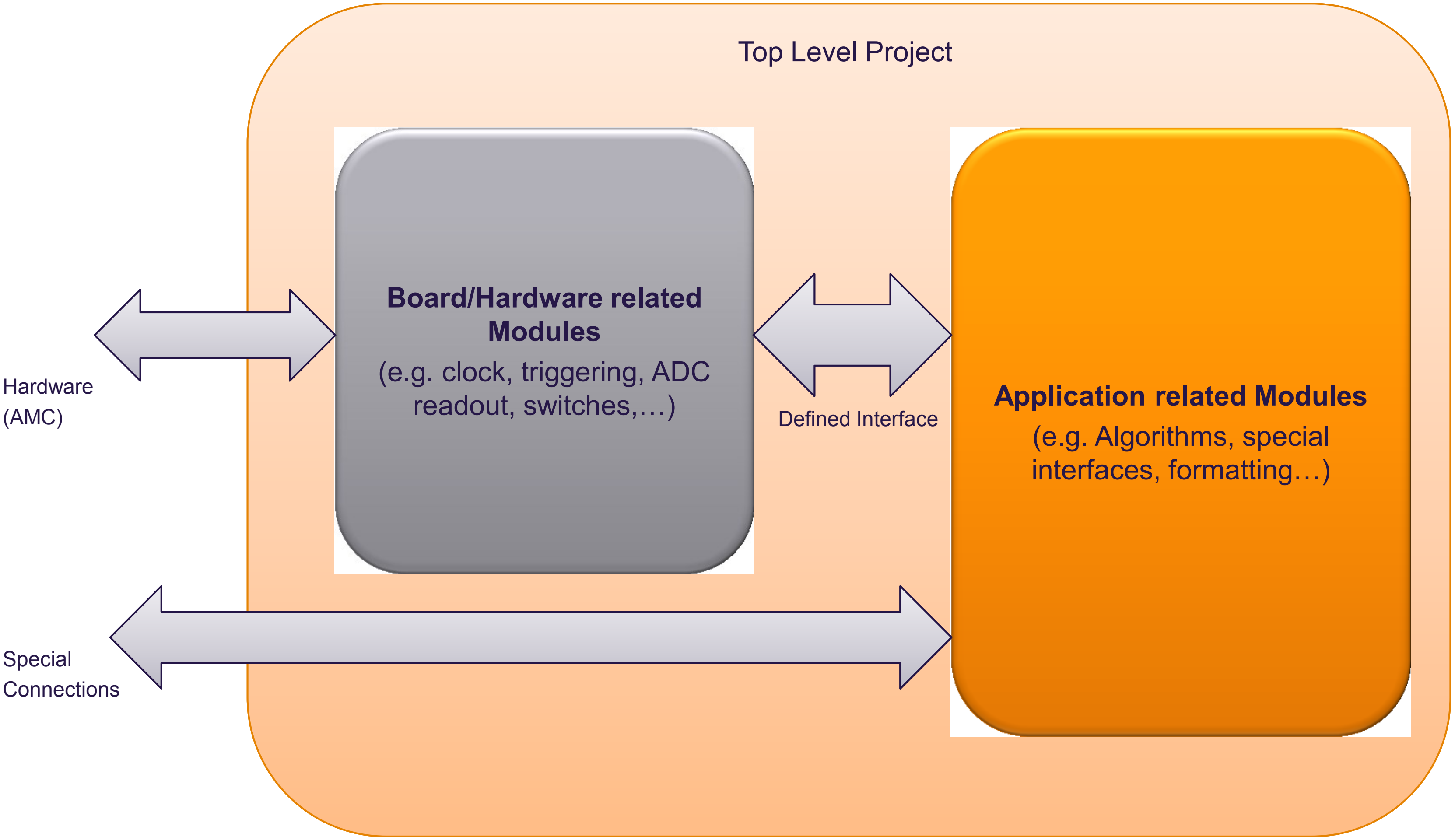

ISE Project Navigator (M.81d) - /home/gessler/fpga/adq_devs/beamline_devs/pes/pes_svn/implementation/xilinx/ADQ412_DevKit.xise - [adq412_user_logic_mtca.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
adq412_user_logic (adq412_user
GTX_eth10gdummyinst - GTX
GTX_eth1gdummyinst - GTX_I
GTX_ethamcgdummyinst - GT
GTX_ppdummyinst - GTX_DUI
high_precision_trigger - tdc_tr
inst_tdc - tdc - rtl (tdc.vhd)
calib_ring_osc - tdc_ringos
icon_for_tdc - icon_tdc (icc
vio_for_tdc - vio_tdc (vio_t
ila_for_tdc - ila_tdc (ila_tdc
tiger_clk50_manager_adq412 (ti
top_adq412 (top_adq412_mtc
ADQ412_4G.ucf
No Processes Running
Processes: adq412_user_logic
Design Utilities
Check Syntax
Start
Design
Files
Libraries
Find:
15]
Next
Previous
Options
Mark All
.vho
vio_tdc.vho
tdc.vhd
ila_tdc.vho
tdc_ringosc.vhd
tdc_ringosc.vhi
adq412_user_logic_mtca.v
Console
Errors
Warnings
Tcl Console
Find in Files Results
Ln 419 Col 11
Verilog

```
      else
         begin
            ext_trig_vector_d2 <= ext_trig_vector_d1;
         end
      ext_trig_vector_o <= ext_trig_vector_d2;

      ext_trig_det_rising_edge <= ~{ext_trig_vector_i[14:0], ext_trig_vector_d1[15]} & ext_t
      ext_trig <= {|ext_trig_det_rising_edge};
   end

   assign mlvds_trigger_masked = user_register_in[5][7:0] & {mlvds_t20_rx_i,
                                                             mlvds_r20_rx_i,
                                                             mlvds_t19_rx_i,
                                                             mlvds_r19_rx_i,
                                                             mlvds_t18_rx_i,
                                                             mlvds_r18_rx_i,
                                                             mlvds_t17_rx_i,
                                                             mlvds_r17_rx_i};

   assign mlvds_trigger_muxed = {|mlvds_trigger_masked};

   assign mlvds_trigger_i = {16 {mlvds_trigger_muxed}};


   assign mlvds_t20_tx_o = user_register_in[6][7] & trigout_i;
   assign mlvds_r20_tx_o = user_register_in[6][6] & trigout_i;
   assign mlvds_t19_tx_o = user_register_in[6][5] & trigout_i;
   assign mlvds_r19_tx_o = user_register_in[6][4] & trigout_i;
   assign mlvds_t18_tx_o = user_register_in[6][3] & trigout_i;
```

```
Number of warnings :   0

Process "Map" completed successfully
```

Xilinx ISE is one tool for VHDL Programming for the FPGA

Providing Structure: Start-Up Projects



Top Level Project

Board/Hardware related Modules

(e.g. clock, triggering, ADC readout, switches,…)

Application related Modules

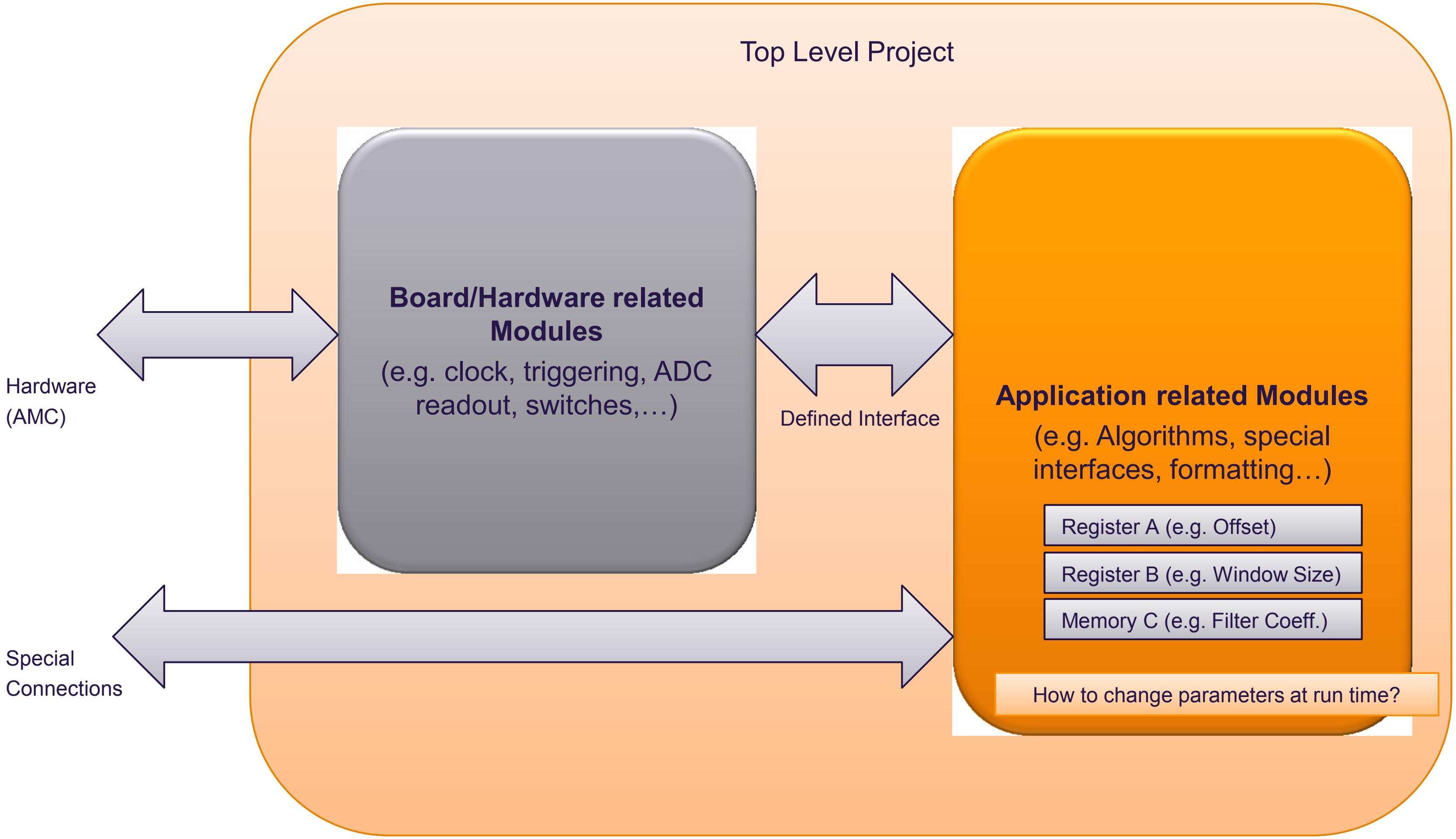
(e.g. Algorithms, special interfaces, formatting…)

Defined Interface

Hardware (AMC)

Special Connections

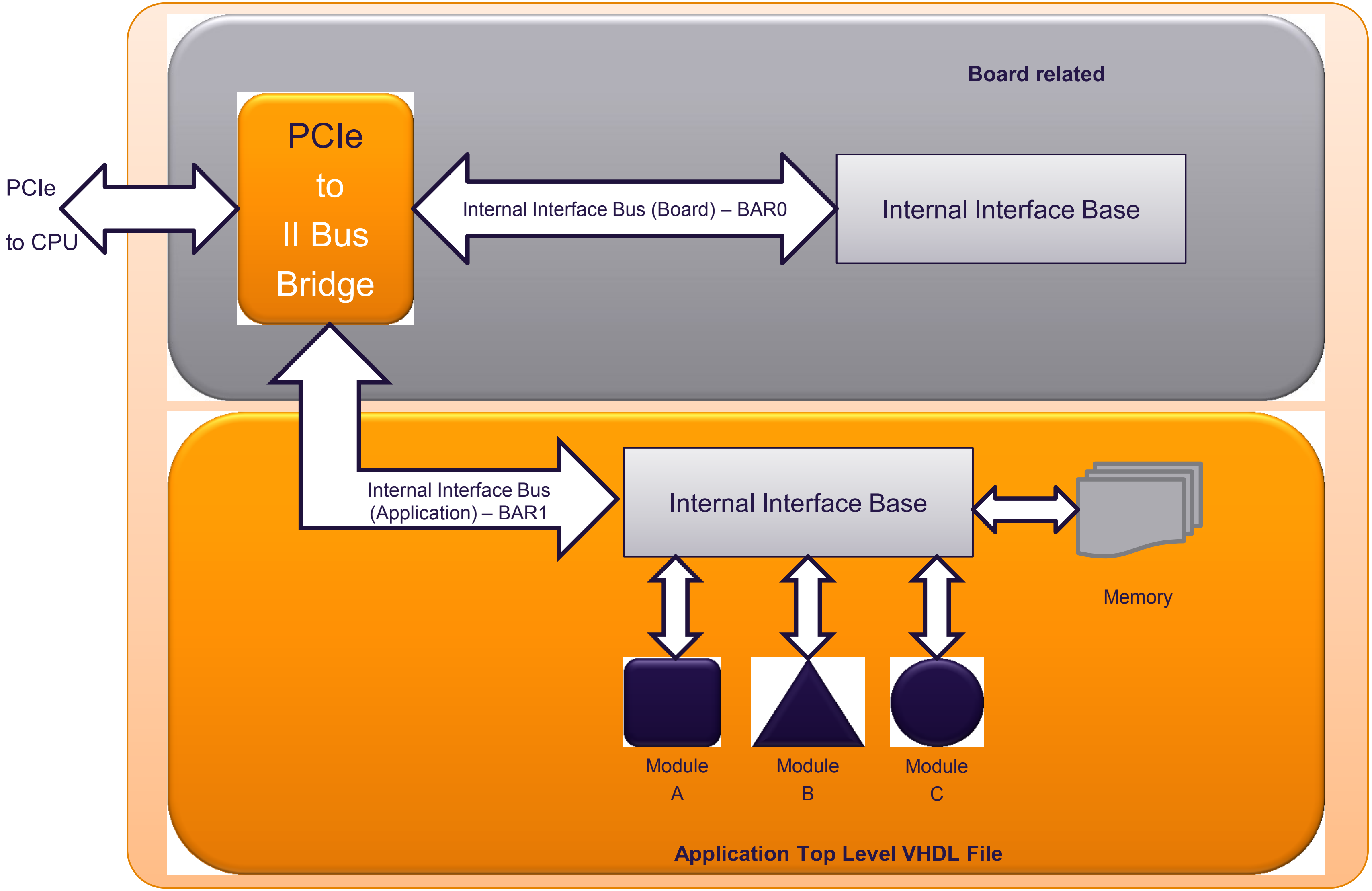
Providing Structure: Start-Up Projects



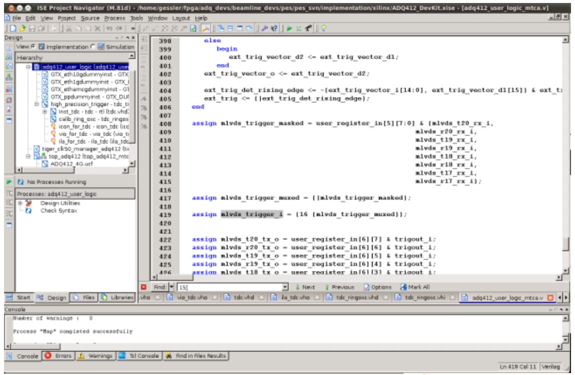
Top Level Project
Hardware (AMC)
Board/Hardware related Modules
(e.g. clock, triggering, ADC readout, switches,…)
Defined Interface
Application related Modules
(e.g. Algorithms, special interfaces, formatting…)
Register A (e.g. Offset)
Register B (e.g. Window Size)
Memory C (e.g. Filter Coeff.)
Special Connections

How to change parameters at run time?

Providing Computer Connection: PCIe / II Bus



Workflow example



ISE Project Navigator (M.81d) - /home/gessler/fpga/adq_devs/beamline_devs/pes/pes_svn/implementation/xilinx/ADQ412_DevKit.xise - [adq412_user_logic_mtca.v]

```
      else
         begin
            ext_trig_vector_d2 <= ext_trig_vector_d1;
         end
      ext_trig_vector_o <= ext_trig_vector_d2;

      ext_trig_det_rising_edge <= ~{ext_trig_vector_i[14:0], ext_trig_vector_d1[15]} & ext_t
      ext_trig <= {|ext_trig_det_rising_edge};
   end

   assign mlvds_trigger_masked = user_register_in[5][7:0] & {mlvds_t20_rx_i,
                                                              mlvds_r20_rx_i,
                                                              mlvds_t19_rx_i,
                                                              mlvds_r19_rx_i,
                                                              mlvds_t18_rx_i,
                                                              mlvds_r18_rx_i,
                                                              mlvds_t17_rx_i,
                                                              mlvds_r17_rx_i};

   assign mlvds_trigger_muxed = {|mlvds_trigger_masked};

   assign mlvds_trigger_i = {16 {mlvds_trigger_muxed}};


   assign mlvds_t20_tx_o = user_register_in[6][7] & trigout_i;
   assign mlvds_r20_tx_o = user_register_in[6][6] & trigout_i;
   assign mlvds_t19_tx_o = user_register_in[6][5] & trigout_i;
   assign mlvds_r19_tx_o = user_register_in[6][4] & trigout_i;
   assign mlvds_t18_tx_o = user_register_in[6][3] & trigout_i;
```

```
Number of warnings : 0

Process "Map" completed successfully
```

ISE DESIGN SUITE

Firmware Image

Programming Path

Communication Path
(e.g. PCIe, Ethernet,..)

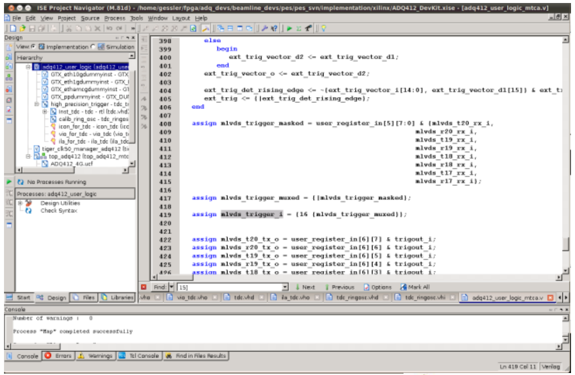
XML FILES

User Registers and Memories Description

Control System Software

This system allows **automatic** detection of installed firmware and loads **correct** XML files with Register und Memory descriptions

Workflow example (with remote f/w upload)



```
    else
      begin
        ext_trig_vector_d2 <= ext_trig_vector_d1;
      end
    ext_trig_vector_o <= ext_trig_vector_d2;

    ext_trig_det_rising_edge <= ~{ext_trig_vector_i[14:0], ext_trig_vector_d1[15]} & ext_t
    ext_trig <= {|ext_trig_det_rising_edge};
  end

  assign mlvds_trigger_masked = user_register_in[5][7:0] & {mlvds_t20_rx_i,
                                                            mlvds_r20_rx_i,
                                                            mlvds_t19_rx_i,
                                                            mlvds_r19_rx_i,
                                                            mlvds_t18_rx_i,
                                                            mlvds_r18_rx_i,
                                                            mlvds_t17_rx_i,
                                                            mlvds_r17_rx_i};

  assign mlvds_trigger_muxed = {|mlvds_trigger_masked};

  assign mlvds_trigger_i = {16 {mlvds_trigger_muxed}};


  assign mlvds_t20_tx_o = user_register_in[6][7] & trigout_i;
  assign mlvds_r20_tx_o = user_register_in[6][6] & trigout_i;
  assign mlvds_t19_tx_o = user_register_in[6][5] & trigout_i;
  assign mlvds_r19_tx_o = user_register_in[6][4] & trigout_i;
  assign mlvds_t18_tx_o = user_register_in[6][3] & trigout_i;
```

```
Number of warnings : 0

Process "Map" completed successfully
```

ISE Project Navigator (M.81d) - /home/gessler/fpga/adq_devs/beamline_devs/pes/pes_svn/implementation/xilinx/ADQ412_DevKit.xise - [adq412_user_logic_mtca.v]
File Edit View Project Source Process Tools Window Layout Help
Implementation
Simulation
Find: 15]
Next
Previous
Options
Mark All
vio_tdc.vho
tdc.vhd
ila_tdc.vho
tdc_ringosc.vhd
tdc_ringosc.vhi
adq412_user_logic_mtca.v
Console
Errors
Warnings
Tcl Console
Find in Files Results

ISE DESIGN SUITE

Firmware Image

Communication Path (e.g. PCIe, Ethernet,..)

XML FILES

User Registers and Memories Description

Control System Software

It also allows centralized remote firmware management (e.g. version tracking and remote upload synchronized with s/w updates)

High Level FPGA Algorithm Development

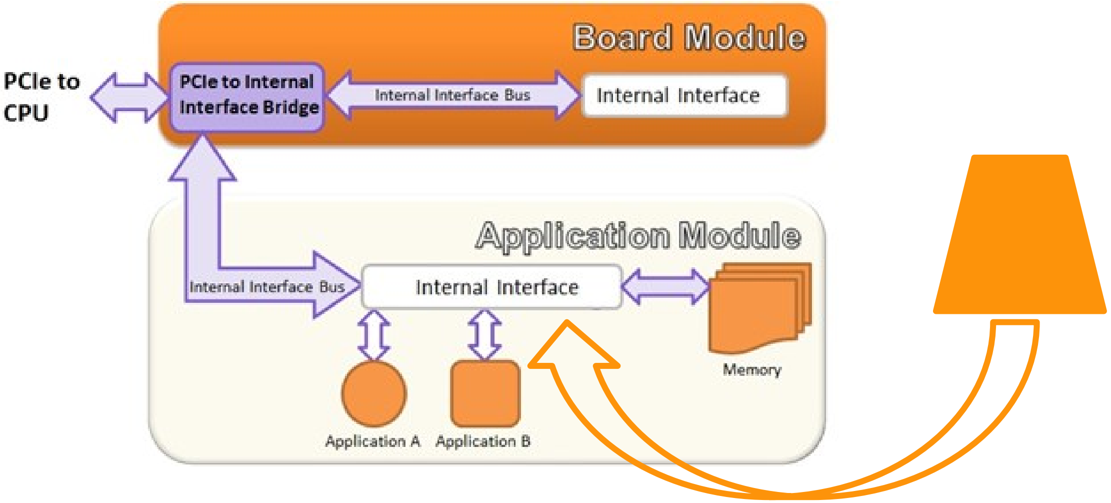
The FPGA should bend time and space according to

$$R_{\mu\nu} - \frac{1}{2} R\, g_{\mu\nu} + \Lambda\, g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$$

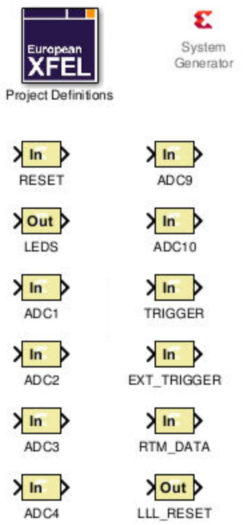
- FPGA programming is time intensive and requires specialists, however **most applications and algorithms are conceptualized by users unfamiliar with FPGA programming**.
- For the European XFEL, a high level FPGA framework is being developed that allows for users with no prior HDL knowledge to develop their algorithm modules which can be integrated in a top VHDL project.

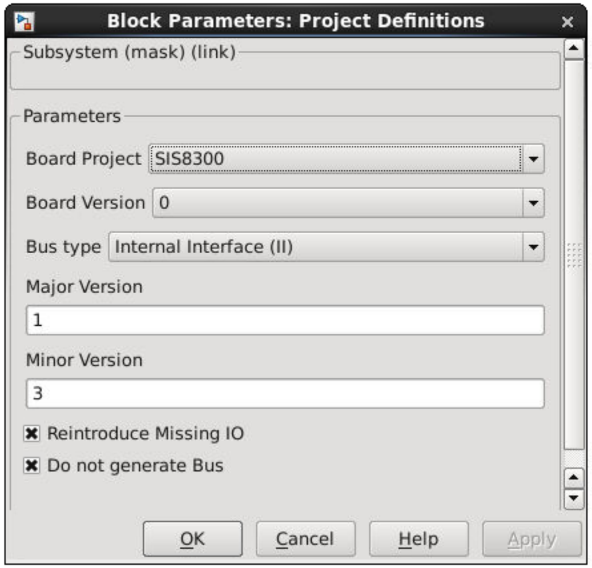
High Level FPGA Algorithm Development

- The Framework should:
  - Abstract end user from hardware programming languages and concepts such as Pin placement, clock routing, etc.
  - User design environment automatically setup for the target board;
  - Library with blocks that simulate the behavior of available features;
  - User defined registers and memories and generate the necessary logic to later communicate with the bus protocol;
  - Easy to port and distribute applications to other projects/boards.



XFEL Simulink Library – Project Definitions




Block Parameters: Project Definitions

Subsystem (mask) (link)

Parameters

Board Project: SIS8300

Board Version: 0

Bus type: Internal Interface (II)

Major Version: 1

Minor Version: 3

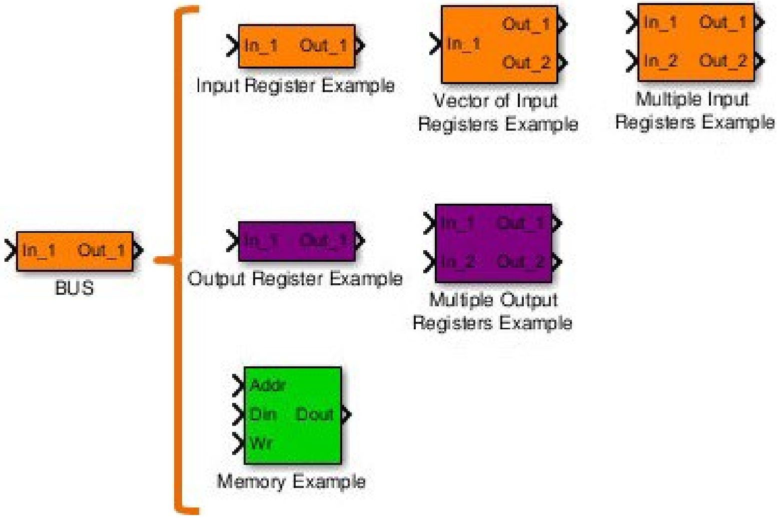
☒ Reintroduce Missing IO

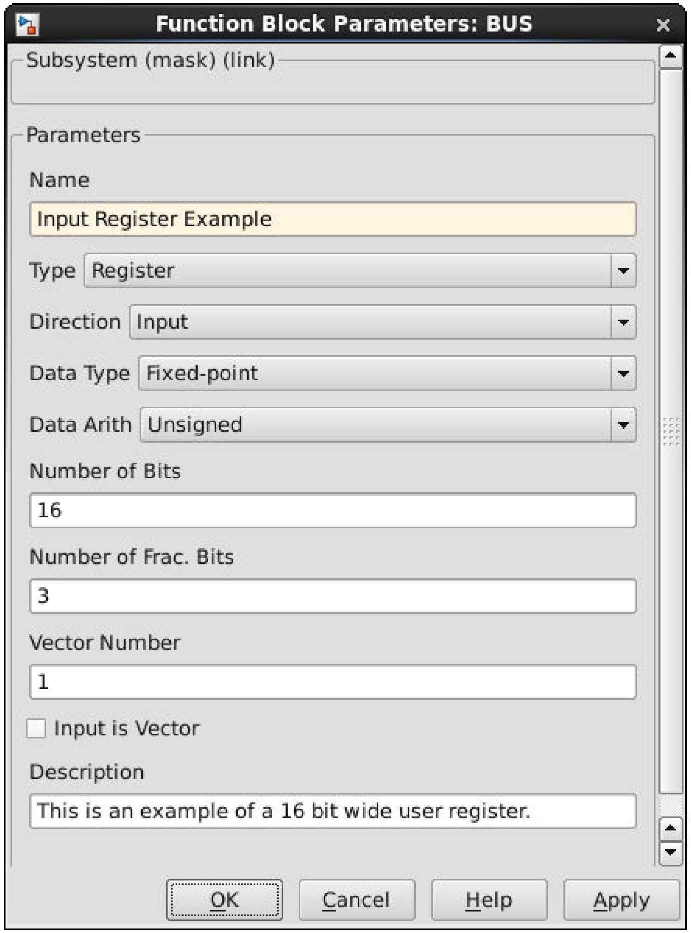
☒ Do not generate Bus

- The Project Definitions block: the user defines for which board the algorithm is going to be develop as well as the bus protocol:
  - Generate the IO available for the chosen board and examples of expected input signals;
  - Includes the System generator block and defines its parameters according to the FPGA board.

XFEL Simulink Library – BUS Block

- BUS block allows for users to define registers and memories which are later accessible by the chosen protocol




Function Block Parameters: BUS

Subsystem (mask) (link)

Parameters

Name: Input Register Example

Type: Register

Direction: Input

Data Type: Fixed-point

Data Arith: Unsigned

Number of Bits: 16

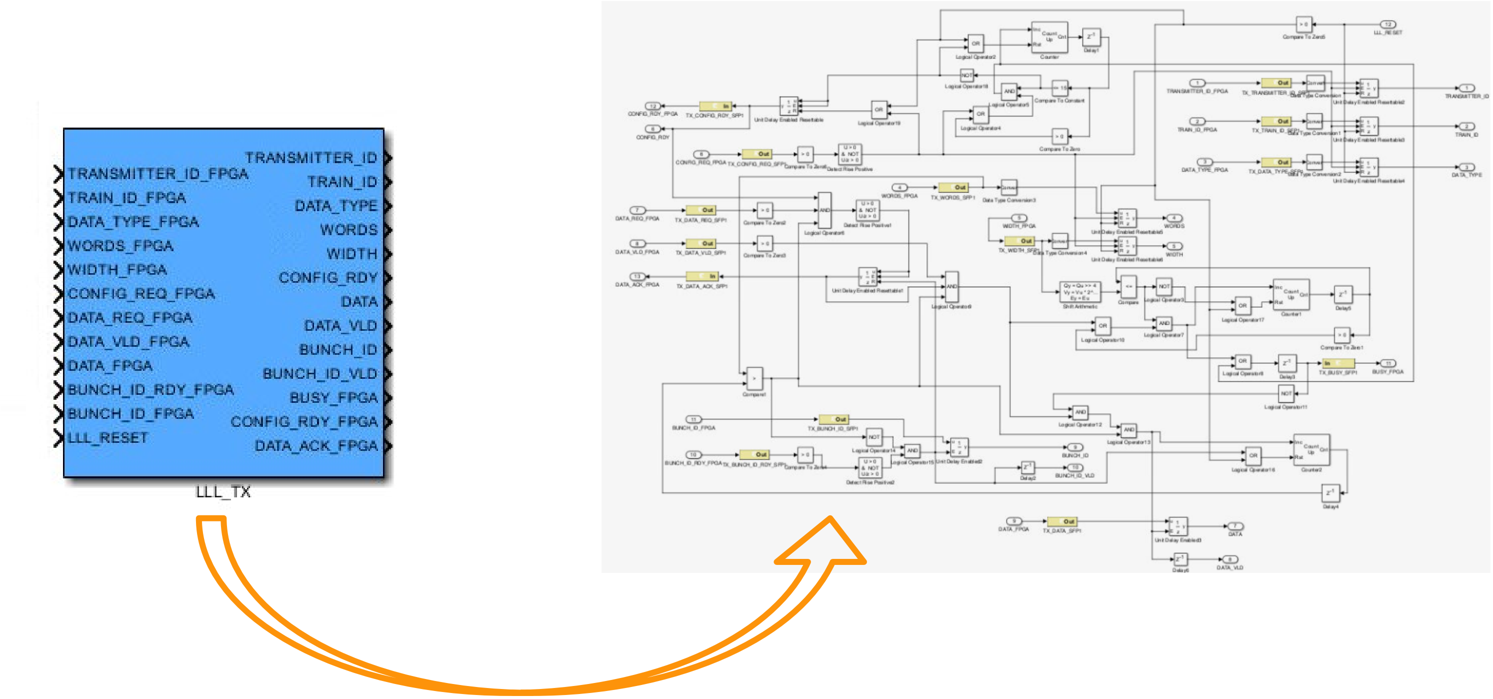
Number of Frac. Bits: 3

Vector Number: 1

□ Input is Vector

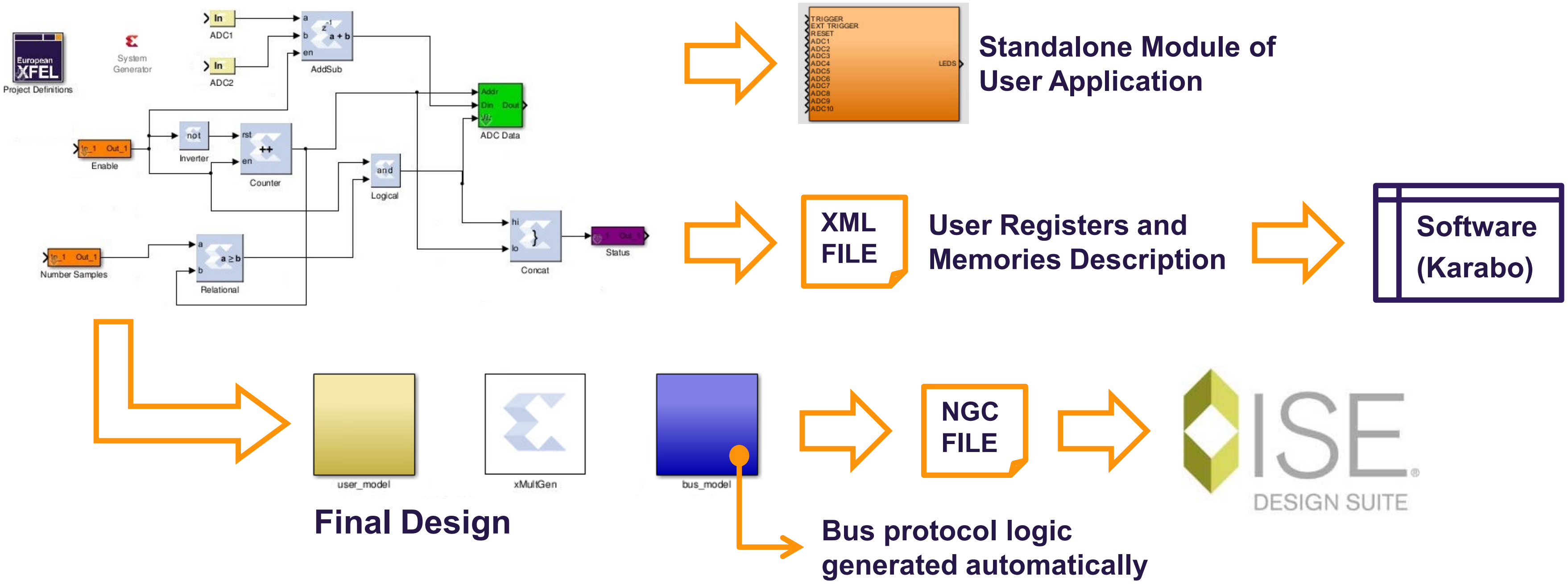
Description: This is an example of a 16 bit wide user register.

XFEL Simulink Library – Board Specific block



- Specific blocks are available on the library which will accurately simulate the behavior of features available on the board;
- In Hardware, the block is replaced with the real implementation.

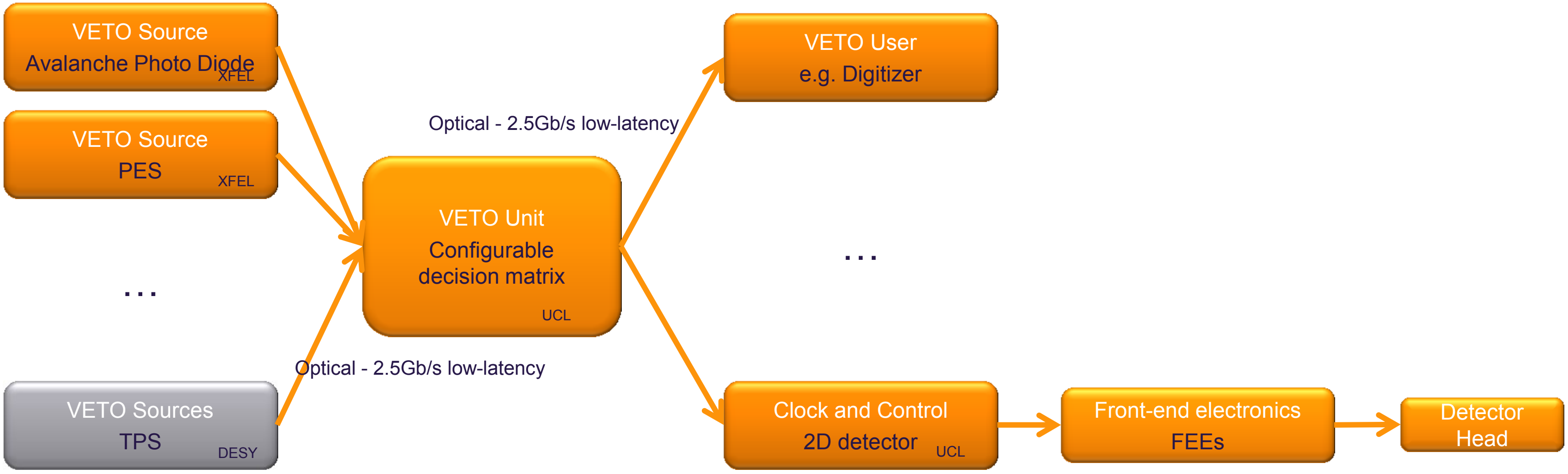
XFEL Simulink Library – Application Integration



- The final Algorithm is processed to integrate the top level FPGA Project:
  - Generate Final Design with bus interface logic based on defined user registers and chosen protocol;
  - Netlist file of final design inserted in ISE project;
  - Standalone Module of User application to share and distribute
  - XML file with register information similar to the VHDL developed applications.

Low-Latency data transmission: VETO System for data reduction and memory optimization

- Optimize picture quality of 2D detectors
  - Limited frame capacity in ASICs (~300-700 frames)
  - Replace bad frames with new ones in ASIC before read out and transmission
- Data reduction
  - Discard useless data before transmission
- Implementation
  - FPGAs of diagnostics and detectors provide bunch information with low-latency
  - Configurable central VETO unit per experiment decides on bunch quality
  - FPGAs of detectors (maybe also diagnostics) receive the decision and react on it
  - Using a common protocol with beam based feedback system

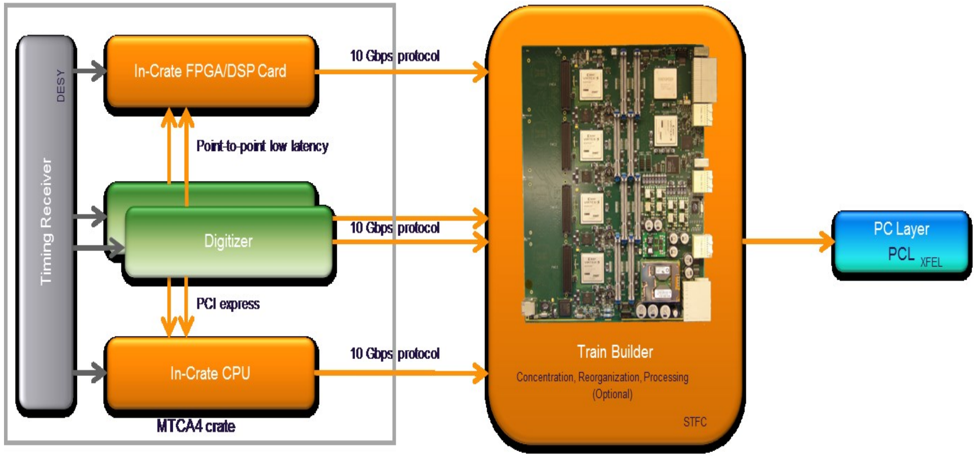


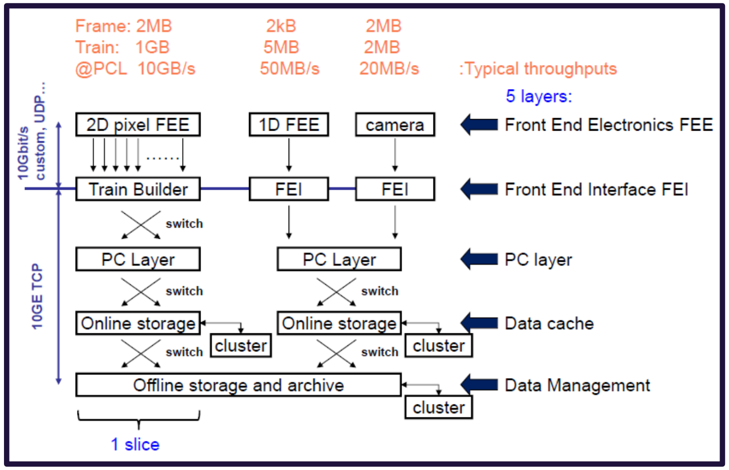
High-Bandwidth data transmission
10Gb Ethernet data streaming

- Driving force: fast and large area detectors

Detector type	Sampling	Data/pulse	Data/train	XFEL/sec	LCLS/sec
1 Mpxl 2D camera	4.5 MHz	~2 MB	~1 GB	~10 GB	~300 MB
1 channel digitizer	5 GS/s	~2 kB	~6 MB	~60 MB	~0.2 MB

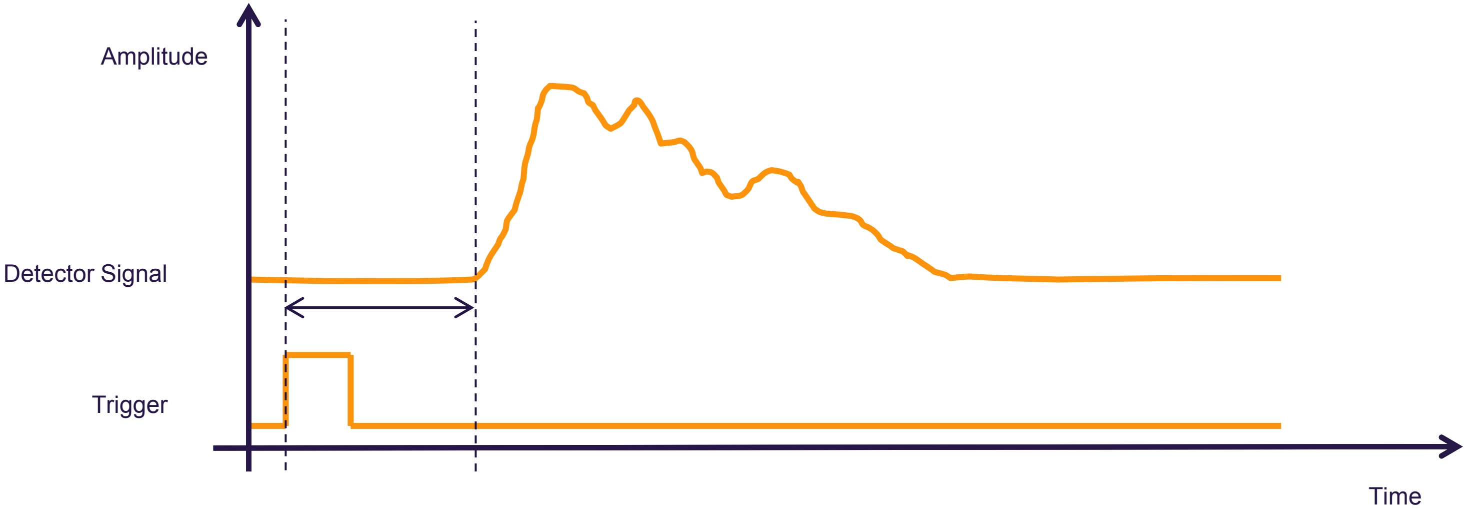
- Chosen architecture:





High Resolution Trigger

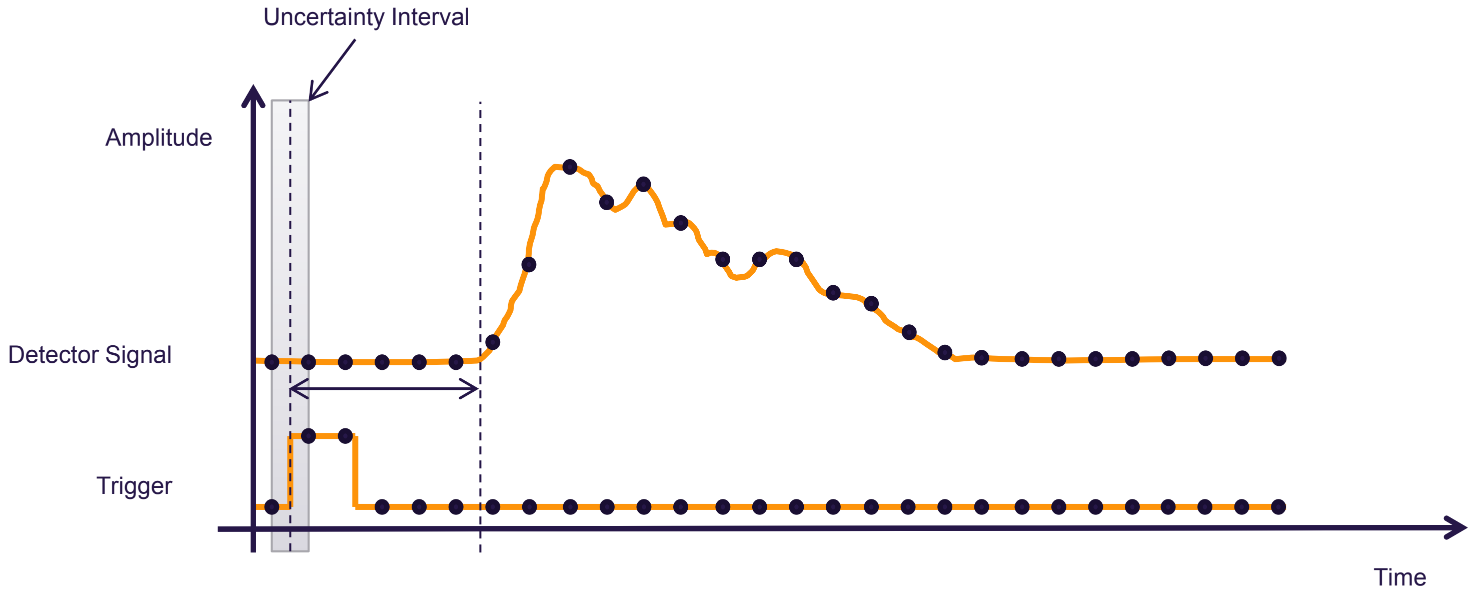
- Triggering Basics:
  - “A trigger defines the point in time relative to an event (e.g. detector signal) to be observed”



  - The trigger is used to start data acquisition and processing

High Resolution Trigger

- In the digital world:
  - Signal and trigger are quantized in discrete time steps



  - The trigger time is only known at the step size of the sampling time of the trigger

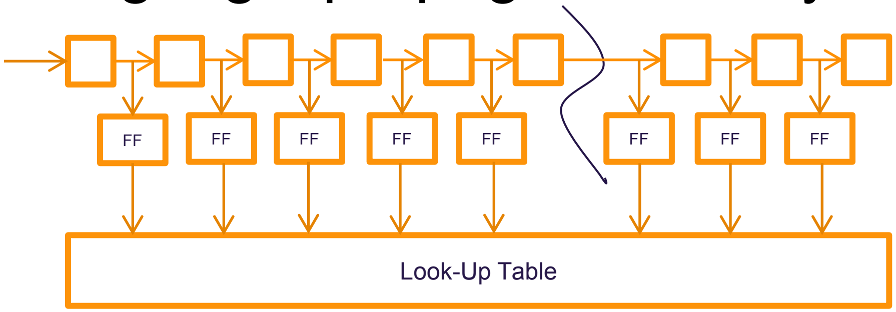
Comment: If the trigger has a constant time to the following sampling clock edge and therefore also to the signal, the problem is not there anymore.

High Resolution Trigger

- In many applications the signal sampling clock is much higher than the trigger sampling clock (factor > 8)
  - In this case the uncertainty is higher than the signal sampling accuracy!
- Solutions:
  - Synchronize clock sampling phase to experiments reference
    - → Not always possible (due to internal PLLs)
    - → Increases noise through external clock
  - Increase sampling speed for trigger
    - → Usual maximum in FPGAs: between 1 and 2.5 GBPS
    - → Using Time-to-Digitial converter (TDC) technology

High Resolution Trigger

- Different TDC implementations in FPGA possible
- The currently under implementation at XFEL:
  - Using logic propagation delays



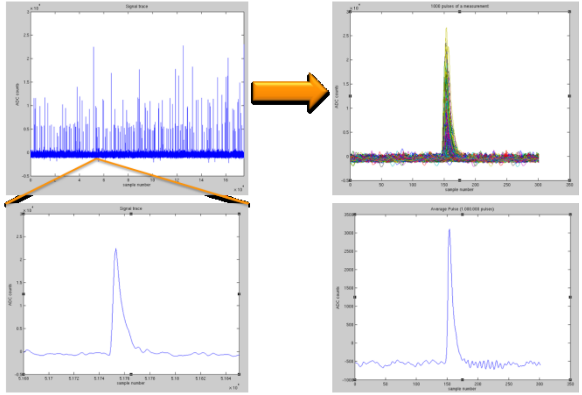
➔ 0000000011111111...1111111

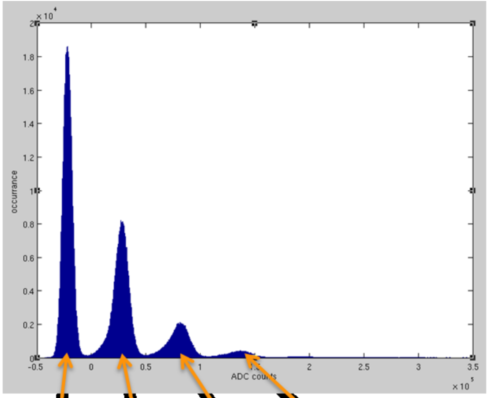
➔ 15

- Challenges:
  - complicated timing relations in FPGA
  - Calibration and temperature induced drift compensation
- Expected resolution: down to 50ps

Pulse Energy Detetction

- Different possibilities tested
  - Integration (summing of samples) and
  - Cross-Correlation (Matched filtering)



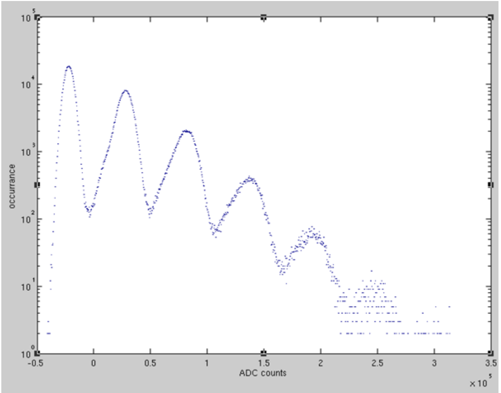


Baseline

1 Photon

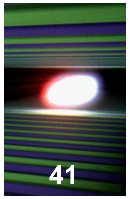
2 Photons

3 Photons



Tests done at Petra III with XFEL FXE, P01 and P11 groups

Conclusion



- Analog front end electronics are required in order to
  - Convert the charges of a sensor to be detected
  - Shape the signal for the digitalization process
  - Amplify and filter the signal to optimize signal-to-noise ratio
- A transition from VME to MicroTCA for the XFEL.EU was required
  - To increase the availability (reduce downtime)
  - Allow for higher data bandwidths for faster digitizers and detectors
- The development of FPGAs
  - Allows for high flexibility and also includes high complexity
  - Many effort had been done on simplify the work to
    - Speed up the development time
    - Reduce errors
    - Improve maintainability
  - Key algorithms are under development or test
  - Non experienced people can do FPGA programming in Simulink