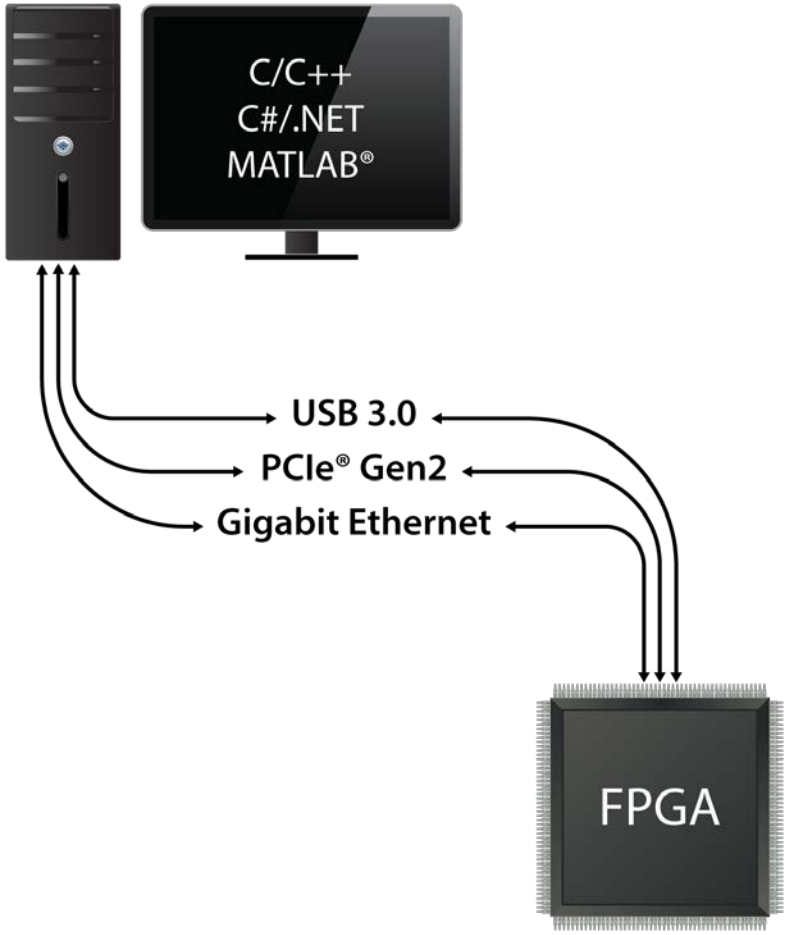
Universal connection between an FPGA and a high level programming language

Streaming, made simple.



Visit us at our booth

Agenda

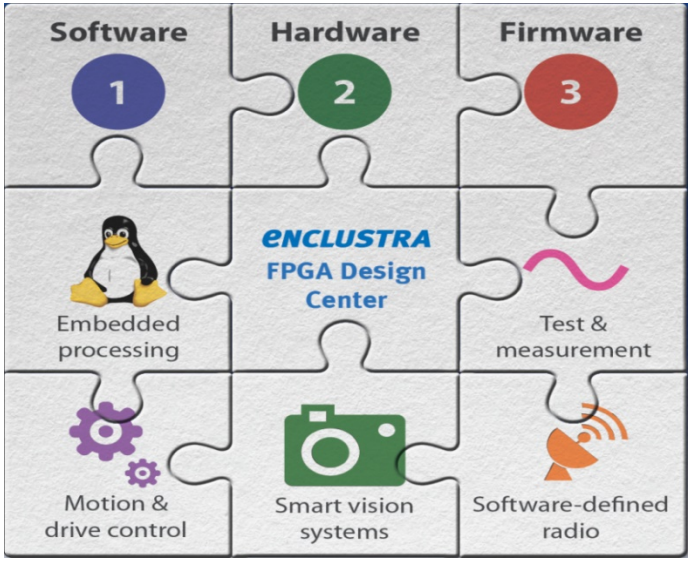
- **Enclustra company profile**
- Reasons for linking an FPGA to a high level language
- Requirements when linking an FPGA to a high level language
- Challenges when linking an FPGA to a high level language
- Why a standard solution makes sense
- FPGA Manager overview

Enclustra company profile

- *Focused on FPGA Technology – Everything FPGA!*
- *Founded in 2004 – successfully in business for 14 years!*
- *Headquarters in Zürich, Switzerland*
- *Branch offices in Germany, USA and China*
- *2 Business units: FPGA Design Center, FPGA Solution Center*
- *30 employees: 15 FPGA engineers plus 11 staff in Zurich, 4 employees abroad*
- *Vendor-Independent*

Enclustra company profile

- FPGA Design Center
  - Hardware (High-Speed, Analog, RF)
  - HDL firmware (VHDL, Verilog)
  - Embedded software (for FPGA processors)
- FPGA Solution Center
  - IP-Cores
  - FPGA & SoC Modules



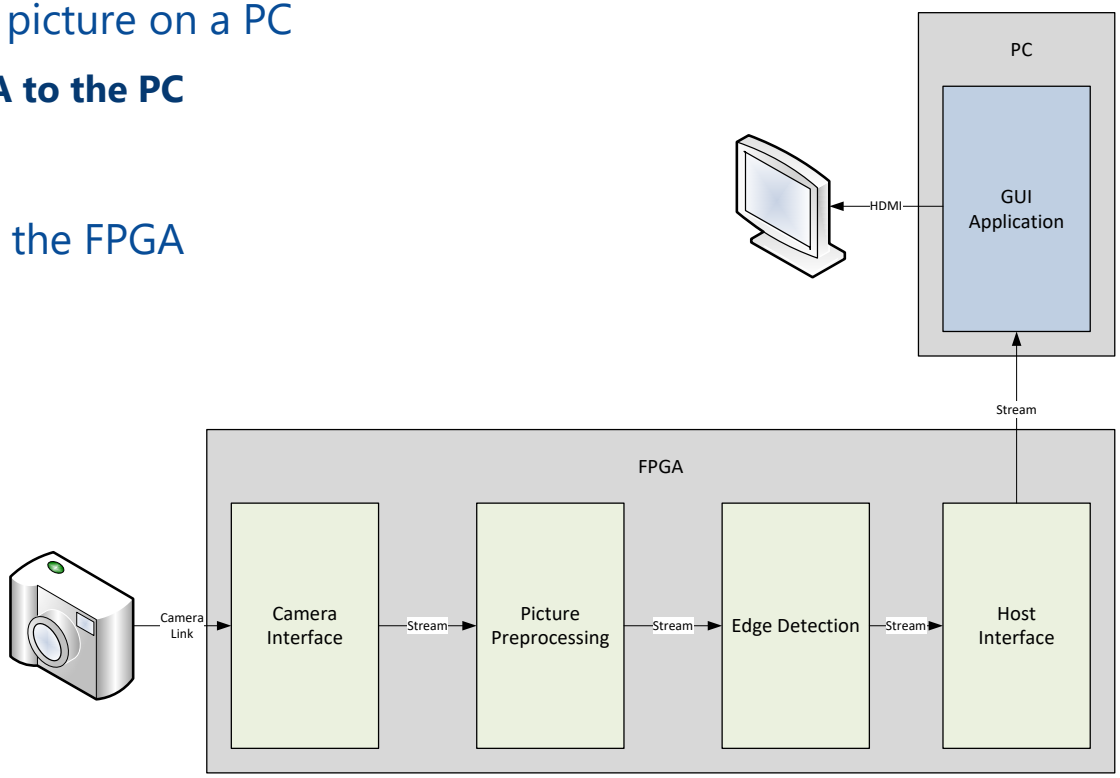
FPGA? Enclustra!

Agenda

- Enclustra company profile
- **Reasons for linking an FPGA to a high level language**
- Requirements when linking an FPGA to a high level language
- Challenges when linking an FPGA to a high level language
- Why a standard solution makes sense
- FPGA Manager overview

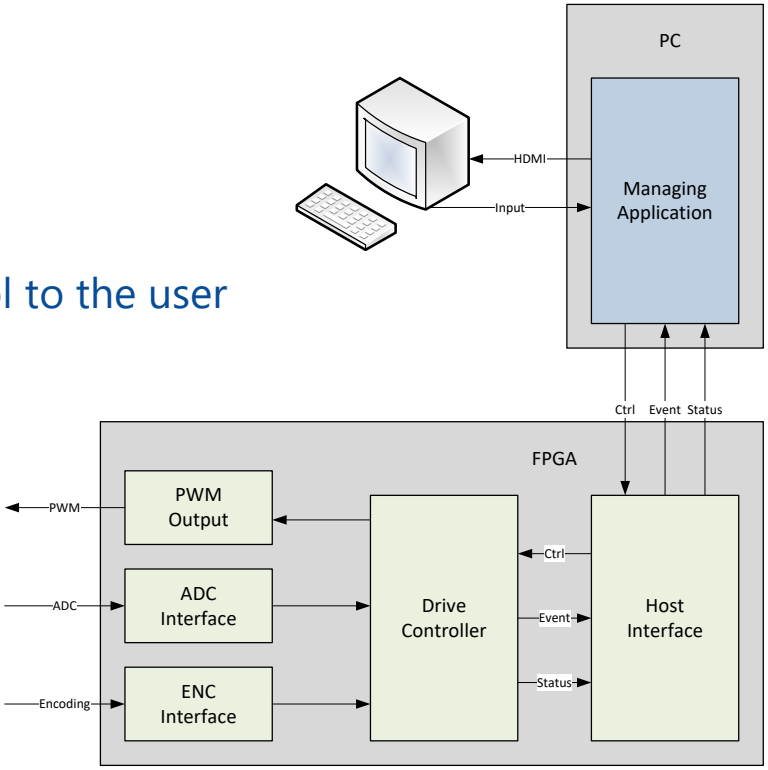
Reasons for linking an FPGA to a high level language

- Use case 1: Camera with edge detection
  - Data preprocessing in the FPGA because of high resolution and very high refresh rate
  - Display of the preprocessed picture on a PC
    - **Data stream from the FPGA to the PC**
  - GUI written in C#
  - PC is remotely connected to the FPGA



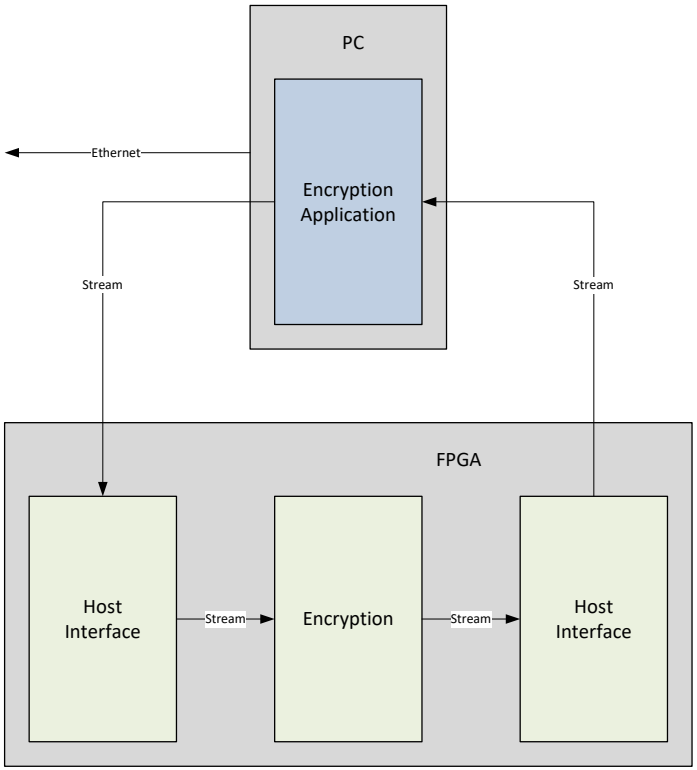
Reasons for linking an FPGA to a high level language

- Use case 2: Configuration and status monitoring of a drive controller
  - Drive controller in the FPGA for high control loop rates
  - Configuration of the motion vectors from a PC
    - **Write register access from the PC to the FPGA**
    - Status monitoring from a PC
    - **Read register access from the PC to the FPGA**
  - Sending of alarms/events from the drive control to the user
    - **Signalling from the FPGA to the PC**
  - Application written in C++
  - PC is remotely connected to the FPGA



Reasons for linking an FPGA to a high level language

- Use case 3: Hardware-accelerated software
  - Encryption of network traffic in real-time
  - Encryption of the network traffic by the FPGA
    - **Data stream from the PC to the FPGA**
  - Sending of encrypted data by the PC
    - **Data stream from the FPGA to the PC**
  - Small application in C, asynchronous transfers
  - PC and FPGA are one system



Reasons for linking an FPGA to a high level language

- Three types of interaction
  - Data stream between FPGA and PC
    - Typically used for data preparation (FPGA to PC) or data processing (PC to FPGA)
    - **Data Stream**
  - Register access on the FPGA by the PC
    - Typically used for writing configuration data or polling status
    - **Memory Mapped**
  - Event signaling from the FPGA to the PC
    - Typically Interrupts (Completion, Errors, etc.)
    - **Signaling**

Agenda

- Enclustra company profile
- Reasons for linking an FPGA to a high level language
- **Requirements when linking an FPGA to a high level language**
- Challenges when linking an FPGA to a high level language
- Why a standard solution makes sense
- FPGA Manager overview

Requirements when linking an FPGA to a high level language (1)

- All three types of interaction shall be supported
  - Data Streams
    - Frame Streams, Byte Streams
    - Appending Metadata to data streams (e.g. Timestamps)
  - Memory Mapped
    - Read, Write, Read-Modify-Write
    - Single Access, Burst
  - Signalling
    - Interrupts
    - Edge, Level
    - Priorities

Requirements when linking an FPGA to a high level language (2)

- Diversity
  - Multiple different links and bandwidths
    - PCIe Gen1/2/..., USB 2.0/3.0/..., ETH100/1000Mbps/...
  - Multiple FPGA vendors, CPU architectures, OS and FPGA boards
    - Altera, Xilinx, ... & X86, ARM, ... & Windows, Linux ....
  - Multiple programming languages
    - C#/.NET, C/C++, Matlab, Java etc.
- Uniformity
  - Same API for all links, FPGAs and boards
  - Same API for different operating systems
  - "Same" API for different programming languages (functional identical, and where possible, also syntactic)
  - Same interface for the user logic in the FPGA for all links, FPGAs and boards
  - Use of standardized interface for user logic in the FPGA

Requirements when linking an FPGA to a high level language (3)

- Multiplexing
  - Multiple connecting channels over one physical link
    - E.g. 3 Data Stream channels, 2 Memory Mapped channels, 2 Signalling channels over the same USB 3.0 link
  - Simultaneous access of multiple applications to different channels
- Performance
  - Bandwidth (MB/s)
  - Latency (s)
  - CPU load (%)
  - Flow control

Requirements when linking an FPGA to a high level language (4)

- Blocking and non-blocking
  - Support for synchronous and asynchronous transfers

Linking to an FPGA should be **simple** for the user!!!

The user should be able to concentrate on his end goals; linking to the FPGA is mainly just a means to that end.

Agenda

- Enclustra company profile
- Reasons for linking an FPGA to a high level language
- Requirements when linking an FPGA to a high level language
- **Challenges when linking an FPGA to a high level language**
- Why a standard solution makes sense
- FPGA Manager overview

Challenges when linking an FPGA to a high level language (1)

- Diversity vs. Uniformity
  - Unified interfaces over all dimensions (Links, FPGA vendors, CPU architectures, operating systems and programming languages)
- Performance vs. Resources
  - To achieve maximum performance, normally more resources are needed
- Simplicity vs. Flexibility
  - How simple can an API be and still give maximum flexibility?
- Various links require additional chips/drivers/libraries
  - PCIe => MGTs and PCIe hard IP
  - USB => USB PHY, FIFO controller (e.g. Cypress FX3 or FTDI 2232H), FIFO controller drivers and access libraries
  - Ethernet => ETH PHY, host MAC driver and socket interface of OS

Challenges when linking an FPGA to a high level language (2)

- Bandwidth/Throughput vs. Latency
  - For maximum bandwidth large data packets are optimal, for low latency small packets are optimal, and for maximum throughput as many operations as possible should be contained in one packet.
- Modularity/maintainability vs. Optimization
  - How much can you optimize before losing maintainability and modularity?

The main challenge is to bring **all these paradigms under one hat!**

How to handle these paradigms? (1)

- Diversity vs. Uniformity
  - Link for the software and the hardware developer shall be a black box
    - Interface abstraction on Host and FPGA
  - Use a link independent streaming protocol which can be encapsulated into link specific protocols/frames
    - Allows multiplexing, error detection, throttling etc.
  - Use a link independent memory mapped protocol which shall be encapsulated into the already link independent streaming protocol
    - Allows memory mapped acccess

How to handle these paradigms? (2)

- Diversity vs. Uniformity
  - Abstract the operating system with an OS-abstraction layer in the software library
  - Create the software library in one programming language and only have access APIs in different languages, e.g. library in C++, APIs for C#, C, C++, Java, Matlab …
  - Don't use FPGA vendor specific constructs in the core part of the firmware
  - Abstract the link with a link abstraction module in the firmware
    - Here also FPGA vendor dependent constructs are sometimes used, e.g. PCIe Hard IP
  - Use standardized interfaces in the firmware, e.g. AXI and AXIS

ENCLUSTRA FPGA SOLUTIONS

How to handle these paradigms? (3)

- Performance vs. Resources
  - Make all parameters that have an influence on performance and resources accessible to the user
    - Allows tweaking the solution to the specific application which shall represent the minimum required resources to achieve the required performance
- Simplicity vs. Flexibility
  - Create a simple API
  - Provide parameters to change behavior of API functionalities
  - Create a default parameter set that matches “most” applications
- Simplicity vs. Complexity
  - Keep the complexity away from the user
  - Don’t try to recover from all possible error cases

ENCLUSTRA FPGA SOLUTIONS

How to handle these paradigms? (4)

- Bandwidth vs. Latency
  - Make all parameters that have an influence on bandwidth, latency and throughput accessible to the user
    - Allows tweaking the solution to the specific application which shall represent the maximum bandwidth with minimum latency and maximum throughput required by the application
- Modularity/maintainability vs. Optimization
  - Create a modular architecture with well defined interfaces so parts could be replaced with optimized versions and still keeping core parts
  - Provide parameters to optimize for a specific application

Agenda

- Enclustra company profile
- Reasons for linking an FPGA to a high level language
- Requirements when linking an FPGA to a high level language
- Challenges when linking an FPGA to a high level language
- **Why a standard solution makes sense**
- FPGA Manager overview

Why a standard solution makes perfect sense

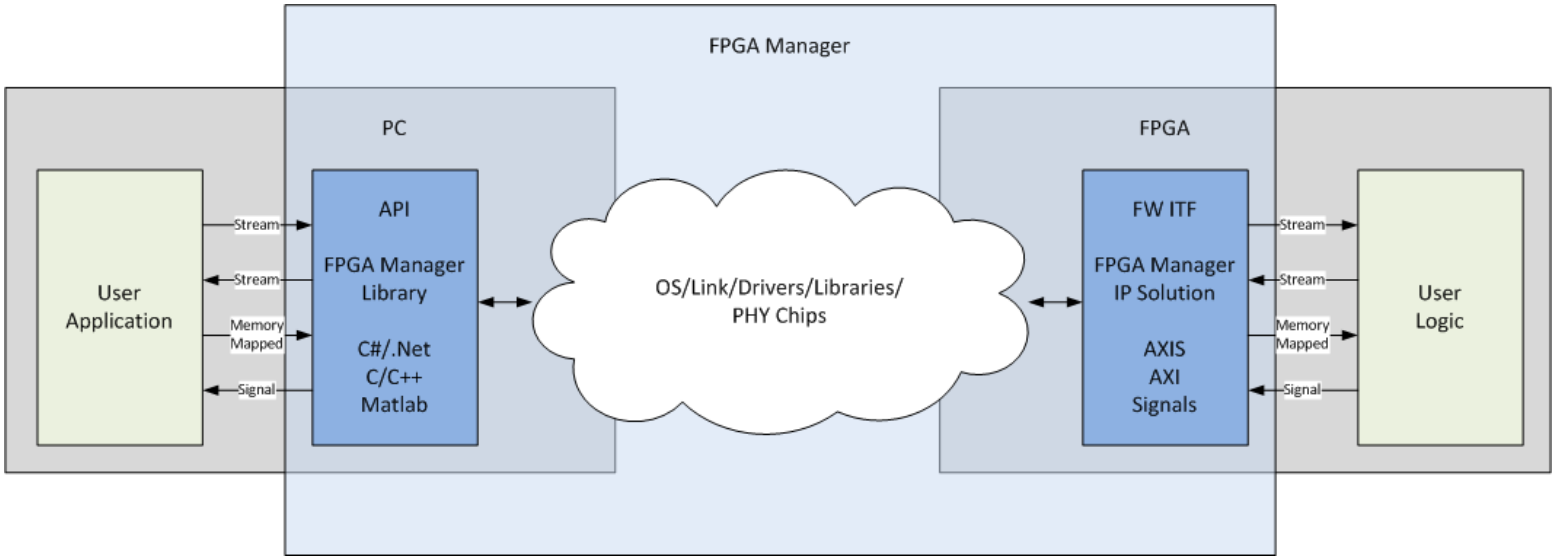
- All these challenges – and the only thing we want to do is communicate with an FPGA
- You "typically" start from a reference design and end up using (wasting?) 50% of your project budget (money, time), with the added bonus of increased stress
- Repeating work: for each link, OS or FPGA again
- Communication with the FPGA is often only a means to an end

Agenda

- Enclustra company profile
- Reasons for linking an FPGA to a high level language
- Requirements when linking an FPGA to a high level language
- Challenges when linking an FPGA to a high level language
- Why a standard solution makes sense
- **FPGA Manager overview**

FPGA Manager overview
What is FPGA Manager?

- A software-firmware co-solution
  - **FPGA Manager Firmware IP solution**
    - Acces to user logic in the FPGA
  - **FPGA Manager Software library**
    - API for user application



ENCLUSTRA FPGA SOLUTIONS

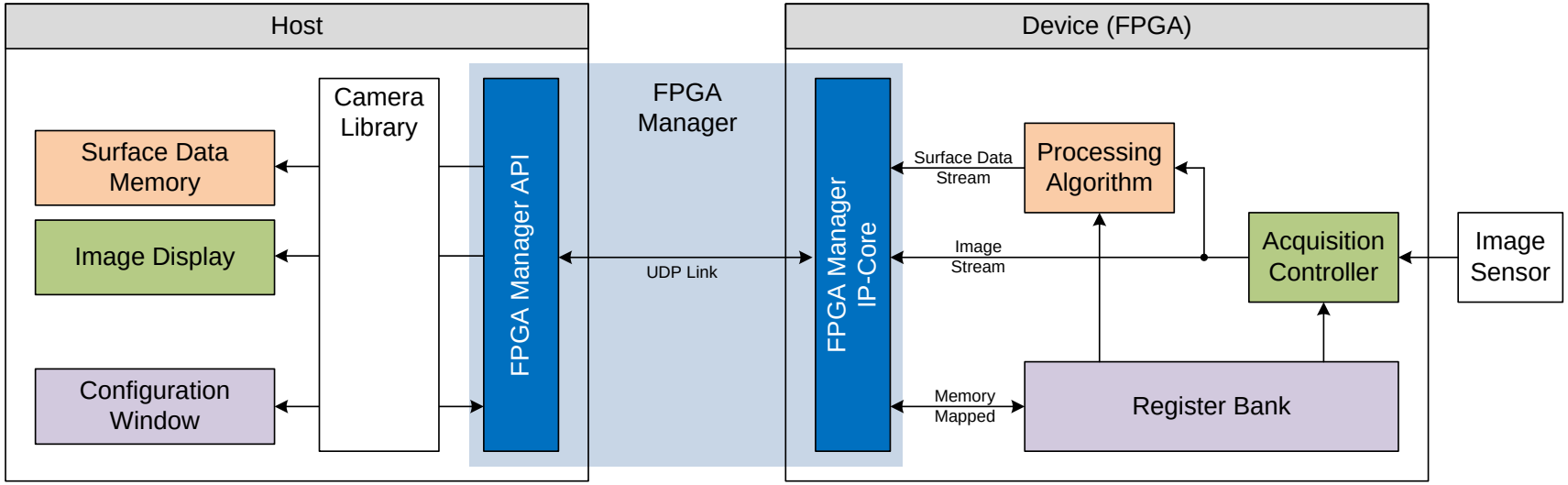
Features

- Interfaces
  - PCIe
  - Ethernet
  - USB 2.0, 3.0
- Operating Systems
  - Windows
  - Linux
- Software API
  - C++
  - C#
  - C
  - Matlab

Performance vs. Resources

- Resource optimized implementation
  - For applications with limited FPGA resources
- Performance optimized implementation
  - For applications with higher demands on data bandwith
- For small data packets no differences
  - Up to ~1 kByte

Camera Example(1)



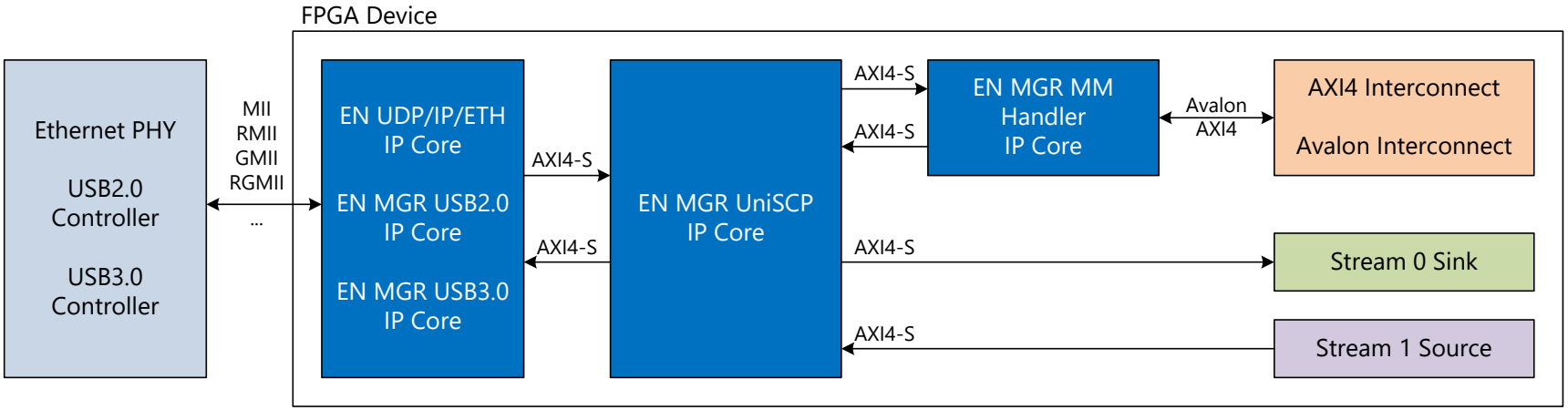
Camera Example(2)

- Ethernet link
  - UDP Link
- 3 different data transmissions
  - 2 Data streams from the FPGA to the Host
  - Memory Mapped access for configuration
- GUI written in C# on Host

Camera Example(3)

- No microprocessor on FPGA side
  - Configuration from Host PC
- Customer Software completely in C#
- Adding another vision algorithm would be simple

Device Side



- Link to Interface
- Core handling streams
- Memory Mapped Handler

ENCLUSTRA FPGA SOLUTIONS

Summary

- Connection of FPGAs to high-level languages is a **typical application**
- Connection of FPGAs is often **only a means to an end**
- Connection shall be **simple**
- Connection shall be **flexible**
- **Migration** shall be **easy**
- In the majority of cases using a **standard solution is cheaper than developing one** itself

Questions?

Next Events:

- Sindex
  August 28-30, 2018
  Berne, Switzerland
- Vision 2018
  November 6-8, 2018
  Stuttgart, Germany

Matthias Frei

Enclustra GmbH

matthias.frei@enclustra.com

Tel. +41 43 343 39 56

Quaterly newsletter: enclustra.com/subscribe