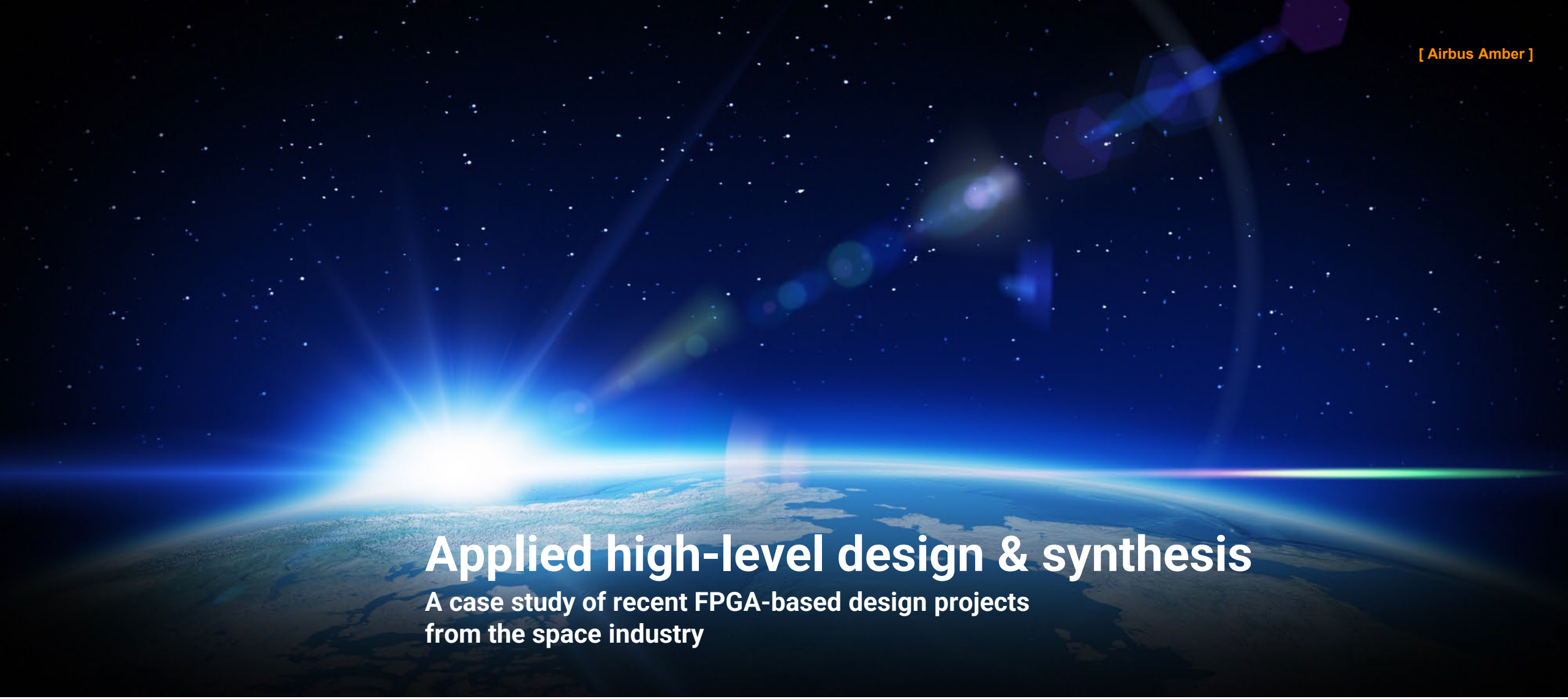


[Airbus Amber]

Applied high-level design & synthesis

A case study of recent FPGA-based design projects from the space industry

DEFENCE AND SPACE

Patrick Gest

14 November 2023

AIRBUS

Introduction

- Devices: KU060 Versal RTG4 NG-Ultra

© Avnet

© Microsemi

© ESA

- Design Flow: Most of our designs are done on the RTL level of abstraction and written in VHDL

- Domains: Most of our designs make use of algorithms from the digital signal processing & artificial intelligence domains

  => Dataflow centric => Strength of high-level synthesis tools

AIRBUS

Use of High-level Design & Synthesis Tools

MathWorks

- MATLAB/Simulink for designing complex signal processing algorithms
- MATLAB HDL Coder for implementing the algorithms on FPGA via VHDL
- MATLAB generated testbenches for co-simulation using Questa Sim

AMD Xilinx

- Vitis HLS for designing data processing pipelines (computer vision, signal processing)
- Vitis HLS for C-simulation of HLS functions
- Vitis IDE for co-simulation of HLS functions together with on-board SW & OS

Deployment of RF Interference Mitigation Algorithms

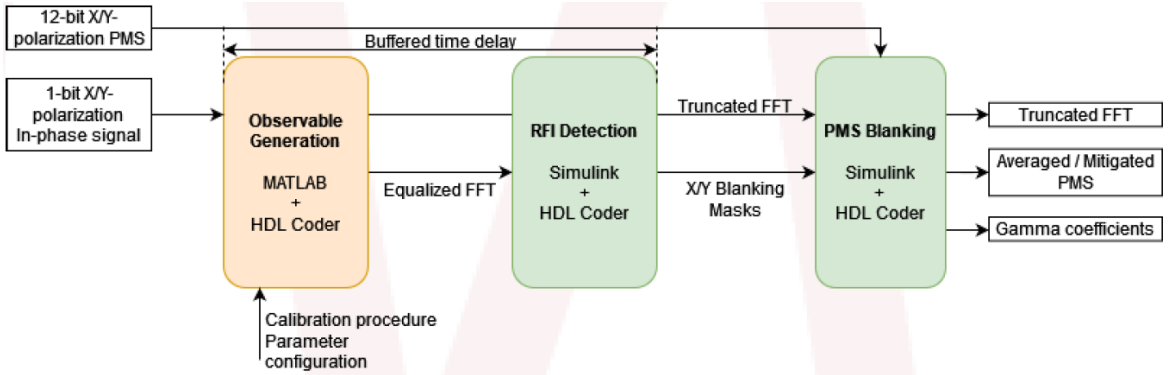
- Project: L-Band Radio Frequency Interference (RFI) Mitigation

- Background:
  - Support of the Soil Moisture and Ocean Salinity (SMOS) mission from ESA
  - Microwave Imaging Radiometer instrument picks up microwave emissions from the Earth's surface
  - Based on the received signals, the processing unit calculates the levels of soil moisture on ground via cross-correlation
  - Operates in the L-Band, which is contaminated by RF interference

- Goal: Develop and deploy an algorithm for mitigating the interference

- Concept:

Blank symbols that shall not be used in the calculation of the correlation



CLOSEOUT PLAN RFI MITIGATION

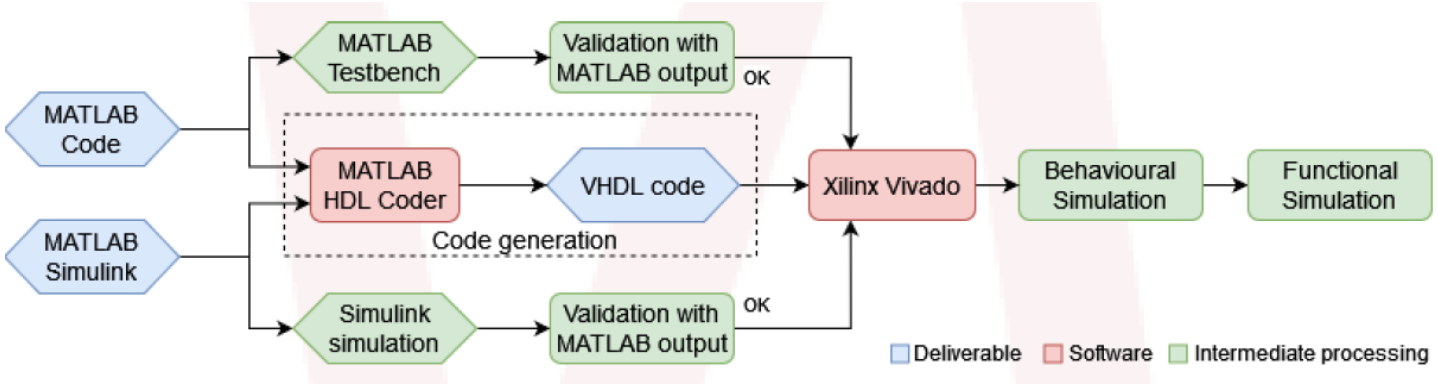
ALGORITHM IMPLEMENTATION

J. QUEROL, A. PÉREZ, A. CAMPS

Deployment of RF Interference Mitigation Algorithms

- Deployment: The algorithm shall be implemented on FPGA and process the antenna data in real-time
- Tools: MATLAB HDL Coder was used for the generation of VHDL code

  1.) This enables a rapid prototyping of the developed algorithm
  2.) The functionality of the algorithm can be verified more easily on system level than on HDL level

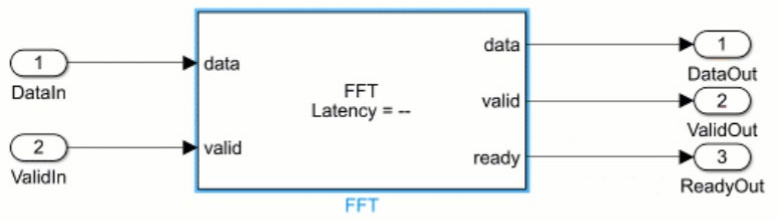


RFI MITIGATION ALGORITHM
BREADBOARD DETAILED DESIGN
J. QUEROL, A. PÉREZ, A. CAMPS

- Challenges: Auto-generated code did not meet timing constraints; manual insertion of pipelining register was necessary

Evaluation of Auto-generated HDL Code on the NG-Ultra

- Goal: Feasibility study of auto-generated VHDL code from MATLAB HDL Coder on NG-Ultra devices
- Difficulty: NG-Ultra is not on the list of supported devices
  => Questionable whether desired performance metrics will be met
- Example: Simulink FFT Implementation



Target Frequency: 180 MHz

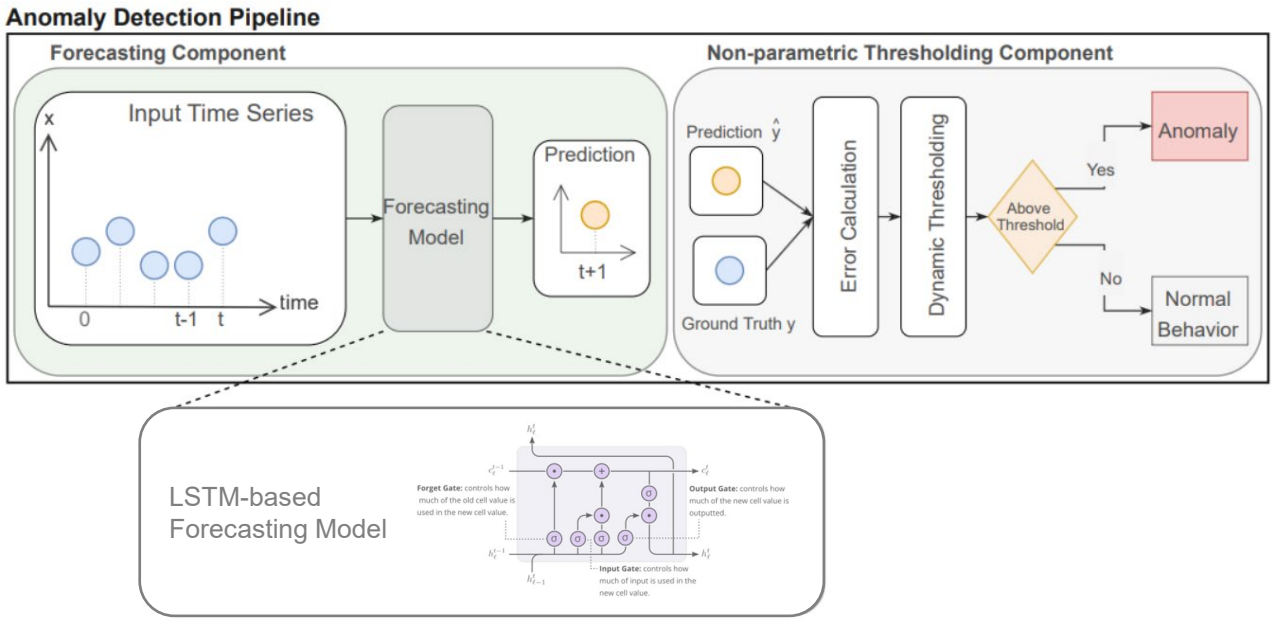
NxMap	Default	Floorplanning
clk (post syn)	210 MHz	200 MHz
clk (post rout)	130 MHz	180 MHz

- Conclusion: Auto-generation HDL code typically works decently well for the NG-Ultra

AIRBUS

Fast Prototyping of AI Models on FPGAs / SoCs

- Project: Anomaly Detection & Anomaly Prognosis (ADAP)
- Goal: Demonstrate that AI can enhance the FDIR capabilities of future satellites
- Task: Detect anomalous behaviours in a satellite‘s telemetry data
- Concept:

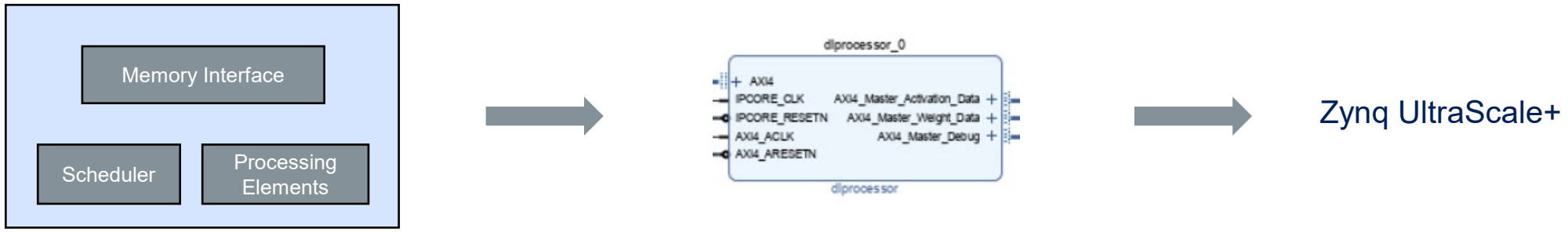


AIRBUS

Fast Prototyping of AI Models on FPGAs / SoCs

- Tools: MATLAB Deep Learning HDL Toolbox & HDL Coder
- Output: Generation of a customized hardware implementation for a given deep learning network

  => VHDL / Verilog for implementation on an FPGA



- Results: Throughput of 0.5 MB / s @ Total Power Consumption of ~5 W

AIRBUS

Summarizing the Experiences with the HDL Coder

Main pros:

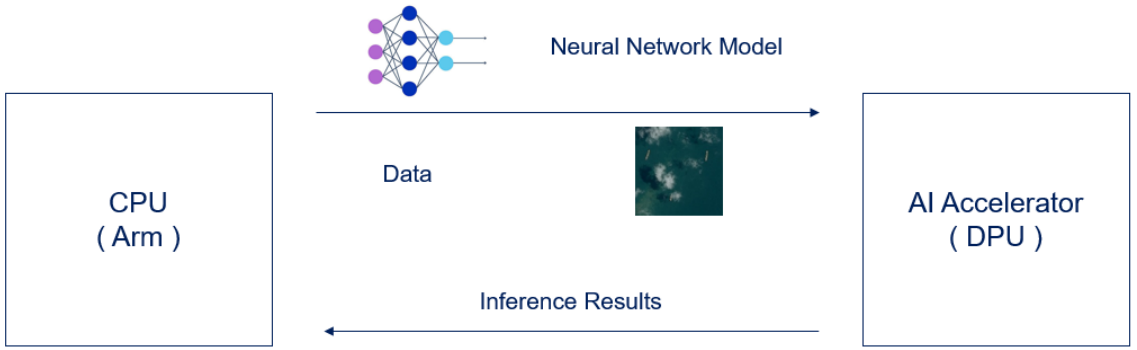
- Very efficient for complex “linear” data flow
- Sophisticated signal processing toolboxes
- Simplification of filter design (with direct quantized implementation)
- Direct link between the MATLAB/Simulink Model and RTL verification

Main cons:

- Still inefficient for state machine generation compared to “hand design”
- Design can become quite complex and unreadable
- Multiple clock domains complex to implement

Extending an AI Processing Platform via HLS Functions

- Background: Many of our AI projects make use of the Deep Learning Processing Unit (DPU) IP Core from AMD Xilinx

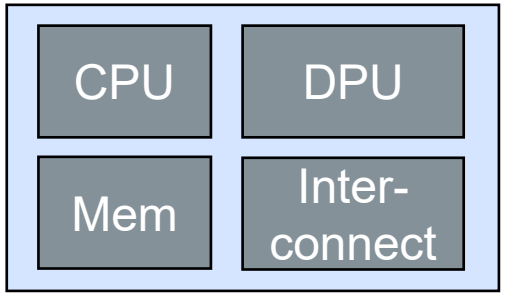


- Demand: Many applications require additional operations to be executed that are not supported by the DPU

- Examples:
  - Earth Observation -> Preprocessing of images (Pixel Correction, Image Filtering, Image Cropping, …)
  - Signal Processing -> Transformation of data (Frequency Filtering, FFT, Interpolation/Decimation, …)

Extending an AI Processing Platform via HLS Functions

- Concept: Functional HLS kernels integrate very well into the development flow with Vitis

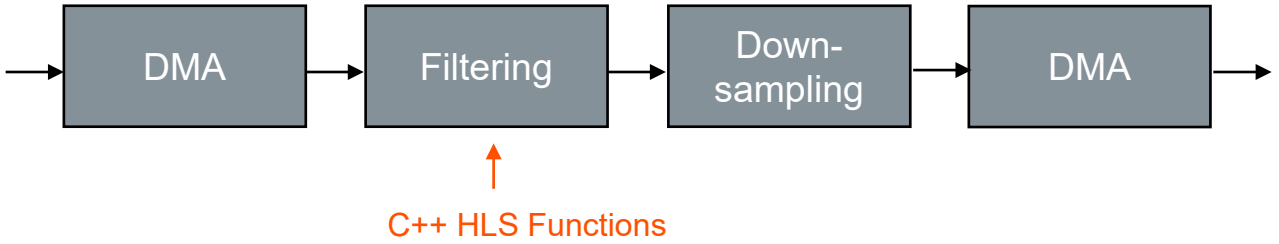


Extensible Vitis Platform

Equipped with „open“ interconnect ports

- Example: Earth observation

- Example: Signal Processing



AIRBUS

Summarizing the Experiences with Vitis HLS

Main pros:

- Simple synthesis of interfaces for integration into existing design
- Sophisticated set of HLS libraries -> Vision, DSP, Linear Algebra, …
- Good integration into Vitis IDE for co-simulation/emulation with hardware and software

Main cons:

- Learning curve; HLS C/C++ differs quite a bit from standard code
- Designs can become quite complex and unreadable

Thank you !