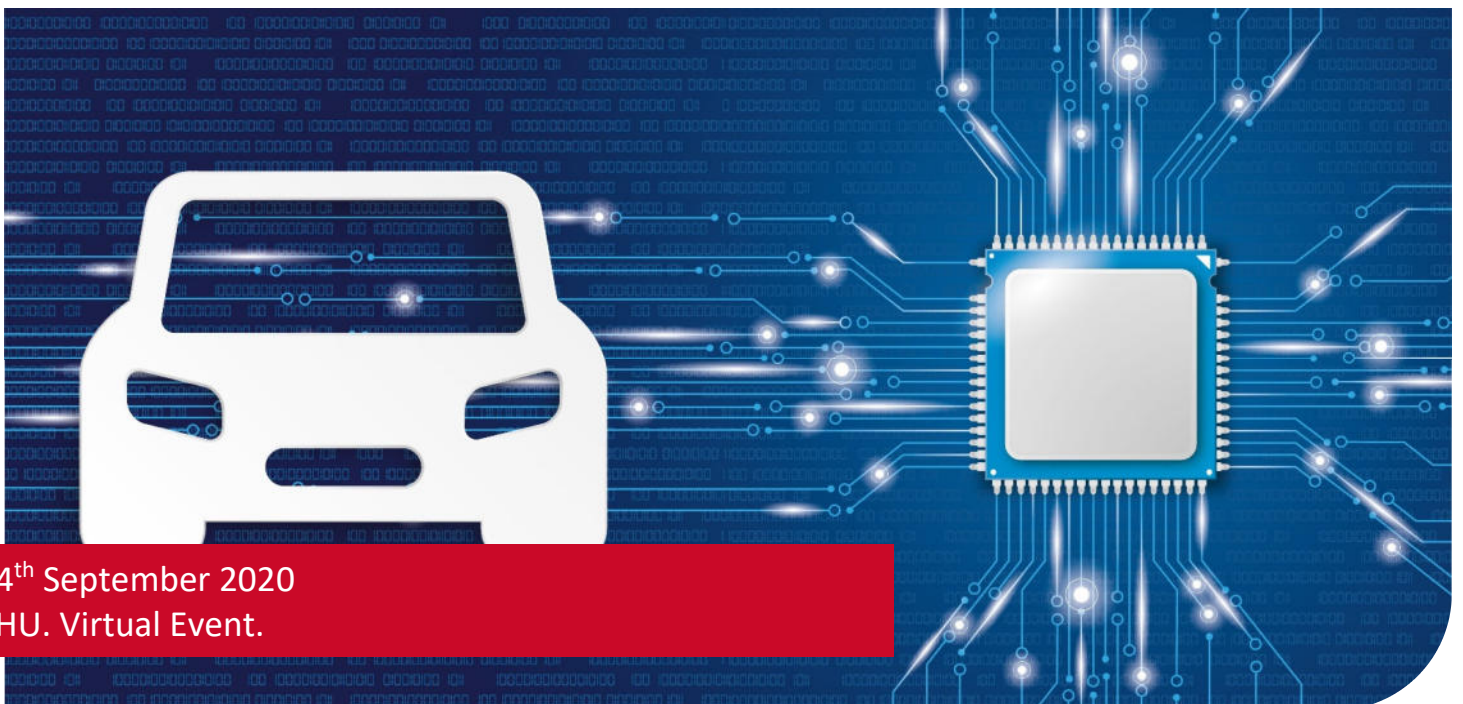


Proceedings of the
4th Workshop

PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE

– FROM CLASSICAL FPGA TO ADAPTABLE COMPUTING

ALGORITHMS. ARCHITECTURE. REALIZATION. TEST.



24th September 2020
THU. Virtual Event.

Imprint

ISBN 978-3-9820843-2-9

Cover photo © Alexander Limbach #233715262

Editor

Prof. Dr. Anestis Terzis
Technische Hochschule Ulm
University of Applied Sciences
Prittwitzstr. 10
89075 Ulm, Germany
Anestis.Terzis@thu.de

This work is subject to copyright. All rights are reserved by the publisher or the copyright holder.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use. The publisher, the authors and the editors are safe to assume that the advice and information in this publication are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Proceedings of the 4th Workshop

Programmable Processing for the Autonomous / Connected Vehicle

Program Committee of the Workshop:
Prof. Dr. A. Terzis, Dr. E. Schubert, M. Güthoff

Contents

- Opening Remarks from the Program Committee.....1**
Anestis Terzis, THU, Endric Schubert, Missing Link Electronics, Mathias Güthoff, Xilinx

- FPGA & SoC Design for Functional Safety.....6**
Dimitri Hamidi, Senior Application Engineer, MathWorks

- PCIe-over-TCP-over-TSN-over-10/25GigE.....28**
Endric Schubert, CTO, Missing Link Electronics

- Mirror Replacement System – An FPGA4ADAS Story.....42**
Stefan Schütz, Managing Director, Solectrix

- Automotive System Architectures from ADAS to AD.....53**
Ralf Neuhaus, Automotive System Architect EMEA, Xilinx

- Xilinx in AI: Versal AI core, AI Engine Architecture, Design Flow.....65**
Daniele Bagni, DSP / ML Specialist for EMEA, Xilinx

- Porting a Gesture Recognition Neural Network composed of CNN and LSTM
to a FPGA SoC.....84**
Robert Briegel, JacoL – FPGA Entwicklungen GmbH

- HAPPi-Net: Hardware-aware Performant Perception of Neural
Networks - Designing Lightweight CNNs on Embedded Platforms.....92**
Alexander Frickenstein¹, Manoj-Rohit Vemparala¹, Nael Fafous², Lukas Frickenstein¹,
Walter Stechele²,
¹BMW Group, Autonomous Driving, ²Technical University of Munich

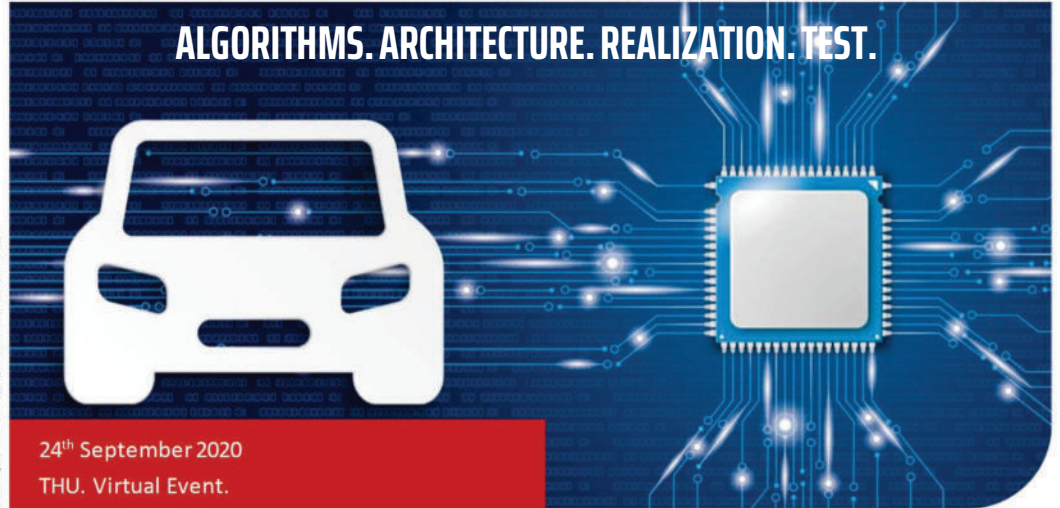
- Security in Automotive.....99**
Ralf Neuhaus, Automotive System Architect EMEA, Xilinx

WELCOME.



PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE – FROM CLASSICAL FPGA TO ADAPTABLE COMPUTING

Partner:

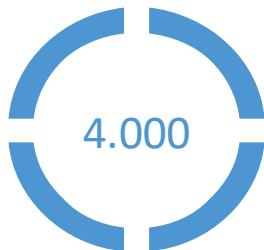


Ulm University of Applied Sciences



› In Numbers

Students



Professors



Other Academics +
Administration



Partnering UAS
worldwide



Studies and Lectures



› Six departments



Electrical Engineering and Information Technology



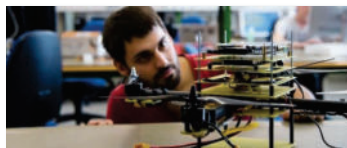
Mechanical and Automotive Engineering



Mathematics, Natural and Economic Sciences



Production Engineering and Production Economics



Computer Science



Mechatronics and Medical Engineering

3 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

Applied Research



› Five focus areas

- **Modern Mobility**
- **Digital Technologies**
- **Sustainable Energy Systems**
- **Technology in Health and Medicine**
- **Intelligent Industrial Systems**



4 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

FPGA and ADAS/AD in B. Eng. EE



Durch die Wahl von 2 Schwerpunkten mit jeweils 4 Modulen kann das Studium nach eigenen Interessen gestaltet und das fachliche Profil geschärft werden.

Elektrotechnik und Informationstechnik B. Eng.						
7	Bachelorarbeit mit Seminar		Wahlmodul	Modul Schwerpunkt 1	Modul Schwerpunkt 2	
6	Projekt Elektrotechnik	Wahlmodul	Wahlmodul	Modul Schwerpunkt 1	Modul Schwerpunkt 2	
5	Praxissemester (Praxisprojekt im Unternehmen mit vorbereitender Laborveranstaltung)					
4	Elektronik	Signalverarbeitung	Software Engineering	Regelungstechnik u. elektrische Maschinen	Modul Schwerpunkt 1	Modul Schwerpunkt 2
3	Elektronik	Systemtheorie	Programmieren in C++	Mathematik für die Elektrotechnik	Modul Schwerpunkt 1	Modul Schwerpunkt 2
Grundstudium						
2	Elektronik	Grundlagen der Komm.technik	Mikrocomputertechnik	Mathematik	Physik	Digitaltechnik
1	Elektrotechnik		Programmieren in C	Mathematik	Physik	Digitaltechnik

— Pflichtmodule
 — Wahlmodule
 — Schwerpunktmodule

Ausführliche Infos zu den einzelnen Studieninhalten und Modulen unter: www.thu.de/et Ab dem 3. Semester einzelne Lehrveranstaltungen in Englisch möglich

Schwerpunkte

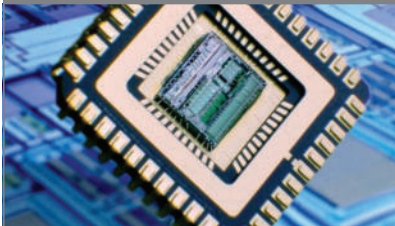
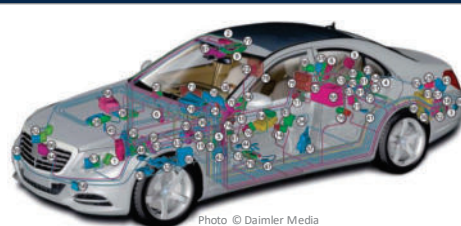
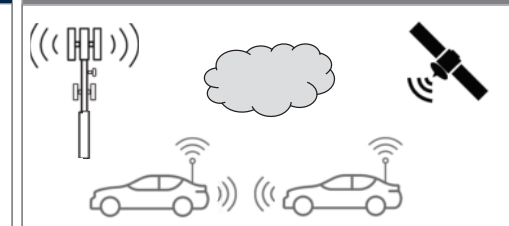
Kommunikationssysteme Methoden der Kommunikationstechnik Leitungsgebundene Kommunikation Simulation von Kommunikationssystemen Funkkommunikation	High Speed Electronics Hochfrequenztechnik Schaltungen der Kommunikationstechnik Elektromagnetische Verträglichkeit	Automatisierung Sensoren und Bussysteme Steuerungstechnik Aktorsysteme Methoden der Regelungstechnik	Fahrzeugsysteme Fahrwerksysteme Sensoren und Aktoren Predictive Engineering Autonomes Fahren
Leistungselektronik und Energietechnik Leistungselektronik Elektrische Energieversorgung Antriebe und Anlagentechnik Elektromagnetische Verträglichkeit	Internet of Things Softwarearchitekturen Verteilte Systeme Data Analysis Datenbanken	Wirtschaft BWL English for special purposes Europäisches Wirtschaftsrecht Projektmanagement	

FPGA **ADAS/AD**

5 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

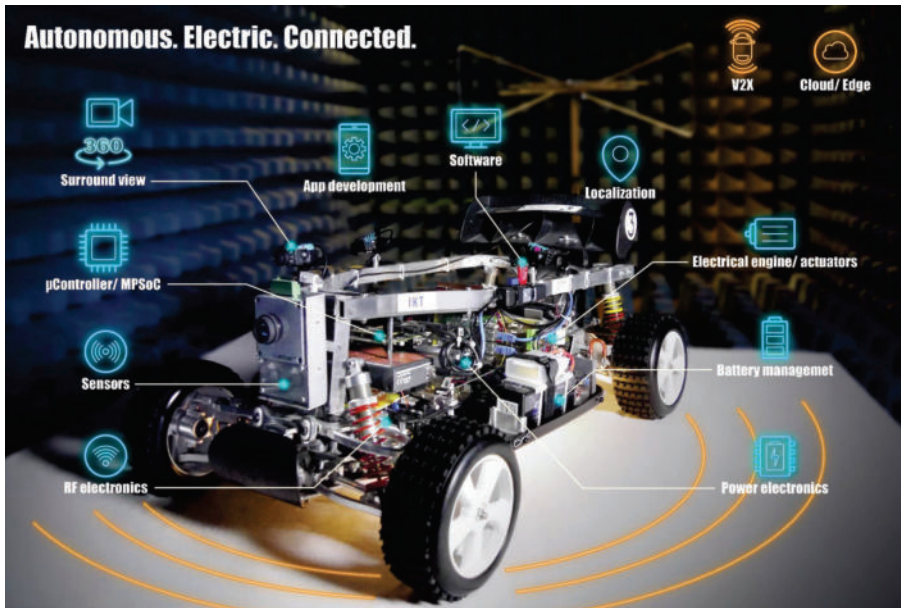
Levels of Connectivity - Connected Car



In Chip / ECU Networks	In Vehicle Networks	Infrastructure / Wireless Networks
	 <p style="font-size: small;">Photo © Daimler Media</p>	
NoC, MGT, AMBA AXI, LPDDR4/4x, Ethernet AVB/TSN, MIPI, HDMI, SCCB, PCI Express, SATA, DAC, ADC, SPI, I ² C, ...	CAN, LIN, Flexray, MOST, Ethernet, LVDS, HDBaseT, Powerline, ...	Bluetooth, Wi-Fi IEEE 802.11, Satellite Navigation GPS, Radio, TV, Wireless Entry, TPMS, V2I, V2N, V2V, 125 kHz – 80 GHz, ...

6 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

Platform for Autonomous Driving



RC Car
as a test and development platform for
Autonomous Driving technology

Institute of Communication Technology



Program



Time (CEST)	PROGRAM
09:45 - 10:00	Registration
10:00 – 10:20	Opening Remarks from the Program Committee Anestis Terzis, THU, Endric Schubert, MLE, Mathias Güthoff, Xilinx
10:20 - 11:10	FPGA and SoC Design using MATLAB and Simulink for Functional Safety Dimitri Hamidi, The MathWorks GmbH
11:10 - 11:40	PCIe-over-TCP-over-TSN-over-10/25 Gig Ethernet Endric Schubert, Missing Link Electronics
11:40 - 12:10	Mirror Replacement System - A FPGA-4-ADAS Story Stefan Schütz, Solectrix GmbH
12:10 - 13:00	Break

9 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

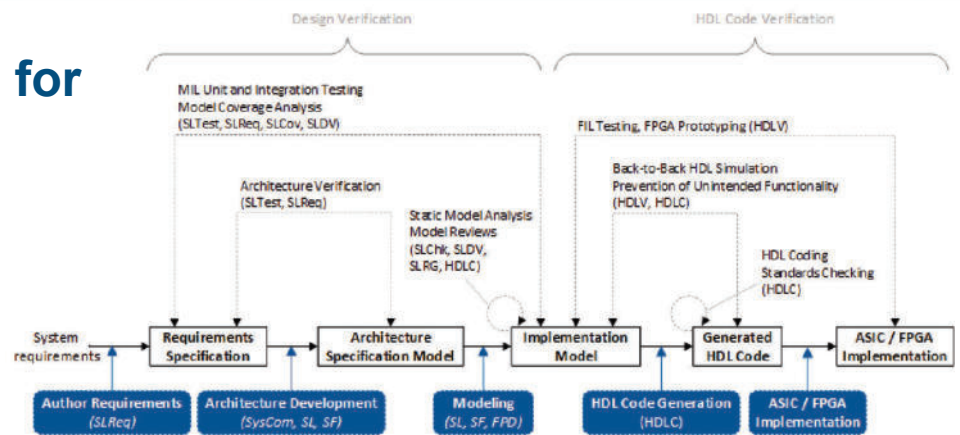
Program



Time (CEST)	PROGRAM
13:00 - 13:30	Overview: Xilinx in Automotive , Mathias Güthoff, Xilinx Automotive System Architectures from ADAS to AD , Ralf Neuhaus, Xilinx
13:30 - 14:00	Xilinx in AI – Versal AI-core, AI-Engine Architecture, Design Flow Daniele Bagni, Xilinx
14:00 - 14:30	Porting a Gesture Recognition Neural Network composed of CNN and LSTM to a FPGA-SoC R. Briegel, JacoL FPGA Entwicklungen GmbH
14:30 - 15:00	HAPPI-Net: Hardware-aware performant perception of Neural Networks Alexander Frickenstein, BMW Group
15:00 - 15:30	Security solution based on FPGA/SoC Ralf Neuhaus, Xilinx

10 © Prof. Dr. Anestis Terzis, PROGRAMMABLE PROCESSING FOR THE AUTONOMOUS / CONNECTED VEHICLE, 24.09.2020

FPGA & SoC Design for Functional Safety



Dimitri Hamidi
 Senior Application Engineer
 MathWorks
 dhamidi@mathworks.com



© 2020 The MathWorks, Inc.

Agenda

- Motivation behind Model Based Design for FPGA/ASIC
- Model Based Design Workflows for ISO26262
 - Development Workflow for HDL Code
 - Verification and Validation
- Deep Learning on FPGA

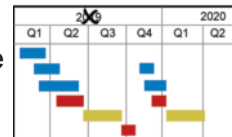
FPGA, ASIC, and SoC Development Projects

Wilson Research Report – Mentor Graphics



67% of ASIC/FPGA projects are **behind schedule**

Over **50% of project time** is spent on **verification**
(42 % on debugging!)



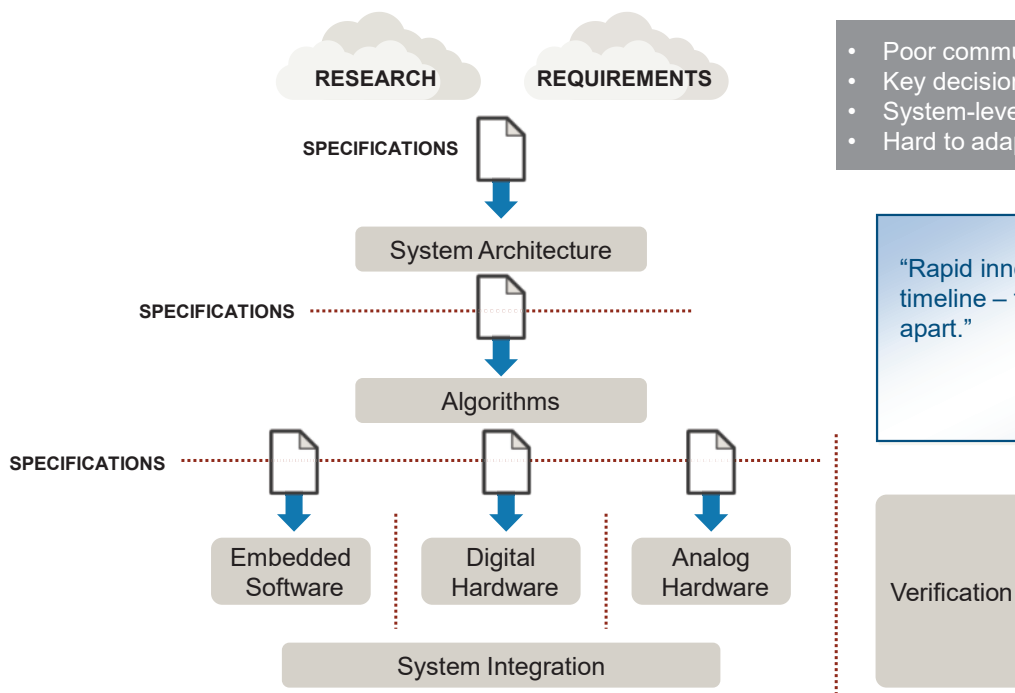
Root cause of functional flaws in 50% of cases originate from **specification**



84% of FPGA projects have non-trivial bugs escape into production

Statistics from 2018 Mentor Graphics / Wilson Research survey, averaged over FPGA/ASIC

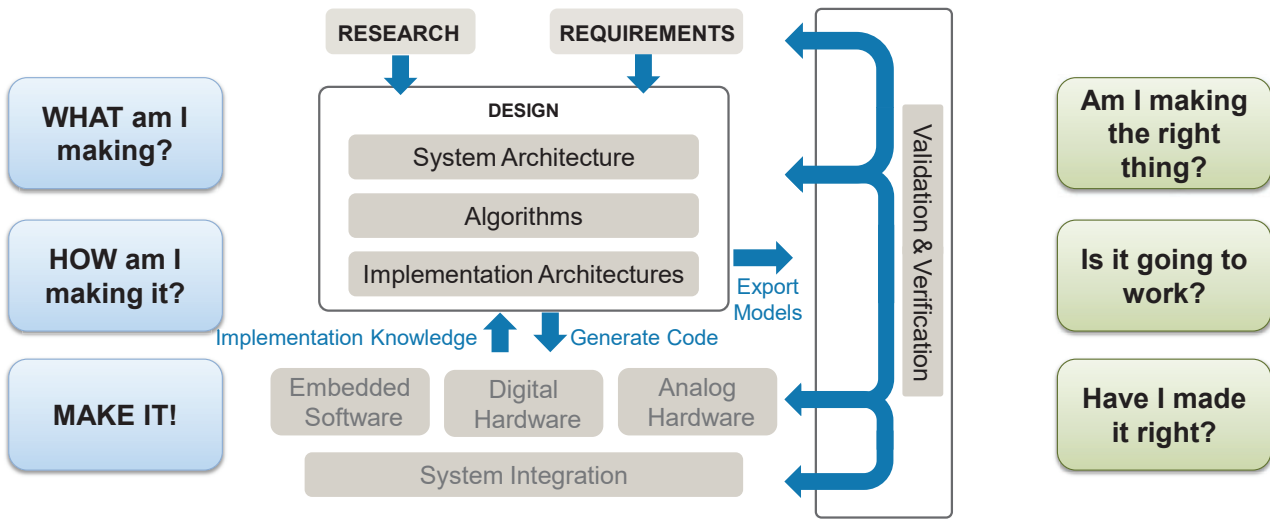
Many Different Skill Sets Need to Collaborate



- Poor communication across teams
- Key decisions made in silos
- System-level issues found in late stages
- Hard to adapt to changing requirements

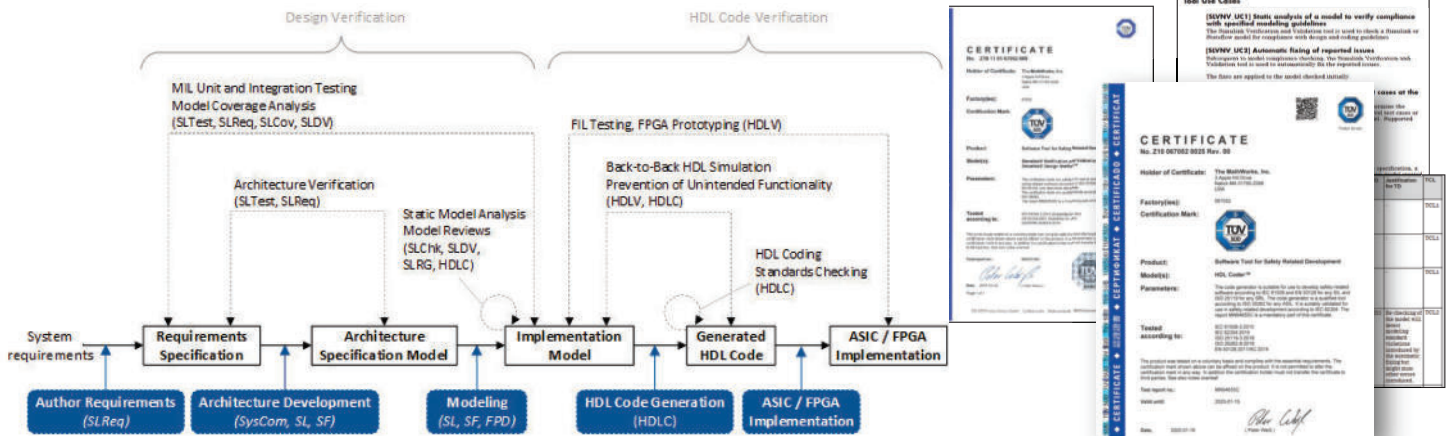
“Rapid innovation under a rapid timeline – that’s when this flow falls apart.”
 Jamie Haas
 Allegro Microsystems

SoC Collaboration with Model-Based Design



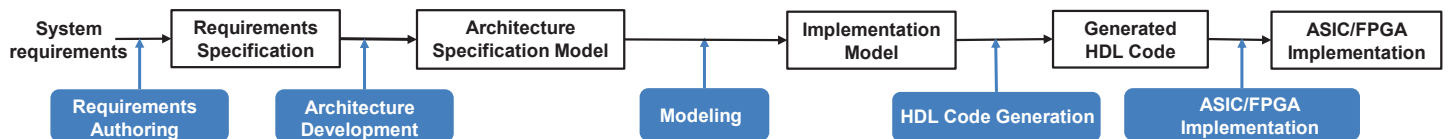
IEC Certification Kit – ISO26262

- I. Reference Workflow for:
 - I. Model-Based Design Development Workflow for HDL Code
 - II. Systematic verification and validation (V&V) of models and generated code
- II. Tool certification/qualification accomplished by tool test suites, vendor audits



Model Based Design Development Workflow for HDL Code

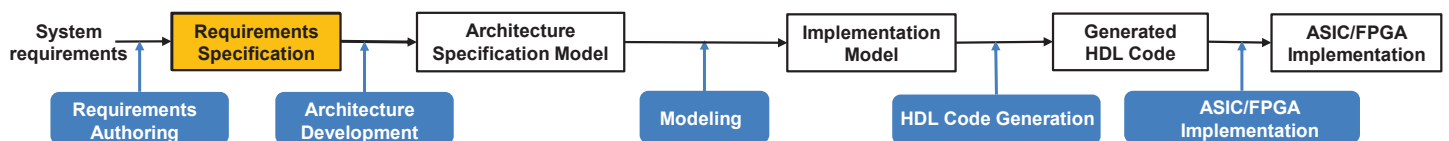
- I. **Executable specification:** a model in early phases of development to conceptually anticipate the functionality to be implemented, demonstrates and verifies the compliance of the input-output behavior of the **model** subject to the **model specifications**.
- II. **Implementation Model:** Enhance first stage by adding design information and implementation details, used as input for the generation of HDL code . Executable, final stage of the model evolution process.
- III. **Production-quality HDL code**



7

Development Workflow: Author Requirements

- Author Requirements
- Establish bidirectional traceability to models test and code
- Monitor and manage implementation status



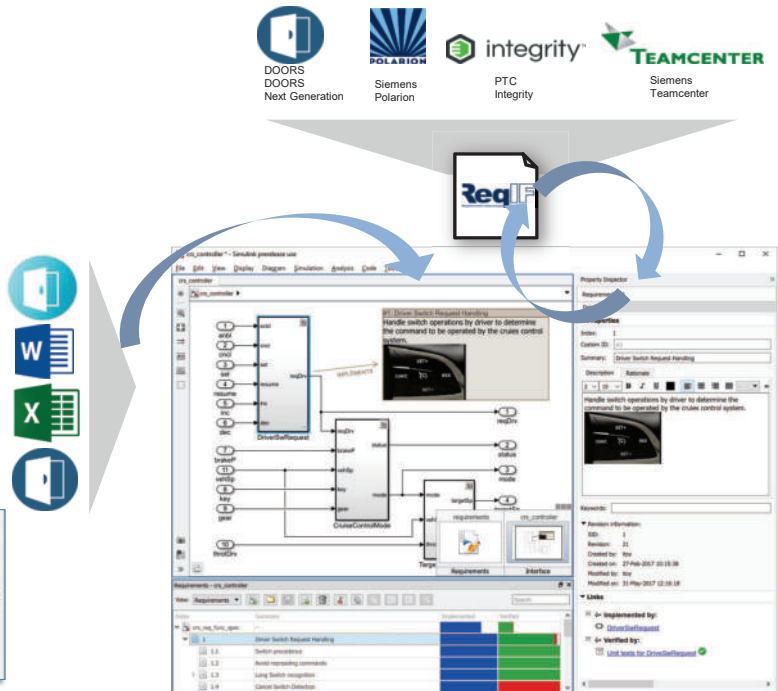
8

Requirements

Captured in

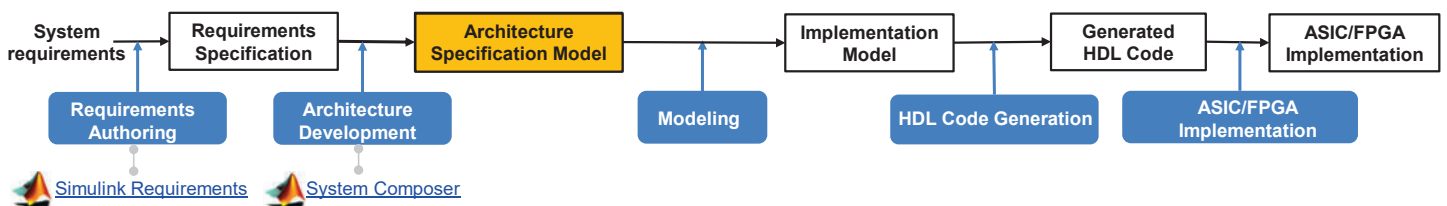
- Word, Excel,
- DOORS, etc.,
- and/or Simulink Requirements

Source	Destination	Change Information
Changed source: 3/12	Changed destination: 4/12	Source: Revision: 1 (Time Stamp: 25-Jun-2017 11:34:04)
Enabling cruise control	#9 Enable Switch Detection	Issue: Destination Changed
Disabling cruise control	#7 Cancel Switch Detection	Source: Revision: 15 (Time Stamp: 29-May-2017)
Activating cruise control	#8 Set Switch Detection	Actual: Revision: 18 (Time Stamp: 29-May-2017)
Deactivating cruise control	#8 Set Switch Detection	Clear Issue
Target Speed Increment	#11 Increment Switch Detection	
Target speed increment	#10 Decrement Switch Detection	
Target speed increment	#10 Decrement Short Switch Detection	
Target Speed Increment	#12 Decrement Short Switch Detection	
Successive Target Speed Increment	#13 Increment Long Switch Detection	
Successive Target Speed Increment	#14 Intermediate state	
Successive Target Speed Increment	#12 Decrement Long Switch Detection	
Successive Target Speed Increment	#18 Intermediate state	



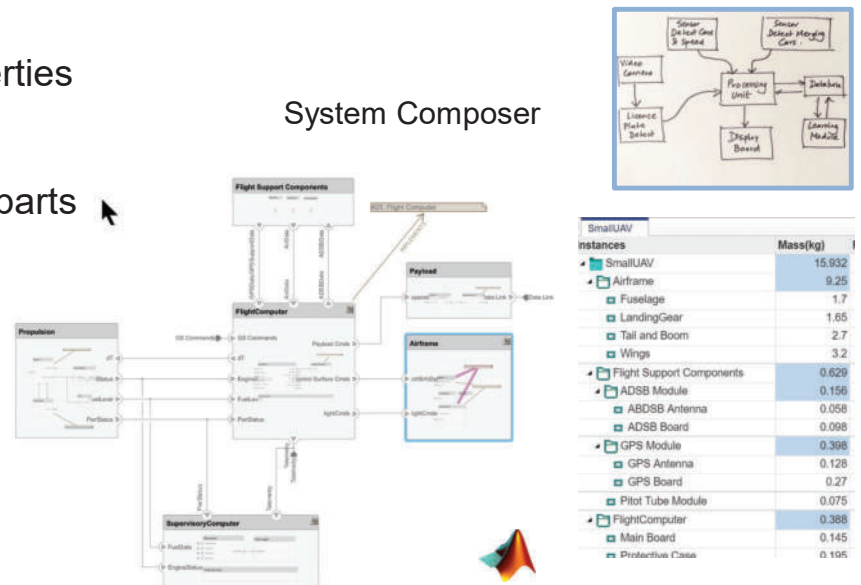
Development Workflow: Architecture Development

- Define model based hardware architecture



Model-Based Systems Engineering

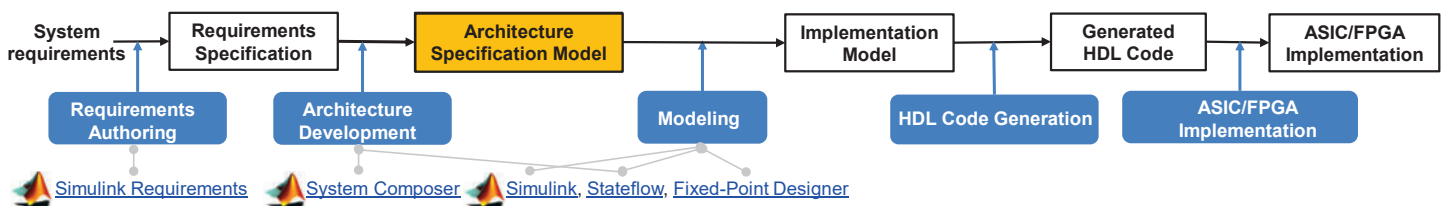
- Architecture Models
- Profiles, stereotypes, properties
- Allocate requirements
- Views to focus on relevant parts
- Perform Analysis



11

Development Workflow: Modeling

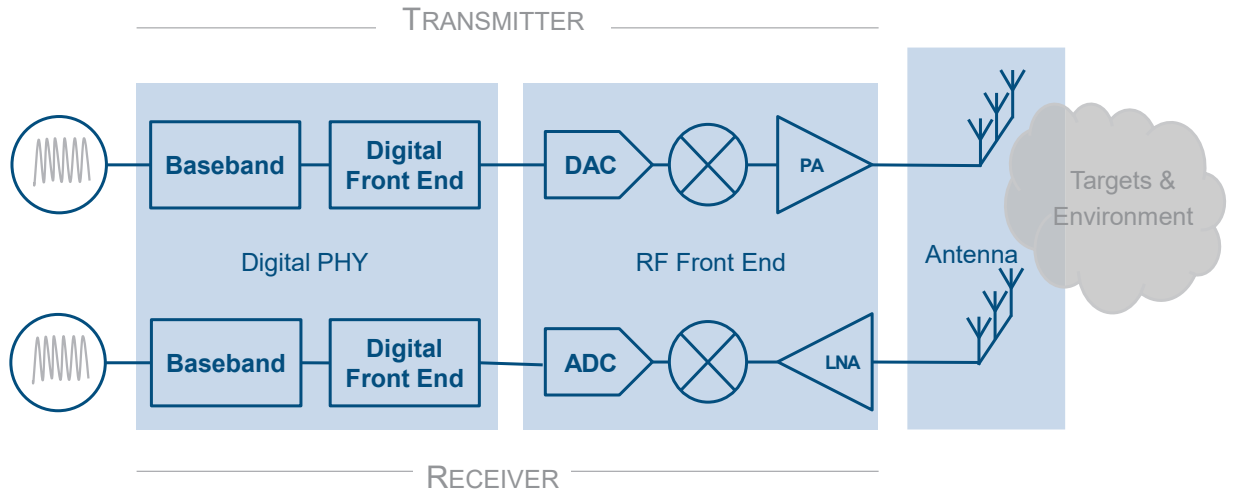
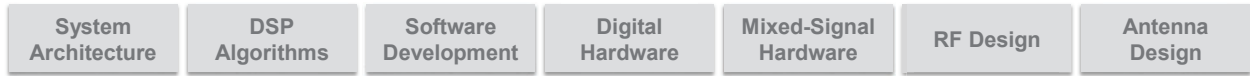
- Use Simulink / Stateflow to model behavior of your system



12

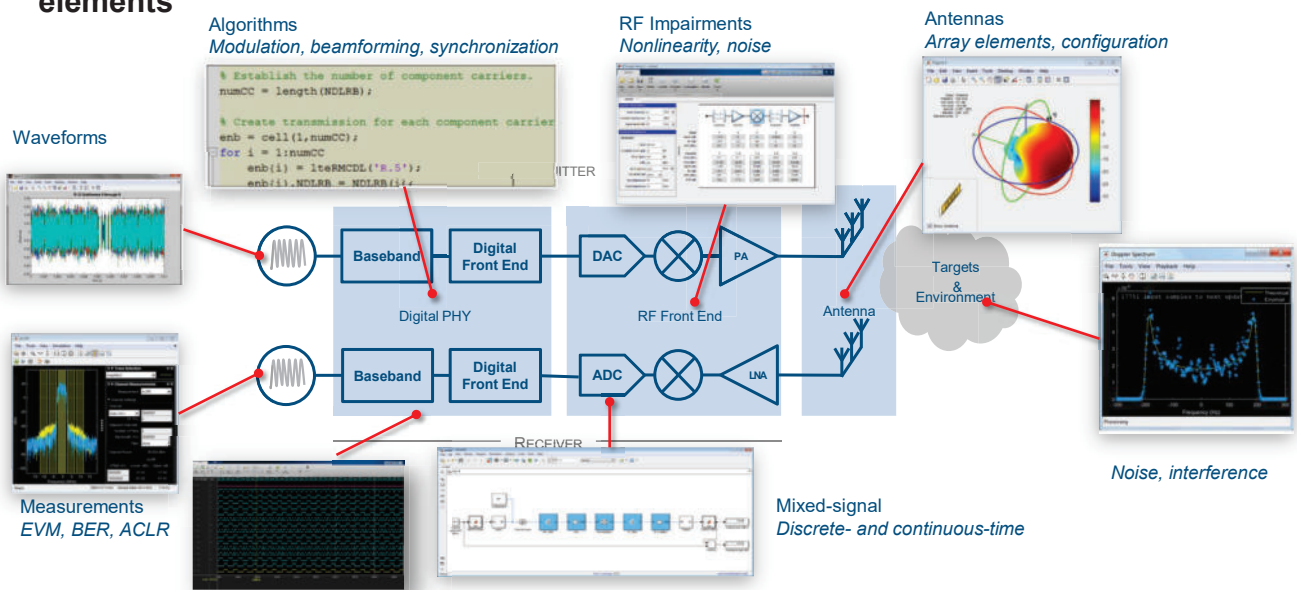
Radar Design as an Example!

Requires ^{at least} 7 different skills to be successful!



Why MATLAB and Simulink?

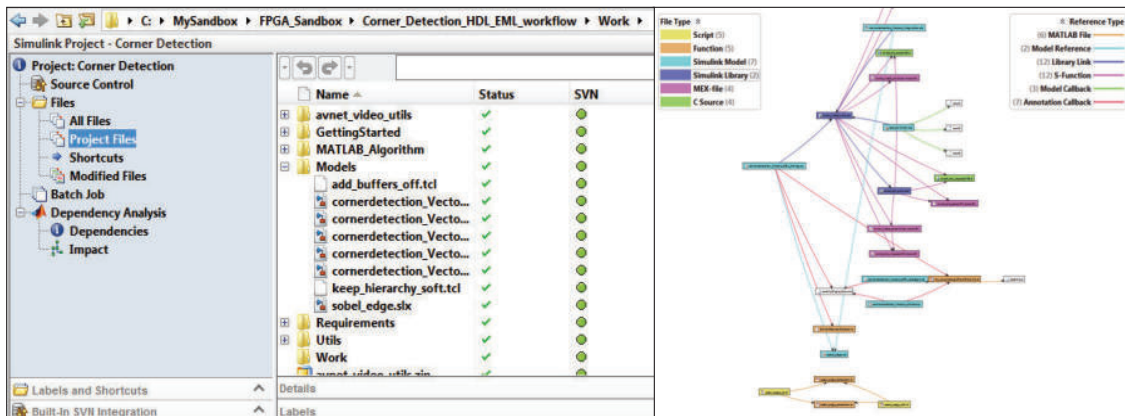
- Rapid and flexible algorithm exploration, design, and analysis
- Unified simulation of digital, RF, mixed signal and antenna elements



Large Scale Modeling

Manage Design Related Files with Simulink Projects

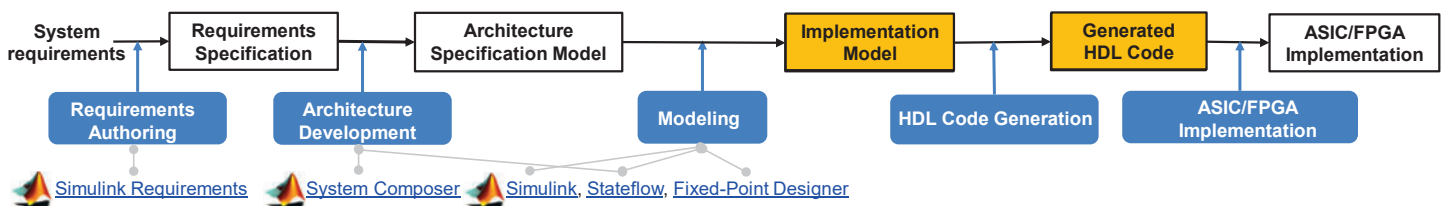
- **Search**, **manage**, and **share** related files in a Simulink project
- Access **version control** functionality (SVN/GIT support built-in)
- **Peer review** of **changes** and **merge** using comparison tools
- **Impact Analysis** before making changes



15

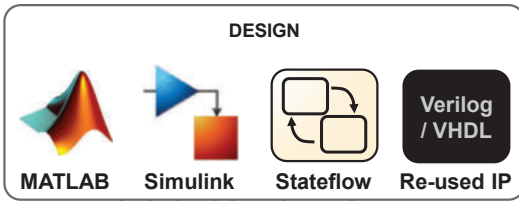
Development Workflow: Modeling

- Add implementation details
- Convert to fixed point
- Optimize architecture
- Generate production quality code



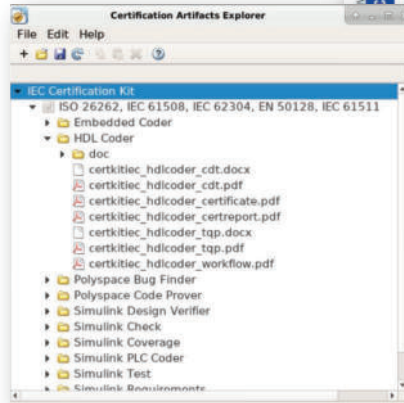
16

HDL Coder - Certified by TÜV SÜD for ISO26262 for any ASIL

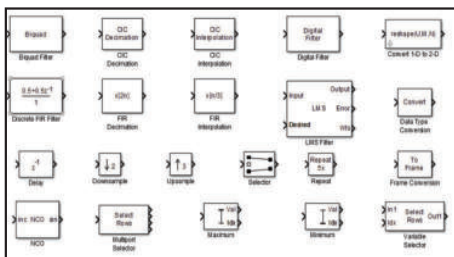


- Conform to ISO 26262 Demonstration Template (QVT)
- TÜV Certificate
- TÜV Report

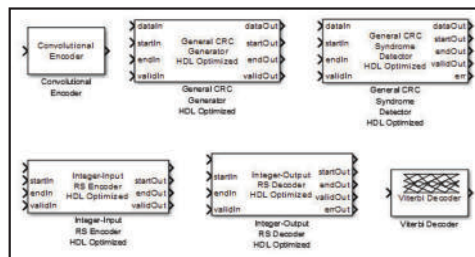
Synthesizable RTL
AXI Interfaces
Synthesis scripts



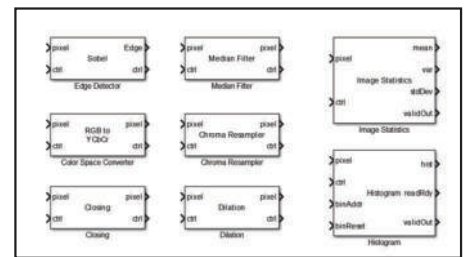
Application Domains for HDL HDL Supported IP



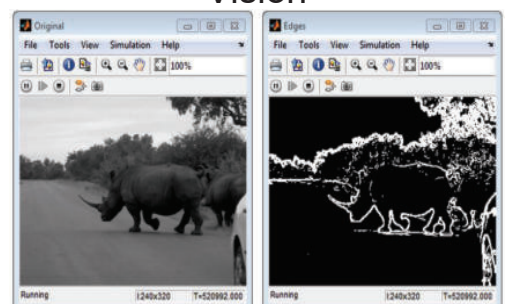
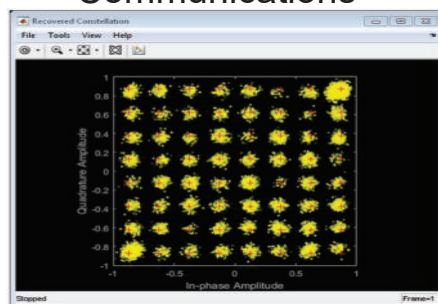
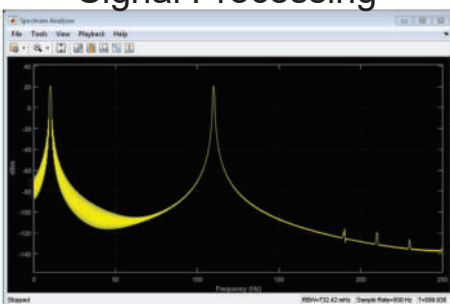
Signal Processing



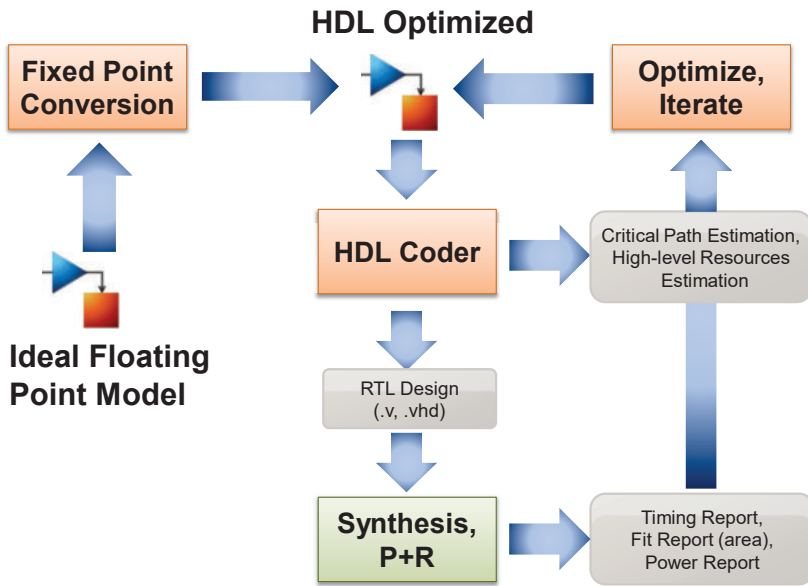
Communications



Vision

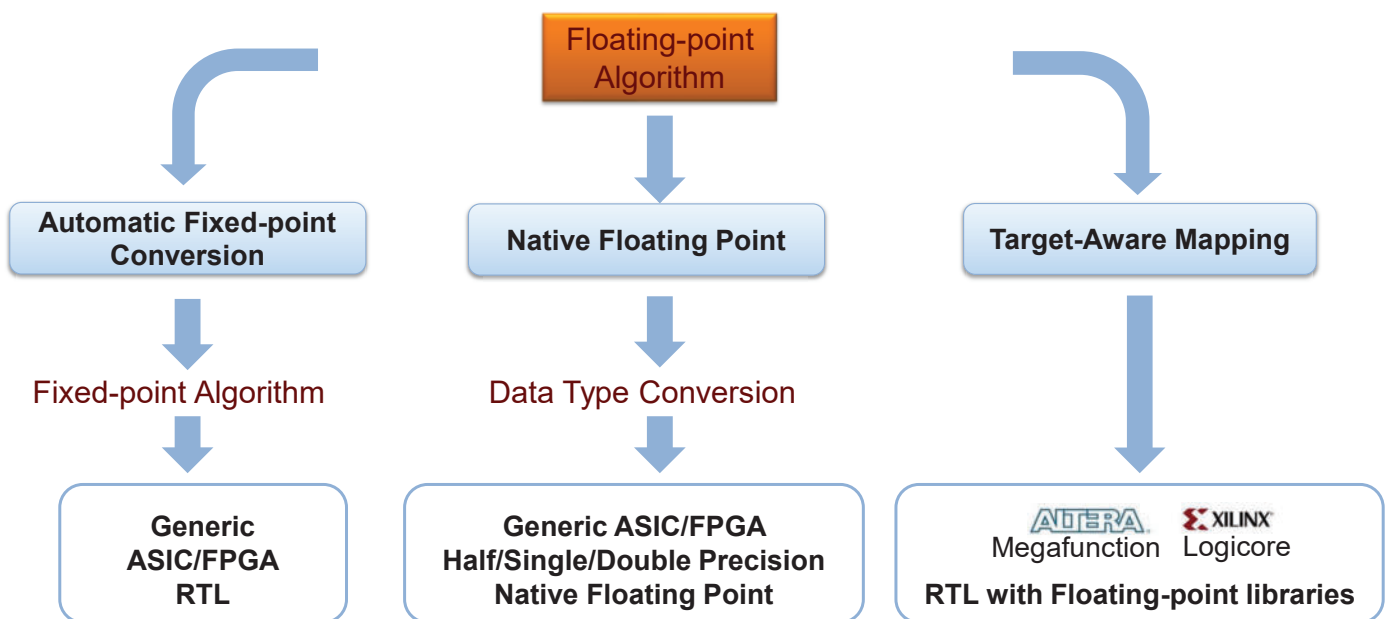


Workflow & Strategies for hardware optimizations



- Fixed-Point Conversion
 - Optimal Fixed-Point will save area and improve critical path
- Architectural choices, e.g.
 - Resource sharing
 - Linear, tree, cascade
 - FCSD, LUT, CORDIC, Shift Add ..
- Pipelining
 - Input / Output pipelining
 - Distributed pipelining

Data Types: HDL Coder Capabilities



Guided and Automated Fixed-Point Quantization

Simulate with representative data to collect required ranges

Fixed-Point Designer proposes data types

Choose to apply proposed types or set your own

Simulate and compare results

Name	Run	CompiledDT	SpecifiedDT	ProposedDT	Accept	SimMin	SimMax
Compute Power/Add: Accumulator	Ranges(...)	double	Inherit: Inh...	n/a	<input type="checkbox"/>	0	0.0714272
Compute Power/Add: Output	Ranges(...)	double	Inherit: Inh...	fixdt(0,16,19)	<input checked="" type="checkbox"/>	0	0.0714272
Compute Power/Product	Ranges(...)	double	Inherit: Inh...	fixdt(0,16,19)	<input checked="" type="checkbox"/>	0	0.0713574
Compute Power/Product1	Ranges(...)	double	Inherit: Inh...	fixdt(0,16,19)	<input checked="" type="checkbox"/>	0	0.0457087

Visualization of Simulation Data

Histograms of all results in the model

Generate Bit & Cycle True, Readable and Traceable Code

Code Generation Report

Find:

Match Case

Contents

- Summary
- Clock Summary
- Code Interface Report
- Timing And Area Report
- High-level Resource Report
- Optimization Report
- Distributed Pipelining
- Streaming and Sharing
- Delay Balancing
- Adaptive Pipelining
- Traceability Report

Generated Source Files

- FOC_Current_Control_pkg.vhd
- Clarke_Transform.vhd
- Saturate_Output.vhd
- D_Current_Control.vhd

Clarke Transform

Converts three-phase quantities into balanced two-phase quantities. The A and B phases are converted to the direct axis (alpha) component and the quadrature axis (beta) component. Components are all dependent on time and speed.

Converts the time domain components of a three-phase system (in abc frame) to two components in an orthogonal stationary frame (αβ). The a and b phases are converted to the direct axis (α) component and the quadrature axis (β) component.

1.1.3.1.a Alpha component

$$i_{\alpha} = i_a$$

1.1.3.1.b Beta component

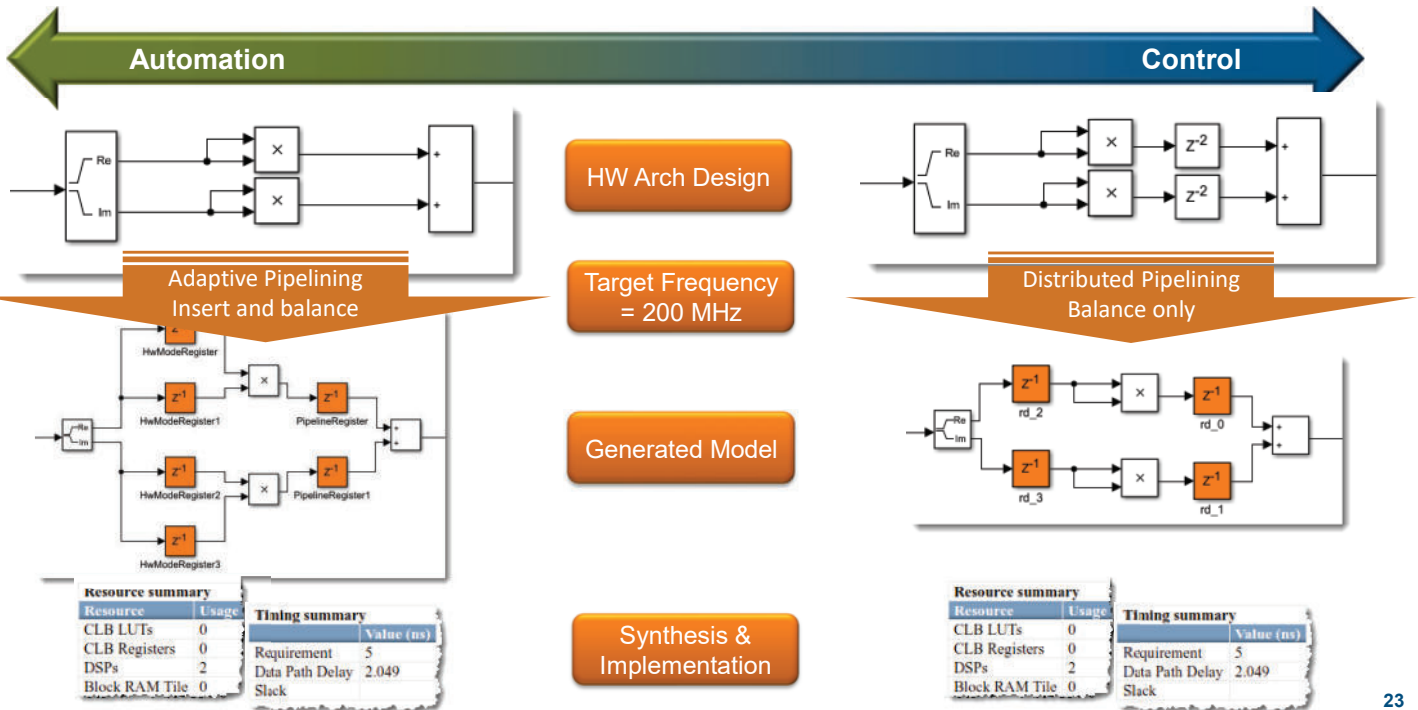
$$i_{\beta} = \frac{1}{\sqrt{3}} i_a + \frac{2}{\sqrt{3}} i_b$$

1.1.3.2. Park Transform

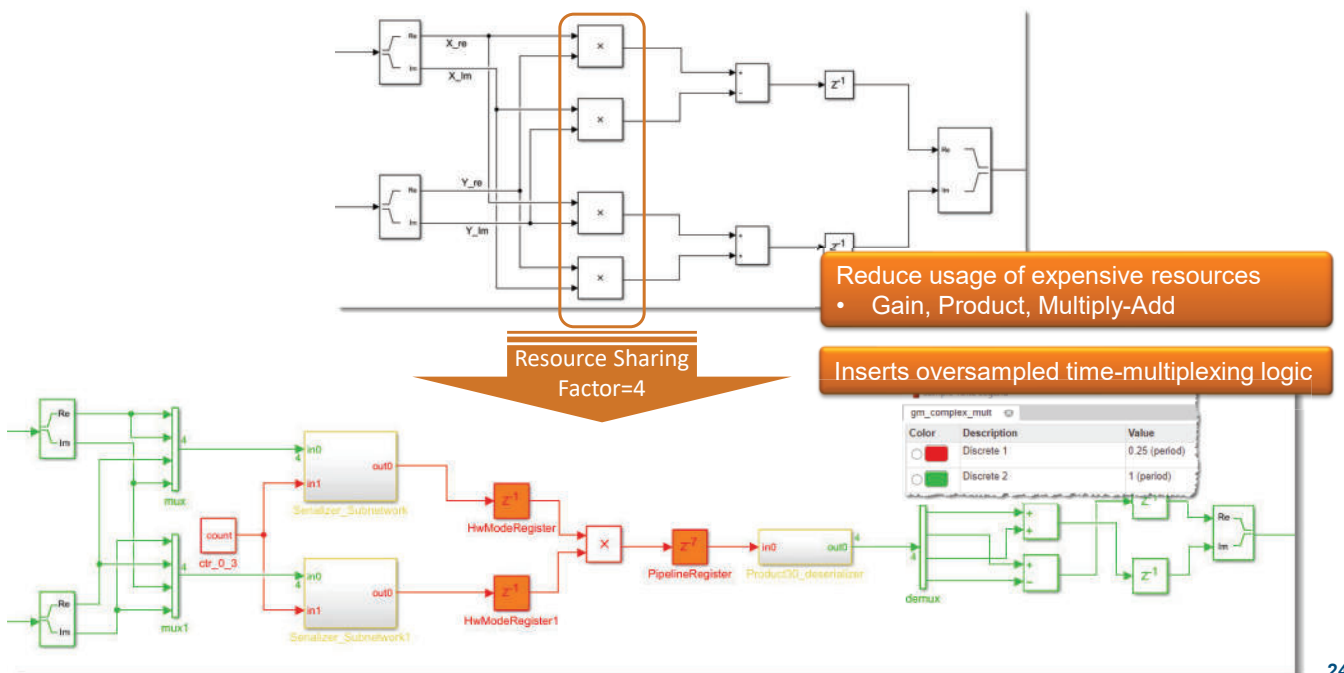
Converts balanced two-phase orthogonal stationary system to an orthogonal rotating reference frame.

Trace between generated RTL to model and requirements

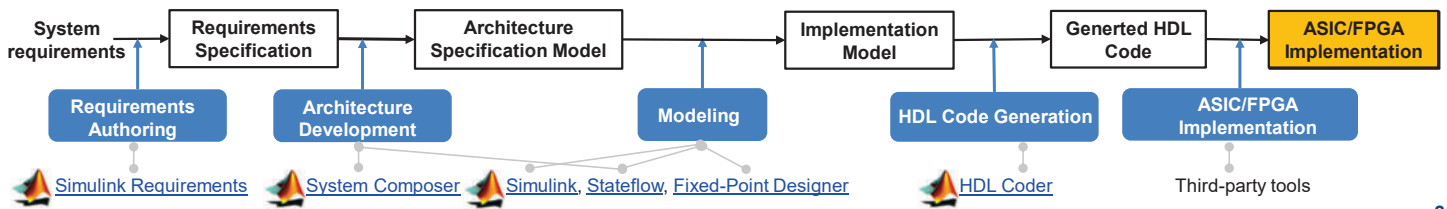
Optimize Timing



Optimize Hardware Resources



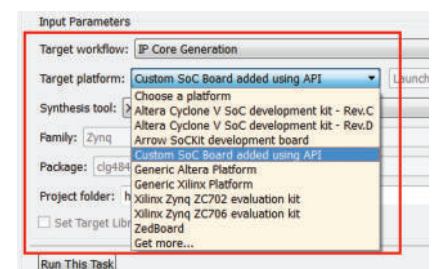
Reference Workflow: Modeling



25

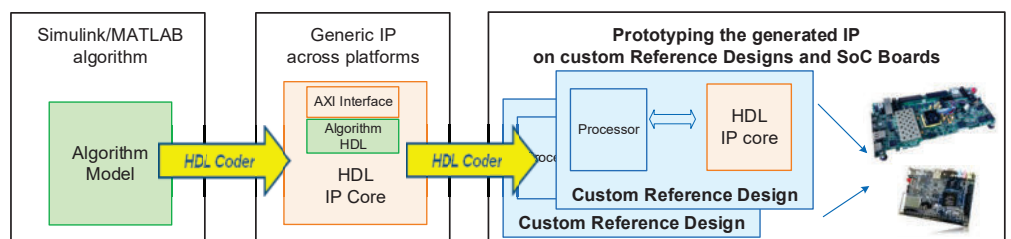
IP Core Generation & Integration

- Generate HDL IP core with standard interfaces
- Use Provided Reference Design
- Define your own Board and Reference Design
- Integrate IP Core Automatically into Reference Design



IP Core Interfaces

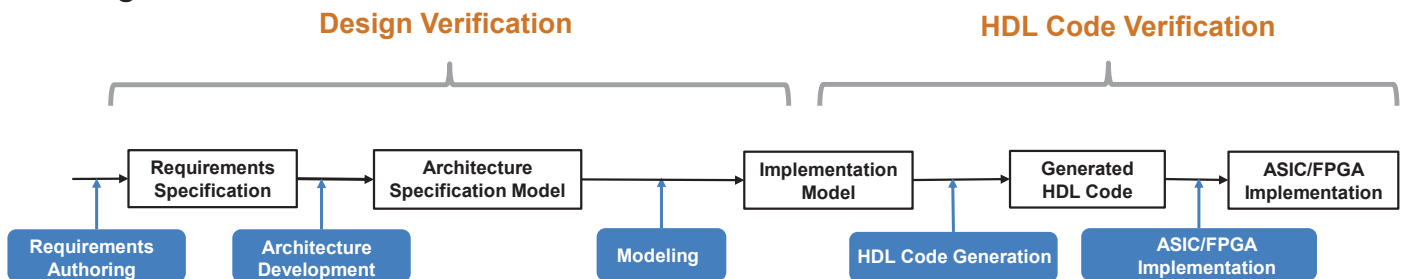
- Internal/External IO
- AXI4 / AXI4-Lite
- AXI4-Stream / Video
- AXI4 Master



26

Verification and Validation in the Model-Based Design Workflow

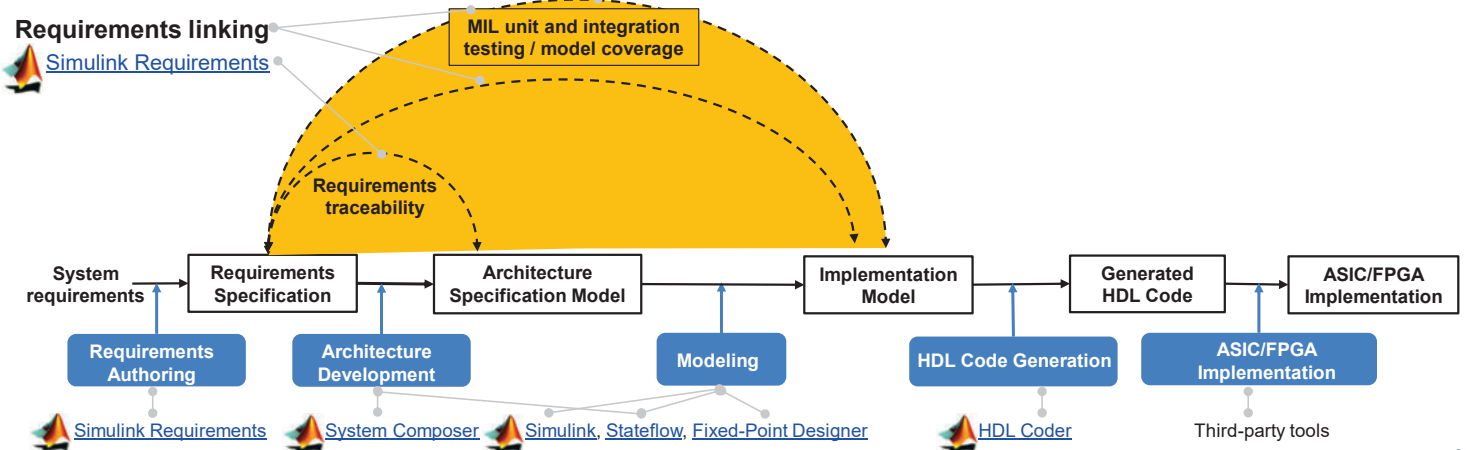
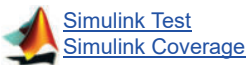
- I. **Verification and validation at the model level (design verification):**
 Demonstrate that the model used for production code generation behaves as specified in its requirements and absence of unintended functionality
- II. **Verification and validation at the HDL code level (HDL code verification):** Demonstrate equivalence between the model and generated HDL code.



27

V&V Workflow: MIL Unit and Integration Testing, Coverage Analysis

Simulation / test authoring / coverage analysis

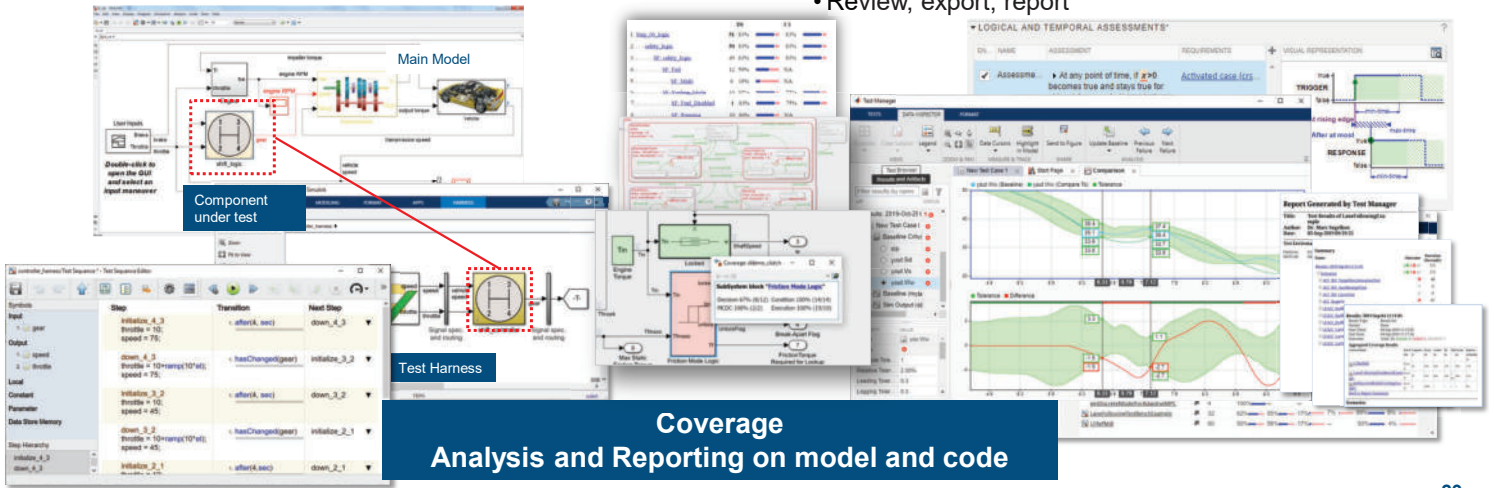


28

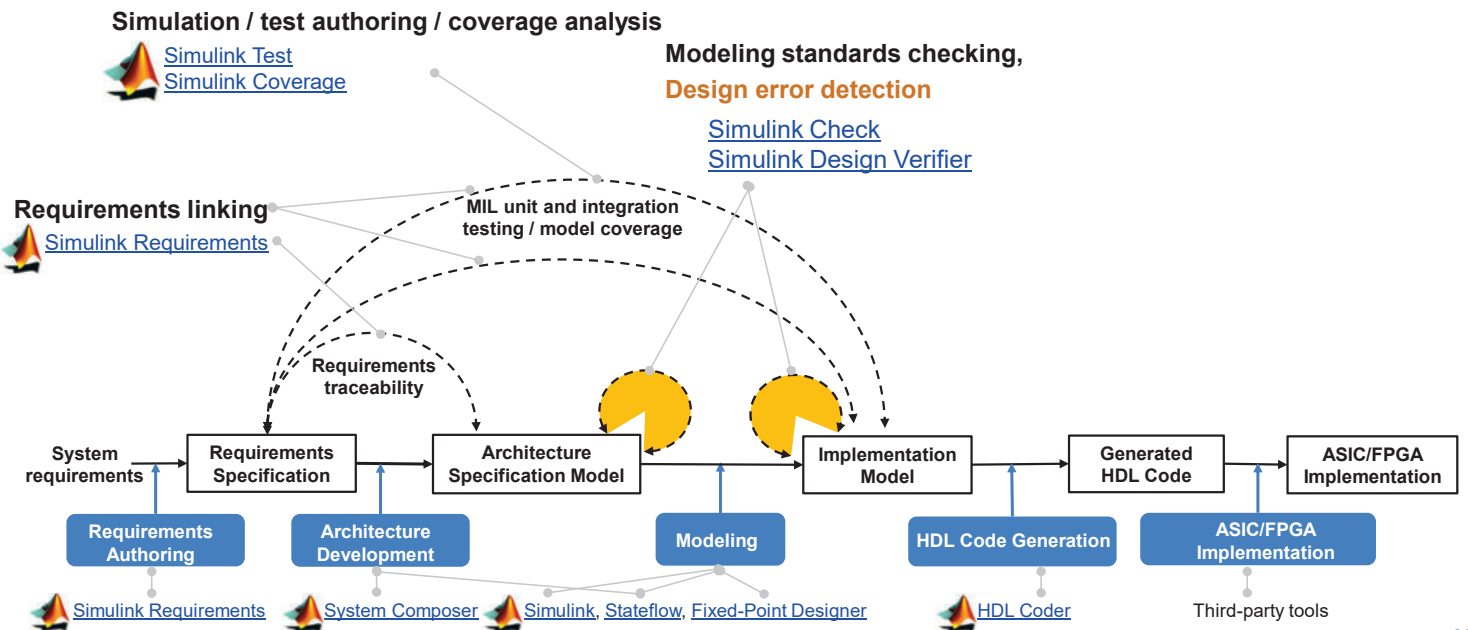
Develop, Manage, and Execute Simulation-based Tests with Coverage Reporting

Test Harnesses
Test Sequence Blocks, Pass/Fail Criteria
 Synchronized, simulation test environment

Test Manager
Test Definitions, Pass/Fail Criteria
 • Author, execute, manage test cases
 • Review, export, report



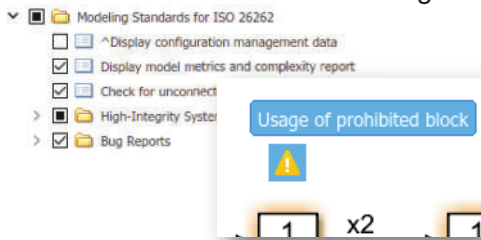
V&V Workflow: Static Model Analysis



Static Model Analysis I : Modeling Standards, Metrics & HDL Compatibility

Standards & Guidelines Checks

- **Automate** compliance to ISO26262
- **Customize** checks
- **Find and fix** compliance issues while you design with Edit Time Checking



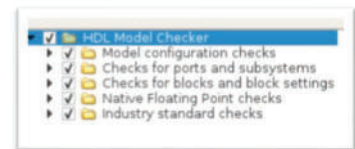
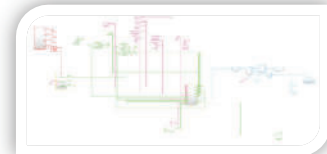
Model Metrics

- **Analyze** complexity, size, reusability
- **Assess** design quality



HDL Code Advisor

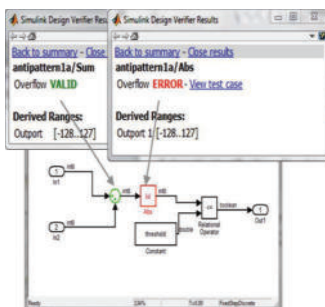
Model compatibility with HDL Code Generation



Static Model Analysis II : Formal Verification

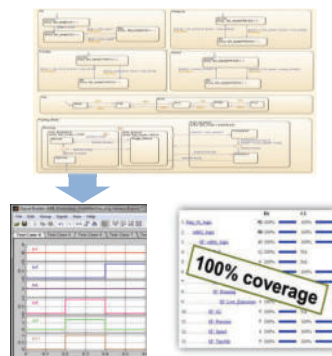
Design Error Detection

- **Uncover** hard to find dead logic and design flaws



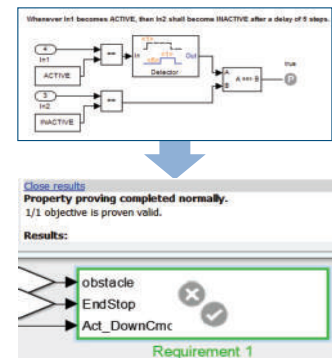
Test Generation

- **Automate** test case generation to complete coverage

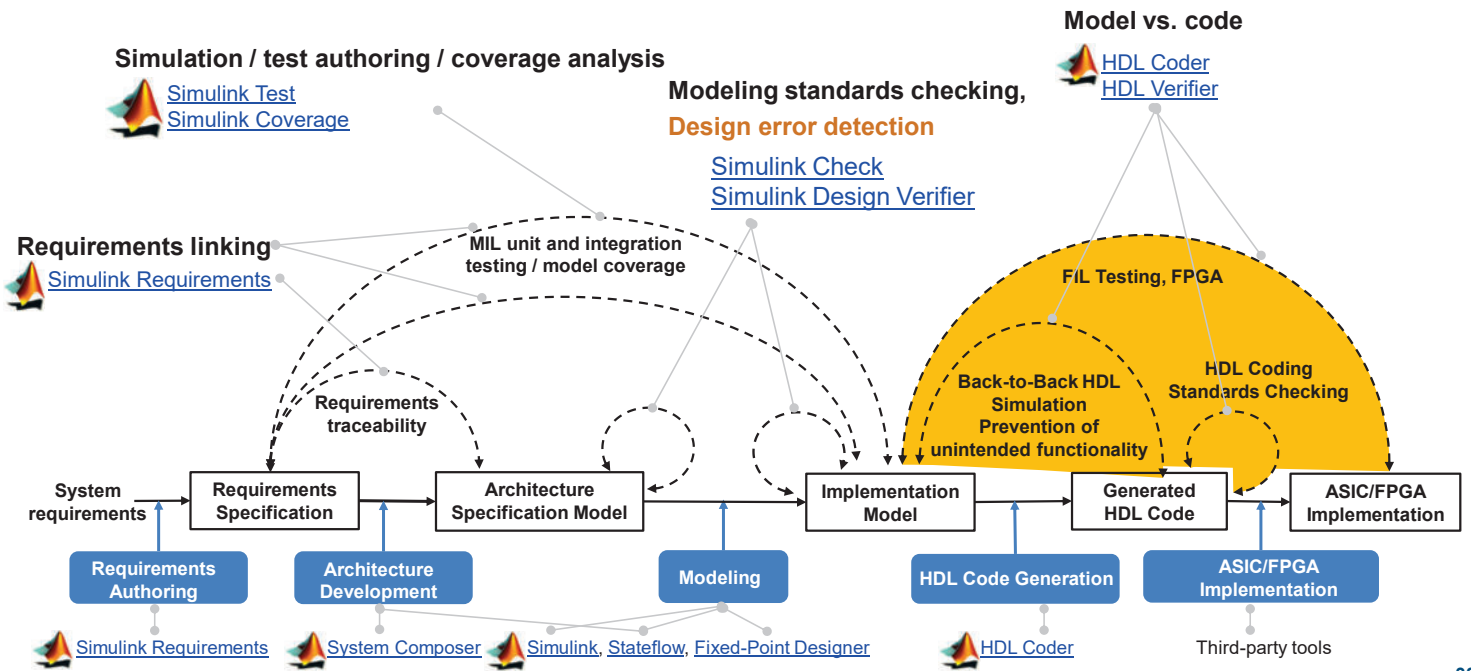


Requirements Proving

- **Prove** formally design meets requirements

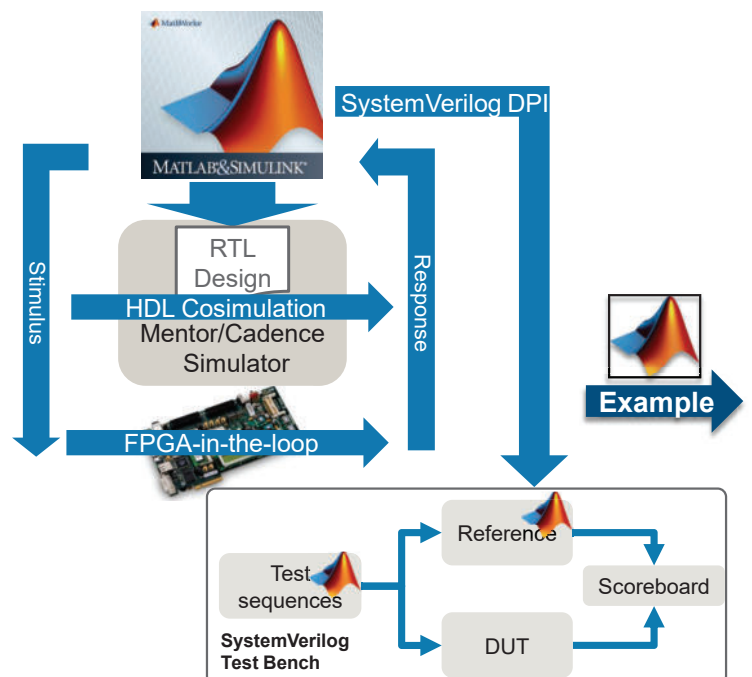


V&V Workflow: Static Model Analysis



Verify and Debug with MATLAB and Simulink

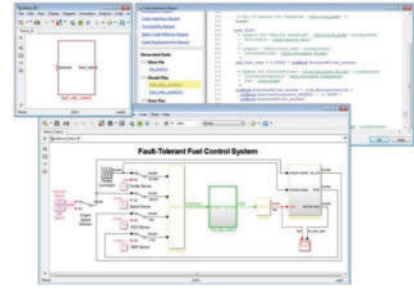
- Use HDL Verifier to:
 - **Cosimulate** RTL back-to-back with the model to debug before FPGA deployment
 - Simulate **FPGA-in-the-loop** with your MATLAB/Simulink tests
 - Generate **SystemVerilog DPI-C** components for the verification team:
 - Reference models
 - Test sequence items
 - External models



Model Based Design for FPGA/ASIC

What is your value?

Enable collaboration by integrating workflows with **Model-Based Design**



Shorten development time and react faster to changing requirements with **Automatic HDL Code generation**

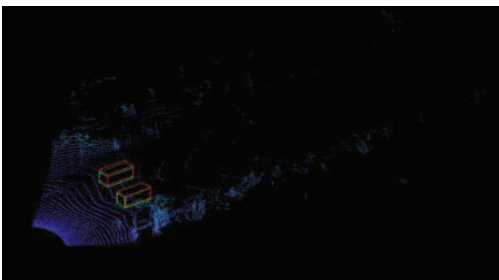
Reduce verification time with **HDL/FPGA Co-simulation** and **increased reuse with automatic test bench generation**

Streamline ISO26262 certification with **IEC Certification Kit** and a **certified toolchain**

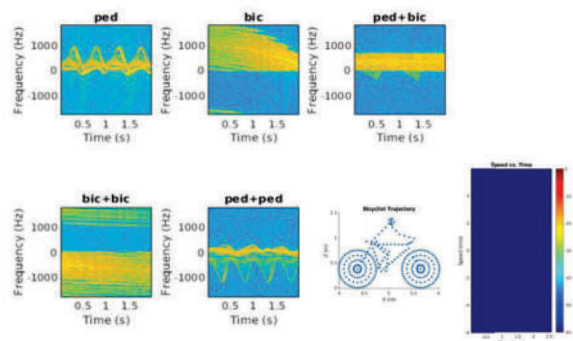
Deep Learning on FPGA

➤ Deep Learning is state of the art for many perception problems for AD

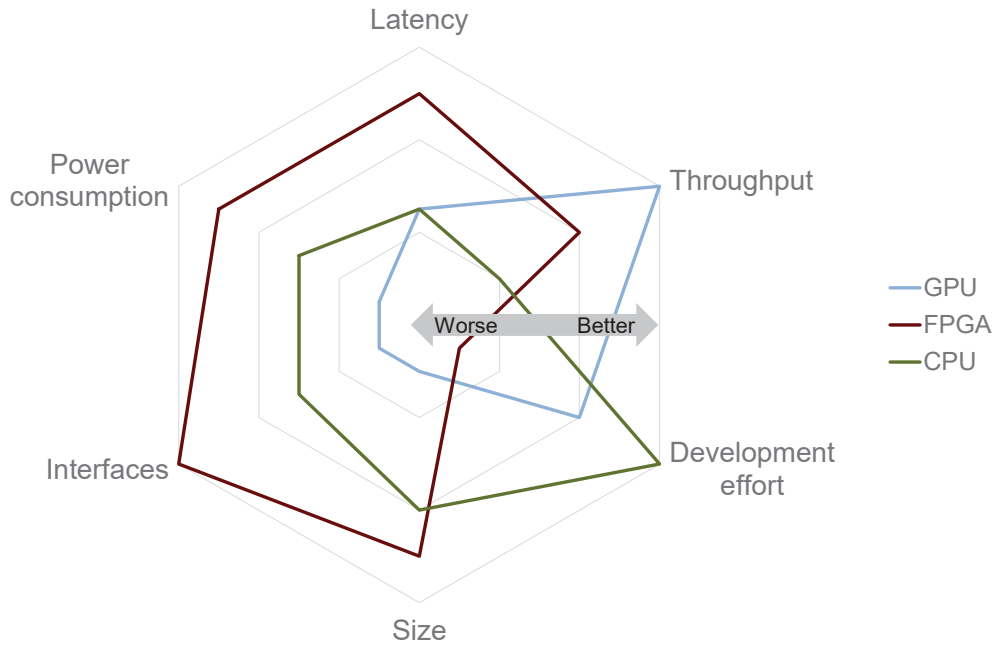
Lidar Object Detection



Radar Signature Classification



Why deploying DNNs on FPGA?



37

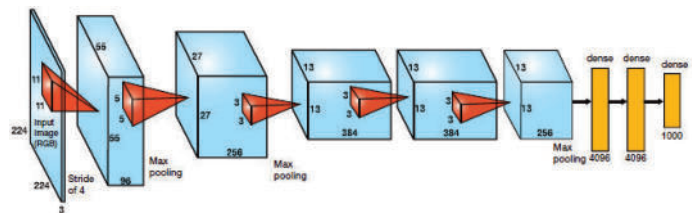
Challenges of deploying Deep learning models on FPGAs

- Large scale matrix computations
 - TFLOPS: 230M weights and 724M MACs
- Complex architecture
 - Scale of data movement across the DDR
- Manual workflows are tedious

	input	conv 1	conv 2	conv 3	conv 4	conv 5	fc6	fc7	fc8	Total	
Parameters (Bytes)	n/a	140K	1.2M	3.5M	5.2M	1.8M	148 M	64M	16M	230 M	DDR
Activations (Bytes)	588K	1.1M	728K	252K	252K	168K	16K	16K	4K	3.1 M	BRAMs
FLOPs	n/a	105 M	223 M	149 M	112M	74M	37M	16M	4M	720 M	DSPs

Workflow:

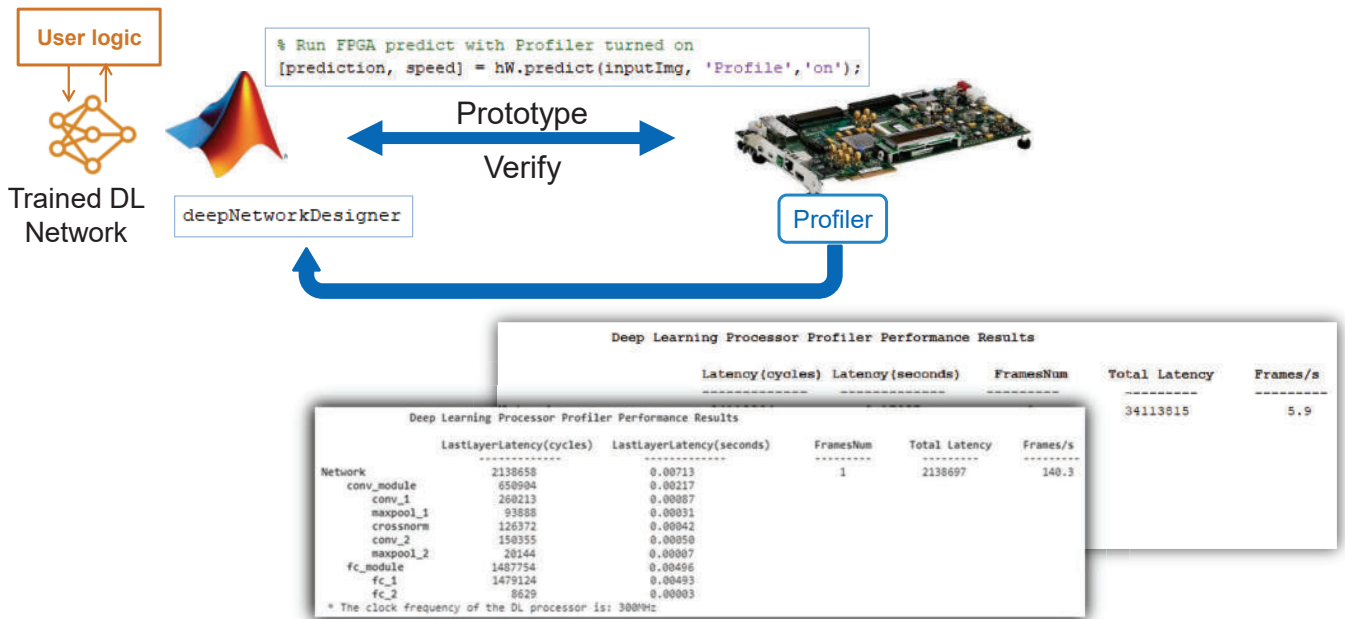
- Exploring multiple networks
- Exploring the resource and performance tradeoffs



Deep learning networks are too big for FPGAs

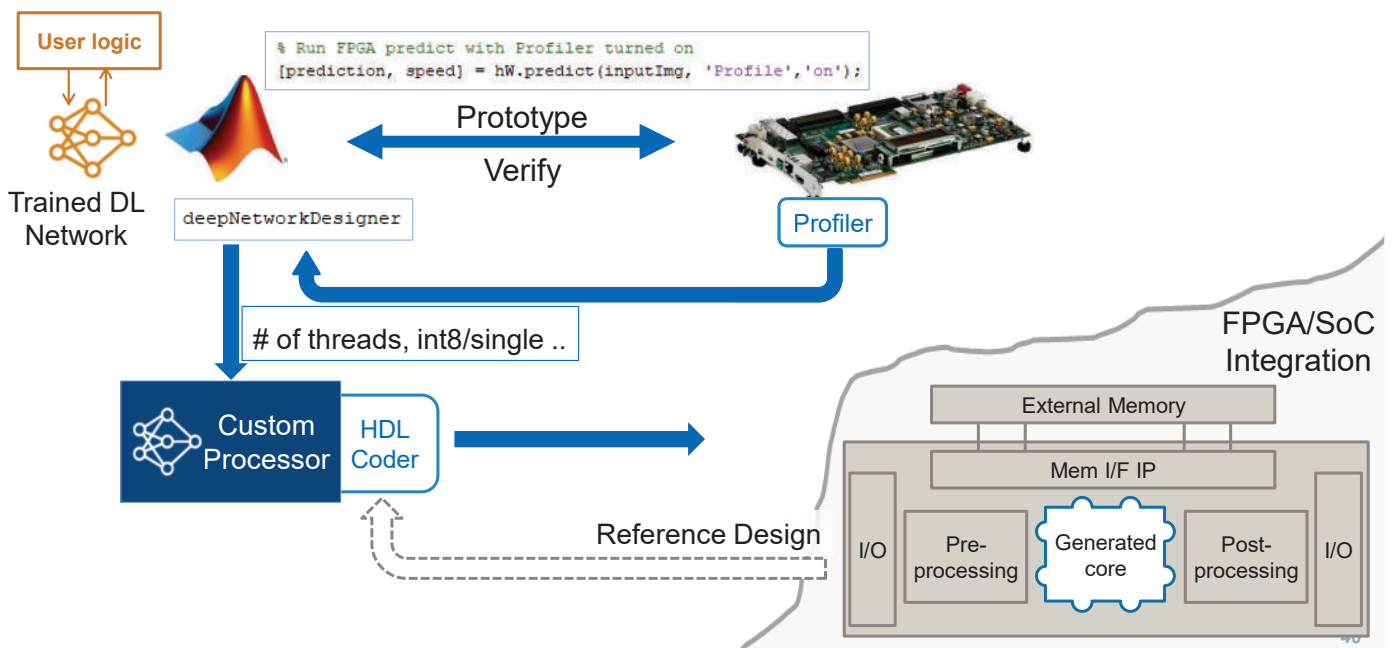
38

Prototyping: Design Exploration and Customization

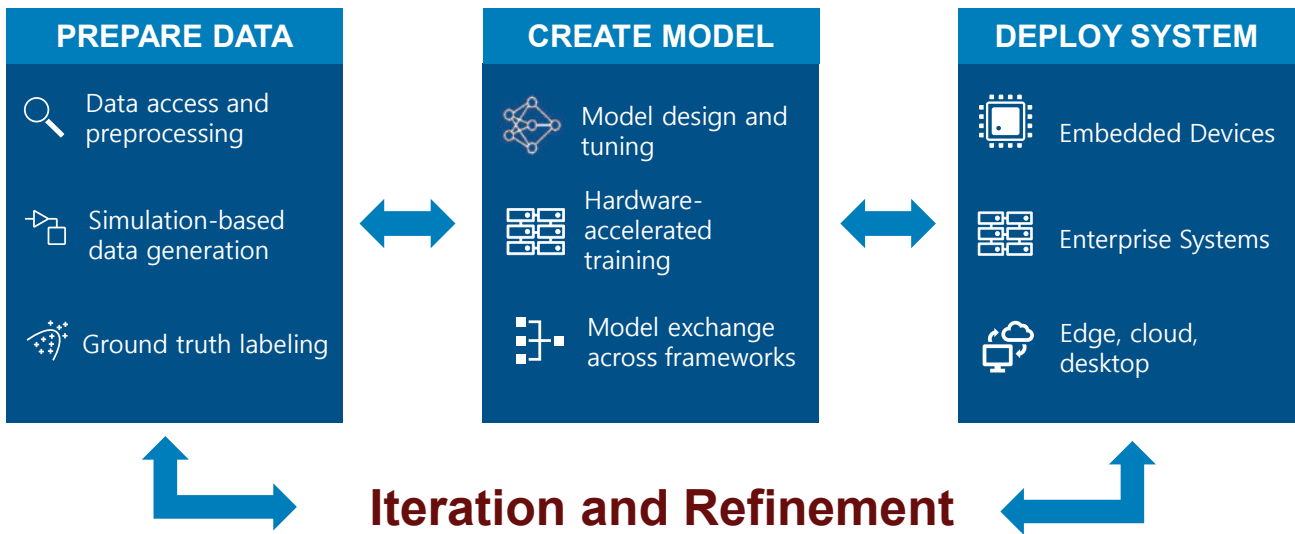


39

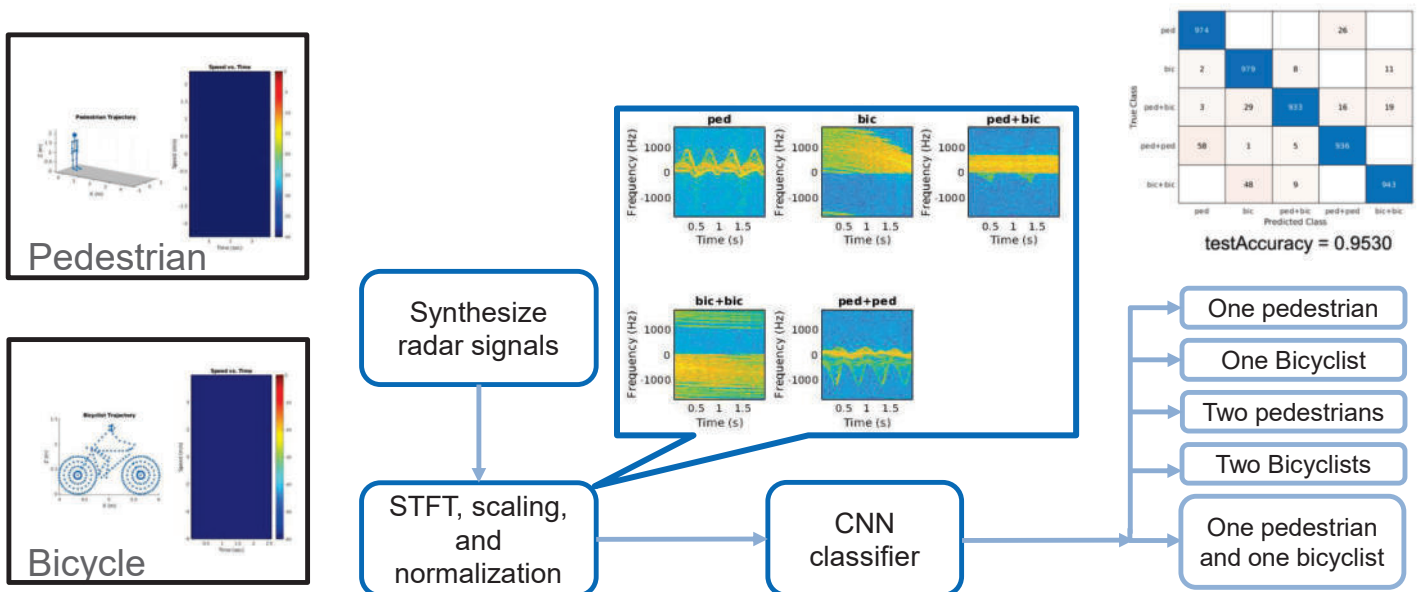
Generate Custom Processor for FPGA/SoC Integration



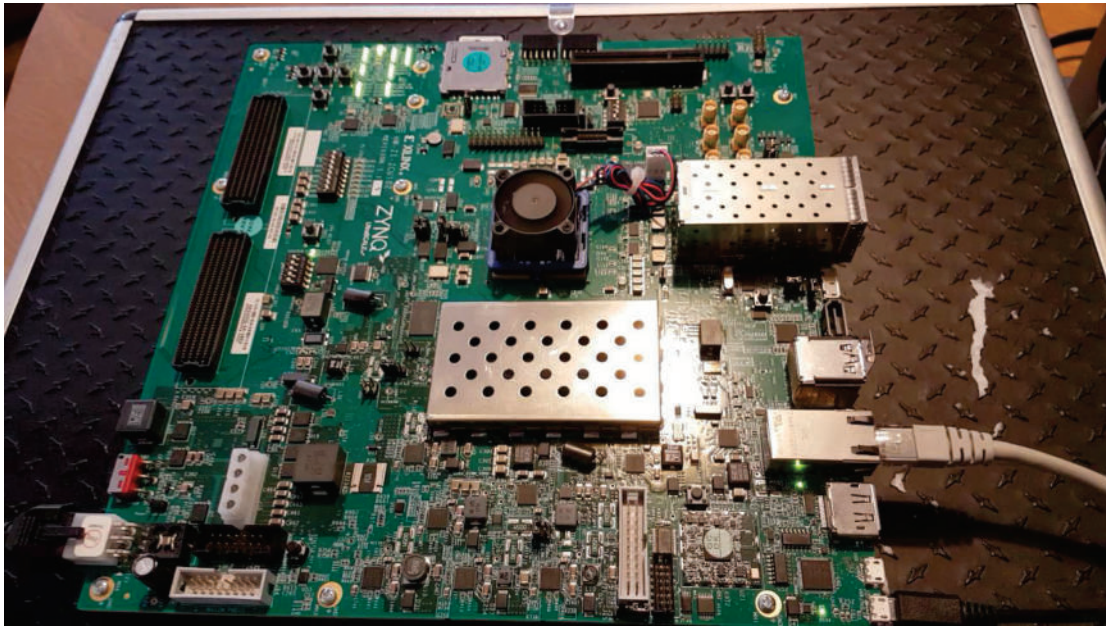
MATLAB supports the entire **deep learning workflow** – from **Data to Deployment**



Pedestrian and Bicyclist Classification Using Deep Learning



Scene Recognition using Radar on Xilinx ZCU102



43

Thanks for your attention
Questions?

Dimitri Hamidi
Senior Application Engineer
MathWorks
dhamidi@mathworks.com



44

PCIe-over-TCP-over-TSN-over-10/25GigE

Dr. Endric Schubert, CTO

Presentation at
"Programmable Processing for the Autonomous / Connected Vehicle" Sep-24 2020

<https://innosued.de/workshop-programmable-processing-for-the-autonomous-connected-vehicle/>

Outline

WHY

- Automotive needs 10 Gig networking, or more! Electric vehicles and ADAS / Automated Driving push the migration from Domain-based over to Zone-based Architectures, which again pushes for more bandwidth and real-time capabilities in the Automotive Network.

WHAT

- Out patent pending technology integrates IEEE Standards for Time-Sensitive Networking with Heterogeneous Packet Tunneling for PCIe, 100Base-T1, MIPI CSI-2, CAN-FD, and others. Utilizing Protocol Acceleration in hardware we can scale to 10 Gbps linerates, and beyond.

HOW

- A Software and Semiconductor IP core subsystem which integrates 3rd party IP from German Fraunhofer with technology from MLE and can be licensed for FPGA and ASIC implementations.

Backgrounder Missing Link Electronics

Our Mission is to

- Apply FPGA technology for Domain-Specific Architectures
- Offer pre-validate FPGA subsystems of FPGA IP blocks and open-source software
- Support customer projects with deep expertise and hands-on design services

Head-quartered in Silicon Valley with Design Offices in Germany

- Founded 2010, employee owned
- 15+ Certified FPGA Designers
- 50+ Presentations at Technology Conferences, 4 Patents

Technology Partnerships



2020-09-24

Company Confidential

3

Our Technology Achievements

- Patented technology in the fields of networking, mixed-signal, functional safety
 - US Patent 9,209,828 - Configurable Mixed-Signal Systems
 - US Patent 10,140,049 - Partitioning Systems Operation in Multiple Domains
 - US Patent 10,509,880 - Automation for Configurable Mixed-Signal Systems
 - US Patent 10,708,199 - Heterogeneous Packet-Based Transport
- 50+ Presentations at Technology Conferences and in Technology Journals
 - Embedded World Conferences
 - PCI-SIG Developers Conferences
 - IBM Open Power Summit
 - SNIA Storage Developers Conferences
 - Flash Memory Summits
 - Xilinx XCELL Magazines
 - XILINX Developer Forum and Security Workshops



2020-09-24

Company Confidential

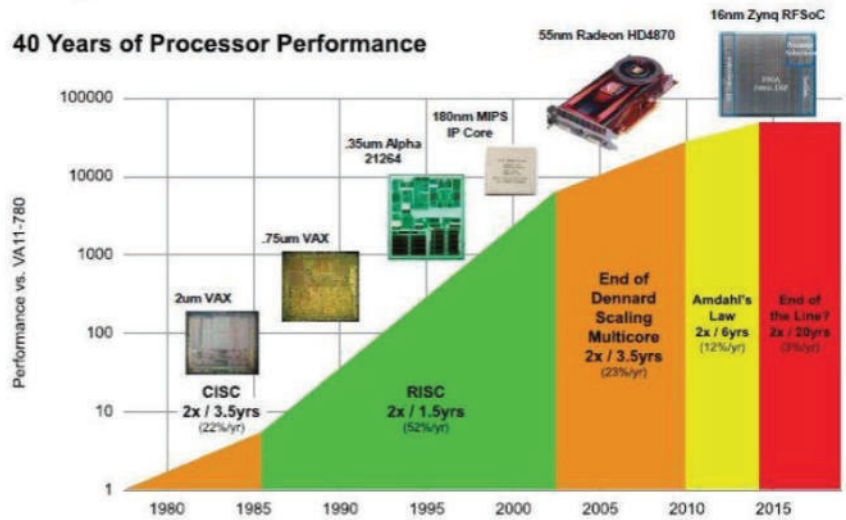
4

Domain-Specific Architectures

FPGAs as a very powerful processor that executes "dataflow software"

Apply HPC/HA Datacenter technology to other verticals - "Proudly borrowed elsewhere!"

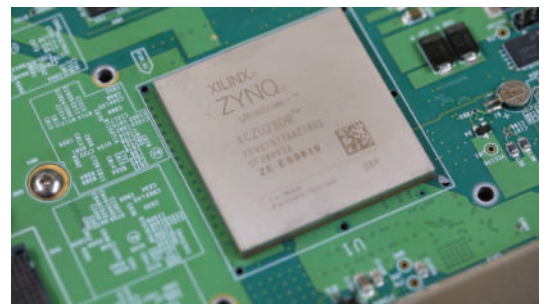
Challenges: The End of Moore's Law and Scaling



Source: John Hennessy and David Patterson, Computer Architecture: A Quantitative Approach, 6e 2018

Our FPGA Design Expertise

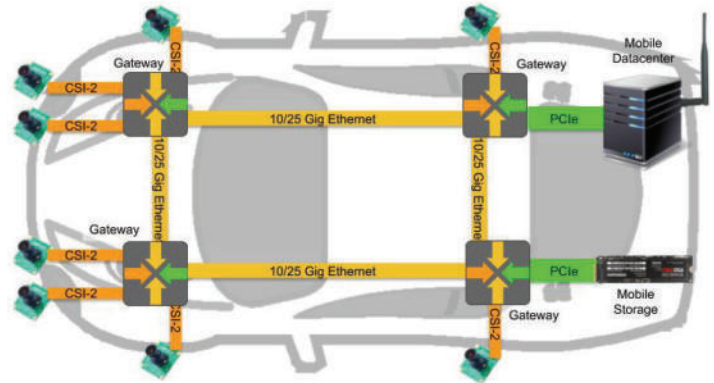
- Mentor, Cadence, Xilinx Toolflows
- Zynq-7000 SoC in designs since Q1/2012
- Zynq Ultrascale+ MPSoC in designs since Q4/2015
- Zynq UltraScale+ RFSoc in designs since Q2/2018
- PetaLinux / Vanilla Linux and Yocto-based SW development
- Multigigabit transceiver configurations
 - PCIe Gen2/3/4, SATA 3/6G, SAS 6/12G, NVMe,
 - CAPI, JESD204B, DP/HDMI, MIPI CSI-2 D-PHY
 - 10/25/4050/100G Ethernet, Low Latency Ethernet
- Radar, civil, mil/aero, automotive
- camera, Lidar, data recording
- Functional Safety Design Flows ISO 26262 (ASIL), IEC 61508 (SIL)
- Security & Trust
 - PUF, Crypto, Trusted Execution Environment, OP-TEE



Zone-Based 10 GigE Automotive Backbone

Do you ...

- Prefer open IEEE standards over closed proprietary ones?
- Need 10+ Gbps bandwidth?
- Need deterministic, low-latency real-time network behavior, namely TSN?
- Need heterogeneous connectivity with PCIe, 100Base-T1, MIPI CSI-2, CAN-FD, etc?*
- Need Functional Safety and Security / Hardware Root-of-Trust?

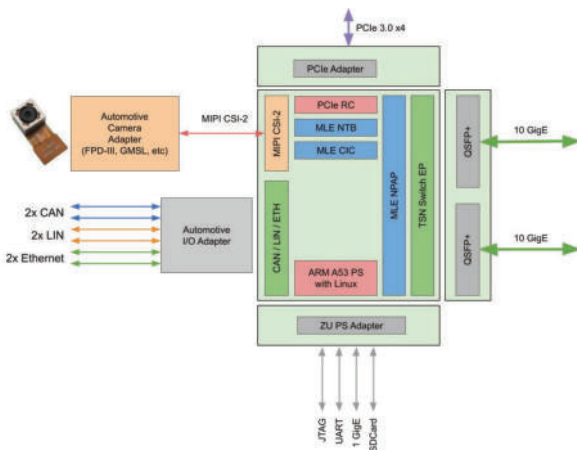


⇒ Use time-to-market solutions from MLE
100% "Made-in-Germany"

MLE LabCar for "Tunneling" PCIe (and else)

FPGA-based Prototyping System for Architecture Exploration and Development

- Example of Zone Gateway Node



- Commercial Product License for FPGA or ASIC implementation
- Sign-once License for complete subsystem including 3rd party IP
- Customization NRE fee

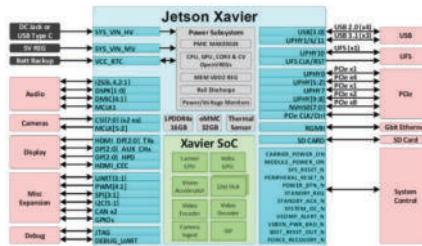
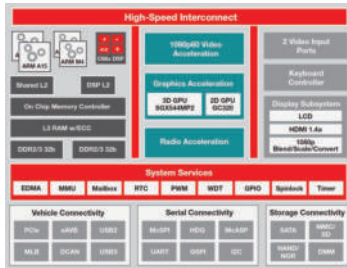


- Prototyping example: 4-node "Lab Car", incl. Hardware, Software, Firmware and licenses for in-house evaluation and FPGA development

⇒ Use time-to-market solutions from MLE
100% "Made-in-Germany"

Why PCIe?

- Future-proof road-map, driven by PCI-SIG
- PC, Cloud Computing, Embedded Systems drive this roadmap
- Best-in-class price (\$) per performance (Gbps) ratio
- Modern automotive SoCs all support PCIe



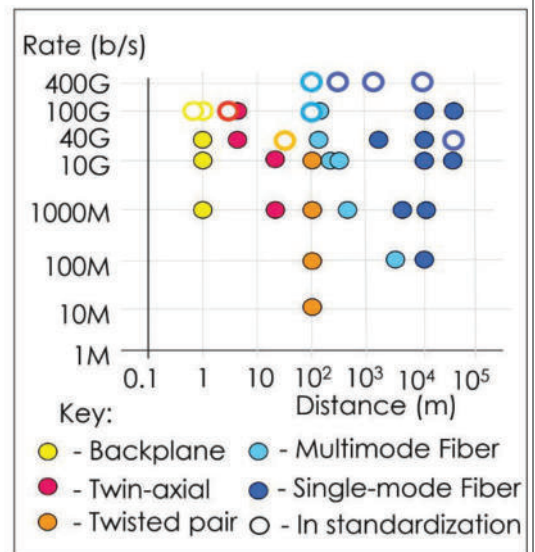
Why Ethernet?

- Future-proof road-map, driven by PCI-SIG
- PC, Cloud Computing, Embedded Systems drive this roadmap
- Best-in-class price (\$) per performance (Gbps) per length (meters) ratio

Distance vs Speed

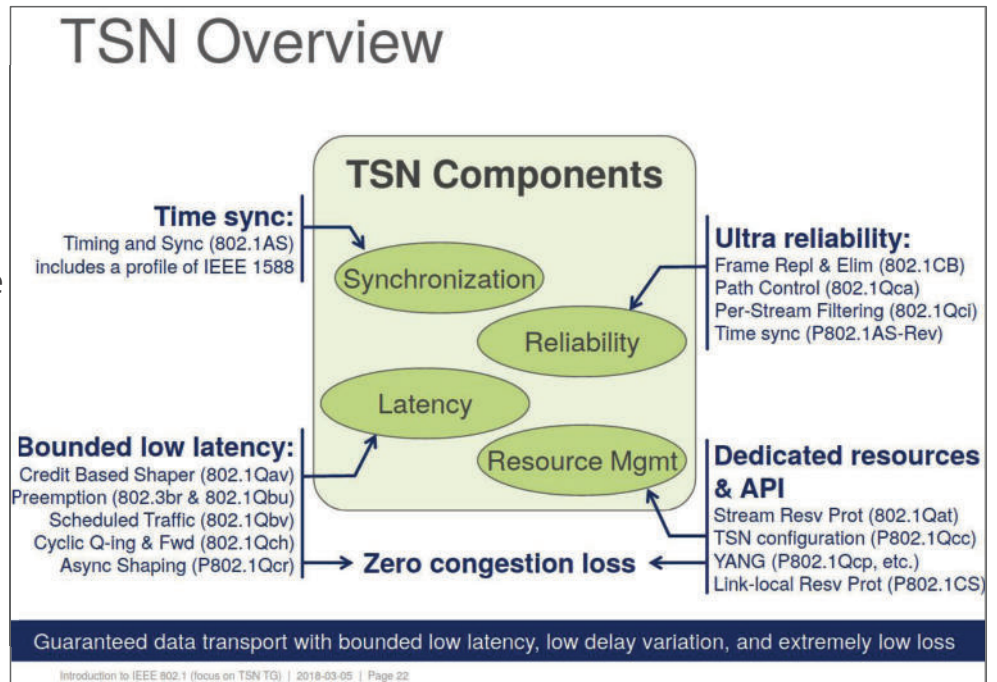
Ethernet operates at different speeds over different distances depending on the media :

- backplanes up to 1m
- Twinax to 15m
- Twisted pair to 100m
- Multimode fiber to 5km
- Single-mode fiber to 40km



Why TSN?

- IEEE open standard for real-time networking
- Telco and Datacenter drive the roadmap



Unique Technology Combination

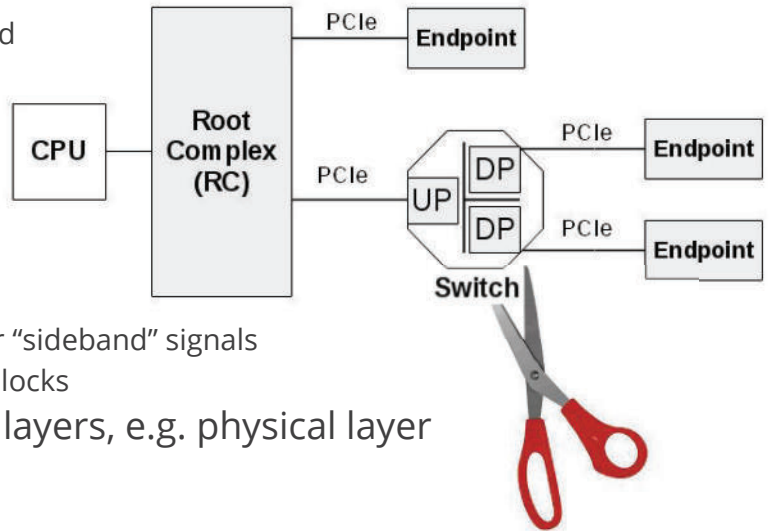
1. PCIe Range Extension via Robust, Long-Range Protocol Tunnels
2. PCIe Non-Transparent Bridging (NTB)
3. PCIe / NVMe Full Acceleration
4. TCP/UDP/IP Full Acceleration (Fraunhofer HHI)
5. Time-Sensitive Network IP (Fraunhofer IPMS)

⇒ Real-Time Multi-Protocol Heterogeneous Packet-Based Transport

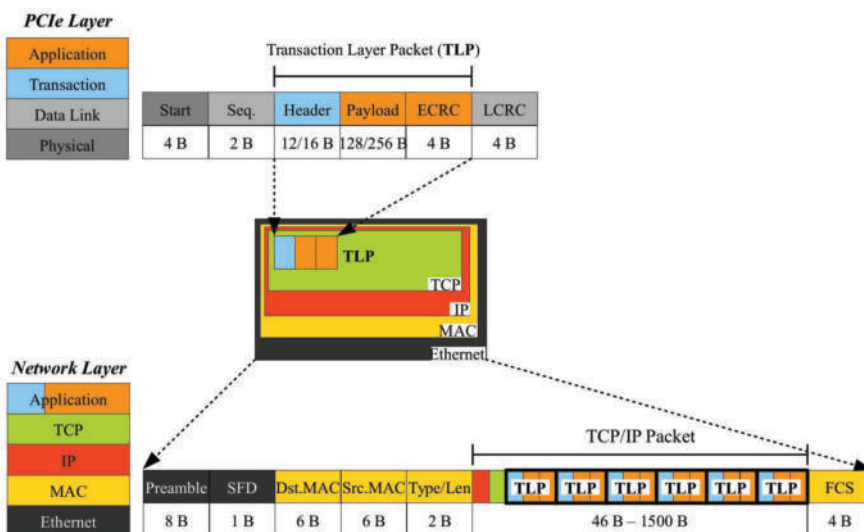
Licensed by other Xilinx customers as Platform Subsystem

PCIe Range Extension via TCP/IP

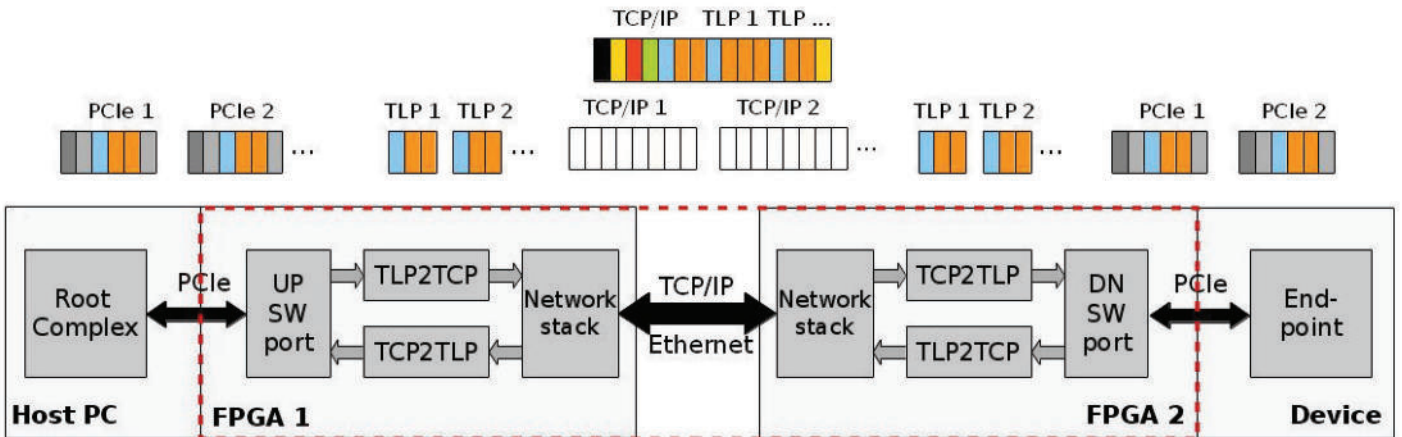
- Fully transparent to network equipment
 - Just a bunch of TCP sessions
 - No special traffic handling required
- Fully transparent to PCIe
 - Reliable transport via TCP
 - Congestion control via TCP
- A “distributed” PCIe Switch
 - In accordance to PCIe Spec
 - Scalable via TCP session count
 - Supports latency requirements for “sideband” signals
 - Special care needed to avoid deadlocks
- Independent of lower network layers, e.g. physical layer



Concept of PCIe-over-TCP (1)



Concept of PCIe-over-TCP (2)



----- Distributed PCIe Switch

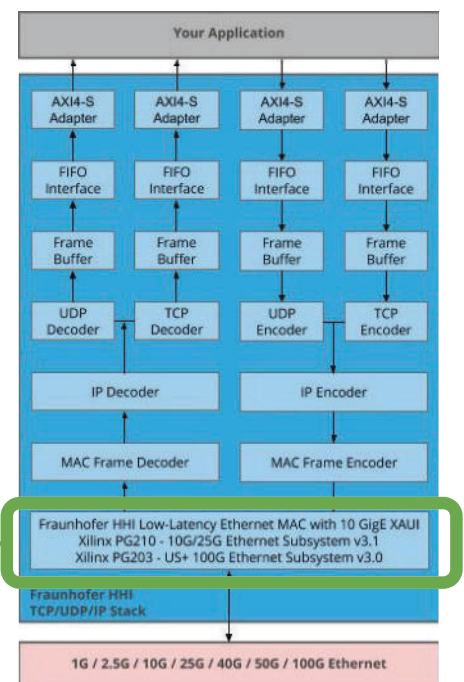
TCP/UDP/IP Full Accelerator

- "Packet Processing" in hardware (FPGA/ASIC)
 - Low latency in microseconds
 - Deterministic (no CPU, no cache misses)
 - Performance scales to 100 Gbps
- In accordance to IETF RFC 1122

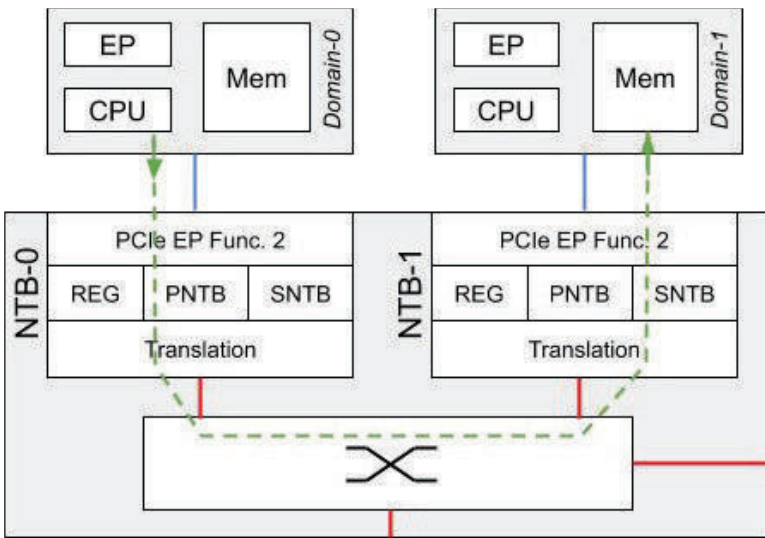
Mature technology licensed from Fraunhofer HHI



Extensible Real-Time Behavior via TSN MAC



PCIe NTB - enables CPU-to-CPU Direct Comm



- Combines Network-on-Chip with PCIe Upstream Ports
- “Remote” DMA plus Write-Only Communication
- Not a PCIe Standard, but “blessed” by PCI-SIG

PCIe (MR-IOV is abandoned as of PCIe Spec 6.0)

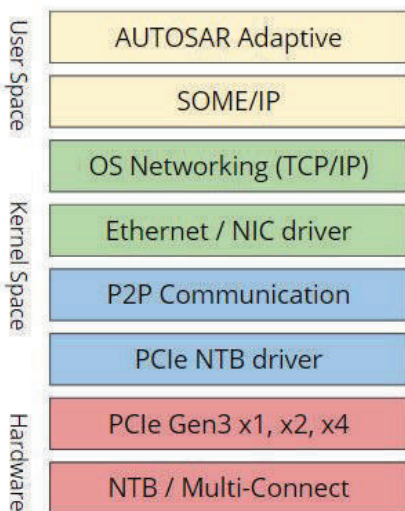
Switch Interconnect

Data Path

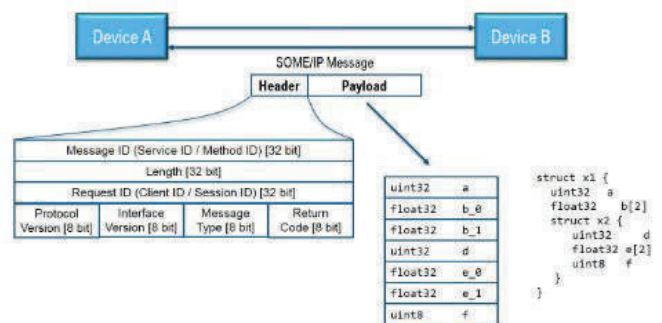
PCIe NTB - Via Well Known Network API

Software Stack on Compute Node
(Linux, QNX, Adaptive AUTOSAR, ...)

Application Programmer's View



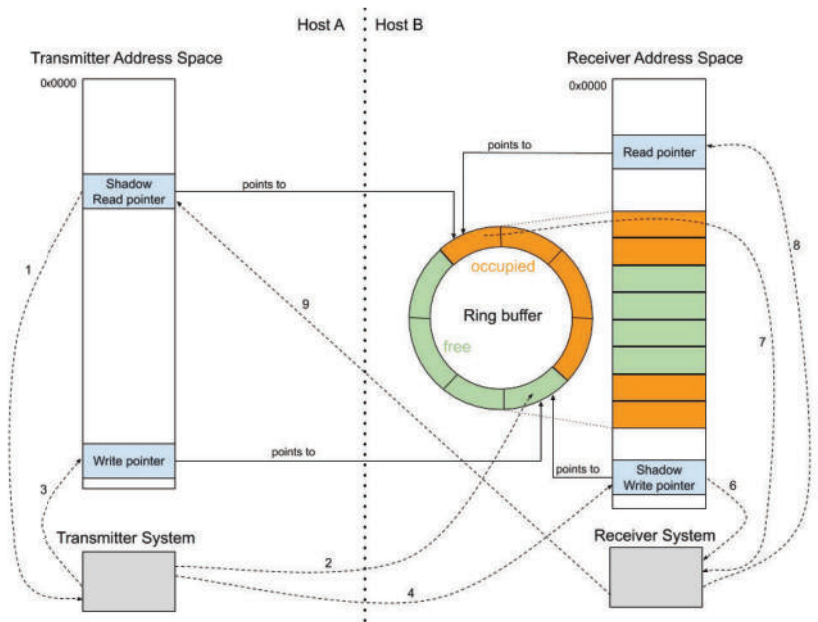
- Fully transparent comm. via PCIe NTB
 - Local (within one ECU)
 - Remote (between multiple ECUs)
- IP address for each Compute Node
- Gateway does routing, fail-over re-routing
- Send/receive TCP/IP, UDP/IP, SOME/IP messages



PCIe NTB High Performance Delivered

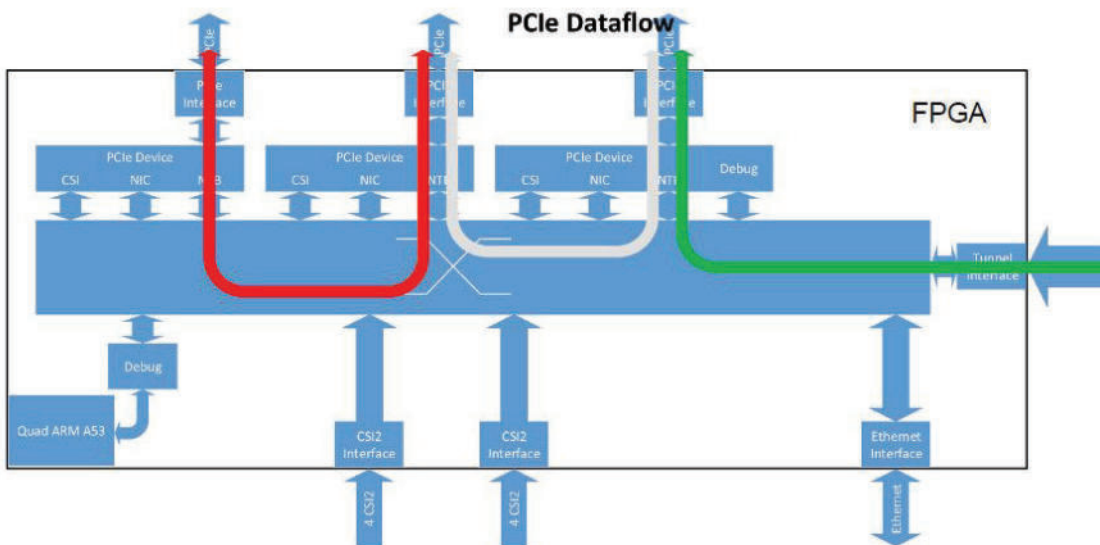
- Write-Only Comm via Doorbells and PCIe Posted Writes
- Avoids difficulties of PCIe multi-device
- Scales to > 32 PCIe RCs

“Borrowed” from NVMe Spec



Network of PCIe - Onchip, Offchip, Backbone

Custom Switch Based Design



Platform Technology Example

Fully integrated system stack for automotive connectivity

- Available for ASIC and/or FPGA implementation

TSN Features:

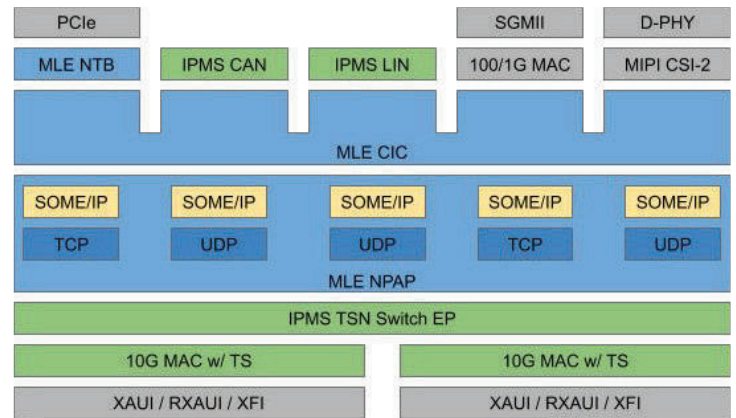
- IEEE 802.1AS, 802.1Qav, 802.1Qbv, 802.1Qci, 802.1Qcb
- Switch and Endpoint mode
- Scales to 10/25 Gbps

TCP/UDP/IP Features:

- IETF RFC 1122 Supported
- Autosar 4.x SOME/IP accelerated
- Scales to 10/25/50/100 Gbps

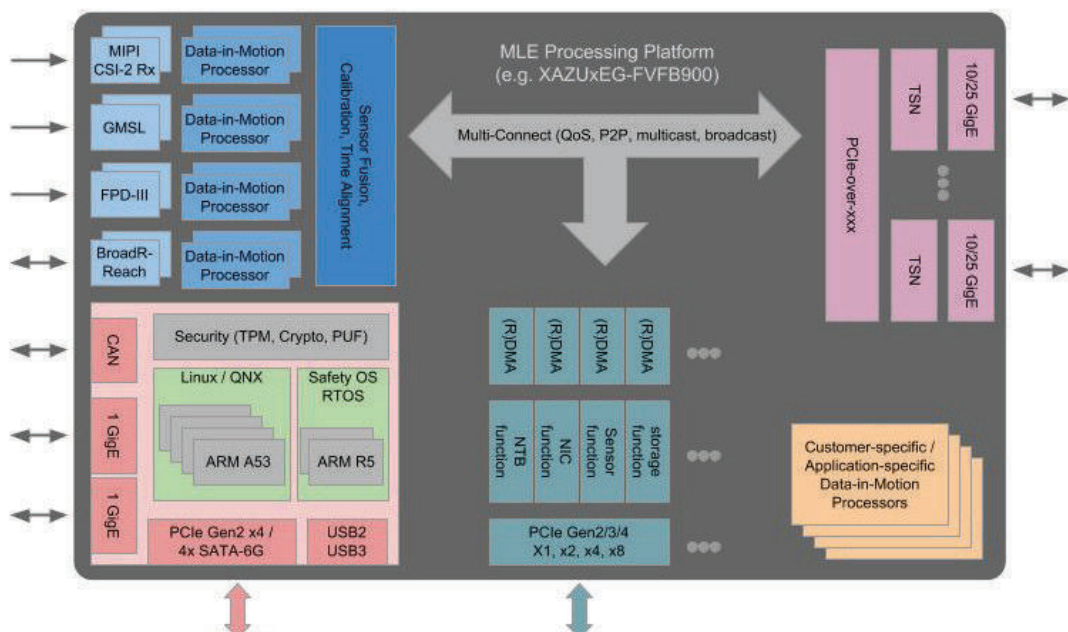
PCIe Features:

- PCI-SIG Base Spec 3.1 or 4.0 using x1, x4, x8
- Scales to 8/16/32/64/128 Gbps



System-level protocol stack example

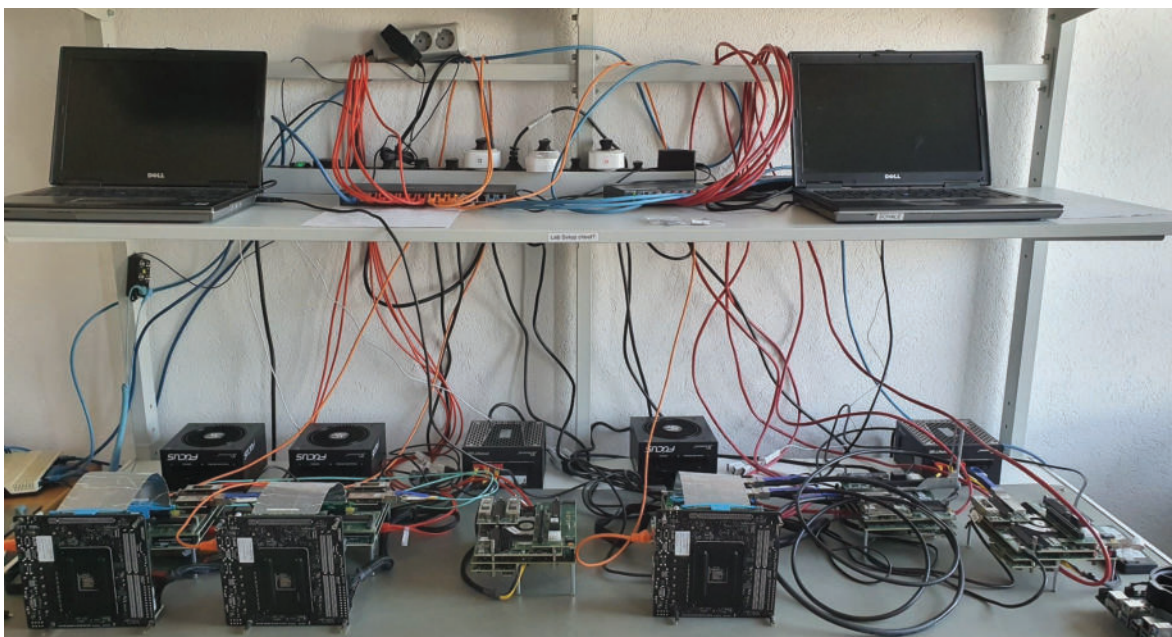
Implementation with Xilinx FPGA



Ongoing Research & Development w/ Partners

- Functional Safety (apply PCIe aspects for High-Availability)
 - Watchdog for PCIe AER (Advanced Error Reporting)
 - PCIe DPC (Downstream Port Containment)
 - IEEE 802.1CB (Frame Replication / Elimination)
- Security
 - PCIe RC/EP Authentication
 - ARM Secure OP-TEE
- Real-Time Behavior
 - IEEE 802.1AS and PCIe PTM

PCIe-over-TCP-over-TSN-over-10GE Lab Car

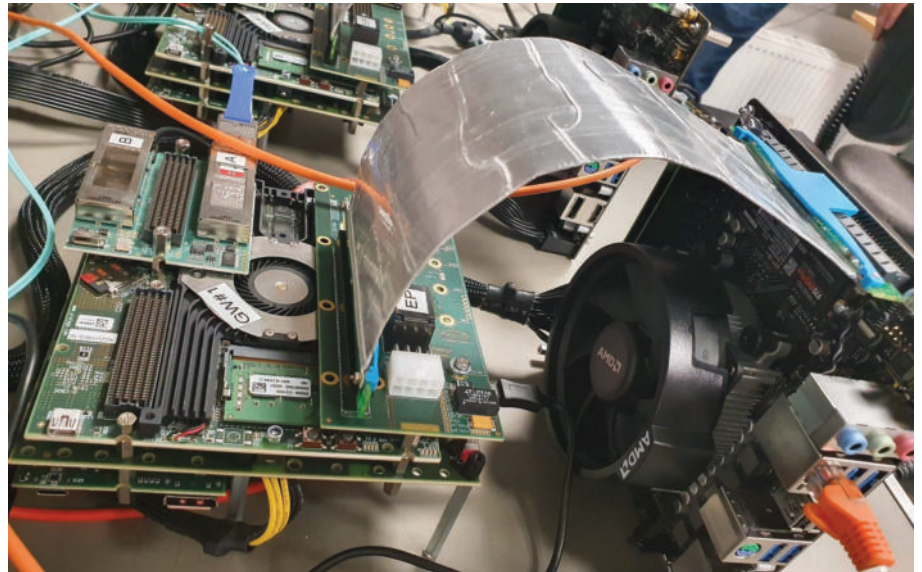


PCIe-over-TCP-over-TSN-over-10GE Lab Car

Uses ASIC Emulators
from ProDESIGN GmbH

“LEGO”-like interface
boards for

- PCIe
- 10/25G Ethernet
- etc



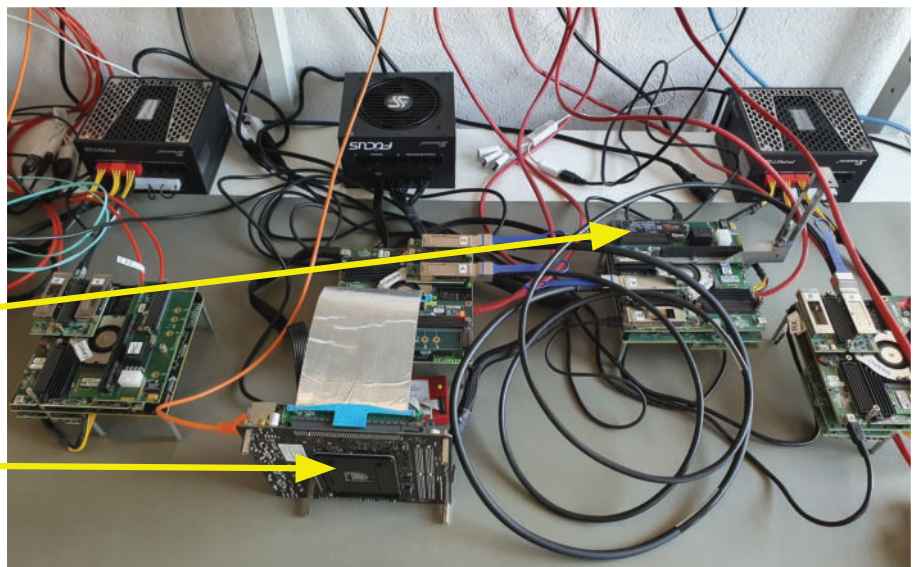
PCIe-over-TCP-over-TSN-over-10GE Lab Car

Backbone with
2 FPGA Gateways

for
PCIe-over-...

to
m.2 NVMe SSD

from
mini-ITX PC



PCIe-over-TCP-over-TSN-over-10GE Lab Car

Backbone with
2 FPGA Gateways

for
PCIe NTB

between
2 mini-ITX PCs



Our Contact Information

Missing Link Electronics, Inc.

+1 (408) 475-1490

2880 Zanker Road, Suite 203

San Jose, CA 95134

United States

Missing Link Electronics GmbH

+49 (731) 141149-0

Industriestrasse 10

89231 Neu-Ulm

Germany

Email contact: sales-web@mlecorp.com

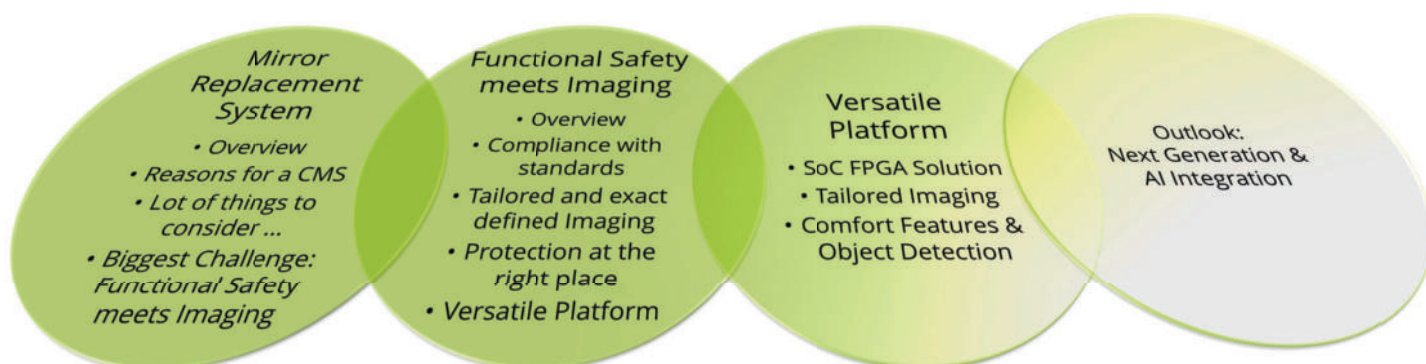


Mirror Replacement System An FPGA4ADAS Story

Dipl. Ing. (FH) Stefan Schütz
Managing Director Solectrix
R&D Director Automotive



Agenda



Mirror Replacement System An FPGA4ADAS Story

Topic: Replacing conventional exterior mirrors with equivalent & suitable **Camera Monitor System (CMS)**
- aerodynamic advantages - allows new design concepts - enabling further ADAS features

• **Challenges:**

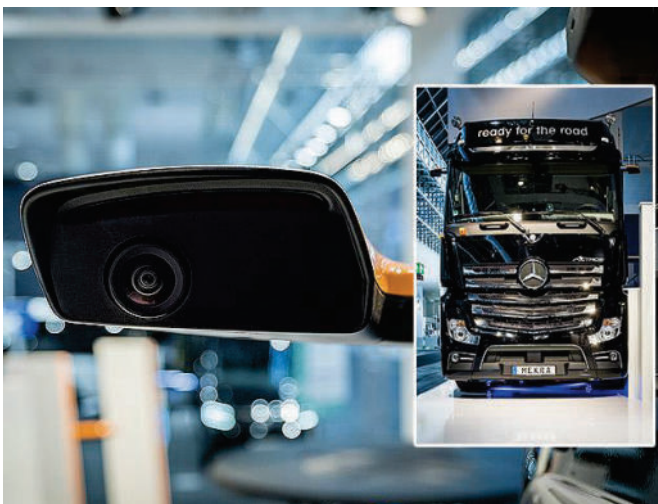
- Exterior mirrors are safety relevant vehicle parts for securing the driver's indirect rear view
- The CMS must meet specific quality criteria to display the rear view sufficiently
- Receive & display more information and integrate certain comfort/ADAS features

• **Solectrix contributions:**

- Research since 2012 regarding CMS Core features, i.e. Image Quality and Field of Views
- Series development for a "truck-CMS" with ASIL B criteria since 2015 with SOP 07/2019
- Research and Development now on 2nd generation of CMS Features

Supplier Awards 2019 - category innovation

Mirror Replacement System on a truck



Reasons for a CMS as Mirror Replacement:

- Better fuel economy due to improved aerodynamics - Reduction of CO2 emissions
- Improvement of aeroacoustics
- Improved vision
 - No glare at direct sunlight or due other high-beam headlights
 - Improved night vision – better than the human eye
- Improvement of field of view, blind spot minimized
- Improvement of direct vision – smaller obstructions
- Integration of smart comfort features and new view options
- Enabling new advanced driver assistance systems

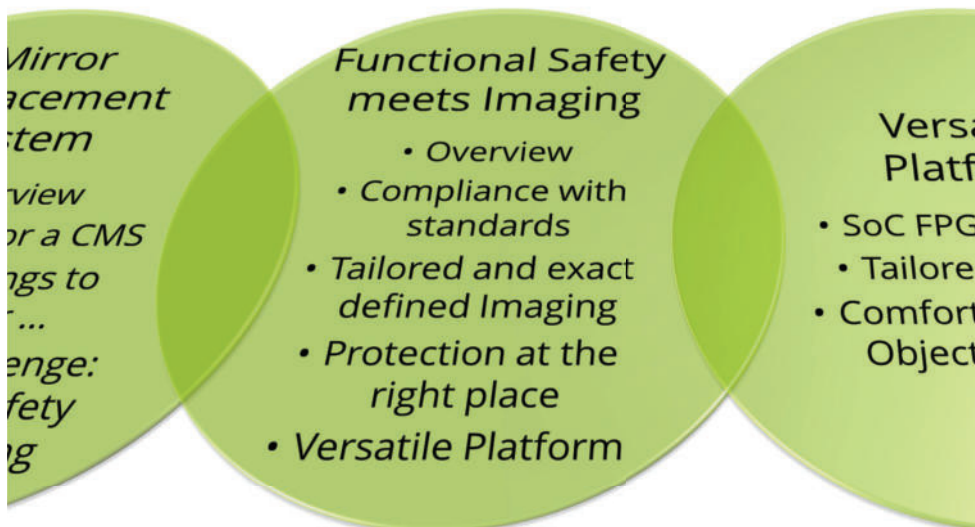
Lot of things to consider ...

- Optical-electrical-optical transfer function has to be optimized for maximum contrast and true color reproduction, especially regarding adequate response to changing lighting conditions
- Perfect resolution has to be shown, which is also relevant for the selection of the sensor and display resolution and size
- Image changes have to be depicted with a minimal time-delay for the whole optical-electrical-optical path
- Different general day and night characteristics & properties has to be considered for the whole imaging path
- Need of Automatic.-Panning, so that the system always shows the entire trailer (because the FOV cannot be changed by moving the head)
- The CMS has to ensure that no image loss and frozen images occurs
- The state of the CMS must be clearly recognizable and secured/ensured (boot up, operation, safe state)
- The whole image reproduction (color and contrast reproduction) has to be ensured to avoid any loss of information's, artifacts, etc. and controlled adaptations to changes in ambient conditions
- Compliance with standards and laws for indirect vision and CMS and end customer acceptance
- High level of integration in vehicle infrastructure and architecture in general

Biggest Challenge: Functional Safety (ISO26262) meets Imaging

- **HQ Image** reproduction - HDR, Anti-Flicker, 3D noise filter, tone mapping, day & night characteristics
- **Variable FOV** (Field of View(s) / Visions) with different view generation
- **Low latency with defined step** function **response** regarding AEG & AWB (e.g. tunnel scenario)
- **Trailer Tracking** for automatic panning of the shown view (to show the entire trailer)
- **Digital Assistance** e.g. with specific overlays to support docking maneuvers or lane changes

Secured and Safeguarded -
To avoid frozen, delayed or
artifact disturbed images



Detection Rules at the right places for ASIL A / B / (C)

Faults: Delayed – Frozen - Disturbed

- Frozen image stream - stream of frames with non-increasing framenumbers
- Delayed image stream - consistent but delayed stream of frames with increasing framenumbers
- Disturbed image stream – loss of information within the content, e.g. due to faults when multiple images are combined together like stitching or HDR processing

Measures for protection and safeguarding at the right places

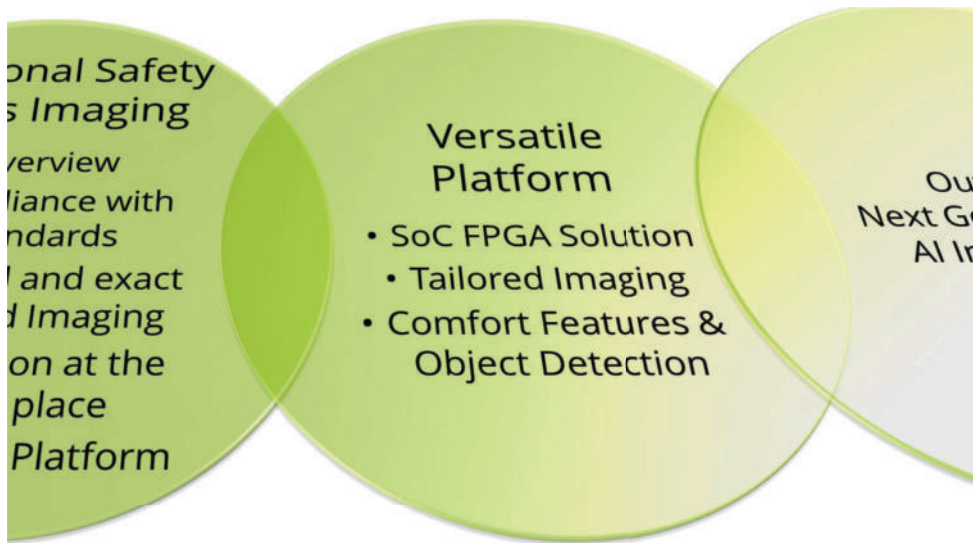
- Control & Monitor Timing & frame rates - ensure buffer management
- Control & Monitor Frame Numbers
- Control & Monitor Sensor, HDR Processing and image characteristics in general

Versatile Platform

- Compliance with standards
- Tailored and exact defined Imaging
- Protection at the right place

+ Scalable and versatile solution for integration of smart comfort features & enabling of new advanced driver assistance systems

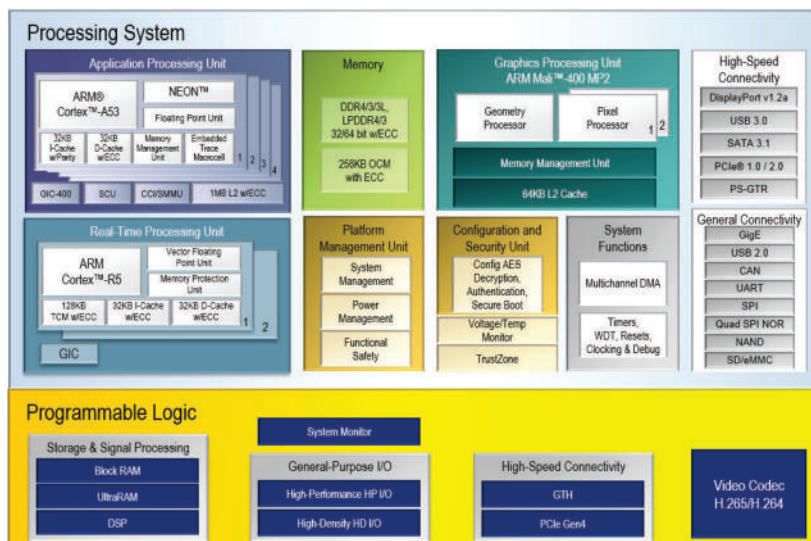
→ Scalable and/or versatile SoCs needed like TI, NVIDIA or FPGA based SOCs



Versatile Platform – FPGA based SoC Solution – an example

- Sensor Configuration
- Image Processing Control Loops
- Overlay Generation
- SW Isolation
- Video Streaming
- Object Detection

- Overlay Generation
- (270° Birdview)



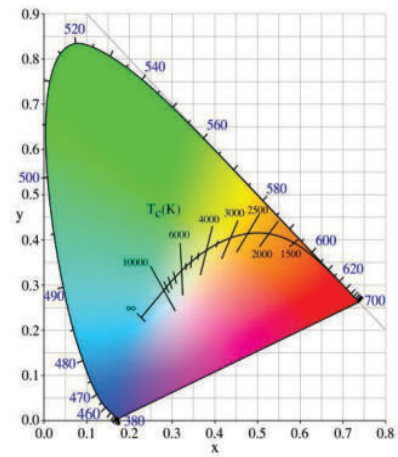
- CAN Communication
- SW Download
- Parameter Memory
- Diagnosis

- Image Processing Pipelines
- FoV Generation
- Object Detection

- Board Monitoring
- Error Handling
- SW Test Library
- Functional Safety - ASIL-C
- H.264/H.265 Video Encoding for Video Streaming / Storage

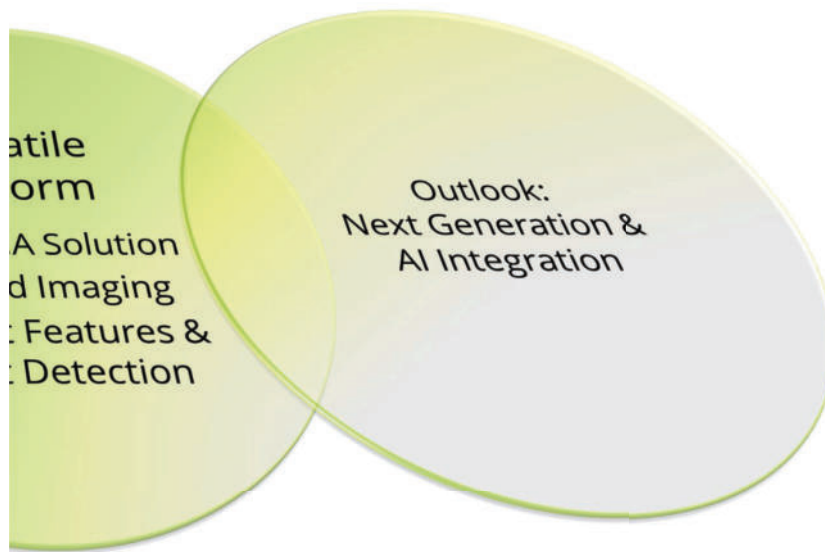
Tailored Imaging:

- Adaptive ToneMapping
- Anti-flicker mechanisms for mitigation disturbing image flicker
- CCT (correlated color temperature) guided AutoWhiteBalance
- Adaptive Luminance & Saturation Control
- Adaptive and optimized noise filters
- Day / night mode
- Special Views
- ...



Intelligent comfort features and object detection

- Trailer Tracking & Automatic Panning
 - Lane Detection
 - Support for VRU detection
- + more complex object detection based on CNN approaches as an option



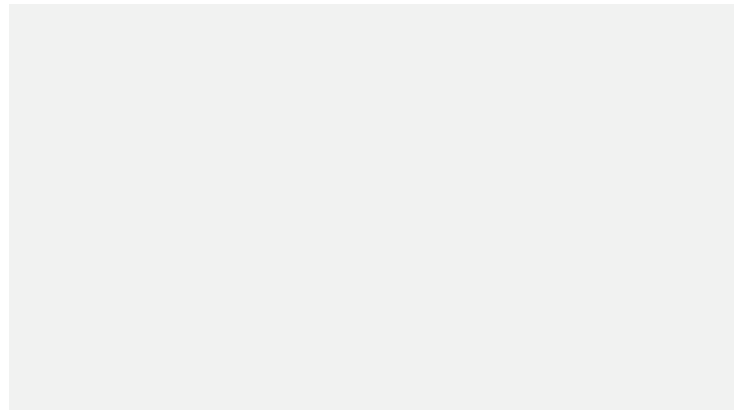
FPGAs ... from an application point of view

- Powerful, fast, and flexible
- Wide range of applications
 - Functional safety
 - Image signal processing
 - ...
- Restricted in resources

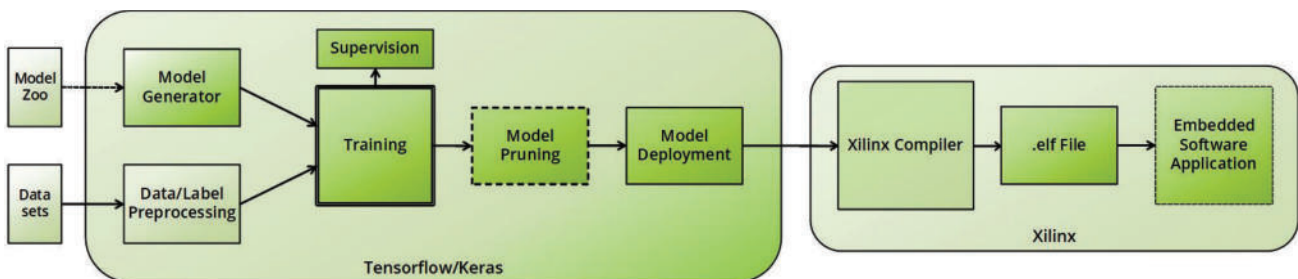


AI / machine learning ... from an algorithmic point of view

- Powerful and flexible
- Wide range of applications (in image processing)
 - Object detection
 - Classification & Segmentation
 - ...
- Large amount of resources required

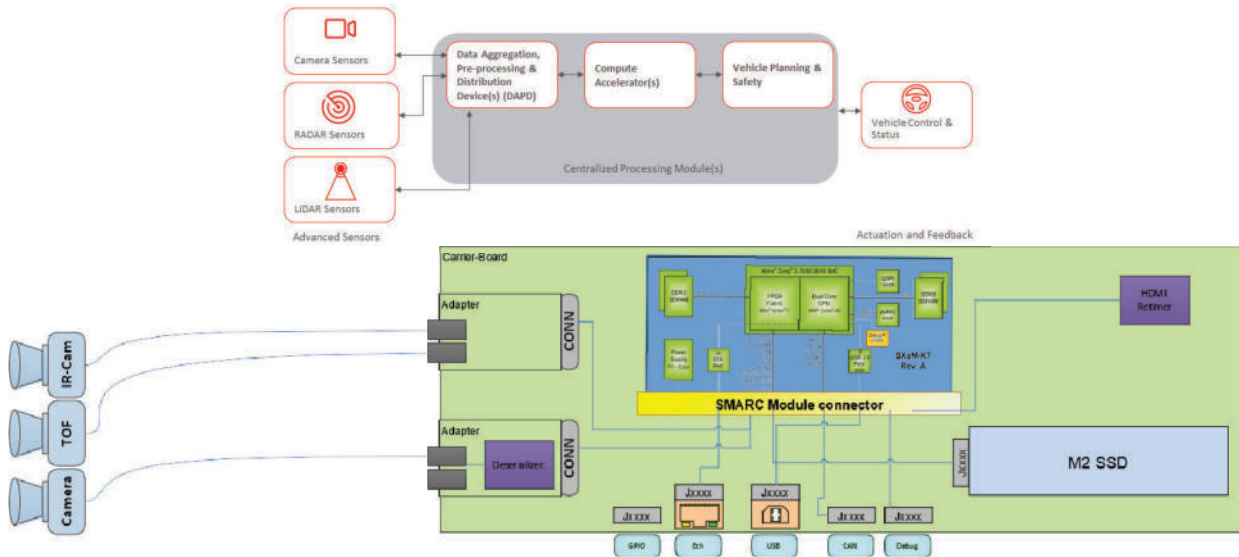


- Bringing both worlds together with a specialized AI-Ecosystem



- Focus on Network selection & Pruning methods
- With the goal either to optimize a network for implementing AI in an embedded device or to optimize a network for pure performance

Platform for data aggregation and AI integration - sysiko



solectrix GmbH
Dieter-Streng-Str. 4
90766 Fürth
Germany

Managing Directors:
Dipl.-Ing. (FH) Lars Helbig
Dipl.-Ing. (FH) Stefan Schütz
Dipl.-Ing. (FH) Jürgen Steinert

Fon: +49 (0) 911 - 30 91 61 - 0
Fax: +49 (0) 911 - 30 91 61 - 299

info@solectrix.de

www.solectrix.de



Automotive System Architectures – from ADAS to AD

Advanced Driver Assistance Systems to Autonomous Drive

Ralf Neuhaus : Automotive System Architect EMEA

September 24, 2020



Agenda

- › SAE Driving Levels: Evolution vs Revolution
- › Automated Driving System Requirements
- › Scaling the Architecture L0 to L4

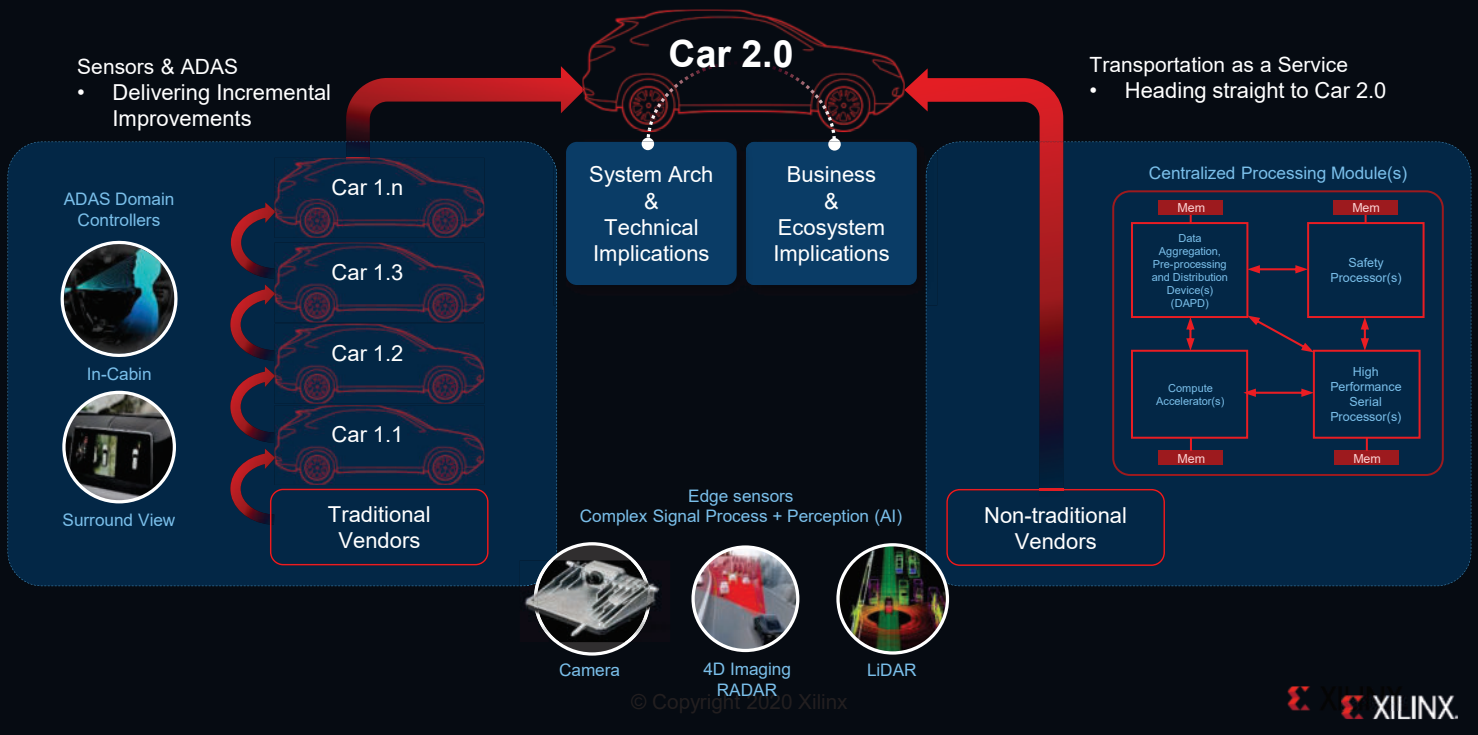
SAE Driving Levels: Evolution vs. Revolution

SAE Driving Levels

Increased System Level Responsibility = Increase in System Complexity

SAE Level	Name	Narrative Definition	Execution of Steering and Acceleration/Deceleration	Monitoring of Driving Environment	Fallback Performance of Dynamic Driving Task	System Capability (Driving Modes)	Examples
Human Driver monitors the driving environment							
0	No Automation	the full-time performance by the <i>human driver</i> of all aspects of the <i>dynamic driving task</i> , even when enhanced by warning or intervention systems	Human Driver	Human Driver	Human Driver	n/a	Blindspot Detection / Surround View
1	Driver Assistance	the <i>driving mode</i> -specific execution by a driver assistance system of either <u>steering</u> or <u>acceleration/deceleration</u> using information about the driving environment and with the expectation that the <i>human driver</i> perform all remaining aspects of the <i>dynamic driving task</i>	Human Driver and system	Human Driver	Human Driver	Some driving modes	Adaptive Cruise Control / Lane Keep Assist / Parking Assist
2	Partial Automation	the <i>driving mode</i> -specific execution by one or more driver assistance systems of <u>both steering and acceleration/deceleration</u> using information about the driving environment and with the expectation that the <i>human driver</i> perform all remaining aspects of the <i>dynamic driving task</i>	System	Human Driver	Human Driver	Some driving modes	Traffic Jam Assist
Automated Driving System ("system") monitors driving environment							
3	Conditional Automation	the <i>driving mode</i> -specific performance by an <i>automated driving system</i> of all aspects of the <i>dynamic driving task</i> with the expectation that the <i>human driver</i> will respond appropriately to a <i>request to intervene</i>	System	System	Human Driver	Some driving modes	Full Speed Range Stop & Go - Highway / Self Parking
4	High Automation	the <i>driving mode</i> -specific performance by an <i>automated driving system</i> of all aspects of the <i>dynamic driving task</i> , even if a <i>human driver</i> does not respond appropriately to a <i>request to intervene</i>	System	System	System	Some driving modes	Automated Driving / Valet Parking
5	Full Automation	the full-time performance by an <i>automated driving system</i> of all aspects of the <i>dynamic driving task</i> <u>under all roadway and environmental conditions</u> that can be managed by a <i>human driver</i>	System	System	System	All Driving Modes	Full Autonomous Driving / Driver-less Vehicle Operation

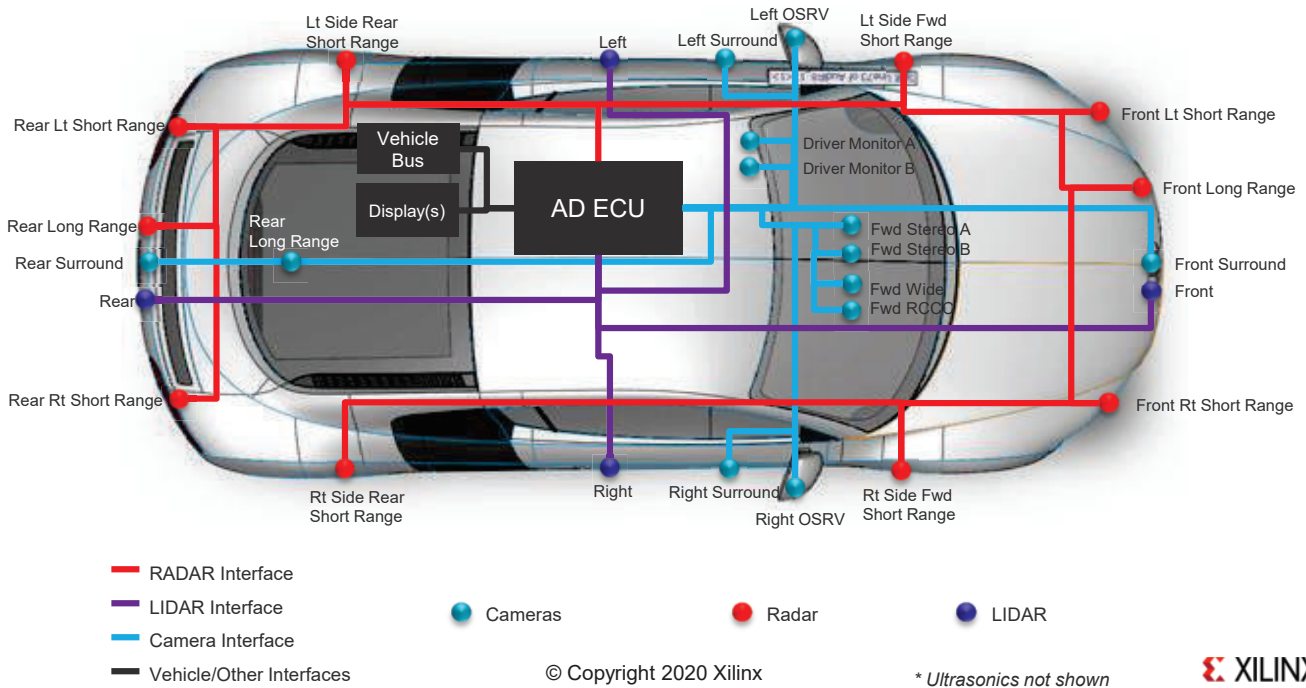
Towards Level 4: PoV Evolution vs. Robotaxi Revolution



Automated Driving System Requirements

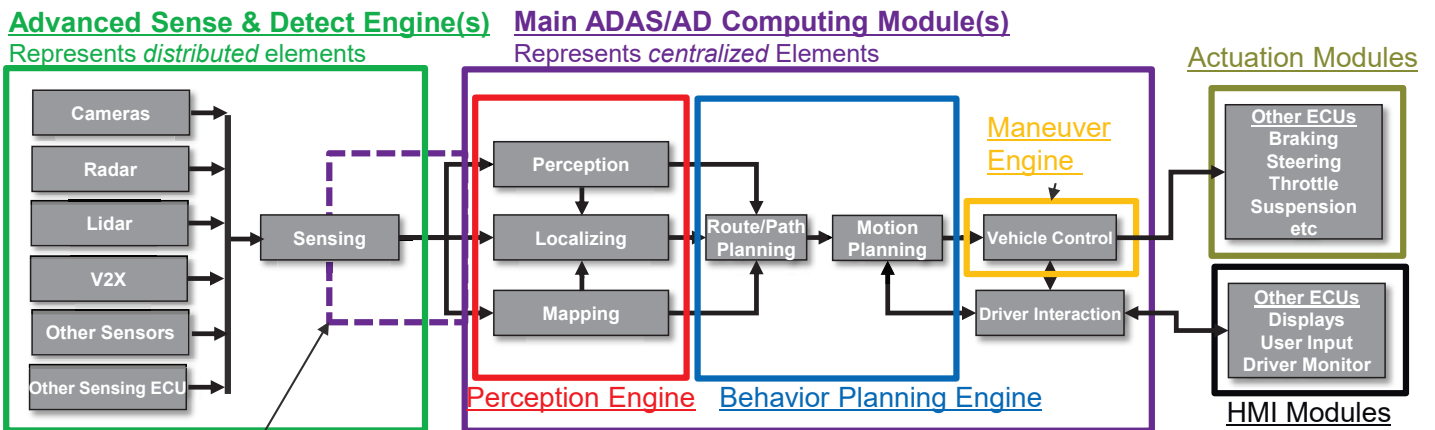
Automated Driving Case: Sensing Suite

Wide variety of distributed sensor configurations from L0 to L4



Automated Driving System Functional Diagram

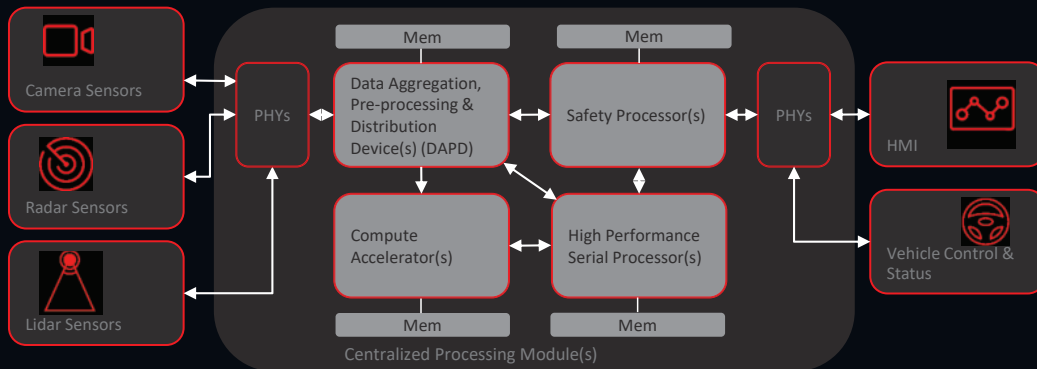
An assortment of different processing functions performed by various “engines” located in distributed modules



Some Sense & Detect Processing may be done in Main ADAS/AD Compute Module

AD Central Module Processing Element Architecture

4 primary processing element types



- › Although sometimes integrated into a single device/package, a centralized AD processing module is commonly comprised of a *heterogeneous set* of processing element types:
 - › Data Aggregator, Pre-Processor and Distributor
 - › High Performance Compute Processor(s)
 - › Computational Accelerators
 - › Safety Processor(s)

© Copyright 2020 Xilinx



Desired AD Platform Architecture Characteristics

What do I want in my platform to address L0 to L4?



Scalability

- › Adjustment of BoM costs across low to high complexity systems



Portability

- › Migration of designs between device family generations over time



Adaptability (Flexibility)

- › Efficient adaptation of algorithms and interfaces across multiple product life cycles
 - New sensing technologies drives new algorithmic and interface approaches (e.g. 1 to 8 Mpix)
 - High volume "field lessons" require changes in deployed algorithms (e.g. new edge cases)
 - Drive for efficiency requires adaptable processing engines (e.g. 32b FP to 8b INT to binary CNN)



Modularity

- › Ability to adjust individual elements of processing performance (e.g. DMIPs vs. TOPs)
- › Partitioning and functional safety advantages

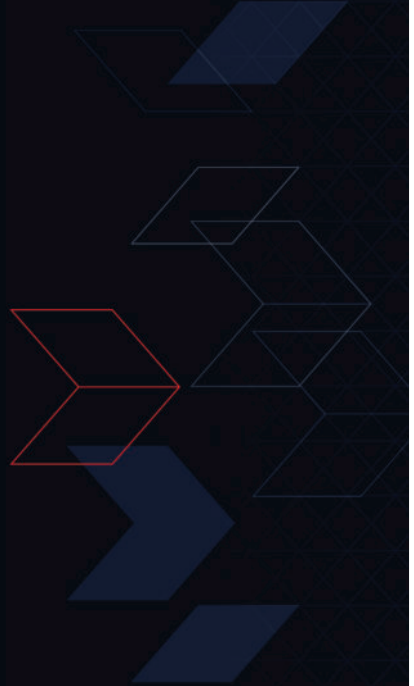


Scaling the Architecture L0 to L4

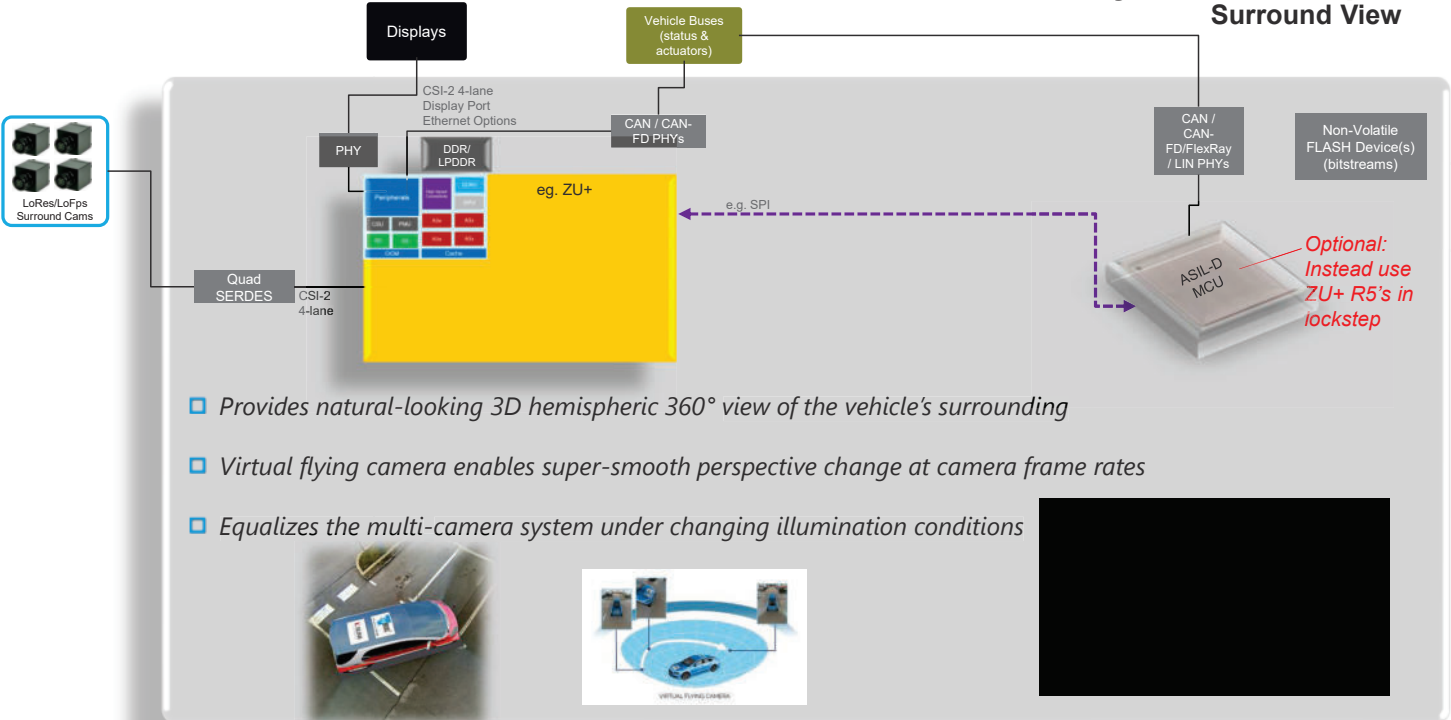


Level 0 Example

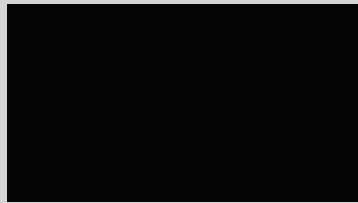
Surround View



Level 0 System Example Surround View



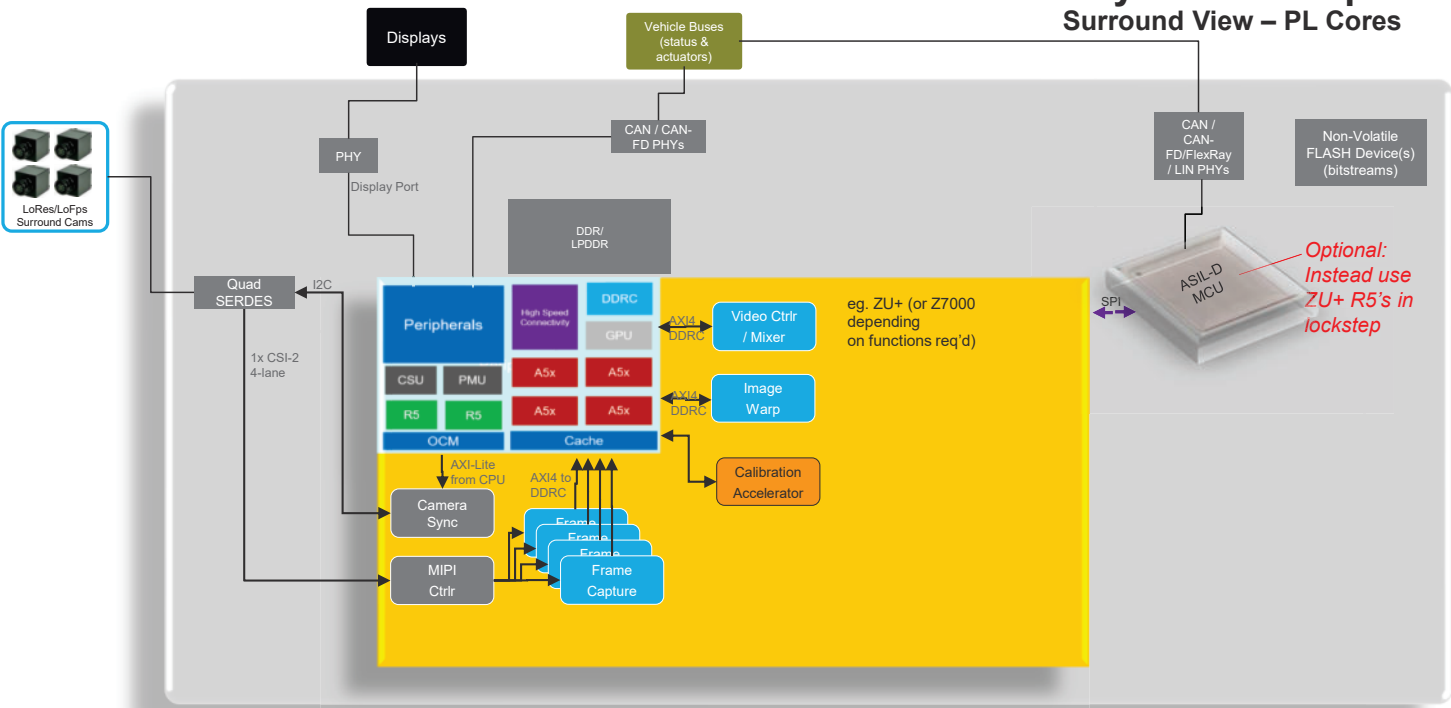
- Provides natural-looking 3D hemispheric 360° view of the vehicle's surrounding
- Virtual flying camera enables super-smooth perspective change at camera frame rates
- Equalizes the multi-camera system under changing illumination conditions



© Copyright 2020 Xilinx



Level 0 System Example Surround View – PL Cores



© Copyright 2020 Xilinx



Level 1 Example

Surround View, NCAP Fwd Cam, ACC



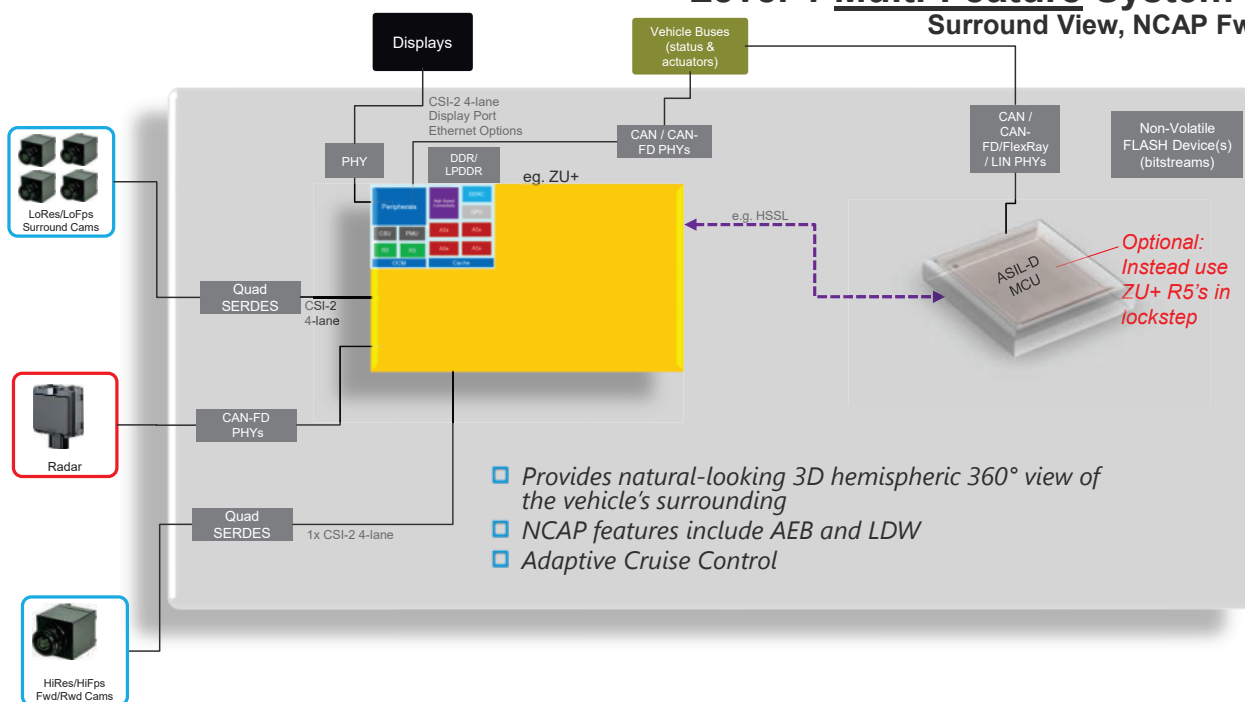
>> 15

© Copyright 2020 Xilinx



Level 1 Multi-Feature System Example

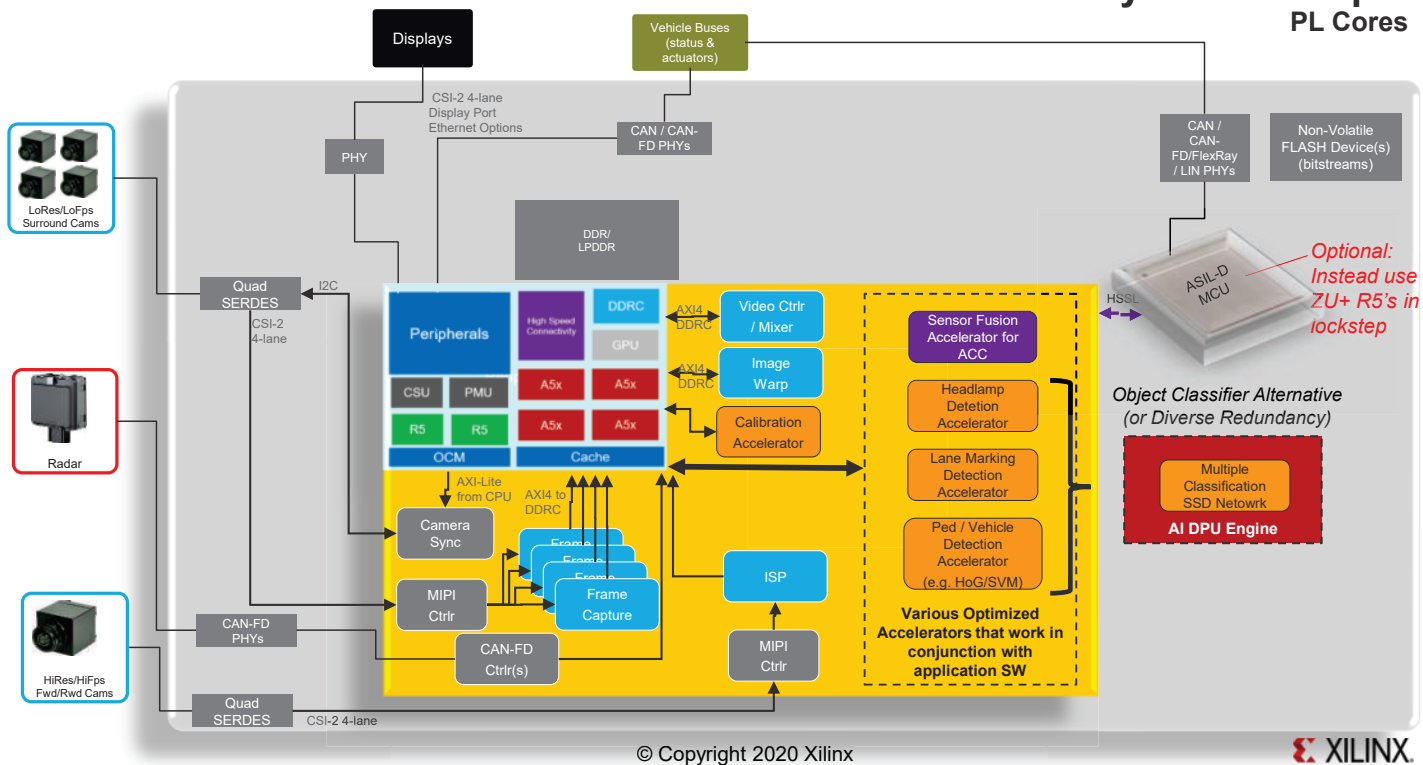
Surround View, NCAP Fwd Cam, ACC



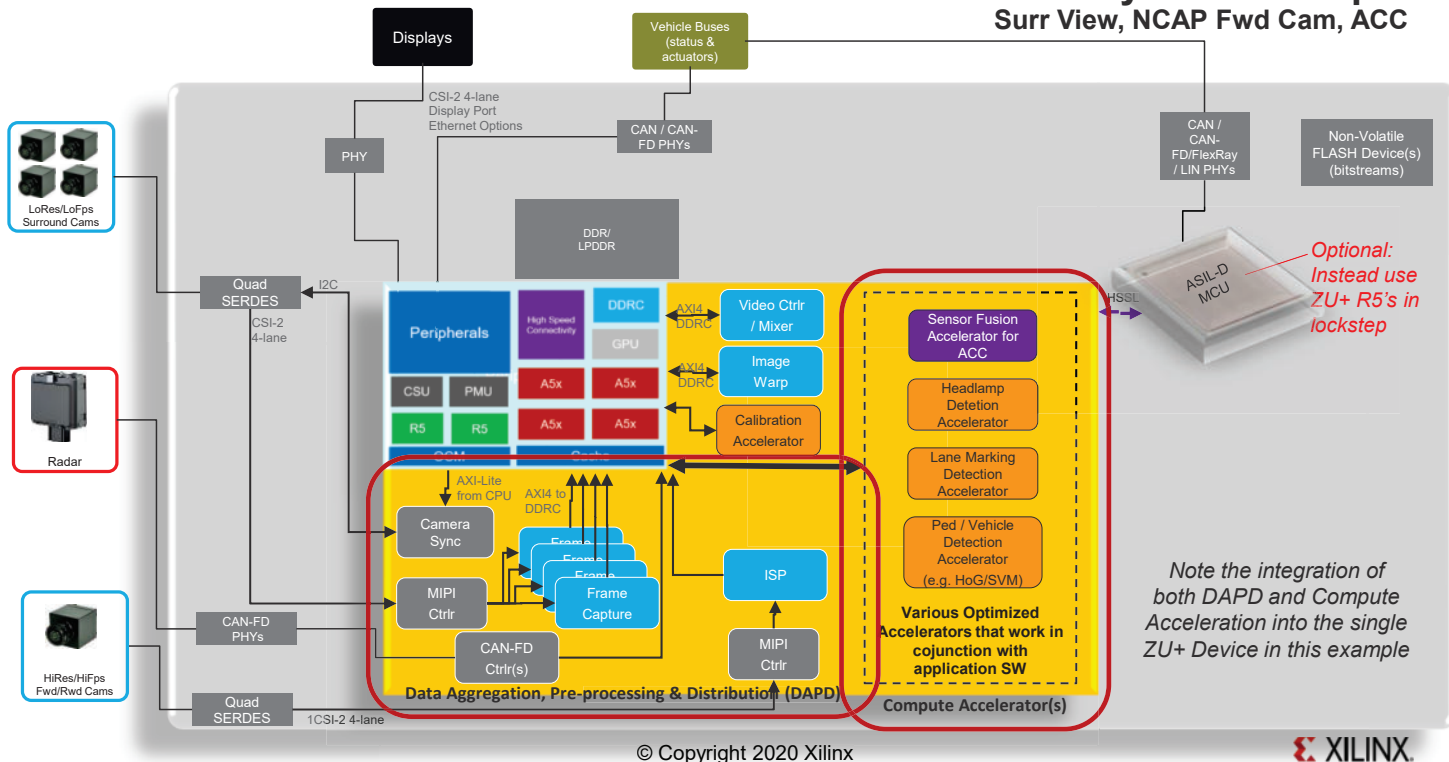
© Copyright 2020 Xilinx



Level 1 Multi-Feature System Example PL Cores



Level 1 Multi-Feature System Example Surr View, NCAP Fwd Cam, ACC



Level 3/3+ Example

NCAP Fwd Cam, DMS, Highway Pilot, APA

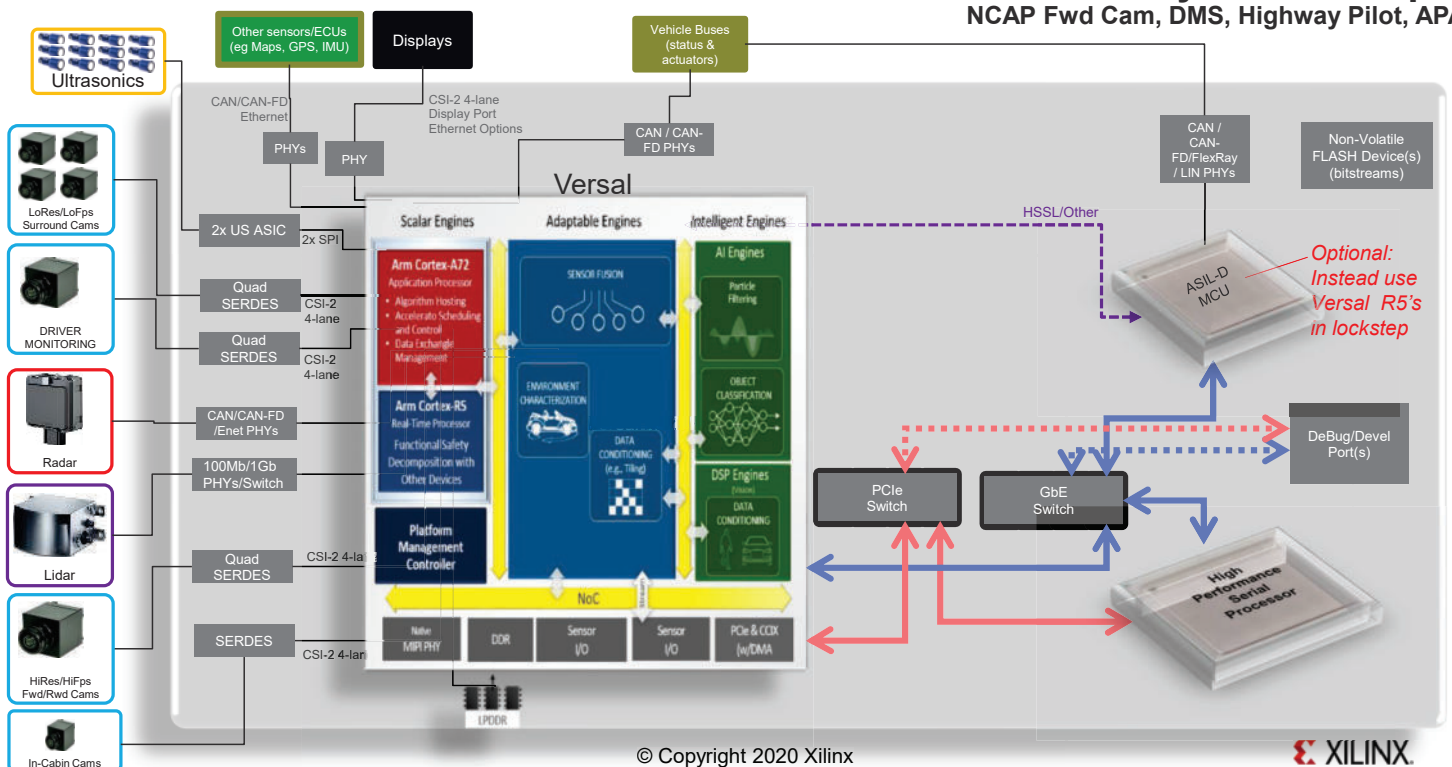


>> 19

© Copyright 2020 Xilinx

Level 3+ System Example

NCAP Fwd Cam, DMS, Highway Pilot, APA



© Copyright 2020 Xilinx



Level 4 AD ECU Summary

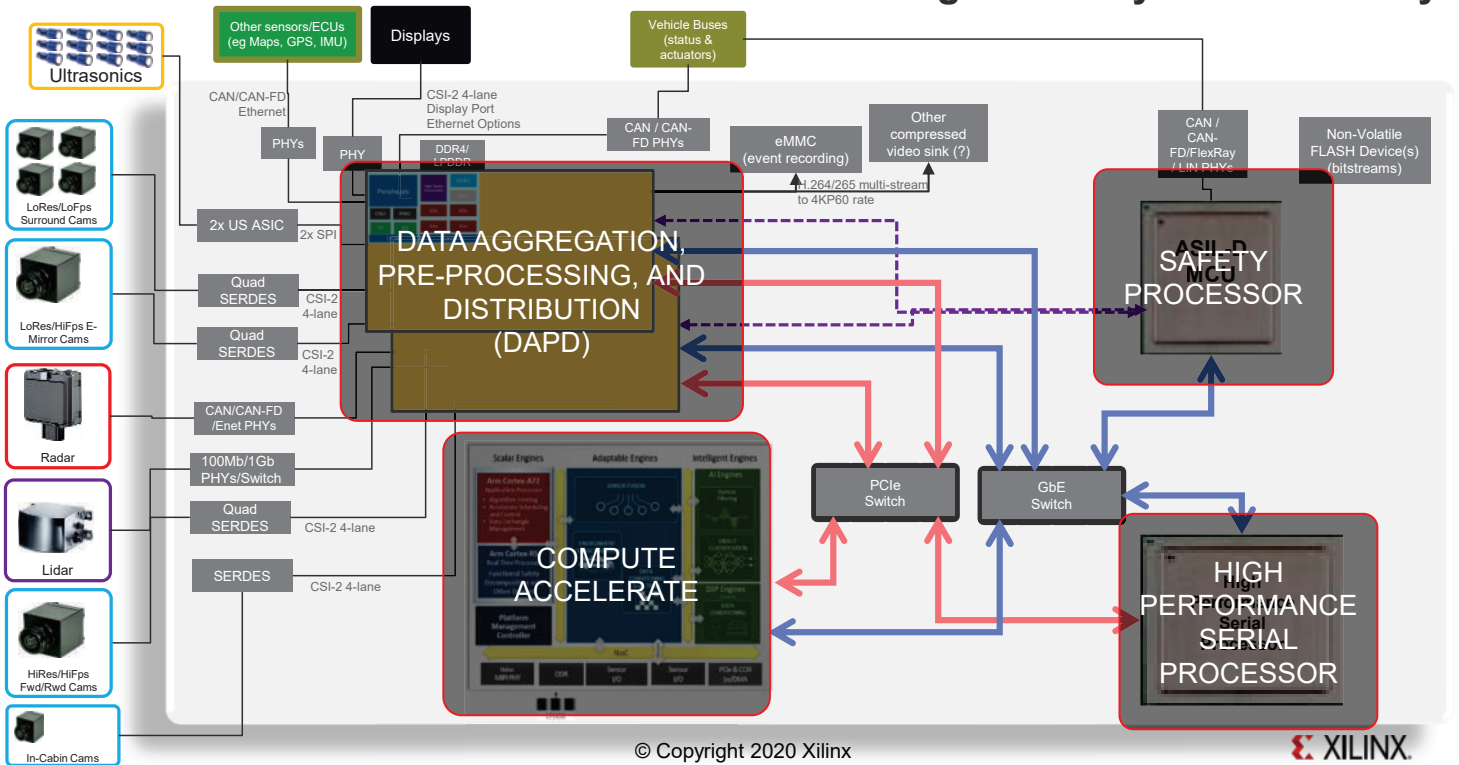


>> 21

© Copyright 2020 Xilinx



A Modular/Scalable AD ECU Architecture offering Flexibility and Portability



© Copyright 2020 Xilinx



Workshop :
Programmable Processing for the
Autonomous / Connected Vehicle



Thank You

Xilinx in AI: Versal AI-core, AI-Engine Architecture, Design Flow

Daniele Bagni
daniele.bagni@xilinx.com
DSP / ML Specialist for EMEA

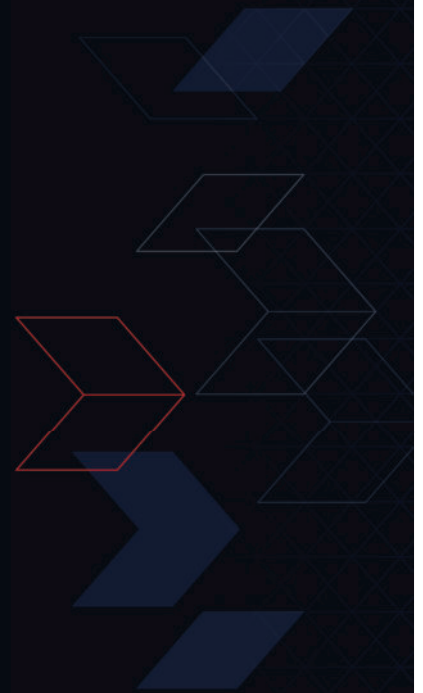
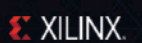
FPGA4ADAS Workshop at Ulm University,
24 September 2020



Agenda

- > Introduction
- > The AI Engine
- > ML inference with Vitis AI

>> 2

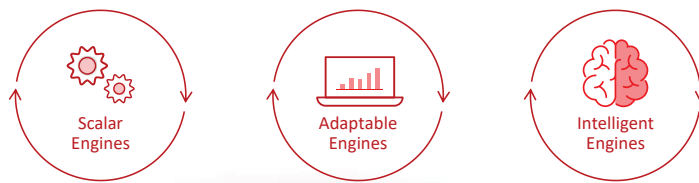


Introduction



New Device Category: Adaptive Compute Acceleration Platform

COMPUTE ACCELERATION



ADAPTIVE

Diverse Workloads in Milliseconds

Future-Proof for New Algorithms

PLATFORM

Development Tools
HW/SW Libraries
Run-time Stack

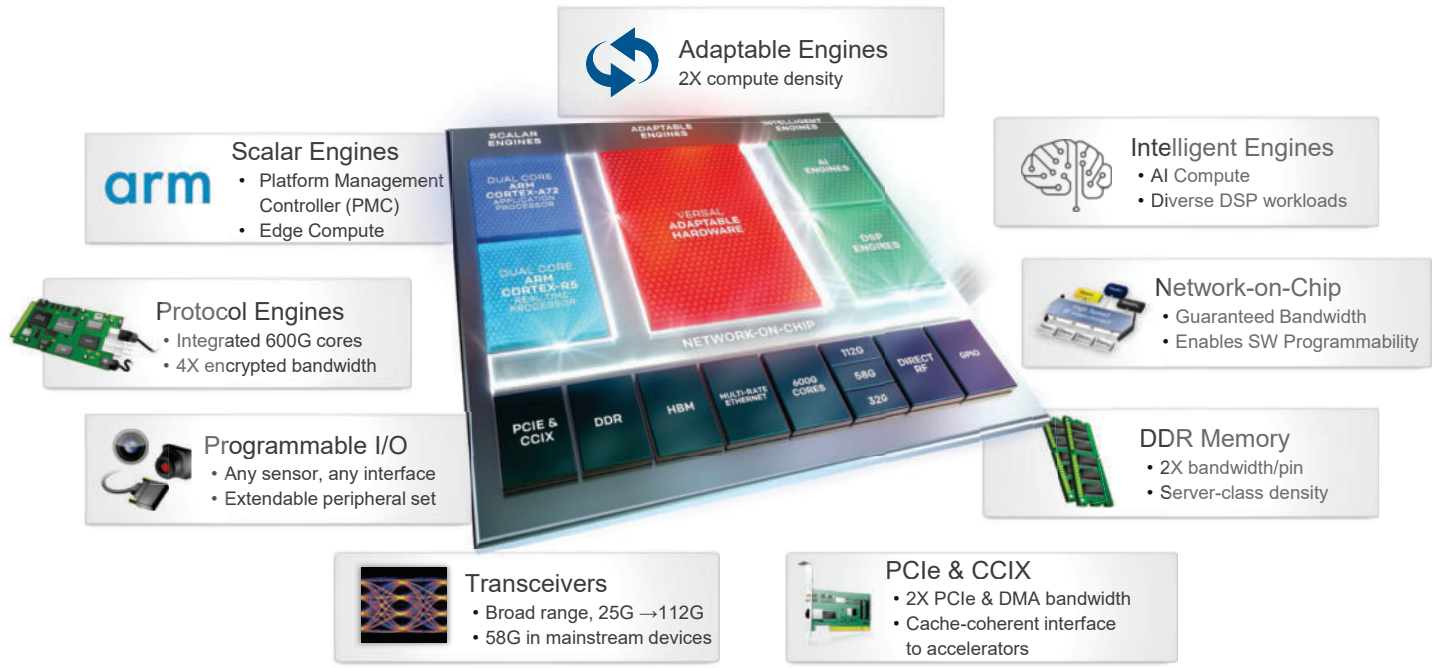
SW Programmable
Silicon Infrastructure

Enabling Data Scientists, SW Developers, HW Developers

© Copyright 2020 Xilinx



Versal Architecture Overview



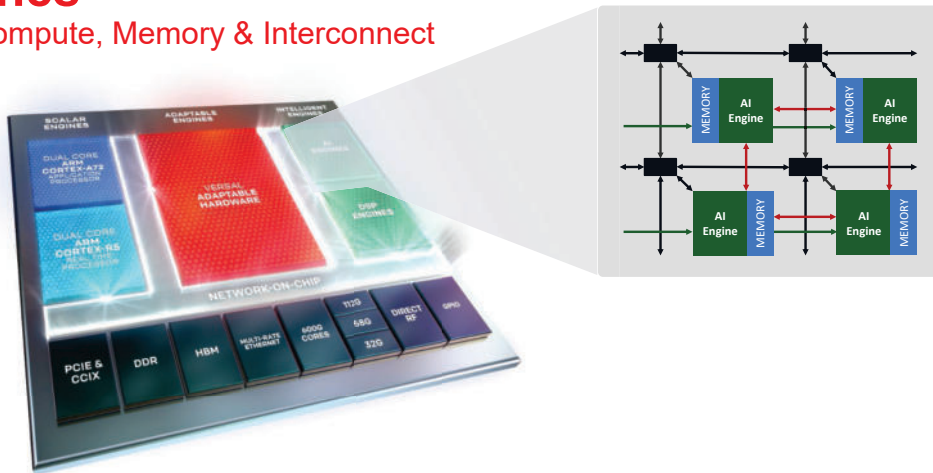
>> 5

© Copyright 2020 Xilinx



AI Engines

Hardened Compute, Memory & Interconnect



Huge performance improvements versus UltraScale+

- > 8x compute density @ 40% lower power

1GHz+ VLIW / SIMD vector processors

- > Versatile core for ML and other advanced DSP workloads

Massive array of interconnected cores

- > Instantiate multiple tiles (10s to 100s) for scalable compute

>> 6

Terabytes/sec of interface bandwidth to other engines

- > Direct, massive throughput to adaptable HW engines
- > Implement core application with AI for "Whole App Acceleration"

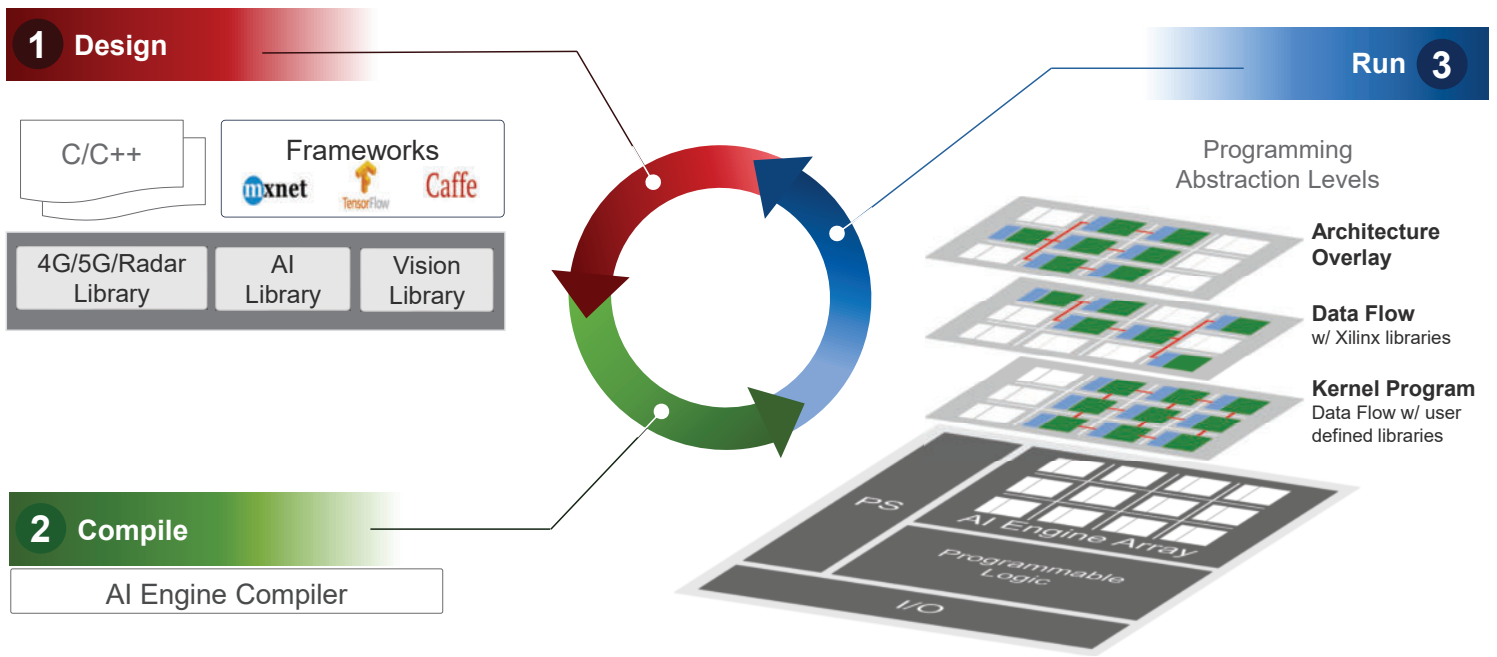
SW programmable for any developer

- > C programmable, compile in minutes
- > Library-based design for ML framework developers

© Copyright 2020 Xilinx



Software Programmable: Any Developer

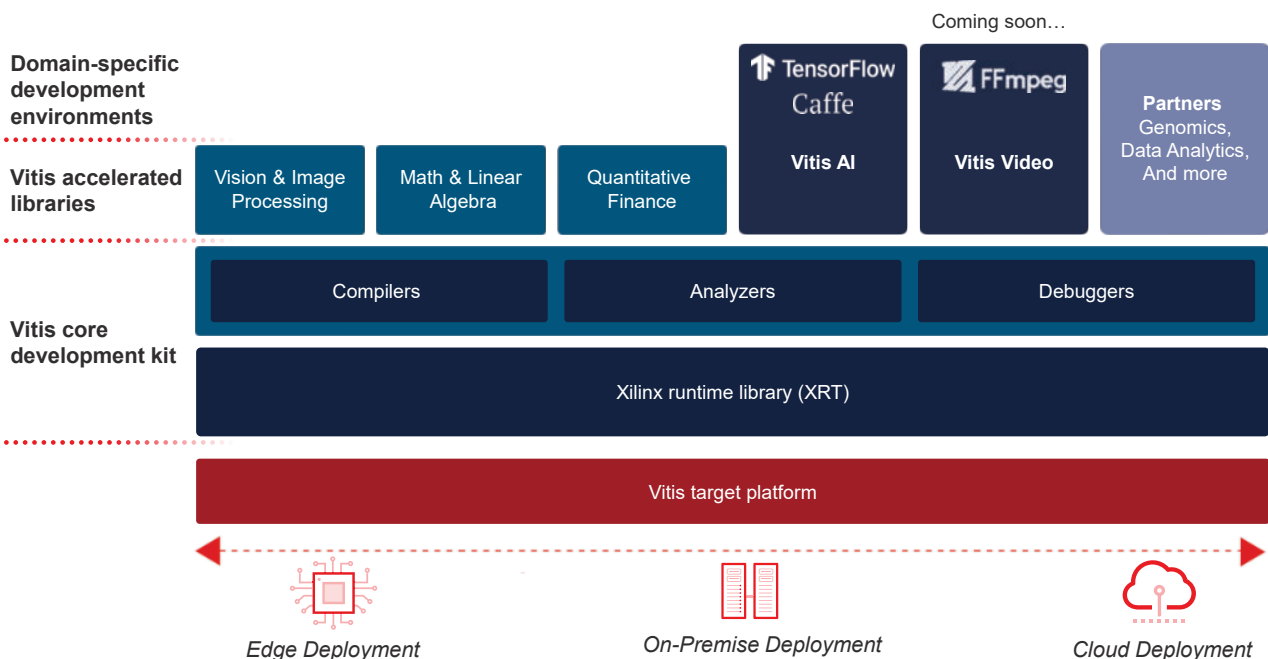


>> 7

© Copyright 2020 Xilinx

XILINX

Vitis Unified Software Platform Design Flow



© Copyright 2020 Xilinx

XILINX

Vitis Target Platform

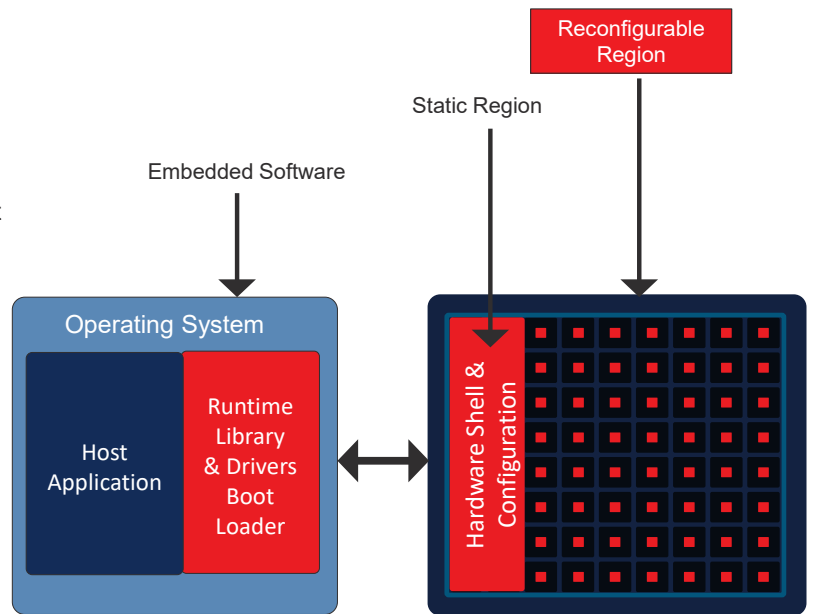
> Pre-Configured Static Region

- >> PCIe® Interface Logic
- >> DDR memory interface controllers
- >> XDMA logic etc.
- >> Hardware Config & Lifecycle Management

> For Embedded Devices, Includes

- >> Operating System
- >> Runtime Library (XRT)
- >> Runtime Drivers (XRT)
- >> Firmware & Boot loader

Use **Ready-to-Use** Vitis Target Platforms
OR
Build Your Own using Vivado Design Suite



>> 9

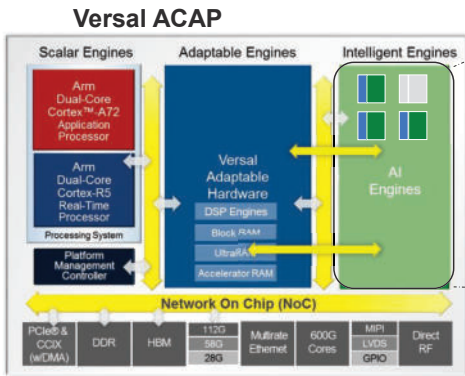
© Copyright 2020 Xilinx

XILINX

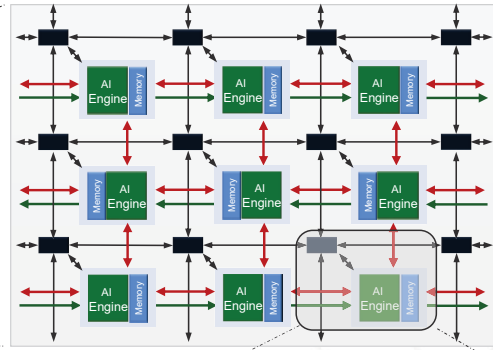
The AI Engine

XILINX

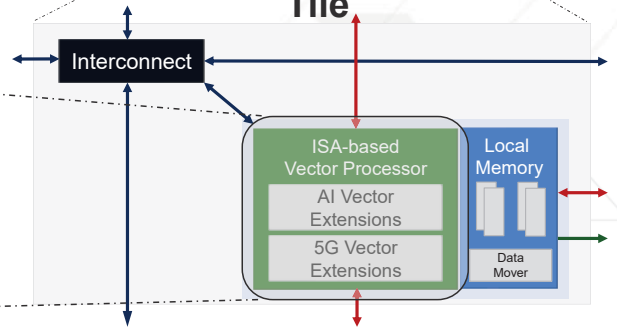
AI Engine: Terminology



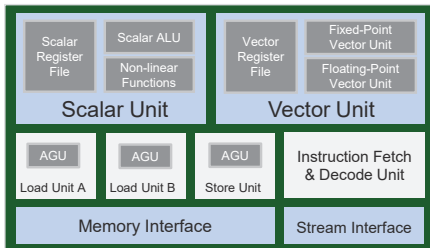
AI Engine Array



AI Engine Tile



AI Engine Core



>> 11

© Copyright 2020 Xilinx

XILINX

AI Engine Tile

> AI Engine core

- >> 512b SIMD vector units
 - Both fixed and floating point
 - 16KB program memory
- >> 32b scalar RISC processor
- >> 256-bit load (x2) and store units with individual AGUs

> 128KB direct core memory access

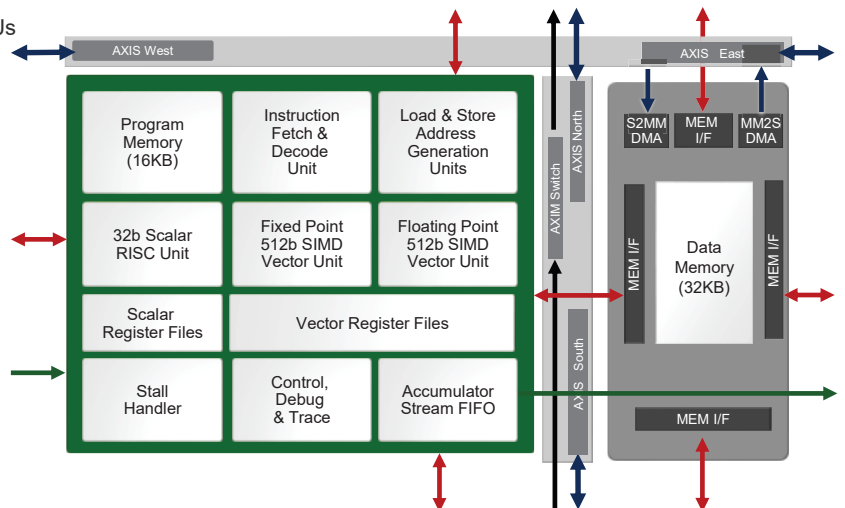
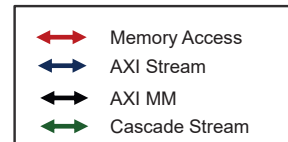
- >> 32KB local
- >> 32KB north, south, east or west

> Interconnects

- >> AXI Memory Mapped (AXI-MM) switch
 - Configuration, control and debug
- >> AXI-Stream crossbar switch
 - Routing N/S/E & west around the array

> Debug/Trace/Profile functionality

- >> Debug using memory-mapped AXI4 i/f
- >> Connect to PMC via JTAG or HSDP



>> 12

© Copyright 2020 Xilinx

XILINX

Multi-Precision Support

Operand 1	Operand 2	Output	Number of GMACs @ 1 GHz
8 real	8 real	48 real	128
16 real	8 real	48 real	64
16 real	16 real	48 real	32
16 real	16 complex	48 complex	16
16 complex	16 real	48 complex	16
16 complex	16 complex	48 complex	8
16 real	32 real	48/80 real	16
16 real	32 complex	48/80 complex	8
16 complex	32 real	48/80 complex	8
16 complex	32 complex	48/80 complex	4
32 real	16 real	48/80 real	16
32 real	16 complex	48/80 complex	8
32 complex	16 real	48/80 complex	8
32 complex	16 complex	48/80 complex	4
32 real	32 real	80 real	8
32 real	32 complex	80 complex	4
32 complex	32 real	80 complex	4
32 complex	32 complex	80 complex	2
32 SPFP	32 SPFP	32 SPFP	8

cfloat is a vector type but is not directly supported by the AI Engine vector processor.
2 instructions have to be issued to perform a mult.

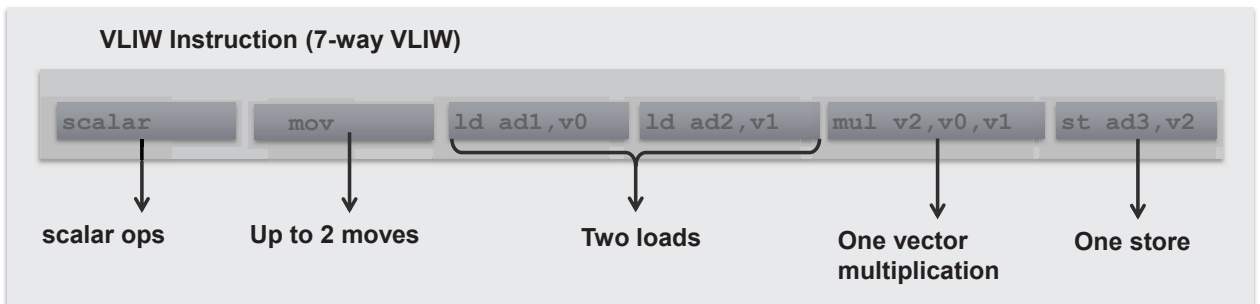
>> 13 SPFP: IEEE Single Precision Floating Point

© Copyright 2020 Xilinx



Multiple-levels of Parallelism

- > 7-way VLIW machine
- > Instruction-level Parallelism (ILP)
 - >> Very long instruction word (VLIW) → 128-bit
 - >> Multiple operations issued in one cycle
- > Data-level Parallelism (DLP)
 - >> Vector data path (SIMD)



>> 14

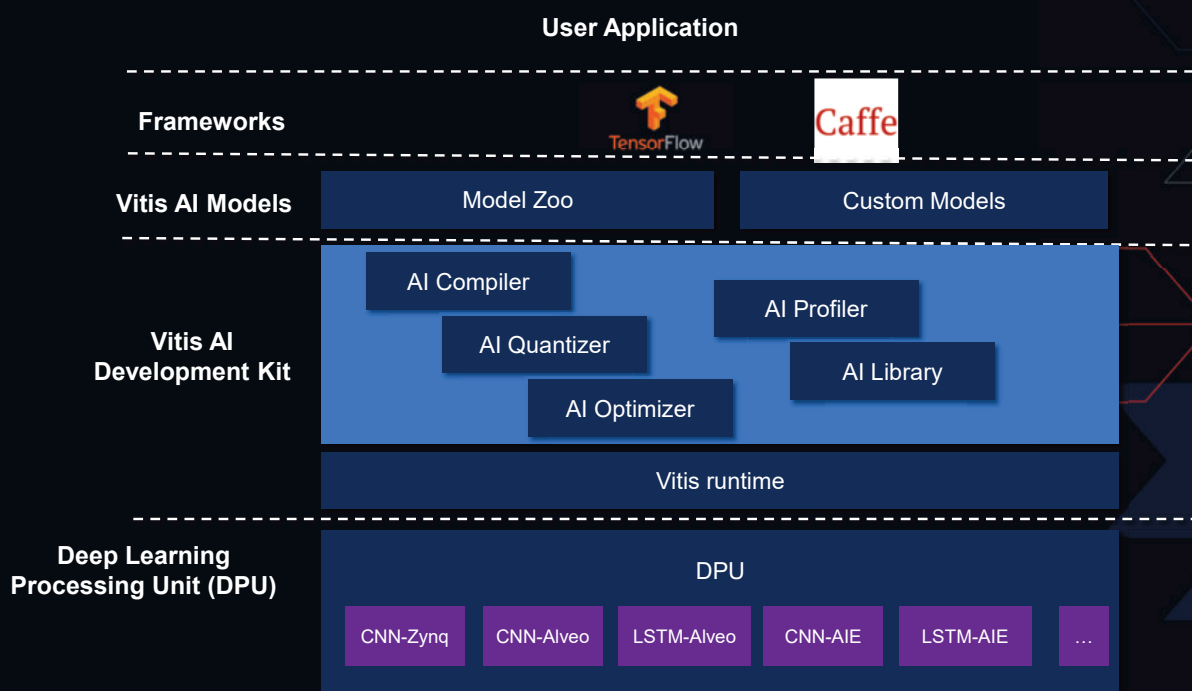
© Copyright 2020 Xilinx



ML inference with Vitis AI

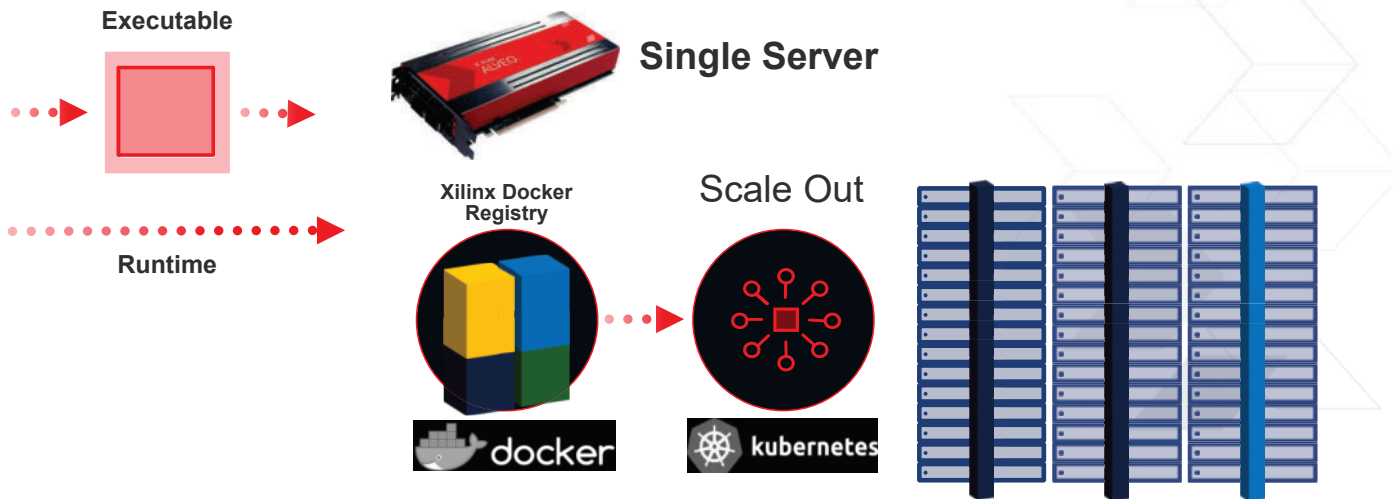


What is Vitis AI?



Vitis AI Deployment

Embedded (Zynq SoC & MPSoC, Versal ACAP)



>> 17

© Copyright 2020 Xilinx



Vitis AI Model Zoo

- > **Shared Repository of Pre-Trained AI Models**
 - >> Ready to Deploy, Pre-Optimized Models
 - >> A lot of Models Supporting Broad Range of Applications
 - >> Open and Available on [GitHub](#)
- > **Leverage Standard Frameworks, Networks, Datasets**
 - >> Trained Using TensorFlow and Caffe
- > **Deploy As-is, Re-Train or Further Optimize**
 - >> Caffe_Xilinx, a custom distribution of Caffe provided to test & finetune caffe models
 - >> Training code, test code and train eval instructions provided

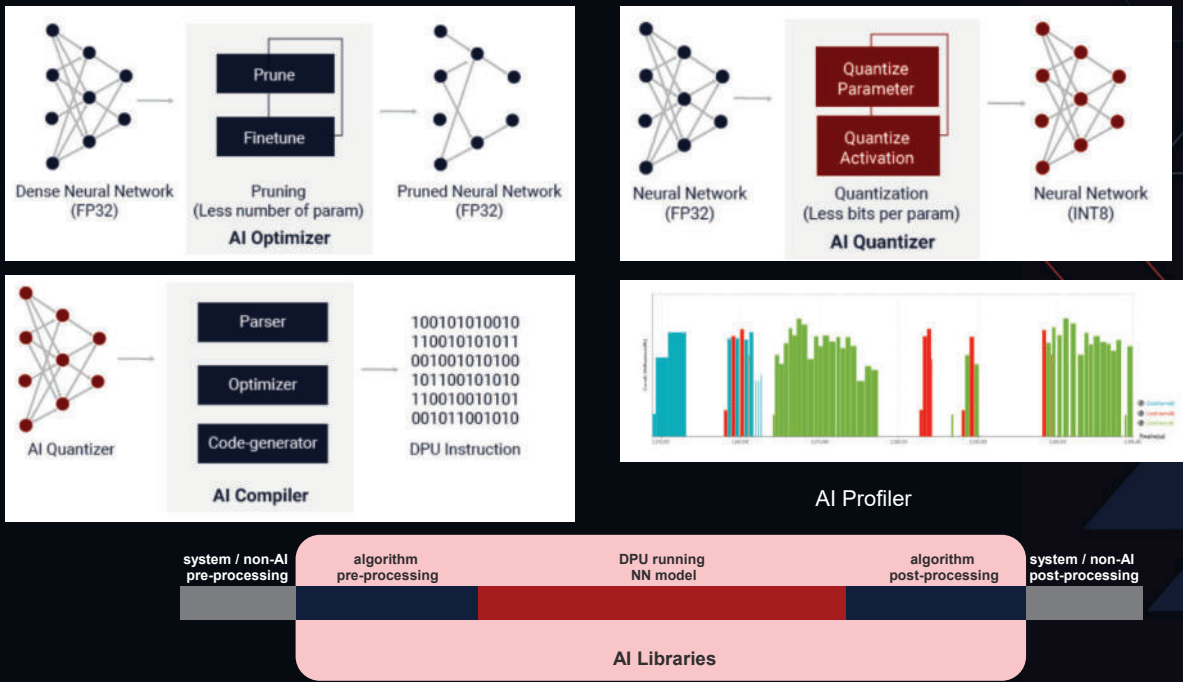
Application	Model
Face	Face detection
	Landmark Localization
	Face recognition
Pedestrian	Face attributes recognition
	Pedestrian Detection
	Pose Estimation
Video Analytics	Person Re-identification
	Object detection
	Pedestrian Attributes Recognition
	Car Attributes Recognition
	Car Logo Detection
	Car Logo Recognition
	License Plate Detection
License Plate Recognition	
ADAS/AD	Object Detection
	3D Car Detection
	Lane Detection
	Traffic Sign Detection
	Semantic Segmentation
	Drivable Space Detection

>> 18

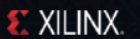
© Copyright 2020 Xilinx



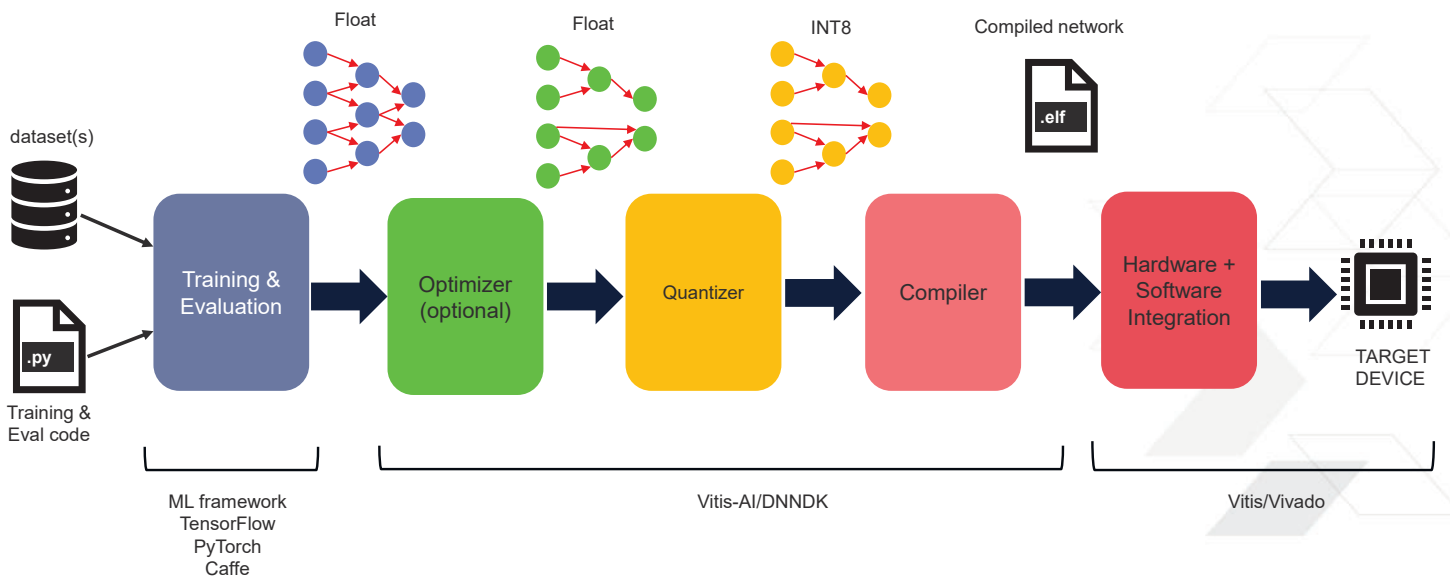
Vitis AI Development Kit



>> 19



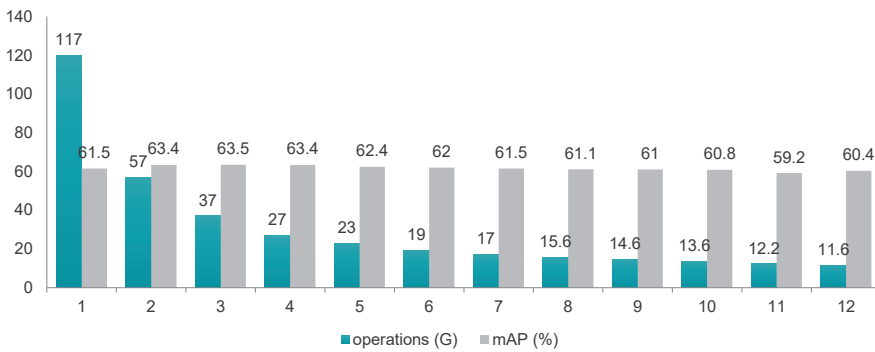
Xilinx AI Flow Overview



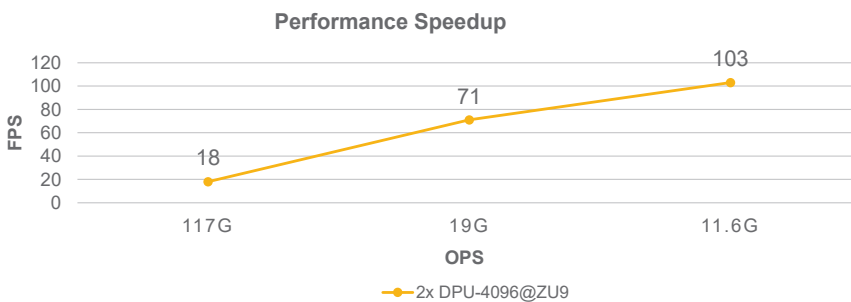
© Copyright 2020 Xilinx



Example Using Vitis AI Optimizer



SSD+VGG @ Surveillance 4 Classes



© Copyright 2020 Xilinx



Vitis AI Quantizer Result

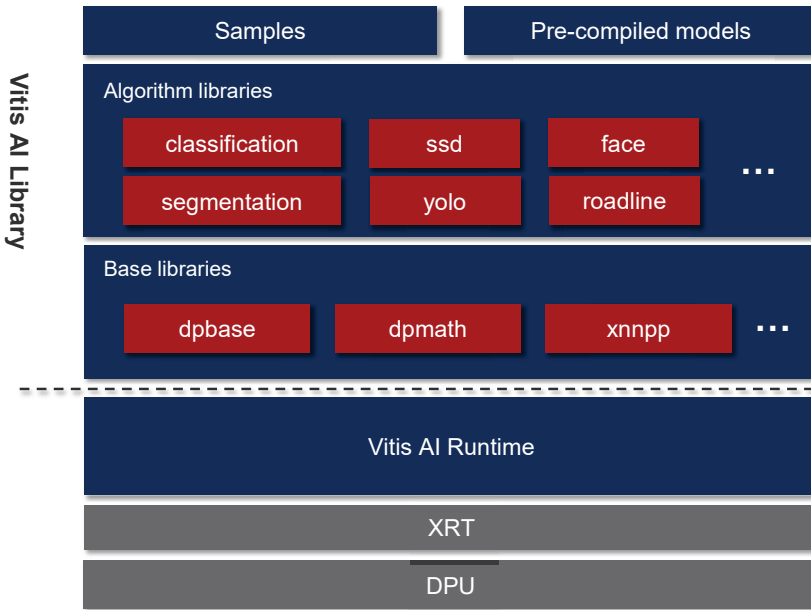
Classification Networks	Float		8 bit quantized				After quantized finetune			
	Top1	Top5	Top1	ΔTop1	Top5	ΔTop5	Top1	ΔTop1	Top5	ΔTop5
Inception_v1	66.90%	87.68%	66.54%	-0.36%	87.58%	-0.10%	66.62%	-0.28%	87.58%	-0.10%
Inception_v2	72.78%	91.04%	71.93%	-0.85%	90.58%	-0.46%	72.40%	-0.38%	90.82%	-0.23%
Inception_v3	77.01%	93.29%	76.26%	-0.75%	92.85%	-0.44%	76.56%	-0.45%	93.00%	-0.29%
Inception_v4	79.74%	94.80%	79.04%	-0.70%	94.53%	-0.27%	79.42%	-0.32%	94.64%	-0.16%
ResNet-50	74.76%	92.09%	73.74%	-1.02%	91.44%	-0.65%	74.59%	-0.17%	91.95%	-0.14%
VGG16-3fc-float	70.97%	89.85%	70.67%	-0.30%	89.72%	-0.13%	70.74%	-0.23%	89.79%	-0.06%
MobileNet_v1	70.61%	89.63%	68.01%	-2.60%	88.14%	-1.49%	69.71%	-0.90%	89.06%	-0.57%

Detection Networks	Dataset	Float mAP	8 bit quantized mAP	ΔmAP
SSD_VGG	VOC 21 classes	76.47%	76.27%	-0.20%
SSD_MobileNet_v2	BDD100k 11 classes	30.80%	29.70%	-1.10%
SSDLite_MobileNet_v2	Customer's data	20.28%	20.12%	-0.16%

© Copyright 2020 Xilinx



Vitis AI Library



- > **Ease of Use**
 - >> Quicker mode deployment
 - >> Easier AI application development
- > **Performance Optimized**
 - >> Optimized pre & post processing
- > **Open**
 - >> Open source
 - >> Fully sync with Vitis AI Model Zoo

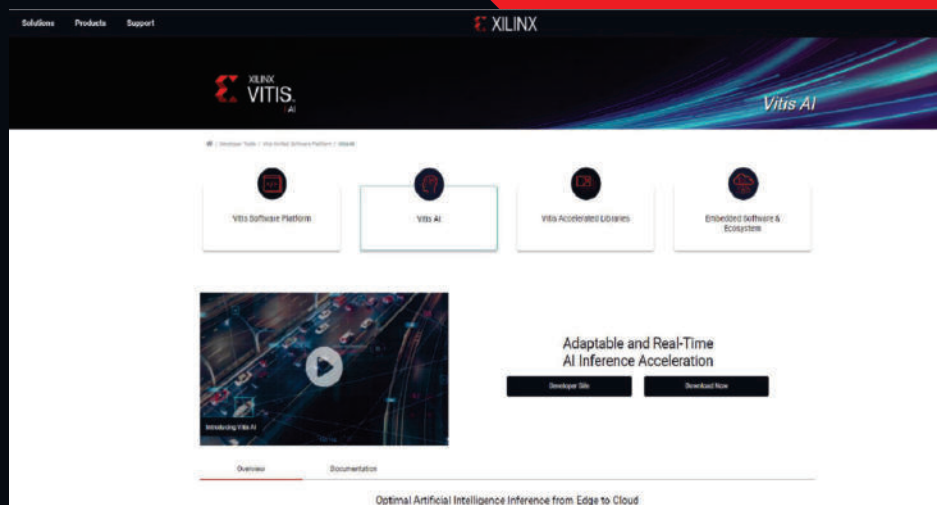
© Copyright 2020 Xilinx



<https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html>

THANK YOU!

daniele.bagni@xilinx.com



Where to find more information



More info about Versal ACAP

- > <https://www.xilinx.com/products/silicon-devices/acap/versal.html>
- > <https://www.xilinx.com/products/silicon-devices/acap/versal-ai-core.html>
- > <https://www.xilinx.com/products/silicon-devices/acap/versal-ai-core.html#productTable>
- > https://www.xilinx.com/support/documentation/white_papers/wp505-versal-acap.pdf
- > https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf
- > https://www.xilinx.com/support/documentation/white_papers/EW2020-Deep-Learning-Inference-AICore.pdf
- > https://www.xilinx.com/support/documentation/white_papers/ACAP%20Paper.pdf
- > <https://www.xilinx.com/products/silicon-devices/acap/versal-ai-core.html#getStarted>

More info about Vitis design flow

- > <https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html>
- > <https://developer.xilinx.com/>
- > https://www.xilinx.com/html_docs/xilinx2020_1/vitis_doc/index.html



>> 27

© Copyright 2020 Xilinx

XILINX

More info about Vitis AI tools and Libraries

- > [Getting Started from Vitis AI Github](#)
- > [Vitis AI Model Zoo](#)
- > [Vitis AI Optimizer Lounge*](#)
- > [Vitis AI DPU TRD](#)
- > [Vitis AI Library](#)
- > [Vitis AI Tutorial](#)
- > [Vitis AI Forum](#)

* Approval required

Name	Last commit	Last update
alioo	Update README.md	1 day ago
dlcc	Update tool_run_docker.md	2 days ago
imprec	Update runtime_docker.md	15 hours ago
UCENSE	ADD LICENSE	3 days ago
README.md	Update README.md	1 day ago

Introduction
This repository includes got platforms. These models co center, etc. You can get star

Model Information
The following table includes comprehensive information about each model, including application, framework, training and validation dataset, backend, input size, computation as well as float and quantized precision.

Use Application

Frameworks: Caffe, TensorFlow

Vitis AI Models: Model Zoo, Custom Models

>> 28

© Copyright 2020 Xilinx

XILINX

Further old (but still good) references

- > <https://www.xilinx.com/publications/events/developer-forum/2018-frankfurt/xilinx-machine-learning-strategies-with-deepi-tech.pdf>
- > <https://www.xilinx.com/publications/events/developer-forum/2018-frankfurt/machine-learning-for-embedded-deep-dive.pdf>
- > <https://github.com/Xilinx/graffitist>

>> 29

© Copyright 2020 Xilinx

 XILINX.

About me

 XILINX.

<http://www.xilinx.com/publications/archives/xcell/Xcell86.pdf>

ASK FAE - X

Median Filter and Sorting Network for Video Processing with Vivado HLS

by **Daniele Bagni**
DSP Specialist
Xilinx, Inc.
daniele.bagni@xilinx.com



Daniele Bagni is a DSP Specialist FAE for Xilinx EMEA in Milan, Italy. After earning a degree in quantum electronics from the Politecnico of Milan, he worked for seven years at Philips Research labs in real-time digital video processing, mainly motion estimation for field-rate upconverters. For the next nine years he was project leader at STMicroelectronics' R&D labs, focusing on video coding algorithm development and optimization for VLIW architecture embedded DSP processors, while simultaneously teaching a course in multimedia information coding as an external professor at the State University of Milan. In 2006 he joined the Xilinx Milan sales office. What Daniele enjoys most about his job is providing customers with feasibility studies and facing a large variety of DSP applications and problems. In his spare time, he likes playing tennis and jogging.

230000 downloads in the first 2 weeks!

© Copyright 2020 Xilinx



<http://www.xilinx.com/publications/archives/xcell-software/xcell-software1.pdf>

XCELL SOFTWARE JOURNAL XCELLENCE WITH SDSOC

Using the SDSoC IDE for System-level HW-SW Optimization on the Zynq SoC

by **Daniele Bagni**
DSP Specialist FAE
Xilinx, Inc.
danieleb@xilinx.com

Nick Ni
Product Manager, SDSoC
Development Environment
Xilinx, Inc.
nickn@xilinx.com

© Copyright 2020 Xilinx





Outline

Abstract

Keywords

- 1. Introduction
 - 2. Related work
 - 3. The proposed method
 - 4. FPGA implementation
 - 5. Experimental results and discussion
 - 6. Conclusions
- References



Computer Vision and Image Understanding

Volume 114, Issue 11, November 2010, Pages 1164-1179



A real-time versatile roadway path extraction and tracking on an FPGA platform

Roberto Marzotto ^a, Paul Zoratti ^b, Daniele Bagni ^c, Andrea Colombari ^a, Vittorio Murino ^{a, d, e}

© Copyright 2020 Xilinx



Application Note: Vivado HLS



XAPP1163 (v1.0) January 23, 2013

Floating-Point PID Controller Design with Vivado HLS and System Generator for DSP

Author: Daniele Bagni, Duncan Mackay

Summary

This application note describes how to quickly implement and optimize floating-point Proportional-Integral-Derivative (PID) control algorithms specified in C/C++ code into an RTL design using Vivado HLS. You can use System Generator for DSP to easily analyze and verify the design. This enables floating-point algorithm designers to take advantage of high-performance, low cost, and power efficient Xilinx® FPGA devices.

Introduction

Floating-point algorithms are widely used in industries from analysis to control applications. Traditionally, such algorithms have been implemented on microprocessors. The primary reason for using microprocessors has been the ease with which floating-point algorithms can be captured, validated, and debugged in C/C++ code, therefore avoiding the complexity and skills required to implement them in hardware. However, implementing these algorithms on optimized and dedicated hardware provides lower cost, higher performance, and power benefits over a standard, or even optimized microprocessor.

This application note presents a new design flow enabled by the Xilinx Vivado™ Design Suite, which allows floating-point algorithms to be quickly specified in C/C++ code, optimized for performance, and implemented on Xilinx FPGA devices. This flow delivers on the cost,

© Copyright 2020 Xilinx





A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS

Author: Daniele Bagni, A. Di Fresco, J. Noguera, F. M. Vallina

Summary

This application note describes how to use Vivado® High Level Synthesis (HLS) to develop a floating-point matrix multiplication accelerator connected via an AXI4-Stream interface to the Accelerator Coherency Port (ACP) of the ARM CPU in the Zynq®-7000 All Programmable SoC (AP SoC) device.

The floating-point matrix multiplication accelerator modeled in C/C++ code can be quickly implemented and optimized into a Register Transfer Level (RTL) design using Vivado HLS. The solution is then exported as an IP core connected with an automatically-created AXI4-Stream interface to the ACP on AP SoC Processing Subsystem (PS). The connection is made through a Direct Memory Access (DMA) core in the AP SoC Programmable Logic (PL) subsystem. Vivado IP Integrator (IPI) is used to design the AP SoC PL hardware, including the matrix multiplier peripheral, the DMA engine, and an AXI timer. The Software Development Kit (SDK) is used to design the AP SoC PS software to manage the peripherals.

© Copyright 2020 Xilinx



Demystifying the Lucas-Kanade Optical Flow Algorithm with Vivado HLS

Authors: Daniele Bagni, Pari Kannan, and Stephen Neuendorffer

Summary

The Lucas-Kanade (LK) algorithm for dense optical flow estimation is a widely known and adopted technique for object detection and tracking in image processing applications. This algorithm is computationally intensive and its implementation in an FPGA is challenging from both a design and a performance perspective. This application note describes how to implement the LK algorithm with the Xilinx Vivado® High-level Synthesis (HLS) tool to achieve real-time performance in the Zynq®-7000 All Programmable (AP) SoC without image quality degradation.

A real-time demonstration on a Zynq-7000 AP SoC reference board was built with the SDSoc™ development environment's integrated tool. The design reads video data from a file and writes back the processed data to a file, instead of reading and writing frame buffers. The design was created in less than eight weeks by an engineer. This application note also serves as a tutorial demonstrating good C/C++ coding techniques for obtaining the best performance from Vivado HLS in image processing. Download the [reference design files](#) for this application note from the Xilinx website. For detailed information about the design files, see [Reference Design](#).

© Copyright 2020 Xilinx



My Vitis AI tutorials on Deep Learning for Computer Vision

- > **Quantization and Pruning of AlexNet CNN trained in Caffe with Cats-vs-Dogs dataset**
 - >> <https://github.com/Xilinx/Vitis-AI-Tutorials/tree/VAI-Caffe-ML-CATSvsDOGS>
- > **Deep Learning with Custom GoogleNet and ResNet in Keras and Xilinx Vitis AI**
 - >> <https://github.com/Xilinx/Vitis-AI-Tutorials/tree/Keras-GoogleNet-ResNet>
- > **FCN8 and UNET Semantic Segmentation with Keras and Xilinx Vitis AI**
 - >> <https://github.com/Xilinx/Vitis-AI-Tutorials/tree/VAI-KERAS-FCN8-SEMSEG>

© Copyright 2020 Xilinx



Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

Automotive Applications Disclaimer

XILINX PRODUCTS ARE NOT DESIGNED OR INTENDED TO BE FAIL-SAFE, OR FOR USE IN ANY APPLICATION REQUIRING FAIL-SAFE PERFORMANCE, SUCH AS APPLICATIONS RELATED TO: (I) THE DEPLOYMENT OF AIRBAGS, (II) CONTROL OF A VEHICLE, UNLESS THERE IS A FAIL-SAFE OR REDUNDANCY FEATURE (WHICH DOES NOT INCLUDE USE OF SOFTWARE IN THE XILINX DEVICE TO IMPLEMENT THE REDUNDANCY) AND A WARNING SIGNAL UPON FAILURE TO THE OPERATOR, OR (III) USES THAT COULD LEAD TO DEATH OR PERSONAL INJURY. CUSTOMER ASSUMES THE SOLE RISK AND LIABILITY OF ANY USE OF XILINX PRODUCTS IN SUCH APPLICATIONS.

© Copyright 2013–2016 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

© Copyright 2020 Xilinx



Porting a Gesture Recognition Neural Network composed of CNN and LSTM to a FPGA-SoC

ROBERT BRIEGEL



24.09.2020

FPGA4ADAS 2020 - ROBERT BRIEGEL

1

Agenda



1. Goals
2. **SiComAs** Algorithm
3. Approach
4. Realization
5. Results
6. Further Improvements

24.09.2020

FPGA4ADAS 2020 - ROBERT BRIEGEL

2

Goals

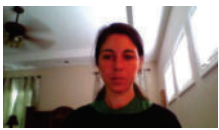
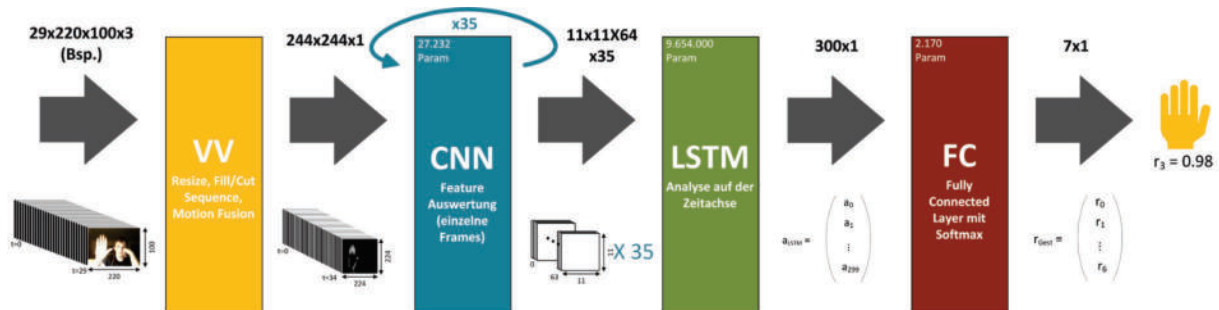
Porting the Neural Network **SiComAs** to a FPGA-SoC platform

- Retention of the architecture of the NN and achieved accuracy
- Maximizing throughput

What's the SiComAs?

- **Sign Communication Assistant** developed by EDAG in Lindau
- Takes video as input and recognizes gestures
- Developed to further improve Human-Vehicle interface while not excluding the hearing impaired

Sign Communication Assistant



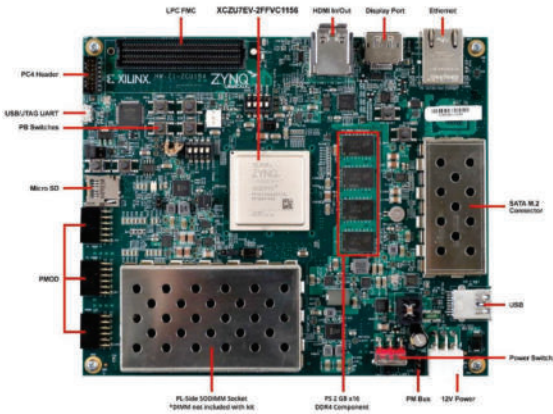
Approach Framework

Xilinx Deep Neural Network Development Kit (Now part of VITIS-AI)

- Allows porting of CNNs
- LSTM cells are **not** (yet) supported, therefore must be computed in other ways
- Tensorflow Interface
- Inference using a co-processor in programmable logic (DPU)

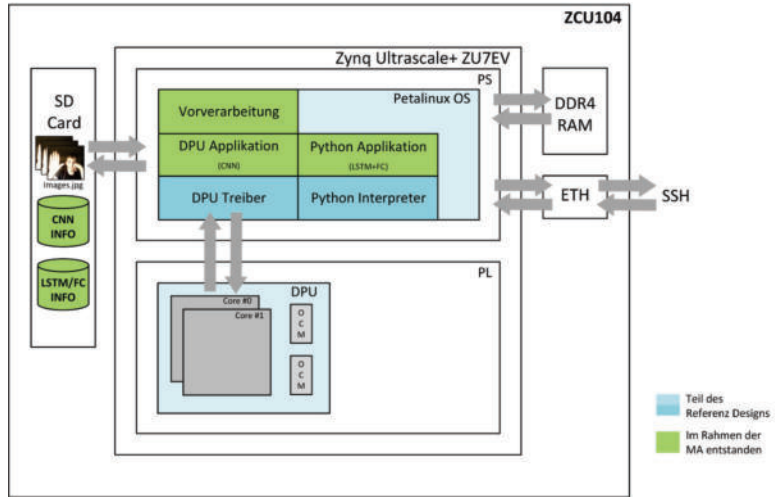
Approach

Hardware and Top-Level Architecture



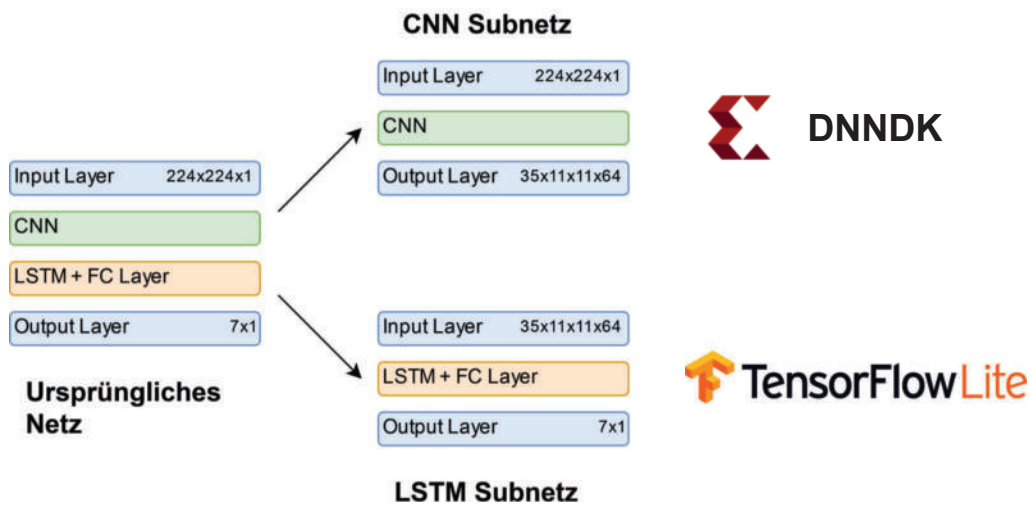
Xilinx ZCU104 Zynq Ultrascale+ MPSoC Evaluation Kit¹

¹<https://www.xilinx.com/products/boards-and-kits/zcu104.html>



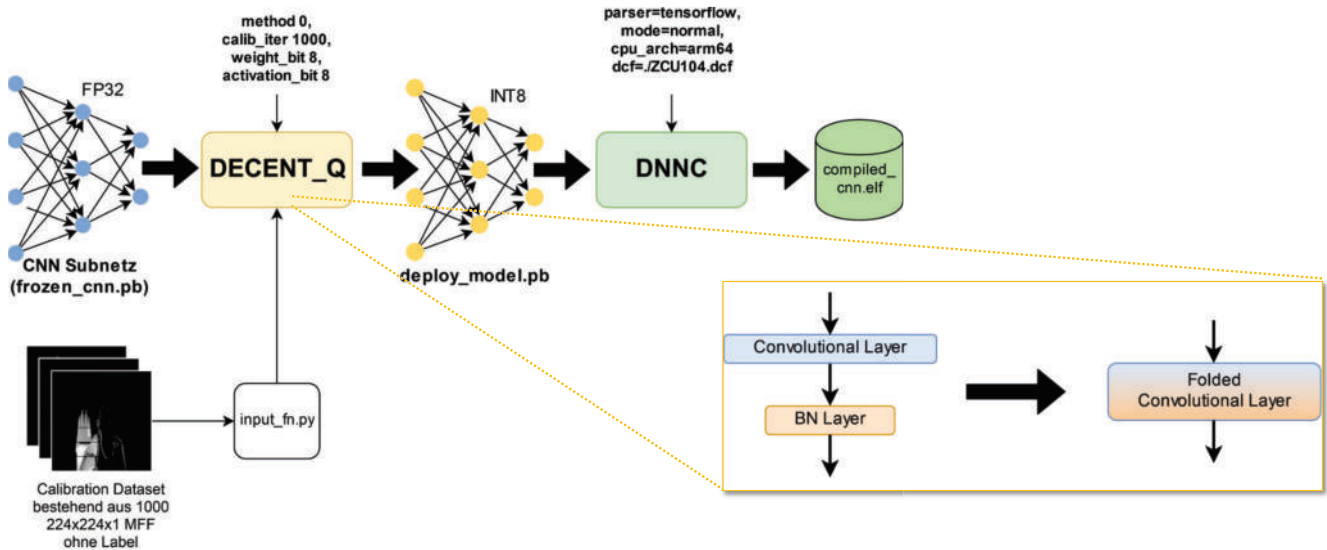
Realization

Splitting into subnets



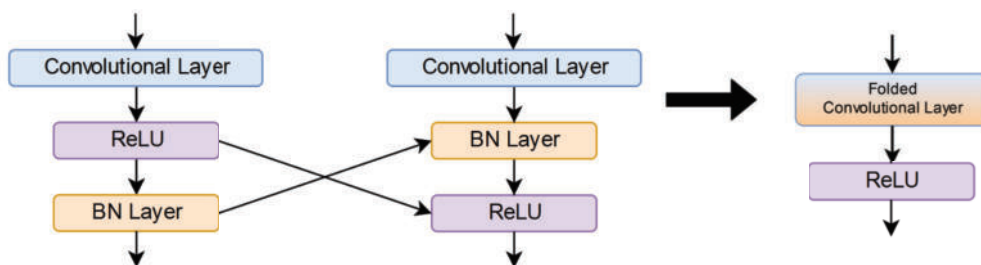
Realization

Preparing the CNN



Realization

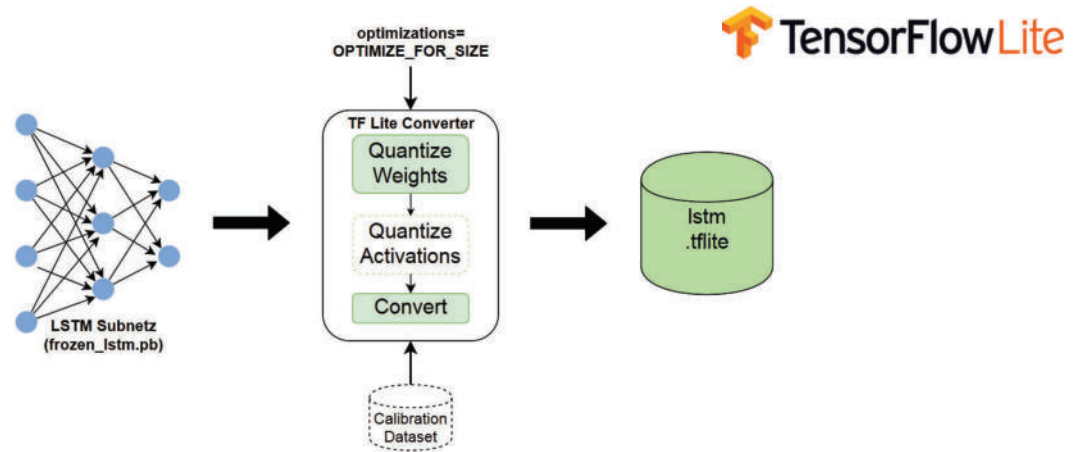
Changing the layer sequence



- Re-training of the network led to a loss of classification accuracy of 2.15% (absolute)

Realization

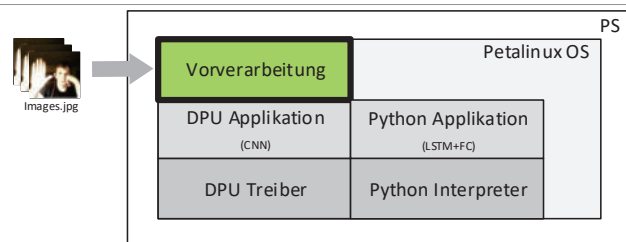
Preparing the LSTM



Realization

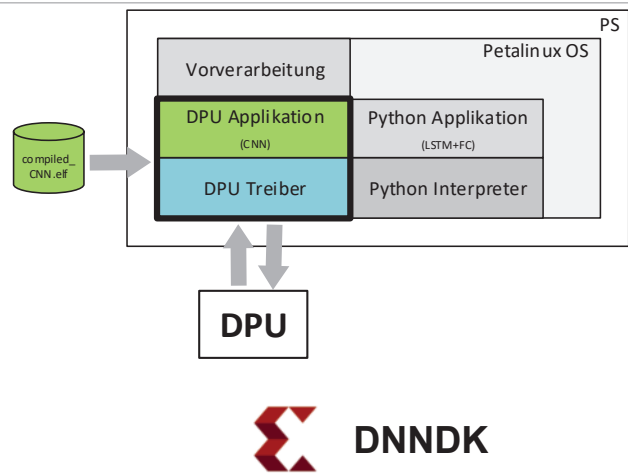
Preprocessing

- Implemented in C++ using the OpenCV library
- Loads videos from the SD card and performs pre-processing:
 - Motion Fusion
 - Resize
 - RGB→SW
 - Fixed video length: 35 Frames



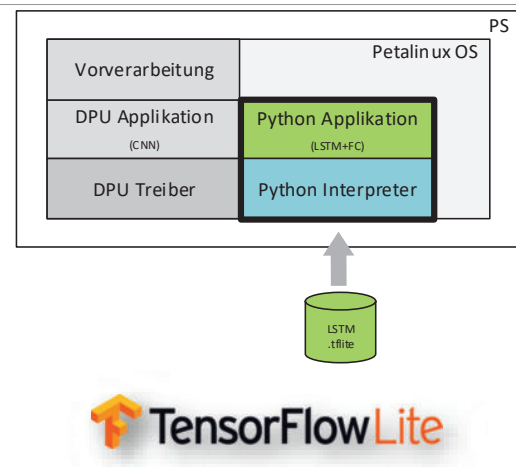
Realization DPU Application

- Implemented in C++ using the DPU driver provided by Xilinx
- Initializes the DPU IP core, controls the data flow and triggers the inference of the individual frames
- Multithreading ensures full utilization of the DPU IP core



Realization Python Application

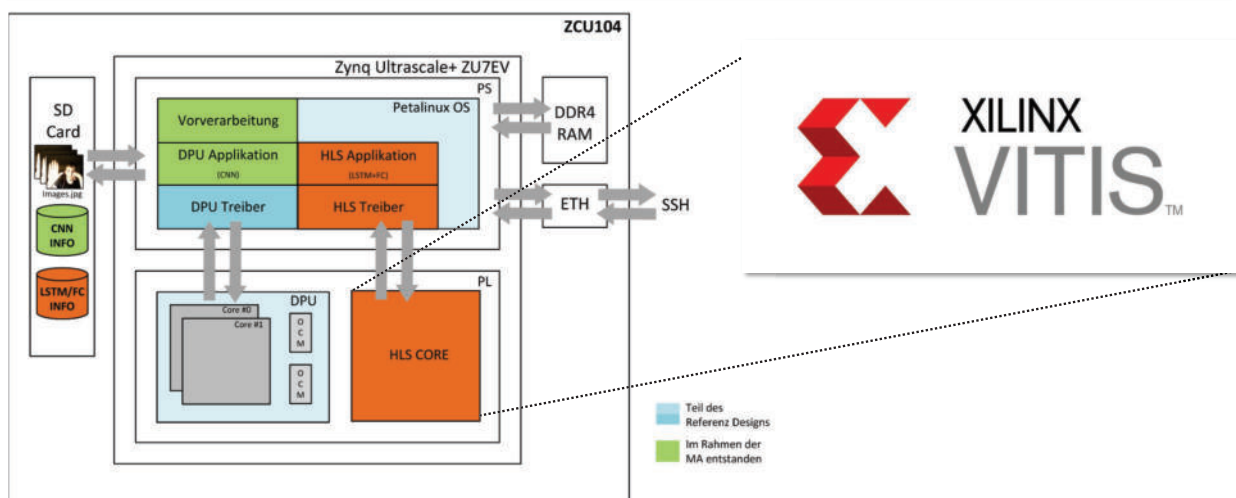
- Implemented using Python and the Tensorflow Lite Interpreter
- performs inference of the LSTM cell and the final classification layer



Results In Summary

- Top 1 classification accuracy was measured at **89.86 %**
(compared to 92.00% originally)
- Inference time **CNN: Ø 22.90 milliseconds**
(35 Frames, 4 Threads)
- Inference time **LSTM: Ø 263.69 milliseconds**

Further Improvements ...yet to be made



HAPPi-Net: Hardware-aware Performant Perception of Neural Networks - Designing Lightweight CNNs on Embedded Platforms

Alexander Frickenstein^{*1}, Manoj-Rohit Vemparala^{*1}, Nael Fafous^{*2}, Lukas Frickenstein^{*1}, Walter Stechele²

Abstract—In the field of autonomous driving and other robotic applications, embedded hardware (HW) platforms are either resource or power constrained limiting the computational complexity, the memory utilization and/or the memory bandwidth. This circumstance hinders modern convolutional neural networks (CNNs) being deployed on such systems, however, emphasizes the role of CNN optimization. Individual approaches for CNN optimization are discussed in the context of a complete HW-CNN co-design process. In this work, we demonstrate one top-down approach leveraging a filter-wise pruning technique, namely the autoencoder-based low rank filter sharing (ALF) technique, to be used on various parallelized algorithms and hardware accelerators. In the context of a meet-in-the-middle design approach, we present a binary drivable area detection neural network (binary DAD-Net) which is accelerated by the run-time reconfigurable processing element OrthrusPE, for embedded friendly applications. Based on the aforementioned co-design processes, it becomes clear that different optimization techniques leverage their efficacy later in the design process or depend on consecutive optimizations.

I. INTRODUCTION

The automobile has formed the basis for private transportation for over 100 years. It has made a significant contribution to the comfort and freedom of people. In the past few years, increased traffic volumes have led to undesirable outcomes, namely traffic jams and environmental pollution. With the electrification and automation of cars, private transportation can be made more productive and relaxing for people and their environment. Here, neural networks are the key technology for autonomous vehicles. More broadly, in the field of computer vision, tasks such as image classification or semantic segmentation form the fundamental complexity of most applications. This has made neural network algorithms the de facto standard for solving many tasks in the field. Only the storage requirements, the computing complexity and the energy demand pose great challenges for electric autonomous cars. Optimization methods offer a good solution to this. Since the optimization has a significant impact on the deployment process, a detailed consideration of the methods used and their interaction with the respective design phase

is crucial. An understanding of the optimization methods in the deployment process is beneficial in two ways. Firstly, more efficient applications are achieved that either allow the use of less expensive control devices and/or provide space for additional features. Secondly, it simplifies the design process in a data driven development by better separation of design steps (CNN design, optimization, hardware implementation), forming understandable dependencies between stakeholders and a clearer assignment of expert knowledge. Since a uni-directional, deep-dive deployment approach can be complex and lead to many inefficiencies in the design, we illustrate alternative approaches of two design methodologies.

II. RELATED WORK

Mitigating the challenges of deploying deep and wide, high-performance neural networks on the resource constrained environment of edge devices is a focal point of enabling CNNs for embedded applications. Thus, both industry and academia focus to reduce redundancies arising from training deeper and wider network architectures [1].

A. Neural Network Compression

Quantization and pruning render compression techniques that can potentially be exploited to reduce the aforementioned redundancies, resulting in efficient CNNs for deployment on embedded hardware.

Pruning, aims to reduce the parameter or feature redundancy in a neural network. The removal of network parameters has a direct impact on the network's memory footprint, as well as the total number of computations necessary. For the latter, the inference platform needs to be compatible with the pruning granularity applied to the network. A common example of mismatched pruning and inference hardware is using element-wise pruning on a general matrix multiplication (GEMM) accelerator. Removing individual elements from a kernel makes it sparse, however, the GEMM operation is inherently rigid and cannot trivially accommodate sparsity in its execution. GEMM flattens the convolution's strided operation into replicated inputs and kernels to reshape the problem at hand and execute it on fast matrix-matrix or matrix-vector multipliers.

^{*}Authors contributed equally

¹BMW Group, Autonomous Driving, Munich, Germany,
<Firstname>.<Lastname>@bmw.de

²Technical University of Munich, Munich, Germany,
<Firstname>.<Lastname>@tum.de

We can classify pruning techniques into two general domains, namely rule-based and learning-based compression. As implied, rule-based compression techniques rely on having static or pseudo-static rules, which enforce/impose the compression of a given CNN accordingly. Hand-crafted pruning represents a further subgroup of rule-based compression, where heuristics are utilized to determine the saliency of neurons. An example of this is where Han et al. [2] utilize the magnitude of weights to determine their saliency, resulting in element-wise pruning and leading to irregular compute structures (irregular pruning). The resulting inefficient memory accesses render irregular pruning as impractical for general purpose computing platforms. To overcome these limitations, introducing regularity in the pruning process becomes an essential criterion for performing accelerator-aware compression. The design process for matching the target hardware to the software/model is simplified such that designing complex hardware, e.g. the Efficient Inference Engine (EIE) [3] or the Sparse-CNN (SCNN) accelerator [4], becomes unnecessary. Due to the compelling characteristics of regular pruning, Frickenstein et al. [5] proposed a structured, kernel-wise magnitude pruning method along with a scalable and sparse algorithm. He et al. [6] utilize a geometric mean heuristic to prune redundant filter in a CNN, resulting in a hardware inference friendly network. However, removing filters directly impacts the input channels of subsequent layers, potentially resulting in a more significant task-related accuracy degradation. Despite the tempting simplicity of rule-based compression techniques, they overly generalize the problem and struggle to find a one-size-fits-all rule considering the varying nature of CNNs with respect to complexity, structure and target task.

To address the shortcomings of rule-based compression, recent works in literature [7], [8] have adopted learning-based compression techniques. Both works utilize a reinforcement learning (RL) agent that is capable of learning the criteria of the pruning process, which is formalized as an optimization problem and a corresponding cost function.

Further, Huang et al. [7] define the environment for an RL-agent as a CNN, where both an accuracy term and an efficiency term are utilized to formulate the training policy for the agent. This non-differentiable policy is used to maximize the contrary objectives and the agent tries to find a balanced trade-off. The fact that an agent needs to be trained individually for each layer and the increasing search complexity for layers with multiple channels results in a slow and greedy process of the model exploration. He et al. [8] cut down the exploration time by utilizing a RL-agent that prunes without fine-tuning at intermediate stages. Additionally, layer characteristics such as size, stride and operations (OPs) serve as further inputs to the agent. However, the formulation of the cost function of such techniques is non-trivial and is dependent on expert knowledge and trial-and-error. The design space of the environment of the agent renders an additional configuration parameter, making it difficult to test many configurations of the underlying problem. Each of such combinations of neural network and target task combination

results in a new and unique problem, requiring a unique solution.

The pruning scheme presented by Guo et al. [9] incorporates pruning in the training flow by introducing learnable parameters, which can be recovered if necessary, to dynamically prune the underlying CNN, resulting in irregular sparsity. Cardinality constraints are incorporated by Zhang et al. [10] into the training objective to obtain different pruning regularities. Bagherinezhad et al. [11] proposed Lookup-based CNNs to learn a dictionary of shared filters at training time. The lookup dictionary is then utilized during inference to perform the convolution on the input, resulting in low computational complexity. When considering the similarities for filter-sharing and weight-sharing, the weight-sharing approach presented by Bagherinezhad is arguably closest to the technique presented in ALF framework [12]. However, the methodology and the training procedure fundamentally differ.

Differently, Neural Architectural Search (NAS) techniques demonstrated the capability to successfully optimize CNN models at design-time. Synergies emerged between CNN design and the target hardware platform by combining NAS with Hardware-in-the-Loop (HIL) testing. MNAS-Net, proposed by Tan et al. [13], comprises NAS with a RL-agent for mobile devices. Cai et al. [14] focus on deriving specialized, hardware-specific CNN architectures from over-parameterized models with their proposed ProxylessNAS.

Quantization, including its most drastic form, binarization, aims to reduce the representation redundancy of model parameters by constraining the range of possible values and mapping them to a discrete domain of quantized representations [15], [16], [17]. It is worth mentioning, that both quantization and binarization are orthogonal to pruning and can be applied in combination with e.g. our ALF method.

Binarization of a CNN constrains its weights and activations to $\{-1, 1\}$. Along with the compelling benefits of computational efficiency and reduction in memory requirements, binarization leads to a degradation in accuracy compared to the full-precision counterpart. To tackle the degradation in accuracy, Rastegari et al. [18] introduced XNOR-Net, where they estimate real-valued weights and activations by introducing a scaling factor and corresponding binary weights and activations. With the introduction of CompactBNN, Tang et al. [19] observed that binarizing activations is more challenging compared to the binarization of weights and thus, focused on improving the approximation of activations. They further improved the training by proposing a ReLU activation function with trainable parameters. In ABC-Net [20], Lin et al. approximate full-precision weights and activations by a linear combination of binary representations with corresponding shifting and scaling factors. They highlighted the appealing characteristic of BinaryNets with multiple weight and activation bases against equivalent fixed-point quantized CNNs with regard to embedded systems. Considering 45nm CMOS technology, a MAC operation consumes $> 8\times$ more power than a bit-wise operation [21].

The previous mentioned publications focused on improv-

ing binary neural networks for image classifications, however another set of work studied binary object detection models such as the work from Hanyu et al. [22]. Further, Zhuang et al. [23] propose GroupNet with multiple binary bases, where they extend the approximation towards the structural level. GroupNet’s structural complexity allows it to extend the effectiveness of BNNs to challenging semantic segmentation tasks. In the scope of GroupNet, Zhuang et al. introduce the Binary Parallel Atrous Convolution (BPAC) module, consisting of multiple dilated convolutions with various dilation rates, up to 16. This results in irregular memory accesses and a higher power-consumption of the memory controller, as pointed out in [5].

B. Efficient Hardware Processing

Several accelerators have been developed with processing elements designed to exploit performance boosts due to variable quantization levels [24], [25], [26], [27]. Other accelerators were designed to solely execute BNNs [28], [29], [30]. With the popularity of quantization as a compression technique, commercial hardware providers also offer support for low-bitwidth operations on their compute platforms [31], [32], [33], [34]. NVIDIA’s Turing architecture employs Tensor Cores which can operate at FP16 precision, while offering higher throughput at lower INT8 and INT4 precision modes [32]. Intel provides Vector Neural Network Instruction (VNNI) libraries which pack more low-precision operations in a single processor instruction. Intel’s Arria 10 FPGAs provide support for efficiently utilizing their DSPs’ native bitwidth to execute parallel low-precision operations, effectively transforming them into vectorized processing elements [35]. Similarly, the Xilinx DSP48 blocks can be utilized to perform up to 48 parallel binary operations [17]. The Xilinx Versal platform [34] provides AI-Engine cores which can perform 32, 16 and 8 bit operations, at higher degrees of vectorization respectively. Coupled with the DSP engines and the programmable logic block, a wide variety of quantization levels can be supported on a single compute platform.

Bit Fusion [24], UNPU [25], Stripes [26] and Loom [27] are all based on ASIC designs. UNPU, Stripes and Loom offer single bit operations while the Bitbricks structure used in Bit Fusion allows the execution of operations at fixed quantization levels, making the smallest possible precision bounded by the size of a single Bitbrick. UNPU, Stripes and Loom are capable of performing both binary and fixed-point operations, however, with a non-negligible overhead, due to the support of variable quantization levels. Further ASIC-based works, BREin [29] and YodaNN [28], were developed precisely to accelerate BNNs. However, they do not implement the binary bases required for accurate binary nets, nor do they support the shifting and scaling of the intermediate maps.

FINN [30] is a popular framework for accelerating BNNs on FPGAs. The framework is geared towards BNNs similar to the ones proposed in [36]. FINN compiles HLS code from a BNN description to create a bit file that exactly

suits that network. The first and last layers are not binarized in their *CNV* network, making the fixed-precision hardware utilized for those layers only useful for those two parts of the network. In our proposed solution, every instantiated DSP can be used for any part of the entire network, due to the dual modes of the PE that can be reconfigured at run-time. Furthermore, FINN is not compatible with multiple binary bases, but rather simpler BNNs suited for problems such as MNIST, CIFAR-10 or SVHN. Other FPGA-based BNN accelerators [37], [38], [39] also execute binary operations purely on LUTs and utilize DSPs for fixed-point operations, where they are supported.

The authors of Double MAC [40] extract more functionality from FPGA hard blocks. They precondition the signals going into DSP blocks such that two multiplications can be obtained with some post-processing. This leverages quantization, since the two results obtained at the output are calculated from operands that are smaller than the maximum possible precision that the DSP can offer. Their work virtually turns DSPs into SIMD multipliers. Similarly, our proposed solution turns DSPs into SIMD binary Hadamard product processing units. Our solution is orthogonal to Double MAC, making it possible to include Double MAC as a *third* mode in OrthrusPE.

Efficient exploitation of hard blocks on FPGAs can play a key role in lowering the efficiency gap between ASIC and FPGA implementations [41]. This is evident in the recent trend of FPGA manufacturers adding more hard blocks to their chips aimed at accelerating deep neural network applications [34].

C. Semantic Segmentation

One of the first prominent semantic segmentation models proposed was the Fully Convolutional Network (FCN), which was successfully adopted by Shelhamer et al. [42]. An important aspect of FCN are the skip connections which capture the intermediate features from the high level feature maps during the up-sampling stage through 1×1 convolutions. This method paved the way to more structured models such as UNet [43]. This structured up-sampling provides higher accuracy than single $\times 8$ up-sampling present in FCN. However, this increases the computational complexity due to additional up-sampling layers.

DeepLab, proposed in [44], utilizes dilated convolution instead of down-sampling the feature maps, maintaining a sufficient receptive field. The pooling or strided convolution is avoided for the last set of feature maps. This would increase the computational costs as the convolution is performed on larger feature maps. The encoder network is downsampled by a factor of 8/16 instead of 32. The down-sampled feature maps are then passed to a spatial pyramid pooling module, which consists of parallel dilated convolution with different rates followed by concatenation and point-wise convolution. This module produces better segmentation results by extracting multi-scale information. Multi-class semantic segmentation has a negative effect on the precision of the drivable area detection algorithm and

their vast number of MAC operations making the application impractical for embedded systems.

III. DESIGN PROCESS OF EFFICIENT DEEP NEURAL NETWORKS

To meet the ever-increasing challenges of automated driving, such as quality of service or security requirements, modern CNN architectures are becoming larger and deeper. In this context, the interest in model compression, *i.e.* the process of deploying a lightweight CNN, is gaining more importance. A lightweight model is the optimal variant from the designer’s point of view, *i.e.* with the lowest memory requirements, the best performance, the lowest energy consumption or a trade-off from different criteria. The deployment process varies for different optimization methods and hardware accelerators. On the one hand, the optimization methods have different dependencies in the algorithmic or hardware implementation. On the other hand, different hardware accelerators require different programming mechanisms. The design process can be formulated as a top-down or bottom-up approach. In the case of a two-sided deployment, *i.e.* a concurrent top-down and bottom-up work, a meet-in-the-middle approach can emerge. Based on the abstraction (architectural, algorithmic and hardware) and separation of dependencies, the effectiveness of different optimization methods can be seen later in the development process or depend on other optimization measures, see Fig. 1.

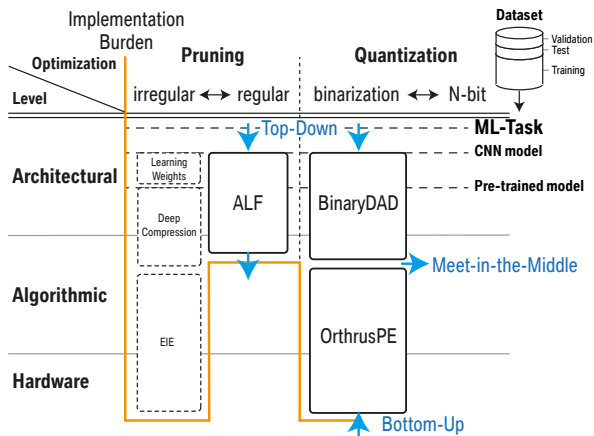


Fig. 1. Design process of hardware-aware performant perception of neural networks.

In the following we introduce two design processes: First, a simple top-down approach that enables lightweight models by means of structured pruning, which in turn can be applied to a variety of hardware accelerators using parallel libraries such as GEMM. Second, a meet-in-the-middle approach where binarization of the neural network has very good task related properties and a specially designed processing element favors its execution on an FPGA.

A. Top-Down Application of Structured Pruned CNNs

In this section we present a learning-based filter-wise pruning method, *i.e.* the autoencoder-based low rank filter sharing (ALF) technique [12]. By means of the method,

a simple top-down design process is leveraged. In detail, ALF makes use of the inherent structure of CNNs, maintains it throughout the compression, and results in an easy to implement representation. A conversion into a matrix form by using standard linear algebra libraries (e.g. BLAS) makes the model applicable to various off-the-shelf hardware accelerators. Furthermore, it can be observed that no algorithmic adjustments nor custom hardware are necessary.

The ALF approach is based on the publication [12] and provides the following contributions:

- Approximation of weight filters of convolutional layers using ALF-blocks, consisting of sparse autoencoders.
- A two player training scheme allows the model to learn the desired task while slimming the CNN.

The goal of the proposed filter-sharing technique is to replace the standard convolution with a more efficient alternative, namely the autoencoder-based low rank filter sharing-block.

Without loss of generality, $A^{l-1} \in \mathbb{R}^{H_i \times W_i \times C_i}$ is considered as an input feature map to a convolutional layer $l \in [1, L]$ of an L -layer CNN, where H_i and W_i indicate the height and width of the input, and C_i is the number of input channels. The weights $W^l \in \mathbb{R}^{K \times K \times C_i \times C_o}$ are the trainable parameters of the layer l , where K and C_o are the kernel dimensions and the number of output channels respectively.

In detail, the task is to approximate the filter bank W in a convolutional layer during training by a low-rank version $W_{\text{code}} \in \mathbb{R}^{K \times K \times C_i \times C_{\text{code}}}$, where $C_{\text{code}} < C_o$. The low-rank version of the weights W_{code} is utilized later in the deployment stage for an embedded-friendly application.

In contrast to previous structured pruning approaches [11], [5], [6], this method does not intend to alter the structure of the model in a way which results in a changed dimensionality of the output feature maps $A^l \in \mathbb{R}^{H_o \times W_o \times C_o}$, where H_o and W_o indicate the height and width of the output. This is done by introducing an additional expansion layer [45]. The advantages are twofold. First, each layer can be trained individually without affecting the other layers. Second, it simplifies the end-to-end training and allows comparison of the learned features.

The expansion layer is comprised of point-wise convolutions with weights $W_{\text{exp}} \in \mathbb{R}^{1 \times 1 \times C_{\text{code}} \times C_o}$, for mapping the intermediate feature maps after an ALF-block $\tilde{A}^l \in \mathbb{R}^{H_o \times W_o \times C_{\text{code}}}$, to the output feature map A^l , as expressed in Eq. 1.

$$A^l = \sigma(\tilde{A}^l * W_{\text{exp}}) = \sigma(\sigma_{\text{inter}}(A^{l-1} * W_{\text{code}}) * W_{\text{exp}}) \quad (1)$$

As the point-wise convolution introduces a certain overhead with regard to operations and weights, it is necessary to analyze the resource demands of the ALF-block compared to the standard convolution and ensure $C_{\text{code}} < C_{\text{code,max}}$, where $C_{\text{code,max}}$ denotes the number of filters which have to be removed to attain an efficiency improvement, see Eq. 2.

$$\frac{C_i C_o K^2}{C_{\text{code}}(C_i K^2 + C_o)} \rightarrow C_{\text{code,max}} = \lfloor \frac{C_i C_o K^2}{C_i K^2 + C_o} \rfloor \quad (2)$$

As stated before, the autoencoder is required to identify correlations in the original weights W and to derive a compressed representation W_{code} from them. The autoencoder is

only required in the training stage and is discarded in the deployment stage.

According to the design of an autoencoder, Eq. 3 gives the complete expression for calculating the compressed weights W_{code} . The encoder performs a matrix multiplication between the input W and the encoder filters $W_{\text{enc}} \in \mathbb{R}^{K \times K \times C_o \times C_{\text{code}}}$. M_{prune} zeroizes elements of \tilde{W}_{code} and σ_{ae} refers to a non-linear activation function, *i.e.* $\tanh(\cdot)$.

$$W_{\text{code}} = \sigma_{\text{ae}}(\tilde{W}_{\text{code}} \odot M_{\text{prune}}) = \sigma_{\text{ae}}((W \cdot W_{\text{enc}}) \odot M_{\text{prune}}) \quad (3)$$

Eq. 4 provides the corresponding formula for the reconstructed filters W_{rec} of the decoding stage. The symbol \cdot stands for a matrix multiplication and \odot for a Hadamard product respectively. The pruning mask M_{prune} acts as a gate, allowing only the most salient filters to appear as non-zero values in W_{code} , in the same manner as sparse autoencoders. The decoder must, therefore, learn to compensate for the zeroized filters to recover a close approximate of the input filter bank.

$$W_{\text{rec}} = \sigma_{\text{ae}}(W_{\text{code}} \cdot W_{\text{dec}}) \quad (4)$$

In order to dynamically select the most salient filters, an additional trainable parameter, denoted mask $M \in \mathbb{R}^{1 \times 1 \times 1 \times C_o}$, is introduced with its individual elements $m_i \in M$. By exploiting the sparsity-inducing property of L1 regularization, individual values in the mask M are driven towards zero during training. Since the optimizer usually reaches values close to zero, but not exactly zero, clipping is performed to zero out values that are below a certain threshold t . Further, the clipping function $M_{\text{prune}} = (M, t) = \mathbb{I}_{\{|m_i| > t\}} m_i$ allows the model to recover a channel when required.

Unlike other pruning approaches which require a pre-trained CNN and incorporate heuristics to determine the saliency of the weights, ALF dynamically prunes a given CNN during task-specific training. With minimal accuracy degradation, we reduce the number of training parameters of ResNet-20 by $3.9\times$ and operations by $2.6\times$ on CIFAR-10 dataset [46]. In summary, no pre-trained CNN is required for ALF-based pruning simplifying the compression. Additionally, as entire filters are removed, various parallel algorithms and hardware accelerators can be used in a subsequent deployment.

B. Meet-in-the-Middle Deployment of Binary Drivable Area Detection

The meet-in-the-middle design approach promotes the productivity of ML-engineers and HW-designers by allowing the expertise of both to meet at an optimal, tightly-coupled deployment. Deployment targets and constraints are allowed to *flow* in both directions of the design (top-down and bottom-up), leading to a better exchange in information among the expert groups. In this section, we show an example of this design methodology through binary DAD-Net [47] and OrthrusPE [17]. Binary DAD-Net provides efficient binary operations which tackle the two class, drivable area detection problem in an efficient and effective manner.

However, to maintain high accuracy, it requires some fixed-point operations, e.g. scale and shift operation. OrthrusPE provides an effective, high-throughput, low-power and low-utilization processing element, which can perform binary and fixed-point operations through run-time reconfigurability, fulfilling the requirements of binary DAD-Net. From the perspective of the HW-designer, binary DAD-Net inherently provides a lightweight CNN that can efficiently be deployed on FPGAs. Conversely, from the perspective of the ML-engineer, OrthrusPE offers the freedom of designing CNNs which have multiple types of operations (binary and fixed-point).

Top-Down: The binary DAD-Net approach provides the following contributions:

- A fully binarized drivable area detection neural network which has binary weights and activations in all parts of the model, *i.e.* encoder, bottleneck and decoder.
- The proposed binary model performs similar to the full-precision network gaining $14.3\times$ computational efficiency and $15.8\times$ memory saving for Cityscapes dataset [48] on the DAD task.
- The performance of binary DAD-Net is increased when pre-trained on automatic annotations.

The proposed drivable area detector is inspired by autoencoder-based networks with skip connections, *i.e.* DeepLabV3 [44]. As the name implies, binary DAD-Net has binary representations in all three parts of the model: the encoder, bottleneck (latent space) and decoder. Binary DAD-Net adopts the binarization scheme of Rastegari et al. [18]. As backbone (encoder) the 18 layered CNN ResNet18 is chosen, where the first convolutional layer is not binarized due to very few trainable parameters and computations compared to the remaining binary DAD-Net's layers. This aspect is one of many where the emphasis is on a processing element that is reconfigurable at run-time.

Furthermore, quantizing input image leads to high information loss leading to severe accuracy degradation. For the remaining layers, the sign-function binarizes the real-valued activations $H_{l-1} \approx \text{sign}(A_{l-1})$. In the inference-stage the weights are considered to $B \approx \text{sign}(W) \in \{-1, +1\}$. Scale factors α and β , introduced in [18], find better estimations for $W \approx \alpha B$ and $A \approx \beta H$, see Eq. 5. The convolution between B^l and H^{l-1} can be computed using *xnor-popcount* operation.

$$A_l = \text{Conv}(W_S^l, f_S^{l-1}) \approx \alpha\beta \text{Conv}(B^l, H^{l-1}) \quad (5)$$

A typical binary convolution block consists of 1) binarization of the activations and weights, 2) binary convolution, 3) Batch Normalization and 4) non-linear activations such as ReLU. The residual block, introduced by He et al. [49], can be easily binarized learning more complex features by adding consecutive binary convolutional layers with Batch Normalization and non-linear activation function along with fused shortcut connections. The shortcut connections in binary residual blocks favor the BNNs by overcoming the gradient saturation problem. Inspired by DeepLabv3 [50], we use dilated convolutions in the bottleneck to increase the

receptive field of the respective convolutions to increase the receptive field of a convolutional layer, dilated convolution introduces zeros to the weights of the respective layer. Our observation for dilated convolution fits previous investigations for vanilla binary convolution layer [18], [19], [20].

The central part of the DeepLab [50] inspired binary DAD-Net is the bottleneck. In detail, the bottleneck consists of two consecutive binary residual blocks and a binary atrous spatial pyramid pooling (ASPP)-block. The dilation rate $d = 2$ is used for the last residual block. For the parallel convolutions in ASPP block, four different dilation rates $\{1, 8, 12, 18\}$ are assigned. Different to previous residual blocks, the dilated residual blocks do not downsample the feature maps. Thus, the feature resolution of the binary bottleneck is efficiently increased. Upsampling by a factor of 16 instead of 32 is required.

The decoder of binary DAD-Net is binarized for the task of drivable area detection. Employing only binary convolutions enlarges the output of the bottleneck to the size of the original input image I generating pixel-wise predictions for the task of drivable area detection. The binary decoder also consists of bilinear upsampling and a binary score layer. In detail, after the binary dilated convolution, described in the previous section, linear combination (binary 1×1 convolution) of the ASPP feature maps and the encoder skip connection (after the first residual block) is computed. Next, the feature maps are fused in two consecutive binary refinement blocks. The binary refinement blocks consist of 3×3 kernels, which is similar to the binary convolutional layer, described above. Instead of transpose convolutions, bilinear up-sampling enlarges the feature maps to the size of the input I . This is important as the binary transpose convolution would introduce additional operations and would lead to an accuracy degradation.

We introduce normalized compute complexity (NCC), allowing an implementation-wise comparison, by determining optimal utilization of fixed-point and binary operations in one compute unit. The reference implementation of OrthrusPE using DSP-48 block is used to compute NCC. In particular, OrthrusPE enables of perform two 16-bit fixed-point multiplications or 48 XNOR operations at once. Binary DAD-Net achieves 96.23% mean intersection over union (mIOU) using 0.9MB of training parameters on the CityScapes dataset. By incorporating the automatic annotations of drivable area using Train Data Generator [51], we increase the mIOU to 96.60%. Moreover, binary DAD-Net shows its superior performance w.r.t. an embedded implementation, by drastically reducing the NCC ($20.4\times$) compared to the full precision implementation of DeepLabv3 [44].

Bottom-Up: The OrthrusPE design provides the following contributions:

- A flexible computation unit for accelerating a wide range of BNNs.
- Execution of SIMD-based binary Hadamard product on FPGA hard blocks.
- A run-time reconfigurable processing element which dynamically supports binary and fixed-point computations.

OrthrusPE is a run-time reconfigurable processing element (PE) which can satisfy all the functions required by accurate BNNs, while capitalizing on resource reuse. Accurate BNNs cannot be achieved without fixed-point operations and reliance on DSP blocks. Instead of separating binary and fixed-point computations to two types of hardware resources, OrthrusPE improves the efficiency of the computation by executing both on FPGA hard blocks. OrthrusPE and OrthrusPE-DS (Dual-Static) were evaluated across multiple target accelerator frequencies. Both solutions achieved improved resource utilization *and* power efficiency compared to typical BNN accelerator processing elements. OrthrusPE presents a well-suited processing element to compute the operations of binary DAD-Net, which involve binary Hadamard products, as well as fixed-point operations in the form of scaling factors, necessary for the XNOR-Net binarization in binary DAD-Net. Accurate BNNs solve many of the computation and memory challenges for deep neural network workloads on edge devices. Efficiently executing their mixed-precision computations can further exploit the advantages they offer at the hardware level.

IV. CONCLUSION

Knowledge of the design process for the deployment of CNNs on embedded ECUs, such as in robotics or autonomous driving, helps to establish lightweight applications. We show the design process by means of two prominent optimization methods, namely pruning and quantization. In the first methodology, channel-wise pruning is conducted to a CNN allowing it to be deployed with minimal implementation overhead on various hardware accelerators (preventing lock-ins). Here, structured learning-based pruning outperforms irregular handcrafted pruning in terms of applicability of the compressed model and facilitates the design process. Taking the autoencoder-based low rank filter-sharing technique into consideration, CNNs are compressed. Its' compressed variants can easily be deployed with off-the-shelf algorithms and hardware accelerators. In the second method, we show the use of a novel binary drivable area detection neural network together with a run-time reconfigurable processing element. In the design phase of the binary neural network we evaluate different local and structural binarization methods. A thorough selection of methods and composition of the CNN architecture brings the BNN close to the accuracy of its full-precision counterpart. For a concurrent bottom-up design, the BNN's requirements from the top-down design are made accessible. This allows the targeted conception of a reconfigurable processing element. All in all, the meet-in-the-middle design shows very fruitful results, where synergies between HW and lightweight CNN are made. In summary, we show that a successful deployment requires data scientists to prepare the training data, ML experts to design the CNN, programmers to develop high-performance algorithms and hardware specialists to provide the target accelerator. The specific expertise of the individual participants makes it necessary to abstract the process of optimizing CNNs.

REFERENCES

- [1] A. Frickenstein, C. Unger, and W. Stechele. Resource-Aware Optimization of DNNs for Embedded Applications. In *CRV*, 2019.
- [2] Song Han, Jeff Pool, John Tran, et al. Learning both weights and connections for efficient neural networks. In *NeurIPS*, 2015.
- [3] S. Han, X. Liu, H. Mao, et al. EIE: Efficient inference engine on compressed deep neural network. In *ISCA*, 2016.
- [4] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally. Snn: An accelerator for compressed-sparse convolutional neural networks. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 27–40, 2017.
- [5] Alexander Frickenstein, Manoj Rohit Vemparala, Christian Unger, et al. DSC: Dense-sparse convolution for vectorized inference of cnns. In *CVPR-W*, 2019.
- [6] Yang He, Ping Liu, Ziwei Wang, et al. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 2019.
- [7] Q. Huang, K. Zhou, S. You, et al. Learning to prune filters in convolutional neural networks. In *WACV*, 2018.
- [8] Yihui He, Ji Lin, Zhijian Liu, et al. AMC: Automl for model compression and acceleration on mobile devices. In *ECCV*, 2018.
- [9] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient DNNs. In *NeurIPS*, 2016.
- [10] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, et al. Structadmm: A systematic, high-efficiency framework of structured weight pruning for dnns. 2018.
- [11] Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. LCNN: Lookup-based convolutional neural network. In *CVPR*, 2017.
- [12] Alexander Frickenstein, Manoj-Rohit Vemparala, Nael Fasfous, Laura Hauenschield, Naveen-Shankar Nagaraja, Christian Unger, and Walter Stechele. ALF: Autoencoder-based low-rank filter-sharing for efficient convolutional neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, June 2020.
- [13] Mingxing Tan, Bo Chen, Ruoming Pang, et al. MnasNet: Platform-aware neural architecture search for mobile. *arXiv:1807.11626*.
- [14] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.
- [15] Sebastian Vogel, Mengyu Liang, Andre Guntoro, et al. Efficient hardware acceleration of CNNs using logarithmic data representation with arbitrary log-base. In *ICCAD*, 2018.
- [16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, et al. Binarized Neural Networks. In *NeurIPS*, 2016.
- [17] Nael Fasfous, Manoj-Rohit Vemparala, Alexander Frickenstein, et al. Orthruspe: Runtime reconfigurable processing elements for binary neural networks. In *DATE*, 2020.
- [18] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, et al. Xnor. 2018.
- [19] Wei N. Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.
- [20] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 345–353. Curran Associates, Inc., 2017.
- [21] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, NIPS’15, pages 1135–1143, Cambridge, MA, USA, 2015. MIT Press.
- [22] Siyang Sun, Yingjie Yin, Xingang Wang, De Xu, Wenqi Wu, and Qingyi Gu. Fast object detection based on binary deep convolution neural networks. *CAAI Trans. Intell. Technol.*, 3:191–197, 2018.
- [23] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Structured binary neural networks for accurate image classification and semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 413–422, June 2019.
- [24] H. Sharma et al. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks. In *ISCA*, 2018.
- [25] J. Lee et al. Unpu: An energy-efficient deep neural network accelerator with fully variable weight bit precision. *IEEE J. Solid-State Circuits*, Jan. 2019.
- [26] P. Judd et al. Stripes: Bit-serial deep neural network computing. In *MICRO*, Oct. 2016.
- [27] S. Sharify et al. Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks. In *DAC*, 2018.
- [28] R. Andri et al. Yodann: An architecture for ultralow power binary-weight cnn acceleration. *IEEE TCAD*, Jan. 2018.
- [29] K. Ando et al. Brein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 tops at 0.6 w. *IEEE J. of Solid-State Circuits*, Apr. 2018.
- [30] Y. Umuroglu et al. Finn: A framework for fast, scalable binarized neural network inference. In *FPGA*, 2017.
- [31] Intel Corp. *Lower Numerical Precision Deep Learning Inference and Training White Paper*, 1 2018.
- [32] Nvidia Corp. *Nvidia Turing GPU Architecture White Paper*, 2018.
- [33] Apple Inc. *Reducing the Size of Your Core ML App*, accessed April 11, 2020.
- [34] Xilinx, Inc. *Versal: The First Adaptive Compute Acceleration Platform (ACAP)*, 10 2018. v1.0.
- [35] Philip Colangelo, Nasibeh Nasiri, Asit Mishra, Eriko Nurvitadhi, Martin Margala, and Kevin Nealis. Exploration of low numeric precision deep learning inference using intel fpgas, 2018.
- [36] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4107–4115. Curran Associates, Inc., 2016.
- [37] R. Zhao et al. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *FPGA*, 2017.
- [38] L. Yang, Z He, and D. Fan. A fully onchip binarized convolutional neural network fpga impelmentation with accurate inference. In *ISLPED*, 2018.
- [39] S. Liang et al. Fp-bnn: Binarized neural network on fpga. *Neurocomputing*, 275, 2018.
- [40] D. Nguyen, D. Kim, and J. Lee. Double mac: Doubling the performance of convolutional neural networks on modern fpgas. In *DATE*, Mar. 2017.
- [41] E. Nurvitadhi et al. Accelerating binarized neural networks: Comparison of fpga, cpu, gpu, and asic. In *FPT*, Dec. 2016.
- [42] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, April 2017.
- [43] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.
- [44] Liang-Chieh Chen, Yukun Zhu, George Papandreou, et al. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.
- [45] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, et al. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size. *arXiv:1602.07360*, 2016.
- [46] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).
- [47] Alexander Frickenstein, Manoj-Rohit Vemparala, Jakob Mayr, et al. Binary DAD-Net: Binarized driveable area detection network for autonomous driving. In *ICRA*, 2020.
- [48] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [49] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016.
- [50] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 833–851, Cham, 2018. Springer International Publishing.
- [51] Jakob Mayr, Christian Unger, and Federico Tombari. Self-Supervised Learning of the Drivable Area for Autonomous Vehicles. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 362–369. IEEE, 01.10.2018 - 05.10.2018.



Security in Automotive

Ralf Neuhaus : Automotive System Architect EMEA



© Copyright 2020 Xilinx

Agenda

- >Automotive Trends & Security Standards
- >Secure Boot for Automotive Applications
- >Security Features for Automotive Applications
- >Safety and Security
- >Summary

Security is important!

www.heise.de/news from 22.Sep. 2020

Danger for cars: "Everything that is networked will also be attacked".

moving tests

regarding this PSA should be directed to your local **FBI Field Office.**

Local Field Office Locations:
www.fbi.gov/contact-us/field

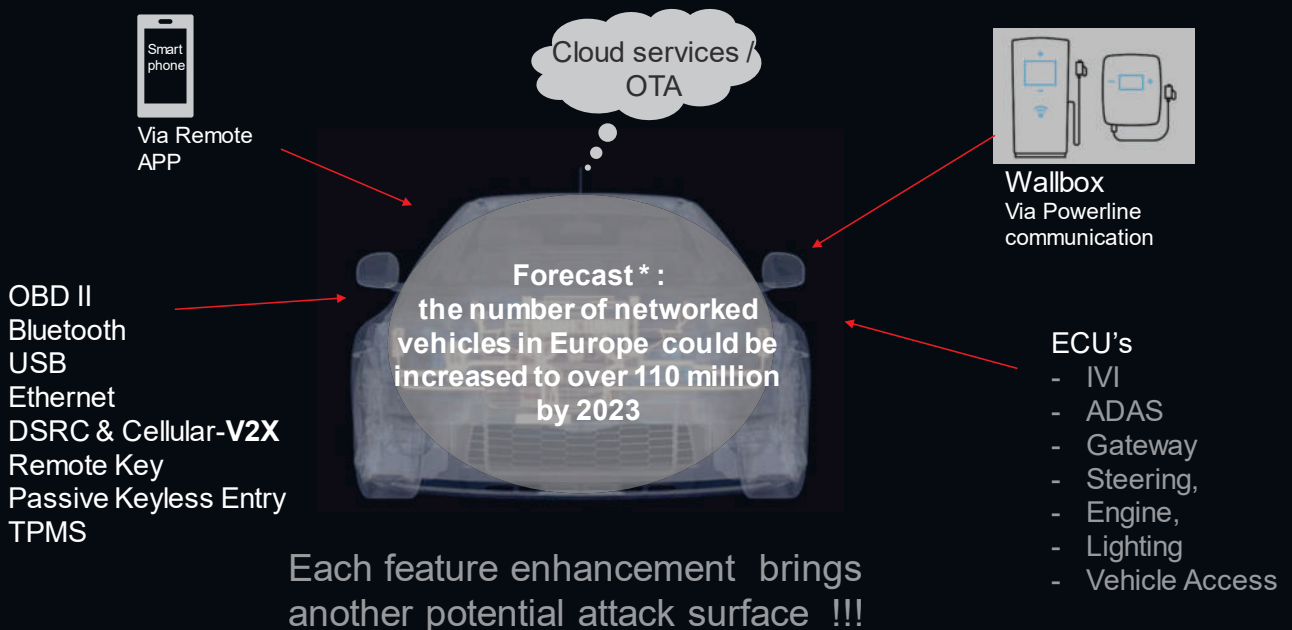
As previously, researchers evaluating remote exploits of motor vehicles could gain significant control over vehicle wireless communications vulnerabilities. While the issues have been addressed, it is important that consumers and manufacturers be aware of the possible threats and how an attacker may seek to remotely exploit vulnerabilities in the future. Third party aftermarket devices with Internet or cellular access plugged into diagnostics ports could also introduce wireless vulnerabilities.



>> 3

© Copyright 2020 Xilinx

Potential attack surfaces of a connected car



>> 4

*IT consulting company Capgemini

© Copyright 2020 Xilinx



Trends in Automotive

- > Moving towards Hardware Root of Trust
- > Growing desire for every network connected device to support OTA
- > Products that will have some unique and evolving security requirements
 - >> Vehicle Gateways and many ECUs
 - Needs for authentication and/or encryption on CAN and Ethernet messaging between ECUs
 - Enhanced AUTOSAR + crypto extensions e.g. with HSM
 - >> Domain Controllers
 - Hypervisors or several secure operating systems may be used to provide another layer of security
 - Need for additional protection our XMPU and XPPU
 - >> Event Recorders
 - Secure storage on encrypted video as a method to protect user data
 - Need for high bandwidth AES accelerator
 - >> C-V2X
 - Large number of key exchanges
 - Need for HSM low latency and high throughput



>> 5

© Copyright 2020 Xilinx



Industry Focus

Source: SAE J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems; Appendix H; Jan 2016

- > Lack of an industry standard creates challenges
- > Xilinx's primary focus is on
 - >> Auto-ISAC
 - >> ISO21434
 - >> HSM Architecture (e.g. EVITA)



> Others?



Project	Timeline	Origin
EVITA	2008-2011	EU
PRECIOSA	2008-2010	EU
SeVeCOM	2006-2010	EU
DRIVE-C2X	2011-2014	EU
PRESERVE	2011-2014	EU
OVERSEE	2010-2012	EU
EURO-MILS	2012-2015	EU
SESAMO	2012-2015	EU
SHIELDS	2008-2010	EU
CVIS	2006-2010	EU
TECOM	2008-2011	EU
SEPIA	2010-2013	EU
NoW	2004-2008	Germany
ARAMiS	2011-2014	Germany
HEAVENS	2013-2016	Sweden

>> 6

© Copyright 2020 Xilinx



Automotive Groups & Standards

> Auto-ISAC (Information Sharing & Analysis Center)

> ISO21434 Road vehicles – Cybersecurity engineering

>> Requirements around development process

- Cybersecurity management
- Risk management & assessment
- Product development
- Verification & validation
- Operations & maintenance

>> Latest <https://www.sae.org/standards/content/iso/sae21434.d1/> (from 202002)

> SAE J3101 – Requirements for Hardware-Protected Security for Ground Vehicle Applications

>> Requirement for security implemented in hardware

>> Latest https://www.sae.org/standards/content/j3101_202002/

> SAE J3061 – Cybersecurity Guidebook for Cyber-Physical Vehicle Systems



>> 7

© Copyright 2020 Xilinx



ISO21434 Details

> Process for OEMs, Tier 1 and Tier 2 Suppliers



> Contains requirements around

- >> Cybersecurity Management
- >> Project Dependent Cybersecurity Management
- >> Risk Assessment
- >> Concept Phase
- >> Development Phase
- >> Production, Operations, Maintenance and Decommissioning
- >> Management Systems
- >> Distributed activities (i.e. between OEM and Suppliers or between Suppliers)

*DIA – Development Interface Agreement

>> 8

© Copyright 2020 Xilinx



Agenda

- > Automotive Trends & Security Standards
- > Secure Boot for Automotive Applications
- > Security Features for Automotive Applications
- > Safety and Security
- > Summary

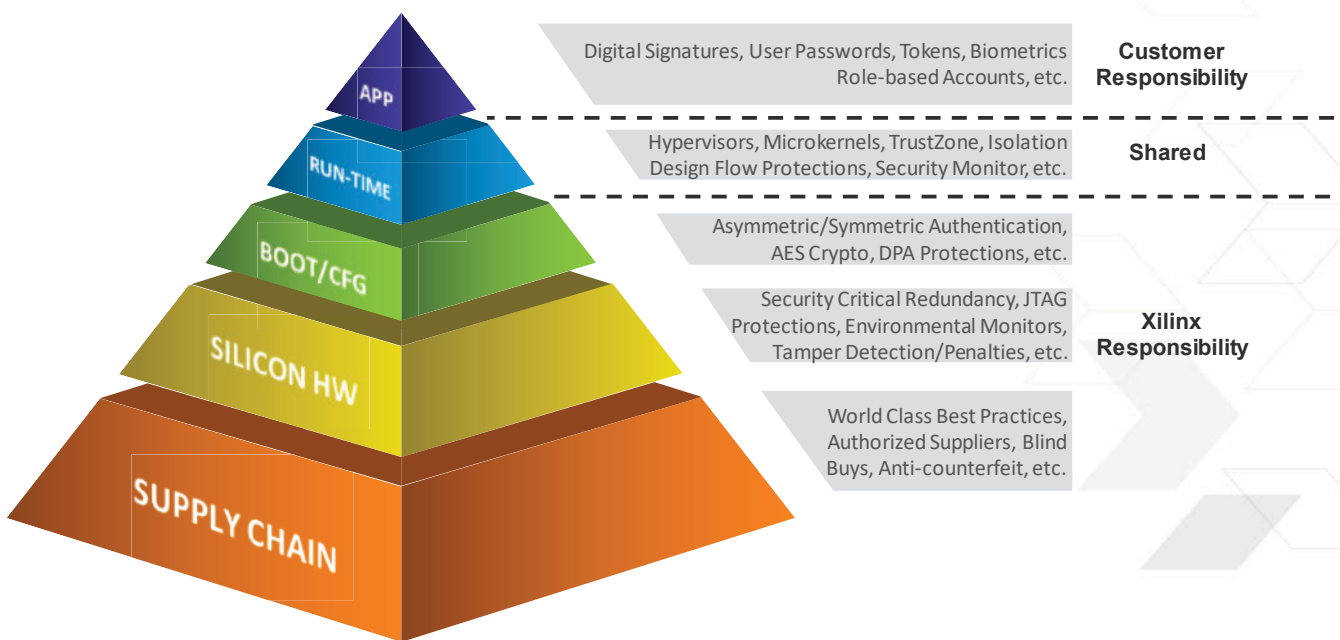


>> 9

© Copyright 2020 Xilinx



Security through Product/System Lifecycle



© Copyright 2020 Xilinx



Two Secure Boot Modes in ZU+

	HW Root of Trust	Encrypt-Only
Asymmetric Authentication	Yes w/ RSA-4096	No
Confidentiality	Optional w/ AES-GCM (256 bit key)	Required w/ AES-GCM ¹ (256 bit key)
Symmetric Authentication	Optional w/AES-GCM (256 bit key)	Required w/ AES-GCM ¹ (256 bit key)
Boot Time ⁴	Longer	Shorter
Differential Power Analysis (DPA) Protection	Yes	No ²
Physical Unclonable Function (PUF) Support for Black Key Storage	Yes ³	No
Key Revocation/Anti-Replay	Yes	No
RMA Support	No	Yes

1. ALL partitions must be encrypted
2. DPA Protection requires RSA and AES
3. 128bit entropy PUF (SCD#4687) is supported in XA devices

4. See Boot Time Estimator (<https://www.xilinx.com/support/answers/67475.html>)

>> 11

© Copyright 2020 Xilinx



Secure Boot and Operation Protections (ZU+)

Attack	Device Countermeasure
Side Channel	Built-In Differential Power Analysis Countermeasures and/or Protocol (Authentication and Key Rolling)
Fault Injection	PMC Triple Redundant Processors and ECC on PMC Memories; Temporal and Physical Redundancy in HW and ROM Code; SHA Integrity Checks on Immutable ROM Code
Physical	PMC Triple Redundant Processors and ECC on PMC Memories; Temporal and Physical Redundancy in HW and ROM Code; SHA Integrity Checks on Immutable ROM Code
Environmental	ECC on PMC Memories; SHA Integrity Checks on Immutable ROM Code
Test / Debug	By Design (disabled upon power up and fault tolerant); JTAG Monitoring; Permanent disable capability
General	Immutable ROM Code; PMC clocked by internal, uninterruptable clock source; Pre-boot: sensitive info is the device key – protected via PUF (eFUSE)

>> 12

© Copyright 2020 Xilinx



Agenda

- > Automotive Trends & Security Standards
- > Secure Boot for Automotive Applications
- > Security Features for Automotive Applications
- > Safety and Security
- > Summary



>> 13

© Copyright 2020 Xilinx



Simplified Per-Device Unique Keying

<https://github.com/Xilinx/bootgen>

- > Enabled by open source **Bootgen** running on **A53/A72**
 - >> Facilitates unique keying architectures that require unique boot images per device



Traditional Provisioning

- Prebuilt boot image
- PUF KEK is unique per device
- One boot image encryption key
- Single signature across devices

Advanced Provisioning

- Boot image encrypted & signed “on-the-fly”
- PUF KEK is unique per device
- Supports unique encryption keys per device
- Unique signature per device



>> 14

*See XSWG2019 Provisioning / Secure Provisioning Service presentation

© Copyright 2020 Xilinx



ZU+ Fielded System Test/Debug

- > **What test capability do you have for a fielded system?**
 - >> Enabling Secure Boot protects test interfaces
 - >> JTAG is automatically protected w/Secure Boot Enabled
 - You do not have to program the JTAG Disable eFUSE to protect the device



>> 16

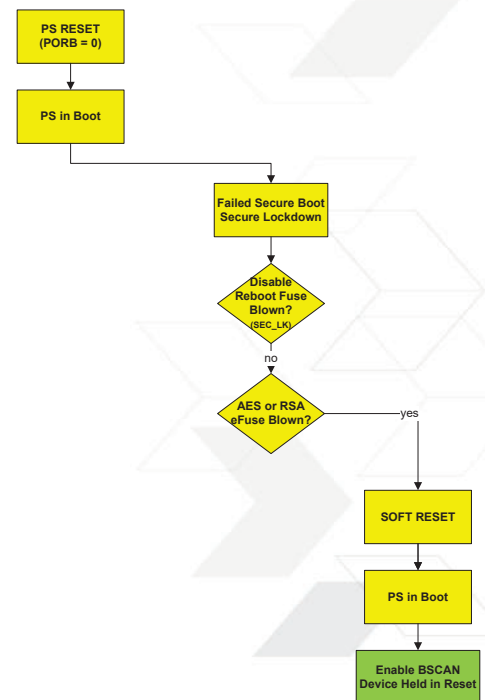
© Copyright 2020 Xilinx

References from ZU+ TRM v1.8 (UG1085)



ZU+ Fielded System Test/Debug

- > **What test capability do you have for a fielded system?**
 - >> Enabling Secure Boot protects test interfaces
 - >> JTAG is automatically protected w/Secure Boot Enabled
 - You do not have to program the JTAG Disable eFUSE to protect the device
- > **Test Capabilities**
 - >> Boundary Scan/Connectivity testing when secure boot fails
 - Automatically enabled if you do not program SEC_LK eFUSE



>> 17

© Copyright 2020 Xilinx

References from ZU+ TRM v1.8 (UG1085)



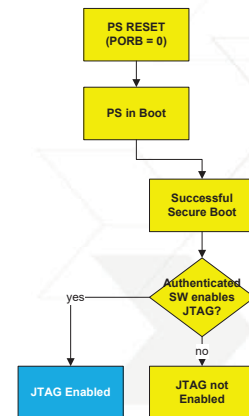
ZU+ Fielded System Test/Debug

> What test capability do you have for a fielded system?

- >> Enabling Secure Boot protects test interfaces
- >> JTAG is automatically protected w/Secure Boot Enabled
 - You do not have to program the JTAG Disable eFUSE to protect the device

> Test Capabilities

- >> Boundary Scan/Connectivity testing when secure boot fails
 - Automatically enabled if you do not program SEC_LK eFUSE
- >> Full test capability via JTAG when secure boot passes
 - Load an FSBL that enables JTAG



>> 18

© Copyright 2020 Xilinx

References from ZU+ TRM v1.8 (UG1085)



ZU+ Fielded System Test/Debug

> What test capability do you have for a fielded system?

- >> Enabling Secure Boot protects test interfaces
- >> JTAG is automatically protected w/Secure Boot Enabled
 - You do not have to program the JTAG Disable eFUSE to protect the device

> Test Capabilities

- >> Boundary Scan/Connectivity testing when secure boot fails
 - Automatically enabled if you do not program SEC_LK eFUSE
- >> Full test capability via JTAG when secure boot passes
 - Load an FSBL that enables JTAG

> What is not covered?

- >> Failure during boot / boot logic; Status available via JTAG
 - PS TAP Controller
 - JTAG Error Status
- >> Do NOT program the JTAG Disable eFUSE

Table 6-13: JTAG Error Register Description

Error source	Bit on JTAG Error Status
CSU ROM error (same as bit 120).	0
PMU pre-boot error (same as bit 78).	1
PMU ROM service error (same as bit 99).	2
PMU firmware error (same as bits 103:100).	6:3
Uncorrectable PMU error. Includes ROM validation, TMR, uncorrectable RAM ECC, and local register address errors.	7
CSU error.	8
PLL lock errors [VideoPLL, DDRPLL, APULL, RPULL, IOPLL].	13:9
PL generic errors passed to PS.	17:14
Full-power subsystem time-out error.	18
Low-power subsystem time-out error.	19
Reserved errors.	24:20
Clock monitor error.	25
XPMU errors [LPD XMPU, FPD XMPU].	27:26

>> 19

© Copyright 2020 Xilinx

References from ZU+ TRM v1.8 (UG1085)



ZU+ Fielded System Test/Debug

> What test capability do you have for a fielded system?

- >> Enabling Secure Boot protects test interfaces
- >> JTAG is automatically protected w/Secure Boot Enabled
 - You do not have to program the JTAG Disable eFUSE to protect the device

> Test Capabilities

- >> Boundary Scan/Connectivity testing when secure boot fails
 - Automatically enabled if you do not program SEC_LK eFUSE
- >> Full test capability via JTAG when secure boot passes
 - Load an FSBL that enables JTAG

> What is not covered?

- >> Failure during boot / boot logic; Status available via JTAG
 - PS TAP Controller
 - JTAG Error Status
- >> Do NOT program the JTAG Disable eFUSE

Table 39-6: PS TAP Controller Status Register

Bit	Value	Description
31-28	PS_VERSION	Indicates the PS version, same as csu.version [ps_version] register bit
27-20	Reserved	Ignore
19	Reserved	Ignore
18	Reserved	Reads 0
17-14	BOOT_MODE	Device boot mode
13	CBR_DONE	Configuration BootROM (CBR) has finished running and the full JTAG instruction set is available
12	SCAN_CLEAR_FAILED	Pre-boot SCAN CLEAR function failed
11	LBIST_FAILED	Pre-boot LBIST function failed
10	BISR_FAILED	Pre-boot BISR function failed
9	PL_PWR_STS	Power status of the PL, cannot connect to the PL TAP if this bit is 0
8-7	Reserved	Ignore
6	PMU_MDM_SEC_GATE	PMU MDM security gate is disabled
5	PL_TAP_SEC_GATE	PL TAP security gate is disabled
4	ARM_DAP_SEC_GATE	Arm DAP security gate is disabled
3	ARM_DAP	Arm DAP is connected in the JTAG chain
2	PL_TAP	PL TAP is connected in the JTAG chain
1	0	
0	1	

>> 20

© Copyright 2020 Xilinx

References from ZU+ TRM v1.8 (UG1085)



How to do an RMA in ZU+ and Versal

ZU+

> Encrypt Only boot mode

- field factory
- >> Step 1 – Provision device for Encrypt Only boot mode
 - Provision ENC_ONLY eFUSE and AES eFUSE key
 - >> Step 2 – Deploy system
 - >> Step 3 – When device fails, encrypt Xilinx RMA boot loader with your symmetric key
 - Xilinx boot loader opens test interfaces
 - >> Step 4 – Return device and encrypted boot load to Xilinx using approved process

> HWRoT boot mode is NOT supported

Versal

- > Encrypt Only boot mode support
- > HWRoT boot mode support

>> 21

© Copyright 2020 Xilinx



EVITA HSM Architecture

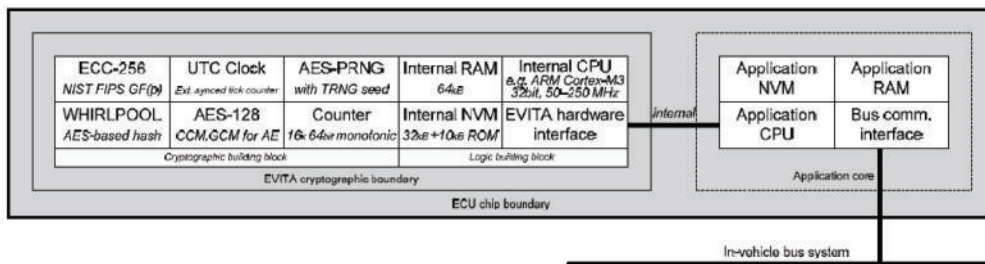


Figure 7: EVITA full hardware security module (V2X level)

> Security Requirements

- >> SR.1 Integrity/Authenticity of e-Safety related events
- >> SR.2 Integrity/Authenticity of ECU/firmware installation/configuration
- >> SR.3 Secure execution environment
- >> SR.4 Vehicular Access Control
- >> SR.5 Trusted In-Vehicle ECU Platform. Integrity/Authenticity of operated software
- >> SR.6 Secure in-vehicle data storage
- >> SR.7 Confidentiality of in-vehicle and external communications
- >> SR.8 Privacy
- >> FR.9 Interference of security functionality
 - Layman's definition – security cannot negatively impact system availability

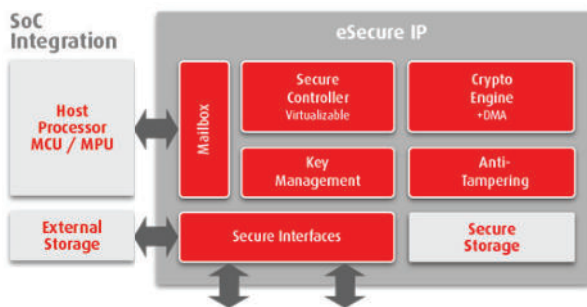
>> 24

© Copyright 2020 Xilinx



Pre-Engineered Solution

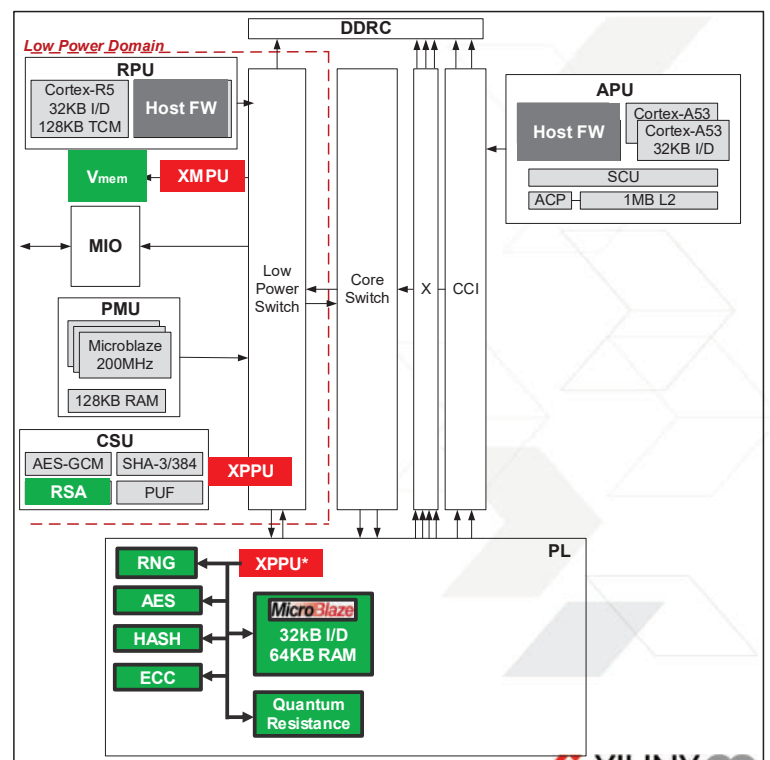
> Silex Insight HSM eSecure IP



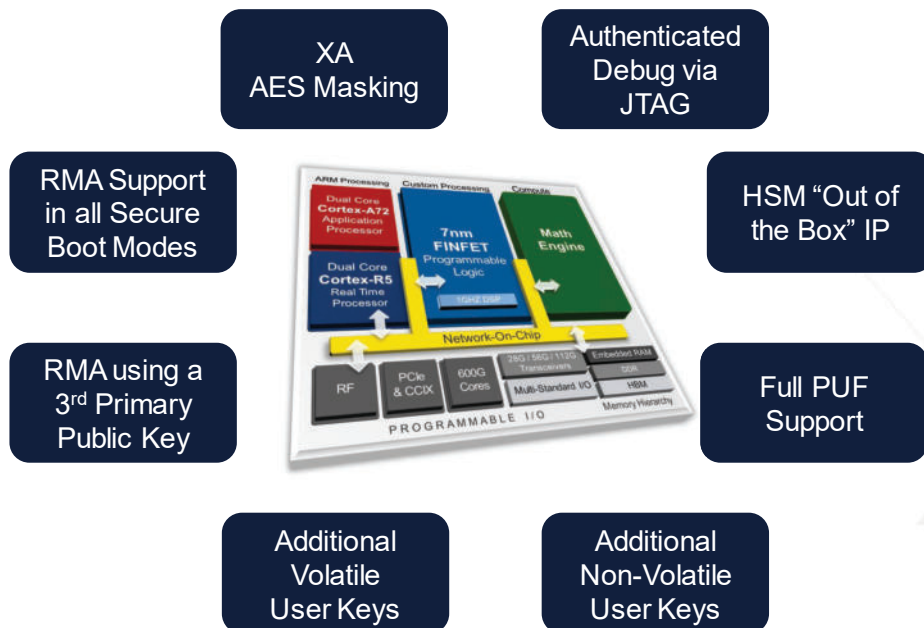
- >> Secure operations in the PL
 - Keys/Secure Processing isolated from PS
- >> Host FW executes on RPU or APU
 - Interface with AUTOSAR API
- >> Flexible: Tradeoff features, logic and performance
- >> See XSWG2019 ZU+ Hardware Security Module (HSM) IP Solution presentation

>> 25

© Copyright 2020 Xilinx



Automotive Security Enhancements in next gen products



>> 26

© Copyright 2020 Xilinx



Agenda

- > Automotive Trends & Security Standards
- > Secure Boot for Automotive Applications
- > Security Features for Automotive Applications
- > Safety and Security
- > Summary

>> 27

© Copyright 2020 Xilinx



Safety Collateral



>> 28

© Copyright 2020 Xilinx

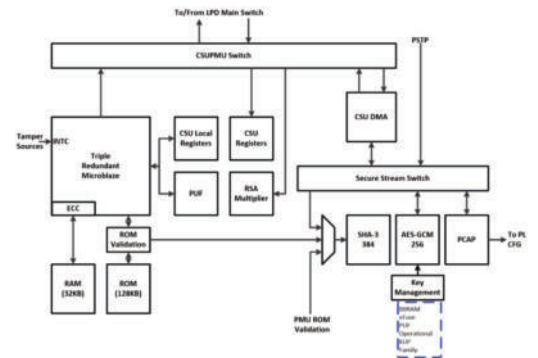


ZU+ Automotive Developments

> CSU Usage in FuSa application

>> Previous version of ZU+ Safety Manual (UG1226) prevented use of hardened crypto accelerators in CSU

>> **Not good for safety applications requiring security !!!**



[SMA_NS RM_003] The following modules contained in the LPD can be used to initialize, service, and support the application but shall never be used during the time when the application is actively performing the safety mission:

- 1) AXI Trace Monitor (ATM)
- 2) AXI Performance Monitor (APM)
- 3) Real Time Clock (RTC)
- 4) JTAG
- 5) ARM debug access port (ARM DAP)
- 6) Crypto Interface Block (CIB)

>> 29

© Copyright 2020 Xilinx



ZU+ Automotive Developments

> CSU Usage in FuSa application

- >> Previous version of ZU+ Safety Manual (UG1226) prevented use of hardened crypto accelerators in CSU
- >> New version of ZU+ Safety Manual defines assumptions of use for using hardened crypto accelerators in CSU
- >> **Much better for security applications!!!**

[SMA_CSU_001] The CSU shall be clocked by the SysOsc clock.

[SMA_CSU_002] The CSU Crypto Interface Block (CIB) contains RSA, SHA3, AES-GCM, PUF, PCAP, and CSU DMA functions. When using these CIB functions, all safety-related communication dependent on these blocks, shall use end-to-end protection measures.

>> 30

© Copyright 2020 Xilinx



Safety Collateral

> [Functional Safety Lounge \(link\)](#)

- >> The Functional Safety Lounge is home to much of Xilinx Functional Safety Collateral
- >> Contact your FAE for access details

> Xilinx Functional Safety Working Group (FSWG)

- >> FSWG is for safety what XSWG is for security
- >> Two day working group
- >> 2020 will be its 4th year

Xilinx Functional Safety Working Group (FSWG)
2020 Virtual Event
Registration to be opened starting 1st of October

>> 31

© Copyright 2020 Xilinx



Agenda

- > Automotive Trends & Security Standards
- > Secure Boot for Automotive Applications
- > Security Features for Automotive Applications
- > Safety and Security
- > Summary

>> 32

© Copyright 2020 Xilinx



Summary

- > **ZU+ security can address the cybersecurity Automotive needs today**
 - >> Multiple Secure Boot modes
 - >> Support for secure external non-volatile memory
 - >> Run-time security through hardened accelerators and tamper monitoring
- > **Versal security continues automotive security with additional enhancements**
- > **Hardware Security Module (HSM) IP provides canned Security Solution**
- > **Safety requires security!!**
 - >> Xilinx's security solutions complement our functional safety solutions

Information on older families can be found in prior year's XSWG presentations

>> 33

© Copyright 2020 Xilinx





Workshop : Programmable Processing for the Autonomous / Connected Vehicle

THANK YOU



© Copyright 2020 Xilinx