BOSCH DAIMLER dSPACE ika INSTITUT FÜR KRAFTFAHRZEUGE RWTH AACHEN UNIVERSITY IMS Hannover infineon

Vorhabensbezeichnung:
Verbundprojekt DESERVE: DEvelopment platform for Safe and Efficient dRiVE
EU-Grant-Nr.:295364

Laufzeit des Vorhabens:
01.09.2012 – 29.02.2016

Zuwendungsempfänger:	**Förderkennzeichen:**
Robert Bosch GmbH	01|S12002A
Daimler AG	01|S12002B
Infineon Technologies AG	01IS12002C
dSPACE GmbH	01|S12002D
Institut für Kraftfahrzeuge (ika)	01|S12002E
Institut für Mikrosystemtechnik (IMS) Hannover	01|S12002F

Projektträger:
Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) | Fr. Tai Ding und Fr. Juliane Jacobi

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Inhalt

1 Allgemeiner Teil

1.1 Aufgabenstellung

Das ECSEL Projekt DESERVE hat sich zum Ziel gesetzt, eine durchgängige und vereinheitlichte Entwicklungsplattform für Fahrerassistenzsysteme zu entwickeln, die durch eine innovative Partitionierung und Modularisierung die bisher nur rudimentär existierenden Einzellösungen im Bereich der Umgebungsperzeption sowohl bezüglich der Entwicklungszeiten als auch bei der Wiederverwendbarkeit von bereits implementierten Komponenten und Subsystemen zu optimieren.

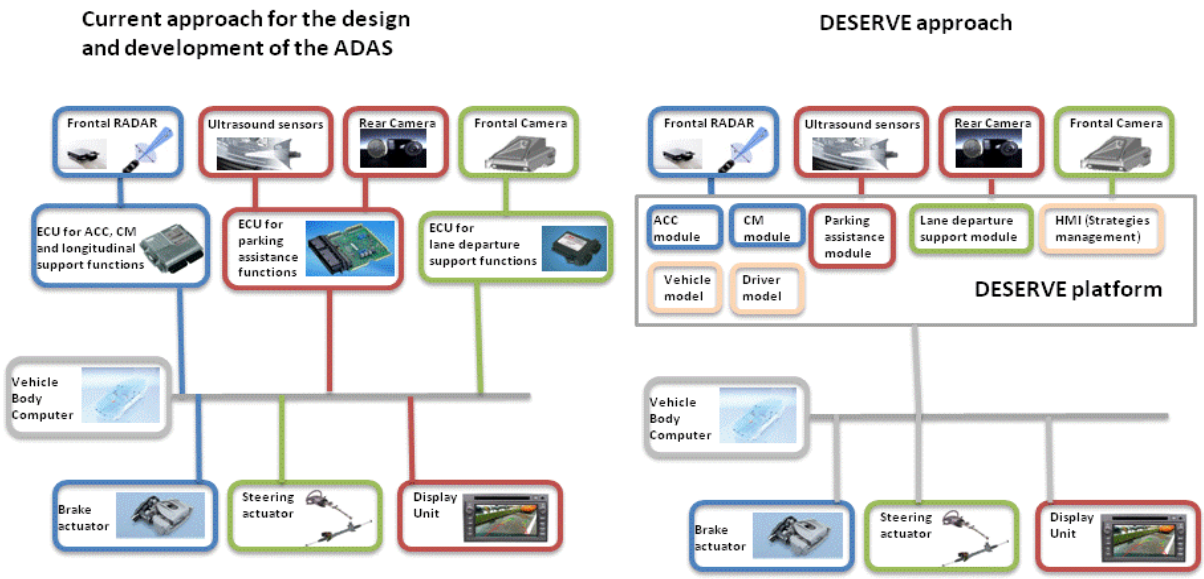


Abbildung 1: Der generelle DESERVE Plattform-Konzeptansatz

In Abbildung 1 ist das grundlegende Konzept der DESERVE Plattform dargestellt. Durch eine Modularisierung der entsprechenden Verarbeitungskomponenten kann der Informationsfluss untereinander maximiert und die für die verschieden Fahraufgaben notwendige Rechenkapazität minimiert werden. Dieses modulare Plattformkonzept zieht sich in DESERVE von der Perzeption über die Planung bis hin zur Aktorik durch, wie in Abbildung 2 dargestellt.

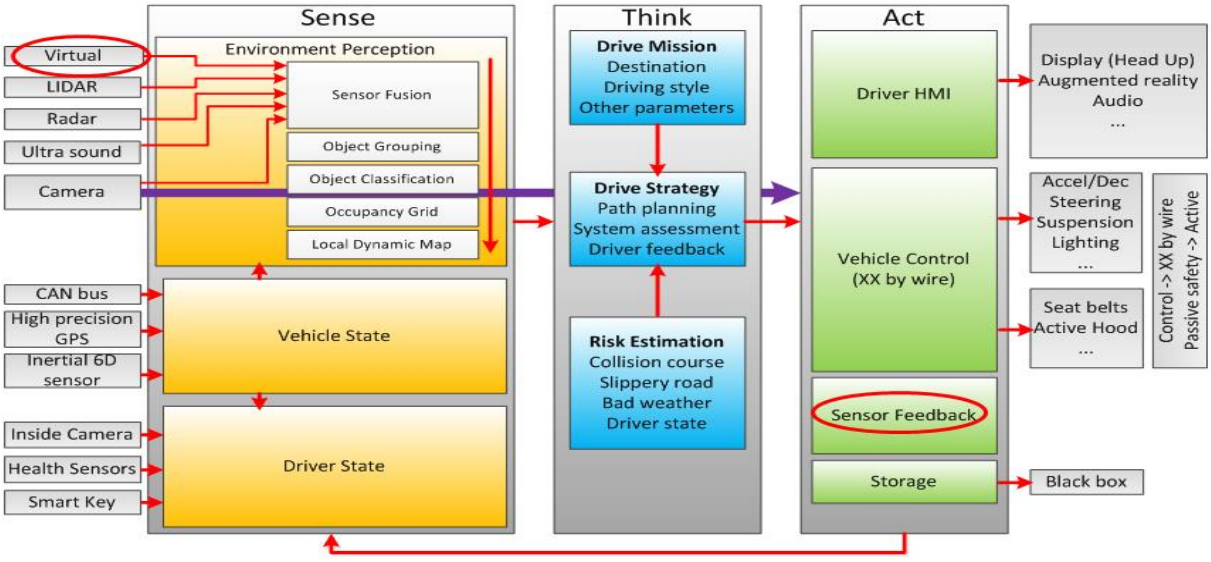


Abbildung 2: Das DESERVE Multi-Plattformkonzept

Die Aufteilung in Perzeptions-, Applikations- und Kontrollplattform wird in DESERVE von allen Europäischen Projektpartnern konsequent angewandt und auf modularer Sub-Ebene funktionsspezifisch für die verschiedenen Implementierungs-Demonstratoren weiterentwickelt und verfeinert.

Das Deutsche DESERVE Konsortium hat sich dabei als Anwendungsfall den Inter-Urban Assist (in Deutsch: Landstraßenassistent) vorgenommen, um die DESERVE Plattform-spezifischen Entwicklungen beispielhaft umsetzen zu können.
Durch die Modularisierung und Vereinheitlichung der Schnittstellen kann sowohl die stetig ansteigende Komplexität der einzelnen ADAS Funktionen beherrscht werden als auch die Entwicklungszeiten zwischen den einzelnen Produktgenerationen signifikant verkürzt werden (ca. 10 bis 15% als Zielwert).

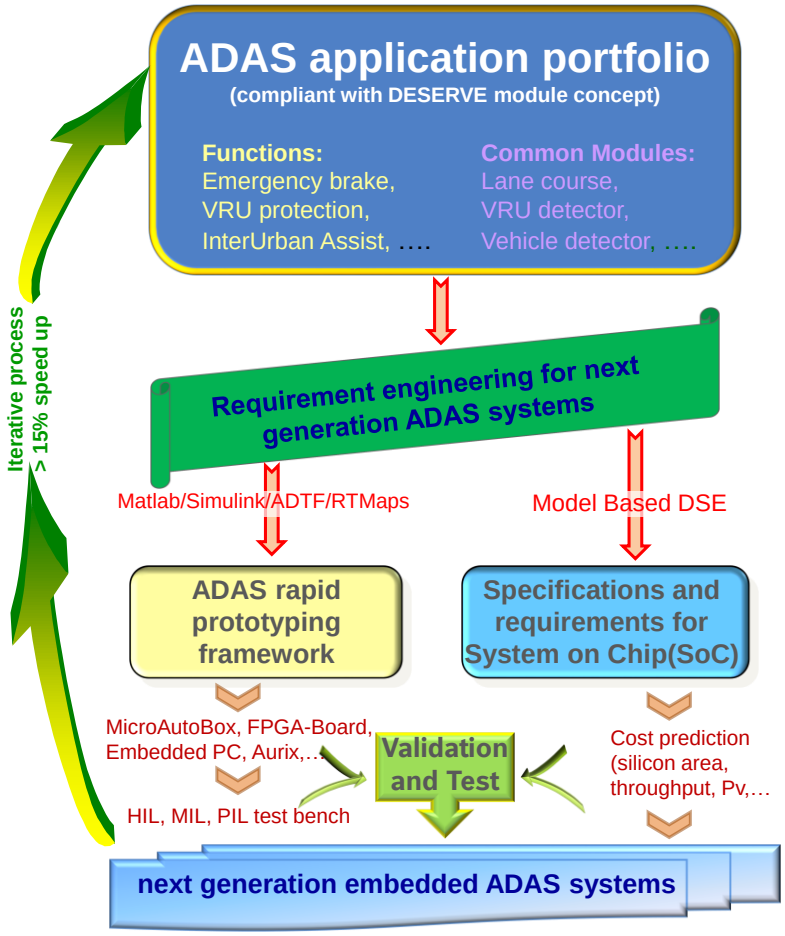


Abbildung 3: Der DESERVE Plattformgedanke – Schnellere ADAS Generationsentwicklung

Abbildung 3 zeigt die Grundgedanken und den prinzipiellen Ablauf des DESERVE Plattformkonzepts bei der Mehrgenerationen-Produktentwicklung. Durch gleichzeitiges Entwickeln und Spezifizieren im DESERVE Entwicklungsrahmen sowohl auf der Prototypen- als auch auf der Serienschiene wird von Anfang an eine höhere Effizienz und wesentlich kürzere Entwicklungszeit erreicht.
Bereits bei der Anforderungsanalyse für die zukünftige ADAS Systeme werden parallel zur Prototypenentwicklung auch schon die grundlegenden Spezifikationen für die letztendliche Serienplattform mit einbezogen, was sich direkt oder auch nur indirekt bereits auf die Prototypenentwicklung auswirkt und das Nachbessern oder sogar Neukonzeptionen für die spätere Serienentwicklung auf ein Minimum reduziert. Damit werden Fehlentwicklungen oder „geschwulst-ähnliche“ Anhängsel bei den Serienprodukten, wie sie hauptsächlich bei verspäteten Nachbesserungen und umständliche Hilfslösungen (sogenannten Work-Arounds) anzutreffen sind, best-möglichst verhindert. Der dabei angewandte Bewertungsansatz, basierend auf der Design Space Exploration (DSE – entwickelt und veröffentlicht von der Universität Hannover, IMS Institut) verhindert das im frühen Entwicklungsprozess oft chaotische und planerisch ineffiziente „Hin- und Her Geschiebe“ zwischen der funktionalen Algorithmik und der hardwaremäßigen Implementierung. Vielmehr ist über entsprechende Modelle und Kostenfunktionen, die aktuell jedoch nur sehr rudimentär und auf oberster Ebene für einige wenige ADAS Funktionen existieren, bereits zu Beginn der Algorithmen Entwicklung genau erkennbar, welcher Hardwareaufwand und andere Leistungsindikatoren notwendig sind, um die entsprechenden Algorithmen später in Serienprodukte umzusetzen.
Das Grundprinzip der DSE ist in Abbildung 4 skizziert.

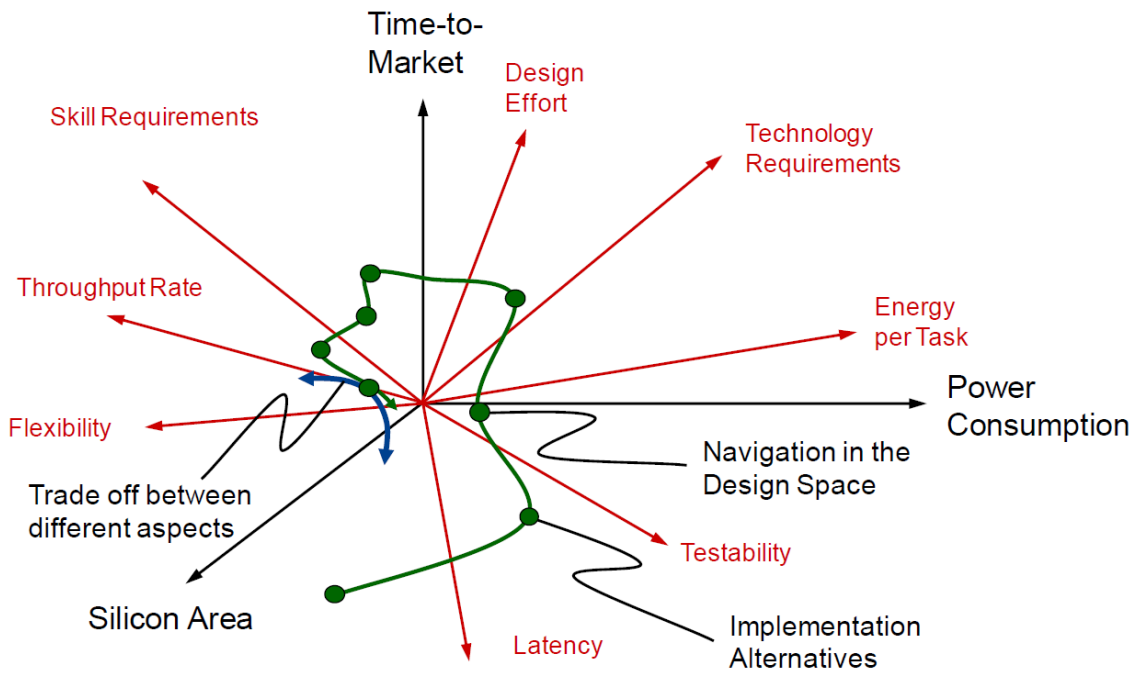


Abbildung 4 : Design Space Exploration Prinzip

Letztendlich ist bei konsequenter Anwendung des DSE Prinzips eine globale, mehrdimensionale Optimierung über die wichtigsten Leistungsfaktoren und entscheidenden Entwicklungsparameter bei der Produktentwicklung möglich und führt abhängig von den als wichtig angesehenen Optimierungskriterien zur effizientesten, leistungsfähigsten, kostengünstigsten und implementierungsfreundlichsten Lösung.
Diese anspruchsvolle und neuartige Methodik ist der inhaltliche Grundgedanke und stellt den Kern der Aufgabenstellung für das Verbundprojekt DESERVE dar.

1.2 Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Die inhaltlichen Voraussetzungen zur Projektdurchführung waren durch die breit aufgestellten Vertreter aus Industrie und Forschung gegeben. Die Robert Bosch GmbH hatte die Rolle des nationalen Koordinators übernommen. Zusammen mit den anderen Projektpartnern waren alle wichtigen Mitwirkenden entlang der Produktions- und Wertschöpfungskette präsent, vom Chip- und Technologie Lieferant Infineon AG bis zum Endkunden, dem OEM Daimler AG.
Die Forschungsschwerpunkte lagen dabei in der Definition und Entwicklung eines Plattformkonzeptes, das Fahrerassistenzsysteme bis hin zum teil-automatisierten Fahren schnell, effizient und ohne den typischen Bruch im Entwicklungsablauf zwischen ersten Prototyp und finalem Produkt realisieren kann.

1.3 Planung und Ablauf des Vorhabens

Die Projektplanung und Durchführung fand in enger Absprache und Zusammenarbeit mit allen Projektpartnern statt. In den ca. vierteljährlichen Deutschen Abstimmungstreffen wurde der Projektfortschritt und die weitere strategische Ausrichtung und Zielsetzung unter den Partnern besprochen und abgestimmt.
Das Projekt wurde dazu in verschiedene Arbeitspakete mit entsprechender Resourcenhinterlegung aufgesetzt, wie in Abbildung 5 ersichtlich.

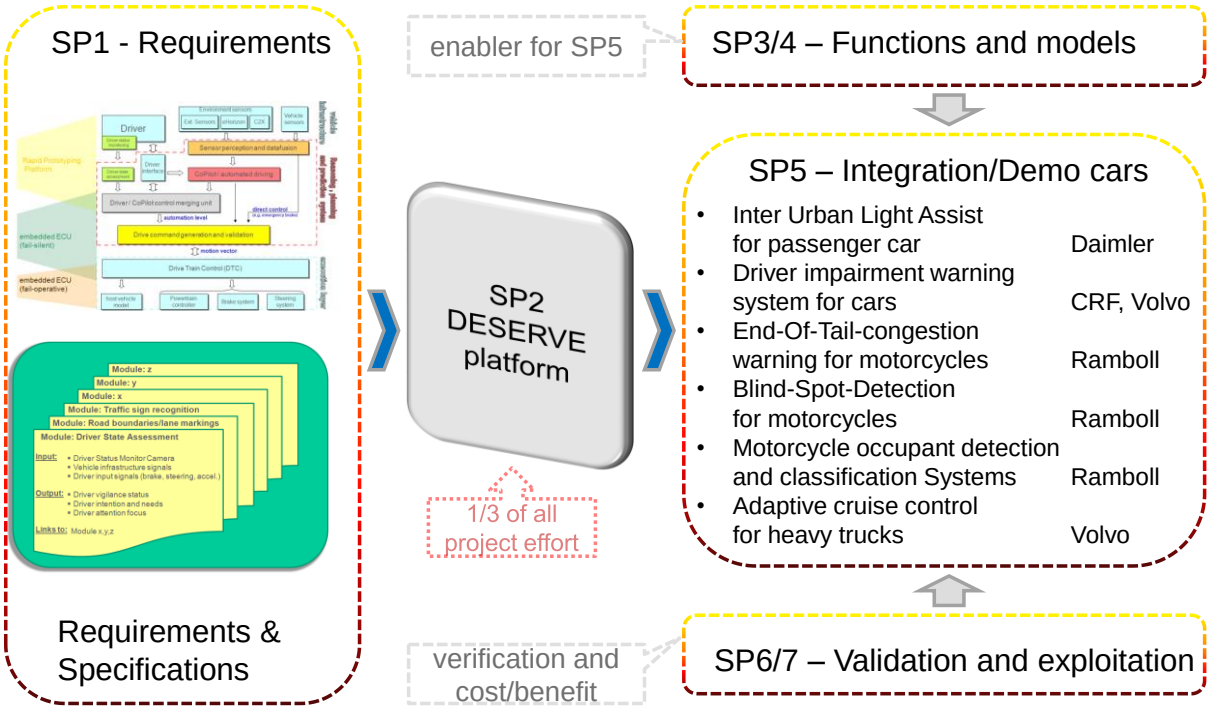


Abbildung 5: DESERVE Arbeitsplan auf SP Ebene

1.4 Wissenschaftlicher und technischer Stand - Ausgangssituation

Das DESERVE Projekt beschreitet mit seinem innovativen Ansatz bei der Produktentwicklung von Fahrerassistenzsystemen neue Wege und versucht erstmals, die einzelnen Entwicklungsschritte etwas besser zu modularisieren und zu standardisieren. Damit ist sowohl eine effizienterer und kürzere Produktentwicklung möglich als auch eine bessere Wiederverwendbarkeit bereits entwickelter Module und Komponenten. Der bisherige Ansatz, jede Funktion der Fahrerassistenz als „stand-alone“ Einzelsystem zu betrachten, wurde aufgegeben und vielmehr nach Gemeinsamkeiten gesucht, um den benötigten Hardware- und Software so gering wie nur möglich zu halten.
Während der Projektlaufzeit wurde der aktuelle Stand der Technik kontinuierlich weiterbeobachtet und inhaltlich verfolgt.

1.5 Wissenschaftlicher und technischer Stand - verwendete Fachliteratur

Für die wissenschaftlichen Arbeiten wurden, soweit vorhanden, die zugänglichen Literaturquellen herangezogen und sachlich objektiv bewertet. Ähnlich wie bei Daimler konnten sich auch Bosch und Infineon auf umfassende Berichte und Informationen aus den hauseigenen Forschungsabteilungen stützen. Einige dieser Unterlagen unterliegen der Geheimhaltungspflicht und wurden im Projekt dementsprechend auch vertraulich behandelt. Der technische Stand bei der Entwicklung von komplexen Fahrerassistenzsystemen auf Modulbasis unter Ausnutzung von Synergieeffekten war zu Projektstart nur sehr rudimentär vorhanden. Einschlägige Veröffentlichungen und Artikel zu dem DESERVE Thema gab es keine. Auch die verfügbare Fachliteratur hielt sich in überschaubaren Größen.

1.6 Zusammenarbeit mit anderen Stellen

Die Zusammenarbeit zwischen den Projektpartnern war harmonisch und in allen Belangen gut abgestimmt. Kooperationen mit projektfremden Partnern auf System- und Funktionsebene hielten sich in engen Grenzen, da die Thematik sehr speziell und vor allem auch neu war und daher auf Förderprojektebene keine geeigneten Partnerprojekte identifiziert werden konnten. Die Zusammenarbeit mit Fremdzulieferern (z.B. für die Robert Bosch GmbH speziell mit dem FPGA Spezialisten Xilinx) gestaltete sich unter den üblichen Rahmenbedingungen und Geheimhaltungsvereinbarungen durchgehend positiv und konstruktiv.

2 Eingehende Darstellung

2.1 Verwendung der Zuwendung

Die Zuwendung wurde im Wesentlichen zur teilweisen Deckung der in DESERVE entstandenen Personal- und Sachkostenaufwendungen eingesetzt. Die F&E Personalkosten und alle anderen geplanten Kostenpositionen beliefen sich im geplanten Kostenrahmen ohne größere Abweichungen und Verschiebungen innerhalb der einzelnen Positionen.

Die Reisekosten fielen wegen der lokalen Nähe der Projektpartner teilweise etwas unter Plan aus. Bei den Material- und Sachkosten wurde durch vorausschauende Planung der benötigten und eingesetzten Ressourcen der geplante Budgetrahmen sehr effizient ausgeschöpft.
Die Unterschreitungen bei einigen Partnern in den verschiedenen Positionsarten wurden durch entsprechende Beantragung von Mittelumwidmungen kompensiert.

2.2 Wichtigsten Positionen des zahlenmäßigen Nachweises

Bitte hierzu die entsprechenden Erläuterungsblätter im Anhang zu diesem Bericht heranziehen.

2.3 Notwendigkeit und Angemessenheit der geleisteten Arbeiten

Die Notwendigkeit und Angemessenheit der geleisteten Arbeiten reflektieren sich bereits an den erzielten Ergebnissen. Der Aufwand und Einsatz an Personal und Sachmitteln entsprach den projektspezifischen Notwendigkeiten, um die geplanten Projektziele und Ergebnisse auch erreichen zu können. Die geleisteten Arbeiten waren weder über- noch unterdimensioniert, sondern standen im Einklang mit den geplanten und letztendlich auch umgesetzten Projektzielen. Die beantragten Fördermittel der deutschen Projektpartner wurden daher auch bis auf kleine Abweichungen wie geplant abgerufen.

2.4 Voraussichtlicher Nutzen und Verwertbarkeit der Ergebnisse

Die in DESERVE erarbeiteten Ergebnisse werden kontinuierlich in den kommenden Jahren nach und nach in die entsprechenden Produktgenerationen für Fahrerassistenzsysteme und automatisierte Fahrfunktionen bei den Industriepartnern einfließen. Bei den universitären Einrichtungen bildet sich die Ergebnisverwertung in Form von Kompetenz- und Know-How – Aufbau ab.
Eine detailliertere Darstellung für die einzelnen Projektpartner kann in den Erfolgskontrollberichten der Projektpartner nachgelesen werden.

2.5 Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen

Zum Projektstart und während der Laufzeit von DESERVE wurden die Fortschritte auf den Forschungsschwerpunkten in regelmäßigen Abständen durch entsprechende Recherchen überprüft. Es gab keinerlei Hinweise auf Veröffentlichungen von anderen Stellen und daher auch keine Möglichkeit, sich mit Externen zu dem DESERVE Thema auszutauschen oder zu diskutieren.

2.6 Erfolgte oder geplante Veröffentlichung der Ergebnisse

Die im Zusammenhang mit dem DESERVE Verbundprojekt entstandenen Veröffentlichungen und Vorträge wurden in der Abschlussveranstaltung am 17. Dezember 2015 bei der Daimler AG in Ulm präsentiert.

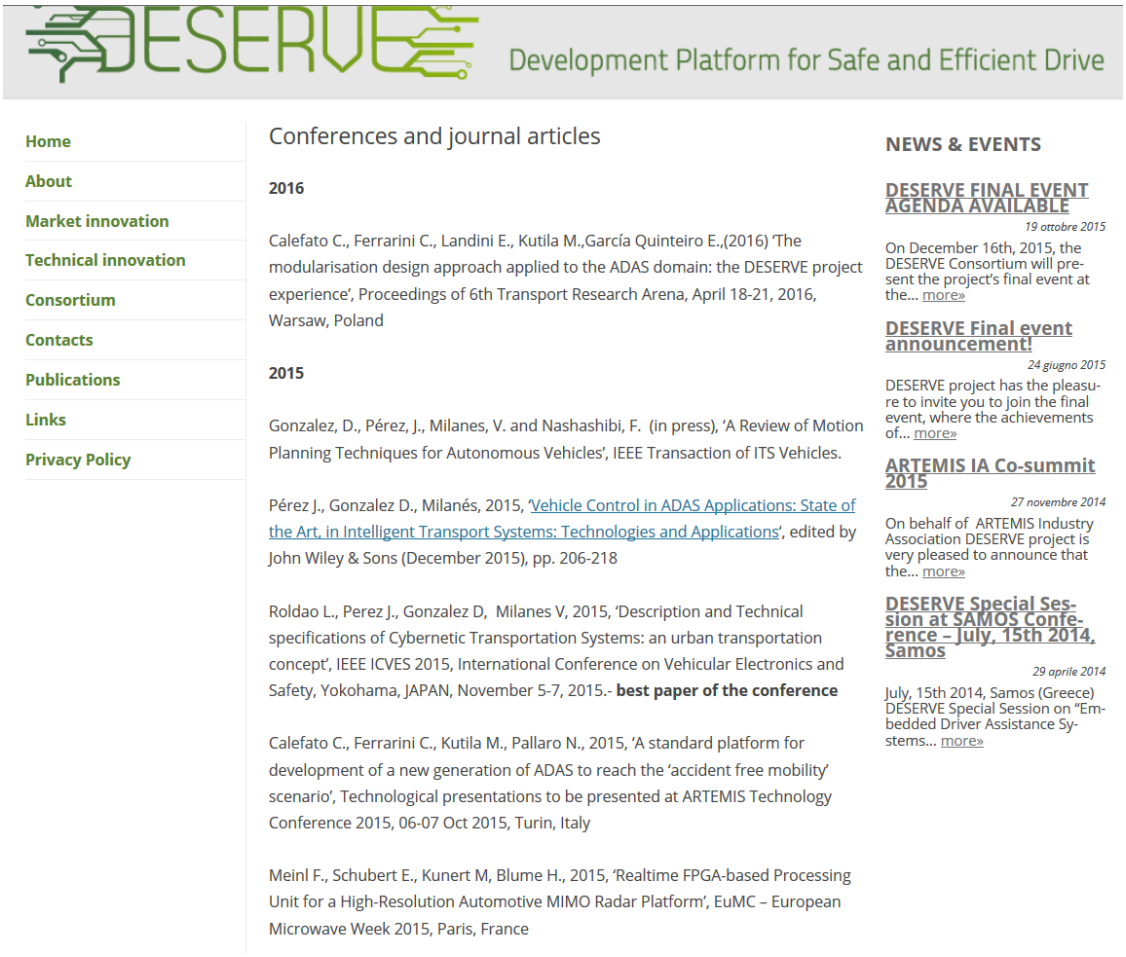


Abbildung 6: DESERVE Webseite mit Dissemination Informationen

Eine Referenz aller DESERVE Veröffentlichungen ist sowohl auf der DESERVE Webseite (siehe Abbildung 6) als auch im Quellennachweis unter [Ref 17] zu finden.

3 Fachlicher Schlussbericht

3.1 Einführung

In den nachfolgenden Abschnitten wird ein Überblick der im Verbundprojekt DESERVE von den Deutschen Partnern erzielten Arbeitsergebnisse gegeben. Die Berichterstattungstiefe ist aufgrund des limitierten Umfanges auf das Wesentliche und die essentiellen Ergebnisse reduziert. Auf viele detailliertere Aspekte, die im Rahmen der Projektarbeiten aufgekommen sind, kann daher nicht weiter eingegangen werden. Eine detailliertere Ergebnisbeschreibung kann den zusätzlichen Informationsquellen im Literaturverzeichnis von Abschnitt 6 entnommen werden.

3.2 Allgemeine Projektziele

Die allgemeinen Projektziele von DESERVE können wie folgt zusammengefasst werden:

1. Definition und Implementierung eines modell-getriebenen Entwicklungsprozesses mit einfachen Anknüpfpunkten für bereits existierende Komponenten und Funktionen;
2. Die Entwicklung einer innovativen, eingebetteten Fahrzeugplattform für ein schnelles und zuverlässiges Fahrerassistenz-Entwicklungssystem;
3. Die Integration von bereits vorhandenen Fahrzeugsensoren und Aktoren in einen vereinheitlichten Software Entwicklungsrahmen;
4. Die Anpassung von bestehenden Datenfusionsalgorithmen, Mensch-Maschine Schnittstellen und Fahrerverhaltensmodellen;
5. Die Implementierung von neuen Methoden und Konzepten für die Entwicklung von Fahrerassistenzfunktionen.

Zusammenfassend: Eine Tool-Plattform und Landschaft für die kompositionelle Entwicklung von zukünftigen ADAS Systemen bis hin zum automatisierten Fahren. Die treibenden Gedanken und die Motivation des DESERVE Plattformkonzeptes manifestieren sich in der Darstellung von Abbildung 7.

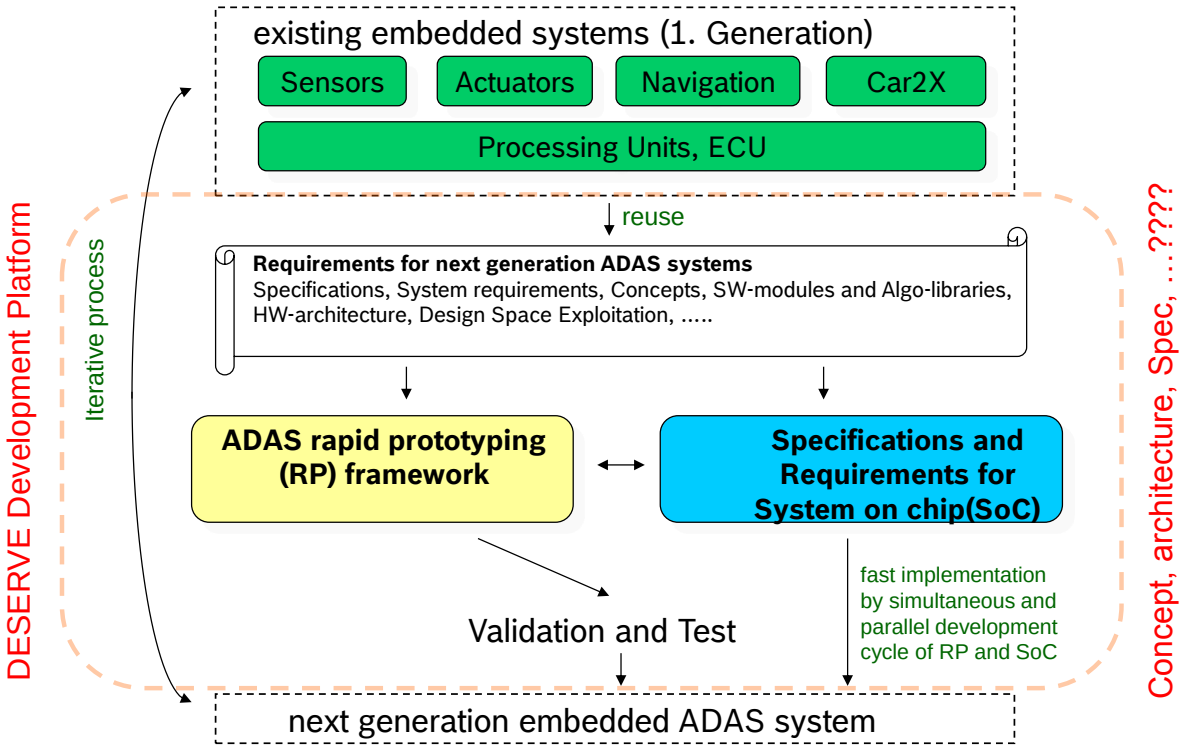


Abbildung 7: Die DESERVE Projektidee

3.3 Beitrag IMS - Methoden der Entwurfsraum-Exploration

Eine geeignete Software/Hardware-Implementierung eines Systems zu identifizieren ist in der Regel ein sehr zeitaufwändiger Prozess. Der Applikations-Designer hat eine Auswahl zwischen verschiedenen Algorithmen zu treffen und muss diese darüber hinaus applikationsspezifisch parametrisieren. Die Evaluation des Hardware-Mappings des betrachteten Algorithmus lässt darauf schließen, ob die Hardware den Anforderungen gerecht wird. Zusätzlich geben die ausgewerteten Implementierungs-Daten dem Entwickler Hinweise bezüglich Änderungen von algorithmischen Parametern, um noch effizientere Hardware-Realisierungen herauszuarbeiten.
Zum Untersuchen verschiedener Realisierungen und zum Finden der besten möglichen Implementierung mit vorgegebenen Ressourcen ist für eine schnelle Evaluation unterschiedlicher Hardware-Plattformen eine Entwurfsraum-Exploration notwendig. Die Evaluation von Hardware-Plattformen bezüglich der Umsetzung der betrachteten Algorithmen und das Zusammentragen der daraus resultierenden Hardware-Kosten ist ein langwieriger Prozess. Um die Entwurfsraum-Exploration durchführbar zu gestalten, lassen sich daher Kostenmodelle anwenden.
Typischerweise existieren diverse mögliche Mappings einer Applikation für unterschiedliche Hardware-Architekturen. In Anbetracht unterschiedlicher Hardware-Charakteristiken und Technologien ist die Möglichkeit einer Entwurfsraum-Exploration unbedingt erforderlich, um in diesem multidimensionalen Entwurfsraum eine Realisierung zu finden, die den vorgegebenen Bedingungen entspricht.

3.3.1 Die Exploration des modell-basierten Entwurfraumes

Der Entwurfsraum ist typischerweise über mehrere Dimensionen aufgespannt, z.B. Implementierungskosten, Leistungsaufnahme, Siliziumfläche, etc. Es ist daher nur bedingt möglich, den gesamten Entwurfsraum durch Implementierung oder Simulation zu Evaluationszwecken abzudecken, da der Aufwand ein vernünftiges Zeitmaß deutlich übersteigt und die Simulation komplexer Hardware enorm zeitaufwändig ist.
Eine Kosten-Evaluation zuvor betrachteter Systeme und eine Entwurfsraum-Exploration auf Basis von quantitativen Kostenmodellen unterstützen den Entwickler, indem eine Exploration des vorliegenden Entwurfsraumes ohne Detailwissen über die zur Verfügung stehenden Hardware-Technologien möglich wird. Die Analyse des betrachteten Systems in frühen Design-Stadien und das fundierte Untermauern notwendiger Entscheidungen im Entwurfsprozess werden somit durchführbar.
Unterschiedliche Hardware-Plattformen bringen eigene Randbedingungen mit sich, die bei der Umsetzung des Algorithmus unter Umständen besondere Beachtung benötigen und völlig unterschiedliche Ansätze mit sich bringen. Im Vergleich der Modelle von z.B. programmierbaren Systeme und dedizierter Hardware sind darüber hinaus architektur-spezifische Eigenschaften zu berücksichtigen.
Zur Modellierung eines programmierbaren Multicore-Prozessors gehören unter anderem die Verteilung der Arbeitslast zwischen den Prozessorkernen, Datenlokalität, Cache-Zugriffe und –Kohärenz, oder die Granularität des Parallelisierungslevel, um das oftmals nicht-deterministische Laufzeitverhalten eines Multicore-Prozessorsystems zu modellieren.
Dem gegenüber steht beispielsweise eine dedizierte Hardware, die ein parametrisiertes Kostenmodell benötigt, um wesentliche algorithmische Parameter zu extrahieren, welche signifikante Auswirkungen auf den späteren Ressourcenaufwand haben. Zusätzlich zu einem Laufzeitmodell sind quantitative Modelle für Lookup-Tabellen, benötigter Block-RAM oder verwendete DSPs des späteren Designs möglich.

3.3.1.1 Modellierung der Entwurfskosten

Das zentrale Element der Modell-basierten Entwurfsraum-Exploration ist die Modell-Bibliothek (siehe Abbildung 8). In dieser Bibliothek sind diverse quantitative Kostenmodelle hinterlegt, welche über die sog. intrinsischen und extrinsischen Anforderungen parametrisiert sind.
Intrinsische Anforderungen, d.h. algorithmische und architektonische Anforderungen, sind Eigenschaften, die durch die Anwendung selbst vorgegeben sind. Hierbei sind algorithmische Randbedingungen beispielsweise Freiheitsgrade eines Parametersatzes, um den Algorithmus an eine Anwendung anzupassen, z.B. Bildgröße, Schwellwerte, Filtereinstellungen, etc. . Architektonischen Randbedingungen umfassen die zur Verfügung stehenden Möglichkeiten einer Architektur, um die Ausführung eines Algorithmus zu beschleunigen. Programmierbare ASIPs bieten z.B. die Möglichkeit, den grundlegenden Instruktionssatz zu erweitern, wohingegen sich dedizierte Hardware-Designs besonders für stark paralle-

lisierbare Datenverarbeitung eignen. Unterschiedliche Architekturen bieten somit mit Blick auf verschiedene Algorithmen spezielle Vor- oder Nachteile und sind daher unter Berücksichtigung der späteren Applikation zu selektieren.
Extrinsische Anforderungen sind Randbedingungen, die sich durch Auswahl einer bestimmten Produktfamilie oder der verwendeten Technologie ergeben. Plattform-spezifische Randbedingungen sind beispielsweise die zur Verfügung stehenden Hardware-Ressourcen, z.B. die Anzahl der DSPs oder Block RAMs auf FPGAs. Technologie-spezifische Randbedingen geben bestimmte Eigenschaften durch den verwendeten Technologieknoten vor und sind somit von grundlegender Bedeutung für bestimmte Entwurfsschritte.

Mehrere Erscheinungsformen von quantitativen Kostenmodellen sind möglich:

1. Eine Kostenfunktion ist aus einem Satz von Stützpunkten aufgebaut, die z.B. aus einem Datenblatt oder aus Messungen extrahiert sind. Diese Datensätze dienen als Tabellen und liefern präzise Resultate für eine definierte Variation von Eingangsparametern. In bestimmten Anwendungsfällen ist es sinnvoll, fehlende Stützpunkte in den Tabellen zu interpolieren, um Auswirkungen zulässiger Eingangsparameter-Kombinationen zu approximieren, die im Vorfeld nicht untersucht worden sind.
2. Eine Kostenfunktion ist eine analytische Funktion, die bestimmte Auswirkungen von Eingangsparameter auf spätere Hardware-Kosten beschreibt. Es ist möglich, solche Funktionen auf Basis verschiedener Implementierungen zu entwickeln.
   Das Resultat einer modellbasierten Entwurfsraum-Exploration sind applikationsspezifische, intrinsische und extrinsische Charakteristika für betrachtete Plattformen, die den Prozess eines Systemdesigns unterstützen und somit den Designprozess signifikant erleichtern. Es ist wichtig, dass die Modellbibliothek kein funktionsfähiges System liefert, sondern vielmehr den Entwurfsraum derart einschränkt, dass Designfehler frühzeitig erkannt und behoben werden.

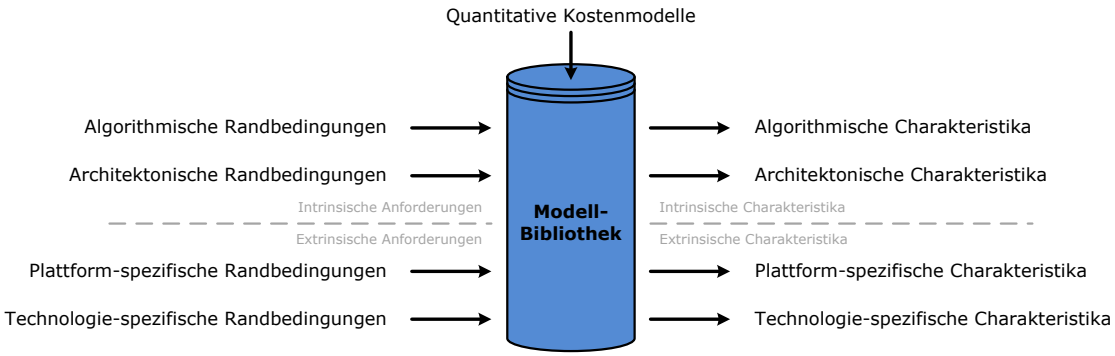


Abbildung 8: Die Modell-Bibliothek ist mit quantitativen Kostenmodellen gefüllt

Basierend auf intrinsischen Anforderungen (algorithmische und architektonische Randbedingungen) und extrinsischen Anforderungen (Plattform-spezifische und Technologie-spezifische Randbedingungen) lassen sich die Charakteristika eines möglichen Designs abschätzen, um den zur Verfügung stehenden Entwurfsraum sinnvoll und zweckgebunden einzuschränken.

3.3.2 Prototyp für modellbasierte Entwurfsraumexploration

Zur Umsetzung der oben genannten Methodik der modellbasierten Entwurfsraumexploration ist ein Software-Prototyp implementiert worden und liegt am Institut für Mikroelektronische Systeme vor. Die folgenden Abschnitte erläutern den Softwareentwurf und den Ablauf einer Entwurfsraumexploration mit dem genannten Software-Tool. Das folgende Diagramm in Abbildung 9 zeigt die Use Cases (UC), die in dem Tool umgesetzt wurden.

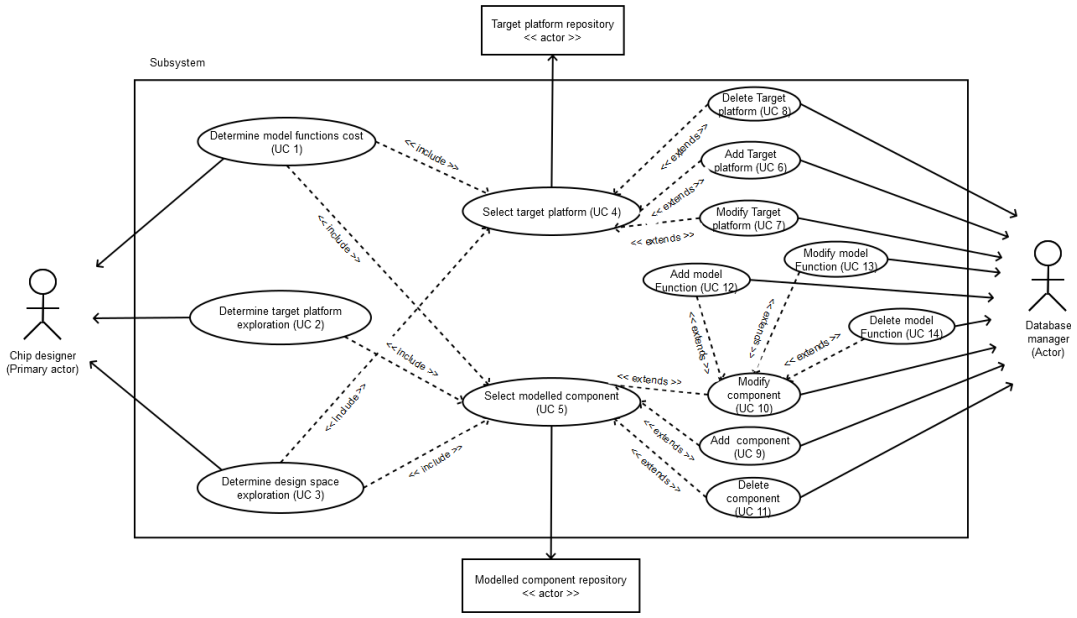


Abbildung 9: Use Cases in Softwareprototypen für modellbasierte Entwurfsraumexploration

Modelle für Hardwareplattformen (Target Platforms) sowie für Anwendungsbausteine (Model Components) werden in einer Datenbank verwaltet (Database Manager). Für die Hardwareplattformen besteht die Möglichkeit, Rahmenbedingungen (constraints) anzugeben, die die Plattform charakterisieren. Zu jedem Anwendungsbaustein werden verschiedene Aspekte modelliert, die dann durch jeweils eine Modellfunktion dargestellt werden. Modellfunktionen können als Formeln eingegeben werden, oder als tabellarische Modellfunktionen hinterlegt werden.
Der Hardwaredesigner, der eine Untersuchung des Entwurfsraums vornehmen möchte, hat drei verschiedene Szenarien (UC, Use Cases) zur Auswahl:

- Auswertung der Modellfunktionen (UC1),
- Evaluation von Hardwareplattformen (UC2),
- Entwurfsraumexploration (UC3).

3.3.2.1 Auswertung der Modellfunktionen (UC1)

In diesem Szenario stellt der Hardwaredesigner ein Anwendungsmodell zusammen. Er greift dabei auf die bestehenden Modellkomponenten zurück, die einzelne Komponenten der Anwendung beschreiben und von dem Tool in einer Datenbank verwaltet werden.

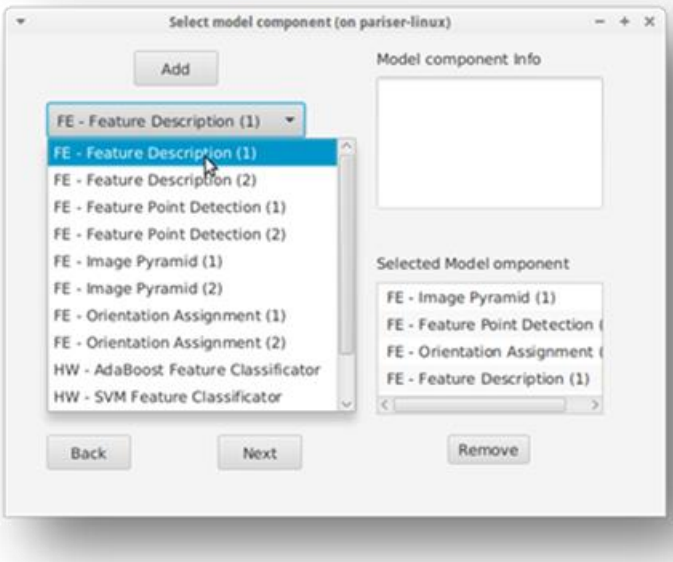


Abbildung 10: Auswahl von Modellkomponenten für ein Anwendungsmodell

Jede Modellkomponente enthält verschiedene Modellfunktionen, zu denen der Designer Werte für die Parameter der Modellfunktionen angibt. Das Tool wertet die Modellfunktionen der ausgewählten Modellkomponenten aus. Das Ergebnis wird in Form einer Tabelle dargestellt und ist für weitere Verarbeitung (z.B. in einer Tabellenkalkulation) exportierbar.
Zusätzlich besteht die Möglichkeit, dass der Designer in diesem Szenario eine Hardwareplattform wählt. Das Tool vergleicht die berechneten Werte der Modellfunktionen mit den Randbedingungen der Hardware und markiert Überschreitungen.

3.3.2.2 Evaluation von Hardwareplattformen (UC2)

Ziel dieses Szenarios ist die Suche nach möglichen Hardwareplattformen zu einem gegebenen Anwendungsmodell. Nachdem der Designer das Anwendungsmodell aus den Modellkomponenten aufgebaut und die Parameter für die in den Komponenten enthaltenen Modellfunktionen angegeben hat, wertet das Tool die Modellfunktionen aus und gleicht die Ergebnisse mit den Hardwareplattformen in der Datenbank ab, um die Plattformen zu identifizieren, auf denen es möglich ist, die modellierte Anwendung umzusetzen.
In der Datenbank ist jede Hardwareplattform mit gewissen Rahmenbedingungen (constraints) abgelegt. Diese beschreiben zum Beispiel die verfügbaren Hardwareressourcen, die maximale Taktfrequenz usw. Der Satz von Parametern, die zu einer Plattform gehören, ist vom Typ der Hardwareplattform abhängig. Das Tool unterstützt die Angabe beliebiger Parameter, die dann über ihren Namen mit den Modellfunktionen der Anwendungsbausteine identifiziert werden. Damit ist die Evaluation von Hard-wareplattformen über beliebige Typen von Hardwareplattformen durchführbar.

3.3.2.3 Entwurfsraumexploration(UC3)

In den beiden vorangegangenen Szenarien wurden durch den Designer die Parameter eines Anwendungsmodells vorgeben und das Tool wertete die Modellfunktionen entsprechend aus.
Die Exploration des Entwurfsraumes wird von dem Tool dadurch unterstützt, dass der Designer nach der Definition des Anwendungsmodells für die Parameter der Modelle nicht nur feste Werte vorgibt, sondern auch Intervalle und Schrittweite für die Parameter. Dadurch berechnet das Tool nicht nur einen Datenpunkt für das gegebene Anwendungsmodell, sondern eine Punktwolke, die verschiedene Konfigurationen des Anwendungsmodells beschreibt. Die resultierenden Tabellen für die einzelnen Modellfunktionen lassen sich wieder zur weiteren Verwendung exportieren.
Auch bei diesem Szenario ist es optional möglich, eine Hardwareplattform aus der Datenbank auszuwählen, gegen deren Restriktionen die einzelnen Punkte der Auswertung geprüft werden. So lassen sich Implementierungsvarianten, die auf der Zielplattform nicht umsetzbar sind, leicht identifizieren.

3.3.3 Fallstudie: Faltungsnetze

Für die Implementierung der Faltungsnetze für das Scene Labeling (siehe Abschnitt 3.7.1 „Straßenverlaufserkennung und Scene Labeling“) sind Modelle für den allgemeinen Berechnungsaufwand der einzelnen Layer in Abhängigkeit der Parameter dieser Layer aufgestellt worden. Diese Modelle erlauben eine Abschätzung der Laufzeit verschiedener Implementierungen unter Berücksichtigung von Implementationsparametern, wie zum Beispiel dem Parallelisierungsgrad. Im Folgenden sind exemplarisch zwei Modelle für verschiedene Layertypen beschrieben. Konkrete Modellparameter sind für eine Implementierung der Layer auf einem Tensilica LX5 ASIP ermittelt worden.
In einem Faltungsnetz werden die Eingabedaten schrittweise durch verschiedene Operationen verarbeitet. Die deutlich aufwändigste Operation dabei ist die Faltung der Daten mit verschiedenen Filterkernen. Andere Operationen, die in einem Faltungsnetz ausgeführt werden, beinhalten das Pooling und die Anwendung der Aktivierungsfunktion (squashing function). Eine eventuelle Fragmentierung der Bilder ist ohne großen Zusatzaufwand in die anderen Operationen integrierbar und ist daher in der Modellierung zu vernachlässigen.

Faltungslayer
Berechnet die Faltung eines Eingangsbildes ($N \times N$ Pixel) mit einem Faltungskern der Größe $n \times n$.
Eingabeparameter:

- Größe des Eingangsbildes $N \times N$
- Größe des Filterkerns $n \times n$

Allgemeines Modell:

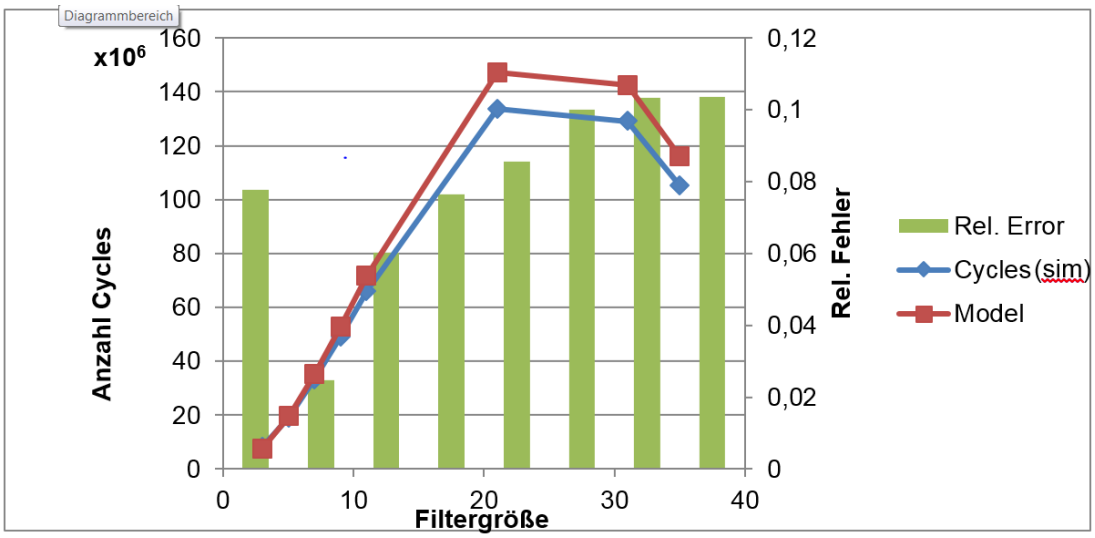
Die Anzahl der Operationen für eine Faltung ist proportional zu der Bildgröße und der Filtergröße, also zu $N^2 n^2$.

Berechnungsaufwand der Faltung (#Cycles)

Gemessen auf Tensilica LX5 ASIP.

$$P(N,n) = 370.628\big(N-(n-1)\big)^2 n^2$$

Bildgröße $\boldsymbol{N = 50 \times 50}$



Das Modell für eine einzelne Faltung hat im verwendeten Beispiel einen maximalen relativen Fehler von unter 10%, d.h. das modellierte Laufzeitverhalten der Faltung weicht um weniger als 10% von der Zyklus-akkuraten Simulation der Faltung ab.

Aktivierungsfunktion (squashing function)

Nicht lineare Aktivierungsfunktion, die auf jedes Ausgangspixel einer Faltung angewendet wird.

Eingabeparameter:

- Größe des Eingangsbildes $N \times N$

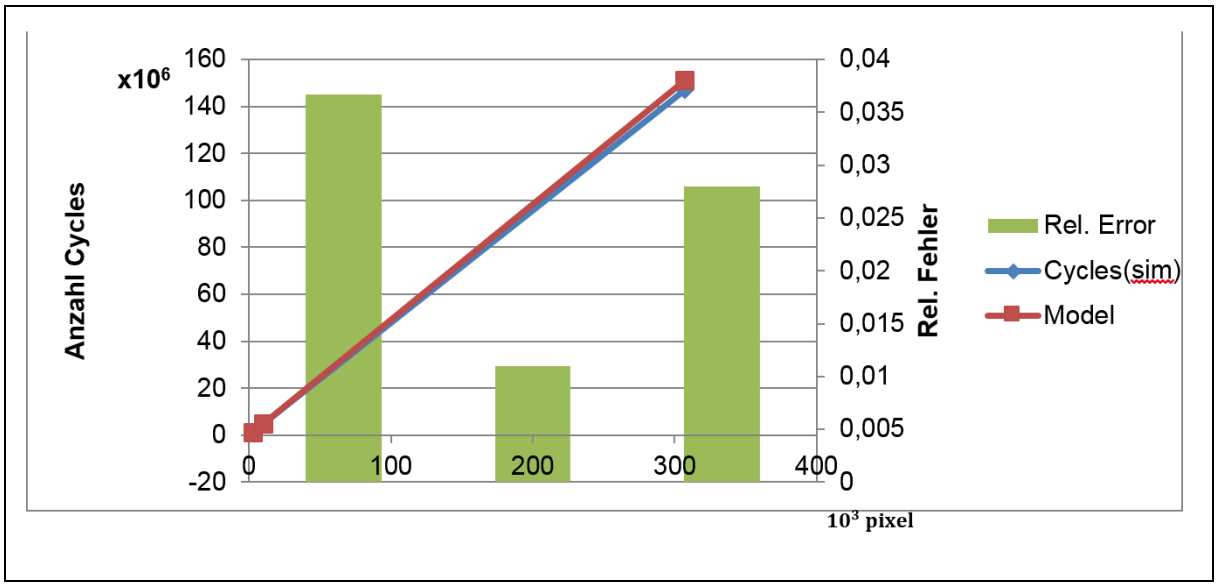
Allgemeines Modell:

Die Anzahl der Operationen für die Aktivierungsfunktion ist proportional zu der Bildgröße N^2.

Berechnungsaufwand der Aktivierungsfunktion (#Cycles)

Gemessen auf Tensilica LX5 ASIP.

$$P(N) = 491.7N^2$$



Der Aufwand zur Berechnung der Aktivierungsfunktion hängt auch von der Funktion selbst ab. In diesem Beispiel ist eine tanh-Aktivierungsfunktion verwendet, die mit ca. 491 Operationen pro Pixel ausgeführt wird.
Modelle für weitere Schritte des Faltungsnetzes (Feature-Pooling, verschiedene Aktivierungsfunktionen) sowie die Vorverarbeitung der Eingangsdaten sind ebenfalls er-mittelt worden. Damit ist die Aufwandabschätzung für eine gegebene Netztopologie möglich.

3.4 Beitrag dSPACE - Rapid Prototyping

Im Rahmen des modellbasierten Entwurfs werden Steuergerätefunktionen in der Regel unter Verwendung eines Rapid Control Prototyping Ansatzes entwickelt. Auf diese Weise ist es möglich neue Regelstrategien frühzeitig in den Entwicklungsprozess einzubringen, kurze Iterationszyklen zu erreichen und die Algorithmen direkt im Fahrzeug zu verifizieren. Die dSPACE MicroAutoBox II [Ref 2] ist ein weitverbreitetes Beispiel für ein Rapid Prototyping System, welches ein Echtzeit-Betriebssystem, einen skalierbaren Embedded Controller und vordefinierte I/O Schnittstellen zur Verfügung stellt. Das System ist kompakt und robust mit kurzer Boot-Zeit, so dass es im Fahrzeug eingesetzt werden kann.
Rapid Protoyping Systeme dienen typischerweise als Implementierungsplattform für die Regelalgorithmen und deren Anwendung. Sensoren, Aktoren und Fahrzeug-Busse können direkt angeschlossen werden. Verglichen mit dem Serien-Steuergerät stellt ein Rapid Prototyping System eine deutlich höhere Rechenleistung und quasi unbegrenzte RAM und Flash Ressourcen zur Verfügung, so dass auch komplexe Algorithmen in Echtzeit berechnet werden können.
Im Rahmen des DESERVE Projektes wurde eine universelle und offene Entwicklungsumgebung aufgebaut, die die Integration verschiedener kommerzieller Lösungen erlaubt. Dabei war es Ziel die Schnittstellen zwischen den verschiedenen Entwicklungsebenen so gut wie möglich zu standardisieren.
Es konnten drei verschiedene Ebenen identifiziert werden, siehe auch Abbildung 23:

- Level 1: PC-basiertes Entwicklungsframework
- Level 2: Embedded Controller Entwicklungsframework
- Level 3: Hardware spezifisches Entwicklungsframework (z.B. FPGA oder ASIC basiert)

Die DESERVE Rapid Prototyping Plattform adressiert alle drei Ebenen. Sie besteht aus den nachfolgend beschriebenen Elementen:

X86 PC Plattform mit RT Maps oder EB Assist ADTF
Die kommerziellen Werkzeuge RTMaps [Ref 4] von Intempora und EB Assist ADTF [Ref 3] von ElektroBit sind im automobilen Markt etabliert und decken verschiedene Phasen der ADAS Entwicklung ab. Diese Software Frameworks werden als Schnittstellen zu den verschiedenen Sensoren verwendet und als Framework für die Implementierung und die Simulation von C/C++/C# basierten Algorithmen, insbesondere für die Perzeption, die Sensor Daten Fusion und das Object Tracking eingesetzt. Sie können

auch zum Prototyping der ADAS Anwendungssoftware verwendet werden. RTMaps und EB Assist ADTF laufen auf Standard Windows oder Linux x86 PC Plattformen. Sie werden vorrangig in frühen Entwicklungsphase z.B. zum Prototyping oder zur Validierung der Algorithmen während Testfahrten im Fahrzeug verwendet. Allerdings ist der Echtzeit-Betrieb der Anwendung insbesondere bei komplexeren Software-Funktionen nicht garantiert.

Echtzeit-Prozessor-Plattform
Um Entwicklern frühzeitig im Entwicklungsprozess die Möglichkeit zu geben, verschiedene Ziel-Implementierungen zu vergleichen, werden die oben beschriebenen Software Frameworks mit einer Echtzeit-Prozessor-Plattform gekoppelt. Die Anordnung besteht aus verschiedenen Komponenten:

- Embedded PC und FPGA basierte Prozessor-Plattform:
  Ausgewählte Perzeptions-, Fusions- und Tracking Algorithmen können auf dem FPGA berechnet werden, dadurch erfolgt die Verarbeitung von komplexen Software Funktionen in Echtzeit.
- Embedded PC, FPGA und embedded Controller basierte Prozessor-Plattform:
  Ausgewählte Perzeptions-, Fusions- und Tracking Algorithmen werden auf dem FPGA berechnet, während Anwendungsfunktionen, die typischerweise in Matlab/Simulink beschrieben werden, auf dem Embedded Controller mit einem Echtzeit-Betriebssystem ausgeführt werden.

Die verschiedenen Eigenschaften der Embedded PC, FPGA und Embedded Controller Plattform können genutzt werden, um die Implementierung von unterschiedlichen ADAS Algorithmen und Hardware Architekturen zu ermöglichen.

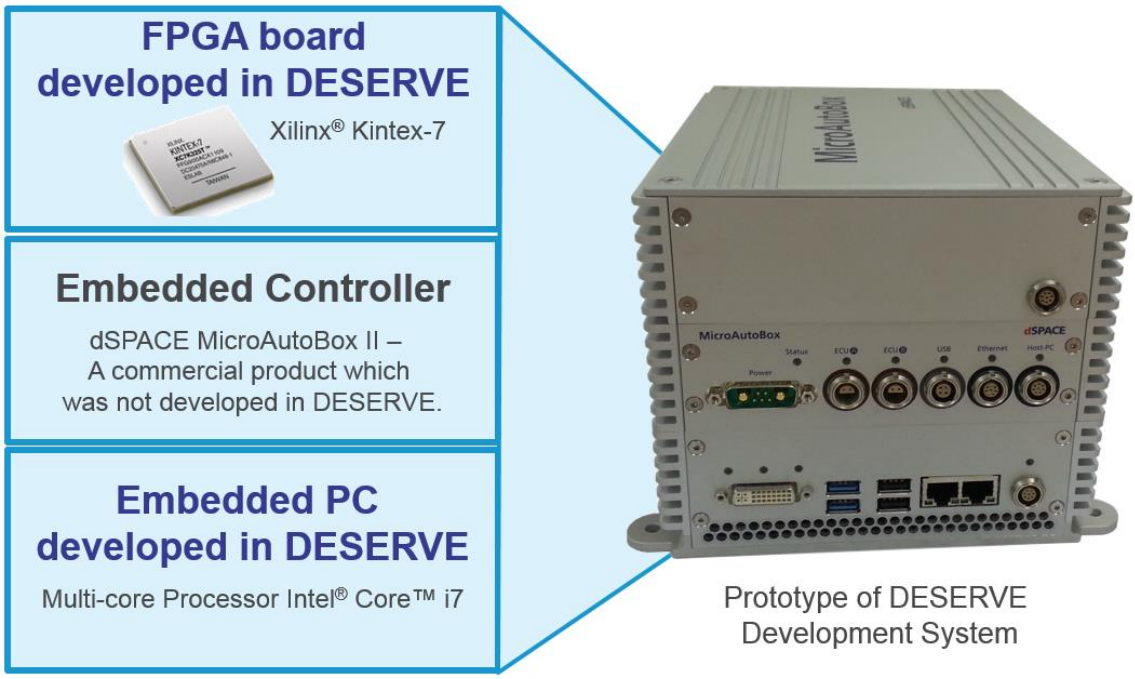


Abbildung 11: DESERVE spezifisches Rapid Prototyping System

Abbildung 11 zeigt das DESERVE Rapid Prototyping System bestehend aus den drei oben beschriebenen Ebenen.
Der embedded PC (Level 1) enthält einen Intel® Core™ i7 Prozessor [Ref 6] mit 2.5 GHz, 8 GB DDR3 SDRAM und 192 GB Flash Memory. Der Embedded PC stellt verschiedene Interfaces wie USB2.0, USB3.0, Gigabit Ethernet, DVI und PCIe zur Verfügung, um Ein- und Ausgänge unterschiedlicher Komponenten anbinden zu können.
Der Embedded Controller (Level 2) wird durch die Standard dSPACE MicroAutoBox II, ein kommerzielles Produkt, welches nicht im Rahmen des DESERVE Projektes entstanden ist, abgedeckt. Die MicroAutoBox ist für die Verwendung im Fahrzeug in Bezug auf Schock und Vibration qualifiziert. Sie beinhaltet einen leistungsfähigen Prozessor, ein Echtzeit-Betriebssystem, verschiedene I/O Schnittstellen und Bus-Anbindungen, wie CAN, LIN, Ethernet oder FlexRay [Ref 7].
Der dritte Level wird aus einem FPGA Board bestehend aus einem Xilinx® Kintex-7 FPGA [Ref 8] gebildet. Dieses ermöglicht die Berechnung von komplexen Algorithmen in Echtzeit.

Sowohl der embedded PC (Level 1) als auch das FPGA Board (Level 3) wurden als Prototypen im Rahmen des DESERVE Projektes entwickelt. Zusätzlich wurde ein FPGA Code Framework basierend auf dem AXI Protokoll [Ref 5] entworfen, welches es erlaubt applikationsspezifischen FPGA Code flexibel zu integrieren.

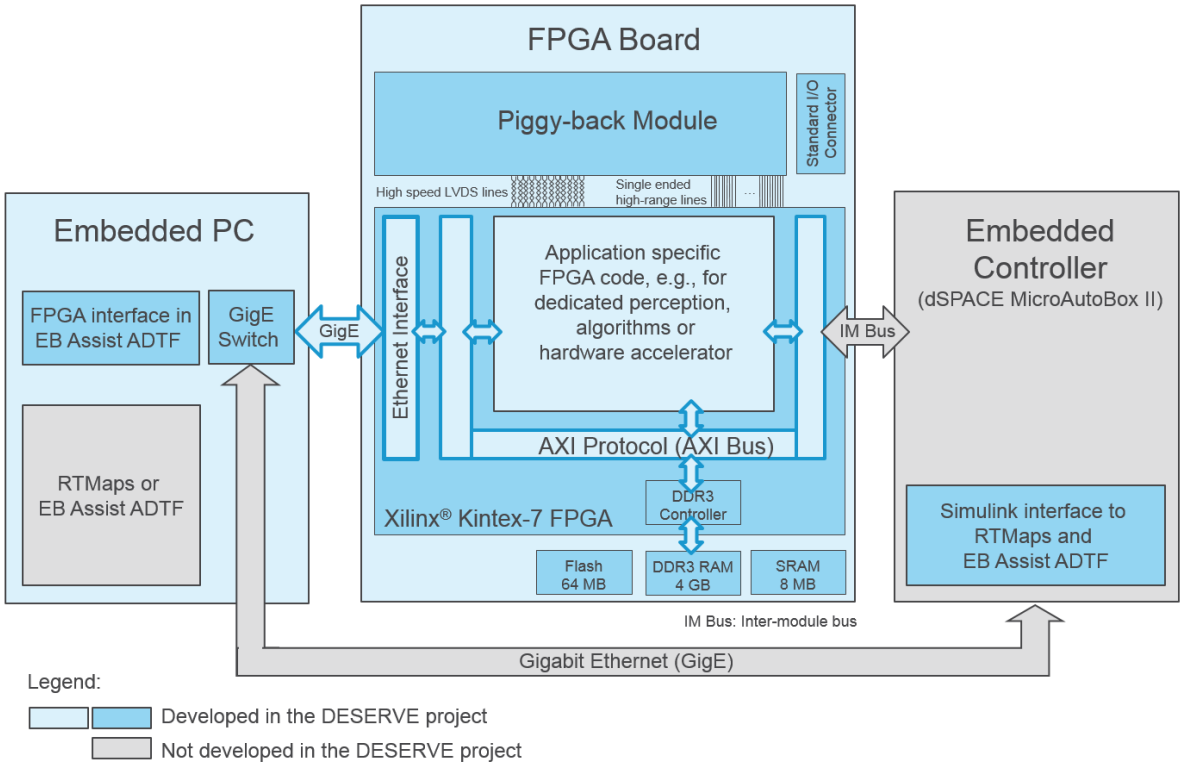


Abbildung 12: Architektur des DESERVE Rapid Prototyping Systems

Abbildung 12 zeigt die Architektur des DESERVE Rapid Prototyping Systems.
RTMaps oder EB Assist ADTF werden auf dem Embedded PC installiert. Die Daten können über Gigabit Ethernet direkt mit der Simulink Applikation, die auf dem Embedded Controller läuft, ausgetauscht werden. Die Simulink Schnittstelle für die MicroAutoBox II zu RTMaps und EB Assist ADTF ist in Form von Blocksets [Ref 9], [Ref 10] verfügbar. Zusätzlich erlaubt ein generisches Kommunikationsinterface basierend auf Gigabit Ethernet die Daten der Umgebungssensoren, die mit EB Assist ADTF verbunden sind, direkt auf das FPGA zu übertragen. Die Architektur ermöglicht es, die Algorithmen in einem ersten Entwicklungsschritt auf dem Embedded PC zu untersuchen. Wenn die Algorithmen komplexer und ausgereifter werden, können sie nach und nach auf das FPGA oder den embedded Controller portiert werden, um einer Implementierung, die der Ziel-Hardware und Software im Fahrzeug näher kommt, zu erreichen. Das FPGA Board ermöglicht weiter die Integration von echten Hardware-Acceleratoren und eine Schnittstelle zu Micro Controllern für die Serie, wie z.B. für den Infineon AURIX Controller [Ref 11]. Außerdem können Umgebungssensoren, wie Kamera- oder Radar-Sensoren direkt mit dem FPGA verbunden werden, in dem ein dediziertes Piggy-Back Modul [Ref 12] verwendet wird. Abbildung 13 zeigt ein Bild des FPGA Boards, welches in einem MicroAutoBox Gehäuse verbaut ist.

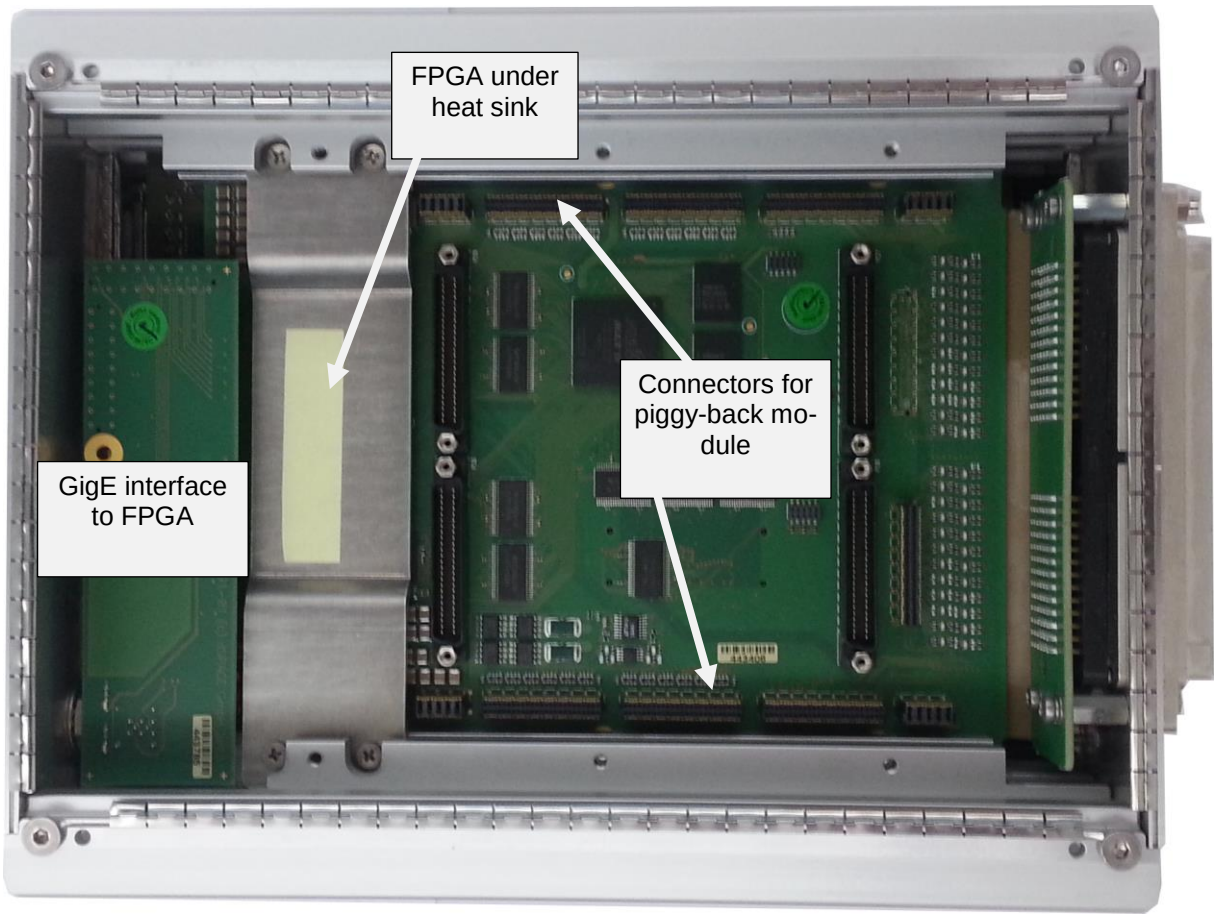


Abbildung 13: FPGA Board

Mit dem DESERVE Rapid Prototyping Konzept können verschiedene Entwicklungsprozesse implementiert werden. Im Rahmen des Projektes sind mit der Prototyping Plattform verschiedene Demonstratoren und Anwendungsfälle aufgebaut worden.
Für den Daimler Demonstrator Inter Urban Assist wurde die Prototyping Plattform in Verbindung mit EB Assist ADTF auf dem PC und dem FPGA Board als Echtzeit-Prozessing Einheit verwendet.
Für den CRF Demonstrator ist eine weitere Variante bestehend aus dem RTMaps Software Framework in Kombination mit der MircoAutoBox II und dem embedded PC als Echtzeit Processing Plattform eingesetzt worden. Es wurden Anwendungen, wie automatische Fußgänger-Notbremsung, Erkennen eines abgelenkten oder ermüdenden Fahrers gezeigt.
Weiter ist die Rapid Prototyping Umgebung eingesetzt worden, um eine Entwicklungsumgebung für Car2x Anwendungen, wie in Abbildung 14 dargestellt, zu erstellen. Dazu ist ein dediziertes Blockset für die Protokollinterpretation von Car2x Botschaften erstellt worden. Der Inhalt jeder Nachricht kann als Signal Vektor in Simulink bereitgestellt werden. Anwender sind in der Lage Filter für Car2x Signale zu konfigurieren, um nur die für eine Anwendung relevanten Signale in der Modellierungsumgebung anzuzeigen. Die Kodierung und Dekodierung eines Blocks wird abgeleitet aus den durch ETSI standardisierten ASN.1 Beschreibungsdateien. Eingehende Car2X-Botschaften werden in der LDM (Local Dynamic Map) gespeichert und die relevanten Botschaftsinhalte an einzelne Car2X-Anwendungen verteilt. Das Simulink-Modell wird über Ethernet mit einem geeigneten Car2X-Hardware Adapter, wie zum Beispiel der MK4 Einheit von NXP/Cohda Wireless, an den Funkkanal angebunden. Neben den Car2X-spezifischen Blöcken bietet die Car2X Lösung ein spezielles Blockset zum Codieren und Decodieren von GNSS-Daten gemäß dem Standard NMEA 0183. Abbildung 14 zeigt die Entwicklungsumgebung für Car2x Anwendungen.

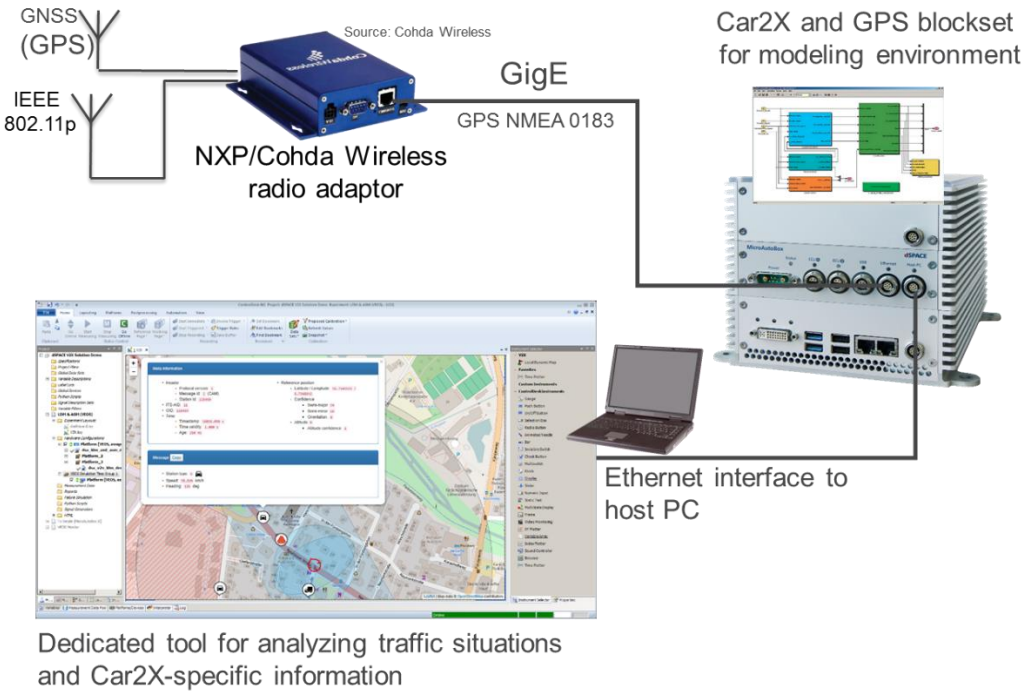


Abbildung 14: Car2x Anwendung

3.5 Beitrag Infineon – Embedded Architekturen

Auf den DESERVE-Plattformgedanken zur nahtlosen Integration von Algorithmen über verschiedene Stufen des Entwicklungsprozesses hinweg wurde in Abbildung 3 bereits genauer eingegangen. Für den Halbleiterhersteller Infineon AG lag der Fokus der Arbeiten in DESERVE auf den höheren Entwicklungsstufen (ab Level 3), d.h. auf einer vollständig eingebetteten (embedded), AUTOSAR-kompatiblen Architektur zur Evaluierung von Algorithmen in Echtzeit in Prototypfahrzeugen, sowie der Implementierung von Sicherheitsanforderungen gemäß ISO 26262 (z.B. vor dem Hintergrund der Notwendigkeit einer Prä-Zertifizierung als Voraussetzung für die Durchführung von Tests auf öffentlichen Straßen).

Für Level 3 wird der Standard PC durch eine embedded PC Lösung substituiert, die hinsichtlich Schock-, Vibrations- und Temperaturanforderungen den automotive Anforderungen genügt. Die Plattform-Lösung ermöglicht die Integration von Hardware-Beschleunigern, so dass selbst sehr rechenintensive Algorithmen in Echtzeit im Fahrzeug evaluiert werden können. Die Plattform verhält sich wie ein Prototyp-Steuergerät, das wie jedes Fahrzeugsteuergerät mit der Zündung an-/abgeschaltet wird.

Zur Unterstützung der modellbasierten Algorithmen Entwicklung muss die Level 3 Plattform vollständig AUTOSAR-kompatibel sein. Hierzu ist anzumerken, dass zum heutigen Zeitpunkt kein vollständig zertifiziertes AUTOSAR 4.0 Echtzeit-Betriebssystem mit Memory Protection verfügbar ist. Die Entwicklung eines solchen Betriebssystems war jedoch nicht Gegenstand von DESERVE.

Weiterhin müssen bei Level 3 bereits Sicherheitsmechanismen entwickelt und implementiert werden. Dazu wird das ADAS-System nach ISO 26262 klassifiziert. Viele Assistenzsysteme greifen in das Motormanagement oder auf Bremse und Lenkung ein, und sind daher in Automotive Safety Integrity Level (ASIL) D einzustufen. Für die Zertifizierung sind strenge Auflagen zu erfüllen. Der Zertifizierungsprozess ist sehr eng mit der Hardware verknüpft. Da die Zielhardware gegenüber der Level 3 Hardware unterschiedlich ist, ist in Level 3 lediglich eine Prä-Zertifizierung möglich.

Level 4 beinhaltet die Zielhardware im späteren Produkt (z.B. Multicore Controller mit integrierten Kundenspezifischen ASICS/FPGA oder Hardwarebeschleunigern, schematisch dargestellt z.B. in Abbildung 12). Mittels dieser Zielplattform kann die Zertifizierung gemäß ASIL-Klassifikation erfolgen.

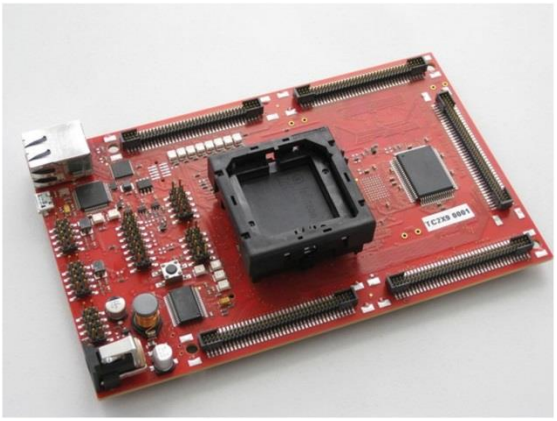


Abbildung 15: DESERVE Plattform der Infineon AG

3.5.1 Plattformarchitektur

Das DESERVE Plattformkonzept wurde im Projekt entwickelt und validiert. Die DESERVE Hardware-Architektur ist sehr flexibel. Sie kann entweder verteilt und skalierbar implementiert sein (mehrere Module, wobei jedes Sensieren, Verarbeiten und Agieren kann) oder konzentriert (wobei alle Sensoren und Aktoren an eine Steuereinheit angeschlossen sind). Für ein Höchstmaß an Flexibilität erfolgte das Design so, dass Subsysteme unterschiedlicher Generationen parallel eingesetzt werden können. Auf diese Weise können gut getestete und zertifizierte Kern-Funktionen „alter" Systeme zumindest teilweise als SoC (System on Chip) implementiert werden.

Die wesentliche DESERVE Idee betrifft die Verwendung ein und derselben Systemplattform für alle funktionalen ADAS-Module (s. Abbildung 16). Dies führt zu drei Herausforderungen:

- Automotive Qualität: Die Plattform muss hohe Verfügbarkeit über den gesamten automobilen Temperaturbereich bereitstellen, sämtliche Umweltanforderungen erfüllen und (wenigstens in Level 3 und 4) die entsprechenden ASIL Anforderungen einhalten;

- Hardware-Erweiterungen: Die Plattform muss von Anfang an so entworfen werden, dass spätere Hardwareerweiterungen möglich sind. Dies gilt für die Anbindung zusätzlicher Sensoren genauso wie für weitere FPGAs/DSPs/Hardwarebeschleuniger etc. im Fall benötigter höherer Rechenleistung. Die externen Komponenten müssen kaskadierbar sein. Insbesondere muss es möglich sein, Serienteile früherer Generationen mit einzubinden;

- Darüber hinaus wird eine lückenlose Tool Chain benötigt, wobei eine Schlüsselanforderung in der Wiederverwendbarkeit existierender Toolketten und Funktionsmodule über mehrere Produktgenerationen liegt. Weiterhin muss es die Plattform gestatten, eine möglichst breite Palette von Use Cases zu bedienen, z.B. - wie im Projekt DESERVE gezeigt – die nächste Generation von Radar- und Videosensoren für anspruchsvolle Fahrsituationen.

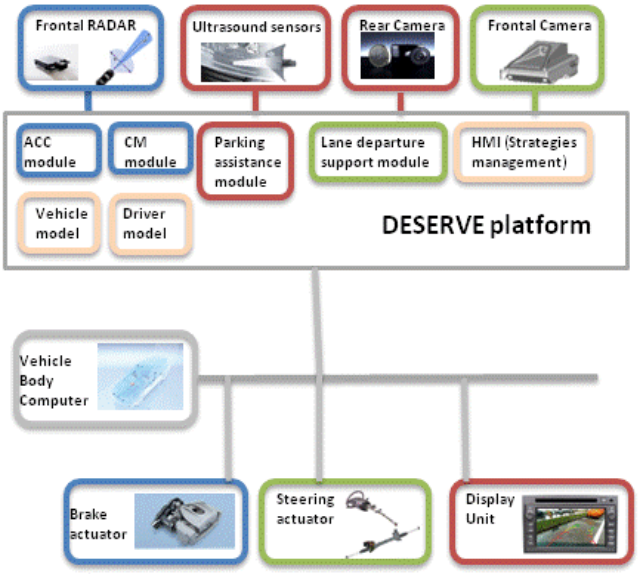


Abbildung 16: DESERVE Plattform – Eine gemeinsame Plattform für alle ADAS Module

Genauso wie die Hardware Applikationsanforderungen der ADAS-Module erfüllen muss, ist dies auch für die Software-Architektur der DESERVE-Plattform der Fall. Für spätere Portierbarkeit wurden AUTOSAR-Standards im Projekt berücksichtigt und bereits im Forschungsprojekt umgesetzt, wo immer die Implementierung mit vertretbarem Aufwand möglich war.

Auf Basis des AUTOSAR Standards wurde die Software-Architektur in drei Haupt-Layer strukturiert:

- Low Level: Die Basis-Software abstrahiert von der zugrundeliegenden Hardware, stellt Basis- und „komplexe“ Treiber und Dienste für darüber liegende Schichten zur Verfügung, z.B. Speicher, I/O.
- Middle Level: Diese Ebene beinhaltet den Virtual Function Bus sowie die Runtime-Infrastruktur. Der AUTOSAR Standard sieht zwei Architektur-Konzepte vor, die die Infrastruktur-unabhängige Softwareentwicklung ermöglichen. Dies sind der Virtual Function Bus (VFB) und die Runtime Infrastruktur (RTE). Der VFB stellt generische Kommunikationsdienste für beliebige AUTOSAR Komponenten bereit. Obwohl diese virtuell sind, werden sie in späteren Entwicklungsphasen der darunter liegenden Hardware-Infrastruktur zugeordnet. Das Runtime Environment (RTE) stellt eine aktuelle Repräsentation des virtuellen Konzepts des VFB für ein spezifisches Steuergerät bereit.
- High Level: Hierunter fallen die (modular realisierten) Applikationssoftware-Komponenten.

Modularität ist eines der wichtigsten Direktiven beim Design der globalen Architektur, ihrer Funktionen und Module des eingebetteten Systems. Verschiedene Multi-Tasks (Prozesse) können im Zuge von Ressourcenteilung auf einer CPU ausgeführt werden. Die Programmierung mehrerer Threads erfolgt meist über C++.

Allen ADAS-Applikationen gemeinsam ist, dass insbesondere die Algorithmen für Radar- und Video-Verarbeitung immense Anforderung an die Rechenleistung stellen. Die Verfügbarkeit leistungsfähiger Mehrkern-Micro Controller mit DSP-optimierten Algorithmen trägt zu hoher Performance und damit letztendlich zu weiterer Verkürzung der Entwicklungszeiten bei.

3.5.2 Schnittstellen

Schnittstellendefinitionen spielen eine entscheidende Rolle für die Kommunikation und den Datenaustausch zwischen den Modulen der DESERVE Plattform und der angeschlossenen Sensoren. In DESERVE wurden dazu abstrahierte Interface-Deskriptoren definiert. Die Interface-Architektur wurde individuell für jeden Layer des OSI-Modells definiert, angefangen vom physical layer bis hoch zum application layer (Abbildung 17).

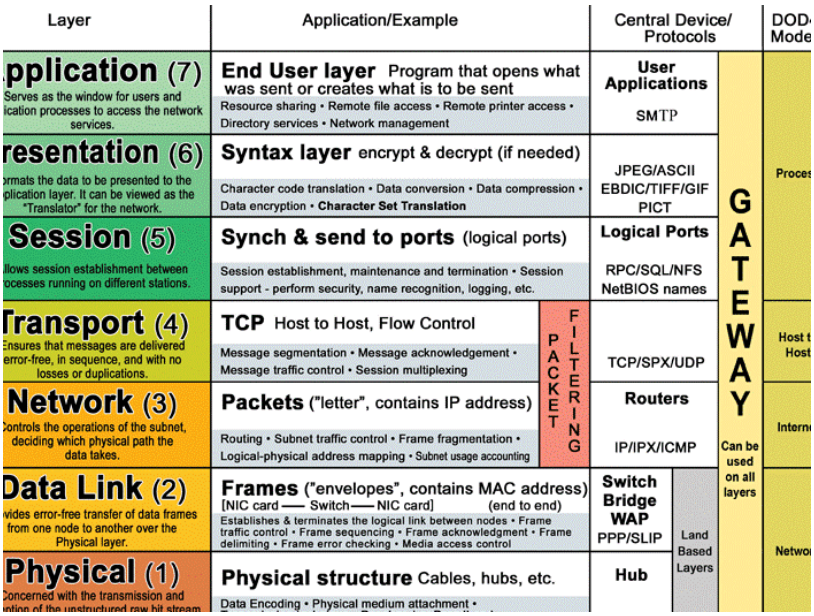


Abbildung 17: OSI 7-Schichtmodell

Basierend auf dem OSI-Modell wurden für die DESERVE Plattform folgende Layer definiert:

- Physical Data Layer: Dieser Layer umfasst die physikalische Struktur der Sensor Komponenten oder Kommunikationseinheiten. Häufig sind proprietäre Datenstrukturen zu berücksichtigen, um Sensoren oder Aktoren anbinden zu können. Abhängig davon, ob Rohdaten oder vorverarbeitete bzw. aggregierte Daten übertragen werden sollen, kann die Datenrate zwischen einigen 100 bit/s bis zu mehreren Gbit/s betragen. Weiterhin ist ein Rückkanal für die Sensorkalibrierung mit vorzusehen.
- Communication Layer: Diese Schicht beinhaltet Data Link, Network und Transport Layer und umfasst die komplette Information darüber, wie die Daten über die Verbindung zwischen Plattformmodulen und -komponenten zu senden sind. Favorisierte Protokollarten für die Kommunikation sind TCP/IP oder UDP. Auch hier können proprietäre und neuartige Busstandards wie Thunderbolt oder USB 3.0 zusätzlich zum Ethernet-Bus-Protokoll zum Einsatz kommen.
- Application Service Layer: Dies ist die höchste Schicht und etabliert die abstrahierte Datenverbindung zu den Anwendungen. Ähnlich dem TCP/IP Protokoll werden der Session Layer und der Presentation Layer zusammengefasst im Application Layer.

Das AUTOSAR Schnittstellenkonzept wurde im Projekt berücksichtigt und implementiert, wo immer sinnvoll und mit angemessenem Aufwand umsetzbar, denn der AUTOSAR-Betriebsmodus korreliert sehr gut mit dem DESERVE-Ansatz (Abbildung 18).

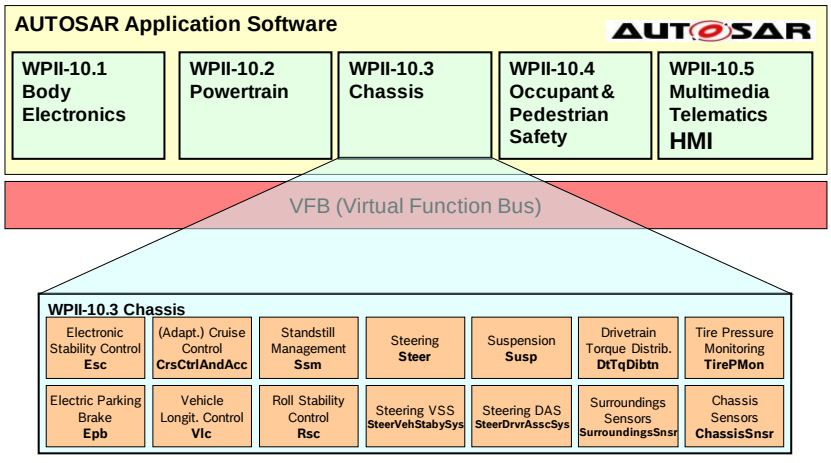


Abbildung 18: AUTOSAR Applikationssoftware Koncept

Heute eingesetzte Kommunikationsprotokolle im Automotive-Sektor umfassen:

- CAN (Controller Area Network)
- VAN (Vehicle Area Network)
- FlexRay (für im Vergleich zu CAN höhere Datenraten; FlexRay unterstützt Fehler-Toleranz-Eigenschaften)
- LIN (Local Interconnect Network)
- SAE-J1939 and ISO 11783 (Anpassungen von CAN für Nutzfahrzeuge)
- MOST (Media-Oriented Systems Transport)
- Keyword Protocol 2000 (KWP2000)
- Andere wie z.B. DC-BUS, IDB-1394, SMARTwireX, SAE-J1850, SAE-J1708, SAE-J1587 and ISO-9141-I/-II

Derzeit am Markt befindliche ADAS-Systeme sind häufig so entwickelt, dass sie in einem Steuergerät eine Funktion bereitstellen. Meist sind dazu in einem Gehäuse Sensor und Signalverarbeitungseinheit untergebracht. Auf diese Weise umgeht man das Problem, Rohdaten – die z.B. von einer Kamera (Pixel) oder einem Radar (Abtastwerte) geliefert werden – über einen Hochgeschwindigkeitsbus zu übertragen. Stattdessen werden über die o.g. Bussysteme nur noch vorverarbeitete, kondensierte Daten übertragen.

Für die Entwicklung von Algorithmen über ADTF, RTMaps o.ä. werden jedoch die Rohdaten der Sensoren benötigt. Spezifische Elektronik (basierend auf Ethernet Kommunikation) und Software zeichnet diese Daten in Echtzeit auf. Heute werden dafür bis zu 4 Ethernet Verbindungen zwischen Sensor und Datenlogger (z.B. PC) integriert. Die Bandbreite der Kommunikation wächst mit zunehmend komplexer der ADAS-Systemen immer mehr an.

Nicht nur für die Entwicklung, sondern auch für die Implementierung im Produkt werden bei Zugrundelegung des DESERVE Plattformkonzepts künftig deutlich höhere Datenraten benötigt. Die Definition geeigneter Hochgeschwindigkeits-Schnittstellen ist ein Schlüssel zur Verbesserung der Leistungsfähigkeit von ADAS-Systemen der nächsten Generation: Optimierte Schnittstellen ermöglicht optimierten Datenfluss und Systemperformanz. Für die beiden Sensorfamilien – Kamera und Radar – werden dedizierte Schnittstellen benötigt. Beide wurden im Rahmen dieses Projekts definiert.

- Parallel Camera Interface (CIF)

  Das Camera Interface (CIF, Abbildung 19) repräsentiert die Schnittstelle für den gesamten Video- und Standbildtransfer vom Bildsensor in den Bildspeicher. Mehrere Hardwareblöcke, die bereits beim Einlesen wichtige Verarbeitungsoperationen rechnen, stehen darüber hinaus im CIF zur Verfügung (z.B. Transfer von RAW Bildformat in nicht Rahmen-synchronisierte Datenpakete). Bei CIF handelt es sich um ein 16 bit paralleles Interface. Die Ausgangsdaten werden über ein Speicherinterface zum BackBone Bus (BBB) System übertragen. Die Programmierung des CIF erfolgt über Register Lese-/Schreib-Transaktionen unter Verwendung des BBB Slave Interface.

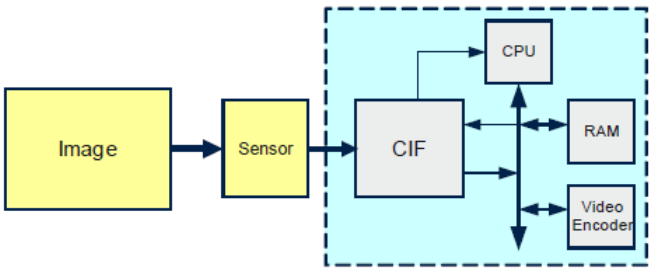


Abbildung 19: Camera Interface (CIF) Übersicht

CIF ist optimiert für höchste Datenraten bei niedrigem Leistungsverbrauch und stellt Sensor-/Kamerainterface für eine Vielzahl von Videoanwendungen zur Verfügung, z.B. Video Capturing/Encoding, Standbild-Capturing mit On-the-fly JPEG Enkodierung und RAW Rahmendaten-Gewinnung.

- Serielles Radar Interface (RIF)

  Analog-zu-Digital (ADC) Wandlerraten sind rückblickend kontinuierlich gestiegen, um den Anforderungen von ADAS, aber auch von Industrieanwendungen, Kommunikation oder Consumer-Elektronik, zu genügen. Verbunden mit der Notwendigkeit, Signale möglichst früh in der Signalkette zu digitalisieren, motivierte dies die Entwicklung von Hochgeschwindigkeits- ADC-Kernen, die bei Taktraten von 100-200 MHz mit 8-12 bit Auflösung quantisieren können.
  Zur Verkürzung von Settling Time der Wandlerausgänge zwischen Schaltvorgängen ist der Einsatz von LVDS (Low Voltage Differential Signalling) für ADC Hochgeschwindigkeitsausgänge vorteilhaft. Infineon integriert LVDS Ausgänge in seine neue Generation von schnellen ADCs, und ebenso LVDS Eingänge in seine neuen Micro-Controller Designs. Die mit ca. 350 mV im Vergleich zu anderen Schnittstellen deutlich niedrigeren Spannungspegelunterschiede bieten intrinsisch Vorteile durch verkürzte Schaltzeiten und verbesserte elektromagnetische Verträglichkeit (durch die Differenzeingänge löschen sich Störungen weitgehend aus).
  Zwei Standards zur Definition von LVDS existieren: Der Erste - ANSI/TIA/EIA-644 – trägt den Titel ”Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits.” Der Zweite - IEEE Standard 1596.3 – ist bezeichnet mit ”IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI)”.

- Generische Schnittstelle zur Kommunikation zwischen ADTF und FPGA basierter Hardware Plattform.

  Die dritte neue Schnittstelle dient der einfachen und standardisierten Kommunikation zwischen einem ADTF-Projekt (das z.B. auf dem PC läuft) und einer FPGA basierten Hardware-Plattform. Für diese generische Schnittstelle wird in DESERVE Gigabit Ethernet vorgeschlagen. Eine prototypische Implementierung und Evaluierung erfolgte im Projekt.

3.5.3 Sicherheitsstandards und Konzepte zur Zertifizierung

Die Modularisierung einer gemeinsamen ADAS-Plattform bewirkt signifikante Auswirkungen auf die (System-)Sicherheit, da die Module interagieren. Wie Abbildung 20 verdeutlicht, impliziert die Interaktion, dass eine Änderung des Betriebs eines einzelnen Moduls Auswirkungen auf eine ganze Reihe anderer Module in seiner Betriebsumgebung besitzt.

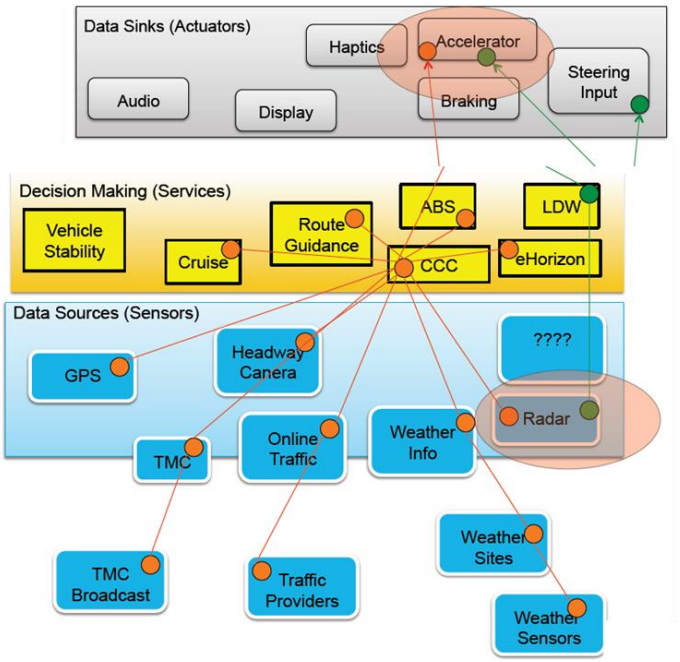


Abbildung 20: Modulinteraktion impliziert Änderungen im Systemverhalten

Die zentrale Frage lautet damit: Wie kann eine Systemzertifizierung erfolgen, wenn ein einzelnes Modul ausgetauscht wird? Der internationale Sicherheits-Standard ISO 26262 für Straßenfahrzeuge (bis 3,5 to) enthält hierfür das sogenannte Konzept des „Safety Element out of the Context“ (SEooC). Ein SEooC ist definiert als eine Komponente, für die keine einzelne vorbestimmte Applikation in einem spezifischen System existiert. Daher weiß ein SEooC Entwickler nichts darüber, welche konkrete Rolle sein „Produkt“ im Sicherheitskonzept des späteren Gesamtsystems spielen wird. Subsysteme, Hardware-Komponenten, auch Software-Module dürfen als SEooC entwickelt werden. Typische Software-SEooCs zeichnen sich durch Wiederverwendbarkeit und Anwendungsunabhängigkeit aus. Beispiele sind etwa Betriebssysteme, Bibliotheken oder Middleware-Komponenten.

Für SEooC Komponenten schlägt der Standard vor, Sicherheitsanforderungen (z.B. ASIL D) anzunehmen und für die Entwicklung zugrunde zu legen. Für den späteren Einsatz des SEooC in einem System muss der Systementwickler prüfen, ob die bei der Entwicklung zugrunde gelegten Sicherheitsanforderungen für sein Gesamtsystem reichen oder nicht. Passen die vom Gesamtsystem geforderten und die durch das SEooC-Modul garantierten Anforderungen zusammen, kann die Komponente in das System integriert werden.

Der ISO Standard spezifiziert relativ grob den Prozess zur Einbettung von SEooC Entwicklungen in den Standard-Sicherheits-Lebenszyklus. Dieser Prozess basiert auf hierarchischer Modularisierung und fokussiert auf die Rolle eines SEooC als Sub-Komponente des Gesamtsystems. Es ist davon auszugehen, dass die Integration einer SEooC Komponente während der Systementwicklung erfolgt. Daher ist eine Unterstützung von offenen Systemen, bei denen dynamisch Komponenten integriert werden können, nicht explizit vorgesehen.

Als Beispiel für die Sicherheitsaspekte des DESERVE Plattform-Konzepts werden zusammenfassend einige Sicherheitsmechanismen betrachtet, die die Multicore-Micro Controller-Plattform AURIX von Infineon bietet. Diese Plattform kann als Beispiel einer Hardware-Instanz der DESERVE-Plattform (Entwicklungs-Level 3-4) interpretiert werden. Die verfügbare Sicherheitsdokumentation umfasst:

- Safety Case Report: Dieser Bericht liefert Argumente und Nachweise, wie die Ziele der ISO 26262 und der Sicherheitsanforderungen an Komponenten vollständig eingehalten werden;
- FMEDA (Failure Modes Effects and Diagnostic Analysis): Kunden- und Infineon-vertrauliches Dokument;
- Safety Manual: Dieses Manual enthält Anwendungsfälle und Empfehlungen für die Applikationsebene, eine Zusammenfassung der Sicherheitsfeatures und der -mechanismen einschließlich der empfohlenen Anwendung der Features sowie der erreichten Sicherheitsmatrix und der daraus resultierenden ASIL-Stufe.

Die AURIX Micro Controller Plattform wurde als SEooC entwickelt (Abbildung 21) und stellt die in Abbildung 22 skizzierten Sicherheitsmechanismen bereit. Ziel der Architektur ist eine sichere Rechenplattform für Sicherheitssysteme bis ASIL D. Diese höchste Automotive Sicherheitsstufe wird für die nächste Generation von ADAS-Systemen benötigt. Um dieses Ziel zu erreichen, sind neben sicherer Hard- und Software auch sichere Software Architekturen und ein sicheres Betriebssystem erforderlich.

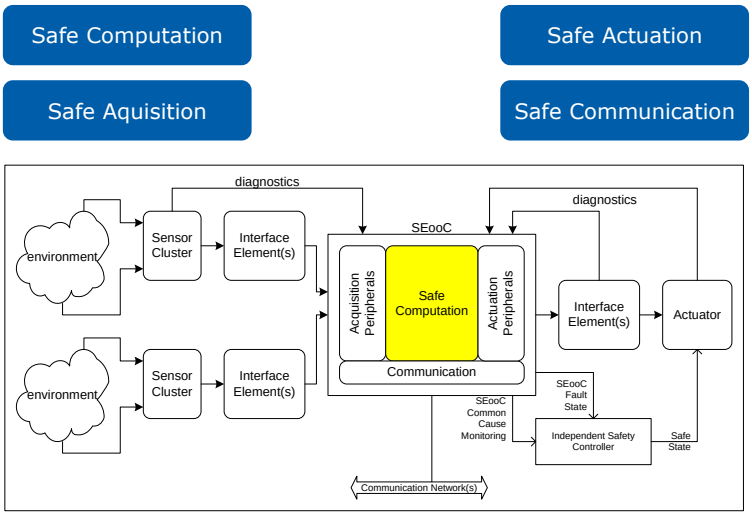


Abbildung 21: SEooC Sicherheitsmechanismen

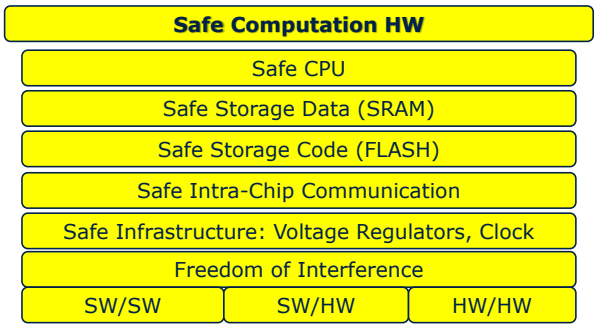


Abbildung 22: Generische Elemente einer sicheren Rechnerhardware-Plattform

Die sichere Hardware-Plattform muss Hardware-Redundanz (z.B. realisiert über verzögerte Lockstep-CPU zusammen mit verbesserten Timing-Funktionalitäten) bereitstellen. Sichere SRAMs ermöglichen Redundanz der Information (realisiert z.B. durch Standard SECDED ECC und Adresssignaturen). Auch ein sicherer Flash-Speicher ist erforderlich (z.B. realisiert über verbesserte ECC). Verbesserte Fehlererkennungs-Codes für Adress-/Datenleitungen ermöglichen sichere Verbindungen und unterstützen Redundanz der Informationen.

Wie erwähnt, erfordert die sichere Software Plattform den Einsatz eines ASIL D kompatiblen Betriebssystems, welches die Funktionalitäten Memory Protection und Time Protection bietet. Weiterhin muss es Dienste wie die Überwachung des Programmflusses, Sicherheitsprotokolle für die Ende-zu-Ende-Kommunikation und einen sicheren Interrupt Vektor besitzen. ASIL D Software muss gemäß ISO 26262 Part 6 entwickelt werden.

Die AURIX Plattform garantiert Rückwirkungsfreiheit der Software durch Software-Isolation. Rückwirkungsfreiheit auf Hardwareebene wird durch Hardware-Isolation garantiert. Die Memory Protection Unit (MPU) der CPU überwacht den direkten Zugriff auf lokale Speicher und die Softwaretasks und erlaubt dynamische Rekonfiguration. Die Bus-MPU überwacht SRAM-Zugriffe. Schließlich überwacht eine Register Access Protection die Schreib-Zugriffsrechte auf die Modulregister.

3.6 Beitrag Bosch - Entwicklung eines MIMO-Radarprototypen unter Zuhilfenahme der DESERVE Entwicklungsplattform

Eine interessante Aufgabenstellung im deutschen Anwendungsfall IUA (Inter-Urban Assist) ist neben der hochkomplexen und sehr rechenintensiven Bildverarbeitung von Videokameras auch bei der Radarsignalverarbeitung zu finden. Der Wunsch und die daraus abgeleiteten Anforderung an den Radarsensor nach immer besserer Szenenauflösung und Situationserkennung wachsen überproportional zur Weiterentwicklung von Laptop und PC Leistung an, wodurch altbewährte Ansätze, die Funktionsentwicklung zuerst prototypenhaft auf einem PC oder Laptop durchzuführen, nicht mehr greifen.
Es sind neue Ansätze im Bereich des Rapid Prototypings gefragt, die dem Anspruch nach erhöhter Rechenleistung und Datendurchsatz unter Echtzeitbedingungen Rechnung tragen.
Genau hier setzt erwartungsgemäß die DESERVE Methodik an und versucht, wie in Abbildung 3 dargestellt, die Entwicklungsschritte der frühen, prototypenhaften Vorentwicklung mit den späteren, finalen Serienumsetzungen weitestgehend zu koppeln und aufeinander auszurichten.
Im von Bosch betrachteten Anwendungsfall des MIMO (multiple input multiple output) Radars werden als Hardware-Beschleuniger FPGA Komponenten von Xilinx verwendet, die den anfallenden Bitdatenstrom der AD-Wandler derart vorverarbeiten und reduzieren, dass eine Weiterverwendung auf einem nachfolgenden Laptop oder PC problemlos möglich wird.
Entsprechend dem DESERVE Plattform Entwicklungsgedanken führt der Entwicklungsverlauf im sogenannten Ko-Designverfahren zwischen Rapid Prototyping und System on Chip (SoC) über mehrere, verzahnte Entwicklungsstufen, wie auch in Abbildung 7 und Abbildung 23 dargestellt.

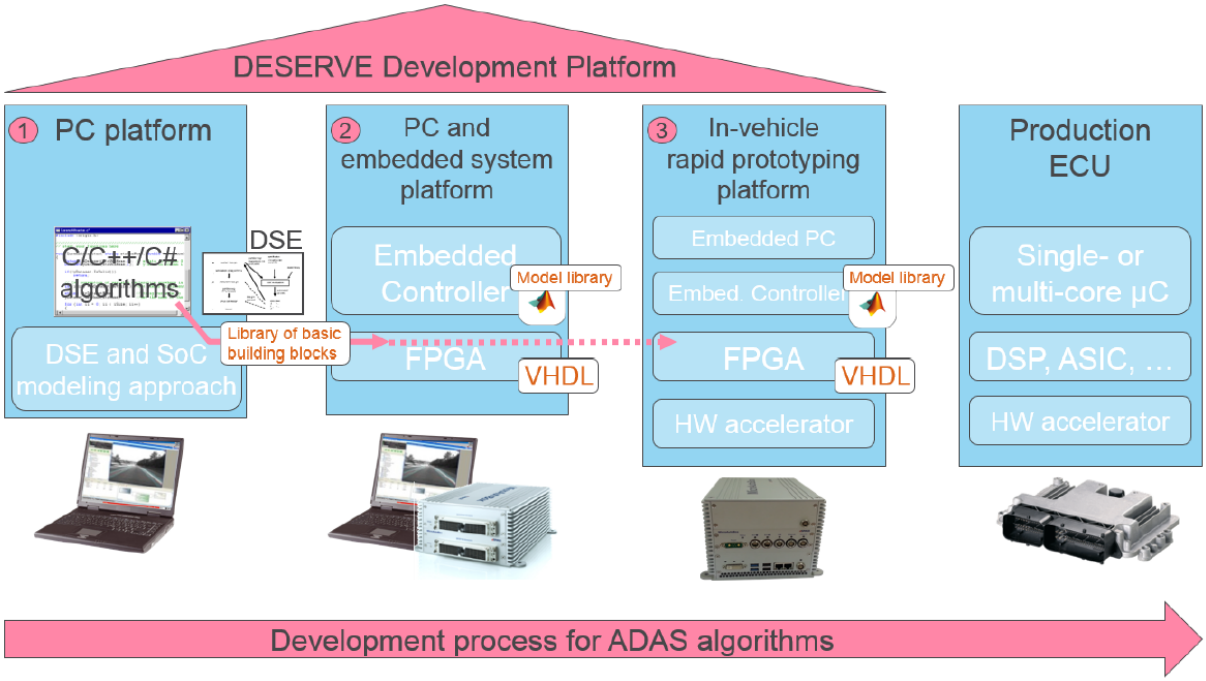


Abbildung 23: Der DESERVE Entwicklungsplattformgedanke (Quelle: DESERVE internal white paper)

Wie in Abbildung 23 ersichtlich, werden die mit DSE erzeugten Systemmodelle sukzessive von hochsprachigen PC-Plattformen über eingebettete Entwicklungsplattformen in die entsprechenden Serien-steuergeräte überführt.
Diesem Entwicklungsgedanken folgend wurde die von Bosch entwickelte MIMO Radar Prototypenplattform nach gleichen Gesichtspunkten gestaltet, wie in Abbildung 24 dargestellt.

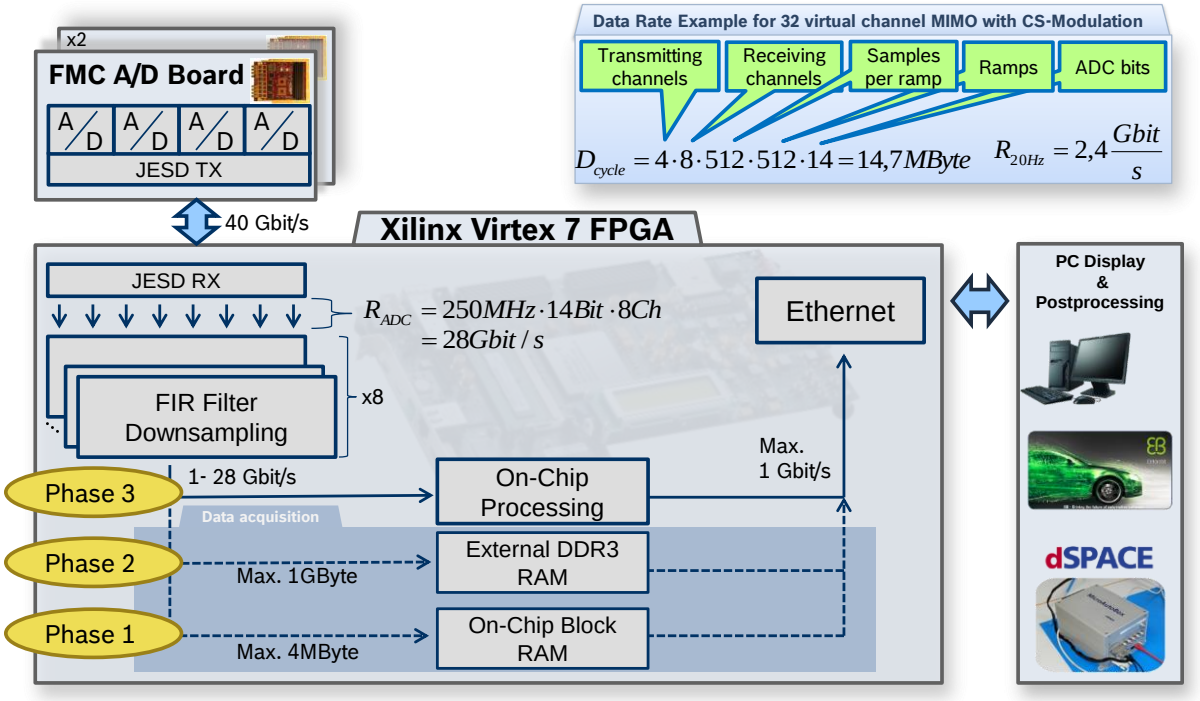


Abbildung 24: Die BOSCH MIMO Radar Entwicklungsplattform

Die Signalnachverarbeitung findet auf einem PC bzw. Laptop statt, der über eine 1 GBit/s Ethernet Anbindung mit dem Xilinx FPGA verbunden ist. Diese 1 Gbit/s Schnittstelle zum PC stellt auch das

„Nadelöhr“ zwischen der FPGA-basierten eingebetteten Plattform und der Darstellungs- bzw. Auswerteeinheit auf dem PC dar und zeigt den großen Nutzen des DESERVE Entwicklungskonzeptes, bei dem sukzessive Software-Algorithmik vom PC in die embedded Hardware verschoben werden kann.
Dies ist in Abbildung 24 durch die drei Datenpfade mit Phase 1 bis Phase 3 angedeutet, wobei in Phase 1 auf dem FPGA lediglich die Datenaufnahme in dem on-Chip SD-RAM erfolgt und die restliche Signalverarbeitung auf dem PC in ADTF erfolgt. Die Datenbitraten bei der Aufnahme (1-28 Gbit/s) ist dabei immer höher als der maximale Verbindungstransfer von 1 Gbit/s zum PC, was einen Echtzeitbetrieb des MIMO-Radars gänzlich unmöglich macht.
In Phase 2 werden die Daten einer kompletten MIMO Sequenz bereits im 1 GB DDR RAM des FPGA Entwicklungsboards zwischengespeichert und prozessiert, sodass hier jetzt zumindest ein vollständiger MIMO Zyklus in Echtzeit abgefahren werden kann.
In Phase 3 wird schließlich die Datenverarbeitung auf dem FPGA Board so weit vorangetrieben, dass aus den prozessierten Messdaten nur noch die relevanten Informationen extrahiert werden und diese dann problemlos in Echtzeit über die 1 GBit Ethernet Schnittstelle zum PC übertragbar sind. Die Datenreduktion geschieht über eine Schwellwertauswahl, die bei Radarsystemen typischerweise mittels eines CFAR (Constant False Alarm Rate) Algorithmus erfolgt, dessen Schwellwert situations-adaptiv angepasst werden kann. Somit ist man in der Lage, das MIMO Radar bereits im Prototypenstatus unter Echtzeitbedingungen betreiben zu können und kann schon sehr frühzeitig die notwendigen Randbedingungen für die spätere Serienimplementierung ausloten und festlegen.
Die vollständige Signalverarbeitungskette auf dem FPGA Entwicklungsboard ist in Abbildung 25 dargestellt.

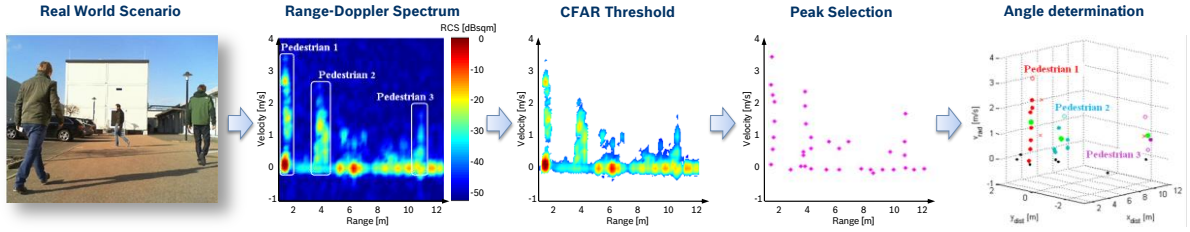


Abbildung 25: Implementierte Signalverarbeitungskette auf dem FPGA Entwicklungsboard

Der Einsatz von FPGAs ist für die primäre Datenvorverarbeitung von MIMO Radarsignalen ideal geeignet, da eine Vielzahl von Frequenztransformationen (FFT) für die große Anzahl an sich schnell wiederholenden Frequenzrampen für alle Empfangskanäle durchgeführt werden muss.
Die prinzipielle Chirp Sequence MIMO Radarsignalverarbeitung ist in Abbildung 26 dargestellt. Für jeden Frequenzchirp ist eine Frequenztransformation der aufgenommenen Signalwerte notwendig. Das Ergebnis liefert die Entfernungsinformation für jedes detektierte Objekt im Radar Erfassungsbereich. Eine weitere, orthogonale Frequenztransformation über die Anzahl der Frequenzchirps erschließt dann die Geschwindigkeit der bereits detektierten Objekte. Die Darstellungsform, wie in Abbildung 26 rechts skizziert, wird als „Range-Doppler Plot“ bezeichnet und kann zur Objektklassifikation (unter dem Stichwort „Mikro-Doppler Analyse“) herangezogen werden.
Im unteren Teil der Abbildung 26 sind die typischen Betriebswerte und Leistungsdaten des MIMO Radar-Prototypen aufgelistet. Mit steigender Messzeit erhöht sich die Mikro-Doppler Auflösungsfähigkeit, wobei die Ortsbestimmung aufgrund der Heisenberg‘schen Unschärferelation bei Szenen mit dynamischen Objekten sich in gleichem Maße verschlechtert. Für Automobilanwendungen sind daher typische Zykluszeiten für eine vollständige Chirp Sequenz bei ca. 20 ms, was einer maximalen Messwiederholrate von etwa 50 Hz entspricht.

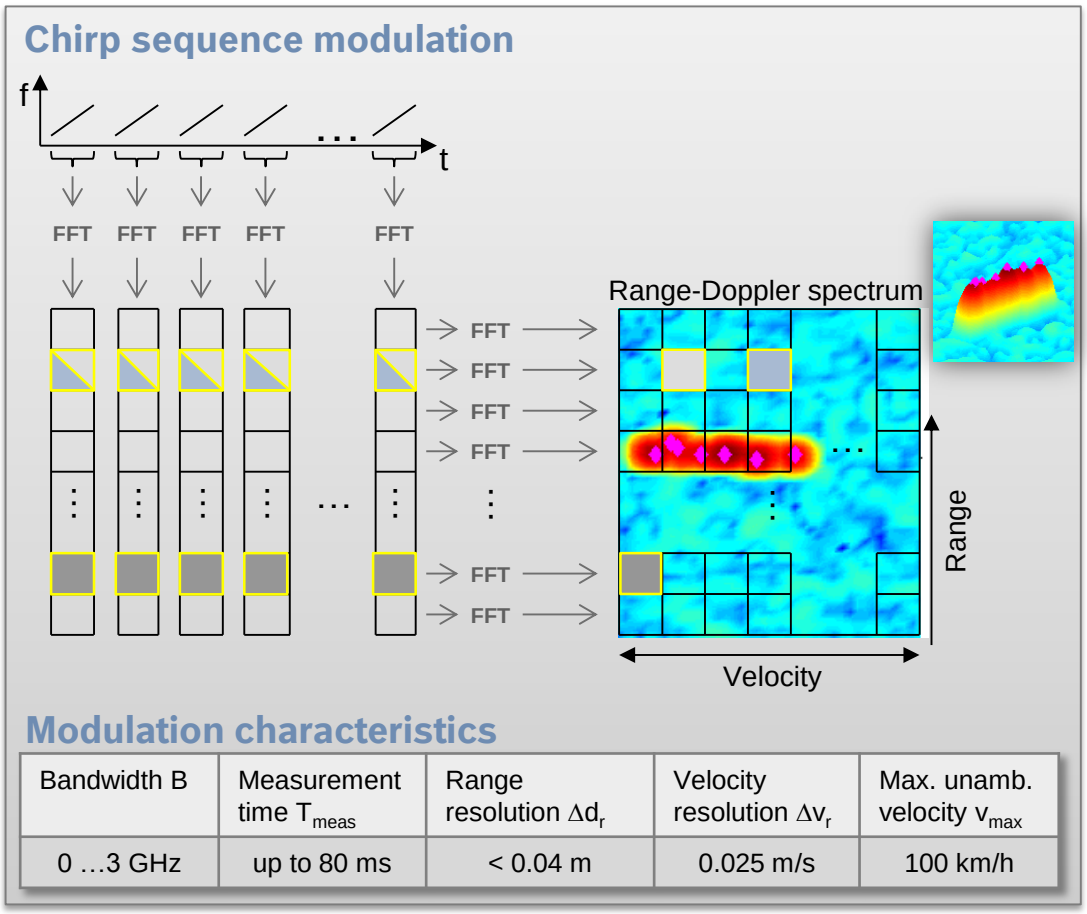


Abbildung 26: Funktionsweise und Betriebsparameter der Chirp Sequence (CS) Modulation

Für ein 4TX/8RX (i.e. 4 Send- und 8 Empfangskanäle) Radar sind für einen kompletten MIMO Zyklus mit z.B. 1000 Chirp Sequenzen und 1024 Abtastwerten pro Chirp-Sequenz insgesamt 32000 (für Range Dimension) plus 32768 (für Doppler Dimension) Frequenztransformationen zu berechnen. Bei einer 20 Hz Zyklusrate müssen diese FFTs unter Echtzeitbedingungen in 50 ms verarbeitet sein, d.h. eine FFT ist in weniger als 0,7 µs zu berechnen. Dies ist für den Virtex7 FPGA von Xilinx problemlos möglich, da die Datenströme der einzelnen Empfänger (RX) sich im FPGA sehr einfach parallelisieren lassen.
Bei entsprechender Reduktion der Abtastwerte oder der Chirp Sequenzen sind sogar noch höhere Zyklusraten möglich, was besonders bei der Auswertung von Mikro-Doppler Signalen von Vorteil ist.
In Abbildung 27 ist das Mikro-Doppler Spektrum von einem fahrenden Lieferwagen dargestellt.

- Bandwidth: 1.6 GHz
- Measurement time: 16ms
- Sampling frequency 80MHz
- 2048 samples and 512 ramps
- Update rate 45 Hz

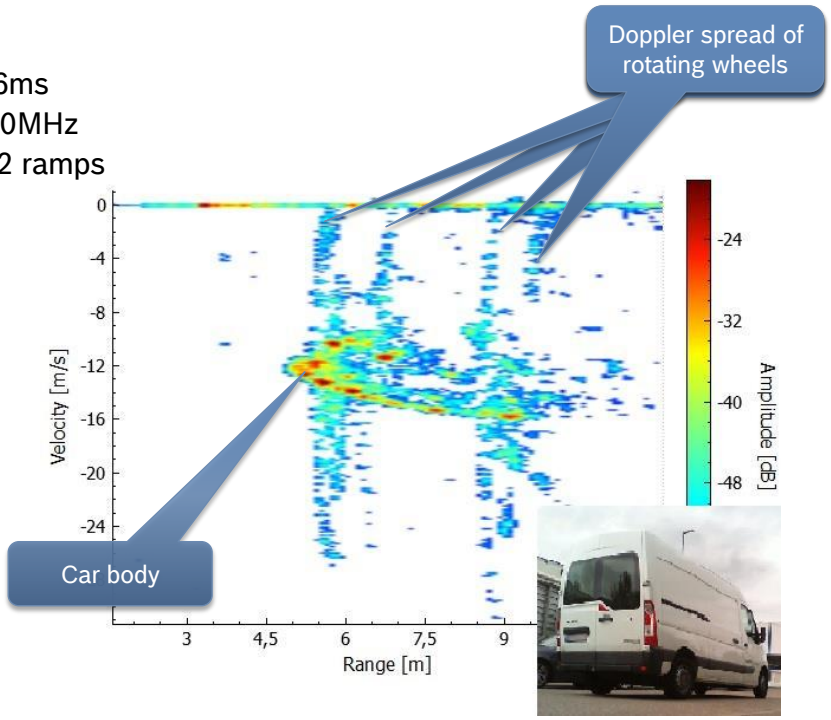


Abbildung 27: Mikro-Doppler Spektrum eines fahrenden Kleintransporters

Aufgrund der gewählten Betriebsparameter (1.6 GHz Chirp-Bandbreite, 16 ms Beobachtungszeit für die 512 Chirp-Rampen und einer Zyklusrate von 45 Hz) ist die Anzahl der detektierten Reflektionen vom Fahrzeug sehr hoch (> 200 Reflexpunkte), wobei auch die vier rotierenden Reifen wegen ihres charakteristischen Verhaltens (Geschwindigkeitsspektrum geht von 0 m/s bis zur doppelten Geschwindigkeit des Fahrzeugchassis) deutlich erkennbar sind.
Jedes Verkehrsobjekt hat im Mikro-Doppler Spektrum seine besonderen Merkmale, sodass ähnlich zur Videobildverarbeitung auch mit den Radarsensoren eine Objektklassifizierung durchgeführt werden kann. Objekte mit Achsen und drehenden Rädern bilden sich im Mikro-Doppler Spektrum als H-Struktur ab (für 2-achsige Fahrzeuge), während Fußgänger ein typisches, „raupenartiges“ Muster aufgrund der charakteristischen Beinbewegungen aufweisen.
Die typischen Mikro-Doppler Muster eines Fußgängers, Fahrradfahrers und PKWs sind in Abbildung 28 aufgeführt.

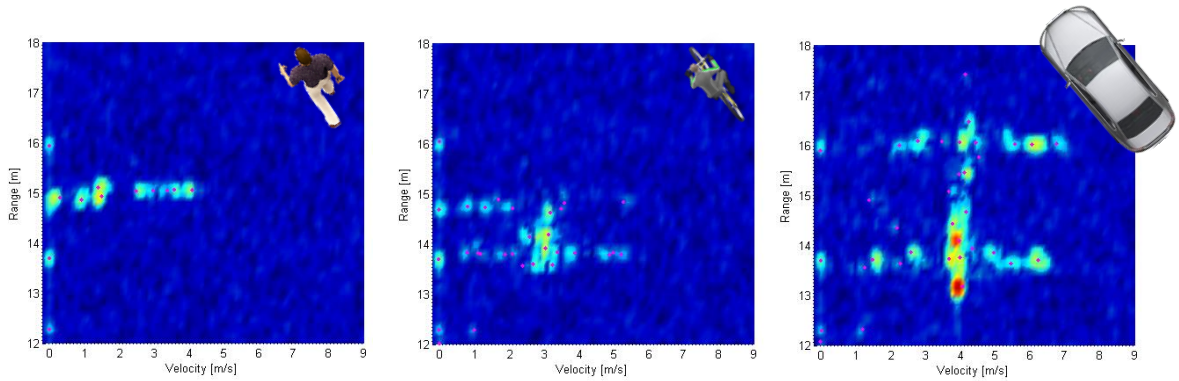


Abbildung 28: Mikro-Doppler Muster eines Fußgängers, Radfahrers und PKW

Der Vorteil der direkten Geschwindigkeitsmessung mittels Radarsensoren wird in Abbildung 29 deutlich. Da der Radarsensor die Rotation der Fahrradreifen direkt messen kann, wird das plötzliche blockieren des Hinterrades sofort erkannt und im Entfernungs-Doppler Diagramm unmittelbar dargestellt.
Sobald die Bremse wieder gelöst wird, ist auch wieder die Drehung des Hinterrads erkennbar.

- FPGA-based FFT and CFAR processing
- Range-Doppler plot at different time instances
- Velocity spread of wheels is clearly visible
- Locked rear wheel effect is immediately visible due to instantaneous speed measurement capability of microwave radar sensor

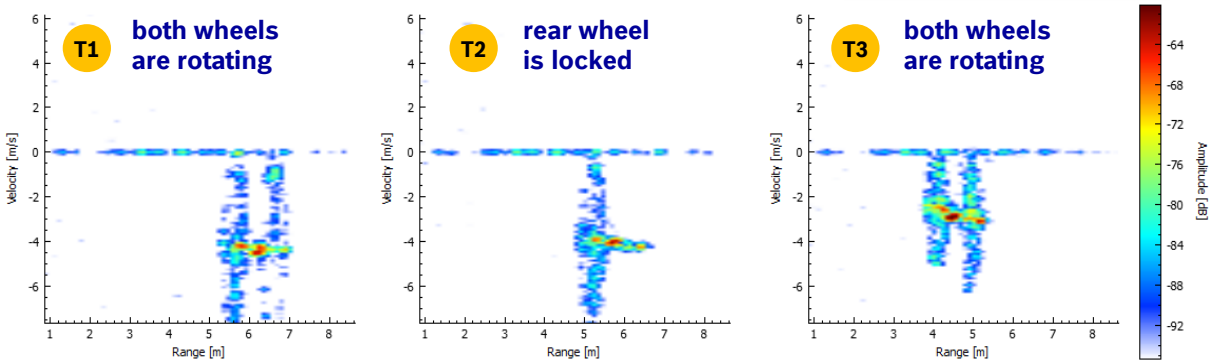


Abbildung 29: Direkte Geschwindigkeitsmessung mit Radarsystemen (Alleinstellungsmerkmal)

Der Versuch, eine ähnliche Präzision und Zeitbezug mit video-basierter Sensorik durchzuführen, würde sich als äußerst schwierig, wenn nicht sogar unmöglich, gestalten. Diese, nur den Radarsensoren innewohnende Eigenschaft der Mikro-Doppler Messung, kann daher als Alleinstellungsmerkmal für Automobilradare gesehen werden und führt besonders bei der Objektklassifikation und der dynamischen Szenenerkennung zu sehr robusten und verlässlichen Ergebnissen.

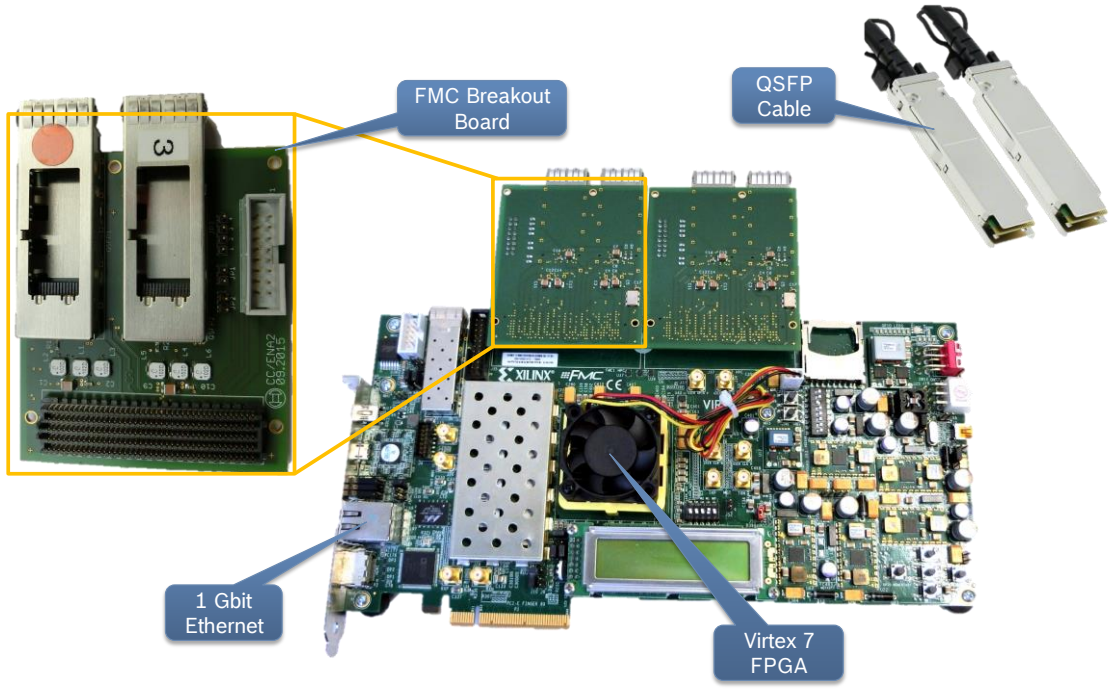


Abbildung 30: Die FPGA-basierte Signalverarbeitungs-Hardware

In Abbildung 30 sieht man die Hardware-Komponenten für die Signalverarbeitung des MIMO Radar Prototypen. Kernstück bildet ein Evaluierungsboard mit einem Virtex7 FPGA, 1 GByte DDR3 RAM und einer 1 GBit/s Ethernet Verbindung. Über die beiden FMC Erweiterungsstecker können die bereits im Radar-Frontend digitalisierten Signale via QSFP Kabel mit 5 Gbit/s pro Signalpfad eingelesen werden. Mit Doppelt-Leitungen je Kabelstecker ergibt sich bei den insgesamt 4 möglichen Kabelschächten eine maximale Datenübertragungsrate von 40 Gbit/s, was für ein 8TX/16RX MIMO Radar mit den in Abbildung 24 und Abbildung 26 spezifizierten Betriebsparametern gerade noch ausreichend ist.

In Abbildung 31 sind die beiden MIMO Front-Ends zu sehen, die flexibel mit der FPGA Recheneinheit über die entsprechenden QSFP-Kabel verbunden werden können. Selbst ein Wechsel zu anderen Front-End Designs ist über die realisierte Ausprägung der Schnittstelle über QSFP Kabel und dem entsprechenden JESD204 Protokoll jederzeit möglich.

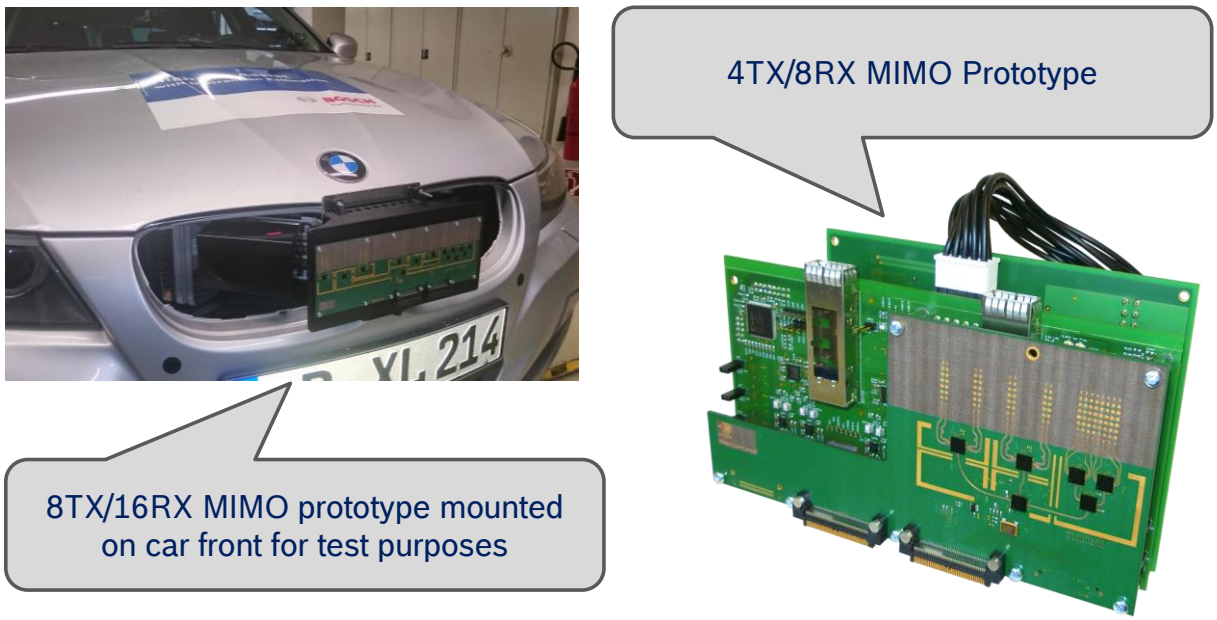


Abbildung 31: Die beiden MIMO Radar Front-Ends

Zusammenfassend lässt sich folgendes Resümee ziehen:

- Durch die konsequente Anwendung der in DESERVE entwickelten Methoden und Konzepte ist es gelungen, einen echtzeitfähigen MIMO Radarprototypen in kurzer Entwicklungszeit zu implementieren. Dies wäre mit konventionellem Vorgehen und etablierten Entwicklungsmethoden nicht oder nur mit sehr viel Mehraufwand möglich gewesen;
- Durch das rudimentär verwendete DSE (Design Space Exploration) ist es bereits in einer frühen Projektphase möglich, die später für die SoC Implementierung notwendigen Parametersätze zu definieren und festzulegen;
- Bereits zu einem sehr frühen Zeitpunkt kann die Gesamtperformance des später in Serie gehenden Systemkonzeptes abgeschätzt und getestet werden. Dies trifft insbesondere auf die Leistungsabschätzung der Mikro-Doppler Auswertung zu, die naturgemäß nur in Echtzeit bewertet werden kann, da kein zeitliches Herunterskalieren möglich ist;
- Mit dem DESERVE Projekt konnten erste Erkenntnisse und Erfahrungen auf dem Gebiet der zukünftigen System on Chip Entwicklungsmethodik und den damit verbundenen neuen Systemarchitekturansätzen gewonnen werden, die nutzbringen bei weiteren Zukunftsprojekten eingearbeitet werden können.

3.7 Beitrag Daimler Video Processing

Die Ursache vieler nächtlicher Verkehrsunfälle auf Landstraßen ist, dass Situationen viel zu spät erfasst werden, und so Fußgänger oder Hindernisse spät oder gar nicht erkannt werden. Daher ist das Hauptziel des Inter-Urban Assists, dass der Fahrer die vor ihm liegende Situation so gut und schnell wie möglich erfassen kann, sowohl hinsichtlich des weiteren Straßenverlaufs als auch die Aufmerksamkeit des Fahrers bei schlechter Sicht so früh wie möglich auf potentielle Hindernisse zu lenken. Eine Möglichkeit ist, dem Fahrer ein Video zu zeigen, und die erkannten Hindernisse oder Fußgänger farbig zu markieren und den weiteren Straßenverlauf anzuzeigen. Um diese Informationen zu liefern, sind dementsprechende Vorverarbeitungsschritte nötig. Diese Vorverarbeitung lässt sich beim Inter-Urban Assist unterteilen in Straßenverlaufserkennung und Scene Labeling, Selbstkalibrierung und 3D-Rekonstruktion.

3.7.1 Straßenverlaufserkennung und Scene Labeling

Für die in Kapitel 3.9 beschrieben Aufgaben im Inter-Urban-Assist wird eine präzise Straßenverlaufsprädiktion benötig. Für diese werden die Daten eines Fernbereich-Radars, einer digitalen Karte und einer Kamera fusioniert. Im Besonderen wurde für das bestehende Fusionssystem ein neues Straßenverlaufserkennungsmodul auf Basis von tiefen neuronalen Faltungsnetzen (CNN) implementiert. Da CNNs eine hohe Rechenkomplexität besitzen, wurden mithilfe der DESERVE Plattform prototypische Umsetzungen auf Grafikkarten und FPGAs (vgl. Kapitel 3.3.3) realisiert. Das Straßenverlaufserkennungsmodul besteht aus drei Komponenten:

1. Pixelklassifikation (Scene Labeling)
2. Straßenobjektbildung
3. Straßenformgebung und –tracking

Im Folgenden wird insbesondere auf die **Scene Labeling,** eine Teilkomponente des Straßenverlaufserkennungsmoduls eingegangen, da diese durch ein CNN realisiert wurde. Eine detaillierte Beschreibung des gesamten Frameworks findet sich in (Limmer, et al., 2016) [Ref18].

CNNs sind eine Erweiterung des Multi-Layer-Perzeptrons (MLP) um sogenannte Faltungs- und Poolingschichten. Anstatt alle Eingabewerte einer Schicht zu summieren (fully-connected), führt die Faltungsschicht eine Faltung auf den Eingabewerten aus, wobei die Faltungskerne die gelernten Gewichte darstellen. Dadurch kann einerseits die Größe der Eingabedaten variabel gestaltet werden und andererseits werden nur nahestehende Eingabewerte miteinander korreliert, was wiederum für Bilder als Eingabedaten durchaus sinnvoll ist. Poolingschichten bilden je nach Ausprägung das Maximum oder den Mittelwert über eine kleine Pixelnachbarschaft, wobei kein Pixel doppelt in die Berechnung miteinfließt. Sie werden einerseits genutzt um die Zwischenrepräsentationen herunter zu skalieren und andererseits um die Translationsinvarianz zu verbessern.
Die Scene Labeling Komponente wird durch ein tiefes multiskalares CNN realisiert. Dazu werden Methoden aus (Simonyan & Zisserman, 2014) [Ref 19] mit dem multiskalaren Ansatz von (Farabet, C., Couprie, Najman, & LeCun, 2013) [Ref 20] kombiniert. (Simonyan & Zisserman, 2014) beschreibt bestimmte Netzarchitekturen, die sich dadurch auszeichnen, dass sie aus vielen Faltungsschichten mit kleinen Faltungskernen und vergleichsweise wenigen Poolingschichten bestehen. Dadurch vergrößert sich die Anzahl der Nichtlinearitäten in einem Netzwerk und damit einhergehend dessen Kapazität komplexe Klassifikationsfunktionen zu lernen. Durch die Verwendung von kleinen Faltungskernen führt die vergrößerte Anzahl der Faltungsschichten nicht unbedingt zu einem drastischen Anstieg der Berechnungskomplexität. Die Verwendung mehrerer Skalen der Eingangsbilder verbessert zudem die Skaleninvarianz ohne dabei die Tiefe des Netzes zu vergrößern. Traditionelle Faltungsnetze arbeiten auf Bildausschnitten, die einen Bruchteil der Größe eines Eingangsbildes besitzen. Aufgrund der Echtzeitanforderung des Inter Urban Assist wurden zusätzlich die Methoden von (Giusti, Ciresan, Masci, Gambardella, & Schmidhuber, 2013) [Ref 21] und (Thom & Gritschneder, 2016) [Ref 22] umgesetzt, die eine effiziente Anwendung eines CNN auf das Gesamtbild ermöglichen.
Multiskalare neuronale Netze (Farabet, C., Couprie, Najman, & LeCun, 2013) prozessieren gleichzeitig mehrere Skalierungen der Eingangsdaten. Dazu muss eine Bildpyramide mit n_l Stufen berechnet werden, wobei für jede neue Stufe beide Bilddimensionen halbiert werden. Jede Pyramidenstufe wird dann lokal auf Mittelwert 0 und Einheitsvarianz normalisiert um Texturen hervorzuheben und helle und dunkle

Regionen im Bild zu egalisieren. Die normalisierten Pyramidenstufen werden jeweils von einem separaten Zweig des Netzes verarbeitet und schlussendlich in einer fully-connected Schicht kombiniert. Abbildung 32 zeigt eine schematische Darstellung der möglichen Netztopologien.

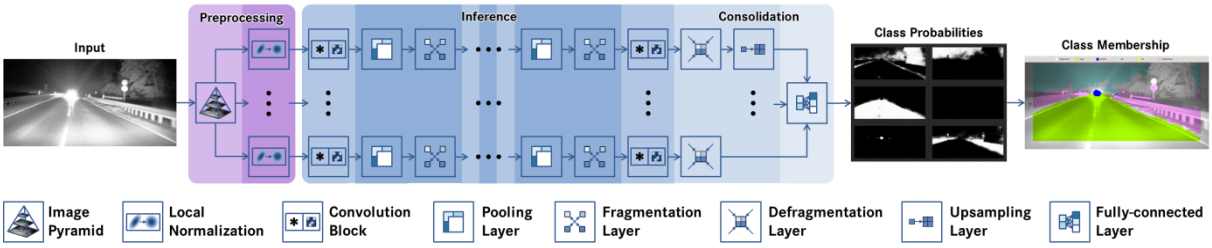


Abbildung 32: Schematische Darstellung von möglichen Netztopologien

Eine Vorverarbeitung erzeugt eine normalisierte Bildpyramide mit n_l Pyramidenstufen. Jede Stufe wird in ihrem eigenen CNN Zweig weiterverarbeitet. Ein Zweig besteht aus alternierend Faltungsblöcken und Pooling-/Fragmentierungsschichten. Die fragmentierten Merkmalskarten aller Zweige werden defragmentiert, hochskaliert und in einer fully-connected Schicht zusammengeführt. Dadurch wird eine Wahrscheinlichkeitskarte für jede trainierte Klasse generiert, aus der eine Klassenzugehörigkeitskarte (Pixelklassifikation) berechnet werden kann.

Jeder Zweig ist strukturell identisch, sie teilen sich aber keine Gewichte. Sie bestehen alternierend aus *Faltungsblöcken* und Poolingschichten. Ein Faltungsblock besteht aus n_c Faltungsschichten inklusive ihrer Aktivierungsfunktion (Nichtlinearität). Als Aktivierungsfunktion wurde die ReLU Funktion gewählt: $\mathrm{ReLU}(x) = \max(0, x)$. Die Poolingschichten führen jeweils ein $k_p = 2 \times 2$ max-Pooling durch.
Für eine bildausschnittbasierte Anwendung des CNNs muss für jede Pixelposition ein passender Ausschnitt aus den jeweiligen Pyramidenstufen extrahiert werden, bevor dieser verarbeitet werden kann. Da dadurch viele redundante Berechnungen ausgeführt würden, müssen ein paar Veränderungen an der Struktur des Netzes geändert werden um eine bildbasierte auflösungserhaltende Ausführung zu ermöglichen (vgl. (Giusti, Ciresan, Masci, Gambardella, & Schmidhuber, 2013) (Thom & Gritschneder, 2016)). Diese werden im Folgenden erläutert:
1) Überlappendes Pooling: Poolingschichten besitzen im Allgemeinen eine Sprungweite ($\mathrm{stride} > 1$), die meist der Größe des Poolingfensters entspricht. Um keine Auflösung zu verlieren muss das Pooling mit $\mathrm{stride} = 1$ durchgeführt werden. Das nennt sich überlappendes Pooling.
2) Fragmentierung: Fragmentierungsschichten müssen nun nach jeder (überlappenden) Poolingschicht eingefügt werden. Diese teilen die Merkmalskarten so auf, dass $\prod stride_{k_p}$ Merkmalskarten mit geringerer Auflösung entstehen um die Auflösungsreduktion der ursprünglichen Poolingschichten zu emulieren. Die Fragmente werden im Weiteren separate prozessiert. Abbildung 33 zeigt eine solche 2×2 Fragmentierung.

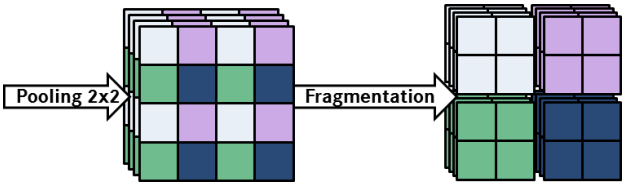


Abbildung 33: Eine 2×2 Fragmentierung

3) Defragmentierung: Sofern von Fragmentierungsschichten Gebrauch gemacht wurde, muss nach dem letzten Faltungsblock eine Defragmentierungsschicht eingeführt werden. Diese kehrt die vorhergehenden Fragmentierungen um und erzeugt dadurch kohäsive Merkmalsbilder mit der ursprünglichen Auflösung der korrespondierenden Pyramidenstufe.
4) Hochskalierung: Nach der Defragmentierungsschicht müssen die Merkmalsbilder der kleineren Pyramidenstufen hochskaliert und eventuell beschnitten werden, dass alle Merkmalsbilder die gleiche Auflösung besitzen. Dadurch können diese aneinandergehängt und als Eingangsdaten für die folgende fully-connected Schicht verwendet werden.
5) Fully-connected Schicht als Faltung: Die ursprüngliche fully-connected Schicht muss für die bildbasierte Anwendung in eine Faltungsschicht umgewandelt werden, da fully-connected Schichten keine dynamischen Bildgrößen unterstützen. Die Gewichte der ursprünglichen Schicht werden dadurch in Faltungskerne überführt, wobei die Anzahl der Kanäle eines Kerns der Anzahl der Merkmalsbilder entspricht.

Die oben beschriebenen Netze wurden mit etwa 6800 gelabelten Bildern aus der in Kapitel 4.10. beschriebenen Kamera trainiert. Der Datensatz umfasst Szenen mit verschiedenen Straßenbeschaffenheiten, Wettersituationen und Jahreszeiten um die Robustheit der Klassifikation zu verbessern. Es wurden diverse Topologien trainiert und auf die jeweilige Zielhardware (Grafikkarte, FPGA) angepasst. Details zum Training und der Performance des Gesamtsystems findet sich in [Ref 18].

3.7.2 Selbstkalibrierung und 3D-Rekonstruktion

Da die Augmentierung dem Fahrer ermöglichen soll, die vor ihm liegende Situation schnell zu erfassen, sollte diese auch die Entfernung der erkannten Objekte beinhalten. Solche Entfernungswerte können mit Stereoverfahren berechnet werden. Um die Reaktionszeit des Fahrers zu begünstigen, sollten Objekte bereits in weiter Entfernung vor dem Fahrzeug erkannt und auch augmentiert werden können. Die Reichweite eines Stereosystems wird von der Basisbreite bestimmt. Um also Objekte in weiterer Entfernung erkennen zu können, ist also eine große Basisbreite nötig.
Da es mit wachsender Basisbreite eines Stereosystems oft nicht mehr praktikabel ist, die beiden Kameras in mechanisch starr verbundener Weise anzubringen, können die Kameras individuell angebracht werden, z.B. an der Windschutzscheibe.
Bekannte Stereo-Algorithmen setzen jedoch die genaue Kenntnis der intrinsischen (z.B. fokale Länge) und extrinsischen Kameraparameter (die Transformation zwischen den beiden Kameras) voraus, denn Kalibrierfehler führen zu fehlerhaften Werten in der Tiefenrekonstruktion. Die Kameraparameter sind die Grundlage für die Rektifizierung, die die beiden Bildebenen auf eine gemeinsame Ebene transformiert und so die Basis für die weitere Verarbeitung bildet.
Die intrinsischen Kameraparameter sind konstant und können durch eine Offline-Kalibrierung festgestellt werden. Da die Kameras aber nicht starr mechanisch verbunden sind, können die extrinsischen Kameraparameter variieren und sich so über die Zeit oder auch rasch von Bild zu Bild verändern. Daher ist eine einmalige Offline-Kalibrierung nicht ausreichend um den Anforderungen für die weitere Stereo-Verarbeitung zu genügen. Um diese Anforderungen zu erfüllen ist eine schritthaltende Online-Kalibrierung nötig.
Da der Gebrauch von Kalibrierobjekten mit bekannter Geometrie während der Fahrt nicht umsetzbar ist, ist ein Selbstkalibrierungsmechanismus nötig. Online-Selbstkalibrierung bedeutet, auf Basis der Bilder der beiden Kameras auf die Kameraparameter zu schließen. Deshalb ist ein nötiger Vorverarbeitungsschritt die Extraktion von Szenenpunkten, die in beiden Kamerabildern sichtbar sind. Da die extrahierten Punktpaare bestimmten Anforderungen an Genauigkeit genügen müssen, kann dieser Schritt sehr rechenaufwendig sein.

Schritthaltende extrinsische Selbstkalibrierung
Die extrinsischen Parameter eines Stereosystems sind durch eine Rotation $\mathrm{R_X} \in \mathrm{SO}(3)$ und eine Translation $\mathrm{t_X} \in \mathbb{R}^3$ beschrieben. Die Transformation eines Punktes $\mathrm{X_l} \in \mathbb{R}^3$ im Koordinatensystem der linken Kamera in das Koordinatensystem der rechten Kamera ist beschrieben durch

$$\mathbf{X}_\mathrm{r} = \mathrm{R_X}(\mathbf{X_l} - \mathrm{t_X}).$$

Üblicherweise beinhaltet eine extrinsische Stereo-Kamera-Kalibrierung die Berechnung von $\mathrm{R_X}$ und $\mathrm{t_X}$.Im Folgenden wird $\mathrm{t_X}$ als konstant angenommen und nur $\mathrm{R_X}$ wird geschätzt.
Während der Rektifizierung wird $\mathrm{R_X}$ zerlegt in

$$\mathrm{R_X} = \mathrm{R}_r^{-1} R_l$$

um die Rotation für die rechte und linke Kamera auf die gemeinsame Bildebene separate zu bestimmen. Da angenommen werden kann, dass die Dekalibrierung sehr klein ist, kann die Re-Kalibrierung auf vorrektifizierten Bildern aufbauen Die Vorrektifizierung kann mit den Parametern der Offline-Kalibrierung oder mit den Parametern eines vorherigen Kalibrierzyklusses durchgeführt werden.

Gegeben N korrespondierende vorrektifizierte Bildpunkte $\tilde{\mathrm{P}}_i$ und $\tilde{\mathrm{Q}}_i$ mit $i = 1, \dots, N$, können mit einer Kamera-Matrix K als Einheitsvektoren berechnet werden

$$\tilde{p}_i \cong \mathrm{K}^{-1}\tilde{\mathrm{P}}_i$$
$$\tilde{q}_i \cong \mathrm{K}^{-1}\tilde{\mathrm{Q}}_i \, .$$

Diese beiden Vektoren sind durch die Epipolargleichung verknüpft

$$0 = \tilde{\mathrm{Q}}_i \mathrm{K}\, \check{\mathrm{R}}\, \mathrm{K}^{-1}\tilde{\mathrm{P}}_i$$

wobei $\check{\mathrm{R}}$ die Rotation ist, die die Dekalibrierung kompensiert.

Da die Dekalibrierung als sehr klein angenommen wird, liegt diese Matrix nahe der Einheitsmatrix. Um eine Überanpassung zu vermeiden, werden die Bildvektoren in das ursprüngliche Kamera-Koordinatensystem transformiert durch

$$\mathrm{p}_\mathrm{i} = \mathrm{R}_\mathrm{l}\tilde{\mathrm{p}}_\mathrm{i}$$
$$\mathrm{q}_\mathrm{i} = \mathrm{R}_\mathrm{r}\tilde{\mathrm{q}}_\mathrm{i}\,.$$

Diese beiden Vektoren können auf die jeweiligen Bildebenen projiziert werden durch

$$\mathrm{P}_i = \mathrm{K}\,p_i$$
$$\mathrm{Q}_i = \mathrm{K}\,q_i.$$

Ein Bildpunkt Q_i kann, gegeben den gemessenen Bildvektor p_i, die Tiefe d_i des Weltpunktes $\mathbf{X}_\mathrm{l}$ und die Dekalibrierung $\check{\mathrm{R}}$, modelliert werden durch

$$\mathrm{Q}'_i(\check{\mathrm{R}}, \mathrm{d}_i) = \mathrm{K}\,\check{\mathrm{R}}\mathrm{R}_\mathrm{X}(\left(p_i \mathrm{d}_i\right) - \mathrm{t}_\mathrm{X}).$$

Da es aufgrund von Messrauschen hier keine exakte Lösung gibt, sollte die Zielfunktion den Rückprojektionsfehler, also den Fehler zwischen gemessenem und modelliertem Bildvektor minimieren zu

$$e_i = \left\|\mathrm{Q}_i - \mathrm{Q}'_i(\check{\mathrm{R}}, \mathrm{d}_i)\right\|.$$

Unter der Berücksichtigung aller Punktkorrespondenzen wird die Zielfunktion so formuliert, dass die Summe über alle Rückprojektionsfehler minimiert wird:

$$\underset{\check{\mathrm{R}},\mathbf{d}}{\operatorname{argmin}} \sum_{i=1}^{N} e_i^2$$

mit $\mathbf{d} = [\mathrm{d}_1 \ldots \mathrm{d}_N]$. Die Lösung kann durch eine nicht-lineare Optimierung gefunden werden, z.B. nach Levenberg-Marquardt.

Dieser schritthaltende Selbstkalibrierungsmechanismus bildet die Basis für die folgende Rektifizierung, die eine Voraussetzung für die Berechnung der dichten Tiefenkarte der Szene ist. Aus der Tiefenkarte der Szene können dann die Tiefenwerte einzelner erkannter Objekte extrahiert und für die Augmentierung genutzt werden.

3.8 Beitrag ika - Regelungskonzepte

Im Zuge des DESERVE-Projekts hat das Institut für Kraftfahrzeuge (ika) eine Regelungsfunktion entwickelt, die den Fahrer im Bereich des Überlandverkehrs bei der Längs- und Querführung des Fahrzeugs unterstützt. Hintergrund der Entwicklung dieser Funktion innerhalb des Projekts ist der Nachweis über die Vorteile, insbesondere des reduzierten Aufwands durch Wiederverwendung bereits existierender Softwarebausteine und Hardwarekomponenten, die die Plattform bei der Entwicklung von Fahrerassistenzsystemen bietet. Zudem wurde eine Methode entwickelt, mit der ein Fahrerassistenzsystem oder eine Funktion des automatisierten Fahrens früh im Entwicklungsprozess unter realistischen Bedingungen und unter bestimmten Gesichtspunkten in der Simulation getestet, bewertet und teilweise sogar validiert werden kann. Dazu entwickelte das ika in einem weiteren Teilprojekt ein Fahrermodell (siehe Kap. 4.8.2) für die Verkehrssimulation PELOPS, dass diese um die Fähigkeit erweitert, Fahrerassistenzsysteme im Bereich des Überlandverkehrs, insbesondere auf Kreuzungen zu testen und zu bewerten. Durch die Erweiterung der Simulation durch das Fahrermodell und der Einsatz dieser in der Entwicklung der Regelungsfunktion, wird die DESERVE-Werkzeugkette vervollständigt. Die Anwendung der Regelungsfunktion auf ein virtuelles Fahrzeug in der Simulation mit realistischem Umgebungsverkehr und realistischer Verkehrsumgebung (Streckenmodell) wurde auf der Abschlussveranstaltung des Projekts im Dezember 2015 in Ulm erfolgreich demonstriert.

3.8.1 Die Überland-ACC/LK-Regelungsfunktion mit Safe-Passing-Funktionalität

Die Regelfunktion ist in der Lage, den Fahrer bei der Umsetzung von freiem Fahren und Folgefahren sowie beim sichern Passieren von Hindernissen zu unterstützen. Dabei wird sowohl die Längs- als auch die Querführung unterstützt. Anders als bei Fahrerassistenzsystem für Manöver mit höheren Geschwindigkeiten, die zurzeit in Serienfahrzeugen verbaut sind (ACC, Lane-Keeping), berechnet die hier entwickelte Funktion eine Trajektorie für die nächsten Sekunden und regelt die Längs- und Querführung des Fahrzeugs auf diese Trajektorie ein. Somit lassen sich komplexere Manöver, wie das Passieren von

Objekten sicher und vorausschauend bewältigen. Längs- und Querführung, also insbesondere die Geschwindigkeit und die Position innerhalb des Fahrstreifens sind aufeinander abgestimmt und optimieren das Fahrverhalten hinsichtlich Sicherheit, Komfort und Effizienz.
Die Funktion wurde in Matlab/Simulink implementiert und auf einer MicroAutobox II, die die Basis für eine der DESERVE-Hardware-Plattformen bildet, installiert. Die MicroAutobox II wurde in dem virtuellen Fahrzeug einer PELOPS-Simulation integriert und getestet.
Die Trajektorienplanung und die Regelung sind die zentralen Elemente für dieses System, da diese das Verhalten maßgeblich vorgeben. Die meisten zusätzlich benötigten Elemente, wie die Perzeption, die für die Aufbereitung und erste Verarbeitung der Sensordaten, sowie HMI-Eingaben zuständig ist und die Umsetzung der Regelung (Aktoren und HMI-Ausgaben), werden durch Standardkomponenten der DESERVE-Plattform durchgeführt. Diese Elemente sind in der „Perception platform“ und in der „IWI platform“ in Abbildung 34 dargestellt. Die eigentliche – im Projekt implementierte – Anwendung ist in der „Application platform“ untergebracht. Dabei werden zunächst die relevanten Objekte identifiziert und klassifiziert (als Folgeobjekt oder zu passierendes Objekt), darauf und auf den Umgebungsdaten basierend wird eine Trajektorie geplant und die für den Fahrer relevanten Informationen auf dem Display angezeigt. Die Trajektorie und der Fahrzeugzustand werden an die Regelung weitergeben.

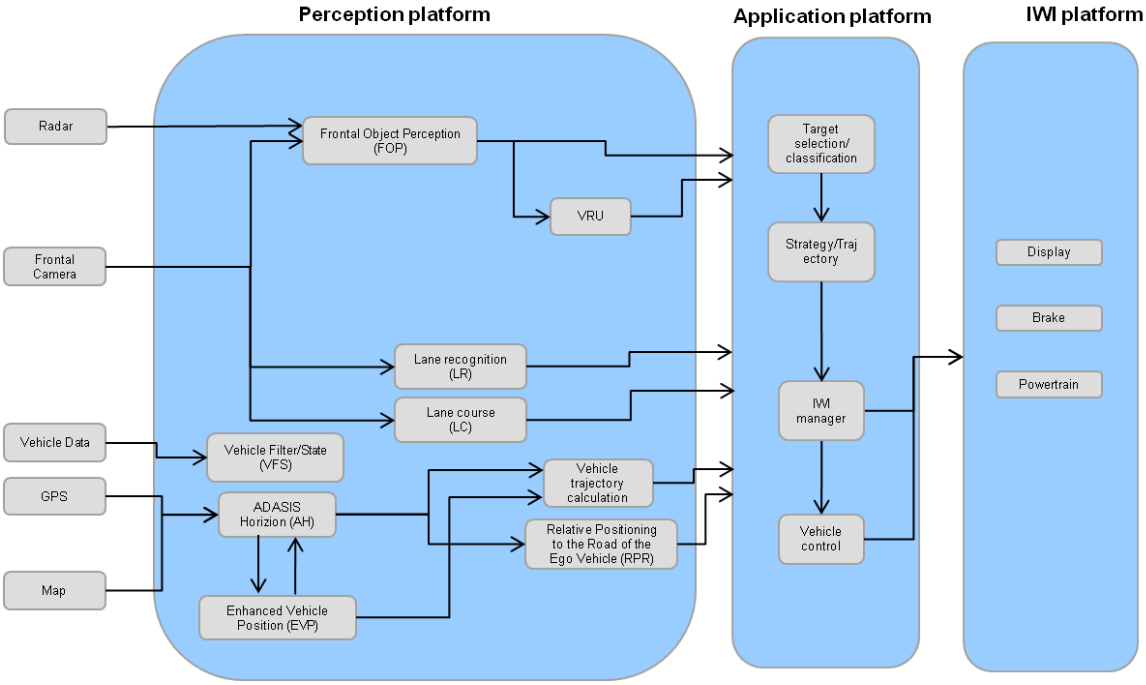


Abbildung 34: Integration der benötigten Bausteine für die Funktion in der DESERVE Architektur

Die Trajektorienplanung generiert einen Pfad (longitudinale Position, lateral Position im Fahrstreifen, Richtungswinkel des Fahrzeugs) mit entsprechendem Geschwindigkeitsprofil. Dabei wird der Pfad dahingehend optimiert, dass Faktoren wie reduzierte Querbeschleunigung, Sicherheitsabstand zu Fußgängern und zu passierenden Objekten, kürzester Weg bzw. reduzierte Reisezeit in einer definierten Kostenfunktion berücksichtigt werden. Zusätzlich wird die Geschwindigkeit der aktuellen Situation angepasst. Zum Beispiel wird die Geschwindigkeit in Kurven reduziert, um Komfort und Sicherheit zu gewährleisten und dem Fahrer ein realistisches Fahrerverhalten vorzugeben. Dies führt in bestimmten Situationen zu einem Dilemma, welches es zu lösen gilt: Die Wahl der Geschwindigkeit hängt von der Wahl des Pfades ab, welcher wiederum durch die zeitabhängige Position des Fahrzeugs, also von der Geschwindigkeit beeinflusst wird. Um dieses „Henne-Ei-Problem“ zu verdeutlichen, soll ein Beispiel herangezogen werden: Ein langsam fahrendes Fahrzeug (z. B. ein Mofa, Zielfahrzeug genannt) soll sicher passiert werden. Der Pfad, der um das Zielfahrzeug herum berechnet wird, beeinflusst das Geschwindigkeitsprofil, das gewählt wird, um dem Ego-Fahrer und dem Fahrer des Zielfahrzeugs ein angemessenes Sicherheitsgefühl zu geben und den Komfort zu erhöhen. Durch die Reduktion der Geschwindigkeit verlängert sich der Weg des Passierens, da sich das Zielfahrzeug unter Umständen mit unveränderter Geschwindigkeit weiter fortbewegt. Dadurch muss sich offensichtlich der Pfad an den längeren Weg anpassen (einscheren), was wiederum die Geschwindigkeit beeinflusst. Dieser Effekt wird insbesondere bei der Betrachtung von kurvigen Straßen und bei Beeinflussung von vorausfahrenden Fahrzeugen kompliziert, da hier das Geschwindigkeitsprofil nicht konstant ist. Um das Problem zu lösen, wird zunächst ein Geschwindigkeitsprofil berechnet, welches auf der Krümmung der Straße, der vorgegeben Geschwindigkeitsbegrenzung, der Vorgabe des Fahrers und eines Standard ACC-Verhaltens

(ISO 22179:2009) basiert. Die Querführung bleibt dabei zunächst unberücksichtigt. Durch das abgeschätzte Geschwindigkeitsprofil, lässt sich eine zeitabhängige longitudinale Position berechnen. Zu der longitudinalen Position wird eine optimierte Querablage zur Fahrstreifenmitte berechnet, die die folgenden Faktoren berücksichtigt: Sicher Abstand zu Hindernissen durch ein Potentialfeld, dass die Gefahr über die Fahrstreifenbreite abbildet, sicherer Abstand zu entgegenkommenden Fahrzeugen, Abstand zu den Fahrstreifenrändern, reduzierte Krümmung und Krümmungsänderung und damit eine reduzierte Querbeschleunigung und ein reduzierter Querruck. Da die Querablage an situationsabhängigen Stützstellen generiert wird, entsteht eine diskrete Beschreibung des optimalen Pfads, der unter Umständen große Lücken ausweist. Um eine kontinuierliche Kurve zu erhalten, wird eine Bézierkurve auf Basis der Punkte berechnet. Diese Kurve hat den Vorteil gegenüber C2-Splines, dass die Punkte nicht zwingend auf der Kurve liegen, sondern die Kurve sich möglichst glatt ausbildet, dabei jedoch von den Stützstellen stark „angezogen“ wird, also bei moderaten Krümmungsverlauf und ausreichend geringem Abstand der Punkte nur geringfügig von diesen abweicht. Der Charakter der Kurve lässt sich gut durch die Dichte der Stützstellen steuern. Dadurch wird eine glatte Kurve generiert, die durch das Regelungsmodul sehr gut eingeregelt und die durch ein reales Fahrzeug, oder in diesem Fall ein Fahrzeugmodell mit realistischer Fahrzeugdynamik, befahren werden kann. Der zeitabhängige Pfad und das Geschwindigkeitsprofil ergeben in Kombination eine Trajektorie, aus der fahrzeugtechnisch relevante Werte, wie Geschwindigkeit, Krümmung und Gierwinkel, abgeleitet werden, welche im Regelungsmodul durch einen kaskadierten Regelungsansatz in eine Beschleunigung und einen Lenkwinkel umgerechnet werden.
In Abbildung 35 oben ist eine Trajektorie für das Durchfahren einer Kurve dargestellt und in der unteren Darstellung die Trajektorie für sicheres Passieren eines geparkten Fahrzeugs. Es ist erkennbar, dass die Kurve „geschnitten“ wird, also eine Querablage in das Kurveninnere berechnet wird. Dies ist durch den Einfluss von absoluter Krümmung und Krümmungsänderung sowie Reisezeit und zurückgelegter Weg auf die Kostenfunktion bedingt – Kurvenschneiden reduziert i. d. R. die Krümmungsänderung und die absolute Krümmung, sowie den zurückgelegten Weg. In der unteren Darstellung ist erkennbar, dass das System einen Sicherheitsabstand zum geparkten Fahrzeug einstellt, der durch die Potentialfelder bedingt ist. Zu erkennen ist zudem, dass die Trajektorien-Planung zu Beginn der dargestellten Situation intensiv reagiert, da sich das Egofahrzeug bereits nah am Passierobjekt befindet, jedoch noch keinen Sicherheitsabstand aufgebaut hat (Anfangsbedingung des Beispiels), dennoch aber den Weg in die Fahrbahnmitten (einscheren) großzügig wählt, um die Bedingungen nach geringer Krümmung und Krümmungsänderung zu erfüllen.

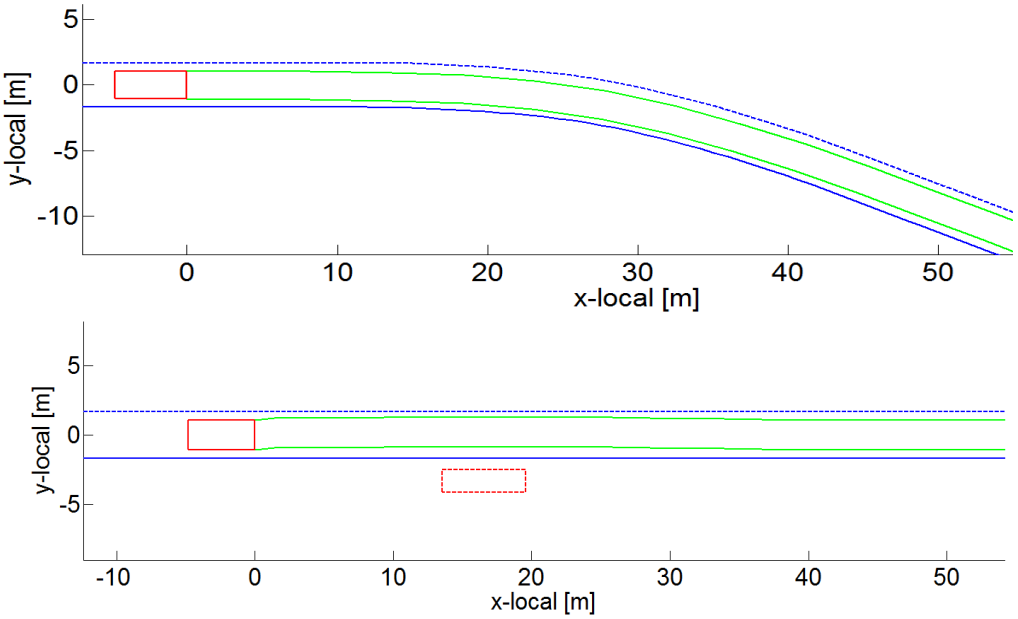


Abbildung 35: Trajektorien für eine Kurvenfahrt (oben) und für das sichere Passieren eines geparkten Fahrzeugs (unten)

Die Funktion wurde im Zuge des Projekts auf einer MicroAutobox II implementiert und in der Simulation unter realen Bedingungen (realistische Strecke, realistischer Verkehr) getestet. Dabei wurde das Fahrerverhalten des virtuellen Egofahrers von dem Assistenzsystem überlagert. Die Funktion zeigt in dieser Simulationsanwendung sehr gute Ergebnisse und kann in Folgeprojekten für den Einsatz im Realfahrzeug optimiert werden.

3.8.2 Das Fahrermodell

Die oben beschriebene Anwendung der Regelungsfunktion und das wissenschaftliche Testen dieser in der Simulation setzt voraus, dass diese den Realverkehr realistisch abbildet. Dies ist nur möglich, wenn entsprechende Modell eingesetzt werden, die die Interaktionen von Realfahrern beschreiben und dabei die Variation unter diesen berücksichtigt. Dazu werden verschiedene Simulationslevel verwendet. Die Untersuchung von Verkehrsdynamikeffekten auf einem groben Straßennetz, wie Stauwellen und Anwendungen bei denen grobe verkehrsspezifische Messgrößen wie Verkehrsdichte und Durchschnittsgeschwindigkeiten von Bedeutung sind, werden in der Regel in makroskopischen Simulationen durchgeführt. Dabei wird nicht das einzelne Fahrzeug mit dem Interaktionsverhalten als Element abgebildet, sondern ein fluiddynamischer Ansatz gewählt, der Verkehr als fließendes Medium abstrahiert. Dennoch bietet sich auch für die makroskopische Untersuchung von Verkehrseffekten der Ansatz an, die Fahrzeuge einzeln als Fahrer-Fahrzeug-Einheit (ein fusioniertes Element) abzubilden. In diesem Fall redet man von einer mikroskopischen Verkehrssimulation. Die Vorteile sind dabei, dass einerseits interaktionsspezifische Verhaltensmuster realistischer abgebildet werden können und die Untersuchung auf einer Situationsebene stattfinden kann, also Konstellationen von Fahrzeugen und deren Reaktion aufeinander abgebildet werden können. Dabei werden jedoch häufig die Fahrzeugdynamik und die Interaktion des Fahrers mit seinem Fahrzeug ungeachtet gelassen, da die Komponenten des Fahrzeugs in der Regel gar nicht oder sehr rudimentär implementiert sind. Üblicherweise wird die Dynamik des Fahrzeugs, insbesondere das begrenzte Beschleunigungsvermögen durch eine geschwindigkeitsabhängige Beschränkung des Fahrerwunsches berücksichtigt und dieser zumeist direkt als Beschleunigung des Fahrzeugs gesetzt. Submikroskopische Verkehrssimulationen bilden die Interaktion zwischen Fahrer und Fahrzeugkomponenten, wie Fahrpedal, Lenksystem und Bremssystem ab. Dabei wird i. d. R. das Fahrzeug aus dem Zusammenspiel seiner Komponenten modelliert, sodass mindestens die Kraftübertragung vom Motor zur Straße modelliert wird und somit die resultierende Beschleunigung bei gegeben Fahrpedalwert eingestellt wird. Dazu muss das Fahrermodell natürlich in der Lage sein, einen Pedalwert einzustellen, der seinen Wunsch umsetzt. Zudem werden in einigen Verkehrssimulationswerkzeugen die Querführung und teilweise sogar die Interaktion mit den Fahrzeugsystemen über ein virtuelles HMI abgebildet. Dadurch lassen sich Effekte wie Fahrereingriffe beim teilautomatisierten Fahren abbilden, z. B. das Überstimmen eines ACC-Systems durch den Fahrer oder das simple ein- und ausschalten dieser Systeme. Dieses ist unter Umständen nicht nur für das Egofahrzeug notwendig. Soll zukünftiger Verkehr realistisch abgebildet werden, müssen FAS und Automatisierungsfunktionen in den virtuellen Fahrzeugen des Umgebungsverkehrs integriert werden und ebenso die entsprechenden Eingriffe durch den Fahrer.
Das Institut für Kraftfahrzeuge der RWTH Aachen University (ika) benutzt in der Entwicklung von Fahrerassistenzsystemen u. A. die Verkehrssimulation PELOPS, die bei der Partnerforschungsgesellschaft (fka) in Zusammenarbeit mit der Firma BMW entwickelt wurde. Die Verkehrssimulation verfolgt einen submikroskopischen Ansatz. Knotenpunkte waren bisher nur rudimentär implementiert. Für die Entwicklung der Regelungsfunktion (siehe Kapitel 4.8.1) musste eine Erweiterung des bestehenden Fahrermodells vorgenommen werden, sodass Verkehr auf Kreuzungen des Überlandbereichs generiert werden konnte. Weitere Anforderungen für das Model sind:

- Verwendung bewährter Modellansätze, um Entwicklungsauswand und Validierungsbedarf zu reduzieren;
- Parametrisierbar, sodass verschiedene Fahrergruppen und Fahrertypen abgebildet werden können;
- Realistisch, durch Realdaten belegt (validiert);
- Übersichtlich, einfach zu testen, zu erweitern, zu adaptieren und zu integrieren, da am ika viele beteiligte Personen für limitierte Zeit an dem Modell arbeiten (studentische Abschlussarbeiten, Dissertationen) und eine Einarbeitung daher schnell gehen muss;
- Flexibel einsetzbar, da das Modell nicht nur für den Einsatz in PELOPS zur Verfügung stehen soll (möglichst hohe Verwertbarkeit).

Aus den Anforderungen lässt sich ableiten, dass eine Implementierung in Matlab/Simulink sinnvoll ist, da dieses Werkzeug insbesondere im technischen, wissenschaftlichen Umfeld weite Verbreitung gefunden hat und leicht zu erlernen ist. Das grafische Programmierungskonzept passt zudem zu der im Folgenden vorgestellten Modellstruktur. Die Integration in Anwendungen kann über eine Kommunikationsschnittstelle (z. B. UDP) ohne großen Aufwand geschehen. Simulink bietet dafür fertige Blöcke. Zudem wurde eine Methode entwickelt mit der sich aus dem Simulink-Modell C-Code generieren lässt, der wiederum mit einem Klassenwrapper – also eingebettet in eine Klassenstruktur – zu einer dynamischen

Programmierbibliothek kompiliert werden oder nativ in z. B. Fortran-, C- oder C++-Code kopiert werden kann. Dynamische Programmbibliotheken lassen sich in den meisten Entwicklungswerkzeugen verwenden. Zudem gibt es das Konzept in fast jedem Betriebssystem und die Kompilierung des Codes funktioniert bei fast allen Kompilierern ähnlich. Eine Implementierung in Java, womit das PELOPS-Kernmodul in der neusten Version entwickelt wurde, wäre dann auf Java-Anwendungen begrenzt und müsste für den Einsatz in anderen Programmiersprachen in diese übersetzt werden.
Ein modularer Implementierungsansatz ist hier sinnvoll, da so Teile des Modells entfernt, neue hinzugefügt und bestehende verbessert oder auf spezielle Anwendungsfälle angepasst werden können. Zudem sind viele existierende Fahrermodelle in logische Teile aufteilbar, die in eine modulare Struktur untergebracht werden können. Das IDM (Intelligent Driver Model, [Ref 13]) setzt sich zum Beispiel aus einem Term für das freie Fahren und einem Term für das Folgen bzw. Annähern zusammen. Das Wiedemann Modell [Ref 14] unterteilt dieses Verhalten in vier Teile, inklusive einer Entscheidung.
In Abbildung 36 ist die Modellstruktur des entwickelten Fahrermodells dargestellt. Die Modellstruktur bildet die gesamte Fahraufgaben der Ebenen Navigation, Bahnführung (Guidance) und Stabilisierung (Stabilisation) nach Donges ([Ref 15]) ab. Das Verhalten des Fahrers ist dabei in die Blöcke Wahrnehmung (Perception), Informationsverarbeitung (Processing) und Handlung (Action) aufgeteilt. Zudem wird ein Modul angelegt – in der Darstellung mit State für Zustand bezeichnet –, der für die Speicherung von kurzfristigen und langfristigen Informationen (Memory) sowie für die Verarbeitung der fahrerspezifischen Parameter (Parameters) zuständig ist. Der Block speichert somit den Zustand des Fahrers (State). Die Wahrnehmung stellt die Eingangsschnittstelle des Fahrermodells zur Umwelt und den umgebenen Verkehr dar, während die Handlung die Ausgangsschnittstelle zu den Fahrzeugbedienelementen darstellt. In der Informationsverarbeitung wird die Fahrstrategie berechnet. Die blauen Rechtecke in der Darstellung bezeichnen die Implementierungsmodule.

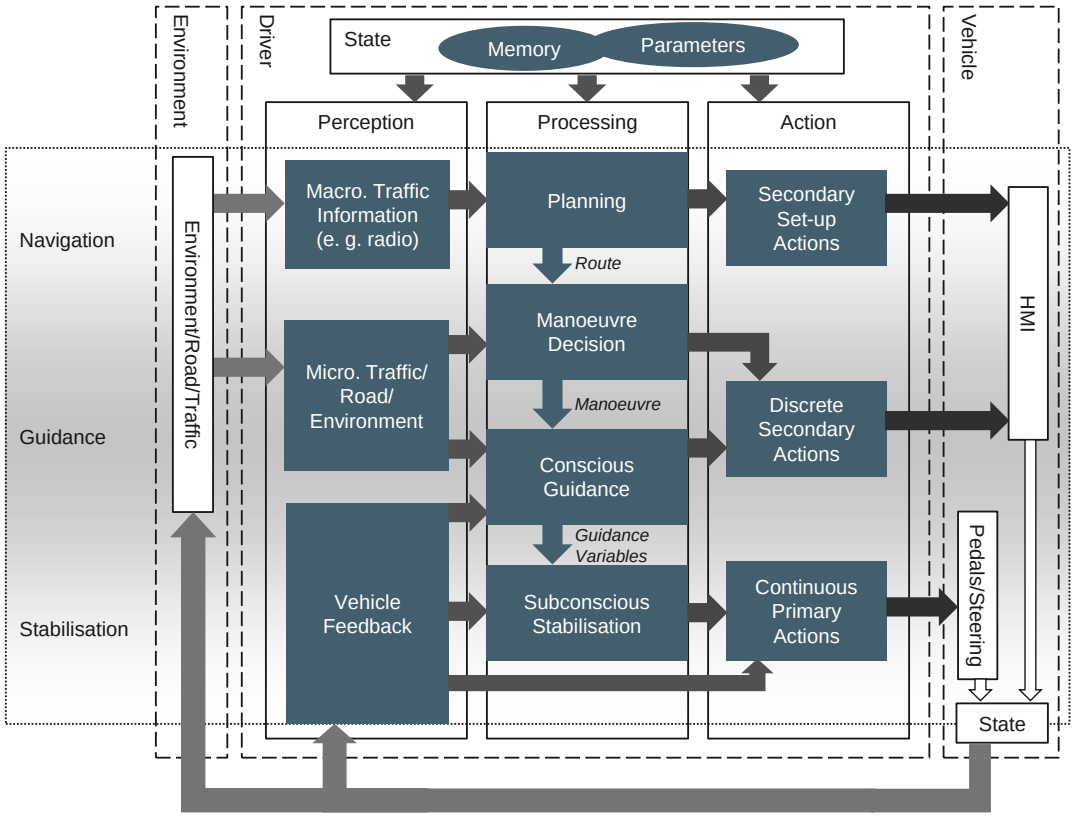


Abbildung 36: Struktureller Aufbau des Fahrermodells mit Implementierungsmodulen

Die Umsetzung der Fahraufgabe verteilt sich – wie oben erwähnt – auf mehrere Module in verschiedene Ebenen. Die Module enthalten Modellansätze, die auf Basis der Wahrnehmung, des aktuellen Zustands und der Parameter des Fahrermodells ein Ergebnis generieren, das an nachfolgende Module der Verarbeitung oder an Handlungsmodule weitergeben und dort weiterverarbeitet wird. Diese Ausgänge werden als Zwischenwerte bezeichnet und stehen als zusätzliche Schnittstelle des Modells zum Testen zur Verfügung. Die wichtigsten Module sind hierbei das Entscheidungsmodul (Manoeuvre Decision), das auf Basis der Planung und der aktuellen Situation eine diskrete Entscheidung für Manöver trifft aber auch die Entscheidung, welches Fahrzeug relevant ist. Die Entscheidung führt dazu, dass in zwei Mo-

dulen kurzfristige Fahrstrategien berechnet werden, die bewusst (Conscious Guidance) und unterbewusst (Subconscious Stabilisation) zu Führungsgrößen führen. Die Details zur Wahl und Implementierung der Modellansätze in den Modulen für das Projekt ist in dem Projektbuch [Ref 16] beschrieben und würde den Rahmen dieses Berichts sprengen, weshalb im Weiteren lediglich das allgemeine Implementierungskonzept und die Ergebnisse präsentiert werden.
Die Umsetzung der Führungsgrößen über das Einstellen von Fahrpedalwerten und Lenkwerten beeinflusst die Fahrzeugdynamik und ergibt nach zeitlicher Integration der dynamischen Zustände des Fahrzeugs eine Bewegung und damit eine Trajektorie. Dabei lässt sich diese über Parameter indirekt beeinflussen. Werden die Parameter extrem gewählt, bilden sich die Randbereiche der Trajektorien, in denen alle Fahrer liegen. Werden die Parameter durchschnittlich gewählt, bildet sich eine durchschnittliche Trajektorie. In Abbildung 37 sind die Geschwindigkeitstrajektorien von136 Fahrern beim Linksabbiegen auf einer Realkreuzung in der Nähe von Aachen dargestellt. Extremfahrer, Durchschnittsfahrer sowie einige Variationen einzelner Parameter sind ebenfalls dargestellt. Es ist erkennbar, dass die meisten der gemessenen Trajektorien innerhalb der Ränder durch die simulierten Extremfahrer liegen. Zudem ist erkennbar welcher Parameter an welcher Stelle des Abbiegevorgangs Einfluss nimmt. Beispielweise hat die Gesetzestreue keinen Einfluss auf die Kurvengeschwindigkeit, jedoch auf die Geschwindigkeitswahl vor und teilweise während der Annäherung. Die Geschwindigkeit im Scheitelpunkt wird hingegen maßgeblich durch das Sicherheitsbedürfnis beeinflusst.

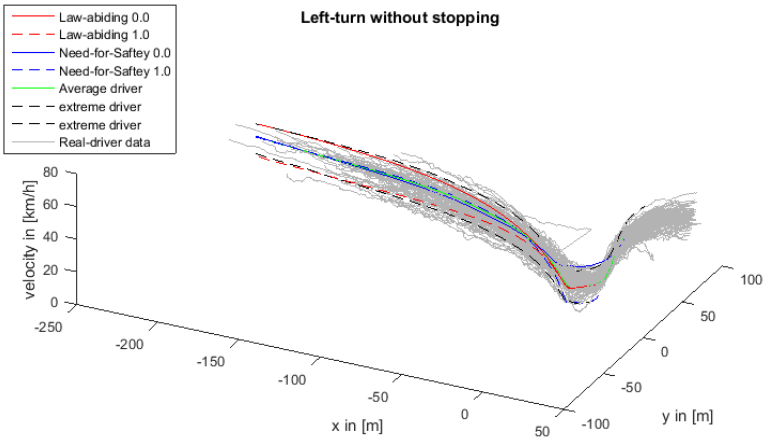


Abbildung 37: Simulationsergebnisse für Extrem- und Durchschnittsfahrer im Vergleich mit Realfahrern beim Linksabbiegen ohne Einfluss durch priorisierten Verkehr auf einer T-Kreuzung im Überlandbereich

Die Funktionen des Fahrermodells, die für das Projekt implementiert wurden sind die Folgenden:

- Freies Fahren und Folgefahren
- Spurhalten und Kurvenfahren
- Anhalten, Stehen und Anfahren
- Fahrstreifenwechsel
- Sicheres Passieren
- Situationsbedingtes reagieren auf Stoppschilder, Ampeln, vorfahrtberechtigen Verkehr (an Konfliktzonen)
- Geschwindigkeitswahl auf Basis von erlaubter Geschwindigkeit und Krümmungsverlauf der Strecke

Das Fahrermodell wurde eingesetzt, um unter verschiedenen Parametereinstellungen die Regelungsfunktion zu testen. Dabei wurde das Modell sowohl zur Generierung des Umgebungsverkehrs genutzt als auch als Fahrer für das Egofahrzeug. Das Fahrermodell verhält sich dabei bei angeschaltetem System passiv, sofern diese nicht zu stark von dem Wunsch des Fahrers abweichen, ansonsten wird das System abgeschaltet und das Fahrermodell übernimmt die Kontrolle.
Durch den Einsatz der Simulation mit realistischem Verkehr und Situation, die alltäglich auf Straßen und Knotenpunkten im Überlandbereich auftreten, lässt sich der Aufwand gerade im Anfangsstadium der Entwicklung eines Assistenzsystems erheblich reduzieren, da fehlende Eigenschaften, Fehler und ungewolltes Verhalten schnell identifiziert werden können und das System optimiert werden kann. Zudem lassen sich gezielt zu überprüfende Situationen generieren und das System unter Einfluss dieser Situ-

ation reproduzierbar testen. Das entwickelte Fahrermodell liefert den Grundstein für eine solche Simulation und wird insbesondere für das realistische Reagieren auf die entsprechenden Situationen und Handlungen des Systems benötigt.

3.9 Der Inter-Urban Assist Demonstrator

Der **Inter-Urban-Assist** Demonstrator **(IUA)** wurde von der Daimler AG und den Projektpartnern in eine Mercedes-Benz S-Klasse integriert (siehe Abbildung 38). Die Hauptaufgabe des IUA ist es, den Fahrer bei Nachtfahrten auf Landstraßen, den Verbindungsstraßen zwischen zwei Städten, zu unterstützten. Der IUA ist ein Fahrerassistenzsystem, das den Fahrer durch Sicht- bzw. Wahrnehmungsverbesserungsfunktionen bei einer der schwierigsten Fahrsituationen in der Nacht unterstützt:

Während der nächtlichen Fahrt auf engen, kurvigen Straßen, bei denen der weitere Straßenverlauf sehr schwierig auszumachen ist und Hindernisse bzw. andere Verkehrsteilnehmer wie z.B. ein dunkel gekleideter Fußgänger in der Regel erst sehr spät erkannt werden.

Das IUA unterstützt den Fahrer bei dieser schwierigen Fahraufgabe durch das Markieren des Straßenverlaufs in einem im Fahrzeug dargestellten Nachtsichtbild der Fahrszene, eine verbesserte Visualisierung der Fahrszene in Fahrtrichtung durch eine perfekte, dem Straßenverlauf angepasste Ausleuchtung der Straße, und dem Hervorheben des in Fahrtrichtung auftretenden Fußgängers durch folgende Fahrerassistenzfunktionen:

- Ein augmentiertes Nachtsichtsystem (**augmented night vision system**) indem dem Fahrer über einen Monitor ein Nachtsichtbild der Fahrszene angeboten wird, in dem die Straße und die Hindernisse so markiert sind, dass sie dieser unmittelbar wahrnehmen kann,
- Ein prädiktives Fahrzeugscheinwerfersystem (**predictive front light)** in dem Ausleuchtung der Straße dem aktuellen Straßenverlauf angepasst ist
- Der als **spotlight** bezeichneten Scheinwerferfunktion, mittels derer ein in Fahrtrichtung auftretender Fußgänger durch dreimaliges Anblinken in der Fahrszene hervorgehoben wird.

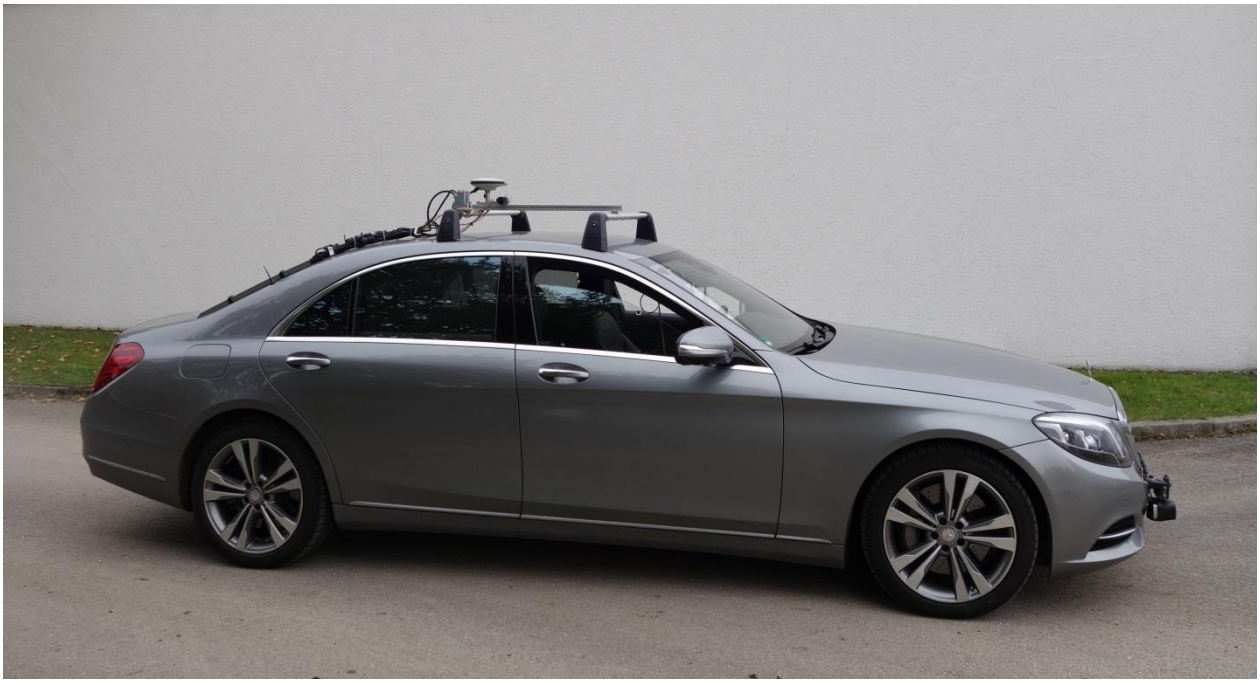


Abbildung 38: Mercedes Benz S-Klasse - Inter-Urban Assist Fahrzeug Demonstrator

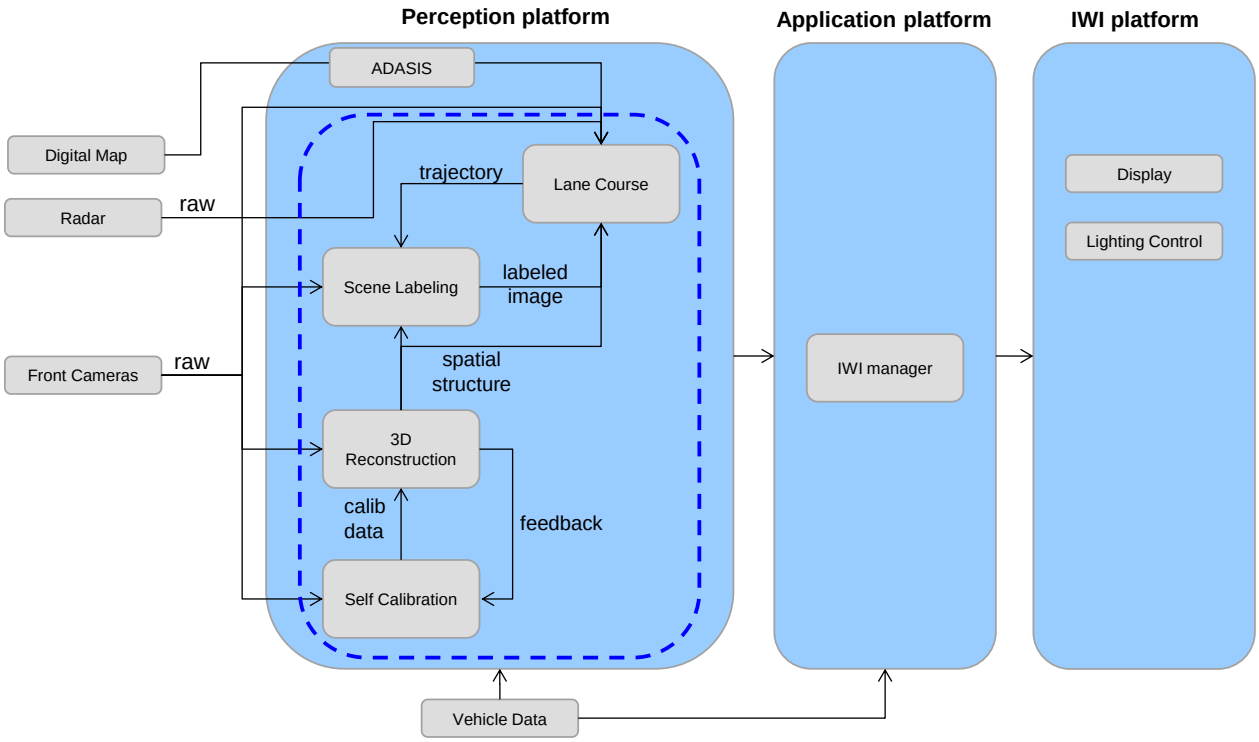


Abbildung 39: Functional blocks diagram for Inter Urban Assist

Im Funktionsblockdiagramm in Abbildung 39 wird die Architektur des IUA mit den wichtigsten Modulen des IUA dargestellt.

In der Perception Plattform wird aus den von den Umgebungserfassungssensoren des Fahrzeugs empfangenen Daten die Objekterkennung des Systems betrieben. Die ermittelten Objekte inklusive ihrer Relationen zueinander werden der Application Platform zugeführt. In der Application Plattform werden diese Daten hinsichtlich der Fahrerassistenzfunktion bewertet und daraus eine Aktionsstrategie abgeleitet. Diese wird dann in der IWI (**I**ntervention, **W**arning, **I**nteraction) Plattform umgesetzt. Die wichtigsten Module dieser Funktionsblöcke werden im Folgenden erläutert.

Perception platform

Zur Realisierung der Darstellungsfunktion bzw. der Lichtsteuerung müssen Daten verschiedener Sensoren zeitgleich genutzt werden. Um z.B. den Fahrer mit dem augmentierten Nachtsichtbild der Fahrszene zu unterstützen, werden die Daten von vier Sensortypen genutzt. Die Video-Daten der Front-Kamera liefern das Nachtsichtbild. Aus diesen Daten wird aber auch die Fahrbahnbegrenzung im Nachbereich bestimmt. Ein Radar-Sensor wird genutzt, um den Erfassungsbereich bis auf 200 m auszudehnen. Zur weiteren Ausdehnung des Erfassungsbereiches müssen die Daten einer digitalen Karte genutzt werden. Ferner müssen für die Berechnung der Fahrzeug-Odometrie Fahrzeugdaten wie Geschwindigkeit, Lenkwinkel, Gierwinkel, etc. verwendet werden. Der Montageort des ARS310-Radarsensors wird in Abbildung 40 gezeigt, der Montageort der Nachtsichtkamera –es werden zwei Nahinfrarotkameras verwendet- wird in Abbildung 41 gezeigt.

In der Perception platform werden aus diesen Daten die für die Realisierung der FAS notwendigen Objektdaten (incl. der dazugehörigen Raum- und Zeitgrößen) berechnet. Die Berechnung erfolgt über die vier Perception Module **Lane Course**, **Scene Labeling**, **3D Reconstruction** und **Self Calibration** mit folgenden Aufgaben:

- **Lane Course:** Bestimmt den Verlauf der Fahrspur vor dem Fahrzeug bis zu einer sinnvollen und für die Fahrerassistenzfunktion notwendigen Entfernung. Der Fahrspurverlauf wird für die prädiktive Lichtfunktion und das augmentierte Nachtsichtbild verwendet.

- **Scene Labeling:** Berechnet die Klassenzugehörigkeit (Straße, Hintergrund, Objekt) zu jedem Bildpunkt des Eingangsbildes. Diese wird als semantische Szenen-Information für das Nachtsichtbild genutzt.
- **3D Reconstruction:** Berechnet die räumliche Struktur der Szene vor dem Fahrzeug und liefert damit zu jedem 2D-Bildpunkt eine 3D-Informationen.
- **Self Calibration:** Berechnet die Kalibrierungsinformationen für jedes Bildpaar, was wiederum für die Berechnung der 3D-Informationen erforderlich ist.

Abbildung 40: Monatageort des Radar-Sensors unter der Frontschürze links.

Abbildung 41: Montageort der zwei Nah-Infrarot-Kameras hinter der Windschutzscheibe

Application platform

Der **IWI-Manager** analysiert anhand der Daten aus dem Perception Modul das Szenario und steuert abhängig vom Analyseergebnis die Module der IWI platform an.

IWI (Intervention, Warning, Interaction) platform

Das **Lighting Control** Modul deckt alle IUA-Funktionalitäten ab, die die Lichtsteuerung beeinflussen. Die vom System detektierten Fußgänger werden mit dem Scheinwerfer 3x angeblinkt. Ferner wird zur bestmöglichen Ausleuchtung des Fahrwegs die Strahlrichtung des Hauptscheinwerfers an den weiteren Straßenverlauf ausgerichtet. Das Ergebnis ist die bezüglich der Strahlverteilung des Scheinwerfers optimale Ausleuchtung des Straßenverlaufs (siehe bis Abbildung 44).

Das **Display** Modul deckt alle IUA-Funktionalitäten ab, die das augmentierte Nachtsichtbild nutzen. Ziel ist es, das Nachtsichtbild mit Informationen so anzureichern, dass diese vom Fahrer besonders schnell erfasst werden können. In dem im Kombiinstrument dargestellten Nachtsichtbild wird der Straßenverlauf im Bild mittels einer Nordlicht-Darstellung markiert (siehe Abbildung 45).

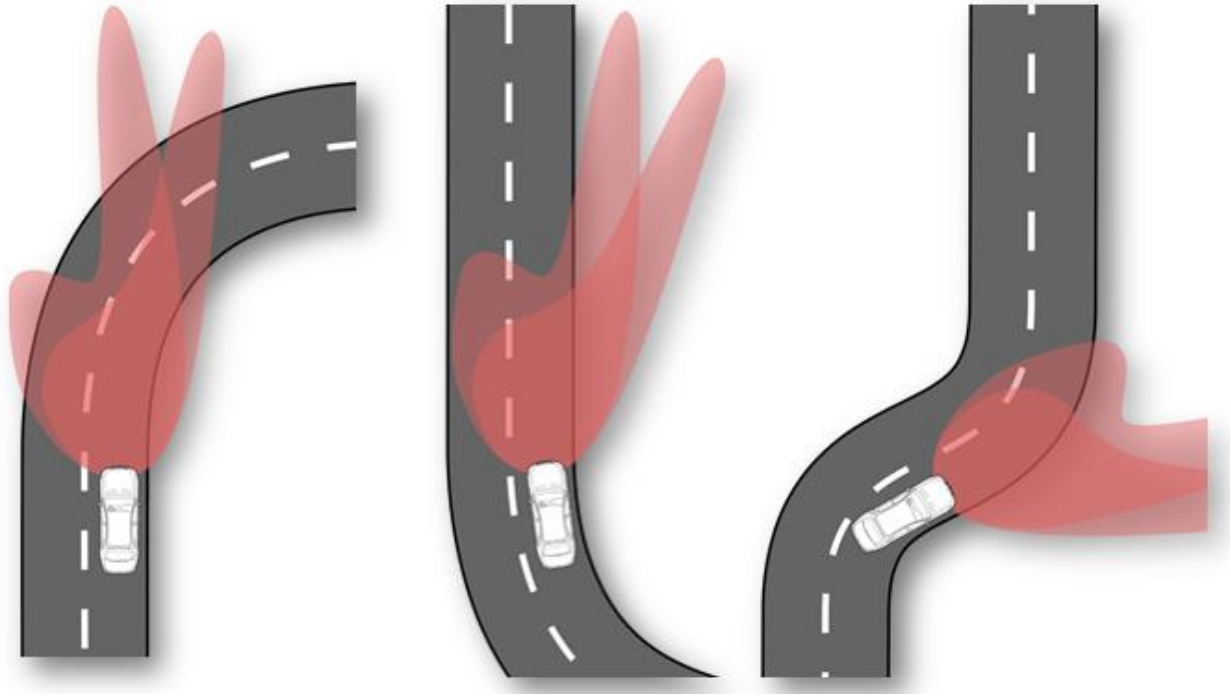


Abbildung 42: Ausleuchtung des Fahrwegs ohne Kenntnis des Straßenverlaufs.

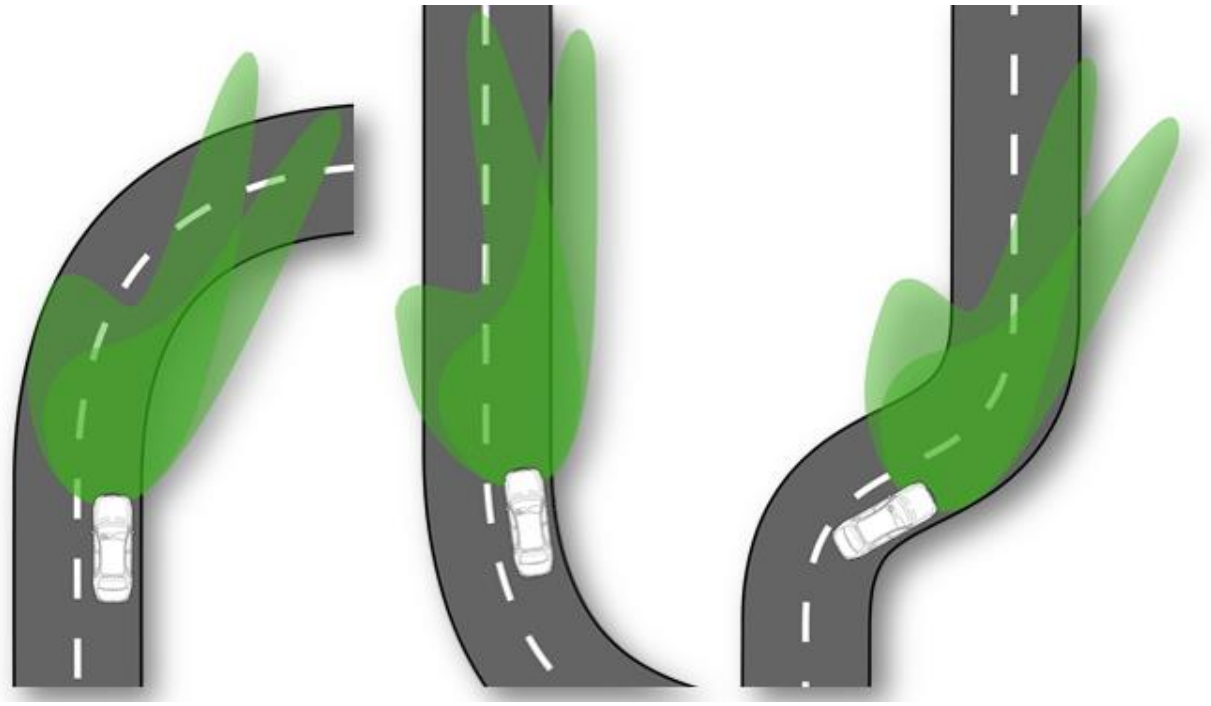


Abbildung 43: Perfekte Ausleuchtung des Fahrwegs angepasst an den Straßenverlauf.

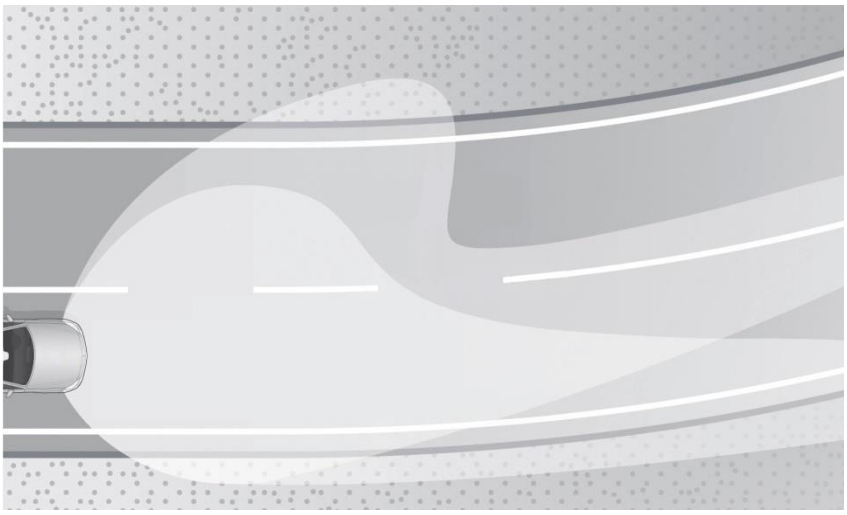


Abbildung 44: Scheinwerferhauptstrahl angepasst an den Straßenverlauf.

Abbildung 45: Augmentiertes Nachtsichtbild im Kombiinstrument des Fahrzeugs. Der Straßenverlauf wird durch die eingeblendete Nordlicht-Markierung angezeigt.

Der IUA Demonstrator wurde auf der DESERVE Abschlussveranstaltung im Dezember 2015 in Ulm präsentiert. Abbildung 46 zeigt einige Impressionen von diesem Event.

Abbildung 46: DESERVE Abschlussveranstaltung im Dezember 2015

4 Danksagung

Die Partner des Deutschen DESERVE Konsortiums danken dem Bundesministerium für Bildung und Forschung herzlich für die nationale Teil-Finanzierung dieses Projekts im Rahmen der Joint Technology Initiative (JTI) ECSEL. Dieser Dank gebührt in gleicher Weise der Europäischen Kommission für deren Teil-Finanzierung auf EU Ebene.

Ein großer Dank geht ebenfalls an den Projektträger DLR für die stets kompetente Bearbeitung und administrative Betreuung während der gesamten Projektlaufzeit und alle Europäischen Projektpartner für die freundliche, kollegiale und effiziente Zusammenarbeit in DESERVE.

5 Quellennachweise

[Ref 1] Bundesministerium für Bildung und Forschung, „Ideen. Innovation. Wachstum - Hightech-Strategie 2020 für Deutschland“, 2010

[Ref 2] MicroAutoBox II, www.dspace.com/en/pub/home/products/hw/micautob.cfm

[Ref 3] EB Assist ADTF, http://automotive.elektrobit.com/driver-assistance/eb-assist-adtf

[Ref 4] RTMaps, http://www.intempora.com/

[Ref 5] AXI Reference Guide, http://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf

[Ref 6] Intel® Core™ i7,http://www.intel.eu/content/www/eu/en/processors/core/core-i7-processor.html?wapkw=intel+core+i7+processor

[Ref 7] FlexRay automotive network communication protocol, https://en.wikipedia.org/wiki/FlexRay

[Ref 8] Xilinx® Kintex-7 FPGA Family, http://www.xilinx.com/products/silicon-devices/fpga/kintex-7/

[Ref 9] ADTF Blockset, https://www.dspace.com/en/pub/home/products/sw/impsw/adtf_blockset.cfm

[Ref 10] RTMaps - Simulink - dSPACE prototyping systems, http://www.intempora.com/services/tutorials-technical-articles/studio/intermediate/329-rtmaps-simulink-dspace.html

[Ref 11] Infineon Aurix Microcontroller, http://www.infineon.com/cms/en/product/channel.html?channel=db3a30433727a44301372b2eefbb48d9

[Ref 12] http://www.dspace.com/en/pub/home/products/hw/modular_hardware_introduction/i_o_boards/ds5202.cfm

[Ref 13] Treiber, Martin; Hennecke, Ansgar; Helbing, Dirk (2000): Congested Traffic States in Empirical Observations and Microscopic Simulations. In: Rev. E 62, Issue 62, S. 2000.

[Ref 14] Wiedemann, Rainer (1974): Simulation des Straßenverkehrsflusses. Karlsruhe: Institut für Verkehrswesen

[Ref 15] E. Donges: Das Prinzip der Vorhersehbarkeit als Auslegungskonzept für Maßnahmen zur Aktiven Sicherheit im Straßenverkehrssystem, VDIGesellschaft Fahrzeugtechnik, Das Mensch-Maschine-System im Verkehr,Düsseldorf, 1992

[Ref 16] DESERVE booklet - “Towards a common software/hardware methodology for future advanced driver assistance systems - The DESERVE approach“, Editors: G. Payá-Vayá and H. Blume, River Publishers, printing foreseen end of 2016

[Ref 17] DESERVE – DEvelopment platform for Safe and Efficient dRiVE
An ARTEMIS Joint Undertaking in the FP7 programme – grant agreement no. 295364

[Ref 18] Limmer, Matthias; Forster, Julian; Baudach, Dennis; Schüle, Florian; Schweiger, Roland; Lensch, Hendrik P.A.: Robust Deep-Learning-Based Road-Prediction for Augmented Reality Navigation Systems

[Ref 19] Simonyan, Karen; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition | International Conference on Learning Representations 2014

[Ref 20] Farabet, C.; Couprie, Camille; Najman, Laurent; LeCun, Yann: Learning Hierarchical Features for Scene Labeling | IEEE Transactions on Pattern Analysis and Machine Intelligence 2013, page 1915-1929

[Ref 21] Giusti, A.; Ciresan, D.C.; Masci, J.; Gambardella, L.M.; Schmidhuber, J.: Fast image scanning with deep max-pooling convolutional neural networks | Image Processing (ICIP), 2013 20th IEEE International Conference, page 4034-4038

[Ref 22] Thom, Markus; Gritschneder, Franz: A Theory for Rapid Exact Signal Scanning with Deep Multi-Scale Convolutional Neural Networks, 2016

6 Begriffserklärungen und Abkürzungen

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
ASIL	Automotive Safety Integrity Level
ASIP	Application Specific Instruction Processor
ASN.1	Abstract Syntax Notation One
AUTOSAR	AUTomotive Open System ARchitecture
ECSEL	Electronic Components and Systems for European Leadership
ECC	Error Correction Code
ECU	Electronic control Unit
ETSI	European Telecommunications Standards Institute
EU	Europäische Union
FAS	FahrerAssistenz Systeme
FPGA	Field Programmable Gate Array
GigE	Gigabit Ethernet
GNSS	Global Navigation Satellite System
HAF	Hoch-automatisiertes Fahren
HMI	Human Maschine Interface
HW	Hardware
IHC	Intelligent Headlight Control
ISO	International Standardization Organization
IUA	Inter Urban Assist
JESD204	JEDEC Standard 204
JU	Joint Undertaking
MIMO	Multiple Input Multiple Output
NMEA	National Marine Electronics Association
LDW	Lane Depature Warning
LKS	Lane Keeping Support
OEM	Original Equipment Manufacturer (deutsch: Erstausrüster)
OSI	Open Systems Interconnection
PED	Pedestrian Protection
QSFP	Quad Small Form-factor Pluggable
µC	Mikrocontroller
RSR	Road Sign Recognition
SD-RAM	Synchronous Dynamic Random Access Memory
SOP	Start of Production
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TMB	Technical Management Boards
UC	Use Case
UDP	User Datagram Protocol
USB	Universal Serial Bus
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLIW	Very Large Instruction Word