Zuwendungsempfänger: Technische Universität München	Förderkennzeichen: 16ES1029
Vorhabenbezeichnung: „KI-FLEX: Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren“	
Laufzeit des Vorhabens: von 01.09.2019 bis 31.08.2023	

Die Kernfrage, mit der sich TUM in diesem Projekt beschäftigt ist wie können die rechenintensiven KI-Algorithmen optimal auf vorhandene HW-Ressourcen verteilt werden, so dass bestimmte Optimierungsziele wie z.B. die Minimierung des Stromverbrauchs während die Systemperformanz eine bestimmte Grenze nicht unterschreitet. Dabei ist die Herausforderung, dass diese Ressourcenzuteilung sehr dynamisch zur Laufzeit stattfinden muss wenn z.B. Fahrsituationen sich ändern und daher bestimmte KI-Algorithmen aktiviert, deaktiviert oder umdisponiert werden müssen. Diese Änderungen basierend auf die Kritikalität des jeweiligen KI-Algorithmus. Das Problem der Planung der Ressourcen kann als ein Scheduling-Problem angesehen werden, das aus der theoretischen Informatik bekannt ist.

In diesem Teilvorhaben betrachtet TUM dieses Problem im Kontext des autonomen Fahrens und die limitierten eingebetteten Ressourcen, die vorhanden sind. Um die Aufgabe der Ressourcenplanung zu lösen, wird zuerst ein Meta-Model definiert. Dieses Meta-Model wird benutzt, um sowohl die Eigenschaften der Hardwareressourcen und deren Kapazitäten und als auch die Anforderungen, die von KI-Algorithmen kommen, zu beschreiben. Diese model-basierte Vorgehensweise ermöglicht die Ableitung der benötigten Informationen zur Ableitung der Scheduling-Bedingunen. Die Abgeleiteten Constraints können dann in der Sprache der SolverTechnologien kodiert und mit Berücksichtigung der Optimierungsziele gelöst werden. Die Vorarbeiten der TUM im Bereich des model-basierten Schedulings für zeitgesteuerte Netzwerke werden bei der Entwicklung der Konzepte in KI-FLEX behilflich sein.

In Anbetracht der obigen Ausführungen haben wir drei Entwicklungen vorgeschlagen, um die Projektziele zu erreichen: einen modellbasierten Softwarerahmen, einen Ansatz zur Analyse von Designfehlern und einen Überwachungsmechanismus. Jeder dieser Vorschläge wird im Folgenden kurz erläutert.

Software Framework

Wir haben ein neuartiges modellbasiertes Framework vorgestellt, das die Synthese von E/E-Architekturen für Fahrzeuge erleichtert und die Modellierung von eingebetteten Systemen für Fahrzeuge unterstützt. Das vorgestellte Werkzeug automatisiert die Zuordnung von Softwarekomponenten zu Hardwareelementen, d.h. die Ressourcenzuweisung und die Berechnung von Zeitplänen für Anwendungsthreads. Es legt das Routing von Netzwerknachrichten fest und plant Kommunikationsaufgaben innerhalb der Fahrzeugtopologie unter Berücksichtigung von Sicherheitsanforderungen, einschließlich Redundanz, homogener Redundanz und Zuverlässigkeit. Das vorgeschlagene computergestützte Tool optimiert auch das Systemmodell und deckt mehrere Optimierungsziele ab. Es unterstützt die Mehrzieloptimierung und verwendet einen Ein-Schritt-Ansatz zur Lösung von MIP-Beschränkungen (Mixed-Integer Programming), wodurch die Lösungszeit reduziert und die Beziehungen zwischen den verschiedenen Beschränkungen berücksichtigt werden. Darüber hinaus bietet das vorgeschlagene Tool ein webbasiertes Frontend, mit dem Benutzer ihre gewünschten E/E-Systeme modellieren und verschiedene Hardware- und Softwareanforderungen und -eigenschaften sowie die Randbedingungen und Optimierungsziele auswählen können. Das entwickelte Frontend visualisiert auch die Lösung des entworfenen Systems, nachdem es gelöst wurde.

Design Error Analysis Approach

Es gibt Situationen, in denen eine entworfene E/E-Architektur nicht zufriedenstellend ist, was bedeutet, dass der MIP Löser keine machbaren Lösungen finden kann. Im Gegensatz zu einfachen Modellen ist das

Navigieren und Korrigieren der Unerfüllbarkeit komplexer E/E-Modelle eine komplexe und zeitaufwändige Aufgabe, die zu erhöhten Entwicklungskosten führt. Um dieses Problem anzugehen, wird in dieser Arbeit auch ein Ansatz zur Identifizierung von Entwurfsfehlern vorgestellt, wenn nach dem Lösungsschritt Verletzungen in der im Systemmodell enthaltenen Constraintmenge auftreten.

Monitoring Mechanism

Was das Laufzeitverhalten des Betriebssystems und der Middleware innerhalb der Hardwareplattform betrifft, so entstehen Unsicherheiten durch ereignisbasierte Aktivitäten, wie z. B. die Erkennung von Anwendungsdiensten und andere dynamische, interagierende Prozesse, die zu einer nicht-deterministischen Nutzung der Systemressourcen führen. Angesichts dieser Komplexität ist die Erstellung und Berücksichtigung einschlägiger Einschränkungen in der Entwurfsphase durch das vorgestellte Werkzeug unpraktisch. Um die Validierung von Anforderungen, z.B. zeitliche Anforderungen, während der Laufzeit nach der Bereitstellung von Lösungen, die durch das vorgestellte Framework auf der Hardwareplattform berechnet wurden, zu ermöglichen, wurde ein Überwachungsmechanismus entwickelt. Dieser Ansatz führt einen Überwachungsmechanismus zur Identifizierung von Timing-Verletzungen in Plattformen für autonomes Fahren. Um Risiken, die sich aus Anforderungsverletzungen ergeben, zu minimieren, erhält das Überwachungsmodul die vordefinierten Anforderungen, die dem Werkzeug ursprünglich vorgegeben wurden. Gleichzeitig sammelt es Echtzeitwerte von der Hardwareplattform. In der anschließenden Phase wird ein kontinuierlicher Vergleichs- und Verifizierungsprozess zwischen den Anforderungen zur Entwurfszeit und dem Echtzeitstatus bezüglich dieser Anforderungen eingeleitet. Im Falle eines Verstoßes wird eine Warnung ausgegeben, um die verletzte Anforderung zu melden.
Es wurde eine grafische Benutzeroberfläche (GUI) für den integrierten Überwachungsmechanismus entwickelt. Wenn der Wert einer Anforderung den festgelegten Schwellenwert überschreitet, wird außerdem eine sofortige Warnmeldung erzeugt. Der Überwachungsmechanismus wurde auf der KI-FLEX-Hardwareplattform implementiert, um die kritischen Parameter der Lokalisierungseinheit zu überwachen.

Evaluation

Das Hauptziel ist eine strenge Bewertung der die Durchführbarkeit, die Wirksamkeit und die praktische Anwendbarkeit des vorgeschlagenen Software-Rahmens zu bewerten. Eine systematische Reihe von Experimenten und Bewertungen, einschließlich zweier Arten von Evaluierungen, zielt darauf ab die im vorgestellten Rahmenwerk formulierten Hypothesen zu validieren und empirische Beweise zu liefern. Anhand verschiedener Fallstudien haben wir die Leistung, Anwendbarkeit und Skalierbarkeit des Tools Skalierbarkeit in der Entwurfsphase. Ziel war es, die Stärken, die Anpassungsfähigkeit und die potenziellen Verbesserungsmöglichkeiten des Tools herauszustellen und empirische Erkenntnisse über seinen praktischen Wert zu gewinnen. Der vorgeschlagene Rahmen ist ein wertvolles Hilfsmittel zur Rationalisierung des Entwurfsprozesses für Systemarchitekten. Der praktische Nutzen dieses Rahmens kann jedoch beeinträchtigt werden durch lange Berechnungszeiten, die sowohl bei der Generierung als auch bei der Lösung von Beschränkungen anfallen. Um dieser Herausforderung zu begegnen, wurde eine umfassende Untersuchung durchgeführt, um die Auswirkungen Einfluss verschiedener Parameter innerhalb des vorgestellten Systemmodells auf die Zeit, die für eine effiziente Generierung und Lösung von Nebenbedingungen.
Bei der Laufzeitevaluierung werden die vom Tool berechneten Lösungen auf einer realen Hardwareplattform eingesetzt, um die Anwendbarkeit der Lösungen aus der Entwurfszeit in einer realen Implementierung zu beobachten. Im Rahmen dieses Abschnitts werden die vom Tool akribisch berechneten Lösungen
Werkzeug berechneten Lösungen durch den Einsatz auf einer realen Hardwareplattform greifbar gemacht. Diese strategische Einsatz dient als Brücke, die den Bereich der Entwurfszeitentscheidungen mit der der dynamischen Umgebung der Laufzeitausführung.

KI-FLEX: Rekonfigurierbare Hardwareplattform zur KI-basierten Sensordatenverarbeitung für das autonome Fahren

Schlussbericht Teilvorhaben Technische Universität München

(eingehende Darstellung)

Laufzeit des Vorhabens: von 01.09.2019 – bis 31.08.2023

Zuwendungsempfänger: Technische Universität München

Förderkennzeichen: 16ES1029

Ansprechpartner: Hadi Askaripoor (askaripo@in.tum.de)

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

Projektträger: VDI/VDE Innovation + Technik GmbH

INHALTSVERZEICHNIS

TABELLENVERZEICHNIS

1 Aufgabenstellung und Zielsetzung

Die Kernfrage, mit der sich TUM in diesem Projekt beschäftigt ist wie können die rechenintensiven KI-Algorithmen optimal auf vorhandene HW-Ressourcen verteilt werden, so dass bestimmte Optimierungsziele wie z.B. die Minimierung des Stromverbrauchs während die Systemperformanz eine bestimmte Grenze nicht unterschreitet. Dabei ist die Herausforderung, dass diese Ressourcenzuteilung sehr dynamisch zur Laufzeit stattfinden muss wenn z.B. Fahrsituationen sich ändern und daher bestimmte KI-Algorithmen aktiviert, deaktiviert oder umdisponiert werden müssen. Diese Änderungen basierend auf die Kritikalität des jeweiligen KI-Algorithmus. Das Problem der Planung der Ressourcen kann als ein Scheduling-Problem angesehen werden, das aus der theoretischen Informatik bekannt ist.

In diesem Teilvorhaben betrachtet TUM dieses Problem im Kontext des autonomen Fahrens und die limitierten eingebetteten Ressourcen, die vorhanden sind. Um die Aufgabe der Ressourcenplanung zu lösen, wird zuerst ein Meta-Model definiert. Dieses Meta-Model wird benutzt, um sowohl die Eigenschaften der Hardwareressourcen und deren Kapazitäten und als auch die Anforderungen, die von KI-Algorithmen kommen, zu beschreiben. Diese model-basierte Vorgehensweise ermöglicht die Ableitung der benötigten Informationen zur Ableitung der Scheduling-Bedingunen (Scheduling Constraints). Die Abgeleiteten Constraints können dann in der Sprache der SolverTechnologien kodiert und mit Berücksichtigung der Optimierungsziele gelöst werden. Die Vorarbeiten der TUM im Bereich des model-basierten Schedulings für zeitgesteuerte Netzwerke werden bei der Entwicklung der Konzepte in KI-FLEX behilflich sein.

2 Durchgeführte Arbeiten und erreichte Ergebnisse

Die von der TUM beigesteuerten Arbeitspakete werden im Folgenden kurz erläutert. Anschließend werden der von uns entwickelte Ansatz und die Ergebnisse vorgestellt.

2.1.1 AP 1.2.9, Anforderungen und Konzepte

Die Ziele dieses Arbeitspakets sind Anforderungsanalyse der Ressourcen-Schedulers, Spezifikation der Architektur des Schedulers, and Entwicklung eines ersten Konzeptes.

2.1.2 AP 2.4.1, Entwicklung einer Testumgebung

2.1.3 AP 2.4.7, Entwicklung der Testszenarien für die Testumgebung

Funktionsweise der Testumgebung wird durch adäquate Testszenarien validiert.

2.1.4 AP 2.4.8, Integration der Dummy-Module in die Testumgebung

2.1.5 AP 2.5.15, Entwicklung der Konzepte enines Optimalen Ressourcen-Scheduler fur das HW-Zielsystem

Model-basierte Entwicklung eines optimales Scheduling für die Ausschöpfung der HW-Features

2.1.6 AP 2.5.16: Die entwickelten Konzepte des Ressourcen-Schedulers werden implementiert.

2.1.7 AP 2.5.17, Testen der Ressourcen-Schedulers und seine Schnittstellen

Die Funktionalität der implementierten Schnittstellen und der Ressourcen-Schedulers selbst werden getestet und gegen Anforderungen validiert.

2.1.8 AP 3.2.3, Plan für SW-Framework-Integration

Zusammenbringen von KI-Algorithmen und dem Ressourcen-Scheduler.

2.1.9 AP 3.2.6, Implementierung des Integrationsplans des SW Frameworks

Der entwickelte Ressourcen-Scheduler wird in das Zielsystems integriert.

2.1.10 AP 3.3.4

Präsentation der Funktionalitäten des Ressourcen-Schedulers zur Systemlaufzeit in unterschiedlichen Fahrsituationen.

2.1.11 AP 3.3.5, Finales Testen der Gesamt-System-Integration mit Fokus auf Ressourcen-Scheduler

Systemweites und intensives Testen der SW-Module und debuggen. Die Probleme müssen behoben werden und den Partnern mitgeteilt werden. Die abschließende Bewertung des von uns entwickelten Rahmens wurde durchgeführt und abgeschlossen. Außerdem wurde unser entwickeltes Überwachungsmodul in unserem eigenen Demonstrator getestet und zur endgültigen Einrichtung auf dem Demonstrator des Projekts an den entsprechenden Partner geschickt.

2.1.12 AP 3.3.9

Die Ziele sind der Ablauf der Demonstrationen ist geplant und dokumentiert und das Gesamtsystem ist ins Testfahrzeug integriert und kann die Daten der verbauten Sensoren korrekt verarbeiten.

2.2 Eingehende Darstellung

2.2.1 Ansatz

Wie in den Beschreibungen der Arbeitspakete erwähnt, schlagen wir ein modellbasiertes Software-Framework für den Entwurf und die Synthese von Fahrzeug-E/E-Architekturen vor, das verschiedene Funktionalitäten unterstützt [1, 2]. Die Architektur des vorgestellten Frameworks, das E/E Designer genannt wird, besteht aus drei Hauptteilen. Wie Abbildung 1 zeigt, stellt die linke Seitenleiste der Abbildung die Eingaben des Frameworks dar, die von einem E/E-Systemintegrator/-architekten bereitgestellt werden können. Zu diesen Inputs gehören verschiedene Eigenschaften und Anforderungen in Bezug auf Hardware- und Softwarekomponenten, den Mapping-Prozess, das Routing von Kommunikationsnachrichten, die zeitgesteuerte Planung und die Sicherheit [1, 3, 4, 6, 7]. Im Einzelnen handelt es sich um ASIL-Level, Turbo-Boost-Technologie, Kommunikations- und Prozess-/Thread-Ausführungszeiten und -perioden, Kostenüberlegungen, erzwungenes Mapping, Verbindungstyp, Knotentyp, Sender- und Empfängerinformationen für Kommunikationsnachrichten, die gewünschte Netztopologie (z.

B. Vollmaschentopologie oder andere Topologien, einschließlich der Anzahl der erforderlichen Knoten und Verbindungen), mittlere Zeit bis zum Ausfall (MTTF), Ausfallrate usw. Darüber hinaus kann ein grafischer E/E- Systemmodellierer verschiedene Optimierungsziele und

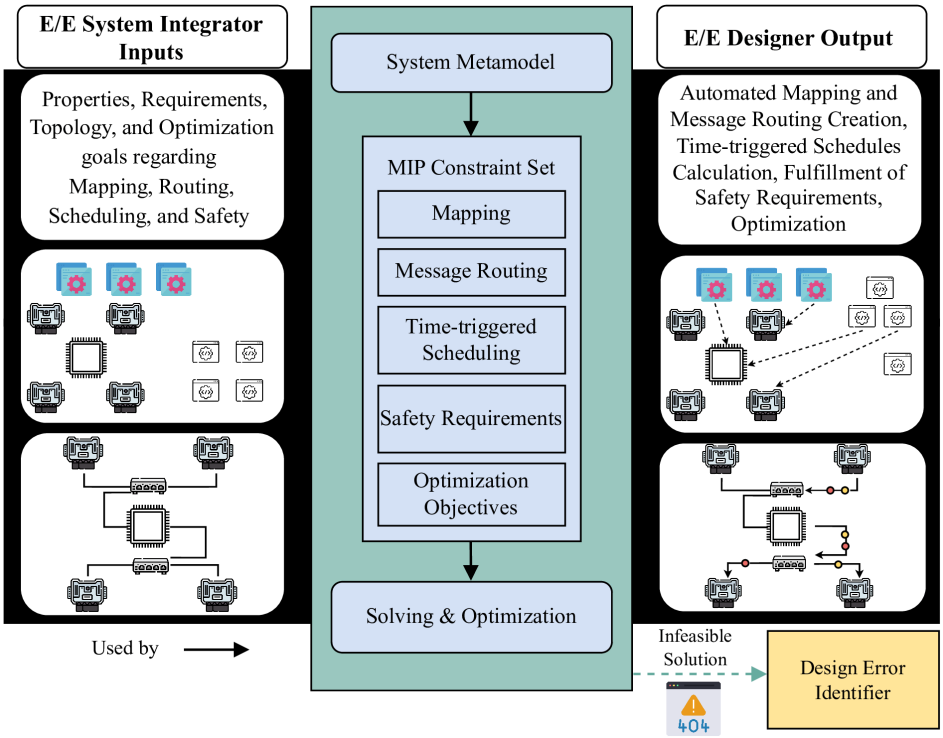

Abbildung 1: Die Architektur des vorgeschlagenen modellbasierten Rahmens. Die linke und rechte Spalte, die die Eingaben des E/E-Systemintegrators bzw. die Ausgaben des E/E-Designers darstellen, bilden das Frontend. Der mittlere Kasten stellt das Backend des Tools dar. Im Frontend des Frameworks wird eine E/E-Architektur von einem E/E-Systemarchitekten modelliert. Diese Modellierung umfasst die Definition von Anforderungen, Eigenschaften und die Behandlung verschiedener Probleme. Die modellierte E/E-Architektur wird mithilfe des MDD-Ansatzes in MIP-Formulierungen umgewandelt. Diese Formulierungen werden dann im Backend des Werkzeugs gelöst und optimiert. Schließlich wird die optimale Lösung im Frontend des Frameworks visualisiert.

Randbedingungen auswählen. Sie können auch eine gewünschte E/E-Architektur oder eine Fahrzeugnetztopologie mit Hilfe von Drag-and-Drop-Funktionalitäten entwerfen.

Der mittlere Teil von Abbildung 1 stellt in erster Linie das Back-End des Frameworks dar, in dem alle mathematischen Formulierungen und Berechnungen durchgeführt werden. Dieser Teil bleibt für den Benutzer der Einfachheit halber verborgen. Wie bereits erwähnt, verwendet der vorgestellte Ansatz ein objektorientiertes Metamodell, das dem Ansatz der modellgetriebenen Entwicklung (MDD) folgt und im folgenden Abschnitt erläutert wird. Dieses Metamodell dient als Grundlage für die grafische Modellierung, die vom Integrator oder Modellierer zur Erstellung grafischer Modellinstanzen verwendet wird. Unter Verwendung eines formalen System-Metamodells werden die grafischen Modellinstanzen, die alle ausgewählten Anforderungen und Eigenschaften umfassen, in einen Satz von MIP-Constraints (mixed-integer programming) im Rahmen der linearen Programmierung (LP) umgewandelt (gekennzeichnet durch den grünen Kasten in Abbildung 1) [1]. Dieser Bedingungssatz umfasst Bedingungen für das Mapping (die auf Multicore-Architekturen und andere Hardware-/Softwarekomponenten angewendet werden können), das Nachrichtenrouting für Fahrzeugnetztopologien, die zeitgesteuerte Planung für Anwendungsthreads und Kommunikationsaufgaben, Sicherheits- und Nicht-

Sicherheitsanforderungen sowie Optimierungs- und Grenzziele (siehe Abbildung 1). Im letzten Schritt, der im mittleren Teil von Abbildung 1 dargestellt ist, wird ein MIP-Solver eingesetzt, um die vorgegebenen Randbedingungen unter Berücksichtigung der definierten Optimierungsziele zu lösen und zu optimieren.Nach dem Lösungsschritt bietet das E/E-Designer-Framework schließlich eine automatische Zuordnung oder Ressourcenzuweisung (z. B. die automatische Zuweisung verschiedener Anwendungsthreads zu verschiedenen Kernen einer Multi-Core-HPCU) und erstellt Nachrichtenrouten (d. h. die Suche nach dem korrekten Pfad vom Sender einer Kommunikationsnachricht zu ihrem Empfänger) für eine modellierte E/E-Architektur oder Netzwerktopologie, die die vorgegebenen Anforderungen erfüllt. Darüber hinaus berechnet der E/E-Designer zeitgesteuerte Zeitpläne für die zugewiesenen laufenden Anwendungsthreads, z. B. auf einer HPCU mit mehreren Kernen, und verwaltet das Routing der Kommunikationsaufgaben über die Netzwerkverbindungen des Fahrzeugs. Die resultierende Lösung, die das Werkzeug ausgibt, erfüllt alle vom E/E-Systemmodellierer ausgewählten Anforderungen und wird auf der Grundlage der gewählten Optimierungsziele optimiert [1, 4].

2.2.2 Vorgeschlagener modellbasierter Software-Rahmen

Bei dem vorgestellten modellbasierten Rahmenwerk handelt es sich um ein Software-Tool zum Entwurf/Modellieren und Synthetisieren von Fahrzeug-E/E-Architekturen (siehe Abb. 2) [1, 2, 4, 7]. Es umfasst ein Frontend, in dem der Benutzer eine gewünschte Fahrzeugtopologie durch Ziehen und Ablegen verschiedener Software- und Hardwarekomponenten entwerfen kann. Das Werkzeug verwendet die Methode der Design Space Exploration (DSE), um verschiedene Syntheseprobleme zu lösen, wie z. B.

- Automatische Zuordnung von Anwendungen zu Kontrollknoten, z. B. elektronischen Steuergeräten (ECU)
- Berechnung von zeitgesteuerten Zeitplänen für Anwendungs-Threads, die auf Kontrollknoten laufen
- Erstellung von Nachrichten-Routen für die Übertragung von Kommunikationsnachrichten von Sendern zu Empfängern, einschließlich einfacher, mehrfacher, redundanter und homogener redundanter Routen
- Berechnung von zeitgesteuerten Zeitplänen für Kommunikationsaufgaben, die über die Verbindungen des Netzes geleitet werden
- Berücksichtigung verschiedener Randbedingungen und Optimierungsziele, die vom Benutzer vor der Lösung des entworfenen Modells ausgewählt werden können, einschließlich Verbindungsbelegungsrate, maximale Bandbreitennutzung, maximale Ressourcennutzung, Ressourcennutzung, Kostenreduzierung, Antwortzeit, End-to-End-Latenz und Zuverlässigkeit
- Vorschlag eines Konzepts zur Ermittlung der Ursache für die Undurchführbarkeit des Modells. Dies zielt darauf ab, die Beschränkungen zu finden, die das Systemmodell unlösbar machen

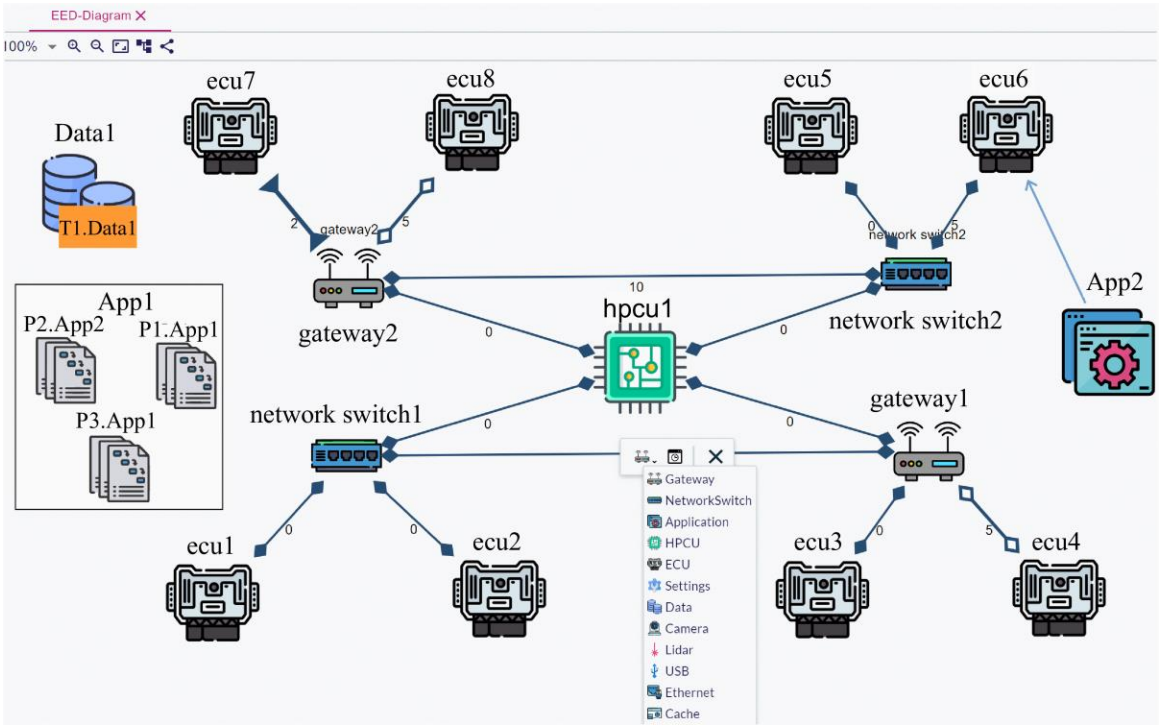

Abbildung 2: Ein Beispiel für ein zonales E/E-Architekturmodell unter Verwendung des E/E Designer Framework.

2.2.2.1 Modellierung

Mit dem E/E Designer Framework können Benutzer E/E-Architekturen und Fahrzeugtopologien grafisch modellieren. Abb. 2 zeigt ein mit unserem Tool erstelltes Beispiel. Das E/E-Designer-Framework verwendet ein webbasiertes Frontend. Im Folgenden werden mehrere Vorteile webbasierter Werkzeuge erläutert. Einer der Hauptvorteile von webbasierten Modellierungswerkzeugen ist ihre Zugänglichkeit. Auf sie kann von jedem Gerät mit Internetanschluss aus zugegriffen werden, so dass keine speziellen Softwareinstallationen erforderlich sind [1].

Nach der Erstellung eines Projekts können die Benutzer verschiedene HW/SW-Komponenten auswählen, z. B. Gateways, Netzwerk-Switches, Anwendungen (einschließlich Threads), ECUs (einschließlich Unterkomponenten wie Prozessoren und Speicher), HPCUs (bestehend aus Unterkomponenten wie Prozessoren, Kernen, Speicher und GPUs), Kommunikationsnachrichten (in diesem Fall Daten), Kommunikationsaufgaben und mehr. Wenn Sie auf ein leeres Feld klicken, wird ein Fenster mit den verfügbaren Objekten angezeigt, die ausgewählt und erstellt werden können. Darüber hinaus kann der Benutzer erzwingen, dass Anwendungen vor dem Lösungsschritt bestimmten HW-Komponenten zugewiesen werden.

In Abb. 2 wird zum Beispiel App_2 gezwungen, auf ECU_6 zu laufen (blauer Pfeil). Das Framework enthält auch eine Option zur Erzeugung einer vollmaschigen Netzwerktopologie für ausgewählte HW-Knoten, bei der jeder Knoten direkt mit allen anderen Knoten verbunden ist.

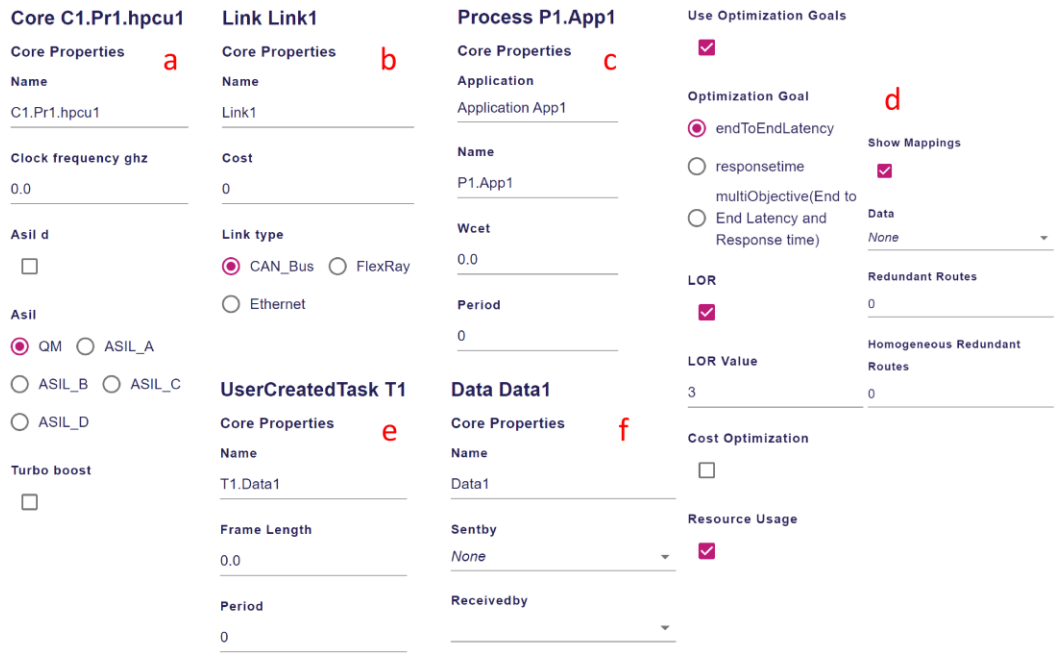

Abbildung 3: Eigenschaften von Software- und Hardwarekomponenten im E/E Designer Tool.

2.2.2.2 Anforderungen und Eigenschaften

Neben der Modellierung können auch die Anforderungen und Eigenschaften jeder Komponente bestimmt werden, die für die Lösung eines entworfenen Modells unerlässlich sind. Die Details zu jeder Komponente werden auf der rechten Seite des Modellierungsfensters angezeigt. Abb. 3 veranschaulicht die Details zu verschiedenen Komponenten und die Lösungseinstellungen. Wie in Abb. 3 (a) gezeigt, kann ein Kern als Teil der HPCU verschiedene ASIL-Stufen als sicherheitskritische Eigenschaft, eine Turbo-Boost-Funktion, eine definierte Taktfrequenz und einen beliebigen Namen haben. Für einen Link können der Typ (Ethernet, FlexRay und zeitgesteuerter CAN-Bus), die maximale Bandbreitenkapazität, die Kosten und der Name ausgewählt werden (siehe Abb. 3 (b)); darüber hinaus hat jeder Link-Typ ein eindeutiges Visualisierungssymbol, z. B. sind ECU_7 und ECU_8 mit zwei verschiedenen Links zu $gateway_2$ verbunden (siehe Abb. 2). Die Ausführungszeit, der Zeitraum und der Name jedes Threads (im Werkzeug Prozess genannt) sind die Thread-Eigenschaften in Abb. 3 (c) [1]. Es sollte hinzugefügt werden, dass die Anwendung selbst in den Anwendungseigenschaften als sicherheitskritisch oder nicht sicherheitskritisch definiert werden kann. Jede Kommunikationsaufgabe enthält einen Namen, die Rahmenlänge und die Periode als Eigenschaften, die in Abb. 3 (e) dargestellt sind. Auch der Absender und der Empfänger einer Kommunikationsnachricht können in den Datendetails in Abb. 3 (f) angegeben werden. Wie bereits erwähnt, unterstützt unser Rahmenwerk mehrere Anforderungen, Optimierungsziele und auch die Mehrzieloptimierung. Daher haben wir eine Einstellung in das Front-End integriert, in der der Benutzer die Art der Multi-Objektive auswählen und auch die gesamte Optimierungsmethode aktivieren oder deaktivieren kann (Kontrollkästchen *Use Optimization Goals* in Abb. 3 (d)) sowie jedes Ziel und jede Anforderung selbst, z. B. LOR maximal tolerierbare Grenze, maximale Bandbreitennutzung, Ressourcennutzung, maximale Ressourcennutzung und Kosten (siehe Abb. 3 (d)).

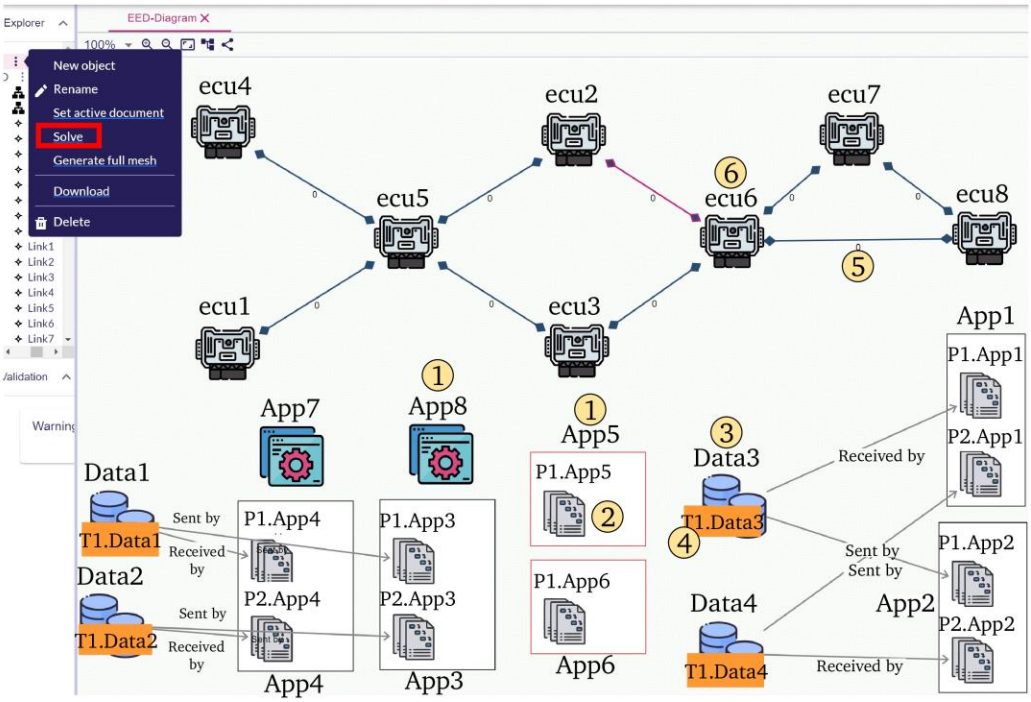

Abbildung 4: Eine mit dem vorgestellten Werkzeug modellierte E/E-Architektur mit Anwendungen (Nr. eins), Anwendungs-Threads (Nr. zwei), Kommunikationsnachrichten (Nr. drei), Kommunikationsaufgaben (Nr. vier), Verbindungen (Nr. fünf) und Steuergeräten (Nr. sechs). Das Modell wird durch Klicken auf die Option *Solve* (rotes Rechteck) gelöst und optimiert [1].

In derselben Abbildung können die Anzahl der redundanten und homogenen redundanten Routen sowie die sicherheitskritischen Anforderungen geändert werden. Um die Lösung im Frontend übersichtlich darzustellen, werden die erstellten Routen und Mappings nur für zusammenhängende Meldungen angezeigt. Der Benutzer kann dies im Abschnitt Daten in Abb. 6 (d) auswählen.

2.2.2.3 Lösungsansätze und Lösungen

Zum Lösen und Optimieren eines erstellten Modells unter Berücksichtigung der definierten Eigenschaften, Anforderungen und Optimierungsziele, z. B. des in Abb. 4 gezeigten Modells, ist eine Option *Solve* integriert [1]. Wenn Sie auf die Option *Solve* klicken, kann das Tool die mit der modellierten Architektur verbundenen Probleme lösen, wobei die Lösung optimiert wird und die festgelegten Anforderungen erfüllt werden. In dem in Abb. 4 gezeigten Modell gibt es acht Anwendungen, von denen sechs einen oder zwei Threads mit denselben Perioden und unterschiedlichen Ausführungszeiten und zwei Anwendungen ohne Thread enthalten. Zwei Anwendungen sind hier als sicherheitskritisch ausgewählt (App_5 und App_6 mit einem roten Rahmen dargestellt). Außerdem besteht das Modell aus vier Kommunikationsnachrichten mit denselben Zeiträumen wie die Anwendungsthreads und verschiedenen Rahmenlängen. Ziel ist es, die Anwendungen automatisch acht Steuergeräten zuzuordnen und dabei bestimmte Anforderungen zu erfüllen.

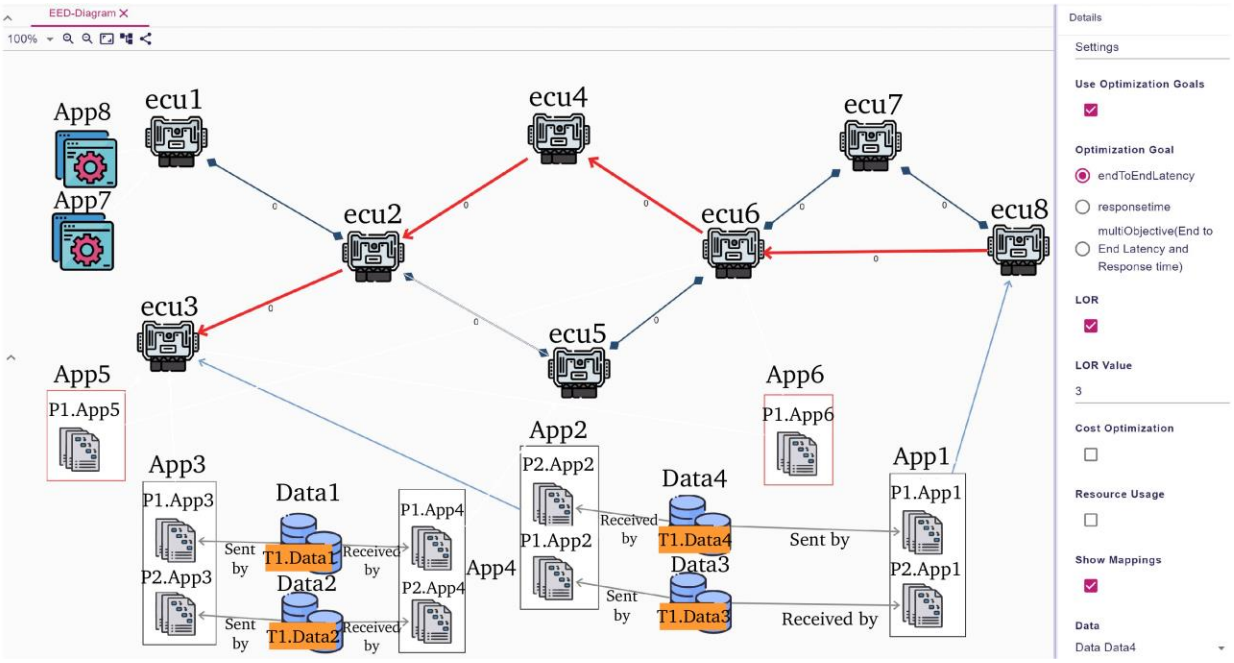

Abbildung 5: Eine Lösung des entworfenen Modells in Abb. 4 einschließlich Mapping, Nachrichtenrouting und Scheduling. Hier werden nur die Mappings für die Anwendungen eins und zwei in Bezug auf die Kommunikationsnachricht vier angezeigt [1].

Es muss sichergestellt werden, dass die auf jedem Steuergerät laufenden Threads die korrekten zeitgesteuerten Zeitpläne haben und dass alle Bedingungen für ein automatisches Mapping von sicherheits- und nicht sicherheitskritischen Anwendungen erfüllt sind [1, 4]. Nach der Lösung des Modells kann der Benutzer die optimierten Lösungen für das Mapping, das Scheduling von Anwendungsthreads und Kommunikationsaufgaben sowie das Nachrichten-Routing betrachten. Abb. 5 zeigt die Lösung des in Abb. 4 dargestellten Entwurfsmodells, die Nachrichten-Routing, Mapping und Scheduling für Anwendungs-Threads und Kommunikationsaufgaben umfasst [1]. Als Mapping-Lösung bestimmt das E/E-Designer-Tool, welche Anwendungen welchen Steuergeräten zugeordnet werden sollen (z.B. blaue Pfeile in Abb. 5). Die aktuellen Mapping-Anforderungen in diesem Beispiel sind Redundanzbedingungen für sicherheitskritische Anwendungen, und die Einschränkung stellt sicher, dass die auf jedem Steuergerät laufenden Threads nicht Sender und Empfänger der gleichen Kommunikationsnachricht sind. In Abb. 5 werden nur Mappings visualisiert, die sich auf Sender- und Empfängeranwendungen der Kommunikationsnachricht $Data_4$ beziehen. Es ist wichtig zu beachten, dass die Visualisierung in Abb. 5 ausschließlich die abgebildeten Anwendungen hervorhebt, die sowohl als Sender als auch als Empfänger für eine ausgewählte Kommunikationsnachricht (hier die Kommunikationsnachricht Nummer vier) fungieren. Der Grund für diese Entscheidung liegt in der Verbesserung der Klarheit und Verständlichkeit des Modells, vor allem wenn es sich um ein sehr komplexes System handelt. Durch die Fokussierung auf diesen spezifischen Aspekt wird potenzielle Verwirrung gemildert, was zu einer klareren Darstellung führt, die ein intuitiveres Verständnis des Mapping-Szenarios ermöglicht. Abb. 6 zeigt die berechneten Zeitpläne für die Anwendungsthreads, die auf ECU_3 in Abb. 5 innerhalb einer berechneten Hyperperiode ausgeführt werden. Wie zu erkennen ist, steht jede Farbe für eine Anwendung und jeder Slot für

den Zeitplan eines Threads. Jeder Slot kann innerhalb der Hyperperiode mehrfach ausgeführt werden, abhängig von der Periode des Threads.

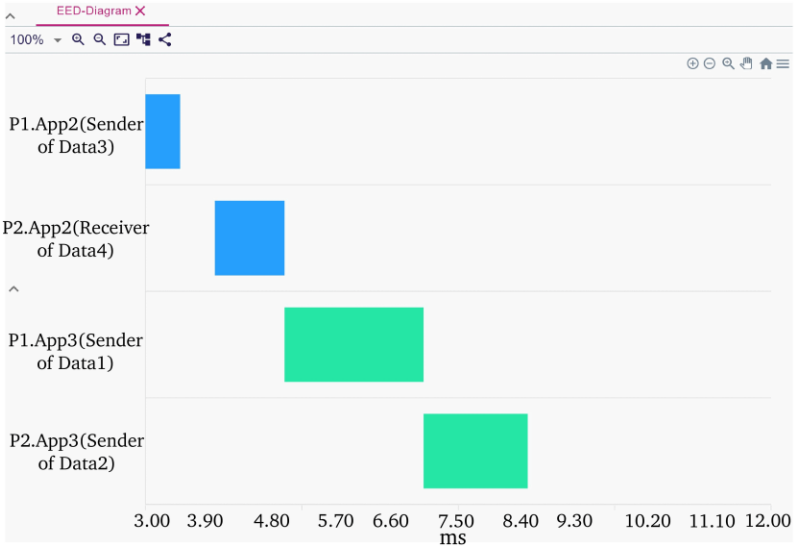

Abbildung 6: Eine Lösung einer modellierten E/E-Topologie, einschließlich Mapping und Nachrichtenrouting. Auf der linken Seite werden mehrere Warnungen bezüglich der Validierung des Modells angezeigt [1].

Eine erstellte Route für eine bestimmte Kommunikationsnachricht zur Weiterleitung einer Nachricht von einem Sender zu einem Empfänger kann im Frontend angezeigt werden. In Abb. 5 müssen vier Kommunikationsnachrichten von Sendern zu Empfängern gesendet werden. Jede Nachricht umfasst eine Kommunikationsaufgabe, die über die Netzwerkverbindungen geleitet wird, und der Benutzer muss vor dem Lösungsschritt die Threads als Sender und Empfänger jeder Nachricht im Frontend angeben. Daher kann der Benutzer nach dem Lösen des Modells den korrekten Pfad für jede Kommunikationsnachricht und die zugehörigen Zuordnungen für Sender- und Empfängeranwendungen, die für dieselbe Kommunikationsnachricht relevant sind, beobachten, indem er eine gewünschte Nachricht auswählt. Basierend auf Abb. 5, zeigt der rote Pfad eine generierte Route von ECU_8 als Sender zu einem Empfänger ECU_3, der die Kommunikationsnachricht d_4 weiterleitet.

Ähnlich wie beim Scheduling für Anwendungsthreads werden die Slots für die Kommunikationsaufgaben während der entsprechenden Hyperperiode angezeigt. Es sollte hinzugefügt werden, dass bei der in Abb. 5 dargestellten Lösung das Ziel der Ende-zu-Ende-Latenzoptimierung verfolgt wurde. Darüber hinaus können die berechneten zeitgesteuerten Zeitpläne für vier Kommunikationsaufgaben, die über Links geleitet werden, ähnlich wie in Abb. 6 visualisiert werden [1].

2.2.2.4 Modell-Validierung

Da verschiedene Modellanforderungen erfüllt sein müssen, um ein gültiges Modell zu erstellen, können Fehler, Warnungen und allgemeine Hinweise den Benutzer bei der Erstellung eines gültigen Modells unterstützen. Die Validierung des vorgeschlagenen Werkzeugs ist ein kontinuierlicher Prozess, der das Modell nach jeder Änderung prüft und dann bei Bedarf die Validierungsmeldungen anzeigt.

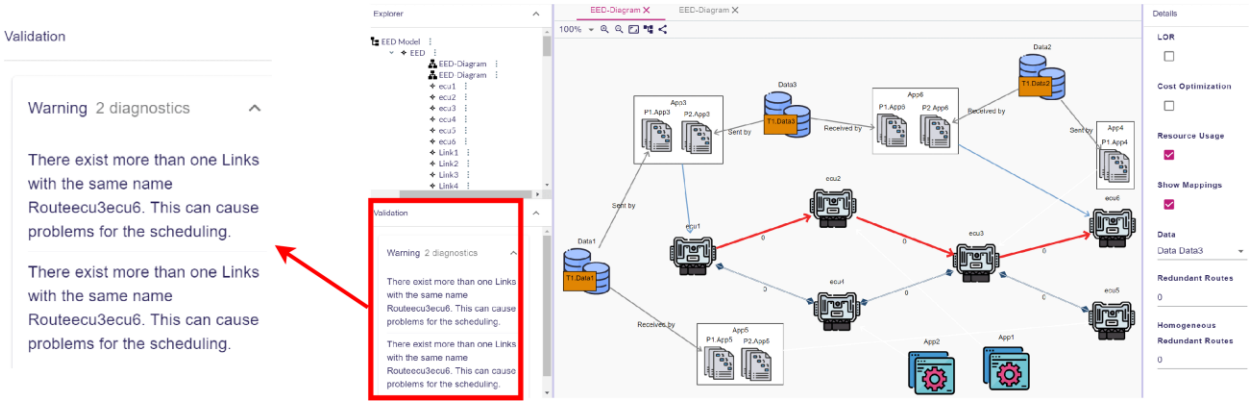

Abbildung 7: Eine Lösung eines vorgefertigten Modells mit Mapping und Nachrichtenrouting. Auf der linken Seite werden mehrere Meldungen zur Validierung des Modells angezeigt.

Die Validierungsmeldung besteht aus einer Problembeschreibung und enthält die Elemente, Mappings oder Attribute, die zu einem Problem führen. Darüber hinaus gibt die Meldung die Schärfe des Problems und die Dringlichkeit einer Änderung des Modells an. Auf dieser Grundlage führen wir drei verschiedene Nachrichtentypen ein, darunter *error*, *warning* und *message*. Ein *error* erscheint, wenn eine Modellverletzung auftritt, die kritische Sicherheitsanforderungen verletzt oder zu einer Verletzung von ILP-Randbedingungen führt. Dieser Meldungstyp bedeutet, dass der Benutzer das Modell ändern muss, während ein *warning* für Verstöße verwendet wird, die nicht begangen werden sollten, aber kein ernstes Problem darstellen. Schließlich visualisiert das System einen *message* für einfache Tipps, Hinweise und andere Informationen. In Abb. 7 gibt es zum Beispiel mehrere *warnings* für das entworfene Modell [1].

2.2.3 Ansatz zur Analyse von Konstruktionsfehlern

Der Ansatz der Entwurfsfehleranalyse zielt darauf ab, die Lücke zu schließen, die bei der Identifizierung der für die Undurchführbarkeit eines Modells in einem Constraint-System verantwortlichen Nebenbedingungen besteht. Während sich frühere Studien auf die Generierung von Mengen nicht erfüllbarer oder undurchführbarer Nebenbedingungen konzentriert haben, wurden die spezifischen Nebenbedingungen, die die Undurchführbarkeit des Systems verursachen, noch nicht gründlich erforscht. Dieser Aspekt ist im Zusammenhang mit dem von uns vorgestellten Werkzeug von Bedeutung, da die Identifizierung verletzter Nebenbedingungen nach der Lösung komplexer E/E-Systemmodelle eine komplizierte und zeitaufwändige Aufgabe ist. Folglich wird ein Ansatz zur Identifizierung der wichtigsten erfüllbaren Randbedingungen eingeführt, der auf den Algorithmen aufbaut, die minimale nicht erfüllbare Mengen erzeugen. Um die unbefriedigbaren Mengen zu extrahieren, verwendet der vorgeschlagene Ansatz ein Verfahren, das irreduzible infeasible subsystem (IIS) genannt wird [8, 9], das im Folgenden erläutert wird.

2.2.3.1 Irreducible Inconsistent Subsystem

Ein Irreducible Inconsistent Subsystem (IIS) ist äquivalent zu Minimal Unsatifiable Subset (MUS) im SAT Problem. Ein IIS ist eine Teilmenge der Constraints und Variablengrenzen mit den folgenden Eigenschaften:

- Sie ist immer noch nicht durchführbar, und

- Wird eine einzige Bedingung oder ein einziger Vorbehalt aufgehoben, wird das Teilsystem machbar.

Um die nicht erfüllbaren Mengen zu extrahieren, berechnen wir die IIS für die nicht erfüllbare Menge mit dem Gurobi Solver [10]. Der vorgeschlagene Ansatz unter Verwendung der IIS-Methode besteht aus mehreren Schritten, die in Abb. 8 als Flussdiagramm dargestellt sind. Der Prozess beinhaltet die Bereitstellung eines nicht erfüllbaren Satzes von MIP-Randbedingungen und leeren Listen als Eingaben. Im nächsten Schritt wird die Berechnung der IIS aus den nicht erfüllbaren Nebenbedingungen durchgeführt, und die Ergebnisse werden in einer Liste gespeichert. Eine IIS stellt eine minimale Teilmenge von Nebenbedingungen dar, die das System, wie bereits erläutert, undurchführbar macht. Dies bedeutet, dass die Einbeziehung beliebiger Nebenbedingungen in eine IIS zu einer undurchführbaren Lösung führt. Nach der Berechnung der IIS wird im nächsten Schritt jede Bedingung innerhalb der IIS-Liste untersucht. Eine Bedingung nach der anderen wird aus der Menge der unzulässigen Bedingungen entfernt. Die IIS-Berechnung wird dann erneut auf den aktualisierten Satz der undurchführbaren Bedingungen angewandt, wobei die entfernte Bedingung ausgeschlossen wird. Die Durchführbarkeit des Modells wird auf der Grundlage des Ergebnisses der IIS-Berechnung während dieser Iteration bewertet. Wenn das Modell nach Ausschluss der Bedingung machbar wird, zeigt dies, dass die entfernte Bedingung wesentlich zur Unerfüllbarkeit des gesamten Systemmodells beiträgt. Um diesen Einfluss zu erfassen, wird der ausgeschlossenen Bedingung ein Gewicht zugewiesen und in einer separaten Liste gespeichert. Das zugewiesene Gewicht einer Bedingung ist proportional zu der Anzahl der Fälle, in denen sie als Ursache für die Unzulässigkeit des Modells identifiziert wurde, wenn sie aus der Menge der unzulässigen Bedingungen ausgeschlossen wurde. Gemäß dem Flussdiagramm in Abb. 8 wird der Prozess fortgesetzt, bis alle Nebenbedingungen in der IIS-Liste abgedeckt sind. Als Ergebnis wird eine Liste von Nebenbedingungen zusammen mit ihren jeweiligen Gewichtungen zur Verfügung gestellt.

Schließlich werden unter Berücksichtigung der den einzelnen Nebenbedingungen zugewiesenen Gewichte die wichtigsten Nebenbedingungen ermittelt, die zur Unerfüllbarkeit des Systemmodells beitragen. Mit Hilfe dieser identifizierten Randbedingungen können die Benutzer das Modell korrigieren, indem sie die Probleme der kritischen Randbedingungen angehen. Dies kann zum Beispiel die Validierung der bereitgestellten Eingaben als Anforderungen an das Werkzeug beinhalten. Nachdem die notwendigen Anpassungen vorgenommen wurden, kann das entworfene E/E-System bei einer erneuten Lösung als zufriedenstellend angesehen werden. Bei der Bestimmung einer IIS für ein nicht durchführbares Modell bestimmt der Parameter *ConstrIISForce* als ein in den Gurobi-Optimierer [10] integriertes Attribut die Einbeziehung oder den Ausschluss einer allgemeinen Einschränkung in die IIS. Wenn der Wert standardmäßig auf *-1* gesetzt ist, wird die Entscheidung dem IIS-Algorithmus selbst überlassen. Wenn der Wert standardmäßig auf *-1* gesetzt ist, wird die Entscheidung dem IIS-Algorithmus selbst überlassen. Wird das Attribut dagegen auf *0* gesetzt, wird die Bedingung als nicht geeignet für die Einbeziehung in die IIS betrachtet. Wenn der Wert standardmäßig auf *-1* gesetzt ist, wird die Entscheidung dem IIS-Algorithmus selbst überlassen. Wird das Attribut dagegen auf *0* gesetzt, wird die Bedingung als nicht geeignet für die Einbeziehung in die IIS betrachtet.

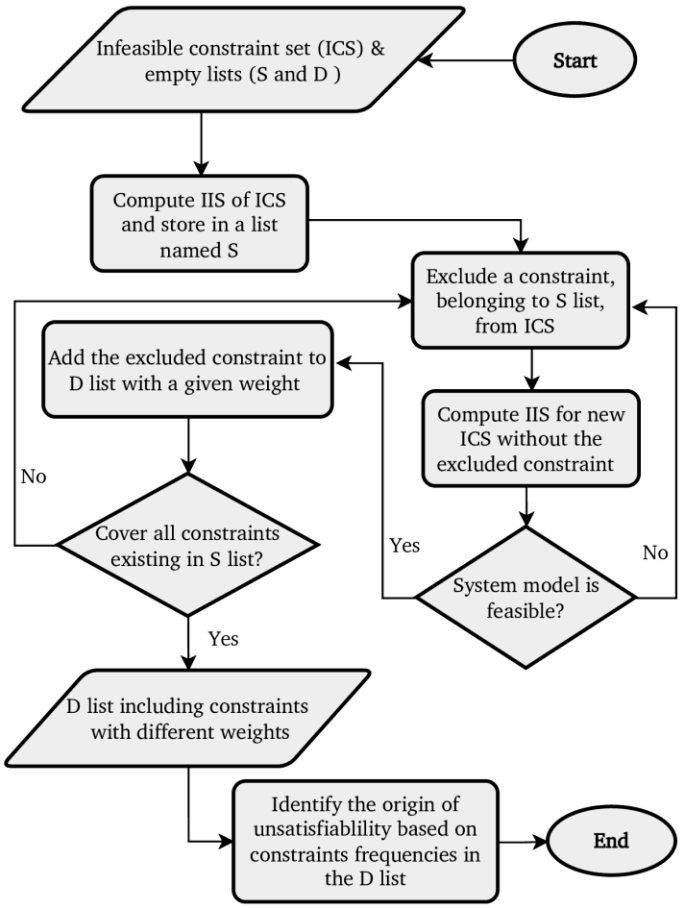

Abbildung 8: Das Flussdiagramm der Entwurfsfehleranalyse nach der IIS-Methode.

Umgekehrt garantiert die Einstellung des Attributs auf *1* die Aufnahme des Constraints in die IIS, wobei der Algorithmus jegliche Überlegungen zu seiner Entfernung außer Acht lässt [10].

2.2.3.2 Bewertung des Ansatzes zur Analyse von Konstruktionsfehlern

In diesem Unterabschnitt werden die mit den vorgestellten Methoden erzeugten Entwurfsfehlerlösungen dargestellt und analysiert. Für den IIS-Ansatz, die primäre Methode zur Analyse von Entwurfsfehlern, die in unserem vorgestellten Werkzeug verwendet wird, werden Lösungen für vier nicht realisierbare Fallstudien präsentiert, wie in Abb. 9 dargestellt. Abb. 9 (a) zeigt die Gewichtung der einzelnen Constraints in einem nicht realisierbaren Systemmodell nach der Lösung, das 15 Steuergeräte, sechs Anwendungen und drei Kommunikationsnachrichten enthält. Die erstellte Analyse zeigt, dass nur acht von 483 Constraints ein Gewicht größer als Null haben. Das bedeutet, dass nur diese neun Nebenbedingungen das Entwurfsmodell unbefriedigend machen. Je höher die Gewichtung einer Randbedingung ist, desto größer ist ihr Einfluss auf die Unerfüllbarkeit des Systemmodells. In Abb. 9 (a) zum Beispiel wurden die Ausführungszeiten der Anwendungs-Threads nahe an ihren jeweiligen Zeiträumen definiert. Da dieser Fehler mehrere Beschränkungen betrifft, wurden verschiedene Beschränkungen entsprechend ihrer Bedeutung für die Entstehung eines unbefriedigenden Modells gewichtet. In diesem Beispiel hat c_1 die höchste Gewichtung in Bezug auf die Beschränkungen des Nachrichtenroutings, während die anderen Beschränkungen mit der Abbildung und den zeitgesteuerten Scheduling-Bedingungen verbunden sind. Um das Systemmodell zu korrigieren,

müssen die gewichteten Randbedingungen analysiert werden, um die Ursache der Verletzung zu ermitteln.

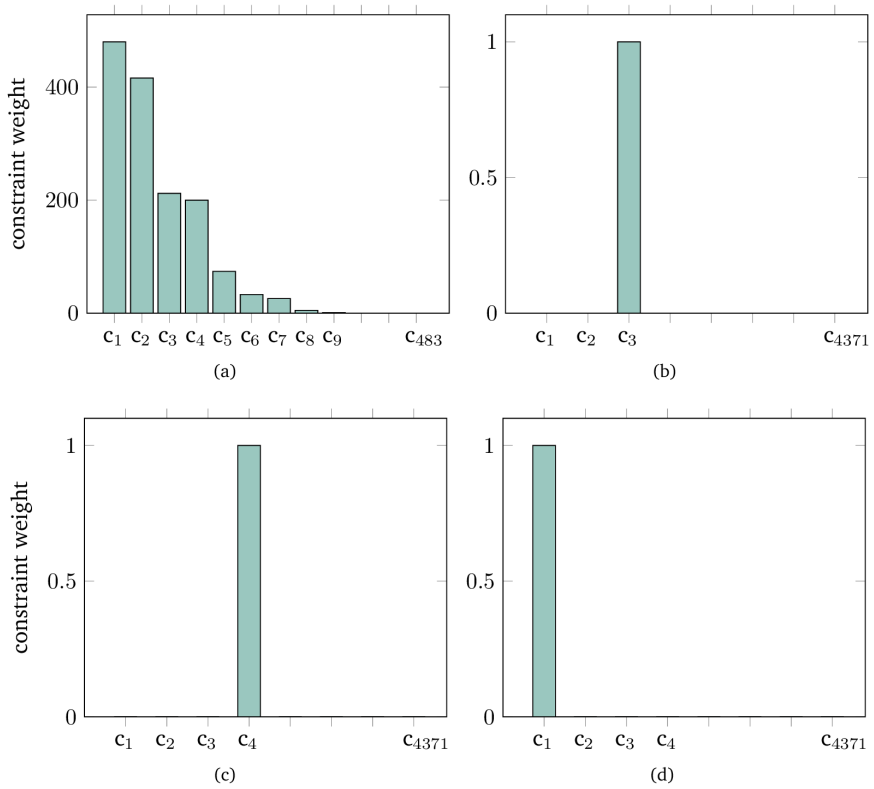

Abbildung 9: Die Lösungen des Ansatzes der Konstruktionsfehleranalyse wurden für verschiedene Fallstudien mit der IIS-Methode erstellt.

In Abb. 9 (b) wurde die Quelle der Verletzung für ein modelliertes E/E-System mit 15 Steuergeräten, 60 Anwendungen und 30 Kommunikationsnachrichten identifiziert. In diesem Modell wurde die Framelänge einer Kommunikationsnachricht so festgelegt, dass sie größer als ihre Periode ist. Wie zu beobachten ist, hat von den 4371 Constraints nur das Constraint c_3 ein Gewicht, was darauf hinweist, dass die Bedingung, dass die Framelänge einer Nachricht kleiner als ihre Periode sein muss, verletzt wurde. Da diese Verletzung die anderen Bedingungen nicht betrifft, erhielt nur c_3 eine Gewichtung nach der Entwurfsfehlerberechnung mit dem IIS-Ansatz. Im folgenden Szenario wird das gleiche Modell wie in Abb. 9 (b) verwendet. Diesmal wird jedoch die Ausführungszeit eines Anwendungsthreads höher angesetzt als seine Periode. Wie in Abb. 9 (c), der Lösung des vorgeschlagenen Ansatzes unter Verwendung von IIS, zu sehen ist, erhielt die zugehörige Bedingung c_4 eine Gewichtung aus den anderen 4371 Bedingungen. Daher kann der Benutzer ein realisierbares Modell erhalten, indem er das Problem im Zusammenhang mit dieser spezifischen Bedingung während des Lösungsprozesses korrigiert.

Schließlich wurde in Abb. 9 (d) eine Routing-Bedingung im Systemmodell geändert, was zu einem nicht durchführbaren Modell nach dem Lösungsschritt führte. Diese Änderung wurde vorgenommen, um zu überprüfen, ob der vorgestellte Ansatz die Quelle der Verletzung genau identifizieren kann. Nach Anwendung des Entwurfsfehler-Ansatzes wurde die Routing-Bedingung (c_1 in Abb. 9 (d)) erfolgreich identifiziert. Beachten Sie, dass die zur Identifizierung der Quelle der Unerfüllbarkeit erforderliche Rechenzeit davon abhängt, ob die Konfliktquelle eine einzelne oder

mehrere Nebenbedingungen betrifft. Sie hängt auch davon ab, ob der Fehler selbsterklärend ist, wie er durch die Nebenbedingungen definiert ist, oder ob der Fehler erst nach dem Lösen der Hälfte des Systemmodells sichtbar wird, weil Werte hinzugefügt wurden, die das System undurchführbar machen.

In dem in Abb. 9 (a) gezeigten Anwendungsfall wurde zum Beispiel die Ausführungszeit eines Threads so definiert, dass sie nahe an seiner Periode liegt. Dies scheint zunächst keine Konflikte zu verursachen, die auf den Zeitplanungsbeschränkungen beruhen. Nachdem jedoch die korrekten Zeitpläne für mehrere Anwendungen ermittelt wurden, zeigte sich, dass die Überschneidungen zwischen Threads aufgrund des geringen Abstands zwischen ihrer Ausführungszeit und ihrer Periode zu Konflikten bei anderen Bedingungen, wie z. B. dem Routing, führten. Daher spielt die Berechnungszeit in umfangreichen Modellen eine zentrale Rolle, vor allem wenn der Fehler mehrere Bedingungen betrifft.

2.2.4 Bewertung

Wir evaluieren die Leistung, Anwendbarkeit und Skalierbarkeit unseres Frameworks zur Entwurfszeit, indem wir sieben Experimente durchführen, und zur Laufzeit, indem wir die Lösungen des vorgeschlagenen Tools, einschließlich Mapping, Message Routing und Scheduling, in einem Versuchsaufbau einsetzen.

2.2.4.1 Bewertung zur Entwurfszeit

Der von uns vorgeschlagene Rahmen erleichtert den Entwurfsprozess für Systemarchitekten. Allerdings wirken sich die hohen Rechenzeiten für die für die Generierung und Lösung der Constraints einen negativen Einfluss auf die praktische Anwendung haben. Daher untersuchen wir, wie sich verschiedene Parameter des mit dem vorgeschlagenen Werkzeug entworfenen Systems auf die Zeit auswirken, die für die Generierung und Lösung der in unserem Rahmenwerk verwendeten Constraints benötigt wird. Darüber hinaus stellen wir eine Skalierbarkeitsanalyse vor und zeigen, dass unsere einstufigen Lösungsalgorithmen und unsere Formulierung auf Systeme mit angemessener Größe skalieren. Wir haben *Gurobi 9.5* [10] zur Lösung der Systemmodelle verwendet und alle Experimente zur Entwurfszeit auf einem Laptop mit einem 2,80 GHz Core i7 Prozessor und 32 GB Speicher ausgeführt. In den ersten drei Fallstudien lösen wir das Mapping-Problem, indem wir das Ziel der Ressourcennutzung (RU) als harte Randbedingung und das Scheduling für Threads, die auf jedem Kontrollknoten laufen, unter Verwendung des vorgestellten Frameworks anwenden. In der ersten Fallstudie entwerfen wir ein verteiltes System, bestehend aus 10 Anwendungen, die jeweils zwei Threads mit zufälligen Ausführungszeiten $t_i.e$ und geraden Perioden $t_i.p$ enthalten, und 10 Steuergeräten als Kontrollknoten. In dieser Fallstudie wird die Systemgröße von 10 auf 50 Anwendungen und Steuergeräte erhöht, um die Lösungszeit des Constraintsatzes und die Zeit für die Generierung von MIP-Constraints zu messen (siehe Abb. 10 (a)). In der zweiten Fallstudie wird ein verteiltes System mit 8 Steuergeräten und zehn Anwendungen, die jeweils aus zwei Threads bestehen, modelliert, und es werden die gleichen Constraints und Lösungsoptionen wie im ersten Anwendungsfall verwendet. Wir erhöhen die Anzahl der Anwendungen von 10 auf 70, während wir die Anzahl der Steuergeräte konstant halten, um das Timing-Verhalten für das Lösen und Generieren von Constraintsätzen zu beobachten.

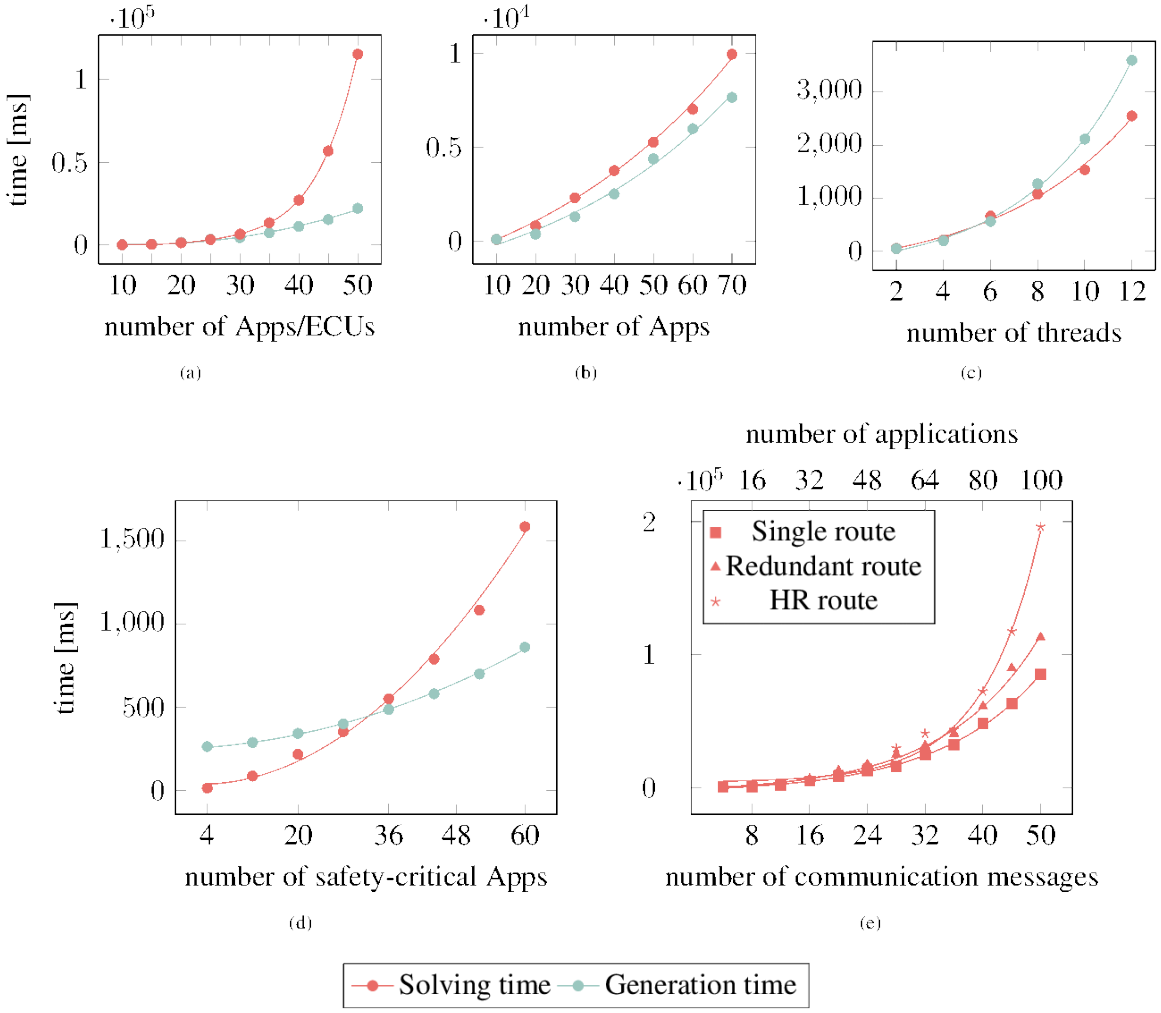

Abbildung 10: Bewertung der Leistung des E/E Designer zur Entwurfszeit. (a) Jede Anwendung besteht aus zwei Threads. (b) Es wird auf eine Topologie mit 8 Steuergeräten angewendet, und jede Anwendung umfasst zwei Threads. (c) Die Topologie besteht aus 8 Anwendungen und 8 Steuergeräten. (d) In diesem Anwendungsfall wird eine zonale Architektur mit 15 Steuergeräten eingesetzt, und jede sicherheitskritische Anwendung hat einen Thread. Die Grenzziele für die Ressourcenauslastung und die maximale Speichernutzung sowie die sicherheitskritischen Mapping-Beschränkungen werden angewendet. (e) Die gleiche Architektur wie in (d) wird verwendet. Diesmal wird eine Mehrzieloptimierung durchgeführt, die die End-to-End-Latenz, die Antwortzeit und die LOR sowie die Ziele für die Ressourcennutzung und die maximale Speichernutzung umfasst.

Basierend auf Abb. 10 (b) steigt die Lösungszeit exponentiell an, da mehr Thread-Schedules für jedes Steuergerät berechnet werden müssen. Betrachtet man beispielsweise 70 Anwendungen, die 8 Steuergeräten zugeordnet sind, so bedeutet dies, dass jedes Steuergerät mindestens acht Anwendungen oder 16 Threads (unter Berücksichtigung des EVU-Ziels) ausführt, die korrekte Zeitpläne haben müssen [1, 4].

Wir führen die exakte Messung mit einem Modell durch, das acht Anwendungen und 8 Steuergeräte umfasst, aber in diesem Anwendungsfall erhöhen wir nur die Anzahl der Threads von 2 auf 12 für jede Anwendung, während wir die Perioden aller Threads als ungerade Werte festlegen. Wie Abb. 10 (c) zeigt, sind exponentielle Zuwächse bei den Lösungs- und Generierungszeiten zu erkennen. Das Finden von Zeitplänen für mehrere Threads mit ungeraden Perioden ist aufgrund des kleinsten gemeinsamen Multiplikators und der Hyperperiodenberechnungen komplizierter als bei geraden Perioden. Abb. 10 (d) zeigt

Messungen zur Lösung eines Mapping-Problems und zur Ermittlung korrekter Zeitpläne für Anwendungsthreads. Es wurde eine zonale Architektur, ähnlich wie in Abb. 2, modelliert, die 15 Steuergeräte umfasst. Die Anzahl der Anwendungen wurde von 4 auf 60 erhöht, wobei alle Anwendungen, jeweils einschließlich eines Anwendungsthreads, als sicherheitskritisch definiert wurden. Während des Lösungsschritts werden die Steuergerätenutzung und die maximale Speicherauslastung jedes Steuergeräts als Randbedingungen festgelegt und weitere Mapping-Bedingungen, wie in den automatisierten Mapping-Bedingungen erläutert, wie z.B. Redundanz für sicherheitskritische Anwendungen, einbezogen. Da die sicherheitskritische Anwendung redundant ausgeführt werden muss, laufen im Fall von 60 sicherheitskritischen Anwendungen 120 Anwendungen in der Topologie. In Abb. 10 (e) untersuchen wir den Einfluss der Erstellung verschiedener Arten von Routen auf die Lösungszeit bei Anwendung der automatischen Zuordnung und der Einplanung von Anwendungsthreads und Kommunikationsaufgaben in einem einzigen Schritt für dieselbe zonale E/E-Architektur, die in Abb. 10 (d) modelliert wurde. Wir messen die Lösungszeit für die Erstellung von drei Arten von Routen: einzelne, redundante und HR-Pfade (die drei disjunkte Routen erzeugen) von Sendern zu Empfängern (die automatisch auf der Grundlage von Mapping-Bedingungen ausgewählt werden), während wir die Anzahl der Kommunikationsnachrichten von 4 auf 50 und die der Anwendungen von 8 auf 100 (jeweils einschließlich eines Threads) erhöhen.

Darüber hinaus wird die Mehrzieloptimierung, einschließlich der Ende-zu-Ende-Latenz, der Antwortzeit und der LOR, unter Verwendung eines hierarchischen Ansatzes und der RU und des maximalen Speicherverbrauchs als einzelne Grenzziele angewendet. Mit anderen Worten, wenn wir beispielsweise in Abb. 10 (e) 50 Kommunikationsnachrichten haben, besteht die Lösung aus 50 Pfaden wie einzelnen, redundanten und HR-Routen, die von Sendern zu Empfängern erstellt werden, den Zeitplänen von Tasks über Links nur für aktivierte Routen, dem Mapping von Anwendungen auf ECUs und den Thread-Zeitplänen auf jeder ECU. Wie erwartet, kann die Anzahl der generierten Routen die Lösungszeit beeinflussen. Folglich benötigt der HR-Pfad gemäß Abb. 10 (e) mehr Zeit, da er in jedem Szenario drei disjunkte Routen finden muss.

Um die Skalierbarkeit des vorgestellten Frameworks zu evaluieren, messen wir die Zeit, die für die Generierung der Systemmodellvariablen und die Lösung des Constraint-Sets für zwei verschiedene Architekturen benötigt wird, wie z.B. eine Full-Mesh-Topologie und eine zonale Topologie, die jeweils 15 Steuergeräte umfassen. Das automatische Message-Routing (zur Generierung einzelner Pfade) und Mapping sowie das Scheduling der Anwendungsthreads und Kommunikationsaufgaben werden für diese beiden Architekturen in einem einzigen Schritt gelöst. Darüber hinaus wenden wir die Mehrzieloptimierung an, die die End-to-End-Latenz, die Antwortzeit und die LOR sowie die RU und die maximale Speichernutzung als grenzwertige Einzelziele umfasst. Alle Threads haben identische Ausführungszeiten und -perioden. Auch die Rahmenlängen der Kommunikationsaufgaben sind gleich, und ihre Zeiträume entsprechen den Werten der Zeiträume ihrer Sender und Empfänger.

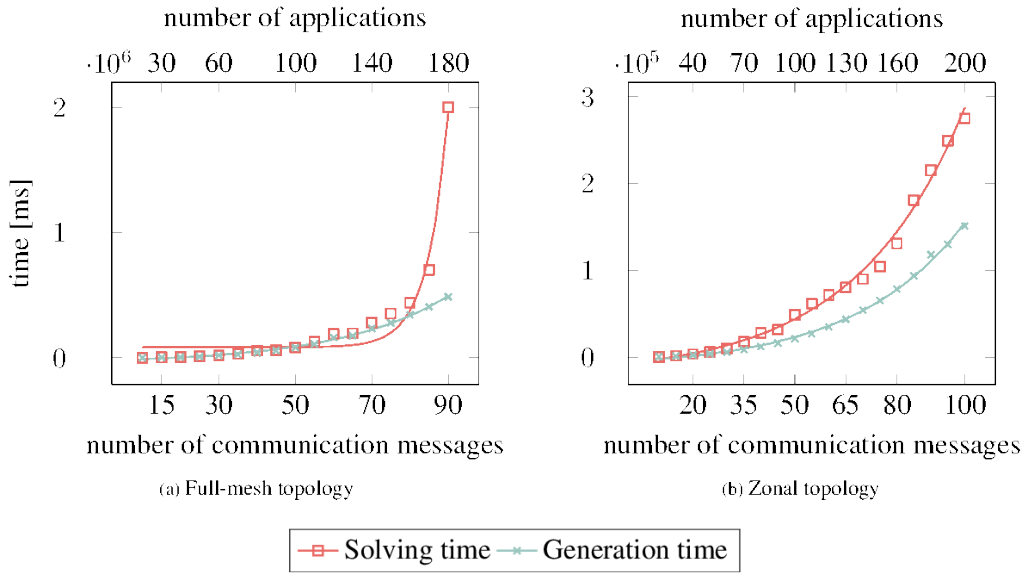

Abbildung 11: Skalierbarkeitsanalyse des vorgeschlagenen Rahmens. Generierungs- und Lösungszeiten für (a) eine Full-Mesh-Architektur und (b) eine zonale Topologie mit jeweils 15 ECUs.

Ausgehend von Abb. 11 (a) erhöhen wir für die Vollmaschentopologie die Anzahl der Nachrichten von 10 auf 90 und der Anwendungen von 20 auf 180, die jeweils einen Anwendungsthread umfassen. Bei der zonalen Topologie in Abb.11 (b) steigt die Anzahl der Nachrichten auf 100 und die der Anwendungen auf 200. Das bedeutet, dass das Tool 100 Routen von Sendern zu Empfängern erstellt, einschließlich ihrer Zeitpläne. Wie Abb.11 zeigt, wächst die Lösungszeit für beide Anwendungsfälle exponentiell. Darüber hinaus ist die Lösungszeit für die Vollmaschentopologie erwartungsgemäß deutlich höher als die für die Zonentopologie, was auf den größeren Untersuchungsraum zurückzuführen ist. Betrachtet man die zonale Topologie in Abb. 11 (b) und ihre Skalierung auf 100 Kommunikationsnachrichten und 200 Anwendungen, ist die Lösungszeit für Probleme wie Mapping, Scheduling und Routing, die bekanntermaßen NP-schwere Probleme sind [1, 3, 4, 7], mit unserem einstufigen Lösungsansatz angemessen. In der Praxis sind die dargestellten Werte für eine Anwendungsdomäne im Automobilbereich angemessen.

Die experimentellen Ergebnisse während der Entwurfsphase zeigen, dass unsere Formulierung und unser Ansatz es den Nutzern ermöglichen, ihr gewünschtes Systemmodell zu synthetisieren, das Mapping, Routing und Scheduling innerhalb eines angemessenen Zeitrahmens unterstützt und gleichzeitig die vordefinierten Anforderungen erfüllt. Darüber hinaus können durch die Anwendung von Einzel- und Mehrzieloptimierungen auch komplexere Probleme und Bedingungen gelöst werden. Es ist erwähnenswert, dass das von uns vorgestellte Tool für die Synthese jeder Art von Fahrzeug-E/E-Architektur und Netzwerktopologie geeignet ist, einschließlich verschiedener Konfigurationen von Anwendungen, Threads und Kommunikationsaufgaben.

2.2.4.2 Bewertung während der Laufzeit

In diesem Abschnitt wird beschrieben, wie die von dem vorgestellten modellbasierten Werkzeug berechneten Lösungen zur Entwurfszeit experimentell auf realer Hardware evaluiert wurden. Für die Experimente wurden drei verschiedene Hardware-Plattformen verwendet, die üblicherweise

in Automobilsystemen eingesetzt werden. Zunächst wurde ein Nvidia Drive AGX Xavier Developer Kit ausgewählt, weil es eine beliebte Wahl für eine HPCU im autonomen Fahren ist. Das Entwicklungskit besteht aus zwei Xavier-SoCs, die jeweils eine 8-Core-ARM-CPU, RAM und Hardware-Beschleuniger für Deep-Learning-Inferenz umfassen. Es läuft eine modifizierte Version von Ubuntu 18.04 mit dem Preempt-RT-Patch als Betriebssystem. Zweitens wurden Experimente auf einem Intel i210-basierten Evaluation Board durchgeführt, um ein Kommunikationsnetzwerk-Gateway zu simulieren. Drittens wurde ein Discovery-Kit mit einem STM32L476VG-Mikrocontroller als Beispiel für ein energieeffizientes Steuergerät verwendet [1, 7].

2.2.4.2.1 Software-Einrichtung

Aufgaben: Jede Aufgabe wird als ein Linux-Prozess einer benutzerdefinierten Benchmark-Anwendung modelliert. Die Benchmark-Anwendung berechnet zehn Ziffern von Pi mit Hilfe des Gauß-Legendre-Algorithmus in einer Endlosschleife und ist daher CPU-gebunden. Außerdem gibt es Sender- und Empfängeranwendungen, die den TCP-basierten Nachrichtentransfer unterstützen.

Zeitplanung und Disposition: Die Zeitplanung wird von einem anderen Standard-Linux-Prozess simuliert, dem die höchste Echtzeitpriorität 99 zugewiesen ist. Sie wird mit einer C-Timer-Funktion implementiert, die alle 1000 Nanosekunden einen Callback aufruft, um zu prüfen, ob ein Thread/eine Aufgabe zu diesem bestimmten Zeitpunkt geplant oder gestoppt werden sollte. Bevor die Simulation beginnt, muss die Anzahl der Hyperperioden angegeben werden, damit die Start- und Stoppzeiten für jede Aufgabe im Voraus berechnet und in einem sortierten Array gespeichert werden können. Daher muss im Timer-Callback nur der aktuelle Zählerwert mit dem obersten Eintrag des oben genannten Arrays verglichen werden, wodurch die Aufrufzeit des Callbacks auf ein Minimum reduziert wird. Wenn ein Task gestartet oder gestoppt werden muss, sendet der Scheduler ein POSIX-Signal an den entsprechenden Task. Die Aufgabe wird mit dem SIGKILL-Signal beendet und mit dem SIGCONT-Signal neu gestartet.

Zuordnung von Aufgaben: Die Tasks werden vor dem Start der Simulation bestimmten Kernen zugewiesen. Ihre CPU-Affinität wird mit dem Befehl *taskset* festgelegt, was den Linux-Scheduler veranlasst, den Prozess an einen bestimmten CPU-Kern zu binden. Um sicherzustellen, dass die Tasks nicht durch andere auf dem System laufende Prozesse unterbrochen werden, wird jeder Task die zweithöchste Echtzeitpriorität von 98 zugewiesen. Mit dieser Priorität werden die Tasks vom Linux-Scheduler gegenüber allen anderen Prozessen bevorzugt, können aber dennoch vom Simulations-Scheduler unterbrochen werden. Darüber hinaus werden alle anderen System- und Fremdprozesse dem CPU-Kern 0 zugeordnet, mit Ausnahme der gebundenen Kernel-Threads, um die simulierten Kerne zu isolieren.

Synchronisierung von Aufgaben: Die Taktsynchronisation zwischen mehreren Knoten wurde erreicht, indem sie durch Überbrückungsdrähte verbunden wurden und ein Triggersignal gesendet wurde, um die Simulation gleichzeitig zu starten.

Überwachung : Die Überwachung ist eine wichtige Komponente der Leistungsbewertung, da die Messungen sehr genau und leichtgewichtig sein müssen, um eine Beeinflussung der gemessenen Metriken zu vermeiden. Die wichtigste Leistungskennzahl war die Start- und

Stoppzeit der einzelnen Aufgaben [1, 5]. Daraus lässt sich der Start- und Stopp-Jitter berechnen, der sich auf die Differenz zwischen den tatsächlichen und den erwarteten Start- und Stoppzeiten bezieht. Die Start- und Stoppzeiten wurden ermittelt, indem die Systemaufrufe, die eine Änderung des Prozessstatus anzeigen, mit *strace* verfolgt wurden. Zu den weiteren Leistungsmetriken gehörten die CPU- und RAM-Auslastung sowie die thermische Entwicklung der CPU.

GUI: Um den Einsatz der berechneten Lösungen zu erleichtern, wird eine grafische Benutzeroberfläche entwickelt, die alle oben beschriebenen Funktionen in einem Programm integriert. Auf diese Weise können Mapping- und Kommunikationslösungen automatisch auf die Hardware aufgespielt werden, um die Experimente einfach zu wiederholen. Darüber hinaus startet die entwickelte GUI das Monitoring und kümmert sich um das Sammeln und Verarbeiten der Messdaten [1].

Tabelle 1: Zeiträume und Ausführungszeiten von Threads für jede Beispielanwendung.

Name	*t.p* (Period)[ms]	*t.e* (Execution time)[ms]
T_0	600	5
T_1	1200	1
T_2	600	20

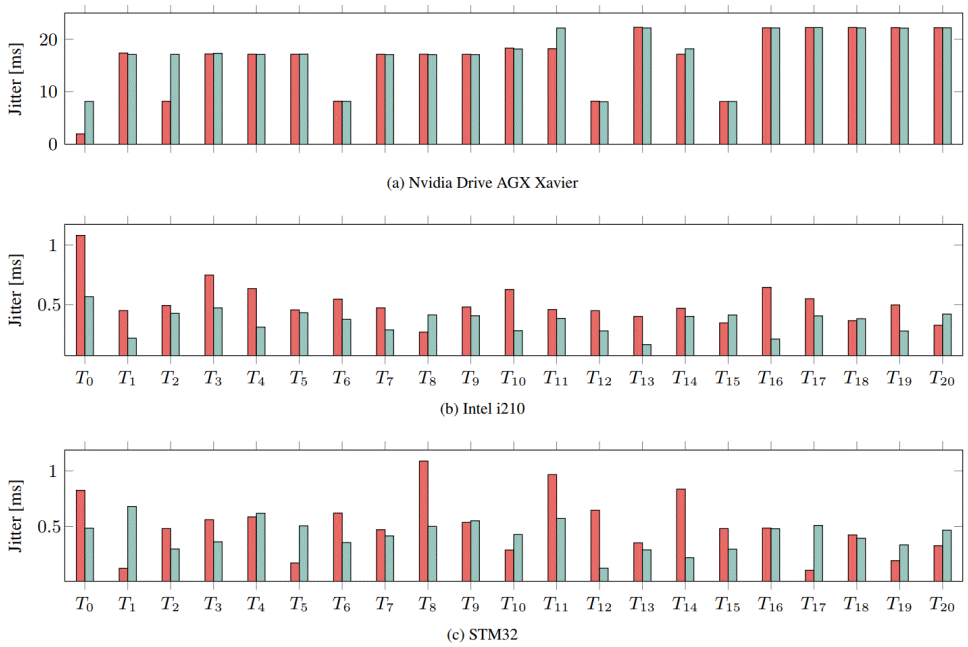

Abbildung 12: Start- und Stop-Zittern der einzelnen Threads, gemessen für die TT-Planung über eine Hyperperiode. Dies sind die Messungen auf CPU-Kern 5. Rote Balken stellen den Start-Jitter dar, grüne Balken den Stop-Jitter.

2.2.4.2.2 Mapping-Auswertung

Die Bewertung des Mappings erfolgte anhand von 40 Anwendungen, die jeweils aus drei Threads bestanden. Die Zeiträume und Ausführungszeiten dieser Threads sind im linken Abschnitt der Tabelle 1 zu finden. Insgesamt wurden 120 Anwendungsthreads auf verschiedene Kerne abgebildet. Das zeitgesteuerte Scheduling wurde im Vergleich zu FIFO (first in, first out) und

Round Robin mit einem Zeitquantum von 1 ms bewertet, das aufgrund der kürzesten Ausführungszeit ausgewählt wurde. Die FIFO-Planung ist ein einfacher und intuitiver Planungsalgorithmus, der nach dem Prinzip "Wer zuerst kommt, mahlt zuerst" arbeitet. Bei diesem Ansatz wird der Prozess, der zuerst eintrifft, zuerst ausgeführt, und die nachfolgenden Prozesse werden in der Reihenfolge ihres Eintreffens ausgeführt [11]. Das Round-Robin-Scheduling [12] hingegen ist ein präemptiver Scheduling-Algorithmus, der jedem Prozess zyklisch ein festes Zeitkontingent zuweist. Er sorgt für Fairness, indem er jedem Prozess erlaubt, eine vordefinierte Zeitspanne oder ein Quantum auszuführen, bevor er zum nächsten Prozess übergeht. Wenn ein Prozess seine Ausführung nicht innerhalb des Zeitquantums abschließt, wird er vorübergehend angehalten, und der nächste Prozess in der Warteschlange darf ausgeführt werden. Der unterbrochene Prozess wird dann wieder in die Bereitschaftswarteschlange gestellt, und die Ausführung wird im nächsten Planungszyklus an der Stelle fortgesetzt, an der sie unterbrochen wurde [12]. Da die Ausführung aller Kerne unabhängig voneinander erfolgt und die Arbeitslast in der evaluierten Lösung annähernd gleichmäßig auf alle Kerne verteilt ist,

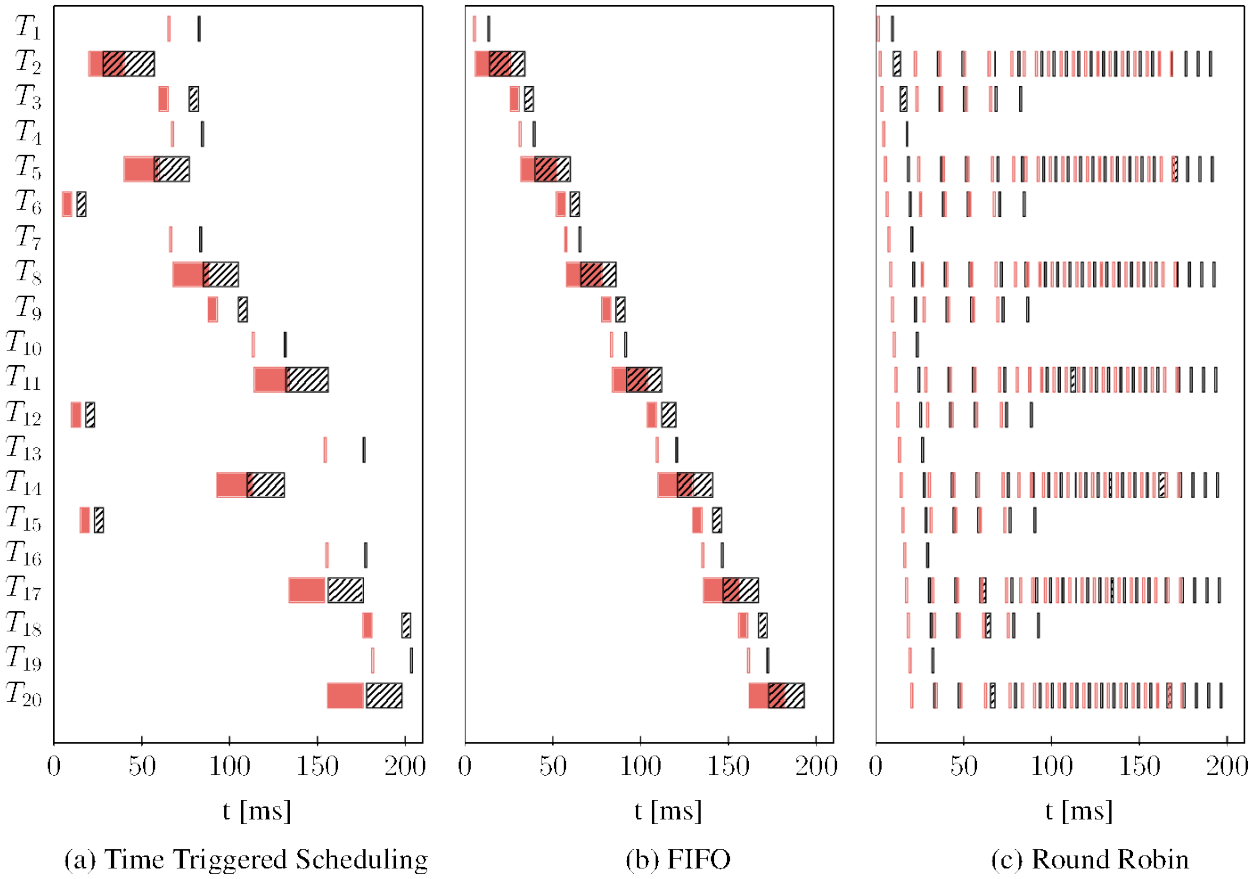

Abbildung 13: Gantt-Diagramme der verschiedenen Planungslösungen für CPU 5. Die roten Balken stellen die geplante Ausführung dar, während die schraffierten Balken für die tatsächliche Ausführung stehen.

werden zur Veranschaulichung nur die Ergebnisse auf einem Kern gezeigt, da andere Kerne ein ähnliches Verhalten zeigen.

Abb. 12 zeigt die Jitter-Messungen für zeitgesteuertes Scheduling auf allen Hardware-Plattformen. Es ist zu erkennen, dass der auf dem Nvidia-Laufwerk erzeugte Jitter im Allgemeinen signifikant ist. Der durchschnittliche Jitter ist auf dem Nvidia-Laufwerk etwa 36,6-mal höher als auf dem Intel i210 für dieselbe Konfiguration. Abb. 13 zeigt die erwarteten und tatsächlichen Start-

und Stoppzeiten für die drei getesteten Scheduling-Strategien auf dem Nvidia-Laufwerk für die ersten 200 ms. Es ist zu beobachten, dass es in allen Konfigurationen Threads gibt, die auch nach der erwarteten Stoppzeit starten. Dies gilt insbesondere für kürzere Ausführungszeiten und ist daher bei der Round-Robin-Konfiguration stärker ausgeprägt [1]. Außerdem bleibt der Abstand zwischen der tatsächlichen und der erwarteten Ausführungszeit bei der FIFO-Planung vergleichsweise konstant. Dies zeigt sich auch in Abb. 14, aus der hervorgeht, dass der Jitter bei FIFO viel besser vorhersehbar ist als bei der zeitgesteuerten Scheduling-Lösung. Dies deutet darauf hin, dass der vom Dispatcher erzeugte Jitter um einen konstanten Wert mit einer geringen Streuung zentriert ist. Nichtsdestotrotz wurden alle Threads innerhalb ihres festgelegten Zeitraums beendet, und in keinem der getesteten Setups kam es zu einer Verletzung der Frist.

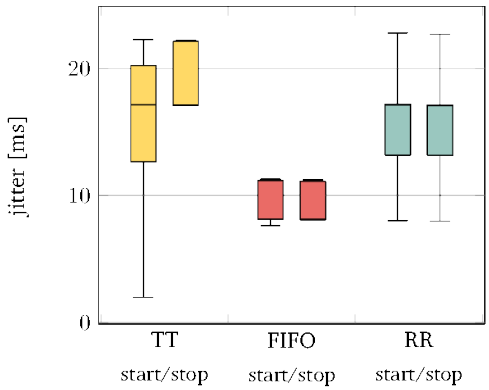

Abbildung 14: Streuung des Start- und Stopp-Jitters für die getesteten Zeitplanungsstrategien. Das Boxplot zeigt die minimalen und maximalen Werte, die unteren und oberen Quantile und den Mittelwert. Die gelben, roten und grünen Kästen stellen die Ergebnisse für zeitgesteuerte, FIFO- bzw. Round-Robin-Planungsschemata dar.

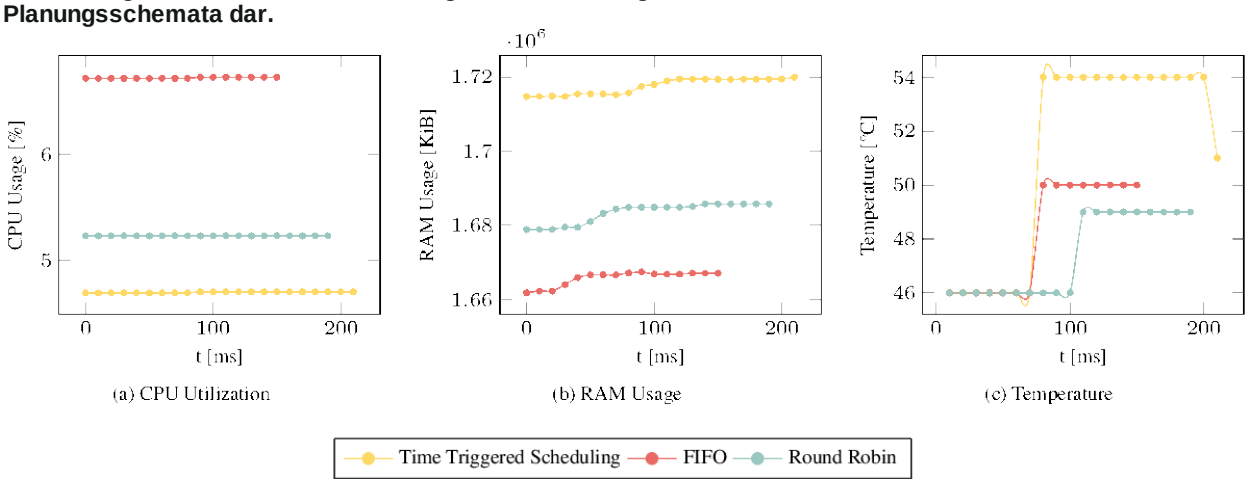

Abbildung 15: Vergleich der verschiedenen Metriken. Gelb ist die time-triggered Planung, rot ist die FIFO-Planung und grün ist die Round Robin-Planung.

Die Ergebnisse der CPU-, RAM- und sind in Abb. 15 dargestellt. Es gibt merkliche, aber nicht signifikante Unterschiede zwischen allen Konfigurationen. Außerdem übertrifft keines der Planungsschemata die anderen in allen Metriken. Die Kommunikation wurde anhand eines

Tabelle 2:Von E/E Designer berechnete Kommunikationslösung.

Name	Start time [us]	Stop time [us]
t_1.st	0.00	90.00

t_2.st	90.00	290.00
t_3.st	290.00	440.00
c_1.stl1	100.0	112.60
c_2.stl1	300.00	312.60
c_3.stl1	450.00	462.00
t_4.st	150.20	250.20
t_5.st	650.20	850.20
t_6.st	500.20	650.20
c_1.stl2	127.60	140.20
c_2.stl2	327.60	340.20
c_3.stl2	477.60	490.20

Szenarios getestet, das aus drei verbundenen Knoten bestand: einem Nvidia TX2 Developer Kit (ECU_1), dem Nvidia Drive AGX Xavier (ECU_2) und einem Intel i210 Developer Kit (ECU_3), basierend auf Abb. 16. Auf ECU_1 liefen drei Anwendungen, jeweils bestehend aus einem Thread, die jeweils ein Kommunikationspaket über ECU_2 an eine entsprechende Empfänger-Task auf ECU_3 senden. Abb. 16 visualisiert den Aufbau. Wie bereits erwähnt, werden die Ende-zu-Ende-Latenzen und Antwortzeiten der drei Kommunikationsnachrichten, die jeweils eine oben definierte Kommunikationsaufgabe enthalten, gemessen [1].

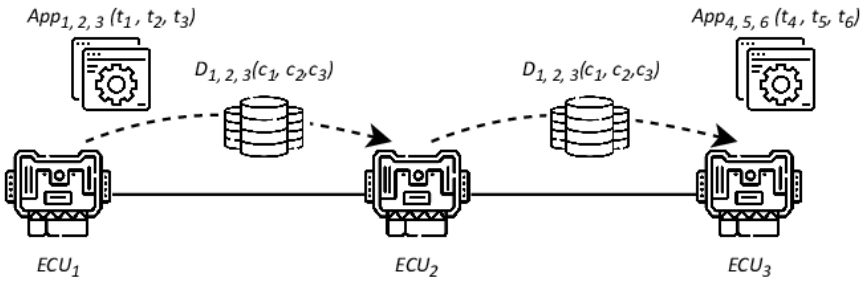

Abbildung 16: Topologie des getesteten Kommunikationsaufbaus. Die Threads 1-3 auf ECU1 senden jeweils ein Kommunikationspaket an die Threads 4-6 auf ECU3. Alle drei Pakete c1, c2, c3 werden über ECU2 geroutet.

2.2.4.2.3 Koomunikationsbewertung

Die Kommunikationsverbindungen hatten eine theoretische Bandbreite von 1000 Mbit/s; es wurden jedoch 940 Mbit/s gemessen, als auch die PTP-Synchronisation über dieselben Verbindungen lief. Die Größe der Kommunikationsnachrichten wurde so gewählt, dass sie in einen einzigen Ethernet-Rahmen von 1500~B Größe passen. Daher wurde die Rahmenlänge jeder Kommunikationsaufgabe auf 12.6 µs festgelegt. Die von dem vorgeschlagenen Tool berechnete Lösung ist in Tabelle 2 dargestellt. Sie zeigt die Start- und Stoppzeiten für die Sender- und Empfängeranwendungen und die Kommunikationsaufgaben über eine Hyperperiode hinweg [1]. Abb. 17 zeigt die Ergebnisse der Experimente. Es fällt auf, dass die gemessenen Ende-zu-Ende-Latenzen nicht stark variieren, sondern in der Größenordnung von 10 ms liegen. Auch wenn die tatsächliche Frame-Länge wesentlich geringer ist als diese Werte, hat der Einfluss von Start- und Stop-Jitters in den Anwendungsthreads einen großen Einfluss auf die

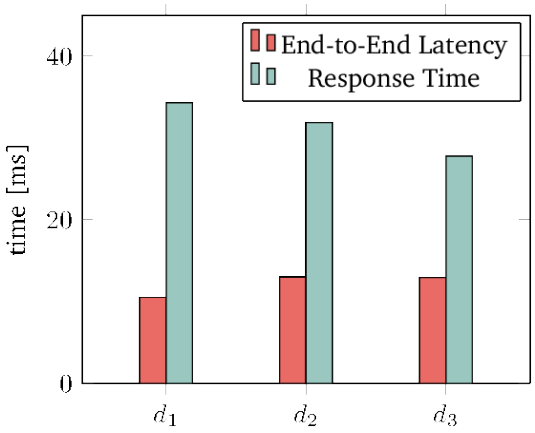

Abbildung 17: Gemessene Ende-zu-Ende-Latenz und Antwortzeiten der drei Kommunikationsnachrichten unter Verwendung von Nvidia Drive AGX, Nvidia TX2 und Intel i210 Entwickler-Kits als Setup.

Gesamtlatenzmetrik. Dies wiederum führte während der Testphase der Kommunikation zu Terminüberschreitungenn [1]. Dieses Phänomen kann auf den erheblichen Jitter zurückgeführt werden, der durch das Nvidia AGX-Laufwerk verursacht wird und in Abb. 12 anschaulich dargestellt ist. Aufgrund dieses Effekts wurde beschlossen, das Kommunikationsexperiment mit STM32-basierten Entwicklungsplatinen durchzuführen [13]. Diese Boards weisen einen deutlich geringeren Scheduling-Jitter auf und gewährleisten eine minimale Unterbrechung der Ausführung der synthetisierten Kommunikationslösung. Es ist wichtig anzumerken, dass bei diesem speziellen Experiment dieselbe Topologie und dieselben Ergebnisse zur Entwurfszeit (siehe Tabelle 2) verwendet wurden wie beim ursprünglichen Kommunikationsexperiment, die beide vom Tool generiert wurden.

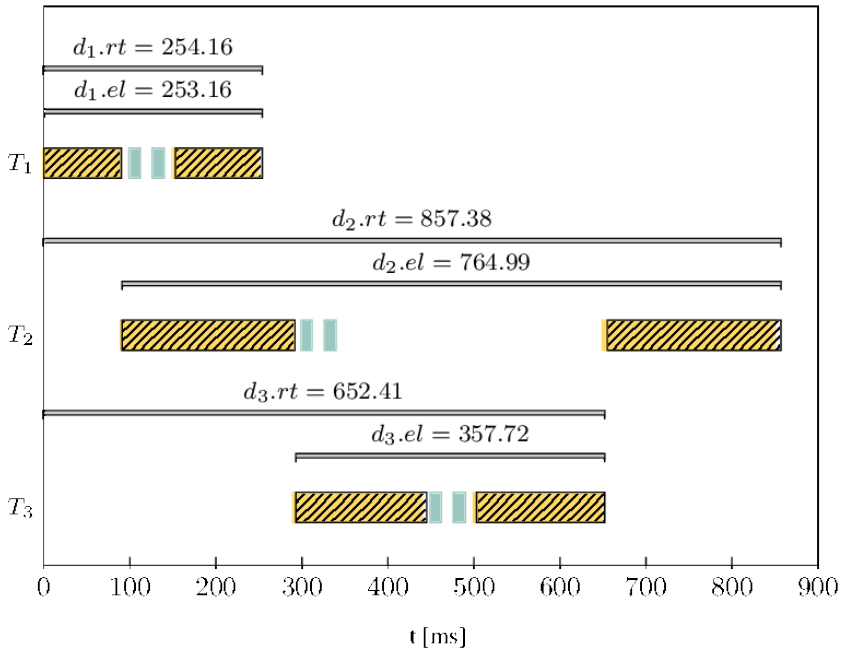

Abbildung 18: Ergebnisse des Experiments zur Bewertung der Kommunikation mit STM-Entwicklungskits. Die gelben Balken stehen für die erwartete Ausführung, während die schraffierten Balken die tatsächlichen Ausführungszeiten der Sender- und Empfängeranwendungen darstellen. Die grünen Balken zeigen den geplanten Zeitplan der Kommunikationsnachrichten an. Die Antwortzeit und die Ende-zu-Ende-Latenzzeit jeder Kommunikationsnachricht sind ebenfalls dargestellt [1].

Die CAN-Konfiguration wurde als Kommunikationsprotokoll zwischen den Steuergeräten in diesem Aufbau gewählt. Diese Wahl resultiert aus seiner weiten Verbreitung im Automobilbereich und seiner vergleichsweise einfachen Handhabung im Vergleich zu anderen Protokollen hinsichtlich der notwendigen Hardware-Einrichtung und Software-Implementierung. Außerdem hat es einen relativ geringen Overhead und vereinfacht somit die Berechnungen. Die Kommunikationsverbindungen hatten eine theoretische Bandbreite von 11.520 Bytes/s, was einer üblichen Baudrate entspricht. Die Größe der Kommunikationsnachrichten wurde mit 145 Byte bewusst klein gehalten, damit sie in einen einzigen Frame passen und eine Fragmentierung und lange Überprüfungszeiten vermieden werden, die die Ergebnisse möglicherweise beeinflussen können. Die Framelänge jeder Kommunikationsaufgabe wurde auf 12,6 ms festgelegt, berechnet durch Division der Größe der Kommunikationsnachricht durch die Bandbreite. Die Lücke zwischen den Paketen wurde auf 10 ms festgelegt. Der Benutzer kann diese Kommunikationseigenschaften für jedes Netzwerkprotokoll im Framework festlegen, ähnlich wie die Zeitparameter im Frontend des Frameworks bestimmt werden können.

Abb. 18 zeigt die Ergebnisse der Kommunikationsexperimente mit den STM-Boards. Es ist zu erkennen, dass die gemessenen Antwortzeiten und Ende-zu-Ende-Latenzen für alle drei Kommunikationsnachrichten d_1, d_2 und d_3 eng mit den erwarteten Werten übereinstimmen, was darauf hindeutet, dass es keine Fristverletzungen gibt.

3 Wichtigste Positionen des zahlenmäßigen Nachweises

Personalkosten –Wissenschaftliche Mitarbeiter, Dienstreisen kosten über Konferenzen und Forschungsaufenthalte.

4 Notwendigkeit und Angemessenheit der geleisteten Arbeiten

Zuvor wurden die Ziele des KI-FLEX-Projekts beschrieben, einschließlich der Abgrenzung der Arbeitspakete, die von der TUM durchgeführt werden sollen. Der von uns vorgeschlagene Rahmen ist mit den übergeordneten Projektzielen abgestimmt, wie eine umfassende State-of-the-Art-Analyse gezeigt hat, die bestehende Lücken und Herausforderungen identifiziert hat, die mit den Projektzielen übereinstimmen. Folglich haben wir versucht, diese Lücken und Herausforderungen durch die Einführung der oben erläuterten Ansätze zu überbrücken. Hinsichtlich der Angemessenheit unserer Arbeit, insbesondere aus technischer Sicht, wurden unsere Ergebnisse sowohl mit den Projektpartnern als auch mit dem VDI/VDE ausgetauscht und diskutiert. Unsere gemeinsamen Bemühungen und Ergebnisse wurden effektiv kommuniziert, was für Transparenz sorgte und ein kohärentes Verständnis unter den Beteiligten förderte. Wichtig ist, dass unsere Ergebnisse im Rahmen der spezifizierten Arbeitspakete für die TUM nahtlos alle Anforderungen erfüllten, die ursprünglich im Projektantrag definiert waren. Aus der Forschungsperspektive haben wir mit Stolz zum akademischen Diskurs beigetragen, indem wir während der Projektlaufzeit sieben begutachtete Publikationen veröffentlicht haben. Diese umfangreiche Arbeit ist ein Beleg für unser Engagement als akademischer Partner, der die Forschungsziele des Projekts untermauert und die Rolle der TUM als wichtiger Beitrag für die breitere akademische Gemeinschaft weiter etabliert. Darüber hinaus hat unser proaktives

Engagement bei der Verbreitung von Wissen und Erkenntnissen nicht nur den Gemeinschaftsgeist innerhalb des Projekts gestärkt, sondern auch unsere Reichweite über die Projektgrenzen hinaus erweitert. Durch die Weitergabe unserer Ergebnisse in Form von Veröffentlichungen, Präsentationen und Interaktionen haben wir zur weiteren Verbreitung von Wissen beigetragen und so die Wirkung und Sichtbarkeit des KI-FLEX-Projekts in der akademischen und beruflichen Welt erhöht.

5 Nutzen und Verwertbarkeit des Ergebnisses

Um die Verbreitung unserer Forschungsergebnisse zu erleichtern und die Zusammenarbeit zu fördern, wurde das von uns entwickelte Software-Framework als Open Source zur Verfügung gestellt und ist über den folgenden Link zugänglich: "https://github.com/hadiaskaripoor/ee-designer". Diese Entscheidung richtet sich sowohl an die Forschung als auch an die Industrie und ermöglicht es ihnen, das Framework nicht nur zu erforschen und zu nutzen, sondern sich auch aktiv an seiner Weiterentwicklung zu beteiligen.

Das Open-Sourcing unseres Frameworks dient einem vielschichtigen Zweck. Erstens entspricht sie unserem Engagement für die Förderung der Zusammenarbeit und bietet Forschern eine gemeinsame Plattform, auf der sie aufbauen und die Arbeit erweitern können. Dies verbessert die Reproduzierbarkeit unserer Ergebnisse und ermöglicht es anderen, unsere Erkenntnisse unabhängig zu validieren und zu verifizieren. Darüber hinaus können Partner aus der Industrie und der Wissenschaft erheblich von unserer Open-Source-Initiative profitieren. Das Framework dient als wertvolle Ressource für ihre zukünftigen Projekte und bietet eine solide Grundlage, die auf spezifische Anforderungen zugeschnitten werden kann. Diese Anpassungsfähigkeit stellt sicher, dass unsere Ergebnisse nicht auf eine einzelne Anwendung beschränkt sind, sondern sich weiterentwickeln und in verschiedenen Kontexten Relevanz finden können. Darüber hinaus sind wir uns des Potenzials bewusst, dass unsere Ergebnisse auf der Grundlage der einzigartigen Anforderungen verschiedener Interessengruppen verfeinert und erweitert werden können. Indem wir die Tür für Zusammenarbeit und Beiträge öffnen, befähigen wir die breitere Gemeinschaft, die Funktionalität, Skalierbarkeit und Anwendbarkeit des von uns entwickelten Tools im KI-FLEX-Projekt zu verbessern. Diese kollektive Anstrengung stärkt nicht nur die Projektergebnisse selbst, sondern trägt auch zum Wachstum des gemeinsamen Wissens auf diesem Gebiet bei.

6 Fortschritt bei anderen Stellen

Nicht zutreffend.

7 Veröffentlichungen

Tabelle 3: Erfolgte und Erfolgte und geplante Veröffentlichungen von Projektergebnissen

Autor(en) / Partner	Titel der Publikation	Publikation auf	Datum Veröffent-lichung	Link
H. Askaripoor, T. Mueller and A. Knoll	E/E Designer: a Framework to Design and Synthesize Vehicle E/E Architecture	Journal of IEEE Transactions of Intelligent Vehicles	13.05.2023	https://ieeexplore.ieee.org/document/10285045
Askaripoor, H., Hashemi Farzaneh, M. and Knoll	E/e architecture synthesis: Challenges and technologies	MDPI Electronics Journal	10.02.2022	https://www.mdpi.com/2079-9292/11/4/518
Askaripoor, H., Hashemi Farzaneh, M. and Knoll	A model-based approach to facilitate design of homogeneous redundant e/e architectures	ITSC 2021 Conference	25.10.2021	https://ieeexplore.ieee.org/document/9565115
Askaripoor, H., Hashemi Farzaneh, M. and Knoll	A platform to configure and monitor safety-critical applications for automotive central computers	ETFA 2021 Conference	30.11.2021	https://ieeexplore.ieee.org/document/9613692
Askaripoor, H., Hashemi Farzaneh, M. and Knoll	Considering safety requirements in design phase of future e/e architectures	ETFA 2020 Conference	05.10.2020	https://ieeexplore.ieee.org/document/9212001
Hadi Askaripoor, Sina Shafaei, Alois Knoll	A flexible scheduling architecture of resource distribution proposal for autonomous driving platforms	VEHITS 2021 Conference	April 2021	https://mediatum.ub.tum.de/doc/1613149/document.pdf
Müller, T., Askaripoor, H. and Knoll, A	Performance analysis of KVM hypervisor using a self-driving developer kit	IECON 2022 Conference	09.12.2022	https://ieeexplore.ieee.org/abstract/document/9968908

Bibliografie

[1] H. Askaripoor, T. Mueller and A. Knoll, "E/E Designer: a Framework to Design and Synthesize Vehicle E/E Architecture," in IEEE Transactions on Intelligent Vehicles, doi: 10.1109/TIV.2023.3324617.

[2] Askaripoor, H., Hashemi Farzaneh, M. and Knoll, A., 2022. E/E architecture synthesis: Challenges and technologies. Electronics, 11(4), p.518.

[3] Askaripoor, H., Farzaneh, M.H. and Knoll, A., 2020, September. Considering safety requirements in design phase of future e/e architectures. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 1165-1168). IEEE.

[4] H. Askaripoor, M. H. Farzaneh and A. Knoll, "A Model-Based Approach to Facilitate Design of Homogeneous Redundant E/E Architectures," 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 2021, pp. 3426-3431, doi: 10.1109/ITSC48978.2021.9565115.

[5] Askaripoor, H., Shafaei, S. and Knoll, A., 2021. A flexible scheduling architecture of resource distribution proposal for autonomous driving platforms. In Proceedings of the 7th International Conference on Vehicle Technology and Intelligent Transport Systems.

[6] T. Müller, H. Askaripoor and A. Knoll, "Performance Analysis of KVM Hypervisor Using a Self-Driving Developer Kit," IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society, Brussels, Belgium, 2022, pp. 1-7, doi: 10.1109/IECON49645.2022.9968908.

[7] H. Askaripoor, M. H. Farzaneh and A. Knoll, "A Platform to Configure and Monitor Safety-Critical Applications for Automotive Central Computers," 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, 2021, pp. 1-4, doi: 10.1109/ETFA45728.2021.9613692.

[8] Gleeson, J. and Ryan, J., 1990. Identifying minimally infeasible subsystems of inequalities. ORSA Journal on Computing, 2(1), pp.61-63.

[9] Liffiton, M.H., Previti, A., Malik, A. and Marques-Silva, J., 2016. Fast, flexible MUS enumeration. Constraints, 21, pp.223-250.

[10] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2022.

[11] Leontyev, H. and Anderson, J.H., 2007, July. Tardiness bounds for FIFO scheduling on multiprocessors. In 19th Euromicro Conference on Real-Time Systems (ECRTS'07) (pp. 71-71). IEEE.

[12] Rasmussen, R.V. and Trick, M.A., 2008. Round robin scheduling–a survey. European Journal of Operational Research, 188(3), pp.617-636.

[13] ST. STM32L476VG. 2023. URL:https://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html.