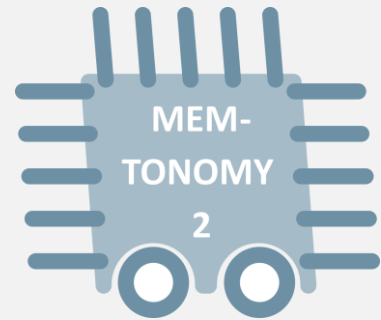


MEMTONOMY-2

Verbundprojekt
Speichersysteme für das autonome Fahren



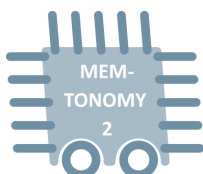
Sachbericht Teil II: Eingehende Darstellung

Zuwendungsempfänger

Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
Lehrstuhl Entwurf Mikroelektronischer Systeme
Erwin-Schrödinger-Straße, Gebäude 12, 67663 Kaiserslautern

Datum	08.04.2026
Projektkronym	MEMTONOMY-2
Förderkennzeichen	16ME0717
Projektlaufzeit	01.11.2022 - 31.12.2025

Projektpartner	RPTU Kaiserslautern-Landau (RPTU) Fraunhofer IESE (IESE) Technische Universität München (TUM) Hyperstone GmbH (HS) Micron Semiconductor (Deutschland) GmbH (MICRON) LUBIS EDA GmbH (LUBIS)
Assoziierte Partner	Aumovio SE, Mercedes-Benz AG
Projektkoordinator	Fraunhofer IESE, Kaiserslautern



Gefördert durch:



PROJEKTRÄGER

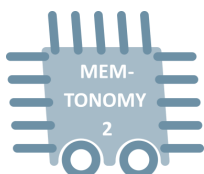
VDI | VDE | IT

Zweck dieses Dokuments

Dieses Dokument beschreibt als Teil II des Sachberichts der RPTU Kaiserslautern-Landau die Projektergebnisse des Verbundprojektes »**MEMTONOMY-2 – Speichersysteme für das autonome Fahren**« in ausführlicher Form. Dazu erläutert das Dokument die durchgeführten Arbeiten, gibt einen Überblick zur finanztechnischen und inhaltlichen Projektumsetzung, macht Aussagen zur Notwendigkeit, Angemessenheit und zum Nutzen der Projektarbeiten, zeigt Fortschritte auf dem Gebiet des Vorhabens bei anderen Stellen auf und gibt einen Überblick zu Veröffentlichungen im Rahmen des Projekts.

Das Projekt wurde unter dem Förderkennzeichen 16ME0716-16ME0721, sowie 16ME0995 vom Bundesministerium für Forschung, Technologie und Raumfahrt gefördert und durch den Projektträger PT-VDI/VDE, Arbeitseinheit TZ-MST betreut. Es wurde vom 01.11.2022 – 31.12.2025 im Rahmen des Programms „Elektronik und Softwareentwicklungsmethoden für die Digitalisierung der Automobilität (MANNHEIM)“ durchgeführt.

Ziel des Vorhabens war es, den Einsatz von DRAM- und Flash-Speichern in sicherheitskritischen Anwendungskontexten in der Automobilindustrie unter Berücksichtigung entsprechender Sicherheitsnormen (z. B. ISO 26262) sowie der Performanzanforderungen zu ermöglichen.



Gefördert durch:



Bundesministerium
für Forschung, Technologie
und Raumfahrt

PROJEKTTRÄGER

VDI | VDE | IT

Inhaltsverzeichnis

Liste der Bilder	1
1 Durchgeführte Arbeiten	2
1.1 Kurzdarstellung.....	2
1.2 AP1 (Anforderungen und Architektur)	2
1.3 AP2 (Safety und Zuverlässigkeit)	3
1.4 AP3 (Optimierungen Speichercontroller)	8
1.5 AP4 (Demonstrator).....	13
2 Projektumsetzung	14
2.1 Finanztechnische Projektumsetzung	14
2.2 Inhaltliche Projektumsetzung	14
3 Notwendigkeit und Angemessenheit der geleisteten Projektarbeiten	14
4 Nutzen der geleisteten Projektarbeiten	14
5 Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen	15
6 Veröffentlichungen	16
7 Literaturverzeichnis	17

Liste der Bilder

Abbildung 1: Retention-Fehler abhängig von Temperatur und Hersteller	3
Abbildung 2: Messplattform für Interfacefehler	4
Abbildung 3: Anschluss der aktiven Tastköpfe	5
Abbildung 4: Verteilung der TLFM	6
Abbildung 5: Erreichbare Bandbreite für ein sequenzielles und zufälliges Zugriffsmuster ohne und mit aktiviertem Inline-ECC.....	7
Abbildung 6: Latenz über verschiedene Bandbreiten für ein sequenzielles Zugriffsmuster.	8
Abbildung 7: Latenz über verschiedene Bandbreiten für ein zufälliges Zugriffsmuster.	8
Abbildung 8: Optimierungen im Speichercontroller - Entwurfsraum.....	8
Abbildung 9: Interner Aufbau eines LPDDR5-DRAMs.....	10
Abbildung 10: Multiplexer-Architektur zur Adressabbildung.....	11
Abbildung 11: Adressabbildung mit Matrix	11

1 Durchgeführte Arbeiten

1.1 Kurzdarstellung

Im Bereich des autonomen Fahrens müssen durch den stetig wachsenden Einsatz von Sensorik und KI-Komponenten für kritische Fahrfunktionen zukünftig sehr große Datenmengen in der Edge echtzeitnah aufgenommen, fusioniert und analysiert werden.

Die Menge an Daten, die in Echtzeit verarbeitet werden müssen, sowie der Anteil an Software nimmt in modernen Fahrzeugen stetig zu. Dabei gibt es an die Hardware hohe Anforderungen an einen niedrigen Leistungsverbrauch und hohe Performanz sowie einen großen Kostendruck, der dazu führt, dass verstärkt Komponenten eingesetzt werden, die ursprünglich für den Consumer-Markt entwickelt wurden. Bei neuen Automobilanwendungen wird die Lücke zwischen der Rechengeschwindigkeit der Prozessoren und dem Zugriff auf den externen Arbeitsspeicher immer stärker sichtbar.

Wie Speichermodule in sicherheitskritischen Anwendungskontexten in Bezug auf Bandbreite, Latenz, Leistung, Temperatur, Zuverlässigkeit und Sicherheit eingesetzt werden können, stellt für die Automobilindustrie eine zentrale Herausforderung dar. Die wissenschaftliche Speicherforschung konzentriert sich allerdings bisher vor allem auf den Einsatz in mobilen Geräten und Rechenzentren. Die fehlenden Forschungsergebnisse bzgl. DRAM-Speicher werden für die Plattformentwicklung zunehmend zu einem Problem, da der Einsatz neuer Speichertechnologien aufgrund der Anforderungen von neuronalen Netzen und anderen KI-basierten Anwendungen insbesondere in sicherheitskritischen Anwendungen wie dem autonomen Fahren immer zwingender wird.

Ziel des Vorhabens war es deshalb, den Einsatz von DRAM- und Flash-Speichern in der Automobilindustrie unter Berücksichtigung der entsprechenden Sicherheitsnormen (z. B. ISO 26262) sowie der Performanzanforderungen zu ermöglichen.

Die RPTU hat sich wegen der verfügbaren Kompetenzen auf mehrere Schwerpunkte konzentriert. Zum einen wurden von RPTU, IESE und Micron hauptsächlich DRAM betrachtet, während Hyperstone und TUM ihren Fokus auf Flash gelegt haben. Durch die langjährige Erfahrung im Bereich der Kanalcodierung war die Fehlercharakterisierung und Entwicklung von Fehlererkennung und -korrektur zudem ein Schwerpunkt. Auch die Optimierung des DRAM-Controllers ist ein langjähriges Forschungsgebiet der RPTU und wurde in diesem Projekt weiter vorangetrieben.

1.2 AP1 (Anforderungen und Architektur)

Die RPTU hat in AP1 das Fraunhofer IESE mit seiner Expertise im Bereich DRAM-Speichersysteme unterstützt. Im Anschluss an die vom Fraunhofer IESE durchgeführten Interview-Termine wurde ein grundlegender Anforderungskatalog für den Einsatz von DRAM-Speichersystemen in autonomen Fahrzeugen ausgearbeitet. Zu den Anforderungen zählt auch die Festlegung bestimmter DRAM-Standards, die für den Einsatz im Fahrzeug infrage kommen. Hierbei haben sich vor allem die Low-Power-Standards (LPDDR_x) und High-Performance-Standards (HBM_x) als favorisierte Kandidaten herausgestellt. Für diese Standards wurden neue Simulationsmodelle entwickelt und bereits bestehende Modelle um

zusätzliche Funktionalitäten ergänzt. Zudem wurden erste Simulationen der Zielarchitekturen durchgeführt.

1.3 AP2 (Safety und Zuverlässigkeit)

Die RPTU hat sich in AP2 hauptsächlich auf DRAM-Speicher konzentriert. Zur Fehlercharakterisierung wurden zunächst Retention-Fehlermessungen für LPDDR4-Devices verschiedener Hersteller bei drei unterschiedlichen Betriebstemperaturen durchgeführt (30 °C, 60 °C und 80 °C) [7]. Die Temperaturabhängigkeit spielt im automobilen Anwendungsfall eine große Rolle, da die Steuergeräte und damit Speicher abhängig von ihrer Positionierung im Fahrzeug großen Temperaturschwankungen ausgesetzt sind. Die Messungen haben gezeigt, dass die Unterschiede zwischen den einzelnen Herstellern gering ausfallen, eine erhöhte Betriebstemperatur jedoch zu deutlich höheren Fehlerraten führt (siehe Abbildung 1: Retention-Fehler abhängig von Temperatur und Hersteller). Durch Extrapolation der Kurven wurde zudem ersichtlich, dass auch bei 80 °C Betriebstemperatur noch mehr als eine Größenordnung zwischen dem vom Standard vorgeschriebenen Refresh-Intervall von 32 ms und der tatsächlichen Worst-Case-Retention-Zeit liegt (der Punkt, an dem die Kurven die x-Achse schneiden würden). Alle getesteten Module entsprachen somit den Vorgaben des Standards und bei Einhaltung des vorgeschriebenen Refresh-Intervalls sind keine Retention-Fehler zu erwarten.

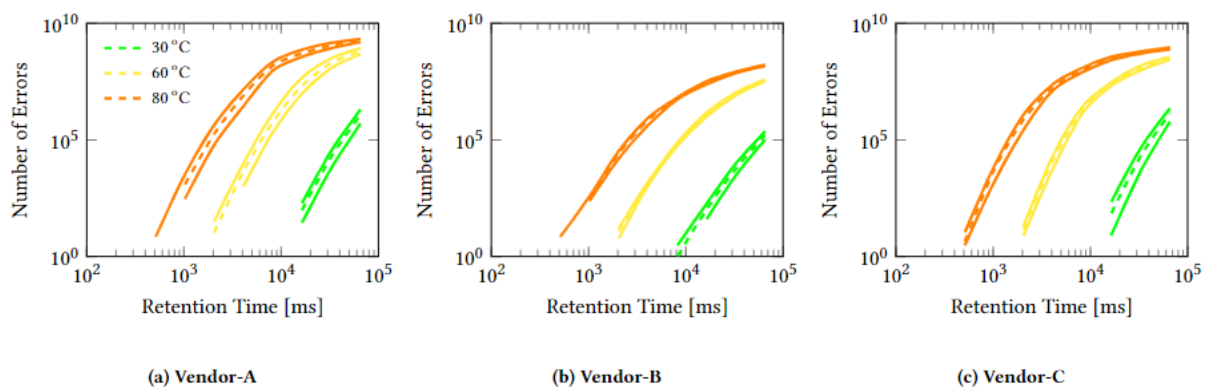


Abbildung 1: Retention-Fehler abhängig von Temperatur und Hersteller

Allerdings erlaubt der Standard auch Betriebstemperaturen über 85 °C, schreibt dann aber eine Verkürzung des Refresh-Intervalls vor. In vergangenen Arbeiten hat sich gezeigt, dass diese Verkürzung deutliche negative Auswirkungen auf die Bandbreite und Latenz der Speicher hat. Die damaligen Simulationen wurden für ältere DRAM-Standards (DDR3, DDR4) durchgeführt, deren Refresh nur auf Device-Granularität durchgeführt werden konnte. Dies bedeutet, dass alle Bänke innerhalb des Speichers gleichzeitig blockiert werden und währenddessen keine Zugriffe erfolgen können. Neuere Standards, so auch LPDDR4 und dessen Nachfolger, verfügen über zusätzliche optimierte Refresh-Kommandos, die eine geringere Granularität besitzen. So werden nur einzelne Bänke innerhalb des Devices durch den Refresh blockiert, während die anderen Bänke weiterhin Lese- und Schreibzugriffe verarbeiten können. Die Simulationen wurden daher für LPDDR4 wiederholt. Es hat sich gezeigt, dass die Verkürzung des Refresh-Intervalls nunmehr kaum messbare Auswirkungen

auf Bandbreite und Latenz hat, sofern die Refresh-Kommandos mit geringerer Granularität genutzt werden und der Speichercontroller in der Lage ist, die eingehenden Zugriffe geschickt umzusortieren. Somit ist der Betrieb von DRAM-Bausteinen auch bei Betriebstemperaturen über 85 °C im automobilen Einsatz problemlos möglich.

Die Fehlercharakterisierung wurde anschließend für Interface-Fehler fortgeführt. Deren Hauptursache ist Jitter, welcher dazu führt, dass die Datenleitungen zum falschen Zeitpunkt abgetastet und noch der vorherige oder schon der nächste Wert erfasst werden. Da reale Interface-Fehlerraten sehr gering sind (Bitfehlerraten von 10^{-12} bis 10^{-15}), ist eine direkte Messung nicht in annehmbarer Zeit durchführbar. Das sogenannte Dual-Dirac-Verfahren ermöglicht es jedoch, die Interface-Fehlerraten aus einem Augendiagramm zu approximieren. Hierzu werden die Ausläufer der Wahrscheinlichkeitsdichtefunktion der Jitterverteilung eines Augendiagrammes gefittet und extrapoliert. Um die notwendigen Augendiagramme aufzuzeichnen, wurde die in Abbildung 2: Messplattform für Interfacefehler gezeigte Messplattform aufgebaut, die die im Projekt angeschafften aktiven Tastköpfe verwendet (siehe Abbildung 3: Anschluss der aktiven Tastköpfe).

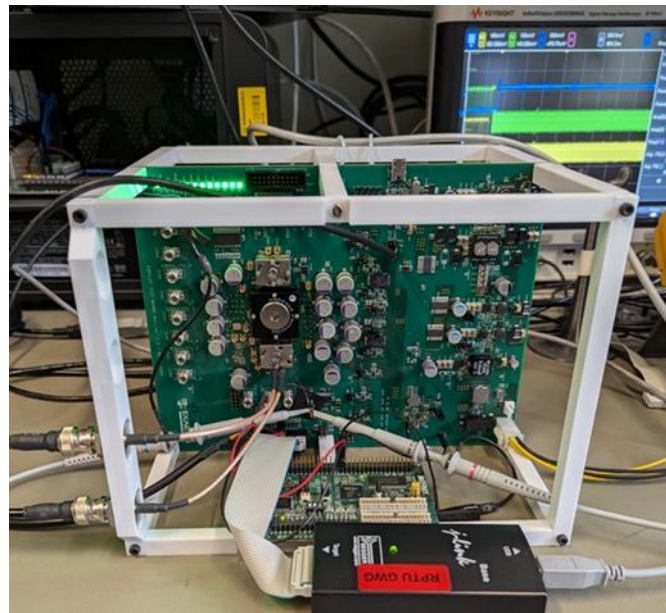


Abbildung 2: Messplattform für Interfacefehler

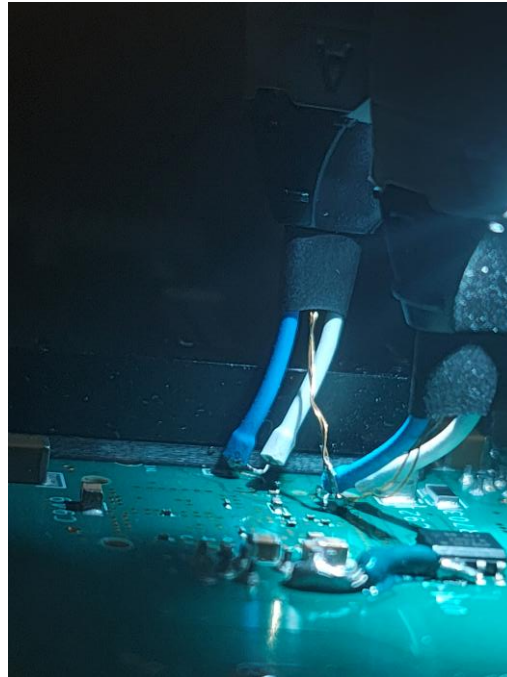


Abbildung 3: Anschluss der aktiven Tastköpfe

Während vom Fraunhofer IESE auf Basis von Augendiagrammmessungen die Approximation der Interface-Fehlerraten mit dem Dual-Dirac-Verfahren durchgeführt wurde, hat die RPTU zudem die unterschiedlichen Jitterquellen untersucht. Der Gesamtjitter kann als Überlagerung einzelner lokaler Jitter (pro Datenleitung) und eines globalen Jitters (mindestens acht Datenleitungen wegen des gemeinsamen Taktsignals) angesehen werden. Die Verteilung ist für die Entwicklung von Fehlererkennung und -korrektur elementar, da lokale Jitter stochastisch unabhängig sind und daher nur Einzelbitfehler verursachen, während globaler Jitter Multibitfehler verursacht. Die Bestimmung der Jitterverteilung kann allerdings nicht auf Basis von Augendiagrammen erfolgen, da diese nicht den Jitter mehrerer Datenleitungen zu einem bestimmten Zeitpunkt zeigen, sondern die Überlagerung vieler Übertragungen über eine einzelne Datenleitung. Daher war es für die Bestimmung der Jitterverteilung notwendig, tatsächlich auftretende Interface-Fehler zu zählen und die Fehlerbilder zu analysieren. Um diese Messungen in „endlicher“ Zeit durchführen zu können, wurde in einem speziellen Messaufbau der Jitter durch externe Störquellen künstlich erhöht. Es hat sich gezeigt, dass in der Realität beide Fehlerbilder, also Einzelbitfehler und Multibitfehler, auftreten und keines davon dominiert. Somit ist ein SECDED-ECC, wie ihn LPDDR5 implementiert, für den Datenbus nicht ausreichend, um die Fehler zu erkennen. LPDDR6 hingegen implementiert eine deutlich umfangreichere Fehlererkennung für das Interface, die entweder 100 % der Einzelbitfehler korrigiert und mehr als 99,5 % der Multibitfehler erkennt, oder keine Fehler korrigiert, aber dafür mehr als 99,998 % der Multibitfehler erkennt. Zudem besitzt LPDDR6 ein dediziertes Signal, um dem Speichercontroller das Auftreten eines Fehlers mitzuteilen. Dadurch wird die Performance nicht beeinträchtigt. Für den Einsatz im Automobilsektor ist daher LPDDR6 eindeutig vorzuziehen.

Für die Charakterisierung der Fehler innerhalb eines Devices (d.h. nicht auf dem Interface) wurde eine umfangreiche Literaturrecherche zu DRAM-Fehlerstudien durchgeführt. Diese Studien analysieren die Fehlerraten verschiedener JEDEC-Standards (DDR4, DDR5, HBM2, LPDDR4, LPDDR5) und Hersteller basierend auf Messungen. Abhängig von der Granularität eines einzelnen Fehlers, d.h. der Anzahl an betroffenen Bits, können Rückschlüsse auf die Fehlerquelle gezogen werden. So betreffen der Großteil der Fehler innerhalb eines Devices nur ein einzelnes Bit (z.B. direkter Partikelschlag auf eine Zelle), was sich leicht durch einen SEC-ECC korrigieren lässt. Deutlich schwieriger zu korrigieren und dadurch kritischer sind Multibitfehler, die z.B. durch einen Defekt der globalen und lokalen Row- und Column-Decoder oder Bitline-Logik entstehen können. Bei der Entwicklung geeigneter Fehlererkennungs- und Fehlerkorrekturmechanismen für diese Art von Defekten kann das Wissen über den internen Aufbau der DRAM-Bausteine einbezogen werden. Eine einfache Möglichkeit zur Erkennung dieser Fehler ist es, eine Prüfsumme der Adresse mit als Daten abzuspeichern. Beim späteren Auslesen werden die Prüfsumme der Leseadresse und die als Daten abgespeicherte Prüfsumme verglichen. Liegt ein Fehler in der Decoder-Logik vor, so stimmen die Prüfsummen mit hoher Wahrscheinlichkeit nicht überein und die Fehler werden erkannt.

Um die Prüfsummen in einem DRAM-Baustein abzulegen, ist zusätzlicher Speicher notwendig. Während LPDDR4 und LPDDR5 keinen dedizierten Speicher dafür bereitstellen, wurde dies in LPDDR6 nun eingeführt. Pro 256-Bit-Speicherzugriff werden zusätzlich 32 Bit an Metadaten übertragen. Die eine Hälfte wird, wie bereits im vorherigen Abschnitt erklärt, zur Absicherung des Links verwendet, die andere Hälfte steht dem Entwickler frei zur Verfügung. Somit hat er volle Flexibilität bei der Implementierung von Fehlererkennung und -korrektur. Ein BCH-Code mit 256 Bit Nutzdaten und 16 Bit Redundanz könnte beispielsweise zwei Bitfehler korrigieren. Wird die Anzahl der Nutzdaten auf 512 Bit verdoppelt, was der üblichen Größe einer Cache Line entspricht, so können mit 32 Bit Redundanz drei Bitfehler korrigiert werden.

ID	Description	Comment	Critical FIT	Failure Type
TLFM-01	Corrupt data: Single-bit error (SBE)	Single corrupted bit in one or multiple words	~70%	SBE
TLFM-02	Corrupt data: Double-bit error (DBE)	Two corrupted bits in one or multiple words		MBE
TLFM-03	Corrupt data: Multiple-bit error (MBE)	Multiple corrupted bits or random data vector		MBE
TLFM-04	Corrupt data: Continuous MBE (CMBE)	Repeated MBE for many/every read access		MBE
TLFM-05	Wrong data	Example: Data from a wrong address	~30%	Address
TLFM-06	Lost / old data	Example: No data written, old data at this address		Address
TLFM-07	No data driven during read operation	Termination pulls data to VSS — all-0 received		MBE
TLFM-08	DQ bus disturbance	Leading to MBE on the shared DQ bus		MBE

Abbildung 4: Verteilung der TLFM

In (Buch, 2024) wurde die Verteilung der sogenannten Top Level Failure Modes (TLFM) für LPDDR4 und LPDDR5 untersucht (siehe Abbildung 4: Verteilung der TLFM). Weil LPDDR6 erst kürzlich veröffentlicht wurde, existieren für diesen Standard noch keine Untersuchungen. Es ist jedoch zu erwarten, dass die Verteilung für LPDDR6 ähnlich aussehen wird. Durch den Link-ECC und einen BCH-Code lassen sich der Großteil dieser Fehler erkennen und auch korrigieren (entweder durch den Code selbst oder durch erneute Übertragung). Problematisch bleiben jedoch insbesondere Wrong Data (TLFM-05) und Lost/Old Data (TLFM-06), da diese bisher nicht abgedeckt sind. Das Lesen von einer falschen Adresse lässt sich durch zusätzliches

Abspeichern oder Verrechnen der Adresse mit den Nutzdaten erkennen. Durch erneutes Auslesen besteht sogar die Möglichkeit, dann die richtigen Daten zu erhalten. Lost Data, d.h. ein zurückliegender Adressierungsfehler, durch den die eigentlichen Daten zerstört werden, lässt sich ebenso durch Abspeichern der Adresse erkennen, jedoch nicht korrigieren. Am schwierigsten zu erkennen ist Old Data, da hier die Adresse übereinstimmt. In diesem Fall müssen zur Fehlererkennung die Daten auf zwei separate DRAM-Bausteine aufgeteilt werden, da man davon ausgeht, dass der Fehler nur in einem Baustein auftritt (Fault Bounds).

Falls dennoch auf einen älteren Standard ohne spezielle Metadaten zurückgegriffen werden soll, oder die Menge der Metadaten nicht ausreichend ist, kann ein Inline-ECC genutzt werden. Dabei wird die Redundanz am Ende einer Row anstelle von Nutzdaten abgelegt. Neben der offensichtlichen Reduzierung der nutzbaren Speicherkapazität wirkt sich ein Inline-ECC jedoch auch negativ auf effektive Bandbreite und Latenz aus, da die Redundanz in einem zusätzlichen Speicherzugriff gelesen oder geschrieben werden muss. Diese Auswirkungen wurden für einen LPDDR-Speicher mit 12,5 % redundanten Daten untersucht, was dem internen Aufbau der NVIDIA® Orin™-Plattform entspricht. Im idealen Szenario mit einem sequenziellen Zugriffsmuster sinkt die erreichbare Performanz um 3,5 %, im ungünstigen Szenario mit einem zufälligen Zugriffsmuster sinkt die Performanz um 14 %. Bezieht man die Bandbreite nur auf die Nutzdaten und nicht auf die ECC-Daten, so beträgt der Verlust bis zu 29 %. Neben der Bandbreite wurde auch das Latenzverhalten in die Analyse mit einbezogen: Im optimalen Fall ist die Beeinträchtigung der Latenz mit nur 1% sehr gering, im ungünstigsten Fall ist eine Erhöhung der Latenz um bis zu 10 % zu verzeichnen. Des Weiteren konnte festgestellt werden, dass mit steigender Bandbreite die Latenz mit Inline-ECC früher saturiert, was dazu führt, dass 25 % weniger zufällige Speicherzugriffe pro Zeiteinheit verarbeitet werden können.

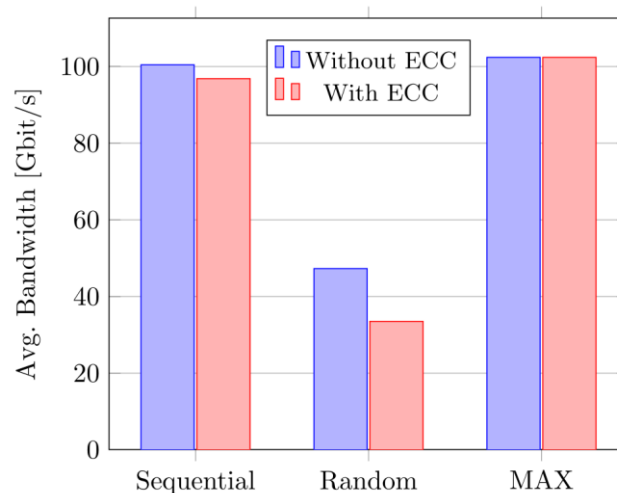


Abbildung 5: Erreichbare Bandbreite für ein sequenzielles und zufälliges Zugriffsmuster ohne und mit aktiviertem Inline-ECC.

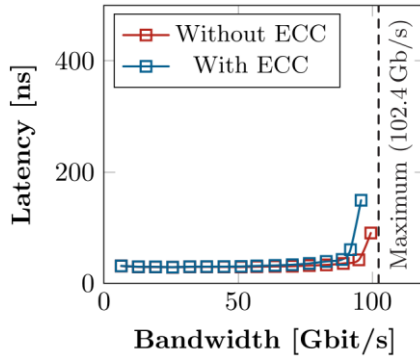


Abbildung 6: Latenz über verschiedene Bandbreiten für ein sequenzielles Zugriffsmuster.

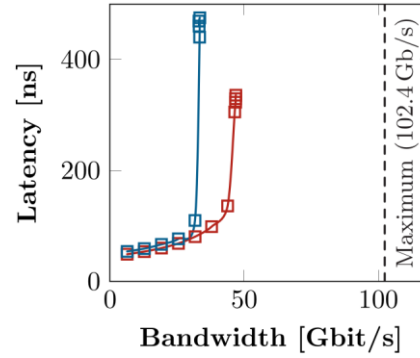


Abbildung 7: Latenz über verschiedene Bandbreiten für ein zufälliges Zugriffsmuster.

Zusammenfassend sind die Leistungseinbußen durch einen Inline-ECC für das Erreichen der Sicherheitsziele akzeptabel, jedoch sind weitere Maßnahmen wie z. B. ein Cyclic Redundancy Check (CRC) erforderlich, um eine höhere ASIL-Stufe als ASIL A zu erreichen. Die Ergebnisse der Analysen wurden im „International Journal of Parallel Programming“ veröffentlicht [16].

Die von RPTU und IESE entwickelten Inline-ECC-Mechanismen im Speichercontroller wurden an LUBIS zur formalen Verifikation übergeben. Insbesondere die Adressierungslogik ist sehr fehleranfällig, weil ein zusammenhängender Speicherbereich (physikalische Adressen) auf einen Speicherbereich mit Löchern (DRAM-Adressen) abgebildet werden muss. Die Löcher entstehen durch die Speicherung der Metadaten. Weiterhin muss sichergestellt werden, dass die Metadaten nach jedem Schreibvorgang durch ein sogenanntes „Read-Modify-Write (RMW)“ aktualisiert werden.

1.4 AP3 (Optimierungen Speichercontroller)

Bei der Erhebung der Anforderungen in AP1 hat sich gezeigt, dass ein gemeinsames DRAM zur Speicherung verschiedener Datenarten mit unterschiedlicher Kritikalität und Resilienz genutzt wird (vgl. Abbildung 8).

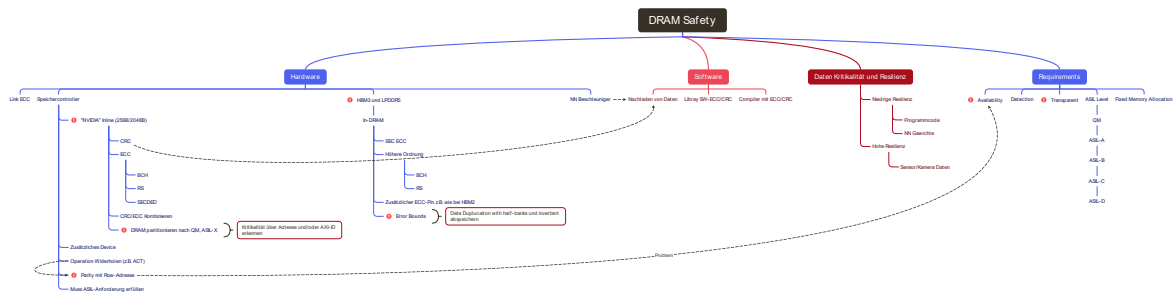


Abbildung 8: Optimierungen im Speichercontroller - Entwurfsraum

Beispielsweise besitzt Programmcode eine hohe Kritikalität und geringe Resilienz, sodass bereits ein Einzelbitfehler zu einem Fehlverhalten des Systems führen kann. Die Inferenz eines

künstlichen neuronalen Netzes kann zwar auch eine hohe Kritikalität besitzen, allerdings ist die Resilienz deutlich höher, sodass sich ein einzelner Fehler in den Eingabewerten (z.B. Bilddaten) meistens nicht in den Ausgabewerten bemerkbar macht. Andere Daten wiederum besitzen eine geringe Kritikalität, bspw. die des Entertainment-Systems.

Da eine höhere Fehlererkennungs- und Fehlerkorrekturfähigkeit immer mit zusätzlichen Kosten einhergeht (z.B. Energie, Latenz, Bandbreite, Speicherbedarf, Chipfläche) ist es deshalb sinnvoll, den Speicher in verschiedene Bereiche mit unterschiedlichen ASIL zu partitionieren. Dies kann besonders flexibel über den Speichercontroller umgesetzt werden. Anhand der Adresse eines Zugriffs entscheidet dieser, welche Fehlererkennungs- und -korrekturverfahren angewendet werden. Eine Möglichkeit ist der im vorigen Kapitel vorgestellte Inline-ECC, bei dem ein Teil jeder DRAM-Row zur Speicherung von Redundanz genutzt wird. Dieses Verfahren sorgt jedoch für eine komplizierte Zuordnung des zusammenhängenden physikalischen Adressbereichs zu DRAM-Adressen (Row, Bank, Column etc.). Für Daten mit hoher Kritikalität und geringem Speicherbedarf (z.B. Programmcode) kann auch eine vereinfachte Variante des Inline-ECC genutzt werden, bei der für die betreffenden Zugriffe vom Speichercontroller die Burstlänge verdoppelt wird (bei LPDDR4/5 von 16 auf 32) und somit die doppelte Menge an Daten transferiert wird. Dadurch steht 100 % der Nutzdatenmenge noch einmal zur Speicherung von Redundanz zur Verfügung. Die Adressierung lässt sich durch einfaches Auslassen eines Adressbits realisieren, allerdings sinkt die nutzbare Speicherkapazität um 50 %, die Latenz der Zugriffe steigt und die verfügbare Bandbreite sinkt.

Durch die Partitionierung des Speichers in verschiedene Kritikalitätsbereiche kann auch das weitere Vorgehen im Falle eines nicht-korrigierbaren Fehlers unterschiedlich gehandhabt werden. Im einfachsten Fall wird der Fehler ignoriert und mit den fehlerhaften Daten weitergearbeitet. Auch das erneute Auslesen der Daten aus dem DRAM kann abhängig von der Fehlerursache zu einer Korrektur führen und vom Speichercontroller durchgeführt werden. Falls es sich bei den Daten nur um lokale Kopien von Daten aus dem Flash-Speicher handelt, können diese auch erneut geladen werden.

Über den Speichercontroller bietet sich auch die Möglichkeit, Daten auf verschiedene Arten im Speicher abzulegen. Bspw. werden bei dem in Servern häufig verwendeten Chipkill-ECC die Nutzdaten und Redundanz so auf die einzelnen Devices eines DIMMs verteilt, dass die Fehler, die durch Ausfallen eines einzelnen Devices entstehen, trotzdem korrigierbar sind. Die Umsetzung eines Chipkill-ECC bei HBM und LPDDR ist meist nicht möglich, da es sich nur um eine geringe Anzahl an Devices handelt (z.B. 1 oder 2). Dennoch kann durch geschicktes Ablegen der Daten im Speicher die Fehlerkorrekturfähigkeit erhöht werden. In der Literaturrecherche zu DRAM-Fehlern in AP2 hat sich gezeigt, dass diese auf eine Vielzahl unterschiedlicher Defekte zurückzuführen sind. Auch wenn ein Großteil der Fehler nur eine einzelne Zelle betreffen, so entstehen dennoch auch eine beachtliche Anzahl an Fehlern durch den Ausfall der Zugriffslogik (globale und lokale Row- und Column-Decoder, Bitline-Logik). Betrachtet man den internen Aufbau von LPDDR-DRAMs, so wird ersichtlich, dass der externe 16-Bit-Datenbus intern in zwei separate 8-Bit-Busse unterteilt wird, die jeweils voneinander unabhängige Zugriffslogik benutzen (siehe Abbildung 9: Interner Aufbau eines LPDDR5-

DRAMs). Somit kann sich ein einzelner Fehler in der Zugriffslogik maximal auf die Hälfte der Daten auswirken. Ähnliches gilt für die DRAM-Bänke innerhalb eines Bausteins. Diese Eigenschaften kann der Speichercontroller bei der Fehlererkennung und Fehlerkorrektur gezielt ausnutzen.

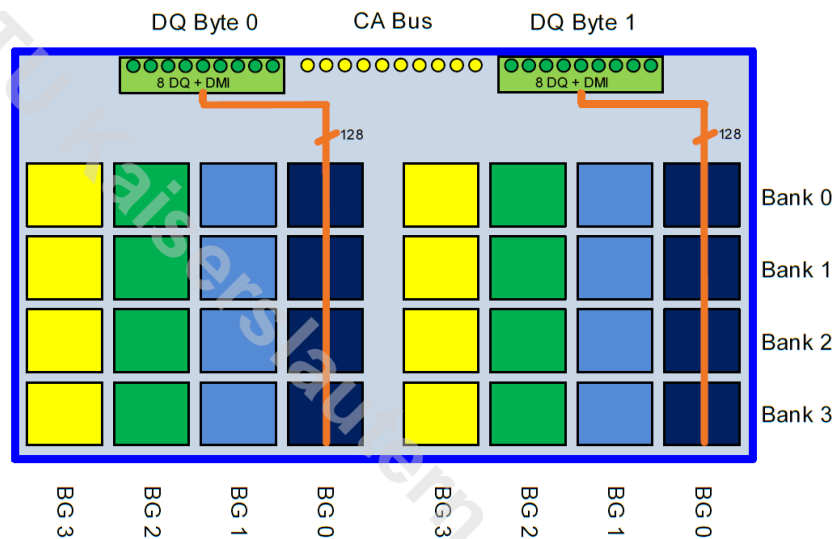


Abbildung 9: Interner Aufbau eines LPDDR5-DRAMs

Die auf dem Papier angegebene Bandbreite eines DRAM-Systems berechnet sich als Produkt aus Datenbusbreite und Datenrate, mit der dieser betrieben wird. In der Realität wird diese Bandbreite jedoch nicht erreicht, weil verschiedene Faktoren wie Page Misses, Wechsel zwischen Lese- und Schreibzugriffen oder Refresh-Kommandos zu internen Verzögerungen im DRAM führen, sodass keine lückenlosen Zugriffe mehr möglich sind. Um diese negativen Einflussfaktoren zu minimieren, gibt es verschiedene Ansätze, die im Speichercontroller umgesetzt werden können.

Eine vom Hardwareaufwand und der zusätzlich entstehenden Latenz sehr günstige Möglichkeit ist die Optimierung des Address-Mappings, d.h. der Abbildung von physikalischen Speicheradressen auf DRAM-Adressen. Diese Optimierung bietet sich insbesondere dann an, wenn die Speicherzugriffe der auszuführenden Applikation bereits im Voraus vollständig bekannt sind, weil die Bestimmung des optimalen Mappings dann als mathematisches Optimierungsproblem formuliert werden kann.

In früheren Arbeiten des Lehrstuhls wurden bereits zwei Ansätze präsentiert. (Jung, et al., 2016) haben beliebige bijektive Abbildungen betrachtet, wodurch jedes beliebige Mapping dargestellt werden konnte. Dies ist in Hardware jedoch nicht realisierbar, da eine Lookup-Tabelle (SRAM) benötigt wird, die in der Größenordnung des DRAMs selbst ist. In (Natale, et al., 2021) wurden daher nur Abbildungen betrachtet, bei denen jeweils ein physikalisches Adressbit mit genau einem DRAM-Adressbit verbunden wird. Dies lässt sich in Hardware mit der in Abbildung 10: Multiplexer-Architektur zur Adressabbildung gezeigten Multiplexer-Architektur realisieren.

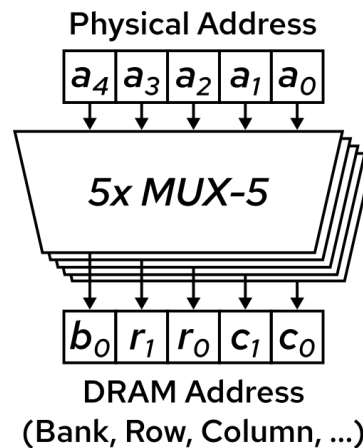


Abbildung 10: Multiplexer-Architektur zur Adressabbildung

Die Anzahl der möglichen Abbildungen ist dadurch jedoch deutlich reduziert: Während für einen Adressbereich mit n Bit $2^n!$ bijektive Abbildungen existieren, lassen sich mit den Multiplexern „nur“ $n!$ bijektive Abbildungen realisieren. In der Praxis eignet sich die Multiplexer-Architektur gut für sequenzielle Zugriffsmuster oder Zugriffsmuster mit einer festen Schrittweite („Fixed Stride“), da die Adressbits mit hohen Umschaltraten auf Bank- und Column-Bits abgebildet werden können. Für den Fall, dass mehrere Anwendungen mit unterschiedlichen Schrittweiten auf den Speicher zugreifen, wird hingegen keine Abbildung gefunden, mit der dauerhaft eine hohe Speicherbandbreite erzielt wird.

Daher wurde nach einer Möglichkeit gesucht, mit ähnlich geringer Hardwarekomplexität mehr Adressabbildungen zu realisieren. Dies wurde durch eine Abbildung erreicht, die wie in Abbildung 11: Adressabbildung mit Matrix auf einer Matrixmultiplikation über $GF(2)$ basiert.

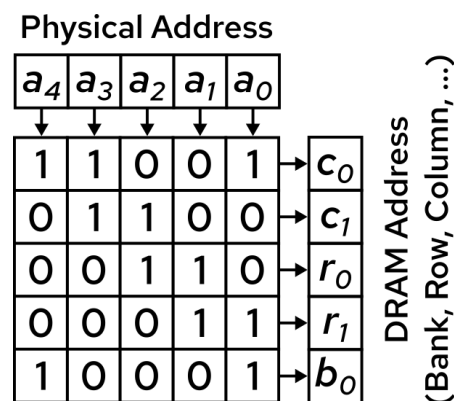


Abbildung 11: Adressabbildung mit Matrix

Für diesen Fall ist die Anzahl der darstellbaren Abbildungen $\prod_{k=0}^{n-1} (2^n - 2^k)$. In Hardware lässt sich die Matrixmultiplikation durch AND-Gatter und XOR-Gatter realisieren. Die Multiplexer-Architektur ist ein Spezialfall der Matrixmultiplikation, bei der in jeder Zeile und Spalte genau eine Eins steht. Die Matrixmultiplikation bietet den Vorteil, dass auch bei mehreren Strides jeweils die Adressbits mit hohen Umschaltraten auf Bank- und Column-Bits abgebildet werden können, weil die Anzahl an Einsen in den Zeilen der Matrix nicht beschränkt ist. Der neue Ansatz wurde auf der „MEMSYS“-Konferenz publiziert [2].

Als letzte Möglichkeit zur Optimierung des Address-Mappings wurde die Kombination von Matrix und Lookup-Tabelle betrachtet. Hierzu wurde die Abbildung in einen linearen und einen nichtlinearen Teil zerlegt. Die Zerlegung erfolgt so, dass der nichtlineare Teil, der später mithilfe von Lookup-Tabellen realisiert werden muss, nur von einer geringen Anzahl der physikalischen Adressbits abhängt. Dadurch bleibt die Hardwarekomplexität gering, die Anzahl an darstellbaren Abbildungen wird aber im Vergleich zur reinen Matrix noch einmal erhöht.

Die aufgezeigten Optimierungen setzen voraus, dass die Speicherzugriffe im Voraus bekannt sind. Ist dies nicht der Fall, oder vermischen sich die Speicherzugriffe mehrerer Applikationen oder Beschleuniger, wie es häufig bei zentralen Rechnern im Auto der Fall ist, dann ist die Bestimmung einer optimalen Adressabbildung hingegen nicht möglich. Dennoch kann auf Basis von statistischen Daten über die Speicherzugriffe eine verbesserte Adressabbildung und Konfiguration des Speichercontrollers gefunden werden. Solche statistischen Daten sind zum Beispiel die Umschaltraten der einzelnen physikalischen Adressbits oder die Anzahl der Wechsel zwischen Lese- und Schreibzugriffen. Um diese Daten auf einem realen System während der Laufzeit zu sammeln, wurde ein Zählermodul entwickelt, welches direkt im Speichercontroller integriert ist. Zur Minimierung der Wechsel zwischen Lese- und Schreibzugriffen wurde ein Logikblock entworfen, der es erlaubt, im Speichercontroller Lese- und Schreibzugriffe zu gruppieren. Dieser Block wurde auch in einen bestehenden Speichercontroller eingebaut und getestet.

Allgemein bietet ein DRAM-Subsystem eine Vielzahl an Konfigurationsmöglichkeiten (Speichercontroller, Devices, Architektur). Sind bestimmte Anwendungen vorgegeben (wie in unserem Fall von Mercedes Benz und Aumovio), dann kann die Performance einer bestimmten Konfiguration für diese Anwendungen durch Simulationen evaluiert werden. Soll jedoch die optimale Konfiguration gefunden werden, dann müssen alle möglichen Konfigurationen simuliert werden („Exhaustive Simulation“). Dies ist in der Realität aus Zeitgründen nicht möglich. Es wurde daher ein Framework zur automatischen Konfigurationssuche entwickelt, dessen Laufzeit nur logarithmisch von der Anzahl der möglichen Konfigurationen abhängt. Die Ergebnisse wurden auf dem „RAPIDO“-Workshop der „HIPEAC“-Konferenz publiziert [8].

Um mögliche Optimierungen im Flash-HW-Controller zu evaluieren ist es notwendig, Simulationen mit hoher Geschwindigkeit und hoher Genauigkeit durchzuführen. Für DRAM nutzt die RPTU hierzu seit vielen Jahren den eigenen SystemC-basierten Speichersimulator DRAMSys, welcher seit 2020 auch auf GitHub veröffentlicht ist und mittlerweile eine breite Nutzerschaft hat. Auf Basis von DRAMSys wurde im Laufe des Projekts der Simulator ONFISys entwickelt. Dieser erlaubt die Simulation von Flash-HW-Controllern und NAND-Flash-Devices, die über das sogenannte „Open NAND Flash Interface (ONFI)“ verbunden sind. Ein entscheidender Unterschied zwischen DRAM und Flash ist die Direktionalität der Kommunikation zwischen Controller und Speicher. Während im DRAM die Kommandos immer vom Controller zum Speicher geschickt werden und eine festgelegte Zeit zur Ausführung benötigen, variiert beim Flash die Ausführungszeit und der Speicher teilt dem Controller die Fertigstellung aktiv mit. Dieser Unterschied musste bei der Entwicklung von

ONFISys beachtet werden. Hierzu waren Anpassungen des Controllersimulationsmodells nötig. Zudem wurde ein Flash-Timing-Modell entwickelt, das auch nichtdeterministische Verzögerungen erlaubt. ONFISys wurde im Anschluss an Hyperstone zur Benutzung und Weiterentwicklung übergeben.

Im Bereich der holistischen Optimierung wurde untersucht, ob der DRAM-Controller den Decoder des Flash-HW-Controllers zur Fehlerkorrektur mitbenutzen kann, um den Flächenbedarf zu senken. Grundsätzlich verwendet Flash deutlich komplexere und leistungsfähigere Fehlerkorrekturmechanismen (z.B. LDPC-Codes mit Soft-Information), da es sich im Vergleich zu DRAM um einen sehr unzuverlässigen Speicher mit deutlich höheren Fehlerraten handelt. Eine fehlerfreie Benutzung von Flash ohne Fehlerkorrektur ist schlichtweg nicht möglich. Auch die Zugriffsgranularität von Flash und damit die Codewortlänge ist größer. Eine erarbeitete Möglichkeit wäre, die einzelnen DRAM-Zugriffe nur mit einer Fehlererkennung abzusichern (wie bspw. bei LPDDR6), und zusätzlich mehrere DRAM-Zugriffe mit einem In-Line-ECC abzusichern, der den gleichen Fehlerkorrekturcode wie Flash verwendet. Wird beim Lesen vom DRAM ein Fehler erkannt, könnte dieser über den Decoder im Flash-HW-Controller korrigiert werden. Die durchschnittliche Performance des Flashs würde nicht beeinträchtigt, da das Auftreten eines Fehlers im DRAM und damit die Notwendigkeit zur Korrektur sehr selten ist. Der Ansatz der gemeinsamen Nutzung ist nur dann umsetzbar, wenn sich beide Controller auf dem gleichen Chip befinden. Während in der Vergangenheit der Flash-Controller meistens auf einem separaten Chip platziert war, beinhalten immer mehr moderne SoCs auch den Flash-Controller.

Zur Evaluierung der Performance von Near- und In-Memory-Computing wurde ein weiterer Simulator namens PIMSys auf der Basis von DRAMSys entwickelt. Dieser bildet prototypisch das von Samsung veröffentlichte PIM-HBM nach. Durch Kopplung des Simulators mit gem5 und eine eigens entwickelte Rust-Softwarebibliothek können die Operationen im DRAM direkt aus der Software heraus gesteuert werden. Der gewählte Modellierungsansatz ist außerdem flexibel, sodass in Zukunft auch andere Near- und In-Memory-Computing-Ansätze nachgebildet werden können. Der Simulator wurde außerdem auf der „MEMSYS“-Konferenz präsentiert [6].

1.5 AP4 (Demonstrator)

Um die von Mercedes-Benz und Aumovio beschriebenen Anwendungsszenarien aus AP 1 nachzubilden, wurden die in den vorigen Arbeitspaketen erstellten Simulationsmodelle parametrisiert. Die entworfenen Initiatoren wurden so konfiguriert, dass sie Speicherzugriffe mit den korrekten Zugriffsmustern, Adressbereichen und Zugriffsfrequenzen erzeugen. Anschließend wurden sowohl LPDDR5- als auch LPDDR6-DRAM-Subsysteme mit unterschiedlichen Fehlerkorrekturschemata nachgebildet (kein ECC, In-Line-ECC wie bei NVIDIA Orin und Kombination von CRC+BCH). Für verschiedene Fehlerraten wurden dann Simulationen durchgeführt und sowohl Bandbreite als auch Energie der Fehlerkorrekturschemata für beide DRAM-Standards verglichen.

2 Projektumsetzung

2.1 Finanztechnische Projektumsetzung

Zur Projektdurchführung waren finanzielle Ressourcen für Personalkosten, Reisekosten und Materialkosten notwendig. Die beantragten Fördermittel betragen in Summe 284.713,45 €. Für wissenschaftliche Mitarbeiter wurden Gelder in Höhe von 218.856,40 € verausgabt (Pos. 812). Die Reisekosten beliefen sich auf 2.548,42 € (Pos. 846) und die Ausgaben für Geräte auf 16.195,90 € (Pos. 850).

Die Materialkosten haben die zu Beginn veranschlagten Kosten um etwa 20 % überschritten. Grund hierfür war, dass der Preis der aktiven Tastköpfe, die zur Umsetzung von AP2 benötigt wurden, von der Beantragung bis zur Anschaffung gestiegen war. Durch eine Reduzierung der Reisekosten konnte dies jedoch kompensiert werden. Daher entsprechen die tatsächlichen Ausgaben nahezu den veranschlagten Kosten.

Der Projektpartnerwechsel wurde durch eine kostenneutrale Verlängerung aufgefangen und hat daher keine finanztechnischen Auswirkungen.

2.2 Inhaltliche Projektumsetzung

Die Projektziele wurden in vollem Umfang erreicht. Die durch den Projektpartnerwechsel entstandenen Verzögerungen konnten durch eine kostenneutrale Verlängerung kompensiert werden. Zudem ergab sich dadurch die Möglichkeit, den erst Mitte 2025 veröffentlichten Standard LPDDR6 noch stärker ins Projekt einzubeziehen.

3 Notwendigkeit und Angemessenheit der geleisteten Projektarbeiten

Die in den einzelnen Arbeitspaketen geleisteten Arbeiten waren notwendig und angemessen, um die Ziele des Vorhabens zu erreichen. Ohne die Förderung wäre es nicht möglich gewesen, die Forschung, Entwicklung und Implementierung in den einzelnen Arbeitspaketen in dem benötigten Umfang umzusetzen. Die geleisteten Projektarbeiten entsprechen im Wesentlichen den im Antrag vorgesehenen Arbeiten. Die durch den Projektpartnerwechsel entstandenen Verzögerungen konnten durch die kostenneutrale Verlängerung ausgeglichen werden. Insgesamt wurden die Projektziele somit erreicht.

4 Nutzen der geleisteten Projektarbeiten

Die wichtigsten Projektergebnisse wurden durch wissenschaftliche Publikationen und Präsentationen auf Konferenzen der Öffentlichkeit verfügbar gemacht. Somit ist die wissenschaftliche Verwertung sichergestellt.

Weiterhin erfolgte während des Projekts eine enge Zusammenarbeit mit dem Industriepartner Micron. Die innerhalb des Projekts gesammelten Ergebnisse konnten damit in die Entwicklung neuer Produkte einfließen. Da Micron als einer der drei weltweit größten DRAM-Hersteller auch maßgeblich an der Spezifizierung neuer DRAM-Standards beteiligt ist,

konnten die Projektergebnisse auch dort einfließen. Die weitere Zusammenarbeit mit Micron in Zukunft ist ebenso geplant.

Im Rahmen des Projekts wurden an der RPTU drei studentische Abschlussarbeiten umgesetzt. Zudem wurde ein Promotionsvorhaben unterstützt.

Die (weiter-)entwickelten Simulationswerkzeuge finden auch in zukünftiger Forschung Anwendung.

5 Fortschritt auf dem Gebiet des Vorhabens bei anderen Stellen

Durch unsere Arbeiten während des Projekts wurde ersichtlich, dass die verfügbaren Safety-Mechanismen in LPDDR4 und LPDDR5 nur ausreichen, um eine ASIL-A-Zertifizierung zu erreichen. Diese wichtigen Informationen wurden auch an Micron übermittelt. Der 2025 neu veröffentlichte DRAM-Standard LPDDR6 führt mehrere zusätzliche Safety-Mechanismen ein und ist damit ein deutlicher Schritt in Richtung ASIL-D.

6 Veröffentlichungen

- [1] **Split'n'Cover: ISO 26262 Hardware Safety Analysis with SystemC**
L. Steiner, K. Kraft, D. Christ, D. Uecker, C. Malek, M. Jung, N. Wehn, International Journal for Parallel Programming (IJPP)
- [2] **A Mathematical Model for XOR-Based Application Specific DRAM Address Mapping Schemes**
A. Rotaru, O. Bachtler, L. Steiner, M. Jung, S. O. Krumke, N. Wehn. 11th International Symposium on Memory Systems (MEMSYS), October 2025, Washington D.C., USA.
- [3] **Performance and Power Analysis of LPDDR6**
D. Christ, L. Steiner, T. Zimmermann, M. Mörz, N. Wilbert, M. Jung, N. Wehn. IEEE Cross-disciplinary Conference on Memory-Centric Computing (CCMCC), October, 2025, Dresden, Germany.
- [4] **DRAMPower 5: An Open-Source Power Simulator for Current Generation DRAM Standards**
L. Steiner, T. Psota, M. Mörz, D. Christ, M. Jung, N. Wehn In HiPEAC RAPIDO 2025 Workshop ACM, Barcelona, Spain, 2025.
- [5] **A Novel System Simulation Framework for HBM2 FPGA Platforms**
H. G. M. Hernandez, V. Iskandar, L. Steiner, P. Holzinger, M. Jung, D. Goehringer, M. Huebner, N. Wehn and M. Reichenbach Springer LNCS International Conference on Embedded Computer Systems Architectures Modeling and Simulation (SAMOS), July, 2024, Samos Island, Greece.
- [6] **PIMSys: A Virtual Prototype for Processing in Memory**
D. Christ, L. Steiner, N. Wehn, M. Jung. 10th International Symposium on Memory Systems (MEMSYS 2024), October, 2024, Washington, DC, USA.
- [7] **A Precise Measurement Platform for LPDDR4 Memories**
J. Feldmann, L. Steiner, D. Christ, T. Psota, M. Jung, and N. Wehn. ACM International Symposium on Memory Systems (MEMSYS 2023), October, 2023, Washington, D.C., USA
- [8] **Automatic DRAM Subsystem Configuration with irace**
L. Steiner, G. Delazeri, I. Prando da Silva, M. Jung, N. Wehn. International Conference on High-Performance and Embedded Architectures and Compilers (HiPEAC), Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO), January, 2023, Toulouse, France.

7 Literaturverzeichnis

- Buch, S. (2024). Error Detecting and Correcting Codes for DRAM Functional Safety. In *Proceedings of the International Symposium on Memory Systems (MEMSYS'23)* (S. 1-5). New York, NY, USA: Association for Computing Machinery (ACM).
- Jung, M., Heinrich, I., Natale, M., Mathew, D. M., Weis, C., Krumke, S., & Wehn, N. (2016). ConGen: An Application Specific DRAM Memory Controller Generator. In *Proceedings of the Second International Symposium on Memory Systems (MEMSYS '16)* (S. 257-267). New York, NY, USA: Association for Computing Machinery (ACM).
- Natale, M. V., Jung, M., Kraft, K., Lauer, F., Feldmann, J., Sudarshan, C., Wehn, N. (2021). Efficient Generation of Application Specific Memory Controllers. In *Proceedings of the International Symposium on Memory Systems (MEMSYS '20)* (S. 233-247). New York, NY, USA: Association for Computing Machinery (ACM).