Entwicklung einer FPGA-basierten asymmetrischen MPSoC Architektur

Heiko Wilken

Seminar Ausarbeitung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Entwicklung einer FPGA-basierten asymmetrischen MPSoC Architektur

Heiko Wilken

Seminar Ausarbeitung
im Studiengang Master Informatik
Department Informatik
in der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Professor : Prof. Dr.-Ing. Bernd Schwarz
Gutachter : Prof. Dr. rer. nat. Kai von Luck
Gutachter : Prof. Dr. rer. nat. Gunter Klemnke

Abgegeben am 22. Februar 2011

Inhaltsverzeichnis

1 Einleitung **4**
1.1 Zielsetzung 5

2 Entwicklungsschwerpunkte **6**
2.1 Shared Memory Speicherarchitektur 7
2.2 Gewährleistung des Wechselseitigen Ausschlusses 8
2.3 Interprozesskommunikation zwischen Prozessoren 9
2.4 Hardware / Software Partitionierung 10
2.4.1 RTOS Implementierung 11

3 Risiken & Herausforderungen **12**

4 Zusammenfassung & Ausblick **13**

Literaturverzeichnis **15**

Abbildungsverzeichnis **17**

1 Einleitung

Multiprozessorsysteme werden eingesetzt um den leistungsbezogenen Anforderungen heutiger Anwendungen gerecht zu werden. Der stetig steigende Bedarf an Rechenleistung erfordert ein Umdenken im Systementwurf. Die Erhöhung der Taktrate bei Singlecores zur Erzielung von Leistungssteigerungen ist aufgrund von Stromverbrauch und Wärmeabgabe nicht realisierbar [9]. Vor allem bei Echtzeitsystemen mit zeitkritischen Anfordungen und hohen Datenraten wird im Vergleich zu einem Einprozessorsystem bei niedrigeren Taktfrequenzen eine Erhöhung des Datendurchsatzes erreicht. Durch den Einsatz von FPGA basierten „Multiprocessor System-on-Chips (MPSoC)“ wird der Systemdurchsatz erhöht, indem verschiedene Programme parallel auf mehreren Prozessoren ausgeführt werden. Die FPGA Technologie nutzt durch die getrennte Konfiguration der Hardwarebeschleunigermodule eine Parallelität auf Hardwareebene und bietet dadurch eine hohe Flexibilität. Beim „HW/SW-Codesign“ werden ressourcenlastige Softwareteile extrahiert und in spezielle Hardwarestrukturen implementiert. Mit der Hinzunahme von zusätzlicher Hardware wird der Durchsatz und damit die Beschleunigung des Gesamtsystems gesteigert. Die HW/SW Partitionierung ist bei der Implementierung von Multiprozessorsystemen von wichtiger Bedeutung. Durch die Auslagerung von sequentiell ablaufenden Befehlsfolgen auf verschiedene Prozessoren werden Aufgaben parallel ausgeführt. Die gleichzeitige Reduzierung der Prozessorauslastung minimiert den Energiebedarf des Systems [6].

Eng gekoppelte Multiprozessoren sind asymmetrisch, wenn die Prozessoren unabhängig voneinander verschiedene Aufgaben bearbeiten. Die Heterogenität der Prozessoren ermöglicht eine unterschiedliche Konfiguration. Als Beispiel für eine AMP Architektur kann der IBM Cell Prozessor der Sony Playstation 3 genannt werden. Durch einen Synchronisationsmechanismus wird der wechselseitige Zugriff auf gemeinsame Hardware Ressourcen gewährleistet. Dies ist vor allem bei der Verwendung eines gemeinsamen Speichers erforderlich. Da die Prozessoren unabhängig voneinander arbeiten, werden Ergebnisse und Kontrollinformationen über eine Interprozesskommunikation ausgetauscht, beispielsweise mit dem Einsatz einer „Mailbox“. Im Gegensatz zu symmetrischen Multiprozessoren, die von einem zentralen Betriebssystem gesteuert werden, können bei asymmetrischen Multiprozessoren verschiedene Betriebssysteme implementiert werden. Durch die Trennung der Prozessoren und durch den Einsatz von Threads kann bei der Implementierung eines Echtzeitbetriebssystems (RTOS) eine doppelte Parallelisierung erreicht werden. Die Leistungssteigerung des Multiprozessorsystems ist hierbei von der Parallelisierbarkeit der Software abhängig. Dies wird durch das Amdahlsche Gesetz, welches aussagt, dass ein Programm nie vollständig parallel ausgeführt werden kann, da einige Teile, wie Initialisierung und Speicher Allokation stets sequentiell ausgeführt werden müssen, bestätigt [1].

1.1 Zielsetzung

"High Performance Embedded Computing"(HPEC) ist ein Forschungsschwerpunkt des Projektes FAUST der HAW Hamburg. Ziel ist der Aufbau einer FPGA-basierten Fahrzeugsteuerungsplattform für ein autonomes Fahrzeug. Im Rahmen der anstehenden Masterarbeit wird eine „FPGA-basierte Multiprozessorplattform“ für den Einsatz in einem autonomen Fahrzeug entwickelt und implementiert. In Abb. 2.1 ist der erste MPSoC Prototyp dargestellt. Dieser wird im weiteren Verlauf der Masterarbeit kontinuierlich weiterentwickelt. Sowohl das FAUST „Sensor Controlled Vehicle“ als auch das FAUST „High Performance Embedded Computing“ Fahrzeug dienen als Einsatzgebiete für die MPSoC Plattform. Folgende Entwicklungsschritte sind für die Masterarbeit geplant:

- Im ersten Schritt werden zwei MicroBlaze Prozessoren in einer asymmetrischen „Shared Memory Architektur“ aufgebaut. Über den externen 256 MB großen DDR2 Speicher sind die Prozessoren eng miteinander gekoppelt. Ein in Hardware implementierter XPS Mutex gewährleistet durch einen Synchronisationsmechanismus den wechselseitigen Ausschluss auf die gemeinsamen Ressourcen. Durch die asymmetrische Architektur haben die Prozessoren eigenständige Aufgaben und kommunizieren über eine Interprozesskommunikation. Je nach Aufgabenkomplexität kann die Anzahl der MicroBlazes varieren.
- Im zweiten Schritt werden zwei verschiedene Echtzeitbetriebssysteme (RTOS) auf den MicroBlazes implementiert. Durch die gewonnene POSIX Thread Implementierung kann softwareseitig eine weitere Quasi-Parallelität erreicht werden. Eine Erhöhung des Datendurchsatzes und eine Verringerung der Ausführungszeit wird erwartet.
- Im dritten Schritt soll die asymmetrische Architektur durch eine symmetrische Architektur ersetzt werden. Ein zentraler MicroBlaze steuert die Verteilung der Aufgaben an die zur Verfügung stehenden Prozessoren. Die Herausforderung besteht in der Parallelisierung der Software. Parallelisierende Compiler übersetzen durch eine Untersuchung der Datenabhängigkeiten sequentielle Programme (C Programme) in ausführbare Multiprozessor Programme [3]. Im Bereich der Eingebetteten Systeme ist die Implementierung von SMP aufgrund des fehlenden Compilers schwer realisierbar. Hierfür ist die Modifikation eines RTOS nötig.

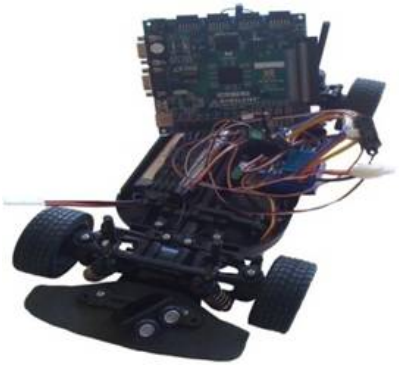

Abbildung 1.1: FAUST HPEC Fahrzeug

Abbildung 1.2: Sensor Controlled Vehicle

2 Entwicklungsschwerpunkte

Das asymmetrische MPSoC besteht aus zwei unabhängigen MicroBlazes, die an zwei getrennte PLB Bussysteme angeschlossen sind (vgl. Abb. 2.1). Durch die Asymmetrie der Architektur können die Prozessoren heterogen konfiguriert werden und somit eigenständige Aufgaben bearbeiten. Der Einsatz der Xilinx spezifischen Bussysteme (FSL und PLB) hat zur Folge, dass die Wiederverwendbarkeit und die Portierbarkeit der Komponenten von Xilinx FPGAs abhängig ist. Aus diesem Grund kann im weiteren Verlauf auf den von ARM entwickelten und zu AMBA[1] gehörenden „Advanced eXtensible Interface Bus (AXI)“ zurückgegriffen werden [2][16].

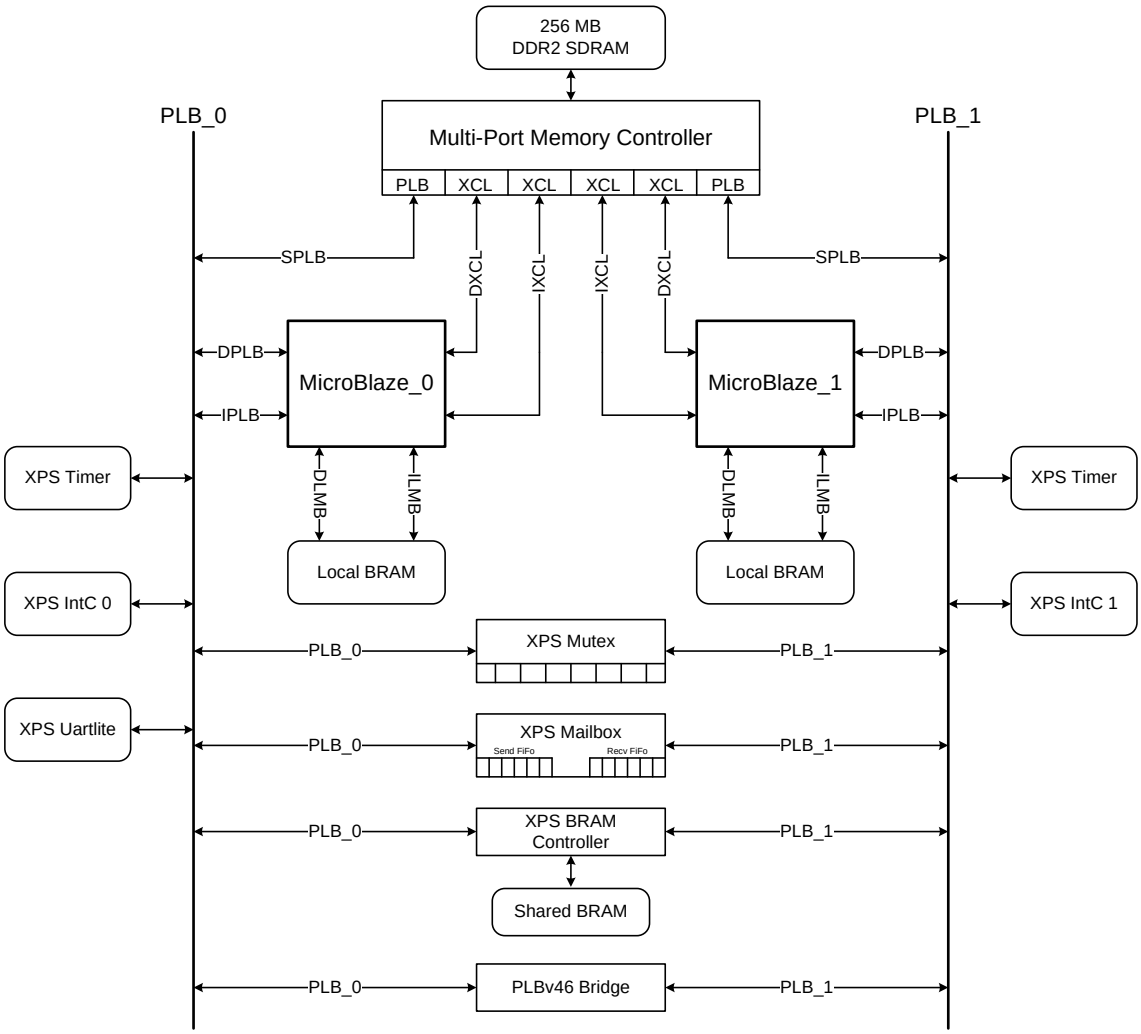

Abbildung 2.1: Dual MicroBlaze MPSoC mit Shared Memory Architektur und Interprozesskommunikation [12]

[1] Advanced Microcontroller Bus Architecture

2.1 Shared Memory Speicherarchitektur

Die UMA[2] Shared-Memory Architektur beinhaltet zwei Softcore MicroBlaze Prozessoren, die über den „Xilinx Cache Link (XCL)“ an den MPMC Speicher Controller angeschlossen sind (vgl. Abb. 2.1). Über den MPMC haben die Prozessoren Zugriff auf den gemeinsamen Speicher. Hierbei ist die Zugriffszeit und der Datendurchsatz jeweils gleich (UMA). Für den Zugriff der Prozessoren ist eine Synchronisation durch einen Mutex erforderlich. Gemeinsame On-Chip Speicherbereiche sind effiziente Kommunikationsinfrastrukturen, die eine schnelle Datenübertragung zwischen zwei Prozessoren bieten (vgl. Abb. 2.1). Die Performance eines On-Chip BRAMs ist ähnlich der Zugriffszeit eines „Cache Hits“ [12]. Eine Synchronisation der BRAM Zugriffe ist nicht nötig, da der Dual-Port RAM über eine eigene Arbitrationslogik verfügt [14].

Durch die Asymmetrie der Prozessorarchitektur hat jeder MicroBlaze ein eigenes ausführbares Programm. Da in einem FPGA-basierten Multiprozessorsystem die Bootbereiche der Software nicht in einem gemeinsamen Speicher liegen können, besitzt jeder Prozessor einen zusätzlichen lokalen BRAM Speicher. Bei Systemstart wird die jeweilige Software aus dem lokalen BRAM gestartet. Der ausführbare Programmcode werden per Linker Script im externen DDR2 Speicher gespeichert (vgl. Abb. 2.2). Eine Partitionierung der Software hat den Vorteil, dass die lokalen BRAMs mit einer minimalen Größe von 8 KB implementiert werden können.

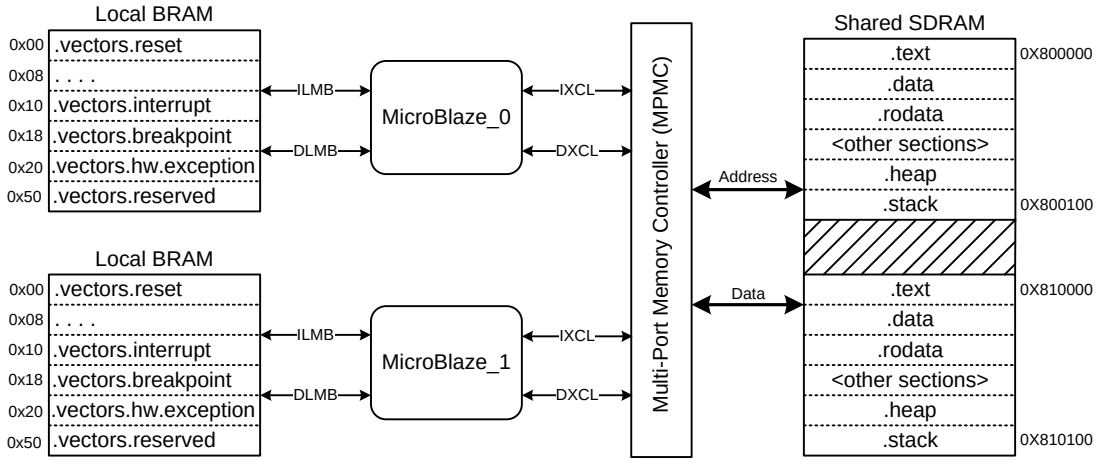

Abbildung 2.2: Software Memory Mapping eines Dual MicroBlazes [12]

Die Herausforderung bei der Implementierung, ist die Minimierung des Speicherflaschenhalses. Dieser entsteht bei der Programmausführung aus dem externen Speicher aufgrund des seriellen Zugriffes des MPMCs auf den DDR2 (nur ein Adress- und Datenbus). Durch den XCL Bus werden sowohl die Daten als auch die Instruktionen gecached und damit der Flaschenhals verringert. Jedoch muss die Kohärenz der Caches durch eine softwareseitige Implementierung sicher gestellt werden. Für die Steigerung der Performance wäre der Einsatz einer symmetrischen Multiprozessorarchitektur von Vorteil. Hierbei wird die Software nur einmalig gespeichert und alle Prozessoren führen das gleiche Programm aus. Ein zentrales Echtzeitbetriebssystem steuert die Verteilung der Aufgaben. Bei „General

[2] Uniform Memory Architectur

Purpose Computer“ übernimmt eine virtuelle Maschine oder ein darauf ausgelegter Compiler die Verteilung der Aufgaben, beispielsweise die „Java Virtual Machine“ oder die „OpenMP“ Programmierschnittstelle. Parallelisierende Compiler übersetzen durch eine Untersuchung der Datenabhängigkeiten sequentielle Programme (C Programme) in ausführbare Multiprozessor Programme [3]. Im Bereich der Eingebetteten Systeme ist die Implementierung von SMP aufgrund des fehlenden Compilers schwer realisierbar.

2.2 Gewährleistung des Wechselseitigen Ausschlusses

In Multiprozessorsystemen, bei denen mehrere Prozessoren auf gemeinsame Komponenten zugreifen, müssen die Zugriffe synchronisiert werden. Der XPS Mutex ist ein in Hardware implementierter IP Block, der mit Hilfe eines 32 Bit großen LOCK Registers die kritischen Speicherbereiche schützt indem er bei nebenläufigen Prozessen die Zugriffe koordiniert (vgl. Abb. 2.1). Sowohl der gemeinsame externe DDR2 Speicher als auch gemeinsame Peripherie Komponenten (UART) dürfen erst nach einem vorherigen Sperrmechanimus benutzt werden. Durch eine „Memory Mapped Adressierung“ kann jeder Prozessor über den PLB Bus direkt auf die Register des Mutexes zugreifen. Die insgesamt acht Mutex Interfaces haben exklusiven Zugriff auf die Register. Da bei einem Zugriff alle anderen Interfaces blockieren, kann eine Manipulation der Mutex Register ausgeschlossen werden. Ein zusätzliches konfigurierbares „User Configuration Register“ speichert zu jedem Mutex Register die zu synchronisierte Speicheradresse. Für ressourcenlastige Systeme empfiehlt es sich, dieses Register zu deaktivieren und das Mapping der Adressen in Software zu implementieren. Damit die Eigentümerschaft bei einem Sperrvorgang eindeutig bestätigt werden kann, wird die zugehörige CPU ID mit einer Kontrollogik überprüft.

In Vorarbeiten wurden die Zugriffszeiten bei einem Sperr- und Entsperrvorgang gemessen. Diese sind ausschlaggebend für die spätere Performance des Gesamtsystems. Im Falle von vielen gemeinsamen Ressourcen kann der Mutex durch den Sperrmechanimus zu einer Verzögerung der Ausführungszeit führen. Der dabei entstehende Flaschenhals ist auch auf den sequentiellen PLB Zugriff zurückzuführen. Die Messung ergab, dass ein Sperren des Mutexes durch die MLOCK() Funktion im Durchschnitt 0,004288 ms benötigt. Bei einer Taktfrequenz von 125 MHz entspricht dies 536 Taktzyklen. Da die Entsperrfunktion bei jedem Aufruf zuerst das Status Register überprüft, ist die Zugriffszeit im Vergleich zur Sperrfunktion leicht erhöht (695 Taktzyklen).

Bei der Implementierung des MPSoC’s werden zwei Synchronisationsmethoden eingesetzt. Über den hardwareseitigen XPS Mutex werden die Prozessoren mit den gemeinsamen Ressourcen, die teilweise an getrennten Bussystemen angeschlossen sind, synchronisiert. Zusätzlich soll ein RTOS mit POSIX Interface implementiert werden. Durch die damit gewonnenen POSIX Threads werden die nebenläufig ablaufenden Aufgaben eines Prozessors softwareseitig miteinander synchronisiert.

Um einen guten Datendurchsatz und eine geringe Ausführungszeit zu erlangen, besteht die Herausforderung darin, den Synchronisationsaufwand möglichst gering zu halten. Unnötige Latenzen sollen weitestgehend vermieden werden.

2.3 Interprozesskommunikation zwischen Prozessoren

Eine wesentliche Herausforderung bei der Integration von Multiprozessorsystemen ist die Kommunikation zwischen den Prozessoren, die sogenannte Interprozesskommunikation (IPC). Damit keine unnötigen Delays entstehen und eine hohe Bandbreite beim Datentransfer ermöglicht werden kann, sollten die Signallaufzeiten so minimal wie möglich sein. Man unterscheidet zwischen zwei Kommunikationsprinzipien [10]:

- Indirekte Kommunikation: Die Prozessoren tauschen per DMA Transfer Daten über einen gemeinsamen synchronisierten Speicher aus.
- Direkte Kommunikation: Die Prozessoren kommunizieren direkt über expliziten einen Nachrichtenversand.

Für das in der Masterarbeit zu entwickelte MPSoC werden beide Kommunikationsprinzipien angewandt. Große Datenmengen werden über den gemeinsamen externen DDR2 Speicher ausgetauscht. Die Prozessoren, die an getrennte PLB Bussysteme angeschlossen sind, können mit einem DMA Transfer direkt über den MPMC auf den Speicher zugreifen. Hierbei können „Data Races“ entstehen, wenn zwei Prozessoren gleichzeitig den gleichen oder auf einen überlappenden Speicherbereich zugreifen [4]. Aus diesem Grund muss der Zugriff auf den gemeinsamen Speicherbereich synchronisiert werden (vgl. Kap. 2.2). Da die Latenz beim Zugriff auf den gemeinsamen Speicher höher als die Latenz auf lokalen On-Chip Speicher ist, werden Cache-Speicher zur Leistungssteigerung und zur Latenzverringerung eingesetzt [5].

Zur direkten Kommunikation werden die Prozessoren zusätzlich durch den unidirektionalen FSL Bus miteinander verbunden. Die Punkt-zu-Punkt Verbindung besitzt eine interne FIFO und kann entweder synchron oder asynchron betrieben werden [13]. Durch die Kopplung der Registerfiles können die Prozessoren direkt auf die Register des anderen zugreifen. Dies hat gegenüber der indirekten Kommunikation einen erheblichen Geschwindigkeitsvorteil. Jedoch nur für geringe Datenmengen. Aus diesem Grund wird eine Kombination aus indirekter und direkter Kommunikation eingesetzt.

Eine weitere direkte Kommunikationsmöglichkeit ist die in Abb. 2.1 dargestellte XPS Mailbox. Durch zwei getrennt konfigurierbare FIFOs für Sender und Empfänger werden zwei unabhängige PLB Bussysteme miteinander gekoppelt. Vorarbeiten haben gezeigt, dass der Einsatz der Mailbox einen hohen FPGA Ressourcenbedarf mit sich bringt. Desweiteren liegt die Zugriffszeit bei einer einzelnen „Read“ oder „Write“ Funktion bei durchschnittlich 4 Mikrosekunden [11]. Für das MPSoC mit strengen Echtzeitanforderungen eignet sich der Einsatz der XPS Mailbox nur bedingt.

Im weiteren Entwicklungsverlauf werden die Xilinx spezifische Bussysteme (PLB und FSL) durch das zu AMBA gehörenden „Advanced eXtensible Interface 4 (AXI)“ ersetzt [16]. Basierend auf einer unidirektionalen Streaming Architektur, maximiert AXI den Datendurchsatz durch eine höhere Datenbreite (256 Bit) und eine höhere Taktfrequenz [2]. Streaming Protokolle dienen dem Verbinden von Komponenten, die große Datenmengen mit wenig Latenz und großem Datendurchsatz austauschen. Durch die „Plug & Play“ Wiederverwendbarkeit beim Einsatz von AXI4 können IP Blöcke schnell ausgetauscht werden. Das MPSoC wird dadurch skalierbar und reduziert die Entwicklungskosten [15]. Außerdem ist eine Portierung auf herstellerunabhängige FPGAs möglich.

2.4 Hardware / Software Partitionierung

Zum ersten Einsatz kommt die FPGA-basierte MPSoC Plattform in einem autonomen Transportsystem, dem FAUST Sensor Controlled Vehicle (SCV). Als Versuchsträger für passive und aktive Fahrerassistenzfunktionen besteht die aktuelle SCV Plattform aus zwei GEME 2000 Industrierechner und zwei ARM7 TDMI Mikrocontroller (vgl. Abb. 2.3).

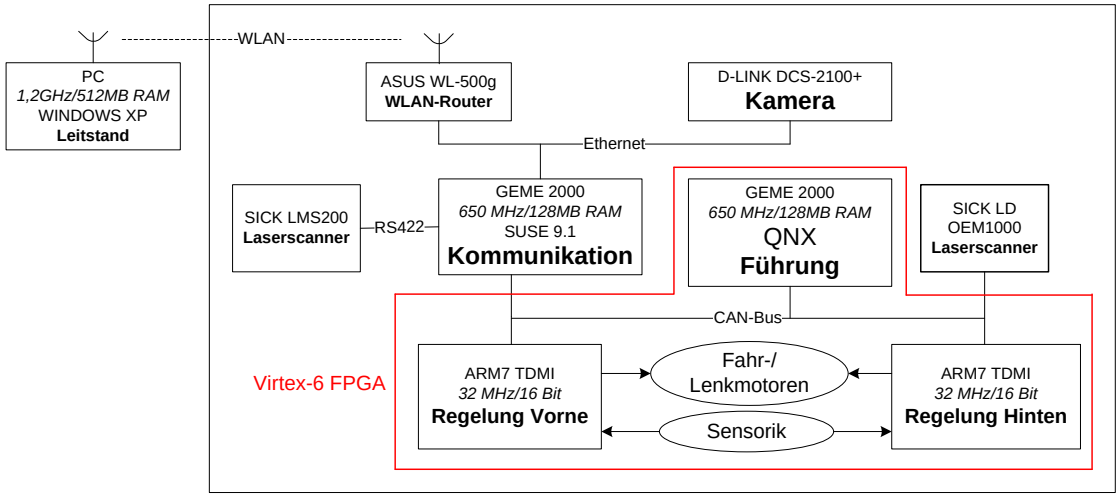

Abbildung 2.3: Faust „Sensor Controlled Vehicle“ Hardware Plattform

Die rot markierten Komponenten aus Abb. 2.3 werden auf einem Virtex-5 FPGA implementiert. Hierbei ist die HW/SW Partitionierung der bestehenden Plattform von wichtiger Bedeutung. Durch die Auslagerung von sequentiell ablaufenden Befehlsfolgen auf verschiedene Prozessoren werden Aufgaben parallel ausgeführt. Der zu ersetzende GEME 2000 Rechner übernimmt die Aufgabe der Fahrzeugsteuerung. Die verschiedenen Aufgaben der Fahrzeugführung, wie Positionsbestimmung, Situationsanalyse und Messwertaufbereitung sind in Threads ausgelagert und laufen auf einem QNX Betriebssystem. Die Mikrocontroller dienen der Geschwindigkeits- und Abstandsregelung.

Die Herausforderung bei der Implementierung des MPSoC besteht in der HW/SW Partitionierung. Die Regelungskomponenten für Geschwindigkeit und Abstand werden komplett in Hardware implementiert. Über Schnittstellensignale werden die Messergebnisse direkt in Register geschrieben und können anschließend von den MicroBlazes gelesen werden. Im Gegensatz zu einer reiner Software Implementierung sind Hardware Beschleuniger (IPs) schneller und bieten einen höheren Datendurchsatz. Unabhängige Aufgaben werden auf verschiedene Prozessoren ausgelagert und reduzieren somit die Ausführungszeit des einzelnen Prozessors. Die bestehende Thread Implementierung des GEME Rechners wird analysiert und gegebenenfalls werden einzelnen Komponenten in Hardware modelliert. Dies eignet sich vor allem bei zeitintensiven und komplexen Algorithmen, wie der Positionsbestimmung (Odometrie). Hierbei werden einzelne Teilaufgaben durch eine Vorverarbeitung in Hardware ersetzt. Die MicroBlazes besitzen einen gemeinsamen Speicher und kommunizieren zusätzlich über eine direkte FSL Verdrahtung. Für die ereignisgesteuerten und zeitgesteuerten Teilfunktionen sind geeignete SW-Implementierungen zu entwickeln.

Das auf dem GEME Rechner implementierte Echtzeitbetriebssystem QNX ist nicht kompatibel mit den MicroBlaze Softcores. Aus diesem Grund muss ein vergleichbares und POSIX fähiges RTOS auf den MicroBlazes des Virtex-5 FPGA implementiert werden.

2.4.1 RTOS Implementierung

Für die Einhaltung von Zeitbedingungen und die Vorhersagbarkeit des Prozessverhaltens wird auf beiden MicroBlazes ein RTOS implementiert. Das Ziel für das asymmetrische Multiporzessorsystem ist eine Implementierung von zwei verschiedenen Echtzeitbetriebssysteme. Hierfür wurde in Vorarbeiten erste Erfahrungen mit der Implementierung eines RTOS auf einem FPGA gesammelt. Für den MicroBlaze Softcore stehen verschiedene RTOS Alternativen zur Verfügung (vgl. Tabelle 2.4.1):

Name	**Features**
MicroC/OS-II	POSIX Interface, AMP, Kostenlos, Preemptive, Kernel > 5K Bytes
Nucleus RTOS	Lizenzpflichtig, AMP, POSIX Interface, Kernel > 13K Bytes
Xilinx XilKernel	Open Source, AMP, POSIX Interface, Kernel > 7K Bytes
QNX Neutrino	nur PowerPC kompatibel, SMP, lizenzpflichtig
FreeRTOS	Open Source, AMP, Preemptive, Kernel < 9K Bytes

Tabelle 2.1: Übersicht der Echtzeitbetriebssysteme (RTOS) [18]

Da das RTOS für die MPSoC Implementierung sowohl kostenlos als auch das POSIX Interface unterstützen soll, kommt lediglich das Micrium MicroC/OS-II und der Xilinx Xilkernel in Frage. Das MicroC/OS-II wird die Aufgabe der Fahrzeugsteuerung übernehmen und dient als Master Prozessor. Der Xilkernel kommuniziert mit den Hardwaremodulen für die Regelungen und wertet die Messergebnisse aus. Über den gemeinsamen DDR2 Speicher haben beide Prozessoren Zugriff auf die jeweiligen Daten. Eine bidirektionale FSL Bus Implementierung wird für die schnelle Kommunikation verwendet. Hierbei ist im Gegensatz zum gemeinsamen Speicher keine Synchronisation notwendig. Dies beschleunigt die Ausführungszeit der Prozessoren. Beide RTOS sind preemptive und unterstützen das POSIX Interface. Durch ein prioritätenbasierter Scheduler werden die POSIX Threads abgearbeitet. Hierbei ist zu beachten, dass die Anzahl der nebenläufigen Threads möglichst gering gehalten wird. Mit steigender Anzahl an Threads steigt auch der Synchronisationsaufwand und die Gefahr von Deadlocks. Um dieses Risiko zu minimieren sollen die Aufgaben weitestgehend eigenständig bearbeitet werden und die Prozessoren nur die Ergebnisse untereinander austauschen.

Das weiterführende Ziel der MPSoC Projektarbeit ist die Implementierung einer „Symmetrischen Architektur“. Hierbei wird lediglich ein RTOS auf dem Master Prozessor implementiert. Dieser teilt die Aufgaben auf die zur Verfügung stehenden Prozessoren auf. Aufgrund des fehlenden parallelisierenden Compilers für FPGA-basierte Systeme ist die SMP Technologie noch weitestgehend unerforscht. Erste Ansätze zu einem FPGA-basierten SMP System sind [8]. Hierbei wurde der Kernel des Xilinx Xilkernels modifiziert.

3 Risiken & Herausforderungen

Bei der MPSoC Entwicklung für die Masterarbeit ist das übergeordnete Risiko der zeitliche Aspekt. Bis Ende des Jahres muss das vollständige Verständnis über die FPGA-basierte Multiprozessortechnologie vorliegen. Hierbei ist das Problem, das die Anwendung und Nutzung der neuen Technologien nur unvollständig dokumentiert ist. Auf Seiten des FPGA Herstellers „Xilinx“ existieren nur sehr wenige „Application Notes“, die die Implementierung eines Multi-MicroBlaze Systems erläutern. Durch MPSoC Prototypen müssen Erfahrungen, bezüglich des Timing-Verhaltens, der Durchsatzsteigerung und des Ressourcenbedarfs, gesammelt werden.

Ein stetiges Risiko bei FPGA Implementierungen ist der Ressourcenbedarf. Vor allem die Verfügbarkeit von BRAMs ist bei komplexen Systemen auschlaggebend. Durch die begrenzten Hardware Ressourcen ist die Hardware/Software Partitionierung von wichtiger Bedeutung. Für die Masterarbeit steht ein Xilinx ML505 LXT Board mit einem Virtex-5 XC5VLX50T-1FF1136C FPGA zur Verfügung [17]. Dieser FPGA verfügt über 7200 Slices[1] und 60 BRAMs. Im Vergleich zum Spartan 3E 1200 FPGA, welcher auf dem HPEC Fahrzeug eingesetzt wird und bei der Hälfte von Slices nur 14 BRAMs besitzt, sind die verfügbaren Ressourcen des Virtex-5 deutlich höher. Für den Einsatz als Fahrzeugsteuerungsplattform muss das MPSoC portiert werden. Hierbei muss der Ressourcenbedarf jeder Komponente verringert werden. Ein MicroBlaze Softcore verbraucht 6 BRAM Blöcke. Da beim Spartan 3E nur 14 BRAM Blöcke zur Verfügung stehen, kann zur Durchsatzsteigerung die Prozessoranzahl nicht erhöht werden.

Weitere Risiken werden als Herausforderung gesehen und können durch geeignete Implementierungsstrategien gelöst werden. Hierzu zählen die folgenden Herausforderungen:

- Timing-Eigenschaften für Echtzeitsysteme: Durch die Interprozesskommunikation und den Synchronisationsmechanismus entsteht eine Verzögerung der Ausführungszeit. Diese kann durch die Auslagerung von parallel ablaufenden Aufgaben kompensiert werden. Hierbei muss der Synchronisationsaufwand durch Trennung von unabhängigen Aufgaben minimiert werden. Die Prozessoren dürfen sich nicht gegenseitig behindern indem sie auf Ergebnisse oder Zugriffe warten müssen. Eine genaue Deadlock Analyse muss erfolgen.
- Shared-Memory: Der Flaschenhals, der beim Zugriff auf den externen Speicher über den MPMC entsteht, muss durch geeignete Speicherorganisation verringert werden. Außerdem muss die Kohärenz der Caches sichergestellt werden.
- SW-Parallelisierung: Bei der Analyse der FAUST SCV Steuersoftware können Abhängigkeiten erkannt werden, die eine Parallelisierung behindern. Daten-Abhängigkeiten können entstehen, wenn eine Berechnung auf vorherige Ergebnisse aufbaut.

[1] Virtex-5 Slice: bestehend aus 4 LUTs und 4 Flip-Flops

4 Zusammenfassung & Ausblick

Der Einsatz von Multiprozessoren beim Design von Eingebetteten System-on-Chips wird in den kommenden Jahren von reger Interesse sein. Durch den Einsatz von FPGA basierten „Multiprocessor System-on-Chips (MPSoC)“ wird ein erhöhter Systemdurchsatz erreicht, indem verschiedene Programme parallel auf mehreren Prozessoren ausgeführt werden. Die FPGA Technologie nutzt durch die getrennte Konfiguration der Hardwarebeschleunigermodule eine Parallelität auf Hardwareebene und bietet dadurch eine hohe Flexibilität. Bei asymmetrischen und speichergekoppelten Systemen greifen alle Prozessoren auf einen gemeinsamen Speicher zu. Wie in Kap. 2.2 beschrieben, ist hierfür eine Synchronisation der Zugriffe erforderlich. Ein Mutex koordiniert die Speicherzugriffe und gewährleistet den wechselseitigen Ausschluss der Prozessoren. Der Synchronisationsaufwand muss durch eine geeignete Speicherorganisiation minimal gehalten werden. Gerade bei Echtzeitsystemen ist ein vorhersehbares Timing-Verhalten unabdingbar. Zur Kommunikation zwischen den Prozessoren wird sowohl eine indirekte Kommunikation über den gemeinsamen Speicher als auch eine direkte Kommunikation über den FSL oder AXI Busse eingesetzt.

Im Rahmen der Masterarbeit werden die Grundlagen der FPGA-basierten Multiprozessortechnologie vertieft. Ein asymmetrisches Dual-MicroBlaze MPSoC wird für den Einsatz in einem autonomen Fahrzeug entwickelt. Die Steuer- und Fahrzeugplattform des FAUST „Sensor Controlled Vehicle“ soll auf ein Virtex-5 FPGA portiert werden. Hierfür werden zwei verschiedene „Real-Time Operating Systems (RTOS)“ auf den MicroBlazes implementiert. Die aktuelle Steuerungssoftware läuft unter einem QNX Betriebssystem auf zwei GEME-2000 Rechnern. Durch die Inkompatibilität von QNX zu den MicroBlaze Softcores muss ein alternatives Echtzeitbetriebssystem implementiert werden. Nach ersten Analysen kommt aufgrund des POSIX Interfaces lediglich das Micrium MicroC/OS-II und der Xilinx Xilkernel in Frage. Die HW/SW Partitionierung ist ausschlaggebend für die Durchsatzsteigerung des Gesamtsystems. Die SCV Software wird durch eine Abhängigkeitsanalyse auf Parallelisierbarkeit untersucht. Anschließend werden die Aufgaben auf die asymmetrischen Prozessoren verteilt. Diese bearbeiten die Aufgaben völlig unabhängig voneinander und tauschen die Ergebnisse über eine Interprozesskommunikation aus.

In zukünftigen Projekten soll das MPSoC mit einem chipinternen Verbindungsnetzwerk, dem sogenannten „Network-on-Chip (NoC)“, erweitert werden [7]. Eine solche Architektur ersetzt die dedizierten Verbindungen und Bussysteme. Sie besteht aus Netzwerk-Interfaces und Routern, die einen Routingalgorithmus zum Weiterleiten der Pakete an die entsprechenden Empfänger IPs implementiert haben. NoC basierte FPGA Systeme bieten gegenüber Singlebussysteme durch die parallele Transaktionsabarbeitung eine effizientere Kommunikation. Gerade bei Multimedia- und Echtzeitanwendungen ist die schnelle und parallele Bearbeitung von Instruktionen wichtig. Abb. 4.1 zeigt die Erweiterung des MPSoCs mit einem NoC.

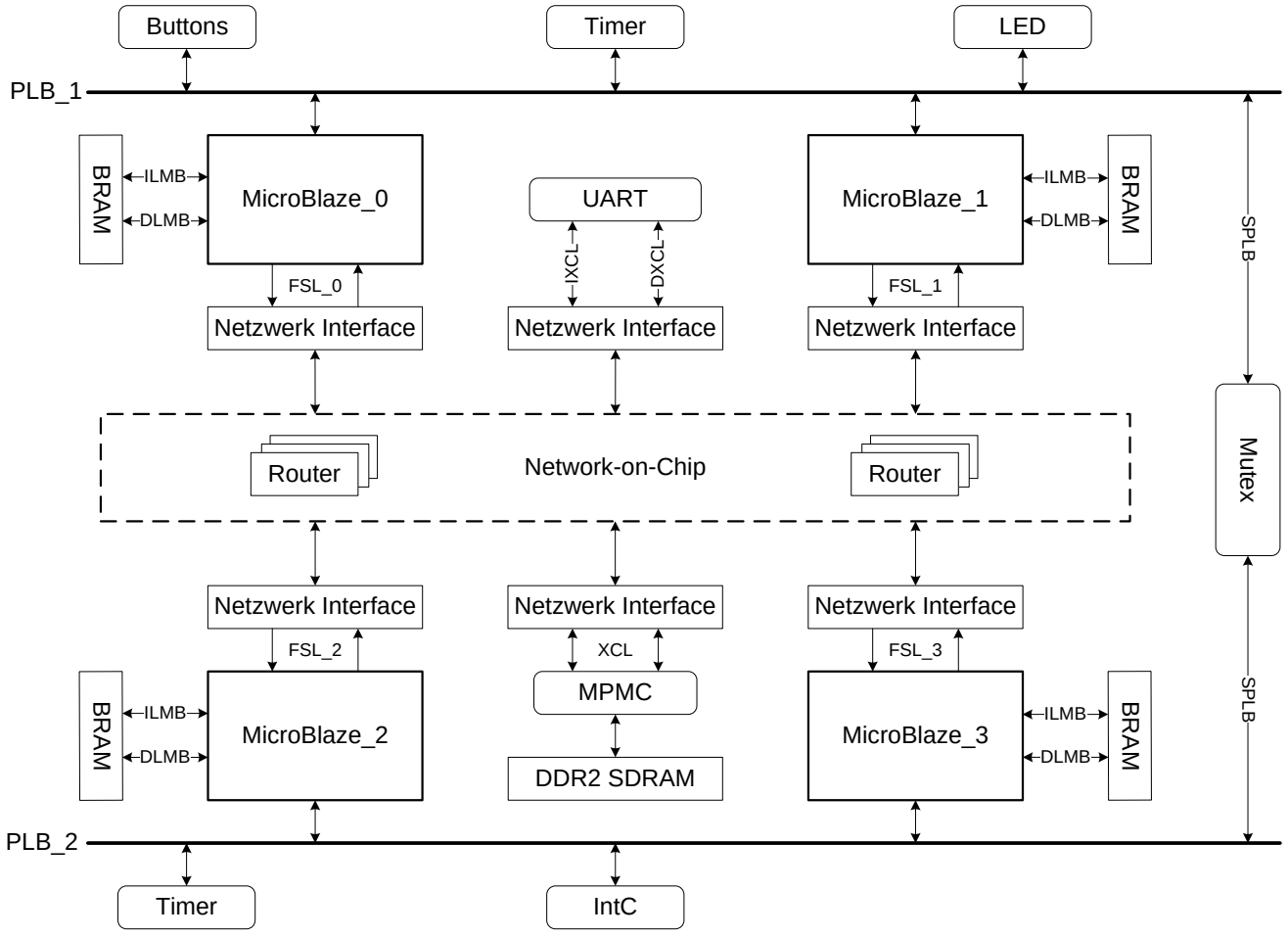

Abbildung 4.1: MPSoC mit „Network-on-Chip“ als Verbindungsnetzwerk

Literaturverzeichnis

[1] AMDAHL, Gene M.: Validity of the single processor approach to achieving large scale computing capabilities. In: *AFIPS spring joint computer conference* (1967), Nr. AFIPS Conferences 30

[2] ARM: AMBA AXI Protocol. In: *Specification* (2010), Nr. Version 2.0

[3] G.BENGEL ; C.BAUN ; M.KUNZE ; K.U.STUCKY: *Masterkurs Parallele und Verteilte Systeme*. Vieweg+Teubner, 2008. – ISBN 978-3-8348-0394-8

[4] GOVE, Darryl: *Multicore Application Programming: for Windows, Linux, and Oracle Solaris (Developers's Library)*. Addison-Wesley Professional, 2011. – ISBN 978-0-3217-1137-3

[5] JERRAYA, Ahmed A. ; WOLF, Wayne: *Multiprocessor Systems-on-Chips*. Morgan Kaufmann, 2005. – ISBN 978-0-123-85251-9

[6] MAHR, Thomas ; GESSLER, Ralf: *Hardware-Software-Codesign: Entwicklung flexibler Mikroprozessor FPGA-Hochleistungssysteme*. Vieweg+Teubner, 2007. – ISBN 978-3-83-480048-0

[7] MICHELI, Giovanni de ; BENINI, Luca: *Networks on Chips: Technology and Tools*. Morgan Kaufmann, 2006. – ISBN 978-0-12-370521-1

[8] P.HUERTA ; J.CASTILLO ; C.SANCHEZ ; J.I.MARTINEZ: Operating System for Symmetric Multiprocessors on FPGA. In: *Reconfigurable Computing and FPGAs - ReConFig 2008* (2008), Nr. IEEE Conferences

[9] RAUBER, Thomas ; RÜNGER, Gudula: *Multicore: Parallele Programmierung*. Springer Verlag, 2008. – ISBN 978-3-540-73113-9

[10] T.DORTA ; J.JIMENEZ ; J.L.MARTIN ; U.BIDARTE ; A.ASTARLOA: Overview of FPGA-Based Multiprocessor Systems. In: *2009 International Conference on Reconfigurable Computing and FPGAs* (2009), Nr. IEEE Conferences

[11] WILKEN, Heiko: Network-on-Chip basierte Multiprozessor-SoC-Plattform. In: *Projektbericht 1* (2010), Nr. HAW Hamburg

[12] XILINX: Designing Multiprocessor Systems in Platform Studio. In: *White Paper: Xilinx Platform Studio* (2007), Nr. WP262(v2.0)

[13] XILINX: Fast Simplex Link (FSL). In: *Xilinx Product Specification* (2008), Nr. 2.11a

[14] XILINX: XPS Block RAM (BRAM) Interface Controller. In: *Product Specification* (2009), Nr. DS596 (v1.00b)

[15] XILINX: AXI4 Interconnect Paves the Way to Plug-and-Play IP. In: *White Paper* (2010), Nr. WP379 (v1.0)

[16] XILINX: Unlock New Levels of Productivity for Your Design Using ISE Design Suite 12. In: *White Paper: Virtex-6 and Spartan-6 FPGAs* (2010), Nr. WP368 (v1.0)

[17] XILINX: *Virtex-5 LXT FPGA ML505 Evaluation Platform*. 2010. – URL http://www.xilinx.com/products/devkits/HW-V5-ML505-UNI-G.htm. – abgerufen 15.10.2010

[18] XILINX: *Third Party Real Time Operating Systems (RTOS) Support*. 2011. – URL http://www.xilinx.com/ise/embedded/epartners/listing.htm#RTOS. – abgerufen 13.02.2011

Abbildungsverzeichnis

1.1 FAUST HPEC Fahrzeug 5
1.2 Sensor Controlled Vehicle 5

2.1 Dual MicroBlaze MPSoC mit Shared Memory Architektur und Interprozesskommunikation [12] 6
2.2 Software Memory Mapping eines Dual MicroBlazes [12] 7
2.3 Faust „Sensor Controlled Vehicle“ Hardware Plattform 10

4.1 MPSoC mit „Network-on-Chip“ als Verbindungsnetzwerk 14