Multiprozessor System on Chip

Heiko Bordasch

Anwendungen 1 - Ausarbeitung

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Multiprozessor System on Chip

Heiko Bordasch

Ausarbeitung eingereicht im Rahmen von Anwendungen 1
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 26. Februar 2010

Inhaltsverzeichnis

1 Einleitung

Schon 1965 sagte Gordon Earle Moore, dass sich die Anzahl der Transistoren auf einem Chip alle zwei Jahre verdoppelt [17]. Jedoch erfordert der stetig steigende Bedarf an Rechenleistung ein Umdenken im Systementwurf. Die Erhöhung der Taktrate bei Singlecores zur Erzielung von Leistungssteigerungen ist aufgrund von Stromverbrauch und Wärmeabgabe nicht realisierbar. Eine weitere Problematik liegt darin, dass die Speicherzugriffszeiten nicht im gleichen Umfang wie die CPU Geschwindigkeiten wachsen und somit ohne eine gute Cache Speicherstrategie ein Flaschenhals entstehen kann [19]. Abb. 1.1 zeigt die heutige Lage, nämlich eine gleichbleibende Taktfrequenz trotz steigender Transisitoranzahl (Mooresches Gesetz). Um den leistungsbezogenen Anforderungen der Anwendungen gerecht zu werden, werden Multiprozessorsysteme eingesetzt.

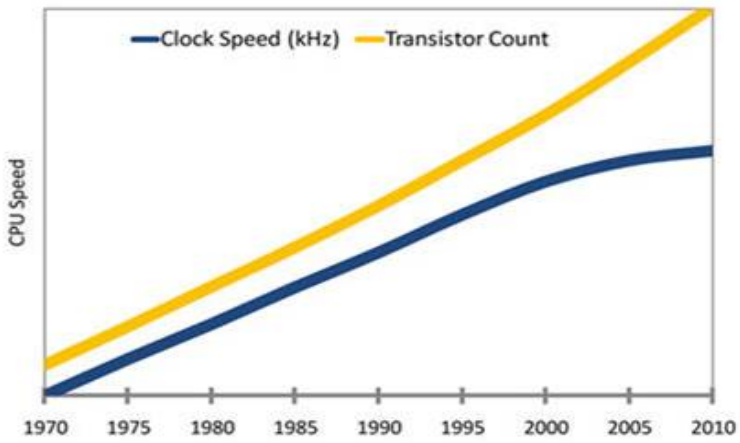

Abbildung 1.1: Mooresches Gesetz im Vergleich zu heutigen Taktfrequenzen [13]

Ziel dieser Ausarbeitung ist eine Themenfindung als Schwerpunktsdefinition für den Masterstudiengang. Multiprozessor System on Chip (MPSoC) Systeme werden vor allem bei Echtzeitanwendungen mit hohen Datenraten benötigt, wie zum Beispiel bei der Audio- oder Videoverarbeitung. Aktuelle Forschungsergebnisse, wie beispielsweise die des Interuniversity Microelectronics Centre (IMEC) in Belgien (vgl. Kap. 4), zeigen, dass der Trend bei Embedded Systemen in Richtung MPSoC Systemen geht [21]. Auch in der Automobilbranche, in der sich in den letzten Jahren die Technologie bei Fahrerassistenzsystemen und Infotainment Systeme stetig weiter entwickelt hat und wo hohe Leistung auf geringer Platzdichte benötigt wird, ist ein Einsatz von MPSoC Plattformen vorstellbar. In den kommenden Semestern soll das Verständis der Multiprozessor Technologie vertieft werden und eine FPGA basierte MPSoC Plattform im FAUST Projekt „High Performance Embedded Computing (HPEC)“ entwickelt und implementiert werden.

1.1 FAUST High Performance Embedded Computing

"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. Für die Entwicklung und die Implementierung werden Spartan3 oder Virtex5 FPGAs eingesetzt. Diese ermöglichen eine Integration sowohl von Software Lösungen mit Hilfe des MicroBlazes als auch von in Hardware implementierten Algorithmen. Gerade für die Echtzeit-Signalverarbeitung, die z.B. in der digitalen Bildverarbeitung zur Fahrspurführung eingesetzt wird, eignet sich die Modellierung der Algorithmen für eine Hardware Implementierung [22]. Die parallelen Arithmetik Ressourcen eines FPGAs können den von der Kamera gelieferten Datenstrom auch bei erheblich niedrigeren Taktfrequenzen beherrschbar machen. Die jetzige SoC Plattform besteht aus einer Vielzahl von Intellectual Properties (IPs), die über ein Bussystem (PLB) mit einem Softcore MicroBlaze Prozessor verbunden sind (vgl. Abb. 1.2).

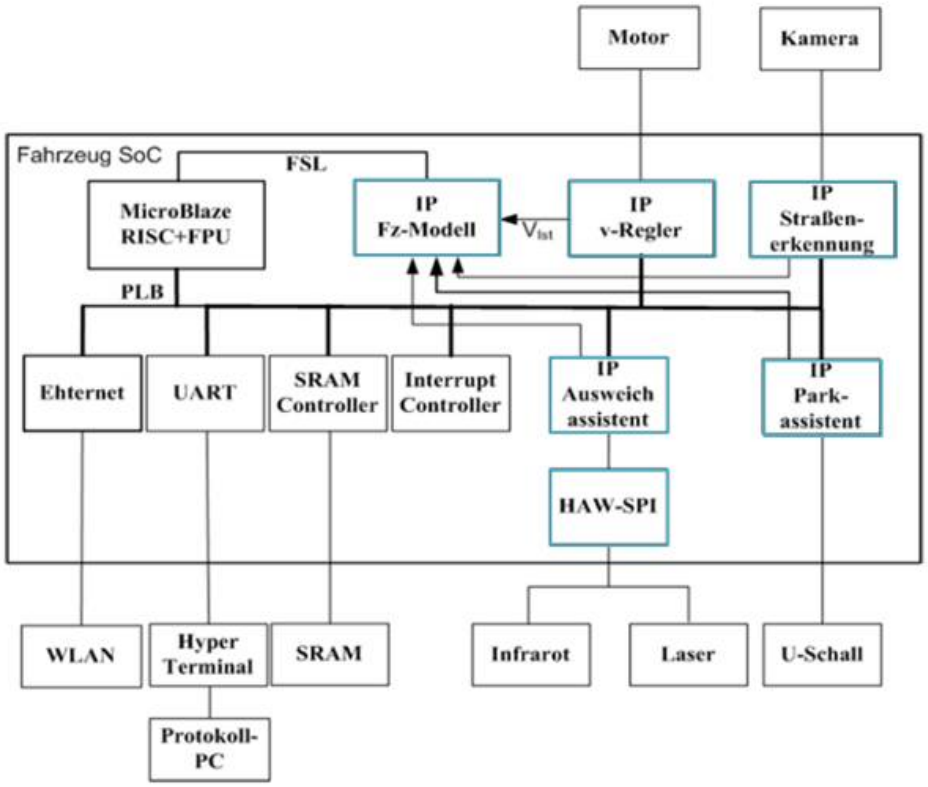

Abbildung 1.2: SoC basierte Fahrzeugsteuerungsplattform

Da es sowohl Realtime IPs, die Sensordaten erfassen und die Fahrspur erkennen, als auch Non-Realtime IPs für Debug- und Kontrollfunktionen gibt, soll die SoC Plattform durch eine MPSoC Plattform ersetzt werden. Hierbei wird ein zusätzliches Bussystem mit einem eigenen MicroBlaze implementiert, welches ausschließlich für die relevanten Steuerungs- und Führungsfunktionen verantwortlich ist. Die so entstandene Trennung von Realtime und Non-Realtime Funktionen führt zu einer Erhöhung der parallelen Instruktionsabarbeitung und somit zu einer Leistungssteigerung. Durch den Einsatz von spezifisch ange-

passten MicroBlaze Varianten mit wahlweisen Floating Point oder Barrel Shifter Komponenten kann zusätzlich eine Verringerung des Energiebedarfs erfolgen (vgl. Kapitel 2). Mit dem Einsatz der MPSoC Technologie für Embedded Systems in Fahrzeugen für den Carolo Cup Wettbewerb, erwarten wir deshalb entscheidende Vorteile bei der zeitlichen Reichweite des Fahrzeuges.
In den folgenden Semestern wird eine MPSoC Architektur für das HPEC Fahrzeug entworfen und implementiert. Die Anzahl der benötigten MicroBlazes und die Partitionierung der einzelnen IPs ist hierbei eine grundlegende Aufgabe. Die mögliche Verringerung von Laufzeiten durch eine Parallelisierung sind durch verschiedene Faktoren, wie der Anzahl der eingesetzen Prozessoren oder den auszuführenden Algorithmen, begrenzt [19].

1.2 Gesetz von Amdahl und Gustafson

Ein Programm kann nie vollständig parallel ausgeführt werden, da einige Teile, wie Intitalisierung und Speicher Allokation stets sequentiell ausgeführt werden müssen. Der Effekt von sequentiellen Programmteilen auf den erreichbaren Speedup S wird durch das Amdahlsches Gesetz erfasst [2]. Sei p der parallele Anteil eines Programms, dann ist $1-p$ der sequentielle Teil des Programms. Bei n eingesetzten Prozessoren ist p/n der beschleunigte parallele Anteil. Der maximale Speedup lässt sich dann wie folgt ausdrücken:

$$S = \frac{1}{(1-p) + \frac{p}{n}} \leq \frac{1}{1-p} \tag{1.1}$$

Aus dem Amdahlschen Gesetz geht hervor, dass bereits ein relativ kleiner, nicht parallelisierbarer Anteil den maximal erreichbaren Speedup begrenzt. Nach Amdahl sind parallele Berechnungen nur für eine kleine Anzahl von Prozessoren sinnvoll [2]. Bei der Ausführung einen Programmes geht Amdahl von einer konstanten Problemgröße aus, die auf die zur Verfügung stehenden Prozessoren verteilt wird. Jedoch hat die Realität bewiesen, dass die Problemgröße mit der Anzahl der Prozessoren wächst [10]. John L. Gustafson veränderte das Amdahlsche Gesetz, indem er für skalierbare Systeme annimmt, dass der parallele Anteil mit der Anzahl der Prozessoren wächst. Außerdem behauptet er, dass der sequentielle Anteil weitgehend unabhängig von der Problemgröße ist und sich bei steigender Prozessoranzahl gegenüber dem parallelen Anteil verringert [9]. Das Gesetz von Gustafson lautet wie folgt:

$$S = (1-p) + n * p \tag{1.2}$$

Der Unterschied zu Amdahl ist, dass der parallele Anteil mit der Anzahl der Prozessoren n wächst und der sequentielle Anteil gesenkt wird. Es gibt bei der Parallelisierung von Programmen eine Reihe von Problematiken, die sich nicht beliebig vergrößern lassen oder die in möglichst kurzer Zeit berechnet werden müssen, wie beispielsweise bei Echtzeitsystemen. Gerade die Skalierbarbeit von Anwendungen spielt hier eine bedeutsame Rolle. Eine Anwendung ist skalierbar, wenn die steigende Problemgröße durch die wachsende Anzahl der Prozessoren so kompensiert werden kann, dass die Zeit zur Lösung des Problems konstant gehalten werden kann. Bei nicht skalierbaren Anwendungen gilt das Amdahlsche Gesetz. Ist aber ein Programm skalierbar, dann gilt das Gesetz von Gustafson [9].

2 Grundlagen von Multiprozessor System on Chip

Nach der Flynnschen Klassifikation gehören Multiprozessoren zu den MIMD Architekturen (Multiple Instruction, Multiple Data) [8]. Sie führen gleichzeitig verschiedene Operationen auf verschiedene Eingangsdatenströme durch, wobei die Aufgaben auf die zur Verfügung stehenden Prozessoren verteilt werden. Herkömmlichen Singlecores verfügen nur über ein Prozessorelement mit Low Level Kommunikation. Der Entwurf eines MPSoC Designs hingegen ist um einiges komplexer. Damit ein energiesparendes und leistungsstarkes System entwickelt werden kann, müssen sowohl die Anzahl und die Topologie der Prozessoren als auch die Interprozesskommunikation und das Speichermanagement untersucht und aufeinander abgestimmt sein [14]. Multiprozessoren lassen sich in zwei Arten aufteilen:

- Homogene Multiprozessoren: Diese sind meist General Purpose Systemem, bei denen alle CPUs identisch sind. Da bei der Erhöhung der Prozessoranzahl die zugrunde liegende Architektur nicht verändert werden muss, sind die Entwicklung des Systems und die Softwareimplementierung einfach.
- Heterogene Multiprozessoren: Diese sind spezifisch auf eine Aufgabe angepasste Prozessoren mit guter Performance und weniger Speicherbedarf. Aufgrund der Spezialisierung und des begrenzten Platzbedarfs auf einem FPGA, sind sie in einem MPSoC System vorteilhafter [21].

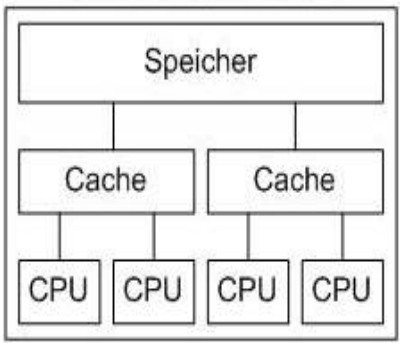

Abbildung 2.1: Hierarchisches Design

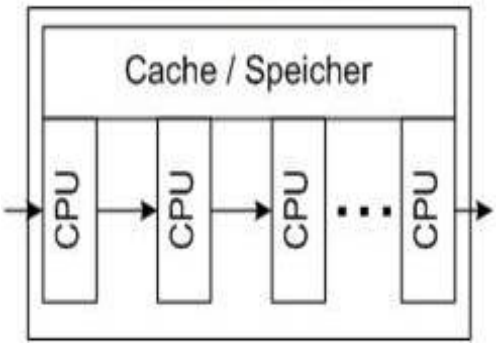

Abbildung 2.2: Pipeline Design

Bei dem Design von MPSoC Systemen gibt es verschiedene Implementierungsvarianten, die sich in der Anzahl der Prozessoren, der Größe und der Anordnung des Speichers, dem Aufbau und dem Zugriff von Caches und dem Einsatz von heterogenen Komponenten unterscheiden. Eine Multiprozessor Architektur beschreibt den Aufbau und die Verbindung der einzelnen Komponenten. Man unterschiedet zwischen folgenden Architekturen [19]:

- Hierarchisches Design: Mehrere Prozessoren teilen sich in einer baumartigen Struktur mehrere Caches (vgl. Abb. 2.1). Die Wurzel der Baumstruktur repräsentiert den Hauptspeicher, auf den alle gleichberechtigt zugreifen können. Ein oder mehrere Master Prozessoren steuern das Verhalten der Slave Prozessoren.
- Pipeline Design: Eingehende Daten werden von mehreren Prozessoren, die jeweils eine spezifische Aufgabe haben, schrittweise verarbeitet (vgl. Abb. 2.2). Der letzte Prozessor legt das Ergebniss im Hauptspeicher ab. Solche Architekturen sind vor allem für Streaming Applikationen vorteilhaft [21].
- Busbasiertes Design: Alle Prozessoren und alle beteiligten Komponenten (Speicher und Caches) werden über ein Bussystem miteinander verbunden. Der komplette Datentransfer läuft über das Verbindungsnetzwerk (bei Xilinx FPGAs meist der PLB Bus). Busbasierte Designs sind vor allem bei dem Einsatz von einer Vielzahl von Prozessoren sinnvoll.

Bei einer symetrischen Multiprozessor Architektur (Symmetric Multiprocessing SMP) verarbeiten die Prozessoren unabhängig voneinander Instruktionen auf jeweils eigenen Dateneingangsströmen [1]. Jede CPU ist funktional identisch und hat keine direkte Verbindung zu anderen Prozessoren (keine Master/Slave Hierarchie). Alle Komponenten, sowohl Prozessoren als auch Speicher und I/O Komponenten, sind an ein Bussystem angeschlossen. Sie teilen sich einen gemeinsamen Adressraum und haben gleichberechtigt Zugriff auf das I/O Subsystem (vgl. Abb. 2.3).

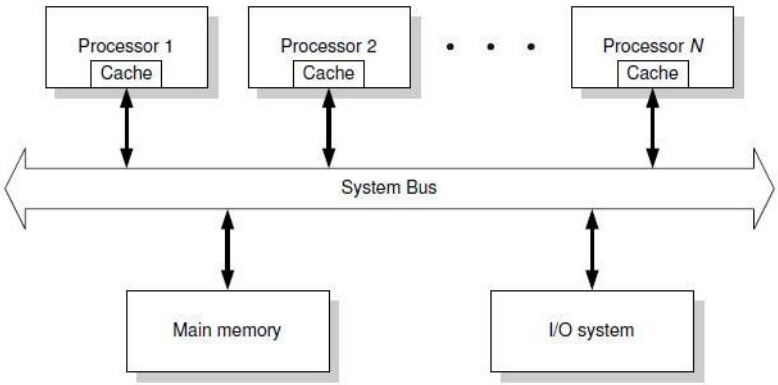

Abbildung 2.3: Aufbau einer typischen SMP Architektur mit gemeinsamen Speicher [1]

In den meisten Fällen bestimmt die Zielapplikation eines FPGA basierten Multiprozessorsystems die Architektur des MPSoC Systems. Die Anzahl der zur Verfügung stehenden Logik Ressourcen eines FPGAs, besonders die Größe des Speichers, beschränkt den Entwurf der Architektur. Da der Spartan3e FPGA der HPEC Plattform nur 63 Kilobyte (28 BRAMs) an Speicher zur Verfügung stellt, ist aufgrund der Speicherauslastung des Gesamtsystems eine Shared Memory Architektur zu bevorzugen [24][21]. FPGA basierte MicroBlazes unterstützen keine atomaren Operationen, die den exklusiven Zugriff auf gemeinsamen Speicher erlauben. Aus diesem Grund wird die Synchronisation meistens mit einem in Hardware implementierten Mutex IP Core gewährleistet (vgl. Kapitel 3.1).

2.1 Kommunikationsmechanismen

Eine wesentliche Herausforderung bei der Integration von Multiprozessorsystemen auf einem Chip ist die Kommunikation, die sogenannte Interprozesskommunikation (IPC), zwischen den Prozessoren. Damit keine unnötigen Delays entstehen und eine hohe Bandbreite beim Datentransfer ermöglicht werden kann, sollten die Signallaufzeiten so minimal wie möglich sein. Bei der Multiprozessor Kommunikation unterscheidet man zwischen:

- Indirekte Kommunikation: Datenaustausch findet entweder über einen gemeinsamen synchronisierten Speicher (Shared Memory) statt oder über einen DMA Transfer zwischen zwei lokalen Speichern. Bei dem Einsatz eines geteilten Bussystems entsteht der Nachteil, dass nur ein Prozessor zur gleichen Zeit senden bzw. empfangen kann.
- Direkte Kommunikation: Prozessoren kommunizieren direkt über expliziten Nachrichtenversand. Bei Xilinx FPGAs kann die Kommunikation über den Fast Simplex Link (FSL) erfolgen. Da der Kommunikationskanal nicht geteilt werden muss, hat die direkte Kommunikation den Vorteil, dass eine hohe Bandbreite genutzt werden kann [21].

Die Speicherorganisation in einem MPSoC Design regelt den Zugriff der Prozessoren auf die Daten. Bei verteilten Speichersystemen hat jeder Prozessor einen fest zugeordneten lokalen bzw. privaten Speicher, auf den nur er zugreifen kann. Bei dem Zugriff auf einen fremden lokalen Speicher eines anderen Prozessors, müssen IPC-Kommunikationsmechanismen wie Message Passing eingesetzt werden. Multiprozessoren mit gemeinsamem Speicher haben über ein Verbindungsnetzwerk (z.B. Xilinx PLB Bus) Zugriff auf eine global adressierbare Speicherressource. Damit Prozessoren nicht gleichzeitig auf die gleichen Daten zugreifen können, müssen diese durch IPC-Mechanismen geschützt werden. Für IPC-Mechanismen gibt es im wesentlichen drei Anwendungsgebiete: Den Datenaustausch zwischen Prozessoren, die Synchronisation und den gegenseitige Ausschluss bei gemeinsamen Ressourcen durch einen Mutex oder einen Semaphor [19].

Da die Latenz bei dem Zugriff auf den gemeinsamen Speicher höher als die Latenz auf lokalen On-Chip Speicher ist, werden Cache-Speicher zur Leistungssteigerung und zur Latenzverringerung eingesetzt [14][7]. Diese Caches können in verschiedener Art und Weise aufgebaut werden. Bei der meist genutzten Cache Architektur teilen sich die eng gekoppelten Prozessoren einen gemeinsamen L2-Cache. Zusätzlich hat jeder Prozessor noch einen privaten L1-Cache, der über den L2-Cache auf den Hauptspeicher zugreifen kann. Dadurch kann jede Hauptspeicherinformation innerhalb des Gesamtsystems mehrfach repliziert sein. Bei verteilten Caches muss aufgrund dessen die Cache Kohärenz sicher gestellt werden. Die einzelnen Caches müssen für die gleiche Speicheradresse die gleichen Daten zurückliefern. Caches besitzen hardwareseitig garantierte Kohärenz. Jedoch bietet Xilinx bei Multiprozessoren keine Unterstützung für Kohärenz Protokolle. Hier muss der Entwickler softwareseitig dafür sorgen, dass die Caches mit bestimmten Protokollen, wie beispielsweise dem „Snoopy Protocol“[16], synchron gehalten werden [4].

2.2 Topologien von Multiprozessoren

Die Topologie eines MPSoC Systems beschreibt die physikalische Struktur der Multiprozessoren und definiert die Kommunikationsarchitektur. Die einfachste Variante stellt ein geteiltes Bussystem dar, bei dem sich alle Kommunikationspartner einen Bus teilen und ein Arbiter durch ein Busprotokoll (Prioritäten- oder Zeitbasiert - TDMA) die Zuteilung übernimmt (vgl. Abb. 2.4). Bei einem hierarchischen Bussystem werden mehrere geteilte Bussysteme über Bridges miteinander verbunden, um so die Busauslastung zu reduzieren. Eine Ringstruktur, wie in Abb. 2.5, hat den Vorteil, dass die Punkt zu Punkt Verbindung zwischen den Prozessoren von kürzester Länge ist. Das System kann so höhere Taktraten unterstützen und lässt keine Kollisionen zu [14]. Die komplexeste Topologieart ist die der „Packet Switched Fabrics“. Unter diese Rubrik fällt sowohl die Crossbar Switch (vgl. Abb. 2.6) als auch das Mesh Netzwerk in Abb. 2.7. Beide benutzen Switching Mechanismen, um verschiedene Kommunikationspartner miteinander zu verbinden.

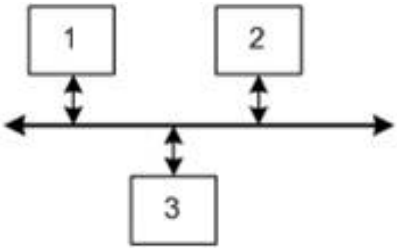

Abbildung 2.4: Bus Topologie

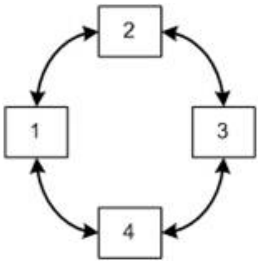

Abbildung 2.5: Ring Topologie

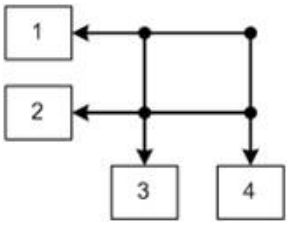

Abbildung 2.6: Crossbar Switch

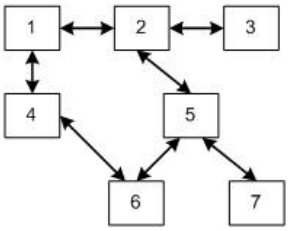

Abbildung 2.7: Mesh Topologie

Singlebus MPSoC Systeme können nur wenige Prozessoren sinnvoll miteinander verbinden. Für jeden Speicherzugriff wird der gemeinsame Bus beansprucht. Somit können keine garantierte Zugriffszeit und keine hohe Bandbreite geboten werden. Nebenläufige Transaktionen von verschiedenen Busteilnehmern sind nicht möglich. Bei netzwerkbasierten Systemen wird das Netzwerk nur zur Kommunikation zwischen den Prozessoren beansprucht. Zukünftige Technologien, wie das „Network on Chip (NoC)“, verbessern die Skalierbarkeit und die Energie Effizienz von SoC Systemen (vgl. Kapitel 2.3), indem bekannte Netzwerktechnolgien in das Chipdesign integriert werden. So ist die Möglichkeit gegeben, dass Transaktionen zwischen mehreren Komponenten parallel stattfinden können. Gerade bei Echtzeitsystemen, die einen hohen Grad an Parallelität fordern, ist dies eine wichtige Eigenschaft [15].

2.3 Network On Chip

Network on Chip (NoC) Architekturen bestehen aus einer Anzahl von Routern, die alle Systemressourcen (Prozessoren, Speicher und IPs) miteinander zu einem Netzwerk verbinden. Durch Punkt-zu-Punkt Verbindungen können die Knoten über ein Netzwerk Interface miteinander kommunizieren. Hierbei wird das Verbindungsnetz meist aus Effizienz- und Skalierbarkeitsgründen mit einer 2D Topologie in X und Y Dimension aufgebaut. Bei einer typischen 2D Torus Topologie ist jeder Knoten über seinen zugeordneten Router mit seinem direkten Nachbar in X und Y Richtung verbunden (vgl. Abb. 2.8).

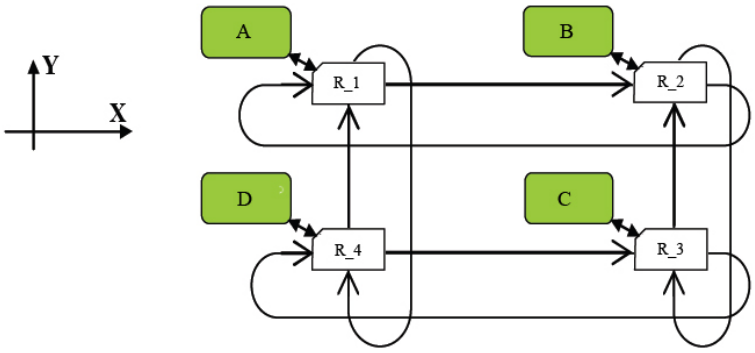

Abbildung 2.8: 2 dimensionale NoC Architektur [12]

Jede Nachricht wird mit einem Routing Header erweitert, der die Zieladresse in X und Y Richtung enthält. Um die Performance und die Effizienz des Netzes zu verbessern, können große Nachrichten fragmentiert werden. Ein Router besteht aus einer logischen Switch, die mit Hilfe einer Routingtabelle die ankommende Nachricht in Richtung des Zieles weiterleitet (vgl. Abb. 2.9). Einer der meist genutzen Routing Algorithmen ist das „Dimension Ordered Routing (DOR)“ oder auch XY Routing genannt [15]. Hierbei wird jedes Paket zuerst in X Richtung geroutet, bis es einen Router erreicht, bei dem die X Adresse mit der des Zielknotens übereinstimmt. Anschließend wird das Paket in Y Richtung bis zum Zielknoten geroutet. Bei unidirektionalen Verbindungen können Pakete nur in Richtung ansteigender Adressen geroutet werden. Wenn nötig, kann am Ende eines Netzwerkes ein Wrap-Arround auf eine niedrigere Adresse erfolgen [12].

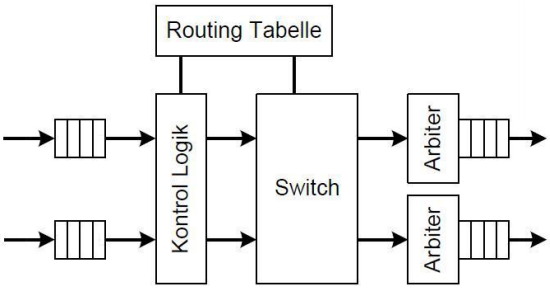

Abbildung 2.9: Aufbau eines NoC Routers [20]

3 Ausblick

In Hinblick auf die Masterarbeit soll eine MPSoC basierte Fahrzeugsteuerungsplattform für das autonome HPEC Fahrzeug entworfen und implementiert werden. Hierbei spielt die Partitionierung der einzelnen IP Blöcke auf die zur Verfügung stehenden MicroBlazes eine bedeutsame Rolle. Abbildung 1.2 ist zu entnehmen, dass die jetzige SoC Plattform sowohl über steuerungsrelevante IP Blöcke, wie beispielsweise die Spurerkennung, als auch über nicht relevante Kontroll IPs verfügt. Das übergeordnete Ziel, die Entwicklung der HPEC Plattform, wird für die kommende Semestern in drei Teilaufgaben zerlegt:

- Network on Chip: Die Technologie der NoC basierten FPGA Systeme soll durchdrungen und erprobt werden. Geeignete Routingverfahren, die für den Einsatz auf einem FPGA in Frage kommen, werden untersucht und in einem ersten MPSoC System getestet. Hierfür muss zuerst ein Netzwerk Interface entworfen werden, welches das zu sendende Paket mit einem Routinginformationsheader ergänzt. Weiterhin sollen erste Erfahrungen bei der Entwicklung eines hardwareseitigen NoC Routers gesammelt werden.

- Software Parallelisierung und Synchronisation bei Shared Memory Multiprozessoren: Ein Multiprozessorsystem ist nur dann sinnvoll, wenn auch die darauf laufende Software parallelisiert wird. Da die Steuerung des Fahrzeuges durch ein RTOS mir partieller Rekonfiguration erfolgen soll [18], muss für die Parallelisierung ein entsprechendes Konzept erarbeitet werden. Erste Ansätze sind beispielsweise OpenMP, das mit Compiler Direktiven arbeitet. Hierbei muss die Möglichkeit der Portierbarkeit auf ein FPGA untersucht werden. Da Amdahl den Speedup eines MPSoC Systems begrenzt, gibt es erste Ansätze, die versuchen den sequentiellen Anteil durch Kommunikation und Synchronisation zu parallelisieren [11]. Diese Software Methoden und geeignete Kommunikationsstrategien sollen erprobt werden. Eine wichtige Anforderung bei der Parallelisierung in MPSoC Systemen ist die Möglichkeit der Identifikation des Prozessors zur Laufzeit durch die Software. Diese Anforderung ist bei Xilinx durch ein „Processor Version Register (PVR)“ gelöst [4]

- Xilinx Dual MicroBlaze Plattform: Da die HPEC Plattform mit Xilinx FPGAs ausgestattet wird, wird zur ersten Erprobung der Multiprozessor Technologie eine MPSoC Plattform mit zwei MicroBlazes implementiert (vgl. Kap. 3.1). Anhand dieser Erprobungsplattform soll das Verständnis der Multiprozessor Technologie auf Basis einer Xilinx Plattform vertieft werden. Die von Xilinx bereitgestellten Multiprozessor IPs müssen für den Einsatz auf der HPEC Plattform auf eventuelle Erweiterungen oder Neuimplementationen überprüft werden.

3.1 Xilinx Dual MicroBlaze System

Als erstes Erprobungsbeispiel wurde ein Xilinx Dual MicroBlaze System mit dem Embedded Development Kit (EDK) implementiert (vgl. Abb. 3.1). Dieses besteht aus zwei getrennten PLB Bussystemen mit je einem MicroBlaze (MB). Jeder Prozessor verfügt über einen lokalen 18 Kbit großen BRAM Speicher. Über einen Multi Port Memory Controller (MPMC) hat jeder Prozessoren Zugriff auf einen gemeinsamen externen Speicher. Über den Xilinx Cache Link (XCL), bestehend aus Datenleitung (DXCL) und Instruktionsleitung (IXCL), kann der Prozessor Daten schnell in den Speicher schreiben und lesen (vgl. rechter MB in Abb. 3.1). Der Xilinx MPMC unterstützt vier Prozessoren die gleichzeitig auf einen gemeinsamen externen Speicher zugreifen können. Eine weitere schnelle und effiziente Shared Memory Strategie ist das Teilen des internen On-Chip BRAMs. Maximal zwei Prozessoren können über den Local Memory Bus (LMB) mit dem BRAM verbunden werden und so geringe Datenmengen schnell austauschen. Zur Synchronisation von gemeinsamen Speicher bietet Xilinx einen eigenständigen XPS_Mutex_Core IP Block an. Mit einem konfigurierbaren Register kann der Entwickler bestimmte kritische Bereiche durch ein Bit schützen. Ein weiterer von Xilinx bereitgestellter IP Block ist der XPS_Mailbox_Core, der über das allgemeine Message Passing Prinzip über eine FIFO den Datenaustausch zwischen zwei Komponenten ermöglicht. Die Ankunft einer Nachricht wird dem Empfänger durch einen Interrupt signalisiert. Solch eine FIFO Kommunikation kann ebenfalls über den unidirektionalen Fast Simplex Link (FSL) zwischen zwei MicroBlazes ermöglicht werden [3].

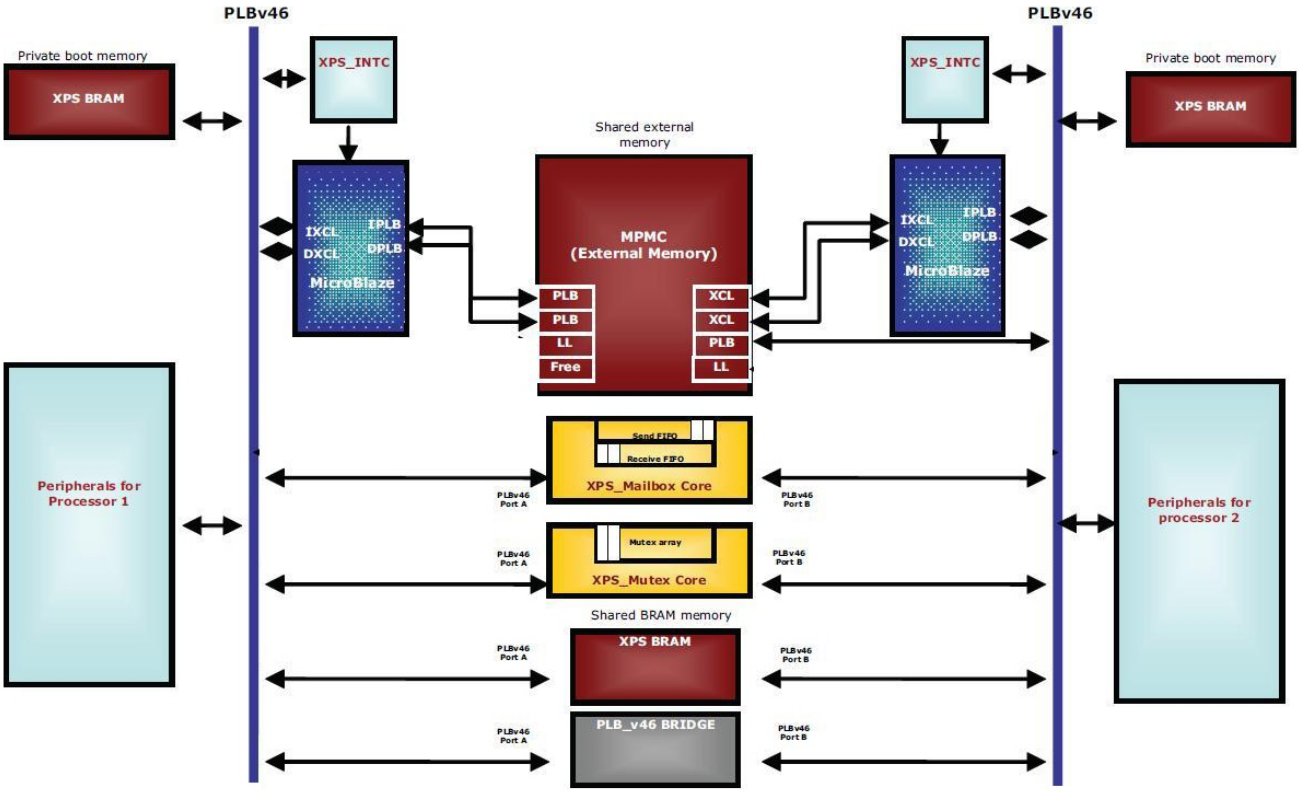

Abbildung 3.1: Xilinx Multiprozessor System [3]

4 Zusammenfassung

In dieser Ausarbeitung wurde ein Überblick über die Integration von Multiprozessorsystemen (MPSoC) auf einem FPGA gegeben. Durch den Einsatz von MPSoC Systemen ist es möglich, den stetig steigenden Leistungsanforderungen in der Computerindustrie gerecht zu werden und gleichzeitig energiesparende Systeme zu entwickeln. Multiprozessoren bieten die Möglichkeit, einzelne Berechnungsschritte auf mehrere Cores zu verteilen, um so die Prozessorlast zu reduzieren. In FPGA basierten Systemen, wie der HPEC Fahrzeugsteuerungsplattform, ist der Einsatz von heterogenen Multiprozessoren von Vorteil, da durch die Spezialisierung der einzelnen Cores der begrenzte Platz auf dem jeweiligen Chip besser genutzt werden kann. Verschiedene Kommunikationsmechanismen, die die Prozessoren miteinander vernetzen, wie die Shared Memory Strategie oder das Message Passing Prinzip, wurden in dieser Arbeit erläutert. Da On-Chip BRAM Speicher auf FPGAs nur in geringen Mengen vorhanden ist, ist der Einsatz von gemeinsamen Speichermodulen, die durch Interprozessmechanismen synchronisiert werden, vorteilhaft.

Network on Chip ist eine neue Technologie, um Prozessoren über ein Netzwerk miteinander zu verbinden. Eine solche Architektur besteht aus Netzwerk Interfaces und Routern, die einen Routingalgorithmus zum Weiterleiten der Pakete an die entsprechenden Empfänger IPs implementiert hat. NoC basierte FPGA Systeme bieten gegenüber Singlebussysteme durch die parallele Transaktionsabarbeitung eine effizientere Kommunikation. Gerade bei Echtzeitanwendungen ist die schnelle und parallele Bearbeitung von Instruktionen wichtig. Aus diesem Grund ist im Kontext der autonomen Fahrzeuge der Einsatz von MPSoC Plattformen gut geeignet, da mehrere unabhängig voneinander agierende Systeme gleichzeitig Berechnungen durchführen müssen.

Bei der Entwicklung eines MPSoC Systems spielt die Parallelisierung der Software eine bedeutsame Rolle. Da nach dem Amdahlschen Gesetz auf Grund der Begrenzung des Speedups kein Programm vollständig parallelisiert werden kann, gibt es erste Ideen, die versuchen, den sequentiellen Programmteil zu parallelisieren. Solche Softwaremethoden, die durch geeignete Kommunikations- und Synchronisationsmechanismen den sequentiellen Anteil auf mehrere Prozessoren verteilen, werden gemeinsam mit der Network on Chip Technologie in den kommenden Semestern untersucht und erprobt. Als erste Erprobungsplattform wird ein Xilinx Dual MicroBlaze System implementiert und die von Xilinx bereit gestellten IP Blöcke getestet.

Glossar

Interuniversity Microelectronic Centre (IMEC) IMEC ist mit knapp 1.700 Mitarbeitern eines der größten Forschungszentren für Nano- und Mikroelektronik in Europa (Belgien). Der Schwerpunkt bei der Forschung liegt bei der Entwicklung von energiesparenden und flexiblen Prozessorsystemen. Ein Beispiel ist die MPSoC Suite. Diese enthält verschiedene Tools, die es dem Entwickler ermöglichen High Level Code automatisch für die Parallelisierung zu optimieren und anschließend auf eine MPSoC Plattform zu mappen. Weitere Produkte sind die Multiprozessor Architektur ADRES und der zugehörige Compiler DRESC [5].

Carolo Cup

Der Carolo Cup ist ein von der TU Braunschweig ins Leben gerufener Wettbewerb für autonome Modellfahrzeuge im Maßstab von 1:10. Die Fahrzeuge müssen drei verschiedene Szenarien bewältigen, dazu gehören das Fahren auf einer Rundstrecke mit und ohne Hindernisse und das parallele Einparken [6]. Seit 2008 nimmt das Team FAUST der HAW Hamburg mit einem Mikrocontroller basierten Fahrzeug ausgestattet mit drei ARM7 Prozessoren teil. Das HPEC Team hat das Ziel im Jahre 2011 mit einem eigenen FPGA basierten Fahrzeug teilzunehmen. Dieses verfügt über eine MPSoC Plattform mit mindestens zwei Spartan3e FPGAs.

MicroBlaze

Der Xilinx MicroBlaze ist ein Softcore Prozessor mit einer 32bit Harvard RISC Architektur. Mit dem Embedded Development Studio (EDK) der Firma Xilinx, kann der MicroBlaze in ein SoC System mit FPGA integriert werden. Über den Local Memory Bus (LMB) werden die getrennten Adress- und Datenspeicher im BRAM mit dem MicroBlaze verbunden. Je nach Größe des FPGAs, kann der Prozessor ab Version 6.00 mit einer 3 bis 5-stufigen Pipeline implementiert werden. Um die Performance des Systems zu erhöhen können ebenfalls separate Instruktions- und Datencaches, die über den Xilinx Cache Link (XCL) angeschlossen werden können, verwendet werden. Je nach Einsatzzweck, kann der Entwickler den Prozessor mit weiteren konfigurierbaren Features ausstatten [3][4].

Fast Simplex Link

Der FSL ist eine schnelle unidirektionale Punkt zu Punkt Verbindung zwischen zwei MicroBlazes. Der Vorteil des Einsatzes des FSLs ist der direkte Zugriff auf das Registerfile des Prozessors. Da die Kommunikation über asynchrone oder synchrone FIFOs erfolgt können die Busteilnehmer, der Master und der Slave, mit unterschiedlichen Taktraten betrieben werden. Der Master kann mit putfsl() Daten in die FIFO schreiben und der Slave holt sie mit dem Befehl getfsl() ab [23].

Abkürzungsverzeichnis

ADRES	Architecture for Dynamically Reconfigurable Embedded Systems
CPU	Central Processing Unit
DMA	Direct Memory Access
DOR	Dimension Order Routing
EDK	Embedded Development Kit
FAS	Fahrerassistenzsysteme
FAUST	Fahrerassistenz- und Autonome Systeme
FPGA	Field Programmable Gate Array
FSL	Fast Simplex Link
HAW	Hochschule für angewandte Wissenschaften
HPEC	High Performance Embedded Computing
IMEC	Interuniversity Microelectronic Centre
IP	Intellectual Property
LMB	Local Memory Bus
MB	MicroBlaze
MIP	Message Passing Interface
MPMC	Multi Port Memory Controller
MPSoC	Multiprocessor System on Chip
NOC	Network on Chip
PLB	Processor Local Bus
RTOS	Realtime Operating System
SoC	System on Chip
TDMA	Time Division Multiple Access
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XCL	Xilinx Cache Link

S	Speedup
p	Paralleler Programmteil
$p-1$	Sequentieller Programmteil
n	Anzahl der Prozessoren

Literaturverzeichnis

[1] A.HUNG ; W.BISHOP ; A.KENNINGS: Symmetric Multiprocessing on Programmable Chips Made Easy. In: *IEEE Design, Automation and Test in Europe* (2009), Nr. IEEE Conferences

[2] 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

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

[4] ASOKAN, Vasanth: Dual Processor Reference Design Suite. In: *Xilinx Application Note* (2008), Nr. XAPP996 v1.3

[5] CENTRE, Interuniversity M.: *Imec*. 2010. – URL http://www2.imec.be/be_en/home.html. – abgerufen 14.09.2009

[6] CUP, Carolo: *Studenten entwickeln autonome Modellfahrzeuge*. 2010. – URL http://www.carolo-cup.de/. – abgerufen 10.02.2010

[7] EICHELE, Hermann: *Multiprozessorsysteme*. B.G. Teubner Stuttgart, 1990. – ISBN 3-519-06128-7

[8] FLYNN, Michael J.: Some Computer Organizations and Their Effectiveness. In: *IEEE Transactions on Computers* (1972), Nr. Volume C-21

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

[10] GUSTAFSON, John L.: Reevaluating Amdahl's law. In: *Communications of the ACM* (1988), Nr. Volume 31 Number 5

[11] HERLIHY, Maurice ; SHAVIT, Nir: *The Art of Multiprocessor Programming*. Elsevier, 2008. – ISBN 978-0-12-370591-4

[12] H.V.THANG ; P.N.NAM: Prototyping of a Network-on-Chip on Spartan 3E FPGA. In: *IEEE Communications and Electronics* (2008), Nr. IEEE Conferences

[13] INSTRUMENTS, National: *NI Developer Zone*. 2008. – URL http://zone.ni.com/devzone/cda/tut/p/id/6616. – abgerufen 12.09.2009

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

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

[16] M.LOGHI ; M.PONCINO ; L.BENINI: Cache Coherence Tradeoffs in Shared-Memory MPSoCs. In: *ACM Transactions on Embedded Computing Systems* (2006), Nr. Volume 5 Nr. 2

[17] MOORE, Gordon E.: Cramming more components onto integrated circuits. In: *Electronics* (1965), Nr. Volume 38 Number 8

[18] OPITZ, Frank: Thread basierte partielle Rekonfiguration von SoC Systemen. In: *INF M1 - Anwendungen 1* (2010)

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

[20] S.LUKOVIC ; L.FIORIN: An Automated Design Flow for NoC-based MPSoCs on FPGA. In: *The 19th IEEE/IFIP International Symposium on Rapid Systemn Prototyping* (2008), Nr. IEEE Conferences

[21] 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

[22] WOLF, Wayne: *High Performance Embedded Computing*. Elsevier, 2007. – ISBN 978-0-12-369485-0

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

[24] XILINX: Spartan-3E FPGA Family: Data Sheet. In: *Xilinx Product Specification* (2009), Nr. DS312 v3.8

Abbildungsverzeichnis