Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung - Anwendungen 2 SS2012

Friedrich Groß

Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte

Fakultät Technik und Informatik
Department Informatik

Friedrich Groß

Thema der Ausarbeitung - Anwendungen 2
SS2012
Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte

Stichworte
TTEthernet, Time-Triggered Ethernet, Real-time Ethernet, Hardware Software Co-Design, Automotive Anwendungen

Kurzzusammenfassung
In dieser Arbeit werden verwandte Arbeiten zu meinem Masterthema *Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte* vorgestellt. Dabei wird eine Hardware- und eine Softwareimplementierung des Protokolls gezeigt. Eine Hardwareimplementierung des PROFINET IRT Protokolls, welches ähnlich dem TTEthernet Protokoll ist, wird ebenfalls vorgestellt. Ein Arbeit der Technischen Universität Dortmund, bei der mehrere Steuergeräte auf ein FPGA synthetisiert werden, zeigt, dass auch auf einfachen Wegen die Komplexität in einem Automobil reduziert werden kann.

Title of the paper
Hardware/Software Co-Design for Real-time Ethernet-based controllers

Keywords
TTEthernet, Time-Triggered Ethernet, Real-time Ethernet, Automotive Applications, Hardware Software Co-Design

Abstract
In this work, related work to my Masterthesis *Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte* is presented. A Hardware- and a Softwareimplementation will be shown. A further Hardware implementation of the PROFINET IRT protocol, which is similar to the TTEthernet protocol, is also presented. A word of the Technical University Dortmund shows a easy way to reduce the amount of control units in a automobile. They do this by synthetisizing several control units on one FPGA.

Inhaltsverzeichnis

1 Einführung **1**
1.1 Einleitung 1
1.2 Aufbau der Arbeit 2
1.3 Time-Triggered Ethernet 2
1.4 Rückblick Anwendungen 1 3
1.5 Ziele Anwendungen 2 3

2 Verwandte Arbeiten **5**
2.1 TTEthernet Implementierungen 5
2.1.1 Hardware 7
2.1.2 Software 9
2.1.3 Weitere TTEthernet Implementierungen 12
2.2 PROFINET IRT - Hardwareimplementierung 13
2.3 Steuergerätekonzept zur Reduzierung der Komplexität im Automobil 14

3 Zusammenfassung **16**

Abkürzungsverzeichnis **17**

Abbildungsverzeichnis **18**

Literaturverzeichnis **19**

1 Einführung

In dieser Arbeit werden verwandte Arbeiten zu meinem Masterthema vorgestellt und analysiert. Es soll festgestellt werden ob bestimmte Teilprobleme von Anderen bereits gelöst wurden und ob diese Lösungen für meine Arbeit relevant sind.

1.1 Einleitung

In einem Automobil werden Bus-Systeme für die Kommunikation zwischen den einzelnen Steuergeräten, Sensoren und Aktoren verwendet. Die steigende Anzahl an Fahrerassistenzsystemen führt zu einer steigenden Anzahl an Steuergeräten, was wiederum zu einer komplexeren Kommunikation und einem erhöhten Bandbreitenbedarf führt. Ein Teil der Kommunikation muss unter Echtzeitbedingungen durchgeführt werden, was insbesondere bei sicherheitsrelevanten Systemen der Fall.

Seit einigen Jahren wird in der Automobilindustrie das Bussystem FlexRay eingeführt, das im Jahr 2000 von einigen Automobil- und Elektronikherstellern entwickelt und standardisiert wurde (vgl. FlexRay Consortium). FlexRay ist echtzeitfähig und lieferte eine Bandbreite von 10Mbit/s. Da die Bandbreite von FlexRay für Bild- und Multimediaübertragungen nicht reicht und die Echtzeitfähigkeit nicht überall benötigt wird, werden weitere Bus-Systeme, die für ihren Einsatzzweck optimiert sind, in den Fahrzeugen verwendet.

Der Einsatz von verschiedenen Bus-Systemen steigert die Komplexität der Steuergeräte und erfordert teilweise verschiedene Schnittstellen und Gateways in einem Steuergerät. Ein einheitlicheres, gleichzeitig schnelles und echtzeitfähiges Bus-System würde das steigende Bandbreitenproblem lösen und die Komplexität verringern.

TTEthernet ist ein echtzeitfähiges Ethernet und gilt als Kandidat für die schrittweise Ablösung von alten Bussystemen im Backbone-Netz eines Automobils. Vorarbeiten haben gezeigt, dass TTEthernet eine Erfolg versprechende Technologie für die Kommunikation in Fahrzeugen ist (vgl. Steinbach u. a., 2010). TTEthernet wurde an der Technischen Universität Wien entwickelt und wird aktuell von der Firma TTTech (vgl. TTTech Computertechnik AG) vermarktet. Dieses Bussystem wird derzeit nicht im Automobil eingesetzt; es befindet sich in der „Evaluierungsphase“ (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009).

In der Masterarbeit geht es darum ein HW/SW-Codesign des TTEthernet-Protokolls zu entwickeln. In dieser Arbeit geht es darum verwandte Arbeiten ausfindig zu machen und diese zu analysieren.

1.2 Aufbau der Arbeit

In diesem Kapitel wird folgend zum besseren Verständnis das TTEthernet Protokoll zusammengefasst beschrieben. Weiterhin wird ein kurzer Rückblick zur vorangegangenen Arbeit *Anwendungen 1* gegeben. Zudem werden Ziele für diese Arbeit definiert.

In Kapitel 2 werden eine Hardware und eine Software Implementierung des TTEthernet Protokolls vorgestellt. Eine Hardwareimplementierung des PROFINET IRT Protokolls, welches ähnlich dem TTEthernet auch ein zeitgesteuertes Ethernet ist, wird ebenfalls vorgestellt.

in Kapitel 3 wird die Arbeit zusammengefasst, dabei werden Überdeckungen zu meiner Arbeit und Differenzen der Arbeiten untereinander erläutert.

1.3 Time-Triggered Ethernet

TTEthernet ist eine Echtzeiterweiterung für das Ethernet. Es ermöglicht u.a. das zeitgesteuerte Kommunizieren innerhalb eines Netzwerks, wobei für die zeitgesteuerten Nachrichten eine deterministische Paketlaufzeit durch das Netzwerk garantiert wird. Um dieses Verhalten zu ermöglichen, werden spezielle TTEthernet-Switches zwischen Netzwerkteilnehmern benötigt.

Das TTEthernet-Protokoll unterstützt drei Nachrichtenklassen, die im Folgenden beschrieben sind:

Time-Triggered-Traffic (TT) zeitgesteuerte Nachrichten mit einem deterministischem Zeitverhalten (konstante Paketlaufzeit mit niedrigem Jitter). Wird von den Netzwerkgeräten mit höchster priorität behandelt.

Rate-Constrained-Traffic (RC) wird für weniger zeitkritische Echtzeitkommunikation verwendet. Die Echtzeitfähigkeit wird durch eine vorher festgelegte garantierte Bandbreite realisiert. Nachrichtenklasse entspricht dem AFDX-Protokoll-Standard (vgl. AIM GmbH).

Best-Effort-Traffic (BE) entspricht dem Standard Ethernet Verkehr. Nachrichten dieser Klasse werden mit niedrigster Priorität behandelt.

Um das zeitgesteuerte Kommunizieren zu ermöglichen, ist eine globale Uhr notwendig, auf die alle Netzwerkteilnehmer synchronisiert werden müssen.

Aus dem Grund beinhaltet die TTEthernet-Spezifikation eine Zeitsynchronisation bei der Netzwerkteilnehmer die aktuelle Uhrzeit vom Synchronisations-Master erhalten. Spezielle Switches durch die diese Nachrichten geleitet werden, addieren Ihre verursachte Weiterleitungszeit in den Paketen auf, so dass Beim Empfang der Synchronisationspakete die tatsächliche aktuelle Netzwerkzeit gelesen wird. Es können mehrere Synchronisations-Master im Netzwerk enthalten sein. Die Switche (Compression-Master) errechnen dann eine Durchschnittszeit, die dann allen Netzwerkteilnehmern (auch den Synchronisations-Mastern) mitgeteilt wird.

1.4 Rückblick Anwendungen 1

Ziel der Masterarbeit ist es ein modularen Kommunikations-Stack für ein Steuergerät im Automobilkontext zu entwickeln. Modular im Sinne von, dass Teile des Stacks, je nach Anwendugsfall, in Hardware oder in Software implementierbar/synthetisierbar sind.

Wird z. B. eine hohe Genauigkeit benötigt, so ist der Entwurf Hardwarelastig zu gestalten. Ist eine hohe Genauigkeit nicht geforder, so führt ein Softwarelastiger Entwurf zu einem geringeren Resourcenbedarf bezüglich logischer Zellen auf einem Entwicklungsboard.

Um das zu ermöglichen muss der Kommunikationsstack in Module aufgeteilt werden und jeweils in Hardware und Software entwickelt werden. Jedes Modul muss dabei eine Hardware- und Softwareschnittstelle anbieten.

1.5 Ziele Anwendungen 2

Das Ziel dieser Arbeit ist es Verwandte Arbeiten zu finden und deren Konzepte zu analysieren. Bei der Analyse wird darauf geachtet, ob Teile der Konzepte für zukünftige Arbeiten relevant sind und mit übernommen werden können. Bei der Recherche wurde mit verschiedenen schärfegeraden das Thema fokussiert wie in Bild 1.1 dargestellt.

Andere Steuergeräte Konzepte Hier wurde aus der „Vogelperspektive“ nach Arbeiten gesucht, deren Ziel es war die Komplexität in einem Automobil zu verringern. In Abschnitt 2.3 wird so eine Arbeit vorgestellt

Andere Time-Triggered Ethernet Implementierungen Zeit gesteuerte Ethernet Implementierungen in Hardware (nicht TTEthernet). In Abschnitt 2.2 wird eine PROFINET IRT Implementierung vorgestellt.

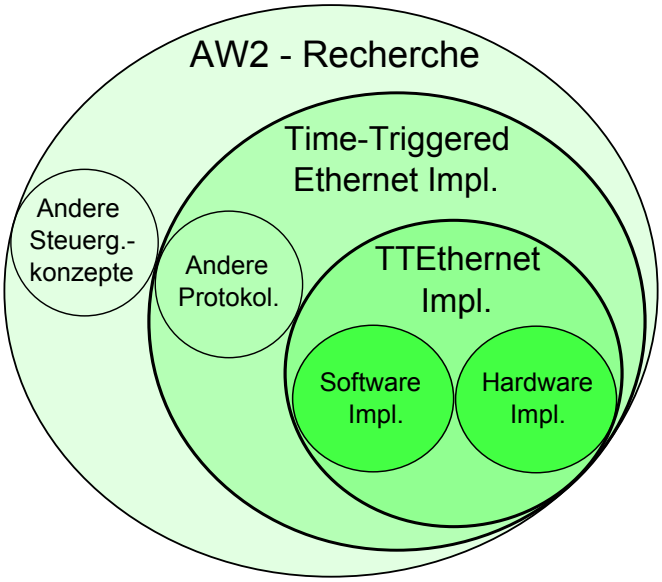

Abbildung 1.1: *Recherchegebiete*

TTEthernet Implementierungen Konkrete TTEthernet Implementierungen in Hardware und Software. In Abschnitt 2.1.1 wird eine Hardwareimplementierung vorgestellt und im Abschnitt 2.1.2 eine Softwareimplementierung.

2 Verwandte Arbeiten

2.1 TTEthernet Implementierungen

Um Überschneidungen konkret finden zu können und diese dann darzustellen, wurde die Problemstellung der Masterarbeit in Module aufgeteilt (siehe Abbildung 2.1). In den kommenden Abschnitten wird bei der Analyse der verwandten Arbeiten die Überschneidung in diesem Bild grafisch dargestellt.

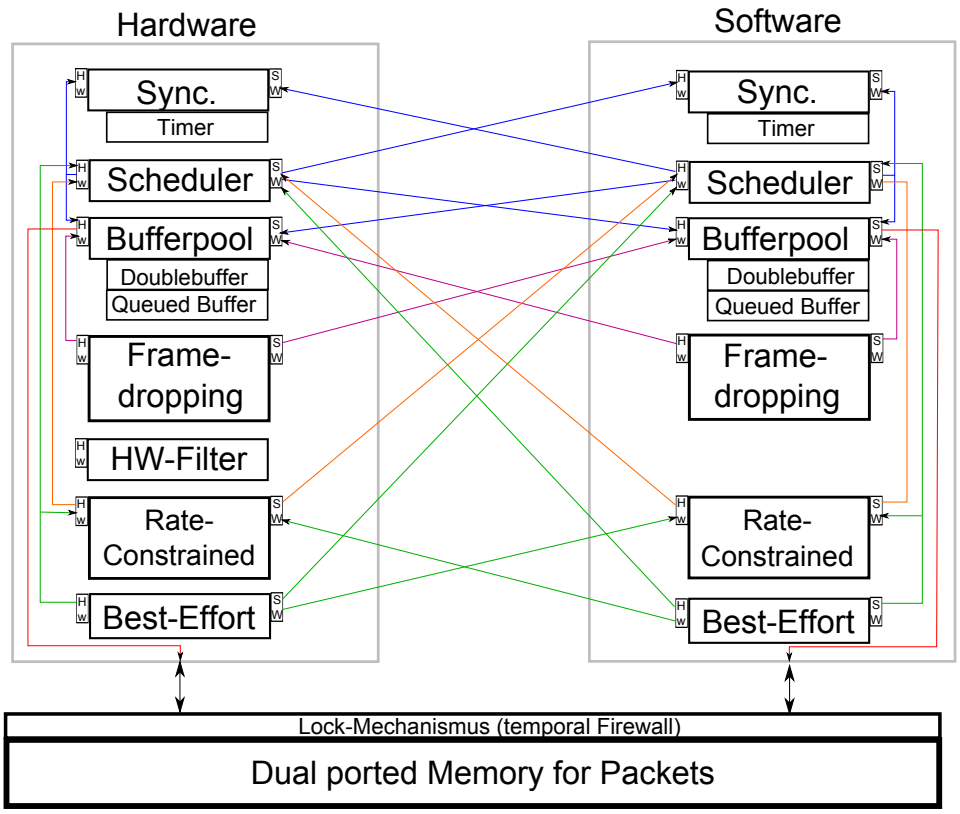

Abbildung 2.1: *Module HW/SW-TTEthernet-Stack*

Abbildung 2.1 zeigt verschiedene Module deren Entwicklung im praktischen Teil der Masterarbeit geplant sind. Vorerst ist geplant jedes Modul in Hardware und in Software zu entwickeln.

Es kann sich aber im weiteren Verlauf der Arbeit ergeben, dass nicht jedes Modul in Hardware oder Software sinnvoll ist.
Die Pfeile zwischen den Modulen sollen Schnittstellen darstellen, die jeweils zwischen den Modulen notwendig sind. Im Folgenden werden die Module kurz beschrieben.

Synchronisation Dieses Modul wird für die Zeitsynchronisation benötigt und sorgt dafür, dass alle Netzwerkteilnehmer mit der gleichen Uhr arbeiten.

Scheduler Dieses Modul wird für die Time-Triggered Nachrichtenklasse benötigt. Hier wird statisch festgelegt, zu welchem Zeitpunkt im TTEthernet-Zyklus, welche Nachricht verschickt wird.

Bufferpool Dieses Modul beinhaltet zwei Buffer. Der *Doublebuffer* hat zwei Speicherplätze um beim Senden Konflikte zu vermeiden. In einen Speicherplatz werden aktuelle Nachrichten reingeschrieben (z. B. Sensorwerte) und aus dem anderen Speicherplatz zum Versenden ausgelesen. Dies kann in verschiedenen asynchron in unterschiedlichen Frequenzen geschehen. Beim Senden wird der zuletzt geschriebene Wert gesendet, auch wenn länger keine Nachricht in den Buffer geschrieben wurde.
Der *Queued Buffer* ist ein FIFO Buffer, in den eine variabel festlegbare Menge an Nachrichten gespeichert werden können. Beim Senden wird die älteste Nachricht gewählt. Ist dieser Buffer leer, wird im Gegensatz zum *Doublebuffer* keine Nachricht versendet.

Framedropping Dieses Modul überwacht den Speicherstand der zu sendenden und der empfangenen Nachrichten. Dabei wird für die kritischen Nachrichten immer ein bestimmter Grad an Speicherplatz freigehalten, indem nieder priorisierte Nachrichten verworfen werden.

Hardware-Filter Dieses Modul ist nicht Bestandteil der AS6802 Spezifikation (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009), soll aber um die Softwareseite zu entlasten implementiert werden. Dieses Modul Soll einstellbar bestimmte Nachrichten im Vorfeld verwerfen können. Dabei sind verschiedene Szenarien denkbar wie z. B. das Filtern von Best-Effort Nachrichten oder Broadcast-Nachrichten.

Rate-Constrained Dieses Modul ist für die Rate-Constrained Nachrichtenklasse zuständig und muss mit Hilfe des Moduls *Scheduler* Lücken in der Nachrichtenklasse Time-Triggered finden und die Rate-Constrained Nachrichten dennoch in Echtzeit (bandbreitenbasiert) verschicken.

Best-Effort Dieses Modul ist für die Best-Effort Nachrichtenklasse zuständig und muss mit Hilfe der Module *Scheduler* und *Rate-Constrained* Lücken finden, um Best-Effort Nachrichten mit niedrigster Priorität zu senden.

Im den folgenden zwei Abschnitten wird auf diese Module eingegangen und dabei beschrieben wie diese dort umgesetzt wurden.

2.1.1 Hardware

In der Arbeit *Hardware Implementation of Time-Triggered Ethernet Controller* Steinhammer und Ademaij (2007) wird die Entwicklung eines *TT Ethernet* Controllers beschrieben. Das *TT Ethernet* und das *TTEthernet* Protokoll nicht genau gleich. Beim *TT Ethernet* Protokoll fehlt die Nachrichtenklasse Rate-Constrained, außerdem ist der Zeit-Synchronisationsalgorithmus vereinfacht.

Im Folgenden wird Bezug zu den in Abschnitt 1.5 definierten Modulen genommen.

Synchronisation

In dieser Arbeit wird für die Synchronisation ein Micro/Macrotick-Timer verwendet. Dieser sorgt dafür, dass die Geschwindigkeit der Zeit einstellbar ist und somit auf die globale Uhr synchronisierbar ist. Die Funktion des Timers wird im Folgenden beschrieben:

Microtick und Macrotick sind zwei verschiedene Timer. Die Frequenz des Microtick-Timers wird durch ein Quartz bestimmt. Mit jeder steigenden Flange wird der Wert im Microtick dekrementiert. Erreicht der Microtick-Timer den Wert 0, wird der Macrotick-Timer inkrementiert, womit die Macrotick-Frequenz bestimmt wird. Der Microtick wird anschließend mit einem Defaultwert beschrieben, der Bestimmt wie oft bis 0 dekrementiert wird. Alle Hardware und Softwaremodule nehmen den Macrotick-Timer als Zeitgeber. Somit kann die Geschwindigkeit des Macroticks durch den Defaultwert des Microticks verändert werden.

Diese Timer besitzen eine Schnittstelle nach außen, so dass eine CPU/Mikrocontroller auf diese zugreifen kann und mit einem auf dem Mikrocontroller existierendem Synchronisationsalgorithmus gesteuert werden kann. Der Synchronisationsalgorithmus ist in der Arbeit nicht beschrieben. Bei allen eintreffenden Ethernet-Paketen wird direkt beim Empfang ein Zeitstempel vom Macrotick-Timer genommen und dieser an die Header-Daten des Ethernet-Pakets angehängt. So kann der Synchronisationsalgorithmus, asynchron anhand des Zeitstempels und der in den Synchronisationsnachrichten enthaltenen Zeit, den Defaultwert des Microticks neu berechnen.

Scheduler

Der Scheduler beinhaltet einen so genannten MEDL-Speicher (Message Descriptor List memory). Diese ist eine verkettete Liste, die für jede Nachricht, die im TT Ethernet Zyklus geplant ist, folgende Informationen Enthält:

- Absendezeitpunkt
- Nachrichtenheader (Ethernet)
- Pointer auf den Datenteil, der in einem Dual ported Memory liegt.

Der MEDL-Speicher ist aufsteigend nach dem Absendezeitpunkt sortiert. Da dieser Speicher eine Linked-List ist, können nachträglich Nachrichten hinzugefügt werden, die dann ohne Kopieroperationen in die Liste einsortiert werden können. Für die richtige Sortierung muss der Softwareteil sorgen, der auf einem Mikrocontroller läuft. Dieser ist in der Arbeit nicht beschrieben.

Ein separates Sendemodul überwacht diesen Speicher und sendet jeweils die Nachricht dessen Absendezeitpunkt gleich des Macrotick-Timers ist. Die MEDL-Liste wird dabei zyklisch in sortierter Reihenfolge abgearbeitet. Abbildung 2.2 zeigt dies schematisch.

Lokale Uhr	==	Absende Zeit	Msg-Header	Ptr-Daten TT-Msg1
		Absende Zeit	Msg-Header	Ptr-Daten TT-Msg2
		Absende Zeit	Msg-Header	Ptr-Daten TT-Msg3
		Absende Zeit	Msg-Header	Ptr-Daten TT-Msg4

Abbildung 2.2: *Ablauf Scheduler und Sendemodul*

Bufferpool

In dieser Arbeit wird der Datenteil einer Nachricht im Dual ported Memory gespeichert. Jeder MEDL-Eintrag hat nur einen Pointer zum Datenteil der beim Senden immer wieder ausgelesen wird. Eine temporale Firewall verhindert, dass der Mikrocontroller und die Hardwaremodule gleichzeitig auf die gleichen Datenteile zugreifen. Diese temporale Firewall wird als Umkehrung des im Modul *Schedule* festgelegten Zeitplans realisiert. Die Begriffe *Doublebuffer* und *Queuedbuffer* werden in diese Arbeit nicht verwendet, jedoch kommt die verwendete Implementierung dem *Doublebuffer* nahe.

Best-Effort

Das Best-Effort Modul wurde in dieser Arbeit wie folgt realisiert:
Steht eine eventbasierte Nachricht an, so wird anhand der aktuellen Zeit und dem Zeitpunkt der nächsten zu sendenden Nachricht durch eine Subtraktion ausgerechnet, wieviel Zeit bis zur nächsten TT-Nachricht bleibt. Anhand der Serialisierungszeit (Bestimmt durch die Bandbreite des Netzwerks) und der Paketgröße kann dann die Sendezeit für die anstehende eventbasierte Nachricht ausgerechnet werden. Ist Sendezeit kleiner als die Zeit bis zur nächsten

TT-Nachricht, so wird die eventbasierte Nachricht verschickt. Wenn nicht, wird so lange gewartet bis eine ausreichend große Lücke da ist. Hierbei wird das Interframe-Gap (12 bytes), dass laut Ethernet-Spezifikation zwischen jeder Nachricht liegen muss mitberücksichtigt.

Zusammenfassung

Diese Arbeit liefert für meine Arbeit wertvolle Informationen, zugleich ist es die einzige Hardwareimplementierung die veröffentlicht wurde. Insbesondere die Konzepte des Hardwaretimers und des Schedulers können in meiner Arbeit Verwendung finden. Der grau schattierte Bereich in Abbildung 2.3 zeigt die Überschneidung mit meiner Arbeit.

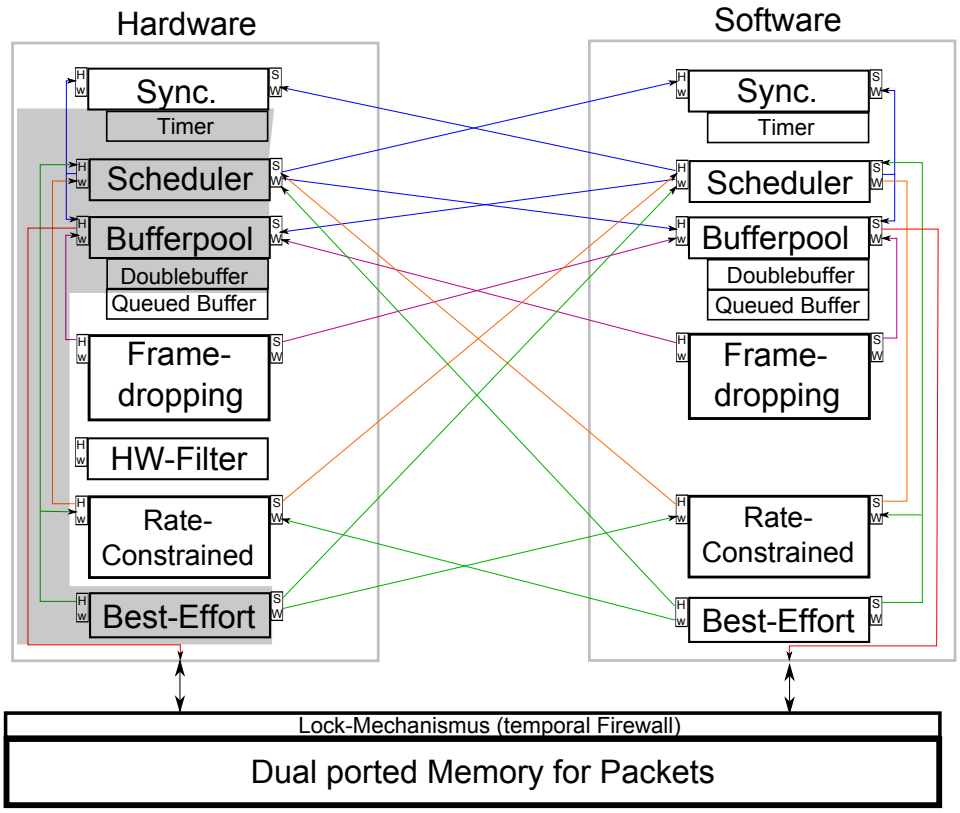

Abbildung 2.3: *Module HW/SW-TTEthernet-Stack - Überdeckung mit (Steinhammer und Ademaij, 2007)*

2.1.2 Software

In dieser Arbeit wurde das TTEthernet Protokoll auf dem Mikrocontrollerboard *Hilscher NXHX500-ETM* realisiert. Es wurde dabei kein Betriebssystem verwendet. Dieses Board ist

mit einer ARM926EJ CPU ausgestattet, die mit 200MHz getaktet wird, alle anderen Komponenten werden mit 100MHz getaktet. Das Board verfügt über 8Mbyte SDRAM, 16Mbyte Flashspeicher und diverse Schnittstellen wie CAN, USB, RS232 und 2x Ethernet. Das besondere an diesem Board ist, dass die Ethernetschnittstellen jeweils eine Zeitstempeleinheit besitzten. Diese zeichnen die Ankunftszeit eines Ethernetframes genau auf. Eine weitere Besonderheit ist, dass ein Timer existiert, der in seiner Geschwindigkeit eingestellt werden kann. Im Folgenden wird anhand der im Kapitel 1.5 festgelegten Module die Arbeit erläutert.

Synchronisation

Für die Synchronisation wurde, ein auf dem Mikrocontrollerboard befindlicher Hardware-Timer verwendet. Dieser entspricht konzeptionell einer Festkommazahl. Der Timer besteht aus zwei 32-Bit Registern. Der Wert im höheren Register kann als Zahl vor den Komma gesehen werden. Der Wert im niederwertigem Register als Zahl nach dem Komma. Nun kann eingestellt werden, welcher Wert bei jedem Takt addiert werden soll. Im Standardfall ist es der Wert 1.0. Bei diesem Wert behält das niederwertige Register immer den gleichen Wert und das höhere taktet pro Takt um eins hoch. Soll de Geschwindigkeit des Timer bspw. um 10% erhöht werden, So stellt man den zu addierenden Wert auf 1.1. Dabei wird das höhere Register bei 9/10 der Takte um eins erhöht und bei jedem zehnten mal um zwei, weil das niederwertige Register dann überläuft.

Der Synchronisationsalgorithmus ist eine Eigenlösung und wurde als Regelung implementiert. Diese weicht von der AS6802 Spezifikation (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009) ab und wird hier deswegen nicht weiter erläutert.

Scheduler

Für das Scheduling wird in der Startup-Phase eine Liste mit den Time-Triggered Nachrichten und deren Absendezeit festgelegt. Diese Liste wird vom Scheduling Modul nach Absendezeit sortiert. Für die erste Nachricht wird ein Event an den Synchronisations-Timer gebunden, dieser Mechanismus wird von der Mikrocontrollerarchitektur bereitgestellt. Dabei wird die Eventzeit mit der Zeit des Synchronisations-Timers verglichen. Wenn diese gleich ist, wird ein Interrupt ausgelöst. In der ISR wird dann die Nachricht verschickt, anschließend die Dauer bis zur nächsten zu verschickenden Nachricht berechnet und die Zeitdifferenz + aktuelle Zeit in das Eventregister geschrieben. Beim nächsten Interrupt wird dann die nächste Nachricht verschickt.

Frame Dropping

Das *Frame Dropping* Modul ist in dieser Arbeit implementiert. Es wird rein Softwaretechnisch bei jeder Empfangenen Nachricht die priorität überprüft. Übersteigt der Speicherfüllstand ein bestimmtes Level, so werden nieder priorisierte Nachrichten verworfen. Ist der Speicher voll, so werden auch neu eingetroffene hoch priorisierte Nachrichen verworfen. Dieses Modul braucht die meiste CPU-Zeit.

Best Effort

Das Versenden von Best Effort Nachrichten wurde wie folgt realisiert. Nach jedem Sendevorgang von Time-Triggered Nachrichten wird geprüft, ob in der Zeit bis zur nächsten Time-Triggered Nachricht ein Fullsize-Frame (1540 Byte) verschickt werden kann. Ist dies der Fall, so wird eine Best-Effort Nachricht verschickt. Wenn nicht wird beim Versenden der nächsten Time-Triggered Nachricht erneut geprüft.

Bufferpool / Rate-Constrained

In dieser Arbeit wurde der Queued- und der Double-Buffer, wie in AS6802 spezifiziert, bzw. wie in Abschnitt 1.5 beschrieben, implementiert. Die Nachrichtenklasse Rate-Constrained wurde ebenfalls implementiert, ist jedoch bis zum Ende der Arbeit größtenteils ungetestet geblieben und wird in zukünftigen Arbeiten behandelt.

Zusammenfassung

In dieser Arbeit wurden große Teile der AS6802 Spezifikation umgesetzt. Der Synchronisationsalgorithmus wurde, nicht wie in der Spezifikation gefordert, als Offsetkorrektur, sondern als Regelung implementiert. Dies hat den Vorteil, das eine höhere Genauigkeit beim Versenden von Time-Triggered Nachrichten erreicht wird und den Nachteil, dass diese Regelung unter schlechten Umständen anfangen könnte zu Schwingen.

In Abbildung 2.4 wird die Überschneidung zu meiner Arbeit blau schattiert dargestellt.

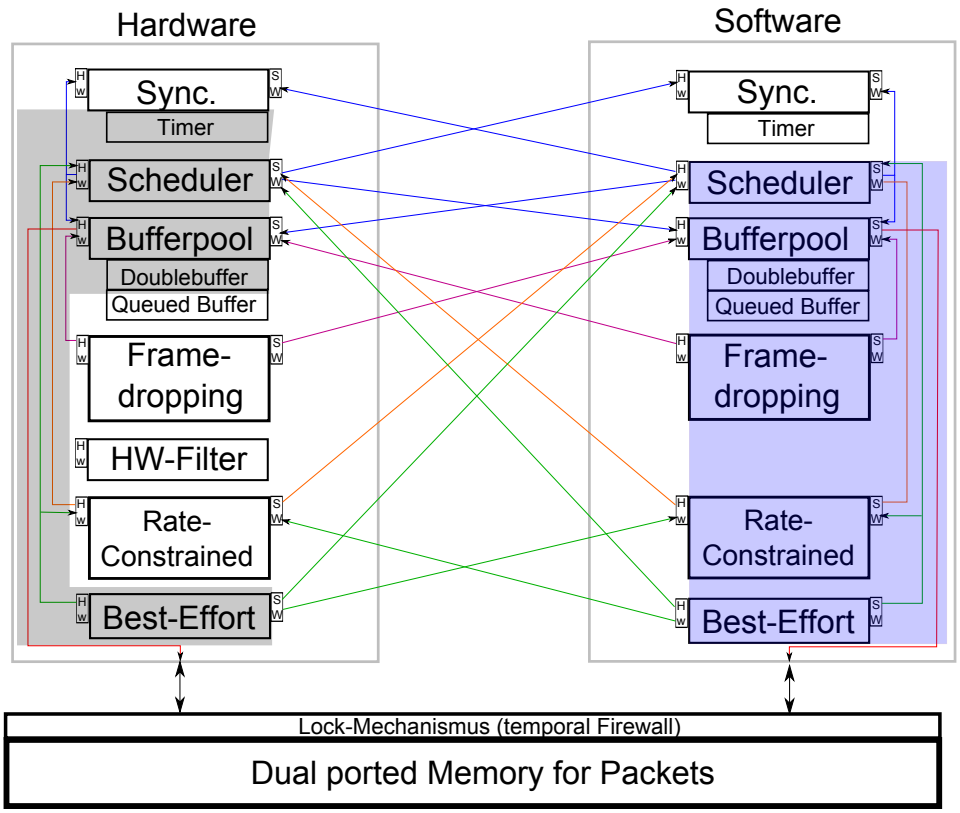

Abbildung 2.4: *Module HW/SW-TTEthernet-Stack - Überdeckung mit (Müller, 2010)*

2.1.3 Weitere TTEthernet Implementierungen

In der Arbeit *Realtime-Ethernet für automotive Anwendungen: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet* (vgl. Steinbach, 2010) wurde das TTEthernet Protokoll für die Simulationsumgebung OMNeT++ in C++ implementiert. Die Implementierung beinhaltet Endsysteme und Switche, womit ein ganzes Netzwerk simuliert werden kann. Die Implementierung beinhaltet nicht die Synchronisation der Endsysteme. Sourcen stehen meiner Arbeit zur Verfügung und kann evtl. für die Softwareteil verwendet werden.

In der laufenden Arbeit *Einbettung einer TTEthernet Synchronisation in eine OMNeT++ basierte Simulationsumgebung* (vgl. Todorov, 2012), wird die eben Beschriebene Simulation um den Synchronisationsmechanismus erweitert. Die Module werden nach der AS6802 (vgl. SAE - AS-2D Time Triggered Systems and Architecture Committee, 2009) Spezifikation implementiert. Die Sourcen dieser Arbeit stehen meiner Arbeit ebenfalls zur Verfügung.

2.2 PROFINET IRT - Hardwareimplementierung

Profinet IRT ist ein Echtzeit Ethernet Protokoll, welches ähnlich dem TTEthernet zeitgesteuert funktioniert. In diesem Protokoll ist eine künstlich geschaffene Grenze von 250 Microsekunden für die kürzeste Zykluszeit festgeschrieben. Dies ist aus der Anforderung entstanden, dass ein Full-Size Ethernetframe, zusätzlich zum Protokolloverhead, innerhalb eines Zykluses übertragen werden kann.

Es gibt jedoch Anwendungsfälle, wie Motion-Control Systeme, bei denen diese Zykluszeit zu lang ist und bei denen keine Full-Size Frames übertragen werden. In diesen Fällen wäre es nützlich vom Protokoll abzuweichen und eine kürzere Zykluszeit im Netzwerk zu verwenden. Die Arbeit *Optimising PROFINET IRT for Fast Cycle Times* der ZHAW Zürich (vgl. Gunzinger, 2010) beschreibt eine FPGA Implementierung des PROFINET IRT Protokolls mit der Zykluszeiten von 31.25 Microsekunden möglich sind.

Das PROFINET IRT Protokoll wird mit dem *Precision Time Protokoll* synchronisiert, welches im IEEE1588 Standard definiert ist. Dieser Standard erfordert es, wie beim TTEthernet, dass eingehende Nachrichten einen genauen Eingangs-Zeitstempel erhalten. Um eine hohe Synchronisationsgenauigkeit zu erhalten, muss in erster Linie die Zeitstempeleinheit genau arbeiten. Aus dem Grund wurde in dieser Arbeit ein Hardwarebaustein entwickelt, der direkt nach dem Erkennen des *Start of Frame delimiter* (das Bit direkt nach der Präambel), einen Zeitstempel aufzeichnet. Dieser Baustein wird mit 400MHz getaktet. Die Zeit wird von einem Timer genommen, dessen Geschwindigkeit sich von einem Synchronisationsalgorithmus einstellen lässt. Der Synchronisationsalgorithmuss wird in einem *Offset/Drift Control Block*, der durch eine Addition oder Subtraktion die Uhr einstellt und zum Anderen die Geschwindigkeit der Uhr anpasst.

Die Zeitstempeleinheit wird außerdem dazu verwendet die Pünktlichkeit von eingehenden Paketen festzustellen. Kommt eine Echtzeitnachricht zu spät an, so wird diese im Vorfeld von der Hardware verworfen.

Der komplette PROFINET IRT Block mit zwei MAC-Einheiten, einer 400MHz Zeitstempeleinheit, einem Hardware Zeitsynchronisation wurde auf einm Altera Cyclon III Board synthetisiert. Auf diesem Board stehen 120.000 *logic elements* (LE) zur Verfügung. Verbraucht wurden davon weniger als 25000. Der Synchronisationsmechanismus verwendet davon 9000 LE's und die Zeitstempeleinheit 2960 LE's.

Die synthetisierte Switch erreicht beim Versenden von Nachrichten ein Jitter von 6ns und braucht 1,4 Microsekunden um ein Paket weiterzuleiten (Cut-Through Technologie). Die Synchronisationsgenauigkeit beträgt <50ns.

Zusammenfassung

In dieser Arbeit wird die Hardwareimplementierung des PROFINET IRT protokolls beschrieben. Der Synchronisierung hat ähnliche Anforderungen bezüglich des Timers, wie beim TTEthernet. Andere Teile des Protokolls unterscheiden sich jedoch deutlicher und wurden hier nicht weiter beleuchtet. Der Aufbau der Zeitstempeleinheit wird in dieser Arbeit am besten beschrieben und wird in meiner Arbeit eine Hilfe sein.

2.3 Steuergerätekonzept zur Reduzierung der Komplexität im Automobil

Die Projektgruppe *CoaCh (Car on a Chip)* der Technischen Universität Dortmund hat in einer Projektarbeit (vgl. Spinczyk, 2009) mehrere Steuergeräte auf ein FPGA synthetisiert und diese mit einem *on Chip CAN-Bus* innerhalb des FPGAs verbunden. Abbildung 2.5 stellt die Zusammensetzung schematisch dar. Die Motivation hinter dem Thema ist, die Komplexität im Automobil zu verringern ohne Steuergeräte neu entwickeln zu müssen. Vorhandener Programmcode kann in den baugleichen Mikrocontrollern fast eins zu eins verwendet werden. Ein weiterer Punkt ist, dass Automobilhersteller bis zu 30 Jahre Ersatzteile liefern können müssen, was zu hohen Kosten bei alten Mikrocontroller führt. Durch diese Methode können alte Mikrocontroller auf neue FPGAs synthetisiert werden. Der Projektgruppe standen echte Automobildaten des Partner Audi zur Verfügung. Außerdem standen der Projektgruppe zahlreiche Mikrocontroller in Form von VHDL/Verilog Code zur Verfügung, viele davon sind als OpenSource erhältlich. In dieser Arbeit wurde ein eigener *on Chip CAN-Bus* entwickelt, da kommerziellen Lösungen den Anforderungen nicht genügten.

Mit dem programmierten FPGA wurde ein Demonstrator aufgebaut, der aus einem Lenkrad, einem Scheinwerfer, einem Gangwahlschalter und einem Gas- und Bremspedal bestand.

Eine der wichtigsten Anforderungen war, dass die einzelnen Mikrocontroller isoliert voneinander arbeiten können. Da aber ein gemeinsamer DDR3-Speicher von allen genutzt wurde, musste hier ein Memory-Multiplexing Modul eingesetzt werden umd die Isolation sicherzustellen. Um die Isolation nachzuweisen hat man alle Mikrocontroller ein 256Byte langes Array, weiles im DDR3-RAM lag mit dem Bubblesort-Algorithmus sortieren lassen. Dies wurde zu erst einzeln und dann gleichzeitig durchgeführt. Die Laufzeiten hatten sich hierbei nur leicht erhöht. Dennoch gilt das Ergebnis in dieser Arbeit als Nachweis für die Isolation der Mikrocontroller.

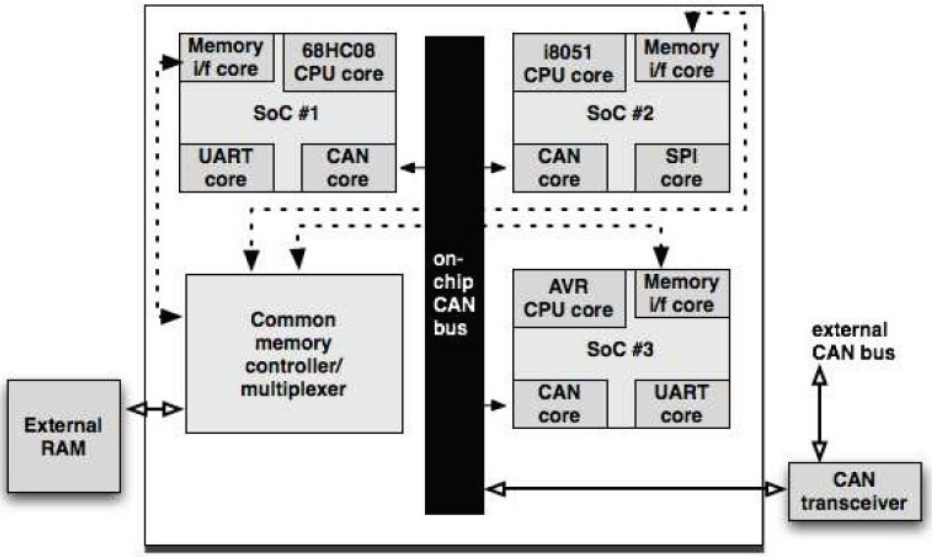

Abbildung 2.5: *Automobiles Network-on-Chip*

Zusammenfassung

Diese Arbeit zeigt eine einfache Lösung die Anzahl der Steuergeräte in einem Automobil zu reduzieren. Hier wird insbesondere der Vorteil darin gesehen, dass keine Lagerbestände, die bis zu 30 Jahre halten müssen nicht gehalten werden müssen, sondern alte Steuergeräte auf aktuell produzierte Hardware synthetisiert werden kann.

3 Zusammenfassung

Die Überscheidungsgrafiken 2.3 und 2.4 zeigen, dass die recherchierten Arbeiten gute Überschneidungen mit meiner Arbeit haben. Es konnten Hardware- und Softwareimplementierungen für das TTEthernet Protokoll gefunden werden. Einige Implementierungen stehen nicht nur konzeptionell, sondern auch als Sourcen zur Verfügung. Es wurde aus jedem, in Kapitel 1.5 definierten Recherchegebiet eine Arbeit vorgestellt. Interessant dabei ist, dass in den Arbeiten unterschiedliche Konzepte für Timer verwendet wurden. In der Hardwareimplementierung die in Kapitel 2.1.1 beschrieben wurde, wird ein Micro-/Macrotimer verwendet wo hin gegen in der Softwareimplementierung (die ein Hardwaretimer verwendet), die in Kapitel 2.1.2 beschrieben wird ein Festkomma-Konzept für den Timer verwendet wird. Welcher Timer für meine Implementierung der beste ist wird in zukünftigen Arbeiten analysiert.

Die OMNeT++ Implementierungen wurden hier nur kurz beschrieben, werden jedoch eine große Hilfe für mich sein, da hier die State-Diagramme der Spezifikation in C++ komplett implementiert worden sind.

Abkürzungsverzeichnis

µC	Mikrocontroller
AFDX	Avionics Full Duplex Switched Ethernet
ASIC	Application-specific integrated circuit
BE	Best-Effort
CM	Compression Master
CoRE	Communication over Real-time Ethernet
CPU	Central processing unit
ES	Endsystem
ESP	Elektronisches Stabilitätsprogramm
FPGA	Field Programmable Gate Array
HW	Hardware
LIN	Local Interconnect Network
MOST	Media Oriented Systems Transport
NAFTA	North American Free Trade Agreement
PCF	Protocol Control Frame
RC	Rate-Constrained
SM	Synchronisation Master
SW	Sowftware
TT	Time-Triggered
TTEthernet	Time-Triggered Ethernet
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Abbildungsverzeichnis

1.1 Recherchegebiete 4

2.1 Module HW/SW-TTEthernet-Stack 5
2.2 Ablauf Scheduler und Sendemodul 8
2.3 Module HW/SW-TTEthernet-Stack - Überdeckung mit Steinhammer und Ademaij 9
2.4 Module HW/SW-TTEthernet-Stack - Überdeckung mit Müller 2011 12
2.5 Automobiles Network-on-Chip 15

Literaturverzeichnis

[AIM GmbH] AIM GMBH: *Avionics Databus Solutions*. – URL http://www.afdx.com/. – Zugriffsdatum: 2010-02-24

[FlexRay Consortium] FLEXRAY CONSORTIUM: *FlexRay*. – URL http://flexray.com/. – Zugriffsdatum: 2011-02-07

[Gunzinger 2010] GUNZINGER, David: Optimising PROFINET IRT for fast cycle times: A proof of concept. In: *Factory Communication Systems (WFCS)* 8 (2010)

[Müller 2010] MÜLLER, Kai: *Entwicklung eines TTEthernetclients als Embeddedsystem auf einem ARM-Board*. Dezember 2010. – Bachelorthesis

[SAE - AS-2D Time Triggered Systems and Architecture Committee 2009] SAE - AS-2D TIME TRIGGERED SYSTEMS AND ARCHITECTURE COMMITTEE: *Time-Triggered Ethernet (AS 6802)*. 2009. – URL http://www.sae.org. – Zugriffsdatum: 2010-12-11

[Spinczyk 2009] SPINCZYK, Olaf: Car on a Chip. In: *Technische Universität Dortmund* (2009)

[Steinbach 2010] STEINBACH, Till: *Realtime-Ethernet für automotive Anwendungen: Metriken und deren simulationsbasierte Evaluierung am Beispiel von TTEthernet*. Februar 2010. – URL http://papers.till-steinbach.de/s-reaam-10.pdf. – Bericht

[Steinbach u. a. 2010] STEINBACH, Till ; KORF, Franz ; SCHMIDT, Thomas C.: Comparing Time-Triggered Ethernet with FlexRay: An Evaluation of Competing Approaches to Real-time for In-Vehicle Networks. In: *8th IEEE Intern. Workshop on Factory Communication Systems*. Piscataway, New Jersey : IEEE Press, Mai 2010, S. 199–202

[Steinhammer und Ademaij 2007] STEINHAMMER, Klaus ; ADEMAIJ, Astrid: Hardware Implementation of Time-Triggered Ethernet Controller. In: *Submission for the IESS - Network and communication systems* (2007). – URL http://www.vmars.tuwien.ac.at/documents/intern/2218/IESS07_paper_33.pdf. – Zugriffsdatum: 2012-08-27

[Todorov 2012] TODOROV, Lazar: *Einbettung einer TTEthernet Synchronisation in eine OMNeT++ basierte Simulationsumgebung*. 2012. – Bachelorthesis

[TTTech Computertechnik AG] TTTECH COMPUTERTECHNIK AG: . – URL http://www.tttech.com. – Zugriffsdatum: 2011-01-17