Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung - Masterseminar WiSe 2012/2013

Friedrich Groß

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

Fakultät Technik und Informatik
Department Informatik

Friedrich Groß

Thema der Ausarbeitung - Masterseminar
WiSe 2012/2013
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 wird die Idee des Masterthemas *Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte* unter der Informationsgewinnung von verwandten Arbeiten konkreter vorgestellt. Verwandte Arbeiten haben gezeigt, dass reine Softwareimplementierungen des TTEthernet bei hoher Bandbreite einen zu hohen CPU-Ressourcenbedarf haben. Aus dem Grund beschäftigt sich diese Arbeit mit einer Hardware/Software Codesign Lösung für den TTEthernet-Stack. Es wird ein Baukastensystem vorgestellt, aus dem sich je nach Anforderungen die optimale Hardware/Software-Partitionierung bilden lässt, so dass Echtzeitanforderungen, CPU-Ressourcen und Kosten optimal sind.

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
This work presents the idea of the mastertheasis *hardware / software co-design for real-time Ethernet-based controllers* under the acquisition of information of related work. Related work has shown that pure software implementations of TTEthernet have a high CPU resource requirements at high bandwidth. For the reason this study deals with a Hardware / Software Co-design solution for TTEthernet stack. A modular system is presented, from which it can be, depending on the requirements form the optimal hardware / software partitioning, so that real-time requirements, cost and CPU resources are optimally.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation 1
1.2 Aufbau der Arbeit 1
1.3 Verwandte Arbeiten 2
1.4 Grundlagen 3

2 Idee der Masterarbeit **5**
2.1 Zielsetzung 5
2.2 Anforderungskriterien 5
2.3 Partitionierungsauswirkungen auf Anforderungskriterien 7
2.3.1 Timestamping 7
2.3.2 Synchronisation 8
2.3.3 Scheduler 9
2.3.4 Framedroppiing 9
2.3.5 Rate-Constrained 10
2.3.6 Best-Effort 10
2.4 Zielfunktionen 11

3 Risiken und Ausblick **12**

Abkürzungsverzeichnis **13**

Literaturverzeichnis **14**

1 Einleitung

1.1 Motivation

Die steigende Anzahl an Fahrerassistenzsystemen in einem Automobil steigert die Komplexität in einem Automobil. Dies gilt auch für die zur Kommunikation notwendigen Bussysteme. Abbildung 1.1 zeigt die durchschnittliche Anzahl an Knotenpunkten in einem Fahrzeug. Die bisher verwendeten Bussysteme kommen momentan an ihre Grenzen (Badstübner, 2008, BMW Group Forschung und Technik), da ein immer höherer Bandbreitenbedarf durch Kamera- oder 3D-Laserscanner -basierte Assistenzsysteme entsteht. Einige dieser Daten müssen in Echtzeit übertragen werden. Um den stetig steigenden Kommunikationsanforderungen gerecht zu werden, ist die Automobilindustrie auf der Suche nach einem neuen Bussystem, welches eine hohe Bandbreite liefert, echtzeitfähig und kostengünstig ist. TTEthernet ist ein Kandidat, der diese Anforderungen erfüllen könnte und deswegen momentan in der Evaluierungphase ist (vgl. Steinbach u. a., 2010). Während der Evaluation wurde festgestellt, dass reine Softwareimplementierungen bei einer hohen Kommunikationslast einen hohen CPU-Ressourcenbedarf haben, so dass kaum CPU-Ressourcen für eine Anwendung übrig bleiben (vgl. Müller, 2011). Ziel dieser Arbeit ist es, eine HW/SW-Codesigen-Lösung zu finden, die je nach Anforderungen bestimmte Module in Hardware auslagert, so dass ein Optimum an benötigter Echtzeitbedinungen, Chipfläche und CPU-Ressourcenbedarf der Anwendung gefunden werden kann.

1.2 Aufbau der Arbeit

Im zweiten Kapitel, erster Abschnitt, werden die verwandten Arbeiten aus der Ausarbeitung (Groß, 2012) zusammengefasst dargestellt. Im zweiten Abschnitt folgen dann die wichtigsten Grundlagen, die zum Verständnis dieser Arbeit beitragen sollen.

Im dritten Kapitel wird die Idee der Masterarbeit präsentiert. Es wird dabei ein Baukastensystem zur HW/SW-Partitionierung gezeigt und eine Bewertung darüber, wie sich die Partitionierung der einzelnen Module auf verschiedene Anforderungskriterien auswirken könnte.

Im dritten Kapitel werden die Risiken des Projekts aufgezeigt und es wird ein Ausblick für das weitere Vorgehen beschrieben.

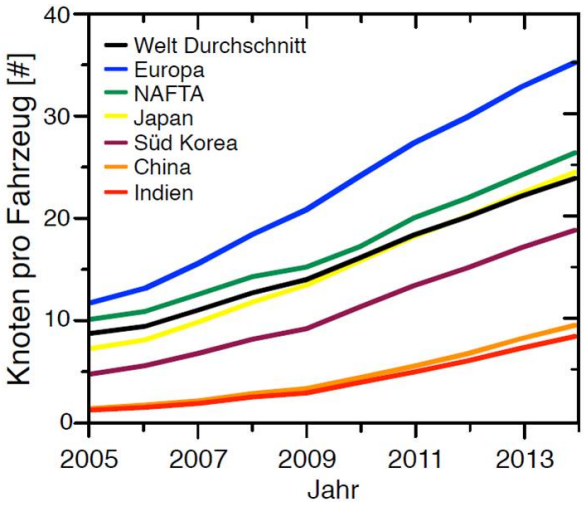

Abbildung 1.1: *Durchschnittliche Anzahl Knotenpunkte in einem Fahrzeug (vgl. Bruckmeier, 2010)*

1.3 Verwandte Arbeiten

Einen ausführlichen Bericht über die verwandte Arbeiten kann in (Groß, 2012) gefunden werden. Hier möchte ich nur ein kurzen Einblick auf die Rechercheergebnisse geben.

In der Arbeit (Steinhammer und Ademaij, 2007) wird eine Hardwareimplementierung des Time-Triggered Ethernet vorgestellt. In dieser Arbeit wurde jedoch die Nachrichtenklasse *Rate-Constrained-Traffic* nicht implementiert und für die Zeitsynchronisation nur ein Timer zur Verfügung gestellt, der sich jedoch in der Geschwindigkeit anpassen lässt. Es werden Konzepte für das Senden von Time-Triggered Nachrichten, das Einordnen von Best-Effort Nachrichten zwischen den Time-Triggered Nachrichten und die Konfiguration des Gesamtsystems vorgestellt.

In der Arbeit (Müller, 2011) wird eine vollständige Softwareimplementierung des Time-Triggered Ethernet vorgestellt. In dieser Arbeit hat sich gezeigt, dass Softwareimplementierungen bei hohen Bandbreiten einen hohen CPU-Bedarf haben. Der CPU-Bedarf kann dabei so ansteigen, dass kaum CPU-Ressourcen für eine Anwendung übrig bleiben. In dieser Arbeit wurde ein alternativer Synchronisationsalgorithmus verwendet, der nicht wie in (SAE International, 2011) eine einfache Offset-Korrektur vornimmt, sondern anhand der eintreffenden Zeitpakete eine Regelung der Uhr vornimmt.

In der Arbeit (Gunzinger, 2010) wird eine Hardwareimplementierung des Protokolls PROFINET IRT (vgl. PROFIBUS & PROFINET International) vorgestellt. PROFINET IRT ist ebenfalls ein zeitgesteuertes Ethernet, so das man hier bei der Implementierung auf ähnliche Probleme stößt. In dieser Arbeit wird insbesondere auf die Synchronisation eingegangen, die mittels

IEEE 1588 (vgl. Institute of Electrical and Electronics Engineers, 2002) vorgenommen wird. Dazu wurde ein Hardwarebaustein entwickelt, der direkt beim erkennen des Start-of-Frame-Delimiter (Das Byte direkt nach der Präambel) einen Zeitstempel setzt. Dadurch wurde ein Synchronisationsgenauigkeit von < 50ns erreicht. In dieser Arbeit hatte man das Ziel, eine möglichst kurze Zykluszeit zu erreichen, um ein Motion-Control System mit dem Netzwerk zu verbinden. Die minimale Zykluszeit beträgt in diese Implementierung 31.25µs.

Es gibt viele Arbeiten, die Methoden zur HW/SW-Partitionierung vorstellen, jedoch ist die Arbeit (D'Ambrosio und Hu, 1994) eine der wenigen Arbeiten, die bei der Partitionierung nicht nur auf Kosten und Performance achten, sondern auch auf Einhaltung von Echtzeiteigenschaften. In diesem Ansatz wird Hardware als Ressource gesehen und Software als Task's betrachtet, die die HW-Ressourcen benutzen (wie in vielen anderen Arbeiten auch). Jedoch wird eine Möglichkeit aufgezeigt, wie die Partitionierung in einem Simulator simuliert werden kann und man dabei eine Machbarkeitsfaktor erhält, der einem einen Wahrscheinlichkeit liefert, ob die Echtzeitanforderungen eingehalten werden könne.

1.4 Grundlagen

Time-Triggered Ethernet ist eine Echtzeiterweiterung des Standard-Ethernet und ist in (SAE International, 2011) spezifiziert. Es unterstützt drei Nachrichtenklassen, die im Folgenden kurz beschrieben werden:

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 bei der Nachrichtenklasse *Time-Triggered-Traffic* ein deterministisches Zeitverhalten erreichen zu können, werden in dem Netzwerk spezielle Switches benötigt. Alle Netzwerkteilnehmer, die eine TT-Nachricht passieren, müssen u.a. den Sendezeitpunkt über die jeweilige Nachricht speichern. So kann der Netzwerkteilnehmer dafür sorgen, dass zu diesem Zeitpunkt die Sendepuffer für die Zielports freigehalten werden. So kann sichergestellt werden, dass TT-Nachrichten mit einem geringem Jitter durch das Netzwerk versendet werden können.

Um diese Funktionalität zu erreichen, müssen die Uhren der Teilnehmer synchronisiert werden. Dazu wird ein 2-Phasen Synchronisationsalgorithmus verwendet, bei dem mehrere Rollen definiert sind. Zu Beginn eines Zykluses senden die *Synchronisation-Master* (meistens Endteilnehmer) an den *Compression-Master*(meistens Switch) ihre aktuelle Uhrzeit. Der *Compression-Master* errechnet aus den empfangenen Zeiten eine Durchschnittszeit und sendet diese an alle Teilnehmer, auch an die *Synchronisation-Client*'s (meistens Endteilnehmer). Alle Teilnehmer korrigieren nun ihre Uhr anhand des empfangenen Pakets. Jeder Teilnehmer berücksichtigt beim Empfang von Zeitpaketen verschiedene Verzögerungen, die auftreten können. So wird die Signallaufzeit auf der Leitung, die fest konfiguriert wird, auf empfangene Zeiten aufaddiert. Auch wird die eigene Verzögerung zum Bearbeiten des Pakets mit aufaddiert.

2 Idee der Masterarbeit

Wie in Abschnitt 1 beschrieben, ist die Automobilindustrie auf der Suche nach einem Bussystem, welches eine hohe Bandbreite liefert, echtzeitfähig und kostengünstig ist. TTEthernet ist ein möglicher Kandidat, der diese Anforderungen erfüllen kann. Bei der Recherche nach verwandten Arbeiten wurde eine Hardwareimplementierung des TTEthernet gefunden, die jedoch nicht vollständig und nicht offengelegt ist. Die existierende Softwareimplementierung zeigt Schwächen bei hoher Netzwerkbelastung, was dazu führt, dass kaum CPU-Ressourcen für eine Anwendung übrig bleiben. In diesem Kapitel wird nun die Idee der Masterarbeit beschrieben, eine dynamische anwendungsfall-spezifische HW/SW-Codesign-Lösung zu finden.

2.1 Zielsetzung

Ziel der Masterarbeit ist es, ein HW/SW-Codesign zu entwickeln, welches je nach Anforderungen eine optimierte Hardware-Software Komposition ermöglicht. Dabei kann es sein, dass es bei einer Anwendung Sinn macht, möglichst viele Module in Hardware zu partitionieren und bei einer anderen Anwendung möglichst viel in Software zu partitionieren.

Die Idee ist, einen Baukasten zu entwickeln, aus dem je nach Anforderungen Module in Hardware oder in Software ausgewählt werden können. Die Module sind Bestandteile des TTEthernet-Stacks. Dabei kann es Sinn machen, bestimmte Module gar nicht auszuwählen, weil z.B. bestimmte Nachrichtenklassen von der Anwendung nicht verwendet werden.

Welche HW/SW-Partitionierung für eine konkrete Anwendung verwendet werden soll, entscheiden die Anforderungen einer Anwendung. Die Kriterien, die für die Partitionierung entscheidend sind, werden im folgenden Abschnitt näher betrachtet.

2.2 Anforderungskriterien

Es gibt 4 Anforderungskriterien, die im Folgenden kurz beschrieben werden.

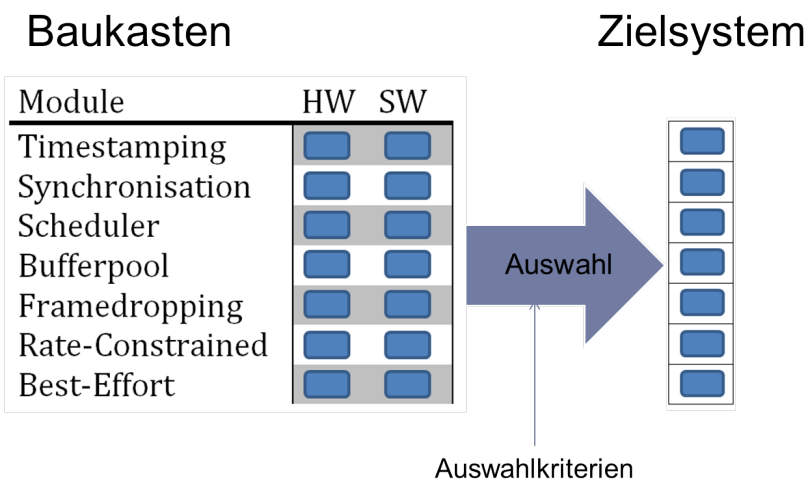

Abbildung 2.1: *Baukasten*

Echtzeitanforderungen Wie schnell und innerhalb welcher Zeitfenster sollen Nachrichten übertragen werden.

Verfügbare CPU-Ressourcen Je höher der Rechenaufwand einer Anwendung ist, umso geringer sind die verfügbaren CPU-Ressourcen für den TTEthernet-Stack.

Verfügbare Chipfläche Sind die Maximalkosten (Chipfläche) festgelegt, so müssen Module möglichst softwarelastig partitioniert werden.

Benötigte Funktionalität Nicht in jeder Anwendung wird die komplette Funktionalität benötigt. Eine Auswahl der benötigten Module soll ermöglichen, nur die nötigste Funktionalität zu implementieren.

Die ersten drei genannten Anforderungskriterien konkurrieren untereinander und können deswegen nicht getrennt voneinander betrachtet werden. Am Ende muss also eine Optimierungsfunktion über die Partitionierung entscheiden.

Im Folgenden wird auf die Anforderungskriterien genauer eingegangen.

Echtzeitanforderungen

Bei der Nachrichtenklasse *Time-Triggered-Traffic* müssen bestimmte Nachrichten zu einem bestimmten Zeitpunkt versendet werden. Für jede Nachricht wird dabei ein Zeitfenster definiert, in dem der Teilnehmer diese Nachricht absenden darf. Die Größe des Zeitfensters wird durch den Jitter des Teilnehmers entschieden. Dabei versucht man, ein möglichst kleines Zeitfenster zu erreichen, weil große Zeitfenster die reservierte Zeit im Netzwerk erhöhen und somit für diese Zeit keine weitere Nachricht über diesen Pfad versendet werden darf.

Die Größe der erlaubten Zeitfenster entscheiden also über den erlaubten Jitter des Teilnehmers. Ein kleiner Jitter erlaubt es, kleine Zeitfenster zu verwenden und ein großer Jitter führt zu einem großen Zeitfenster. Um einen kleinen Jitter zu erreichen, muss die Partitionierung hardwarelastig erfolgen. Kann man sich einen großen Jitter erlauben, so kann softwarelastig partitioniert werden und somit Chipfläche gespart werden.

Verfügbare CPU-Ressourcen

Über die verfügbaren CPU-Ressourcen entscheidet die Anwendung, die auf der CPU ausgeführt wird. Ist der Rechenaufwand der Anwendung hoch, bleiben wenig CPU-Ressourcen für den TTEthernet-Stack übrig und es muss hardwarelastig partitioniert werden. Bei geringem Rechenaufwand können mehr Module in Software partitioniert werden um Chipfläche zu sparen.

Verfügbare Chipfläche (Kosten)

Um ein Gerät so günstig wie möglich zu realisieren, muss dieser Punkt bei der Suche nach dem optimalen Partitionierungsprofil mit berücksichtigt werden. So kann es sinnvoll sein, die Echtzeitanforderungen auf das Nötigste zu reduzieren, um möglichst viel in Software partitionieren zu können.

Benötigte Funktionalität

Um so wenig wie möglich CPU- und Hardwareressourcen zu verbrauchen, macht es in vielen Anwendungen Sinn, Funktionalität wegzulassen. Beispielsweise kann es sein, dass eine Anwendung nicht alle Nachrichtenklassen benötigt. Die entsprechenden Module müssen in diesem Fall nicht mit in die Partitionierung aufgenommen werden.

2.3 Partitionierungsauswirkungen auf Anforderungskriterien

In diesem Abschnitt werden die Module des TTEthernet-Stacks beschrieben. Außerdem wird beschrieben, welche Auswirkungen eine Hardware- oder Softwareimplementierung auf die Anforderungskriterien hat. Eine hypothetische Bewertung auf die Auswirkung wird in Tabellenform je Modul dargestellt. Konkrete Werte werden sich erst im Laufe der Entwicklung ergeben.

2.3.1 Timestamping

Dieses Modul sorgt dafür, dass eintreffende Nachrichten einen Ankunftszeitstempel erhalten. Dies ist zum Einen für die Zeit-Synchronisationspakete wichtig, weil die Zeitkorrektur anhand

dieses Zeitstempels und einer im Paket eingetragenen Zeit vorgenommen wird. Zum Anderen wird der Zeitstempel für die Nachrichtenklasse *Time-Triggered-Traffic* benötigt, um festzustellen, ob diese Nachrichten rechtzeitig eingetroffen und somit gültig sind.

HW/SW-Bewertung

Bei einer Hardwareimplementierung kann dieses Modul zwischen OSI-Layer 1 und 2 implementiert werden, so dass exakte Zeitstempel möglich sind. Dies führt dazu, dass die Synchronisation der Uhr sehr genau vorgenommen werden kann. Dies wirkt sich positiv auf die Echtzeiteigenschaften auf, so lange auch bestimmte andere Module in Hardware implementiert sind.

Eine Softwareimplementierung würde zum extremen Gegenteil führen. Zwischen dem Netzwerkinterface und der CPU liegen viele Puffer und Busse, deren Zeitverhalten stark jittert. Dies gilt insbesondere bei hoher Belastung. Hier wäre ein Zeitstempel mit einem hohen Jitter belastet, was zu einer schlecht synchronisierten Uhr und somit zu schlechten Echtzeiteigenschaften führt.

Tabelle 2.1: *Timestamping-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	+++	-	+
Softwareimpl.	- - -	+	-

2.3.2 Synchronisation

Dieses Modul nimmt anhand der eintreffenden Zeit-Synchronisationspakete eine Korrektur der Uhr vor. Im einfachsten Fall wird nur eine Offset-Korrektur vorgenommen (vgl. SAE International, 2011). Diese Korrektur wird anhand des Ankunftszeitstempel und einem Zeitstempel innerhalb des Pakets berechnet.

HW/SW-Bewertung

Nach Spezifikation muss die Offset-Korrektur der Uhr innerhalb eines Netzwerkzykluses vorgenommen werden. Dieser Zyklus ist nicht fest definiert, beträgt aber in den bisher aufgebauten Systemen oft mehrere Millisekunden. Die Offset-Korrektur muss also nicht sofort erfolgen und bringt also für die Echtzeiteigenschaften kaum Vorteile, wenn diese in Hardware implementiert wird. Hier hat eine Hardwareimplementierung nur den Vorteil, wenn CPU-Ressourcen gespart werden sollen und eine Softwareimplementierung wenn Hardware-Ressourcen gespart werden sollen.

Tabelle 2.2: *Synchronisation-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	o	-	+
Softwareimpl.	o	+	-

2.3.3 Scheduler

Dieses Modul speichert den Zeitplan, zu welchem Zeitpunkt im Zyklus welche Nachricht der Klasse *Time-Triggered-Traffic* versendet werden soll und stößt den Sendevorgang an.

HW/SW-Bewertung

Ist dieses Modul in Hardware implementiert, kann es den Sendevorgang zu exakten Zeitpunkten anstoßen. Zudem ist eine Hardwareimplementierung, wie in der verwandten Arbeit Steinhammer und Ademaij (2007) beschrieben, nicht komplex. Eine Softwareimplementierung würde dazu führen, dass zum jeweiligen Sendezeitpunkt ein Timer einen Interrupt auslöst und dabei der Sendevorgang in einer ISR angestoßen wird. Dies würde zu einen Jitter führen, der die Echtzeiteigenschaften des Teilnehmers verschlechtert.

Tabelle 2.3: *Scheduler-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	+++	- -	++
Softwareimpl.	- - -	++	- -

2.3.4 Framedroppiing

Dieses Modul stellt sicher, dass ein bestimmter Grad an Speicher für eintreffende Nachrichten mit hoher Priorität frei bleibt. Dazu muss regelmäßig der Speicherfüllstand abgefragt werden und bei hohem Füllstand alte, niedrig priorisierte Nachrichten gelöscht werden.

HW/SW-Bewertung

In der verwandten Arbeit Müller (2011) verbraucht dieses Modul die meisten CPU-Ressourcen. Hinzu kommt, dass der CPU-Bedarf bei hoher Netzwerkbelastung steigt, weil es da mehr unbearbeitete Nachrichten gibt. Bei einer Hardwareimplementierung können direkt bei der Ankunft von Nachrichten die Füllstände der Nachrichtenpuffer abgefragt werden und bei hohem Füllstand niedrig priorisierte Nachrichten verworfen werden, ohne das die CPU einen Interrupt für

diese Nachricht erhält. Eine Hardwareumsetzung dieses Moduls würde die größten Vorteile bezüglich der CPU-Ressourcen bringen.

Tabelle 2.4: *Scheduler-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	o	- -	+ + +
Softwareimpl.	o	+ +	- - -

2.3.5 Rate-Constrained

Dieses Modul ist für das Versenden von Nachrichten der Klasse *Rate-Constrained-Traffic* zuständig. Die Implementierung in (Müller, 2011) arbeitet dabei folgendermaßen:
Wenn eine RC-Nachricht ansteht, wird diese in eine Liste eingetragen. Wenn die CPU gerade „sonst nichts zu tun hat“ wird geprüft, ob eine RC-Nachricht in der Liste eingetragen ist und diese versendet, falls genug Serialisierungszeit bis zur nächsten TT-Nachricht vorhanden ist.

HW/SW-Bewertung

Das Suchen nach einem möglichen Sendezeitpunkt kann in Hardware und in Software durch einfache Subtraktionen erfolgen und verbraucht in beiden Fällen wenig Ressourcen (vgl. Steinhammer und Ademaij, 2007). Jedoch muss bei einer Softwareimplementierung, aufgrund des höheren Jitters, ein größeres Zeitfenster eingeplant werden, was dazu führt, dass weniger Bandbreite für andere Nachrichten übrig bleibt. Auf die Echtzeiteigenschaften haben beide Implementierungen keine Auswirkung. Tabelle 2.5 zeigt eine hypothetische Bewertung zu den Auswirkungen der Anforderungskriterien.

Tabelle 2.5: *Rate-Constrained-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	o	- -	+ +
Softwareimpl.	o	+ +	- -

2.3.6 Best-Effort

Dieses Modul ist für das Versenden von Nachrichten der Klasse *Best-Effort-Traffic* zuständig. Es ist dem vorangegangenem Modul sehr ähnlich. Best-Effort Nachrichten werden nach den RC-Nachrichten versendet und haben somit niedrigere Priorität. **HW/SW-Bewertung**

Ähnlich wie beim *Rate-Constrained-Modul* verbraucht die Implementierung weder in Hardware, noch in Software viel Ressourcen. Hier muss bei der Softwareimplementierung auch mit einer geringeren Restbandbreite gerechnet werden, da aufgrund des höheren Jitters größere Zeitfenster reserviert werden müssen. Auf die Echtzeiteigenschaften haben beide Implementierungen keine Auswirkung. Tabelle 2.6 zeigt eine hypothetische Bewertung zu den Auswirkungen der Anforderungskriterien.

Tabelle 2.6: *Best-Effort-Modul - hypothetische Bewertung*

	Echtzeit	HW-Ressourcen (Kosten)	SW-Ressourcen
Hardwareimpl.	o	- -	+ +
Softwareimpl.	o	+ +	- -

2.4 Zielfunktionen

Das Ziel ist es, am Ende eine Funktionen zu konstruieren, die eine Partitionierung anhand der Anforderungskriterien liefern kann. Dafür sind für jedes Modul konkrete Werte für die Auswirkung auf die Anforderungskriterien notwendig. Im vorangegangen Abschnitt wurden diese Werte hypothetisch in Tabellen dargestellt, um einen ersten Eindruck für die Auswirkungen zu vermitteln. Die Partitionierungsfunktion ist in Abbildung 2.2 dargestellt.

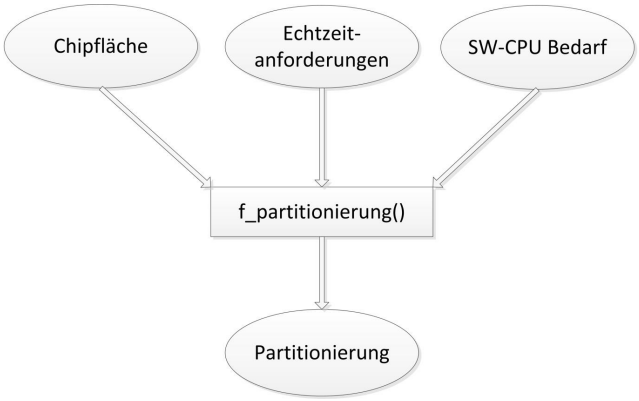

Abbildung 2.2: *Partitionierungsfunktion*

3 Risiken und Ausblick

Risiken

Ein großes Risiko ist, dass der Aufwand dieser Arbeit falsch eingeschätzt wird. Um einen vollen Baukasten zu ermöglichen, wie er in Abschnitt 2.1 beschrieben ist, ist es notwendig, jedes Modul in Hardware und in Software zu implementieren und zusätzlich Schnittstellen pro Modul in Hardware und in Software bereitzustellen.

Ein weiteres Risiko ist, dass nach der Analysephase nur noch wenige Partitionierungen übrig bleiben, die Sinn ergeben. Es macht z.B. wenig Sinn, dass *Timestamping*-Modul in Software zu realisieren und andere echtzeitkritische Module in Hardware zu realisieren. In dem Fall hätte eine Hardwareimplementierung der anderen Module nur noch wenige Vorteile.

Bei der Realisierung wird darauf geachtet, dass die Sourcen leicht portierbar sind. Es kann jedoch sein, dass einige Module speziell für Xilinx und Altera FPGA's entwickelt werden müssen. Da diese unterschiedliche Bussysteme, Memory-Controller und CPU's verwenden.

Ausblick

In Projekt 1 wird die Zeitstempeleinheit und die Synchronisation entwickelt. Die Zeitstempeleinheit wurde zwischen das PHY- und MAC-Modul (OSI-Layer 1 und 2) platziert. Eine Messung ergab, dass der Zeitstempel zu Beginn des vierten Bytes der Präambel genommen wird und um nur 1ns jittert.

Im weiteren Verlauf werden zunächst die Komponenten entwickelt, die am meisten CPU-Ressourcen verbrauchen und am effektivsten die Echtzeiteigenschaften verbessern. Es wird versucht, bestehende Softwarelösungen in dieses Projekt zu portieren und Schnittstellen zu Hardware und Software je Modul zu bilden. So soll es gelingen, am Ende einen möglichst gefüllten Baukasten, wie er in Abschnitt 2.1 beschrieben ist, zu füllen.

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

Literaturverzeichnis

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

[Badstübner 2008] BADSTÜBNER, Jens: Kollaps im Bordnetz: Schluss mit Can, Lin und Flexray. In: *KFZ-Betrieb* (2008), Nr. 17, S. 68–70

[Bruckmeier 2010] BRUCKMEIER, Robert: *Ethernet for Automotive Applications.* Vortrag. Juni 2010. – URL http://www.freescale.com/files/ftf_2010/Americas/WBNR_FTF10_AUT_F0558.pdf. – Zugriffsdatum: 2012-02-02

[D'Ambrosio und Hu 1994] D'AMBROSIO, Joseph G. ; HU, Xiaobo: Configuration-level hardware/software partitioning for real-time embedded systems. In: *Hardware/Software Codesign, 1994., Proceedings of the Third International Workshop on* IEEE (Veranst.), 1994, S. 34–41

[Groß 2012] GROSS, Friedrich: *Hardware/Software Co-Design für Real-time Ethernet basierte Steuergeräte - Verwandte Arbeiten*. August 2012. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2012-aw2/gross/bericht.pdf. – Anwendungen 2 Ausarbeitung

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

[Institute of Electrical and Electronics Engineers 2002] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems / IEEE. 2002 (IEEE Std. 1588). – Standard. – ISBN 0-7381-3369-8

[Müller 2011] MÜLLER, Kai: *Time-Triggered Ethernet für eingebettete Systeme: Design, Umsetzung und Validierung einer echtzeitfähigen Netzwerkstack-Architektur*. 2011

[PROFIBUS & PROFINET International] PROFIBUS & PROFINET INTERNATIONAL: *Profinet*. – URL http://www.profibus.com/technology/profinet. – Zugriffsdatum: 2011-02-07

[SAE International 2011] SAE INTERNATIONAL: Time-Triggered Ethernet AS6802. In: *As-2d2 Deterministic Ethernet And Unified Networking*, November 2011

[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