Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Seminararbeit

Dirk Ewerlin

## Schedulingverfahren für zeitgesteuerte Anwendungen in verteilten embedded Systemen

*Fakultät Technik und Informatik*
*Department Informatik*

*Faculty of Engineering and Computer Science*
*Department of Computer Science*

Dirk Ewerlin

# Schedulingverfahren für zeitgesteuerte Anwendungen in verteilten embedded Systemen

Seminararbeit eingereicht im Rahmen der Seminarprüfung
im Studiengang Master Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Bernd Schwarz
Zweitgutachter : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 28. Februar 2008

**Dirk Ewerlin**

## Thema der Seminararbeit

Schedulingverfahren für zeitgesteuerte Anwendungen in verteilten embedded Systemen

## Zusammenfassung

Die Seminararbeit stellt einen Ansatz zur Scheduling-Analyse einer zeitgesteuerten Anwendung in einem verteilen Netzwerk basierend auf FlexRay vor. Es wird gezeigt, dass es einen Bedarf an Methoden zur Scheduling-Analyse zeitgesteuerter verteilter Software-Hardware-Plattformen gibt und keine Lösungen verfügbar sind. Zudem wird an Hand des FlexRay ProductDay 2007 gezeigt, welche Qualifikationen zum beruflichen Einstieg in diesem Bereich notwendig sind. Darufhin wird der Aufbau eines zeitgesteuerten Systems verdeutlicht. Es wird die Zusammenarbeit zwischen dem Kommunikations-Controller und dem Microcontroller gezeigt und hieran das Zusammenspiel von Kommunkation und Taskausführung in einem zeitgesteuerten System verdeutlicht.

Abschließend wird der Ansatz zur Scheduling-Analyse des Embedded Systems Lab der Universität Linköping dargestellt. Dieser Ansatz beinhaltet eine umfassende Scheduling-Analyse und eine Bus-Analyse. Die Anwendung wird in einem Conditional Procress Graph modelliert. Mit Hilfe eines List-Scheduling-Algorithmus werden die Aktivierungszeitpunkte der Tasks und die Nachrichtenübermittlung geplant. Der Algorithmus bietet zudem einen Algorithmus zur Optimierung der Paramter des Bus-Protokolls.

# Inhaltsverzeichnis

# 1 Einleitung

Verteilte Echtzeit-Systeme in ereignisgesteuerten und zeitgesteuerten Netzwerken sind insbesondere in den Bereichen Automobilindustrie, Avionik und Medizinische Geräte zu finden. In der Automobilindustrie kommen vor allem ereignisgesteuerte Netzwerke (CAN etc.) mit ereignisgesteuerten verteilten Anwendungen zum Einsatz. Das in der Abb. 1 schematisch dargestellte Netzwerk von Steuereinheiten eines Fahrzeugs ist in zwei Teile aufgeteilt, die über einen Gateway miteinander verbunden sind. Im vorderen Teil befindet sich das Netzwerk für die Motorsteuerung (engine network), während das Netzwerk im hinteren Teil für das Brake-by-wire, ABS, ESP etc. (power train network) zuständig ist. Die Steuereinheiten eines Fahrzeugs in einem Netzwerk sind meist über das ganze Fahrzeug verteilt und besteht oft aus unterschiedlichen Hardware-Komponenten. Eine Aufteilung der Infrastruktur in zeitgesteuerte und ereignisgesteuerte Netzwerke kann durch funktionsspezifische Anforderungen an das deterministische Antwortverhalten der Netzwerkkomponenten verbunden sein.

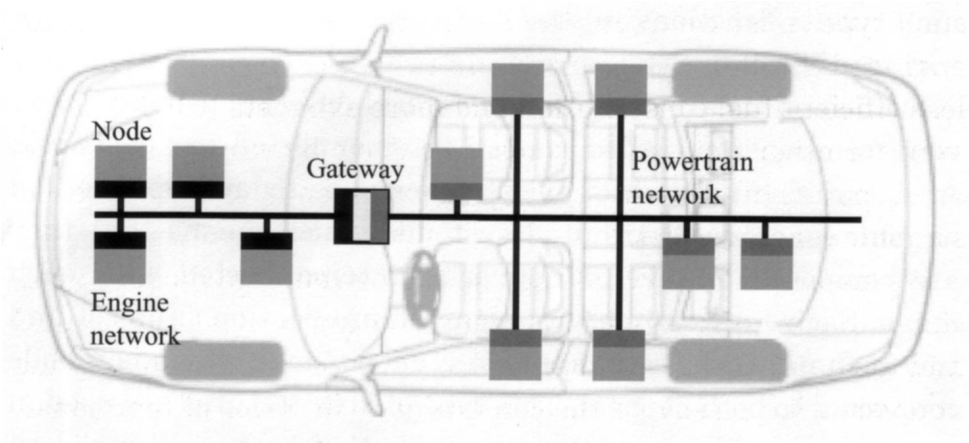


Abb. 1: Verteiltes Embedded System im Automobil

In Automobilen ist eine stetige Zunahme des Funktionsumfangs zu beobachten. Vor allem im Bereich der Fahrerassitenzsysteme werden neue Funktionen entwickelt, die durch Anwendungen und Komponenten in ein Netzwerk zu integrieren sind. Die steigende Anzahl an Anwendungen, die über einen ereignisgesteuerten Bus kommunzieren, bedeutet auch eine höhere Last auf dem Bus. Durch eine hohe Buslast wird das Antwortverhalten nicht mehr vorhersagbar und die Anwendungen des Systems weisen somit keinen Determinismus auf. Betrachtet man ein sicherheitskritische Anwendungen wie z. B. einen Bremsassistenten, so können verzögerte Nachrichten zu fatalen Folgen führen (Elektr. Automot. (2002)).

Eine Lösung hierfür ist ein zeitgesteuertes System mit einem zeitgesteuerten Bus wie FlexRay. Ein zeitgesteuertes Busssystem hat den Vorteil, dass es ein garantiertes Antwortverhalten gibt. Dieses führt zu Determinismus, da auf dem Bus jedem angeschlossenen System eine feste Zeitspanne zur Übertragung seiner Daten garantiert wird. Die Kopplung aus einer zeitgesteuerten Taskausführung und Datenübertragung liefert hohe Reaktionszeiten und Determinismus. Die Umsetzung einer solchen Architektur macht es notwendig, die Taskausführung und die Kommunikation zeitlich aufeinander abzustimmen. In der Industrie sind Werkzeuge wie der Electrobit tresos Designer im Einsatz, die die Konfiguration des zeitgesteuerten Busses FlexRay unterstützen (EB (2007)). Hierbei werden Parameter wie Slotlänge und Slotreihenfolge festgelegt. Mit Hilfe eines Schedules und Timers ist zudem eine zeitgesteuert Tasksausführung implementierbar. Auf dem Markt ist jedoch kein Werkzeug verfügbar, das sowohl die Taskausführung und Kommunikation plant als auch die Abstimmung zwischen Taskausführung und Kommunikation

in einem verteilten zeitgesteuerten System unterstützt. Im FAUST-SCV-Projekt ist bereits eine zeitgesteuerte Anwendung in einem prototypischen System umgesetzt worden. Hierzu wurde ein Software-Konzept im Rahmen einer Masterarbeit entwickelt (Sellentin (2006)). Das Konzept ist jedoch keine Lösung für komplexere Anwendungen, die Nachrichten großer Länge austauschen. Im April 2007 wurde das Forschungsprojekt TIMMO im C-Lab der Universität Paderborn gestartet, an dem u. a. Audi und Volkswagen beteiligt sind (Siemens (2007)). Ziel des Projekts ist es, eine standardisierte Infratruktur für die Behandlung von Zeitspezifikationen im Entwurfsablauf für vernetzte embedded Echtzeit-Systeme zu entwickeln. Es soll eine Beschreibungssprache für zeitliche Aspekte in der Entwicklung automobiler Steuergeräte und -netzwerke entstehen. Im Rahmen des Projekts sind bisher keine Veröffentlichungen herausgegeben worden. Dieses zeigt, dass dieses Thema in der Industrie von Interesse ist und es noch keine passenden Lösungen für diese Problematik gibt.

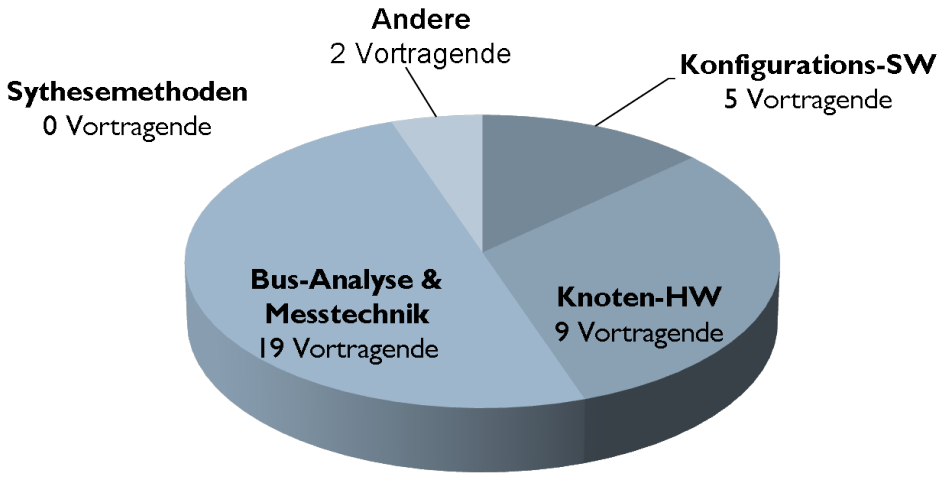


Abb. 2: Themenschwerpunkte der Vorträge auf dem FlexRay ProductDay 2007

Eine Analyse der Vortragsthemen auf dem FlexRay ProductDay 2007 zeigt die 4 Themenbereichen mit denen sich die Hersteller zur Zeit auseinandersetzen (vgl. 2, Hanser (2007))[1]. Der Bereich Bus-Analyse und Messtechnik umfasst Messgeräte und Software zur Analyse des Busses. In diesem Bereich wurden 19 von den insgesamt 35 Vorträgen gehalten. Im Bereich der FlexRay-Knoten-Hardware sind Hersteller von Kommunikations-Controllern und Microcontrollern mit integrierten Kommunikations-Controller enthalten. Zu diesem Thema waren neun Vorträge zu hören. Weitere fünf Vortäge sind zum Thema Konfigurationssoftware vorgetragen worden. Sie beinhalteten Software zur Unterstützung der Konfiguration des zeitgesteuerten Busses. Weitere zwei Vorträge wurden zu anderen Themen gehalten. Vortäge zu Methoden zur Scheduling-Analyse wurden auf dem FlexRay ProductDay 2007 nicht vorgetragen.

Aus dieser Übersicht ist zu erkennen, welche Qualifikationen für einen Informatiker in diesem Marktsegmemt zu erlangen sind. Eine berufliche Vorbereitung sollte sich daher mit den aktuell relevanten Themen auseinandersetzen. Im Bereich der ist die Verwendungen der zur Verfügung stehende Hardware interessant. Sie muss konfiguriert und programmiert werden, um im Netzwerk mit anderen Komponenten zusammenzuarbeiten. Ein weiteres Ziel ist die Anwendung der Konfigurationssoftware zur Konfiguration des Kommunikationsbusses. Diese Kenntnisse sind notwendig, um eine Anwendung au der Hardware implementieren zu können. Ausserdem sind Kenntnisse in der Bus-Analyse zur Kontrolle der entwickelten Anwendugen und der Fehlersuche sinnvoll. Ein besonderes Ziel ist die Durchdringung von wissenschaftlichen Methoden zum Timing der Taskaktivierung und der Abstimmung auf das Kommunikationssystem.

[1] Die Vorträge befinden sich auf der CD zum FlexRay ProductDay 2007

Im folgenden Kapitel zeigt den Aufbau eines verteilten zeitgesteuerten Systems in einem FlexRay-Netzwerk. Darauf folgt eine Bescheibung wie zeitgesteuerte Anwendungen auf einer verteilten zeitgesteuerten Arcitektur modelliert werden. Anschließend folgt der Ansatz zur umfangreichen Scheduling-Analyse und Bus-Analyse des Embedded Systems Lab der Universität Linköping.

# 2 Zeitgesteuerte Systeme

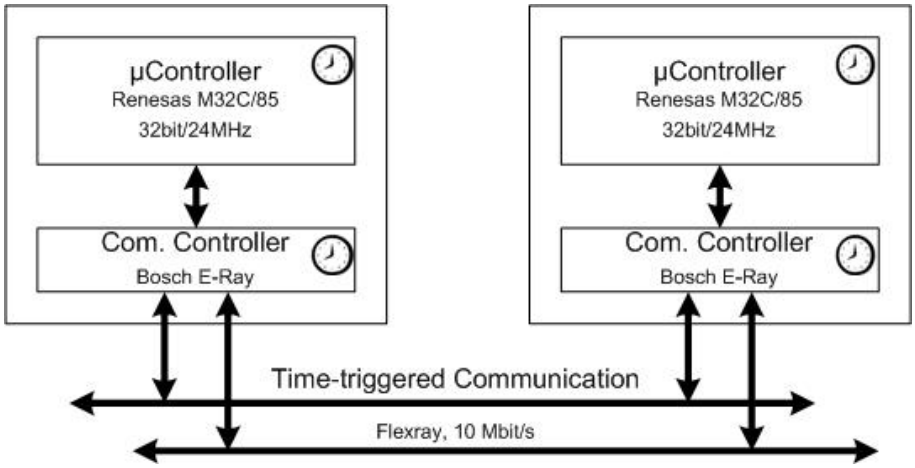


Abb. 3: Zeitgesteuertes Hard-Real-Time System mit zeitgesteuertem Kommunikationssystem

Das Senden und Emofangen von Nachrichten über den Kommuniktions-Controller erfolgt nach einer unabhängigen Zeitbasis. Die Zeitpunkte der Taskaktivierungen muss jedoch mit den Zeiten der Nachrichtenübertragungen synchronisiert werden, da die Nachrichten Ein- oder Ausgabedaten von Tasks beinhalten. Auf dem Microcontroller werden durch einen Dispatcher die einzelnen zyklischen Tasks aktiviert. Wird ein Timer des Microcontrollers verwendet, so muss die Zeit des Microcontrollers mit der Zeit des Kommunikations-Controllers synchronisiert sein, um einen rechtzeitigen Nachrichtenaustausch zu gewährleisten. Für eine genaue Synchronisation müssen die Zeiten in kleinen Abständen miteinander synchronisiert werden und eine der Zeiten angepasst werden. Da die Zeit des Kommunikations-Controllers sich mit den anderen Busteilnehmern abgleicht, kann nur die Zeit des Microcontrollers angepasst werden. Um keine Prozessorzeit für die Synchronisation zu verwenden, stellt der Kommunikations-Controller einen Timer-Interrupt bereit, der durch den Microcontroller programmiert werden kann. Die Taskaktivierung erfolgt auf Basis des Timer-Interrupts. Durch diese Synchronisation wird das Auseinanderdriften der beiden Zeitbasen verhindert.

Die Kommunikation im zeitgesteuerten System des SCV-Projekts findet über den Kommunikationsbus FlexRay statt. Sie erfolgt nach dem Time-Division-Multiple-Access (TDMA)-Verfahren. Der Kommunikationszyklus im FlexRay-Protokoll besteht aus einem statischen und einem dynamischen Segment. Im statischen Segment werden Nachrichten nach dem TDMA-Verfahren zu den festgelegten Zeitpunkten in den statischen Slots übertragen. Das Format der Nachrichten und die Länge der Slots wird vorher festgelegt. Im dynamischen Bereich können Nachrichten übertragen werden, die nicht zuvor festgelegt wurden. In diesem Bereich wird nur die Buszugriffsreihenfolge, aber nicht die Länge des Buszugriffs festgelegt. Der Zeitplan des FlexRay-Systems, der die Übertragungszeitpunkte der Nachrichten vorschreibt, muss auf alle Bus-Teilnehmer abgestimmt und für alle identisch sein. Dadurch ist ein höhere Spezifikationsaufwand während des Entwurfs des Systems erforderlich. Die Abbildung 4 stellt einen festen Kommunikationszyklus dar, nach dem der Buszugriff erfolgt. Das dargestellte System besteht

aus fünf Knoten, die über zwei FlexRay-Kanäle kommunizieren. Jedem dieser Knoten ist ein Slot aus dem statischen Segment zugeordnet, wobei die Kommunikation je nach Konfiguration auf beiden Kanälen gleichzeitg oder nur auf einem Kanal stattfinden kann. In diesem Beispiel sind zusätzlich Nachrichten im dynamischen Segment eingetragen. Im statsichen Segment kann jeder Bus-Teilnehmer den ihm zugeteilten Slot exlusiv nutzen. Die Zeiten der Teilnehmer müssen synchronisiert werden, um den exklusiven Zugriff zu garantieren. Am Ende jedes Komunikationszyklus werden die Uhren der Teilnehmer synchronisiert.

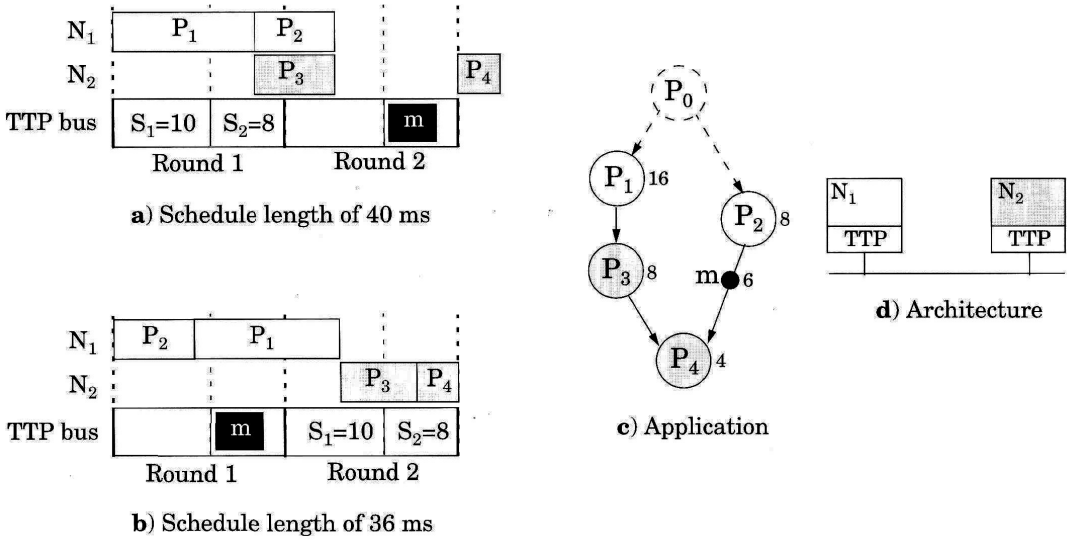


Abb. 5: Optimierung der Ausführungszeit durch Anpassung der Taskreihenfolge

Die Abstimmung der Kommunikation und der Taskaktivierung ist bei der Realisierung eines verteilten zeitgesteueren Systems nicht das einzige Problem. Zeitgesteuerte Anwendungen in sicherheitkritischen Bereichen unterliegen meist zeitlichen Vorgabe wie einer Deadline, die vorgibt, nach welcher Zeitdauer die Anwendung abgeschlossen sein muss. Bei der Umsetzung einer Anwendung wird daher versucht, eine möglichst kurze Ausführungsdauer zu generieren. Die Ausführungsreihenfolge der Tasks auf einem Knoten und die Zuordnung von Knoten zu den Slots einer TDMA-Runde beeinflusst die Ausführungsdauer einer Anwendung. Die Optimierung dieser Zuordnungen unterstützt die Erzeugung eines minimalen Ausführungsdelays. Die Abbildung 5 veranschaulicht die Problematik der Taskreihenfolge. Der Graph, der in Abb. 5 c dargestellt wird, zeigt vier Tasks, die auf zwei Knoten ausgeführt werden. Die Task $P_2$ sendet Daten an die Task $P_4$ in der Nachricht $m$. Unter Abb. 5 a werden auf dem Knoten $N_1$ die Tasks $P_1$ und $P_2$ nacheinander ausgeführt. Durch das vorgegebene TDMA-Schema kann die von $P_2$ erzeugte Nachricht erst im zweiten Slot der Runde 2 übertragen werden. Die Task $P_4$ kann erst nach Ankommen

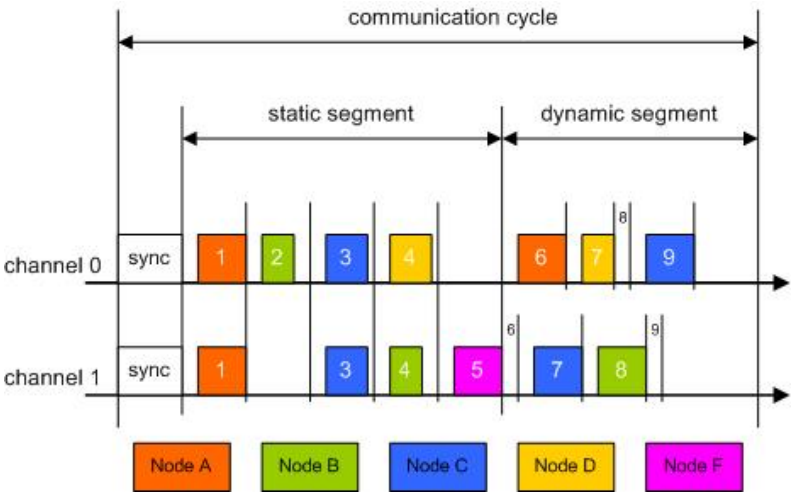


Abb. 4: FlexRay TDMA-Runde

der Nachricht gestartet werden. Hieraus ergibt sich ein Delay von 40 ms für diese Anwendung. Unter Abb. 5 b wird die gleiche Anwendung mit einer anderen Taskreihenfolge dargestellt. Auf dem Knoten $N_1$ wird erst die Task $P_2$ und dann die Task $P_1$ ausgeführt. Hierdurch kann die Nachricht $m$ bereits im zweiten Slot der ersten Runde gesendet werden. Das Delay verkürzt sich somit auf 36 ms. Durch die Anpassung der Slotzurodnung im TDMA-Schema kann ebenfalls eine Verkürzung des Anwendungsdelays erreicht werden.

In der Praxis ist zur Zeit nur der X5 von BMW zu finden, in dem FlexRay als Kommunikations-Protokoll zum Einsatz kommt. An der HAW Hamburg ist in prototypischen Umgebung des FAUST-SVC-Projekts eine Fahrerassistenz-Anwendung in einem FlexRay-Netzwerk umgesetzt worden. Der Einsatz im HAWKS-Racing ist zudem vorgesehen.

# 3 Anwendungsmodellierung eines zeitgesteuerten Systems

Der Entwurf einer zeitgesteuerten Anwendung für ein verteiltes embedded System lässt sich im Allgemeinen in sechs Phasen unterteilen (s. Abb. 6, Sellentin (2006)).

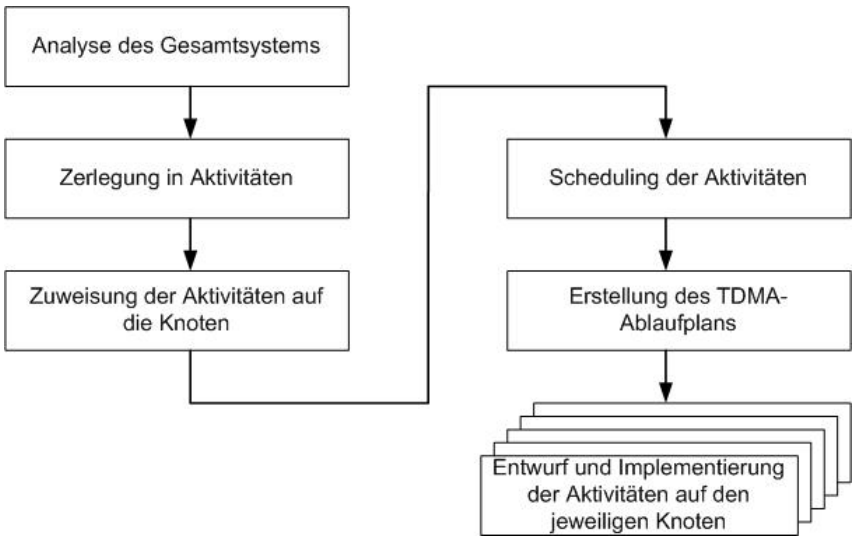


Abb. 6: Entwurfsschritte einer zeitgesteuerten Anwendungen (Sellentin (2006))

1. **Analyse des Gesamtsystems:**
   In der ersten Phase ist die Funktionalität des Gesamtsystems und die dafür notwendige Hardware zu analysieren.
2. **Zerlegung der Aktivitäten:**
   Die zweite Phase umfasst die Zerlegung der Gesamtfunktionalität in mehrere Tasks. Die Kommunikation und die Abhängigkeiten zwischen den Tasks müssen spezifiziert werden. Anschließend muss die Datenmenge, die übertragen werden soll, und die Ausführungszeiten der Tasks analysiert werden.
3. **Zuweisung der Aktivitäten auf die Knoten:**
   In der dritten Phase werden die Tasks den Knoten zugeordnet (Mapping). In einem verteilten embedded System sind Tasks oft von der Hardware im Bezug auf die Schnittstellen und die angeschlossene Peripherie der Mikrocontroller abhängig. Daher lässt sich nur ein Teil der Tasks frei verteilen.

4. **Scheduling der Aktivitäten:**
   Die vierte Phase umfasst die Planung der Aktivierungszeitpunkte der Tasks auf den einzelnen Knoten. Diese Planung kann von Hand oder unterstützt durch entsprechende Algorithmen durchgeführt werden.

5. **Erstellung eines TDMA-Ablaufplans:**
   Die fünfte Phase umfasst die Erstellung eines TDMA-Ablaufplans. Es wird festgelegt, welche Daten zu welchen Zeitpunkten übertragen werden dürfen. Für die Festlegung muss die Datenmenge, sowie die Aktivierungszeitpunkte, die maximale Ausführungszeit und die Periode der Tasks bekannt sein.

6. **Entwurf und Implementierung der Aktivitäten auf den jeweiligen Knoten:**
   In der letzten Phase werden die Aktivitäten entworfen und auf den Knoten implementiert. Die Entwicklung der Aktivitäten muss sich hierbei an die in den vorherigen Phasen festgelegten Rahmenbedingungen halten.

Der vorgestellte Ansatz ermöglicht die Modellierung einer zeitgesteuerten Anwendung auf einem zeitgesteuerten Bus. Hierbei werden zuerst die Ausführungszeitpunkte der Tasks festgelegt und anschließend der TDMA-Ablaufplan erstellt. Dieses ermöglicht nicht die optimale Einplanung von Tasks, die Nachrichten von anderen Tasks über den BUS erhalten. Eine optimale Task- und Nachrichteneinplanung erhält man durch parallele Bearbeitung beider Aufgaben. Ein Ansatz hierzu wird im folgenden Abschnitt dargestellt.

# 4 Scheduling-Analyse zeitgesteuerter Systeme und Analyse des zeitgesteuerten Busses

Im folgenden Abschnitt wird ein Ansatz zur Scheduling-Analyse vogestellt, der u. a. durch die Professoren Pop und Eles des Embedded Systems Lab der Universität Linköping entwickelt wurde (ESLAB (2007), Pop u. a. (2004), Eles u. a. (2000), Pop u. a. (2006)). Dieses ist der einzige in einem Buch und mehreren Aufsätzen umfangreich dokumentierte Ansatz, der zur Zeit zu finden ist. Der vorgestellte Ansatz beinhaltet eine umfassende Analyse des Gesamtsystems. Der Ansatz geht von einem zeitgesteuerten Kommunikationsbus aus, der nach dem TDMA-Verfahren arbeitet. Es wird von einer verteilten Architektur bestehend aus programmierbaren Prozessoren und dedizierter Hardware ausgegangen. Die Komponenten sind über mehrere Busse miteinander verbunden. Jeder programmierbare Prozessor kann eine Task zur Zeit ausführen während die dedizierte Hardware Tasks parallel verarbeiten kann. Der Bus kann nur eine Übertragung zur Zeit ausführen. Die Tasks sind Prozessoren zugeordnet. Für jede Task ist die Worst-Case-Execution-Time (WCET) auf der entsprechenden Ressource bekannt. Für eine Kommunikations-Task ist die zu übertragende Datenmenge bekannt. ( Pop u. a. (2004)) Ziel ist es, das Worst-Case-Delay, das das System zur kompletten Ausführung benötigt, zu minimieren, ein statischen Schedule zu generieren und die Parameter des Kommunikationsprotokolls so anzupassen, dass ein definiertes Delay garantiert wird.

Die Anwendung wird in diesem Ansatz als Conditional Process Graph (CPG) modelliert, wie er in Abb. 7 dargestellt ist. Der Graph ist polar, azyklisch und gerichtet. Polar bedeutet, dass es eine Quelle und eine Senke gibt, die die erste und letzte Task einer Anwendungsrunde repräsentieren. Es handelt sich dabei meist um Dummy-Tasks ohne Ausführungszeit und ohne Ressourcenzuordnung. Im Beispiel sind dieses die Tasks $P_0$ und $P_{15}$. Tasks werden als Knoten dargestellt, neben denen ihre WCET notiert wird. Die Einfärbung der Knoten repräsentiert dabei die Zuordnung zu einer Ressource. Zwischen diesen beiden Dummy-Tasks gibt es eine azyklische Folge von Tasks, deren Abhängigkeiten durch gerichtete Kanten dargestellt werden.

Die Kommunikation zwischen zwei Tasks auf unterschiedlichen Prozessoren wird als Kommunikationstask abgebildet. Diese wird durch einen schwarzen Kreis auf einer Kante dargestellt (in der Abbildung z. B. zwischen $P_2$ und $P_3$). Neben einem Kreis wird die zu übertragende Datenmenge notiert. Gerichtete Kanten können mit Konditionen versehen werden (in der Abbildung auf der Kante zwischen $P_1$ und $P_2$/$P_{14}$). Sie werden nur dann ausgeführt, wenn die Kondition wahr ist. In einem CPG wird davon ausgegangen, dass es sich um Tasks mit der selben Periode handelt. Tasks mit unterschiedlichen Perioden werden gehandhabt, indem mehrere Instanzen der Tasks erzeugt werden und ein CPG mit der Periode des kleinsten gemeinsamen Vielfachen erzeugt wird.

Beim Scheduling einer Anwendung besteht die Schwierigkeit darin, dass bei jeder Ausführung des Graphen nur eine Untermenge an Tasks ausgeführt wird, da einige Tasks aufgrund der Konditionen nicht ausgeführt werden. Der verwendete Scheduling-Algorithmus basiert auf einer List Scheduling Heuristik (Abb. 8). Es wird dabei eine Tabelle, der Schedule Table, erzeugt. Auf den einzelnen Komponenten befindet sich ein minimaler Scheduler, der anhand dieses Schedule Tables die Taskaktivierung vornimmt.

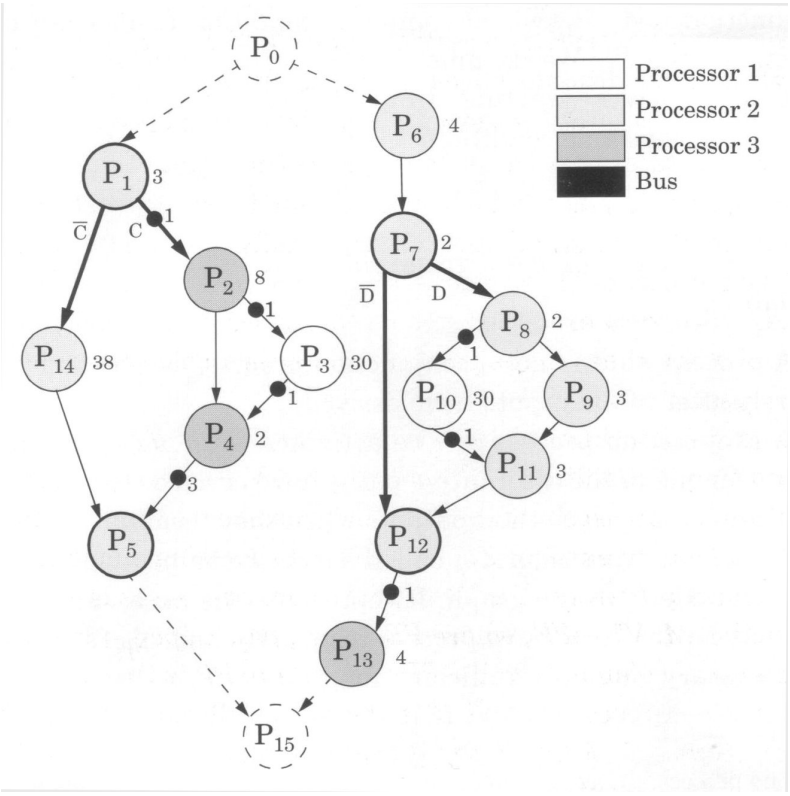


Abb. 7: Beispielanwendung modelliert als Conditional Process Graph (Pop u. a. (2004))

Der List Scheduling Algorithmus beginnt zum Zeitpunkt Null. Als Eingangsinformationen werden eine ReadyList und die bekannten Konditionen (KnownConditions) benötigt. Die ReadyList beinhaltet die Prozesse, die zum aktuellen Zeitpunkt ausführbar sind. Ausführbar sind Prozesse, die keine Vorgänger haben und denen alle Eingangsinformationen zur Verfügung stehen. Der Algorithmus beinhaltet eine Schleife, die mit dem Aktualisieren der ReadyList beginnt. Anschließend wird für jede Ressource, sofern sie nicht belegt ist, mit Hilfe der GetReadyProcess-Funktion eine Task aus der ReadyList entnommen. Die GetReadyProcess-Funktion enthält eine Prioritätsfunktion, die die Ausführungsreihenfolge konkurrierender Tasks bestimmt. Die Prioriätsfunktion wird später genauer beschrieben. Gibt es für die aktuelle Ressource eine Task, die ausgeführt werden soll, so wird diese unter Verwendung der Funktion Insert in den Schedule Table eingetragen. Als Informationen werden die Task mit deren Aktivierungszeitpunkt und die bekannten Konditionen, unter denen dieser Eintrag gültig ist, eingetragen. Erzeugt die eingeplanten Tasks einen Konditionswert, so wird die List Scheduling Funktion rekursiv aufgerufen. Der rekursive Aufruf wird für den wahren und falschen Konditionswert ausgeführt, um beide Fälle abzudecken. Die KnownConditions werden bei dem Aufruf um die entsprechenden Konditionswerte erweitert. Die aktuelle Zeit wird anschließend auf den nächsten Zeitpunkt, zu dem

eine Task beendet wird, gesetzt. Die Schleife wird wiederholt bis alle Tasks des alternativen Pfades eingeplant wurden.

Die Einplanung der Nachrichten findet während des ListSchedulings statt. Der Kommunikationsbus wird dabei als Ressource betrachtet, auf der die Kommunikationstask eingeplant werden. Für die Einplanung wird eine MessageSchedule Funktion verwendet. Diese Funktion beachtet die Besonderheiten des Kommunikationsprotokolls wie die Slotzuordnung und die Slotlänge. Ist z. B. ein Slot bereits mit Nachrichten gefüllt, so muss die Nachricht in der folgenden Kommunikationsrunde eingeplant werden.

```
ListScheduling(CurrentTime, ReadyList, KnownConditions)
1  repeat
2      Update(ReadyList)
3      for each processing element PE do
4          if PE is free at CurrentTime then
5              P_i = GetReadyProcess(ReadyList)
6              if there exists a P_i then
7                  Insert(P_i, ScheduleTable, CurrentTime, KnownConds)
8                  if P_i is a disjunction process then
9                      C_i = condition calculated by P_i
10                     ListScheduling(CurrentTime,
11                         ReadyList ∪ ready nodes from the true branch,
12                         KnownConditions ∪ true C_i)
13                     ListScheduling(CurrentTime,
14                         ReadyList ∪ ready nodes from the false branch,
15                         KnownConditions ∪ false C_i)
16                 end if
17             end if
18         end if
19     end for
20     CurrentTime = earliest time when a scheduled process terminates
21 until all processes of this alternative path are scheduled
end ListScheduling
```

Abb. 8: ListScheduling-Funktion (Pop u. a. (2004))

Die Prioritätsfunktion, die in der GetReadyProcess Funktion zur Anwendung kommt, bestimmt die Ausführungsreihenfolge konkurrierender Tasks. Die Priorität eines Prozesses wird bestimmt, indem die Prozesse betrachtet werden, die auf die einzuplanende Task folgen. Eine zeitlich lange Folge aus Tasks, die auf anderen Ressourcen als die, auf der die einzuplanende Task läuft, führt zu einer höheren Priorität. Die Priorität bildet sich zusätzlich zu den Tasks auf anderen Ressourcen auch durch die Kommunikationszeit. Die Zeit der Nachrichtenübermittlung beinhaltet hierbei die Zeit, die sie benötigt bis die Daten beim Empfänger vorliegen.

Neben der Generierung eines Schedules mit minimalen Delay, zeigt der Ansatz auch die Optimierung der Parameter des Bus-Protokolls. Die Anpassung der Slotgröße und die Reihenfolge werden zur Optimierung verwendet. Die Abb. 9 zeigt ein Beispiel wie sich durch die Slotreihenfolge und Slotgröße das Delay der Anwendung verkürzen lässt. Eine Anpassung der Slotgröße ist dann sinnvoll, wenn es viele Nachrichten gibt, die erst eine TDMA-Runde später gesendet werden müssen, da der Slot bereits voll ist. Kann eine Nachricht früher gesendet werden, so kann dieses zu einem kürzerem Delay führen, wenn durch diese Änderung andere Nachrichten nicht wieder das Delay vergrößern. Die Anpassung der Slotreihenfolge wird zusätzlich zur Optimierung verwendet. Durch die Vertauschung der Zuordnung kann ein kürzeres Delay erzeugt werden. Der hier vorgstellte Ansatz nutzt zur Optimierung der beiden Paramerter Feedback aus dem Scheduling-Algorithmus. Passt z. B. eine Nachricht nicht mehr in einen Slot, so wird eine größere Slotgröße aus dem Algorithmus heraus vorgeschlagen. Die vorgeschlagenen Slotgrößen werden nach dem ersten Durchgang getestet. Die Konfiguration mit dem kürzesten Delay gibt anschließend die Parameter für das Bus-Protokoll vor.

Die Qualität des Scheduling-Algorithmus wurde in einem Experiment des Embedded Systems Lab überprüft. Im Test wurden Anwendungen mit 2, 4, 6, 8 und 10 Knoten getestet. Auf jedem

Knoten wurden 40 Tasks eingeplant. In jeder Dimension wurden dabei 30 Anwendungen erzeugt. Die mit dem vorgstellten Algorithmus erzeugten Schedules weichen von der optimalen Lösung um bis zu 1,5 % ab, wobei die Ausführungszeit bei 400 Tasks unter 400 ms lag. Bei der Optimierung der Bus-Parameter lag die Ausführungszeit für 400 Prozesse bei ca. 8 Sekunden. Die Abweichung von der optimalen Lösung lag bei 40 Tasks unter 2%, während sie bei 400 Tasks ca. 10% betrug (Pop u. a. (2004)).

Zu dem hier vorgestellten Ansatz sind auch Erweiterungen für ereignisgesteuerte Taskausführung und Kommunikation vom Embedded Systems Lab erarbeitet worden. Sie nutzen die volle Funktionalität des FlexRay-Protokolls. Die zeitgesteuerten Tasks werden durch einen statischen Schedule eingeplant, bei dem die Ausführung von ereignisgesteuerten Tasks berücksichtigt wird. Die zeitgesteuerten Nachrichten werden im statischen Segment des FlexRay-Protokolls eingeplant. Für die ereignisgesteuerten Tasks wird eine Response-Time-Analyse durchgeführt. Die Nachrichten der ereignisgesteuerten Tasks werden im dynamischen Segment übertragen.

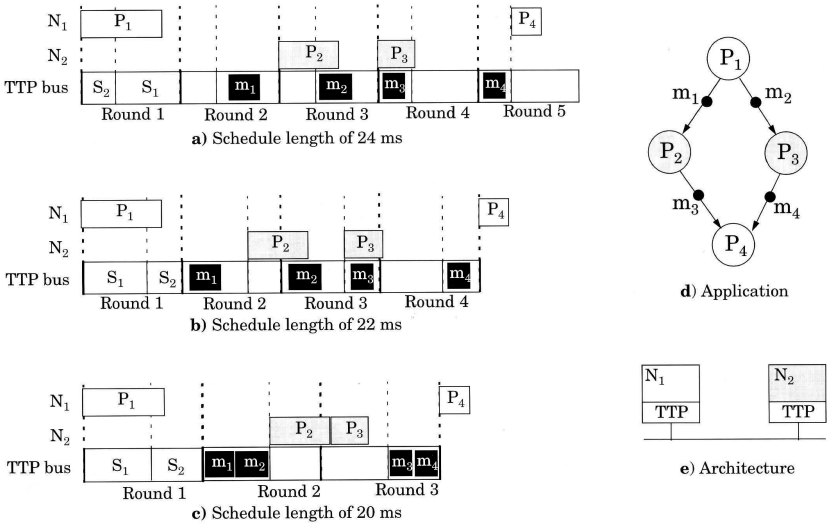


Abb. 9: Delay-Verkürzung durch Bus-Optimierung (Pop u. a. (2004))

# 5 Thesis Outline

Das Ziel der Master Thesis umfasst die Scheduling-Analyse für zeitgesteuerte Anwendungen in verteilten embedded Systemen in einem FlexRay-Netzwerk. Hierzu gehören die folgenden Arbeitsschritte:

- Untersuchung und Aufbereitung der Scheduling-Analyse des Emebedded Systems Lab der Universität Linköping. Darunter fällt die Zusammenstellung der Algorithmen-Komponenten in Pseudocode und UML-Grafiken sowie die Abbildung auf ein Software-Paket soweit dieses noch nicht durch das Embedded Systems Lab geschehen ist.
- Abgrenzung gegenüber vergleichbaren Verfahren.
- Anwendung des Ansatzes auf die Task-Ausführung in einer SCV-Anwendung auf der 2-Knoten-Konfiguration der Masterarbeit von Herrn Sellentin und Vergleich der Ergebnisse und Bus-Analyse.
- Ersetzung eines der verfügbaren Fahrerassistenzsysteme, verteilt über drei Knoten zur Ersetzung der PC104/QNX-Threads Konfiguration durch Renesas-$u$Controller/TT-Software Konfiguration.
- Ein weiteres mögliches Arbeitspaket ist die Betrachtung des Mapping-Aspektes

# 6 Zusammenfassung

Die Seminararbeit befasst sich mit der Scheduling-Analyse zeitgesteuerter Anwendungen in verteilten embedded Systemen. Es wird der Aufbau und die Anwendungsmodellierung eines zeitgesteuertes Systems in einem FlexRay-Netzwerk am Beispiel des FAUST-SCV-Projekts dargestellt. Anschließend wird der Ansatz zur Scheduling-Analyse des Embedded Systems Lab der Universität Linköping vorgestellt.

Ereignisgesteuerte verteilte Anwendungen sind u. a. in der Automobilindustrie zu finden. Steigende Funktionalität erfordert zusätzliche Systeme im ereignisgesteuerten Netzwerk. Die dadurch steigende Buslast führt zu einem nicht mehr vorhersagbaren Antwortverhalten und damit zu fatalen Folgen. Ein System mit zeitgesteuerten Kommunikationsbus wie FlexRay liefert ein garantiertes Antwortverhalten. Die Kopplung aus einer zeitgesteuerten Datenübertragung und Taskaktivierung liefert schnelle Reaktionszeiten und ein vorhersagbares Verhalten der Anwendungen. Zur Realisierung zeitgesteuerter Anwendungen bietet der Markt Konfigurationssoftware für den zeitgesteuerten Bus und Microcontroller zur zeitgesteuerten Task-Ausführung. Auf dem Markt sind jedoch keine Hilfsmittel verfügbar, die die Abstimmung zwischen der zeitgesteuerten Taskausführung und Datenübertragung unterstützen.

Eine Analyse der Vorträge und Aussteller auf dem FlexRay-ProductDay 2007 verdeutlichte noch mal, dass keine Produkte zur Scheduling-Analyse auf dem Markt verfügbar sind. Aus dieser Analyse lässt sich ableiten, welche Ziele für einen Informatiker zum Einstieg in diesen Markt von Interesse sind. Ziele sollten die Anwendung der Knotenhardware sowie die Anwendung von Techniken zur Busanalyse sein. Ein weiters Ziel ist die Durchdringung wissenschaftlicher Methoden zum Timing der Taskaktivierung und der Abstimmung mit dem Kommunikationssystem.

Der Aufbau eines zeitgesteuerten Systems im SCV-Projekt besteht aus Microcontrollern, die über einen Kommunikations-Controller an den FlexRay-Bus angeschlossen sind. Auf den Komponenten des gesamten Systems muss eine einheitliche Zeitbasis hergestellt werden. Das FlexRay-Protokoll bietet Algorithmus zur Synchronisation der Kommunikations-Controller-Uhren. Zur Abstimmung der Microcontroller-Uhren wird die Taskaktivierung durch die Kommunikations-Controller angestoßen. Die Anwendungsmodellierung zeitgesteuerter Anwendungen findet in sechs Phasen statt. Sie umfassen die Analyse des Gesamtsystems, die Zerlegung der Aktivitäten, die Zuweisung der Aktivitäten zu den Knoten, das Scheduling der Aktivitäten sowie die Erstellung des TDMA-Plans und den Entwurf der Aktivitäten.

Das Embedded Systems Lab der Unviversität Linköping hat einen Scheduling-Analyse Ansatz für zeitgesteuerte Anwendungen entwickelt. Dieser führt eine umfassende Analyse des Gesamtsystems durch. Es wird eine List Scheduling Heuristik verwendet, die sowohl die Tasks auf den einzelnen Ressourcen als auch die Nachrichten auf dem Bus einplant. Zur Einplanung konkurrierender Tasks wird eine Prioritätsfunktion verwendet, die das parallele Arbeiten fördert. Bei der Einplanung der Nachrichten werden die Eigenschaften des Kommunikationsbusses berücksichtigt, wobei die Parameter bei der Optimierung während der Scheduling-Analyse optimiert werden. Es können Slotgrößen und die Reihenfolge der Slotzuordnung optimiert werden, um ein geringeres Delay der Anwendung zu erhalten.

Der hier vorgestellte Ansatz zeigt die Durchführung der Scheduling-Analyse für zeitgesteuerte Anwendungen in verteilten Systemen. In der Masterarbeit sollte dieser Ansatz daher umfangreich untersucht werden und gegenüber anderen Ansätzen abgegrenzt werden. Zusäzlich ist eine Anwendung mit Hilfe dieses Ansatzes zu implementieren und mit den Ergebnissen der Masterarbeit von Herrn Sellentin zu vergleichen.

# Literatur

[Ahmed Amine Jerraya 2005] AHMED AMINE JERRAYA, Wayne W.: *Multiprocessor System-on-Chips*. Morgan Kaufmann, 2005. – ISBN 0-12385-251-X

[Andrew Tannenbaum 2003] ANDREW TANNENBAUM, Marten van S.: *Verteilte Systeme: Grundlagen und Paradigmen*. Pearson Studium, 2003. – ISBN 3-8273-7057-4

[EB 2007] ELECTROBIT CORPORATION: *EB tresos Designer*. 2007. – URL http://www.elektrobit.com/index.php?770

[Elektr. Automot. 2002] HEINECKE, Prof. H. ; SCHEDL, Dr. A.: FlexRay - ein Kommunikationssystem für das Automobil der Zukunft. In: *Elektronik Automotive* September 2002 (2002), S. 36–45

[Eles u. a. 2000] ELES, P. ; DOBOLI, A. ; POP, P. ; PENG, Z.: Scheduling with Bus Access Optimization for Distributed Embedded Systems. In: *IEEE Transactions on VLSI Systems* 8 (2000), oct, Nr. 5, S. 472–491. – URL http://www2.imm.dtu.dk/pubdb/p.php?4620

[ESLAB 2007] EMBEDDED SYSTEMS LAB ; UNIVERSITY LINKÖPING: *Design Environment for Real-Time Embedded Systems in Control-Related Applications*. 2007. – URL http://www.ida.liu.se/labs/eslab/new_research/design_control_related.shtml

[Hanser 2007] HANSER VERLAG: *FlexRay ProductDay 2007*. 11 2007. – URL http://www.flexray-productday.de

[Kopetz 1997] KOPETZ, Hermann: *Real-Time Systems: Design Principles for Distributes Embedded Applications*. Kluwer Academic, 1997. – ISBN 0-7923-9894-7

[Pop u. a. 2004] POP, Paul ; ELES, Petru ; PENG, Zebo: *Analysis and Synthesis of Distributed Real-Time Embedded Systems*. Kluwer Academic Publishers, 2004. – ISBN 1-4020-2872-5

[Pop u. a. 2006] POP, Traian ; POP, Paul ; ELES, Petru ; PENG, Zebo ; ANDREI, Alexandru: Timing Analysis of the FlexRay Communication Protocol. In: *18th Euromicro Conference on Real-Time-System (ECRTS'06)* 0 (2006), S. 203–216. – ISSN 1068-3070

[Sellentin 2006] SELLENTIN, Jörn: *Ein zeitgesteuertes, verteiltes SW-Konzept implementiert auf FlexRay-Komponenten für ein fahrerloses Transportsystem*, HAW-Hamburg, Diplomarbeit, 2006

[Siemens 2007] SIEMENS ; UNIVERSITÄT PADERBORN: *Forschungsprojekt TIMMO*. 2007. – URL http://www.c-lab.de/de/forschungsprojekte/timmo/index.html

# Abbildungsverzeichnis