VERTEILT UND DOCH ZUSAMMEN – ZUM STEUERSYSTEM VON HYBRID III

Hans Otto Trutmann

Institut für Technische Informatik und Kommunikationsnetze TIK
Eidgenössische Technische Hochschule Zürich ETH
Gloriastr. 35, 8092 Zürich
email: trutmann@tik.ee.ethz.ch

Abstract: The ETH project Hybrid III, targeting the fuel efficiency of private vehicles, introduced a number of novel components into the drive train. Control system design had to take into account two very different modes of operation: components running separately on their own test beds during development on the one hand or assembled and driving a real car on the other hand. To this end a flexible multiprocessor solution that mirrors the system structure was proposed. Flexibility was a key issue also regarding data I/O with a miniature cabling system and software generators easing the task of adapting to ever changing environments. The entire functional program code was generated automatically from CASE specifications. Designers of the drive train elements worked with a dedicated graphical software tool and were thus able to concentrate on their actual work instead of getting involved with control system details.

Keywords: automotive, computer control, embedded system, realtime, specification.

1. EINLEITUNG

Das Institut für Technische Informatik und Kommunikationsnetze ist verantwortlich für die Informatikmittel im Projekt Hybrid III. Nebst der Steuerung selbst und von Teilen der Software bedeutete das auch die Bereitstellung einer geeigneten Entwicklungsumgebung und die Begleitung des Projekts von den Anfängen bis zum Betrieb.

Das *Steuersystem* selbst musste sehr flexibel konzipiert werden; die Anzahl der Ein- und Ausgabesignale, deren zeitliche Anforderungen, die benötigte Rechenleistung waren bei Projektbeginn allesamt unbekannt. Dazu kam, dass während der Entwicklung jede Antriebskomponente sowohl alleine wie auch in verschiedenen Kombinationen mit anderen – am Schluss dann als Gesamtsystem – betrieben wird.

Von der *Arbeitsumgebung* wurde vor allem gefordert, dass sie die an der Entwicklung der Antriebskomponenten beteiligten Spezialisten von Programmieraufgaben entlastet. Als Nebeneffekt sollten so moderne Werkzeuge und neue Erkenntnisse im Bereich von industriellen Informatikanwendungen Eingang in die am Projekt beteiligten Institute finden. Eine wichtige Voraussetzung dafür war eine einheitliche Arbeitsumgebung mit identischen, vernetzten Arbeitsplatzrechnern, die einerseits als selbständige Entwicklungsplätze für Elemente der Gesamtsteuerung ausgerüstet sind, über die andrerseits auch alle zentralen Dienste und die Projektdaten koordiniert werden können.

2. STEUERSYSTEM

Das Antriebssystem besteht aus fünf Hauptkomponenten und genau so ist auch die Steuerung aus einzelnen, lose kooperierenden Blöcken zusammengesetzt. Diese sogenannten Cluster brechen das Ganze in Teilsysteme auf, die ihrerseits lokal eingrenzbare Abläufe selbsttätig ausführen können. Konstruktiv zusammengehörige Gruppen sind auch in der Steuerung vereint, wodurch die getrennte Entwicklung und separate Tests einzelner Baugruppen innerhalb der endgültigen Struktur erst möglich werden. Cluster ermöglichen damit auch, das System hierarchisch zu zerteilen.

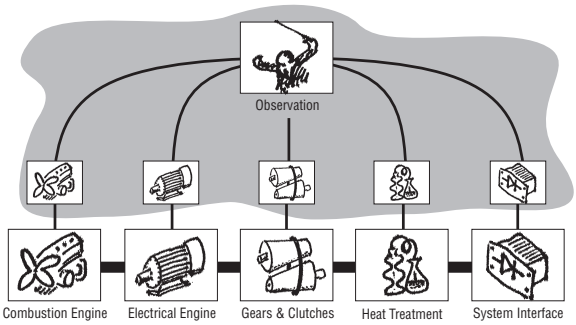

Abb. 1: Logische Struktur des Steuersystems

Ein zusätzlich eingefügter Cluster mit höheren Steuerfunktionen überwacht und koordiniert die einzelnen Teilsysteme. Die Ein- und Ausgaben erfolgen über

dezentrale, für den Einsatz im Fahrzeug miniaturisierte Anschlusseinheiten (und einen weiteren Cluster, der mit diesen Aufgaben betraut ist). Diese Struktur bildet die Basis für die Arbeit der einzelnen Ingenieurgruppen, die nun damit ihren Teil des Antriebssystems steuern können.

Die Cluster von Hybrid III heissen:

O Observation
C Combustion
E Electrical
M Mechanical
H Heat Treatment
S System Interface
IO Input & Output.

3. KONZEPTE

Die oben skizzierte Aufteilung in einzelne Cluster und deren Verbindung untereinander können in der Implementation nicht direkt übernommen werden; die verwendeten Prozessorknoten haben zu wenige Verbindungen, um eine Sternstruktur mit so vielen Partnern aufzubauen. In der Implementation gewährleisten zwei gegenläufige Ringe, dass alle Partner auf zwei Arten erreichbar sind. In diesen Ring einbezogen ist auch der Cluster **IO**, der zudem über einen Broadcast-Mechanismus allen anderen Clustern die Eingabedaten gleichzeitig übermitteln kann.

Die grosse Anzahl von Sensoren und Aktoren (ca. 150 Sensoren und 80 Aktoren, auf den Prüfständen sogar noch mehr) verleihen guten Lösung für die Datenerfassung und -ausgabe einiges Gewicht. Hardwareseitig kommt ein auf dem eigentlichen Peripheriebus aufsetzender Mezzaninbus, zusammen mit dezentralen, eigens entwickelten Anschlusseinheiten (Satelliten) zum Einsatz. Dazu gehören auf der Softwareseite massgeschneiderte Algorithmen, die gemäss einer Spezifikation für die jeweilige Installation generiert werden.

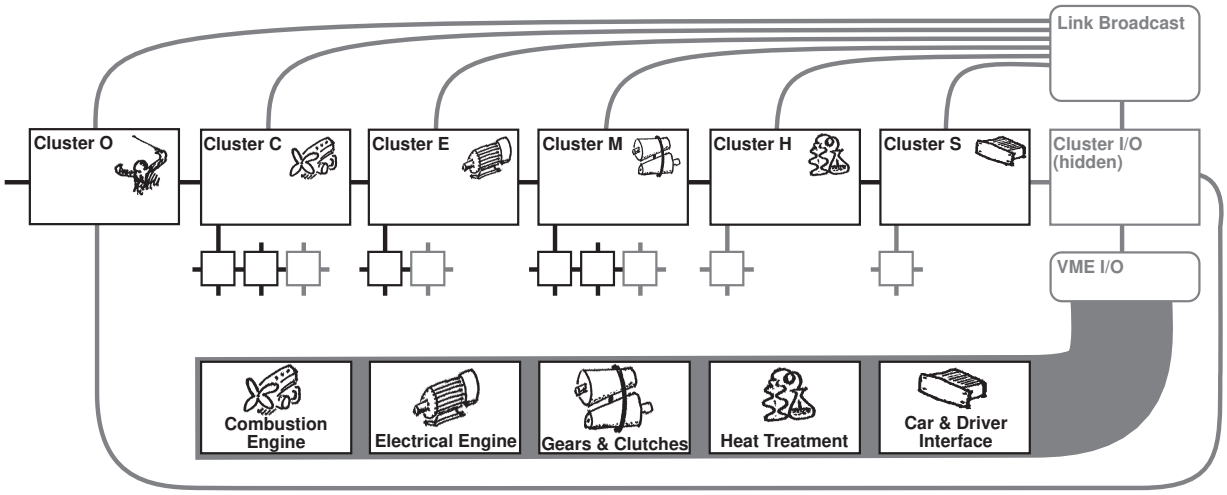

Abb. 2: Implementationsstruktur des Steuersystems

Alle *globalen Funktionen* (Betriebsarten, Fehlerbehandlung etc.) werden vom übergeordneten *SystemMaster* im Cluster Observation **O** erledigt, der *SystemMaster* kontrolliert damit alle langsamen systemübergreifenden Vorgänge.

Die *lokalen Funktionen* all jener Cluster, die mit einer Maschineneinheit verknüpft sind (d.h. **C**, **E**, **M**, **H** und **S**), werden von einem lokalen *SubMaster* gesteuert; die meisten schnellen Vorgänge im System bleiben so lokal.

Die Detailausführung der Fahrstrategien (Kupplungsbetätigungen, Getriebebetätigung und -umschaltung etc.) ist sehr stark auf den Cluster Mechanical **M** konzentriert und wird in dessen *StrategyMaster* angesiedelt, der damit diejenigen schnellen Vorgänge beherrscht, die unter Umständen auch systemübergreifend sein können.

Manchmal bestehen Widersprüche zwischen der *funktionalen* Zugehörigkeit und den *konstruktiven* Gegebenheiten. Da die Konstruktion die Randbedingungen auch für das Steuersystem vorgibt, sind alle Elemente denjenigen Clustern zugeordnet, zu denen sie konstruktiv gehören. Ein *RemoteService* Mechanismus macht solche in *funktionaler* Hinsicht falsch plazierte Elemente den Clustern zugänglich, zu denen sie von der Funktion her gehören.

Cluster verfügen bei Bedarf über zusätzliche lokale Ressourcen (Rechenkapazität, Speicher etc.).

Solange es erforderlich ist, kann ein Entwicklungssystem am Cluster **O** angeschlossen werden.

3.1. Prozessoren

Die Prozessoren der Steuerung sind Transputer [7], RISC-Prozessoren mit allen Fähigkeiten zum Mehrprozessorsystem. Die klare Gliederung einer komplexen Aufgabe in einzelne Prozesse ist auf unsere Systemanforderungen ideal zugeschnitten; jeder Aufgabenblock wird von einer Gruppe von Prozessen auf einem unabhängigen Rechner gesteuert, der über Kanäle mit seinen Partnern kommuniziert. Die dazu nötige Kommunikation (Synchronisation, Datenaustausch) geschieht via Links (je deren vier), durch die mehrere Transputer zu einem fast beliebigen Netzwerk zusammengeschaltet werden können. Kommunikation und Arbeitsteilung werden von der Hardware unterstützt, so dass ohne ein Betriebssystem direkt auf die Rechner aufgesetzt werden kann.

Einzelne Transputer vom Typ T805 erreichen bei einer Taktfrequenz von 30 MHz ca. 15 MIPS. Die Rechenleistung der integrierten FPU beträgt 2 MFLOPS und die Kommunikationsgeschwindigkeit der Verbindungen zwischen Transputern 20 Mbit/s.

Cluster können bei Bedarf jederzeit weitere Rechenleistung erhalten, indem ihnen ein oder mehrere lokale Sklaven beigefügt werden.

3.2. Kommunikation

Die Strukturpfade im Gesamtsystem gehen sternförmig vom Cluster **O** aus. Diese Strukturpfade sind auch Kommunikationspfade und garantieren, dass Implementationen von unvollständigen Systemen alle ihre Schnittstellen vorfinden; alle Prüfstände fallen in diese Kategorie. Verschiedene Protokolle für Punkt zu Punkt- und Broadcast-Verbindungen sind global vereinbart und decken die vorkommenden Fälle ab.

Auf der ganz anders aussehenden Hardware-Topologie wird diese hierarchische Cluster-Struktur mithilfe des Rings implementiert: ein spezieller Prozess (*Communicator*) auf jedem der verarbeitenden Cluster besorgt den Ring-Datenverkehr von und zu diesem Element. Für andere Adressaten bestimmte Meldungen leitet er weiter. Der Ringmechanismus wird aber auch für die Ausgabedaten und überhaupt für jede Art Meldungsaustausch im System benützt.

3.3. Ein- und Ausgabe

Ein- und Ausgabesignale sind nicht global; sie gehören demjenigen Cluster, auf dessen Maschineneinheit sich der betreffende Sensor, bzw. Aktor befindet. Auch hier besteht jedoch ein Strukturkonflikt mit der Implementation, diesmal mit den Gegebenheiten von Ein- und Ausgabeschnittstellen. Diese sind im allgemeinen mehrkanalig und es ist daher unumgänglich, die vorhandenen Kanäle auf verschiedene Cluster zu verteilen. Erschwerend kommt dazu, dass für die Eingabe drei verschiedene Geschwindigkeitsklassen existieren (z.B. 1 kHz, 100 Hz, 10 Hz nebeneinander).

Damit die Rechenlast im Cluster **IO** und damit verbunden im ganzen System möglichst ausgeglichen ist, wird jeweils zusammen mit denjenigen der höchsten Geschwindigkeitsklasse eine jedesmal andere Gruppe von Signalen aus den niederen Klassen eingelesen. Die grosse Anzahl Signale und die verschiedenen, auch häufig wechselnden Installationen verunmöglichen hier Handarbeit vollends, so dass der notwendige Programmcode aus einer Spezifikationsliste erzeugt wird.

Eingabe und Ausgabe von Prozessdaten sind standardisiert. *Eingaben* werden in fixen Intervallen vorgenommen (d.h. synchron), *Ausgaben* sofort dann, wenn ein Cluster einen neuen Ausgabewert abgesetzt hat (asynchron). Trotzdem ist jeweils ein 'Burst' von Ausgabewerten mit der vorangehenden Gruppe von Eingabewerten verbunden; sie liegen nämlich zusammen zwischen zwei Ticks des Eingabesystems. Wird die Leistungsgrenze der Anordnung überschritten, ist das nicht mehr der Fall!

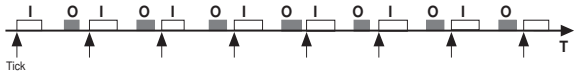

Abb. 3: Zyklischer Ablauf von Ein- und Ausgabe

Ein einzelner Zyklus beginnt auf den 'Tick' mit der Erfassung, Verarbeitung und dem Weitersenden einer Gruppe von *Eingabedaten*. Alle Cluster erhalten eine identische Kopie der neu erfassten, konvertierten Daten simultan über den Broadcast-Mechanismus. Irgendwann später, nachdem die Cluster ihre Verarbeitung beendet und neue Werte errechnet haben, treffen die *Ausgabedaten* im Cluster **IO** ein. Diese Daten sind bis auf Spezialfälle schon konvertiert und können deshalb sofort ausgegeben werden.

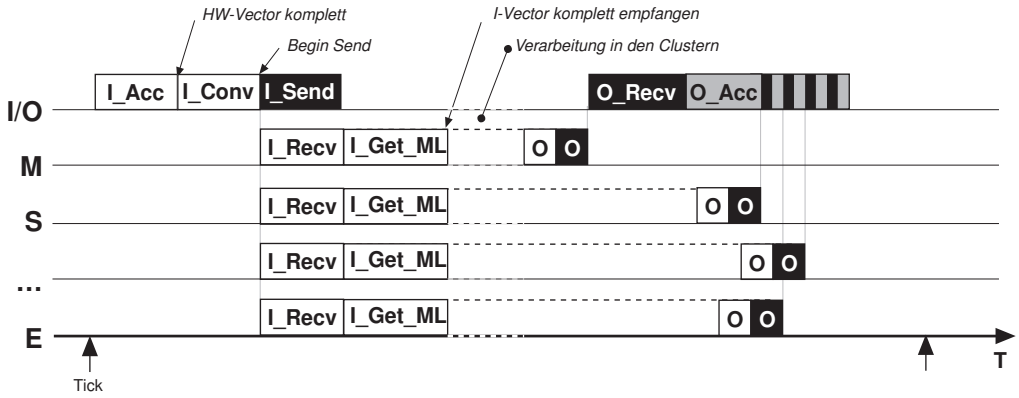

Abb. 4: Synchroner Input und asynchroner Output

3.4. Bussystem

Die Verwendung des M-Modul-Mezzaninbus [5] (aufbauend auf dem VME-Bus [8]) für alle Ein- und Ausgaben öffnet den Zugang zu einer grossen Auswahl an käuflichen Ein- und Ausgabekomponenten. Für Eigenkonstruktionen bringt er den Vorteil von kleinen Karten mit einfachen Schnittstellen. Der VME-Bus gibt aber auch Grund zur Besorgnis: als einziges nicht-, oder nur schwer erweiterbares Element bildet er die Kapazitätsgrenze im Gesamtsystem. Nötigenfalls sind Ergänzungen in zwei Richtungen möglich, einerseits können lokal Signalprozessoren eingesetzt werden (für die Leistungselektronik wurde das von Anfang an vorgesehen), andrerseits sind in Ausnahmefällen auch direkte Peripherieanschlüsse eines Clusters machbar.

3.5. Mechanische und elektrische Ausführung

Die Steuerung des fertigen Fahrzeugs soll klein sein, wenig Energie verbrauchen und den erschwerten Betriebsbedingungen in einem Fahrzeug gewachsen sein. Ein geschlossener, möglichst kleiner 19"-Einschub mit geregelten Lüftern und ausschliesslich steckbaren Verbindungen erfüllt diese Bedingungen. Ausreichende Schirmung ist durch die gewählte Anordnung bereits gegeben; diese wird durch ein rigides Speisungs- und Erdungskonzept auf die ganze Anlage ausgedehnt. Die Leistungsschalter des Steuersystems und einiger anderer gewichtiger Verbraucher sind spezielle Ausführungen mit besonders geringem Haltestrom.

Die Speisung des Steuersystems selbst erfolgt von der Steuereinheit aus, wo alle erforderlichen Spannungen aus dem 12V-Fahrzeugnetz erzeugt werden. Die Versorgung der peripheren Systeme kann zentral geschaltet werden, zudem kann die Steuerung ihr Funktionieren auf jedem Satellit im Betrieb überprüfen. Die gesamte Installation ist aufgeteilt in drei galvanisch getrennte Bereiche und weist einen einzigen zentralen Massepunkt auf.

Damit die Steuereinheit beim Wechsel vom Labor ins Fahrzeug nicht angetastet werden muss, wird sie auch im stehenden Betrieb mit 12 VDC betrieben.

Die zentralen Spannungswandler sind auch für den im Fahrzeugnetz nötigen weiten Eingangsspannungsbereich erhältlich und arbeiten mit hohem Wirkungsgrad.

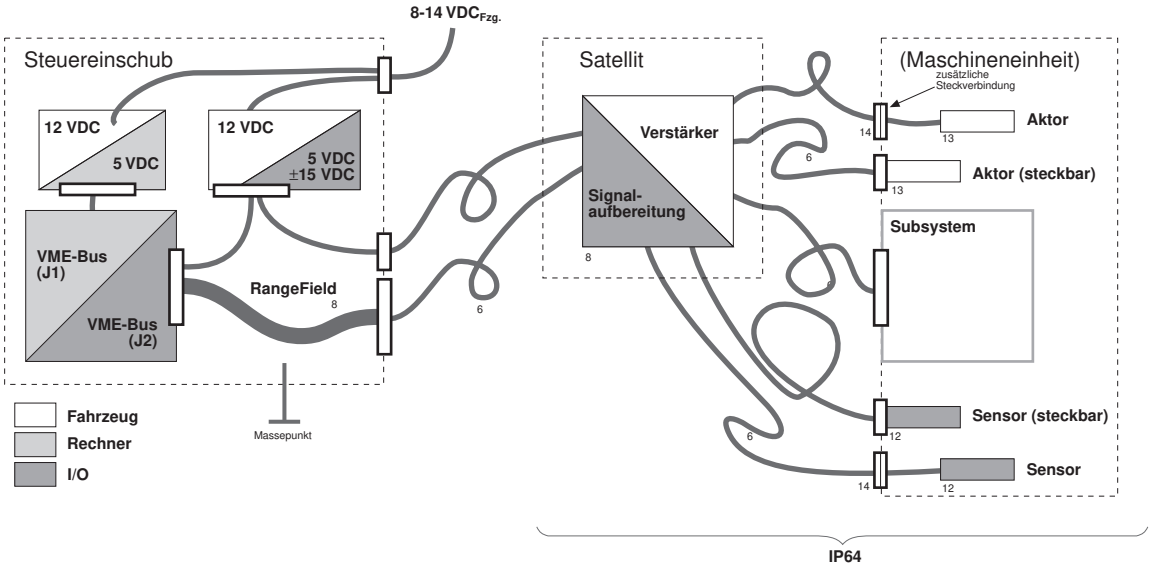

Abb. 5: Verkabelungselemente & Galvanische Trennung

3.6. Verkabelung

Die Verbindungen zwischen der Steuereinheit und den einzelnen Maschinenteilen werden von verschiedenen Randbedingungen diktiert:

Wegen der schwierigen Umgebungsbedingungen sind alle Kabel geschirmt ausgeführt. Die einzelnen Maschineneinheiten wurden zudem separat verkabelt, da die Entwicklung der Komponenten nicht am gleichen Ort erfolgt. Alle Verbindungen sind an gewissen, festgelegten Orten steckbar.

Die verwendeten Kabel sind lang genug, um eine flexible Positionierung der Steuereinheit zu ermöglichen. Während der Fahrversuche wird die Steuereinheit im Fahrzeuginnern zusammen mit anderen Geräten plaziert, später kann sie im Kofferraum Platz finden.

3.7. Codierung

Entwicklungsarbeit bringt es mit sich, dass wiederholt Teile zusammengefügt werden müssen, die in sich selbst noch unfertig und in ständiger Wandlung begriffen sind. In einem komplexen Projekt ist das ein Alptraum ohne Unterstützung durch leistungsfähige Werkzeuge. Leider gibt es noch keine vollständig integrierten Lösungen, so dass wir bei unserem Ansatz aus verschiedenen Quellen schöpfen.

Das Ein- und Ausgabesystem, der Ringmechanismus, die Ablaufumgebung für den generierten Code, gewissermassen als *Systemsoftware* das Fundament für alles andere, muss nahe an der Hardware und effizient programmiert sein und ist deshalb von Hand geschrieben.

Regleralgorithmen werden am besten in einer besonders dafür geschaffenen Umgebung erarbeitet und dann als fertige Bausteine übernommen.

Die eigentlichen *Steuerfunktionen*, die ereignisorientiert arbeiten, sowie die allgemeine Kontrollstruktur jedoch nehmen wir den Programmierern vollkommen aus der Hand. Die Verwendung einer Spezifikationsmethode und der dazugehörigen Arbeitsumgebung erlauben den Entwicklern, problemorientiert zu arbeiten und die eigentliche Codierung dem Werkzeug zu überlassen.

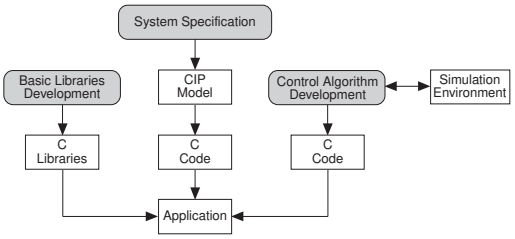

Abb. 6: SW-Entwicklungsstränge

Die fertige Applikation setzt sich aus drei Strängen zusammen. In einer späteren Phase sollten Änderungen einzig noch im mittleren Strang vorkommen, d.h. man bleibt dann von Programmierfehlern verschont (wenn auch nicht von logischen…).

4. METHODEN UND WERKZEUGE

4.1. Arbeitsumgebung

Die einzelnen Gruppen arbeiten selbständig und an verschiedenen Orten an ihrem Teilsystem. An einer Workstation mit allen nötigen Werkzeugen hängt ein reduziertes Steuersystem mit den benötigten Rechnerknoten sowie Ein- und Ausgabemöglichkeiten. Zur Koordination der Arbeiten und auch zum sonstigen Datenaustausch bestehen Netzwerkverbindungen.

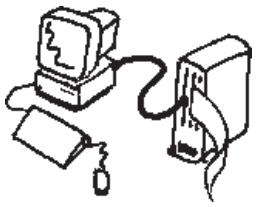

Abb. 7: Entwicklungsstation

Die Schnittstellen sowohl der Maschine wie der Steuerungselemente sind für den späteren Zusammenbau vorbereitet, diese können also für Tests ohne umwälzende Änderungen an der Hard- und Software zusammengefügt und dann von einer einzigen Entwicklungsstation aus bedient werden. Beim Betrieb im Fahrzeug, wo Platz und Energie knapp sind, wird das besonders wichtig sein.

Sind Entwicklung und Testbetrieb einmal abgeschlossen, wird die gleiche Steuerungsstruktur ohne eine Entwicklungsstation weiter verwendet, die Bedieneroberfläche wird dann die eines ganz normalen Fahrzeugs sein.

Um ein reduziertes Steuersystem mit einer beliebigen Anzahl Rechnerknoten sowie Ein- und Ausgabemöglichkeiten zu betreiben, sind keinerlei Änderungen am unterliegenden Programmgebäude nötig. Alle Anpassungen erfolgen in den Spezifikationen der beteiligten Cluster (es müssen etwa die in der aktuellen Installation fehlende Kommunikationspartner simuliert werden).

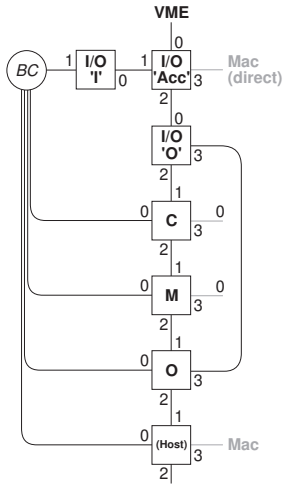

Abb. 8: Teilsystem

4.2. Spezifikation und Modellierung

Das hybride Antriebskonzept bedeutet eine grosse Steigerung der Komplexität gegenüber konventionellen Fahrzeugen, was sich besonders im verwikkelten Zusammenspiel der Einzelkomponenten niederschlägt. Die komplizierten Abläufe entziehen sich weitgehend der direkten Vorstellung, weshalb die Darstellung der diskreten Zustände und die Simulation der Abläufe mittels geeigneter Werkzeuge in der Planungsphase grosse Bedeutung erlangt. Ideal ist, wenn die dort geleisteten Arbeiten direkt in die spätere Applikation einfliessen können, was nur mit der Unterstützung eines leistungsfähigen, durchgängigen Werkzeugs gelingen kann.

Das gilt für alle 'Embedded Systems', eingebettete Systeme also, die als integrierender Bestandteil von Geräten umgebende Prozesse steuern und regeln, wozu sie ausserordentlich zuverlässig und in Echtzeit funktionieren müssen (Maschinen- und Prozessteuerungen, Prozessleitsysteme u.a.).

Wie arbeiten Ingenieure an der Lösung ihrer regeltechnischen oder anderen Steuerprobleme? Üblich ist, dass sie sich neben der eigentlichen Aufgabe allzusehr mit der eigentlichen Codierung von Algorithmen befassen müssen, zum 'Was' gesellt sich gewichtig das 'Wie' und im Laufe der Arbeit, besonders dann bei späteren Änderungen, lassen sich die beiden überhaupt nicht mehr auseinanderhalten. Wir beschreiten einen anderen Weg: anstatt zu programmieren, wird das funktionale Verhalten der einzelnen Blöcke in einem Werkzeug als grafisches Modell beschrieben. CIP (Communicating Interacting Processes) ist eine unter anderem am TIK entwickelte formale Entwicklungsmethode, mit der sich verteilte, eingebettete Systeme durch grafische Modelle ablauffähig spezifizieren lassen. Der Entwicklungsansatz geht vom Verhalten der realen Objekte der Umgebung aus und führt in konstruktiven Schritten zum kompositionell definierten Gesamtsystem. Aus solchen Spezifikationen kann direkt Programmcode erzeugt werden, der in einer vorbereiteten Umgebung ohne weiteres Zutun laufen kann.

Hybrid III ist ein geradezu beispielhaftes eingebettetes System, dazu ein recht kompliziertes. Dafür ist 'CIP Tool' das geeignete Software-Entwicklungswerkzeug; die Designer der Antriebskomponenten brauchen sich als reine Computeranwender nicht um die Einzelheiten des Steuersystems zu kümmern und können sich so ganz auf das richtige funktionale Verhalten ihres Maschinenteils konzentrieren.

Obwohl es keine Einbussen bei der Funktionalität gibt (jedenfalls nicht bei eingebetteten Systemen) bedeutet nicht zu programmieren auch eine Einschränkung, da nicht mehr die ganze Mächtigkeit einer Sprache (z.B. C, mit dem sich alles machen lässt…) zur Verfügung steht. Dieser Verzicht ist ein Vorteil, denn so lange man in der Frage der Korrektheit von Programmen noch im Dunkeln tappt, sollte es darum gehen, einfache Lösungen zu suchen, die sich auch einem späteren Verständnis nicht entziehen.

4.3. Communicating Interacting Processes CIP

Die CIP-Methode modelliert das Systemverhalten durch kooperierende Zustandsmaschinen mit einer Notation von asynchronen Clustern, die ihrerseits synchron untereinander kooperierende Prozesse in verschiedenen Betriebsmodi enthalten.

Das mathematische Modell eines Systems ist als Produkt dieser vielen Zustandsmaschinen definiert, es stellt also eine einzige Maschine im mehrdimensionalen Zustandsraum dar.

Ereignisse aus der Umgebung (Event) veranlassen einen Zustandsübergang im empfangenden Prozess, der seinerseits mit Pulsen weitere Prozesse des Clusters anstossen kann, etc. Die aktivierten Prozesse können dann auf die Umgebung zurück wirken (Action). Eine so entstehende Kettenreaktion ist ununterbrechbar und definiert einen einzigen Zustandsübergang des Clusters; dass sie nicht zyklisch ist, dafür sorgt das Werkzeug.

Die Prozesse haben auch die Möglichkeit, systemweit über Datenströme asynchron untereinander zu kommunizieren (Communication).

Man kann einen Prozess durch Modi erweitern; er weist dann verschiedene Verhalten entsprechend unterschiedlichen Zustandsmaschinen auf.

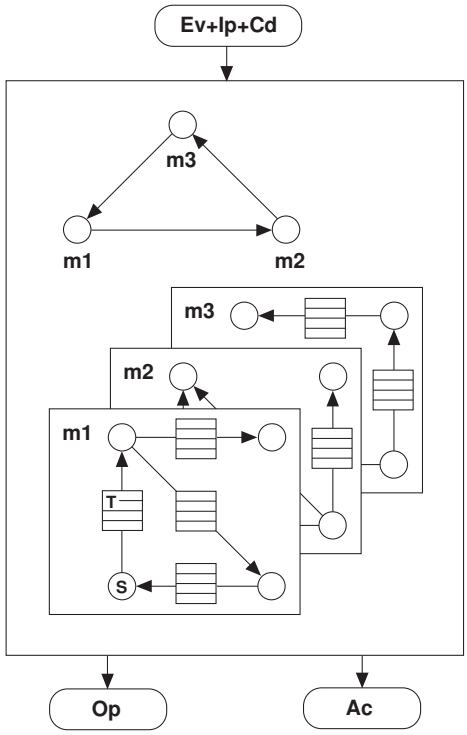

Abb. 9: CIP Prozesstruktur

Prozesse sind entweder Modellprozesse, die einen Teil der Maschinenrealität abbilden, oder Funktionsprozesse, wenn sie das reaktive Verhalten des Modells definieren. Die logische Struktur vieler Steuersysteme, auch diejenige von Hybrid III, stimmt mit dieser Art der Modellbildung ideal überein.

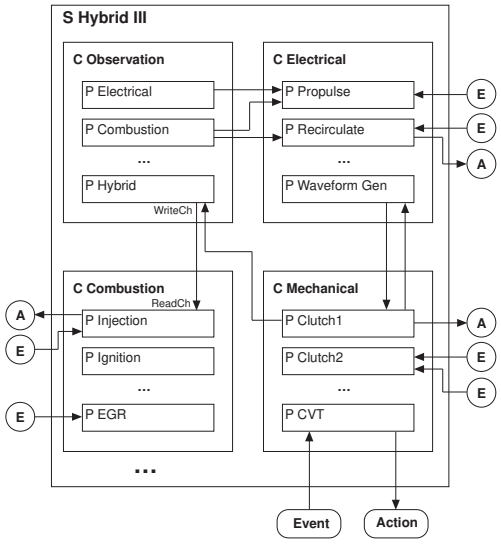

Abb. 10: Hybrid III Projektstruktur (Ausschnitt)

5. HARDWARE-REALISIERUNG

5.1. Steuereinschub

Der Steuereinschub enthält Speisungen für die Rechner und die Peripherie, einen VME-Bus mit 12 Steckplätzen und eine Karte für die Link-Verbindungen der Transputer. Bis auf den einen mit Zugang zum VME-Bus finden Transputer auf TRAM-Modulen Verwendung. Daneben gibt es sieben Trägerkarten für höchstens 28 M-Module. Folgende Ein- und Ausgabemöglichkeiten (nebst den Spezialfällen) sind vorhanden:

80 binäre Eingänge,
64 analoge Eingänge,
8 Drehzahlmessungen,
2 Drehwinkelmessungen,
64 binäre Ausgänge,
24 analoge Ausgänge.

Alle Anschlüsse sind über ein Rangierfeld nach hinten zu versenkten und codierten Steckern herausgeführt; jeder Satellit hat sein eigenes Daten- und Speisungskabel.

Die Steuereinheit wird über einen externen Schalter gestartet (Zündschlüssel), kann aber so nicht abgeschaltet werden; das erledigt sie selbsttätig und nur dann, wenn der Systemzustand es erlaubt.

Die Ein- und Ausgabeschnittstelle hat eine Erweiterung erfahren um zwei weitere Transputer, so dass sich jetzt drei Rechnerknoten die Aufgaben von Cluster **IO** teilen: **IO_ACC** erledigt die Buszugriffe, **IO_I** konvertiert und filtert die Eingabewerte und sendet sie an die anderen Cluster, **IO_O** gibt Daten an die Aktoren aus und bedient die Ring-Schnittstelle.

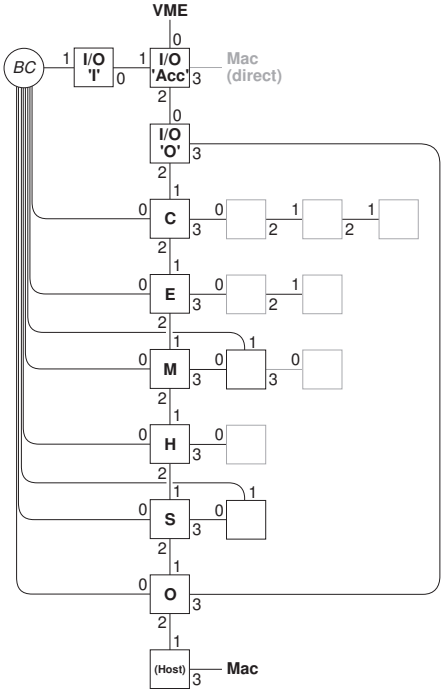

Abb. 11: Implementation Gesamtsystem

5.2. Peripherie

Alle peripheren Anschlüsse sind in Satelliten zusammengefasst. Zehn solcher Satelliten sind in der Nähe der Maschinenkomponenten plaziert und einzelnen Clustern zugeordnet. Die Anschlüsse von Sensoren, Aktoren einerseits und der Steuereinheit andrerseits können dort rangiert werden und nötige Anpassschaltungen, Wandler, Verstärker etc. sind ebenfalls dort untergebracht.

Satelliten enthalten also eine Reihe verschiedenster Funktionen, die sich aber dennoch in Gruppen aufteilen lassen. Diese Voraussetzungen sind ideal für ein frei kombinierbares System mit modularen Elementen für alle vorkommenden Fälle, Verstärker, Wandler etc. Existierende derartige Lösungen kamen nicht in Betracht, da sie alle viel zu viel Platz beanspruchen.

Die Satelliten sind klein (250 x 75 x 75 mm), verfügen über massive Gehäuse mit guter Schirmung und Schutzart IP64 und können deshalb überall im Fahrzeug eingesetzt werden. Sie sind mit einer Basiskarte bestückt, die Anschlüsse für die Versorgungsspannungen, das Steuersystem sowie für die Sensoren und Aktoren aufweist. Acht Steckplätze nehmen übereinander gestapelt bis zu drei Schnittstellenkarten, sogenannte S-Karten auf.

15 verschiedene solche Kartentypen wurden entwikkelt; eventuell vorhandene Leistungselemente arbeiten mit geringen Verlusten, was nicht nur Energie spart, sondern auch Abwärmeprobleme in den geschlossenen Gehäusen verhindert.

Die Sensoren und Aktoren stammen teils aus der Automobiltechnik, teils aus der industriellen Steuerungstechnik. Die Sammlung besticht durch grösstmögliche Variation aller Anschlusswerte; mit Hilfe der S-Karten schaffen wir einheitliche Bedingungen für die Schnittstelle des Steuersystems.

5.3. Spezielle HW

Fast überall wurden käufliche Komponenten verwendet, das war jedoch nicht immer möglich. So enstanden die Satelliten, sämtliche S-Karten und einige Eingabe- und Ausgabekarten (M-Module) neu. Das Multiprozessor-Netzwerk wurde um eine Broadcast-Möglichkeit erweitert.

Das M-Modul für die Ansteuerung der Leistungselektronik stammt vom IEM, alle anderen Karten wurden am TIK entwickelt.

Broadcast BC

Der Cluster **IO** ist für Datenerfassung und -ausgabe an zentraler Stelle vorgesehen. Es handelt sich um den einzigen Transputer mit direktem Zugang zum VME-Bus. Besonders bei der Datenerfassung ergeben sich dabei Probleme, die mit den für Transputer gängigen Kommunikationsmitteln schwer in den Griff zu bekommen sind. Der Vektor mit den aktuellen, aufbereiteten Daten wird periodisch an die verschiedenen verarbeitenden Cluster übermittelt. Da die verwen-

deten Transputer nur über vier einzelne Links verfügen, müsste dieser Vektor von einem zum anderen weitergereicht werden, was unerwünschte zusätzliche Belastungen der einzelnen Cluster mit sich brächte; zudem ginge dabei die angestrebte Gleichzeitigkeit im System verloren.

Die Erweiterung des Link-Mechanismus um eine Broadcast-Möglichkeit erlaubt, diesen Vektor gleichzeitig allen Abonnenten zu senden. Beide genannten Nachteile werden damit vermieden. Unsere Karte kann einen Transputerlink-Ausgang auf bis zu acht Empfänger aufspalten.

Drehzahlmessung M-RPM

Drehzahlen und Drehwinkel gehören zu den wichtigsten Messwerten in einer Antriebssteuerung; es gibt eine ganze Anzahl davon. Digitale Methoden zur Drehzahlmessung, wo von einem Sensor erzeugte Pulse entweder gezählt oder deren Periodendauer bestimmt wird, sind mit einigen Unschönheiten versehen, so ist die für eine bestimmte Genauigkeit erreichbare kürzeste Messdauer drehzahlabhängig (bzw. umgekehrt) und höhere Auflösung bei gleichbleibend kleiner Messdauer bedingt eine steigende Anzahl Pulse pro Umdrehung.

Beides ist in diesem Fall unerwünscht: auch bei kleinen Drehzahlen müssen wir wegen der im ganzen Bereich zu bewältigenden Kupplungsvorgänge schnell und genau messen können. Die theoretisch kürzeste Antwortzeit eines digitalen Geschwindigkeitsreglers beträgt 1.5 Pulse, praktisch liegt dieser Wert aber 5-200 mal höher. Die damit nötig werdenden Sensoren mit hohen Pulszahlen pro Umdrehung sind wenig robust und in einer Fahrzeugumgebung nicht erwünscht, zudem kann die Übertragung von Signalen mit hoher Frequenz Probleme schaffen. In unserem M-Modul nützen wir die physikalische Tatsache, dass keine Geschwindigkeitssprünge auftreten können und lassen auch die Werte der vorangegangenen Messungen in das neue Resultat einfliessen [4, 6].

Das Verfahren füllt die Periode zwischen zwei vom Sensor kommenden Pulsen mit einer generierten Pulsreihe auf. Aufgrund der früheren Messungen wird der Zeitpunkt, an dem der nächste Puls eintreffen wird, vorausgesagt. Wirklich eintreffen wird er aber ein wenig früher oder später als erwartet, was dann Anlass zu einer Korrektur der nächsten Erwartung gibt. Mit der Masszahl Z_k für den Zeitpunkt des erwarteten Pulses gilt $Z_{k+1} := Z_k + (N\text{-}P_k)$.

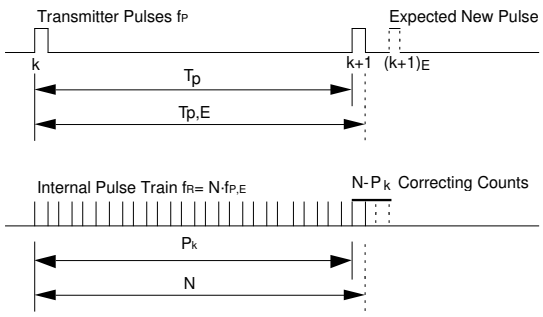

Abb. 12: Drehzahlmessung, Prinzip

Der generierte Pulsstrom entspricht gedachten zusätzlichen Zähnen auf dem Codierrad, gleichbedeutend einem Rad mit N-mal so vielen Zähnen wie tatsächlich vorhanden sind. Die Anforderungen an Codierräder, Sensoren, Übertragung und Auswertung konnten damit erheblich verkleinert werden.

Verbrennungsmotorsteuerung M-ICX

Das M-Modul dient der Ansteuerung eines Vierzylinder-Verbrennungsmotors und kann ausgehend von der Drehwinkelinformation selbsttätig Einspritz- und Zündimpuls-Zeitpunkte am Motor steuern. Die Ausgabe des Steuersystems wird so erheblich entlastet; der Aufwand reduziert sich auf das periodische Bereitstellen von neuen Werten für die diversen Winkel und Zeiten. Die beiden Pulsreihen ϕ (Drehwinkel der Kurbelwelle) und σ (Synchronisations-Impuls, alle zwei Umdrehungen) werden direkt von einer benachbarten M-RPM übertragen [3].

6. SOFTWARE-REALISIERUNG

6.1. Systemsoftware

Die beiden systemübergreifenden Verbindungen (Ring und Broadcast) sind auf allen Clustern mit entsprechenden Prozessen vertreten. Zahlreiche Puffer entkoppeln globale und lokale Vorgänge voneinander. Der Code aller dieser Prozesse ist im ganzen System der gleiche. Darüber hinaus sind auch alle verarbeitenden Cluster identisch und gliedern sich um den einen Prozess, in dem die aus der Spezifikation generierten eigentlichen Steuerfunktionen ablaufen.

In diesem Teil der Arbeit haben sich Nachteile der parallelen Programmierung, bzw. der Mangel an brauchbaren käuflichen Werkzeugen zur Fehlersuche und Profilierung bemerkbar gemacht. Um die Leistungsfähigkeit des Gesamtsystems auszuschöpfen, mussten vor allem die Prozessprioritäten aufwendig aufeinander abgestimmt werden.

Die drei vorkommenden Clustertypen:

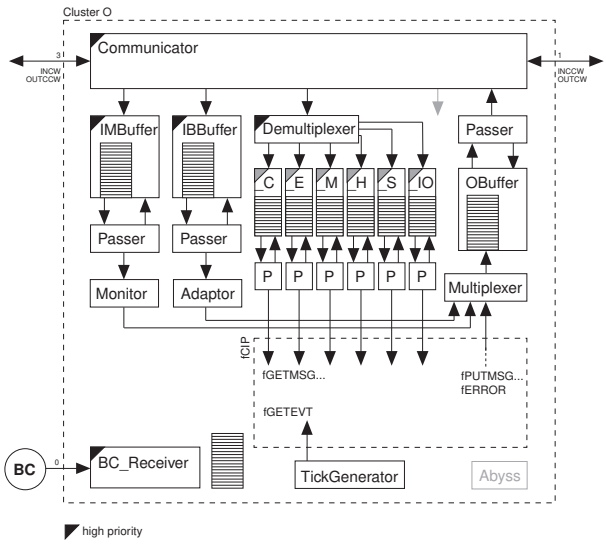

Abb. 13: Prozesstruktur von Cluster O

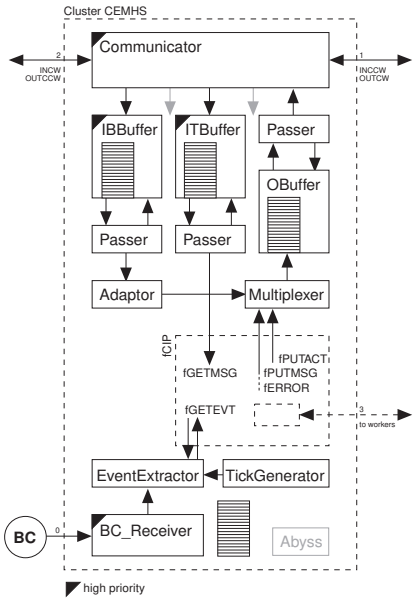

Abb. 14: Prozesstruktur der Cluster C, E, M, H und S

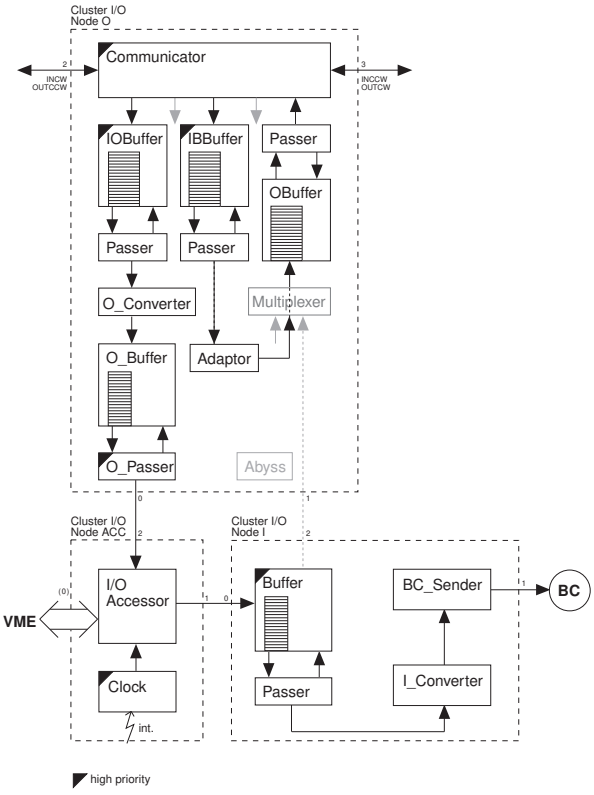

Abb. 15: Prozesstruktur von Cluster IO

6.2. CIP-Shell

Für die Cluster **C**, **E**, **M**, **H** und **S** können Eingabedaten zwei grundsätzlich verschiedene Bedeutungen haben, entweder wie gewohnt direkt als Wert, oder als ein Wert, der ein von CIP zu verarbeitendes Ereignis enthält. Der CIP-Prozess, eine den CIP-Code einhüllende Shell, muss die Daten in geeigneter Form aufbereiten und zwischen den gelesenen Werten und der in CIP spezifizierten Struktur einen Zusammenhang herstellen.

Die Art und Weise dieser Anbindung beschreiben die Betreiber der einzelnen Cluster in einer Konfigurationsliste, abgestimmt auf die verwendete Ereignisstruktur und die Namen etc. in der CIP-Spezifikation, wie auch aufgrund der Namen auf der Eingabeseite. Aus der Liste wird Programmcode erzeugt für den *EventExtractor*, der dann im laufenden System CIP mit Ereignissen bedient.

Die Beschreibung kann Zeitereignisse, binäre Ereignisse, sowie Ereignisse, die sich aus kontinuierlichen Messwerten ergeben, enthalten.

Ereignisse im Zusammenhang mit binären Signalen sind einfach zu verstehen: das Ereignis tritt ein, wenn der Signalwert einer spezifizierten Maske entspricht.

Ereignisse aus kontinuierlichen Signalen treten dann ein, wenn bestimmte Grenzwerte über- oder unterschritten werden. Diese Grenzwerte können an beliebige Stellen innerhalb des Wertebereichs eines Signals gelegt werden. Der *EventExtractor* garantiert für die Einhaltung der Kontinuität, d.h. es werden keine beliebigen Folgen von Ereignissen erzeugt.

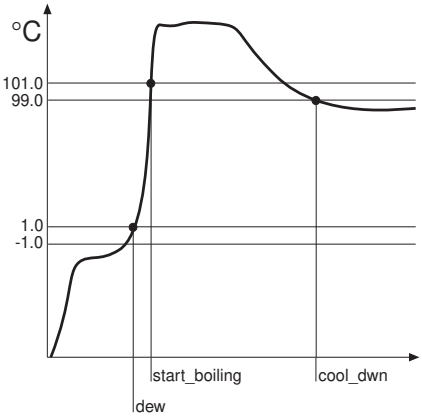

Abb. 16: Ereignisse in einem kontinuierlichen Signal

Die CIP-Shell bindet darüber hinaus den CIP-Code an die darunterliegenden Systemroutinen für Kommunikation und Datenausgabe.

Ausser den Regleralgorithmen wird die gesamte Anwendersoftware, also der Code mit den eigentlichen Clusterfunktionen, aus CIP-Spezifikationen generiert.

7. SCHLUSSFOLGERUNGEN

Hybrid III ist ein sehr komplexes System – und es läuft. Die drei wichtigsten Aufgaben, Entlastung der Entwickler von Programmieraufgaben, Miniaturisierung der Elemente und Flexibilität sind gelöst worden, mit unterschiedlichem Erfolg.

Wir probierten ein neues Vorgehen für die Steuersystemtechnologie. Da sich das CIP-Konzept an der Realität orientiert, sind klare und verständliche Spezifikationen möglich. Die Abstützung auf formale Modelle erlaubt die Entwicklung von robusten und zuverlässigen Systemen. In unserem Fall ist es wohl

sogar so, dass das Steuersystem ohne ein solches Hilfsmittel nicht zum Laufen gekommen wäre. Das Werkzeug fordert eine gewisse Disziplin und schwächt so die Folgen der 'akademischen Arbeitsweise' ein wenig ab. Das zeigt sich auch an der Häufigkeit, mit der sich fatale Fehler in den *nicht* generierten Code einschleichen.

Die sozusagen banalen Aufwendungen für den Einbau des Steuersystems in ein Fahrzeug – Robustheit, kleine Bauformen, Störsicherheit – haben einen grossen Anteil am Gesamtaufwand verschlungen. Der nun folgende Einbau in ein wirkliches Fahrzeug, wo diese Eigenschaften erst nötig werden, ist schon aus diesem Grund sehr begrüssenswert.

Im Laufe der Projektdauer sind Transputer aus politischen Gründen fast von der Bildfläche verschwunden. Das hat dazu geführt, dass sie heute in Hinsicht der Rechenleistung nicht mehr auf dem neuesten Stand sind. In der Folge erlitten auch die verfügbaren Werkzeuge eine erhebliche Ausdünnung und wir mussten mehr Dinge selbst machen als uns lieb war. Vom technischen Standpunkt aus ist das Konzept der Transputer nach wie vor attraktiv und sogar besonders geeignet für komplexe, eingebettete Systeme.

8. LITERATUR

[1] *CIP Tool User Manual 1.2,* CIP System AG, Solothurn 1996

[2] H. Fierz, H. Müller et al.: *CIP, Communicating Interacting Processes, A Formal Method for the Development of Reactive Systems,* Poznan 1993

[3] H.P. Geering: *ICX: A Custom VLSI-Chip for Engine Control*, IJVD 1989

[4] *Ignition and Injection with a Minimum of CPU Load,* Siemens Automotive Engine Management System

[5] *Mezzanine Bus Mini-Specification*, VITA 1991

[6] V. Török et al.: *High Accuracy and Fast Response Digital Speed Measurement for Control of Industrial Motor Drives,* IFAC

[7] *The Transputer Databook*, inmos 1989

[8] *The VME-Bus Specification*, VITA 1987

Studienarbeiten um Hybrid III:

P. De Carli: *Fahrzeugprojekt Hybrid III: Sicherer Datenverkehr vom Auto zur Steuerung und umgekehrt*, TIK SS 94

P. De Carli: *Fahrzeugprojekt Hybrid III: Ein falsches bit und nichts geht mehr?* TIK WS 94

G. Eger: *Die Schnittstelle des neuen Antriebssystems von Hybrid III zum normalen VW Golf*, TIK WS 93

A. Erne, H.P. Schmid: *Fahrzeugprojekt Hybrid III:Wandler- und Filterkarte zur Drehzahlmessung*, TIK SS 93

D. Karaiskakis: *Fahrzeugprojekt Hybrid III: Bedienung wie gewohnt – nur besser!* TIK WS 94

Ch. Koch: *Auto der Zukunft nur noch für Ingenieure? Zur Benutzerschnittstelle von Hybrid III*, TIK WS 93

Ch. Kupper, F. Wunderli: *Fahrzeugprojekt Hybrid III: Mess- und Ansteuerungskarten zur Motorsteuerung*, TIK SS 93

M. Manhart: *Fahrzeugprojekt Hybrid III: Broadcast-Mechanismus für Transputer Links*, TIK SS 93

A. Meyer: *Zur Benutzerschnittstelle im Hybrid III: Fahren und Bremsen*, TIK SS 94

P. Reali: *Fahrzeugprojekt Hybrid III: das Konfigurationswerkzeug für Ein- und Ausgabe – neu gesehen*, TIK WS 93

M. Sturm: *Programmbibliotheken Hybrid III,* Semesterarbeit TIK SS 92

F. Tondi: *Fahrzeugprojekt Hybrid III: Graphische Datenausgabe am Armaturenbrett*, TIK SS 94