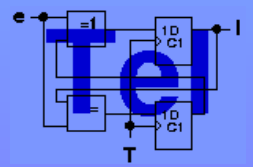
Untersuchungen von HW-Architekturkonzepten für Agentensysteme

Marcel Naggatz

Institut für Technische Informatik

18. Oktober 2006

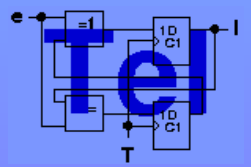


Entwicklung der Elektronikkomponenten

- 1886 befanden sich im ersten Fahrzeug keinerlei elektronische Komponenten
- in heutigen Fahrzeugen stecken bis zu 70 elektronische Komponenten
- 90 Prozent aller Innovationen in Fahrzeugen beruhen zukünftig auf Elektronik

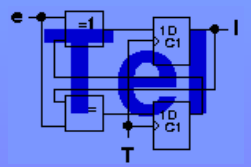
Probleme

- Zuverlässigkeit sinkt
- Entwicklungsaufwand steigt
- Komplexität der Systeme steigt



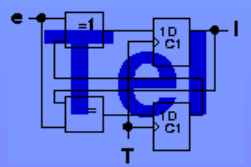
Hardwareagenten

- autonom handelnde Hardware
- bei der Betrachtung wird in externe und interne Eigenschaften unterteilt
- haben Organic Computing und Self-X als Voraussetzung
- helfen bei der Bewältigung komplexester Aufgaben
- agieren selbstständig in ihrer jeweiligen Umgebung
- besitzen Schnittstellen zu anderen Agentensystemen (Hard- und Software)
- können Mobile Roboter, komplexe selbstagierende Schaltungen, etc. sein



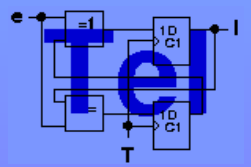
Eigenschaften von Hardwareagenten

- Reaktivität: die Funktionseinheit reagiert auf Änderung der Umgebung
- Proaktivität/Zielorientiertheit: die Funktionseinheit löst Aktionen aufgrund eigener Initiative aus
- Schlussfolgerungs-/Lernfähigkeit: die Funktionseinheit lernt aufgrund zuvor getätigter Entscheidungen bzw. Beobachtungen
- Autonomes Handeln: die Funktionseinheit arbeitet weitgehend unabhängig von Benutzereingriffen
- Mobilität
- Kommunikation/Kooperation: die Funktionseinheit kommuniziert mit anderen Agenten



Organic Computing

- biologisch inspirierte Forschungsinitiative
- Organisation komplexer Systeme
- durch Entwurfsziele gekennzeichnet
- wendet sich von algorithmischer Methodik ab
- dynamische Anpassung an die jeweilige Umgebung
- Self-X-Eigenschaften
  - selbst-konfigurierend
  - selbst-heilend
  - selbst-optimierend
  - selbst-schützend



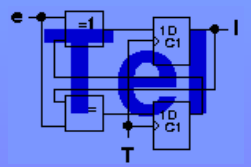
Vor- und Nachteile

Vorteile

- flexibel und robust gegenüber (Teil-)Ausfällen
- in der Lage sich selbst zu optimieren
- können sich an Anwenderbedürfnisse anzupassen
- geringerer Entwicklungsaufwand

Nachteile

- lernende technische Systeme können, analog zu biologischen Systemen, Fehler machen



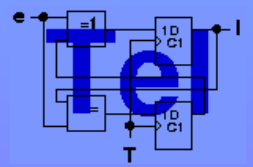
Self-X Systeme

Bei Self-X Systemen handelt es sich um digitale Elektronik, die sich bei Bedarf oder aufgrund eines Programmes in ihrer logischen Funktionsweise (Schaltung) verändert.

Anforderugen an die Hardware:

- rekonfigurierbar (dynamisch, statisch)
- große Logikfläche
- es existiert keine zentrale Instanz
- flexible, dynamische Kommunikation untereinander
- sie muss Testbar sein

"Aus der Summe der autonomen Handlungen der einzelnen Einheiten resultiert schließlich das evolutionäre Verhalten des Gesamtsystems."



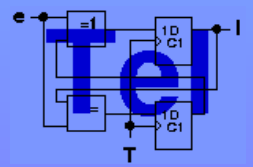
Vorhandene Hardware und VHDL-Designs

Hardware

- Xilinx Spartan3 FPGA Experimentierboard
- ein Programmierboard mit einem ATmega32 Microkontroller

VHDL-Designs

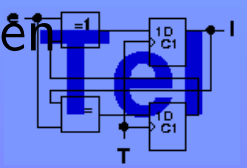
- VHDL-Beschreibung eines Atmel AVR ATtiny2313
- VHDL-Beschreibung eines Atmel AVR ATmega32 (Light)



Atmel AVR ATtiny2313

Vorteile

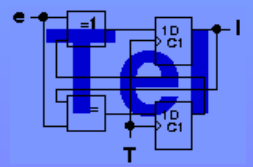
- voller Befehlssatz wurde implementiert
- Taktbar bis zu 64MHz
- 2 Pipelinestufen zu je 2 Takten
- doppelt so schnelle Ausführung der meisten Befehle wie beim Original (z.B. JUMP, RCALL, etc.)
- es passen 3 ATtiny2313 auf einen Spartan 3 mit 200.000 Gatterelementen mit Platz für evtl. Steuereinheiten
- Aussnutzung der FPGA-spezifischen Komponenten, wie Dual-Port-RAM, DCM, HW-Multiplizierer
- es können mehr I/O-Ports als im Original verarbeitet werden (Registerplätze des EEPROM werden genutzt)



Atmel AVR ATtiny2313

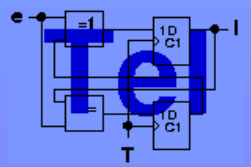
Nachteile

- Fehler im Design der UART und Timer
- zu geringer Funktionsumfang der Interrupt-Behandlung und der UART (keine Multiprozessor-Fähigkeit
- kein Bootloader vorhanden



Atmel AVR ATmega32

- Aufbauend auf dem ATtiny wurde das VHDL-Modell redesigned
- es wurde die stärkere UART (ohne die synchrone Datenübertragung) des ATmega implementiert, hier besteht die Möglichkeit, 9Bits an Daten zu übermitteln und somit einfach Adressen zwischen zwei Cores auszutauschen
- die Timer wurden überarbeitet und von Fehlern bereinigt
- der WDTimer wurde in eine gesonderte Komponente ausgegliedert
- insgesamt wurden die Komponenten für einen einfacheren Austausch überarbeitet
- Bootloader-Support

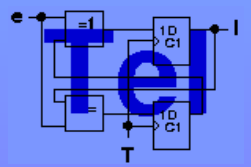


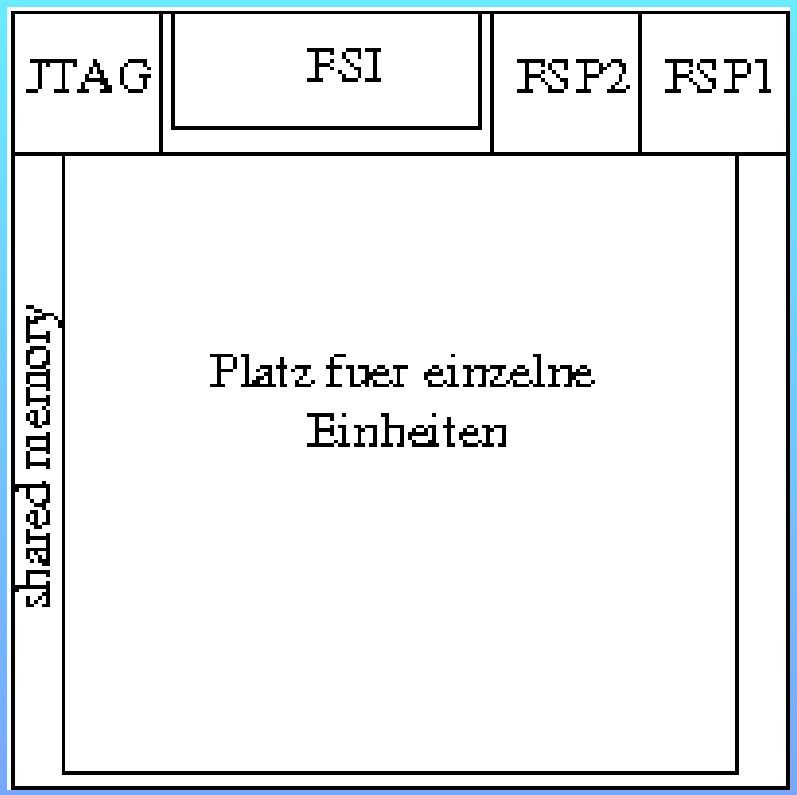
Komponentenaustausch im Design

- es wurde ein VHDL-Parser mit JavaCC entworfen
- die Strukturen und die Komponenten der VHDL-Files, des kompletten Designs werden eingelesen

TODO

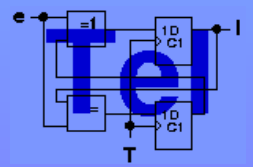
- unter Ausnutzung des neuen Designs, sollen nicht benötigte Komponenten aus dem Design entfernt werden
- es muss ein Programm entworfen werden, in welchem die gewünschten Komponenten ausgesucht/eingegeben werden




JTAG
FSI
RSP2
RSP1
shared memory
Platz fuer einzelne Einheiten

verwendete Komponenten

- Grunddesign ist der ATmega32
- spezielle Service-Prozessoren FSP
- eine I/O-Komponente für die einzelnen Einheiten
- universelle Testkomponente
- Verbindungsnetzwerk für die einzelnen Komponenten
- Shared Memory

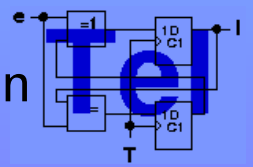


Flexible Service Processor

- zwei Stück zur gegenseitigen Überwachung
- Atmega32 ergänzt um weitere I/O-und Testfunktionalitäten
- Übernehmen die Kommunikation, des kompletten Designs nach außen
- überwachen die Komponenten auf dem Chip
- Vortstellbar währe, dass diese FSP auch die Umprogrammierung des FPGA übernehmen, somit könnte der externe Microkontroller verschwinden (Gedankenexperiment)
- Verteilung der Aufgaben an die einzelnen Komponenten

TODO

- fertigstellen des Designs
- Test der Aufgabenverteilung
- Zusammenarbeit mit den Testkomponenten implementieren
- Gegenseitige Überwachung und Ausfall testen

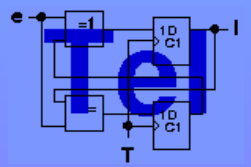


Flexible Service Interface

- kümmert sich um die Ansteuerung der Ports, der einzelnen Komponenten nach außen
- alle externen Kommunikationen, der auf dem FGPA enthalten Einheiten laufen über FSI
- wenn diese Komponente ausfällt ist keine Kommunikation mehr möglich, da jedoch nur die Ports des FPGA angesprochen werden, reicht bei einem Ausfall einfach ein erneutes Programmieren des FPGA, sollten die Ports defekt sein, muss der FPGA gänzlich getauscht werden
- flexible Nutzung der, auf dem FPGA vorhandenen Pins

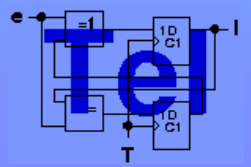
TODO

- Test mit anderen Komponenten als dem ATmega



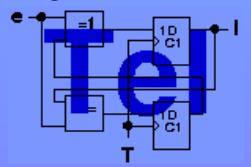
Shared Memory

- SRAM auf dem Spartan3 Board
- Befehlsspeicher für FSP
- Zwischenspeicher für den Programmcode der einzelnen Komponenten
- dient zur einfachen Übergabe von Werten
- einfache Kommunikationen, der auf dem FGPA enthalten Einheiten
- über die UART könnten sich z.B. zwei ATmega Adressen für den Shared Memory austauschen



Konfiguration der FPGA

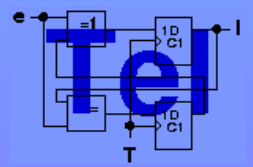
- die Konfiguration des FPGA wird von dem externen Programmierboard übernommen
- es werden vorcompilierte Einheiten mittels JTAG auf den FPGA gespielt (wurde in der Diplomarbeit von Herrn Niederlein behandelt)
- der Microkontroller soll jetz neu, auch die Programmierung des Befehlsspeichers übernehmen
- auf dem FPGA wurde eine BSCAN-Komponente implementiert, die die selben JTAG-PINS wie die Konfiguration des FPGA benutzt
- es sollen auch mehrere FPGA von dem selben Microkontroller gesteuert werden können



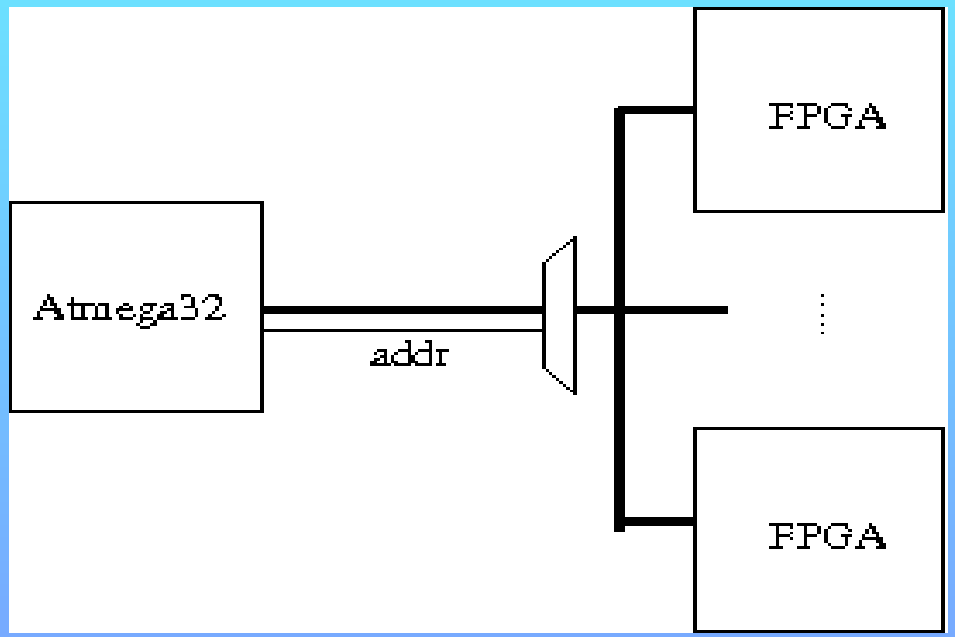
Konfiguration der FPGA

TODO

- Test der BSCAN-Komponente
- Test der FPGA-Konfiguration

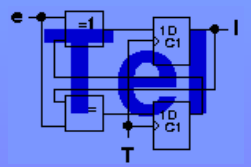


Konfiguration der FPGA



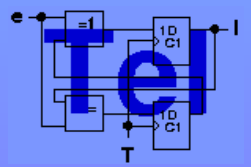
Test der einzelnen Einheiten

- Test und die damit verbundene Ausfallerkennung ist immer eine Frage des Aufwandes, denn irgendwann fällt alles einmal unerkennbar und unreparierbar aus
- ATmega32 besitzt einen Self Test, welcher über Interruptroutinen angesteuert wird
- universelle Testkomponente (Voraussetzung für IP)
- Auswertung vom FSP auf dem FPGA und evtl. Übermittlung nach außen zum Microkontroller
- gegenseitige Überwachung der beiden Serviceprozessoren



Universelle Testkomponente

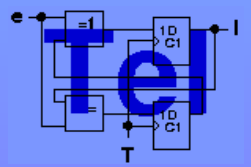
- abgespeckter ATmega32
- nach ersten Überlegungen hängt vor jeder Komponente eine universelle Testkomponente
- jede einzelne Einheit auf dem FPGA registriert sich bei seiner Testkomponente
- die Testkomponente führt den jeweiligen Test durch
- im Falle des ATmega32 ist es ein einfaches Interrupt-Signal, dieser gibt ein Testergebnis zurück



Universelle Testkomponente

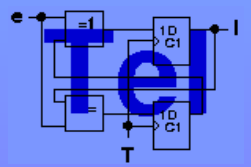
TODO

- Testkomponente für andere Einheiten bereit machen, so z.B. FSI, Schrittmotorsteuerung, Sensorüberwachung, oder FPU
- Testszenarien für solche Einheiten überlegen
- Test und Überwachung der Testkomponenten regeln
- Implementation der Testkomponenten in das feste FPGA-Design, somit Abkopplung von den einzelnen Einheiten



Reorganisation

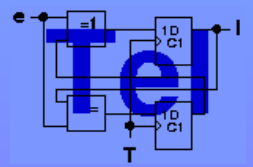
- je nach gewählter Aufgabe wird eine nicht mehr benötigte Einheit entfernt und eine neue implementiert
- dazu ist sehr viel Logik und Voraussicht, beim Empfang des Programmcodes notwendig (schwierig)
- somit ist immer noch ein Eingriff des Systemerstellers notwendig, beim erstellen des Programmcodes wird schon festgelegt welche Einheit dafür benötigt wird (anders ist dies schwer realisierbar)
- wird in einer theoretischen Betrachtung hauptsächlich behandelt



Reorganisation

TODO

- Einheitendatenbank vervollständigen und abschließen
- Allgemeingültige Schnittstellenbeschreibung erstellen, damit auch andere fremde IP-Komponenten verwendet werden könnten

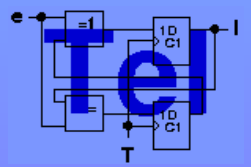


Fehlerbehandlung

- wird ein Fehler in einer Einheit erkannt, kann der FSP die Neuprogrammierung des FPGA veranlassen
- fällt ein FSP aus, übernimmt der Zweite diese Arbeit und meldet dem externen Microkontroller den Ausfall
- sind evtl. doppelte Komponenten vorhanden, kann eine zweite die Arbeit der ausgefallenen übernehmen
- Hauptaufgabe besteht erst einmal in einer korrekten Fehlererkennung und Überwachung des Systems

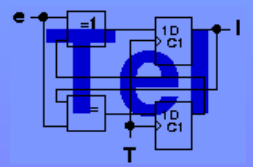
TODO

- Übernahme der Aufgabe einer doppelten Komponente



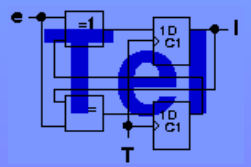
Was ist geschafft?

- VHDL-Design des Hauptkerns
- VHDL-Design des FSI
- VHDL-Design der FSP mit BSCAN-Komponente
- VHDL-Design einer FPU, die an den ATmega, über dessen I/O-Ports angeschlossen werden kann
- Entwurf eine Test-Komponente für den ATmega32
- VHDL-Parser, zum Einlesen der VHDL-Struktur
- Kommunikation zweier ATmega über die UART-Komponenten und den Shared Memory



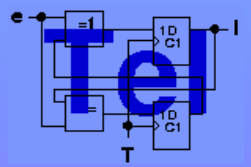
Was muss noch gemacht werden? Die nächsten Schritte!

- ausgiebiger Test der vorhandenen VHDL-Designs
- weitere Anpassungen des ATmega32-Designs an andere Aufgaben (Schrittmotorsteuerung, oder Sensorüberwachung)
- Fertigstellung des VHDL-Parsers
- eine interne Kommunkationsstruktur entwerfen, welche nicht nur zwei, oder mehrer ATmega32 verbinden kann, sondern auch einzelne Einheiten, die von der Kommunikation nie etwas mitbekommen (z.B. arbeitet der ATmega 32 mit der FPU)
- Verfeinerung und Erweiterung der Testkomponente, welche universell für alle Einheiten eingesetzt werden kann



Was muss noch gemacht werden? Die nächsten Schritte!

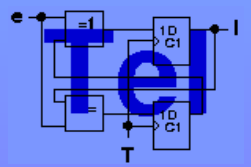
- theoretische Betrachtung der Kommunikation mehrerer FPGA untereinander und eine evtl. Entfernung des externen Microkontrollers
- theoretische Betrachtung der Dezentralisierung der "Gehirne" (FSP), so dass dies Aufgabe ein anderer implementierter ATmega32 kann
- es muss noch untersucht werden, wie eine doppelte Komponente am schnellsten die Arbeit einer ausgefallenen Komponente übernehmen kann, ohne den FPGA neu konfigurieren zu müssen



Erfüllung der Self-X Eigenschaften

"Die Voraussetzung zur Erfüllung der Self-X-Eigenschaften ist eine hochdynamische Kultur, in der es keinerlei starre Strukturen, keine zentralen Instanzen und auch kein zentrales Wissen gibt."

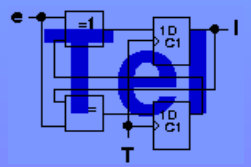
- selbst-konfigurierend
  - Hinzunahme, oder Entfernen einer Einheit
  - Umorganisation der Einheiten
- selbst-heilend
  - ersetzen einer ausgefallenen Einheit durch eine neue Einheit
  - Umorganisation der Einheiten, eine andere Einheit übernimmt die Aufgabe



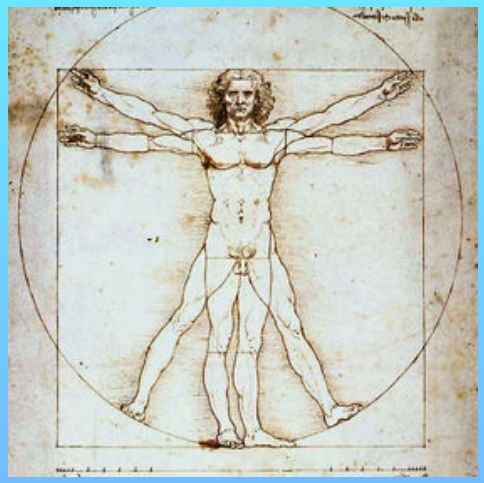
Erfüllung der Self-X Eigenschaften

"Die Voraussetzung zur Erfüllung der Self-X-Eigenschaften ist eine hochdynamische Kultur, in der es keinerlei starre Strukturen, keine zentralen Instanzen und auch kein zentrales Wissen gibt."

- selbst-optimierend
  - erlaubt selbstständige Anpassung an die Umwelt und Aufgabe
  - Umorganisation der Einheiten für andere Aufgabenbereiche
- selbst-schützend
  - ständige Überwachung der Einheiten untereinander
  - Selbsttest der einzelnen Einheiten und gegenseitige Auswertung



Der vitruvianische Mensch



"Der Mensch versucht die biologischen Eigenschaften der Natur zu kopieren und dabei zu verbessern, wird deren Komplexität aber nie erreichen."

Fragen ??