Gesellschaft für Informatik e.V. (GI)

Curriculum Technische Informatik in Bachelor- und Masterstudiengängen Informatik

Fachbereich
Technische Informatik

GI-Empfehlungen
Stand: Mai 2011

Curriculum Technische Informatik in Bachelor- und
Master-Studiengängen Informatik
Empfehlung der Gesellschaft für Informatik e. V.

Arbeitskreis Curriculum Technische Informatik

Christian Hochberger, TU Dresden

Wolfgang Karl, Karlsruher Institut für Technologie (KIT)

Reinhold Kröger, FH Wiesbaden

Erik Maehle, Universität zu Lübeck (Sprecher FB Technische Informatik)

Peter Marwedel, TU Dortmund

Uwe Schmidtmann, FH Oldenburg-Ostfriesland-Wilhelmshaven

Klaus Waldschmidt, Goethe-Universität Frankfurt am Main

Der Fachbereich Technische Informatik hat dieses Curriculum am 17. Juni 2010 beschlossen und am 31. Mai 2011 in seiner vorliegenden Form verabschiedet.

Das Präsidium der GI hat in seiner Sitzung vom 24./25. Juni 2010 diese Empfehlung zustimmend zur Kenntnis genommen.

Gesellschaft für Informatik e. V. (GI)
Wissenschaftszentrum
Ahrstraße 45
D-53175 Bonn
www.gi-ev.de

Kontakt: Fachbereich Technische Informatik
maehle@iti.uni-luebeck.de

Inhaltsverzeichnis

Vorwort

Die Gesellschaft für Informatik e.V. (GI) sieht sich in besonderer Verantwortung, Empfehlungen für die Ausgestaltung von Informatikstudiengängen an Universitäten und Fachhochschulen auszusprechen und damit einen Beitrag zur Qualitätssicherung zu leisten.

Im Jahr 2005 wurde in Kooperation mit dem Fakultäten- und Fachbereichstag Informatik eine Empfehlung für Bachelor- und Masterprogramme im Studienfach Informatik herausgegeben. Sie wurde wie die erste Version aus dem Jahr 2000 bundesweit von Hochschulen und Akkreditierungsagenturen genutzt und hat sich als sehr erfolgreich und einflussreich erwiesen. Ergänzend zu diesen umfassenden Empfehlungen für das gesamte Informatikstudium existieren bereits eine Reihe fachspezifischer GI-Empfehlungen, z. B. zu den Themenbereichen Mensch-Computer-Interaktion oder IT-Sicherheit sowie zum Studienplan Wirtschaftsinformatik.

Solche fachspezifischen Empfehlungen liegen jetzt hiermit auch für das Curriculum in Technischer Informatik in Bachelor- und Masterstudiengängen Informatik an Hochschulen vor. Sie wurden von Experten der Technischen Informatik in einem Arbeitskreis des zuständigen GI-Fachbereichs Technische Informatik (TI) ausgearbeitet und mit dem GI-Fachbereich Informatik und Ausbildung/Didaktik der Informatik (IAD) sowie dem Fakultätentag Informatik abgestimmt.

Die einzelnen Hochschulen weisen heute im Zuge der politisch gewünschten Profilbildung unterschiedliche Profile und Schwerpunkte auf. Die vorliegenden Empfehlungen stellen daher nur eine Orientierungshilfe zu den Lehrinhalten und -formen in Technischer Informatik dar, von denen je nach Profil der Hochschule selbstverständlich abgewichen werden kann.

Der Übergang der Technischen Informatik zu den anderen Informatikdisziplinen ist nicht immer scharf umrissen, sondern teilweise fließend. Hier wurden diejenigen Themengebiete mit einbezogen, die auch im Fachbereich TI vertreten sind. Insbesondere Rechnernetze und Betriebssysteme werden allerdings nicht überall der Technischen Informatik zugerechnet, sondern sind in anderen Bereichen, wie z. B. der Praktischen Informatik, angesiedelt. In diesen Fällen ist das empfohlene Curriculum im Kontext der jeweils gegebenen Struktur des Informatikstudiums zu verstehen.

Ich hoffe, dass die vorliegenden Empfehlungen zur Ausbildung in Technischer Informatik eine weite Verbreitung finden werden und zur Qualität der Lehre an unseren Hochschulen beitragen mögen.

Prof. Dr. Stefan Jähnichen, Präsident der Gesellschaft für Informatik e.V. (GI)

1. Einleitung

1.1 Motivation

Die Technische Informatik ist eines der Hauptgebiete der Informatik, weil sie die technischen Grundlagen für viele andere Informatikdisziplinen legt. Entsprechend ist sie in den Curricula von Informatikstudiengängen fest verankert. Aufgrund neuer Anwendungen und Systemanforderungen einhergehend mit der schnellen technologischen Entwicklung und der dadurch bedingten stark wachsenden Verbreitung von Rechensystemen, insbesondere eingebetteten und ubiquitären Systemen, wird sie in Zukunft weiter an Bedeutung gewinnen.
Im Zuge des Bologna-Prozesses und dem damit verbundenen Übergang zu Bachelor- und Masterstudiengängen sind einerseits Mobilität zwischen Hochschulen im In- und Ausland sowie Beachtung von Akkreditierungsrichtlinien, anderseits aber auch Profilbildung, Flexibilität und die Anpassung an sich wandelnde fachliche Entwicklungen gefordert. Die Curricula müssen daher stetig weiterentwickelt und angepasst werden. In diesem Prozess möchte die vorliegende Empfehlung eine Orientierungshilfe geben, welche Themengebiete der Technischen Informatik in einem modernen Informatikstudium unverzichtbar enthalten sein sollten und exemplarisch wichtige fakultative Bereiche nennen. Erarbeitet wurden diese Empfehlungen von einem Arbeitskreis des Fachbereichs Technische Informatik (FB TI) der Gesellschaft für Informatik.

1.2 Zielgruppen

Hauptzielgruppe des Curriculums sind Studierende in Bachelor- und Masterstudiengängen in Informatik an Universitäten und Fachhochschulen. Hierbei wird entsprechend der Klassifikation der GI je nach Informatikanteil in Studiengänge vom Typ 1, Typ 2 oder Typ 3 differenziert [1]. Teile des Curriculums können aber auch auf den Anteil in Technischer Informatik in Nicht-Informatikstudiengängen Anwendung finden, wobei natürlich die Randbedingungen des jeweiligen Faches zu berücksichtigen sind. Diese werden z. B. in den Ingenieurwissenschaften anders sein als in den Naturwissenschaften. Weiterhin kann das Curriculum als ein Leitfaden für Weiterbildungsangebote in der Industrie oder anderen Aus- und Fortbildungseinrichtungen dienen. Nicht explizit abgedeckt sind eigenständige Studiengänge in Technischer Informatik. Das hier empfohlene Curriculum kann aber als ein Kern dienen, um den sich weitere Themengebiete der Technischen Informatik gruppieren können. Für die Informatikausbildung an Schulen sei auf die zugehörigen Empfehlungen der GI verwiesen [2], in denen die entsprechenden Inhalte unter ‚Informatik der Systeme' enthalten sind.

1.3 Definition Themengebiete

Die vorliegenden curricularen Empfehlungen beziehen sich bewusst nicht auf konkrete Lehrveranstaltungen oder Module, sondern auf inhaltlich zusammengehörige Themengebiete. Die Umsetzung dieser Inhalte in einzelne Lehrveranstaltungen und deren Gruppierung zu Modulen soll den einzelnen Hochschulen überlassen bleiben, wobei mehrere Themengebiete zu einem größeren Modul oder auch Teile verwandter Themengebiete zu einer gemeinsamen Veranstaltung zusammengefasst werden können.. Wichtig ist dabei vor allem, dass alle genannten Pflichtthemengebiete abgedeckt sind, nicht in welcher Form dies im Einzelnen geschieht.

2. Einführung

2.1 Technische Informatik

Technische Informatik beschäftigt sich mit der Architektur, dem Entwurf, der Realisierung, der Bewertung und dem Betrieb von Rechensystemen sowohl mit Hardware als auch systemnaher Software. Architektur ist dabei die Zusammenfassung der Prinzipien für die Konstruktion von Rechensystemen mit Hardware und Software. Ziele sind insbesondere Funktionalität, Leistungsfähigkeit, Zuverlässigkeit, Sicherheit, Effizienz und Wirtschaftlichkeit. Die methodischen und theoretischen Grundlagen sowie die in der Praxis anzuwendenden Verfahren und die entsprechenden Werkzeuge für die Konstruktion und Bewertung von Rechensystemen und ihren Einsatz in den verschiedenen Anwendungsbereichen werden behandelt. Die Anforderungen der einzelnen Anwendungsbereiche liefern Anregungen und Zielvorgaben für Weiterentwicklungen. Diese Randbedingungen sind zudem Maßstab für die Tragfähigkeit von Methoden, Verfahren und Hilfsmitteln.

2.2 Lehrformen

Vermittelt werden die Inhalte der Technischen Informatik wie an Hochschulen üblich in Form von Vorlesungen, Übungen, Praktika und Projekten, die im Zuge der Modularisierung der Bachelor- und Masterstudiengänge zu Modulen zusammengefasst werden, wobei ein Modul eine oder mehrere Vorlesungen, Übungen, Praktika und ggf. Projekte enthalten kann. Eine wichtige Rolle spielen in der Technischen Informatik Werkzeuge, die z. B. zum Hardware- und Softwareentwurf oder zur Bewertung von charakteristischen Eigenschaften wie Leistung und Zuverlässigkeit eingesetzt werden. Besonders in den Übungen, Praktika und Projekten sollten diese Werkzeuge Verwendung finden, damit die Studierenden Kompetenzen mit deren praktischen Einsatz erwerben können. Oft bietet es sich daher an, die Übungen als praktische Übungen im Labor, in Form von Praktika oder als Projekte durchzuführen. Zu den Themengebieten existiert eine Vielzahl von Fach- und Lehrbüchern, die zur Vertiefung des Lehrstoffes herangezogen werden sollten. Methoden des E-Learning, multimediale Lehrformen und andere neue Lehr- und Lernformen können an geeigneten Stellen integriert werden. Inhalte können dabei sowohl in induktiver als auch deduktiver Lehrform vermittelt werden.

3. Bachelorstudiengänge

3.1 Struktur und Einordnung

Bachelorstudiengänge besitzen in der Regel einen relativ hohen Anteil von Pflichtveranstaltungen, in denen die fachlichen Grundlagen vermittelt werden und verfügen nur über einen relativ kleinen Wahlpflichtbereich, in dem die Studierenden sich weiter vertiefen können. Im Folgenden werden daher die Themengebiete der Pflichtveranstaltungen ausführlich wiedergegeben. Die dort aufgeführten Inhalte sollten für alle Informatikstudierenden eine unverzichtbare Grundlage sein. Unterschieden wird allerdings nach dem Typ des Studiengangs nach GI-Richtlinien [1], da mit dem unterschiedlichen Anteil an Informatikinhalten natürlich auch der Anteil an Technischer Informatik entsprechend angepasst werden muss:

Typ 1: Vollstudium Informatik
Typ 2: spezieller Anwendungsbereich, ca. 40 - 50 % Informatik
Typ 3: andere gleichwertige beteiligte Fachdisziplin, ca. 30 - 40 % Informatik.

Die Inhalte orientieren sich dabei sowohl an den Erfordernissen eines weiterführenden Masterstudiums als auch an den Anforderungen einer Berufsqualifizierung.

Abb. 1 gibt einen schematischen Überblick über die einzelnen Themengebiete und deren Abhängigkeiten.

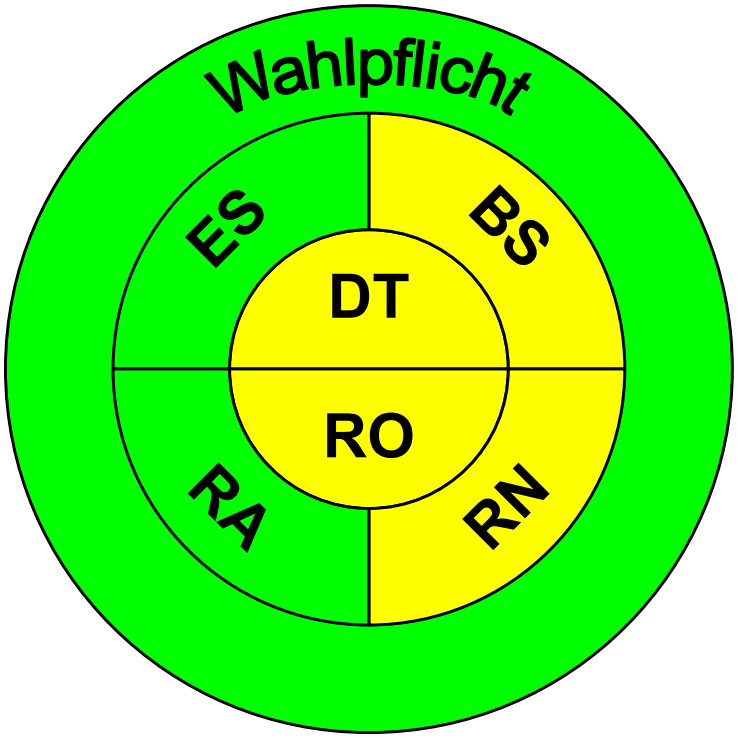

Abb. 1: Struktur des Curriculums Technische Informatik im Bachelorstudiengang

Die Grundlagen in den ersten Semestern legen die Themengebiete Digitaltechnik (DT) und Rechnerorganisation (RO), welche den Kern des Curriculums in Technischer Informatik bilden. DT beschäftigt sich mit digitalen Schaltungen auf Gatter- und Registertransferebene und legt damit die Grundlagen für den Hardwareentwurf, während RO grundlegende Prozessorkonzepte und deren Programmierung auf Assemblerebene behandelt. RO sollte dabei für alle Studiengangtypen verpflichtend sein, DT dagegen nur für Typ 1-Studiengänge. Zur praktischen Umsetzung des in DT und RO vermittelten Stoffes bieten sich Laborübungen an. Diese können entweder in den Übungsbetrieb integriert sein oder je nach Möglichkeiten und Profil der Hochschule auch als ein eigenständiges Praktikum vorgesehen werden. Denkbar ist auch ein Modul ‚Technische Grundlagen der Informatik', der die Inhalte der Digitaltechnik und Rechnerorganisation und ein darauf abgestimmtes, eigenständiges oder in den Übungsbetrieb integriertes Praktikum zusammenfasst. Der Gesamtumfang dieses Kernbereichs mit DT, RO inklusive praktischem Anteil sollte bei Typ 1-Studiengängen bei ca. 12 bis 16 ECTS liegen.
Ebenfalls zum Pflichtbereich gehören Betriebssysteme (BS) und Rechnernetze (RN) im Umfang von zusammen ca. 8 bis 12 ECTS.

Weiterführende Themengebiete in den höheren Semestern sind Eingebettete Systeme (ES) und Rechnerarchitektur (RA) mit einem Gesamtumfang von ca. 8 bis 10 ECTS. Beide bauen auf RO auf, sind aber unabhängig voneinander. ES und RA sollten sowohl für Typ 1- als auch für Typ 2-Studiengänge zum Wahlpflichtangebot gehören, können aber je nach Profil der Hochschule und dem damit zur Verfügung stehenden Anteil an Technischer Informatik auch im Pflichtangebot enthalten sein. Für Typ 3-Studiengänge kann dagegen je nach fachlicher Ausrichtung des Studiengangs eines der beiden Themengebiete selektiert werden. Beispielsweise dürfte sich für Mechatronik eher Eingebettete Systeme und für Wissenschaftliches Rechnen eher Rechnerarchitektur eignen.
Bei Typ 2- und Typ 3-Studiengängen kann es je nach Ausrichtung wünschenswert sein sowohl Inhalte aus der Digitaltechnik als auch der Rechnerorganisation gemeinsam zu vermitteln. Hier bietet sich eine eigene, auf den Studiengang zugeschnittene Einführungsveranstaltung in die Technische Informatik an, die nicht zugleich für Studierende aus Typ 1-Studiengängen geeignet ist.

Zum Wahlpflichtbereich gehören noch weitere Veranstaltungen, in denen sich die Studierenden in einzelnen Gebieten der Technischen Informatik vertiefen können. Diese können in der Regel alternativ zu Vertiefungsveranstaltungen anderer Informatikgebiete ausgewählt werden.

Während die Pflichtveranstaltungen an allen Hochschulen in etwa gleiche Inhalte vermitteln und damit auch einen Wechsel erleichtern, orientieren sich die Wahlpflichtveranstaltungen in der Regel an dem Profil der Hochschule. Im Folgenden sind daher die Empfehlungen für den Pflichtbereich recht detailliert ähnlich wie in einem Modulhandbuch wiedergeben, gegliedert nach Lernzielen, Inhalten und Übungen/Praktika. Letztere geben exemplarisch an, wie der gelehrte Stoff vertieft und gefestigt werden kann. Insgesamt soll eine Orientierung gegeben werden, wie die einzelnen Themengebiete ausgestaltet werden können.

3.2 Themengebiete im Pflichtbereich

3.2.1 Digitaltechnik

Die Informatik basiert ganz wesentlich auf digitalen Schaltungen zur technischen Realisierung konkreter Systeme der Datenverarbeitung. Ein fundiertes Verständnis der Digitaltechnik und darauf aufbauender Schaltungen zur Datenverarbeitung ist daher unabdingbar für eine vollständige Informatikgrundausbildung.

Lernziele

Studierende sollen durch dieses Themengebiet folgende Kompetenzen erwerben:

- Verständnis der verschiedenen Darstellungsformen von Zahlen und Alphabeten in Rechnern,
- Fähigkeiten der formalen und programmiersprachlichen Schaltungsbeschreibung,
- Kenntnisse der technischen Realisierungsformen von Schaltungen,
- basierend auf dem Verständnis für Aufbau und Funktion aller wichtigen Grundschaltungen und Rechenwerke die Fähigkeit, unbekannte Schaltungen zu analysieren und zu verstehen, sowie eigene Schaltungen zu entwickeln,
- Verständnis für die digitaltechnische Umsetzung der in anderen Veranstaltungen (Rechnerorganisation, Rechnerarchitektur und Eingebettete Systeme) behandelten Systeme.

Inhalte

- Theoretische Grundlagen
  - Informationsdarstellung, Zahlensysteme
  - Binärdarstellungen negativer Zahlen, Gleitkomma-Zahlen
  - Alphabete, Codes
- Elektronikgrundlagen der Technischen Informatik
  - einfache elektrische Zusammenhänge wie Ohmsches Gesetz, Kondensatorgleichung
  - MOS-Transistoren, CMOS-Schaltungen, Energieeffizienz
  - Tri-State und Open Drain Schaltungen
- Gatterebene
  - Boolesche Algebra/Schaltalgebra, Normalformen
  - Schaltfunktionsoptimierung (zweistufig und mehrstufig, z.B. KV-Diagramme, Quine-McCluskey, Notwendigkeit heuristischer Verfahren)
  - Schaltnetze, Grundschaltungen (Multiplexer, Halb/Voll-Addierer)
  - Schaltwerke, Automatentypen (FlipFlops, Mealy und Moore)
  - Automatenrealisierungen, Zustandskodierung
  - Realisierungsformen von digitalen Schaltungen: Gatter, PLDs, FPGAs, ASICs
- Entwurf auf Register-Transfer-Ebene
  - Hardware-Beschreibungssprachen
  - Schaltungssimulation, Delay-Modelle
  - Synthese
- Rechenwerke
  - Addierer-Varianten (Ripple-Carry, Carry-Look-Ahead, Carry-Save), Realisierung der Subtraktion
  - Multiplizier-Schaltungen (parallel, seriell, Block-Multiplizierer), Divisionsschaltungen
- Steuerwerke
  - Binär codiert
  - 1-aus-n codiert
- Speichertechnologien
  - Halbleiterspeicher
  - magnetische und optische Speicherung

Übungen

- Umwandlung verschiedener Zahlensysteme ineinander
- Rechnungen mit binär dargestellten Zahlen
- Analyse einfacher Transistorschaltungen: Logische Funktion, Schaltzeitpunkt, Stromverbrauch
- Minimierung von Schaltfunktionen, am besten rechnergestützt
- Verschiedene Realisierungen von Schaltnetzen: mit Gattern, mit PLDs, mit FPGAs
- Analyse einfacher Digitalschaltungen bzgl. Funktion und Zeitverhaltens
- Entwurf von Digitalschaltungen
- Simulation einfacher Schaltungen mit interaktiven graphischen Schaltungssimulatoren
- Darstellung und Simulation einfacher Schaltungen mit Hardware-Beschreibungssprachen (z.B. serieller oder Block-Multiplizierer)

3.2.2 Rechnerorganisation

Kenntnisse grundlegender Prozessorkonzepte und der Assemblerprogrammierung sind die Voraussetzungen für eine sinnvolle und ressourcenbewusste Nutzung von Rechnerstrukturen.

Lernziele

Studierende sollen durch dieses Modul folgende Kompetenzen erwerben:

- die Fähigkeit, einen einfachen Rechner aus Grundkomponenten komponieren zu können, dies insbesondere am Beispiel eingebetteter Systeme. Dazu werden folgende Kenntnisse benötigt:
  - Verschiedene Realisierungsformen komplexer Schaltungen,
  - die relevanten Speichertechnologien,
  - Verständnis für Aufbau und Programmierung von Prozessoren,
- die Fähigkeit, spezifische Eigenschaften des Rechners bei der Programmierung zu berücksichtigen,
- Voraussetzung für das Verständnis weitergehender Veranstaltungen besitzen (Rechnerarchitektur, Betriebssysteme und Eingebettete Systeme).

Inhalte

- Steuerwerke
  - Fest verdrahtete Steuerwerke, mikroprogrammierte Steuerwerke
- Grundlegende Rechnerorganisation
  - Von-Neumann Rechnermodell, Assemblerprogrammierung, Befehle, Adressierungsarten
  - grundlegende Befehlssatzarchitekturen: Akkumulator oder Einadressmaschine, Register-Maschine (mit zwei oder drei Adressen), Stackmaschinen (Null-Adressmaschine)
  - Befehlsabarbeitung, Interrupts
  - Einfache Ein-/Ausgabe Schaltungen (z.B. binäre Ports, serielle Schnittstellen und Timer, Analog-Digital Umsetzer)
  - Fallstudie (z.B. Mikrocontroller)
- Diskussion RISC/CISC-Architektur
- Pipelining des Maschinenbefehlszyklus
  - Pipeline-Hemmnisse
  - Methoden zur Auflösung von Pipeline-Konflikten
- Speicherhierarchien, Speicherorganisation
  - Cache-Speicher, Cache-Organisation, Aktualisierungsstrategien
- Unterstützung von Betriebssystemfunktionen
  - Speicherverwaltung
  - Schutzmechanismen
  - Virtualisierung
- Ein-/Ausgabe-System, Schnittstellen, Interrupt-Verarbeitung
- Bus-Systeme

Übungen

- Realisierung eines Steuerwerkes als Hardware-Steuerwerk mittels One-Hot oder binärer Kodierung und als Mikroprogramm-Steuerwerk

- Erstellen einfacher, exemplarischer Assemblerprogramme (möglichst für alle verschiedenen Befehlssatzarchitekturen, auf dem Papier oder anhand entsprechender Simulatoren)
- Vergleich des Verhaltens und quantitative Bewertung verschiedener Cache-Organisationsformen
  - Berechnen des Zugriffsverhaltens von Cache-Speichern
  - Berechnen von Fehlzugriffsraten bzw. Trefferraten
- Pipeline-Organisation
  - Berechnen von Beschleunigungsfaktoren beim Einsatz von Pipelines
  - Erkennen und Auflösen von Pipeline-Konflikten in Beispielprogrammen
- Sprungvorhersage
  - Vergleich des Verhaltens verschiedener Sprungvorhersagetechniken
- Optimierung von RISC-Programmen zur Vermeidung von Pipeline-Konflikten
- Ermitteln physikalischer Adressen anhand gegebener segmentierter virtueller Adressen und der Seitentabelle

3.2.3 Laborübungen zu Digitaltechnik und Rechnerorganisation

Der Stoff aus den Themengebieten Digitaltechnik und Rechnerorganisation sollte möglichst durch Laborübungen, entweder integriert in den Übungsbetrieb oder als eigenständiges Praktikum, vertieft werden. Durch die praktische Umsetzung von Aufgaben wird die tatsächliche Relevanz des in der Vorlesung vermittelten Stoffes noch einmal betont. Dabei können auch bereits weiterführende Themengebiete anklingen.

Lernziele

Generelles Lernziel von Praktika und Laborübungen ist die Elaboration des in Vorlesungen vermittelten Wissens anhand einer praktischen Umsetzung des Stoffes. Studierende sollen speziell zu den Themengebieten Digitaltechnik und Rechnerorganisation folgende Fähigkeiten erwerben:

- Den Entwurf und die Integration einfacher Hardware-Komponenten in ein bestehendes System beherrschen.
- Ein Gefühl für die Komplexität und die Probleme bei der Umsetzung einfacher Programmieraufgaben in Assembler entwickeln.
- Interrupts und Ausnahmen behandeln können.
- Die Nutzung einfacher Ein-/Ausgabeschnittstellen z.B. zur Implementierung von Steuerungen/Regelungen beherrschen.

Inhalte

Die hier geplanten Laborübungen bzw. Praktika können grundsätzlich in zwei verschiedenen Konstellationen ablaufen.

In der einen Konstellation wird für alle Aufgabenteile eine einheitliche Plattform zur Realisierung von Hardware-Komponenten, zur Assemblerprogrammierung und zur Ein/Ausgabe verwendet. In diesem Fall bietet es sich an, als Plattform FPGAs einzusetzen, da diese sowohl zur Realisierung vollständiger Systeme geeignet sind als auch die nachträgliche individuelle Realisierung von zusätzlichen Hardware-Komponenten ermöglichen.

In der anderen Konstellation verwendet man zur Realisierung der einzelnen Aufgabenteile verschiedene Realisierungsplattformen, um unterschiedliche Schwerpunktsetzungen zu

ermöglichen. Dabei können Standardschaltkreisbausteine, FPGAs und fertige Mikrocontroller-Systeme zum Einsatz kommen. Es sollte auch die Gelegenheit geboten werden, mit Entwurfswerkzeugen und typischen Messgeräten aus dem Hardware-Entwurfsbereich zu arbeiten (einfache Messgeräte, Oszilloskop, Logikanalysator).

In beiden Fällen sind nur elementare Kenntnisse der Elektrotechnik erforderlich, wie sie in der Vorlesung Digitaltechnik leicht vermittelt werden. Der Schwerpunkt liegt auf der digitalen Schaltungstechnik und der Assemblerprogrammierung unter Nutzung der zugehörigen Entwurfsverfahren und -werkzeuge.

Aufgabenteile

Konkret werden folgende Aufgabenteile empfohlen, aus denen sich einzelne Laborübungen oder ein eigenständiges Praktikum zusammensetzen sollten:

- Entwurf und Realisierung einfacher Schaltnetze und Schaltwerke
- Entwurf und Integration von Hardware-Komponenten, z.B. UART, ALU, Befehlsdekoder, Adress-Rechenwerk, Interrupt-Controller, ...
  - Entwurf und Simulation einer reinen Verhaltensbeschreibung einer Prozessorkomponente
  - Synthese und Post-Layout-Simulation
  - Inbetriebnahme und Test der realisierten Schaltung
- Assemblerprogrammierung
  - Einfache Assemblerprogramme zur Realisierung von z.B. arithmetischen Funktionen, Textverarbeitung, Display-Steuerung, einfache Eingebettete Systeme
  - Interruptprogrammierung
- Ein-/Ausgabe-Programmierung
  - Bedienung einer E/A-Komponente per Polling
  - Realisierung einer einfachen Steuerung/Regelung
  - Kooperation einer Kontrollschleife und einer interruptgesteuerten Eingabeverarbeitung

3.2.4 Betriebssysteme

Betriebssysteme sind komplexe, nebenläufige Systeme, die dem Programmierer einerseits eine geeignete, langlebige Abstraktion der zugrunde liegenden Hardware zur Verfügung stellen und andererseits die Betriebsmittel eines Rechners effektiv verwalten.

Lernziele

In diesem Teil des Curriculums sollen die Studierenden folgende Kompetenzen erwerben:

- Sie besitzen ein grundlegendes Verständnis über Aufgaben, Architektur, Funktionsweise und Schnittstellen moderner Betriebssysteme.
- Sie kennen Mechanismen und Strategien von Betriebs- und Laufzeitsystemen und sind in der Lage, diese zu beurteilen und bewerten.
- Sie können Konzepte nachvollziehen, die bei der Konstruktion von Betriebssystemen Verwendung finden.

Insbesondere soll dieser Teil die Voraussetzung liefern, weiterführende Veranstaltungen etwa zu Betriebssystemkonstruktion, Echtzeitverarbeitung, Verteilten Systemen, Fehlertoleranz oder zur Leistungsbewertung erfolgreich zu besuchen und aktuelle Forschungsthemen zu verstehen.

Inhalte

Im Rahmen dieses Themengebiets werden die Basisdienste eines klassischen Betriebssystems vorgestellt:

- Einführung, Geschichte der Betriebssysteme, Arten und Einsatzbereiche von Betriebssystemen
- Systemkonzepte und -Strukturen
- Prozesse und Threads
- Scheduling
- Prozesssynchronisation
- Interprozesskommunikation
- Speicherverwaltung, Virtual Memory Management
- I/O-Management
- Dateisysteme
- Sicherheit und Schutzmechnismen
- Virtuelle Maschinen
- Beispiele, Fallstudien

Übungen/Praktikum

Die Übungen sollten, wenn möglich, praktische Anteile enthalten.

- Kennen lernen der Programmierschnittstelle (API) eines Betriebssystems, z.B. UNIX
- Praktische Erfahrungen in der Nutzung externer Schnittstellen von Betriebssystemen sammeln, z.B. durch Programmieraufgaben unter Nutzung der Systemdienste
- Praktische Erfahrungen in der Nutzung interner Schnittstellen von Betriebssystemen sammeln, z.B. Gerätetreiber und Interrupt Handler, Dateisystemtypen entwickeln
- Inkrementelle Entwicklung eines elementaren Betriebssystems auf einem PC-Emulator

3.2.5 Rechnernetze

Dieses Themengebiet gibt eine Einführung in das Gebiet der Rechnernetze, wobei alle Schichten des Kommunikationssystems betrachtet werden.

Lernziele

Die Studierenden sollen folgende Kenntnisse und Fähigkeiten erwerben:

- Sie kennen sehr gut die Bedeutung von Schichtenmodellen, die Aufgaben und Funktionsweise der verschiedenen Schichten sowie die wichtigsten Protokoll- und Dienstvertreter in jeder Schicht.
- Sie können für ein gegebenes Anwendungsproblem entscheiden, welche Netztechnologien in den verschiedenen Schichten eingesetzt werden sollten.
- Sie wissen, wie das Internet im Kern und in den Endsystemen funktioniert und sind in der Lage, eigene kleine Anwendungen zu programmieren. Sie sind sehr gut in der Lage, die Konzepte der Protokolle TCP, IP, HTTP und SMTP wiederzugeben.

Inhalte

Das Themengebiet beschäftigt sich mit folgenden Inhalten:

- Computernetzwerke und das Internet
- Anwendungsschicht

- Transportschicht
- Vermittlungsschicht
- Sicherungsschicht und Bitübertragung
- Zusammenfassung und Ausblick

Übungen/Praktikum

In den Übungen sollen die Studierenden das in der Vorlesung vermittelte Wissen sowohl anhand von Papierübungsaufgaben nachvollziehen als auch in kleineren Programmieraufgaben einüben. Beispiele für typische Aufgaben:

- Bandbreitenberechnungen für Medien
- Datenratenberechnungen für bestimmte Protokolle
- Nachvollziehen von Protokollen mit Message Sequence Charts
- Verwendung von telnet für verschiedene höherwertige Dienste wie Versenden von Emails
- Socket-Programmierung
- Programmieren eines kleinen http-Servers und -Clients
- Verwendung von Mail-APIs z.B. von Java

3.2.6 Rechnerarchitektur

Die Rechnerarchitektur ist eine allgemeine Strukturlehre mit deren Hilfsmitteln, die als ingenieurwissenschaftliche Disziplin die Aufgabe hat, bestehende und zukünftige Rechnersysteme zu beschreiben, zu bewerten, zu beurteilen und zu entwerfen. Sie betrachtet den Aufbau und die Eigenschaften des Ganzen (das Rechnersystem) sowie seiner Teile (Komponenten) und seiner Verbindungen (Globalstruktur, Infrastruktur). Die grundlegenden Kenntnisse der Hardwarestruktur eines Rechners auf einer konzeptionellen Darstellungsebene, die Sicht des Anwenders auf einen Rechner sowie das Operationsprinzip bilden die Voraussetzung für den effizienten Einsatz von Rechnern in den verschiedenen Anwendungsgebieten.

Lernziele

In diesem Teil des Curriculums sollen die Studierenden folgende Fähigkeiten und Kenntnisse erwerben:

- Sie besitzen ein grundlegendes Verständnis über den Aufbau, die Organisation und das Operationsprinzip von Rechnersystemen.
- Sie verstehen den Zusammenhang zwischen Hardware-Konzepten und den Auswirkungen auf die Software, um effiziente Programme erstellen zu können.
- Sie können aus dem Verständnis über die Wechselwirkungen von Technologie, Rechnerkonzepten und Anwendungen die grundlegenden Prinzipien des Entwurfs nachvollziehen und anwenden.
- Sie sind in der Lage, Rechnerkonzepte zu verstehen, zu bewerten und zu vergleichen.

Insbesondere soll dieser Teil die Voraussetzung liefern, vertiefende Veranstaltungen über moderne Mikroprozessorarchitekturen, Parallelrechner, Fehlertoleranz und Leistungsbewertung zu besuchen und aktuelle Forschungsthemen zu verstehen.

Inhalte

- Einführung in die Rechnerarchitektur, Klassifikation
- Grundprinzipien des Rechnerentwurfs: Kompromissfindung zwischen Zielsetzungen, Randbedingungen, Gestaltungsgrundsätzen und Anforderungen, Leistungskriterien
- Superskalartechnik, VLIW-Prinzip, spekulative Ausführung, Sprungvorhersage
- Mehrfädige Befehlsausführung
- Speichersysteme
  - RAID
  - Network-Attached Storage
  - Storage-Area-Networks (SAN)
- Parallelrechnerkonzepte, speichergekoppelte Parallelrechner (symmetrische Multiprozessoren, Multiprozessoren mit verteiltem gemeinsamen Speicher), nachrichtenorientierte Parallelrechner, Multicore/Manycore-Architekturen, parallele Programmiermodelle, Konsistenzmodelle und Kohärenzprotokolle
- Verbindungsnetze (Topologien, Routing)
- Grundlagen der Vektorverarbeitung, SIMD, Multimedia-Verarbeitung
- Leistungsbewertung
- Energie-effizienter Entwurf
- Grundlagen der Fehlertoleranz, Zuverlässigkeit, Verfügbarkeit, Ausfallsicherheit

Übungen/Praktikum

Die Übungen sollten, wenn möglich, auf parallelen Rechensystemen bzw. mit Hilfe geeigneter Simulationswerkzeuge (z. B. für Prozessorarchitekturen, Speicherhierarchien, Verbindungsnetze) durchgeführt werden.

- Grundlagen des Aufbaus von Rechnern: Herausarbeiten der Ausgewogenheit von Rechnerentwürfen
  - Prozessorgeschwindigkeit vs. Bus-Bandbreite / Latenz vs. Speicherkapazität / Speicherzugriffszeiten vs. E/A-System
- Parallelverarbeitung
  - Berechnen der Parameter des Amdahl-Gesetzes und Ableiten von Parallelitätsparametern
  - Berechnung von Speedup- und Effizienzwerten
  - Erstellen einfacher paralleler Programme (MPI, OpenMP) und Ausführen auf Simulator, parallelem System und Berechnen von Leistungsparametern
- Verbindungsstrukturen
  - Berechnen der Leistungsparameter verschiedener Verbindungsnetze und Vergleich der Ergebnisse
  - Wegeberechnungen für verschiedene Routing-Verfahren
- Rechnerbewertung, Leistungsbewertung
  - Berechnen von Parametern der CPI-Formel mit einfachen Beispielprogrammen
  - Berechnen von SPEC-Parametern an Fallstudien
  - Vergleich von Rechnern anhand von Benchmark-Daten
- Fehlertoleranz
  - Erstellen von Zuverlässigkeitsgraphen und Ermittlung der Systemfunktion
  - Berechnen von Funktions-, Fehler- und Ausfallwahrscheinlichkeiten

3.2.7 Eingebettete Systeme

Eingebettete Systeme sind informationsverarbeitende Systeme, die in ein umgebendes technisches System eingebettet sind. In der Regel interagieren eingebettete Systeme mit einer physikalischen Umgebung. Eingebettete Systeme müssen effizient mit Ressourcen umgehen, u.a. auch mit den Ressourcen „Zeit“ und „Energie“. Beispiele Eingebetteter Systeme finden sich u.a. im Verkehrwesen, in der Telekommunikation, in der Produktion, in der Medizintechnik und im Consumerbereich.

Lernziele

Dieser Teil des Curriculums soll die Studierenden in die Lage versetzen, einfache Eingebettete Systeme zu entwickeln und dazu insbesondere

- Unterschiede zwischen Desktop- und Eingebetteten Systemen zu kennen und berücksichtigen zu können,
- Spezifikations- und Modellierungstechniken für Eingebettete und Realzeitsysteme miteinander vergleichen und geeignete Techniken auswählen zu können,
- die Prinzipien einschlägiger Basistechnologien im Bereich der Hardware und Systemsoftware zu kennen und einsetzen zu können,
- Entwicklungstechniken theoretisch und praktisch zu beherrschen,
- Eingebettete Systeme beurteilen und optimieren zu können,
- Risiken in besonderen Bereichen beurteilen zu können,
- die Zuverlässigkeit bewerten zu können.

Inhalte

- Spezifikations- und Modellierungstechniken Eingebetteter Systeme, insbesondere
  - Modell-basierter Entwurf
  - Models of Computation
  - Techniken für daten- und kontrollflussdominierte Systeme
  - Techniken für lokale und heterogene verteilte Systeme
- Prinzipien von Hardware-Plattformen, insbesondere von
  - Hardwareklassen (applikationsspezifische Schaltkreise (ASICs), Field-Programmable Gate Arrays (FPGAs), programmierbare Prozessoren)
- Hardwarekomponenten (Analog/Digital- und Digital/Analogumsetzer einschl. Abtasttheorem, Prozessoren, Speicher, Peripheriebausteine)
- Kommunikationssysteme
  - Netzwerkanbindung
  - Echtzeitverhalten
- Prinzipien der Basissoftware
  - Echtzeit-Betriebssysteme
  - Middleware
- Abbildung von Anwendungen auf Verarbeitungsplattformen
  - Software-Engineering und Software-Entwicklung für Eingebettete Systeme
  - Grundzüge des Echtzeit-Scheduling
  - Abbildung auf (heterogene) Mehrprozessorsysteme
  - Werkzeugunterstützung
  - Debugging
  - Echtzeitverhalten von Anwendungsschichten
- Evaluation
  - Simulative und analytische Verfahren
  - Bewertung von Systemen z.B. hinsichtlich der Latenzzeit, des Energieverbrauchs
  - Laufzeitgarantien

- Optimierung
  - Anwendung von Optimierungstechniken auf den Entwurf Eingebetteter Systeme
  - Mehrkriterien-Optimierung
  - exemplarische Betrachtung spezieller Optimierungstechniken, z.B. von Codekompressionstechniken oder des Energiemanagements
- Grundlagen verlässlicher Eingebetteter Systeme (Sicherheit, Zuverlässigkeit, Verfügbarkeit)
- Test von Eingebetteten Systemen (z.B. Scan Design, JTAG)
- Exemplarische Betrachtung von Anwendungen

Übungen/Praktikum

- Spezifikation hierarchischer Zustandsautomaten und deren Umsetzung in Hardware oder Software
- Spezifikation von Systemen unter Nutzung von Echtzeit-Profilen z.B. von UML und Java
- Programmierung einfacher Beispielsysteme
- Programmierung mobiler Eingebetteter Systeme
- Datenerfassung physikalischer Daten (z.B. von Sensoren der Medizintechnik)
- Realisierung der Steuerung von technischen Systemen (z.B. Modelleisenbahnen, Kfz-Systeme)
- Ermittlung des Energiebedarfs
- Realisierung von Hardware-in-the-Loop-Systemen
- Praktische Demonstration von Auswirkungen der Abtastung
- Nutzung eines Echtzeit-Betriebssystems
- Lösung von Echtzeit-Scheduling-Aufgaben
- Erprobung der Effizienzsteigerung durch Programm-Optimierungen
- Konzeption und Durchführung von Tests eingebetteter Systeme
- Demonstration der Auswirkungen der Verlässlichkeitsanforderungen

3.3 Themengebiete im Wahlpflichtbereich

Exemplarisch seien hier einige mögliche Themengebiete für vertiefende Wahlpflichtveranstaltungen genannt. Je nach Profil der Hochschule können weitere Themen ergänzt werden.

- Parallelrechner/Parallelverarbeitung
- Robotik
- Rechnergestützter Schaltungsentwurf
- Test von Hardwareschaltungen
- Programmierung Eingebetteter Systeme
- Rechnertechnologie
- Automatisierung technischer Prozesse
- Quantitative Modellbildung und -analyse
- Technische Aspekte der IT-Sicherheit
- ...

4. Masterstudiengänge

Da Masterstudiengänge in der Regel weniger streng strukturiert sind als Bachelorstudiengänge und sich stark an den Forschungsschwerpunkten der jeweiligen Hochschulen orientieren, werden hier nur relative grobe Empfehlungen gegeben. Oft haben die Studierenden eine Vielzahl von Wahlmöglichkeiten und können sich das Curriculum nach ihren Neigungen und Vorlieben zusammenstellen. Im Folgenden wird zwischen zwei Bereichen unterschieden, aus denen jeweils Themengebiete ausgewählt werden können: der generische und der anwendungsbezogene Bereich. Dabei sind nicht nur Themengebiete aus dem Kernbereich der Technischen Informatik aufgeführt, sondern auch solche, in denen Technische Informatik eine wesentliche Rolle spielt. Damit wird dem Zusammenwachsen verschiedener Informatikgebiete, vor allem im Anwendungsbereich, Rechnung getragen.
Die nachfolgende Zusammenstellung ist exemplarisch gemeint, d. h. je nach Profil der Hochschule können noch weitere Themengebiete in beiden Bereichen hinzugefügt bzw. weggelassen werden.

4.1 Themengebiete generischer Bereich

In den generischen Bereich gehören Themengebiete, die eher methodischen Charakter haben und grundlegende Verfahren der Technischen Informatik auf hohem wissenschaftlichen Niveau behandeln. Die einzelnen Themengebiete können dabei jeweils durch mehrere Veranstaltungen, die einzelne Aspekte vertiefen, aufgeteilt sein.

- Systemmodellierung und -simulation
- Hardware/Software Co-Design
- Verlässlichkeit und Fehlertoleranz
- Leistungsanalyse und -bewertung (empirisch bzw. modellbasiert)
- Organic Computing
- Echtzeitsysteme
- Rekonfigurierbare Systemarchitekturen
- Hardwaretest und -verifikation
- E-CAD-Algorithmen
- ....

4.2 Themengebiete anwendungsbezogener Bereich

Der anwendungsbezogene Bereich behandelt den Einsatz von fortgeschrittenen Methoden der Technischen Informatik in konkreten Applikationen.

- Höchstleistungsrechnen
- Nichtkonventionelles Rechnen (Quanten-Computing, DNA-Computing etc.)
- Systemarchitekturen für Multimedia
- Digitale Signal- und Bildverarbeitung
- Ambient Intelligence
- Ubiquitäres Rechnen
- Drahtlose Sensornetze
- Industrieroboter
- Mobile Roboter
- Medizinische Roboter

...

5. Literatur

[1] Empfehlungen für Bachelor- und Masterprogramme im Studienfach Informatik an Hochschulen. Empfehlungen der Gesellschaft für Informatik 2005
http://www.gi-ev.de/fileadmin/redaktion/empfehlungen/GI-Empfehlung_BaMa2005.pdf

[2] Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards für die Sekundarstufe I. Empfehlungen der Gesellschaft für Informatik 2008
http://www.gi-ev.de/fileadmin/redaktion/empfehlungen/Bildungsstandards_2008.pdf

[3] Empfehlungen zur Einrichtung von konsekutiven Bachelor- und Masterstudiengängen in Informatik an Universitäten. Fakultätentag Informatik 2004
http://www.ft-informatik.de/uploads/tx_sbdownloader/bachelor_master_empfehlungen.pdf