Embedded Systems Grundlagen

Inhaltsverzeichnis

Einführung in eingebettete Systeme

Eingebettete Systeme (embedded systems) sind Computersysteme, die aus Hardware und Software bestehen und die in komplexe technische Umgebungen eingebettet sind. Diese Umgebungen sind meist maschinelle Systeme, in denen das eingebettete System mit Interaktion durch einen Benutzer arbeitet oder auch vollautomatisch (autonom) agiert. Die eingebetteten Systeme übernehmen komplexe Steuerungs-, Regelungs- und Datenverarbeitungsaufgaben für bzw. in diesen technischen Systemen.
Die eingebetteten Systeme zeigen dabei eine große Spannweite, denn es ist ein großer Unterschied, eine Kaffeemaschine oder ein Flugzeug zu steuern. Zunächst muss also einmal klassifiziert werden, um die Vielfalt zu beherrschen, und dann werden bestimmte Teile näher behandelt.
Es ist zugleich deutlich, dass es sich um ein interdisziplinäres Feld handelt, da viele Komponenten hier hineinspielen werden. Im ersten Kapitel werden einige Begriffe definiert, um für Einheitlichkeit zu sorgen.

Klassifizierung

Definition:
Ein *eingebettetes System (embedded system)* ist ein binärwertiges, digitales System (Computersystem), das in ein umgebendes technisches System eingebettet ist und mit diesem in Wechselwirkung steht.

Das Gegenstück zu Embedded System wird Self-Contained System genannt. Als Beispiele können Mikrocontroller-basierte Systeme im Auto, die Computertastatur usw. genannt werden.

Allgemeine Klassifizierung von Computersystemen

Die heute verfügbaren Computersysteme können in drei unterschiedliche Klassen eingeteilt werden: (rein) transformationelle, interaktive und reaktive Systeme. Die Unterscheidung erfolgt in erster Linie durch die Art und Weise, wie Eingaben in Ausgaben transformiert werden.

Transformationelle Systeme transformieren nur solche Eingaben in Ausgaben, die zum Beginn der Systemverarbeitung vollständig vorliegen. Die Ausgaben sind nicht verfügbar, bevor die Verarbeitung terminiert. Dies bedeutet auch, dass der Benutzer bzw. die Prozessumgebung nicht in der Lage ist, während der Verarbeitung mit dem System zu interagieren und so Einfluss zu nehmen.

Merkmale:

- Verarbeitung ist **nicht dialogorientiert**
- Benutzer gibt Daten ein → System verarbeitet → Ergebnis wird ausgegeben
- Keine kontinuierliche Rückmeldung oder Steuerung durch den Benutzer
- Oft **stapelverarbeitet** (Batch Processing)

Beispiele:

- Compiler (Quellcode → Maschinencode)
- Bildkonverter (JPEG → PNG)
- Steuerberechnungssoftware, die nach Eingabe ein PDF erzeugt

Interaktive Systeme erzeugen Ausgaben nicht nur erst dann, wenn sie terminieren,

sondern sie interagieren und synchronisieren stetig mit ihrer Umgebung. Wichtig hierbei ist, dass diese Interaktion durch das Rechnersystem bestimmt wird, nicht etwa durch die Prozessumgebung: Wann immer das System neue Eingaben zur Fortführung benötigt, wird die Umgebung, also ggf. auch der Benutzer hierzu aufgefordert. Das System synchronisiert sich auf diese *proaktive* Weise mit der Umgebung.

Merkmale:

- **Dialogorientiert**, oft mit GUI oder CLI
- Benutzer kann den Ablauf beeinflussen
- Reaktion auf Benutzeraktionen in Echtzeit
- Häufig in Benutzeroberflächen, Spielen, Anwendungen mit Feedback

Beispiele:

- Texteditor (z. B. VS Code)
- Webbrowser
- Smartphone-App
- Steuerungssysteme mit Touchscreen

Bei ***reaktiven Systemen*** schreibt die Umgebung vor, was zu tun ist. Das Computersystem reagiert, oft in Echtzeit und mit hoher Zuverlässigkeit, die Prozessumgebung synchronisiert den Rechner (und nicht umgekehrt).

Merkmale:

- **Ereignisgesteuert**
- **Schnelle und zuverlässige Reaktion** auf Änderungen
- Oft in sicherheitskritischen oder verteilten Systemen
- Kann sowohl interaktiv als auch autonom sein

Beispiele:

- Airbag-Steuerung im Auto
- Überwachungssysteme (z. B. Rauchmelder)
- Echtzeit-Datenstromverarbeitung (z. B. Sensoren am Raspberry Pi)

Worin liegen die Auswirkungen dieses kleinen Unterschieds, wer wen synchronisiert? Die wesentlichen Aufgaben eines interaktiven Systems sind die Vermeidung von Verklemmungen (deadlocks), die Herstellung von ”Fairness“ und die Erzeugung einer Konsistenz, insbesondere bei verteilten Systemen. Reaktive Systeme hingegen verlangen vom Computer, dass dieser reagiert, und zwar meistens rechtzeitig. Rechtzeitigkeit und Sicherheit sind die größten Belange dieser Systeme.

Zudem muss von interaktiven Systemen kein deterministisches Verhalten verlangt werden: Diese können intern die Entscheidung darüber treffen, wer wann bedient wird. Selbst die Reaktion auf eine Sequenz von Anfragen muss nicht immer gleich sein. Bei reaktiven Systemen ist hingegen der Verhaltensdeterminismus integraler Bestandteil. Daher hier die Definition von Determinismus bzw. eines deterministischen Systems

Hinweis – „deterministisches Verhalten“ :

- **Vorhersagbarkeit:** Das Verhalten ist vollständig vorhersehbar.

- **Reproduzierbarkeit:** Wiederholte Ausführung mit denselben Bedingungen ergibt dasselbe Ergebnis.
- **Keine Zufallseinflüsse:** Es gibt keine nicht-deterministischen Elemente wie Zufallszahlen, externe Zustände oder konkurrierende Prozesse, die das Ergebnis beeinflussen könnten.

Definition:
Ein System weist *determiniertes* oder *deterministisches Verhalten* (*Deterministic Behaviour*) auf, wenn zu jedem Satz von inneren Zuständen und jedem Satz von Eingangsgrößen genau ein Satz von Ausgangsgrößen gehört.

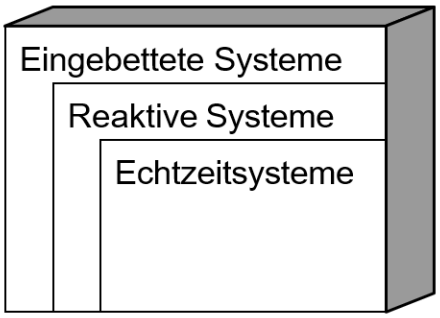
Als Gegenbegriffe können stochastisch oder nicht-deterministisch genannt werden. Diese Definition bezieht sich ausschließlich auf die logische (algorithmische) Arbeitsweise, und das klassische Beispiel sind die endlichen Automaten (DFA, Deterministic Finite Automation). Nicht-deterministische Maschinen werden auf dieser Ebene in der Praxis nicht gebaut, beim NFA (Non-Deterministic Finite Automaton) handelt es sich um eine theoretische Maschine aus dem Gebiet der Theoretischen Informatik.

Klassifizierung eingebetteter Systeme

Eingebettete Systeme, die mit einer Umgebung in Wechselwirkung stehen, sind nahezu immer als reaktives System ausgebildet. Interaktive Systeme sind zwar prinzipiell möglich, doch die Einbettung macht in der Regel eine Reaktivität notwendig. Die wichtigsten Eigenschaften im Sinn der Einbettung sind: Nebenläufigkeit (zumindest oftmals), hohe Zuverlässigkeit und Einhaltung von Zeitschranken.

Noch eine Anmerkung zum Determinismus: Während man davon ausgehen kann, dass alle technisch eingesetzten, eingebetteten Systeme deterministisch sind, muss dies für die Spezifikation nicht gelten: Hier sind nicht-deterministische Beschreibungen erlaubt, z.B., um Teile noch offen zu lassen.

Wird die Einhaltung von Zeitschranken zu einer Hauptsache, d.h. wird die Verletzung bestimmter Zeitschranken sehr kritisch im Sinn einer Gefährdung für Mensch und Maschine, dann spricht man von Echtzeitsystemen. Echtzeitfähige, eingebettete Systeme sind eine echte Untermenge der reaktiven Systeme, die ihrerseits eine echte Untermenge der eingebetteten Systeme darstellen.



Eingebettete Systeme lassen sich weiterhin nach einer Reihe von unterschiedlichen Kriterien klassifizieren. Hierzu zählen:

- *Kontinuierlich* versus *diskret*: Diese Ausprägung der Stetigkeit bezieht sich sowohl

auf Datenwerte als auch auf die Zeit. Enthält ein System beide Verhaltensweisen, wird es als ”hybrides System“ bezeichnet.

- *Monolithisch* versus *verteilt*: Während anfänglich alle Applikationen für eingebettete Systeme als monolithische Systeme aufgebaut wurden, verlagert sich dies zunehmend in Richtung verteilte Systeme. Hier sind besondere Anforderungen zu erfüllen, wenn es um Echtzeitfähigkeit geht.
- *Sicherheitskritisch* versus *nicht-sicherheitskritisch*: Sicherheitskritische Systeme sind solche, deren Versagen zu einer Gefährdung von Menschen und/oder Einrichtungen führen kann. Viele Konsumprodukte sind sicherheits-unkritisch, während Medizintechnik, Flugzeugbau sowie Automobile zunehmend auf sicherheitskritischen eingebetteten Systemen beruhen.

Definitionen

Unter einem *System* versteht man ein mathematisches Modell, das einem Eingangssignal ein Ausgangssignal zuordnet.
Wenn das Ausgangssignal hierbei nur vom aktuellen Wert des Eingangssignals ab- hängt, spricht man von einem *gedächtnislosen* System (Beispiel: Schaltnetze in der digitalen Elektronik). Hängt dagegen dieser von vorhergehenden Eingangssignalen ab, spricht man von einem *dynamischen* System (Beispiel: Schaltwerke).

Ein *reaktives System* (*reactive system*) kann aus Software und/oder Hardware bestehen und setzt Eingabeereignisse, deren zeitliches Verhalten meist nicht vor- hergesagt werden kann, in Ausgabeereignisse um. Die Umsetzung erfolgt oftmals, aber nicht notwendigerweise unter Einhaltung von Zeitvorgaben.

Ein *verteiltes System* (*distributed system*) besteht aus Komponenten, die räumlich oder logisch verteilt sind und mittels einer Kopplung bzw. Vernetzung zum Erreichen der Funktionalität des Gesamtsystems beitragen. Die Kopplung bzw. Vernetzung spielt bei echtzeitfähigen Systemen eine besondere Herausforderung

Ein *Steuergerät* (*electronic control unit*, ECU) ist die physikalische Umsetzung eines eingebetteten Systems. Es stellt damit die Kontrolleinheit eines mechatronischen Systems dar. In mechatronischen Systemen bilden Steuergerät und Sensorik/Aktorik oftmals eine Einheit.

Wird Elektronik zur Steuerung und Regelung mechanischer Vorgänge räumlich eng mit den mechanischen Systembestandteilen verbunden, so spricht man von einem *mechatronischen System*. Der Forschungszweig, der sich mit den Grundlagen und der Entwicklung mechatronische Systeme befasst, heißt *Mechatronik* (*mechatronics*).

Mechatronik ist ein Kunstwort, gebildet aus Mechanik und Elektronik. In der Praxis gehört allerdings eine erhebliche Informatik-Komponente hinzu, da nahezu alle mechatronischen Systeme auf Mikrocontrollern/Software basieren

Aufbau und Komponenten eingebetteter Systeme

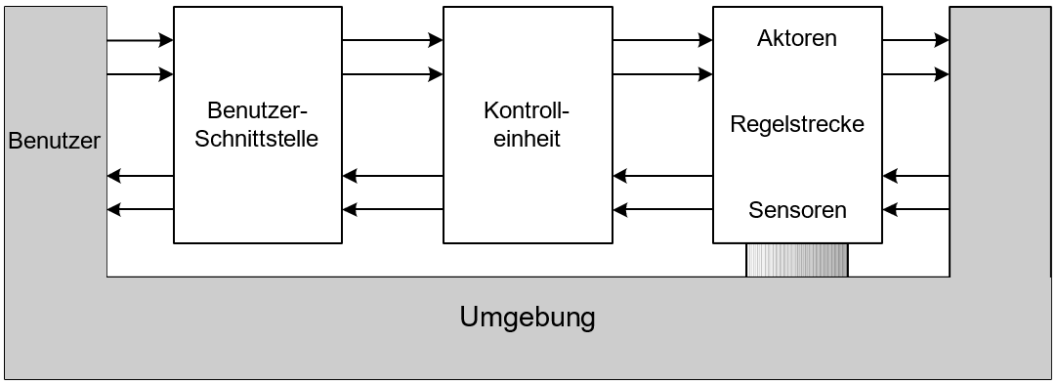
Während der logische Aufbau eingebetteter Systeme oftmals sehr ähnlich ist, hängt die tatsächliche Realisierung insbesondere der Hardware stark von den Gegebenheiten am Einsatzort ab. Hier können viele Störfaktoren herrschen, zudem muss das eingebettete System Sorge dafür tragen, nicht selbst zum Störfaktor zu werden.

Einige *Störfaktoren* sind: Wärme/Kälte, Staub, Feuchtigkeit, Spritzwasser, mechanische Belastung (Schwingungen, Stöße), Fremdkörper, elektromagnetische Störungen und Elementarteilchen (z.B. Höhenstrahlung). Allgemeine und Hersteller-spezifische Vorschriften enthalten teilweise genaue Angaben zur Vermeidung des passiven und aktiven Einflusses, insbesondere im EMV-Umfeld (Elektromagnetische Verträglichkeit).

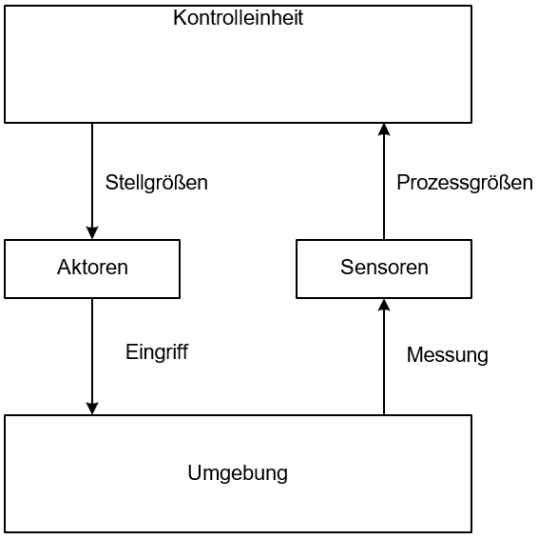
Der *logische Aufbau* der eingebetteten Systeme ist jedoch recht einheitlich, in der Regel können 5 strukturelle Bestandteile identifiziert werden:

- Die Kontrolleinheit bzw. das Steuergerät (siehe Definition Steuergerät/ECU), d.h. das eingebettete Hardware/Software-System,

- die Regelstrecke mit Aktoren (bzw. Aktuatoren) (actuator) und Sensoren (sensor), d.h. das gesteuerte/geregelte physikalische System,
- die Benutzerschnittstelle,
- die Umgebung sowie
- den Benutzer.



Das Bild stellt diese Referenzarchitektur eines eingebetteten Systems als Datenflussarchitektur dar, in der die Pfeile die gerichteten Kommunikationskanäle zeigen. Solche Kommunikationskanäle können (zeit- und wert-)kontinuierliche Signale oder Ströme diskreter Nachrichten übermitteln. Regelstrecke und Umgebung sind hierbei auf meist komplexe Weise miteinander gekoppelt, die schwer formalisierbar sein kann.



Wirkungskette System/Umgebung

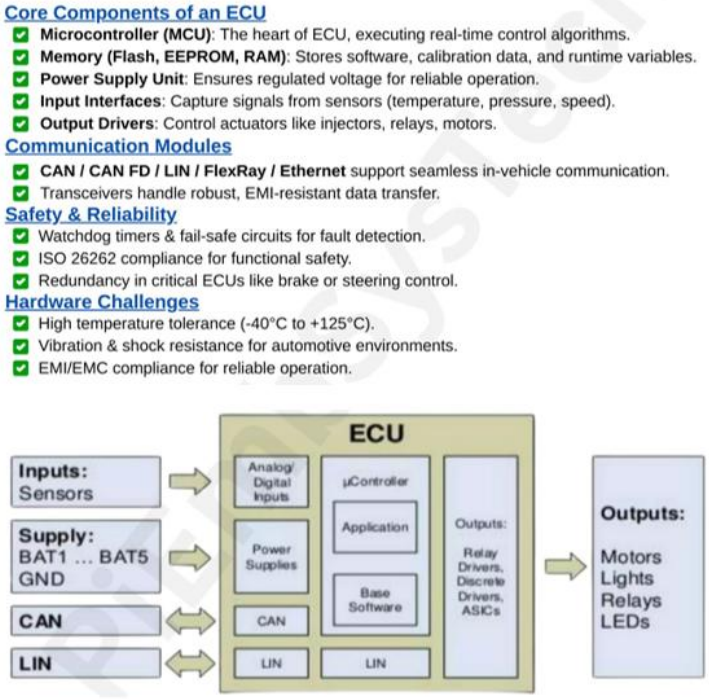
Dieses Bild zeigt die geschlossene Wirkungskette, die ein eingebettetes System einschließlich der Umgebung bildet. Der zu regelnde oder steuernde Prozess ist über Sensoren und Aktoren an das Steuergerät gekoppelt und kommuniziert mit diesem darüber. Sensoren und Aktoren fasst man unter dem (aus dem Von-Neumann-Modell bekannten) Begriff Peripherie (peripheral devices) oder I/O-System (input/output) zusammen.

Zu den einzelnen Einheiten seien einige Anmerkungen hier eingeführt:

Kontrolleinheit

Die Kontrolleinheit bildet den Kern des eingebetteten Systems, wobei sie selbst wieder aus verschiedenen Einheiten zusammengesetzt sein kann. Sie muss das Interface zum Benutzer (falls vorhanden) und zur Umgebung bilden, d.h., sie empfängt Nachrichten bzw. Signale von diesen und muss sie in eine Reaktion umsetzen.
Wie bereits in vorhergehenden Abschnitt dargestellt wurde, ist diese Kontrolleinheit fast ausschließlich als reaktives System ausgeführt. Die Implementierung liegt in modernen Systemen ebenso fast ausnahmslos in Form programmierbarer Systeme, also als Kombination Hardware und Software vor. Hierbei allerdings gibt es eine Vielzahl von Möglichkeiten: ASIC (Application-Specific Integrated Circuit), PLD/FPGA (Programmable Logic Devices/Field-Programmable Gate Arrays), General-Purpose Mikrocontroller, DSP (Digital Signal Processor), ASIP (Application Specific Instruction Set Processor), um nur die wichtigsten Implementierungsklassen zu nennen. Man spricht hierbei von einem Design Space bzw. von Design Space Exploration.

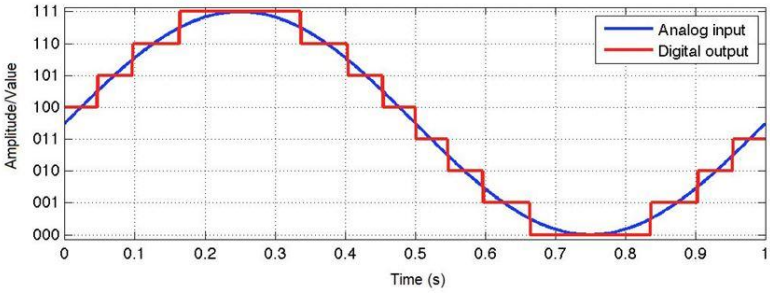
Beispiel Hardware Design einer Automotive ECU



Peripherie: Analog/Digital-Wandler

Ein Analog/Digital-Wandler (Analog/Digital-Converter, ADC), kurz A/D-Wandler, erzeugt aus einem (wert- und zeit-)analogen Signal digitale Signale. Die Umsetzung ist komplexer, nicht reversibler Prozess.

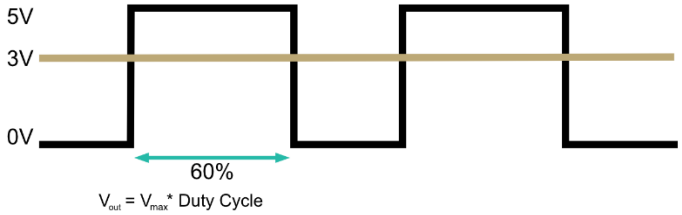
Der technisch eingeschlagene Weg besteht aus der Abtastung zuerst, gefolgt von einer Quantisierung und der Codierung. Die Abtastung ergibt die Zeitdiskretisierung, die Quantisierung die Wertediskretisierung. Man beachte, dass mit technischen Mitteln sowohl die Abtastfrequenz als auch die Auflösung zwar ”beliebig“ verbessert werden kann, aber niemals kontinuierliche Werte erreicht werden. In eingebetteten Systemen werden diese Werte den Erfordernissen der Applikation angepasst.



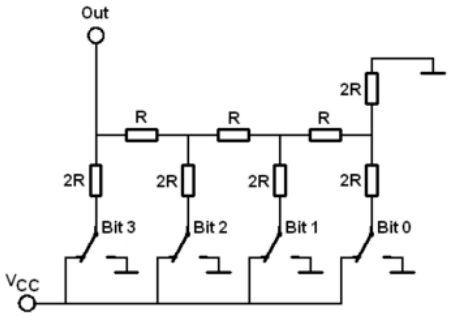
Peripherie: Digital/Analog-Wandler

Der Digital/Analog-Wandler, kurz D/A-Wandler (Digital/Analog-Converter, DAC) erzeugt aus digitalen Signalen ein analoges Signal (meist eine Spannung). Dies stellt die Umkehrung der A/D-Wandlung dar. Die Umsetzung erfolgt exakt, abgesehen von Schaltungsfehlern, d.h. ohne prinzipiellen Fehler wie bei der A/D- Wandlung.

Gängige Verfahren sind: Pulsweiten-Modulation (pulse width modulation, PWM)



und R/2R-Netzwerke.



Peripherie: Sensoren

Definition:

Ein *Sensor* ist eine Einrichtung zum Feststellen von physikalischen oder chemischen Eingangsgrößen, die optional eine Messwertzuordnung (Skalierung) der Größen treffen kann, sowie ggf. ein digitales bzw. digitalisierbares Ausgangssignal liefern kann.

Sensoren stellen also das primäre Element in einer Messkette dar und setzen variable, im Allgemeinen nichtelektrische Eingangsgrößen in ein geeignetes, insbesondere elektrisches Messsignal um. Hierbei können ferner *rezeptive Sensoren*, die nur passiv Signale umsetzen (Beispiel: Mikrofon), sowie *signalbearbeitende Sensoren*, die die Umwelt stimulieren und die Antwort aufnehmen (Beispiel: Ultraschall- Sensoren zur Entfernungsmessung), unterschieden werden.

Als *Smart Sensors* bezeichnete Sensoren beinhalten bereits eine Vorverarbeitung der Daten.

Peripherie: Aktuatoren

Aktuatoren bzw. Aktoren verbinden den informationsverarbeitenden Teil eines eingebetteten Systems und den Prozess. Sie wandeln Energie z.B. in mechanischen Arbeit um.

Die Ansteuerung der Aktuatoren kann analog (Beispiel: Elektromotor) oder auch digital (Beispiel: Schrittmotor) erfolgen.

Die Rolle der Zeit und weitere Randbedingungen

(Drei) Verschiedene Ausprägungen der Zeit

In den vorangegangenen Abschnitten wurde bereits verdeutlicht: Die Zeit spielt bei den binärwertigen digitalen *und* den analogen Systemen (Umgebungsprozess) eine Rolle, die genauer betrachtet werden muss.

Zeit-analoge Systeme / analoge Zeitbindung

- Die Zeit läuft kontinuierlich, wie eine Uhr ohne Unterbrechung.
- Beispiel: Temperaturverlauf, Musiksignal, …

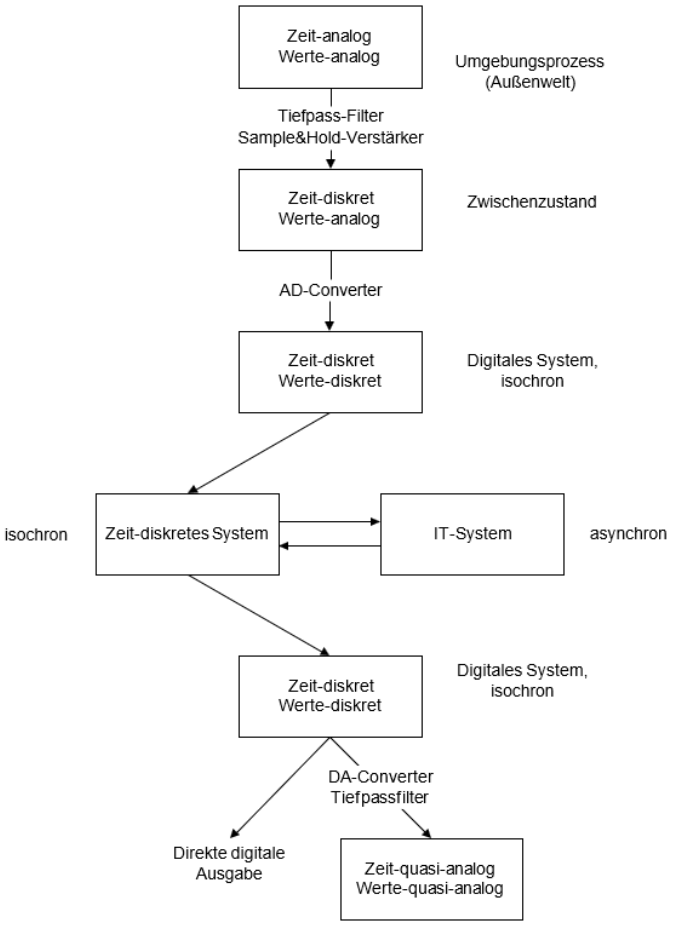
Zeit-diskrete Systeme / diskrete Zeitbindung (digital)

- Die Zeit ist in Einzelschritte unterteilt, z. B. jede Millisekunde.
- Beispiel: Ein Mikrocontroller liest alle 10ms einen Sensorwert.

Zeitunabhängige Systeme (IT-Systeme)

- Diese Systeme arbeiten ohne direkten Bezug zur Zeit, z. B. Datenbanken oder Webseiten.
- Sie reagieren, wenn etwas passiert, aber nicht regelmäßig

Übergänge zwischen den Zeitbindungen



Weitere Randbedingungen für eingebettete Systeme

Die Zeit spielt in Embedded Systems aus dem Grund eine übergeordnete Rolle, weil der Rechner in eine Maschine eingebettet ist, deren Zeitbedingungen vorherbestimmt sind. Insofern hat die Zeit eine übergeordnete Bedeutung.

Aber: Es existieren noch weitere Randbedingungen, insbesondere für den Entwurfsprozess:

- Power Dissipation/Verlustleistung: Welche Durchschnitts- und/oder Spitzenleistung ist vertretbar, gefordert, nicht zu unterschreiten usw.?

- Ressourcenminimierung: Nicht nur die Verlustleistung, auch die Siliziumfläche, die sich in Kosten niederschlägt, soll minimiert werden.

Als vorläufiges Fazit kann nun gelten, dass die Entwicklung für eingebettete Systeme bedeutet, eine Entwicklung mit scharfen und unscharfen Randbedingungen durchzuführen.

Design Space Exploration

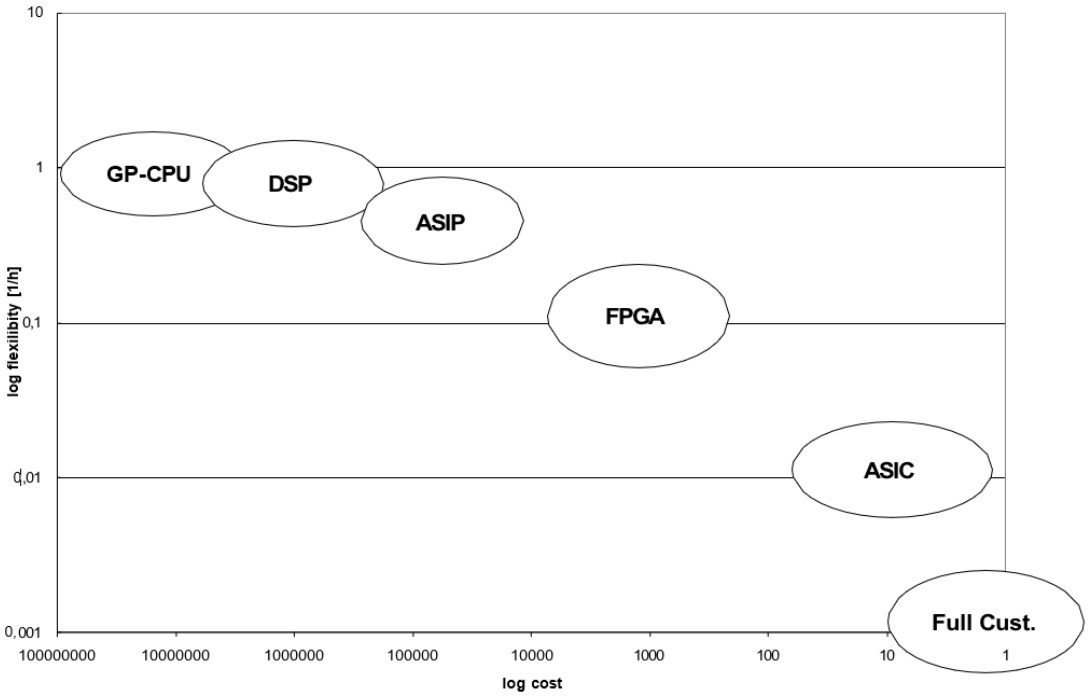
Design Space Exploration (DSE) bedeutet auf Deutsch etwa „Erkundung des Entwurfsraums“ und ist ein wichtiger Begriff in der Systementwicklung, besonders bei eingebetteten Systemen, Hardware-Design oder Software-Architektur.
Wenn man ein technisches System entwirft (z. B. einen Mikrocontroller, ein Steuerungssystem oder eine Software), gibt es viele Möglichkeiten, wie man es umsetzen kann. Diese Möglichkeiten nennt man den „Design Space“ – also den Raum aller möglichen Entwurfsvarianten.

Ziel von DSE

Design Space Exploration bedeutet:

- Man testet und vergleicht verschiedene Varianten eines Designs.
- Man sucht nach der besten Lösung – z. B. die schnellste, sparsamste oder günstigste.

Bei all den bisher dargestellten Fakten sollte eines deutlich geworden sein: Ein eingebettetes System besteht im Kern aus einem programmierbaren Rechner (zumindest ist das die Regel), und die Wahl des Rechners sowie dessen Programmierung müssen sich Randbedingungen unterwerfen, z.B. der Erfüllung von zeitlichen Bedingungen.



Das Bild zeigt das Ergebnis einer Studie, die als Implementierungsbasis

- General-Purpose Prozessoren (z.B. AVR Mikrocontroller)
- Digitale Signalprozessoren (DSP)
- Applikations-spezifische Instruktionssatz-Prozessoren (ASIP)
- Field-Programmable Gate Arrays (FPGA, als spezifische Klasse der PLDs)
- Applikations-spezifische Integrierte Schaltungen (ASIC)
- Fully-Customized Integrated Circuits (voll- kundenspezifische Schaltungen)

nimmt.

Die Kosten, die hier als Produkt von Siliziumfläche, Verlustleistung und Ausführungszeit genommen werden, variieren über einen Bereich von 8 Zehnerpotenzen, bei den Werten zur Flexibilität, gemessen am Zeitverbrauch für eine Änderung, immerhin noch um 3 Zehnerpotenzen.

Dies sagt aus, dass die Implementierung die daraus resultierenden Nebenwirkungen in drastischem Maß beeinflusst. Bei programmierbaren Architekturen ergeben sich immerhin noch 4 Zehnerpotenzen. Leider muss man deutlich sagen, dass sich die Entwicklung für FPGAs deutlich von der für Mikrocontroller unterscheidet (typische Programmiersprache:

VHDL versus C, Programmiermodell: parallel versus sequentiell).

Echtzeitsysteme

Definitionen

Definition nach DIN 44300 des Deutschen Instituts für Normung

Unter *Echtzeit* (*real time*) versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.

Definition nach „Oxford Dictionary of Computing“

Ein Echtzeitsystem (real-time system) ist ein System, bei dem der Zeitpunkt, zu dem Ausgaben vorliegen, bedeutend ist. Das liegt für gewöhnlich daran, dass die Eingabe mit einigen Änderungen der physikalischen Welt korrespondiert und die Ausgabe sich auf diese Änderungen beziehen muss. Die Verzögerung zwischen der Zeit der Eingabe und der Zeit der Ausgabe muss ausreichend klein für eine akzeptable ”Rechtzeitigkeit“ (timeliness) sein.

Echtzeitsysteme sind also Systeme, die korrekte Reaktionen innerhalb einer definierten Zeitspanne produzieren müssen. Falls die Reaktionen das Zeitlimit über- schreiten, führt dies zu Leistungseinbußen, Fehlfunktionen und/oder sogar Gefährdungen für Menschen und Material.
Die Unterscheidung in harte und weiche Echtzeitsysteme wird ausschließlich über die Art der Folgen einer Verletzung der Zeitschranken getroffen:

Ein Echtzeitsystem wird als *hartes Echtzeitsystem* (*hard real-time system*) bezeichnet, wenn das Überschreiten der Zeitlimits bei der Reaktion erhebliche Folgen haben kann. Zu diesen Folgen zählen die Gefährdung von Menschen, die Beschädigung von Maschinen, also Auswirkungen auf Gesundheit und Unversehrtheit der Umgebung.

Typische Beispiele hierfür sind einige Steuerungssysteme im Flugzeug oder im Auto, z.B. beim Antrieb.

Eine Verletzung der Ausführungszeiten in einem *weichen Echtzeitsystem* (*soft real- time system*) führt ausschließlich zu einer Verminderung der Qualität, nicht jedoch zu einer Beschädigung oder Gefährdung.

Beispiele hierfür sind Multimediasysteme, bei denen das gelegentlich Abweichen von einer Abspielrate von 25 Bildern/Sek. zu einem Ruckeln o.ä. führt.

Als Anmerkung sei hier beigefügt, dass fast immer nur die oberen Zeitschranken aufgeführt werden. Dies hat seine Ursache darin, dass die Einhaltung einer oberen Zeitschranke im Zweifelsfall einen erheblichen Konstruktionsaufwand erfordert, während eine untere Schranke, d.h. eine Mindestzeit, vor der nicht reagiert werden darf, konstruktiv unbedenklich ist. Ein Beispiel für ein System, bei dem beide Werte wichtig sind, ist die Steuerung des Zündzeitpunkts beim Verbrennungsmotor: Dieser darf nur in einem eng begrenzten Zündintervall kommen.

Ereignissteuerung vs. Zeitsteuerung