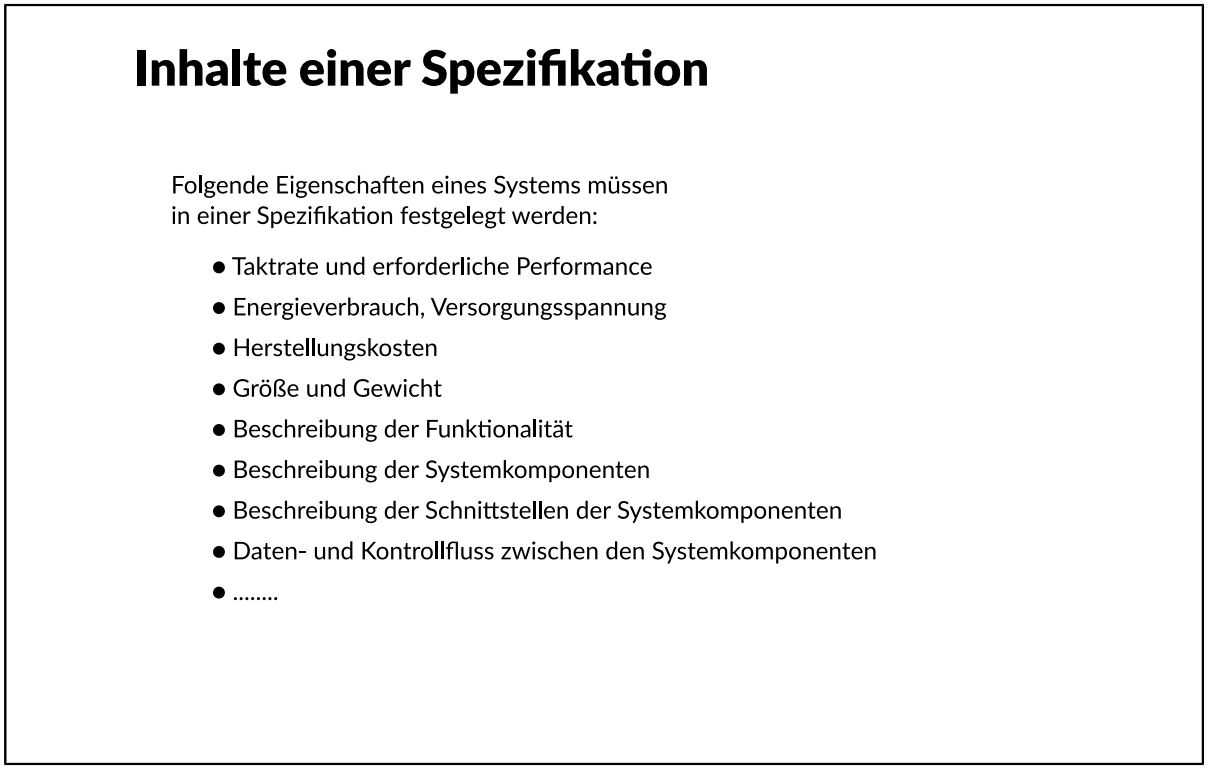
Electronic Design Automation (EDA)

Spezifikation

Spezifikation: Inhalte einer Spezifikation



Ausgangspunkt jedes Entwurfsprozesses ist die Spezifikation des zu entwerfenden Systems. Diese soll dazu dienen, alle zu Beginn des Entwurfsprozesses gewünschten Eigenschaften und Randbedingungen formalisiert zu dokumentieren. Insbesondere bei der Arbeit in größeren Teams ist eine solche formale Spezifikation wichtig, damit alle an einem Projekt Arbeitenden einen Überblick über das Gesamtsystem haben. Gleichzeitig dient die (formale) Spezifikation als Referenz bei der Verfeinerung des Systems im Laufe des Entwurfsprozesses. Idealerweise sollte eine Spezifikation vollständig und widerspruchsfrei sein.

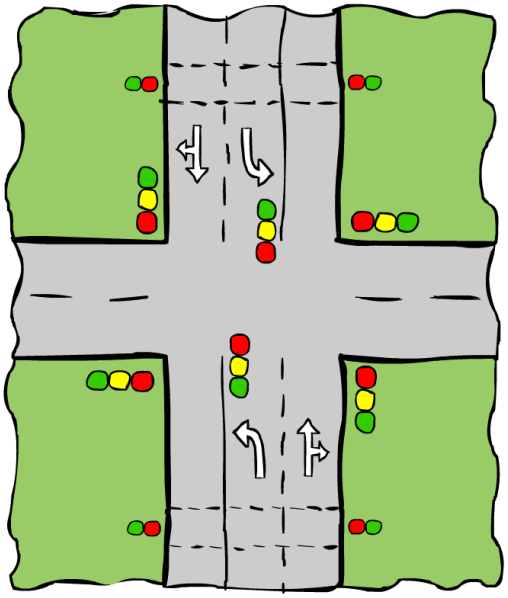
Eine Spezifikation wird auf einer hohen Abstraktionsebene erstellt. Sie kann eine Vielzahl von unterschiedlichen Informationen enthalten. Je nach System – elektrisch, elektro-mechanisch, digital, analog, mixed-signal – wird die Spezifikation nicht alle der oben genannten Daten bzw. zusätzliche Informationen beinhalten.

Spezifikation: Beispielspezifikation Ampelsteuerung

Beispielspezifikation Ampelsteuerung

Steuerung einer Hauptstraßen/Nebenstraßen-Ampel mit Fußgänger-Ampel für die Hauptstraße

- Akustischer Signalgeber für die Fußgängerampel
- Linksabbiegersensor
- Versorgungsspannung 220V
- Treiberleistung je Kanal 1.000 W
- Gesamtleistungsaufnahme < 10.000 W
- Wetterfestes Gehäuse
- Größe maximal 10x10x5 cm
- Taktrate 10 MHz

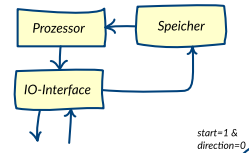


Als einfaches Beispiel sei hier eine Ampelsteuerung spezifiziert, also ein System, das im Wesentlichen lediglich ein Steuerwerk darstellt, das einen Kontrollfluss realisiert. Es beschreibt die Steuerung einer Hauptstraßen/Nebenstraßen-Ampel mit zusätzlicher Fußgänger-Ampel für die Hauptstraße und soll die in der Abbildung dargestellten Eigenschaften aufweisen.

Neben diesen globalen Eigenschaften werden der detaillierte Aufbau der Ampelsteuerung und die Funktionalität mit verschiedenen formalen Methoden beschrieben.

Spezifikation: Formale Beschreibung

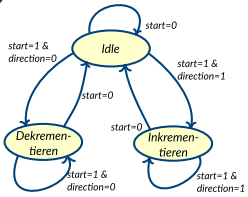
Formale Spezifikationsbeschreibung



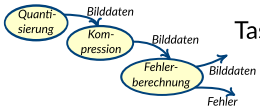
Blockdiagramme

Beschreiben den strukturellen Aufbau eines Systems

Zustandsüber-gangsdiagramme



Dienen zur Beschreibung der Funktionalität, insbesondere des Kontrollflusses.



Task-Flow-Graphen

Beschreiben den Datenfluss des Systems.

System-Level-Design-Language

```
#include "systemc.h"
SC_MODULE(nand2)
// declare nand2
sc_module
{
  sc_in<bool>
...
```

Sind spezielle Sprachen zur Systembeschreibung.

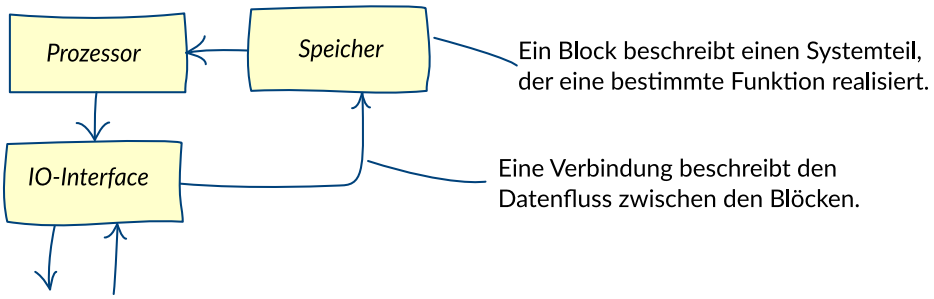
In den meisten Fällen wird die Spezifikation in natürlicher Sprache abgefasst, da bislang keine einheitliche Sprache zur Beschreibung von Spezifikationen existiert. Neben der natürlichen Sprache werden für Teile der Spezifikation auch Blockdiagramme, Zustandsübergangsdiagramme oder Task-Flow-Graphen verwendet. Während Blockdiagramme der Beschreibung der Struktur eines Systems dienen, beschreiben Zustandsübergangsdiagramme das Verhalten des Systems. Mit Task-Flow-Graphen wird der Datenfluss eines Systems spezifiziert.

Grundsätzlich kann die Spezifikation auch mit Hilfe einer formalen Sprache erfolgen. Dazu werden häufig Erweiterungen regulärer Programmierspachen (z. B. C) oder aber auch spezielle Systembeschreibungssprachen benutzt.

Spezifikation: Blockdiagramme

Blockdiagramme

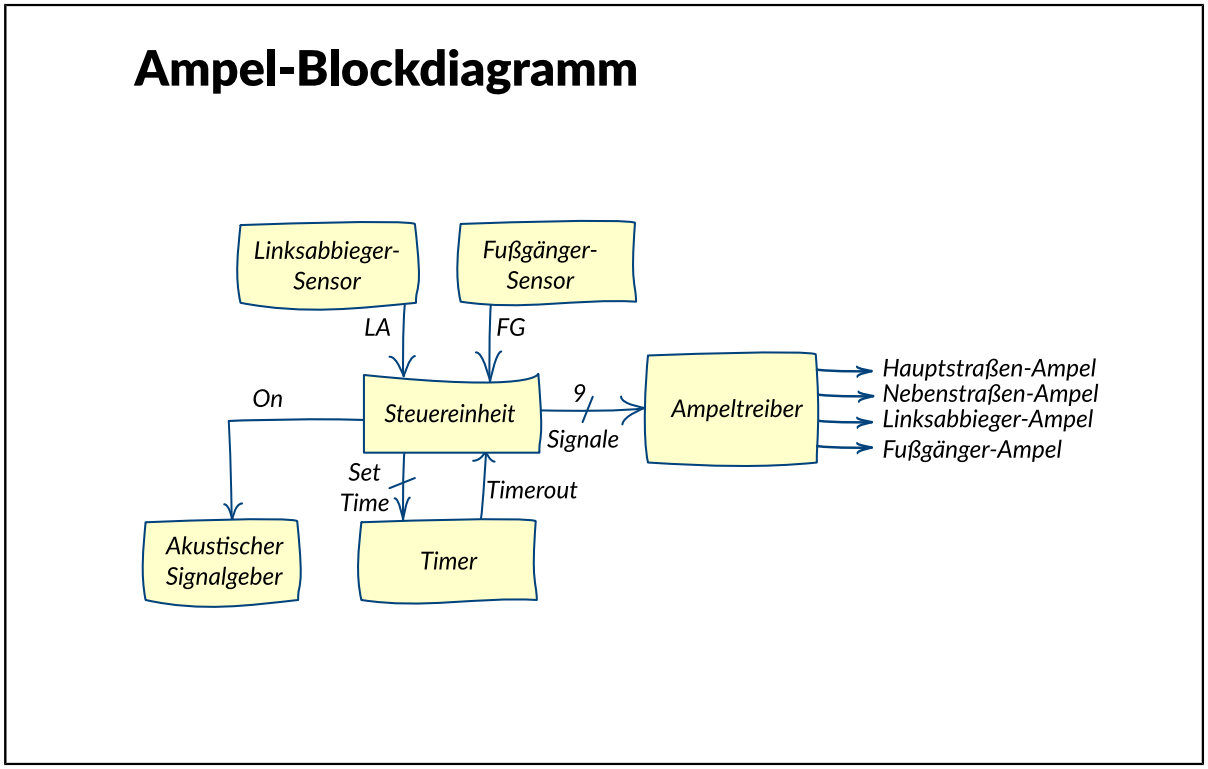
- Beschreiben die grundlegende Systemstruktur.
- Bestehen aus Blöcken und Verbindungen.



- Signalbreiten und Signalnamen können spezifiziert werden.
- Keine Beschreibung von Pegeln oder Protokollen.

Blockdiagramme werden in vielen Spezifikationen zur Beschreibung der grundlegenden Struktur eines Systems verwendet. Blockdiagramme bestehen aus Blöcken und Verbindungen zwischen den Blöcken, wobei jeder Block einer auszuführenden Aufgabe des Gesamtsystems entspricht. Die Verbindungen zwischen den Blöcken entsprechen dem Datenfluss zwischen den Blöcken und können bezüglich der Signalbreiten in einem Blockdiagramm spezifiziert werden. Ein Blockdiagramm basiert auf der Annahme, dass die Aufgaben des Gesamtsystems auf die Blöcke verteilt worden sind. Eine weitergehende Festlegung des Datenflusses zwischen den Modulen wie beispielsweise das Protokoll oder das Datenformat ist in einem Blockdiagramm nicht vorgesehen.

Spezifikation: ... für die Ampel



Im Bild ist das Blockschaltbild der Ampelsteuerung dargestellt.

Die Ampelsteuerung besteht aus zwei Sensoren, die registrieren, ob Linksabbieger an der Ampel stehen bzw. ob eine Querungsanforderung durch Fußgänger besteht.

Die Steuereinheit berechnet aus den Sensorsignalen sowie einem Timer den nächsten Ampelzustand.

Der Timer dient der Steuerung der Intervalllängen, in denen eine Ampel auf grün geschaltet ist. Dabei kann das Zeitintervall, nach dem das Timermodul den Timeout auslöst, von außen gesetzt werden. Dazu wird ein 4-bit breites Signal benutzt. Das Signal Timeout signalisiert den Ablauf des Zeitintervalls.

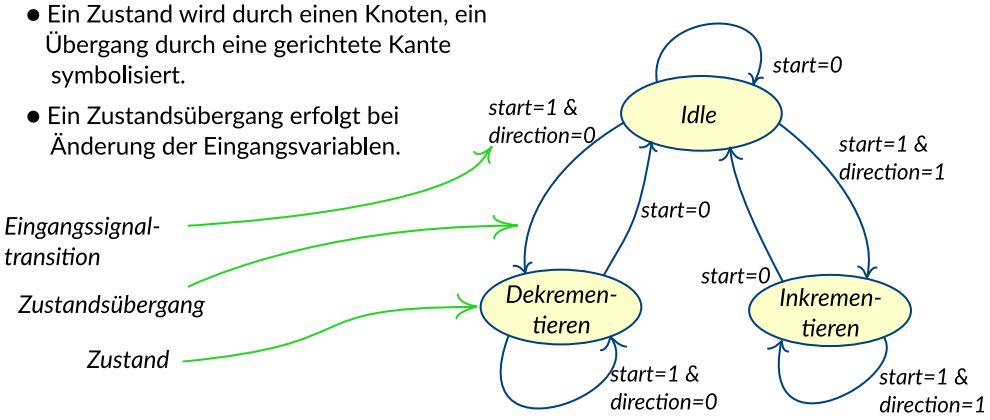
Die Ampeltreiber enthalten die Treiber für die Lampen der Ampel.

Der akustische Signalgeber signalisiert durch ein Tonsignal die Grünphase der Fußgängerampel.

Spezifikation: Zustandsübergangs-diagramme

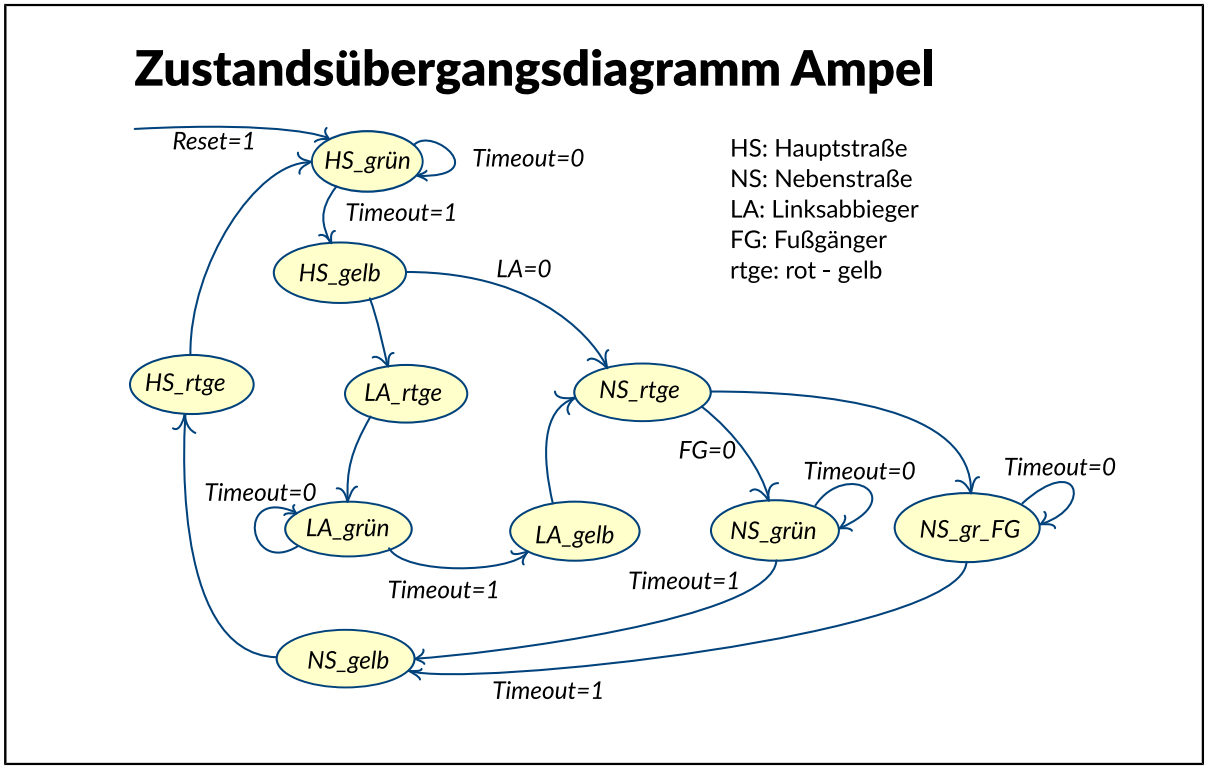
Zustandsübergangsdiagramme

- Beschreiben die Funktion eines Systems in Form eines gerichteten Graphen.
- Bestehen aus Zuständen und Übergängen.
- Ein Zustand wird durch einen Knoten, ein Übergang durch eine gerichtete Kante symbolisiert.
- Ein Zustandsübergang erfolgt bei Änderung der Eingangsvariablen.



Zustandsübergangsdiagramme beschreiben Zustände und Übergänge zwischen den Zuständen. Ein Systemzustand entspricht im Regelfall einem Satz von inneren Zuständen und/oder Ausgangswerten des Systems. Ein Zustandsübergang erfolgt jeweils bei der Änderung der Eingangsvariablen.
Die Zustandsübergänge werden in Zustandsübergangsdiagrammen durch Kanten dargestellt. Die Zustände werden durch Knoten beschrieben. An den Zustandsübergängen werden jeweils die Eingangssignaltransitionen notiert, unter denen der entsprechende Zustandswechsel erfolgt. Zustandsübergangsdiagramme eignen sich besonders gut, um endliche Automaten zu beschreiben. Die Beschreibung von Systemeigenschaften oder Strukturinformationen ist in Zustandsübergangsdiagrammen nicht möglich. Je nach Art des beschriebenen endlichen Automaten erfolgt die Darstellung der Ausgangssignale.

Spezifikation: ... für die Ampel

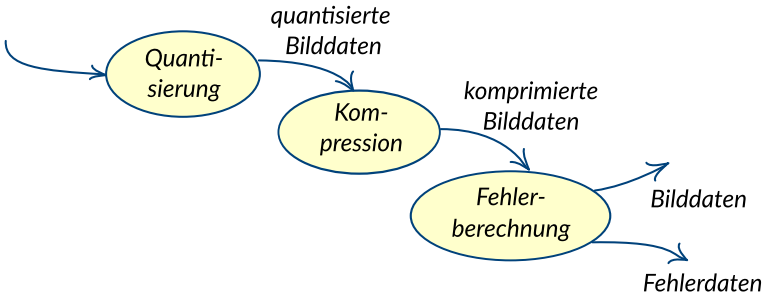


Die Abbildung zeigt das Zustandsübergangsdiagramm für das Modul "Ampelsteuerung" aus dem Blockschaltbild der Ampel. Nach einem Reset startet der endliche Automat in dem Zustand, in dem die Hauptstraße grün hat. Nach Ablauf der Grünzeit, was durch das Signal Timeout signalisiert wird, wechselt die Ampel der Hauptstraße auf gelb. In Abhängigkeit von den Sensoren der Linksabbieger (LA) bzw. Fußgänger (FG) erfolgt der nächste Zustandsübergang. Wurde der Linksabbieger-Sensor ausgelöst, erfolgt jetzt die Linksabbiegerphase. Anschließend folgt die Grünphase der Nebenstraße, wobei die Länge dieser Phase vom Auslösen des Fußgängersensors abhängt. Danach erfolgt wieder der Wechsel in den Initialzustand.

Spezifikation: Task-Flow-Graphen

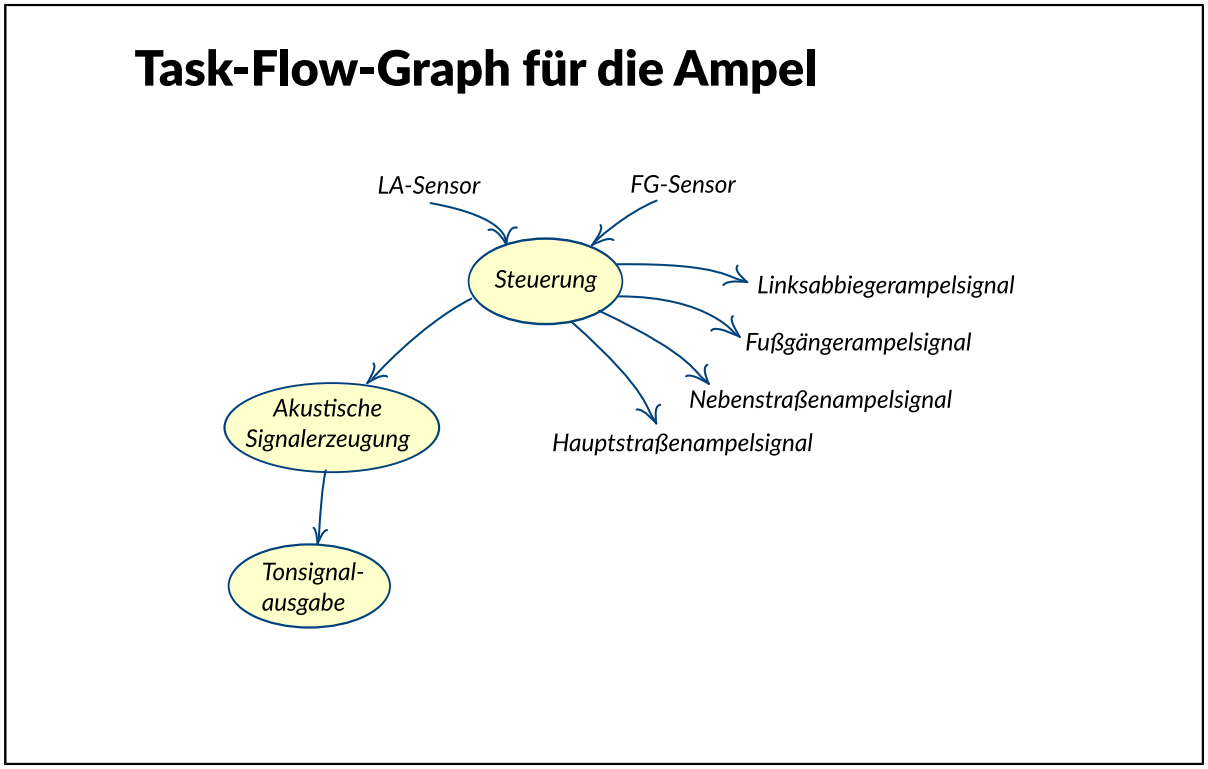
Task-Flow-Graphen

- Bestehen aus Tasks- und Taskgruppen.
- Ggf. erfolgt eine Annotierung des Kontroll- und Datenflusses.
- Ein hierarchischer Aufbau ist möglich.



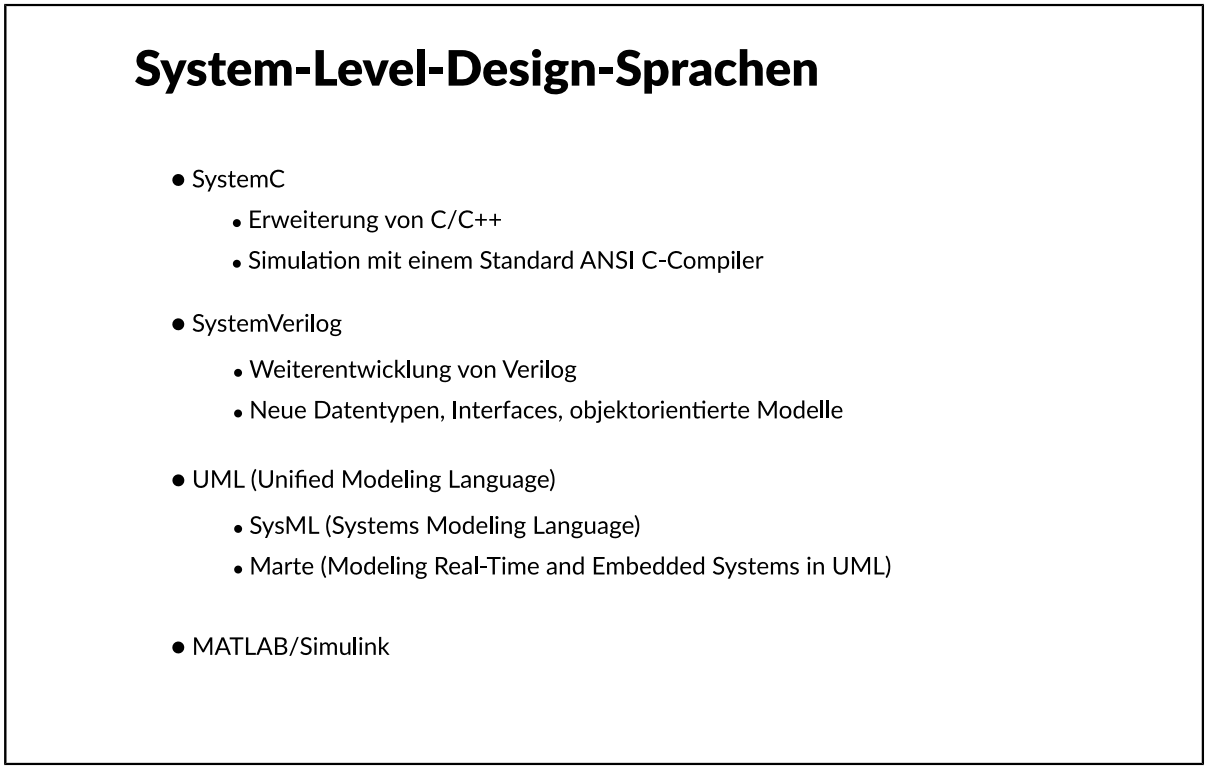
Task-Flow-Graphen werden häufig bei der System-Spezifikation genutzt. Sie stellen den Kontroll- und Datenfluss zwischen den einzelnen Aufgaben (Tasks) eines Systems dar. Jeder Knoten des Graphen stellt dabei eine Aufgabe des Systems dar, die aus vielen einzelnen Schritten bestehen kann. Jede Kante des Graphen repräsentiert einen Datenfluss. Die einzelnen Aufgaben des Systems können zu Aufgabengruppen zusammengefasst werden. Ferner können Task-Flow-Graphen hierarchisch aufgebaut sein, so dass ein Knoten des Graphen in einem weiteren Task-Flow-Graphen detailliert beschrieben werden kann.

Spezifikation: ... für die Ampel



Die Abbildung zeigt den Taskflow-Graphen für die Ampelsteuerung. Der Datenfluss besteht hier in der Signalisierung, dass etwas bestimmtes ausgeführt werden muss.

Spezifikation: System Level Design Language

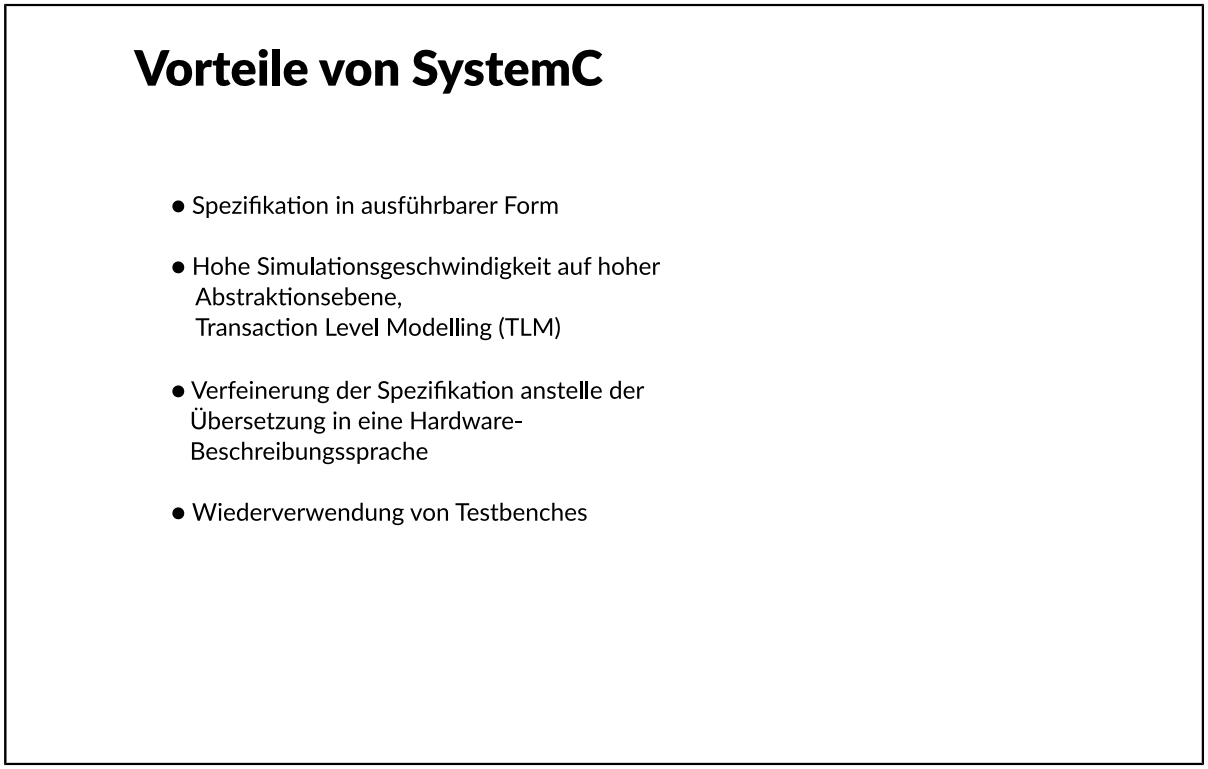


System-Level-Entwurfssprachen sollen eine einheitliche Darstellung für die Beschreibung eines vollständigen Systems – bestehend aus Hard- und Software - liefern. Bisher sind diese Sprachen wenig verbreitet. Zurzeit wird international versucht, einen Standard für eine solche Sprache zu schaffen. Als Quasi-Standard hat sich mittlerweile SystemC etabliert.

SystemC ist eine OpenSource-Initiative, die aus einem Zusammenschluss von Halbleiter-, IP- und EDA-Firmen besteht. SystemC ist frei verfügbar. Zur Simulation von SystemC-Code wird lediglich ein ANSI C++ Compiler benötigt.

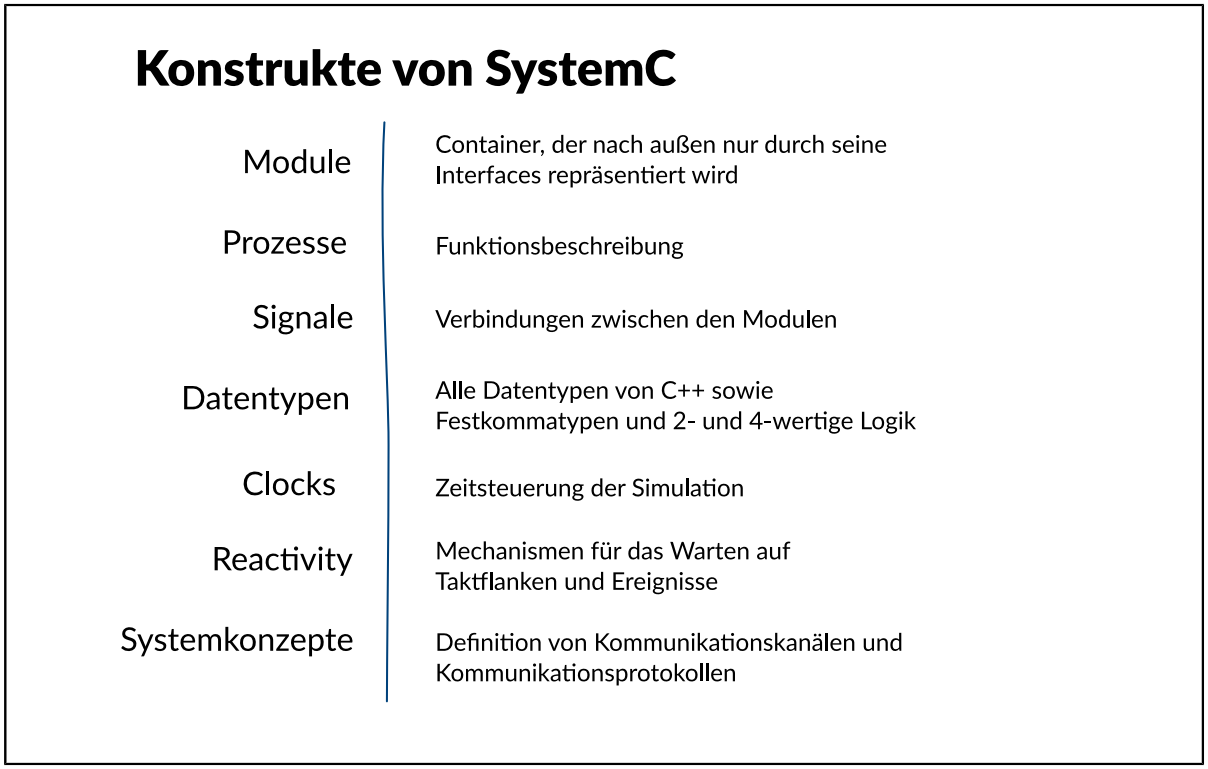
Neben der Entwicklung einer speziellen System-Level-Entwurfssprache wurde auch die Hardware-Beschreibungssprache Verilog um Konstrukte auf höheren Ebenen ergänzt.

Spezifikation: ... Vorteile



Der Einsatz von SystemC bietet verschiedene Vorteile gegenüber dem konventionellen HDL-Designflow. Wichtigster Vorteil ist die Ausführbarkeit der Spezifikation. Die auf hoher Abstraktionsebene beschriebenen Eigenschaften des Systems können bereits simuliert werden. Aufgrund des geringen Detailliertheitsgrads der Spezifikation erfolgt die Simulation auf dieser Ebene sehr schnell. Anschließend ist keine manuelle Umsetzung der Spezifikation in eine Hardware-Beschreibungssprache erforderlich, der C-Code wird lediglich verfeinert.

Spezifikation: ... Konstrukte



SystemC stellt verschiedene Konstrukte zur Systembeschreibung zur Verfügung. Die wichtigsten sind:

- Module: Diese definieren Container, die nach außen nur durch ihre Interfaces repräsentiert werden. Dadurch wird die Implementierung versteckt, was eine Möglichkeit zur Einbindung von bereits vorhandenen, sogenannten Intellectual-Property (IP)-Blöcken schafft.
- Prozesse: In Prozessen wird die eigentliche Funktionalität beschrieben. Dabei unterscheidet man verschiedene Typen von Prozessen. Method-Prozesse verhalten sich wie Funktionen. Thread-Prozesse werden dagegen nur einmal gestartet und werden immer wieder durchlaufen. CThread-Prozesse sind synchrone Thread-Prozesse.
- Signale: Diese definieren Verbindungen zwischen den Modulen und werden wie in anderen Hardware-Beschreibungssprachen verwendet, d.h. die Zuweisung von Signalwerten kann verzögert erfolgen.
- Datentypen: In SystemC stehen alle Datentypen von C++ sowie Festkommatypen und 2- und 4-wertige Logik zur Verfügung.
- Clocks dienen der Zeitsteuerung der Simulation. Dabei sind mehrere Clocks mit beliebigen Phasenlagen zueinander möglich.
- Reactivity: Diese definieren Mechanismen für das Warten auf Taktflanken und Ereignisse sowie das Überwachen von bestimmten Signalen.
- Systemkonzepte: Diese ermöglichen die Beschreibung von Kommunikationskanälen und abstrakten Kommunikationsprotokollen (Handshakes).