Technische Grundlagen der Informatik – Kapitel 4

Prof. Dr. Jan Peters
Fachgebiet Intelligente Autonome Systeme (IAS)
Fachbereich Informatik

TECHNISCHE UNIVERSITÄT DARMSTADT

WS 14/15

Meisenantworten … vielen Dank für die Hinweise!

TECHNISCHE UNIVERSITÄT DARMSTADT

- Folien zeitiger hochladen. *Good point, will try to*!
- Korrekte Studentenantworten über das Mikro wiederholen. *OK, remind me*!
- "Tut mir leid, dass ich geredet habe." *Danke für Deine Entschuldigung und angenommen* ;)
- Liebesbriefe. *Chris, Jan & Rudi sind vergeben. Weitere TGDI Betreuer nehmen aussagekräftige Bewerbungen nur nach erfolgreichem Abschluss von TGDI entgegen*.
- Philosophische Fragen*. Bitte an Profs. Gehring, Hubig & Nordmann im philosophischen Institut*.
- Pro Windows Propanda. *Bitte schnellstmöglich selbst-exmatrikulieren*!!
- Übungsblätter druckerfreundlicher? *Sorry, TU Da Corporate Identity*
- Wöchentliche Sniggersstatistiken: Gerechte Veteilung, minimale Anzahl von Verletzungen. *Super*!
- **Fortsetzung der Meisenantworten in der Vorlesung morgen!**

Kapitel 4: Themen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Einleitung
- Kombinatorische Logik
- Strukturelle Beschreibung
- Sequentielle Logik
- Mehr kombinatorische Logik
- Endliche Zustandsautomaten
- Parametrisierte Modelle
- Testumgebungen

Einleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Hardware-Beschreibungssprachen
  - *Hardware Description Languages (HDL)*

- Erlauben textuelle Beschreibung von Schaltungen
  - Auf verschiedenen Abstraktionsebenen
    - Struktur (z.B. Verbindungen zwischen Gattern)
    - Verhalten (z.B. Boole‘sche Gleichungen)

- Entwurfswerkzeuge erzeugen Schaltungsstruktur daraus automatisch
  - Computerprogramme
  - Computer-Aided Design (CAD) oder Electronic Design Automation (EDA)
  - Schaltungssynthese
    - Grob vergleichbar mit Übersetzung (Compilieren) von konventionellen Programmiersprachen

Einleitung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Fast alle kommerziellen Hardware-Entwürfe mit HDLs realisiert
- Zwei HDLs haben sich durchgesetzt
- Ihr werdet beide lernen müssen!
  - Es gibt keinen klaren Gewinner

Verilog

TECHNISCHE UNIVERSITÄT DARMSTADT

- 1984 von der Fa. Gateway Design Automation entwickelt
- Seit 1995 ein IEEE Standard (1364)
  - Überarbeitet 2001 und 2005
  - Neuer Dialekt SystemVerilog (Obermenge von Verilog-2005)
- Weit verbreitet in zivilen US-Firmen
- In Darmstadt im Fachbereich Informatik
  - Eingebettete Systeme und ihre Anwendungen (ESA, Prof. Koch)
- In Darmstadt im Fachbereich Elektrotechnik
  - Rechnersysteme (RS, Prof. Hochberger)

VHDL

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Very High-Speed Integrated Circuit Hardware Description Language*
- Entwickelt 1981 durch das US Verteidigungsministerium
  - Inspiriert durch konventionelle Programmiersprache Ada
- Standardisiert in 1987 durch IEEE (1076)
  - Überarbeitet in 1993, 2000, 2002, 2006, 2008

- Weit verbreitet in
  - US-Rüstungsfirmen
  - Vielen europäischen Firmen

- In Darmstadt im Fachbereich Elektrotechnik
  - Integrierte elektronische Systeme (IES, Prof. Hofmann)

In dieser Iteration der Vorlesung

TECHNISCHE UNIVERSITÄT DARMSTADT

- In den Vorlesungen Verilog
  - Häufig kompakter zu schreiben
  - Eher auf Einzelfolien darstellbar
- In den Übungen auch VHDL
- Hier gezeigte Grundkonzepte sind in beiden Sprachen identisch
- Nur andere Syntax
  - VHDL-Beschreibung ist aber in der Regel länger
- Im Buch werden beide Sprachen nebeneinander gezeigt
  - Kapitel 4
  - Moderne Entwurfswerkzeuge können in der Regel beide Sprachen

SystemVerilog

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verwendet in 2. Auflage des Lehrbuchs von Harris & Harris

- SystemVerilog ist im Rahmen der Veranstaltung sehr ähnlich zu Verilog
  - Teilweise einfacher
    - Verwendet nur Datentyp **`logic`** statt separaten **`wire`** und **`reg`** Typen
  - Teilweise aufwendiger
    - Getrennte Anweisungen für
      - Flip-Flops: **`always_ff`**
      - Latches: **`always_latch`**
      - Kombinatorische Logik: **`always_comb`**
    - Wird in Verilog alles mit **`always`** beschrieben

  - Übersicht der Unterschiede im Buch in Kapitel 4.7.1
  - Für Übungen und Klausuren relevant: Verilog (gezeigt auf Folien!)

Von einer HDL zu Logikgattern

TECHNISCHE UNIVERSITÄT DARMSTADT

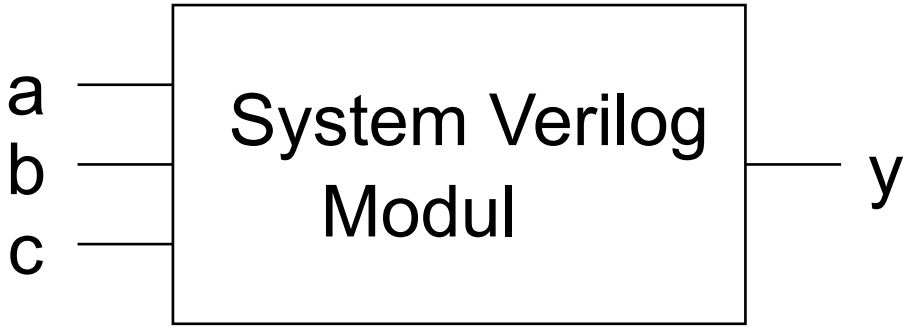
- **Simulation**
  - Eingangswerte werden in HDL-Beschreibung eingegeben
    - Beschriebene Schaltung wird stimuliert
  - Berechnete Ausgangswerte werden auf Korrektheit geprüft
  - Fehlersuche viel einfacher und billiger als in realer Hardware

- **Synthese**
  - Übersetzt HDL-Beschreibungen in Netzlisten
    - Logikgatter (Schaltungselemente)
    - Verbindungen (Verbindungsknoten)

WICHTIG:

Beim Verfassen von HDL-Beschreibungen ist es essentiell wichtig, immer die vom Programm beschriebene **Hardware** im Auge zu behalten!

Verilog-Module



Zwei Arten von Beschreibungen in Modulen:

- Verhalten: Was tut die Schaltung?
- Struktur: Wie ist die Schaltung aus Untermodulen aufgebaut?

Beispiel für Verhaltensbeschreibung

TECHNISCHE UNIVERSITÄT DARMSTADT

System Verilog:

```
module example (input  logic a, b, c,
                output logic y);

  assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;

endmodule
```

LEHRE WIKI TEST IN ZWEI FOLIEN

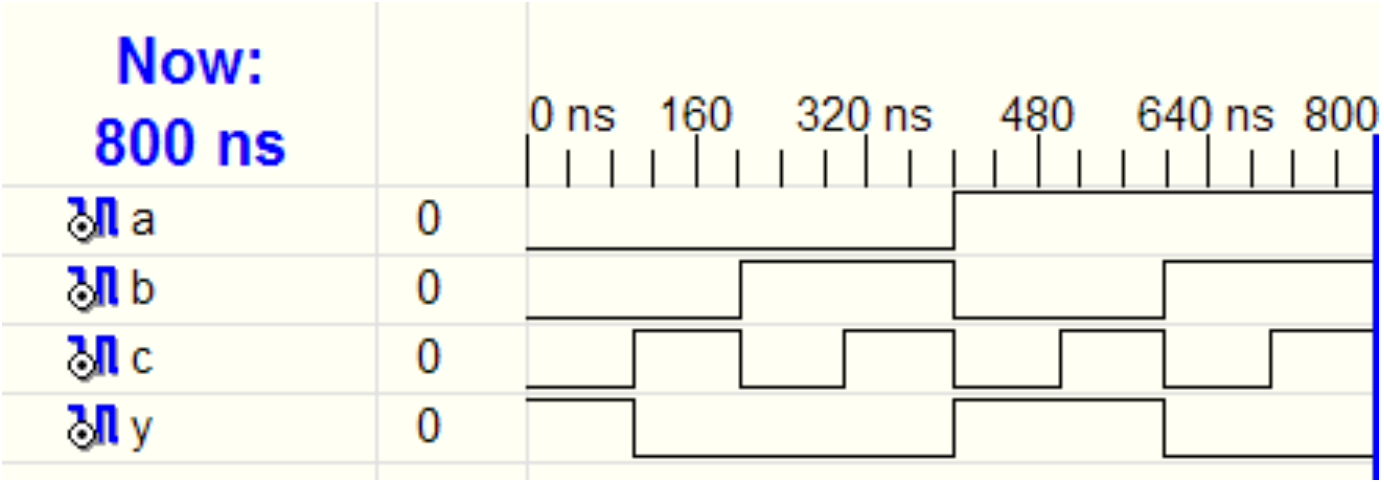
Simulation von Verhaltensbeschreibungen

TECHNISCHE UNIVERSITÄT DARMSTADT

System Verilog:

```
module example (input  logic a, b, c,
                output logic y);
  assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;
endmodule
```

Signalverlaufsdiagramm (*waves*)



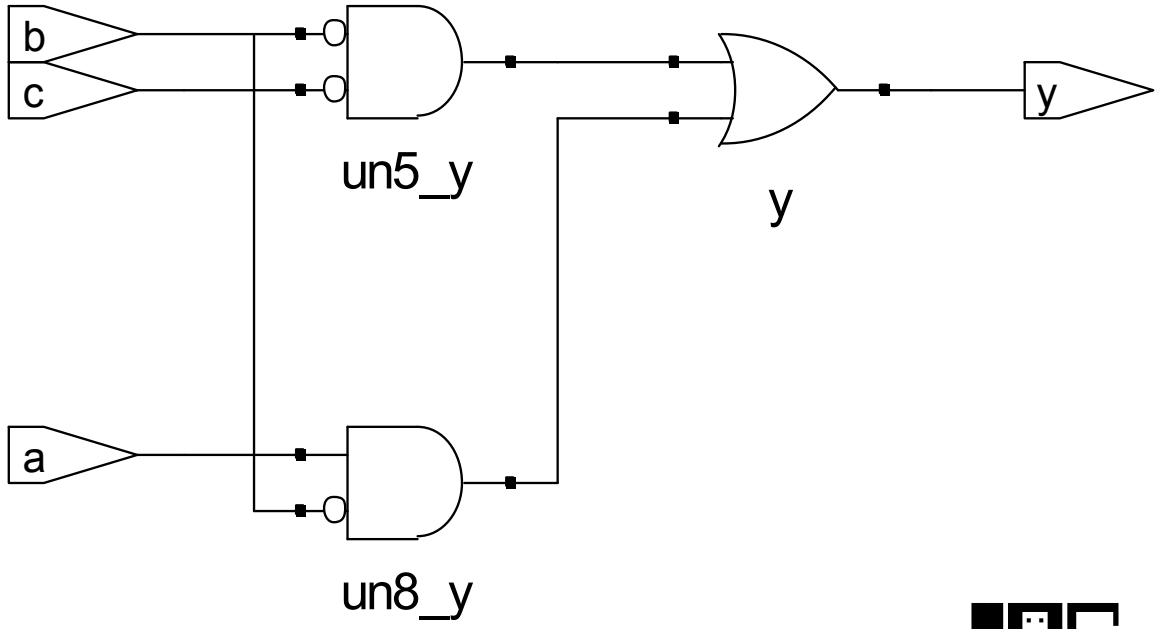
Synthese von Verhaltensbeschreibungen

TECHNISCHE UNIVERSITÄT DARMSTADT

SystemVerilog:

```
module example (input  logic a, b, c,
                output logic y);
  assign y = ~a & ~b & ~c | a & ~b & ~c | a & ~b & c;
endmodule
```

Syntheseergebnis:



Verilog Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

- Unterscheidet Groß- und Kleinschreibung
  - Beispiel: `reset` und `Reset` sind nicht das gleiche Signal
- Namen dürfen nicht mit Ziffern anfangen
  - Beispiel: `2mux` ist ein ungültiger Name
- Anzahl von Leerzeichen, Leerzeilen und Tabulatoren irrelevant
- Kommentare:
  - `//` bis zum Ende der Zeile
  - `/*` über mehrere Zeilen `*/`

Sehr ähnlich zu C und Java!

LEHRE WIKI FRAGE

- Bitte jetzt auf LEHRE WIKI eine Frage beantworten!

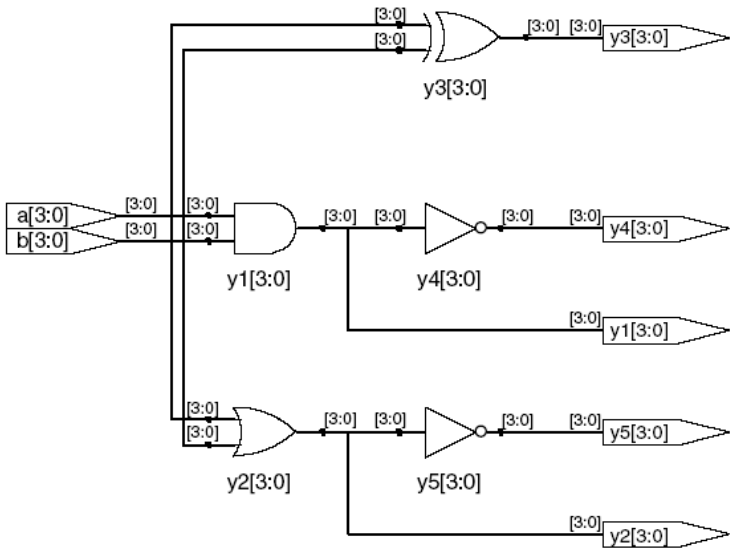
Strukturelle Beschreibung: Modulhierarchie

TECHNISCHE UNIVERSITÄT DARMSTADT

```
module and3 (input  logic a, b, c,
             output logic y);
  assign y = a & b & c;
endmodule

module inv (input  logic a,
            output logic y);
  assign y = ~a;
endmodule

module nand3 (input  logic a, b, c,
              output logic y);
  logic n1;            // internes Signal(Verbindungsknoten)

  and3 andgate  (a, b, c, n1); /* Instanz von and3 namens
                                  andgate */
  inv  inverter (n1, y); // Instanz von inv namens inverter
endmodule
```

Bitweise Verknüpfungsoperatoren

```
module gates (input  logic[3:0]  a, b,
              output logic [3:0] y1, y2, y3, y4, y5);
   /* Fünf unterschiedliche Logikgatter
      mit zwei Eingängen, jeweils 4b Busse */
   assign y1 = a & b;     // AND
   assign y2 = a | b;     // OR
   assign y3 = a ^ b;     // XOR
   assign y4 = ~(a & b); // NAND
   assign y5 = ~(a | b); // NOR
endmodule
```



`//` Kommentar bis zum Zeilenende

`/*...*/` Mehrzeiliger Kommentar

Reduktionsoperatoren

```
module and8 (input  logic[7:0] a,
             output logic y);
   assign y = &a;
   // &a ist Abkürzung für
   // assign y = a[7] & a[6] & a[5] & a[4] &
   //            a[3] & a[2] & a[1] & a[0];
endmodule
```