

Bounded Model Checking mit SystemC

S. Kinder , R. Drechsler, J. Peleska
Universität Bremen

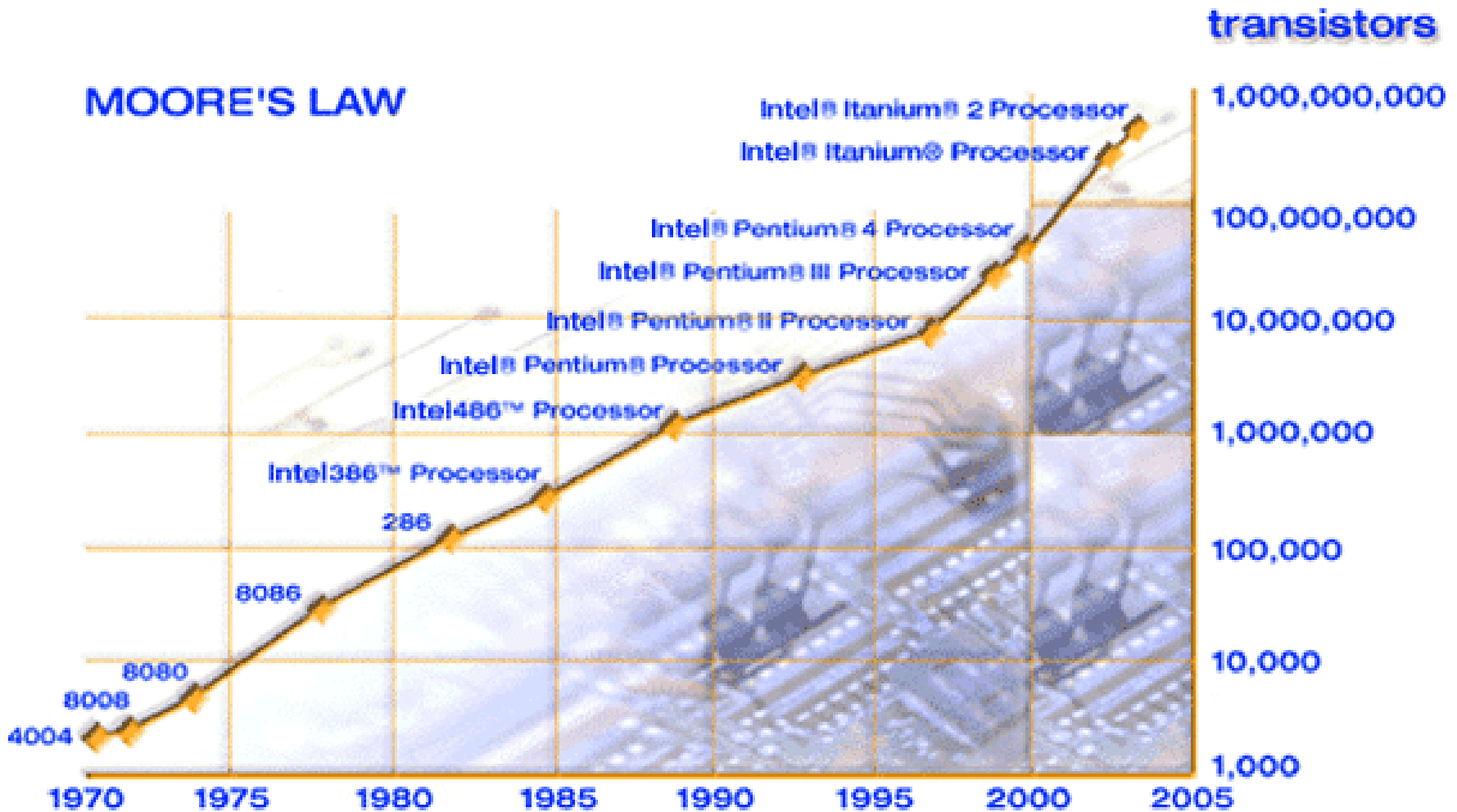
{kinder,drechsle,jp}@informatik.uni-bremen.de



Überblick

- Motivation
- Formale Verifikation
 - Äquivalenzvergleich
 - Eigenschaftsprüfung
- System Level Beschreibungssprache SystemC
 - Konzepte & Modellierung
 - Bounded Model Checking
- Zusammenfassung

Motivation



Quelle: Intel

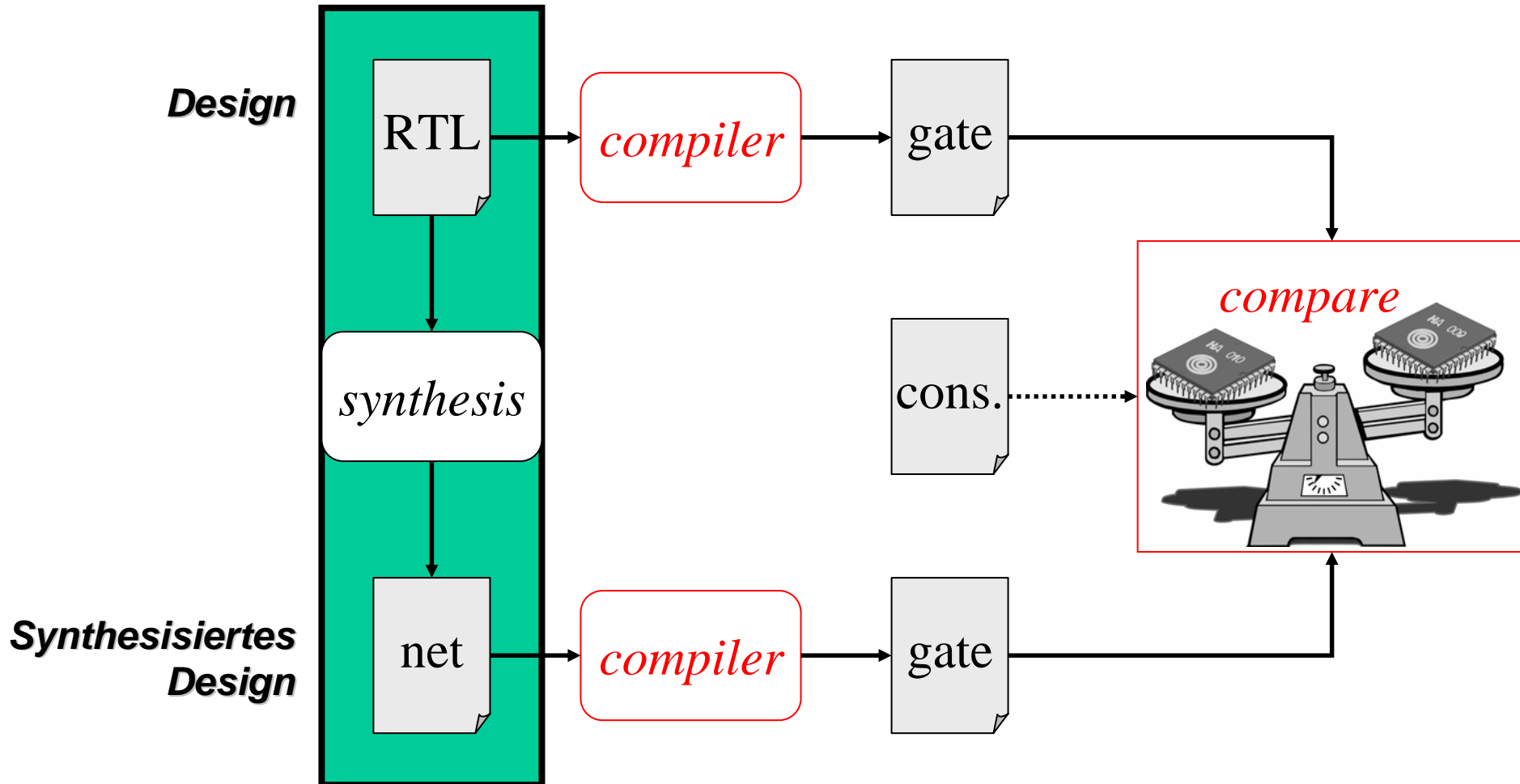
Motivation

- Verifikation immer wichtiger
 - Bis zu 80% der Entwurfskosten
 - Schnell wachsender Markt
- Notwendigkeit von:
 - Effizienten *push-button Werkzeugen*
 - Schnellen Verifikationsalgorithmen
- System Level
 - Top-down Entwurf
 - Hardware-Software Co-Design
 - Hohe Abstraktionsebene

Existierende Techniken

- Validierung von Entwürfen
 - *Coverage* durch Simulation zu niedrig
 - Randfälle und Details kaum zu finden
 - Uneffizient bei der Fehlerlokalisierung
 - Zu viel Handarbeit
- Äquivalenzvergleich
 - Benötigt korrektes Referenzmodell
- Eigenschaftsprüfung (Property Checking – PC)
 - Model Checking
 - Bounded Model Checking

Äquivalenzvergleich



Äquivalenzvergleich

- Funktionale Äquivalenz
- Prüft komplette Designs
- Moderate Kompilierungs- und Verifikationszeiten

Eigenschaftsprüfung

- Prüft eine Eigenschaft (Property) gegen das RT Design
- Eigenschaften werden in temporaler Logik formuliert
- Ergebnis:
 - Beweis der Gültigkeit der Eigenschaft
 - Gegenbeispiel für Fehlerbehebung

Eigenschaftsprüfung

- Anwendbar auf Blöcke des Designs – nicht auf komplette Designs
- Leicht zu nutzen für Designer und zur Verifikation
- **Frühe** Fehlererkennung

Eigenschaft

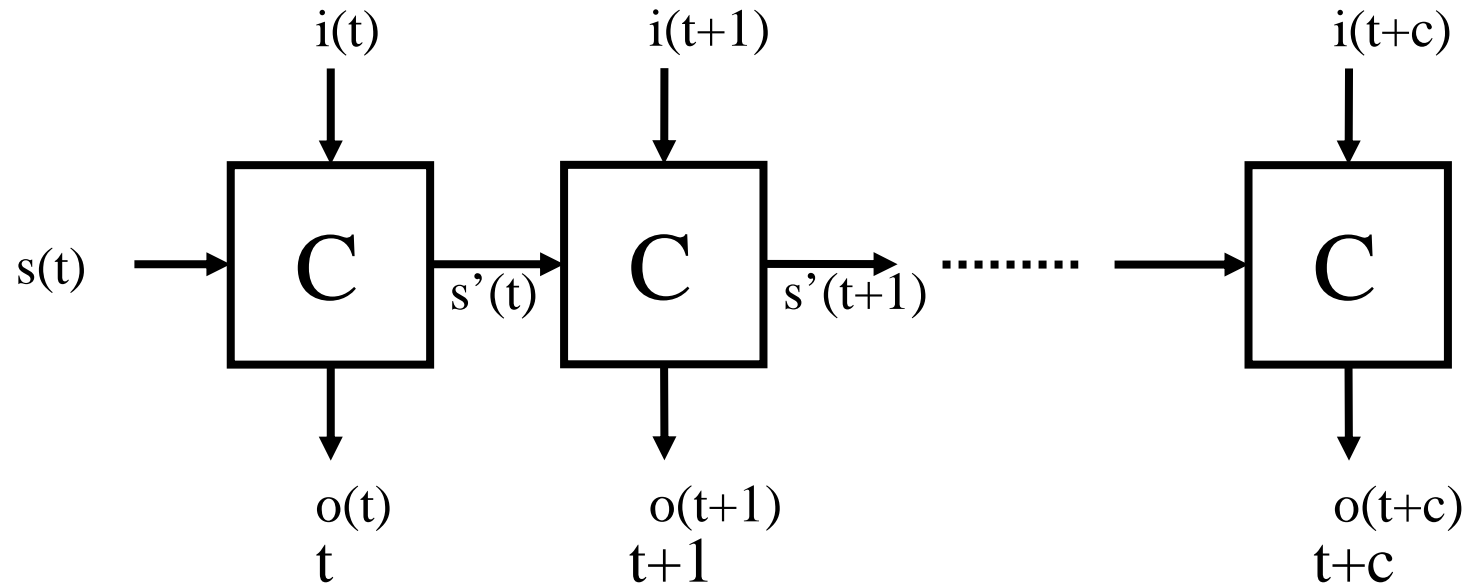
- Eigenschaften bestehen aus zwei Teilen:
 - Liste von Annahmen (*assumptions*)
 - Liste von Zusicherungen (*commitments*)
- Wenn alle Annahmen gelten, müssen auch die Zusicherungen gelten

Beispiel Eigenschaft

- Immer wenn Signal $x = 1$ wird, muss Signal $y = 2$ sein (2 Taktzyklen später)

```
always(  
    //assumption  
    (x = 1)  
    //proof  
    ->  
    next[2] (y = 2)  
)
```

Bounded Model Checking



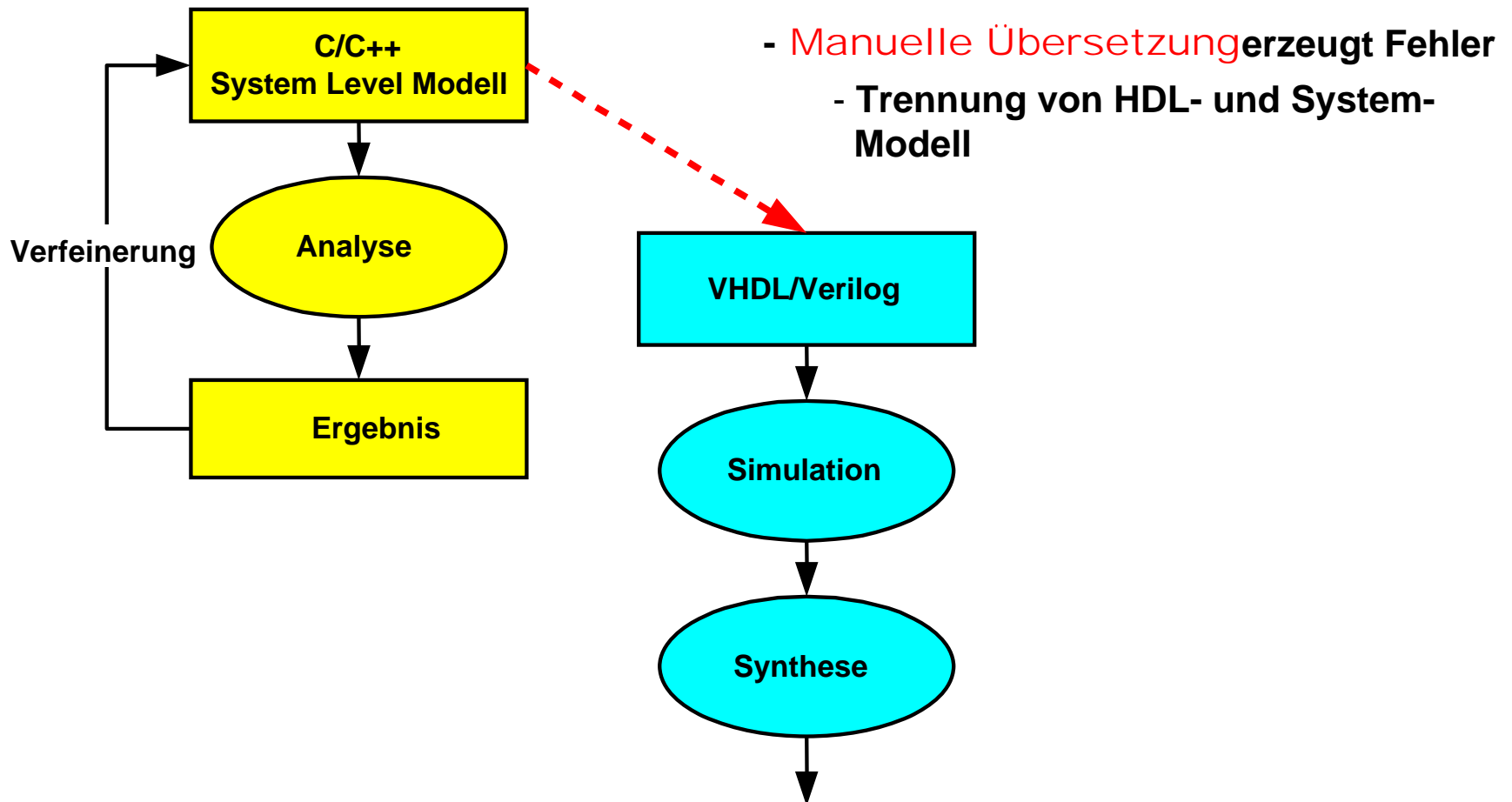
- Sequentielles Problem in kombinatorisches Problem umgewandelt durch abrollen des Designs
- Beobachtungszeitraum: $t - t+c$

System Level

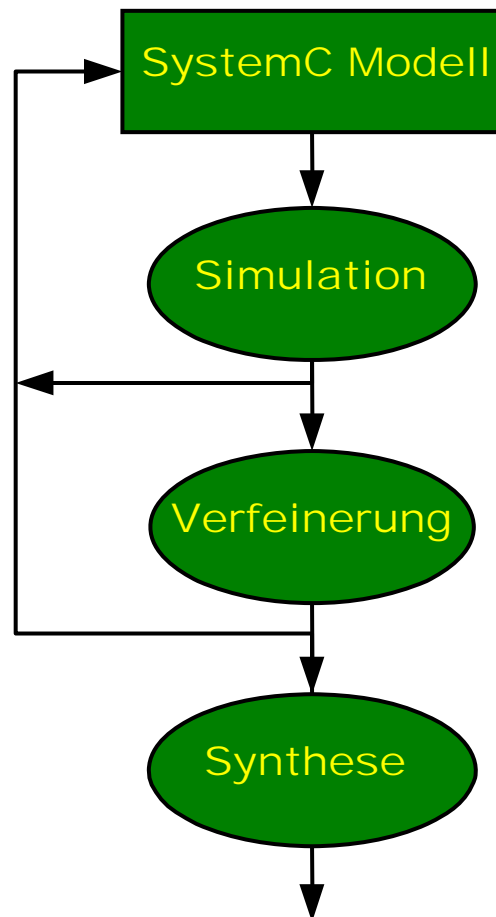
- Höhere Abstraktionsebenen
- **Frühe** Fehlererkennung

- C-ähnliche Beschreibungssprachen
 - Hardware-Software-Co-Design in der gleichen Umgebung
 - **Hier:** SystemC

Gegenwärtige Methodik



SystemC Methodik



- Verfeinerungsmethodik:

Keine Übersetzung von C nach HDL
(Timing Konstrukte)

- Vom System- bis zum RTL Modell: Eine Sprache

SystemC Konzepte

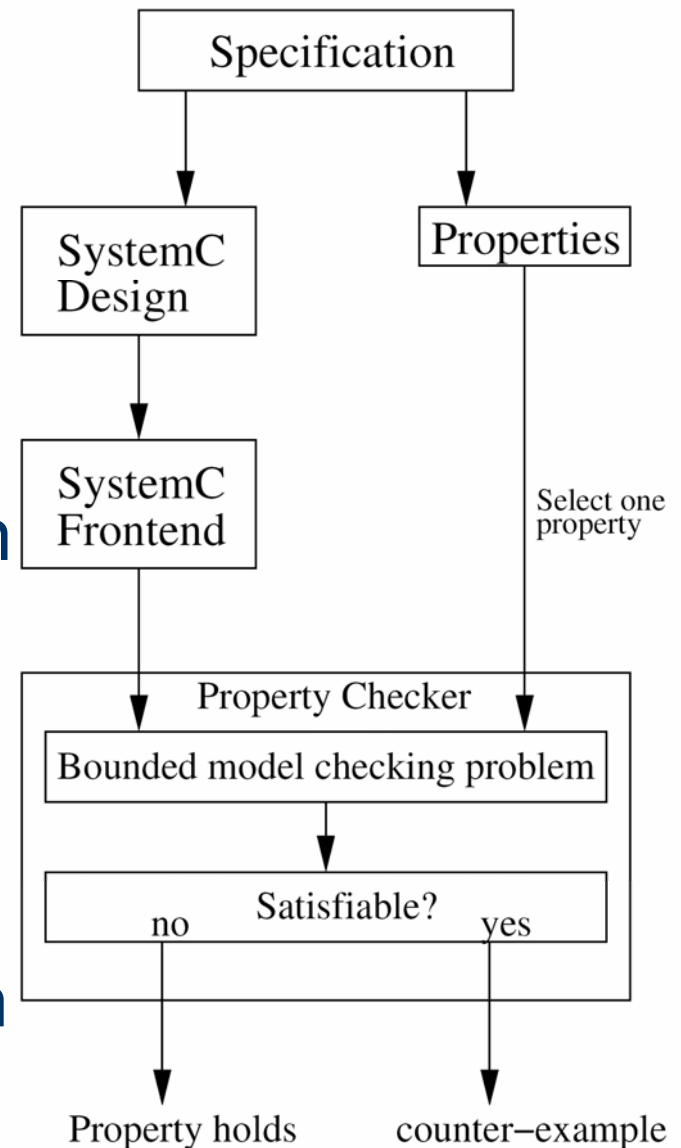
- C++ Klassenbibliothek
- System Level Design
- Schnelle Simulation
- Hardware-Software Co-Design
- Simulationskernel
- Ausführbare Spezifikation

SystemC Modellierung

- Module
(Aufteilung des Designs, Hierarchie)
- Prozesse
(Funktionalität, Nebenläufigkeit)
- Interfaces, Ports, Channels
(Kommunikation)
- Hardware Datentypen
(clocks, bit vectors, ...)

BMC mit SystemC

- Hierarchische SystemC Beschreibung
- Transformation der SystemC Beschreibung in seinen endlichen Zustandsautomat (FSM)
- Eigenschaft + FSM Repräsentation zu einer SAT Instanz konvertieren



BMC mit SystemC

- Formale Verifikation
 - Bestehende Techniken
- Höhere Level der Beschreibung
- Wenn RT Ebene verfügbar
 - Keine signifikanten Änderungen

Beispiel: Bubble Sort (1)

```
typedef sc_uint<4> T; // scalable
```

```
SC_MODULE( bubble ) {  
    sc_in< T >    in[8];  
    sc_out< T >   out[8];  
    T buf[8];
```

```
void do_it(); // sort process
```

```
    SC_CTOR( bubble ) {  
        SC_METHOD(do_it);  
        sensitive<<in[0]<<in[1]<<in[2]<<in[3]  
                <<in[4]<<in[5]<<in[6]<<in[7];
```

```
    }  
};
```

Beispiel: Bubble Sort (2)

```
void bubble::do_it() {
    for( int i = 0; i < 8; i++ ) buf[i] = in[i];
    for( int i = 0; i < 8-1; i++ ) {
        for( int j = 0; j < (8-i)-1; j++ ) {
            if( buf[j] > buf[j+1] ) {
                T tmp;
                tmp = buf[j];
                buf[j] = buf[j+1];
                buf[j+1] = tmp;
            }
        }
    }
    for( int i = 0; i < 8; i++ ) out[i] = buf[i];
}
```

Experiment

- SystemC 2.0.1
- Pentium IV 3 GHz mit 1GB RAM
- Linux
- Laufzeiten in CPU-Sekunden

Beispiel: Bubble Sort

- Eigenschaft: Ausgänge sind sortiert
- Ergebnisse für unterschiedlich Bitbreiten

Bitbreite	Klauseln	Literale	Zeit in s
4	6390	14458	17.18
8	12754	28894	286.93
16	25482	57766	125.25
32	50938	115510	560.48

System Level PC

- Kann generalisiert werden
- Aber:
 - Komplexitätsprobleme
- Alternative:
 - Verifikation von (Kommunikations-) Protokollen für verifizierte Blöcke

Zusammenfassung

- Formale Verifikation
 - Mächtig auf Chip-Ebene
 - PC effektiv auf Blockebene
- System Level
 - PC und BMC wenn RTL verfügbar
 - Top-down Entwurf