Entwicklung einer Systemumgebung für “System Level Design” in SpecC

Dipl.-Ing. (FH) David Berner, ASIC Design Center
Fachhochschule Offenburg, Badstrasse 24, 77652 Offenburg
Tel. 0781/205-274, Fax 0781/205-174
david.berner@fh-offenburg.de

Um bei ständig verkürzten Entwicklungszyklen immer komplexere Designs bearbeiten zu können, muss auf immer höhere Abstraktionsebenen gewechselt werden. SpecC ist eine Methodik und eine Sprache zur Entwicklung auf Systemebene.

Um Akzeptanz auf industrieller Ebene zu erhalten, müssen Tools vorhanden sein, die einen anschaulichen und interaktiven Gang durch den Design-Flow ermöglichen. Dies war bis jetzt ein Schwachpunkt von SpecC, soll sich jedoch mit der Entwicklung von RESpecCT ändern.

1. Einführung

Im Jahr 1965 hat Gordon Moore, Mitgründer von Intel, vorausgesagt, dass sich die Transistordichte von Halbleiterchips ungefähr alle 18 Monate verdoppeln werde. Im Februar 2001 kündigte Pat Gelsinger (Intels Technologie-Kopf) auf der International Solid State Circuit Conference (ISSCC) den 1 TIPS-Prozessor mit 10 Milliarden Transistoren und 30 GHz für das Jahr 2010 an. Ein aktueller P4 Prozessor besteht aus 42 Millionen Transistoren. Da die Produktivität der Entwickler in der Vergangenheit nur um ca. 20% pro Jahr gestiegen ist, wird klar, dass ein solches Ziel mit herkömmlichen Mitteln nicht erreichbar ist.

Ein weiteres Phänomen heutzutage ist, dass Produktentwicklungszyklen immer kürzer werden. Der Markt entwickelt sich derart schnell, dass das, was heute sehr gefragt ist in einem Jahr schon keiner mehr haben will. Die "time to market" entscheidet über den Erfolg oder Misserfolg eines Produkts. Wie ist jedoch eine Verkürzung der Entwicklungszeiten bei immer komplexeren Systemen möglich?

Möglichkeiten die Produktivität derart entscheidend zu steigern sehen wir heute in der konsequenten Wiederverwertung von Intellectual Property (IP reuse) und in dem Wechsel zu höheren Abstraktionsebenen - besser noch: in der Kombination von beidem.

2. System Level Design

Historisch gesehen ging die Entwicklung immer von niederen Ebenen zu höheren Ebenen. Als Parallele kann man das Programmieren betrachten: Ausgehend von dem Stanzen von Lochkarten in Maschinencode über Assembler-Programmierung und der Programmierung in strukturierten

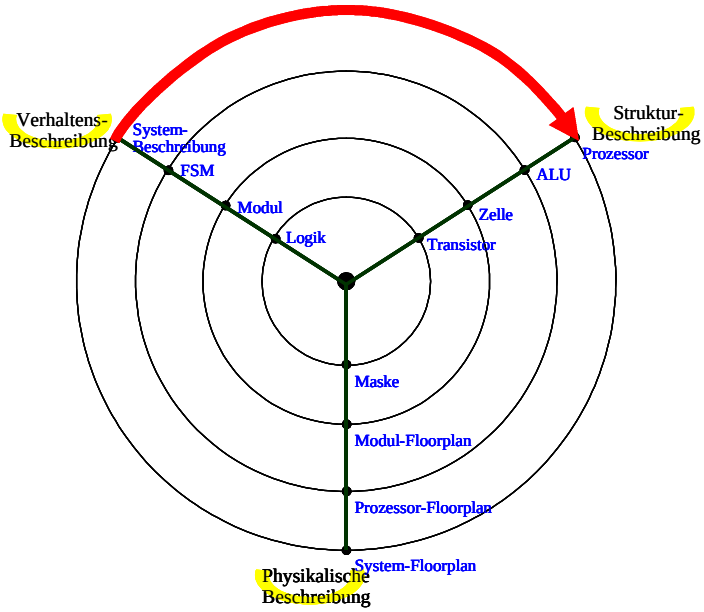

Abbildung 1: *Gajski Y mit System Level Design*

Hochsprachen ging man über zur objektorientierten Programmierung. Dabei wird die innere Struktur von Problemen direkt modelliert ohne auf die letztendliche Realisierung in Maschinensprache Rücksicht zu nehmen. Sicher werden alle anderen Arten von Programmierung heute immer noch verwendet, doch sobald Probleme eine gewisse Komplexität erreichen führt heute kein Weg mehr an der objektorientierten Programmierung vorbei.

In der Schaltungsentwicklung begann es mit dem Designen der eigentlichen Schaltung mit allen Bauteilen auf Silizium. Der Weg führte über Transistor-Logik und Gatter-Logik zu High-Level-Design mit Sprachen wie VHDL und Verilog, wo aus einem lösungsorientierten Sprachkonstrukt die niederen Ebenen automatisch erzeugt werden.

Um noch komplexere Systeme schnell zu realisieren, ging man dazu über, die funktionale Spezifikation in einer Hochsprache wie C zu schreiben, zu testen und zu verifizieren. Dann wurde diese Beschreibung (von Hand) überführt in eine VHDL-Beschreibung. Auf diese Weise kann die Funktionalität des Systems zu einem sehr frühen Zeitpunkt überprüft werden. Es können Modifikationen vorgenommen werden bevor schon zu viel Entwicklungsarbeit investiert wurde. Diese Art von Design-Vorgehen wird "System Level Design" genannt (*Abbildung 1*).

Ein grosses Problem dabei ist die Überführung des C-Modells auf die VHDL-Ebene. Da sich diese beiden Ebenen in einigen Punkten grundsätzlich unterscheiden, ist eine Umsetzung mit erheblichem Aufwand verbunden und fehlerbehaftet. Man ist deshalb bestrebt diese Lücke zu schliessen und einen Design-Fluss zu bekommen, der von der Systemspezifikation bis zur Transistorebene konsistent ist.

3. SpecC

Wer sich mit System Level Design beschäftigt, wird zwangsläufig auf den Begriff SpecC stossen. Der Begriff SpecC taucht jedoch auch im Zusammenhang mit vielen anderen Stichworten auf wie z.B.:

- Hardware Software Co-Design
- SOC, Embedded System
- Unterschiedliche Abstraktionsebenen
- Benutzer geführte Verfeinerung
- Simulation auf verschiedenen Ebenen
- Trennung von Funktion und Kommunikation
- Vollständige und orthogonale Sprache

Doch was ist denn nun SpecC genau?

3.1. Methodik

Zuerst ist SpecC eine Methodik die einen Synthesefluss beschreibt (*Abbildung 2*).

Dabei werden von der System-Spezifikation bis zur Implementierung vier verschiedene Abstraktionseneben definiert:

- Spezifikationsebene
- Architekturebene
- Kommunikationsebene
- Implementierungsebene

Der Übergang zwischen diesen Ebenen erfolgt weitgehend automatisiert, es müssen nur wenige Design-Entscheidungen getroffen werden wie:

- Aufteilung Hardware / Software
- Auswahl von Prozessoren
- Festlegung von Ausführungsreihenfolgen
- Auswahl von Protokollen

Dieser Vorgang wird auch "Benutzer geführte Verfeinerung" genannt.

Der konsistente Synthese-Fluss wird ergänzt durch einen Validierungs-Fluss (*Abbildung 2*).

Der Validierungs-Fluss besteht darin, dass das System auf jedem Abstraktionslevel simulierbar und verifizierbar sein soll. Auf dem Spezifikationslevel wird die Funktion des Gesamtsystems nachgewiesen. Durch den konsistenten Synthese-Fluss reicht es aus in den unteren Ebenen nur noch die sogenannten Hot-Spots zu verifizieren, die Gesamtfunktionalität ist nach wie vor gegeben. Da in niedrigeren Ebenen die Simulationsperformance durch mehr Detailinformationen drastisch zurück geht und eine Simulation des Gesamtsystems praktisch unmöglich wird, ist dies ein entscheidender Vorteil.

SpecC war also erst nur eine Methodik, was fehlte war eine Implementierung davon.

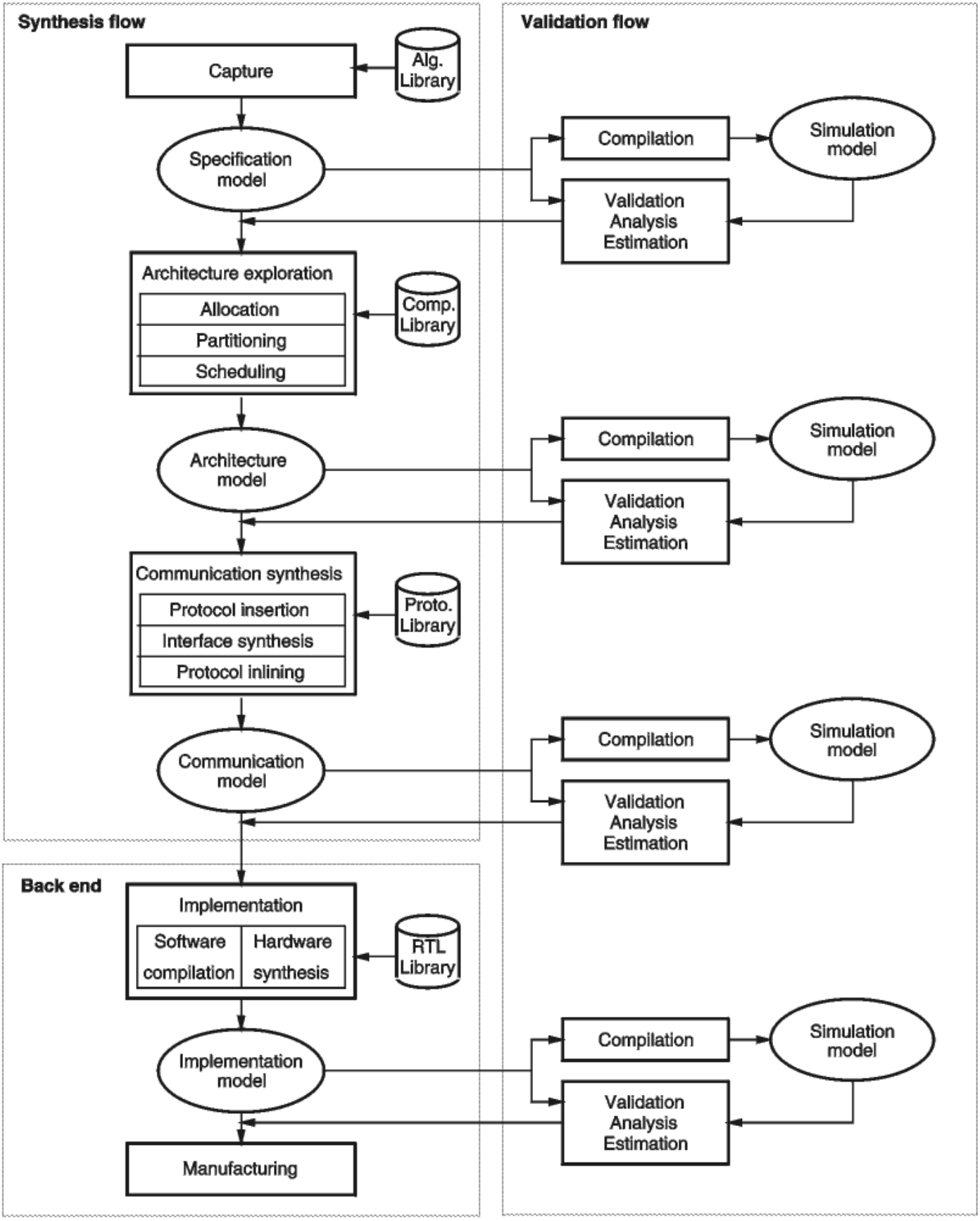

Abbildung 2: *SpecC Sythese- und Validierungs-Fluss*

3.2. SpecC - Yet another language?

Auf der Suche nach einer Implementierung für die SpecC Methodik wurden erst bestehende Sprachen evaluiert (

Abbildung 3).

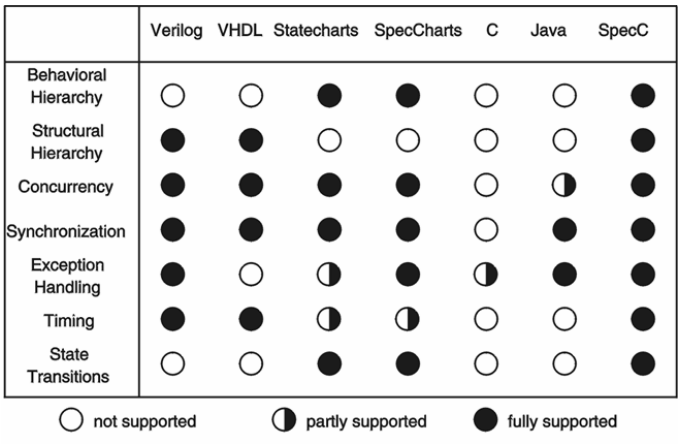

Abbildung 3: *Vergleich bestehender Sprachen*

Dabei wurde festgestellt, dass es bis jetzt keine Sprache gibt, die die Problematik von System Level Design vollständig abdeckt. Es wurde also klar, dass eine eigene Sprache entwickelt werden musste, die der Problematik gerecht wird.

Um den Einarbeitungsaufwand von Ingenieuren möglichst gering zu halten wurde SpecC auf Basis von ANSII C entwickelt. Dies macht Sinn, da heute schon herkömmliches "System Level Design" mit C beginnt.

3.3. Sprache

Die Sprache SpecC ist also ein echtes Superset von ANSI-C. Das heisst jedes C Programm ist ein gültiges SpecC Programm. Es führt nur wenige Syntax-Erweiterungen ein, die der Sprache sieben Konzepte hinzufügen:

- Verhaltens Hierarchie
- Strukturelle Hierarchie
- Parallelismus
- Synchronisation
- Timing
- Zustandsübergänge

Diese Erweiterungen machen SpecC zu einer vollständigen und orthogonalen Implementierung der SpecC Methodik. Die Beschreibung von beliebigen Systemen ist auf allen Ebenen möglich.

Um die Entwicklungsenergie um SpecC zu bündeln ist ein Standardisierungsgremium gegründet worden. Das SpecC Technology Open Consortium (STOC) hat über 40 Mitglieder aus Industrie und Forschung und berät über Modifikationen des offenen Standards.

```
interface I1
{
  bit[63:0] Read(void);
  void Write(bit[63:0]);
};

channel C1 implements I1;

behavior B1(in int, I1, out int);

behavior B(in int p1, out int p2)
{
  int v1;
  C1  c1;
  B1  b1(p1, c1, v1),
      b2(v1, c1, p2);
  void main(void)
  { par { b1.main();
          b2.main();
        }
   }
};
```

Abbildung 4: *SpecC Beispielcode*

4. RESpecCT

4.1. Idee

Um nun SpecC für den Anwender interessant zu machen ist es essentiell Tools zu haben, deren Umgang den Nutzer möglichst intuitiv durch die zu erledigenden Aufgaben führt.

Die Idee war nun ein Tool zu entwickeln, das eine grafische Implementierung der SpecC Methodik ist mit der SpecC-Sprache als Basis. Es sollte Modellierung, Analyse, Verfeinerung und Synthese in sich vereinen. Dieses Tool entstand im Rahmen meiner Diplomarbeit am Center for Embedded Computer Systems der University of California, Irvine und bekam den Namen RESpecCT (=

Refinement and Exploration tool for the SpecC Technology).

Seine wesentlichen Merkmale sind:

- Visualisierung von Designs
- Anzeige von Analyse-Ergebnissen
- Eingabe von Design-Entscheidungen
- Leicht zu bedienen
- Übersichtlich
- Plattform-unabhängig

Die Implementierung der Übergänge zwischen den Abstraktionsebenen wurden von Doktoranden gemacht und als shared libraries eingebunden.

Die Programmierung erfolgte mit Python, einer sehr flexiblen High-Level Sprache und QT als grafischem Toolkit.

4.2. IDE

Abbildung 5 zeigt die RESpecCT Oberfläche. Grundsätzlich kann links die Behavior-Hierarchie eingesehen werden. Auf der rechten Seite ist ein MDI (Multiple Document Interface) in dem Informationen über bestimmte Teile des Designs angezeigt werden können oder der SpecC Code direkt editiert werden kann.

Der Benutzer wird nach und nach durch die verschiedenen Designebenen geführt und bekommt Dialoge präsentiert in denen gewisse Designentscheidungen getroffen werden müssen.

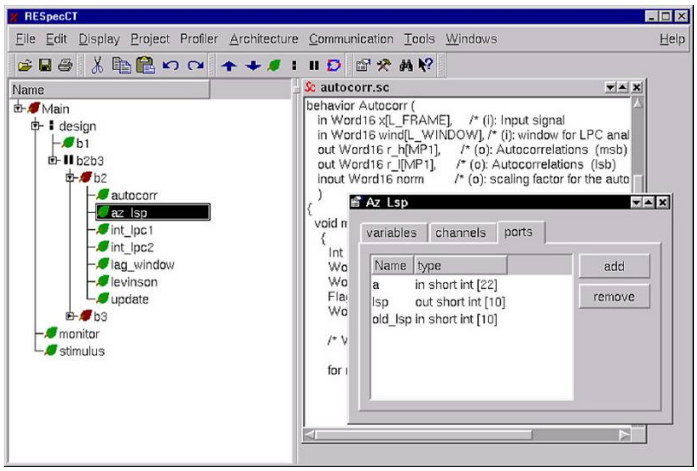

Abbildung 5: *RESpecCT IDE*

4.3. Designanalyse (Profiling)

Die Designanalyse (Profiling, Estimation) dient dazu möglichst viele Informationen über das Design zu sammeln und dem Nutzer zur Verfügung zu stellen, damit er bessere Entscheidungen fällen kann. Er bekommt dabei z.B. Auskunft über Art und Häufigkeit von bestimmten Operationen in beliebigen "Zweigen" des Designs (*Abbildung 6*).

Ausserdem kann die Designanalyse dabei helfen kritische Punkte im Design (sog. hot spots) zu detektieren und durch Änderungen von Designentscheidungen oder Anpassung des Codes zu entschärfen.

Dabei kann auch die Einhaltung von Design-constraints überprüft werden.

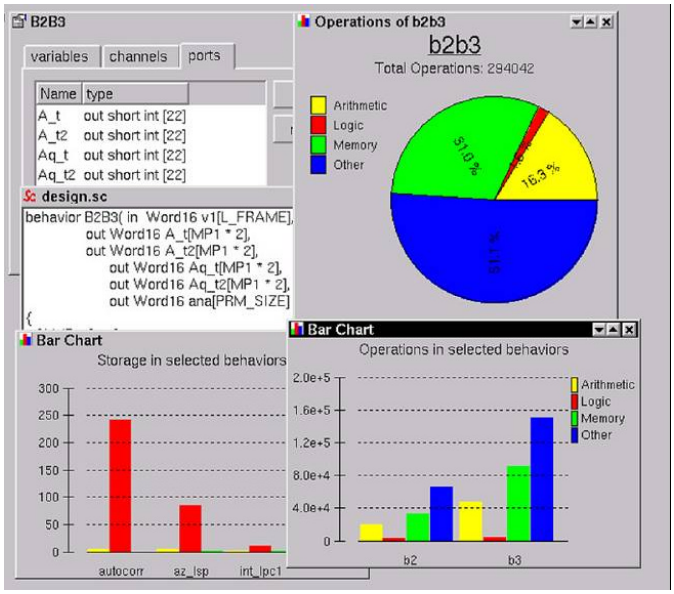

Abbildung 6: *Ergebnisse der Designanalyse*

5. Zusammenfassung

Jedem in der Chipbranche ist längst klar, dass in Zukunft ganz andere Tools zur Entwicklung von Chips, Embedded Systemen und Systemen allgemein benötigt werden. Welches Tool sich letztendlich auf dem Markt durchsetzen wird wird die Zukunft entscheiden.

Im Vergleich mit anderen Ansätzen wie System-C ist SpecC deutlich konsequenter und konsistenter.

Alle Ansätze sind immer noch weit von einer breiten industriellen Nutzung entfernt, jedoch kommt SpecC mit RESpecCT diesem Ziel einen entscheidenden

Schritt näher. Auch wenn noch viel Arbeit gemacht werden muss in Bezug auf Bibliotheken und im Backend so zeigt sich heute schon, dass die Kombination von:

a) der SpecC Methodik, geschaffen für die Komplexität zukünftiger Designs, in Verbindung mit

b) der SpecC Sprache, die perfekt an die Problematik von System Level Design angepasst ist und

c) RESpecCT, einer grafischen Erweiterung die intuitive Interaktion mit den Designs ermöglicht,

auf dem besten Weg ist ein mächtiges Werkzeug für die effiziente Entwicklung von komplexen Systemen der Zukunft zu werden

6. Referenzen

- [1] David Berner, *Development of a Visual Refinement and Exploration Tool for SpecC*, Technical Report ICS-01-12, Irvine, Februar 2001.
- *[2] D.Gajski et al, SpecC: Specification Language and Design Methodology*, Kluwer Academic Publishers, 2000.
- [3] Gerstlauer, Dömer, Peng, Gajski, *System Design: A Practical Guide with SpecC*, Kluwer Academic Publishers, 2001.
- [3] R.Dömer, J. Zhu, D. Gajski, *The SpecC Language Reference Manual*, Technical Report ICS-98-13, Irvine, Januar 1999.

Online:

- SpecC homepage an der University of California Irvine http://www.cecs.uci.edu/~specc/
- SpecC Open Technology Consortium (STOC) http://www.specc.org/