System Level Design Technologies
and
System Level Design Languages

SLD Study Group
EDA-TC, JEITA

Problems to Be Solved

1. Functional Design
   • Ambiguity of specification
   • Change of system specification
   • Slow simulation speed

2. Architecture Design
   • No methodology for verification
   • Low accuracy of estimation

3. Implementation
   • No effective design tool from High Level to RTL design flow
   • Mismatch between implementation and specification

4. Design database, Co-simulation
   • Difficulty of IP customization
   • Lack of performance of co-simulation tools for SW design
Solutions

1. Functional Design
For handling the ambiguity of specification ...
- Methodology to define functional specification
- Executable SLD language

2. Architecture Design
For earlier estimation of cost, performance, etc. ...
- Fast exploration method of finding optimum architecture
- Estimation model and design database

3. Implementation
For easy link to implementation phase ...
- Interface synthesis of HW/SW
- Equivalency check

4. Design database, Co-simulation
For easiness of IP use, high speed co-simulation ...
- Standardized I/F of IP, IP synthesis, Highly abstracted HW model
System Level Design Technologies

- Research Target
  - “EDA Technology Roadmap Toward 2002”

- Universities/Research Centers
  - POLIS
    http://www-cad.eecs.berkeley.edu/Respep/Research/hsc/abstract.html
    “Hardware-Software Co-Design of Embedded Systems, The POLIS Approach”
  - OCAPI-xl
    http://www.imec.be/ocapi
  - IpChinook

- Vendors
  - CoCentric
    http://www.synopsys.com/
  - N2C
    http://www.coware.com/
  - VCC
    http://www.cadence.com/
### Roadmap of EDA Technologies

#### Items

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardization of system model</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standardization of system level design language (SLDL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulation of SLD Language</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance estimation in system level design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(application soft, compiler, performance estimation of HW, optimization)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High speed prototyping</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal verification (Specification - System)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW/SW partition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System level library (IP core, middleware, etc.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision support of system level test strategy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tradeoff of analog/digital</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardization of architecture model</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standardization of architecture description language</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimation in architecture level (area, delay, power...)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision of RTL constraint</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(area, timing, power, budgeting of floor plan)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal verification (System - Architecture)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation/Simulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture synthesis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-synthesis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision support of architecture level test strategy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(test method/area/length of test)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DFT in architecture level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

In the prediction, almost all technologies are in the trial phase in 2000!! How in the real world?

Cited from EIAJ/EDA Technology Roadmap Toward 2002

©JEITA 2001 All rights reserved
Design Tools (1) by Universities and Research Centers

**POLIS (UCB, U.S.A.)**
- Co-design Environment for embedded system based on the CPU and DSP
- Supporting functional design to prototyping
- Architecture exploration by mapping behavior to architecture
- Accurate and high-level performance estimation using a fuzzy instruction set

**IpChinook (U of Washington, U.S.A.)**
- Based on IP-based/Reused-Design methodology
- Automatic generation of control and I/F between functional blocks
- Lots of communication protocol libraries
- Support of Java, Pia C

**OCAPI-xl (IMEC, Belgium)**
- C++ model which can describe HW and SW uniformly
- Optimal block partition using a performance analysis
- Automatic generation of VHDL / Verilog-HDL / C

---

- **System Specification**
  - Capture Behavior
  - Capture Architecture
  - Verify Behavior
  - Verify Architecture
  - Map Behavior to Architecture
  - Verify Performance
  - Refine HWSW to Architecture
  - Link to Architecture Verification
  - Link to HW / SW Implementation

- **Behavioral Description**
  - Control Composition
  - Data Composition & Timing Constraints

- **Target Description**
  - Allocation
  - Heterogeneous Distributed Architecture

- **Verification**
  - Control Synthesis
  - Communication & Interface Synthesis
  - Real-time Scheduling
  - Simulation

- **Implementation**
  - Output Schematic & Code
CoCentric (Synopsys)

- Support HW side and co-verification of SOC Design
- Conversion from float to fix-pointed
- HW Synthesis from behavior (Verilog, VHDL)
- Supporting SystemC

N2C (CoWare)

- Supporting functional level to implementation
- System Definition in Untimed, BCASH, BCA abstraction levels by original C-based language (CoWareC)
- Communication refinement
- Interface synthesis between HW and SW
- Supporting SystemC (in future)

VCC (Cadence)

- Architecture exploration by mapping function to architecture
- Performance estimation using a fuzzy instruction set
- Supporting IP reuse
- Communication synthesis
- Interface to implementation

©JEITA 2001 All rights reserved
Summary of EDA Trend

- May be solved in near future…?
  - Function Verification
  - Architecture Mapping
  - Performance Estimation
  - Interface Synthesis
  - HW/SW Co-Verification

- Unsolved Problems
  - Function Partition
  - High Speed Estimation
  - Estimation of Area, Cost, and Power
  - Standardized Description of system specification, constraints
  
  etc. . . .
Multi-function/
High performance
Requirement of system level optimization

Large/Complex
Heterogeneous components

High IP Utilization

Multi-product, Small-lot Production

Short TAT
Ambiguity of specification causes misunderstanding
Inaccuracy of estimation causes re-spin

Conventional Design Flow
Proposed Design Flow

System Specification
- Constraints
- Function

Function Determination
- Function Definition
- Veri.

Architecture Determination
- Design Space Generation
  - Partitioned Function Definition
  - Candidates of architecture
- Design Space Exploration
- Architecture Mapping
- Est.

Interface to Implementation
- HW/SW Co-Design
Proposed Design Flow

System Specification
- Constraints
- Function

Function Determination
- Function Verification
- Function Definition

Design Space Generation
- Profiling
- Equivalence Checking
- Block Partitioning
- Architecture Generation

Design Space Exploration
- Estimation
- Architecture Mapping

Interface to Implementation
- HW/SW Co-Design

Test Description
- Expected Value

Test Bench

Design Database
- Document DB
- Algorithm IP
- Function Block IP
- Architecture IP
- Estimation DB
- SW IP
- Bus, I/F IP
- HW IP

©JEITA 2001 All rights reserved
Design Space Generation

Block partitioning and architecture generation from function model

**Function Model**
- Represents system by functional blocks and communications between them

**Architecture Model**
- Represents system by components

**Design Space Generation**
- Block partitioning
- Architecture generation
- Profiling

```
func1()
main()
```

```
FB1 ┌─┐ FB2 ┌─┐ FB3
  │   │   │   │
FB4 └─┘FB5 └─┘

CPU ┌─┐ MEM
  │   │
HW   HW
```

©JEITA 2001 All rights reserved
Analyze function models statically/dynamically and extract information for block partitioning and architecture generation

<table>
<thead>
<tr>
<th>Class</th>
<th>Information</th>
<th>Usage (in design space generation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Variable Access</td>
<td>• Number of read/write</td>
<td>• Choice of algorithm</td>
</tr>
<tr>
<td></td>
<td>• Access position</td>
<td>• Decision of block candidates</td>
</tr>
<tr>
<td></td>
<td>• Read-write distance</td>
<td>• Selection of memory/register</td>
</tr>
<tr>
<td>Amount of Data Transfer</td>
<td>• Number of (bits)*(number of exec.) of read/write</td>
<td>• Decision of partitioning units</td>
</tr>
<tr>
<td></td>
<td>• Amount of parameters</td>
<td>• Decision of HW/SW partitioning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Selection of comm. protocol</td>
</tr>
<tr>
<td>Amount of Calculation</td>
<td>• Execution count of lines</td>
<td>• Decision of partitioning units</td>
</tr>
<tr>
<td></td>
<td>• Execution count of blocks</td>
<td>• Performance estimation on HW/SW</td>
</tr>
<tr>
<td>Statistical Analysis</td>
<td>• Type and count of arithmetic/control instructions</td>
<td>• Choice of processor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Refinement of instruction set</td>
</tr>
</tbody>
</table>
Mapping each functional block to a component and select the optimum architecture which satisfies constraints.

**Transaction Model**
represents only the processing cost for estimation

**Architecture Mapping**

**Example**
Processing time = \( \frac{\text{process cost}}{\text{process capability per units}} \)

From transaction model
From architecture model
Transaction Model and Accuracy

Tradeoff of transaction model

Coarse (Fast)  Precise (Slow)

Static Analysis

- Simple summation of average properties (design data)
- Considering algorithm (branch, loop)
- Instruction set

Dynamic Analysis

- Dynamic simulation (using constraints)

Method

Advantages and Disadvantages

- High speed/Low accuracy
- Need DB of past data
- Impossible to critical analysis

- Middle speed/Middle accuracy
- Estimation of average property/range

- Low speed/High accuracy
- Need detailed analysis of implementation
Solution to Problems

- **System Specification**
  - Function
  - Function Definition
  - Function Determination
  - Profiling
  - Block Partitioning
  - Design Space Generation
  - Equivalence Checking
  - Architecture Mapping
  - Architecture Estimation

- **Test Bench**
  - Test Spec.
  - Test Description
  - Expected Value

- **Design Database**
  - Document DB
  - Algorithm IP
  - Function Block IP
  - Architecture IP
  - Estimation DB
  - SW IP
  - Bus, I/F IP
  - HW IP

- **HW/SW Co-Design**
  - Interface to Implementation

- **Test Strategy of Overall Design Flow**
  - Insufficient performance of co-simulator in software development
  - Inaccuracy of estimation
  - Ambiguity of specification
  - Lack of architecture decision tool
  - Handling of ambiguity
  - Difficulty of IP customization
  - Lack of design path to implementation

- **Slow Simulation Speed**
## Technology Map of Tools

<table>
<thead>
<tr>
<th>Technology</th>
<th>Tool1</th>
<th>Tool2</th>
<th>Tool3</th>
<th>Tool4</th>
<th>Tool5</th>
<th>Tool6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Constraints</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
<td>❌</td>
</tr>
<tr>
<td>Function Definition</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Function Verification</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Profiling</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Block Partitioning</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Architecture Generation</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Equivalence Check</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Architecture Mapping</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Performance Estimation</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Interface Synthesis</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>HW/SW Co-Design</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
<tr>
<td>Test Bench Generation</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
<td>❌ &amp; ❌</td>
</tr>
</tbody>
</table>

- □: automated by the tool
- □: tool has supporting function
- □: tool does not have supporting function
- □: not covered by the tool

**Notes:**
- Some tools support this phase but speed/accuracy of the estimation is insufficient.
- Only the verification is supported by all tools.
- No tool supports the test bench generation.
- No tool support is indicated.

©JEITA 2001 All rights reserved
Investigation of System Level Design Languages

- **Background**
  - Important role for system level design flow
  - Standardization activities (OSCI, STOC, Accellera)

- **Objectives**
  - Understand standardization activities (SystemC, SpecC, UML, etc.)
  - Evaluation of the proposed design flow
  - Feedback to standardization groups
SystemC

- **Background**
  - Proposed based on the technologies of Synopsys Inc., CoWare Inc., Frontier Design Inc. (Sept 1999)

- **Features**
  - Representation from system level to HW/SW by adding dedicated class libraries, and without enhancing C/C++ grammar
    - Availability of standard C/C++ development environment
    - Less readability of description
  - Developed from HW design using C/C++ to system level
    - Easy to move from HDL environment

- **Trends**
  - Open SystemC Initiative (OSCI) promotes standardization
    - More than 70 of EDA vendors, IP vendors, semiconductor companies join to OSCI (Apr 2000)
SpecC

- **Background**
  - Developed by Prof. Gajski in UCI (1997)

- **Features**
  - *Unified description of HW/SW by enhancing ANSI-C grammar*
    - High readability
    - Needs special tools and environment
  - *Clear design methodology which covers over system modeling to architecture determination*

- **Trends**
  - SpecC Technology Open Consortium (STOC) promotes the standardization
  - System houses, embedded software tool vendors have strong interest
UML

- **Background**
  - Developed as modeling language for complicated large system
  - Started the standardization since 1996 by OMG *1
  - Unification of existing object-oriented modeling methods*2

- **Features**
  - **Visual language with high modeling ability**
    - Visual and multi-viewed system modeling
    - use case, class, component, layout, state chart, sequence, activity, collaboration
  - Scheme for describing constraints
  - Methodology of applying HW design is incomplete

- **Trends**
  - Wide use in business software
  - Started to use in embedded software
    - trial phase for HW design

---

*1 Object Modeling Group  *2 Booch Method, OMT Method, OOSE Method
## Evaluation Results

<table>
<thead>
<tr>
<th>Item</th>
<th>SystemC (v2.0)</th>
<th>SpecC (v1.0)</th>
<th>UML (v1.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>System Specification</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Constraints</td>
<td>not supported</td>
<td>not supported</td>
<td>visual description</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>constraint language</td>
</tr>
<tr>
<td><strong>Function Determination</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function Definition</td>
<td>less readability</td>
<td>clear methodology</td>
<td>only for SW</td>
</tr>
<tr>
<td>Function Verification</td>
<td>available C++ env.</td>
<td></td>
<td>only state chart</td>
</tr>
<tr>
<td><strong>Architecture Determination</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Profiling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block Partition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equivalence Check</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture Generation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture Mapping</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Interface to Implementation</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW/SW Co-Design</td>
<td>support until RTL</td>
<td>planned to support Accellera’s RTL model</td>
<td>not supported</td>
</tr>
<tr>
<td><strong>Test Bench Generation</strong></td>
<td></td>
<td></td>
<td>only specification</td>
</tr>
<tr>
<td><strong>IP Design</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary of System Level Design Languages

- SystemC
  - Representation from system level to HW/SW by adding dedicated class libraries and without enhancing C/C++ grammar
  - Need to clarify the methodology
  - Strong support for implementation, but need to consider the architecture generation and the estimation

- SpecC
  - Unified description of HW/SW by enhancing ANSI-C grammar
  - A key to prevalence is to arrange special design environment to implementation
  - Methodology for system modeling exists, but weak for the architecture generation and later stages

- UML
  - Large ability of function/constraint description by visual modeling
  - Need to consider design phases after architecture design