Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary


Outline

- System Design Methodology
- Specification Model Generation
- Bus Model Generation
- Implementation Model Generation
- Summary

**SpecC vs. SystemC**

**Execution sequence**

1. **Static scheduling**: In this case, the sequence of execution of behaviors is explicitly specified in the code. SpecC and SystemC support this approach.
2. **Dynamic scheduling**: SpecC uses a non-deterministic approach to schedule behaviors dynamically. SystemC has limited support for dynamic scheduling. When the system state transitions to execution, the behavior comes to execute until the next event occurs. The behavior is verified using a statement outputs.

**Execution sequence**

SystemC only supports dynamic scheduling of execution sequence. There are two main approaches:

1. **Static scheduling**: Developers use a static workflow to schedule processes. The order of execution is defined in the code.
2. **Dynamic scheduling**: The dynamic scheduling model allows processes to execute concurrently. However, in contrast to SpecC, SystemC can only execute processes through the `exec` function. SpecC allows for more flexible scheduling using the `exec` function.

**Data transfer**

Data transfer in SystemC is modeled using channels. The data transfer between processes in different modules is performed through channels. Channels are used to synchronize processes between different modules.
Specification Modeling (cont.)

1. SpecC uses a behavior, which is a consolidated representation for both structure and behavior. But in case of SystemC, there is a separation of the basic structural and behavioral entities. The structure is modeled using modules and behavior is modeled using processes. In summary, SpecC supports behavioral hierarchies which is not available in SystemC.

2. SpecC supports static scheduling while SystemC has to depend only on dynamic scheduling of execution sequence. Therefore, synchronization between concurrently executing processes in SystemC is complex and tedious to model. (e.g. B4 in Figure 1).

3. SpecC doesn’t support static sensitivity mechanism while SystemC does.

Architectural Exploration

Designers perform architecture exploration on the basis of following two factors:

- Complexity of functional blocks: The complexity of functional blocks is estimated by profiling the specification model.

- Execution sequence: The execution sequence of functional blocks is determined by analyzing the specification model.

Figure 12. Design example: architecture exploration result

Architectural Exploration (cont.)

Figure 13. Design example in SpecC: after PE allocation and behavior mapping

Figure 14. Design example in SpecC: after PE allocation and behavior mapping

Figure 15. Design example in SpecC: after PE allocation and behavior mapping

Figure 16. Design example in SystemC: after PE allocation and behavior mapping

Architectural Exploration (cont.)

Figure 17. Design example in SystemC: after step 1 of memory mapping

Figure 18. Design example in SystemC: after steps 2 and 3 of memory mapping
Architectural Exploration (cont.)

Figure 20. Design example in SystemC: after step 4 of memory mapping.

Figure 21. Design example in SystemC: after step 5 of memory mapping.

Architectural Exploration (cont.)

Figure 22. Design example in SystemC: after static scheduling.

Architectural Exploration (cont.)

Figure 23. Design example in SystemC: bus arbitration model.

Figure 24. Design example in SystemC: bus arbitration model.

Bus Model Exploration

Figure 25. Design example in SystemC: after step 5 of memory mapping.

Bus Model Exploration (cont.)

Figure 26. The timing diagram for the double handshake protocol.
Bus Model Exploration (cont.)

The decision rules of the communication exploration are implemented during the communication refinement. The communication refinement consists of two steps:

1. The chosen bus protocol is modeled at the bus-functional level.

Figure 27. Design example in SpecC after step 1 of communication refinement.

Figure 28. Design example in SystemC after step 1 of communication refinement.

Implementation Model Generation

1. Compile hardware models: The behavior description is synthesized into a netlist of layoutable hardware components.

2. Synthesize software: The software implementation is synthesized into a set of objects that are compiled into the processor instruction set, and linked against an RTOS.

Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>SpecC</th>
<th>SystemC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>Hardware includes executable model. Various versions of the executable model can be obtained from the SpecC model.</td>
<td>Hardware includes executable model. Various versions of the executable model can be obtained from the SystemC model.</td>
</tr>
<tr>
<td>Software</td>
<td>Software includes executable model. Various versions of the software can be obtained from the SpecC model.</td>
<td>Software includes executable model. Various versions of the software can be obtained from the SystemC model.</td>
</tr>
</tbody>
</table>

Table 2. Overall comparison in terms of exploration and refinement.