Overview

Intel CoFluent Studio can be used to model and simulate the behavior, timing requirements, architecture and performance estimates (throughput, latency, load, power, memory, cost) of any electronic system: HW IP, embedded SW application, mixed HW/SW multiprocessor system. Use cases of the system are modeled so the automatically generated transaction-level SystemC code can be used as verification testbench.

Behaviors are described with intuitive graphical notations and ANSI C/C++ code, although algorithms can be left undefined and abstracted to their sole execution time. Platforms are built by assembling generic models of universal components like processors, integrated circuits, memories, busses, interfaces. Each generic model provides variable design parameters to easily adjust its behavior and performance characteristics.

Intel® CoFluent™ Studio allows for behavioral and performance estimation without the need for embedded software application code, firmware, or a precise description of the platform with models of each component/IP core.

- No hardware IPs are needed
- No embedded software is needed
- No firmware / OS is needed
- No ISS is needed

Toolset

Intel® CoFluent™ Studio is an integrated Electronic-System-Level modeling and simulation environment composed of:

- A graphical modeler for capturing the system’s use cases, behavior and execution platform (the frontend)
- A simulation framework for automatically generating and instrumenting a transaction-level SystemC model of the complete system (including HW/SW partitioning) and interpreting simulation traces with a rich set of analysis tools
- A SystemC library (extending the Accellera SystemC 2 and TLM libraries) constituting the computation (multiprocessor, multicore, multitask/multithread) and communication (message queues, events, busses, interconnects, networks) simulation engine at the heart of Intel® CoFluent™ Studio
- A DML code generator for translating graphical models into fast functional models for the Wind River Simics® simulation platform
Systems
Intel® CoFluent™ Studio targets complex multiprocessor systems with rich application content, whether they are:

- Multi-board (large equipment, distributed/networked system)
- On-board (embedded system)
- On-chip (SoC, FPGA, ASIC, ASSP)

Packages

**Intel® CoFluent™ Studio for Timed-Behavioral Modeling**

- Targeted at system/specification engineers, application modeling/testbench developers and software engineers
- For modeling and simulating the behavior and time properties of electronic applications with simple graphics and C/C++ code
- For automatic generation of SystemC or DML timed-functional model
- For obtaining timed-executable specifications of the application for further system architecting or implementation

**Intel® CoFluent™ Studio for System Architecting**

- Targeted at system architects and software architects
- For exploring prospective system architectures (application-to-platform) and predicting their performance and power consumption
- For obtaining a virtual system (platform + application) description in transaction-level SystemC serving as executable specifications for further hardware/software design and implementation

Benefits

Intel CoFluent Studio allows developers of electronic systems and chips to:

**Decide**

| Problems | 80% cost-impacting decisions are taken in project's first 20% with few data  
Current processes can't cope with growing complexity and time-to-market  
Existing tools only help verify late in projects already made choices  
Cost to fix late design errors is prohibitive |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Solutions</td>
<td>Securely predict behavior and performance from partial software and hardware for design validation at all stages</td>
</tr>
</tbody>
</table>
| Advantages | Run short iterations for taking the right decisions at the right time  
Validate choices and detect errors earlier  
Optimize the system's architecture |
| Benefits | Mitigate risks of project delays and cancellations  
Reduce development time  
Decrease bill of materials |

**Share**

| Problems | Too large gap from specification to implementation  
Hardware & software engineers have no common understanding  
Difficult communications between multicultural and multidisciplinary teams across distributed sites  
Ambiguous specifications lead to misinterpretation and incompleteness  
Difficulties to define subsystems and manage subcontracted work |
|---|---|
Solutions
Provide system executable specifications rather than static documents for common hardware/software reference

Advantages
Facilitate implementation
Foster and organize teamwork
Ease interactions between all project stakeholders

Benefits
Reduce development time
Mitigate risks of project delays and cancellations

**Capitalize**

**Problems**
Little to no reuse of non-implementation work
Projects at risk if loss of key contributors
New projects can't benefit from past experience/expertise

**Solutions**
Capture projects' system design expertise in enterprise model libraries for easier and faster innovation

**Advantages**
Build a knowledge repository from project to corporate level
Reduce organizations’ dependency on individuals
Accelerate new and derivative/maintenance project inception

**Benefits**
Increase new product introduction rate
Mitigate risks of project delays and cancellations

<table>
<thead>
<tr>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported hosts</strong></td>
</tr>
<tr>
<td>Microsoft Windows OS</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux (for simulation only)</td>
</tr>
</tbody>
</table>

**Supported C++ development environments**

For Windows OS-based simulation: Microsoft Visual Studio*, MinGW GCC*

For all Linux*-based simulation: GNU GCC*

**Supported SystemC platform**

Accelera SystemC 2.2 & TLM 2.0
Synopsys
Cadence
Mentor Graphics