

Behavioral, Real-Time and Performance Analysis of Multicore Embedded Systems

CoFluent Studio™ is a visual embedded system modeling and simulation toolset based on Eclipse. Models are captured in graphical diagrams using CoFluent optimized domain-specific language (DSL) or standard UML notations – a combination of SysML and the MARTE profile. ANSI C or C++ is used as action language to capture data types and algorithms. Non-functional system requirements or model calibration data such as execution durations, power, or memory values, are added through model attributes.

Models are translated into transaction-level modeling (TLM) SystemC code for execution. The SystemC code is instrumented and generates traces that can be monitored with various analysis tools. Fast host-based simulations allow designers to observe the real-time execution of their models on multiprocessor/multicore platforms. Performance figures such as latencies, throughputs, buffer levels, resource loads, power consumption, memory footprint, and cost, can be extracted.

CoFluent Studio addresses two principal design challenges:	CoFluent Studio is available in two packages, addressing each challenge:
<ul style="list-style-type: none"> How to specify an embedded system and validate its specification How to predict and optimize performance of a multicore real-time embedded system 	<ul style="list-style-type: none"> CoFluent Studio for Timed-Behavioral Modeling (TBM) CoFluent Studio for System Architecting (SA)

CoFluent Reader™ is a free viewer and simulator/player for models that were created with CoFluent Studio. It allows designers to share models, simulation results, and observations, with project stakeholders that may not have access to CoFluent Studio. CoFluent Reader enables other engineers, managers, end-users, customers, and marketers, to view the model and its behavior. CoFluent Reader simplifies reading, understanding, and sharing, both models and simulations.

Timed-Behavioral Modeling

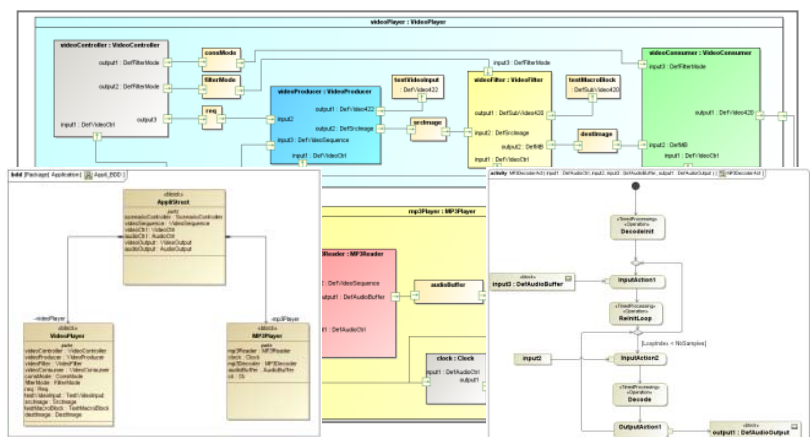
No one questions the importance of good specifications for developing embedded systems and chips. But once the system is specified, how can one be sure that the chosen design meets the system's functional and non-functional requirements? Creating models is a far better solution than simply writing documents and drawing figures. Executing models in order to verify and validate them is the ideal solution.

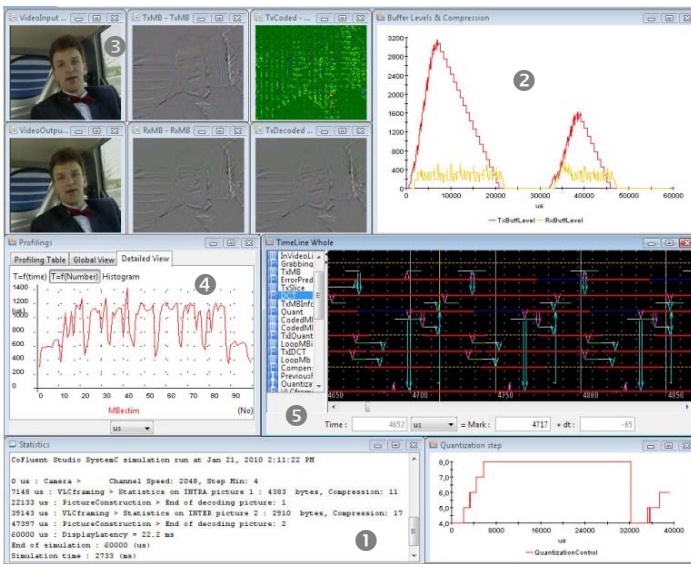
CoFluent Studio for Timed-Behavioral Modeling (TBM) allows capturing the design intent and system use cases in simple graphical models. The system and use case behavior is represented as a network of concurrent processes that act and interact to accomplish some logical end. Data and control flows between and for each process are described using graphical languages: CoFluent DSL or UML. ANSI C/C++ is used as action language.

TBM model representation	CoFluent DSL	UML
<i>Hierarchical structure</i>	CoFluent structural and behavioral diagram	SysML block definition diagram
<i>Data flow</i>		SysML internal block definition diagram + MARTE stereotypes
<i>Control flow</i>		SysML activity diagram + MARTE stereotypes
<i>Execution times</i>	<u>Declaration</u> : MARTE tagged values	
<i>Control statements</i>	<u>Definition</u> : ANSI C or C++	
<i>Algorithms</i>		
<i>Data types</i>		

Two types of modeling can be achieved:

- Behavioral** (also called statistical or token-based): Data types and algorithms can be left empty so that data communications and computations are abstracted to their sole execution time. This is useful for validating the parallel executions of processes and inter-process communications and synchronization in time.
- Functional**: Data types and algorithms correspond to the real data definition and processing done by the system. They can be defined in ANSI C or C++ or described with MATLAB or Simulink.





Diagrams are automatically translated in TLM SystemC code. The code is instrumented and built by a C++ compiler for host-based execution and monitoring. Analysis tools include:

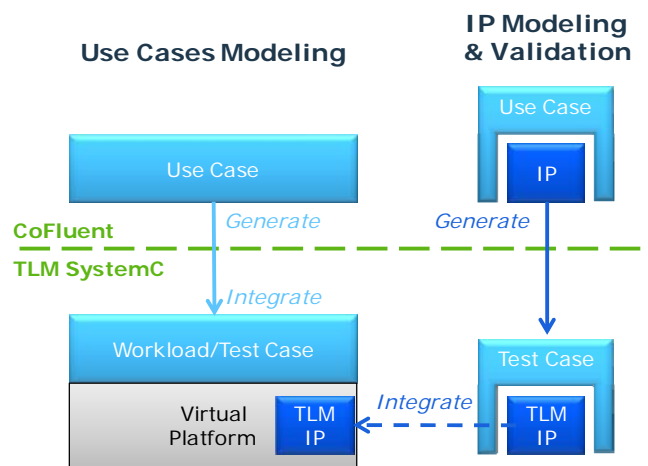
- Text printouts console. ①
- Plot charts: Variable in function of time, variable in function of another variable. ②
- Image viewing. ③
- Algorithms profiling: Measuring the execution times of algorithms (min, max, average), timing and counting the number of executions. ④
- Timeline: Displaying the parallel execution of processes in time with their state (running, blocked, idle) and the inter-process communications (data read/write, event get/set) – equivalent of UML sequence diagram with timings. ⑤

As the system use cases are captured, the resulting SystemC test cases can be reused in any SystemC-based simulation and verification environment for validating a lower-level, more detailed model or the final implementation. The graphical abstraction and automatic code generation provides an estimated 10x productivity improvement compared to manual coding, allowing for more complex and realistic test case creation.

This is especially useful for hardware systems:

- Complex use cases for SoC virtual platforms can be created and drive simulations through traffic generators or standard TLM-2 bus interfaces.
- Functional models of complex proprietary HW IP can be modeled and reused for integration into a TLM virtual platform, while the same test case can be used to validate the model and its RTL implementation.

By providing executable models of the system behavior and use cases, CoFluent Studio can be used for delivering executable specifications and validation test cases for any embedded system or chip.



CoFluent Studio for Timed-Behavioral Modeling Value Proposition

Facilitating transition from concept to design

- Creating realistic workload scenarios driving simulations
- Models enable experimentation and enhance innovation
- Simulations facilitate analysis and exchanges between teams
- Models support for patent applications

Increasing productivity

- Graphics are better suited to handle complexity
- Graphics are 10x more efficient than C/C++ programming

Optimizing design

- Pipeline, buffers, timings...

Facilitating implementation and validation

- Providing executable specifications for implementation
- Reusing test cases for validation
- Reusing model for integration to system simulation

Simulation Licenses & Modes

CoFluent Studio models can run on a variety of host machines and SystemC kernels. Each TBM or SA tool license comes with a bundled default simulation license for running simulations on Windows. The standard OSCI kernel and GCC compiler are shipped with the software product. Additional simulation licenses are available for a number of host OS (Windows or Linux) / SystemC kernel (OSCI, CoWare, Innovator or Qesta) / compiler family (GCC or Visual Studio) combinations. By default, simulation is run "live" from the CoFluent Studio GUI on Windows, whether locally (the SystemC executable runs on Windows) or remotely (the SystemC executable runs on Linux). Batch or standalone simulation is available (without CoFluent Studio GUI), as well as playing back simulation traces from CoFluent Studio (no actual simulation is running). This last mode is used by CoFluent Reader in its free version. Additional Reader licenses allow running simulations and changing parameters at runtime for design space exploration without model editing.

Tool License (Windows)	Simulation License	Live Local	Live Remote	Batch	Playback
Studio TBM/SA	Windows	✓		✓	✓
	Linux		✓	✓	✓
Free Reader					✓
Reader with Simulation	Windows	✓		✓	✓

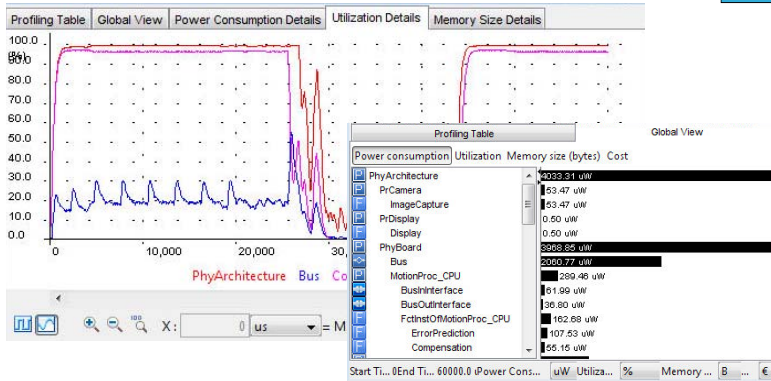
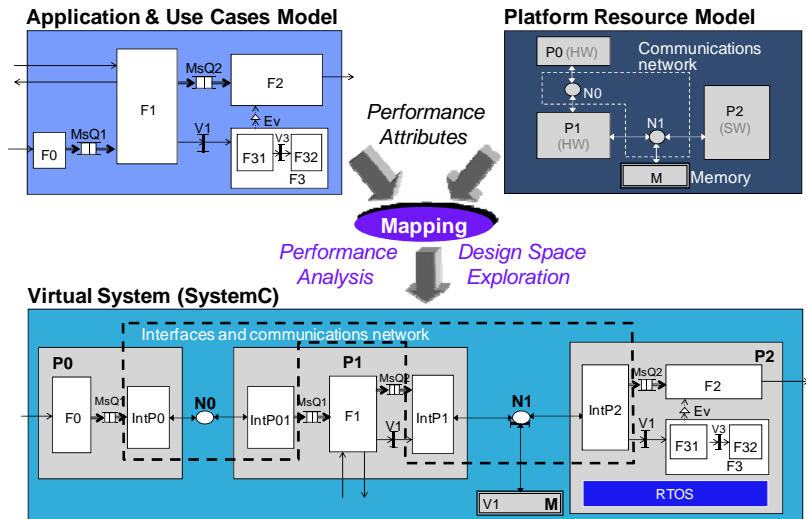
System Architecting

Multiprocessor/multicore embedded systems must address both present and future application demands for performance and low-power. In addition, as new challenges emerge, designers must also:

- Predict the behavior and performance of hardware and software components running on the different cores
- Evaluate required memory, power, bandwidth and computation resources
- Choose the right function for hardware acceleration
- Analyze the impact of asymmetric multiprocessor (AMP), supervised AMP (sAMP) or symmetric multiprocessor (SMP) architecture choices with multiple levels of hardware virtualization and scheduling layers – supervisor, hypervisor, real-time operating system (RTOS)
- Analyzing the software task scheduling and priorities
- Analyze the system communication topology: busses, networks and interconnects

An additional model refinement is necessary when modeling mixed hardware/software applications running on multiprocessor/multicore platforms with shared resources. The resources can include cores, busses, and shared memories. CoFluent Studio for System Architecting (SA) allows designers to describe their system computation, communication and storage units, and the application execution on the described platform.

First, designers create a static model of the system execution platform. Next they allocate processes of the TBM to cores, store shared data into memories, and route data on inter-processor physical links. This operation is called **mapping** or allocation. It is achieved by simple and interactive drag-and-drop editing. The resulting mapped or allocated model can be translated into SystemC and simulated in a similar way to the TBM.



In addition to TBM simulation results, SA simulation delivers:

- Richer timeline: Displaying processor communication interfaces, OS scheduling, and bus transactions.
- Performance profiling: Resource loads (in percentage or MCycles/s), power consumption, memory footprint, and cost values – min, max, average and dynamic profile against time –, are obtained for the system and each of its components.

CoFluent Studio can be used at any point of the project lifecycle for modeling and validating the real-time behavior of an embedded system or chip application and use cases. CoFluent Studio can predict performance data from the application and use cases model execution on a multicore/multiprocessor platform model.

CoFluent Studio for System Architecting Value Proposition

Improving productivity

- ➔ Simplifying multicore application decomposition
 - Processes and inter-process communications & synchronizations
- ➔ Graphical model = implementation blueprint
 - Guiding embedded designers
 - Providing execution time budgets for algorithms development

Reducing risks

- ➔ Avoiding potential real-time problems
 - Even before HW and SW availability
 - Deadlocks, race conditions, CPU starvation, etc.
- ➔ Reducing project cancellations and delays

Optimizing end-product

- ➔ Flexible architecture exploration free of any limitations
- ➔ Finding optimal architecture for parallel execution
 - Tasks allocation, CPU loads, communications topology
- ➔ Trade-off analysis between performance and other requirements
 - Memory footprint, power, cost

The allocated model is a **virtual system**, since it encompasses the description and simulation for the full hardware/software system. Unlike a TLM or cycle-accurate virtual platform or virtual prototyping model, the virtual system does not rely on the assembly of IP models to model the hardware, and instruction-set simulation (ISS) of the actual embedded software code and firmware (drivers, operating system, input/output libraries, etc.) to extract performance data.

CoFluent platform models are built from generic performance models of computing units, busses, network routers, point-to-point links, memories, DMA and I/O interfaces and schedulers. Parameters and calibration data are used to set the characteristics of each component. Hardware and software execution is simulated by computation and communication sequences determined by the TBM. Execution times are also set in the TBM. Such durations can be back-annotated from calibration data, measured by host-based profiling of the C/C++ code, or explored in a variation range. This enables describing models and obtaining simulation results with limited effort in a very short timeframe.

Virtualization Technology	Virtual System / CoFluent	Virtual Platform	Virtual Prototype
<i>Purpose</i>	Performance prediction	Early SW development	HW/SW co-verification
<i>SW/FW code</i>	Not required	Required	Required
<i>HW IP</i>	Not required	Required (TLM)	Required (RTL)
<i>Abstraction & accuracy</i>	Message-level – no ISS	TLM/CA + ISS	RTL + ISS
<i>Exploration & optimization factor</i>	High	Medium/low	Low
<i>Simulation Speed</i>	~1000x faster	~100x faster (TLM)	1x
<i>Modeling effort</i>	Low	Medium/high	High
<i>Performance prediction</i>	Few weeks	Few months	Project end

CoFluent Reader

CoFluent Reader allows CoFluent Studio users to freely share their models and simulation results and observations with all project stakeholders: end-users, customers, contractors, marketers, other engineers, managers.

CoFluent Reader is convenient. It can be freely downloaded, installed and used without time limits. Model information and results are exchanged in one single.cof file that can be easily attached to emails or placed on a FTP or HTTP server. It is safe since models are viewed in read-only mode. For further protection, the model creators select what features and information they want to share with others.

CoFluent Reader Value Proposition

Increasing productivity	<ul style="list-style-type: none"> ➤ Facilitating and accelerating communications and exchanges between project members ➤ Accelerating discussion cycles, avoiding tedious expert and white board sketch discussions ➤ Spreading the use of system-level thinking process to other individuals/teams
Mitigating risks	<ul style="list-style-type: none"> ➤ Establishing an unambiguous and common reference view and understanding of the problem and its solution between all project members and subcontractors

CoFluent Reader provides the same tools and environment as CoFluent Studio, but in read-only mode. In its free version, it allows playing back recorded simulation. Running simulations and varying parameters is possible with a dedicated simulation license.

CoFluent Reader comes with full documentation to guide newcomers.

Example models are available to give users an overview of the modeling and simulation capabilities offered by CoFluent Studio for different application cases.

Technical Specifications

Supported host machines	<i>Tools host OS</i>	Windows XP, Vista, 7
	<i>Simulation host OS</i>	Windows XP, Vista, 7 Linux RHEL 4, 5
Supported UML environments	<i>UML 2.2, SysML 1.1, MARTE 1.0</i>	No Magic MagicDraw 16.5, 16.6
Supported SystemC platforms	<i>SystemC 2.2 & TLM-2.0.1</i>	OSCI SystemC Library 2.2 CoWare 2009 Synopsys Innovator 2009 Mentor Graphics Questa 6.5
Supported C++ compilers	<i>Windows-based simulation</i>	Microsoft Visual Studio 2005, 2008 MinGW GCC 3.4, 4.1
	<i>Linux-based simulation</i>	GNU GCC 3.4, 4.1