Product Brief

Intel® CoFluent™ Studio

Behavioral, real-time, and performance analyses of software/hardware systems

Intel® CoFluent™ Studio is a modeling and simulation tool that offers a base for early tradeoff analysis. By providing executable models of system behavior and use cases, Intel CoFluent Studio can be used to deliver timed-executable specifications and validation test cases for any embedded system or chip, as well as performance predictions at any point in a project lifecycle.

OVERVIEW

Intel® CoFluent™ Studio (CoFluent) is a system modeling and simulation tool for performance predictions and early optimization of architecture and design.

Intel CoFluent Studio helps you innovate faster by making it easier to explore the design space and speed up "what if" analyses. It can help you reduce development costs by optimizing architecture for performance, and by maximizing resource utilization. With CoFluent, you can analyze system architecture early in the design cycle. In turn, this helps you minimizing design redo's and shorten time to market.

Since 2004, the CoFluent solution has been adopted in system-on-a-chip (SoC) and consumer electronic markets. The latest releases of CoFluent are now being successfully used to simulate end-to-end Internet of Things (IoT) systems and determine optimal dimensions for big-data clusters.

Intel CoFluent Studio can be used to model and simulate the behavior, timing requirements, architecture, and performance estimates of any electronic system. This includes performance estimates for throughput, latency, load, memory, and cost. For example, CoFluent can model and simulate systems such as IoT, hardware (HW) intellectual property (IP), embedded software (SW) applications, and mixed HW/SW multiprocessor systems. The resulting CoFluent models can be used for behavioral validation, executable specifications, performance analysis, and functional architecture.

Behaviors in CoFluent are described with intuitive graphical notations and C/C++ code. CoFluent also allows you to leave algorithms undefined and abstracted to their sole execution time. With Intel CoFluent Studio, you can estimate behavior and performance without software application code, firmware, or a precise description of the platform with models of each component/IP core.

MODELING FLOW

No one questions the importance of good specifications for developing embedded systems and chips. But once the system is specified, how can you be sure that the chosen design meets the system's functional and other requirements? A better solution is to create models instead of simply writing documents and drawing figures. The ideal solution? Create and execute models in order to verify and validate them.

Intel CoFluent Studio allows you to capture design intent and system use cases in simple graphical models. In these 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 processes, and the flows for each process individually are described using graphical language. C/C++/SystemC is used as an action language.

CoFluent allows you to perform two distinct types of modeling:

- Behavioral modeling. This is also called statistical or token-based modeling (see Figure 1). Behavioral modeling allows you to leave data types and algorithms 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 interprocess communications and synchronizations in time.

- Functional modeling. In functional modeling, data types and algorithms correspond to the real data definition and processing done by the system. These data types and algorithms can be defined in C/C++/SystemC or described with MATLAB.*

In addition, execution times for models can be back-annotated from calibration data. These execution times can be measured by host-based profiling of the C/C++ code, or explored in a

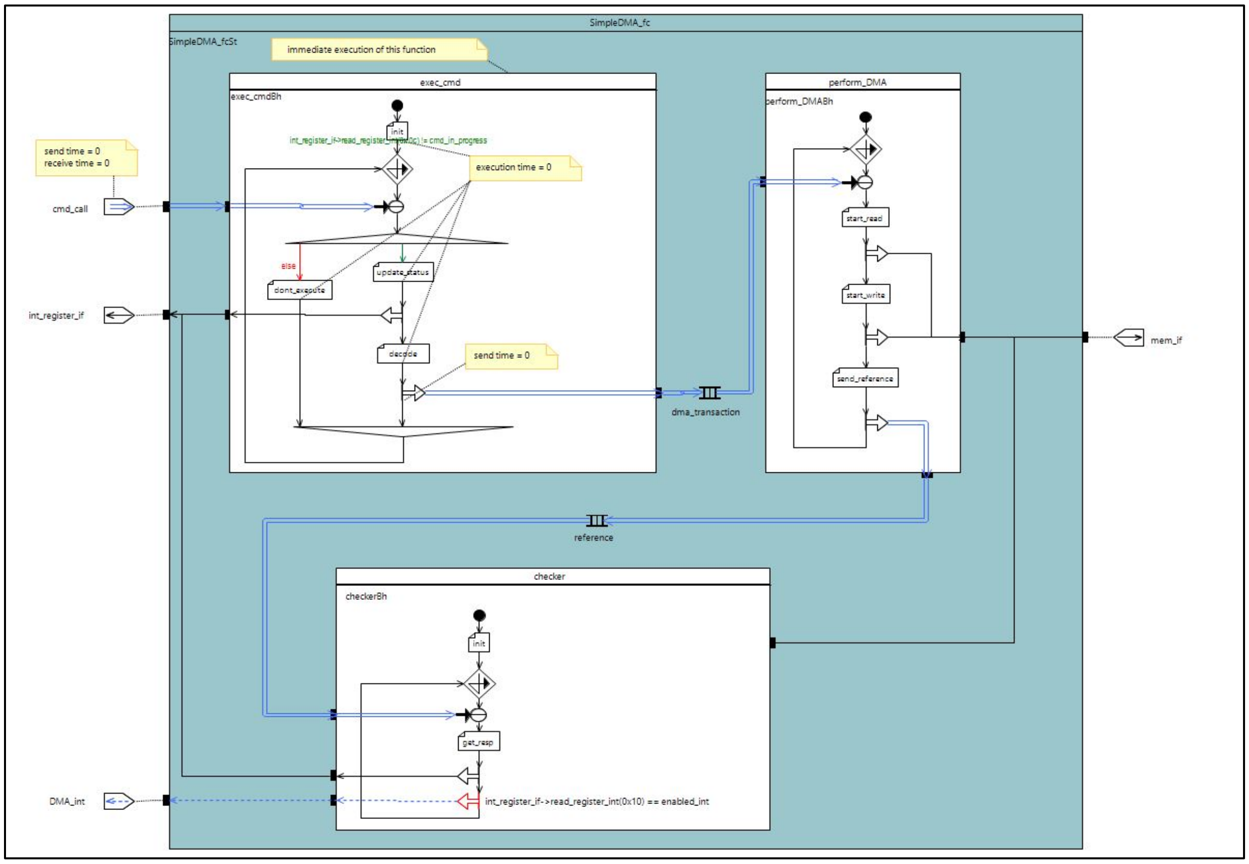

Figure 1. Behavioral modeling with Intel® CoFluent™ Studio

variation range. With CoFluent, you can describe models and obtain simulation results with limited effort in a very short timeframe.

As system use cases are captured, the resulting SystemC test cases can be reused in any SystemC-based simulation and verification environment. This then allows you to validate a lower level of the model, a more detailed model, or a final implementation. When compared to manual coding, CoFluent's graphical abstraction and automatic code generation provide a significant productivity improvement. In turn, this allows you to create more complex and realistic test cases.

CoFluent modeling is especially useful for hardware systems (see Figure 2):

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

By providing executable models of the system behavior and use cases, CoFluent can be used to deliver executable specifications and validation test cases for any embedded system or chip, as well as deliver performance predictions at any point of the project lifecycle.

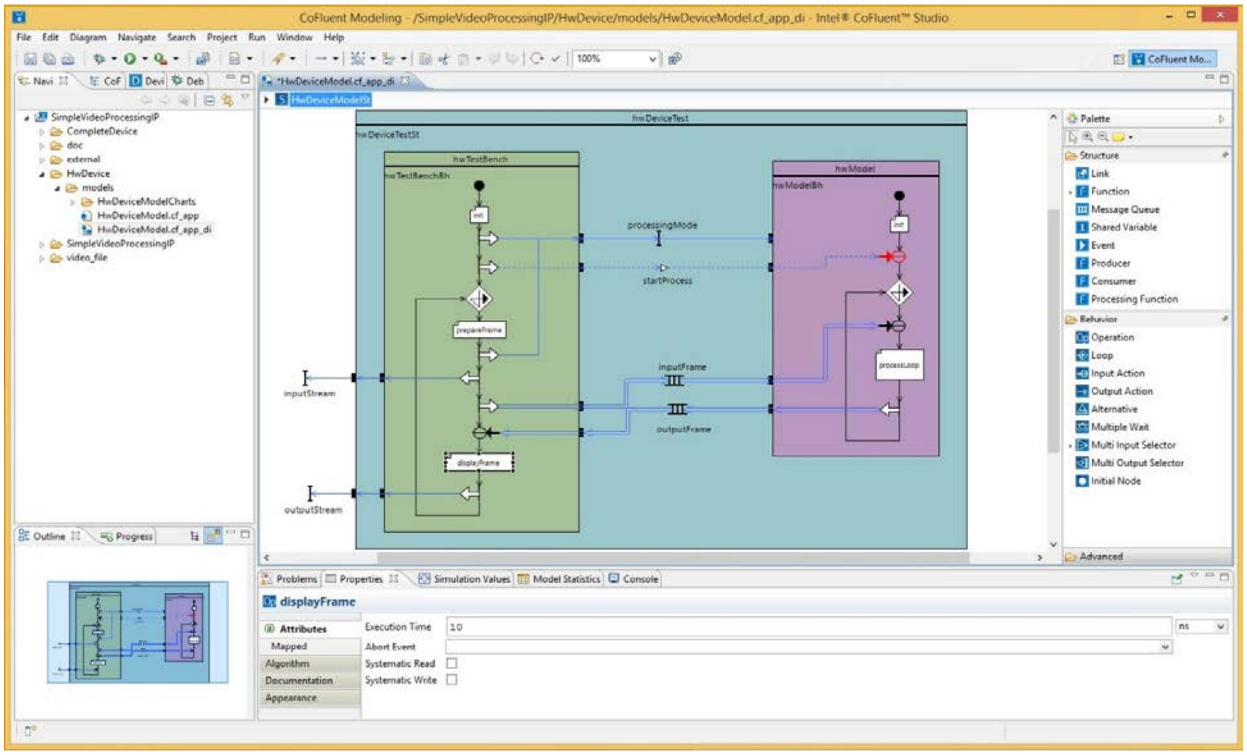

Figure 2. Example of modeling hardware with Intel® CoFluent™ Studio

TOOLSET

Intel CoFluent Studio is an integrated electronic system-level modeling and simulation environment based on Eclipse.* CoFluent consists of a graphical modeler and a simulation framework:

- **Graphical modeler** for capturing the system behavior (the front end) using CoFluent domain-specific language (DSL). SystemC, C, or C++ is used as action language to capture data types and algorithms. Nonfunctional system requirements or model calibration data (such as execution durations or memory values) are added through model attributes.
- **Simulation framework** for automatically generating and instrumenting a transaction-level SystemC model of the complete system and interpreting simulation traces with a rich set of analysis tools:
  - **Text printouts console**.
  - **Plot charts.** Variable in function of time, variable in function of another variable.
  - **Image viewing.**
  - **Algorithms profiling.** Measuring the execution times of algorithms, timing, and counting the number of executions. Algorithm execution times can be identified as minimum, maximum, and average times.
  - **Timeline.** Displaying the parallel execution of processes in time with their state (running, blocked, idle) and the interprocess communications — the equivalent of a UML sequence diagram with timings. Interprocess communications includes data read/write and event get/set.
  - **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 also allows you to extract performance figures, such as latencies, throughputs, buffer levels, resource loads, memory footprint, and cost:

- **A SystemC library,** which extends the Accellera SystemC 2* and TLM libraries. This library is the computation and communication simulation engine at the heart of Intel CoFluent Studio. Computations are performed for multiprocessors and multicore/multithread tasks.

Communications include message queues, events, and interconnects.

The all-in-one system application modeling environment allows you to:

- Model structure, control flow, data flow, and time properties all within the same model.
- Describe use cases, along with complete system functionality.
- Easily integrate external C/C++ or SystemC code/IPs.
- Export models to libraries, and control the import mode (black or white box) for reuse in other projects.

Thorough verification of behavior and timing constraints includes:

- Fully-timed model — duration of algorithms and communications.
- Fast simulation, based on transaction-level modeling (TLM) SystemC (message-passing).
- Automatic instrumentation, with a rich set of dynamic visualization and analysis tools.

You can prepare further architectural designs and implementations with:

- Timed-behavioral model that is directly usable for system architecting and early performance analysis.
- Functional architecture that is compatible with software (real-time operating system abstraction) and hardware implementation models.
- Automatic SystemC code generation for reuse in SystemC-compliant virtual platforms.
- Automatic DML code generation for reuse in Wind River Simics.*
- Automatic documentation generation.
- Software development kit (SDK), which allows you to create your own code generator, validation rules, and trace analyzer.

You can also facilitate interactions between all project stakeholders with:

- Generic models based on concurrent communicating functions (or processes), that represent system dynamics independently of technological considerations and physics.
- Unambiguous executable specifications that provide significant results for verifications.
- High-level graphical notations and simulation results that are more intuitive to project stakeholders.

READER

Intel® CoFluent™ Reader is a free viewer and simulator/player for models that were created using Intel CoFluent Studio. Intel CoFluent Reader allows designers to share models, behaviors, simulation results, and observations with project stakeholders that might not have access to Intel CoFluent Studio; such as other engineers, managers, end-users, customers, and marketers. With Intel CoFluent Reader, you can simplify reading, understanding, and sharing of both models and simulations.

SUMMARY

By providing timed-executable specifications, facilitating communications and offering a base for early tradeoff analysis, Intel CoFluent Studio and Intel CoFluent Reader contribute to bridging the gap in projects that range from specifications to implementations.

For more information about Intel CoFluent Studio, visit http://cofluent.intel.com