Data Sheet

SYNOPSYS®

CoCentric System Studio

Overview

CoCentric® is the family name for Synopsys' SystemC Design and Verification tool suite that spans from concept to implementation in hardware and software. A member of this family, CoCentric System Studio is a SystemC simulator and specification environment for joint verification and analysis of algorithms, architecture, hardware, and software at multiple levels of abstraction. The design flow for all design phases from concept engineering to implementation in hardware and software is vastly simplified using a single simulator. System Studio simulates hardware and software interaction at higher levels of abstraction with several orders of magnitude speed-up over RTL simulations. System Studio bridges the gap from abstract algorithms to synthesizable SystemC, making the creation and exchange of reusable IP based on SystemC easy. System Studio ships with over 2000 models for frequently used algorithms in its model library. Rapid standards compliance verification for the wireless telecommunications, multimedia, and computer networking sectors is made possible with a library of reference design kits. CoCentric System Studio enables verification of complex hardware/software (HW/SW) designs early in the design cycle, getting the specification right in a timely manner. It is also used to create synthesizable SystemC code for CoCentric SystemC Compiler, which provides hardware synthesis from SystemC.

The Problem

System-on-a-chip (SoC) technology marries processor cores, memories and peripherals together, creating unique products. However, SoC technology also presents unique design and verification challenges. Software plays a dominant role in the SoC solution, yet tools for designing hardware and software are disparate. Traditional RTL verification is no longer enough. Hardware elements may originate from different vendors, based on dissimilar tools. How can a design team hope to quickly design and verify an SoC with a high level of confidence?

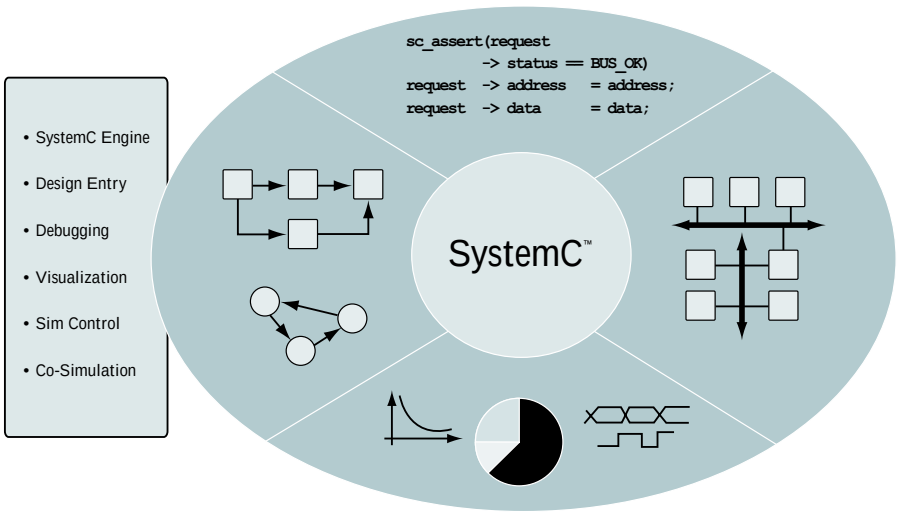

Figure 1. System Studio is a SystemC simulator and specification environment for joint verification and analysis of algorithms, architecture, hardware, and software at multiple levels of abstraction

The Solution: CoCentric System Studio

System Studio is a SystemC simulator and specification environment for joint verification and analysis of algorithms, architecture, hardware and software at multiple levels of abstraction. Based on C/C++, SystemC is the standard design and verification language that spans from concept engineering to implementation in hardware and software. As such, System Studio is the unified tool that effectively addresses the following important aspects of SoC design and verification:

- Design capture and management
- The verification of algorithms and of architectures
- Fast simulation
- Powerful debugging and analysis
- A path to implementation

Design Capture and Management

In System Studio, designers use hierarchical, graphical and language abstractions in order to capture and verify their entire system in one unified environment based on SystemC. It has a powerful integrated textual and graphical editor that enables users to capture algorithmic behavior and system architecture at various levels of abstraction.

SystemC designers, in addition to being able to work at the source code level, can now use System Studio's graphical editor to increase overview and visibility in the design. Pulling together models from design libraries is simply a drag-and-drop operation. Interconnections are easy to see. With the push of a button, System Studio performs formal checks on the design to identify common mistakes before compilation and simulation.

SystemC users are especially appreciative of System Studio because of the powerful dynamically changeable parameter features, the Tcl scripting and simulation control file capability, and the visualization and debug features. Another advantage of System Studio is the ability to visualize the design in several views (graphical schematic and symbol views, source code view, interface view and header view). System Studio maintains synchronicity between the views. It also facilitates design reuse by the automatic generation of model documentation in HTML format.

A very important aspect of any SoC design is the necessity to manage a wealth of design data. System Studio has a powerful and intuitive design management infrastructure including workspaces and library management. Designers can maintain multiple versions of a module as it evolves through the design process. System Studio offers revision control that is compatible with industry standard revision control software tools such as Rationale ClearCase®, GNU RCS and the Open Source CVS.

Design Capture and Management Highlights

- Capture
  - Schematic capture via block diagram editor
  - Built-in source code editor
  - Use of external source code editors (vi, GNU Emacs, …)
  - Model wizard, eases creation of SystemC models
  - Hierarchy browser
- Linked model views
  - Interface (all model types)
  - Source code (leaf level models and SystemC)
  - Header file (architectural models)
  - Schematic / Finite State Machine (hierarchical models)
  - Symbol (all model types)
- Design checks
  - Connectivity
  - Parameterization
  - Syntax
  - Absence of deadlocks (static DFGs)
  - Dataflow consistency (DFGs)
  - Causality (control models)
- Management
  - Automatic generation of HTML documentation
  - Organization of models in workspaces and libraries
  - Detection (+replacement) of out-of-date model instances
  - Configurable compiler and linker options
  - Automatic Makefile generation (models, libraries, simulation)
  - Configurable link to revision control systems
  - Search models by name, description, or category
  - Generate an executable specification from any level of abstraction

Verification of Algorithms

Algorithms are captured, verified and optimized in System Studio using intuitive dataflow graphs (DFGs) and finite state machines (FSMs). System Studio is a fast simulator of algorithmic control and dataflow systems. At the lowest level (e.g. leaf level), blocks of C/C++ and SystemC source code form a natural representation of the input and output behavior and the signal processing function that the block performs. Blocks are cleanly organized to form hierarchical structures by means of dataflow graphs and control models. System Studio is integrated with a powerful finite state machine editor that can be used to create hierarchical and concurrent finite state machines for modeling reactive control, dynamic switching, and concurrent execution. Both modeling styles (dataflow and control) work together and can be nested to any level of depth. For example, a designer can model a state machine in which each state is best modeled as a dataflow, and the algorithm contained within such a dataflow block can contain further dataflow or control models. The natural and intuitive hierarchical structure avoids errors, simplifies debugging and promotes the reuse and sharing of design blocks with other projects.

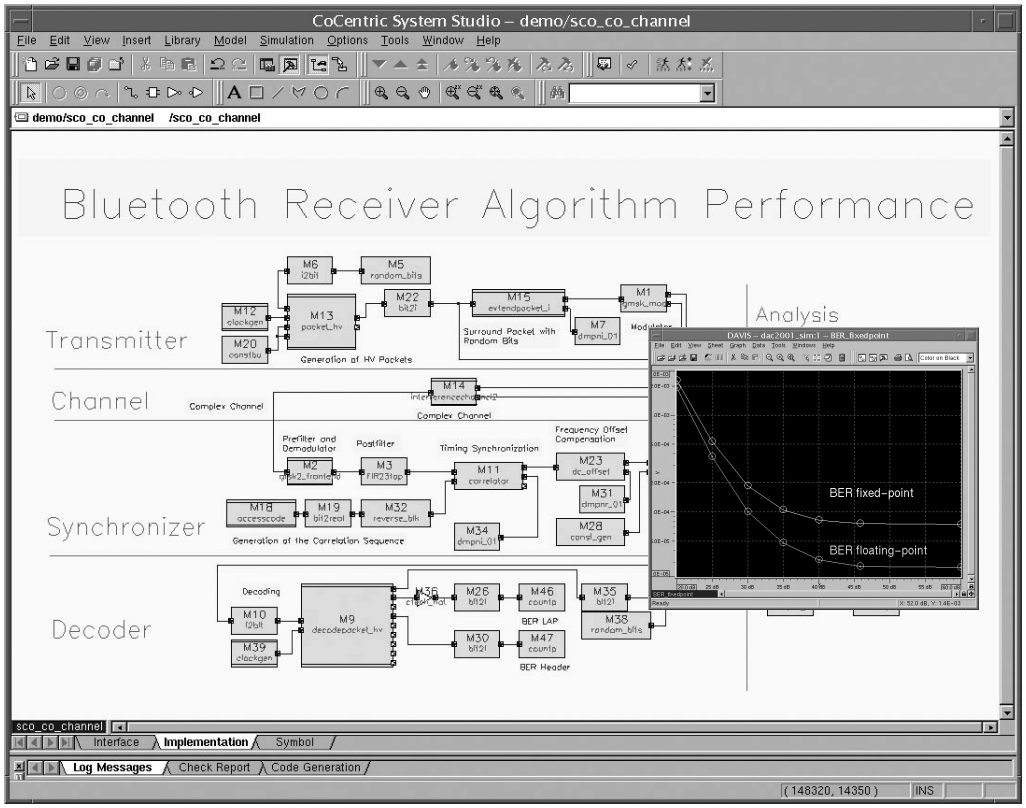

Figure 2. Algorithms are captured, verified and optimized in System Studio using intuitive dataflow graphs and finite state machines

System Studio supports the use of "type parameters". Type parameters enable users to create models that are independent of the data type (floating-point, fixed-point, bit-vectors, etc.) allowing the same block to be reused in a variety of designs. The benefit is a dramatic reduction in the number of library models to be maintained. The same model can be reused employing appropriate data types.

Using its unique "transparent modeling" capability, System Studio can parse the design and analyze it for various characteristics such as data rate consistency and static versus dynamic behavior.

Algorithm Verification Highlights

- Very easy and intuitive to use – can model even the most complicated algorithms very rapidly
- Arbitrary nesting of dataflow and control
  - Full control from dataflow
  - Control other models
    - Reset
    - Suspend/resume
    - Change parameters
- Different model types (PRIM, SDS, DFG, GATED, AND, OR)
- Multi-rate control models
- Static and dynamic dataflow
- Type parameters
- Different types of functional parameters
  - Dynamic parameters
  - Read-on-reset parameters
  - Structural parameters

Algorithm Libraries and Reference Design Kits

A vast library containing over 2000 algorithmic models is shipped with System Studio. Additionally, complete reference design kits (RDKs) are available for all major multimedia coding and wireless telecommunications standards, even including fully compliant receivers and channel models that are required, yet not specified in the standard. Time is saved and risk averted by checking conformance to standards very early in the design. Existing Synopsys COSSAP® models and designs can be reused freely as part of any new system definition. This completely protects existing COSSAP investments by enabling the reuse, as well as the modification of COSSAP models, libraries, and systems in the System Studio environment without any porting efforts.

Reference Design Kits and Algorithm Libraries

- Broadband Access
  - ADSL
  - DOCSIS cable modem
- 3G Wireless
  - cdma2000 1xRTT
  - WCDMA/FDD
  - EDGE
  - TD-SCDMA
- Other Wireless
  - Bluetooth™
  - GSM/GPRS
  - IS-136
  - cdmaOne
  - DECT
  - PDC
- Digital Video
  - MPEG-4 video
  - MPEG-2 video
- Broadcast standards
  - DVB
  - DAB
- Error Correction Coding
- Speech Coding
  - ITU G.72x speech
  - AMR speech
  - GSM speech

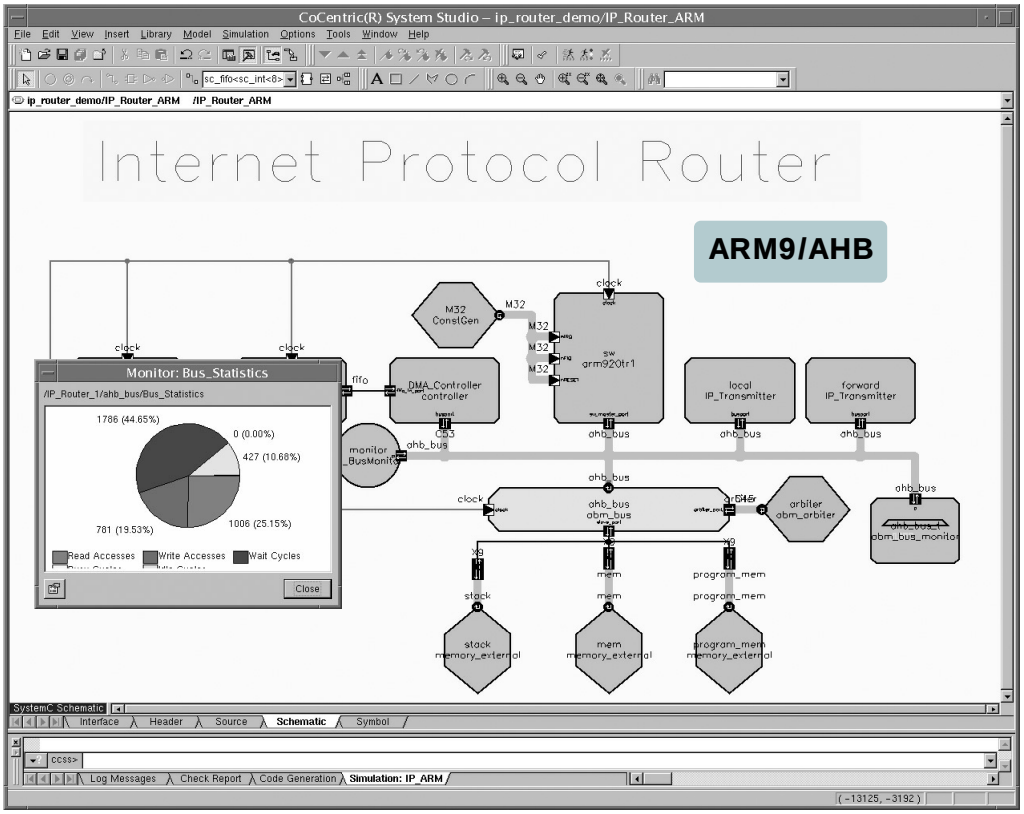

Figure 3. TLM in System Studio greatly benefits the design and verification architectures. Software developers can use the TLM-based model of the architecture to evaluate the throughput of the software and study the interaction of the software and the hardware models

Verification of Architectures

System Studio provides complete support for SystemC. Existing SystemC models can easily be used in System Studio. New SystemC models are quickly created using the model creation wizard and the context sensitive editor. Hierarchical blocks can be created graphically by drag and drop operations.

Full support of SystemC is one reason why System Studio is ideal for architecture modeling and exploration. An SoC architecture contains processing elements (CPUs, DSPs), interconnection elements (buses) and storage elements (memories, caches) in addition to other peripherals (address generators, multiply-accumulators, I/O). System Studio supports transaction-level modeling (TLM). TLM tremendously benefits the design and verification of architectures. As opposed to algorithm modeling, where communication is modeled as point-to-point between blocks, in architecture modeling, a finite set of resources transacts with each other over shared communication channels (e.g., buses). Using the TLM capability, it is possible to achieve significant simulation performance speedups compared to traditional RTL-based methods. System-level bottlenecks can be found and fixed much more productively at the transaction level as models can be developed easily and become available early in the design.

Software developers can use the TLM-based model of the architecture to evaluate the throughput of the software and study the interaction of the software and the hardware models. System Studio designers can create and import pin-level models, which can be simulated together with TLM models enabling the verification of synthesizable models in a high-speed system context.

Architecture Verification Highlights

- Full support of SystemC at all levels of abstraction
- User-defined SystemC channels
- Parametizable models
  - Template parameters
  - Custom constructors
  - Dynamic parameters
    - Change during runtime
    - Models can detect data changes (event notification) and react
    - Change from simulation control file
- Use existing SystemC code developed with the OSCI reference simulator
- Export SystemC models
- Architecture performance analysis early in design
- Executable TLM architecture models for joint HW/SW design and verification
- Pin-accurate models (e.g., for HW synthesis)
- Mix and match models of different abstraction levels
- Integrate algorithmic models simply via drag-n-drop
  - Model SoC environment
  - Stimuli generation
  - Data post-processing
  - Use timed or untimed algorithmic models as placeholder for still-to-be designed components

Simulation

System Studio has a simulation engine, which automatically evaluates the different models being used (dataflow, FSM, SystemC) and optimizes the simulation of the total system through different compile and scheduling techniques. This powerful functionality enables the user to freely mix and match static and dynamic algorithm models along with architectural models and FSMs, while the tool optimizes the simulation execution. It benefits users by allowing them to focus on design and verification

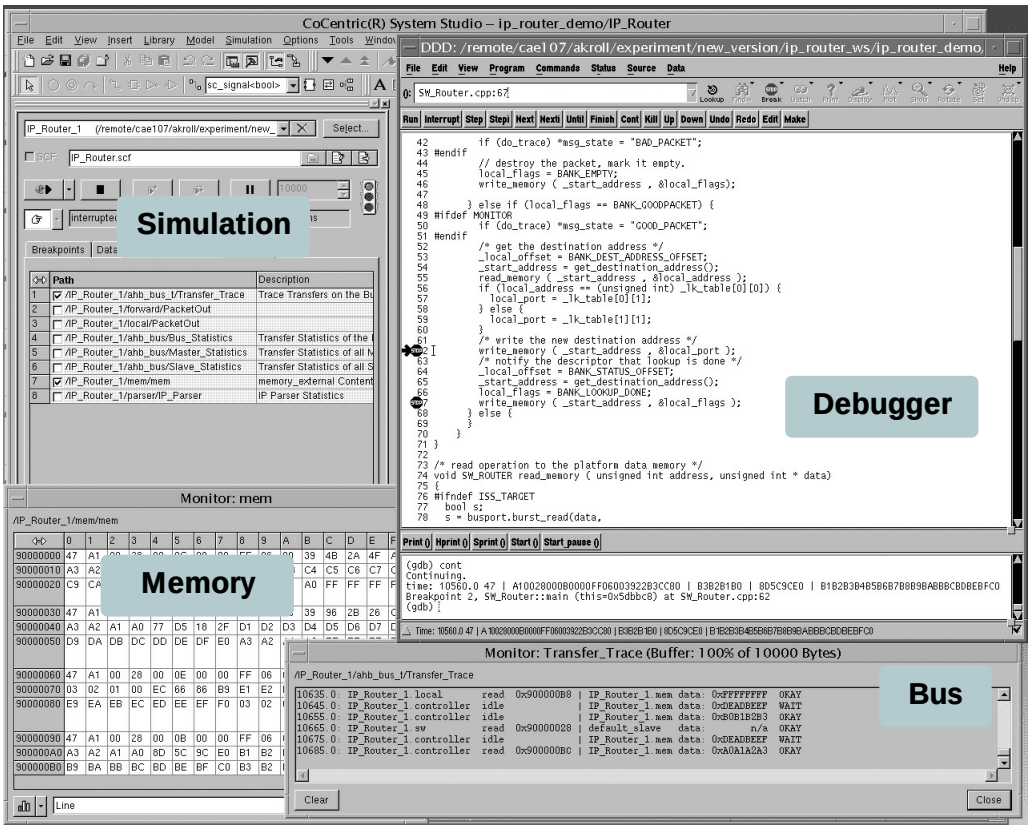

Figure 4. The debugging and analysis capability in System Studio provides designers with all the feedback they need for design productivity

instead spending efforts to port models between different abstraction levels or tools. Mixing of architectural and algorithmic models in the same simulation ensures a mixed-mode capability that enables software designers to simulate software in a hardware context.

CoCentric System Studio works seamlessly with leading HDL simulators such as Synopsys VCS™ and Scirocco™, Cadence® Verilog-XL® and Mentor Graphics® ModelSim® to co-simulate RTL blocks written in VHDL or Verilog. Users of Matlab® from The Mathworks, Inc. can import their models into the System Studio environment for co-simulation. System Studio models can be integrated into the ITUZ.100 Specification and Description Language (SDL) context by making use of System Studio's slavable simulation capability. For example, imagine a software engineer wishes to verify protocol stack software using the Telelogic Tau® SDL Suite to simulate the system from an SDL point of view. Operation of the baseband, simulated by System Studio, automatically communicates bi-directionally with SDL Suite. This unlocks the opportunity for early verification of the full complement of HW/SW across multiple ISO layers.

During simulation, an interactive control panel gives users control over the execution of the simulation. A designer may interactively set breakpoints at any node in the design and single-step through simulation sequences of interest. Once the simulation is paused, the designer has access to variable values, can view the internal states and parameters of models, see values of nets connecting models, or examine the active state of any control model. In short, System Studio offers complete visibility into the design.

System Studio simulations can be controlled via simulation control files (SCF). Based on the widely used Tcl language, an SCF can be used to automatically sweep a range of parameter values for critical blocks in the design. It allows polling (or setting) model parameter values to change simulation scenarios based on intermediate simulation result values, thereby enabling automatic iterative optimization of parameters. This allows converging on a desired output result more rapidly than by using simple iterative simulation over a range of parameter values. In addition, SCF permits the distribution of simulations over a network of workstations using the popular Platform LSF® load sharing facility from Platform Computing, Inc. and Sun™ Grid Engine from Sun Microsystems, Inc.

An incremental compilation option ensures that the simulation setup time is optimized. When this option is invoked, simulation code is generated and compiled only on the sections of the design that have been modified. This is useful in a development environment where designers change minor sections of the design and require a rapid turnaround in simulation results.

Simulation Highlights

- Simulation engine
  - Fastest control/dataflow simulator
    - Compiled statically scheduled code
    - Dynamic dataflow
    - Slavable simulations
  - SystemC simulation
    - Fully compatible with OSCI reference implementation
  - Mix and match models of different levels of abstraction
- Co-Simulation
  - VCS, Scirocco
  - ModelSim
  - Verilog-XL
  - Matlab

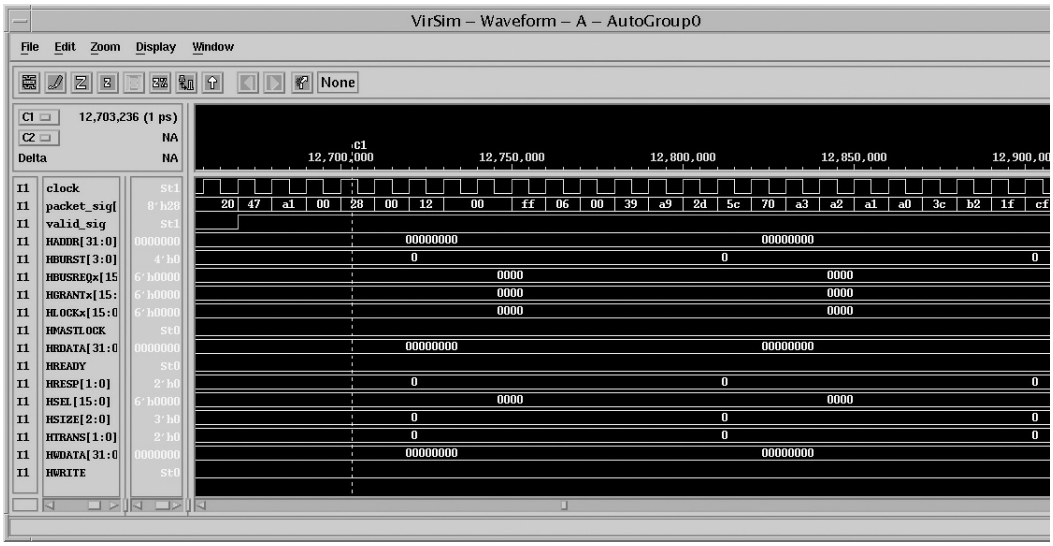

Figure 5. System Studio works seamlessly with leading HDL simulators. Signal transitions can be analyzed interactively, or written to a file and post-processed

- Slavable simulation enables integration of algorithmic models into
  - Telelogic SDL Suite environment
  - HDL context
  - User-proprietary tools
- Tcl simulation control files
  - Regression testing
  - Parameter optimization
  - Load sharing (Platform/SUN)
- Integration of processor models into architectural and algorithmic models
- Incremental compilation (short modify-compile-simulate cycles)
- Optional run-time checks (range checks)
- Simulation reports (activation counts, # samples, etc.)
- Orders of magnitude faster than RTL
- Cycle-accurate hardware simulation

Debugging and Analysis

The debugging and analysis capability in System Studio provides designers with all the feedback they need for design productivity. Designers can choose from numerous options to portray or graph the data in a manner that is most helpful for analysis. Multiple levels of debugging are available: the macro-debugging environment, which is extremely efficient for block level debugging, and micro-debugging, which can be used to debug detailed block implementation at the C++ source-code level.

The macro-debugger enables designers to debug designs at the block level. The tool allows users to set breakpoints when a given block is executed or when a given state in an FSM is entered. This provides for efficiency in the debugging process since the overheads are very minimal.

Once a problematic block is identified, System Studio allows the designer to dive into the source code of that block at the push of a button. Source code debugging, also called “micro-debugging” is performed using standard source-code debuggers. Special print and display functions aid the designer in tracking down bugs.

Signal transitions can be analyzed interactively, or written to a file and post-processed, using either the integrated DAVIS or VirSim data visualization environments. DAVIS provides several capabilities including creation of signal graphs and complex signal calculation on sets of graphs in both the analog and the digital world. VirSim is a waveform viewer for viewing contents of VCD data files at the end of a simulation. VirSim can also steer the simulation and interactively display SystemC signals and parameters. On-demand data tracing ensures the highest possible simulation speed.

While signal transitions are essential to understand the behavior of an individual block, it is difficult to analyze system-level parameters with signal transitions. For example, it is very difficult to study bus utilization with signal transitions. Therefore, System Studio provides designers with unique data monitoring capabilities such as message and table monitors. These monitors display aggregate data using several displays such as pie charts, tables and histograms.

Debugging and Analysis Highlights

- Macro-debugging (e.g., at the block level)
  - Start/stop/pause simulation
  - Run for number of cycles, number of model activations, specified amount of time (architectural models)
  - Step-into (dive into source code; architectural models only)
  - Breakpoints - stop when model is activated or suspended
  - Highlighting of active model instance in schematic
  - Highlighting of state transitions
  - Peek/poke (signals/nets, parameters)
  - Online statistics of model activations and data flow
  - Hierarchy browser of data/parameters (level watch)
  - Create custom data collection (data watch)

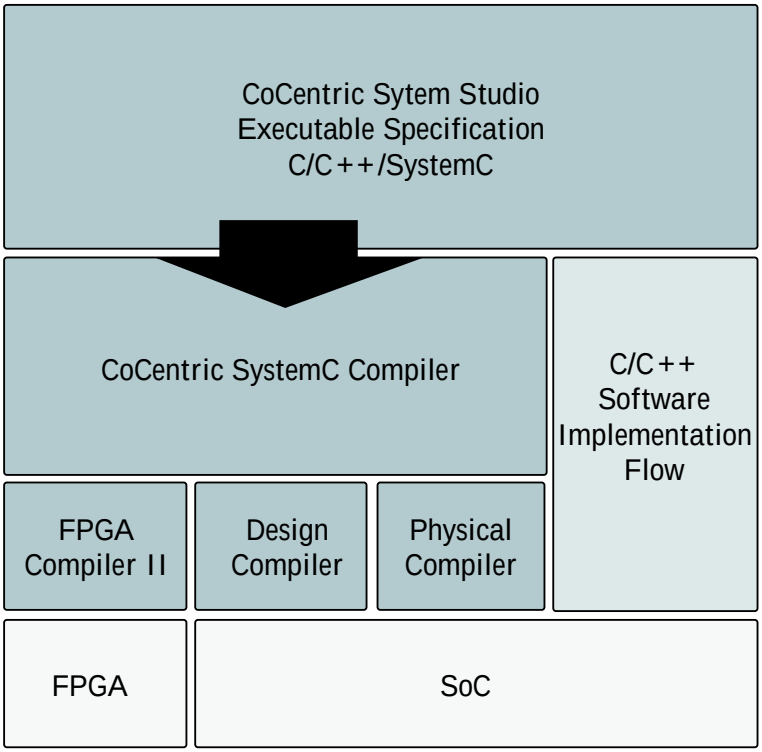

Figure 6. System Studio simulates any SystemC model at any abstraction level. The companion product, CoCentric SystemC Compiler, closes the gap from system design to gates by providing HW synthesis from SystemC. Quality of Results (QoR) is identical to using a traditional RTL starting point but has the additional advantage of early and more complete HW/SW verification

- Micro-debugging (e.g., at the source code level)
  - Print and display values of ports and signals
  - Print and display SystemC data types
- Interactive and post-simulation waveform displays
  - DAVIS
    - Waveforms
    - Calculator
    - Eye diagram
    - Scatter diagram
    - Histogram
    - Analog + digital
    - Timed and untimed
  - VirSim
    - Standard HDL waveform viewer
    - Simulation control
    - Signals and parameters
  - Both DAVIS and VirSim can be connected interactively
    - Data is only recorded upon request
    - No penalty if data is not recorded
    - High simulation speed
- Data monitors
  - Easy-to-use model instrumentation; on-demand data visualization
  - Analyze bus traffic
  - Peek at memories
  - Visualize statistics
  - Present status information
  - Table monitor
    - Table representation
    - Graphical representation (pie, bar, Gantt, …)
  - Message monitor
  - Data monitor

Path to Implementation

System Studio enables the fastest route to implementation of hardware blocks right from the system level of abstraction. For unit-rate algorithms specified using data flow diagrams, FSMs and leaf level C-based models, System Studio can automatically generate synthesizable SystemC code. This synthesizable SystemC code output is then input into CoCentric SystemC Compiler. SystemC Compiler accepts behavioral and RTL SystemC descriptions and generates hardware using state-of-the-art synthesis technology from Synopsys. This automatic path to hardware boosts design team productivity by enabling the quick capture of a design and the subsequent rapid generation of prototype FPGA or ASIC hardware.

Path to Implementation Highlights

- Design synthesizable SystemC models
  - RTL coding style
  - Behavioral coding style
- SystemC code generation
  - Automatic synthesis of hardware
  - Generated directly from unit-rate algorithmic models
    - PRIM models
    - DFGs
    - Control models
    - Nested dataflow and control
  - Flattened or hierarchical implementation
  - Generation of scripts to aid synthesis using SystemC Compiler and Design Compiler™
  - Powerful FSM optimization
- Verification of generated hardware in the system context

SYNOPSYS®

700 East Middlefield Road, Mountain View, CA 94043 www.synopsys.com