With software development becoming the fastest growing component of NRE costs for both SoC and final product development, the challenges of developing, integrating, validating, and optimizing software in the context of hardware dominates the embedded design process. Thus it has become a necessity to make a fast, accurate, low-cost simulation model of the hardware available early to the embedded software team.

Part of the Mentor Vista™ platform, Vista Virtual Prototyping provides an early, abstract functional model of the hardware to software engineers even before the hardware design is implemented in RTL. It can run software on embedded processor models at speeds par with physical hardware boards; yet it provides additional capabilities and benefits, such as the visibility and control to debug complex software/hardware interactions and optimize the software to meet final product performance and power goals.

Platform Creation

Transaction-level modeling (TLM) is an abstracted approach to modeling functional units of digital systems, whereby the details of communication among units are separated from the implementation details of the units. A TLM platform is composed of interconnected transaction-level models represented by SystemC structural code. Vista contains a library of pre-defined TLMs and allows users to customize key attributes (such as timing and power) in these models. Vista also enables users to create their own TLMs or import external models and add these to the Vista model library.

TLM Platform Creation

The Vista Block Diagram editor provides a simple way to create TLM platform SystemC structural code and link graphical symbols of individual TLMs to each other, thus defining the topology of the design. Upon each save operation, the Block Diagram editor automatically generates the structural code corresponding to the schematic view. The Block Diagram editor supports a wide range of operations, including selecting individual elements or groups; defining attributes and default names for graphic elements; resizing, moving, and copying elements; and editing symbols in either a symbol editor or the Block Diagram editor. It also allows users to rotate, delete, and duplicate graphics; add and edit text; and add TLM symbols. Block Diagram editor functions are readily accessed from menus, toolbars, and sidebars, providing flexibility and ease-of-use for TLM platform creators.
Virtual Prototype Creation

A virtual prototype is a stand-alone executable derived from a TLM platform. Virtual prototypes are created either by using the virtual prototype dialog in the Vista GUI or by using the vistaa_create_vp command. Vista allows users to define the Vista parameters file, which contains the parameters of the design that can be modified during runtime (without recreating the executable). Users can also set the values of the default runtime environment variables used during virtual prototype simulation and copy additional directories and files into the virtual prototype runtime environment. Vista virtual prototypes can be generated to run on either Linux or Windows workstations. An optional license agreement can be attached to the virtual prototype to allow the platform creator to control who uses the virtual prototype.

Benefits of Platform Creation

■ Predefined library of TLMs
■ TLM customizable attributes, such as timing and power
■ Can create and add own TLMs to library
■ Intuitive graphical platform assembly editing and visualization
■ Automated creation of virtual prototype executable
■ Virtual prototype runs on Linux or Windows workstations

Simulation Control and HW/SW Co-Debug

Simulation Control

The Vista virtual prototype can be invoked at the command line or controlled from CodeBench Virtual Edition based on the Sourcery CodeBench IDE environment. In command line mode, the virtual prototype runtime variables can be passed as command line arguments. When controlled from CodeBench Virtual Edition, it can provide visibility into the hardware, tight hardware/software debugging, and file system interactions. These capabilities are provided through a plug-in into Sourcery CodeBench that interacts with the virtual prototype API.

Software Development

Vista Virtual Prototyping enables software development, integration and validation with hardware by providing in the virtual prototype equivalent capabilities to those provided in prototypes and physical boards. It supports running the actual software that will be later run on the final product. Vista Virtual Prototyping allows running the software consistently and deterministically. It supports UI, application stacks, middleware, firmware, and drivers running on top of operating systems such as Linux, Android and Nucleus, as well as bare-metal mode. It provides the facilities and the methodology for building Linux kernels and for booting the operating system in a matter of seconds.

Vista virtual prototypes can be linked with physical devices such as terminals and displays on the host workstation, allowing the software engineers to control the virtual prototype. It also allows using the host workstation’s USB and Ethernet connections to run the virtual prototype under realistic environment conditions. Furthermore, Vista offers the ability to run Graphic applications (such as OpenGL and QT) where they are bridged and executed on the GPU of the host. Using semi-hosting it allows the user to print messages to the screen, print current simulation time and set error message verbosity. It also provides visibility and profiling facilities non-intrusively, such that these do not change the behavior of the prototype and have only a minor effect on its performance.

Hardware Timing Modes

Vista Virtual Prototyping can run the hardware in two modes. The functional mode supports integration, validation, and debugging the software. The performance mode allows analyzing and optimizing the software to achieve better performance and reduced power consumption. To enable these modes, the virtual prototype is modeled at two levels of timing detail: loosely timed (LT) and approximately timed (AT).

In functional mode, the virtual prototype uses a loosely timed abstraction level where the transactions represent a complete data transfer across a hardware bus, independent of both how the transfer actually occurs and the time it consumes. In this
mode simulation speed is in the range of hundreds of MIPS, equal or close to real-time speed.

In performance mode, the virtual prototype uses an approximately timed abstraction level in which transactions represent the phases of data transfer in a specific bus protocol (for example the address and data phases of an AHB read or write). The increased accuracy of performance mode does cause the virtual prototype to simulate at about two orders of magnitude slower than functional mode.

Vista also supports higher levels of timing accuracy and performance using the Accuracy-Tunable functional mode. In some applications, the accuracy of this mode can reach that of performance mode but with substantially higher simulation speeds. Simulating in this mode allows the user to concentrate on parts of the system where the timing accuracy is needed while letting the rest of the system run in functional mode.

Vista Virtual Prototyping allows users to select and switch between timing modes during runtime. For most upper-level software, such as the application and OS levels, the functional mode will suffice to validate and debug the software. But for performance and power analysis of hardware under software control and for lower-level software (such as drivers and real-time software), running the virtual prototype in performance mode or in Accuracy-Tunable functional simulation mode is required.

**Hardware Visibility**

CodeBench Virtual Edition Debugger is connected to Vista processor models via GDB server remote protocol. It allows the user to choose the cores the debugger is connected to. Vista Virtual Prototyping also provides the CodeBench Virtual Edition with direct visibility and control to the hardware objects in the platform. These hardware objects include peripheral registers and local variables as declared by the TLM component creator. The objects are designated by their hierarchical design path, and their values are shown in separate sub-trees on the CodeBench Register view. Each of the object values is editable, and new values can be set during debugging.

**Tightly Coupled HW/SW Debugging**

The user can conduct tight hardware-software debugging by setting breakpoints in the hardware that stops the hardware simulation once the breakpoint condition occurs. The simulation can then be resumed such that it stops the software debugger after completing the instruction which triggered the breakpoint. This allows the user to inspect the state of the software at that point and then step through the following software instructions while viewing the state of the hardware objects resulting from the execution of each software instruction.

**Hardware Simulation Control**

Hardware simulation can be controlled using a set of *gdb* monitor (*mon*) commands available from the Sourcery CodeBench console. Using these commands the user can reset the CPU core; load image files into memory; set, query, and delete a breakpoint; set or display the hardware timing mode; display information about the simulation process; and terminate the simulation process.

In addition, Vista Virtual Prototyping allows customizing the CPU core by providing pre-defined callbacks on significant events, and it provides a simulation control API that supports semi-hosting and other useful tasks. It also enables manipulating files from the embedded operating system prompt that is booted on the virtual prototype. For example, files can be copied from the host file system to a local directory on the virtual prototype target (and vice versa) using the Linux cp command issued from the embedded Linux OS prompt.

**Benefits of Simulation Control and HW/SW Co-Debug**

- Simulation control from CodeBench Virtual Edition
- Enables switching between functional and performance mode during run time
- Provides visibility and control to the hardware object from CodeBench Virtual Edition
- Stops simulation on hardware or software breakpoints
- Provides simulation and software control commands
- Supports host file manipulation for the embedded operating system

**Software and Hardware Analysis**

CodeBench Virtual Edition provides a rich set of analysis capabilities for the embedded software and the underlying modeled hardware. The analysis results are provided through analysis viewing windows and analysis summary reports.

**Software Analysis**

Software analysis on the Vista virtual prototype is conducted using the Sourcery Analyzer tool contained in CodeBench Virtual Edition. The Analyzer works with Vista virtual prototypes, integrates data from single and multi-core CPUs, and supports Linux, RTOS, or bare-metal modes. It includes built-
in analysis agents: a library of popular and intuitive system analysis and visualization tools that address the most common views desired by software engineers for evaluating the impact of their software and CPU/core operation on the final product functionality, performance, and power.

Sourcery Analyzer agents enable users to view CPU states and statistics, file system activity over time, function calls and statistics, latency caps, lock wait and hold times, and process and thread states. CodeBench Virtual Edition also contains built-in VP software analysis agents the enable viewing:

- **VP Current Function** — The current function running on a given cluster over time.
- **VP Function Call Activity** — Software function activity over time.

Sourcery Analyzer also allows the user to write custom agents. Written in Java, these provide the means to conduct application specific analysis and visualization, and improve and optimize design performance. Sourcery Analyzer provides users with access to the complete API that the built-in analysis agents use. It also provides a wizard to facilitate the creation of new analysis agents.

**Coverage and Function Reports**

CodeBench Virtual Edition can generate profiling and line coverage reports and places them under a Reports subnode of the SW node. When you click on the lines in the coverage report, it shows the corresponding code.

**Hardware Analysis**

CodeBench Virtual Edition also contains built-in VP hardware analysis agents that enables viewing the following key hardware design attributes:

- **VP Throughput** — The activity level of selected objects measured by the number of transactions or number of bytes within a specified time interval.
- **VP Latency** — The average time within a specified time interval it takes to complete a specific transaction type.
- **VP Power** — Dynamic power, static (leakage), and clock tree power are shown for the entire design and for selected instances in the design.
- **VP Attributes** — Values of attributes derived from user-defined variables specified in the TLM model over time.
- **Cache Hit/Miss Ratio Analysis** — Provides the cache hit and miss ratios for CPU cores that include Icache and Dcache components. These ratios can be further divided into a cache for READs and a cache for WRITEs.
- **VP Transactions** — Shows the transactions of the given module sockets. This allows the user to correlate the data being communicated with the software functions involved in producing or consuming the data.

The Vista Virtual Prototyping allows the user to control the zoom level and the start and end endpoints of the viewed information. It allows comparing multiple simulation sessions to determine the effects of system configuration changes and software changes on the design behavior and on its performance and power attributes.

**Benefits of Software and Hardware Analysis**

- Supports Linux, RTOS, or bare-metal modes
- Provides key analysis views of the CPU states, function calls, and processes
- Provides hardware power views under software control
- Provides performance views of the hardware including throughput and latency
- Enables contention, arbitration, and hit/miss cache ratios analysis
- Provides comprehensive software profiling and line coverage reports

Visit our website at www.mentor.com/esl/vista/virtual-prototyping for more information.