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 necessary to validate software against a fast, accurate, low-cost hardware simulation model as early as possible.

Platform Description

The Altera® Arria® V virtual prototype kit (VPK) was created using the Mentor Vista™ tool. It provides an abstract functional model of the Altera Arria V SoC to embedded software engineers before the hardware design is implemented in RTL. It runs software on embedded processor models at speeds on par with the FPGA; 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 goals. The Arria V virtual platform is compliant with the Altera Arria V Hard Processor System Technical Reference Manual (TRM), Revision: av_5v4 February 28, 2014. The block diagram of the entire platform is shown above. The highlighted blocks are the implemented models in the virtual platform.

The main modules in the platform are:

- **Processor** — An ARM® Cortex®-A9 MPCore processor
- **Buses** — ARM L3 NIC-301 interconnect bus, and five APB buses responsible for connecting various modules
- **Memory Modules** — Various memory modules
- **Devices** — Different models have been developed specifically for the Arria V platform including the GIC, SCU, L2Cache, GPIO, UART, I2C, EMAC, DDRC, SPI, QSPI, SDHC, SYSMGR, RSTMGR, DMA, FPGAMGR, TIMER, FPGA2HPS, HPS2FPGA, and the LWHPS2FPGA device

**KEY BENEFITS**

- Validation of software against early hardware model
- Visibility of key hardware registers and attributes
- Fast software execution speed
- Advanced software performance analysis
- Support for Altera® Linux Yocto kernel and bare-metal applications
- Non-intrusive trace, profiling, and coverage
- Intuitive software debug environment based on Sourcery™ CodeBench
- Embedded software analysis based on Sourcery™ Analyzer
- Easy to integrate and test early FPGA prototype along with the Arria V virtual platform
Simulation Control

The Altera Arria V virtual prototype kit is provided as an executable that can be invoked at the command line or provided as an Eclipse plugin that can be installed and controlled from CodeBench Virtual Edition (which is based on the Sourcery CodeBench IDE environment). At the command line mode, the virtual prototype can be configured by adding or modifying the parameters in the adjunct parameters.txt file. When controlled from CodeBench Virtual Edition, it provides visibility into the hardware, tight hardware/software debug, and file system interactions.

Software Development

The Altera Arria V virtual prototype supports software development, integration, and validation by providing equivalent capabilities to those provided in prototypes and physical boards. It allows developers to run the actual software that will run on the final product. Altera Arria V runs the Linux OS application software consistently and deterministically as well as in bare-metal applications. It provides the facilities and methodology for downloading and installing the Altera Linux Yocto kernel as well as editing the kernel configuration. Once Linux has been installed, booting the operating system is done in a matter of seconds.

The Altera Arria V virtual prototype manual provides procedures for:

- Booting using U-Boot
- Booting using Linux
- Booting using Linux with Minimum Root FileSystem
- Booting from QSPI Flash Memory
- Booting from SD Card

The Altera Arria V virtual prototype includes additional bare-metal applications:

- A bare-metal factorial application runs on the Altera Arria V virtual prototype executable
- A stopwatch application example shows how to run, debug, and analyze an application in CodeBench Virtual Edition

Hardware Timing Modes

The Altera Arria V virtual prototype 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. 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 runs at the LT 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 runs at the AT 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 functional-type 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.

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

**Hardware Visibility**

CodeBench Virtual Edition Debugger is connected to the Altera Arria V virtual prototype processor models via a GDB server remote protocol. It allows the user to choose the cores the debugger is connected to. It 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 transaction-level modeling (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 debug.

**Tightly Coupled HW/SW Debug**

Users can conduct tight hardware-software debug 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 that triggered the breakpoint. This allows users 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 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.

**Software and Hardware Analysis**

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

**Software Analysis**

Altera Arria V virtual prototype software analysis is conducted using the Sourcery Analyzer tool contained in CodeBench Virtual Edition. The Analyzer integrates data from single and multi-core CPUs and supports Linux, RTOS, and bare-metal programs. 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 functionality and performance of the final product.

The 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 virtual prototype (VP) software analysis agents that enable viewing:

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

**Function Calls Statistics**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Time with Children</th>
<th>Time without Children</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>3.220990255 s</td>
<td>3.00002764 s</td>
</tr>
<tr>
<td>UART0_Receive_Bye</td>
<td>228.941628 ms</td>
<td>228.941628 ms</td>
</tr>
<tr>
<td>UART0_Send_Bye</td>
<td>17.234 µs</td>
<td>17.234 µs</td>
</tr>
<tr>
<td>UART0_Send_String</td>
<td>31.057 µs</td>
<td>15.090 µs</td>
</tr>
<tr>
<td>UART0_Send_Decimal</td>
<td>2.020 µs</td>
<td>920.0 ns</td>
</tr>
<tr>
<td>_cct_start</td>
<td>3.250000005 s</td>
<td>654.0 s</td>
</tr>
<tr>
<td>UART0_Receive_Decimal</td>
<td>16.8040031 ms</td>
<td>368.0 s</td>
</tr>
<tr>
<td>_cct_preamble</td>
<td>3.220990425 s</td>
<td>355.0 s</td>
</tr>
<tr>
<td>Display_Time</td>
<td>9.400 µs</td>
<td>258.0 s</td>
</tr>
<tr>
<td>UART0_init</td>
<td>17.0 ns</td>
<td>27.0 ns</td>
</tr>
<tr>
<td>ssdtimer1_start</td>
<td>64.0 ns</td>
<td>64.0 ns</td>
</tr>
<tr>
<td>Display_Start_Messages</td>
<td>11.130 µs</td>
<td>48.0 ns</td>
</tr>
</tbody>
</table>

*Function call statistics on the Altera Arria V virtual prototype*
The Analyzer also allows users to write custom agents. Written in Java, these provide the means to conduct application specific analysis and visualization, and they improve and optimize design performance. The 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 be configured to generate code coverage and profiling reports, which can be imported into the Analyzer for inspection. The reports are generated in HTML format for greater portability.

Hardware Analysis
CodeBench Virtual Edition also contains special built-in VP hardware analysis agents that enable 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 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

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

Customizing Arria V Virtual Platform in Vista
Vista now supports the Arria V virtual prototype as a set of self-extracting files that can be installed either locally or under the Vista install root. If installed under the Vista root, the installer recognizes the platform self-extracting file and creates a platform directory automatically. Once installed under VISTA_ROOT, the platform can be simulated as-is without needing to copy or build anything.

Once the platform is installed in Vista, Vista allows users to customize the design by modifying the block diagram of the design and adding TLMs representing the custom blocks *(requires a separate Vista Architect license)*. Once simulated and validated, users can create a virtual prototype executable representing the modified platform that can be used, debugged, and analyzed with CodeBench Virtual Edition.

---

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