Hybrid Approach to Testbench and Software Driven Verification on Emulation

Debdutta Bhattacharya  Ayub Khan
Mentor, A Siemens Business

### Hybrid Approach to Testbench and SW Driven Verification

This setup enables software development for SoC blocks in a modular fashion wherein parts of the testbench responsible for configuration and providing stimulus to a particular sub-block can be swapped with corresponding software application running on the virtual machine once ready.

#### Concurrent HW and SW Debug

A virtual software based verification flow on emulation enables –
- Full capabilities of SW stack in pre-silicon.
- Complex software can be written and run in Guest OS running in QEMU, e.g. DMA.
- Simultaneous verification of hardware and software by enabling the full debug capabilities of a hardware emulator. E.g. –
  - Waveform upload and viewing
  - Enabling triggers to capture bugs
  - Streaming live waveforms during tests
- Software debug using gdb or kernel messages in QEMU

#### DMA in Testbench Based Verification

Certain software features can be very complex to model in testbenches. One such example is Direct Memory Access (DMA) -
- DMA enables CPU concurrency and boosts overall system performance by handling large memory operations through a DMA engine
- DMA-engines are often multi-threaded, handling multiple descriptors to write to or read large chunks of memory at a time using queues

DMA applications running on a software stack which handle complex DMA transactions are difficult to model in testbench based verification.

#### Software Driven Verification in Emulation

Software verification in pre-silicon is enabled in hardware emulation through the use of virtual CPU emulators like Quick EMulator (QEMU) or VirtualBox or using a physical machine in ICE along with an emulator. In a virtual setup, the platform consists of –
- Guest OS on CPU emulator like QEMU
- QEMU running on a host network connected to the hardware emulator
- Synthesised DUT on hardware emulator
- Communication via protocol like PCIe using comodelling infrastructure

#### Testbench Driven Verification in Emulation

Virtual testbench driven verification for SoCs can be done on hardware emulation in various ways, including running a virtual Testbench written in C/C++, SystemC/UVM as a standalone executable or on a simulator on a comodel server connected to the emulator. Testbenches can also be synthesized along with DUT and run on the emulator.

Virtual testbench based verification provides flexibility in terms of –
- Running a variety of tests by changing testbench stimulus
- Altering stimulus to find specific bugs
- Flexible debug in terms of stopping clocks, checking or setting register values etc.
- Uploading waves for entire DUT for debug based on signal triggers or time points

Transaction based communication between a Virtual Testbench and hardware emulator often provides 50x-1000x speedup as compared to simulation verification flows.

#### Blockwise Shift From Testbench to SW

This approach has the following advantages –
- Software driver and application development with real RTL starts early, months in advance to post-silicon validation
- Eliminates need of modelling complex software scenarios like DMA in testbenchs. Re-use of software blocks saves time and effort spent in modelling these in testbenches
- Bugs are flushed out early by enabling early software validation
- Improved corner-case testing is made possible with real software configurations
- Overall project cycle is shorter with software in better shape in pre-silicon verification

#### Concurrent HW and SW Debug

A hybrid method for testbench based and software based verification in emulation is proposed. One part of the setup consists of single or multiple threads of a testbench communicating to a synthesised DUT running on the emulator. The second part of the setup consists of a virtual machine like QEMU, running a Guest OS. Both parts communicate among each other via APIs and use the same protocol for SoC configuration.

For software development, this enables –
- Virtual environment for driver and application development
- Application testing capabilities across OS
- Lab-like SW debug flows

QEMU along with the testbench allows running software applications which are complex and time-consuming to model in testbench based verification, eg. multi-threaded DMA applications.

#### Software Driven Verification in Emulation

Software in pre-silicon is enabled in hardware emulation through the use of virtual CPU emulators like Quick EMulator (QEMU) or VirtualBox or using a physical machine in ICE along with an emulator. In a virtual setup, the platform consists of –
- Guest OS on CPU emulator like QEMU
- QEMU running on a host network connected to the hardware emulator
- Synthesised DUT on hardware emulator
- Communication via protocol like PCIe using comodelling infrastructure

#### DMA in Testbench Based Verification

Certain software features can be very complex to model in testbenches. One such example is Direct Memory Access (DMA) -
- DMA enables CPU concurrency and boosts overall system performance by handling large memory operations through a DMA engine
- DMA-engines are often multi-threaded, handling multiple descriptors to write to or read large chunks of memory at a time using queues

DMA applications running on a software stack which handle complex DMA transactions are difficult to model in testbench based verification.

#### Testbench Driven Verification in Emulation

Virtual testbench driven verification for SoCs can be done on hardware emulation in various ways, including running a virtual Testbench written in C/C++, SystemC/UVM as a standalone executable or on a simulator on a comodel server connected to the emulator. Testbenches can also be synthesized along with DUT and run on the emulator.

Virtual testbench based verification provides flexibility in terms of –
- Running a variety of tests by changing testbench stimulus
- Altering stimulus to find specific bugs
- Flexible debug in terms of stopping clocks, checking or setting register values etc.
- Uploading waves for entire DUT for debug based on signal triggers or time points

Transaction based communication between a Virtual Testbench and hardware emulator often provides 50x-1000x speedup as compared to simulation verification flows.

#### Testbench Driven Verification in Emulation

Software in pre-silicon is enabled in hardware emulation through the use of virtual CPU emulators like Quick EMulator (QEMU) or VirtualBox or using a physical machine in ICE along with an emulator. In a virtual setup, the platform consists of –
- Guest OS on CPU emulator like QEMU
- QEMU running on a host network connected to the hardware emulator
- Synthesised DUT on hardware emulator
- Communication via protocol like PCIe using comodelling infrastructure

#### DMA in Testbench Based Verification

Certain software features can be very complex to model in testbenches. One such example is Direct Memory Access (DMA) -
- DMA enables CPU concurrency and boosts overall system performance by handling large memory operations through a DMA engine
- DMA-engines are often multi-threaded, handling multiple descriptors to write to or read large chunks of memory at a time using queues

DMA applications running on a software stack which handle complex DMA transactions are difficult to model in testbench based verification.

#### Testbench Driven Verification in Emulation

Virtual testbench driven verification for SoCs can be done on hardware emulation in various ways, including running a virtual Testbench written in C/C++, SystemC/UVM as a standalone executable or on a simulator on a comodel server connected to the emulator. Testbenches can also be synthesized along with DUT and run on the emulator.

Virtual testbench based verification provides flexibility in terms of –
- Running a variety of tests by changing testbench stimulus
- Altering stimulus to find specific bugs
- Flexible debug in terms of stopping clocks, checking or setting register values etc.
- Uploading waves for entire DUT for debug based on signal triggers or time points

Transaction based communication between a Virtual Testbench and hardware emulator often provides 50x-1000x speedup as compared to simulation verification flows.

#### Testbench Driven Verification in Emulation

Virtual testbench driven verification for SoCs can be done on hardware emulation in various ways, including running a virtual Testbench written in C/C++, SystemC/UVM as a standalone executable or on a simulator on a comodel server connected to the emulator. Testbenches can also be synthesized along with DUT and run on the emulator.

Virtual testbench based verification provides flexibility in terms of –
- Running a variety of tests by changing testbench stimulus
- Altering stimulus to find specific bugs
- Flexible debug in terms of stopping clocks, checking or setting register values etc.
- Uploading waves for entire DUT for debug based on signal triggers or time points

Transaction based communication between a Virtual Testbench and hardware emulator often provides 50x-1000x speedup as compared to simulation verification flows.