Verification of a PowerPC Multiprocessor Design

As embedded system designs become more complex, new methodologies are needed to verify both hardware and software. The use of multiple processors has migrated from workstation design to embedded systems. This article outlines a set of methods that can be used to verify both hardware and software for shared-memory multiprocessor embedded system designs. The article is based on the MPC7400 microprocessor, and the use of Simpod’s DeskPOD product to provide the primary verification vehicle.

Hardware Verification

Traditionally, hardware engineers have used bus functional models (BFM) to verify interfaces between microprocessors and custom hardware such as ASICs and FPGAs. Bus models were adequate when there was only one processor in the design and the bus protocol was simple enough to write an accurate BFM in a matter of weeks. Two things have changed the landscape of hardware verification. First, processor busses have become more complex. No longer are simple memory reads and writes enough to model a bus. Features such as deep pipelining, out-of-order transaction completion, and complex arbitration schemes make writing a BFM a task that takes months instead of weeks. Add to this protocol complexity the use of write-back caching and cache snooping, and the concept of the master-only BFM breaks down. In multiprocessor shared-memory architecture interactions between processors and other bus masters become very important. This limits the usefulness of the bus functional model, since BFMs are only capable of generating master transactions on the bus. DeskPOD uses actual device silicon to provide a model of the entire microprocessor. While full-functional models are useful for system simulation and initialization software debugging, they are not usually embraced for ASIC verification because of the necessary overhead for hardware engineers who prefer not to write software and learn to use cross-compiler tools. To meet the challenges of providing a model with the necessary accuracy and functionality, Simpod has added a BFM interface that uses the device model and achieves the necessary hardware verification without all of the overhead associated with full-functional models. DeskPOD is used in 3 distinct modes to achieve the verification goals. The following sections outline the use of DeskPOD for a hypothetical design that contains multiple MPC7400 microprocessors in a symmetric multiprocessor (SMP) configuration as shown in Figure 1. There are three main verification goals: verify the ASIC logic when the processor is acting as a master, verify the master operation of the ASIC when the processors are acting as snoopers, and verifying the initialization software and system diagnostics.

Single CPU Environment

A single CPU environment is used to generate PowerPC master transactions on the MAX bus for the purpose of verifying the functionality of slaves on the bus. ASIC logic in the form of a memory controller or register set can be verified very quickly.
DeskPOD provides a verilog task interface to the MPC7400 model that allows MPX master transactions to be generated on the bus by the CPU. A summary of the transactions is given below. Full details of the interface are given in the Simpod Bus Functional Model Interface specification.

```verilog
$spbfm_pcc_write(`SP_PATH,address,tt,tsiz,tbst,data,data1,data2,data3,status);
$spbfm_pcc_read(`SP_PATH,ad,tt,tsiz,tbst,wait_mode,status);
$spbfm_pcc_read_result(`SP_PATH,ad,result_data,status);
$spbfm_pcc_address_only(`SP_PATH,address,tt_attr,wait_mode,status);
```

- This interface is not much different than any bus functional model interface. It is used from a testbench to generate bus transactions. What is different is that the MPC7400 device is used to generate the transactions. The technology to accomplish this described below.

### Technology

The Simpod technology that enables the BFM interface to run at very high performance levels is outlined below. The BFM interface looks no different than any other BFM and runs at about the same speed.

1. **DeskPOD Memory Models** are used to store a small C program (including RAM, ROM, and STACK) which runs local to the MPC7400 microprocessor on DeskPOD. There is no code fetching or any other memory accesses to/from verilog memories except that specified by the user testbench.

2. **Simulation Disengage** is used to detach DeskPOD from the verilog simulation during code fetching and other uninteresting bus activity. This allows DeskPOD to run in the tens of kilohertz range instead of the tens of hertz range that verilog runs at. DeskPOD will engage with verilog when a bus transaction starts (TS_active) with an address that is interesting to the user testbench.

3. A verilog TS filter is also used to mask pipelined transactions from being seen by the other agents on the shared bus. Because of the deep bus pipelining, deciding when to disengage during an interesting transaction without reflecting a partial bus transaction to the system is difficult. There is rarely a time when the MPX bus is idle and a safe disengage can be done. The TS_filter addresses this problem by not presenting the TS_signal to the system during the data tenure of an interesting testbench transaction. These transactions are no interesting to the testbench and there is no need for the system to see or react to them.

4. Stub to Verilog communication is done via verilog registers. Special addresses are used by the stub to communicate status to the user testbench. The stub can write to these addresses and the verilog model can set a register to indicate status to the testbench. Currently, 3 such registers are used to indicate the stub is ready to receive testbench commands, a testbench command has been completed, and the results of a read command are ready.

To facilitate the fastest performance and best use of DeskPOD’s Simulation Disengage feature, the following restrictions are imposed for the single CPU environment:

- **BG** to the MPC7411 must be always driven active to from the system arbiter. Since arbitration takes place before the address of the transaction is known, it would be impossible to use address tenure arbitration from verilog.
- There is no cache snooping available. This is a consequence of the previous restriction. When **BG** is always active, no other masters can use the MPX bus.

DeskPOD provides a BFM solution that is equivalent to the software BFM for the hardware engineer to write a verilog testbench and generate CPU bus master activity.
Multiple Master Environment

The BFM interface is sufficient for verifying ASIC slave logic, but to verify ASIC master activity additional functionality must be provided beyond what is available in a BFM. A multi-master environment has the additional requirements of arbitration and cache snooping. Unlike the single CPU setup, the system arbiter must be used (and verified) to provide the BG_ and the DBG_ signals to each master on the MPX bus for every address and data tenure. The primary goal of multi-master verification is to verify the arbitration and the cache snooping interactions. Each master must react properly for the different snoop conditions and intervention protocols. To create a specific set of snoop interactions the DeskPOD BFM Interface provides a testbench command to set a PowerPC cache line to a known state. After using this command to setup the cache, a master transaction can be run on the bus and the interaction between the ASIC and the PowerPC cache can be observed and verified. The task is summarized below:

```
$spbfm_ppc_magic_bean(`SP_PATH, address, state, data, status);
```

To facilitate good performance and still meet the goals of arbitration verification and snooping, the Simulation Disengage feature is used less aggressively. The simulation runs in disengage mode until it is ready to run transactions. This allows the testbench to start immediately since the processor initialization is almost instantaneous in disengage mode. But now the simulation runs in the engage mode when the testbench starts. This is the best way to achieve adequate performance and centralized arbitration. The use of engage mode places one restriction on the simulation environment. Transactions will now appear on the bus for the stub fetches and stack accesses. This requires the user design to provide appropriate BG_, AACK_, DBG_ and TA_ responses for these transactions that were not specified from the testbench. If the address ranges used by the stub are the same as those of the system ROM or Flash memory, this can be transparent and require no additional work for the user. If there is no way for the ASIC to provide this bus handshaking, Simpod provides a simple bus controller in Verilog that can be used for this purpose. The data for all of these transactions will still remain in the DeskPOD memory modes.

Software Debugging Environment

DeskPOD can also be used to test initialization software and system diagnostics. The requirements for software debugging and development are much different than hardware verification. For hardware verification, engineers do not want to write any software, only verilog testbenches. Software engineers also like to work with a software debugger. DeskPOD provides software debugging via the COP interface on the MPC7400. COP or Common On-Chip Processor, uses the JTAG scan chain to control the execution of the CPU. It provides all of the hardware support to make software debugger integration possible, and does not require the software running the MPC7400 to be modified or customized for the target.

Conclusion

DeskPOD provides all of the needed functionality to verify both hardware and software using the BFM techniques hardware engineers prefer for ASIC verification. Using the device silicon is the best way to create models that meet the verification requirements of the entire project.