Embedded Systems Verification Solutions – where FPGA meets the processor

Author: Zbigniew (Zibi) Zalewski

For today's projects it is no longer enough to use FPGAs in a traditional way. As the number of mobile and personal applications grows usage of embedded processors becomes a necessity. New FPGA devices with so called soft or hard core processors enable fast migration from the FPGA-only to the SoC world of appliances and projects. This affects not only the chips, but also the tools for SoC designers supporting the latest devices. To catch the wave, the major FPGA vendors are trying to outdo their competitors in delivering new design entries for a smooth integration of the software and hardware sides of the design.

A positive effect of this competition is the availability of quite good design environments for embedded systems, but what happens after I build my SoC design in the typical system level utility? My next step is to move onto simulation and testing. Let's assume I have a typical structure with one processor, system bus and peripherals. After importing the design into the HDL simulation environment, the functional simulation is based on the generated HDL sources and BFM (Bus Functional Model) processor model. This is a good start, but when the design grows and gets more complicated we encounter a dramatic slowdown of the simulation. One of the main causes for this is the simulation of the processor models. It takes hours to simulate a bit more complicated algorithms and data processing.

The natural thing to do would be to go to the prototyping board and run everything in hardware, but this is way too soon for me. My peripherals are still under development and I need to have the whole design or at least a few modules in the simulator. But if I keep it like this my software side (processor and program memory) gets completely stuck. It's not only that simulation time kills everything, but there is also no way to debug my C/ASM code for the processor.

Vendor	Actel	Altera	Xilinx
Processor	CoreMP7 (ARM7)	Nios II	MicroBlaze/PowerPC
Design Environment	Core Console	SOPC Builder	Platform Studio (EDK)
Software Development Environment	Soft Console	Nios II IDE	Platform Studio SDK (EDK)
System Bus	AHB	Avalon	OPB/PLB

Table 1.   Embedded processors comparison (selected vendors in alphabetical order).

Why don't I use a hybrid tool to resolve my SoC problems?

It looks like there are two major problems when verifying SoC designs. One is slow simulation time, the other limited debugging functionality, especially when we talk about processor debugging. To resolve the simulation time issue we could use a hardware accelerator which is basically a hardware board and simulator connected via a high-speed interface. To keep it synchronized with the simulator the interface to the accelerator has to be event-driven.

Let's assume we are using an accelerator, the simulation time has been decreased, and we're also benefiting from enhanced debugging capabilities for the hardware peripherals located in the simulator and the accelerator board. But still there remains one problem: How can I debug my processor? What if we connect the processor debugger to this configuration and receive a concurrent debugging environment? In that case, the HDL simulator delivers the hardware device debugging (residing in the simulator and the board), while the processor debugger assures access to the processor memory and executed program with the typical functionality such as stepping or breakpointing. That kind of hybrid configuration could resolve our major visibility issues and save a lot of simulation time.
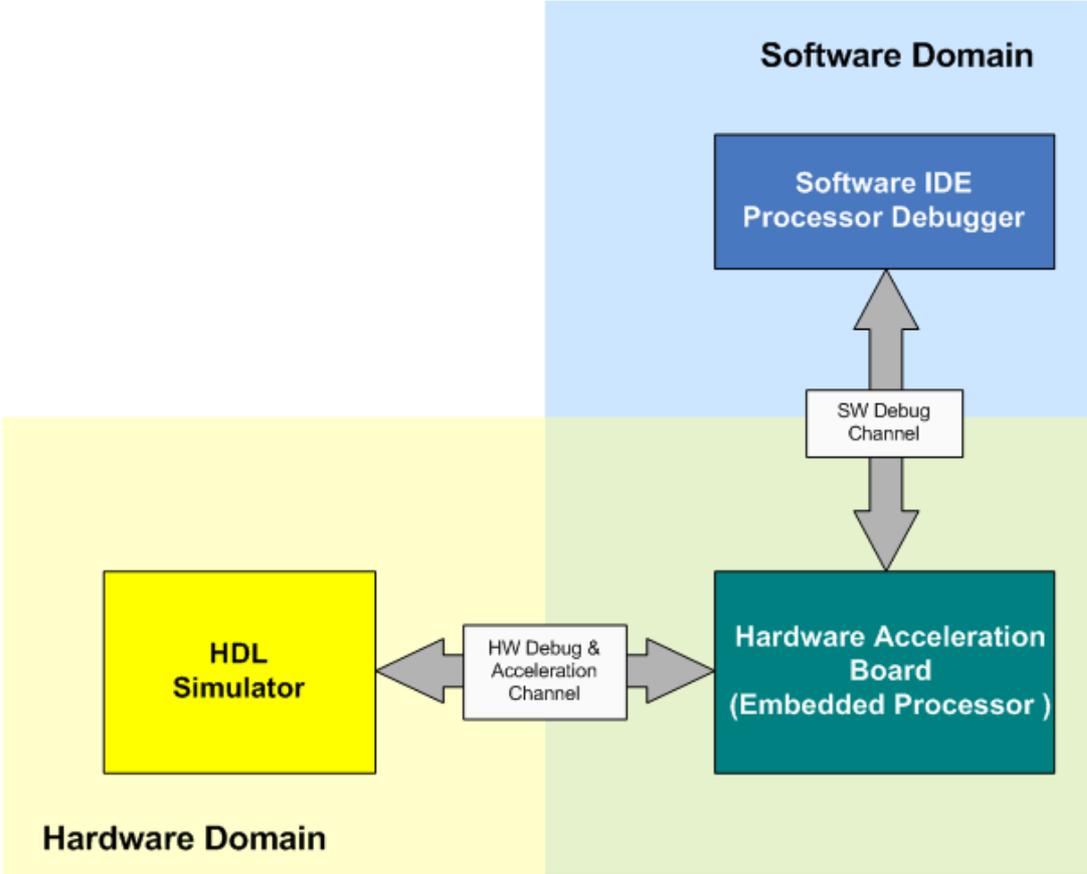

Figure 1.   Hybrid co-verification solution with integrated HDL simulator and processor debugger.

When the real problems begin

Of course everything looks nice in the diagram, but when dealing with real tools we need to resolve a few more problems. One thing is when the HDL simulator drives the whole system the processor is still working on the slow software clock. While this is much better than in a simulator-only environment since the simulation of the peripherals is accelerated, we still want more and are looking for ways of optimizing our solutions. Therefore we need to go a bit deeper with the SoC verification issues.

The other problem is the debugging channel. Sometimes embedded processor cores don't even work on the slow clock (delivered from the simulator) or debug channels cannot accept such a big response time from the processors. In other words, we might step into difficulties with keeping a stable debug channel for the processors. Gathering all that knowledge together

it looks like it would be a good idea to run the processor on the fast speed when its debug is required, but when transactions appear between the master (e.g. processor) and slave devices it would be better to synchronize everything by the simulator driven clock. In that case let's put a bridge that cuts the design in two parts – one hardware related (mostly peripherals) and the other software based (processor and its program memory). One side of such a bridge should accept the event-driven simulator clock (Hz-kHz), the other should be able to cooperate with the fast (MHz) hardware clock delivered to the simulator and the program memory (Figure 2).
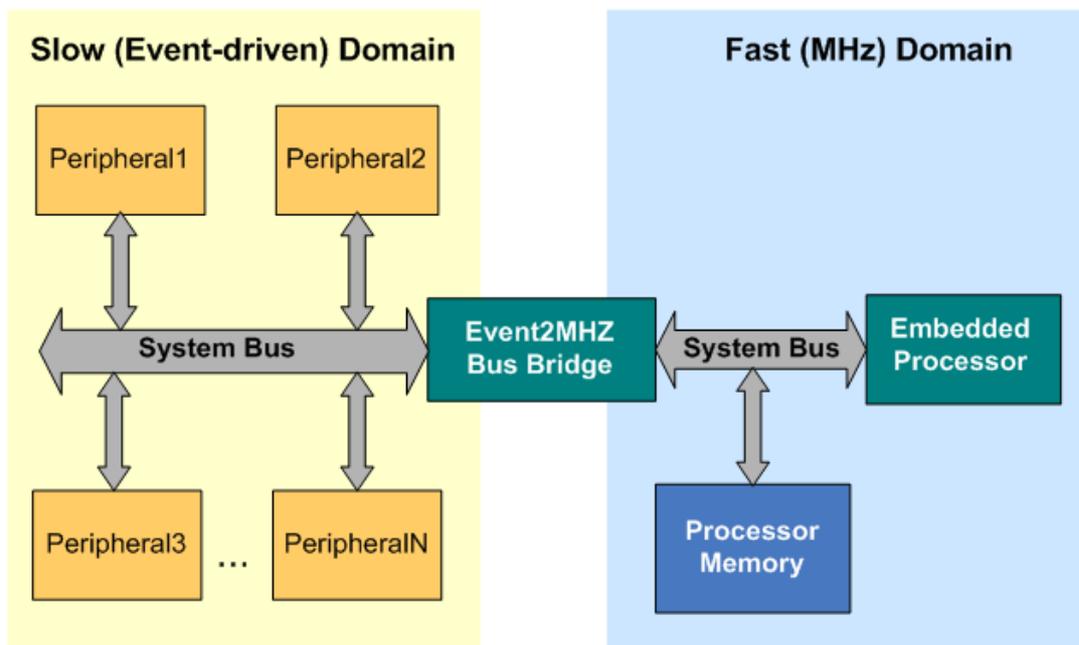

Figure 2.   Software/hardware simulation domains synchronized by the Event2MHz SoC bridge.

The Event2MHz bridge separates the hardware part of the project from the software section. It synchronizes the fast requests from the processor with the "slow" responses from the hardware. Such a solution enables fast debugging of the processor comparable to the prototyping speed in an event driven accelerated simulation of the HDL peripherals. Unfortunately, such a bridge is not a simple module to develop. It depends upon the system bus specification, the processor parameters and the transaction routines processed during verification. Therefore the usage of professional EDA tools is strongly recommended in order to avoid many obstacles and unexpected problems.

So what have we finally received? We have a simulation system with the debugging capabilities for the hardware modules (VHDL/Verilog/SystemVerilog) in an HDL simulator and the software program (C/C++/ASM) in the processor debugger. In other words, we have a verification solution for embedded systems based on FPGA devices.

In summary, the major benefits from such a mixed solutions are:

- Ability to verify the whole project during the development stage, way before the code is ready for prototyping. Event-driven co-simulation channel allow for the incremental mapping of the models in the simulator and the accelerator board keeping the advanced debugging functionality.
- Access to the latest design version during the design process without waiting for the prototype, which is a main obstacle for software developers.
- Working on the latest firmware version delivered from the software guys, verified on the current hardware version.
- Verifying with a real, hardware model of the processor instead of slow and inaccurate software models. When properly synchronized such a hardware processor model delivers a great simulation speedup and makes the easier migration to the prototyping stage.

Main engineering targets include:
- SoC design teams working on software/hardware related tasks for modern appliances.
- Development design stages when simulation time and strong debug capabilities during the development process are a priority.
- Concurrent development and verification activities of hardware designers and software developers without wasting time on final prototype release.