Title: Challenges of ASIC Prototyping in FPGA.

Author: Zbigniew (Zibi) Zalewski

To prototype or not to prototype my ASIC in FPGA? The obvious answer seems to be YES. ASIC prototyping in FPGA saves a lot of money on tape-out and allows the detection of major problems before the ASIC goes to production. However, the decision is not quite so simple, especially when the ASIC is big and during its development there was no consideration of the prototyping environment. At a first glance, ASICs and FPGAs represent two different worlds, but with the increasing speed and capacity of the latest FPGAs they are becoming the major testing environment for ASIC verification. The best results during ASIC prototyping in FPGA can be obtained when the design team is aware that verification is going to be performed on the FPGA platform.

The first problem is the selection of the hardware platform. Mostly there are two choices – you can use either your own board or the universal board from the prototyping vendor. In most cases it is much better to spend the money for the universal board as this will save you a lot of time - between 3 to 12 months for high-speed prototyping board development, depending on requirements.

Assuming we have the hardware platform, we still need to address the following issues:

- ASIC Library Mapping – this is a starting point of the setup process. Each of the ASIC designs is based on the specific vendor library. The library is built of the primitives that cannot be mapped directly in FPGA. What has to be done is primitives remapping from ASIC to FPGA technology, with of course securing the compatibility of the FPGA representatives. Different techniques can be used to confirm the functionality of the ASIC2FPGA library, the example might be standard simulation comparison with controlled code coverage or formal verification methods. It is worth to add if you start from RTL level and synthesize your ASIC targeting to FPGA technology only the directly instantiated ASIC primitives must be remapped. In case of post-synthesis ASIC netlist practically all primitives from ASIC library must be remapped to enable further processing and implementation to FPGA.

- FPGA Capacity – we're talking here about the medium and big sized ASICs so even the latest FPGAs like Virtex 4/5 or Stratix II/III are not able to contain the whole ASIC in one chip. Fortunately FPGA vendors are working constantly on the capacity, so usually 2-4 FPGAs are enough for quite big ASIC projects. However using multiple FPGA hardware platforms creates partitioning needs. Most popular are two methods, manual partitioning and automatic partitioning. Manual way might a good choice when design is quite well divided into smaller modules, with resource control it gives very efficient results. Automatic partitioning gives better results for very big designs where single modules require big capacity, it is not worth to struggle with it using manual partitioning. Software tools will handle the design clustering and proper interfacing between available hardware resources (FPGAs).

- I/O Limitation – the problem doesn't exist when we use only one FPGA (you should be so lucky!), but when you need to map the design into a few FPGAs there is always a lack of pins on the FPGA. The biggest packages offer 1000-1200 user I/O pins; sometimes this is enough, but that's not very often the case. Good solution for this bottleneck might be signal multiplexing between inter-FPGA interfaces. This operation is very tightly connected with a partitioning, so the good choice might to use partitioning with signal multiplexing, it delivers efficient design mapping with automatic generation of multiplexed interfaces between available FPGA.

- Timing / Clock Issues – this is a real nightmare; how to control multiple clock domains with extensive use of clock gating and dividing techniques, how to receive the correct results during the prototyping and still verify the design. ASIC designs with sophisticated clocking and fine tuned timing cannot be directly mapped to FPGA. Fortunately when following certain rules this migration might be a lot easier and save us a lot of effort. One of the problem solving techniques is a conversion of arbitrary ASIC designs into equivalent circuits clocked by one global clock line. Gated and divided clocks are replaced with clock enables, creating a single clocked circuits much easier for the FPGA implementation.

- Memory Mapping – ASIC memory models usually don't match those used in FPGA. What's even worse, in many cases there are no synthesizable models for those memories. Best solution is to use prototyping boards with a specific memory devices, but it is quite rare to find a board that meets all of our requirements or it takes time to develop a board with a specific devices. The other way might be to use memory partitioning tools that are able to map user modules into available memory resources, of course it is a compromise between time available for the memory verification and a testing environment.

- Debugging – when we finalize the setup stage we need to think about the debugging of our prototype. Visibility of the prototyped design is very important. We mustn't forget we're running on full speed, so there's tons of data to be transferred to software JTAG-based debuggers or measuring equipment. When using JTAG-based debuggers it is good to make sure that probes inserted by the debugging setup tools are not changing the timing parameters of our design. Such tools offer quite well visibility with an extensive triggering conditions allowing for optimizing the stream of data transferred through JTAG interface.