DESIGNING MULTI-FPGA PROTOTYPES THAT ACT LIKE ASICS

May 2009

ABSTRACT

FPGA prototyping has become indispensable for functional verification and early software integration of prospective ASIC designs. If the ASIC in question is large, it is often necessary to spread the functionality across multiple FPGAs on a special prototype board. This white paper discusses the importance of choosing the right tools and methods to most efficiently partition ASIC functions into multiple FPGAs for development and evaluation purposes.

Authors:
Ehab Mohsen
DCS Division, Mentor Graphics Corporation
ehab_mohsen@mentor.com

Nang-Ping Chen
President, Auspy Inc
npc@auspy.com

DESIGNING MULTI-FPGA PROTOTYPES THAT ACT LIKE ASICS

When prototyping an ASIC, ASSP, or SoC, mapping the functionality onto a single FPGA is all the challenge most designers care to confront. There is plenty to think about: dealing with differences in ASIC and FPGA architectures, optimizing for performance and area requirements, and accounting for a debug strategy. But these steps are just the tip of the iceberg when compared to implementing a large ASIC's functions on a multi-FPGA platform.

Over the years, FPGA prototyping has become indispensable for functional verification and early software integration of prospective ASIC designs. By conservative estimates, the largest FPGAs currently in production have a capacity of about 1.5 M equivalent ASIC gates. Prototyping an ASIC larger than this means spreading the functionality across multiple FPGAs. It is a complicated task, but one that is well worth the effort.

With mask costs approaching $3M for 45nm designs, avoiding a re-spin by prototyping with FPGAs is a price worth paying, even if it means minor deviations from the final ASIC environment in terms of clocking, memory, or speed. The larger the ASIC design, the greater its development and manufacturing costs. Therefore FPGA development becomes even more urgent and beneficial when the target is a large, complex ASIC device. Of course, multi-FPGA prototyping is itself a complex undertaking but a little planning can go a long way.

FPGA SYNTHESIS TOOLS MUST BE ABLE TO "SPEAK ASIC"

There are certain fundamental technologies that every prototyping flow must include, whether it be for a single- or multi-device platform. If the goal is to accurately approximate the behavior of the target ASIC design, constructs such as gated clocks and Synopsys® Designware® components are indispensable.

Clock gating is necessary in the ASIC world to conserve power in portable devices, but it can lead to poor results in FPGAs. Hence, conversion of these gated clocks to their FPGA functional equivalents is a must. Most clock nets in an FPGA should be mapped to high-speed, low-skew clock lines.

Nets directly driving sequential elements are typically routed this way but when clocks are gated, they are taken off these high-speed routes. The result can be poor performance and potential setup and hold-time violations. Mentor Graphics Precision® Synthesis converts gated clocks to functionally equivalent logic by means such as using the Enable (CE) pin found on most sequential elements.

Figure 1 depicts the solution for a gated clock. It is important to remember that not all gated clocks are this basic, though. Clocking schemes can be extensive, implemented with multi-level logic that drives not only registers but also memory and DSP blocks. Clock divider circuits such as simple counters are another example in which a clock is taken off the clock-line, potentially resulting in severe skew. Precision synthesis is equipped to handle such structures.

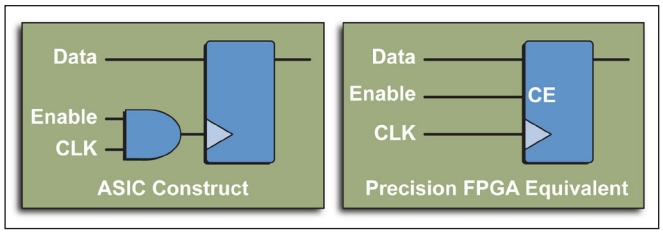

Figure 1: Equivalent gated clocks can be re-created automatically during synthesis.

In addition to gated clocks, ASIC designs commonly use Synopsys Designware library components such as data path elements, memories, and FIFO controllers. Precision provides transparent support of these components, meaning that instantiations of these blocks within the HDL need not be modified.

Performance is always important for FPGA projects, and ASIC Prototyping is no exception. More often than not, the prototype is expected to run not at actual ASIC speeds, but fast enough either to handle real-time input or communicate with an external interface. At the very least it should provide a verification environment several times faster than simulation. In such cases synthesis needs all the usual optimization capabilities, such as advanced technology inference, retiming, and easy control of resource allocation.

One of the more critical performance optimizations in today's complex designs is physical synthesis—the ability to use physical characteristics of the target FPGA to improve frequency. While regular RTL synthesis considers only logic cell delays and simple timing models of interconnect delays, physical synthesis takes into account the actual logic and how may be placed on the device.

With advanced delay models of routing resources, physical synthesis can more effectively produce a netlist optimized for performance. This is particularly useful for the high-end devices typically used in ASIC prototyping. Extensive device support of physical synthesis is critical, since high-end FPGAs are available from many vendors. Broad device support keeps one's options open when seeking the most suitable FPGA for prototyping. Precision RTL Plus offers physical synthesis support for 20+ devices from all major FPGA vendors including Actel, Altera, Xilinx, and Altera.

THINKING AHEAD ABOUT PARTITIONING

Once it is established that synthesis can support a basic prototyping flow, the next hurdle is devising a partitioning strategy for the FPGAs that have been chosen. Carefully addressing this issue can affect the platform's system performance, the cost of hardware, and time spent on manual intervention.

It is critical to think about "high level" logic partitions at the very beginning of the design cycle. This is easier said than done, particularly within large teams in which the designers and verification engineers may be working at different sites or with less-than-perfect sharing and coordination. Experience has shown that SoCs designed with prototyping realities in mind can achieve much greater system performance improvement than designs that don't receive that consideration. Though much can still be done at the back end, the benefit of pre-planning cannot be overstated.

TOOL INTEGRATION APPLIES LEVERAGE TO THE PROBLEM

It is important to develop an early estimate of the number of FPGAs that will be required, as well as their interconnect structure. This sizes the task that lays ahead. Automated or semi-automated partitioning software can be of immense help in this exploratory phase. Using such tools can be as straightforward as importing all RTL and post-synthesis design files, or a combination of any, and letting the tool perform an accurate gate-level estimate using encapsulated, bottom-up synthesis. Older generation RTL partitioning software limited itself to rough area estimates through pure RTL analysis, thereby ignoring gate-level details. More recently, flows have moved beyond this by performing full up-front synthesis to extract accurate timing and area data. The integration and certification between the partitioning software and synthesis tool is critical at this stage.

The Auspy Custom Emulator Compiler (ACE Compiler™) from Auspy Development, Inc. integrates Precision Synthesis to create such a flow. It allows multi-language design import and full bottom-up synthesis for accurate gate-level estimation. The Auspy compiler is the right tool for the mapping phase that determines the number of FPGAs and the complexity of the ASIC partitions.

VENDOR-INDEPENDENT SOLUTIONS PROVIDE MORE HARDWARE CHOICES

Eventually the time to choose the prototyping hardware arrives. Pre-loaded PCBs housing two, four, or more FPGAs are commercially available, and of course it is always possible to create one's own proprietary hardware platform. Accurate gate-level data is useful when evaluating the best solution, whether it be an off-the-shelf product or a "roll your own" PCB.

Two issues dominate the choice of prototyping boards. First—obviously—there is the number of FPGAs and capacity of each. But the connectivity among these devices, hardwired or flexible, can be just as important. Off-the-shelf solutions tend to have specific FPGA interconnect layouts—some interconnects are literally "flexible" via cables; others have programmable cross-bars; some are fixed. Or perhaps the interconnect requirements are so specialized that a custom PCB is the only good choice. In either case the ACE compiler makes it possible to explore these options with either a fully automated or semi-automated partition flow.

In a purely software-guided flow, the ACE tool manages the partitions to minimize the number of FPGAs and/or maximize performance. Alternatively, users can semi-manually investigate various partitioning possibilities via an impact table. Graphically dragging and dropping logic blocks into different partitions yields a potential interconnectivity map, which the compiler software uses to complete the partition based on the specified settings. Either of these methods produces a high-level view of the hardware requirements.

Note here that "hardware" implies not only the prototype PCB layout but also the FPGAs it will serve. When comparing FPGA vendors and their products, one FPGA family is likely to offer features that make it more suitable for the application than others. Understanding the requirements and having the flexibility of vendor independence before selecting the hardware can prevent buyer's remorse later in the implementation phase.

The ACE-Precision flow is among the few truly hardware-independent solutions. It has no ties to specific hardware vendors, which allows for objective platform selection.

TIMING-BASED, HIERARCHICAL PARTITIONING CAN SAVE TIME

Full bottom-up synthesis is not only important for accurate gate-level estimation, but for accurate timing analysis to achieve a partition optimized for performance. High operating speeds cannot be achieved without careful analysis of the timing paths. The ACE Compiler performs hierarchical analysis, partitioning, and optimization in order to keep the netlist database to a manageable size, allowing for reasonable run-times for large designs. This is in contrast to "flattening" the design, or effectively reducing the netlist into one "module" that creates a copy (with a new name) of each lower-level module definition each time it is re-used. This approach not only leads to long run times but may not even be feasible for extremely large ASICs. A hierarchical approach optimizes the entire design according to its original hierarchy, allowing for effective clock and data-path analysis.

FINDING THE BALANCE IN CLOCK DISTRIBUTION

Clock distribution is a key element of timing-based partitioning. The skew in multiple-FPGA systems consists of the combination of skews on the board and inside the FPGAs. When a clock is generated inside one FPGA and distributed to others, the board skew for each receiving FPGA must be well-balanced; otherwise it will cause hold-time violations. The loop-back structure shown in Figure 2 is one method to ensure all design modules receive the same clock at essentially the same time. Other alternatives to eliminate board

skew include isolating the clock circuitry into an FPGA by itself or replicating that circuitry in each FPGA. All methods are supported by the ACE compiler, which strives to balance skew of the clocks driving the design.

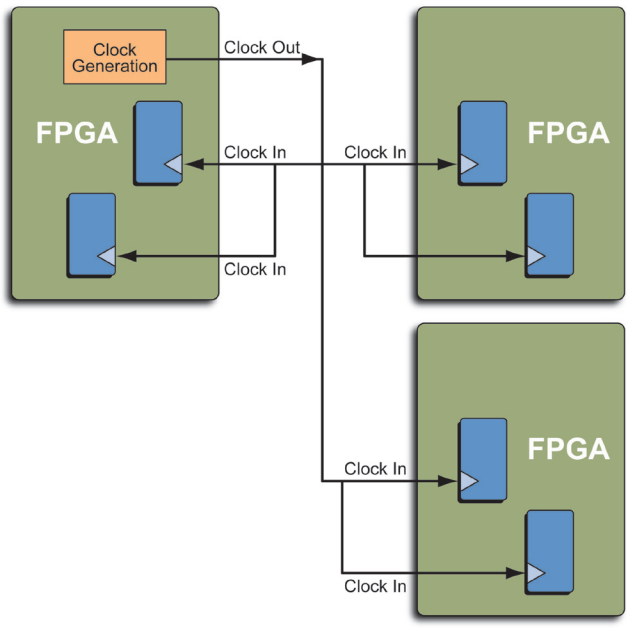

Figure 2: Clock loop-back techniques avoid clock skew on the prototype board

Given the limited clock distribution resources on a typical PCB, multiple schemes may coexist to ensure minimal board skew. A well-planned partition, done manually or via an automatic partitioner, will limit the number of clock domains to be split across FPGAs so every clock gets the support from the FPGA's low-skew clock lines.

OPTIMIZING AROUND HARDWARE LIMITATIONS

Even with the right clock distribution, however, the system may be running at an order of magnitude slower than its optimum rate if the rest of the design is not partitioned properly.

Excessive delays often come from purely combinatorial signals traversing a single FPGA as a result of a design partition. These "combinatorial hops" introduce extra delays through board traces and FPGA input and output buffers. They also disable potential timing optimizations that synthesis could conceivably perform on complete paths as opposed to partial segments. The partitioner's timing engine must recognize "hops" and avoid them when possible—though it may not be possible to eliminate all hops.

Degradation in performance can also come from pin multiplexing, typically done to overcome the limited pin count per FPGA. The lower the partitioned logic block's I/O count, the less pin multiplexing is needed, and the higher the system performance. This is a well-known challenge for partitioning software. The ACE compiler uses data-path clustering algorithms that not only consider gate count but also the timing criticality of data paths and the potential for combinatorial hops.

CONSTRAINT GENERATION ENSURES PROPER TIME BUDGETING

After the partitioner has done its job of clock and data-path optimization, it should generate a set of timing constraints to pass accurate timing budget information to synthesis in order to perform FPGA-wide optimization. The ACE compiler generates timing specifications for the whole design, adjusted per partition to account for the delays on interconnect wires, combinatorial hops that cannot be eliminated, and pin multiplexing. This timing budget is efficiently performed with the help of the partitioner's built-in timing engine. The new set of the timing constraints is passed on to Precision Synthesis to ensure each FPGA is optimized for best performance by synthesis and place-and-route.

INCREMENTAL FLOWS SAVE RE-SYNTHESIS TIME

Given the debug orientation of the methodology, ASIC prototyping requires incremental flows to reduce iteration time when homing in on a genuine design

bug. The ACE compiler supports a re-synthesis of the relevant FPGA. Verification schedules cannot afford re-synthesis of the entire design of four or more FPGAs. Parallel synthesis and place-and-route can also mitigate this problem; both are supported with the Auspy-Precision flow.

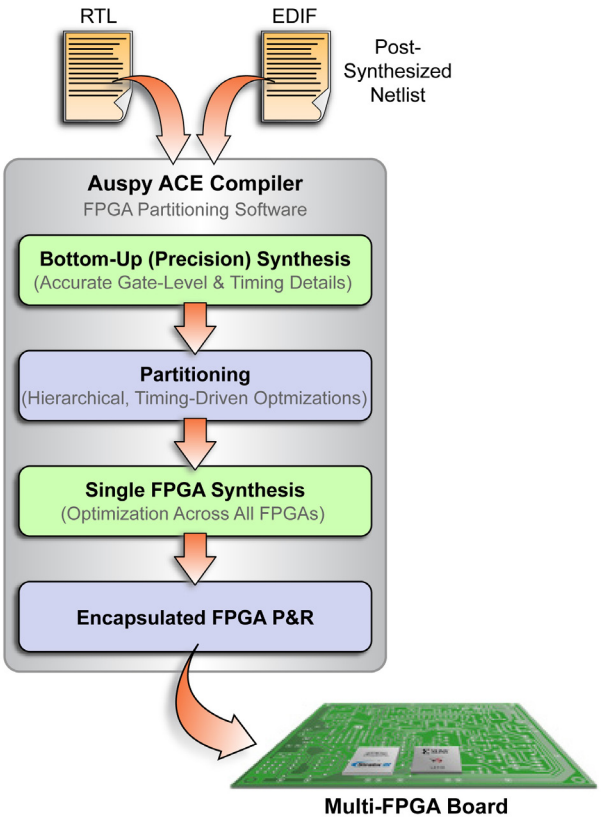

Figure 3: ACE Compiler and Precision Synthesis Integrated Flow

CONCLUSION

Experience has shown that complex multi-FPGA prototyping requires careful integration of advanced synthesis and intelligent partitioning tools, as summarized in Figure 3.

But the importance and impact of careful planning must not be overlooked. Many of the problems that have plagued prototyping engineers over the years have been either eliminated or simplified with the latest software flows, but an ASIC team can save itself even more grief by considering a partitioning strategy up-front and by carefully selecting the right platform with a truly hardware independent flow.

For more information, call us or visit: www.mentor.com/precision

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777 USA
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000

Silicon Valley
Mentor Graphics Corporation
1001 Ridder Park Drive
San Jose, California 95131 USA
Phone: 408.436.1500
Fax: 408.436.1501

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
Room 1001, 10F
International Trade Building
No. 333, Section 1, Keelung Road
Taipei, Taiwan, ROC
Phone: 886.2.87252000
Fax: 886.2.27576027

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo
Japan 140-0001
Phone: +81.3.5488.3030
Fax: +81.3.5488.3021

MGC 05-09 TECH8280-w