Field Programmable Gate Arrays (FPGAs) are becoming more attractive than Application Specific Integrated Circuits (ASICs) to design teams developing next-generation electronic products. That’s due to a variety of reasons including their increased gate counts, versatility, and lower development and manufacturing costs. FPGA devices are now being fabricated in advanced, ultra-deep submicron technology with multi-million gate capacity, and clock speeds approaching 400 MHz. With these larger, more complex FPGAs come new design challenges, including problems associated with interconnect delay. In this article, Salil provides tips for FPGA design.

Interconnect delay
As witnessed when high gate-count, deep submicron ASIC designs first appeared, interconnect delay accounted for as much as 70 to 90 percent of overall circuit delay as feature sizes shrunk below 0.18 µm. Large designs also impact cycle time because of increased place and route runtimes, and an increased number of design iterations needed to reach performance goals. Unfortunately, Electronics Design Automation (EDA) software used to design FPGAs has remained largely unchanged over the years, and is unable to adequately address these problems.

Hierarchical design
The latest advancements in FPGA software provide hierarchical design capabilities that enable designers to partition their physical design into smaller, more manageable pieces. This significantly reduces the time to understand, verify, and implement the design. Partitioning also provides an incremental design methodology that reduces iteration times for implementing Engineering Change Orders (ECOs). Breaking a design into smaller pieces can also reduce runtimes, and the computer resources necessary to place and route the design.

Advanced EDA tools
What’s needed are advanced ASIC-style EDA software tools and methodologies that provide FPGA designers hierarchical, block-based design techniques to identify, analyze, and correct problems early in the design cycle. This, in turn, will lead to fewer design iterations. Advantages to implementing advanced ASIC-style techniques include quicker incremental design changes, improved performance, Intellectual Property (IP) reuse, faster place and route, and tighter utilization control. Designers perform early analysis and planning to maximize performance, and thereby avoid lengthy and repeated iterations.

Place and route
Using the latest FPGA design tools, designers can improve circuit performance by applying advanced floorplanning and static timing analysis techniques early in the design process. Designers can use early congestion analysis to help them select the appropriate FPGA device, and optimally place the logic within it. This results in a reduced number of iterations between each phase of the design process, and a reduced overall time to reach timing closure.

Place and route has far greater difficulty with designs that are flattened – that is, without hierarchy – yet most FPGA software
still operates on a flat netlist. With a flat methodology, even a minor design change isolated within a single logic block necessitates a complete rerouting of the entire netlist. Not only is place and route lengthy for a flattened netlist, it may take many iterations before the design again reaches the desired performance.

Using ASIC-style design techniques, designers draw on hierarchy to help place and route achieve better results, often using less computational resources. By breaking their designs into smaller blocks, they don’t need to run place and route on the entire design each time they make an incremental design change. Instead, they run place and route on the block affected by the design change, while they leave the rest of the physical implementation intact. Using such a block-based flow, they can also sequentially implement and assemble their designs. They can start with the most critical blocks, and successively place and route additional blocks as designers complete the logic contained within.

When designing using a flat methodology, a designer often tries to reach timing goals by trying multiple routing runs, each with different random seed values, hoping that one of them will produce a design with adequate performance. A hierarchical methodology provides a more deterministic process, by enabling designers to define area groups to steer place and route towards acceptable timing values. This serves to further stabilize the place and route process, which makes the results more reliable and predictable.

**Parameter management**

Designers of large FPGAs iterate their physical implementations many times because they are trying to cope with too many simultaneous physical parameters such as connectivity, utilization, I/O placement, clock regions, power, and timing. Without ASIC-style techniques, there is little guidance for designers to know which parameters must be manipulated to reach their requirements until place and route has already occurred. Worse, many requirements are interrelated; manipulating parameters to achieve one requirement often causes problems in achieving several others. Consequently, designers spend too much time iterating their designs.

**Floorplanning**

ASIC-style design tools help designers reach their requirements, prior to place and route, through early design analysis tightly coupled with floorplanning. With floorplanning and analysis, the designer can substantially reduce the number and length of place and route iterations by reviewing early feedback about the physical design.

For example, early connectivity analysis may uncover a potential area of routing congestion. The designer can then change the floorplan to alleviate the problem, rather than wait for lengthy and inconclusive place and route results. By adjusting the floorplan, place and route algorithms have the necessary partitioning, block arrangement, and physical constraints for guiding place and route to an early success. More efficient floorplans generally lead to improved timing, and reduced place and route time.

**Static timing analysis**

Advanced FPGA design tools also provide physical optimization later in the design cycle, to address implementation problems that may arise after place and route. For instance, designers can run static timing analysis to find all the remaining critical paths in their design that do not meet timing requirements. They can then grab all the logic in these critical paths, and place them in a small block, thereby constraining interconnect and minimizing delay. The designer can then run a timing analysis to see if performance has benefited. By this process, designers get nearly instantaneous feedback, rather than waiting for hours of place and route to complete.

**Utilization**

Utilization is an important aspect of FPGA design. In some cases, it’s necessary to crowd as much logic into a given device as possible to meet production volume requirements. In other cases, some amount of spare space is desirable to accommodate bug fixes, naturally occurring design changes, ECOs, or even future planned field upgrades.

Using block-based, hierarchical design techniques, it is easier to control utilization. Designers trying to maximize utilization can set utilization controls to a given level, then run block-level place and route. If it produces successful results, they can change the utilization setting to shrink the block, then re-run block-level place and route. This process can continue until place and route fails, which means the designer has achieved the maximum possible utilization for the given block.

Designers who need to leave some amount of spare space in their designs can use a lower utilization setting for blocks where extra space is needed. A wise technique is to leave more space in blocks that have yet to be verified to accommodate anticipated bug fixes. Designers may need less spare space in blocks that are field-proven, since it is unlikely that extra space will be needed to fix bugs within them. Designers can also create hierarchical blocks that make efficient use of fixed resources such as RAM and multipliers.

**Design workflow**

Like ASIC designers, FPGA designers can accelerate design time by working as a team, and by reusing intellectual property from previous designs. Using ASIC-style design techniques enables teamwork and IP reuse. Designers can take a block-based approach to divide their work into more manageable pieces, and assign responsibilities of designing them to individual team members. Designers can also reuse blocks from previous designs, or even purchase them from a third party to save design and verification time.

By employing ASIC-style techniques, designers can fully characterize their design blocks by freezing the placement within them so that power, timing, and other characteristics remain constant wherever they are used. They know that these blocks meet their physical requirements, and they can connect them to form larger designs that also meet their requirements.

**An example**

One challenge when designing with today’s complex FPGAs is that they have a limited amount of internal memory, and moving data to and from external memory frequently becomes a bottleneck. It would save a significant amount of time if FPGA designers could reuse standard high-speed memory interface blocks, with consistent performance, from one design to another.

Figure 1 shows how one company designing FPGAs did just that. In this case, they used the PlanAhead software from Hier Design, to create the memory interface blocks and to reuse them with floorplanning. Using the tool they were able to lock down and ensure the consistent performance of critical memory interface blocks such as PCI and DDR. This ASIC-style flow created and managed the physical constraints for all the logic inside of the
memory interface blocks, which led to a successful place and route. Figure 1 shows reused external memory interface blocks within a floorplan.

The same reuse methodology also comes in handy for internal, block-level interface logic. Figure 2 shows module interfaces as IP blocks, locked down for reliable performance so they can be reused on other designs.

In this case, there was a PowerPC core (shown in yellow) that was surrounded by IP blocks that interface it to the rest of the design. As with the memory interface blocks, the logic blocks that surround the PowerPC core were locked down using physical constraints created and managed by an ASIC-style methodology.

**Conclusion**

FPGA designers are grappling with long place and route time, too many design iterations, and difficulty reaching and maintaining physical design requirements. They can overcome these problems by adopting ASIC-style techniques that enable them to reach design closure faster, improve performance, reuse intellectual property with consistent results, and ease incremental design changes.

**Salil Raje** is the Chief Technology Officer (CTO) and cofounder of Hier Design. Previously, he was director of research and development at Monterey Design Systems where he was responsible for the development of physical prototype and analysis software. At Monterey, he was awarded four patents, with another two that are pending. Prior to Monterey, he worked at IBM’s T.J. Watson Research Center.

Dr. Raje received a Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, in Madras, India. He completed a Master of Science degree, and a Ph.D. in Computer Science at Northwestern University in Chicago.

For further information, contact Salil at:

**Hier Design, Inc.**

2350 Mission College Boulevard

Suite 850

Santa Clara, CA 95054

Tel: 408-982-8240

Fax: 408-982-3838

E-mail: salil@hierdesign.com

Website: www.hierdesign.com