CMP United Business Media

PROGRAMMABLE LOGIC DesignLine

March 01, 2006

How to achieve fast timing closure on FPGA designs

Using graph-based physical synthesis to achieve fast timing closure on FPGA designs.

By Jeff Garrison, Synplicity

Technological advances in FPGA devices have created both application opportunities and, at the same time, tough design challenges for developers. In new devices, for example, logic routing rather than propagation delays dominate timing paths. Placement of logic and selecting routing to minimize delays complicates timing closure. Designers need tools that understand how to exploit the capabilities of new devices and realize their full performance potential in applications.

Achieving FPGA performance requirements using conventional tools has traditionally involved multiple iterations through synthesis and Place-and-Route (P&R) in order to minimize routing delays. After each pass, delay information is back-annotated into the design database and used to adjust placement and routing to improve timing. This process is inefficient and the outcome uncertain because the approach implements designs as a series of disconnected steps in which synthesis and optimization are separated from placement and routing. That is, placement and routing are performed only after synthesis has completed, by which time the P&R tools may face an impossible task with regard to achieving timing closure. The underlying problem is that conventional logic synthesis fails to adequately account for routing delays and other device characteristics at the outset of the process.

What is needed is a synthesis algorithm that includes placing the design while considering routing resources and their delays as part of optimization so that the routing that follows is a straightforward task. Such an optimization process would yield predictable results with the best possible timing performance.

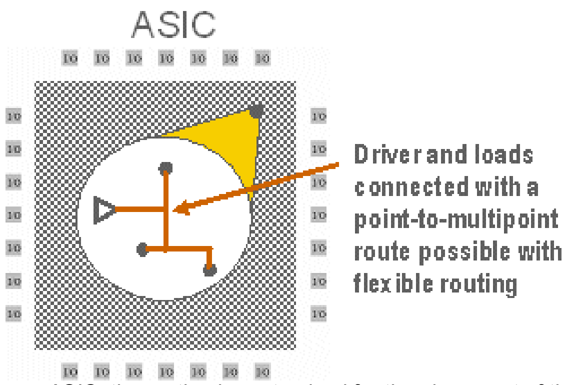
Physical synthesis answers the requirements of FPGA designs by using accurate timing estimations of available logic and routing resources for physical optimization during synthesis. Physical synthesis is an essential enabling process in achieving highly optimized and predictable results for FPGA designs. Automated physical synthesis yields superior design performance for the majority of designs.

Expert designers can add their expertise and knowledge of the system to physical synthesis by guiding placement to boost performance, reduce power, and improve clock management. Other advanced design tool features include support for partitioning of designs among teams and the reuse of intellectual property (IP).

The following sections discuss in greater detail the difficulties of implementing physical synthesis for FPGAs by contrasting FPGA with ASIC technologies. The benefits of physical synthesis are shown in the context of understanding the limits of conventional synthesis when applied to an FPGA design. There is also a discussion of pushbutton synthesis verses design planning.

ASIC and FPGA delay models
Conventional synthesis and placement tools implement designs assuming that routing delays are a function of the proximity of logic elements. This model works reasonably well for ASIC designs, but it breaks down in the case of FPGA architectures. Unlike an ASIC, FPGA routing resources are of pre-determined length and density. Thus, the key in achieving FPGA design timing closure is the use of accurate delay path estimates during synthesis that includes placement performed so as to ensure optimal routing.

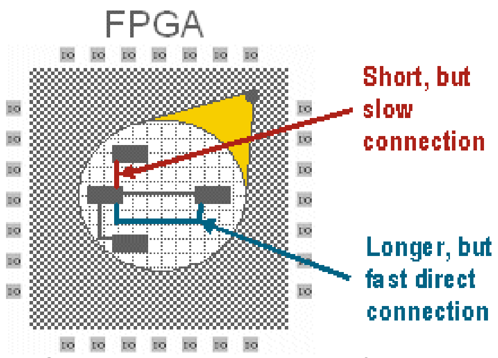
In an ASIC, the routing is customized for the placement of the logic. Once the placement has been performed, it is relatively easy to get a good estimation of routing delay simply by measuring distances from one point to another as illustrated in *Fig 1*.


ASIC
IO
Driver and loads connected with a point-to-multipoint route possible with flexible routing

1. In an ASIC, the routing is customized for the placement of the logic.

The approach that works well for ASICs with their flexible routing does not, however, work for FPGAs due to their dedicated routing networks. In an FPGA, the fastest routing between two points may very well not be the shortest. Routing tracks have different delays and several tracks may be connected to form a path. Tracks are limited, which means that if no short, fast tracks remain unused, then adjacent logic modules must use longer slower tracks.

The situation can be compared with commuting to work. It may be faster to get to work by going slightly out of your way to get on a freeway where you can drive faster with fewer stop lights than by taking the shorter distance on side streets. The same concept applies in FPGA routing, in which some direct routing resources (freeways) are faster than those that have to go through switch matrices (side streets) as illustrated in *Fig 2*.


FPGA
IO
Short, but slow connection
Longer, but fast direct connection

2. In an FPGA, longer (direct) tracks may be faster than shorter tracks.

Thus, a design tool needs to understand the types, lengths, and associated delays of the fixed routing resources when performing placement in order to understand how the design will use those resources. The tool also has to analyze the design itself to foresee – and perform placement to avoid – routing congestion that may accompany instances of dedicated blocks such as memory or DSP blocks. When placing modules, the tool must consider the effect on routing of all modules. Only by analyzing the available resources and the design itself during placement can a tool consistently achieve high levels of performance.

The following sections contrast conventional Back Annotated (BA) synthesis flows and netlist-based floorplanning with graph-based physical synthesis. The shortcomings of BA synthesis and netlist floorplanning can be understood when seen in the context of the requirements for successful FPGA implementation. Those requirements led to the development of physical synthesis.

Multi-pass back-annotation
Conventional physical synthesis tools perform synthesis and optimization prior to the place and route of designs. In this type of flow, a design is first synthesized, then placed and, finally, routed. Timing delay information from the P&R results are subsequently back-annotated into the system for successive synthesis passes using the delay information.

This delay back-annotation (BA) technique had some success when delay paths were dominated by easily calculated logic module propagation delays and when FPGA logic resources were more orthogonal with few dedicated resources.

However, BA method has faltered as routing has come to dominate path delays that vary depending on what routing is used. Delay analysis has become more complex as it is required to analyze multiple track delays and delay dependencies between paths. It is no longer adequate to defer placement and routing until after synthesis hoping to iterate them to meet requirements. Placement that is performed without respect to routing and the design cannot be satisfactorily improved by iteration. Improving slow paths by iteration tends to be at the expense of other paths.

Synthesis must be integrated with a placement that considers routing timing and an analysis of the design. FPGA devices with a hierarchy of routing resources – whose delay calculation depends on which tracks are used – cannot be divorced from the synthesis and placement operations. Decisions to use one resource over another must consider the delay differential between them.

The BA approach can produce satisfactory results when there are small numbers of critical paths. In FPGA designs requiring performance, however, the IPO method of optimizing timing paths individually often leads to degradation of other paths and incomplete timing closure. Moreover, BA does not produce a stable result over repeated design