

Fast Timing Closure on FPGA Designs Using Graph-Based Physical Synthesis



Introduction

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 performance in applications.

Achieving FPGA performance requirements using conventional tools has traditionally involved multiple iterations through synthesis and Place & Route (P&R) to minimize routing delays. After each pass, delay information from the previous is back-annotated into the design database and used to adjust placement and routing to improve timing. The process is inefficient and the outcome uncertain because the approach implements designs in a series of disconnected steps where synthesis and optimization are separated from placement and routing which are performed only after synthesis has completed. By that time these tools may face an impossible task in achieving timing. 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 reuse of IP.

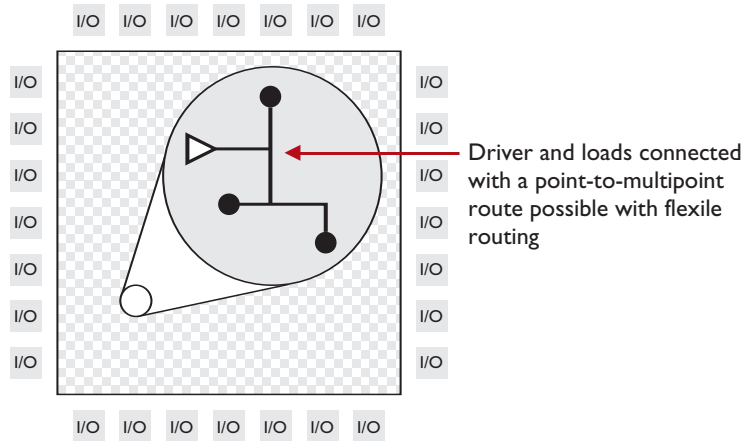
The following sections discuss in greater detail the difficulties of implementing FPGA by contrasting FPGA with ASIC technologies. The benefits of physical synthesis are shown in the context of understanding the limits of conventional synthesis when applied to an FPGA design. There is also a discussion 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. That model works reasonably well for ASIC designs, but it breaks down in FPGA designs. Unlike an ASIC, FPGA routing resources are of pre-determined length and density. The key in achieving FPGA design timing closure is the use of accurate delay path estimates during synthesis that includes placement performed to ensure optimal routing.

In an ASIC the routing is customized for the placement of the logic. Once the placement has been done, it is relatively easy to get a good estimation of routing delay simply by measuring distances from one point to another as shown in the figure below.

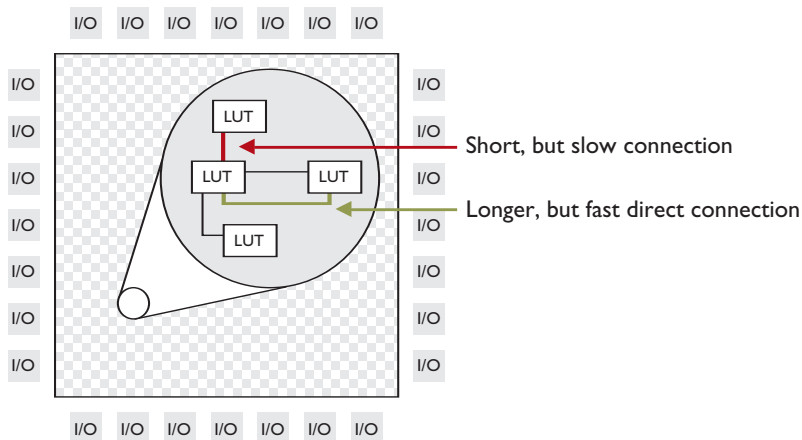
ASIC



The approach that works well for ASIC with their flexible routing does not, however, work for an FPGA due to its dedicated routing network. 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 and if no short, fast tracks remain unused, 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 where some direct routing resources (freeways) are faster than those that have to go through switch matrices (side streets) as illustrated in the diagram below:

FPGA



A design tool needs to understand the types, lengths and delays of the fixed routing resources when performing placement 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.

FPGA Design Synthesis

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.

Multiple Pass Back-Annotation

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

Delay back-annotation 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.

The 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 iterations because optimizing critical paths from one version can create new critical paths in the next. Constraints that are added, for example, to improve timing in one area could worsen it in another.

Designers need reliable results that allow them to change the design without losing what they have achieved in earlier versions. Only by combining synthesis and placement with a clear understanding of routing resources available can a stable design be created that will use constraints to improve critical nets without impacting other logic.

Netlist Floorplanning

Floorplan tools allow designers to define physical regions on the device and to assign logic to the regions. Logic can be assigned hierarchically and compiled incrementally. Floorplan tools support design re-use and team design and let designers control the use of specialized resources.

The technique can improve timing and reduce power, but requires a high level of expertise of the device and analysis of the design. Moreover floorplanning is performed using netlists and cannot optimize the logic, but only change placement. Floorplanning does not, by definition, consider routing resources globally and cannot analyze timing of all paths until after the design is routed.

Graph-Based Physical Synthesis

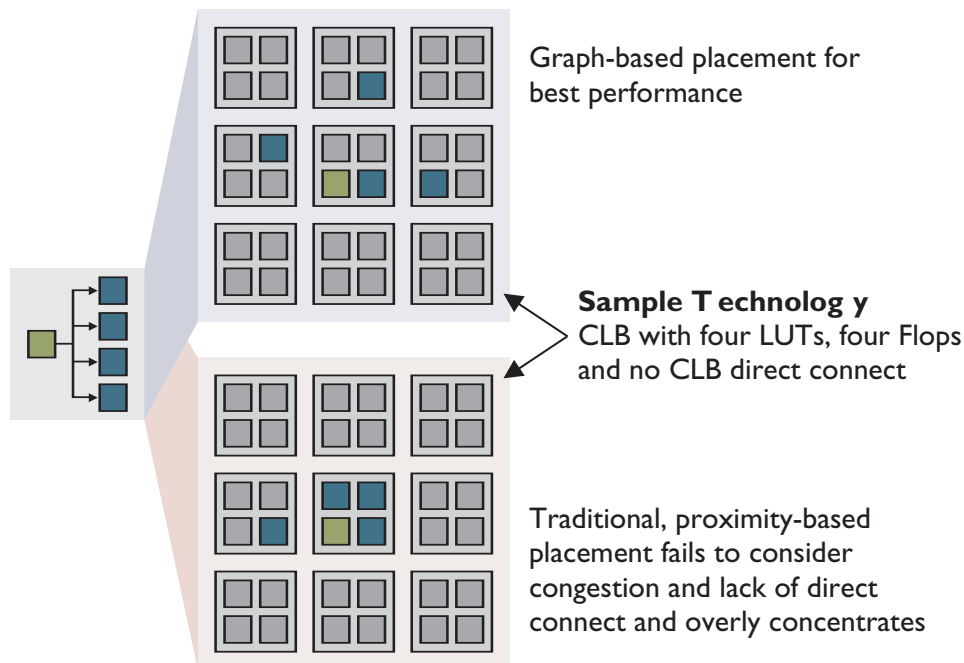
Synopsys' Synplicity Business Group invented graph-based physical synthesis to improve timing closure in a single-pass physical synthesis flow for 90nm FPGAs. The essence of the graph-based approach is that the pre-existing wires, switches and placement sites used for routing can be represented as a detailed routing resource graph. The notion of distance then changes from proximity to a measure of delay and availability of wires.

The graph-based physical synthesis technology merges optimization, placement and routing to generate a fully placed and physically optimized design, providing rapid timing closure and a 5% to 20% timing improvement over the vast number of designs.

Graph-based physical synthesis effectively merges placement and routing because the placement considers routing resources when assigning locations to logic elements and accounts for the effects of congestion and logic module pin delays during the placement to ensure available, efficient routing along critical paths.

The algorithm begins with mapping and optimization followed by an initial placement. The algorithm then optimizes the placement considering the routing resources and congestion. When placement is complete the tool performs another device-level placement optimization and then calls the vendor router for final routing. The router completes much faster than it would following conventional synthesis because the placement allowed sufficient resources. The routing may be followed by timing analysis or programming file generation.

Placement software that considers routing creates a placement that routes much faster and with more uniform path delays rather than a mixture of faster and slower paths. In results from testing hundreds of designs, Synplicity has demonstrated timing improvement in a range of 5% to 20% over logic synthesis alone when using graph-based synthesis.



Example: Comparing proximity with graph-based placement

Graph-based physical synthesis can be guided by providing a graphical design plan (floorplan). A plan for the placement of blocks of logic in specific regions is useful in very high-performance designs or where special consideration for IP and I/O assignments must be made.

Design Planning

Design planning is a method that may be employed by power users to assign logic modules to specific regions of a device. The rationale is that expert designers understand where to place the logic better than the tool because they understand the application environment and have a global view of how best to place modules relative to the fixed locations of I/O cells, clock drivers and specialized functions such as memories to better optimize placement. Design planning can reduce power by constraining logic to a clock region thereby shortening the clock line, for example. It is also important to keep in mind that design planning does nothing to optimize the design netlist – it only helps create better placement.

Design planning can be done at different stages of the design cycle. Design planning after compilation, but prior to optimization, usually involves assigning physical locations and constraints to some or all of the logic modules. This device-level planning allows each module to be optimized separately and to be used in team or performance optimized design. After the modules are individually optimized the entire design is re-optimized for the paths between modules.

Another method of design planning is used after routing to find and extract paths that do not meet timing and may require manual placement. Design planning for timing path optimization requires identification and grouping of those paths that did not meet timing and leaves the balance of the design to the automatic tools.

Frequently a critical path may have sequential end points in common with other paths which may or may not meet timing, but whose timing would be affected by moving the path. Such related paths as well as logic connected to the common paths at only the source or destination must be extracted and grouped in a module that can be placed so as to meet timing for all the paths. The alternative of moving a single path is that the timing of the related paths deteriorates.

The benefits of both types of design planning include better performance through better placement at a high level and more stability because the planned modules may be locked down and won't change. The drawback is that some cross-module optimizations may be missed once modules are locked.

Unlike many planning tools that are used with netlists and require a new plan for each design iteration, design plans used with graph-based physical synthesis remain valid throughout because they are used with the RTL at the beginning of synthesis.

Synthesis tools should help designer understand the results and offer analysis that guide design planning. The tools should include timing and area analysis and be tightly integrated with the device vendor place and route for easy migration within families of devices.

Summary

The increased effects of routing on FPGA path timing requires EDA tools that completely understand the physical properties of the target device and can analyze designs during the placement process in order to achieve timing closure.

Design tools employing techniques suitable for ASIC technologies cannot reliably achieve timing closure in high-performance FPGA design. FPGA tools require another approach because the routing architecture is fixed and, unlike ASICs, proximity does not always imply better timing. Synplicity's graph-based physical synthesis includes placement that uses the device routing resource data and analyzes the design to achieve timing closure. The result is faster designs in less time.

SYNOPSYS[®]
Predictable Success

Synopsys, Inc.
Synplicity Business Group
600 West California Avenue
Sunnyvale, CA 94086 USA
www.synplicity.com

Copyright © 2008 Synopsys, Inc. All rights reserved. Specifications subject to change without notice. Synopsys, Synplicity, the Synplicity logo, "Simply Better Results," and Synplify are registered trademarks of Synplicity, Inc. All other names mentioned herein are trademarks or registered trademarks of their respective companies.