Physical Synthesis for Programmable Logic Devices Technology Backgrounder

Advances in process technology today are enabling a profound increase in the number of applications that can be realized using programmable logic, but along with these technology advances come new design challenges. Densities now in excess of 2 million gates, short design cycles and reduced development costs make programmable devices more and more attractive for a broader range of applications, from networking and telecommunications to high volume consumer products. The underlying technology that makes this possible, ultra deep submicron processing (UDSP), however, presents a new and difficult problem for designers. The physical interconnections between logic elements now dominate IC delay, and high productivity design methods such as conventional logic synthesis fails to adequately account for these effects. In order to restore productivity to the design cycle, the synthesis process must now comprehend physical effects. *Physical synthesis*, including physical optimization during synthesis, is an essential enabling technology for such advancement. Synplicity believes that it is the first company to address this need with the industry's only physical synthesis and optimization technology devised specifically for programmable logic. Advanced techniques employed by the Synplicity tools are designed to restore productivity to the design cycle and enable PLD designers to take full performance advantage of today's silicon technology.

Programmable Device Usage on the Rise

Today, programmable logic devices are addressing a rapidly expanding number of applications. Shorter design cycles, lower development cost, and increasing parity of gate count, cost and performance are making FPGAs and PLDs attractive alternatives to ASIC implementation. In particular, networking and telecommunications applications are increasingly materializing in programmable devices. The fast turnaround and low entry cost of PLDs suit them ideally for such markets, characterized by fluctuating standards and narrow market windows. Now that available PLD gate counts accommodate the functionality required of these applications, programmable devices may indeed surpass ASIC as the preferred implementation method for these market segments.

At the same time, programmable devices are becoming increasingly attractive in other applications segments. Process advancements also translate into higher density PLDs, and PLDs are now a viable option even for cost sensitive, high volume production applications. More and more designers are also employing FPGA-based prototyping for low-cost, low-risk, early market entry for highly complex silicon systems.

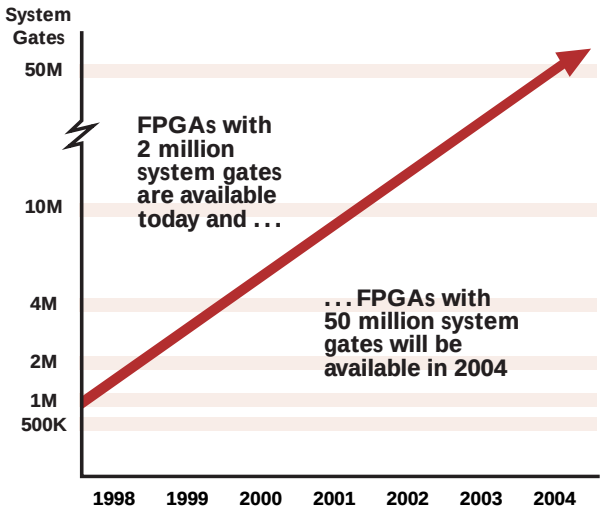

Figure 1: FPGA Speed and Gate Capacity Trend

Today's PLD design challenges

Thus, multi-million gate capacity and clock speeds approaching 200 MHz for PLD-based designs are coming into the mainstream, and the Electronic Design Automation (EDA) technology that support these applications is rapidly changing by necessity. PLD devices are now being fabricated in advanced, ultra deep submicron (UDSM) process technology, and along with the multi-million-gate IC capacity enabled by this technology comes a new and difficult design problem: interconnect-dominant delay. As witnessed when high gate count deep submicron ASIC designs first emerged, the predominance of interconnect between logic elements has a major impact on PLD device performance. Indeed, interconnect can account for as much as 70-90% of overall circuit delay as critical dimensions descend below 0.25µm.

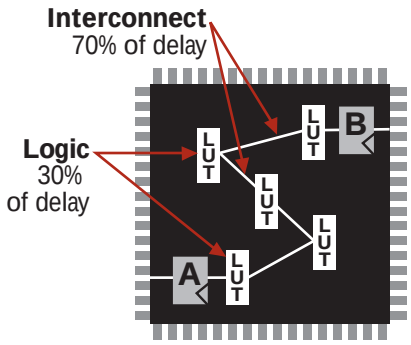

Figure 2: Interconnect vs. Logic Delay in Large ICs

Unfortunately, traditional design methodologies are ill-suited for the task of accurately accounting for interconnect effects, and design teams pay a performance penalty for this inadequacy. Synthesis-based design is imperative for high productivity design cycles these days, but performance optimization in a conventional synthesis environment is performed in the absence of physical interconnect information. Only after subsequent, time-consuming placement and routing steps is such critical information available. Because UDSM interconnect effects are significant, designers must iterate the synthesis and optimization processes with back-annotated, post-route physical information. Newly synthesized results are placed and routed, and more iterations may be necessary. The amount of time required to converge on an acceptable timing solution on UDSM circuits is neither cost nor time efficient. The extra time and uncertainty inherent in iterations between front and back-end design stages jeopardizes a fundamental premise of PLD-based implementation time to market. The only alternative to such iterations is to overestimate physical effects and produce an overly-conservative design that is more costly and fails to take full advantage of the FPGA.

Some designers attempt to address the inadequacies of traditional PLD design technology by employing more "physically intelligent" ASIC synthesis tools to their programmable designs, but these tools lack sufficient intelligence of PLD architectures. While ASIC synthesis technology today does indeed account for physical effects much better than it has in the past, these tools are inadequate for PLD design. Not only are the physical interconnections of PLD devices formed differently, but the rules for how those connections are made as well as the electrical characteristics of those connections are significantly different than those of ASIC devices. Moreover, PLD interconnect characteristics differ substantially between vendors, and standardized physical models such as those used by ASIC design tools are infeasible. Neither ASIC synthesis nor ASIC interconnect models are adequate for designing todays large programmable devices. Creating physical constraints for synthesis using traditional methods is exceedingly difficult, or even impossible.

Team design exposes yet another difficulty of traditional design methods when applied to large designs. Team design is essential to meeting design turnaround objectives for today's complex circuits. Unfortunately, this approach typically involves dividing a design into sub-circuits in very early design stages, before physical effects are comprehended. Design modules are most frequently delineated according to logical heirarchy with little or no consideration for physical effects. Only after place and route does the design team have information about the location of critical paths, and these paths may cross the boundaries of several or all modules. This makes critical path optimization extremely difficult. The team must either re-partition the design to better isolate the critical path or work through a cumbersome and tedious process of floorplanning the design at the gate level.

Improving Synthesis Technology

Thus, improving the synthesis process is pivotal to EDA advancement in the UDSM era. The traditional synthesis scenario in which HDL is compiled to gates, mapped into the chosen technology and optimized must be given intelligence about interconnect effects. Only when this happens is it possible to productively turn out designs with acceptable performance levels. Achieving a design methodology that brings accurate information about how logic elements will be physically connected into the synthesis process entails the use of new synthesis technology: physical synthesis with integral physical optimization capabilities.

Physical Synthesis refers to the utilization of physical placement information during the synthesis process. Only when physical effects are taken into account during synthesis is it possible to glean the most performance from UDSM devices and still hit target market windows. Physical synthesis is a superior approach to back-annotation of post-place and route timing and gate-level resynthesis because it actually restructures the logic of a design based on physical characteristics and creates placement. Intelligent reordering of logic with clear mapping to the logic and gate levels represents a vast improvement over the traditional "card shuffling" approach of back-annotation. Such restructuring reduces or eliminates iterations between synthesis and place and route, and improves not only productivity but also design performance.

We believe *physical optimization* (simultaneous placement and optimization) represents a major technology leap in the design of high performance UDSM PLD devices. Physical optimization enhances results of the synthesis process by using physical design characteristics to affect the actual topology of the circuit. This technology makes it possible to communicate such physical constraint information as RTL placement and regional interconnect delay to the synthesis process and thus to physically optimize a design prior to time-consuming placement and routing stages.

Amplify™ Physical Optimizer™—Bringing Physical Synthesis to PLD Design

Synplicity, a leading provider of advanced EDA solutions for programmable logic, addresses the shortcomings of conventional EDA technology with what Synplicity believes is the first and only physical synthesis tool designed specifically for PLDs, Amplify Physical Optimizer. Amplify combines innovative new physical optimization technology with the production-proven logic synthesis algorithms of Synplicty's Synplify synthesis environment, tuned over the years specifically for PLDs. Amplify consists of a hierarchical optimization engine that leverages topology and placement knowledge to produce significantly improved netlist results after physical optimization synthesis. Advanced optimization techniques, embedded intelligence specific to PLD architectures and high productivity features such as an intuitive graphical user interface (GUI) are key aspects of the Amplify environment that help designers achieve the highest performance possible in short design cycles.

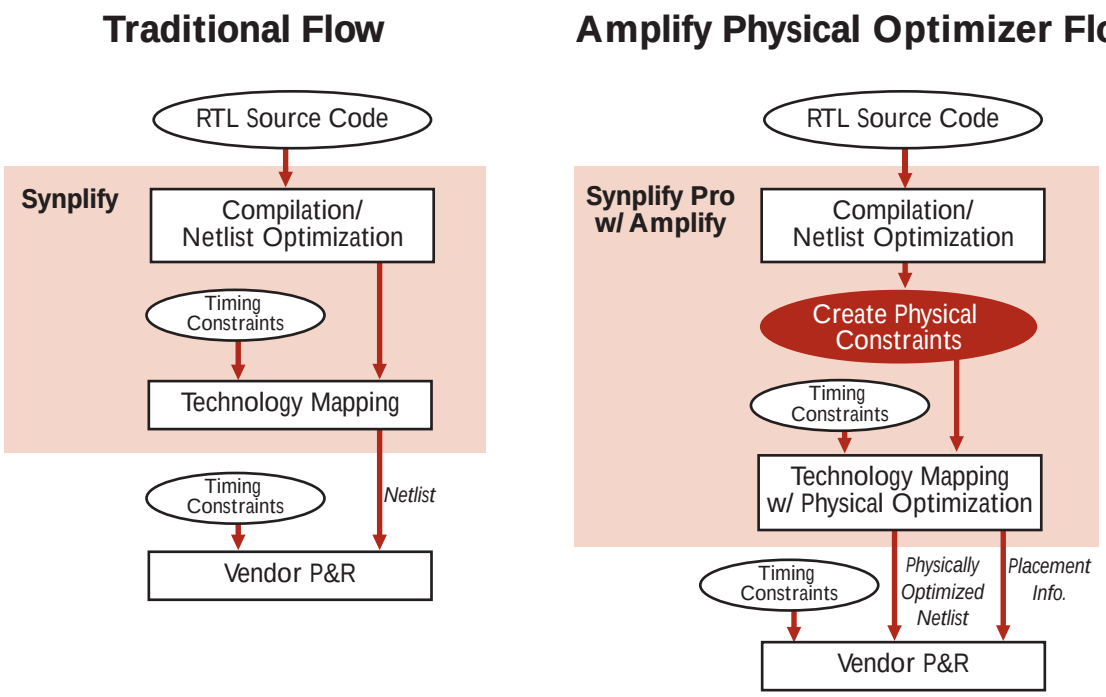

Figure 3: Traditional Flow vs. Amplify Physical Optimizer Flow

RT-Level Optimization Solves the Timing Estimation Problem

Using physical constraint information based on RTL objects, Amplify optimizes a design for performance during synthesis and is designed to eliminate iterations and timing estimation inaccuracies of traditional flows. Amplify Physical Optimizer leverages the designer's knowledge of the circuit and familiarity with post-compiled RTL blocks to create a set of physical constraints rapidly at a higher level of abstraction. Instead of globally applying these constraints to a design, Amplify uses its knowledge of PLD architectures to more predictably estimate timing within defined physical regions (for Xilinx Virtex devices) or rows/megalabs (for Altera Flex / APEX devices). With this physical information, Amplify derives more accurate timing estimations and uses them to perform additional optimization techniques during synthesis, producing a more highly optimized circuit in fewer iterations.

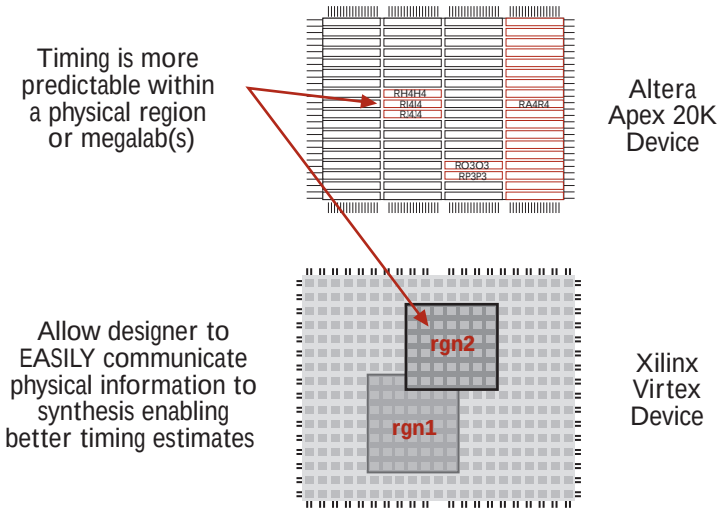

Figure 4: Improving Timing Using Physical Constraints

Abstract Floor Plan: A Better Way to Manage Design Performance

Amplify provides an easy way to optimize and control design placement. Not to be confused with traditional gate-level floorplanners, the Amplify environment operates on the design at the RT level during synthesis. Understanding the designer's knowledge of the code through timing and physical constraints, Amplify enables physical placement of the design at the RT level. After physical constraint creation, the resulting physical hierarchy is optimized for performance and also provides an additional level of designer control for achievement of performance objectives. The performance-driven partitioning and placement embodied in an abstract floor plan enables more effective team design by making it possible to isolate the critical path in a single region. Timing estimation accuracy for a design thus partitioned is vastly improved over that of traditional global estimation.

Amplify also supports designs that contain multiple clocks. After establishing the post-place and route critical path for all clock domains that fail to meet performance requirements, the designer may physically place the RTL source code into regions (Altera Mega Lab-based regions, Xilinx custom regions). For example, two unrelated clock domain critical paths can be partitioned during synthesis so that they are located in two separate physical regions. In this way, Amplify creates physical or regional corridors for clock domains, avoids resource contention and makes subsequent place and route easier. Designers can also save valuable time by avoiding logic assignment in clock domains that meet performance requirements after the first pass through Amplify and place and route.

Throughout the process, the designer maintains a high level of control and flexibility. Amplify allows the designer to change the physical hierarchy (by allowing the creation of regions) without having to actually change the RTL source code. Synplify compilers generate stable, predictable names that allow for incremental changes in the RTL source that result in incremental changes to the netlist. This allows RTL functions to be scalable and provides flexibility for control logic changes without impacting the existing floor plan. For example, a 4 Bit Adder can be upgraded to an 8 Bit Adder, or control logic can be changed to add an extra branch condition. Amplify also provides a project-based flow with the option to rapidly create multiple implementations with individual physical constraints targeting the latest PLD architectures. This capability makes it possible to explore multiple design implementation options (saved in separate folders) using a single set of source code.

Facilitating Team Design

Using Amplify Physical Optimizer, high performance PLD designers can employ a team-design approach with great success. Amplify's approach enables teams to divide design responsibilities without advance knowledge of a design's critical path. This way, the team is helped to avoid the perils and inefficiencies of logically partitioning a design. These teams can achieve performance goals by taking advantage of Amplify techniques, shown to yield a performance improvement ranging from 10% to 45% for Altera Apex and Xilinx Virtex architectures.

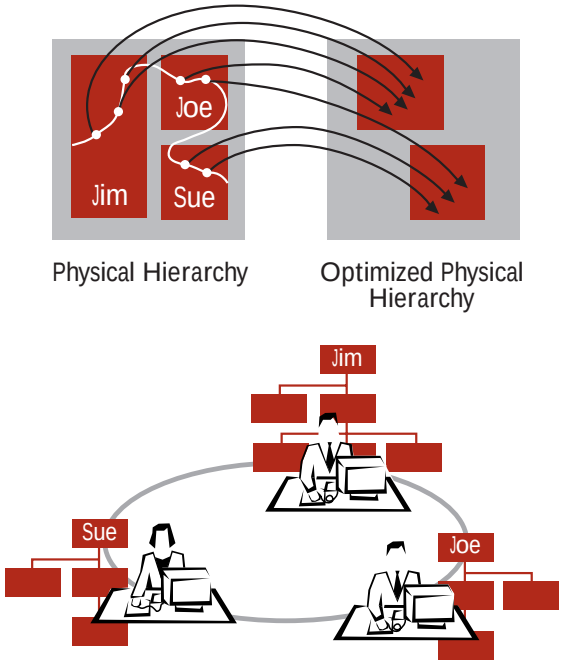

Figure 5: Optimal Team Design

GUI Unleashes Productivity

Amplify is built on a state-of-the-art graphical user interface. The interface provides several key features that facilitate rapid HDL coding, support team design and gives designers a high degree of control over the RTL synthesis process. Using this intuitive interface, even historically difficult tasks are made trivial. For example, designers can easily enter physical constraints by "dragging and dropping" critical path elements from the RTL view to the physical constraint view in the project window. Since these constraints are not a physical floorplan in the traditional, gate-level sense, logic can move outside of a region and can be replicated. Fast area and utilization estimates are facilitated via the integrated Synplify synthesis engine. Amplify's GUI presents data at the right level of abstraction for easy physical constraint creation.

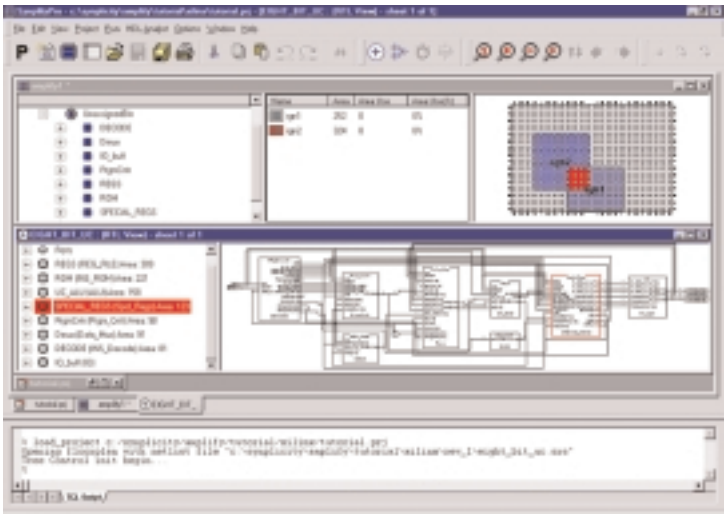

Figure 6: Amplify Graphical User Interface

The Amplify project flow is based on the ability to work with multiple projects and a new concept of "implementations" represented as Amplify revisions for a given design. The revisions referred to in the context of Amplify are not intended to take the place of external source code control software and processes. Each implementation maintains the device options, synthesis options, and project related files. Multiple implementations allow the user to modify device and synthesis controls to explore design options. Often, a designer makes tradeoffs in the size and speed of an FPGA device along with synthesis environment settings (FSM, resource sharing, etc.), and the Amplify user interface makes this process easier to use and manage. For example, to enable design exploration, a simple menu command automatically invokes Amplify to optimize all existing implementations in the active project. This allows designers to create implementations with various device and synthesis options and run Amplify in a graphical batch mode to rapidly produce results for every implementation.

Advanced Optimization Techniques

Amplify features many highly advanced optimization techniques to ensure the best possible performance for a given design. These techniques include critical path restructuring, logic tunneling, feed-through optimization, constant propagation, logic replication, I/O replication, and wire delay re-timing. Amplify differs from common synthesis tools in that it offers the designer the option to implement some or all of these optimization techniques, as his or her specific needs warrant. A sample of Amplify's optimization techniques are briefly described:

Logic Replication

Applying automatic replication during optimization is traditionally an effective means to manage high fan-out nets during synthesis. Amplify takes the replication concept one step further with algorithms that allow designer-controlled module, register, and gate level replication. Such replication minimizes critical path delay, even when interface-related logic is spread out across a chip.

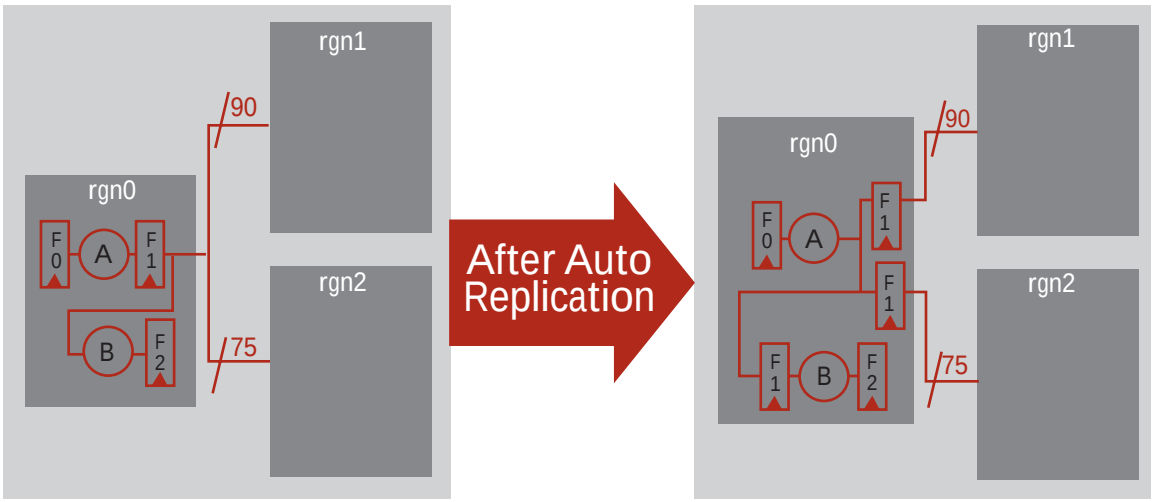

Figure 7: Timing-Based Logic Replication

Tunneling

Automatic tunneling uses boundary optimization techniques to resolve detailed timing issues. Region assignments in Amplify are considered soft when negative slack is present in the circuit and logic may migrate across region boundaries (tunneling) to improve timing along a given path. Such migration can reduce interconnect delay and thus improve speed. Tunneling is not employed when positive slack is determined along any given path in the design. In other words, tunneling is not performed if it is not needed to make timing.

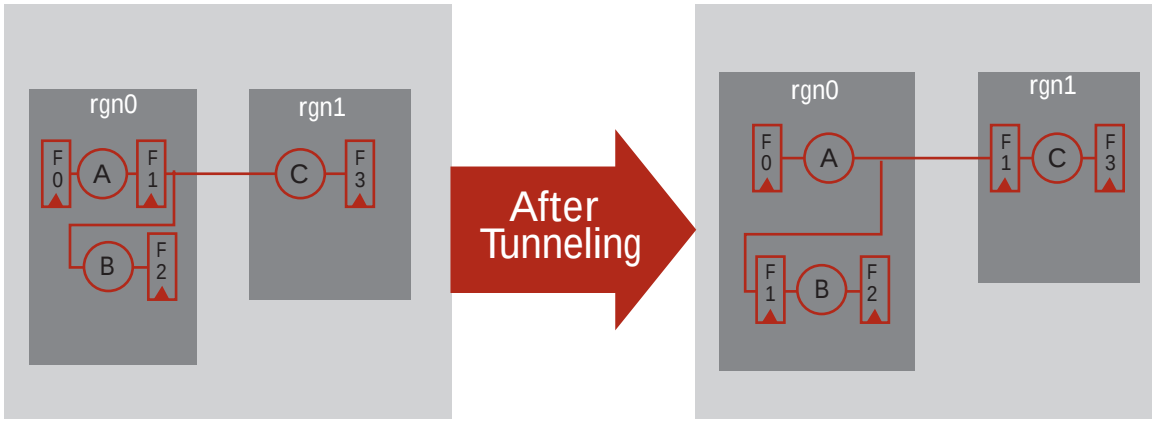

Figure 8: Tunneling

Time Budgeting

Automatic time budgeting across hierarchical design blocks eliminates the need to physically constrain the entire design. The ability to create physical constraints for a design saves time and focuses physical optimization on the most critical path. At the same time, the remaining logic is not regionally grouped with the critical path, allowing place and route to create the most direct routes and minimize delay. Automatic time budgeting across hierarchical design blocks makes it possible to physically constrain only mission critical portions of the design while Amplify manages the remaining logic.

A Multi-Pass Design Flow

A design flow that employs Amplify is a multi-pass flow. The methodology consists of creating two implementations. The first implementation is one without physical constraints that is used to establish a critical path for the design. This is accomplished by first synthesizing the design through Amplify, then running the design through place and route to determine the critical path. The second implementation uses the critical path information from the first pass to create a physically-constrained critical path and optimize the design. In this subsequent pass through Amplify, the designer creates physical constraints by placing RTL source code objects associated with the critical path into regions on the FPGA. The output of this process is a unique, physically optimized, netlist along with place and route constraint files that are based on the physical assignment of RTL source objects into regions.

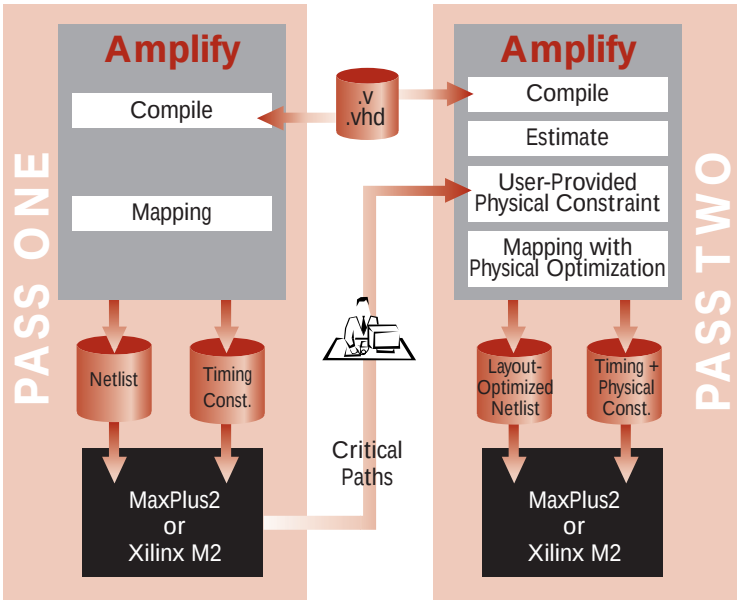

Figure 9: A Multi-Pass Design Flow

Results

Actual customer results show that designs run through Amplify exhibit a significant performance improvement over traditionally placed-and-routed netlists. These results clearly demonstrate that the benefits of Amplify are not simply the result of forward annotating placement constraints. The Amplify netlist differs significantly in structure from that of traditional synthesis alone due to optimizations that become possible when physical constraints are present. Creating physical constraints by functionally grouping logic can, in theory, improve design performance. However, it does not prevent a critical path from traversing through some or all functional modules, thereby nullifying any advantage that functional grouping may provide. Amplify combines the benefits of improved delay estimation based on physical constraints with critical path optimization techniques to deliver increased performance. Amplify merges physical characteristics such as interconnect delay into the synthesis process, thereby increasing performance.

Substantial Performance Gains

Device		Performance Improvement	
Altera	10K100	16.0%	
	10K130	27.3%	
	10K130	31.1%	
	20KE100	11.0%	
	20KE100	18.0%	
	20KE100	24.6%	
	20KE200	45.8%	
	20KE200	36.0%	
	20KE400	37.6%	
	20KE400	33.0%	
	20KE400	34.0%	
Xilinx	XCV150	10.0%	**27% average**
	XCV150	11.2%	
	XCV300	23.2%	
	XCV600	10.4%	
	XCV600	12.8%	
	XCV800	24.0%	
	XCV1000	22.2%	
	XCV1000	30.2%	
	XCV1000	29.5%	
	XCV1000	57.7%	
	XCV2000E	45.0%	

Figure 10: Design Results

Summary

The Amplify Physical Optimizer provides programmable logic designers with technology to enhance design performance, minimize impact of design changes, increase productivity, and reduce time to market. Specifically, Amplify:

- Restructures logical hierarchies to produce a structural netlist optimized for physical design.
- Improves timing estimation by comprehending timing constraints and technology specific mapping in an abstract floor plan (physical constraints) generated during synthesis.
- Improves upon a user-generated abstract floor plans via automated techniques such as tunneling that reduce the physical effects of interconnect on device performance.
- Optimizes design performance based upon placement information in user-generated abstract floor plans via techniques such as buffering and replication.
- Reduces design iterations and improves design results by supplying place and route with 1) a unique netlist optimized for physical design and 2) forward-annotated timing and physical constraints.

Designers wishing to meet speed requirements on large designs, employ a team design approach, improve design productivity or achieve performance objectives in a lower speed grade part can especially benefit from Amplify's capabilities. Such physical optimization tools, working in conjunction with RTL synthesis, enable the designer to achieve the full performance potential of today's high-density FPGAs.