

# Physikalische Synthese für komplexe PLDs

## Einhalten von Timing-Vorgaben ohne zeitraubende Iterationen

In komplexen PLDs geht der Großteil der Signallaufzeiten auf das Konto der Interconnect-Strukturen. Wenn der kritische Signalweg eines Designs ein FPGA in seiner ganzen Länge durchquert, verwundert es nicht, wenn das gewünschte Timing nicht eingehalten wird. Gewisse Abhilfe ist möglich, wenn eine Feinabstimmung des Timings erfolgt und die tatsächlichen Laufzeiten im Anschluss an das Routing an das Design zurückgemeldet werden. Dies setzt aber meist zahlreiche Iterationen voraus und verlängert so die Markteinführungszeiten. Die ‚physikalische Synthese‘ kann die Hauptursache vieler Timing-Probleme direkt lösen helfen.

JEFF GARRISON

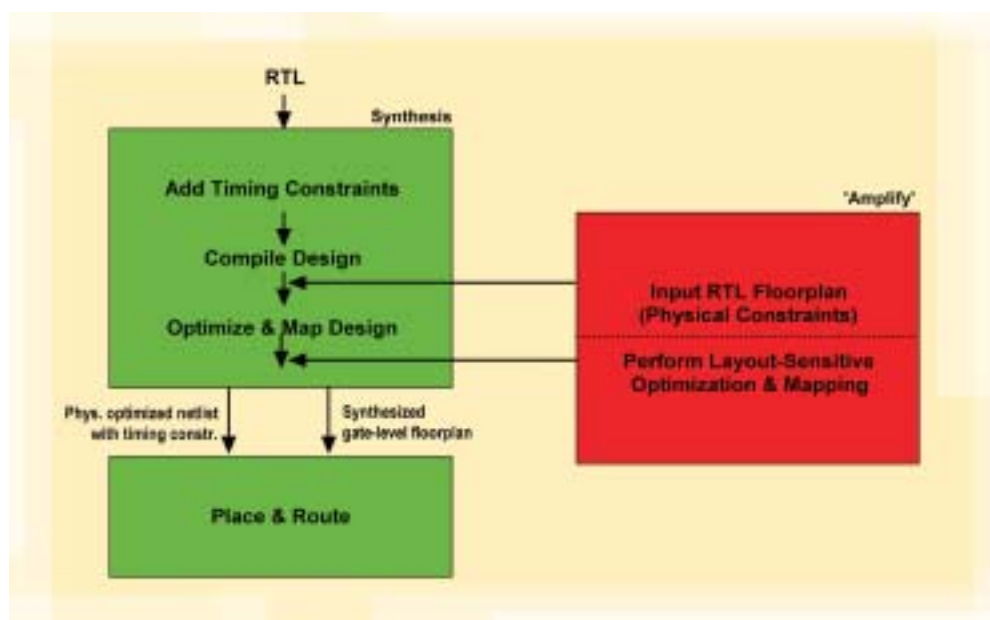


Abb. 1: Ablauf der ‚Physical Synthesis‘



Jeff Garrison ist Director of Marketing bei Synplicity, Inc.

Die physikalische Synthese bietet eine unkomplizierte Methode, über die Möglichkeiten der Logiksynthese hinaus strukturelle Optimierungen vorzunehmen und gleichzeitig die Platzierung der Logik entlang der kritischen Signalwege eines Designs zu kontrollieren. Zusätzlich besteht die Wahlmöglichkeit zwischen automatischer und interaktiver physikalischer Synthese. Die automatische Version erzielt gleichsam auf Knopfdruck eine gewisse Performance-Verbesserung, doch ein Maximum an Leistungsfähigkeit lässt sich erst mit einem interaktiven Ablauf erzielen. Hier liefert der Anwender dem Tool gewisse Vorgaben, indem er physikalische Regionen definiert und die Logik auf der Register-Transfer-Ebene (RTL) diesen Regionen zuweist. Die interaktive Methodik ist sehr intuitiv und benötigt

wenig Zeit, setzt aber voraus, dass der Benutzer mit einigen Aspekten des physikalischen Layouts des Bausteins vertraut ist.

Diese Mehrarbeit zahlt sich dadurch aus, dass deutlich weniger Iterationen benötigt werden und sich das Timing des Bausteins erheblich verbessert. Benchmarks, die die Physical-Synthesis-Software ‚Amplify Physical Optimizer‘ und die Logiksynthese-Software ‚Synplify Pro‘ von Synplicity an ungefähr 100 verschiedenen FPGAs von Altera und Xilinx miteinander verglichen haben, ergaben, dass sich das Timing durch die physikalische Synthese durchschnittlich um mehr als 20 Prozent verbessert. Während sich die Performance bei unzureichenden physikalischen Vorgaben in Einzelfällen sogar verschlechtern kann, wurden bei einigen dieser Designs Verbesserungen um bis zu 50 Prozent erreicht.

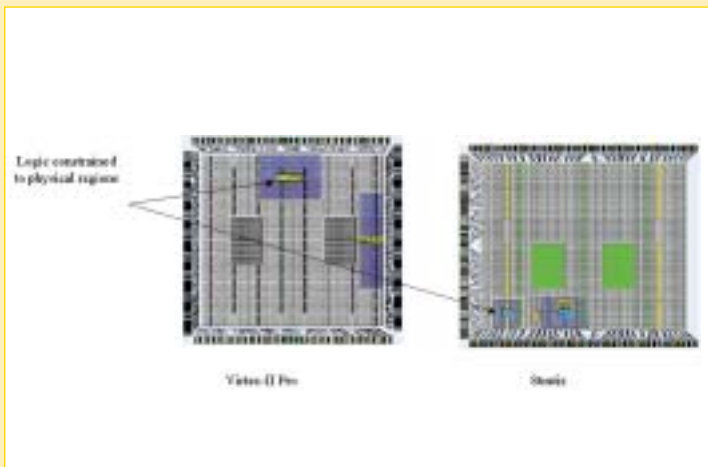


Abb. 2: ‚Amplify Physical Optimizer‘ nutzt Informationen über die Architektur des jeweiligen Bausteins zum Erarbeiten präziser Timing-Schätzungen innerhalb umgrenzter physikalischer Regionen

Auch bei Designs mit weniger ehrgeizigen Timing-Vorgaben zahlt sich die physikalische Synthese aus, denn dank der erzielten Timing-Verbesserungen kann möglicherweise eine langsamere Version des Bausteins gewählt werden. Da die Preisunterschiede zwischen den verschiedenen schnellen Versionen großer Bausteine durchaus mehrere hundert Dollar betragen können, sind auf diese Weise speziell bei hohen Stückzahlen beträchtliche Kostensenkungen möglich.

Auch wenn sich die physikalische Synthese beim ASIC-Design als nützlich erwiesen hat, sind ASIC-orientierte Tools nicht für programmierbare Logikbausteine geeignet, denn ASICs und PLDs besitzen vollkommen verschiedene Strukturen. Abgesehen von den grundlegend anderen Interconnect-Merkmalen weisen programmierbare Bausteine eine ‚grobkörnige‘ Struktur auf, die vom Einsatz spezieller Logikoptimierungs-Verfahren profitieren (hierzu später mehr).

### Der Ablauf der physikalischen Synthese

Synplicity empfiehlt für komplexe programmierbare Logikbausteine eine zweiteilige Vorgehensweise bei der Logiksynthese:

- ▶ Wenn die nötige Timing-Performance mit dem Logiksynthese-Tool nicht erreicht wird, kann der Anwender die vollautomatische physikalische Synthese probieren. Für den automatischen Ablauf sind im Falle der Amplify-Software keine zusätzlichen Restriktionen (Constraints) erforderlich, und da Amplify für die Synthese einiger hunderttausend Gatter nur wenige Minuten benötigt, liegen die Ergebnisse umgehend vor. Diese automatische Methodik ist sinnvoll, wenn eine Performance-Steigerung von ungefähr 10 Prozent ausreicht.
- ▶ Stellt sich das gewünschte Timing auch mit dem automatischen Ablauf nicht ein, kann der Anwender auf die interaktive physikali-

sche Synthese ausweichen. Hierzu wird im Anschluss an die normale Logiksynthese ein Place-and-Route-Arbeitsgang eingefügt, um die kritischen Signalwege des Designs zu ermitteln. Auf dieser Grundlage wird ein mit physikalischen Restriktionen versehener kritischer Signalweg definiert und das Design anschließend mit der Physical-Synthesis-Software neu synthetisiert. In diesem zweiten Arbeitsgang stellt der Designer physikalische Restriktionen auf, indem er die Logik des kritischen Signalwegs bestimmten physikalischen Regionen auf dem FPGA zuweist. Als Resultat dieses Prozesses entsteht eine physisch optimierte Netzliste, die mit einer detaillierten Platzierung für die kritischen Signalwege verbunden ist und an die Place-and-Route-Software weitergegeben wird.

Aus dem Blickwinkel des Designers liegt der Hauptunterschied zwischen beiden Abläufen im Hinzufügen der physikalischen Restriktionen für ‚Amplify Physical Optimizer‘ (in Abb. 1 rot hervorgehoben). Auf der Grundlage dieser physikalischen Constraints können die layout-sensitiven Mapping-Algorithmen der Optimizer während der Synthese eine physikalische Optimierung des Designs im Interesse maximaler Performance vornehmen. Die Software nutzt dabei das Wissen des Designers über die Schaltung und gibt ihm die Möglichkeit, auf der RT-Ebene schnell und einfach einen Satz physikalischer Restriktionen zu definieren.

Anstatt diese Restriktionen nun global auf das gesamte Design anzuwenden, nutzt Amplify Physical Optimizer die Architektur des jeweiligen Bausteins zum Erarbeiten präziser Timing-Schätzungen innerhalb umgrenzter physikalischer Regionen. Sobald die tatsächliche Platzierung der Logik bekannt ist, ergibt sich ein wesentlich besser vorhersagbares Timing (Abb. 2).

In der Praxis erarbeitet der Designer die physikalischen Restriktionen mit Hilfe einer grafischen Benutzeroberfläche, die nicht wie traditionelle Floorplanning-Tools mit Gattern, sondern mit RT-Objekten arbeitet. Im Drag-and-Drop-Verfahren zieht der Designer die Logik aus dem Modulhierarchie-Browser oder der RT-Ansicht in die physikalische Darstellung des Bausteins. Die integrierte Synthese-Engine der Synplify-Software liefert daraufhin zügig Schätzwerte zur Fläche und Performance, die Hilfestellung beim Festlegen der Constraints leisten und dem Anwender mitteilen, wann die Logikressourcen in einer bestimmten Region aufgebraucht sind.

Der Designer kann übrigens durch Einrichten von Regionen die physikalische Hierarchie seines Designs verändern, ohne Modifikationen am RT-Quellcode vorzunehmen. Dies kann den Zeitaufwand für ein Designprojekt verringern, da Performance-Steigerungen ohne

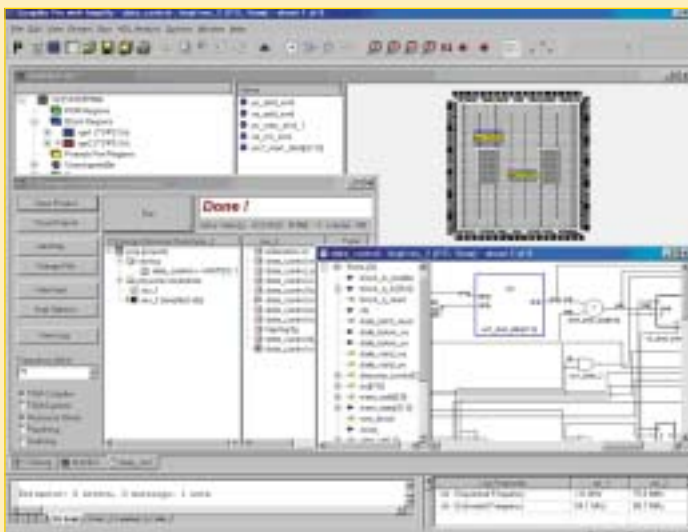


Abb. 3: Der Anwender von ‚Amplify Physical Optimizer‘ kann durch Einrichten von Regionen die physikalische Hierarchie seines Designs verändern, ohne Modifikationen am RT-Quellcode vornehmen zu müssen.

die sonst häufig nötigen zeitraubenden Änderungen am HDL-Code möglich sind (Abb. 3).

### Gezielte Optimierung für PLD-Architekturen

Nach Eingabe der physikalischen Restriktionen nimmt Amplify Physical Optimizer gleichzeitig das Technologie-Mapping und die physikalische Optimierung vor. Hierbei können spezielle PLD-Optimierungstechniken das Timing verbessern. Amplify Physical Optimizer wendet diese Techniken automatisch an,

doch kann der Designer auf Wunsch angeben, ob alle oder nur bestimmte Techniken zum Einsatz kommen sollen. In einigen Fällen kann der Designer zudem definieren, wie die Techniken genutzt werden. Ebenso wie bei allen anderen Manipulationen durch Amplify Physical Optimizer werden auch hier im Zuge der Optimierung keine Änderungen am RT-Quellcode vorgenommen.

Zu den Optimierungstechniken gehören die Logik-Duplizierung, die Logik-Tunnelung, die Zeit-Budgetierung über verschiedene hierarchische Blöcke hinweg, die Restrukturierung kritischer Signalwege, die Feed-through-Opti-

mierung, die Weiterleitung von Konstanten, die I/O-Duplizierung und das Re-Timing von Leitungslaufzeiten. Drei Beispiele sollen zeigen, wie physikalische Optimierung das Timing verbessern können.

#### Duplizieren von Logik

Das automatische Duplizieren von Logik während der Optimierung wird von Synthesengines schon lange genutzt, um Leitungen mit hohem Fan-out zu bewältigen. Wenn ein Interface-Logikblock drei Logikmodule in verschiedenen physikalischen Regionen eines

## Anzeige

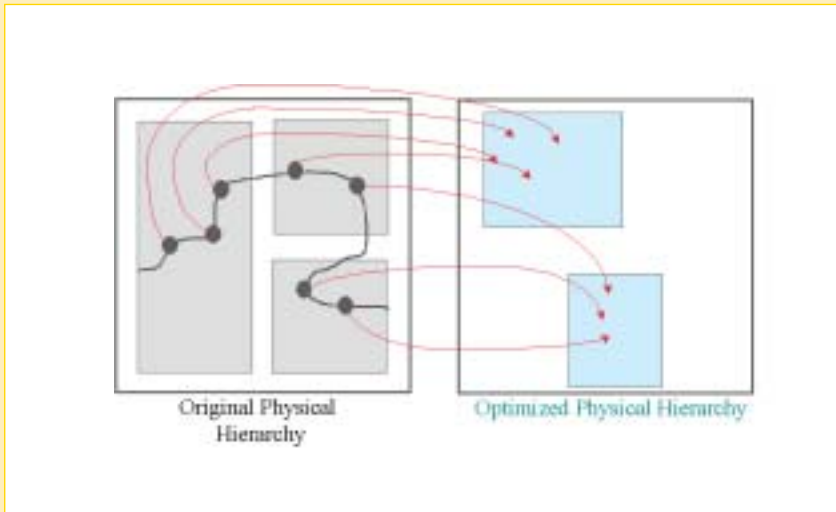


Abb. 4: Die Partitionierung eines Designs auf einer hohen Abstraktionsebene kann die Genauigkeit der Timing-Abschätzung gegenüber einer traditionellen globalen Schätzung verbessern helfen

Bausteins ansteuert, kann das Kopieren der Interface-Logik in diese drei Bereiche hinein das Timing verbessern. Als Fortführung dieses Konzepts ermöglicht Amplify Physical Optimizer dem Designer die Kontrolle der Duplizierung auf der Modul-, Register- und Gatter-Ebene. Mit dieser fein abgestuften Einflussmöglichkeit kann der Designer die Laufzeiten kritischer Signalwege selbst dann minimieren, wenn die mit dem Interface zusammenhängende Logik auf den gesamten Chip verteilt ist.

### Logik-Tunnelung

Ein zweites Optimierungsbeispiel ist die Logik-Tunnelung. Liegt in einem Timing-Pfad ‚negativer Slack‘ (Unterschreitung des angesetzten Spielraums) vor, so veranlasst Amplify Physical Optimizer zur Verbesserung des Timings automatisch eine Verlagerung der Logik (Tunnelung) über Bereichsgrenzen hinweg. Eine solche Migration kann die Interconnect-Laufzeiten verringern und die Geschwindigkeit verbessern helfen.

### Zeit-Budgetierung

Als drittes Beispiel vermeidet die Zeit-Budgetierung über hierarchische Designblöcke hinweg die Notwendigkeit, Restriktionen für ein komplettes Design aufzustellen. Während der Designer seine Constraints auf die kritischen Bereiche beschränkt, nimmt sich der Optimizer der übrigen Logik an. Diese Vorgehensweise spart Zeit, da die physikalische Optimierung auf die kritischsten Signalwege beschränkt bleibt. Da die verbleibende Logik gleichzeitig nicht regional mit dem kritischen Signalweg verknüpft ist, ist es möglich, in der Place-and-Route-Phase die direktesten Verbindungen zu finden und die Signallaufzeit zu minimieren.

### Vorteile der Arbeit auf der RT-Ebene

Amplify Physical Optimizer erlaubt dem Designer das Zuweisen physikalischer Restriktionen nach dem Compilieren des HDL-Quellcodes, aber noch vor dem Mapping und somit in einer Phase, in der maximal auf die Performance eingewirkt werden kann. Mit dem Physical-Constraints-Editor dieser Syntheseumgebung kann der Designer ebenfalls Einfluss auf die physikalische Hierarchie des Designs nehmen.

Hierdurch lässt sich die beim Implementieren isolierter Module von verschiedenen Designern möglicherweise verlorengegangene Performance einer Schaltung wiederherstellen. Verschiedene Abschnitte eines Designs können ohne zu wissen, wo sich der kritische Signalweg befindet, an verschiedene Designer vergeben werden. Der kritische Pfad kann durch mehrere, von unterschiedlichen Designern erstellte Module verlaufen und lässt sich zum Zweck der Optimierung aber dennoch in einen gemeinsamen Bereich legen. Diese Fähigkeit zur Partitionierung eines Designs auf einer hohen Abstraktionsebene kann die Genauigkeit der Timing-Abschätzung gegenüber einer traditionellen globalen Schätzung verbessern helfen (Abb. 4).

Einige Optimierungsverfahren unterstützen außerdem die Maximierung der Design-Performance über Module verschiedener Designer hinweg. Logik-Duplizierung und -Tunnelung sind Beispiele hierfür. Dank dieser Fähigkeiten können mehrere Designer unabhängig vom physikalischen Layout tätig werden, bis die Logik für die abschließende Implementierung zusammengefügt wird. Diese Art team-basierten Designs wird immer wichtiger, da die Gatter-Anzahl der PLDs von bisher ein bis zwei Millionen mittlerweile auf einige zehn Millionen wächst. Die Unterstützung für das Team-Design ist einer der zahlreichen

Vorteile des in Amplify Physical Optimizer verwirklichten RTL/Physical-Management-Konzepts.

### Inkrementelles Design

Ein weiterer Vorzug der Physical-Synthesis-Methodik von Amplify Physical Optimizer besteht darin, dass der Designer schrittweise Änderungen an seinem Design vornehmen und den Ort dieser Modifikationen isolieren kann. Hiermit wird verhindert, dass nach einer kleinen Änderung das komplette Design neu bearbeitet werden muss. Die ‚MultiPoint‘-Technologie von Synplicity bietet somit eine Lösung für das inkrementelle Design. Durch Zuweisen von ‚Compile-Points‘ kann der Benutzer festlegen, welche Module neu zu synthetisieren sind und welche unverändert bleiben. Bei alledem ist die MultiPoint-Technologie ausreichend intelligent, um echte Funktionsänderungen zu erkennen und zu bestimmen, welche Module neu synthetisiert werden müssen. Zum Beispiel veranlasst die Verwendung einer anderen HDL-Syntax für die gleiche Funktion keine erneute Compilierung. Ohne Abstriche an der Leistungsfähigkeit resultiert dieses Konzept in einer deutlichen Beschleunigung des Designprozesses.

### Constraints bei Designänderungen

Mit Amplify Physical Optimizer können unveränderte physikalische Constraints verwendet werden, auch wenn am Design zahlreiche Arten von Änderungen vorgenommen werden. Zum Beispiel kann ein 4-Bit-Addierer zu einem 8-Bit-Addierer erweitert oder Steuerungslogik durch eine zusätzliche Verzweigung ergänzt werden, ohne dass neue physikalische Constraints erstellt werden müssen. Bei Floorplanning-Tools dagegen müssten die Restriktionen für alle elementaren Logikfunktionen mühevoll neu definiert werden.

Da sich mit Amplify Physical Optimizer rasch Modifikationen an den physikalischen Constraints auf der RT-Ebene vornehmen lassen und die Synthese weniger Zeit beansprucht als bei anderen Synthese-Tools, kann der Designer mit wenig Zeitaufwand eine Vielzahl von PLD-Implementierungen ausprobieren. Diese schnelle Rückmeldung hilft dem Designer, mehr Designimplementierungen zu erkunden und letztlich schneller, als es mit der Logiksynthese allein möglich wäre, zu einer Lösung mit optimaler Performance zu kommen.

Beitrag als PDF im Internet:

[www.publish-industry.net](http://www.publish-industry.net)

more @ click DV63253

