



Synplicity® FPGA Synthesis

Synplify Premier Quick Start Guide for Xilinx

July 2008

Synplicity, Inc.
600 West California Avenue
Sunnyvale, CA 94086, USA
(U.S.) +1 408 215-6000 direct
(U.S.) +1 408 990-0263 fax
www.synplicity.com

Preface

Disclaimer of Warranty

Synplicity, Inc. makes no representations or warranties, either expressed or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose of for any indirect, special or consequential damages.

Copyright Notice

Copyright © 2008 Synplicity, Inc. All Rights Reserved.

Synplicity software products contain certain confidential information of Synplicity, Inc. Use of this copyright notice is precautionary and does not imply publication or disclosure. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the prior written permission of Synplicity, Inc. While every precaution has been taken in the preparation of this book, Synplicity, Inc. assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

Trademarks

Synplicity, the Synplicity logo, “Simply Better Results”, Amplify, Amplify FPGA, Behavior Extracting Synthesis Technology, Certify, HDL Analyst, Identify, SCOPE, Synplify, Synplify ASIC, and Synplify Pro are registered trademarks of Synplicity, Inc. BEST, IICE, MultiPoint, Physical Analyst, and System Designer are trademarks of Synplicity, Inc. All other names mentioned herein are trademarks or registered trademarks of their respective companies.

Restricted Rights Legend

Government Users: Use, reproduction, release, modification, or disclosure of this commercial computer software, or of any related documentation of any kind, is restricted in accordance with FAR 12.212 and DFARS 227.7202, and further restricted by the Synplicity Software License Agreement. Synplicity, Inc., 600 West California Avenue, Sunnyvale, CA 94086, U. S. A.

Printed in the U.S.A
July 2008

Synplicity Software License Agreement

Important! READ CAREFULLY BEFORE PROCEEDING

BY INDICATING YOUR ACCEPTANCE OF THE TERMS OF THIS AGREEMENT, YOU (“LICENSEE”) ARE REPRESENTING THAT YOU HAVE THE RIGHT AND AUTHORITY TO LEGALLY BIND YOURSELF OR YOUR COMPANY, AS APPLICABLE, AND CONSENTING TO BE LEGALLY BOUND BY ALL OF THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL THESE TERMS DO NOT INSTALL OR USE THE SOFTWARE, AND RETURN THE SOFTWARE TO THE LOCATION OF PURCHASE FOR A REFUND. This is a legal agreement governing use of the software program provided by Synplicity, Inc. (“Synplicity”) to you (the “SOFTWARE”). The term “SOFTWARE” also includes related documentation (whether in print or electronic form), any authorization keys, authorization codes, and license files, and any updates or upgrades of the SOFTWARE provided by Synplicity, but does not include certain “open source” software licensed by third party licensors and made available to you by Synplicity under the terms of such third party licensor’s license (such as software licensed under the General Public License (GPL)) (“Third Party Software”). If Licensee is a participant in the University Program or has been granted an Evaluation License or Subscription License, then some of the following terms and conditions may not apply (refer to the sections entitled, respectively, **Evaluation License** and **Subscription License**, below).

License. Synplicity grants to Licensee a non-exclusive right to install the SOFTWARE and to use or authorize use of the SOFTWARE by up to the number of nodes for which Licensee has a license and for which Licensee has the security key(s) or authorization code(s) provided by Synplicity or its agents for the purpose of creating and modifying Designs (as defined below). If Licensee has obtained the SOFTWARE under a node-locked license, then a “node” refers to a specific machine, and the SOFTWARE may be installed only on the number of “nodes” or machines authorized, must be used only on the machine(s) on which it is installed, and may be accessed only by users who are physically present at that node or machine. A node-locked license may only be used by one user at a time running one instance of the software at a time. If Licensee has obtained the SOFTWARE under a “floating” license, then a “node” refers to a concurrent user or session, and the SOFTWARE may be used concurrently by up to the number of users or sessions indicated. All SOFTWARE must be used within the country for which the systems were licensed and at Licensee's Site (contained within a one kilometer radius); however, if Licensee has a floating license then remote use is permitted by employees who work at the site but are temporarily telecommuting to that same site from less than 50 miles away (for example, an employee who works at a home office on occasion), but the maximum number of concurrent sessions or nodes still applies. In addition, Synplicity grants to Licensee a non-exclusive license to copy and distribute internally the documentation portion of the SOFTWARE in support of its license to use the program portion of the SOFTWARE. For purposes of this Agreement the “Licensee’s Site” means the location of the server on which the SOFTWARE resides, or when a server is not required, the location of the client computer for which the license was issued.

Evaluation License. If Licensee has obtained the SOFTWARE pursuant to an evaluation license, then, in addition to all other terms and conditions herein, the following restrictions apply: (a) the license to the SOFTWARE terminates after 20 days (unless otherwise agreed to in writing by Synplicity); and (b) Licensee may use the SOFTWARE only for the sole purpose of internal testing and evaluation to determine whether Licensee wishes to license the SOFTWARE on a commercial basis. Licensee shall not use the SOFTWARE to design any integrated circuits for production or pre-production purposes or any other commercial use including, but not limited to, for the benefit of Licensee's customers. If Licensee breaches any of the foregoing restrictions, then Licensee shall pay to Synplicity a license fee equal to Synplicity's perpetual list price plus maintenance for the commercial version of the SOFTWARE.

Subscription (Time-Based) License. If Licensee has obtained a Subscription License to the SOFTWARE, then, in addition to all other terms and conditions herein, the following restrictions apply: (a) Licensee is authorized to use the SOFTWARE only for a limited time (which time is indicated on the quotation or in the purchase confirmation documents); (b) Licensee's right to use the SOFTWARE terminates on the date the subscription term expires as set forth in the quotation or the purchase confirmation documents, unless Licensee has renewed the license by paying the applicable fees.

Project Based License. If Licensee has obtained a Project-Based License to the SOFTWARE, in addition to all other terms and conditions herein, the terms of Exhibit A will apply.

Copy Restrictions. This SOFTWARE is protected by United States copyright laws and international treaty provisions and Licensee may copy the SOFTWARE only as follows: (i) to directly support authorized use under the license, and (ii) in order to make a copy of the SOFTWARE for backup purposes. Copies must include all copyright and trademark notices.

Use Restrictions. This SOFTWARE is licensed to Licensee for internal use only. Licensee shall not (and shall not allow any third party to): (i) decompile, disassemble, reverse engineer or attempt to reconstruct, identify or discover any source code, underlying ideas, underlying user interface techniques or algorithms of the SOFTWARE by any means whatever, or disclose any of the foregoing; (ii) provide, lease, lend, or use the SOFTWARE for timesharing or service bureau purposes, on an application service provider basis, or otherwise circumvent the internal use restrictions; (iii) modify, incorporate into or with other software, or create a derivative work of any part of the SOFTWARE; (iv) disclose the results of any benchmarking of the SOFTWARE, or use such results for its own competing software development activities, without the prior written permission of Synplicity; or (v) attempt to circumvent any user limits, maximum gate count limits or other license, timing or use restrictions that are built into the SOFTWARE.

Transfer Restrictions/No Assignment. The SOFTWARE may only be used under this license at the designated locations and designated equipment as set forth in the license grant above, and may not be moved to other locations or equipment or otherwise transferred without the prior written consent of Synplicity. Any permitted transfer of the SOFTWARE will require that Licensee executes a "Software Authorization Transfer Agreement" provided by Synplicity. Further, Licensee shall not sublicense, or assign this Agreement or any of the rights or licenses granted under this Agreement, without the prior written consent of Synplicity.

Security. Licensee agrees to take all appropriate measures to safeguard the SOFTWARE and prevent unauthorized access or use thereof. Suggested ways to accomplish this include: (i) implementation of firewalls and other security applications, (ii) use of FLEXlm options file that restricts access to the SOFTWARE to identified users; (iii) maintaining and storing license information in paper format only; (iv) changing TCP port numbers every three (3) months; and (v) communicating to all authorized users that use of the SOFTWARE is subject to

the restrictions set forth in this Agreement.

Ownership of the SOFTWARE. Synplicity retains all right, title, and interest in the SOFTWARE (including all copies), and all worldwide intellectual property rights therein. Synplicity reserves all rights not expressly granted to Licensee. This license is not a sale of the original SOFTWARE or of any copy.

Ownership of Design Techniques. “Design” means the representation of an electronic circuit or device(s), derived or created by Licensee through the use of the SOFTWARE in its various formats, including, but not limited to, equations, truth tables, schematic diagrams, textual descriptions, hardware description languages, and netlists. “Design Techniques” means the data, circuit and logic elements, libraries, algorithms, search strategies, rule bases, techniques and technical information incorporated in the SOFTWARE and employed in the process of creating Designs. Synplicity retains all right, title and interest in and to Design Techniques incorporated in the SOFTWARE, including all intellectual property rights embodied therein, provided that to the extent any Design Techniques are included as part of or embedded within Licensee’s Designs, Synplicity grants Licensee a personal, non exclusive, nontransferable license to reproduce the Design Techniques and distribute such Design Techniques solely as incorporated into Licensee’s Designs and not on a standalone basis. Additionally, Licensee acknowledges that Synplicity has an unrestricted, royalty-free right to incorporate any Design Techniques disclosed by Licensee into its software, documentation and other products, and to sublicense third parties to use those incorporated design techniques.

Protection of Confidential Information. “Confidential Information” means (i) the SOFTWARE, in object and source code form, and any related technology, idea, algorithm or information contained therein, including without limitation Design Techniques, and any trade secrets related to any of the foregoing; (ii) either party's product plans, Designs, costs, prices and names; non-published financial information; marketing plans; business opportunities; personnel; research; development or know-how; (iii) any information designated by the disclosing party as confidential in writing or, if disclosed orally, designated as confidential at the time of disclosure and reduced to writing and designated as confidential in writing within thirty (30) days; and (iv) the terms and conditions of this Agreement; provided, however that “Confidential Information” will not include information that: (a) is or becomes generally known or available by publication, commercial use or otherwise through no fault of the receiving party; (b) is known and has been reduced to tangible form by the receiving party at the time of disclosure and is not subject to restriction; (c) is independently developed by the receiving party without use of the disclosing party's Confidential Information; (d) is lawfully obtained from a third party who has the right to make such disclosure; and (e) is released for publication by the disclosing party in writing.

Each party will protect the other's Confidential Information from unauthorized dissemination and use with the same degree of care that each such party uses to protect its own like information. Neither party will use the other's Confidential Information for purposes other than those necessary to directly further the purposes of this Agreement. Neither party will disclose to third parties the other's Confidential Information without the prior written consent of the other party.

Open Source Software. The SOFTWARE may be delivered with software that is subject to open source licensing terms (“Open Source Software”) which are available at http://www.synplicity.com/products/license_agreement.html. If the Open Source Software license also requires source code to be made available, Licensee may reference <http://www.synplicity.com/products/opensource.html> for information on how to obtain such source code. Licensee agrees that all Open Source Software shall be and shall remain subject to the terms and conditions under which it is provided. The Open Source Software is provided “AS IS,” WITHOUT ANY WARRANTY OF ANY KIND, AND SYNPLICITY FURTHER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO OPEN SOURCE SOFTWARE,

INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER SYNPLICITY NOR THE LICENSORS OF OPEN SOURCE SOFTWARE SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE ECLIPSE SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Copyrights to the Open Source Software are held by the copyright holders indicated in the copyright notices in the corresponding source files.

Termination. Synplicity may terminate this Agreement immediately if Licensee breaches any provision, including without limitation, failure by Licensee to implement adequate security measures as set forth above. Upon notice of termination by Synplicity, all rights granted to Licensee under this Agreement will immediately terminate, and Licensee shall cease using the SOFTWARE and return or destroy all copies (and partial copies) of the SOFTWARE and documentation.

Limited Warranty and Disclaimer. Synplicity warrants that the program portion of the SOFTWARE will perform substantially in accordance with the accompanying documentation for a period of 90 days from the date of receipt. Synplicity's entire liability and Licensee's exclusive remedy for a breach of the preceding limited warranty shall be, at Synplicity's option, either (a) return of the license fee, or (b) providing a fix, patch, work-around, or replacement of the SOFTWARE. In either case, Licensee must return the SOFTWARE to Synplicity with a copy of the purchase receipt or similar document. Replacements are warranted for the remainder of the original warranty period or 30 days, whichever is longer. Some states/jurisdictions do not allow limitations, so the above limitation may not apply. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES OR CONDITIONS, EITHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, ARE MADE BY SYNPLICITY OR ITS LICENSORS WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING DOCUMENTATION, AND SYNPLICITY EXPRESSLY DISCLAIMS ALL WARRANTIES AND CONDITIONS NOT EXPRESSLY STATED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. SYNPLICITY AND ITS LICENSORS DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET LICENSEE'S REQUIREMENTS, BE UNINTERRUPTED OR ERROR FREE, OR THAT ALL DEFECTS IN THE PROGRAM WILL BE CORRECTED. Licensee assumes the entire risk as to the results and performance of the SOFTWARE. Some states/jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply.

Limitation of Liability. IN NO EVENT SHALL SYNPLICITY OR ITS LICENSORS OR THEIR AGENTS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTIONS, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF SYNPLICITY AND/OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. FURTHER, IN NO EVENT SHALL SYNPLICITY'S LICENSORS BE LIABLE FOR ANY DIRECT DAMAGES ARISING OUT OF LICENSEE'S USE OF THE SOFTWARE. IN NO EVENT WILL SYNPLICITY OR ITS LICENSORS BE LIABLE TO LICENSEE FOR DAMAGES IN AN AMOUNT GREATER THAN THE FEES PAID FOR THE USE OF THE SOFTWARE. Some states/jurisdictions do not allow the limitation or exclusion of incidental or consequential damages, so the above limitations or exclusions may not apply.

Intellectual Property Right Infringement. Synplicity will defend or, at its option, settle any claim or action brought against Licensee to the extent it is based on a third party claim that the SOFTWARE as used within the

scope of this Agreement infringes or violates any US patent, copyright, trade secret or trademark of any third party, and Synplicity will indemnify and hold Licensee harmless from and against any damages, costs and fees reasonably incurred that are attributable to such claim or action; provided that Licensee provides Synplicity with (i) prompt written notification of the claim or action; (ii) sole control and authority over the defense or settlement thereof (including all negotiations); and (iii) at Synplicity's expense, all available information, assistance and authority to settle and/or defend any such claim or action. Synplicity's obligations under this subsection do not apply to the extent that (i) such claim or action would have been avoided but for modifications of the SOFTWARE, or portions thereof, other than modifications made by Synplicity after delivery to Licensee; (ii) such claim or action would have been avoided but for the combination or use of the SOFTWARE, or portions thereof, with other products, processes or materials not supplied or specified in writing by Synplicity; (iii) Licensee continues allegedly infringing activity after being notified thereof or after being informed of modifications that would have avoided the alleged infringement; or (iv) Licensee's use of the SOFTWARE is not strictly in accordance with the terms of this Agreement. Licensee will be liable for all damages, costs, expenses, settlements and attorneys' fees related to any claim of infringement arising as a result of (i)-(iv) above.

If the SOFTWARE becomes or, in the reasonable opinion of Synplicity is likely to become, the subject of an infringement claim or action, Synplicity may, at Synplicity's option and at no charge to Licensee, (a) obtain a license so Licensee may continue use of the SOFTWARE; (b) modify the SOFTWARE to avoid the infringement; (c) replace the SOFTWARE with a compatible, functionally equivalent, and non-infringing product, or (d) if Synplicity determines that options (a), (b), and (c) are not commercially reasonable, then Synplicity shall have the right to terminate the licenses granted hereunder and refund to Licensee the amount paid for the SOFTWARE, as depreciated on a straight-line 5-year basis, or such other shorter period applicable to Subscription Licenses.

THE FOREGOING PROVISIONS OF THIS SECTION STATE THE ENTIRE AND SOLE LIABILITY AND OBLIGATIONS OF SYNPLICITY, AND THE EXCLUSIVE REMEDY OF LICENSEE, WITH RESPECT TO ANY ACTUAL OR ALLEGED INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS BY THE SOFTWARE (INCLUDING DESIGN TECHNIQUES) AND DOCUMENTATION.

Export. Licensee warrants that it is not prohibited from receiving the SOFTWARE under U.S. export laws; that it is not a national of a country subject to U.S. trade sanctions; that it will not use the SOFTWARE in a location that is the subject of U.S. trade sanctions that would cover the SOFTWARE; and that to its knowledge it is not on the U.S. Department of Commerce's table of deny orders or otherwise prohibited from obtaining goods of this sort from the United States.

Miscellaneous. This Agreement is the entire agreement between Licensee and Synplicity with respect to the license to the SOFTWARE, and supersedes any previous oral or written communications or documents (including, if you are obtaining an update, any agreement that may have been included with the initial version of the Software). This Agreement is governed by the laws of the State of California, USA excluding its conflicts of laws principals. This Agreement will not be governed by the U. N. Convention on Contracts for the International Sale of Goods and will not be governed by any statute based on or derived from the Uniform Computer Information Transactions Act (UCITA). If any provision, or portion thereof, of this Agreement is found to be invalid or unenforceable, it will be enforced to the extent permissible and the remainder of this Agreement will remain in full force and effect. Failure to prosecute a party's rights with respect to a default hereunder will not constitute a waiver of the right to enforce rights with respect to the same or any other breach.

Government Users. If the SOFTWARE is licensed to the United States government or any agency thereof,

then the SOFTWARE and any accompanying documentation will be deemed to be “commercial computer software” and “commercial computer software documentation”, respectively, pursuant to DFAR Section 227.7202 and FAR Section 12.212, as applicable. Any use, reproduction, release, performance, display or disclosure of the SOFTWARE and accompanying documentation by the U.S. Government will be governed solely by the terms of this Agreement and are prohibited except to the extent expressly permitted by the terms of this Agreement.

March 2008

CHAPTER 1

Synplify Premier Tool for Xilinx Devices

This document describes the design flow for the Synplify Premier tool using Xilinx technologies. Topics include:

- [FPGA Implementation Solution](#)
- [Synplify Premier Overview](#), on page 1-7
- [Physical Synthesis Flow Diagram](#), on page 1-8
- [Physical Synthesis Setup](#), on page 1-17
- [Physical Synthesis Flow—Task Summary](#), on page 1-29
- [Define the Project](#), on page 1-30
- [Run Logic Synthesis](#), on page 1-37
- [Validate Logic Synthesis Results](#), on page 1-38
- [Run Physical Synthesis](#), on page 1-42
- [Analyze Physical Synthesis Results](#), on page 1-45
- [Improve Performance](#), on page 1-48
- [Design Planner](#), on page 1-51
- [Design-Plan Based Physical Synthesis Flow](#), on page 1-52

FPGA Implementation Solution

The Synplify Pro, Synplify Premier, and Identify tools are an integrated FPGA implementation solution. These tools include features that address specific FPGA design applications, such as:

- [Timing Closure](#)
- [Single-Chip ASIC Verification](#)
- [System Design](#)
- [Design Analysis](#)
- [Comprehensive HDL Support](#)
- [RTL Debugging](#)
- [Incremental Flows](#)
- [DSP Synthesis](#)

As the FPGA grows in size and complexity at a high rate, designers are continually challenged to achieve timing closure. FPGA routing interconnects now dominate the overall percentage of delay in the circuit, which leads to less timing correlation and less design stability across iterations.

FPGAs now contain the equivalent logic capacity of a four million gate ASIC design with 10Mb RAM, IPs such as Gigabit I/O, DSP blocks, and microprocessors. FPGAs drive new process technologies; currently, leading-edge FPGAs use 40nm process technology. At 40nm and below physical synthesis is required to achieve timing closure.

Timing Closure

The top issue for FPGA designers, timing closure is the process required to converge on the timing goals for a design. Timing closure is primarily a function of:

- Design stability
- Timing correlation
- Design iterations
- Timing performance

For timing closure, use the:

- Synplify Pro tool to handle timing closure with best in class logic synthesis results, fast runtimes, and good timing correlation.
- Synplify Premier tool to improve timing closure with Graph-based Physical Synthesis. Graph-based Physical Synthesis provides design stability, accurate timing correlation, minimal design iterations, and great timing performance.

Single-Chip ASIC Verification

Synplify Premier supports single-chip ASIC verification with:

- Gated Clock Conversion
- Generated Clock Conversion
- DesignWare Support
- ASIC Component Translation (lib2syn)
- RTL Debug

System Design

System Design supports the ReadyIP Encryption flow, which includes:

- IP Encryption
- Third-party Partner IP Libraries
- System Designer

Design Analysis

Use the following tools to analyze your design:

- [HDL Analyst](#)
- [Physical Analyst](#)
- [Cross Probing](#)

HDL Analyst

The HDL Analyst tool consists of the RTL and Technology views. The RTL View represents the generic Boolean structure of the design after the HDL has been compiled, whereas, the Technology View represents the technology-specific mapped gate-level netlist.

Physical Analyst

The Physical Analyst tool, a Synplify Premier only feature, provides a physical representation of the design placement (at the gate level) on the selected target device. You can cross probe paths from the Technology View to easily locate their placement in the Physical Analyst View.

Cross Probing

Cross probing is a powerful debugging feature that can be used for the HDL Analyst views and the Physical Analyst view. Additionally, the cross probing feature allows you to cross probe from vendor-generated timing reports to the Technology view and/or the Physical Analyst view.

Comprehensive HDL Support

The Synplify Pro and Synplify Premier products provide comprehensive HDL support for:

- VHDL 99
- Verilog 2005
- SystemVerilog—While not comprehensively supported; tools do support many mainstream language functions.

Some of the main SystemVerilog features include:

- unique/priority if/case support
- do...while loop support
- continue, break support
- records (structures) support
- typedef support

- .name, .* for instantiations
- new SystemVerilog data types
- always_comb, always_ff, always_latch
- immediate assertions
- packed and unpacked arrays
- pattern assignments
- interface support
- System Verilog enhancement to functions/tasks
- new generates support

The Synplify Pro and Synplify Premier software contains HDL support that handles design portability and mixed HDL languages. A design constructed strictly using generic RTL, which does not contain FPGA vendor-specific code or instantiations, can easily be ported from one FPGA vendor by simply re-targeting to the other FPGA vendor. Both Verilog and VHDL languages are supported.

RTL Debugging

The Identify feature provides RTL debugging capability for the Synplify Pro and Synplify Premier tools. Hardware debugging used to require a logic analyzer on a board. However, as more designs became integrated onto a single FPGA device, vendors began to embed logic analyzer functionality onto the devices as well. Only the Identify tool provides an embedded HDL analyzer for the device at the RTL level, similar to an RTL simulator, instead of only at the gate-level like a logic analyzer.

Incremental Flows

The Xilinx SmartCompile place-and-route feature consists of the SmartGuide and partition flows.

For the SmartGuide flow, use the:

- MultiPoint Synthesis flow of the Synplify Pro tool to stabilize the gate-level naming conventions and interfaces with the partition flow.
- Synplify Premier tool to mitigate design perturbation due to incremental design changes. The Synplify Premier software can handle the perturbation with accurate timing correlation and the coreloc feature.

DSP Synthesis

The Synplify Premier software automatically infers common DSP functions such as Multiply-Accumulate from your RTL code and efficiently maps them into the DSP block of the target FPGA. For example, the tool can map to DSP48 blocks on the Xilinx Virtex5 device. The RTL code is kept vendor-independent while providing a highly efficient implementation for the target FPGA.

The Synplify DSP product can synthesize a Simulink model into RTL code that has been optimized for the target silicon. From a single Simulink model, you can quickly explore performance and area utilization enhancements. Then, the RTL is synthesized by Synplify Premier software into a netlist for implementation for the specified hardware.

See the following sections for complete details on how to run the Synplify Premier Physical Synthesis flows.

Synplify Premier Overview

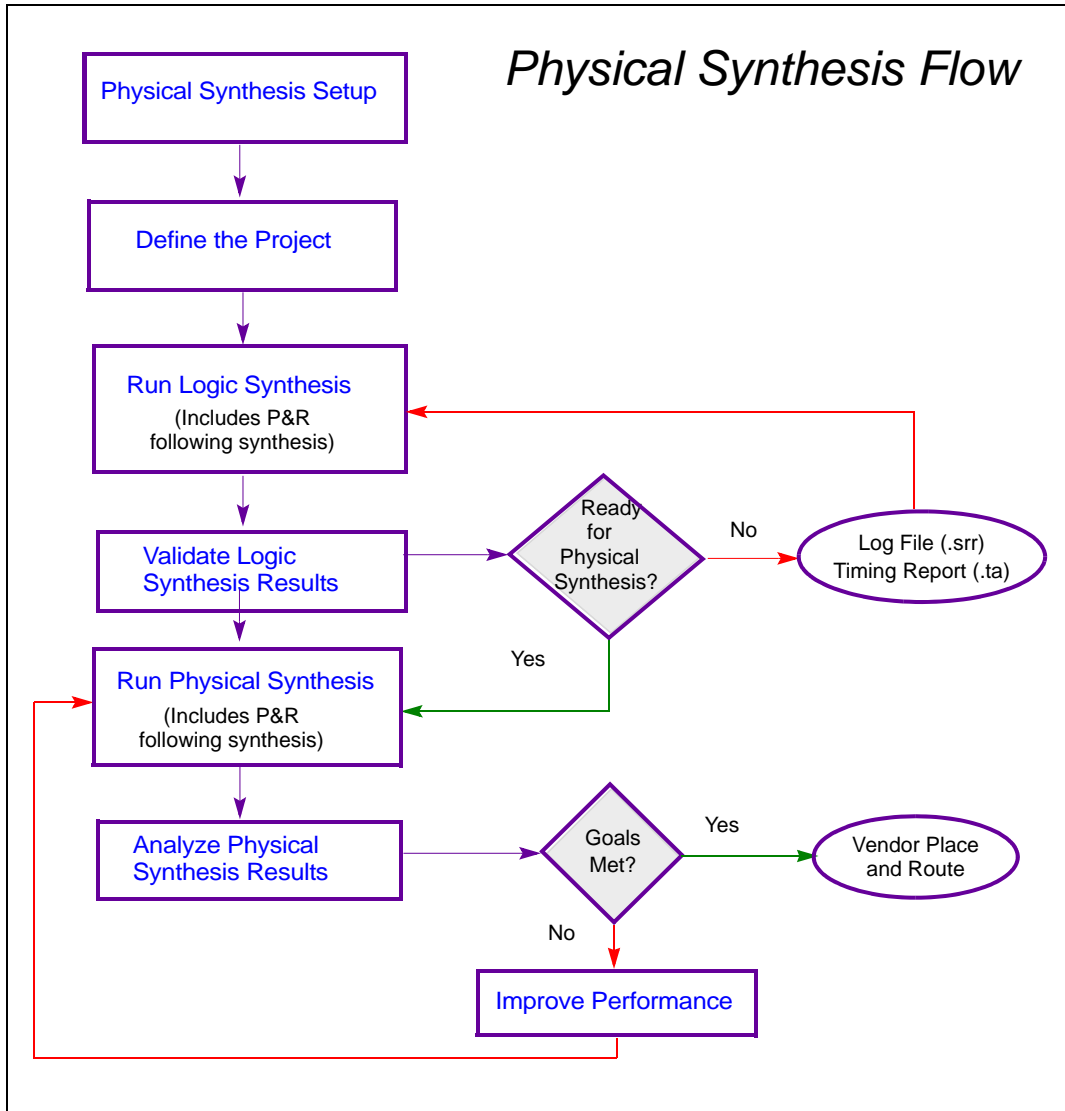
The Synplify Premier product is a physical synthesis timing closure solution that provides more accurate timing correlation and faster timing closure than could be achieved through previous design methodologies. This tool offers a push-button, graph-based design flow for improving overall device performance while simultaneously delivering tight correlation between pre-route timing estimates and final post place-and-route results. The essence of the graph-based approach is that preexisting wires, switches and placement sites used for routing an FPGA can be represented as a detailed routing resource graph. The notion of distance then changes to a measure of delay and availability of wires. Graph-based physical synthesis technology merges optimization and placement to generate a fully placed and physically optimized netlist, providing rapid timing closure and increased timing improvement. You can enable the re-timing feature to provide an additional performance boost.

For details, see the following topics:

- [Physical Synthesis Flow Diagram](#)
- [Types of Physical Synthesis Flows](#)
- [Supported Xilinx Devices](#)
- [About the Xilinx ISE Place-and-Route Tool](#)

Physical Synthesis Flow Diagram

The figure below shows the flow for graph-based physical synthesis.



Types of Physical Synthesis Flows

The Synplify Premier tool supports these flows:

- [Logic Synthesis Validation Phase](#)
Synthesis validation phase to ensure that the design has realistic constraints and can successfully complete synthesis and place and route.
- [Graph-based Physical Synthesis](#)
Push-button, single pass flow that merges optimization and placement to generate a fully placed, physically optimized netlist.
- [Graph-based Physical Synthesis with Design Planner](#)
Single-pass flow that includes a design plan physical constraint file for guiding the global placement process.
- [Design-plan Based Physical Synthesis](#)
Flow requiring manual placement of the critical path to improve the design. This flow is for older technologies and requires the Synplify Premier tool with Design Planner option.

See Also

- To determine the supported physical design flows for your Synplify Premier tool, see:
 - [Supported Xilinx Devices](#), on page 1-10
 - [About the Xilinx ISE Place-and-Route Tool](#), on page 1-10
- For details on the flows, see [Logic Synthesis Validation Phase](#), on page 1-11

Supported Xilinx Devices

Synplify Premier physical synthesis is supported for the following technologies:

Physical Optimization Flows	Xilinx Devices
Graph-based Physical Synthesis	Spartan-3 Virtex-II Pro Virtex-4 Virtex-5
Graph-based Physical Synthesis with Design Planner ¹	Spartan-3 Virtex-II Pro Virtex-4 Virtex-5
Design Plan based Physical Synthesis ¹	Virtex Virtex-II Virtex-E

1. Requires the Synplify Premier tool with the Design Plan option.

About the Xilinx ISE Place-and-Route Tool

Before you run physical synthesis, you must:

- Install the Xilinx ISE place-and-route tool.
- Consult the release notes for the most current information on supported ISE versions.

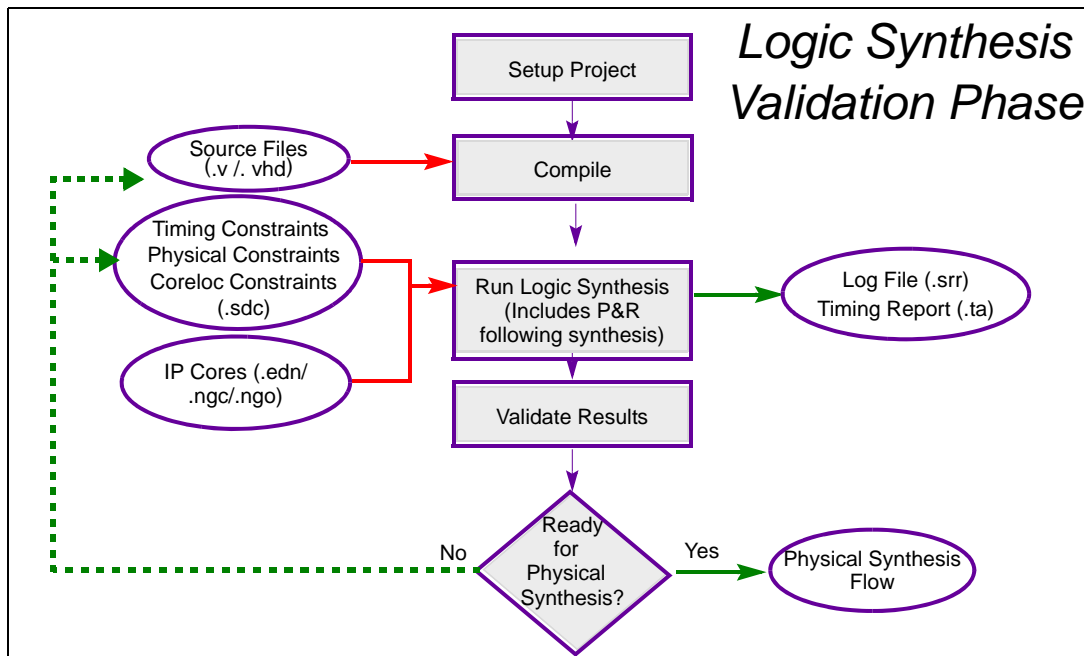
(From the Synplify Premier tool: Help->Online Documents->release_notes.pdf
->Third Party Tool Versions)

Logic Synthesis Validation Phase

Always begin any physical synthesis flow with the logic synthesis validation phase. This is to ensure that a design can successfully complete synthesis and an initial place and route before going into the physical synthesis process. This can save valuable time by identifying obvious problems early in the process. The logic synthesis validation phase ensures that the design:

- Can successfully complete synthesis.
- Can successfully complete place and route.
- Has been assigned accurate, realistic constraints.

The figure below shows the flow for logic synthesis validation phase.



See Also

- Steps 1 through 3 in *Physical Synthesis Flow—Task Summary*, on page 1-29 for a summary of tasks required to complete this flow.
- *Define the Project*, on page 1-30 to start the physical synthesis design flow.

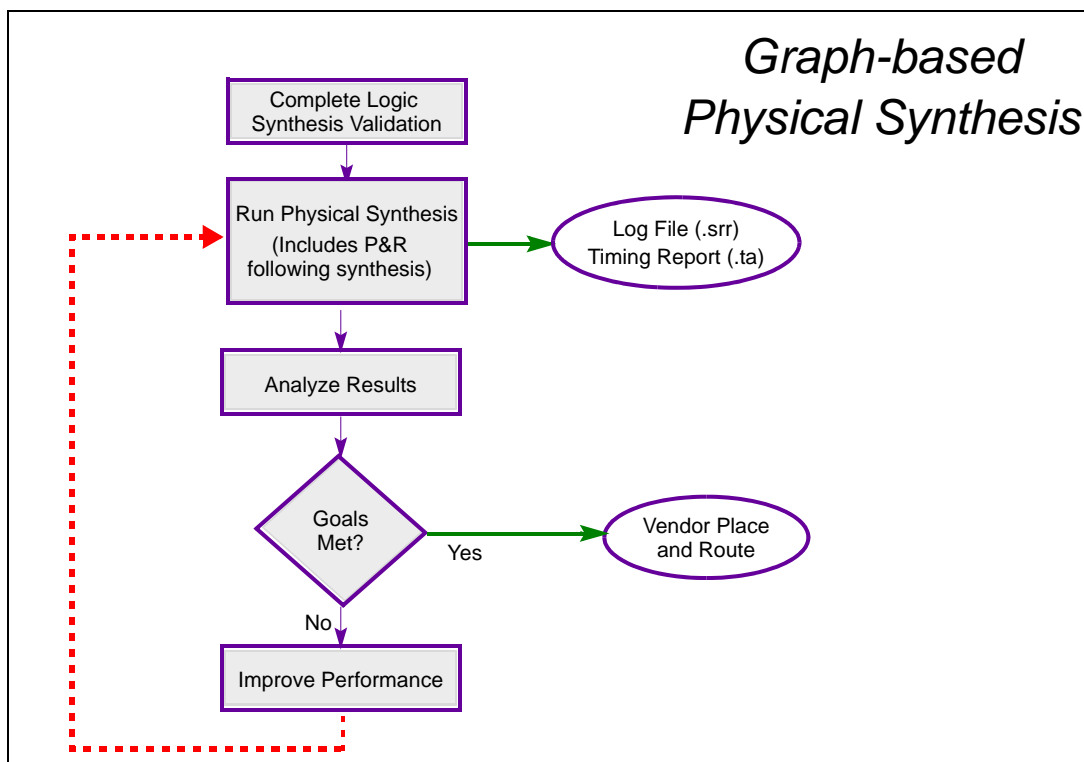
Graph-based Physical Synthesis

This is a single-pass, push-button physical synthesis flow that merges design optimization and placement to generate a fully-placed, physically-optimized netlist, providing rapid timing closure and increased timing improvement. Synthesis and placement are integrated by performing concurrent placement and optimization based on timing constraints and device technology. The output netlist contains placement information. Graph-based physical synthesis also simplifies the process for critical path timing improvements.

This flow can be used with the following Xilinx technologies:

- Spartan-3
- Virtex-II Pro
- Virtex-4
- Virtex-5

The figure below shows the flow for graph-based physical synthesis.



For a flow that includes added performance improvement, see [Graph-based Physical Synthesis with Design Planner](#), below.

See Also

- [Physical Synthesis Flow—Task Summary](#), on page 1-29 for a summary of tasks required to complete this flow.
- [Run Physical Synthesis](#), on page 1-42 to start the physical synthesis design flow.

Graph-based Physical Synthesis with Design Planner

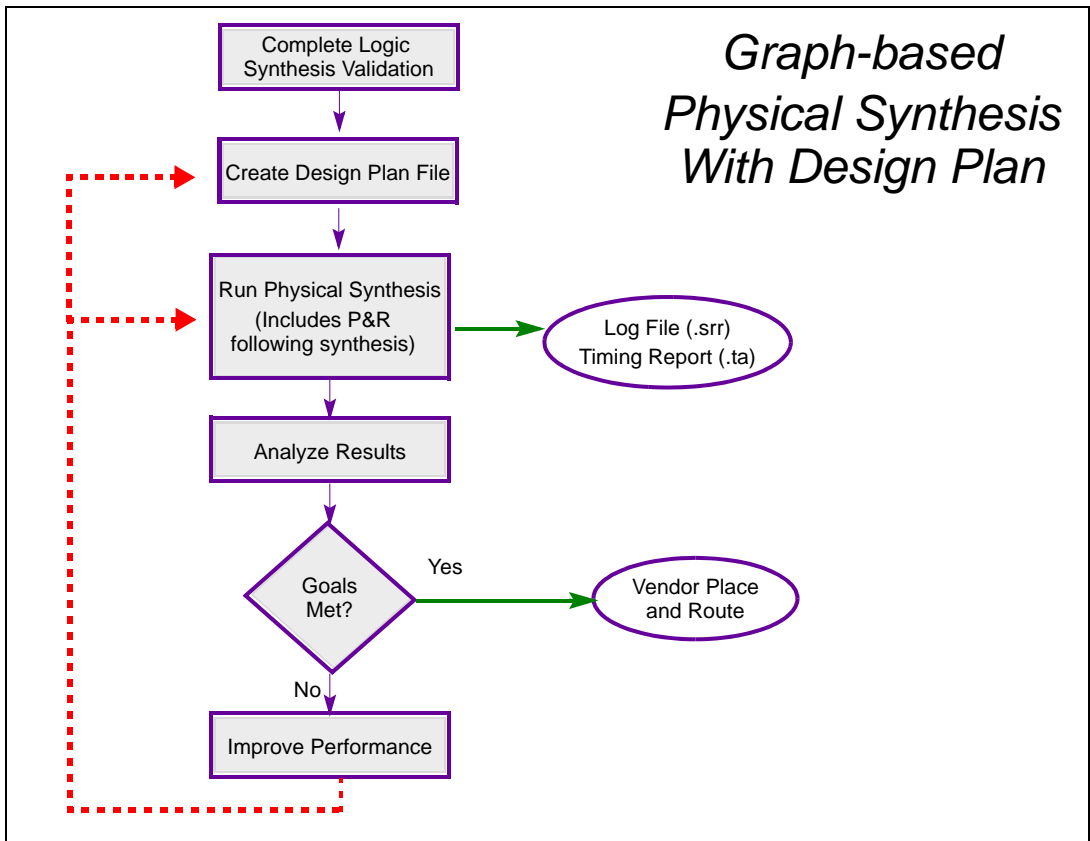
This flow is available when you have the Design Plan option with the Synplify Premier tool and can be used with the following Xilinx technologies:

- Spartan-3
- Virtex-II Pro
- Virtex-4
- Virtex-5

This flow is similar to the push-button graph-based physical synthesis flow described in the previous topic, except a design plan file is included to guide global placement. Use this flow to improve performance.

Placement constraints are generated when you assign RTL logic from the RTL view (HDL Analyst) to ports or regions in the Design Plan view (Design Planner). These regions constrain logic to the areas you specify on the device. During optimizations, the regions guide global placement and subsequently influence physical optimizations.

The figure below shows the flow for graph-based physical synthesis using a design plan file.



See Also

- [Design Planner](#), on page 1-51 for general information on this feature.
- [Create Design Plan File](#), on page 1-28 for information on how to create a design plan file.
- [Physical Synthesis Flow—Task Summary](#), on page 1-29 for a summary of tasks required to complete the flow.
- [Run Physical Synthesis](#), on page 1-42 to start the physical synthesis design flow.

Design-plan Based Physical Synthesis

This flow is for Synplify Premier users that:

- Have Design Planner option and
- Use the following Xilinx technologies:
 - Virtex
 - Virtex-II
 - Virtex-E

The flow requires using Design Planner to manually create physical constraints by assigning critical path logic to a specific location on the die to improve performance. Constraints are saved to the design plan file and added to the project to complete physical synthesis for the design. See [Design-Plan Based Physical Synthesis Flow, on page 1-52](#) for details on using this flow.

Physical Synthesis Setup

The Synplify Premier physical synthesis tools require that you provide all the timing constraints, I/O standard assignments, I/O placements, and pre-placement of memories and DSP blocks, if possible. Previously, this was reserved for placement and routing. This section contains information for defining timing, physical constraints, and attributes so you can run your design through the physical synthesis flows successfully. Topics include:

- [Timing Constraints Guidelines](#)
- [Setting Up Constraints](#)
 - [Create the Timing Constraint File \(.sdc\)](#)
 - [Translate Xilinx UCF Physical Constraints](#)
 - [Run Constraint Check](#)
- [Using the -route Constraint](#)
- [Using IP Cores](#)
 - [Xilinx RPM Management](#)
 - [Coreloc – Core Locked-Placement Constraints](#)
 - [Using Secure/Non-secure IP Cores](#)
- [Understanding Synplify Premier Placement and Routing](#)
 - [Global Placement](#)
 - [Detail Placement](#)
 - [Routing](#)
- [Create Design Plan File](#)

Timing Constraints Guidelines

The Synplify Premier tool requires that you provide accurate and complete timing constraints to run physical synthesis effectively. The Synplify Premier software outputs a placed design, for which the place-and-route tool cannot correct constraints used inaccurately during physical synthesis. For example, a false path constraint provided only to the place-and-route tool allows the Synplify Premier software to move components affected by the false path far apart to resolve critical path issues.

It is extremely important you verify timing constraints. Use the following guidelines:

- Define all clocks.
- Assign realistic, accurate timing constraints. Do not over-constrain the tool.
- Assign clocks to the correct clock group. Clocks assigned to separate groups are cross-clock paths, which are treated as false paths.
- Specify all multicycle paths.
- Specify all false paths.
- Specify all input and output delays.
- Ensure that all I/Os have I/O standards and drive strengths specified.
- Top-level clocks that do not use DCM/PLL should have the constraint specified on the port to ensure insertion delay is modeled correctly.
- Clocks derived through the DCM/PLL should have the constraint specified on the port driving the DCM/PLL, if possible.
- Make sure constraints are valid. Check for the following types of messages in the log file:
 - Cannot find object `<clock>` to apply `define_clock`.
 - Timing constraint `<x> <y>` never applies in design and was not found.

You can use the `ucf2sdc` utility to help you translate some of this information. Ultimately, sometimes manual intervention and review is necessary for this information to be complete.


Setting Up Constraints

This section provides information on defining timing and physical constraints and attributes for synthesis. Topics include:

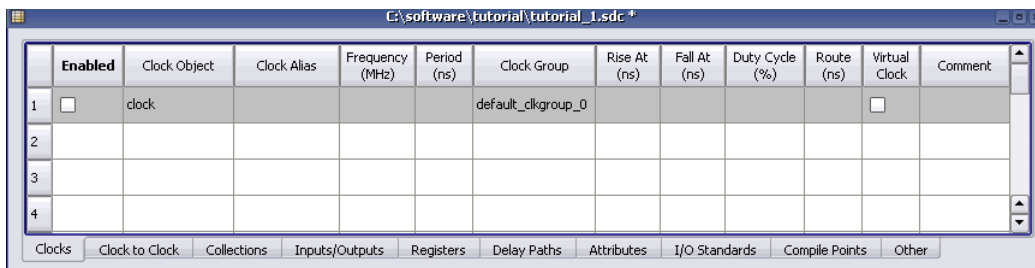
- [Create the Timing Constraint File \(.sdc\)](#)
- [Translate Xilinx UCF Physical Constraints](#)
- [Run Constraint Check](#)

Create the Timing Constraint File (.sdc)

Use timing constraints to specify performance goals for the design and describe the environment. You can specify constraints in a text file using any text editor, however it is easier to use the SCOPE GUI (Synplicity synthesis constraint optimization environment), a spreadsheet-like interface that automatically formats the constraints you define. To create the constraints file:

1. Create the project file (see [Create Project, on page 1-31](#)).
2. Compile the design (Run->Compile Only). This can be done after adding the source files to the project.
3. Open an .sdc file:
 - Click the New Constraint file (SCOPE) icon in the toolbar. ()
 - Click OK to Initialize Constraints.

The SCOPE window opens with the design objects initialized, such as clock frequency, ports, input/outputs. This means you can use the pull-downs in the SCOPE columns to fill in object names and types when specifying constraints.



Specify the desired constraints for the design. See [Using the SCOPE UI, on page 5-2](#) of the *User Guide* for descriptions of the constraints and complete details on creating constraint files.

4. Save the file; click Yes in the dialog box to add the file to your project.
5. Save the project file.

See Also

- [Translate Xilinx UCF Physical Constraints](#), next, for information on how to complete the .sdc file by adding the UCF constraints for physical synthesis.

Translate Xilinx UCF Physical Constraints

A unified set of timing and physical constraints is required to run physical synthesis. Earlier, you created Synplify Premier timing constraints using the SCOPE editor. This section describes how to use the `ucf2sdc` utility to translate constraints in your Xilinx UCF to .sdc format. This ensures that UCF timing and physical constraints are also honored during physical synthesis.

You can run the utility from any shell window; `ucf2sdc` is located in the `bin` directory where your software is installed:

```
<install_dir>/bin
```

To convert UCF constraints:

1. Run `ucf2sdc` using the following syntax:

```
<install_dir>/bin/ucf2sdc -iucf <constraints_file>.ucf  
-osdc <constraints_file>.sdc
```

2. After translating the constraints, edit the new .sdc file.

The translator converts the most common timing and physical constraints. However, because of the diversity and complexity of UCF format, the resulting .sdc file requires manual intervention. To do this:

- Visually inspect the translated file.

The original UCF commands are written as comments in the new .sdc file so that you can validate the translated constraints.

- Manually edit the SDC file to complete the translation of constraints, as necessary.

For reference information on `ucf2sdc`, see [Xilinx `ucf2sdc` Utility](#), on page 12-3 in the *Reference (Reference Manual)*.

3. To run physical synthesis, timing and physical constraints can be combined into one `.sdc` file. Include the timing constraints (created in [Create the Timing Constraint File \(.sdc\)](#)) into the `.sdc` file containing the translated physical constraints. Make sure that all of the following types of constraints are combined into the `.sdc` file:
 - Timing Constraints:
 - Clock
 - Clock-to-clock
 - IO delays
 - IO standard, drive, slew and pull-up/pull-down
 - Multi-cycle and false paths
 - Max-delay paths
 - DCM parameters
 - Physical Constraints ¹:
 - Register packing into IOB
 - LOC on IO pads
 - LOC/RLOC constraints on macros (BUFG, DCM, RAMB, DSP, MULT, etc.)
 - LOC/RLOC constraints on instances (Register, LUT, SRL, RAMS, RAMD, etc.)
 - AREA_GROUP constraints
4. Also, include any synthesis attributes, from logic synthesis, such as `syn_ramstyle`, into the `.sdc` file.
5. Remove all successfully-translated constraints from the original `.ucf` file.

1. Physical constraints applied to invariant objects (such as registers, instantiated macros and modules) can be safely translated to SDC constraints. Please use the Design Planner™ tool for advanced physical constraints.

6. After translating and editing, the resulting .ucf might not be empty because there are a few constraints that cannot be translated (such as the PROHIBIT constraint). If there are any UCF commands remaining in the original file, add the .ucf file to your project. The file must have the same base name as the .edf netlist so that the Xilinx place-and-route tool can source the file.
7. Save the project file.

Run Constraint Check

After you set up your project, you can check your constraints and their syntax. Do the following:

1. Make sure you target a technology that supports this feature.
2. Generate a constraint file, then select Run->Constraint Check.

This command generates a report that checks the syntax and applicability of the timing constraints in the .sdc file(s) for your project. The report is written to the *project_name_cck.rpt* file.

Using the -route Constraint

For the Synplify Premier product, the -route constraint usually should be:

- Removed when converting a Synplify Pro project to a Synplify Premier project. See [Logic Synthesis Usage, on page 1-22](#).
- Used at a global level in the Synplify Premier tool to produce a better netlist during global placement. See [Physical Synthesis Usage, on page 1-23](#).

Logic Synthesis Usage

During logic synthesis the -route option either tightens or loosens timing constraints, without affecting constraints forward annotated to the Xilinx ISE place-and-route tool. The Synplify Pro software tunes the delay estimates to compensate for differences between logic synthesis and the actual place-and-route delays. The Synplify Premier tool uses a different timing model for interconnect delays, therefore, the -route constraint generally should not be used.

Physical Synthesis Usage

The Synplify Premier graph-based timing estimation for critical paths automatically handles the correlation between the synthesis and place-and-route tools. For physical synthesis, apply the `-route` constraint to global clocks from the SCOPE Clock panel. An example of the Tcl command is:

```
define_clock {clk} -period 4 -clockgroup cg1 -route 1
```

Remember to monitor the effects of using the `-route` constraint from the Pre-placement Timing Snapshot report in the log file. A clock constrained by the `-route` option should target a slightly negative slack.

Using IP Cores

Refer to this section whenever you include IP cores in your designs. Topics include:

- [Xilinx RPM Management](#)
- [Coreloc – Core Locked-Placement Constraints](#)
- [Using Secure/Non-secure IP Cores](#)

Xilinx RPM Management

The Synplify Premier tool honors all Relationally Placed Macro (RPM) constraints supported by the Xilinx ISE place-and-route tool. RPM constraints can be created by the user or embedded in the Xilinx IP core. Use the `xc_use_rpms` attribute to help you manage RPMs for physical synthesis.

Values for `xc_use_rpms` can be:

- 0 – removes all RPMs for global placement and physical synthesis. With this value, RPMs can be moved during global placement and physical synthesis. This eliminates the chance of physical synthesis errors related to illegal placement, however, QoR advantages for using RPMs are missed.
- 1 – (default) maintains RPMs for global placement and physical synthesis. With this value, the QoR advantages of using RPMs are maintained, however, if any of the RPMs are illegally placed, physical synthesis will error out.

- 2 – honors RPMs during global placement and removes them for physical synthesis. With this value, some QoR advantages can be maintained for global placement, however, the tool has the option to optimize placement during physical synthesis.

Coreloc – Core Locked-Placement Constraints

The Synplify Premier generates a `coreloc` constraint file that contains the placement for all “anchor” or “core” instances in the design:

- IOs
- BlockRAM
- BlockMULT
- FIFO
- DSP48
- DCM
- BUF*

The major applications for `coreloc` constraints are:

- Add stability – comparisons between logic and physical synthesis should ensure that block and I/O placement are equivalent. To guarantee consistent and stable comparisons, include the `coreloc.sdc` to both logic and physical synthesis runs.
- Timing convergence – by stabilizing block and I/O locations, you can eliminate variations in results that stem from placement variations when running small design changes.

The file is generated during place-and-route backannotation and is written to the results directory as:

```
filename_coreloc.sdc
```

where *filename* is the same base name as the EDIF output netlist (*filename.edf*).

To utilize the coreloc feature:

1. Make sure the switch Backannotate placement and timing Data following Place & Route is enabled when you create the place-and-route implementation before running synthesis. See [Create Place and Route Implementation, on page 1-34](#).
2. After synthesis, add the `_coreloc.sdc` file to your project.
3. Re-run physical synthesis.

Using Secure/Non-secure IP Cores

The tool supports the following types of IP cores:

- EDN
- NGC, secure/non-secured
- NGO

For a description of the types of IP cores and how they are implemented during synthesis, see [Xilinx IP Cores, on page 4-49](#) in the *User Guide*.

To utilize these secure/non-secure IP cores:

1. Create your project file.

Make sure to instantiate each Xilinx IP core in the top-level RTL source code.
2. Add the appropriate IP core files (`.edn`, `.ngc`, or `.ngo`) to your project.
3. Run physical synthesis.
 - For non-secure IP cores, the contents of the core are absorbed into the top-level EDIF file and are visible in the tool.
 - For secure IP cores, the contents of the core are absorbed into a separate and secure core EDIF file. However, the contents of the core are not visible in the EDIF netlist and HDL Analyst tool.
4. Run placement and routing.

The following files are automatically forward annotated to the place-and-route tool:

- Secure IP core EDIF files

- Secure IP core timing constraint file (.ucf)
- Secure IP core placement constraint file (.ncf)

For complete details when working with Xilinx secure and non-secure IP cores, see [Including Secure and Non-Secure Cores for Physical Synthesis](#), on page 4-52 and [The Synplify-EDK Design Flow](#), on page 4-55 in the *User Guide*.

See Also

- [Working with Xilinx Regions](#), on page 10-36 in the *User Guide* for more details defining physical constraints for cores using the Design Planner.
- [Working with Xilinx IP](#), on page 4-49 in the *User Guide* for general information on including Xilinx IPs for synthesis.)

Understanding Synplify Premier Placement and Routing

The following Synplify Premier processes are defined to provide more insight into how the tool works and ways for your design to run through the Synplify Premier flow successfully:

- [Global Placement](#)
- [Detail Placement](#)
- [Routing](#)

Global Placement

The 9.0.1 and later versions of the Synplify Premier tool use the Xilinx placer to generate locations for I/Os and block components. Global placement is responsible for spreading the design evenly on the chip. This is especially important for large block RAM and DSP48 components because physical synthesis does not move them. To avoid block component placement problems, you need to lock placement. See [Coreloc – Core Locked-Placement Constraints](#), on page 1-24 for information.

Detail Placement

The detail placer performs complete legality checks for placed components to ensure critical paths are routed optimally. Some local routing and optimizations are performed as well during placement.

The Xilinx Virtex-5 CLB placement and packing rules are complex. Despite extensive testing, the Synplify Premier flow might still encounter placement violation rules for this device. If you encounter a problem, report it to technical support and:

- Include the Xilinx map log file, Synplify Premier *.aux files, and .srm netlist file to view the problem slice.
- You can also report the problem from the Technology view using HDL-Analyst->Technology->Flattened View. For example, specify the following commands to select components placed in the problem slice:

```
set x [find -hier -inst * -filter @location==SLICE_X23_Y54
select $x
filter
```

Or you can use the Filter Schematic icon to create a schematic with just the instances for this slice. Send this schematic.

Routing

In the Synplify Premier flow final routing is implemented by the Xilinx place-and-route tool. The Synplify Premier tool only performs local routing during detail placement and optimization. In Synplify Premier 9.0.1 and later versions, local routing is forward annotated as initial pin assignments on LUTs. Placement and routing can change pin assignments to accommodate longer distance routing, but generally result in equivalent path delays.

The default routing effort level is recommended for most designs to route successfully with good timing closure. However, if high density routing causes routing detours, increase the effort level to enable extra effort using the `-sc c` switch. You can detect routing detours in the following ways:

- Look for messages in the log file which say it detects a congested design that is switching to a non timing driven mode.
- Open the Xilinx FPGA Editor and select the long delay nets. When wires show an indirect path to the load, then the design likely has detours.

If detours happen on high fanout nets, you can set the `MAXSKEW` option on the net to a fairly loose value, such as 1.0 ns. This setting uses a different router algorithm with higher priority on the applied net. If the design is still congested, then contact technical support.

Create Design Plan File

Use Design Planner to interactively assign RTL modules, paths or components to regions on the device. These placement constraints created in the Design Planner are implemented in global placement and guide detail placement.

1. See [Working with Xilinx Regions, on page 10-36](#) in the *User Guide* for complete details on creating the design plan file.

When you create a RTL region, select Region Type and then configure this region with one of the following modes:

Set the option to...	To...
Soft (Tunneling On)	Allow components to be moved across the boundaries in both directions. This is the default.
Hard (Tunneling Off)	Ensure that components are not moved out of the region, but allow other objects to be moved into the region.
Keep-out	Ensure that no placement occurs in the region. Use this option to create decongestion areas for optimizing your design.

2. After you have completed the design plan file (`.sfp`), add it to the project and enable the file in the Implementation Options ->Design Planning tab.
3. See [Run Physical Synthesis, on page 1-42](#) when you are ready to run physical synthesis. To do this, you must have already completed the initial phases of physical synthesis:
 - [Define the Project, on page 1-30](#)
 - [Run Logic Synthesis, on page 1-37](#)
 - [Validate Logic Synthesis Results, on page 1-38](#)

Physical Synthesis Flow—Task Summary

Here is a summary of the tasks required to complete physical synthesis. These tasks coincide with the flow diagrams for the different design flows described in *Logic Synthesis Validation Phase*, on page 1-11.

1. [Define the Project](#)
 - [Create Project](#)
 - [Set up Timing and Physical Constraints](#)
 - [Set Implementation Options](#)
 - [Create Place and Route Implementation](#)
 - [Specify Global Placement Options](#)
2. [Run Logic Synthesis](#)
3. [Validate Logic Synthesis Results](#)
4. [Create Design Plan File](#) (optional)
5. [Run Physical Synthesis](#)
6. [Analyze Physical Synthesis Results](#)
7. [Improve Performance](#) and Rerun Physical Synthesis, as required

Note: Design-plan only flow: if your technology supports only a design-plan based flow, you will need to complete a different set of tasks than those listed above. See *Design-Plan Based Physical Synthesis Flow*, on page 1-52 for information.

The remaining sections of the chapter provide details on how to complete the tasks above.

See Also

- [Logic Synthesis Validation Phase](#), on page 1-11
- [Graph-based Physical Synthesis](#), on page 1-12
- [Graph-based Physical Synthesis with Design Planner](#), on page 1-14

Define the Project

Project setup is the first phase of the physical synthesis design process. The project file (.prj) is a collection of input files and optimization switches required to synthesize your design. This section contains details on how to setup the file.

First, here are some guidelines to consider before setting up your project:

- Make sure the design is complete including all IPs (no black boxes). See [Working with Xilinx IP, on page 4-49](#) in the *User Guide* for more information.
- Make sure the design is properly constrained. (See [Improve Performance, on page 1-48](#) for tips.)
- Depending on your target Xilinx technology, a design plan file (.sfp) for physical synthesis is optional. However, to use an .sfp file requires the separately-licensed Synplify Premier Design Planner option.
- If you are using one of the graph-based physical synthesis flows, make sure you select a target technology that is supported. See [Supported Xilinx Devices, on page 1-10](#).
- Use the top-down design methodology. (A bottom-up flow is not supported.)
- Do not use the MultiPoint Synthesis flow with graph-based physical synthesis.

The following tasks are required to set up your project for physical synthesis:

- [Create Project](#)
- [Set up Timing and Physical Constraints](#)
- [Set Implementation Options](#)
- [Create Place and Route Implementation](#)
- [Specify Global Placement Options](#)

See the following topics for details.

Create Project

To create a project file for physical synthesis:

1. Bring up the Synplify Premier tool.
2. Click on Open Project, then New Project.
3. Click the Add File button and add the following design files:
 - .v and/or .vhd (HDL source files).
 - .edn/.ngc (core IP files – see [Working with Xilinx IP, on page 4-49](#) in the *User Guide* for more details on using IPs in the flow).
 - .sfp (optional design plan file. You can use this file only if your tool includes the Design Planner option. See [Design Planner, on page 1-51](#) for details).
4. Save the project file.

For details on creating a project file, see:

- [Setting Up HDL Source Files, on page 3-2](#) in the *User Guide*
- [Setting Up Project Files, on page 6-2](#) in the *User Guide*

Set up Timing and Physical Constraints

Constraints for physical synthesis include timing constraints and physical constraints for your design. Timing constraints are used to specify performance goals and describe the design environment. Xilinx physical constraints, as well as all constraints from the UCF should also be included in the project for physical synthesis. All of these constraints must be read from a single .sdc constraint file. To specify timing constraints for the Synplify Premier tool and to translate your Xilinx constraints (ucf2sdc) to use for physical synthesis, see [Setting Up Constraints, on page 1-19](#).

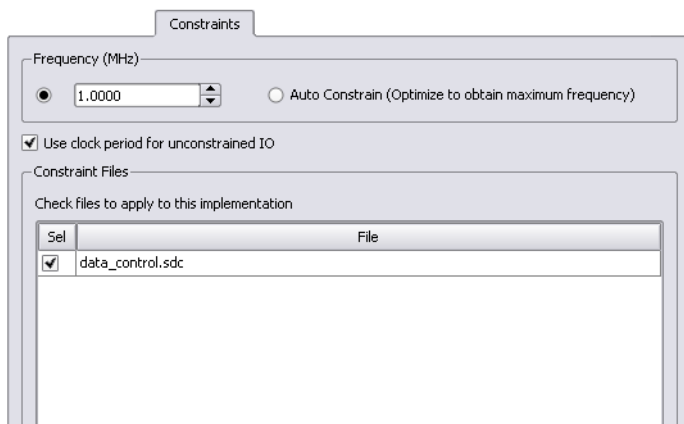
After you have completed the constraint file, add it to the project.

Set Implementation Options

Specify the implementation options for synthesis.

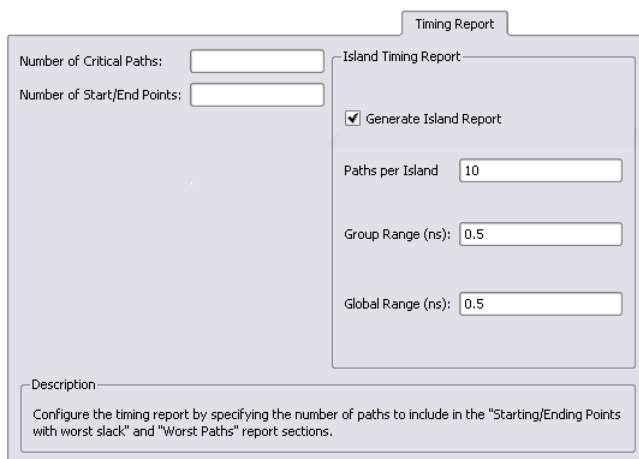
1. Bring up the Implementations Options dialog box (Implementation Options button).

2. In the Device panel, set options for:
 - Technology, part, speed, and package
 - Device mapping options
3. In the Options panel, set the optimization switches for synthesis. You can use the default switches for physical synthesis, which include the following: FSM Compiler, Resource Sharing, and Pipelining. For descriptions of all of the optimization switches, see [Setting Optimization Options, on page 6-17](#) of the *User Guide*.
4. In the Constraints panel:
 - Set an overall target frequency for the design. See [Specifying Global Frequency and Constraint Files, on page 6-19](#) of the *User Guide* for information.
 - Make sure the constraint file that you want to use for synthesis is selected.



5. In the Implementation Results panel, specify the output results directory and output file options. See [Specifying Result Options, on page 6-21](#) of the *User Guide* for details.

6. In the Timing Report panel specify:
 - Number of critical paths and start/end points to display in the timing report.
 - Island timing report; parameters to use for the report.



Timing Report

Number of Critical Paths:

Number of Start/End Points:

Island Timing Report

Generate Island Report

Paths per Island

Group Range (ns):

Global Range (ns):

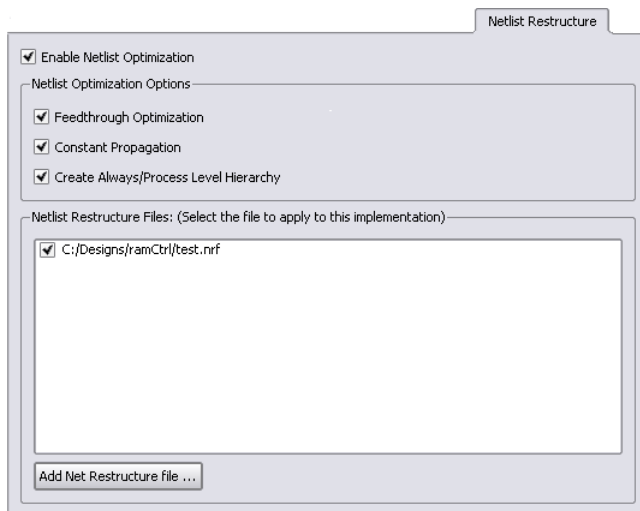
Description

Configure the timing report by specifying the number of paths to include in the "Starting/Ending Points with worst slack" and "Worst Paths" report sections.

If you do not wish to generate an island timing report at this time, you can choose to use the interactive Island Timing Analyst after synthesis. See [Generating the Island Timing Report Automatically, on page 14-14](#) in the *User Guide*. You can also change timing parameters and rerun the timing analyzer after synthesis. See [Using the Island Timing Analyst, on page 14-12](#) in the *User Guide*.

7. In the Verilog/VHDL panel, specify the desired HDL options. See [Setting Verilog and VHDL Options, on page 6-24](#) in the *User Guide*.

8. Specify options, as appropriate, in the Netlist Restructure panel for:
 - Any necessary netlist optimizations.
 - Netlist restructure file (.nrf) for which bit slicing or zippering might have been performed.



See [Setting Synplify Premier Netlist Restructuring Optimizations, on page 7-3](#) in the *User Guide* for descriptions of these switches.

9. Click OK to apply the implementation options.
10. Save the project file.

Create Place and Route Implementation

You can set up an implementation to run Xilinx ISE place and route after synthesis completes. To do this:

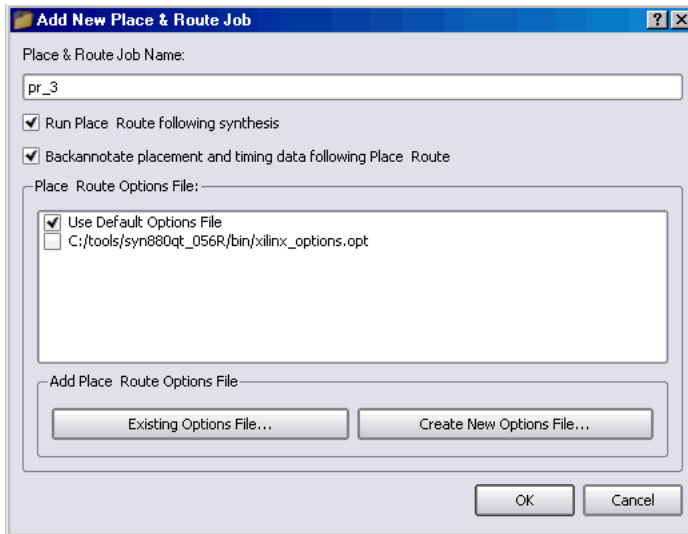
1. Make sure you are using the correct ISE version for your tool. See [About the Xilinx ISE Place-and-Route Tool, on page 1-10](#) for information.
2. Set the XILINX and PATH environment variables to point to a valid installation of the place and route tool.

Note, the Synplify Premier UI automatically sets the environment variable for:

PAR_BELDLYRPT to 1

This environment variable is set so that the Xilinx place-and-route placement file (.xdl) is generated with a particular format. The software uses this .xdl file to backannotate the placement information.

- From the project view, click on the New P&R button.

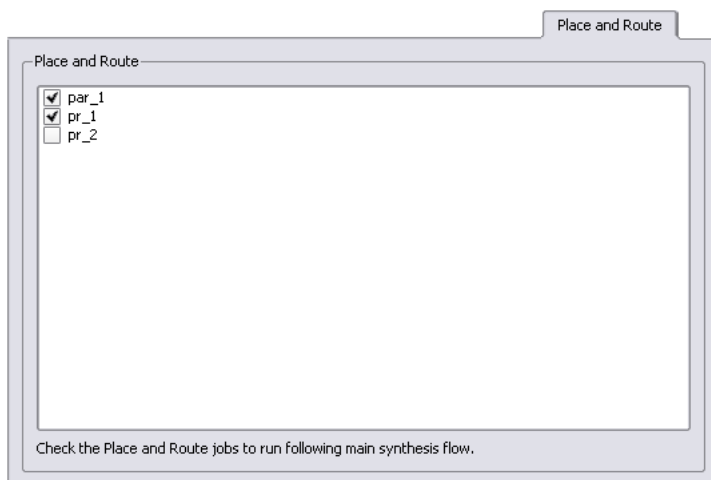


- Specify the Place & Route Job Name. Default is `pr_n`.
- Make sure the Run Place and Route following synthesis switch is enabled.
- If you want to generate a core location file (recommended), enable the Back Annotate placement and timing data following Place & Route switch. See [Coreloc – Core Locked-Placement Constraints, on page 1-24](#), for information about this file.
- Specify the place-and-route options file. The tool automatically uses default options located in

```
<install_directory>\lib\xilinx\xilinx_par.opt
```

You can change or override the default options. See [Specifying Xilinx Place-and-Route Options, on page 7-13](#) in the *User Guide* for details.

8. Enable the P&R implementation to use (Implementation Options->Place and Route tab).



9. Save the project file.

Specify Global Placement Options

For global placement, the tool uses a default options file. However, for graph-based physical synthesis only, you can override these default values. Use the `SYN_XILINX_GLOBAL_PLACE_OPT` Environment variable to direct the tool to the file that contains your preferred options. To override the defaults, set the variable to point to the path of your options file. For example, to point to the options file `test.opt` in the `C:/Temp` directory, you would update the variable as follows:

```
SYN_XILINX_GLOBAL_PLACE_OPT = "C:/Temp/test.opt"
```

Run Logic Synthesis

If this is the first time you are running synthesis on the design, run in logic synthesis mode. This means the Physical Synthesis switch is disabled. The initial synthesis run is to determine if there are any problems that need to be addressed before going on to the physical synthesis stage.

1. Before running logic synthesis, the following phases must be complete:
 - Create the project – [Create Project, on page 1-31](#).
 - Specify constraints – [Set up Timing and Physical Constraints, on page 1-31](#).
 - Set implementation options – [Set Implementation Options, on page 1-31](#).
 - Setup for place and route – [Create Place and Route Implementation, on page 1-34](#).
2. Disable the Physical Synthesis switch:
 - Located in Project view
 - or
 - From Implementation Options->Options.
3. Click the Run button.

The Synplify Premier tool goes through Compiling and Mapping phases. When physical synthesis completes, the place and route implementation that was set up in the project file is also run. When the job completes, Done! (or Warnings!) displays in the Project view. Output results files are shown in the right pane of the Project view. Double-click on the files to display them.

4. Go on to the next phase, [Validate Logic Synthesis Results, on page 1-38](#).

Validate Logic Synthesis Results

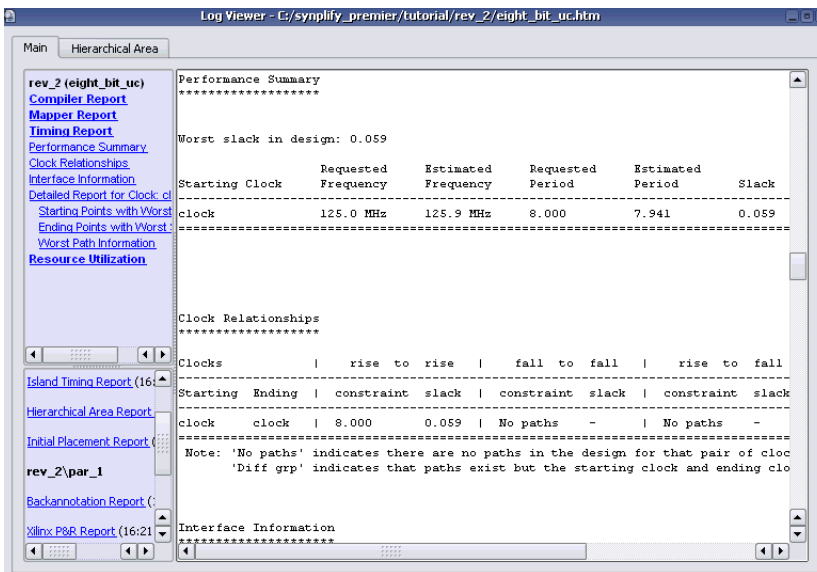
Check initial logic synthesis results. Topics in this section include:

- [Log File](#)
- [Guidelines for Validating Results](#)

Log File

The log file contains default timing and area reports. This section provides the steps required to validate your results.

Click the View Log button in the Project view to display the log file in either text (.srr) or HTML (.htm) format.



Guidelines for Validating Results

Use the following guidelines to validate your results:

1. Was the logic synthesis run successful? (Physical Synthesis switch disabled; successful logic synthesis and place-and-route?)

2. Did you use the correct PAR ISE version for your tool? (See the Release Notes, Help->Online Documents->release_notes.pdf->*Third Party Tool Versions*).
3. Are there black boxes in the design? Search the synthesis .srr log file for black box.

See [Working with Xilinx IP, on page 4-49](#) in the *User Guide* if more information is needed.

4. Are there any combinational feedback loops? Search the synthesis .srr log file for:

```
Found combinational loop
```

Combinational loops cause random timing analysis results that invalidate any comparison and should be eliminated from the design.

5. Are the clock constraints correct? Check the Clock Relationships table in the .srr log file.
6. Are the forward annotated timing constraints (NCF) consistent with the post place-and-route timing constraints?
7. Are the DCM parameters correctly defined in the source code or .sdc constraint file? Check the Clock Relationships table in the .srr log file.
8. Are the false and multi-cycle paths constraints correctly defined in the .sdc file? Ensure that the back-annotation timing report (.srr log file in the PAR directory) matches the .twr report.
9. Are the clocks routed on global resources? Check the Clock Path Skew numbers in the .twr file. Clocks routed on general routing resources usually result in large skews. Because the tool does not take clock skew into account (except for Virtex-5 devices), large skews can degrade the quality of results (QoR) and result in poor timing correlation.

For Virtex5 designs, see [Clock Skew for the Virtex-5 Technology, on page 1-40](#).

See Also

- [Analyze Physical Synthesis Results, on page 1-45](#)

Clock Skew for the Virtex-5 Technology

The Synplify Premier software version 9.0.1 and later supports:

- Clock skews for Virtex-5 devices only
- The clock insertion delay SRM property (Clock input arrival_time). Clock insertion delay models are included for the following components:
 - DCM
 - BUFG
 - BUFR
 - BUFIO
 - Flip-flops generating clocks

This feature ensures that cross-clock paths are compared correctly. Also, it has a large impact on timing constraints for I/O paths, since any clock delay will be added to the output delay and subtracted from the setup delay. Hence, this results in improved timing correlation between the Synplify Premier software and Xilinx timing.

Example 1—Virtex-5 Clock Skew

Clock skew is utilized to calculate the slack in the following example, where:

- Source DCM (clock insertion delay = 0.000ns)
- Load IBUFG (clock insertion delay = 4.157ns)

```
Requested Period: 5.000
- (Setup Time): 0.004
+ (Clock Delay at Ending Point): 4.157
+ (Clock Latency at Ending Point): 0.000
= Required Time: 9.153
- (Propagation Time): 0.746
- (Clock Latency at Starting Point): 0.000
= Slack (non-critical): 8.407
```

Example 2—Virtex-5 Clock Skew

Clock skew is utilized to calculate the slack in the following example, where:

- Source IBUFG (clock insertion delay = 4.157ns)
- Load DCM (clock insertion delay = 0.000ns)

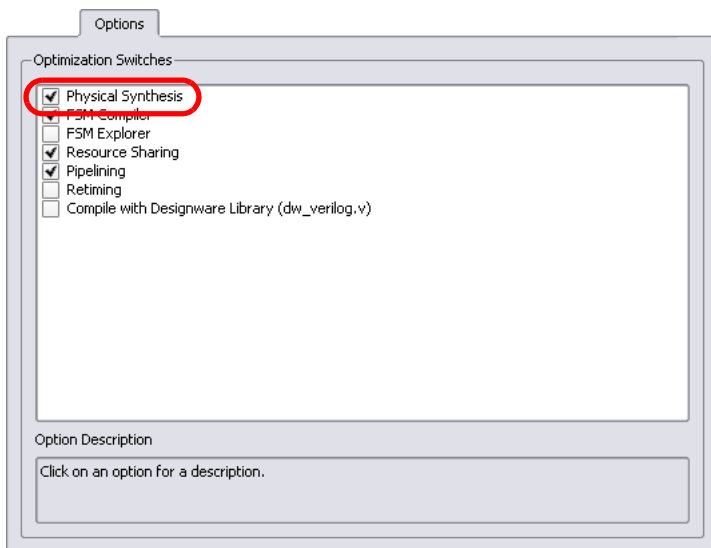
```
Requested Period: 5.000
- (Setup Time): 0.004
+ (Clock Latency at Ending Point): 0.000
= Required Time: 4.996
- (Propagation Time): 0.745
- (Clock Delay at Starting Point): 4.157
- (Clock Latency at Starting Point): 0.000
= Slack (critical): 0.094
```

The Synplify Premier software does not automatically forward annotate constraints for derived clocks. Therefore, a clock generated from a set of flip-flops and logic requires you add a constraint to the UCF file. As a recommendation, derive the clock period the same as the original clock and add a 2-cycle multicycle path from the clock to itself. Better solutions will be provided in the future.

Run Physical Synthesis

Once you complete the logic synthesis validation phase (see [Logic Synthesis Validation Phase, on page 1-11](#) for details), you are ready to run physical synthesis. The project you created for logic synthesis ([Define the Project, on page 1-30](#)) requires these additional steps:

1. Enable the Physical Synthesis switch, located either:
 - in the Project view
 - or
 - Implementation Option ->Options tab



2. If you want a different directory for your physical synthesis results, click on the Implementation Results tab and specify the result options.

The screenshot shows the 'Implementation Results' dialog box. It has a title bar with the text 'Implementation Results'. Inside, there are several fields and controls:

- Implementation Name:** A text box containing 'rev_1'.
- Results Directory:** A text box containing 'C:/Designs/ramCtrl_prod/rev_1' and a 'Browse...' button to its right.
- Results File Name:** A text box containing 'eight_bit_uc.edf'.
- Result Format:** A dropdown menu currently set to 'edif'.
- Optional Output Files:** A section containing three checkboxes:
 - Write Mapped Verilog Netlist
 - Write Mapped VHDL Netlist
 - Write Vendor Constraint File

3. If you are using Design Planner, click on the Design Planning tab and enable the desired design plan file (.sfp).

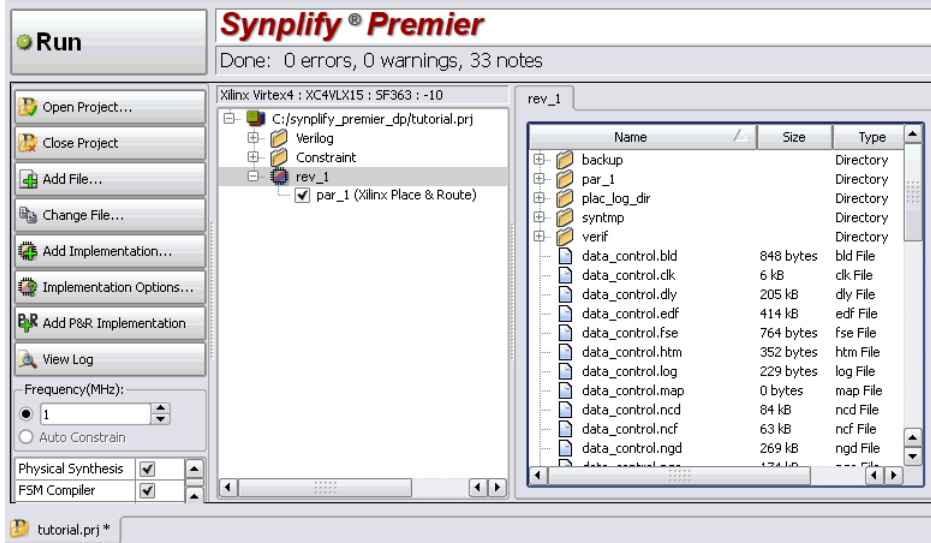
The screenshot shows the 'Design Planning' dialog box. It has a title bar with the text 'Design Planning'. Inside, there is a section titled 'Design Plans: (Select design plan file to apply to this implementation)'. Below this title, there is a list box containing one entry: 'C:/synplify_premier/tutorial.sfp', which has a checked checkbox to its left.

Note: You do not need to select a design plan file to run graph-based physical synthesis. However, if you are using a graph-based flow and want to use a design plan file, see [Create Design Plan File](#), on page 1-28.

For older Xilinx technologies, you must create a design plan (.sfp) to run physical synthesis. See [Design Planner](#), on page 1-51 for more information.

4. Click OK to apply the implementation options.
5. Make sure the place-and-route implementation is enabled, Implementation Options->Place and Route tab.
6. Click Run in the Project view.






Optimizations are performed on the design using placement-aware synthesis. Synthesis and placement are integrated by performing concurrent placement and optimization based on timing constraints and device technology.




7. Analyze results. See [Analyze Physical Synthesis Results](#), next for details.

Analyze Physical Synthesis Results

To determine if your design has met performance goals, use the Synplify Premier analysis tools, which include:

- Log file (.srr or .htm), includes the default timing report
- HDL Analyst:
 - RTL View ()
 - Technology View ()
- Physical Analyst ()
- Timing Report ()
- Island Timing Report ()

Use these tools to analyze the critical path(s) with negative slack and identify potential solutions to improve performance. Use Design Planner to create physical constraints to also aid in improving performance ().

Here are some guidelines for analyzing results:

1. Are start and end points being constrained by the proper clocks?
The timing report is the primary tool for checking this. You can also trace the clock network using HDL Analyst Technology view.
2. Is the critical path a multi-cycle path or false path?
Use the timing report and HDL Analyst tool to get best view of the design's timing.
3. Can pipelining be used to close timing?
Use the HDL Analyst tool; also, see [Pipelining, on page 9-5](#) in the *User Guide* for details on this feature.
4. If the path is inside a state machine, is the FSM being fully optimized?
Use the HDL Analyst. Open the RTL view and push down into the state machine module to display the FSM viewer.
5. Look at the timing report that provides the % breakdown of delay for each path (Do a find on "Total path delay"). Are the net delays contributing to the highest percentage on the critical path?
Use the Physical Analyst to analyze the instance placement of the critical path.

6. Can performance be improved using a physical constraints file? Use Physical Analyst and Design Planner to determine if constraining logic to specific regions can provide improved performance.

For more details and guidelines on improving design performance, see [Improve Performance, on page 1-48](#).

See Also

- [Log File, on page 1-38](#)
- [Guidelines for Validating Results, on page 1-38](#)

HDL Analyst

The RTL and Technology views provide schematics to analyze the design.

Display the RTL schematic for a compiled design (compile phase complete only). Select HDL Analyst->RTL->Hierarchical View or ->Flattened View.

Display the Technology schematic for a synthesized design (technology mapping complete). Select HDL Analyst ->Technology->Hierarchical View, or ->Flattened View.

For an overview of using the HDL Analyst views, see:

- [HDL Analyst Views and Commands, on page 6-2](#) in the *User Guide*
- [Finding Schematic Objects, on page 6-13](#) in the *User Guide*
- [Basic Operations on Schematic Objects, on page 6-13](#) in the *User Guide*
- [Exploring Design Hierarchy, on page 6-21](#) in the *User Guide*

Stand-alone Timing Analyst

You can run the stand-alone timing analyzer to produce a timing report (.ta) that displays more or less information than the default timing report in the log file (.srr). Use the stand-alone timing analyzer for your more specific report requirements.

- Select Analysis->Timing Options.
- Fill in the parameters for the report (see [Timing Report Generation Parameters](#), on page 3-92 in the *Reference* for details on completing the fields).
- Click Generate to run the report.

For more information, see [Using the Stand-alone Timing Analyst](#), on page 14-6 in the *User Guide*.

Physical Analyst

The Physical Analyst provides a visual display of the device and design placement. Select HDL Analyst->Physical Analyst. The Physical Analyst view can display instances and nets. For complete details, see [Chapter 13, Analyzing Designs in Physical Analyst](#) in the *User Guide*.

Island Timing Analyst

Use the Island Timing Analyst to generate a hierarchical display for groups of connected critical paths called islands. The island timing report is useful when you are creating a design plan and for analyzing critical paths (routing versus logic delay), because it identifies the instances or pins belonging to multiple paths and how the critical paths in an island group are connected. You can also cross probe these critical paths to the HDL Analyst view. Select HDL Analyst->Island Timing Analyst. For details, see [Using the Island Timing Analyst](#), on page 14-12 of the *User Guide*.

Improve Performance

The Synplify Premier tool is timing-driven; optimizations depend on timing constraints and are applied until all constraints are met. Therefore, it is very important that you adequately apply timing constraints and not over-constrain the tool. This section includes guidelines for applying constraints.

- Verify constraints consistency between synthesis and P&R:
 - Clock constraints
 - Clock-to-clock constraints
 - IO delays
 - IO standard, drive, slew and pull-up/pull-down
 - Multi-cycle and false paths
 - Max-delay paths
 - DCM parameters
 - Register packing into IOB
 - LOC on IO pads
 - LOC/RLOC constraints on macros (BUFG, DCM, RAMB, DSP, MULT, etc.)
 - LOC/RLOC constraints on instances (Register, LUT, SRL, RAMS, RAMD, etc.)
 - AREA_GROUP constraints
 - IDELAYCTRL and IDELAY constraints
- Ensure the final physical synthesis slack is negative, but no more than 10-15% of the clock constraint.
- Check the log file for the “Pre-placement timing snapshot.”
If it indicates that a clock has positive slack at this point, but in the final results the clock has negative slack, use the `-route` constraint for the clock. This option allows you to control the amount of early timing optimizations for the clock domain. However, large `-route` values can degrade performance. Therefore, to determine the correct `-route` value to use, start with smaller values and increase iteratively. For example, start with half the difference between the estimate and actual slack, or 5% of the clock estimate, whichever is the smallest.

- Experiment with ignoring the relationally-placed macro (RPM) constraints.

RPMs (also known as RLOCs) can negatively affect results. You can compare placement results using the Synplify Premier tool by setting the global attribute `xc_use_rpms` to 0. For details on this attribute, see [xc_use_rpms Attribute, on page 8-271](#) in the *Reference*.

- Ensure placement for I/Os, Block Rams, and DSP48 devices.

This version of the tool uses the Xilinx placer to generate locations for I/Os and block components. To avoid block component placement problems, you need to lock placement. See [Coreloc – Core Locked-Placement Constraints, on page 1-24](#) for information.

Performance Results Comparison

Synplify Premier physical synthesis provides a timing closure solution that yields more accurate timing correlation and faster timing closure for your design. This section provides details on how to analyze results from logic synthesis, physical synthesis and place and route to show more accurate timing correlation between physical synthesis and final place and route results. The diagram below provides an overview of how to use the Synplify Premier tool and its features to analyze performance.

See Also

[Improve Performance, on page 1-48](#), for details on how to use the tools features for determining how to enhance performance.

Running Multiple Implementations

You can create multiple implementations of the same design so that you can compare the results of each implementation and place-and-route run. This lets you experiment with different settings for the same design with different place-and-route options. Implementations are revisions of your design within the context of the Synplify Premier software and do not replace external source code control software and processes.

For the Graph-based physical synthesis with a design plan flow (Virtex-II Pro, Spartan-3, Virtex-4, and Virtex-5), you can run the first pass using the Synplify Premier software without a design plan file (.sfp) to synthesize the design. Placement and routing runs automatically. Then, create a new implementation and apply a design plan for Design plan-based physical synthesis.

See [Working with Multiple Implementations, on page 6-10](#) of the *User Guide* for more information.

Design Planner

If you have the Synplify Premier tool with the Design Planner option you can specify placement constraints to guide global placement. Physical constraints, specified in a Design Plan file (.sfp), constrain logic to specified regions on the device. The .sfp file also serves as a guide for initial placement for the following objects types:

- - I/Os
- - RAMs
- - ROMs
- - DSPs
- - IP modules
- - clock pins

Note: For Xilinx technologies that support graph-based physical synthesis, the .sfp file is not required because you can run full-chip physical synthesis in a single-pass flow without the need for a .sfp file.

See [Supported Xilinx Devices, on page 1-10](#) to determine the physical synthesis flows that can be used with the specific Xilinx technologies and consider using the design planner option if:

- You are using a Xilinx technology that does not support graph-based physical synthesis thus requiring the design-plan based physical synthesis flow. In a design-plan based flow, effective manual placement of the critical path is required to improve the design using placement-based optimization, register replication for high fanouts, and register tunneling across region boundaries. See [Design-Plan Based Physical Synthesis Flow, on page 1-52](#) for details.
- You are using a Xilinx technology that supports graph-based physical synthesis, but you want to apply physical constraints to guide global placement.

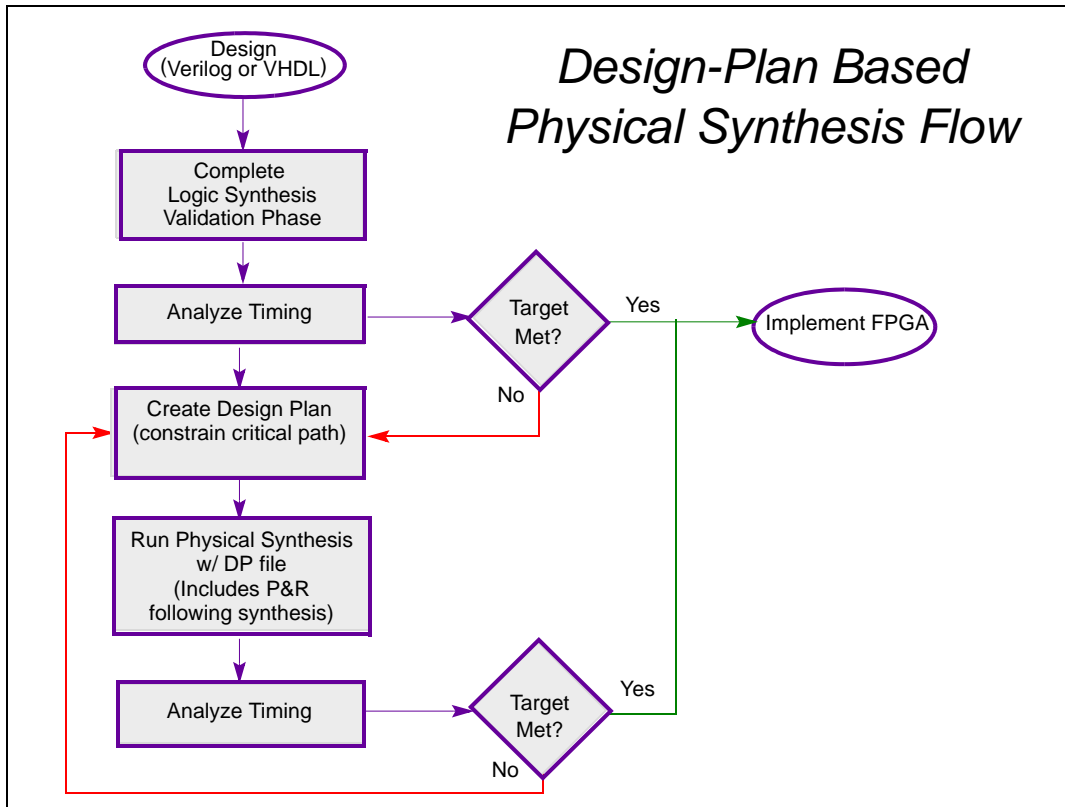
Design-Plan Based Physical Synthesis Flow

Use this flow if you have the Synplify Premier tool with the Design Planner option and are using any of the following Xilinx technologies:

- Virtex
- Virtex-II
- Virtex-E

For all other Xilinx technologies, see [Supported Xilinx Devices, on page 1-10](#) for the flow to use with your technology.


The figure below shows the design-plan based flow; accompanying task descriptions follow the flow diagram.



Design Tasks

These tasks reflect the flow diagram above:

1. Synthesize the design in logic synthesis mode—using timing constraints and no physical constraints. This phase is to determine if the design can successfully complete synthesis and if timing performance enhancements are needed. The logic synthesis validation phase includes running the netlist through place-and-route after synthesis completes. For details on how to complete this phase, see:
 - [Logic Synthesis Validation Phase, on page 1-11](#)
 - [Define the Project, on page 1-30](#)
2. Analyze timing results. See [Validate Logic Synthesis Results, on page 1-38](#) for details.

If timing goals are met, you are done. Otherwise, go to the next step.
3. Determine the critical paths from PAR; these are the candidates for logic assignments to regions.
4. Bring up the Design Planner () and:
 - Create regions for the critical paths and interactively assign the critical paths to regions of the chip. See [Working with Regions, on page 10-19](#) of the *User Guide* for details.
 - Obtain a size estimation for each RTL block in the design. See [Checking Utilization, on page 10-29](#) of the *User Guide* for details.
 - For multiple clocks, assign critical logic associated with each clock domain (that does not meet design requirements) to a unique region to avoid resource contention.

You can also bring up Physical Analyst to view the design and critical path placement.

Consult the following sections of the *User Guide* for more information on how to complete the Design Plan file (.sfp).

- [Creating and Using a Design Plan File for Physical Synthesis, on page 10-8](#)
- [Working with Regions, on page 10-19](#)
- [Assigning Pins and Clocks, on page 10-9](#)

5. Save the design plan file (.sfp) and add it to your project.
6. Run physical synthesis. Use the same project file that you created in step 1 above. This time enable the Physical Synthesis switch and include the physical constraints file (.sfp). This phase also includes running the netlist through place-and-route after synthesis completes.
7. Analyze the timing in the Synplify Premier tool. Use the log file and graphical analysis tools. See [Analyze Physical Synthesis Results, on page 1-45](#) for details.

If the target is met, you can continue to the next design phase. If not, you should re-evaluate timing and placement. Perhaps there is a new critical path or the one that is already assigned to regions needs tweaking. See [Improve Performance, on page 1-48](#) for more suggestions.