

# **Synopsys FPGA Synthesis**

## **Synplify Premier**

### **Quick Start Guide for Xilinx**

December 2009

<http://www.solvnet.com>

**SYNOPSYS®**

---

## Disclaimer of Warranty

Synopsys, 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 © 2009 Synopsys, Inc. All Rights Reserved.

Synopsys software products contain certain confidential information of Synopsys, 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 Synopsys, Inc. While every precaution has been taken in the preparation of this book, Synopsys, Inc. assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

## Trademarks

### Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, Design Compiler, DesignWare, Formality, HDL Analyst, HSPICE, Identify, iN-Phase, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SiVL, SCOPE, Simply Better Results, SNUG, SolvNet, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

### Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, Direct Silicon Access, Discovery, Eclipse, Encore, EPIC, Galaxy, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierar-

---

chical Optimization Technology, High-performance ASIC Prototyping System, HSIM, HSIM<sup>plus</sup>, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC are trademarks of Synopsys, Inc.

## **Service Marks (SM)**

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license. ARM and AMBA are registered trademarks of ARM Limited. Saber is a registered trademark of SabreMark Limited Partnership and is used under license. All other product or company names may be trademarks of their respective owners.

## **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 Synopsys Software License and Maintenance Agreement. Synopsys, Inc., Synplicity Business Group, 700 East Middlefield Road, Mountain View, CA 94043 USA.

December 2009



# Contents

---

FPGA Implementation Solution	8
Timing Closure	8
Single-Chip ASIC Verification	9
System Design	9
Design Analysis	9
Comprehensive HDL Support	10
RTL Debugging	11
DSP Synthesis	12
Synplify Premier Overview	13
Physical Synthesis Flow Diagram	14
Types of Physical Synthesis Flows	15
Supported Xilinx Devices	16
About the Xilinx ISE Place-and-Route Tool	16
Logic Synthesis Validation Phase	17
Graph-based Physical Synthesis	18
Graph-based Physical Synthesis with Design Planner	20
Design-plan Based Physical Synthesis	22
Physical Synthesis Setup	23
Timing Constraints Guidelines	24
Setting Up Constraints	25
Using the -route Constraint	27
Using IP Cores	28
Understanding Synplify Premier Placement and Routing	32
Create Design Plan File	34
Physical Synthesis Flow—Task Summary	35
Define the Project	36
Create Project	37
Set up Timing and Physical Constraints	37
Set Implementation Options	38
Create Place and Route Implementation	40
Specify Global Placement Options	42

Run Logic Synthesis .....	43
Validate Logic Synthesis Results .....	44
Log File .....	44
Guidelines for Validating Results .....	45
Clock Skew for the Virtex-5 Technology .....	46
Run Physical Synthesis .....	48
Improve Performance .....	54
Performance Results Comparison .....	55
Running Multiple Implementations .....	56
Design Planner .....	57
Design-Plan Based Physical Synthesis Flow .....	58

## 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, on page 8](#)
- [Synplify Premier Overview, on page 13](#)
- [Physical Synthesis Flow Diagram, on page 14](#)
- [Physical Synthesis Setup, on page 23](#)
- [Physical Synthesis Flow—Task Summary, on page 35](#)
- [Define the Project, on page 36](#)
- [Run Logic Synthesis, on page 43](#)
- [Validate Logic Synthesis Results, on page 44](#)
- [Run Physical Synthesis, on page 48](#)
- [Analyze Physical Synthesis Results, on page 51](#)
- [Improve Performance, on page 54](#)
- [Design Planner, on page 57](#)
- [Design-Plan Based Physical Synthesis Flow, on page 58](#)

# 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](#)
- [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 retargeting 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.

## 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 FGPA. 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 FGPA.

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 on the specified hardware.

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

# Synplify Premier Overview

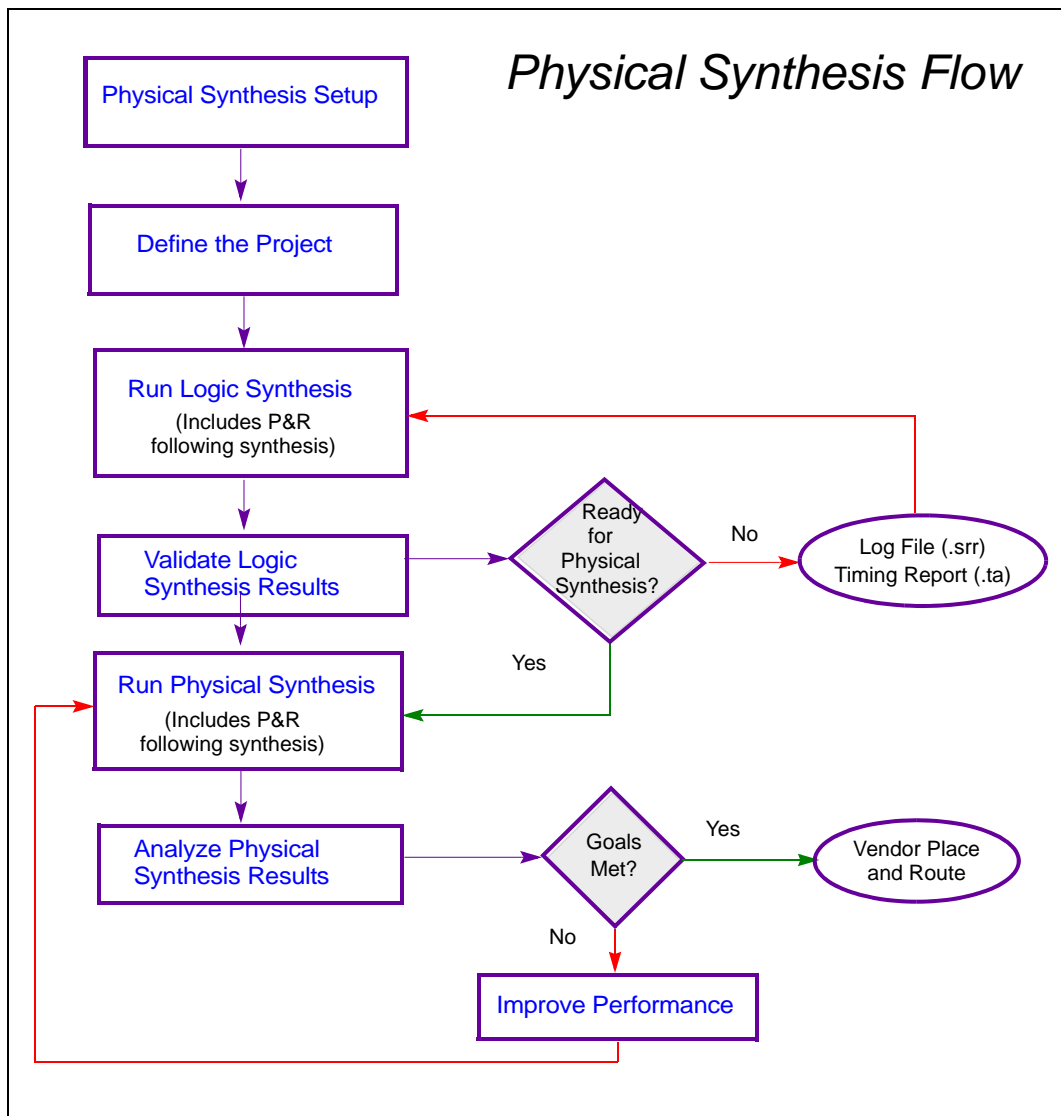
The Synplify Premier product is a physical synthesis timing closure solution that provides more accurate timing correlation and faster timing closure than could be achieved through previous design methodologies. This tool offers a push-button, graph-based design flow for improving overall device performance 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 16](#)
  - [About the Xilinx ISE Place-and-Route Tool, on page 16](#)
- For details on the flows, see [Logic Synthesis Validation Phase, on page 17](#)

## 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 Virtex-6 (Beta)
Graph-based Physical Synthesis with Design Planner <sup>1</sup>	Spartan-3 Virtex-II Pro Virtex-4 Virtex-5
Design Plan based Physical Synthesis <sup>1</sup>	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 download page on SolvNet, select Release Notes-> *Third Party Tool Versions*)



## Logic Synthesis Validation Phase

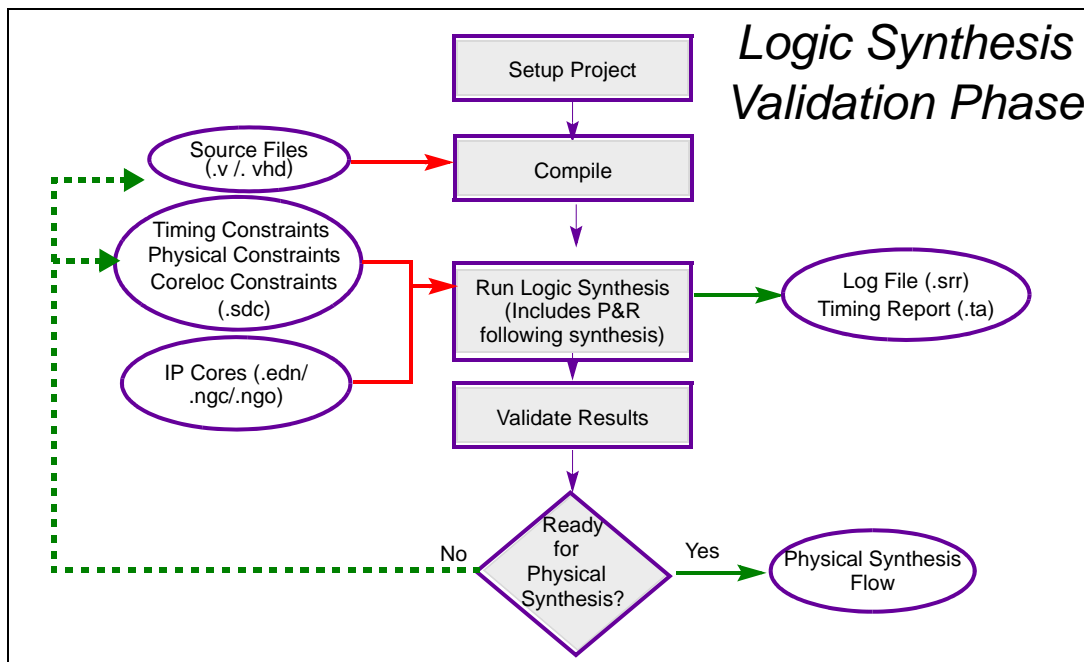
Always begin any physical synthesis flow with the logic synthesis validation phase. This is to ensure that a design can successfully complete synthesis and an initial place and route before going into the physical synthesis process. This can save valuable time by identifying obvious problems early in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

Run Synplify Premier logic synthesis with one of the following flows:

- Synplify Pro logic synthesis
- Logic synthesis with Fast Synthesis
- Logic synthesis with Enhanced Optimization

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 35 for a summary of tasks required to complete this flow.
- [Define the Project](#), on page 36 to start the physical synthesis design flow.
- [Synplify Premier Synthesis Design Flows](#), on page 35 for a description of the Synplify Premier logic synthesis flows.

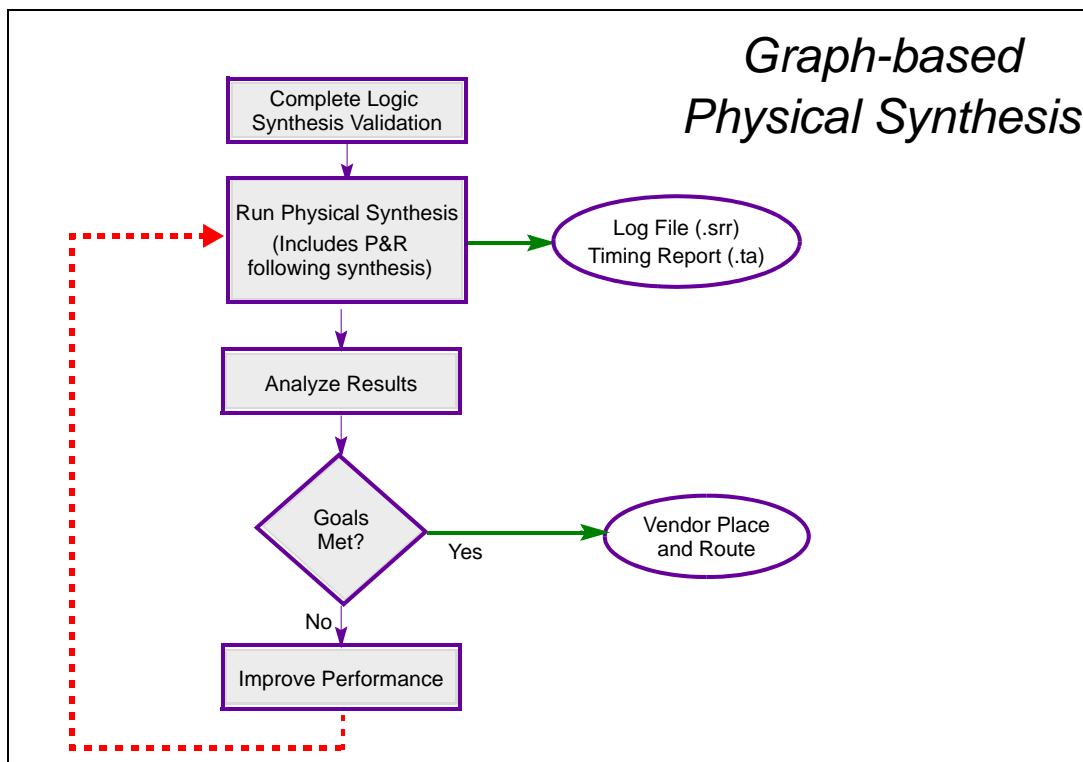
## Graph-based Physical Synthesis

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
- Virtex-6 (Beta)

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 35 for a summary of tasks required to complete this flow.
- [Run Physical Synthesis](#), on page 48 to start the physical synthesis design flow.

## Graph-based Physical Synthesis with Design Planner

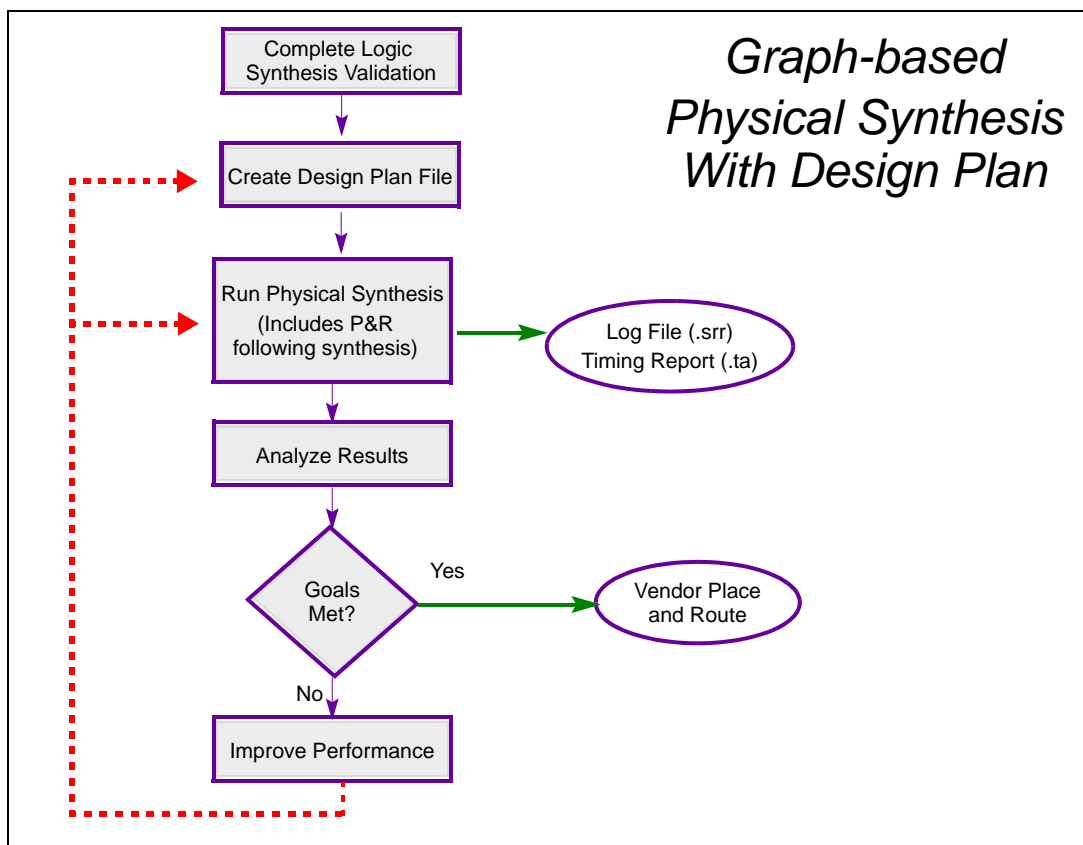
This flow is available when you have the Design Plan option with the Synplify Premier tool and can be used with the following Xilinx technologies:

- 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 57 for general information on this feature.
- [Create Design Plan File](#), on page 34 for information on how to create a design plan file.
- [Physical Synthesis Flow—Task Summary](#), on page 35 for a summary of tasks required to complete the flow.
- [Run Physical Synthesis](#), on page 48 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 58](#) 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 \(.sdc\)](#)
  - [Convert Xilinx UCF 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.




## Setting Up Constraints

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

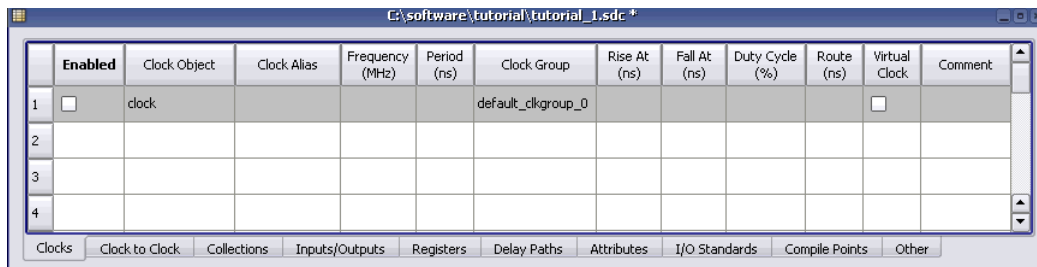
- [Create the Timing Constraint File \(.sdc\)](#)
- [Convert Xilinx UCF Constraints](#)
- [Run Constraint Check](#)

### 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 37](#)).
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 pulldowns 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 110](#) 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

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

## Convert Xilinx UCF Constraints

When you use UCF constraints in a physical design, it is assumed that you did a baseline logic synthesis run with place-and-route, and that you have certain files for the design. This application creates a new project that contains all supported UCF constraints converted to SDC constraints. However, your original project and project files are not altered.

1. To convert UCF constraints, select Project->Convert Vendor Constraints, specify parameters on the UCF to SDC Conversion dialog box and click the Convert button.

For more information, see [Converting and Using Xilinx UCF Constraints, on page 155](#) in the (*User Guide*).

2. 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 <sup>1</sup>:
    - 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

3. Also include any synthesis attributes from logic synthesis, such as `syn_ramstyle`, into the `.sdc` file.
4. 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 28](#).
- Used at a global level in the Synplify Premier tool to produce a better netlist during global placement. See [Physical Synthesis Usage, on page 28](#).

- 
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.

## 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 inter-connect 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 cgl -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 40](#).
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 844](#) 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 Xilinx Cores for Logic and Physical Synthesis, on page 845](#) and [The Synplify-EDK Design Flow, on page 849](#) in the *User Guide*.

## See Also

- [Working with Regions, on page 405](#) in the *User Guide* for more details defining physical constraints for cores using the Design Planner.
- [Working with Xilinx IP, on page 844](#) 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 29](#) 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 Regions, on page 405](#) 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)	Allows components to be moved across the boundaries in both directions. This is the default.
Hard (Tunneling Off)	Ensures that components are not moved out of the region, but allow other objects to be moved into the region.
Keep-out	Ensures 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 48](#) 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 36](#)
  - [Run Logic Synthesis, on page 43](#)
  - [Validate Logic Synthesis Results, on page 44](#)

# 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 17](#).

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 58](#) 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 17](#)
- [Graph-based Physical Synthesis, on page 18](#)
- [Graph-based Physical Synthesis with Design Planner, on page 20](#)

# 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 844](#) in the *User Guide* for more information.
- Make sure the design is properly constrained. (See [Improve Performance, on page 54](#) 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 16](#).
- Use the top-down design methodology. (A bottom-up flow is not supported.)

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/ .ngo/ .ngc (core IP files – see [Working with Xilinx IP, on page 844](#) 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 57](#) for details).
4. Save the project file.

For details on creating a project file, see:

- [Setting Up HDL Source Files, on page 84](#) in the *User Guide*
- [Setting Up Project Files, on page 170](#) in the *User Guide*

## Set up Timing and Physical Constraints

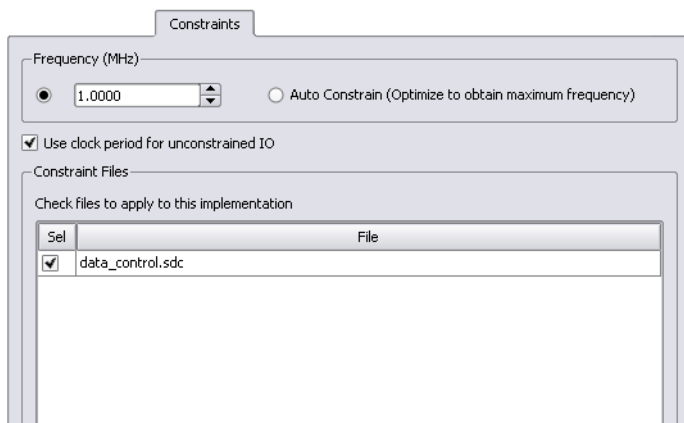
Constraints for physical synthesis include timing constraints and physical constraints for your design. Timing constraints are used to specify performance goals and describe the design environment. Xilinx physical constraints, as well as all constraints from the UCF should also be included in the project 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 UCF to use for physical synthesis, see [Setting Up Constraints, on page 25](#).

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 191](#) 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 193](#) 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 195](#) of the *User Guide* for details.

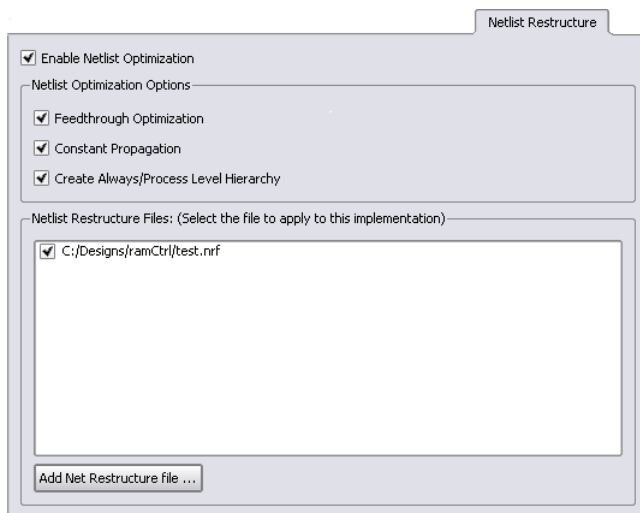
6. In the Timing Report panel specify the number of critical paths and start/end points to display in the timing report.



The screenshot shows the 'Timing Report' panel. At the top, there is a tab labeled 'Timing Report'. Below the tab, there are two input fields: 'Number of Critical Paths:' and 'Number of Start/End Points:'. The main area of the panel is a large, empty light gray rectangle. At the bottom, there is a 'Description' section with the text: '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.'

7. In the Verilog/VHDL panel, specify the desired HDL options. See [Setting Verilog and VHDL Options, on page 197](#) 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 230](#) 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 16](#) for information.
2. Set the XILINX and PATH environment variables to point to a valid installation of the place and route tool.

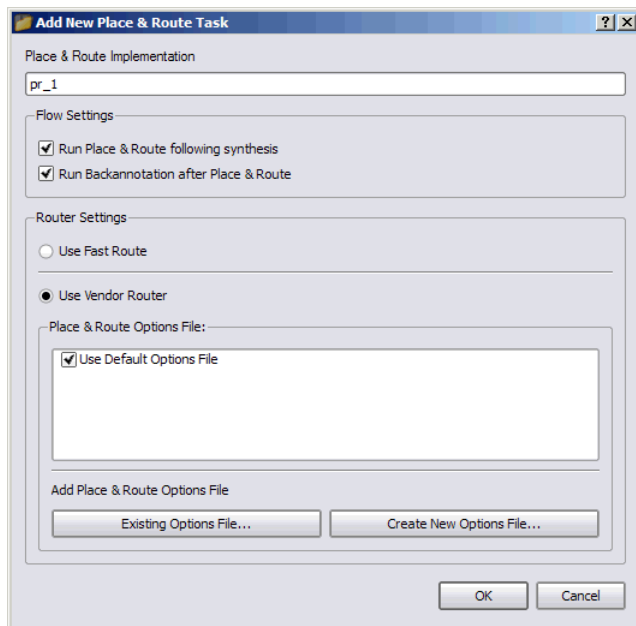
Note, the Synplify Premier UI automatically sets the environment variable for:



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.

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



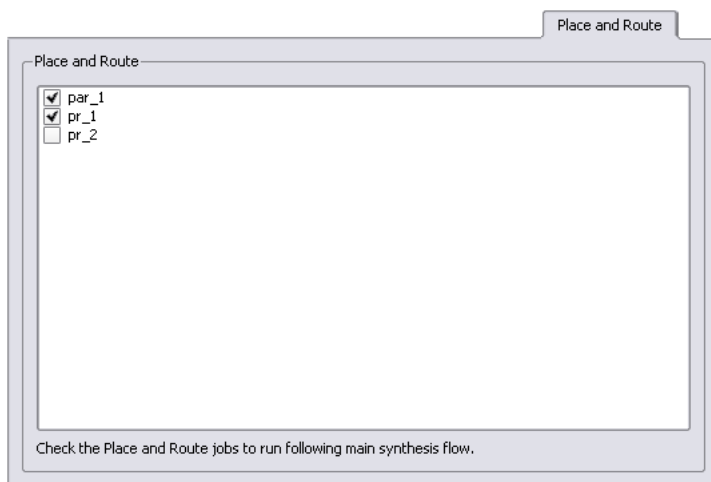
4. Specify the Place & Route Job Name. Default is `pr_n`.
5. Make sure the Run Place and Route following synthesis switch is enabled.
6. If you want to generate a core location file (recommended), enable the Back Annotate placement and timing data following Place & Route switch. See [Coreloc – Core Locked-Placement Constraints, on page 29](#), for information about this file.
7. For Xilinx devices, you can choose the Router Setting of:
  - Fast Route — Virtex-5 devices only (See [Synplify Premier Fast Route \(Beta\), on page 709](#) for more information.)
  - Use Vendor Router — This option is selected in this flow.

- Specify the place-and-route options file. The tool automatically uses default options located in

```
<install_directory>\lib\xilinx\*.tcl
```

This file is used by the Xilinx xtcsh executable to run the P&R tool. You can change or override the default options. See [Specifying Xilinx Place-and-Route Options in a Tcl File, on page 240](#) in the *User Guide* for details.

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



- 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 37](#).
- Specify constraints – [Set up Timing and Physical Constraints, on page 37](#).
- Set implementation options – [Set Implementation Options, on page 38](#).

Select a logic synthesis mode: Synplify Pro logic synthesis, Enhanced Optimization, or Fast Synthesis. For details, see [Synplify Premier Synthesis Design Flows, on page 35](#) in the *User Guide*.

- Setup for place and route – [Create Place and Route Implementation, on page 40](#).

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 44](#).

# Validate Logic Synthesis Results

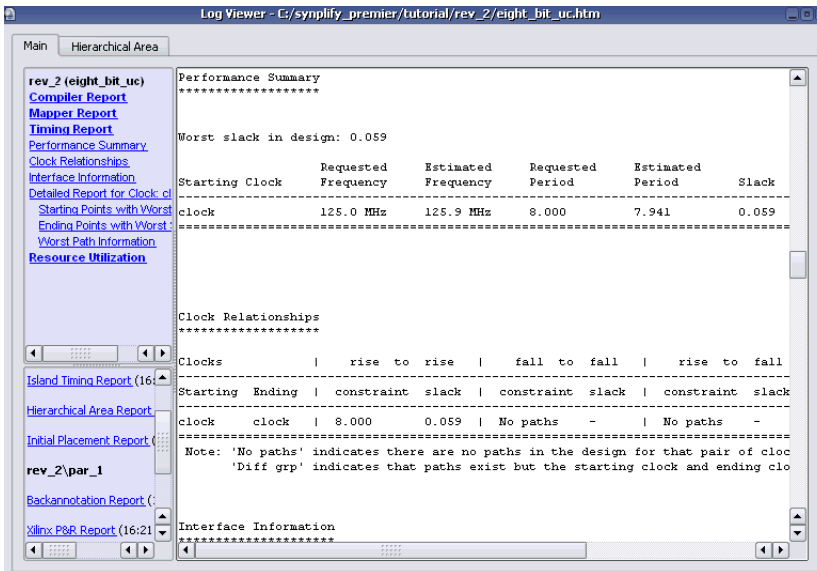
Check initial logic synthesis results. Topics in this section include:

- [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? (From the Synplify Premier tool download page on SolvNet, select Release Notes-> *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 844](#) 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 46](#).

## See Also

- [Analyze Physical Synthesis Results, on page 51](#)

## 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

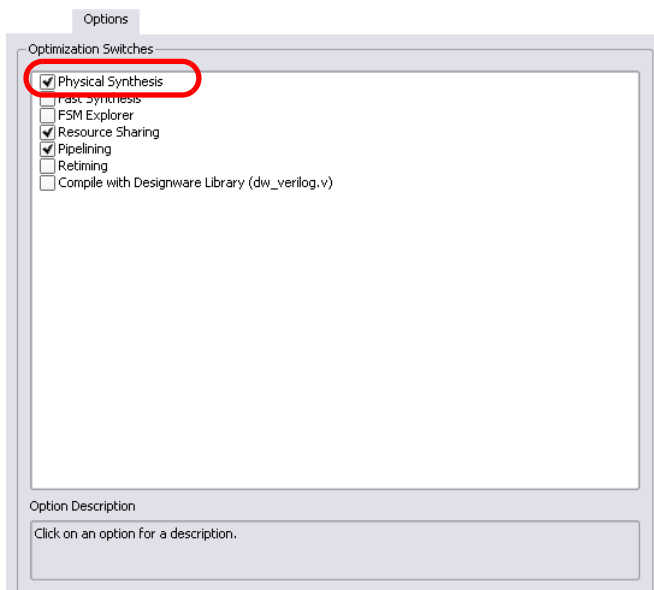
= Required Time: 4.996
- (Propagation Time): 0.745
- (Clock Delay at Starting Point): 4.157
- (Clock Latency at Starting Point): 0.000
= Slack (critical): 0.094
```

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

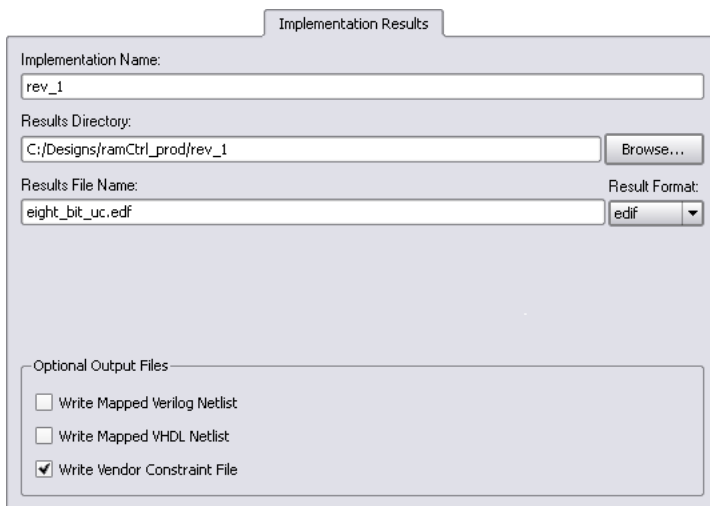
Once you complete the logic synthesis validation phase (see [Logic Synthesis Validation Phase, on page 17](#) for details), you are ready to run physical synthesis. The project you created for logic synthesis ([Define the Project, on page 36](#)) requires these additional steps:

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





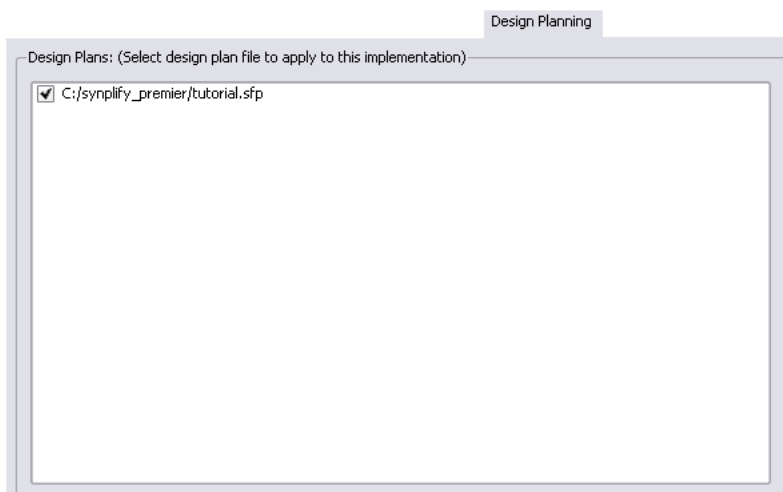
2. If you want a different directory for your physical synthesis results, click on the Implementation Results tab and specify the result options.



The **Implementation Results** dialog box contains the following fields and options:

- Implementation Name:** rev\_1
- Results Directory:** C:/Designs/ramCtrl\_prod/rev\_1 (with a **Browse...** button)
- Results File Name:** eight\_bit\_uc.edf
- Result Format:** edf (dropdown menu)
- Optional Output Files:**
  - ☐ 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).



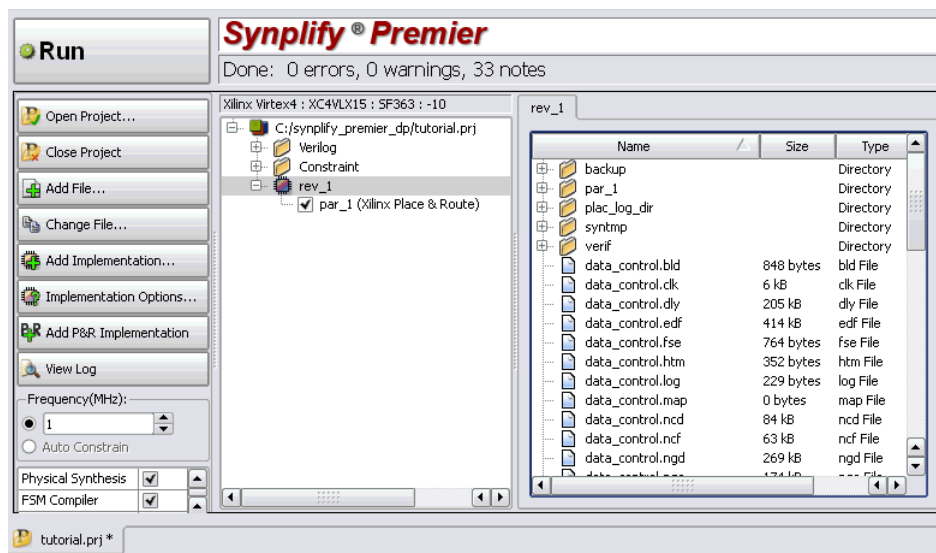
The **Design Planning** dialog box contains the following field:

- Design Plans: (Select design plan file to apply to this implementation)**
  - ☒ C:/synplify\_premier/tutorial.sfp

**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 34](#). For older Xilinx technologies, you must create a design plan (.sfp) to run physical synthesis. See [Design Planner, on page 57](#) 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 ()

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 332](#) 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 54](#).

## See Also

- [Log File, on page 44](#)
- [Guidelines for Validating Results, on page 45](#)

## 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 360](#) in the *Reference Manual*
- [Finding Schematic Objects, on page 371](#) in the *Reference Manual*
- [Basic Operations on Schematic Objects, on page 371](#) in the *Reference Manual*
- [Exploring Design Hierarchy, on page 379](#) in the *Reference Manual*

## 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 Analyst.
- Fill in the parameters for the report (see [Timing Report Generation Parameters, on page 272](#) 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 602](#) 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 12, Analyzing Designs in Physical Analyst](#) in 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 1213](#) 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 29](#) 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 54](#), 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 184](#) 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 16](#) 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 58](#) 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.

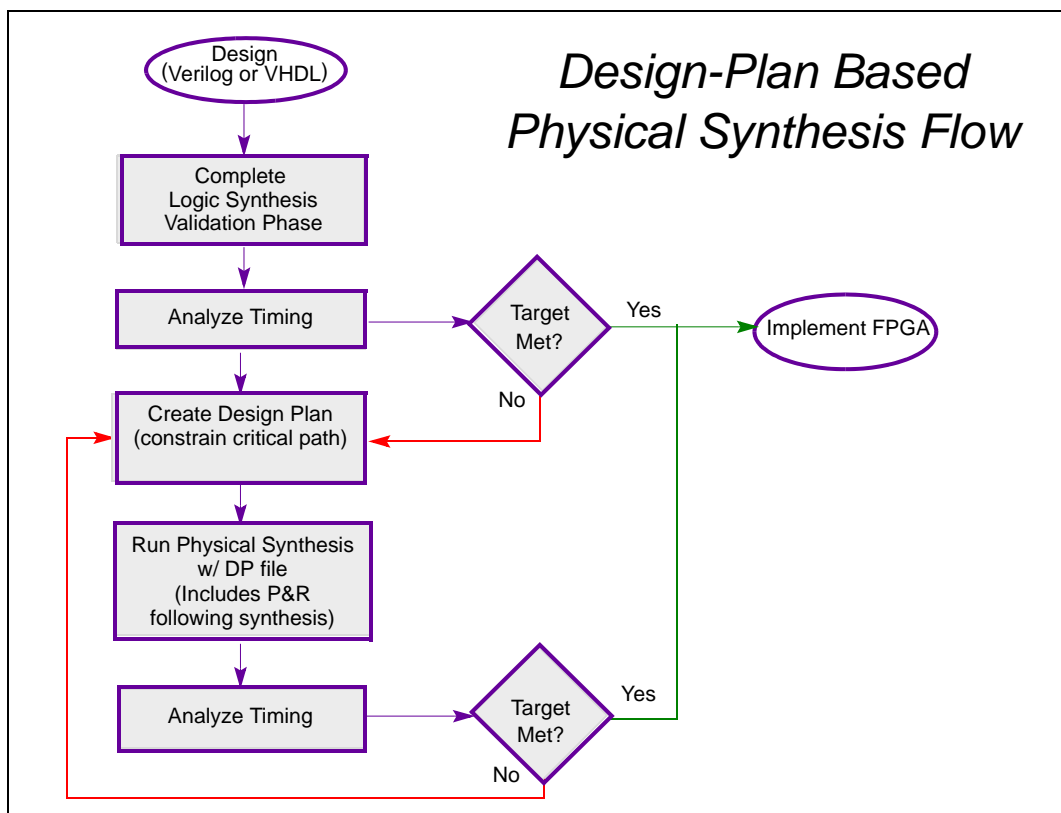
## Design-Plan Based Physical Synthesis Flow

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 16](#) and [Physical Synthesis Flow Diagram, on page 14](#) 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 17](#)
- [Define the Project, on page 36](#)

2. Analyze timing results. See [Validate Logic Synthesis Results, on page 44](#) 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 405](#) of the *User Guide* for details.
- Obtain a size estimation for each RTL block in the design. See [Checking Utilization, on page 416](#) 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 394](#)
- [Working with Regions, on page 405](#)
- [Assigning Pins and Clocks, on page 395](#)

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 51](#) 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 54](#) for more suggestions.

# Index

---

## C

- clock path skew (Synplify Premier) [45](#)
- clock skew (Synplify Premier) [45](#)
  - Virtex5 technologies [46](#)
- clock skew example (Synplify Premier) [46, 47](#)
- constraints
  - checking (Xilinx) [27](#)
- coreloc constraints (Xilinx) [29](#)

## D

- design-plan based physical synthesis flow (Xilinx) [22](#)

## E

- environment variables
  - PAR\_BELDLRPT [41](#)

## G

- graph-based physical synthesis flow (Xilinx) [18](#)
- graph-based w/ dp physical synthesis flow (Xilinx) [20](#)

## I

- IP cores
  - EDN [30](#)
  - NGO [30](#)
  - using in Synplify Premier [30](#)
- IPcores
  - NGC [30](#)

## P

- PAR\_BELDLRPT [41](#)
- physical synthesis

- analyze results (Xilinx) [51](#)
- constraints setup (Xilinx) [37](#)
- coreloc (Xilinx) [29](#)
- create PAR implementation (Xilinx) [40](#)
- defined (Xilinx) [13](#)
- design flow task summary (Xilinx) [35](#)
- design plan file (Xilinx) [34](#)
- design-plan based flow (Xilinx) [22](#)
- flow diagram (Xilinx) [14](#)
- flows (Xilinx) [17](#)
- graph-based flow (Xilinx) [18](#)
- graph-based w/ dp flow (Xilinx) [20](#)
- implementation options (Xilinx) [38](#)
- improve performance (Xilinx) [54](#)
- logic synthesis validation phase (Xilinx) [17](#)
- run constraint checker (Xilinx) [25](#)
- timing constraint guidelines (Xilinx) [24](#)
- timing constraints (Xilinx) [25](#)
- translate UCF (Xilinx) [25](#)
- using -route constraint (Xilinx) [27](#)
- validate results (Xilinx) [44](#)
- Xilinx supported devices [16](#)
- physical synthesis (Xilinx)
  - timing constraint guidelines [24](#)

## S

- secure/non-secure IP cores (Xilinx) [30](#)
- Synplify Premier
  - clock skew [46](#)
  - description (Xilinx) [15](#)
  - physical synthesis flow (Xilinx) [17](#)
  - supported devices (Xilinx) [16](#)

## X

- Xilinx device support
  - physical synthesis [16](#)
- Xilinx IP cores (Synplify Premier) [30](#)

