

# SmartXplorer for ISE Project Navigator Users

*Tutorial (ISE 11.2)*

UG689 (v1.1) July 20, 2009



Xilinx is disclosing this user guide, manual, release note, and/or specification (the "Documentation") to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU "AS-IS" WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© 2009 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/12/09	1.0	Initial Xilinx release.
7/20/09	1.1	Updated for 11.2



---

# Table of Contents

---

## **Preface: About This Guide**

Guide Contents .....	7
Additional Resources .....	8
Conventions .....	8
Typographical .....	8
Online Document .....	9

## **Chapter 1: SmartXplorer Overview**

Introduction .....	11
Key Benefits .....	11
Design Strategies .....	11
Exploiting Parallel compute platforms .....	12
Linux OS .....	12
Microsoft Windows OS .....	12
Using a Single Linux or Windows Machine .....	12

## **Chapter 2: Tutorial Description**

Knowledge Prerequisites .....	16
-------------------------------	----

## **Chapter 3: Preparing the Design for Labs**

Design Description .....	17
Instructions (Linux & Windows) .....	17

## **Chapter 4: Lab 1: Basic Flow**

Objectives .....	19
Lab .....	19
Step 1: Open Lab project .....	19
Step 2: Run SmartXplorer with Default Options .....	19
Step 3: Using SmartXplorer Results .....	23
Step 4: Running Additional Iterations to Improve Timing .....	25
Step 5: Run the Predefined Strategies and Additional Iterations at Once .....	27
Conclusion .....	27

## **Chapter 5: Lab 2: Creating Custom Strategies**

Objectives .....	29
Lab .....	29
Step 1: Open Lab Project .....	29
Step 2: Identify the Best Two Strategies .....	29
Step 3: Creating the Custom Strategy File .....	30
Conclusion .....	32

---

## Chapter 6: Lab 3W: Running Multiple Strategies in Parallel (Windows)

Objectives .....	33
Lab .....	33
Step 1: Open Lab Project .....	33
Step 2: Run SmartXplorer Enabling Two Parallel Runs .....	34
Conclusion.....	35

## Chapter 7: Lab 3L: Running Multiple Strategies in Parallel (Linux)

Objectives .....	37
Lab .....	37
Step 1: Key Items – Xilinx Environment and Results Storage .....	37
Step 2: Pre-run Checklist .....	38
Step 3: Open the Lab Project .....	38
Step 4: Run SmartXplorer on a Regular Linux Network .....	39
Step 5: Run SmartXplorer on LSF or SGE .....	40
Conclusion.....	41

## Appendix A: Custom Files

Objectives .....	43
Custom Strategy File .....	43
Host List Files (Linux) .....	44
Regular Linux Network .....	44
LSF Compute Farm .....	44
SGE Compute Farm .....	44
SmartXplorer Configurations for Various Tasks .....	45
Task 1: Run All Built-in Predefined Strategies .....	45
Task 2: Run the first 3 Built-in Predefined Strategies .....	46
Task 3: Run all built-in, predefined strategies and five additional iterations with different Cost Tables .....	47
Task 4: Run all custom strategies and 3 additional iterations with different Cost Tables .....	48

# About This Guide

---

The goal of this tutorial is to provide a quick introduction to SmartXplorer and how its capabilities can be used to help achieve timing closure.

This tutorial is delivered in two forms, each centered on the main user flow using the ISE® toolset:

- The first is targeted to ISE Project Navigator users. In this section, we show how SmartXplorer can be used from Project Navigator.
- The second is targeted to command line users. Since a majority of Microsoft Windows users use Xilinx tools from the ISE Project Navigator environment, for command line users, we will mainly focus on using SmartXplorer on the Linux platform. Of course, you can easily adapt the provided material for Microsoft Windows operating systems, if required.

Both forms contain a similar set of labs, each with step-by-step exercises allowing you to learn different aspects of SmartXplorer. Please note that not every SmartXplorer option and functionality is covered in this tutorial.

## Guide Contents

This manual contains the following chapters:

- [Chapter 1, “SmartXplorer Overview”](#), gives a description of SmartXplorer capabilities.
- [Chapter 2, “Tutorial Description”](#), provides general information about the features covered in each lab. We also describe the time required to complete each lab segment.
- [Chapter 3, “Preparing the Design for Labs”](#), contains instructions on how to obtain the designs for each lab.
- Chapters 4, 5, 6, and 7 contain detailed information for each lab. Each includes an overall summary for that lab segment as well as the detailed instructions required for you to complete the lab.
  - ♦ [Chapter 4, “Lab 1: Basic Flow”](#)
  - ♦ [Chapter 5, “Lab 2: Creating Custom Strategies”](#)
  - ♦ [Chapter 6, “Lab 3W: Running Multiple Strategies in Parallel \(Windows\)”](#)
  - ♦ [Chapter 7, “Lab 3L: Running Multiple Strategies in Parallel \(Linux\)”](#)
- [Appendix A, “Custom Files”](#), provides examples for a custom strategy file and host list files. In addition, it contains a set of SmartXplorer configurations dedicated to various tasks.

## Additional Resources

To find additional documentation, see the Xilinx® Web site at:

<http://www.xilinx.com/support/documentation/index.htm>.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx Web site at:

<http://www.xilinx.com/support/mysupport.htm>.

## Conventions

This document uses the following conventions. An example illustrates each convention.

### Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays	<code>speed grade: - 100</code>
<b>Courier bold</b>	Literal commands that you enter in a syntactical statement	<code>ngdbuild design_name</code>
<b>Helvetica bold</b>	Commands that you select from a menu	<b>File → Open</b>
	Keyboard shortcuts	<b>Ctrl+C</b>
Italic font	Variables in a syntax statement for which you must supply values	<code>ngdbuild design_name</code>
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as <code>bus [7:0]</code> , they are required.	<code>ngdbuild [option_name] design_name</code>
Braces { }	A list of items from which you must choose one or more	<code>lowpwr = {on   off}</code>
Vertical bar	Separates items in a list of choices	<code>lowpwr = {on   off}</code>



Convention	Meaning or Use	Example
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	<b>allow block</b> <i>block_name loc1 loc2 ... locn;</i>

## Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section “ <a href="#">Additional Resources</a> ” for details. Refer to “ <a href="#">Title Formats</a> ” in <a href="#">Chapter 1</a> for details.
<a href="#">Blue, underlined text</a>	Hyperlink to a Web site (URL)	Go to <a href="http://www.xilinx.com">http://www.xilinx.com</a> for the latest speed files.



# SmartXplorer Overview

---

## Introduction

Delivering timing closure in the shortest amount of time – is the ultimate SmartXplorer goal.

Timing closure is undoubtedly one of the most challenging aspects in modern FPGA design. Xilinx is committed to assisting designers overcome timing challenges by:

- Improving synthesis and implementation algorithms,
- Providing powerful graphical analysis tools such as PlanAhead™ software and FPGA Editor.

Although FPGA tools have become easier to use while offering ever more advanced features, it is difficult to anticipate all design situations. Some may stay hidden until the very last stages of a design cycle.

Regardless of their experience level, designers usually try to explore several possibilities by changing different tool options before deciding to make a change in their HDL code or trying placement constraints. Changing tool operations is a very easy thing to do. The main question facing the designer is this: How can a better set of options be identified in the first place?

## Key Benefits

SmartXplorer has two key features:

- It automatically performs design exploration by using a set of built-in or user created implementation strategies to try to meet timing.  
**Note:** A design strategy is a set of tool options and their corresponding values that are intended to achieve a particular design goal such as area, speed, or power.
- It allows running these strategies in parallel on multiples machines, completing the job much faster.

## Design Strategies

SmartXplorer is delivered with a set of predefined, built-in strategies. These strategies are tuned and selected separately for each FPGA family. This selection is revised for each major release to ensure that we have the best possible correlation with current software version.

Many designers create their own design strategies or scripts based on their experience. SmartXplorer allows users to integrate these custom strategies into the system and either use them exclusively or combine with some predefined strategies.

SmartXplorer is not simply a tool to use during the late, time-limited portion of the design cycle. It can be used during the entire project cycle, preventing or reducing emergency situations at the end of the design cycle. We suggest running it on a regular basis to monitor your design and ensure timing results stay within an acceptable range.

## Exploiting Parallel compute platforms

Executing several design strategies (jobs) in parallel is a powerful feature which allows designers to complete the project faster. This specific nature of this feature depends on the operating system in use.

### Linux OS

SmartXplorer can run multiple jobs in parallel on different machines across the network. This can be done in two ways:

- You have a regular Linux network: in this case, SmartXplorer manages the jobs distribution across the network. You must provide a list of machines which can be used.
- SmartXplorer supports LSF (Load Sharing Facility) or SGE (Sun Grid Engine) compute farms. In this case, LSF or SGE manages jobs distribution. You must specify the number of machines which can be simultaneously allocated to SmartXplorer.

If you do not have access to the Linux network, but you have a personal Linux machine with multi-core processor or several processors, you can still run several jobs in parallel on this machine.

### Microsoft Windows OS

In the current release, SmartXplorer allows several strategies to be run in parallel on a single Windows machine, if it has a multi-core processor or several processors.

## Using a Single Linux or Windows Machine

If you do not have access to multiple Linux servers on a network and can only use your local computer, make sure your machine has at least one multi-core processor or several processors.

First, you must estimate how many jobs your machine can run simultaneously.

Theoretically, the number of jobs you can run in parallel is calculated in the following way:

$$\text{Nb\_Of\_Jobs} = P * C$$

Where **P** is the number of processors and **C** is the number of cores per processor.

If you have 4 dual-core processors, then you can run 8 jobs in parallel.

However, depending on the available memory, its speed, the speed of your hard drive, etc., your computer might not be able to deal with the maximum number of jobs calculated using the above formula.

Depending on your calculations, following are some tips you can use:

**Tip1:** If due to the memory requirements of your design, your machine can run only a single strategy at a time, then you will need to run all strategies sequentially. This is a good situation for using an overnight run of SmartXplorer.

**Tip2:** When trying to solve timing problems, you can work on smaller blocks separately from the rest of the design. Your machine might be able deal with multiple strategies in parallel for these blocks. If this is the case, parallel jobs can save you a lot of time.



# Tutorial Description

---

Throughout the tutorials, we will use a small design to allow you to complete the labs as quickly as possible. Less than 30 minutes is required to complete the entire tutorial, which covers all of the major features of SmartXplorer.

We strongly suggest running the labs in order (Lab 1, Lab 2, etc.). However, the labs are independent and therefore can be run out of order if you wish to immediately focus on one particular functional area.

SmartXplorer has two key features:

- It helps to achieve timing closure by using the predefined built-in or user-defined design strategies.
- It allows running these strategies in parallel on multiple machines, completing the job much faster.

For the sake of clarity these two key features are represented separately within this tutorial:

- Lab 1 and Lab 2 are dedicated to the aspects of timing closure. All design strategies in these labs will be run sequentially (the feature that allows running several strategies in parallel will be *intentionally disabled*).
- In contrast, Lab 3 is dedicated to running multiple strategies in parallel.

Since the job distribution across multiple machines is supported on the Linux OS only, Lab 3 is divided into two parts:

- ◆ Lab 3W is dedicated to the Windows users having either a multi-core processor or a multi-processor machine.
- ◆ Lab 3L covers Linux users having access to a Linux network, LSF and SGE compute farms, or running either a multi-core processor or a multi-processor machine.

The following table provides a brief overview of all the labs:

**Table 1: Lab Overview**

Lab No. and Title	Duration	Covered Features
<a href="#">Lab 1: Basic Flow</a>	15 min	<ul style="list-style-type: none"> <li>• How SmartXplorer is integrated into Project Navigator and can be easily launched</li> <li>• How final results are reported, stored, and can be further used</li> <li>• How to run additional iterations to improve upon previously obtained</li> <li>• How to configure SmartXplorer to run the predefined strategies and additional iterations at once (for example, overnight runs)</li> </ul>
<a href="#">Lab 2: Creating Custom Strategies</a>	5 min	<ul style="list-style-type: none"> <li>• How to create a custom strategy file and use it in SmartXplorer</li> </ul>
<a href="#">Lab 3W: Running Multiple Strategies in Parallel (Windows)</a>	5 min	<ul style="list-style-type: none"> <li>• How to run several strategies in parallel</li> </ul>
<a href="#">Lab 3L: Running Multiple Strategies in Parallel (Linux)</a>	10 min	<ul style="list-style-type: none"> <li>• How to set up the environment and run multiple strategies in parallel</li> <li>• A regular Linux network is presented, as well as LSF and SGE compute farms</li> </ul>

## Knowledge Prerequisites

The labs require some basic knowledge of the ISE Project Navigator environment. Before starting these labs, you should know:

- The major steps of the Xilinx FPGA implementation flow and how to run them: Synthesis, Translate (NGDBuild), Map, Place & Route, and Timing Analysis (TRCE).
- How to open and close an existing project.



# Preparing the Design for Labs

---

This section provides detailed instructions on how to prepare the design for each lab.

## Design Description

You will use the small *stopwatch* design throughout this tutorial for each lab, targeting a Spartan®-3E xc3s100e-4-vq100 device.

## Instructions (Linux & Windows)

1. Create an `sx_labs` directory, where you will store all SmartXplorer lab data.
2. Copy the `sx_tutorial_examples.zip` file to the `sx_labs` directory.
3. Unzip the `sx_tutorial_examples.zip` file. Ultimately, you should have a directory structure similar to the following:



Each lab directory contains an ISE project, named `watchvhd.xise`.

**Note:** New for ISE 11.1, the ISE project is an XML file with a “.xise” extension.



# Lab 1: Basic Flow

---

## Objectives

The goal of this 15-minute lab is to cover:

- How SmartXplorer is integrated in the Project Navigator environment
- How to run each of the predefined SmartXplorer Strategies
- How final results are reported, stored and can be further used
- How to run additional iterations based on the previously obtained results in order to improve timing

## Lab

### Step 1: Open Lab project

Open the **stopwatch.xise** project located in the **lab1** directory.

### Step 2: Run SmartXplorer with Default Options

For this step, we will run SmartXplorer with all default options. We will highlight the main steps of the flow and comment on the results.

1. Select **Tools > SmartXplorer > Launch SmartXplorer**, or click the SmartXplorer toolbar button  .

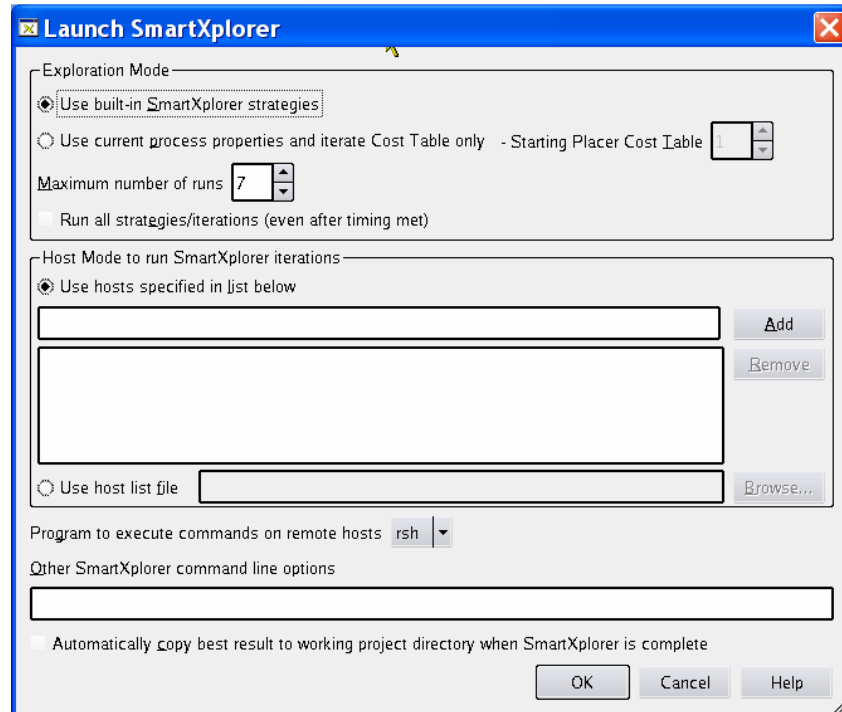


Figure 4-1: SmartXplorer Dialog Box (Linux)

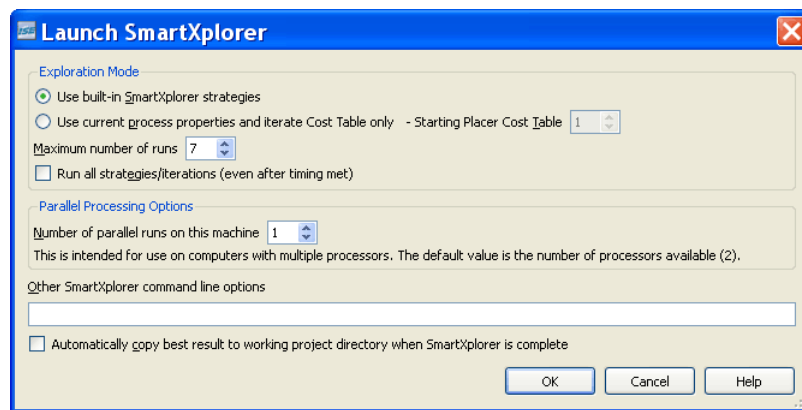
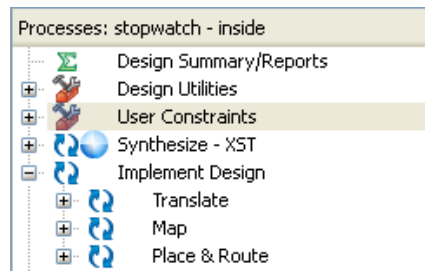


Figure 4-2: SmartXplorer Dialog Box (Windows)

2. Launch SmartXplorer with the default options selected by clicking the OK button.

The SmartXplorer flow consists of multiple Map, Place & Route and TRCE runs with different options. At the beginning of each run, Project Navigator automatically runs the Synthesis and Translate processes if necessary.



Once running, SmartXplorer creates a status table where it will display work progress and final results summary. Each row in this table represents one of the 7 predefined built-in SmartXplorer strategies; in this case, for the Spartan-3E family.

This table is progressively updated during SmartXplorer run. Here is an example of an intermediate status:

Strategy	Host	Status	Timing Score	Run time	Copy Result
ParHighEffort2	None	None	None	None	Copy Result
ParHighEffort1	None	None	None	None	Copy Result
MapUseIOReg	None	None	None	None	Copy Result
MapTimingExtraEffort	tripoli	Done	4413	00h 00m 49s	Copy Result
MapTiming2	None	None	None	None	Copy Result
MapTiming1	None	None	None	None	Copy Result
MapPhysSynthesis	tripoli	Mapping	None	00h 00m 04s	Copy Result

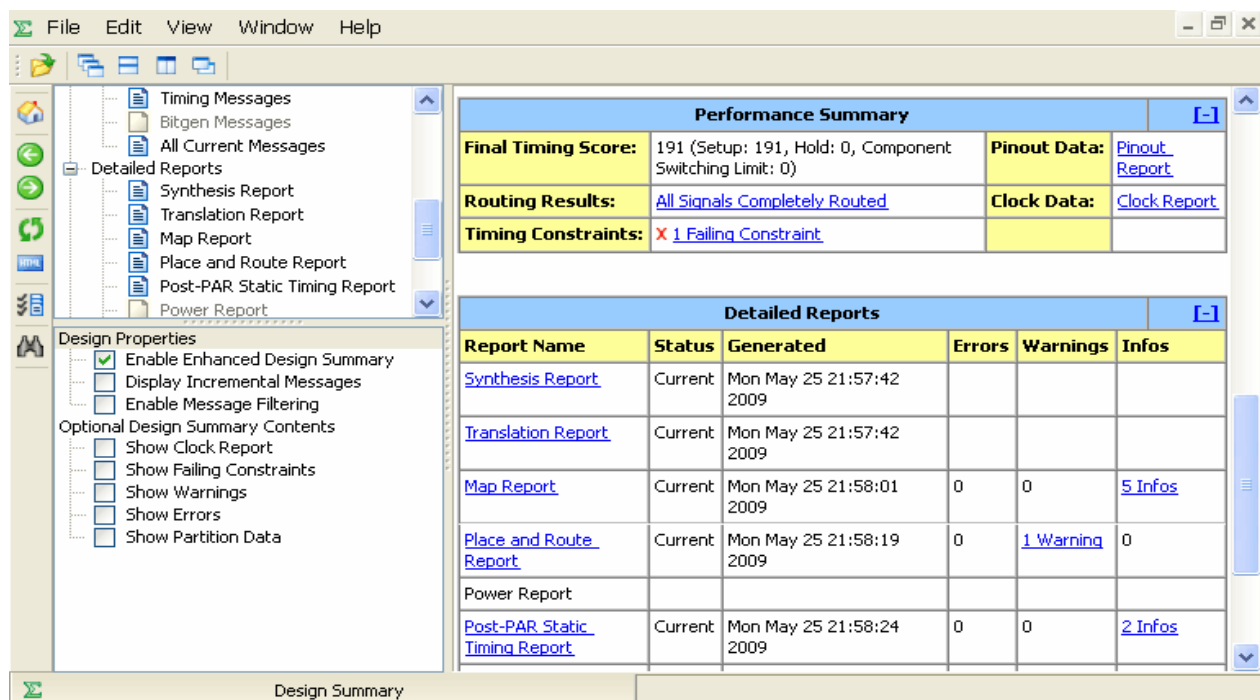
As you can see from the **Done** status in the Status column, the MapTimingExtraEffort strategy has been completed. The process took **49 seconds** and the final timing score is **4413** (timing constrains were not met). This row has a green background meaning that, so far, this strategy provides the best timing results in the current SmartXplorer session.

The MapPhysSynthesis strategy is still running and it is going through the Mapping step: **Mapping**.

Eventually, SmartXplorer completes all strategies and we have the following status table. From a timing perspective, the MapPhysSynthesis is the best strategy as it has the smallest timing score.

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">ParHighEffort2</a>	tripoli	<a href="#">Done</a>	3086	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">ParHighEffort1</a>	tripoli	<a href="#">Done</a>	2508	00h 00m 55s	<a href="#">Copy Result</a>
<a href="#">MapUseIOReq</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 05s	<a href="#">Copy Result</a>
<a href="#">MapTimingExtraEffort</a>	tripoli	<a href="#">Done</a>	4413	00h 00m 49s	<a href="#">Copy Result</a>
<a href="#">MapTiming2</a>	tripoli	<a href="#">Done</a>	2512	00h 00m 50s	<a href="#">Copy Result</a>
<a href="#">MapTiming1</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesis</a>	tripoli	<a href="#">Done</a>	191	00h 00m 49s	<a href="#">Copy Result</a>

- Click on the **Done** link in the Status column of the MapPhysSynthesis strategy to open a design summary and then access all reports generated for MapPhysSynthesis.



**Performance Summary**

<b>Final Timing Score:</b>	191 (Setup: 191, Hold: 0, Component Switching Limit: 0)	<b>Pinout Data:</b>	<a href="#">Pinout Report</a>
<b>Routing Results:</b>	<a href="#">All Signals Completely Routed</a>	<b>Clock Data:</b>	<a href="#">Clock Report</a>
<b>Timing Constraints:</b>	<a href="#">X 1 Failing Constraint</a>		

**Detailed Reports**

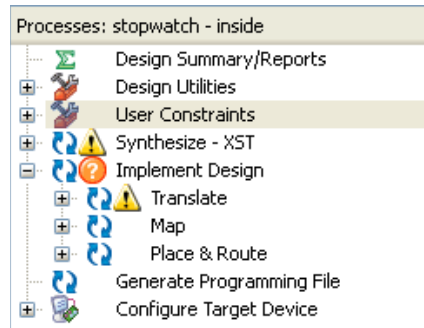
Report Name	Status	Generated	Errors	Warnings	Infos
<a href="#">Synthesis Report</a>	Current	Mon May 25 21:57:42 2009			
<a href="#">Translation Report</a>	Current	Mon May 25 21:57:42 2009			
<a href="#">Map Report</a>	Current	Mon May 25 21:58:01 2009	0	0	<a href="#">5 Infos</a>
<a href="#">Place and Route Report</a>	Current	Mon May 25 21:58:19 2009	0	<a href="#">1 Warning</a>	0
Power Report					
<a href="#">Post-PAR Static Timing Report</a>	Current	Mon May 25 21:58:24 2009	0	0	<a href="#">2 Infos</a>

Design Summary

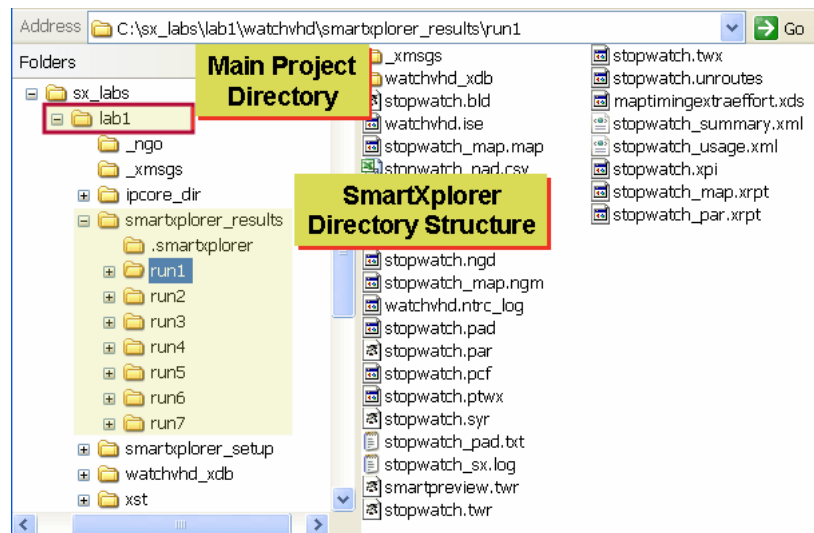
## Step 3: Using SmartXplorer Results

In this step, we will show how generated SmartXplorer Results can be further used.

In the Processes window, the **Map** and **Place & Route** processes do not have status icons, which indicates that they were not run. This is because the SmartXplorer flow is as a parallel flow that coexists with the regular (non-SmartXplorer) flow in the ISE Project Navigator environment.

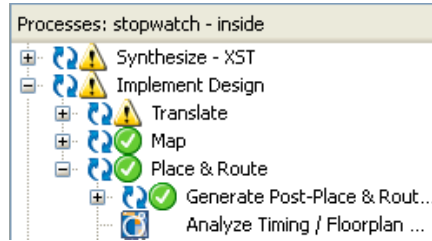


All SmartXplorer results are stored in the **smartexplorer\_results** directory, independently from the regular flow. Each SmartXplorer strategy is stored in a separate directory: **run1**, **run2**, ..., **run7**:

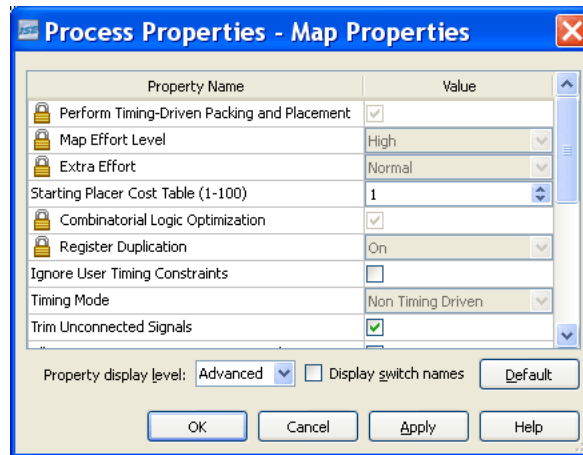


SmartXplorer results can be imported (copied) into the regular Project Navigator flow for further use. You can do this to generate a programming file or to run detailed timing analysis, simulation or power estimation. To copy results, do the following:

- To copy the MapPhysSynthesis strategy results to the regular Project Navigator flow, click **Copy Result**. Project Navigator does the following:
  - Copies all Map, PAR, and TRCE files to the **watchvhd** project directory. This will automatically update the Design Summary for the regular flow as well.
  - Updates Map, PAR, and TRCE process statuses.



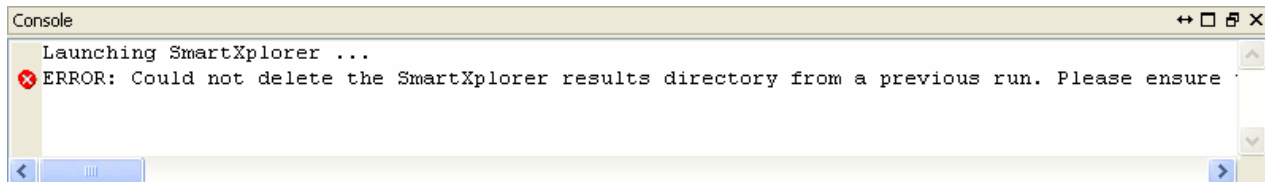
- ◆ Overwrites your current Map and PAR options with the ones from the MapPhysSynthesis strategy. In addition, Project Navigator locks all options explicitly specified in the strategy (indicated by a locked padlock icon).



**Note:** If you want to modify one of the locked options, you must unlock them first by selecting the **Design Goals & Strategies...** option.

2. Continue your design using the standard Project Navigator flow, while taking advantage of SmartXplorer results.

**IMPORTANT:** If you want to restart SmartXplorer, you should first close all design summary windows or reports in text editors opened for SmartXplorer results. The main reason is that SmartXplorer removes all previously generated data files before each start. If one of the reports is not closed, SmartXplorer might fail to clean up previously generated results. In this case, it will issue a **permission denied** error in the Console window and stop execution.





## Step 4: Running Additional Iterations to Improve Timing

In the previous step, the MapPhysSynthesis strategy gave the best timing result. We will run it 5 additional times with different Cost Tables to further improve timing (please refer to the *Command Line Tools User Guide* for more information on cost tables).

**Note:** The MapPhysSynthesis results must first be imported to the regular Project Navigator environment (this has already been completed at “[Step 3: Using SmartXplorer Results](#)”).

1. In the SmartXplorer dialog box, select **Use the current process properties and iterate Cost Table only**.
2. Taking into account that originally the MapPhysSynthesis strategy was run with default Cost Table of 1, we now choose to set starting placer cost table to 2. This means that SmartXplorer will use 2, 3, etc., cost tables for additional iterations.
3. Set maximum number of runs to 5.

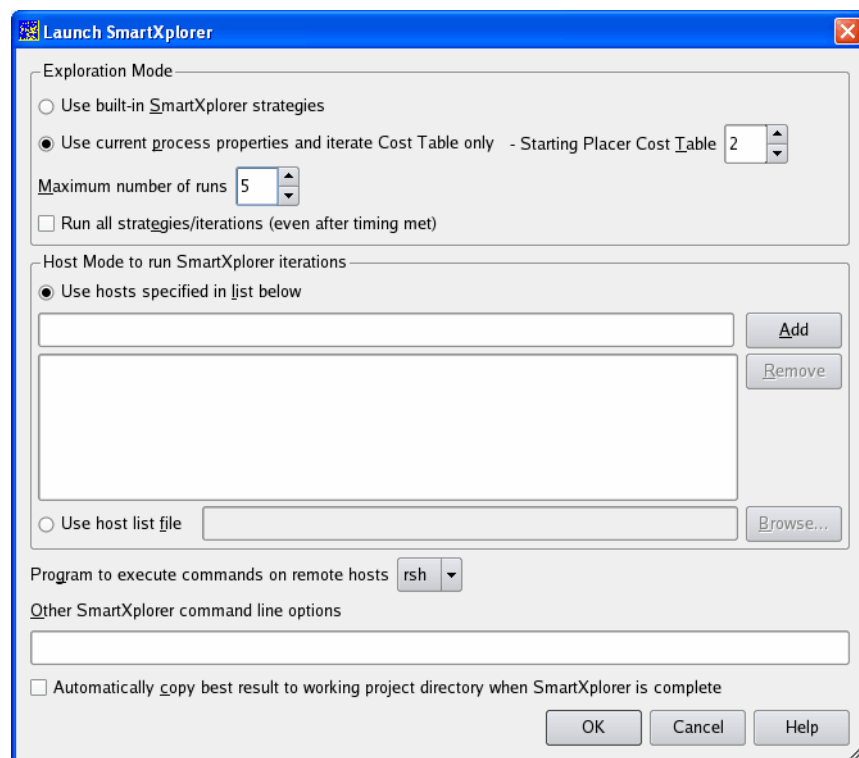


Figure 4-3: Cost Table and Number of Runs Settings (Linux)

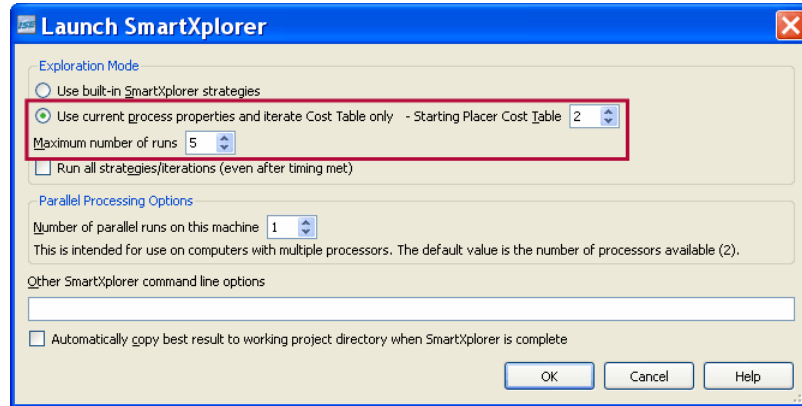


Figure 4-4: Cost Table and Number of Runs Settings (Windows)

4. Click OK.

The SmartXplorer results show that additional SmartXplorer runs were able to meet timing.

Strategy	Host	Status	Timing Score	Run time	Copy Result
CurrentProjectNavigatorSettingsCT5	tripoli	Done	0	00h 00m 49s	Copy Result
CurrentProjectNavigatorSettingsCT4	tripoli	Done	367	00h 00m 54s	Copy Result
CurrentProjectNavigatorSettingsCT3	tripoli	Done	198	00h 00m 54s	Copy Result
CurrentProjectNavigatorSettings	tripoli	Done	330	00h 00m 59s	Copy Result

As you can see, only four strategies were run. The main reason is that by default SmartXplorer stops strategies execution as soon as timing constraints are met. This is the case for iteration with Cost Table 5: CurrentProjectNavigatorSettingsCT5.

You can check the **Run all strategies/iterations (even after timing met)** option to run all iterations and get the complete picture on the relative performance of each cost table.

**Note:** The methodology described in this step can be used in a different way. For example, if you used the regular Project Navigator Flow and were able to obtain a very small timing score, you may use SmartXplorer to run your design with current project settings and just apply different Cost Tables.

## Step 5: Run the Predefined Strategies and Additional Iterations at Once

In the previous steps, we were able to meet timing constraints using a two-phase approach. In the first phase, “[Step 3: Using SmartXplorer Results,](#)” we ran all predefined strategies and identified the best one. In the second phase, “[Step 4: Running Additional Iterations to Improve Timing,](#)” we selected the best strategy from the first phase and iterated on it by changing cost tables.

The same procedure can be achieved in a single phase. This is a very useful approach to running jobs overnight. In the Launch XmartXplorer dialog box, do the following:

1. Select **Use built-in SmartXplorer strategies.**
2. Specify **Maximum number of runs equal to 12** (7 predefined + 5 additional iterations).
3. If you are planning to run SmartXplorer overnight, you may want to run all strategies to get a complete picture. Select **Run all strategies/iterations (even after timing met).**
4. Click **OK.**

The SmartXplorer Results window shows the following final results:

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">ParHighEffort2</a>	tripoli	<a href="#">Done</a>	3086	00h 00m 45s	<a href="#">Copy Result</a>
<a href="#">ParHighEffort1</a>	tripoli	<a href="#">Done</a>	2508	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapUseIOReq</a>	tripoli	<a href="#">Done</a>	4652	00h 00m 45s	<a href="#">Copy Result</a>
<a href="#">MapTimingExtraEffort</a>	tripoli	<a href="#">Done</a>	4413	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapTiming2</a>	tripoli	<a href="#">Done</a>	2512	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapTiming1</a>	tripoli	<a href="#">Done</a>	4652	00h 00m 49s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT6</a>	tripoli	<a href="#">Done</a>	15	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT5</a>	tripoli	<a href="#">Done</a>	0	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT4</a>	tripoli	<a href="#">Done</a>	367	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT3</a>	tripoli	<a href="#">Done</a>	198	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT2</a>	tripoli	<a href="#">Done</a>	330	00h 00m 44s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesis</a>	tripoli	<a href="#">Done</a>	191	00h 00m 49s	<a href="#">Copy Result</a>

## Conclusion

In this lab, we ran SmartXplorer from the Project Navigator environment and obtained results for the 7 predefined strategies (Phase 1, [Step 3: Using SmartXplorer Results](#)).

Then, we showed how one of the generated SmartXplorer results can be imported into the regular Project Navigator flow in order to be further used in the standard flow.

We then ran the best strategy 5 additional times with different Cost Tables to further improve timing (Phase 2, [Step 4: Running Additional Iterations to Improve Timing](#)).

Finally, we showed how to run the predefined strategies and additional runs in a single pass.



## Lab 2: Creating Custom Strategies

---

### Objectives

Working on your design and running the 7 predefined strategies, you found out that two of them always gave you much better results than the other 5. Therefore, for future runs, you ultimately choose to run only those two strategies and skip the others.

Alternatively, you may have manually found a set of strategies which give you the best results for your design and now you would like to use these strategies with SmartXplorer.

In both cases, you can create a custom strategy file and use it in SmartXplorer.

The goal of this 5-minute lab is to show how custom strategies can be created and run from SmartXplorer.

### Lab

#### Step 1: Open Lab Project

Open the `stopwatch.xise` project, located in the `lab2` directory.

#### Step 2: Identify the Best Two Strategies

In Lab 1, we obtained the following results for 7 predefined strategies, where `MapPhysSynthesis` and `MapTiming2` were the best ones:

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">ParHighEffort2</a>	tripoli	<a href="#">Done</a>	3086	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">ParHighEffort1</a>	tripoli	<a href="#">Done</a>	2508	00h 00m 55s	<a href="#">Copy Result</a>
<a href="#">MapUseIOReq</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 05s	<a href="#">Copy Result</a>
<a href="#">MapTimingExtraEffort</a>	tripoli	<a href="#">Done</a>	4413	00h 00m 49s	<a href="#">Copy Result</a>
<a href="#">MapTiming2</a>	tripoli	<a href="#">Done</a>	2512	00h 00m 50s	<a href="#">Copy Result</a>
<a href="#">MapTiming1</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesis</a>	tripoli	<a href="#">Done</a>	191	00h 00m 49s	<a href="#">Copy Result</a>

## Step 3: Creating the Custom Strategy File

Before we can create a custom strategy file based on MapPhysSynthesis and MapTiming2, we must find out the set of options SmartXplorer used for these two strategies.


This information was stored in the `smartexplorer.txt` file located in the `smartexplorer_results` directory of lab1. This file contains the options for all previously run strategies. The contents of this file can be found in the `lab1_smartexplorer.txt` file located in the lab2 directory.

```
-----
Strategy : MapPhysSynthesis
-----
Run index   : run2
Map options : -timing -ol high -xe n -register_duplication on -logic_opt on
Par options : -ol high
...
-----
Strategy : MapTiming2
-----
Run index   : run6
Map options : -timing -ol high -t 9
Par options : -ol high -t 9
...
-----
```

Using the above information, we can create a strategy file. In addition, we will rename the MapPhysSynthesis and MapTiming2 strategies, using the names `My_Strat_1` and `My_Strat_2`, respectively.

1. Open `my_strategy.txt` file located in lab2 directory. This file contains `My_Strat_1` and `My_Strat_2` definitions:

```
{
  "spartan3e":
  (
    {"name": "My_Strat_1",
     "map": " -timing -ol high -xe n -register_duplication on -logic_opt on ",
     "par": " -ol high"},
    {"name": "My_Strat_2",
     "map": " -timing -ol high -t 9",
     "par": " -ol high -t 9"},
  ),
}
```

2. Select **Tools > SmartXplorer > Launch SmartXplorer**, or click the SmartXplorer toolbar button  .
3. Specify the `my_strategy.txt` file via the `-sf` switch in the “Other SmartXplorer command line options” field.
4. To ensure that only two custom strategies will be run, you must set the **Maximum number of runs** value to **2**.

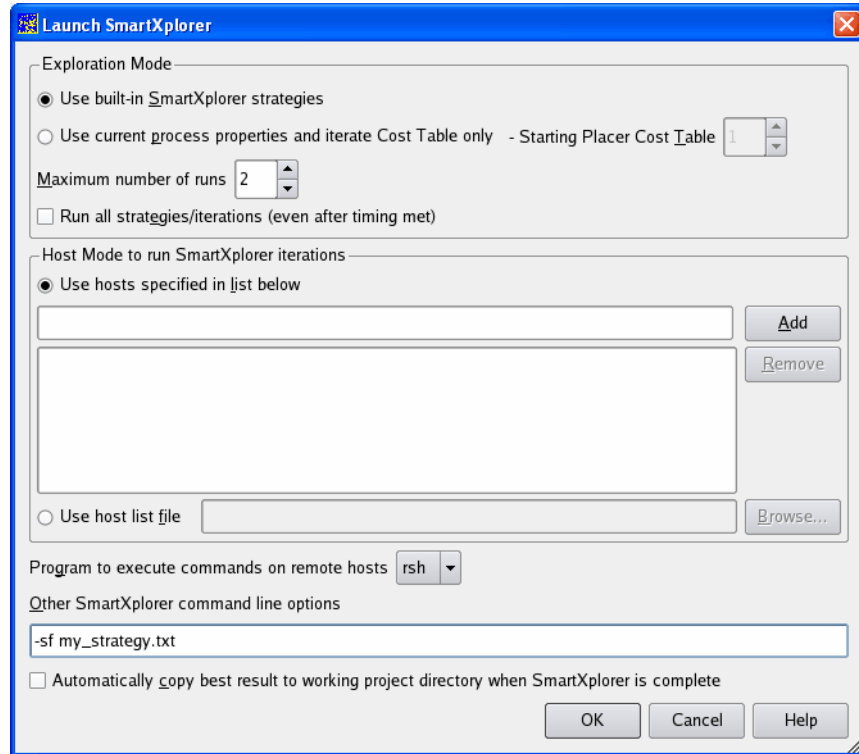


Figure 5-1: SmartXplorer Number of Runs and Command Line Settings (Linux)

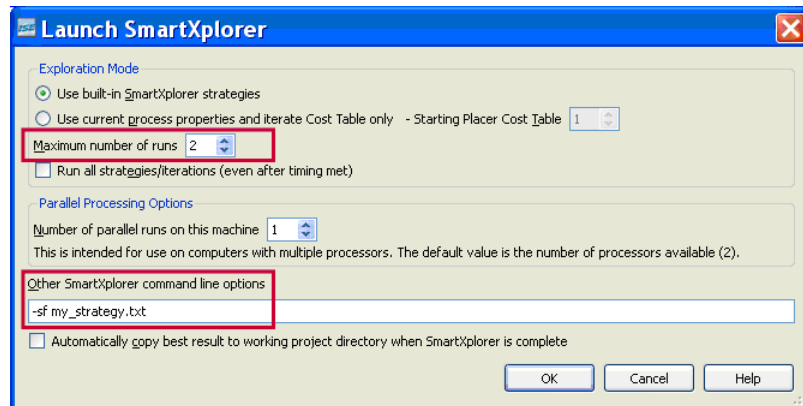


Figure 5-2: SmartXplorer Number of Runs and Command Line Settings (Windows)

Please note that if, for example, you specify 5 as the “Maximum number of runs,” instead of 2, then SmartXplorer will select the best strategy of the two in the strategy file and run three additional iterations using cost tables (i.e., using the same mechanism as for the predefined strategies). This is a valid scenario.

5. Click **OK**.

SmartXplorer generates the following results, which match the results from the preceding lab:

Strategy	Host	Status	Timing Score	Run time	Copy Result
My_Strat_2	tripoli	<a href="#">Done</a>	2512	00h 00m 50s	<a href="#">Copy Result</a>
My_Strat_1	tripoli	<a href="#">Done</a>	191	00h 00m 55s	<a href="#">Copy Result</a>

**Note:** Currently, the **Copy Result** operation is not properly supported for custom strategies. The software will copy files to the project directory, however, Map and PAR options will not be updated in the Map and PAR Process properties. You have to manually setup the corresponding options.

## Conclusion

In this lab, we created a custom strategy file and used it to run a design.

Then, we pointed out the current differences in custom strategies support as compared to using predefined strategies.



## *Lab 3W: Running Multiple Strategies in Parallel (Windows)*

---

**IMPORTANT:** This lab is targeted to users having either a multi-core processor or a multi-processor Windows machine (not a single core processor machine).

### Objectives

One of the main SmartXplorer features is the possibility to run several strategies in parallel.

The goal of this 5-minute lab is to show how you can specify the number of strategies SmartXplorer will run simultaneously on your Windows machine and determine the Global runtime gain you can expect from that.

### Lab

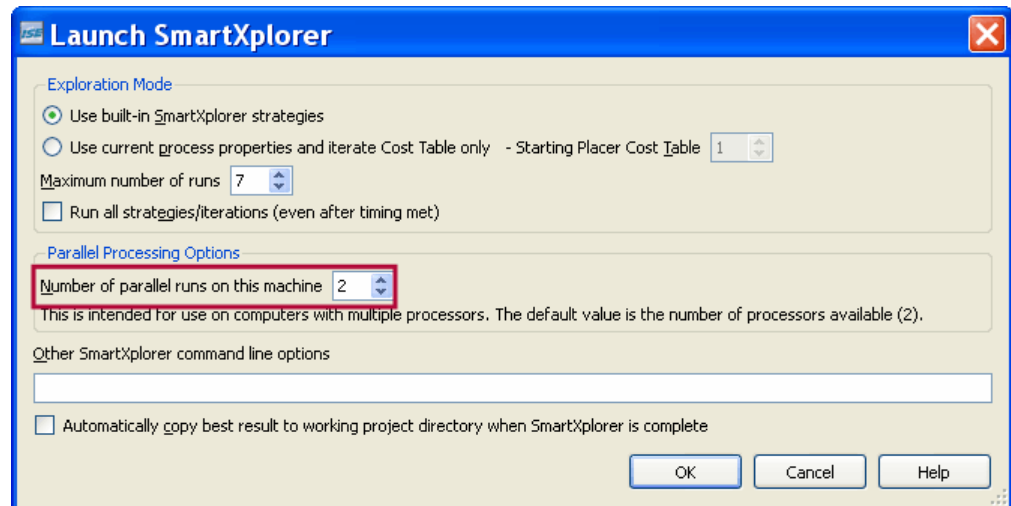
#### Step 1: Open Lab Project

Open the **stopwatch.xise** project located in the **lab3** directory.

## Step 2: Run SmartXplorer Enabling Two Parallel Runs

At this step, we will run the 7 predefined SmartXplorer strategies enabling two parallel runs.

1. Open the SmartXplorer dialog box.
2. Set the “Number of parallel runs on this machine” to 2.



3. **Launch SmartXplorer.** SmartXplorer generates the following results:

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">ParHighEffort2</a>	tripoli	<a href="#">Done</a>	3086	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">ParHighEffort1</a>	tripoli	<a href="#">Done</a>	2508	00h 00m 55s	<a href="#">Copy Result</a>
<a href="#">MapUseIOReq</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 05s	<a href="#">Copy Result</a>
<a href="#">MapTimingExtraEffort</a>	tripoli	<a href="#">Done</a>	4413	00h 00m 49s	<a href="#">Copy Result</a>
<a href="#">MapTiming2</a>	tripoli	<a href="#">Done</a>	2512	00h 00m 50s	<a href="#">Copy Result</a>
<a href="#">MapTiming1</a>	tripoli	<a href="#">Done</a>	4652	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesis</a>	tripoli	<a href="#">Done</a>	191	00h 00m 49s	<a href="#">Copy Result</a>

4. Comparing global runtime when all seven strategies were executed sequentially versus the case with two parallel runs enabled, we observed that all seven predefined strategies were completed **30% faster** when the two parallel runs were enabled.

These experiments were run on a machine with the following configuration:

Intel(R)  
Xeon(TM) CPU 3.20GHz  
3.20 GHz, 3.00 GB of RAM

**Note:** SmartXplorer may stop all currently running strategies if another strategy is completed and met timing. This situation appears when multiple strategies are run in parallel and the **Run all strategies/iterations** option is not checked.

<a href="#">MapTiming1</a>	xgr-storojev-l2	<a href="#">Done</a>	4652	00h 01m 00s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT6</a>	xgr-storojev-l2	<a href="#">Stopped</a>	None	00h 00m 33s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT5</a>	xgr-storojev-l2	<a href="#">Done</a>	0	00h 00m 45s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT4</a>	xgr-storojev-l2	<a href="#">Done</a>	367	00h 01m 00s	<a href="#">Copy Result</a>

Let’s review the status of the two strategies that are surrounded by the red rectangle. Chronologically, MapPhysSynthesisCT6 was started after MapPhysSynthesisCT5 and then MapPhysSynthesisCT5 was subsequently able to meet timing before MapPhysSynthesisCT6 was completed. Therefore, SmartXplorer stopped MapPhysSynthesisCT6 execution and you see the Stopped status in the Status column.

## Conclusion

In this lab, we described how multiple strategies can be run simultaneously on a single Microsoft Windows machine.

We have seen how we can obtain 30% of global runtime reduction just by enabling two parallel runs in SmartXplorer.

**Note:** What you actually gain will depend on your machine configuration.



# Lab 3L: Running Multiple Strategies in Parallel (Linux)

---

**IMPORTANT:** This lab is dedicated to users having access to a regular Linux network, LSF and SGE compute farms, or either a multi-core processor or a multi-processor machine (not a single core processor machine).

## Objectives

The goal of this 10-minute lab is to show how to setup the Xilinx environment and use SmartXplorer on a regular Linux network, LSF and SGE compute farms, or on a single, multi-core processor or a multi-processor machine.

We will list a set of key items you must take into account before using SmartXplorer across such networks.

## Lab

### Step 1: Key Items – Xilinx Environment and Results Storage

There are two important items we must clarify before launching SmartXplorer:

- How to setup the Xilinx software environment on each machine.
  - Where the SmartXplorer results will be stored.
1. Setting up the Xilinx environment

First, let's consider the case of a regular Linux network, where three Linux machines (L1, L2, and L3) will be used to run design strategies in parallel. In addition, L1 will be used to launch ISE Project Navigator and therefore, SmartXplorer.

Before launching a job on L2 (L3), SmartXplorer will automatically setup \$XILINX environment variable on the L2 (L3) machine. For that, it will use the value of \$XILINX from L1. This means that, if Xilinx software is installed on:

- ◆ The network, then L2 (L3) must have access to this installation as well using the same network mount points so that the network paths defined for L1 are valid for all machines.
- ◆ The L1 local disk, then L2 (L3) must have the same version of Xilinx software installed on a local disk and placed in the directory with the same path name as on L1.

This mechanism of using environment variables was created to ensure that each design strategy will be run under the same conditions. In addition to \$XILINX,

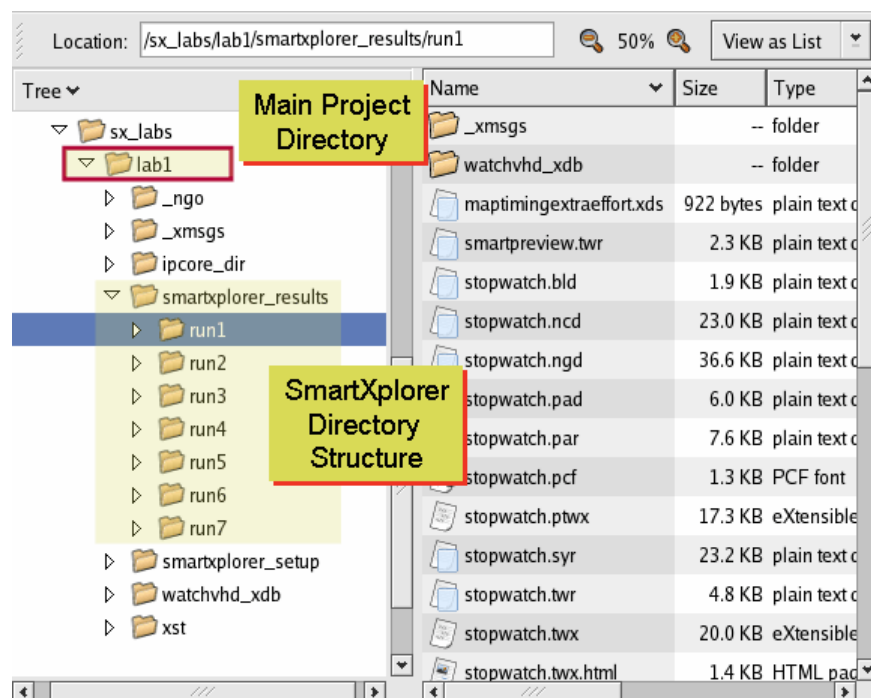
SmartXplorer will automatically collect all Xilinx environment variables (having the 'XIL\_' prefix) set on L1 and propagate them to L2 and L3.

The same rules must be used for LSF and SGE compute farms; each machine eligible to run Xilinx software must be able to use the same Xilinx environment as set on the machine running ISE Project Navigator.

## 2. Results Storage

As we have seen in Lab 1, all SmartXplorer results are stored in a separate directory, named **smartexplorer\_results**.

The same method is used in the case of multiple machines. By default all SmartXplorer results are stored on a machine running ISE Project Navigator. Therefore, all machines must have access to this disk area and have read and write permissions. You can use the `-wd` option to store results in a different location.



## Step 2: Pre-run Checklist

1. Obtain the list of machines you will use.
2. Ensure that the same Xilinx environment can be setup on each of them.
3. Verify that all machines have access and read and write permissions to the SmartXplorer results directory.

## Step 3: Open the Lab Project

Open the **stopwatch.xise** project located in the **lab3** directory.

## Step 4: Run SmartXplorer on a Regular Linux Network

In this step, we will run the 7 predefined SmartXplorer strategies on two Linux machines: xgr-sweng109 and xgr-sweng135. Suppose that xgr-sweng109 can run a single job and xgr-sweng135 two jobs in parallel. We will use xgr-sweng135 to run ISE Project Navigator.

1. Open **my\_hostlist.txt** file located in **lab3** directory using any text editor. This file contains the list of machines which will be used to run design strategies. The format of this file is straightforward; each machine name must be placed in a separate line:

```
xgr-sweng135
xgr-sweng135
xgr-sweng109
```

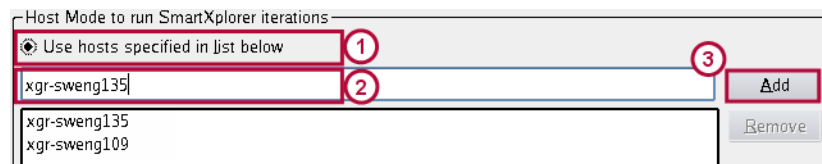
To enable running two jobs in parallel on the same machine, you must list this machine twice, as it is done for xgr-sweng135.

**Note:** Replace xgr-sweng135 and xgr-sweng109 with your machine names in the **my\_hostlist.txt** file and save it before continuing.

2. Select **Tools > SmartXplorer > Launch SmartXplorer**, or click the SmartXplorer toolbar button  .

3. Specify the **my\_hostlist.txt** file in the “Use host list file” field.

**Note:** You can also define the list of your machines directly in the SmartXplorer dialog box.



4. Click **OK**. You might get the following status table, where more powerful xgr-sweng135 was able to complete 5 out of 7 strategies.

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">ParHighEffort2</a>	xgr-sweng135	Done	3086	00h 00m 45s	Copy Result
<a href="#">ParHighEffort1</a>	xgr-sweng109	Done	2508	00h 02m 44s	Copy Result
<a href="#">MapUseIOReg</a>	xgr-sweng135	Done	4652	00h 00m 51s	Copy Result
<a href="#">MapTimingExtraEffort</a>	xgr-sweng135	Done	4413	00h 01m 29s	Copy Result
<a href="#">MapTiming2</a>	xgr-sweng135	Done	2512	00h 00m 45s	Copy Result
<a href="#">MapTiming1</a>	xgr-sweng135	Done	4652	00h 00m 56s	Copy Result
<a href="#">MapPhysSynthesis</a>	xgr-sweng135	Done	191	00h 01m 35s	Copy Result

**Note:** SmartXplorer might stop all currently running strategies if another strategy is completed and met timing. This situation appears when multiple strategies are run in parallel and the **Run all strategies/iterations** option is not checked.

Strategy	Host	Status	Timing Score	Run time	Copy Result
<a href="#">MapTiming2</a>	xgr-sweng135	<a href="#">Done</a>	2512	00h 00m 46s	<a href="#">Copy Result</a>
<a href="#">MapTiming1</a>	xgr-sweng135	<a href="#">Done</a>	4652	00h 01m 01s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT6</a>	xgr-sweng135	<a href="#">Stopped</a>	2764	00h 00m 53s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT5</a>	xgr-sweng135	<a href="#">Done</a>	0	00h 00m 50s	<a href="#">Copy Result</a>
<a href="#">MapPhysSynthesisCT4</a>	xgr-sweng135	<a href="#">Done</a>	367	00h 00m 55s	<a href="#">Copy Result</a>

Let's review the status of the two strategies surrounded by a red rectangle. Chronologically, MapPhysSynthesisCT6 was started after MapPhysSynthesisCT5 and MapPhysSynthesisCT5 was able to meet timing before MapPhysSynthesisCT6 was completed. Therefore, SmartXplorer stopped MapPhysSynthesisCT6 execution, as you can see by the **Stopped** status in the Status column.

## Step 5: Run SmartXplorer on LSF or SGE

Dealing with LSF and SGE compute farms is not much different than working with a regular Linux network shown above. You have to create a host list file and specify it in the in the "Use host list file" field.

The main difference consists in the definition of LSF and SGE in the host list file. The definition format for both compute farms is shown in the following table.

LSF	:LSF {"queue_name": "MYQUEUE", "max_concurrent_runs":N, "bsub_options": "additional_options"}
SGE	:SGE {"queue_name": "MYQUEUE", "max_concurrent_runs":N, "qsub_options": "additional_options"}

Where:

- **queue\_name** defines the queue name. You must replace MYQUEUE by a LSF or SGE queue name.
- **max\_concurrent\_runs** defines the maximum number of jobs which can be run in parallel. You must replace N with a positive integer value.
- **bsub\_options** allows to define additional LSF options and `additional_options` must be replaced by the LSF options. If no options are used, then replace `additional_options` with an empty string: "".
- **qsub\_options** allows to define additional SGE options and `additional_options` must be replaced with the SGE options. If no options are used, then replace `additional_options` with an empty string: "".

Example:

If the queue name is `lin64_q`, the maximum number of parallel jobs is 6, and there are no specific LSF and SGE options. The host list files should contain the following information:

LSF	:LSF {"queue_name": "lin64_q", "max_concurrent_runs":6, "bsub_options": ""}
SGE	:SGE {"queue_name": "lin64_q", "max_concurrent_runs":6, "qsub_options": ""}



## Conclusion

In this lab we showed how you can use SmartXplorer to run several design strategies in parallel on multiple Linux Machines.

We listed a set of key items we have to take into account before using SmartXplorer across the network.



# Custom Files

---

## Objectives

This appendix provides examples for:

- [Custom Strategy File](#)
- [Host List Files \(Linux\)](#)
- [SmartXplorer Configurations for Various Tasks](#)

## Custom Strategy File

Following is an example of a custom strategy file. It contains two strategies for Spartan®-3E devices and two strategies for Virtex®-5 devices.

```
{ # This is a comment
"spartan3e":
(
{"name": "My_Strat_1",
"map": " -timing -ol high -xe n -register_duplication on -logic_opt on ",
"par": " -ol high"},
{"name": "My_Strat_2",
"map": " -timing -ol high -t 9",
"par": " -ol high -t 9"},
),
"virtex5":
(
{"name": "My_Strat_3",
"map": " -ol high -xe n -pr b -t 7 -w ",
"par": " -ol high -xe n -t 7 "},
{"name": "My_Strat_4",
"map": " -ol high -xe n -t 3 -w",
"par": " -ol high -t 3"},
),
}
```

## Host List Files (Linux)

### Regular Linux Network

This example shows a host list file for a regular Linux network.

```
lin_machine_1  
lin_machine_1  
lin_machine_2  
lin_machine_3
```

According to this example SmartXplorer will run four strategies simultaneously. They will be run on three different Linux machines and lin\_machine\_1 will run two strategies in parallel.

### LSF Compute Farm

This example shows a host list file for LSF compute farm. In this example the name of the queue is lin64\_q and the maximum number of parallel jobs is six.

```
:LSF {"queue_name":"lin64_q", "max_concurrent_runs":6, "bsub_options": ""}
```

### SGE Compute Farm

Here is an example of a host list file for SGE Compute Farm. In this example the name of the queue is lin64\_q and the maximum number of parallel jobs is 6.

```
:SGE {"queue_name":"lin64_q", "max_concurrent_runs":6, "qsub_options": ""}
```

# SmartXplorer Configurations for Various Tasks

## Task 1: Run All Built-in Predefined Strategies

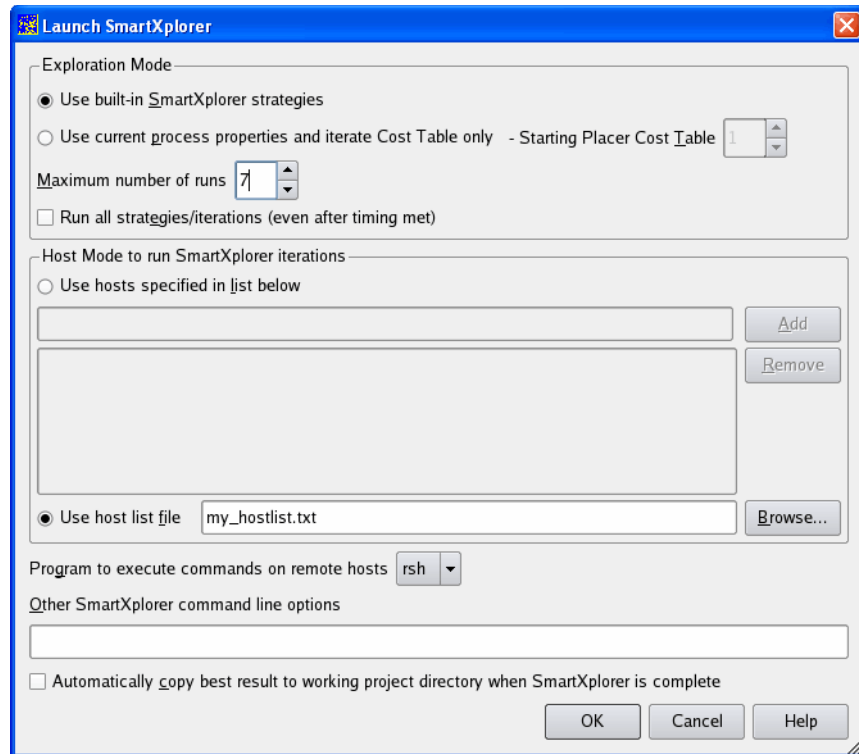


Figure A-1: All Built-in Predefined Strategies (Linux)

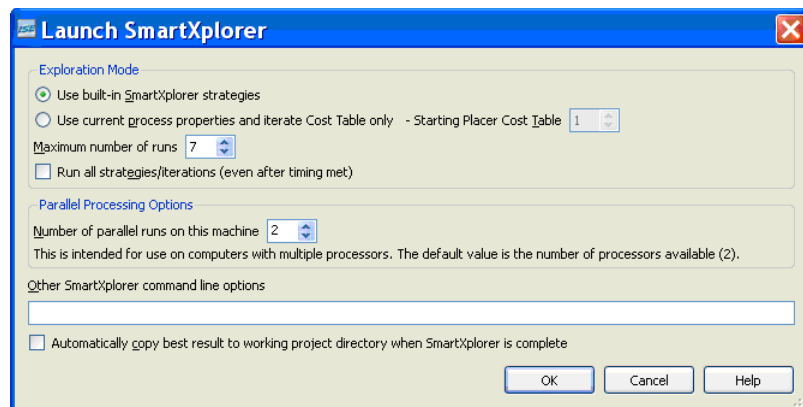


Figure A-2: All Built-in Predefined Strategies (Windows)

This configuration runs all 7 built-in predefined strategies (“Maximum Number of Runs” option). On Linux, they are executed on the machines specified in the host list file. On Windows operating system, two strategies are run in parallel on the same host. The execution stops as soon as timing is met.

## Task 2: Run the first 3 Built-in Predefined Strategies

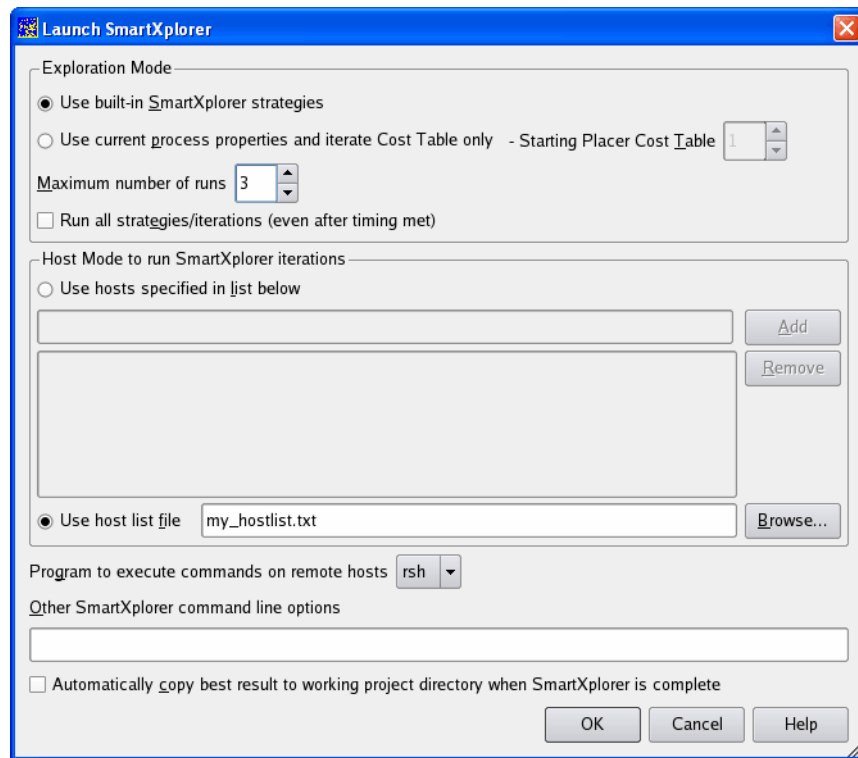


Figure A-3: First 3 Built-in Predefined Strategies (Linux)

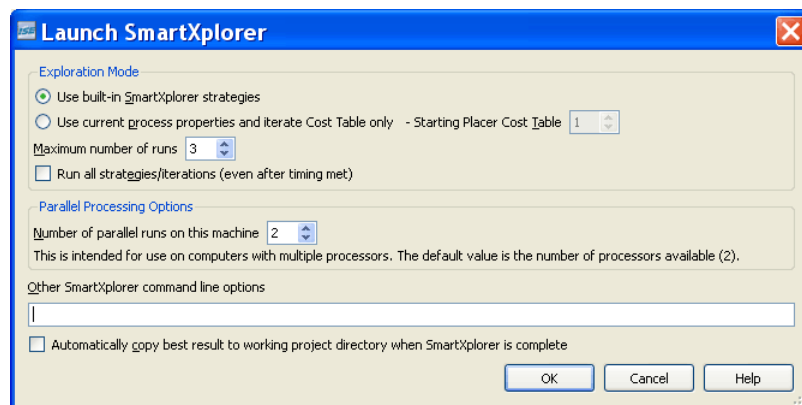


Figure A-4: First 3 Built-in Predefined Strategies (Windows)

This configuration runs the first 3 built-in predefined strategies (“Maximum Number of Runs” option). On Linux, they are executed on the machines specified in the host list file. On Windows operating system, two strategies are run in parallel on the same host. The execution will be stopped as soon as timing is met.

### Task 3: Run all built-in, predefined strategies and five additional iterations with different Cost Tables

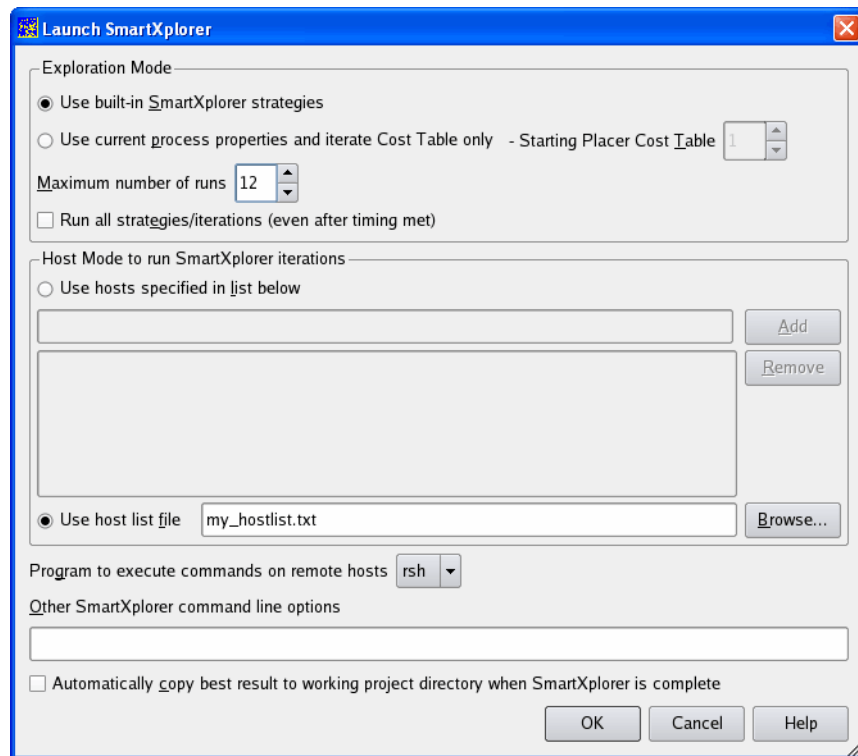


Figure A-5: All Built-in Predefined Strategies and 5 Additional Iterations (Linux)

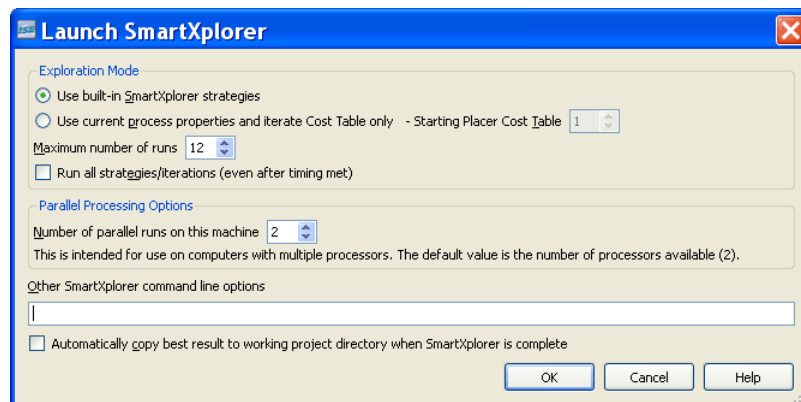


Figure A-6: All Built-in Predefined Strategies and 5 Additional Iterations (Windows)

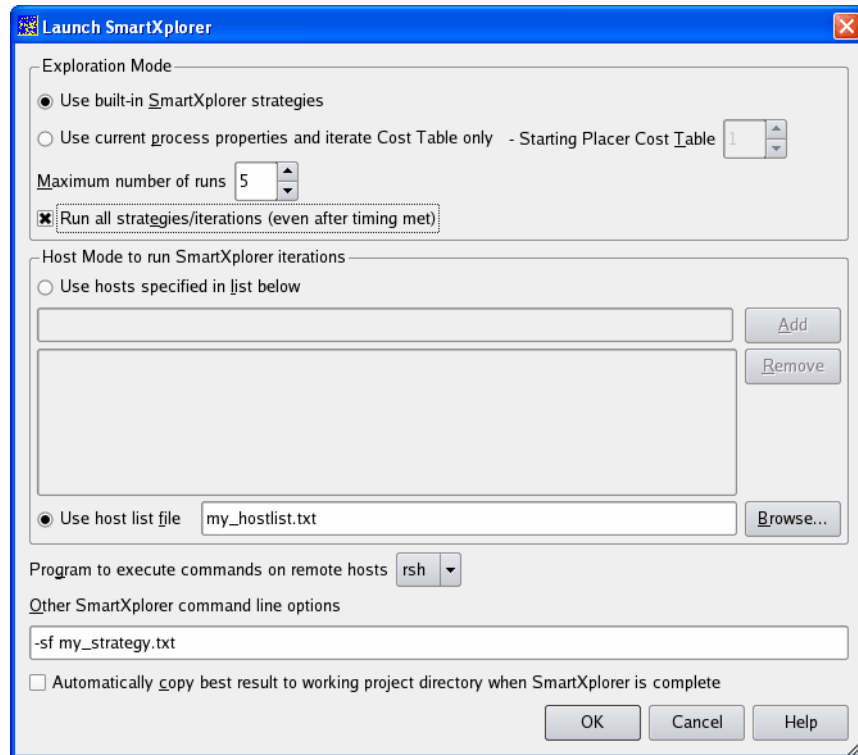
**Phase 1:** This configuration runs all 7 built-in predefined strategies first.

**Phase 2:** Then, SmartXplorer selects the best strategy from the Phase 1 and runs it 5 additional times with different Cost Tables (“Maximum Number of Runs” option: 7+5=12).

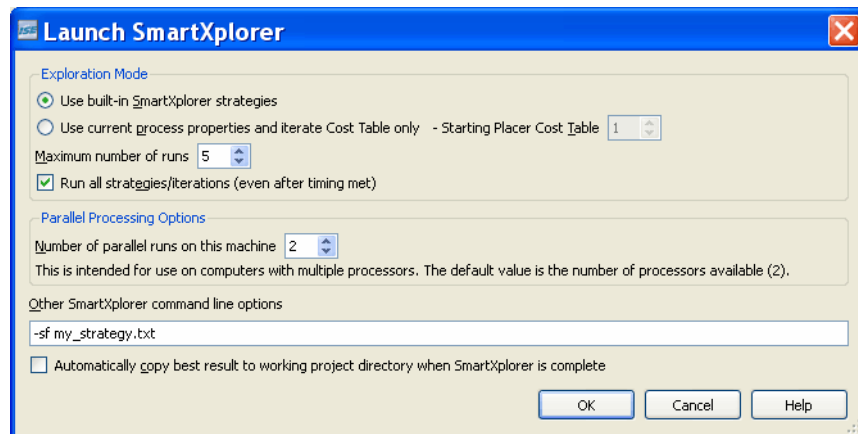
On Linux, strategies are executed on the machines specified in the host list file. On Windows operating system, two strategies are run in parallel on the same host.

The execution stops as soon as timing is met.

## Task 4: Run all custom strategies and 3 additional iterations with different Cost Tables



**Figure A-7: All Custom Strategies and 3 Additional Iterations with Different Cost Tables (Linux)**



**Figure A-8: All Custom Strategies and 3 Additional Iterations with Different Cost Tables (Windows)**



In this example, we suppose that the custom strategy file contains only two strategies.

**Phase 1:** This configuration runs the two strategies specified in the custom strategy file (-sf option specified in the "Other SmartXplorer command line options" field).

**Phase 2:** Then, SmartXplorer selects the best strategy from the Phase 1 and runs it three additional times with different Cost Tables ("**Maximum Number of Runs**" option: 2+3=5).

On Linux, strategies are executed on the machines specified in the host list file. On Windows operating system, two strategies are run in parallel on the same host.

All 5 strategies will be run regardless of the timing results ("**Run all strategies/ iterations**" option).

