SmartXplorer for Command Line Users

Tutorial (ISE 12.1)

UG688 (v12.1) April 19, 2010

XILINX®

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.

© 2010 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
6/12/09	1.0	Initial Xilinx release.
7/20/09	1.1	Updated for 11.2.
4/19/10	1.2	Updated for 12.1

Table of Contents

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® design tools:

- 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," describes the capabilities of SmartXplorer.
- 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 Designs for Labs," contains instructions on how to obtain the designs for each lab.
- Chapters 4, 5, and 6 contain detailed information for each lab, and 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 3: Running Multiple Strategies in Parallel"
- 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	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File →Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *Development System Reference Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr`** `={`**`on`**`|`**`off`**`}`
Vertical bar |	Separates items in a list of choices	**`lowpwr`** `={`**`on`**`|`**`off`**`}`

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	**`allow block`** *block_name loc1 loc2 ... locn;*

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 "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a Web site (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

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 this 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 of all, you have to estimate how many jobs your machine may run simultaneously.

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

```
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 may 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 may 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 to 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 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 the major features of SmartXplorer.

We strongly suggest running the labs in order (Lab1, Lab2, Lab3). 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:

1. It helps to achieve timing closure by using the predefined built-in or user-defined design strategies.
2. 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:

- Lab1 and Lab2 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, Lab3 is dedicated to running multiple strategies in parallel.

The following table gives you a brief overview of all the labs:

Table 2-1: **Lab Overview**

Title	Duration	Covered Features
Lab 1: Basic Flow	15 min	• How SmartXplorer can be easily launched • How final results are reported, stored and can be further used • How to run additional iterations to improve upon previously obtained • How to configure SmartXplorer to run the predefined strategies and additional iterations at once (for example, overnight runs)
Lab 2: Creating Custom Strategies	5 min	• How to create a custom strategy file and use it in SmartXplorer
Lab 3: Running Multiple Strategies in Parallel	10 min	• How to run several strategies in parallel

Knowledge Prerequisites

The labs require some basic knowledge of the major steps of the Xilinx® FPGA implementation flow and how to run them: Synthesis, Translate (NGDBuild), Map, Place & Route, and Timing Analysis (TRCE).

Chapter 3

Preparing Designs for Labs

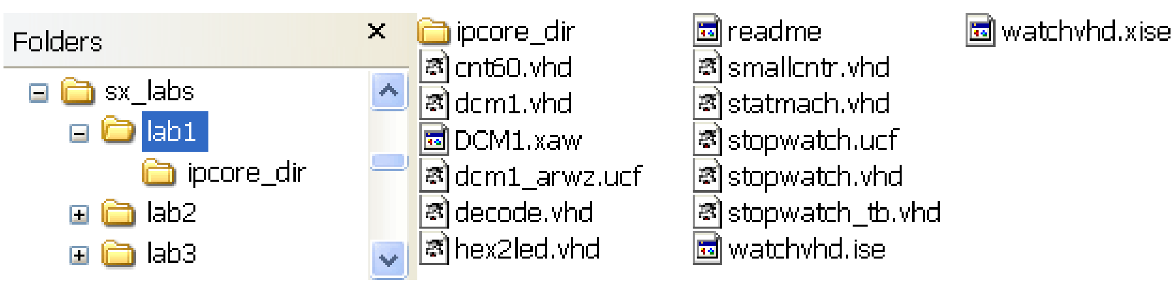
This section provides detailed instructions on how to prepare the design for each lab.

Design Description

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

Instructions

1. Download **UG688.zip** file from here.
2. Create an **sx_labs** directory, where you will store all SmartXplorer lab data.
3. Copy the **UG688.zip** file to the **sx_labs** directory.
4. Unzip the **UG688.zip** file. Ultimately, you should have a directory structure similar to the following view:



Chapter 4

Lab 1: Basic Flow

Scope

Objectives

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

- 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 to improve timing.

Lab

Step 1: Setup Xilinx Environment

1. Open a terminal window and setup the Xilinx® environment.
2. Go to the lab1 directory: **`cd .../lab1`**

Step 2: Run SmartXplorer with Mandatory Options

To run SmartXplorer (**`smartxplorer`**) you have to specify the following mandatory options:

- A target device: **-p xc3s100e-4-vq100**
- A UCF constraint file: **-uc stopwatch.ucf**
- A netlist file: **stopwatch.ngc**

In addition, we will use the following two options:

- Our design contains an external core (**tenth.ngc**) located in the **ipcore_dir** directory. We will use the **–sd** switch to reference to this directory.
- For the sake of clarity SmartXplorer will store its results in the **smartxplorer_results** directory: **–wd smartxplorer_results**

Now we are ready to run SmartXplorer.

1. In the current Terminal window, launch SmartXplorer using the following command:

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -wd
smartxplorer_results -sd='.;ipcore_dir' stopwatch.ngc
```

At the beginning, SmartXplorer runs NGDBuild. NGDBuild results are shared by all designs strategies.

```
Running Initial NGDBUILD on host xgr-sweng135...
```

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

- in the Terminal Window, and
- in the **smartxplorer.html** HTML file, located in the **smartxplorer_results** directory. You should use a Web browser to open this file.

These tables are progressively updated during SmartXplorer run. Following is an example of an intermediate status (**smartxplorer.html**):

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort	xgr-sweng135	run1	**Done**	4413	0h 0m 54s
MapPhysSynthesis	xgr-sweng135	run2	Mapping	None	0h 0m 3s
ParHighEffort1	None	None	None	None	None
ParHighEffort2	None	None	None	None	None
MapTiming1	None	None	None	None	None
MapTiming2	None	None	None	None	None
MapUseIOReg	None	None	None	None	None

As you can see from the **Done** status in the Status column, MapTimingExtraEffort strategy has been completed. The process took **54 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.

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

Note: To stop SmartXplorer execution before it completes all strategies, press CTRL-C in the Terminal window.

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

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort	xgr-sweng135	run1	**Done**	4413	0h 0m 54s
MapPhysSynthesis	xgr-sweng135	run2	**Done**	191	0h 0m 44s
ParHighEffort1	xgr-sweng135	run3	**Done**	2508	0h 0m 40s
ParHighEffort2	xgr-sweng135	run4	**Done**	3086	0h 0m 35s
MapTiming1	xgr-sweng135	run5	**Done**	4652	0h 0m 44s
MapTiming2	xgr-sweng135	run6	**Done**	2512	0h 0m 35s
MapUseIOReg	xgr-sweng135	run7	**Done**	4652	0h 0m 39s

2. Click on the **run2** link in the Output column of the MapPhysiSynthesis strategy to open a strategy log file: **stopwatch_sx.log** (it contains MAP, Place & Route and TRCE reports).

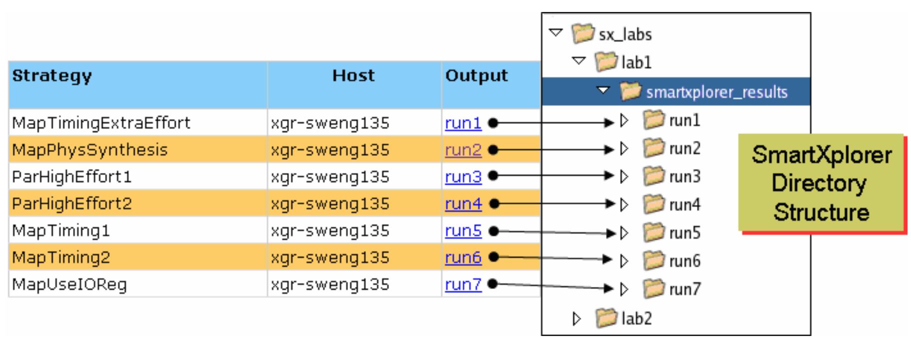
```
Release 11.1 - Map L.33 (lin64)
Copyright (c) 1995-2009 Xilinx, Inc.  All rights reserved.
Using target part "3s100evq100-4".
Mapping design into LUTs...
Writing file stopwatch_map.ngm...
Running directed packing...
Running delay-based LUT packing...
Updating timing models...
Running timing-driven placement...
Total REAL time at the beginning of Placer: 5 secs
Total CPU  time at the beginning of Placer: 2 secs

Phase 1.1  Initial Placement Analysis
Phase 1.1  Initial Placement Analysis (Checksum:1424) REAL time: 6 secs
```

3. Click on the **191** link in the Timing Score column of the MapPhysSynthesis strategy to open a timing report summary.

Asterisk (*) preceding a constraint indicates it was not met.
This may be due to a setup or hold violation.

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_Inst_dcm1_CLK0_BUF = PERIOD TIMEGRP "Inst_dcm1_CLK0_BUF" TS_CLK HIGH 50%	SETUP	-0.155	4.355	3	191
	HOLD	1.310		0	0
TS_CLK = PERIOD TIMEGRP "CLK" 4.2 ns HIGH 50%	MINPERIOD	0.034	4.166	0	0

4. Review the results directory structure. All SmartXplorer results are stored in the **smartxplorer_results** directory. Each SmartXplorer strategy is stored in a separate directory: **run1, run2, …, run7.**



IMPORTANT: If you want to restart SmartXplorer, you should close all reports in text editors opened for SmartXplorer results first. 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 Terminal window and stop execution.

```
ERROR: SmartXplorer could not clean up directory run1 from previous run. Please
ensure that no other programs are using this directory.
```

Step 3: 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).

1. To run the MapPhysSynthesis strategy 5 additional times, we must use the Variability Passes option: **–vp 5**.
2. We have to specify MAP and PAR options from the MapPhysSynthesis strategy using the **–mo** and **–po** options respectively. These options can be found in the **smartxplorer.txt** file located in the **smartxplorer_results** directory. This file contains the options for all previously run strategies.

```
-----------------------------------------------------------------------
BestStrategy : MapPhysSynthesis
------------------------------------------------------------------------
 Run index    : run2
 Map options  : -timing -ol high -xe n -register_duplication on -logic_opt on
 Par options  : -ol high
```

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: **-t 2**. These options must be added to the MAP and PAR commands:

```
-mo "-timing -ol high -xe n -register_duplication on -logic_opt on -t 2"
-po "-ol high -t 2"
```

3. In the current Terminal window, launch SmartXplorer using the following command:

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -wd
smartxplorer_results -sd='.;ipcore_dir' -vp 5 -mo "-timing -ol high -xe
n -register_duplication on -logic_opt on -t 2" -po "-ol high -t 2"
stopwatch.ngc
```

The result is that additional SmartXplorer runs were able to meet timing.

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort*	xgr-sweng135	run1	Done	192	0h 1m 5s
MapTimingExtraEffortCT3*	xgr-sweng135	run2	Done	198	0h 0m 44s
MapTimingExtraEffortCT4*	xgr-sweng135	run3	Done	367	0h 0m 54s
MapTimingExtraEffortCT5*	xgr-sweng135	run4	Done	0	0h 0m 44s

As you can see, only 4 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: MapTimingExtraEffortCT5.

You can use the **–ra** (Run All) option to run all iterations and get the complete picture on the relative performance of each cost table.

Step 4: Run the Predefined Strategies and Additional Iterations at Once

In the previous steps, we were able to meet timing constraints using a 2-phase approach. In the first phase, "Step 2: Run SmartXplorer with Mandatory Options," we ran all predefined strategies and identified the best one. In the second phase, "Step 3: Running Additional Iterations to Improve Timing," we selected the best strategy from the first phase and iterated 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.

1. Specify Maximum number of runs equal to 12 (7 predefined + 5 additional iterations): **-m 12**.
2. Use the **–ra** (Run All) option to get the complete picture.
3. In the current Terminal window launch SmartXplorer using the following command:

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -wd
smartxplorer_results -sd='.;ipcore_dir' -m 12 -ra stopwatch.ngc
```

You will receive the following final results:

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort	xgr-sweng135	run1	Done	4413	0h 0m 45s
MapPhysSynthesis	xgr-sweng135	run2	Done	191	0h 0m 45s
ParHighEffort1	xgr-sweng135	run3	Done	2508	0h 0m 39s
ParHighEffort2	xgr-sweng135	run4	Done	3086	0h 0m 34s
MapTiming1	xgr-sweng135	run5	Done	4652	0h 0m 44s
MapTiming2	xgr-sweng135	run6	Done	2512	0h 0m 35s
MapUseIOReg	xgr-sweng135	run7	Done	4652	0h 0m 44s
MapPhysSynthesisCT2	xgr-sweng135	run8	Done	192	0h 0m 49s
MapPhysSynthesisCT3	xgr-sweng135	run9	Done	198	0h 0m 44s
MapPhysSynthesisCT4	xgr-sweng135	run10	Done	367	0h 0m 44s
MapPhysSynthesisCT5	xgr-sweng135	run11	Done	0	0h 0m 49s
MapPhysSynthesisCT6	xgr-sweng135	run12	Done	15	0h 0m 44s

Conclusion

In this lab, we ran SmartXplorer and obtained results for the seven predefined strategies (Phase 1).

We then ran the best strategy five additional times with different Cost Tables to further improve timing (Phase 2).

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 seven predefined strategies you found out that two of them always gave you much better results than the other five. 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: Setup Xilinx Environment

1. Open a terminal window and setup the Xilinx® environment.
2. Go to the **lab2** directory: **`cd .../lab2`**

Step 2: Identify the Best Two Strategies

In Lab 1, we acquired the following results for seven predefined strategies, where MapPhysSynthesis and MapTiming2 were the best ones:

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort	xgr-sweng135	run1	Done	4413	0h 0m 54s
MapPhysSynthesis	xgr-sweng135	run2	Done	191	0h 0m 44s
ParHighEffort1	xgr-sweng135	run3	Done	2508	0h 0m 40s
ParHighEffort2	xgr-sweng135	run4	Done	3086	0h 0m 35s
MapTiming1	xgr-sweng135	run5	Done	4652	0h 0m 44s
MapTiming2	xgr-sweng135	run6	Done	2512	0h 0m 35s
MapUseIOReg	xgr-sweng135	run7	Done	4652	0h 0m 39s

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 is stored in the **smartxplorer.txt** file located in the **smartxplorer_results** directory of Lab 1. This file contains the options for all previously run strategies. The contents of this file can be found in the following location:

1. **lab1_smartxplorer.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
...
```

2. 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.

   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"},
),
}
```

3. Specify the **my_strategy.txt** file via **–sf** switch in SmartXplorer command.

   To ensure that we will run only two custom strategies, you must set the **Maximum number of runs** value to **2**.

   Please note that if, for example, you specify 5 as the Maximum number of runs (instead of two), then SmartXplorer will select the best strategy from 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.

4. In the current terminal window, launch SmartXplorer using the following command:

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -wd
smartxplorer_results -sd='.;ipcore_dir' -sf my_strategy.txt -m 2
stopwatch.ngc
```

5. SmartXplorer generates the following results, matching the ones we obtained at the beginning (see **smartxplorer.html**):

Strategy	Host	Output	Status	Timing Score	Total Run Time
My_Strat_1	xgr-sweng135	run1	Done	191	0h 1m 0s
My_Strat_2	xgr-sweng135	run2	Done	2512	0h 0m 35s

Conclusion

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

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

Chapter 6

Lab 3: Running Multiple Strategies in Parallel

IMPORTANT: This lab is dedicated to users having one of the following:

- Access to a regular Linux network, LSF and SGE compute farms.
- 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 Xilinx software environment on each machine.
- Where SmartXplorer results will be stored.

1. Setting up 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 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:

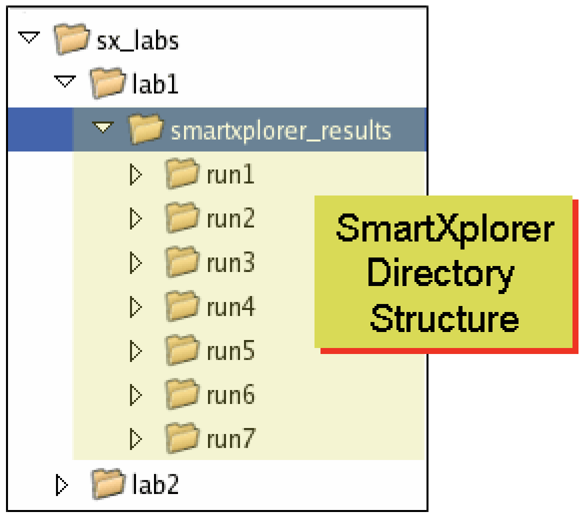
   - the network, then L2 (L3) must have access to this installation as well using the same network mount points so that the network paths defined for L1 are valid for all machines.
   - the L1 local disk, then L2 (L3) must have the same version of Xilinx software installed on a local disk and placed in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

2. Results Storage

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



   The same method is used in the case of multiple machines; all results are stored in the same disk area. Therefore, all machines must have access to this disk area and have read and write permissions.

Step 2: Pre-run Checklist

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

Step 3: Setup Xilinx Environment

1. Open a terminal window and setup Xilinx environment.
2. Go to the **lab3** directory: **`cd .../lab3`**

Step 4: Run SmartXplorer on a Regular Linux Network

In this step, we will run the seven 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 SmartXplorer.

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, we must list this machine twice, as it is done for xgr-sweng135.

   Note: Please replace xgr-sweng135 and xgr-sweng109 with your machine names in the my_hostlist.txt file and save it before continuing.

2. Specify the **my_hostlist.txt** file via **–l** option.
3. In the current terminal window launch SmartXplorer using the following command:

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -wd
smartxplorer_results -sd='.;ipcore_dir' -l my_hostlist.txt
stopwatch.ngc
```

4. You might get the following status table, where more powerful xgr-sweng135 was able to complete five out of seven strategies:

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapTimingExtraEffort	xgr-sweng135	run1	Done	4413	0h 0m 54s
MapPhysSynthesis	xgr-sweng135	run2	Done	191	0h 1m 0s
ParHighEffort1	xgr-sweng109	run3	Done	2508	0h 1m 26s
ParHighEffort2	xgr-sweng135	run4	Done	3086	0h 0m 41s
MapTiming1	xgr-sweng135	run5	Done	4652	0h 0m 46s
MapTiming2	xgr-sweng109	run6	Done	2512	0h 1m 12s
MapUseIOReg	xgr-sweng135	run7	Done	4652	0h 0m 39s

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 **–ra** (Run all) option is not used.

Strategy	Host	Output	Status	Timing Score	Total Run Time
MapPhysSynthesisCT3	xgr-sweng135	run9	**Done**	198	0h 0m 40s
MapPhysSynthesisCT4	xgr-sweng135	run10	**Done**	367	0h 0m 36s
MapPhysSynthesisCT5	xgr-sweng135	run11	**Done**	0	0h 0m 30s
MapPhysSynthesisCT6	xgr-sweng135	run12	Stopped	None	0h 0m 19s

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.

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 must create a host list file and specify it in the "Use host list file" field.

The main difference exists 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 with an 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 you 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 you to define additional SGE options and additional_options must be replaced by 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 **six** 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 that must be taken into account before using SmartXplorer across the network.

Appendix A

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

The following 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 runs four strategies simultaneously. They are run on three different Linux machines and **lin_machine_1** runs two strategies in parallel.

LSF Compute Farm

The following 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

Following 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

The following script runs all seven built-in predefined strategies (the **-m** option with a default value of **7** is skipped). They are executed on the machines specified in the host list file
(**-l** option). The execution stops as soon as timing is met.

```
smartxplorer -p xc3s100e-4-vq100 -uc <file>.ucf -l <host_list_file> <design>.ngc
```

Task 2: Run the first three built-in predefined strategies

The following script runs the first three built-in predefined strategies (**-m 3** option). They are executed on the machines specified in the host list file (**-l** option). The execution stops as soon as timing is met.

```
smartxplorer -p xc3s100e-4-vq100 -uc <file>.ucf -l <host_list_file> -m 3
<design>.ngc
```

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

Phase 1: this script runs all seven built-in predefined strategies first.

Phase 2: then SmartXplorer selects the best strategy from phase-1 and runs it five additional times with different Cost Tables (**-m 12** option: 7+5=12).

Strategies are executed on the machines specified in the host list file (**-l** option).

The execution stops as soon as timing is met.

```
smartxplorer -p xc3s100e-4-vq100 -uc <file>.ucf -m 12 -l <host_list_file>
<design>.ngc
```

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

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

Phase 1: this script runs the two strategies specified in the custom strategy file (**-sf** option).

Phase 2: then SmartXplorer selects the best strategy from phase-1 and runs it three additional times with different Cost Tables (**-m 5** option: 2+3=5).

Strategies are executed on the machines specified in the host list file (-l option).

All five strategies are run regardless of the timing results (**-ra** option).

```
smartxplorer -p xc3s100e-4-vq100 -uc <file>.ucf -m 5 -l <host_list_file> -sf
<strategy_file> -ra <design>.ngc
```