ORCA®
FPGA Express™ Interface Manual

ispLEVER® version 3.0

For Use With
Synopsys® FPGA Express™ version 3.5 or lower, ORCA 2002, and
ispLEVER 2.0 and higher

Technical Support Line: 1-800-LATTICE or 408-826-6002 (international)
ORCA/FPGA Express Interface
CONTENTS

OVERVIEW .................................................................1
RELATED DOCUMENTATION .........................1
SOFTWARE REQUIREMENTS ..........................2
SETTING THE DESIGN ENVIRONMENT ............2
Environment Variables .................................2
ORCA MACRO LIBRARY ...............................3
Library Contents .......................................3
DESIGN FLOW ................................................5
Series 4 Synthesis Flow ............................7
DESIGN ENTRY ...............................................9
DESIGN IMPLEMENTATION ......................11
Mapping the Design in the ORCA Environment 12
DESIGN VERIFICATION .............................14
APPENDIX .................................................. A-1
Programmable GSR Support (Series 3) ........ A-1
Passing FREQUENCY Preferences (Series 3 and 4) A-2
OVERVIEW

This manual describes the Optimized Reconfigurable Cell Array (ORCA) Library and interface between the Synopsys™ FPGA Express™ design tool and the ORCA place and route tools. Together, the tools provide a powerful and integrated high-level design environment for ORCA Field-Programmable Gate Arrays (FPGA).

Note

Note that version numbers for the interface software are continually being updated with each release. Check with the vendor on version number and how it affects the file names or syntax that make it possible to perform tasks as they relate to creating designs with ORCA. Check with technical support with compatibility and support issues.

SOFTWARE REQUIREMENTS

The ORCA/FPGA Express Interface is compatible with the following software:

- Latest version of ORCA
- Synopsys FPGA Express 3.5 or lower

SETTING THE DESIGN ENVIRONMENT

This section helps you customize your Synopsys FPGA Express environment for designing an ORCA FPGA.
Environment Variables

Make sure you have installed the latest version of ORCA software and that the FOUNDRY environment variable is set. The FOUNDRY variable points to the library installation directory.

On the workstation, the FOUNDRY variable should be set as follows:

```
$ setenv FOUNDRY <library_installation_directory>
```

Note

This manual assumes that you are using the c-shell. If not, use the appropriate syntax and procedures to set the environment variable for your shell.

This command may be entered into your .cshrc or .login file so that the environment is properly set upon login.
ORCA MACRO LIBRARY

Library Contents

Library elements are cells (AND gates, counters, I/O buffers, etc.) with which you implement an ORCA design through the synthesis flow. All the ORCA Macro library elements are in our Synopsys cell library. Not all the functions may be used automatically by Synopsys, but all the cells can be instantiated in behavioral code.

For the 2CA/2TA/2TB architecture, the library elements are divided into the following categories:

- Combinational Logic Gates
- Comparators
- I/O Buffers
- Latches
- Flip-Flops
- Multiplexers
- Demultiplexers
- Registers
- Counters
- Loadable Counters
- Adders
- Subtracters
- Multipliers
- Memory
- Special ORCA Architecture Elements
- ORCA Look-Up-Tables (LUTs)
For the Series 3 and Series 4 architectures, the library elements are divided into the following categories:

- Combinational Logic Gates
- Latches
- Flip-Flops
- Multiplexers
- Counters
- Loadable Counters
- Adders
- Subtractors
- Multipliers
- Comparators
- Memory
- SLIC Gates
- PLC 9th Flip-Flop
- I/O Buffers
- Bidirectional Buffers
- PIC 2-Input Function Generators
- PIC Latches
- PIC Registers
- PIC Latched Flip-Flops
- PIC Multiplexers
- Wide Functions
- Special ORCA Series 3 and Series 4 Architecture Elements
- ORCA Look-Up-Tables (LUTs)

For more detailed information about the ORCA FPGA architectures, refer to an available *FPGA Data Book*. For information on the functionality and the pin order of the cells in the library, refer to the appropriate topic in the online help system.
DESIGN FLOW

This section describes the interface between FPGA Express and ORCA. The interface allows you to:

- Synthesize an ORCA design using FPGA Express.
- Use the ORCA tools to read in the EDIF file, map the design into a selected ORCA device, then place and route the design.
- Program the final design into the selected ORCA device.

Figure 1 shows a typical design flow for generating logic designs with FPGA Express and the ORCA software. For additional information on the synthesis flow, consult your Synopsys FPGA Express documentation. For information on the ORCA Development System, consult your ORCA manuals and tutorials.
Figure 1. ORCA/FPGA Express Interface Design Flow
Series 4 Synthesis Flow

This section contains any information that is necessary for performing design synthesis using VHDL and Verilog® HDL for Series 4 designs.

New Series 4 library elements, for example, new Block RAMs (BR512X18), Programmable PLLs (PPLL), and LVDS buffers, are currently supported by only some synthesis vendors, so there is some user intervention required for instantiation and assigning any properties to these elements.

Actions necessary to ensure proper instantiation and for assigning properties for Series 4 elements are described in the following subsections.

Using VHDL Through Synthesis

Since Series 4 elements are not yet supported by all synthesis vendors, VHDL designs may require these elements to be defined as black boxes or macros during synthesis Please check with your vendor on ORCA Series 4 support.

To enter properties for Series 4 elements using VHDL, the syntax structures are the same for most vendors. An example of assigning properties for FPGA Express is provided below.

VHDL syntax for use with Synopsys FPGA Express:

attribute initval: string;

attribute initval of mem_0_0_15 : label is "0x00000000000000000000000000000000";

Using Verilog Through Synthesis

When you instantiate new Series 4 elements in Verilog HDL, synthesis tools issue warning or error messages. To avoid this, we have provided Verilog header files, which contain port definitions for new Series 4 elements. These files are provided for all the synthesis tools and for the Synopsys FPGA Express/FPGA Compiler™, it is located in the file path as:

$FOUNDROY/userware/unix/4E_Verilog_Headers/fc2.v

Depending on what tool is being used, you should add proper header file along with other Verilog files to the project.
Verilog syntax for use with Synopsys FPGA Express:

// synopsys attribute INITVAL "0x00000000000000000000000000000000"
(in each instance)

Ex:- PLLT PLLT_1 (.CLKIN(CKINT), .FB(SCLKT), .ECLK(ECLKT),
    .SCLK(SCLKT), .LOCK(LOCKT)); //synopsys attribute div0 "1" div1 "2" div2
    "2" fbdelay "1" disabled_gsr "1"
DESIGN ENTRY

The following steps outline how to synthesize designs for ORCA with Synopsys FPGA Express and ORCA:

1. Create a design in Verilog HDL, VHDL, or EDIF. The designs may be technology-independent or contain ORCA-specific functions; however, they cannot contain instances of functions from other technology libraries. See Illegal Names and Characters in the ORCA Libraries Manual for a list of names and characters that should not be used for components, nets, sites, or instances.

2. SCUBA (Synthesis Compiler for User programmable Arrays), is a parameterized module generator optimized for ORCA FPGAs. SCUBA accepts options (command-line or through GUI) that specify parameters for parameterized modules such as data-path modules and memory modules, and produces a circuit description with ORCA Library elements. Since the modules generated are optimized for the ORCA architecture, they provide speed and area benefits over synthesis.

   The output from SCUBA can be in EDIF, VHDL, or Verilog. The output can be directly read into Design/FPGA Compiler or can be integrated with the user design as a sub-module. For a complete description of SCUBA and a list of parameterized modules supported by SCUBA, refer to the appropriate topic in the online help system.

3. (Optional) Verify that the design description is correct by simulating the HDL description.

4. Set up the design by creating a project in FPGA Express and identifying and analyzing the source files.

5. Select the target architecture and device, including the package and speed grade.

6. Create (Synthesize) the design implementation.
7. Specify the Design Constraints using the design implementation spreadsheets. The design implementation spreadsheets enable the designer to control the optimization of specific modules, entities, and sub-designs. The tables are customized for the ORCA device family. The tables will help you capture the period, rise time, and fall time for each system clock, the input and output delays at each port, and other design-specific port attributes such as built-in I/O registers and pull-up and pull-down registers. The designer can also lock the I/Os by specifying a pad location in the Pad Loc column for the corresponding port. The optimization controls for hierarchy, primitives, and arithmetic operator sharing have a strong effect on the final results. The designer may choose to preserve hierarchical boundaries during optimization or flatten the design to optimize logic across the hierarchy.

8. After the constraints and controls are fully captured, close the design implementation window and optimize the design.

9. Finally, export the design netlist as an EDIF file for ORCA, and generate a design report for documentation and review.

10. If you want to export timing constraints to ORCA place and route tools, check the Expand Timing Specifications option. This will generate a logical preference file (.lp) for your design which then can be used to generate a physical preference file (.prf) for your design. The physical preference file contains timing constraints for the design.

For more detailed information, refer to the Synopsys FPGA Express User’s Guide and Online Help.
DESIGN IMPLEMENTATION

In this step of the design process, the circuit is mapped, placed, and routed using the following steps:

- Translate Synopsys FPGA Express timing constraints into an ORCA preference file.
- Compile the EDIF netlist for ORCA with the ORCA Map tool. (See the section Mapping the Design in the ORCA Environment.) The output of the mapping phase is an NCD database, named <design>.ncd, which is submitted to the placer and router (PAR) along with timing preferences.
- The order of execution of the tools are: FPGA Express -> Map -> Trace -> PAR -> Trace -> EPIC.
- To analyze the timing of your design, use the Trace static timing analysis tool.
- To interactively edit the physical design, use EPIC.

Figure 2 shows a typical design implementation flow using the ORCA tools. For a complete description of how to use the ORCA design implementation tools, see the appropriate topic in the online help system.

Figure 2. Design Implementation

Mapping the Design in the ORCA Environment
This section discusses the method for reading Synopsys-created EDIF designs into the ORCA design environment. It addresses only the process of mapping a Synopsys EDIF netlist into a Circuit Description (.ncd) file. Once the design is represented as an .ncd file, the full complement of the ORCA tools may operate on it. This includes providing detailed static timing analysis of the design’s structure, creating a physical implementation of the design using timing driven PAR, and Bit Generation.

Each of the ORCA tools may be run from within ORCA Control Center (OFCC), from the command line, or from the shells from prior versions of ORCA. Please refer to the appropriate topic in the online help system for guidelines for mapping Synopsys-created EDIF designs into the ORCA environment using OFCC, and for guidelines for mapping designs from the command line.

### Note

The ORCA shells are no longer a part of the ORCA Development System but still may be installed from the CD. For HP users, the file path for accessing the shells is `$FOUNDRY/bin/HP`, for Solaris users the file path is `$FOUNDRY/userware/unix/bin/sol`, and for PC users the file path is `%FOUNDRY%/userware/NT/bin/nt`.

The following describes how to use the MAP Shell dialog box to map the Synopsys-created .edn file into the ORCA .ncd and .prf files.

1. Invoke the MAP Shell dialog box by entering the command `mapsh` on the UNIX command line.

2. Enter the name(s) of the design(s) to be mapped in the Input Files List. The top level of a hierarchical design must be the first file in the list.

3. Fields that require special consideration when mapping a design produced by Synopsys are contained in the Options dialog box. Push the Options button to display this form.

4. Turn on the Remove Unused Logic option to optimize your designs.

5. Press the OK button in the MAP Options dialog box to accept its configuration and return to the MAP Shell dialog box.
6. On the MAP Shell dialog box, press the **OK** button to begin the mapping operation.

The EDIF2NGD and NGDBUILD operations create the generic database (.ngd) design file, which contains the fully expanded design. Finally, the MAP program is run to create the circuit description (.ncd) file, which represents the mapped physical FPGA design. The MAP shell creates a script file (output_filename.msc) which is run (and may be later modified and rerun). A MAP report file (.mrp) file is also created which is cross-referencing logical instances to their corresponding physical instances in ORCA.

The timing constraints are written into the .ngo files as Preference Language preferences. These constraints are then written into the .prf preference file during the technology mapping operation of MAP.

For additional information on how to run the other ORCA tools, refer to the appropriate topic in the online help system on Preference Language, TRACE, and PAR. A list of related documents is provided at the beginning of this manual.
DESIGN VERIFICATION

Design verification involves three operations:

• *(Optional)* Using a standard supported simulator to perform full timing simulation of the Verilog, VHDL, or EDIF output file generated by ORCA. Currently, we support Model Technology® V-System™ and Synopsys VSS™ simulators for VITAL (VHDL) simulations, Cadence Verilog-XL® for Verilog simulations, and Viewlogic® ViewSim® for EDIF simulations.

• Producing a data bitstream (using Bit Generation) that will be written into the device.

• Physically loading the configuration data into the device.

For a complete description of how to use the ORCA design verification tools, see the *ORCA User’s Guide* and the appropriate simulator interface and reference manuals.
Programmable GSR Support (Series 3)

Programmable GSR support allows the user to disable the GSR functionality on selected register elements. This feature gives the user the ability to retain design information on a PFU basis when using GSR to set/reset the rest of the design.

The PUR component will now be used to set/reset register elements at power-up. GSR is used for global set/reset at all other times of operation. The PUR signal will need to be instantiated in your HDL design since it is currently not inferred.

In ORCA 2002 programmable GSR will be supported via an attribute attached to instantiated library cell instances in the HDL design netlist. The attribute is "DISABLED_GSR" with value "1". The resulting synthesized output EDIF netlist will contain the property "DISABLED_GSR" with value "1" for each component instance containing the attribute. The ORCA tools will recognize this property and disable the GSR functionality accordingly.

Example VHDL to implement programmable GSR:

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
entity attr_chk is
  port(d, clk, preset : in std_logic;
       qout : out std_logic);
architecture flop of attr_chk is
  component FD1S3BX
    generic (disabled_gsr: String);
    port(D, CK, PD : in std_logic;
         Q : out std_logic);
  end component;
  attribute disabled_gsr: String;
  attribute disabled_gsr of u1:label is "1");
begin
  u1 : FD1S3BX generic map (disabled_gsr => "1")
    port map(
      D => d, CK =>clk, PD => preset, Q => qout);
end flop;
```
Example Verilog to implement programmable GSR:

```verilog
module attr_chk (d, clk, preset, qout);
    input d, clk, preset;
    output qout;

    FD1S3BX u1(.D(d),.CK(clk),.PD(preset),.Q(qout));
    defparam u1.DISABLED_GSR=1
    // synopsys attribute DISABLED_GSR "1"
endmodule
```

*Note that boldfaced code is necessary for simulation.

The HDL code examples include "generic map" entries (VHDL) and a "defparam" entry (Verilog). These must be manually added to your post-synthesis HDL netlist for simulation after synthesis. Synthesis tools currently do not produce these entries in their synthesized netlists.

### Passing FREQUENCY Preferences (Series 3 and 4)

FREQUENCY attributes will be converted into properties in the EDIF during synthesis which the mapper will then translate into FREQUENCY preferences on the output of the PLL. You must ensure that the FREQUENCY values are consistent with the input clock frequency, DIV0, DIV1, DIV2, DIV3 values and external divider, if any.

The following example shows how attributes are passed in the HDL code using Synopsys FPGA Express:

```verilog
PPLL macro_express_0_0 (.CLKIN(clk), .FB(fb),
    .MCLK(oclkl_t), .NCLK(oclk2_t), .LOCK(lock), .INTFB(fb)); //
    synopsys attribute VCOTAP "2" NCLKMODE "DELAY" MCLKMODE
    "DELAY" DIV3 "2" DIV2 "3" DIV1 "4" DIV0 "1"
```
INDEX

A
Attributes
  passing FREQUENCY preference in HDL, A-2

D
Design
  entry, 9
  environment setting, 2
  flow, 5
  implementation, 11
  verification, 14

E
Environment variables, 2

F
FPGA Express
  and synthesis, 9
  FREQUENCY preference, A-2

L
Libraries
  ORCA Macro, 3

M
MAP
  mapping the design, 12

P
passing using Synopsys FPGA Express, A-2

R
Requirements
  software, 2

S
Series 4 architecture

ORCA/Synopsys FPGA Express Interface
  Index I-1