Designing UltraLogic™ With Exemplar and Synopsys™

Introduction

Galileo™ from Exemplar Logic and the Design Compiler from Synopsys™ provide two pathways for programmer logic users to use Cypress's UltraLogic™ devices with third-party design environments. They provide behavioral Hardware Description Language (HDL) synthesis through the support of a wide variety of HDL design entry formats and powerful constraint-driven synthesis and optimization capabilities. Both of these tools integrate tightly with Cypress's *Warp*™ design tool to complete the design flow when targeting UltraLogic devices.

This application note is intended to familiarize the reader with these two third-party design tools, as well as the Cypress-specific design pathway by covering the following topics:

- Design entry formats
- UltraLogic device support
- Software Requirements
- Design flow and integration with *Warp*
- Design Synthesis and Optimization Capabilities

EXEMPLAR LOGIC – GALILEO

Galileo consists of three separate modules—the Logic Explorer (the synthesis engine), the Time Explorer (the timing analysis engine), and the V-System (the simulation engine). We will focus mainly on the capabilities of the Logic Explorer and its integration with *Warp*™.

Design Entry Formats

The Logic Explorer provides powerful behavioral synthesis by supporting a wide variety of design entry formats:

- VHDL (IEEE 1164 & 1076)
- Verilog™
- Palasm 2™
- OpenABEL™

Various formats of netlist are also supported for design retargeting and conversion:

- EDIF 2 0 0
- Berkeley PLA
- Actel ADL
- Xilinx XNF

The following design entry format is also provided to facilitate the integration of multiple designs in different formats:

- Exemplar Logic Integration Language (EIL)

UltraLogic Device Support

Logic Explorer currently supports the following family of programmable logic devices from Cypress:

- MAX340® EPLDs
- FLASH370™ CPLDs
- pASIC380™ FPGAs

Software Requirements

To design with the MAX340 and FLASH370 devices, *Warp2* alone is sufficient.

To design with the pASIC380 devices, *Warp2+* is required as a minimum.

Design Flow and Integration with *Warp*

The Logic Explorer–*Warp* design flow includes design entry, synthesis and optimization, fitting (for MAX340 and FLASH370) or place & route (for pASIC380), simulation, and programming (see *Figure 3*). Designs in design entry formats supported by Exemplar can be entered using any text editor, which then goes through the Logic Explorer for synthesis and optimization. The output from the Logic Explorer then goes into *Warp* for fitting or place & route, and programming files and/or timing models are generated by *Warp* for device simulation and programming.

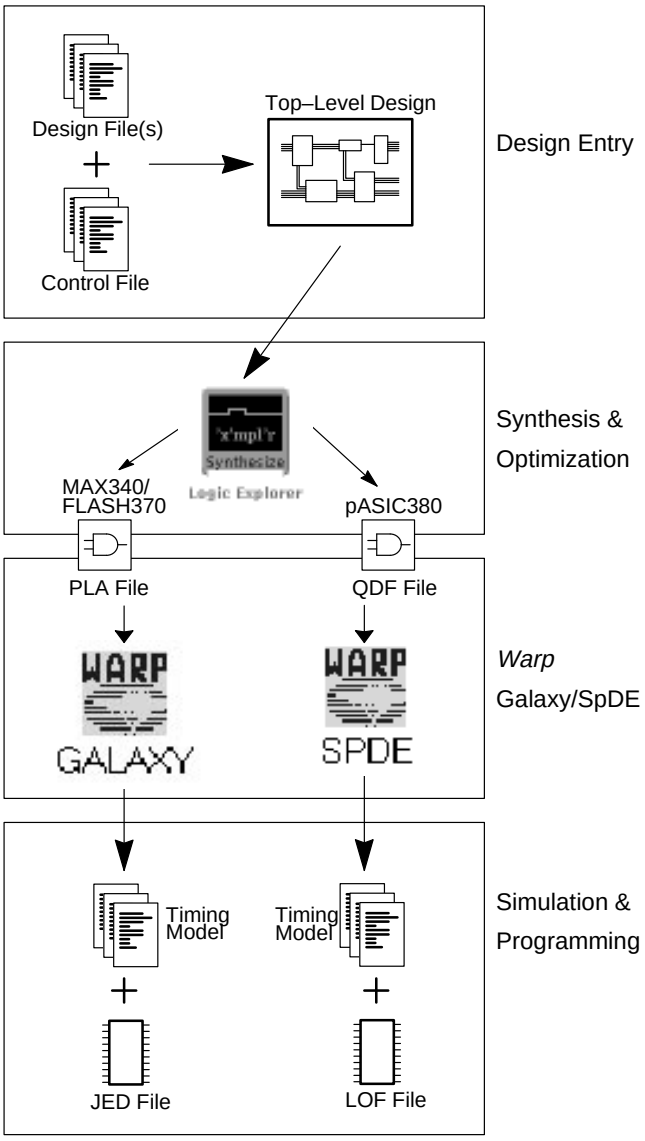

Figure 3. Logic Explorer–*Warp2* Design Flow

Details about each of the design stages are described below:

(A) Design Entry

Designs (in description languages, netlist, or EIL) are entered using any text editor and saved as ASCII text files. Hierarchical designs can be described across multiple design files. The Exemplar Logic Integration Language (EIL) can also be used to link multiple design files in different entry formats into one large design.

An optional control file can be included to specify design-specific parameters such as defining I/O pad mappings and timing requirements. The control file should have the same name as the design file with a .ctr extension.

(B) Design Optimization and Synthesis using the Logic Explorer

The next step is to synthesize and optimize the design(s) using the Logic Explorer.

The Logic Explorer main window allows the user to specify design-specific information like input and output filenames, source and target technology, and entry format, as well as synthesis and runtime options (for details refer to Design Environment below). Depending on the target technology and user-specified constraints, a number of optimization passes (ranging from one to eleven) will be run. The results of these passes will be plotted on an Area vs. Delay graph. The user can save the pass that best fulfills the user-specified constraints.

(C) Device Fitting or Place & Route using *Warp*

After synthesis and optimization, the results generated by the Logic Explorer will be used in *Warp* for fitting or place & route.

Depending on the target technology, results from the Logic Explorer will be saved in the following output formats for interfacing with *Warp:*

Table 1. Logic Explorer to *Warp* Output Formats

Target Technology	Output Format
MAX340	PLA
FLASH370	PLA
pASIC380	QDIF

For MAX340 and FLASH370 devices, a PLA file will be generated by the Logic Explorer which will be taken by the *Warp* Galaxy fitter as input to perform device fitting. A JEDEC file will be produced for device programming and timing models will be generated for device simulation.

For pASIC380 devices, a QDIF file will be generated by the Logic Explorer which will be taken by the *Warp* SpDE place & route tool as input to perform place & route and timing analysis. A LOF file will be generated for device programming and timing models will be generated for device simulation.

Design Synthesis and Optimization Capabilities

We will now highlight some of the features offered by the Logic Explorer . We will begin by summarizing how to access these features by describing its user interface and options in Design Environment. We will then move on to describe these features as categorized by Design Synthesis and Optimization capabilities, Design Integration, and Command and Control File creation.

(A) Design Environment

The Logic Explorer main window has a simple, user-friendly graphical user interface. It consists of a main menu where the user can specify Input and Output Filenames, Source and Target Technology, and Entry Format. In addition, four submenus are available to set Synthesis and Runtime options:

(1) Input Options

- Lower-level design filename(s) for hierarchical designs
- VHDL style (IEEE 1164, 1076, or View-Logic)
- State machine encoding style (binary, gray, random, and one-hot)
- Module generation library names

(2) Output Options

- Target device
- Target package

(3) Synthesis Options

- Optimization constraints (specific area and/or delay)
- Number of optimization passes to be performed
- Derating factors for delay calculations
- Retarget switches for remapping to different technologies
- Command and additional source library filename(s)
- Report file options

(4) Runtime Options

- Global optimization goal (area or delay)
- Technology mapping effort level

Most of these options are self-explanatory. Some of them will be explained in further detail below.

(B) Design Optimization

The Logic Explorer allows users to exert control over the synthesis of their designs by providing ample features in the following areas:

(1) Global and Local Optimization

Global optimization refers to the synthesis of a design that has multiple building blocks as a whole, while local optimization refers to the synthesis of a design's individual building block before combining them together.

Global optimization can be accessed through the Synthesis and Runtime Options menus in the Logic Explorer main window. Optimization is applied to the design overall, including all lower-level modules.

Local optimization trades off control over global optimization with overall design results. The flexibility of being able to optimize each of a design's lower-level building block locally before linking them together can be achieved by using the Exemplar Logic Integration Language (EIL).

EIL is a simple language that describes a netlist of instances of building blocks, each of which can be written in any input format, and can be optimized using different constraints. EIL allows the user to specify the I/O interface of the top-level design and the interconnection of instances which make up the design.

(2) Constraint-Driven Optimization

By setting optimization constraints, the user can specify different design requirements which will affect the synthesis outcome. The Logic Explorer will try its best to synthesis and optimize in such a way that all constraints are met. Design constraints can be applied to an overall design or to individual signals:

- Area
- Delay
- Max Fan-in (for MAX340 only)
- Max PT (for MAX340 only)
- Max Load (for pASIC380 only)

All of the above constraints can be specified as command line options or placed in the control file (see Command and Control File below). Area and delay constraints can be accessed through the Synthesis and Runtime Options menus in the Logic Explorer main window. The user can either let the Logic Explorer synthesize to the best area or delay that it can achieve (Runtime Options), or set specific constraints by specifying values for the maximum area and/or the delay allowed (Synthesis Options).

(3) Technology Mapping

For technology-independent design entry (e.g., VHDL), the Logic Explorer will first translate the design into their internal technology-independent Logic Data Structures. Architecture-specific logic optimization will then begin. When this is done, the design will be mapped into gates that are available from the target technology library. Multiple passes of this technology-mapping step can be run (using different strategies) to achieve results that will best fulfill the design constraints set by the user.

For technology-dependent design entry (e.g., netlist), the source technology library also needs to be specified. The technology mapper will then perform device-specific transformations to map gates from the source technology to the target technology. Some of these retargeting switches, like the mapping of internal three-states into combinatorial logic, can be accessed through the Synthesis Options menu.

(4) I/O Mapping

Automatic synthesis of I/O pads is part of the Logic Explorer's default mode. Howev-

er, the user can also assign pads manually to have better control over pad assignments. Manual pad assignments can be done either through component instantiations in the design input files (e.g., in VHDL or Verilog), or through the use of the control file (see Command and Control File below).

Pin assignments can also be done through the control file.

(C) Design Integration with EIL

The Exemplar Logic Integration Language (EIL) can also be used to link multiple design files in different entry formats (e.g., mixing HDLs and netlists) into one large design.

EIL is a simple language that essentially describes a netlist of instances of building blocks each of which can be written in any input format and can be optimized using different constraints. EIL allows the user to specify the I/O interface of the top-level design and the interconnection of instances which make up the design.

(D) Command and Control File

Command files can be used to store command line options that the user will want to reuse. Any Logic Explorer command line options (e.g., Input and Output filenames, target and source technology libraries, etc.) can be saved in a command file. The user can specify any command file to be reused in subsequent runs of the Logic Explorer. A Command File Editor is available from the File menu in the Logic Explorer main window.

Control files can be used to store design constraints, manual pad assignments, and pin assignments for a specific design. Any control file can be specified to be used with a specific design by specifying it's name in the Control File menu in the Logic Explorer main window.

Please refer to the Galileo Reference Manual for specific formats of the command and control file. However, here is a summary of some useful command file (*Table 2)* and control file options *(Table 3):*

Table 2. Useful Command File Options

Function	Command File Option
Optimize for Area	−area
Optimize for Delay	−delay
Design Constraint for Max Area	−maxarea=<n>
Design Constraint for Max Delay	−maxdelay=<n>
Design Constraint for Max Fan-in	−max_fanin=<n>
Design Constraint for Max PT	−max_pt=<n>
Design Constraint for Max Load	−maxload=<load>
Control File Name	−control=<name>
FSM Encoding Style	−encoding=<encoding style>
Package Type	−package=<name>
Part Name	−part=<part_number>
Source Library Name	−source=<library_name>
Target Library Name	−target=<library_name>

Table 3. Useful Control File Options

Function	Control File Option
Design Constraint for Max. Load	MAX LOAD
Signal Name Preservation	PRESERVE SIGNAL
Manual Pad Assignment	PAD or GATE
Pin Assignment	SET...PIN NUMBER

SYNOPSYS − DESIGN COMPILER

Like the Logic Explorer, the Design Compiler from Synopsys also aims to provide powerful synthesis through the support of a variety of behavioral HDLs, as well as some netlist support for design entry formats. It is also tightly integrated with the *Warp* design tool to provide a seamless design pathway for designing with UltraLogic devices.

Design Entry Formats

Hardware Description Language (HDL) support for the Design Compiler is as follows:

- VHDL (IEEE 1164 & 1076)
- Verilog

Netlist support is as follows:

- Berkeley PLA
- EDIF 2 0 0

UltraLogic Device Support

The Design Compiler currently supports pASIC FPGAs.

Software Requirements

To design with the pASIC380 devices, *Warp2+* is required as a minimum.

Design Flow and Integration with *Warp*

The Design Compiler-*Warp* design flow includes design entry, synthesis and optimization, place & route (for pASIC380), simulation, and programming (see *Figure 2*). Designs in HDL and netlists can be entered using any text editor, which then goes through the Design Compiler for synthesis and optimization. The output from the Design Compiler then goes into *Warp* for place & route, and programming files and/or timing models are generated by *Warp* for device simulation and programming.

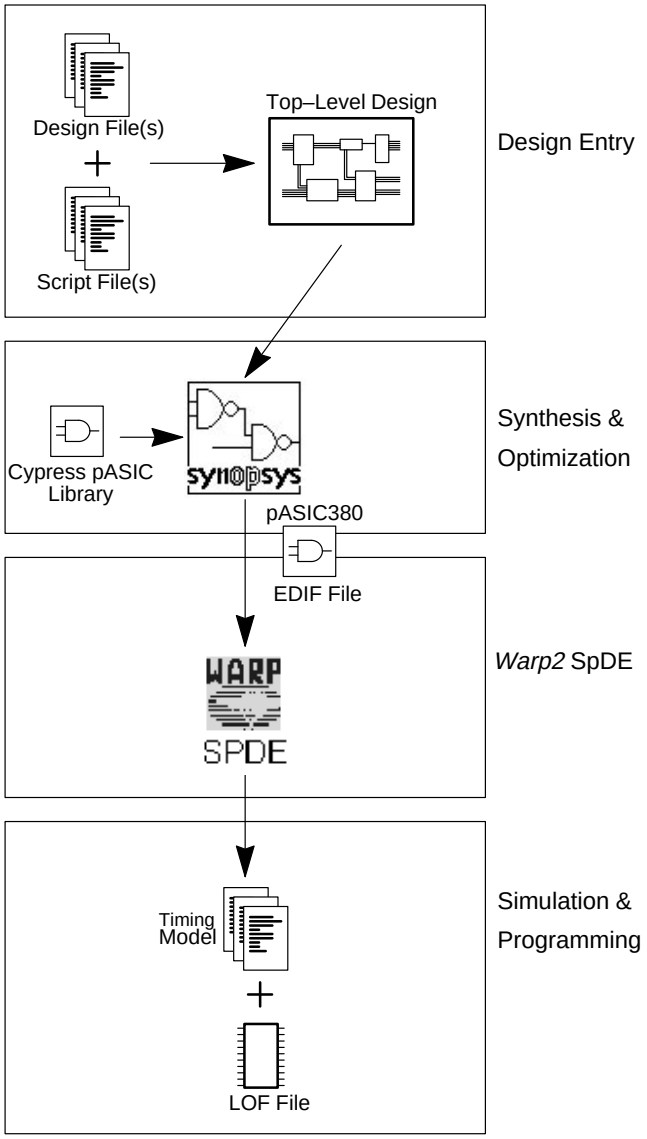

Figure 4. Design Compiler-*Warp* Design Flow

Details about each of the design stages are described below:

(A) Design Entry

Designs (in HDL or netlist) are entered using any text editor and saved as ASCII text files. Hierarchical designs can be described across multiple design files.

Optional Design Compiler Shell Script files can be included to specify synthesis commands as well as design-specific parameters such as input and output filenames, source and target libraries, design constraints, I/O pad mappings, and pin assignments.

(B) Design Optimization and Synthesis using the Design Compiler

The next step is to synthesize and optimize the design(s) using the Design Compiler.

The Design Compiler's graphical interface is called the Design Analyzer. Its main window allows the user to specify design-specific information like input and output filenames, source and target technology, and entry format, as well as synthesis and design constraint options (for details refer to Design Environment below). Upon completion of synthesis and optimization, the resulting netlist will be displayed in graphical form in the Design Analyzer. Users can then push in and out of design hierarchies, examine the timing of critical nets, and generate report files.

(C) Device Place & Route using *Warp*

After synthesis and optimization, the results are generated by *Warp* for place & route. The output format for interfacing with *Warp* is EDIF 2 0 0.

For pASIC380 devices, an EDIF file (containing pASIC primitives will be generated by the Design Compiler which will be taken by the *Warp* SpDE place & route tool as input to perform place & route and timing analysis. A LOF file will be generated for device programming and timing models will be generated for device simulation.

Design Synthesis and Optimization Capabilities

We will now highlight some of the features offered by the Design Compiler. We will begin by summarizing how to access these features by describing its user interface and options in Design Environment. We will then move on to describe these features as categorized by Design Synthesis and Optimization capabilities, Design Integration, and DC Shell Script creation.

(A) Design Environment

The Design Analyzer is a graphical user interface that consists of pull-down menus where the user can specify input and output filenames, source and target libraries, design constraints, I/O pad mappings, and pin assignments. It is also a hierarchical netlist viewer that allows users to view the design in terms of functional blocks before synthesis and in mapped gates after technology mapping. The user can also interactively examine the timing of the critical nets.

The user can open a Command Window to enter Design Compiler commands interactively in command line form. Options that are available from the pull-down menus have an equivalent command line format.

The user can also execute commands in batch form by using DC Shell Scripts. Please refer to the section on DC Shell Scripts for further details.

Some useful options that are available from the pull-down menus are summarized below:

(B) Design Synthesis and Optimization

Synthesis in the Design Compiler involves the translation of an HDL design into a Synopsys built-in generic logic representation and the op-

timization and mapping of that representation using the Cypress pASIC library elements.

As in the Logic Explorer, various synthesis and optimization features are available to the user to better control the results of synthesis.

(1) Constraint-Driven Optimization

Users can control the synthesis outcome by setting optimization constraints on individual signals, on modules under any level of the design hierarchy, or on the overall design. The Design Compiler will try its best to synthesize and optimize in such a way so that all constraints are met. Design constraints that are available to the user for pASIC380 devices are:

- Area
- Delay
- Fanout

All the above constraints can be specified graphically from the Design Analyzer or placed in the DC shell script (see DC Shell Script below and Appendix D). For example, an adder that is constrained by area will be synthesized using a ripple-carry algorithm, while one that is constrained by speed will be synthesized using a carry-lookahead algorithm.

(2) FSM Extraction

Designs that include descriptions of finite state machines (FSMs) can be extracted into a State Table format. Once extracted into this format, the Design Compiler can perform the following FSM optimization techniques on the extracted design(s):

- Automatic state assignments, or completion of partial assignments,
- Optimization of the FSM(s) for Area or Delay,
- Optimization of "don't care" sets,
- Removal of redundant states, and
- Allows users to explore alternative FSM implementations with different state-encoding schemes (e.g., sequential, one-hot, gray, or manual).

For details on how to extract and optimize FSMs refer to Design Examples and Appendix E.

(3) Synthetic Cells

Arithmetic or relational operators are inferred from HDL descriptions as individual logic blocks to allow for more specific and optimal synthesis for these modules. For example, in the following VHDL code fragment:

```
ADD8 <= A8 + B8;

SIX  <= '1' when (ADD8 >
        "00000110") else '0';
```

The '+' sign in the first statement and the '>' sign in the second one will be inferred as an adder and a comparator respectively. These modules are referred to as synthetic cells, and will be synthesized according to design constraints that are set on them by the user (if any).

(4) Resource Sharing

Resource sharing is the using of a single hardware resource for multiple operations. In the following VHDL code fragment:

```
Z <= A + B when X else C + D;
```

Instead of inferring two synthetic adder cells due to two occurrences of the '+' operator, a single synthetic adder cell will be inferred, with the inputs A and C passing through one

two-to-one multiplexer, and inputs B and D passing through another one. In this way, additional logic for generating an extra adder is avoided. This is made possible because depending on the condition of 'X', either A and B or C and D uses the adder exclusively. And hence the resource (adder) can be shared. Other arithmetic and relational operators can be shared in the same fashion.

Resources are automatically shared during design compilation (and can be overridden) and are constraint-driven.

(C) Design Hierarchy

Designs with multiple levels of hierarchy can be viewed, manipulated, and synthesized using the Design Analyzer. Users can select signal paths or logic modules and set constraints on them, or push into lower hierarchical levels to view their gate-level implementations.

In addition, users can manipulate hierarchical designs using the following commands:

(1) Uniquify:

Each instance of the same cell (e.g., an 8-bit adder) is set to be unique (not referenced) so that each instance can be optimized individually through different constraints.

(2) Set Don't Touch:

Lower level modules specified with the set_dont_touch attribute will not be optimized or recompiled.

(3) Ungroup:

Hierarchical designs can be ungrouped or flattened into one single level before compilation and synthesis.

(D) DC Shell Script

DC shell scripts can be specified when invoking the Design Analyzer to perform design compilation and synthesis in batch mode. Any command that are accessible from the Design Analyzer's graphical menus has a command line equivalent that can be used from with a DC shell script. Shell scripts allows users to re-use part or all of the commands that make up the compilation and synthesis procedures.