Xilinx/Cadence PCB Guide

UG629 (v 13.2) July 6, 2011

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	
03/01/2011	13.1	Added this table. Structural changes to match current style guide. No content changes.
07/06/2011	13.2 and later	Updates to the Additional Resources chapter. Recomposed to insure proper formatting of code snippets with updated stylesheet.

Table of Contents

XILINX®

Chapter 1

Introduction

This guide contains information for FPGA designers and Printed Circuit Board (PCB) engineers about processes and mechanisms available within the Xilinx® ISE® Design Suite and various Cadence tools to efficiently implement an FPGA on a PCB.

The first section of the guide covers the PCB and FPGA designs flows, highlighting steps where data is exchanged between these two software environments. Then for each identified step the guide details processes, files, and options available to perform the identified task.

With Cadence's broad software package availability, this document cannot cover all of the features available for implementing a printed circuit board with FPGAs. For details about these tools, refer to the Cadence documentation available at: http://www.cadence.com/support/pages/sourcelink.aspx.

If you use software tools from multiple vendors for your PCB design flow, such as Cadence OrCAD for schematic capture with Mentor Graphics PADs for PCB layout, refer to vendor specific documentation. For Mentor Graphics tools, refer to the *Xilinx/Mentor Graphics PCB guide*.

XILINX®

Chapter 2

Implementing a Xilinx FPGA on a Printed Circuit Board

In recent years, the design of FPGAs and printed circuit boards (PCBs) have become increasingly parallelized as opposed to the traditional sequential model. This is mostly due to market pressure which demands a fast design cycle and rapid adaptability to specification changes. In the past the FPGA was typically designed before the board or was added to an already designed board to perform some glue logic function, voltage or protocol conversion. Often the same PCB engineers were doing both the FPGA and PCB designs.

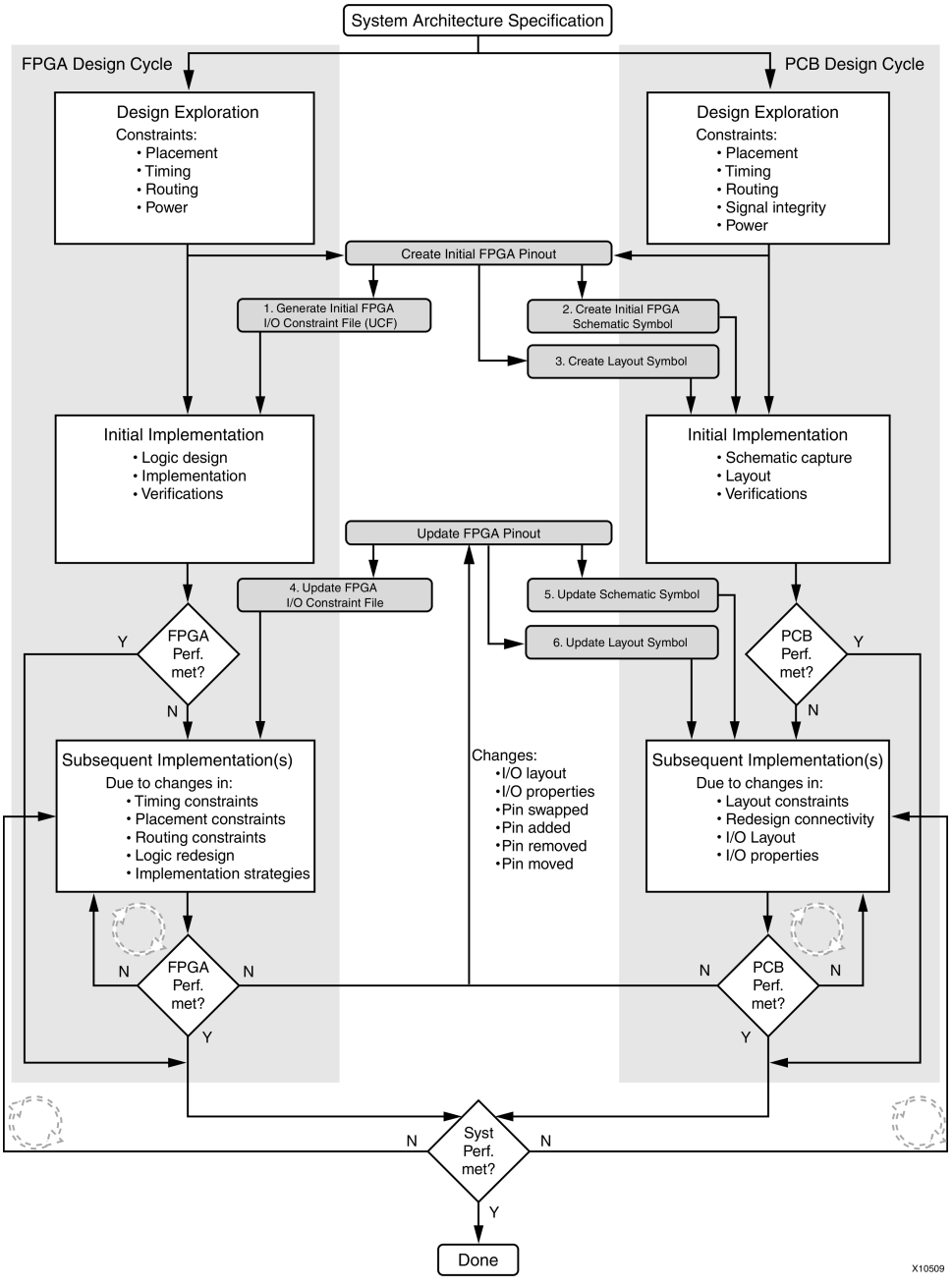
Today, with their increasing internal and I/O capabilities, FPGAs can take on more core features of an application which require longer development time and greater expertise and manpower. On the board side, tight form factor, signal integrity, and electromagnetic regulations require sharp skills and dedicated personnel. Therefore, FPGA and PCB are now two separate design teams working in different environments and often physically distant.

Paradoxically, pressures in terms of time and adaptability to market requires many more interactions between these design environments so that functionality, performance and cost objective are delivered on time. In practice, this translates into back and forth data exchanges throughout the design process between design teams to update the board symbols and FPGA constraints.

Design Flow

The following System Development Cycle illustration shows a typical flow in the PCB and FPGA development cycle (white boxes). It also highlights the steps that require communication between FPGA and PCB software tools (grey boxes). Common Tasks details the mechanisms and processes available to perform each of these data exchanges.

System Design Cycle with FPGA and PCB Databases Synchronization Steps Highlighted



PCB design requires two main tools; a schematic capture tool and a layout tool. These tools are described in the following sections.

Schematic Capture Tool in the FPGA Design Flow

The schematic capture tool enables designers to create a graphical representation of connections between components on the PCB. This data helps anyone involved in the project to understand how components on this board are connected between themselves and with the outside world. The layout designer also uses this information to physically place and route all signals on the PCB.

Tips: Since an FPGA is a programmable component, its requirements on the PCB are unique to your application. Xilinx recommends that you add within all the schematic the specific components necessary for both the programming and the behavior of this device in your particular application.

- Add decoupling capacitors. Since FPGAs can be programmed to perform in a wide range of applications which translate into a wide range of decoupling needs, it is not practical for Xilinx to embed decoupling networks inside the device. The schematic engineer often adds all the decoupling network details on the schematic so as to let the PCB designer place these components in the vicinity of the FPGA package.
- Add other external components necessary to enable specific FPGA features. For instance the schematic designer needs to attach digitally controlled impedance (DCI) calibration resistors to VRN and VRP pins when I/Os on the device have the on-chip termination option enabled.
- Add debug, probe, and test points.
- Add pin swapping information. It is often useful at this point to define which pins can be swapped without violating FPGA pinout rules. This is very useful information for the PCB designer as it provides flexibility when trying to minimize wire crossover, congestion, and signal integrity in placing and routing signals.

PCB Layout Tool in the FPGA Design Flow

The PCB layout tool reads the component and connectivity description in the schematic capture tool and physically places and routes these components on the PCB. The output is a set of masks and geometries that allow manufacture of the PCB.

Tips In order to efficiently place and route a programmable device, the PCB designer needs the following information

- Board physical dimensions. Dimension of the board, mandatory position of connectors, etc.
- Stackup dimensions. Number and orientation of signal layers, number and location of power and ground planes, board material, traces properties, etc.
- Components footprint. Exact dimensions of each component package.
- Components landing pattern. Shape of the junction area between the component and the board including manufacturing tolerances.
- Board environment properties. Available space around the PCB (air flow, obstacle, vibrations, cooling system, access to power and connectors, etc.)

Cadence PCB Design Tools

Below is a brief description of the Cadence tool chains and capabilities available for designing printed circuit boards. Please refer to the tool's documentation for further details.

Cadence OrCAD Series

This tool set is typically appropriate for low to medium complexity boards and single site design teams. This document refers to Capture as the schematic capture tool and Layout as the layout tool. Depending on your exact software configuration, your Cadence documentation might refer to slightly different names, however all features and methodologies presented here are available to you.

Cadence Allegro Series

This tool set is typically appropriate for medium to high complexity designs and for single to many site design teams. This document refers to Design Entry as the schematic capture tool and PCB Editor as the layout tool. Depending on your exact software configuration, your Cadence documentation might refer to slightly different names, however all features and methodologies presented here are available to you.

Multi-Vendor Flow

You can also use multiple vendor software tools for your PCB design. For instance, Cadence OrCAD Capture can be used for schematic capture with Mentor Graphics PADs or Expedition series for PCB layout. Mentor Graphics users should refer to the *Xilinx/Mentor Graphics PCB Design Guide* for information about Mentor Graphics tools. When using other PCB software packages please refer to your vendor's documentation.

XILINX®

Chapter 3

Common Tasks

The following section covers the process, available software features, and file manipulations needed to accomplish each task in the PCB or FPGA design flow related to the FPGA pinout. Each of these tasks is illustrated in the flow chart in chapter 1. Tasks associated exclusively with either PCB or FPGA design flow are represented as white boxes. Tasks common to both design flows are represented as grey shaded boxes.

Create an Initial FPGA Pinout

After taking into account the system specification describing the different parts and connectors on the board along with communication channels linking them together, the next step for the designer is to infer a device and package that can accommodate these communication channels then expand, classify, and assign each signal to a particular pin on the chosen FPGA package. This task requires FPGA architecture knowledge allowing PCB designers to find an optimal pinout for the PCB design. Therefore, this task is typically done by an FPGA engineer.

Necessary Information

I/O placement requirements may come from a variety of sources. To save time Xilinx recommends that you draw the list of I/Os and learn about the FPGA architecture before starting I/O placement.

Tip FPGA requirements can be found in the device data sheets and user guides.

- System requirements:
  - Identify properties and number of each I/O standards required (including direction, input and output voltages, drive strength, slew rate, data rate, etc.).
  - Identify differential pairs.
  - Identify global/regional clock signal with their associated data signals.
  - Identify Multi Gigabit Serial Transceivers.
  - List I/O location constraints imposed by predefined IP Blocks (third party IP or IP generated by the CORE Generator™ tool, the Architecture Wizard, the Memory Interface Generator or the Embedded Development Kit (EDK)).

- FPGA Requirements (refer to the device user guides for more information):
  - Acquire device knowledge including:
    - I/O compatibility or I/O banking requirements.
    - Device package properties such as I/O count for the entire device and per bank and clock region, I/O data rate and signaling capabilities such as single ended/differential or single/dual data rate support, etc.
    - I/O access to internal resources. Resources such as clock buffer, RAM, serializer/deserializer, etc.
    - Device clocking capabilities such as internal clock management resources, I/O with direct access to clock networks, etc.
    - Device Simultaneous Switching Output specifications.
    - Package trace delays.
  - Reserve and Prohibit usage of special purpose pins:
    - Prohibit package pins. Because of die or package migration, or future design growth constraints.
    - Reserve configuration pins.
    - Reserve JTAG pins.
    - Reserve DCI pins.
- PCB Requirements:
  - Formulate package escape strategies. Determine board number and direction of layers, pin spacing, etc.
  - Signal integrity. Estimate amplitude and timing margins for each signal type.
  - Air flow. Ensure work area has sufficient airflow.
  - Placement and orientation of neighboring parts which could constrain the FPGA placement or access of PCB signals to the FPGA.
  - Connectivity to other devices. Other device may impact the optimal FPGA I/O design.

Process

Depending on your preferences and company policies, there are different mechanisms you can use to assign pins on an FPGA. The following methods are the most common.

Create a Pinout in a Spreadsheet Environment

To create a pinout in a spreadsheet environment, create two spreadsheets, the first with your design I/O requirements (signal name, I/O standards, direction, etc.), and a second with properties for each pin in the package (pin number, I/O bank number, pin name, etc.). Then going down the list of your design signals, filter out and sort package pins of the second spreadsheet to determine compatible device I/Os. Finally, go back to your original design I/O spreadsheet and assign pin numbers (or I/O bank numbers) to your signal names. Once this is done, you will export this pinout to the schematic and FPGA tools as detailed in the next paragraphs.

This method is most often used by advanced users with extensive knowledge of the FPGA's capabilities. Since there is no DRC done by any tool during this process, the resulting I/O assignment could fail during FPGA or board implementation.

Tip You can easily create the package property spreadsheet with one of the following:

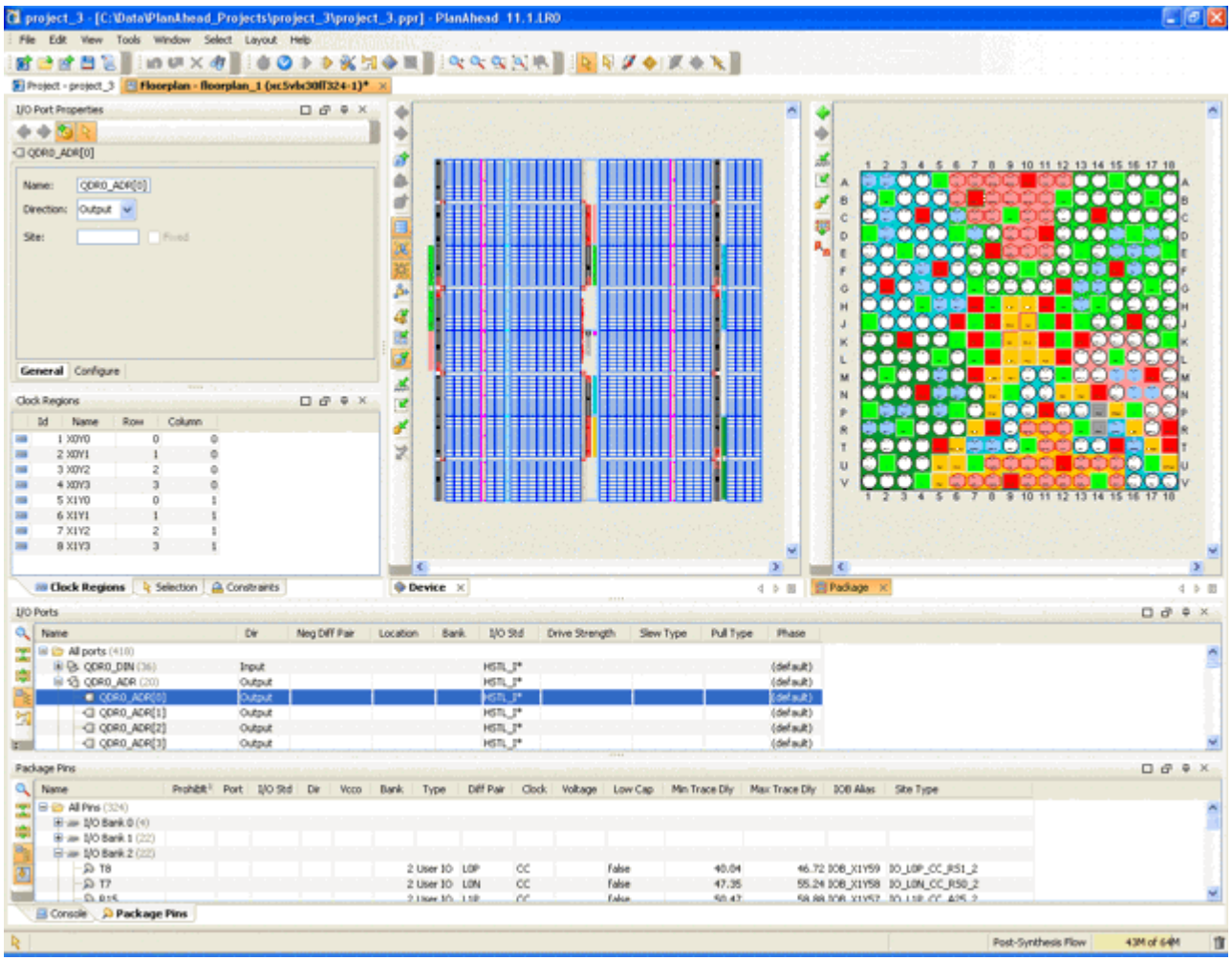
- Partgen utility. (For example, **`partgen -v xc5vlx30ff676`**)
- Download the package file for your target:
  - Virtex® families: Refer to Answer Record 20578 at: http://www.xilinx.com/support/answers/20578.htm
  - Spartan® families: Refer to Answer Record 21035 at: http://www.xilinx.com/support/answers/21035.htm

Create a Pinout using I/O Planner

The I/O Planner graphical tool is integrated into the PlanAhead™ software. As illustrated below, the I/O Planner environment consist of a split workspace showing both the device I/O capabilities and a view of your design I/O settings.

- The device I/O capabilities presented are the different package and I/O properties (standards, voltage banks, differential pins, dedicated pins, clocking resources, package trace delay, prohibits, etc.) supported in the targeted device.
- The design I/O configuration summarizes and allows you to enter details for all of your design I/O signals (voltage, direction, number of pins, on-chip termination, etc.).

Pinout Creation Using I/O Planner



You may also import an I/O port list from HDL, CSV or UCF format into I/O Planner.

These views make it simple to identify package pins that support your signal properties. In addition to the interactive mode where you can drag and drop sets of signals into groups of package pins, you can also enable I/O Planner's I/O placer engine to automate I/O placement.

To check I/O placement against the FPGA pin assignment rules, I/O Planner can prevent incorrect assignment on-the-fly or you can run a set of design rules checks. You can also run a Weighted Average Simultaneous Switching Output analysis to verify the pinout is correct.

Finally you can export this pinout to a UCF, CSV, Verilog or VHDL file and read this information into your schematic or spreadsheet entry tool.

For more information on the I/O Planner tool, see the I/O planning section of the *PlanAhead User Guide*.

Create an Initial FPGA I/O User Constraint File (UCF)

In the ISE® Design Suite, I/O constraints can be entered in a unique User Constraints File (UCF) attached to the design project. They can also be attached to the HDL (Verilog or VHDL) source code, the synthesis constraint file (XCF) or embedded in the logic netlist (NGC, EDF or EDN files, and associated NCF files). The problem with entering I/O properties and location constraints in multiple files is that maintenance, portability and updates to the design become much more complex. Xilinx recommends that you specify the maximum number of I/O related constraint within a single UCF file.

Create a UCF File with a Text Editor

You can create a UCF file by simply typing the constraints into a text editor. When creating a UCF file in this way, please refer to the *Constraints Guide* for the syntax of all I/O related constraints. This method is most often used by companies that have developed their own scripts that read in a spreadsheet and convert the data into UCF syntax.

If you already have your I/O constraints defined in a spreadsheet format, you can use the import function in I/O Planner. In this case, the tool parses your spreadsheet and converts recognized data into UCF syntax. At a minimum, the Signal Name field must be present.

Data is recognized for all column headers that match the following:

- Signal Name (Mandatory)
- IO Bank
- Pin Number
- IOB Alias
- Site Type
- Min/Max Trace Delay
- Prohibit
- Interface
- Direction
- DiffPair Type
- DiffPair Signal
- IO Standard
- Drive
- Slew Rate

For more information on the I/O Planner tool, see the I/O planning section of the *PlanAhead User Guide*.

Create a UCF with I/O Planner

Once the pinout is defined, use the "Export I/O ports" command in I/O Planner to generate the I/O assignment in UCF format.

Create a UCF with the PIN2UCF Utility

With the other methods for creating a pinout described in this section you can either create a complete pin assignment or create a partial one (assign a signal names to sets of pin numbers) and let the back-end place and route tool perform the actual assignment within this specified set. In the second case, you can assign a signal or a set of signals to a pin, a set of pins, a bank or a set of banks and thus give the implementation tool the task of assigning an exact package pin number to each individual user I/O.

If after the implementation you are satisfied with this pinout and want to preserve it for future implementation runs then you can do one of the following:

- In Project Navigator go to the **Process** window and expand the **Implement Design** process. Next, expand the **Place & Route** process, and double-click **Back-Annotate Pin Locations**. A UCF file is created and has all your I/O signals locked to a specific package pin number.
- Use the PIN2UCF utility to lock a particular pinout for the next implementation iteration. To use the PIN2UCF utility, type the following at the command lint.

  `pin2ucf` *ncd_file_name* **`|`** *gyd_file_name* **`-o`** *ucf_file_name*

Create a Schematic Symbol (Schematic Shape and Content)

An FPGA schematic symbol is used to describe the electrical connectivity between each device and its environment (other parts, connectors, etc.). Unlike most other components, FPGA symbols are not likely to be available in a predefined library. One purpose of a library is to allow reuse of its elements across different applications. FPGAs, by definition are programmable and application specific so no two designs will have the same connectivity (signal names and pinout) with the outside world. Therefore few symbol properties can be reused from one project to another.

Necessary Information

Depending on your Cadence tool flow, an FPGA may be represented either as a graphical symbol to be placed on a schematic or as a spreadsheet. In either case, global properties area attached to the resulting representation. As a result, each spreadsheet or symbol lists available I/Os and may also tag I/Os with its additional properties.

Typically FPGAs have more I/Os than can be represented on a single schematic sheet, therefore FPGA symbols are often split into multiple fractures and hierarchy levels to simplify readability.Each company and sometimes each engineer has their own process and opinion as to what an FPGA schematic symbol should look like or contain. Below is the minimum set of data required for an FPGA schematic signal plus some additional information which could make the FPGA schematic symbols more useful.

- **Graphical symbol -** Such that the component can be placed on the schematic. Choose a shape that allows placement of a fair number of I/Os. Some engineers use a different shape depending on the type of interface.
- **Device name -** Component ASCII name that makes it easy for someone reading the schematic to know what this component is.
- **Reference designator -** Unique and short identifier for each component on the schematic.
- **I/O name (or pin name) -** A separate name for each I/O on the symbol. Xilinx recommends using the same name as in the top level HDL description. Since this is the name FPGA designers will be familiar with, this makes for easier communications between Schematic and FPGA designers. In addition each I/O can be tagged with additional visible or non-visible properties. Therefore we recommend adding data sheet, pin name, and I/O direction.
- **Non-user I/Os -** Ensure that all pins available on the package have an entry in the schematic symbol. Some of them may be visible such as unused or logistical I/O (DCI reference). Others may be hidden as they are not of interest for describing the board functionality, such as power, ground, no connect, or reserved I/Os. Having all I/Os present on the schematic symbol will be appreciated because it helps with mapping the schematic symbol to the layout symbol. It will also facilitate the maintenance of symbols when pin swaps occur.
- **Pin Number -** Locates the I/O on the package ball array.

Process

Depending on the engineers' preferences and company policies, there are several mechanisms for creating schematic symbols for FPGAs. Most engineers use one of the three methods described below.

The basic symbol information such as pin name and pin number may be gathered from multiple sources. For instance, it can be copied and pasted from the device user guide. This is not recommended since it is easy to mistakenly copy the wrong information or omit some pins. A more common way if you have a placed and routed FPGA design, is to generate the I/O file, called a PAD file, from within the ISE® Design Suite. This PAD file contains all the specific I/O information about your particular design. A PAD file requires an implemented FPGA design which you may not already have, especially if the PCB and FPGA design processes are highly parallel. In such cases you can use the CSV output from an I/O assignment done in I/O Planner. If this information is not available, you can create a generic symbol using the Partgen utility. Partgen creates a PKG file containing all I/O names and numbers available on the device. This is essentially the information you can find the within the device user guide with the benefit of being rapidly converted into a spreadsheet.

Tips:

- To create a PAD file with ReportGen, type the following:

  ```
  reportgen ncd_filename –pad
  ```

- To get details on all package pins, use the partgen utility:

  ```
  partgen -v target_device
  ```

  For example:

  ```
  partgen -v xc5vlx30ff676
  ```

Note When importing PAD files into Cadence tools, you might find some parsing errors depending on which version of ISE is combined with which version of Allegro. In such cases please refer to the Cadence support site (http://www.cadence.com/products/pcb) for a PERL script utility that converts the ISE software file into a format directly readable into Allegro Librarian.

Create a Schematic Symbol in a Text Editor

To create a schematic symbol in a text editor please refer to OrCAD or Allegro Design Entry documentation on how to set the location, shape and all other properties of the symbol.

This method is most often used by companies that have developed their own scripts that read in a spreadsheet, HDL, or the post implementation I/O report (PAD file) and automatically convert the data onto a symbol.

Create a Schematic Symbol with OrCAD Capture

Within Capture, select the **Generate Part** option and follow the guided process which will successively let you import a spreadsheet based I/O assignment then let you add symbol graphics and pin properties.

Create a Schematic Symbol with Allegro Design Entry

Within PCB Librarian, directly read in the ISE software PAD file or a I/O Planner CSV spreadsheet and follow the guide process to add pin properties. For instance, Power pins and unconnected pins can be tagged for special handling.

Tip You can find documentation for displaying "intelligent" symbol graphics to indicate pin type and other useful symbol properties on the IEEE website at http://www.ieee.org.

Create a Layout Symbol

The layout symbol contains the device physical dimensions such that copper traces can be accurately routed to and from the FPGA pins or balls. The layout symbols are not design specific and can therefore be stored in a predefined library and shared among many FPGA designs with the same package.

Allegro has dialog boxes that let you enter package physical dimensions such as ball location and dimension.

Tips

- Use the Mechanical Drawing section in the Packaging and Pinout Specification of the specific device user guide to capture this information.
- Go to www.xilinx.com and click **Documentation** to find the user guide for your device.

Map Schematic Symbols to the Layout Symbol

This is the process of mapping the pin numbers on the schematic symbols to the pin numbers on the layout symbol. Whenever possible, also enter pin swapability information during this step. Check the tool's documentation for further information.

Update ISE Software Files with Pinout Changes Made in the Schematic Tool

There are several occasions throughout the design process where pinout changes made in the schematic tool must be propagated to the FPGA user constraint file (UCF). For instance, the board design may have started before the FPGA internal logic. Therefore, pins may have been added, removed, renamed, or relocated. The schematic engineer may also discover improperly assigned pins or that the system specifications have changed requiring more, fewer or different I/O properties.

The ISE software user constraint file (UCF) must be kept in sync with the board I/O to avoid a system malfunction.

Process

In either OrCAD Capture or Allegro Design Entry there is no direct link to update the ISE software user constraint file (UCF). Typically the schematic engineer will compile a list of pinout changes in the form of a spreadsheet or meet with the FPGA engineer to ensure those changes are propagated and possible within the FPGA environment.

You can also export the updated pinout in a format which I/O Planner can read (Verilog, VHDL, or CSV) and use I/O Planner to generate the updated UCF file.

Update the PCB Database with Pinout Changes Made in ISE Software

Whether the I/O layout change is due to a timing constraint change, a new piece of logic being added, or a change to existing logic, pinout changes at the FPGA level happen throughout the design cycle. Whenever a pinout change occurs, it is important to propagate this change to the PCB schematic and layout environments to ensure they are not designing with an obsolete I/O assignment and that the board constraints have not been violated. First synchronize the schematic database with the new FPGA I/O layout. Next, synchronize the layout database to the schematic database.

Process

Within either OrCAD or Allegro Design Entry design environment, there are no direct processes to update an FPGA schematic symbol. Typically the FPGA engineer compiles a list of pinout changes made in the form of a spreadsheet or a meets with the schematic engineer. Alternatively the engineer can regenerate an updated Verilog, VHDL or CSV file using I/O Planner and import this new pinout into the schematic tool.

You can also regenerate an updated Verilog, VHDL or CSV file using I/O Planner and import this new pinout into the schematic tool.

Then the schematic engineer will regenerate the symbol (using the Capture Generate Part option or Allegro Librarian) or manually enter the changes.

Tips Whenever a schematic symbol is regenerated there is a risk of this symbol loosing connectivity with the other parts on the schematic. Here are several tips to minimize this possibility.

- Ensure that the pin locations stay the same on the symbol by using the same settings as much as possible.
- Use find and replace features to reconnect the symbols properly.
- Avoid direct connections of the nets on the schematics.

To propagate pin swaps or more generally I/O layout or property changes from the schematic editor to the layout tool, please refer to the Cadence documentation for your layout tool.

Update ISE with Pinout Changes Made in the Layout Tool

At this stage in the design flow there are many ways to take advantage of FPGA I/O programmability by modifying its pinout to optimize the PCB. For instance, there may be a need to reduce wire cross over to be able to complete the PCB routing without requiring additional routing layer. Another common practice is to move or swap pins in order to match or reduce trace length or reduce the number of vias or layer changes due to signal integrity or board timing concerns. This results in a PCB database that is out of sync with both the schematic and FPGA databases.

To synchronize these environments, first propagate the pinout changes to the schematic database then to the FPGA database as previously described in Update ISE Software files with Pinout Changes Made in the Schematic Tool.

Process

This is done in two steps. First propagate pin swaps from the layout tool to the schematic symbol. Refer to Cadence documentation for instructions. The second step is to update the ISE software files by synchronizing the UCF file with the schematic symbol data base as previously described in Update the PCB Database with Pinout Changes Made in the ISE Software.

Appendix

Additional Resources

- **Xilinx Glossary -** http://www.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- **Xilinx Documentation -** http://www.xilinx.com/support/documentation
- **Xilinx Support -** http://www.xilinx.com/support
- **Cadence design tools documentation -** http://www.cadence.com/support/sourcelink.aspx