Xilinx/Mentor Graphics PCB Guide

UG630 (v 11.2) June 24, 2009

XILINX®

Xilinx Trademarks and Copyright Information

Preface

About This Guide

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 Mentor 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 Mentor's broad software package availability, we cannot cover all of the features available to implement a printed circuit board with FPGAs in this document. For full details regarding these tools, please refer to Mentor's documentation available at: http://www.mentor.com/products/pcb-system-design/design-flows/.

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

This guide contains the following chapters:

- Implementing a Xilinx FPGA on a Printed Circuit Board
- Common Tasks

Additional Resources

To find additional documentation, see the Xilinx website 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® website at:

http://www.xilinx.com/support.

To find additional documentation regarding Mentor Graphics PCB design tools, see:

http://supportnet.mentor.com.

Table of Contents

Chapter 1

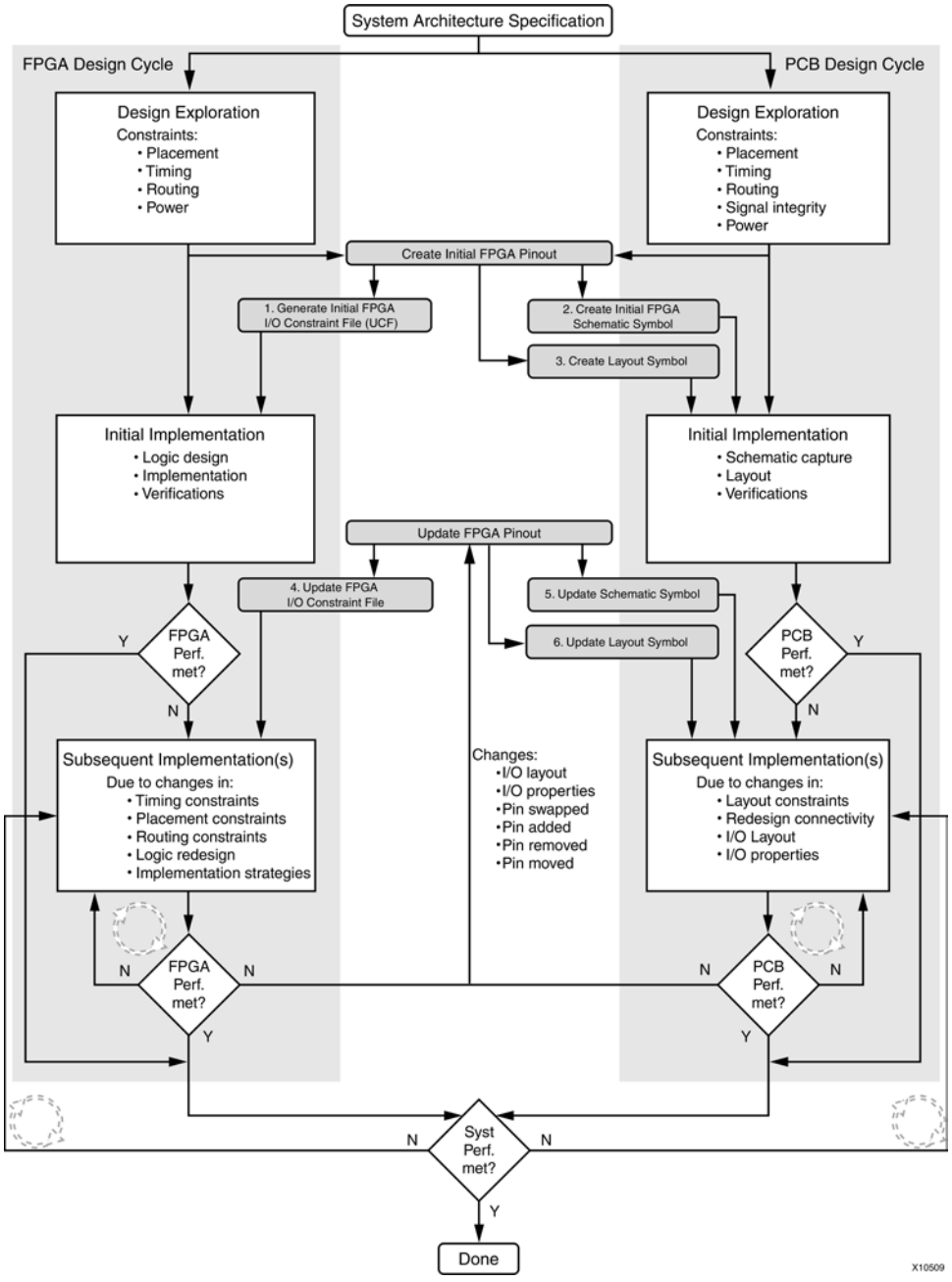
Implementing a Xilinx FPGA on a Printed Circuit Board

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; namely the schematic capture tool and the PCB 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 the schematic all 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 banking, clocking, or SSO rules. This is crucial information for the PCB designer when trying to minimize wire crossover and congestion in placing and routing the FPGA on the board.

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 route these components on the PCB. The output is a set of masks and geometries that allow manufacturing 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.)

Mentor Graphics PCB Design Tools

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

Mentor Graphics PADS Series

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

Mentor Graphics Expedition Enterprise

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

Mentor Graphics I/O Designer Flow

This flow is meant to complement either of the previous flows. I/O Designer adds many features that assist with the creation and maintenance of FPGA symbols. On the FPGA side, I/O Designer understands all Xilinx device families and their I/O properties and restrictions. It can therefore guide you through the pin assignment process along with the definition of pin swappability. On the PCB side, I/O Designer offers wizards and automation to generate and fracture symbols and automatically connect them to the hierarchical schematics while maintaining those connections when pin assignments change. I/O Designer integrates with the Xilinx® ISE® Design Suite and keeps the User Constraints File (UCF) up to date with board schematic symbols. Therefore, consistency can be maintained when changes are made either in the FPGA or PCB domain in order to meet overall system performance.

Multi-Vendor Flow

Some of you may also use multiple vendor software tools for their PCB design. For instance Cadence Concept-HDL or OrCAD can be used for schematic capture with PADs or Expedition series for PCB layout. Cadence users, please refer to the *Xilinx/Cadence PCB Guide* for information about Cadence tools. When using other PCB software packages please refer to your vendor's documentation.

XILINX®

Chapter 2

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 Design Flow. 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. With the help of tools such as I/O Designer this may now also be driven from the PCB software environment allowing PCB designers to find an optimal pinout for the PCB design.

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…).
  - 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 devices 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 three 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:

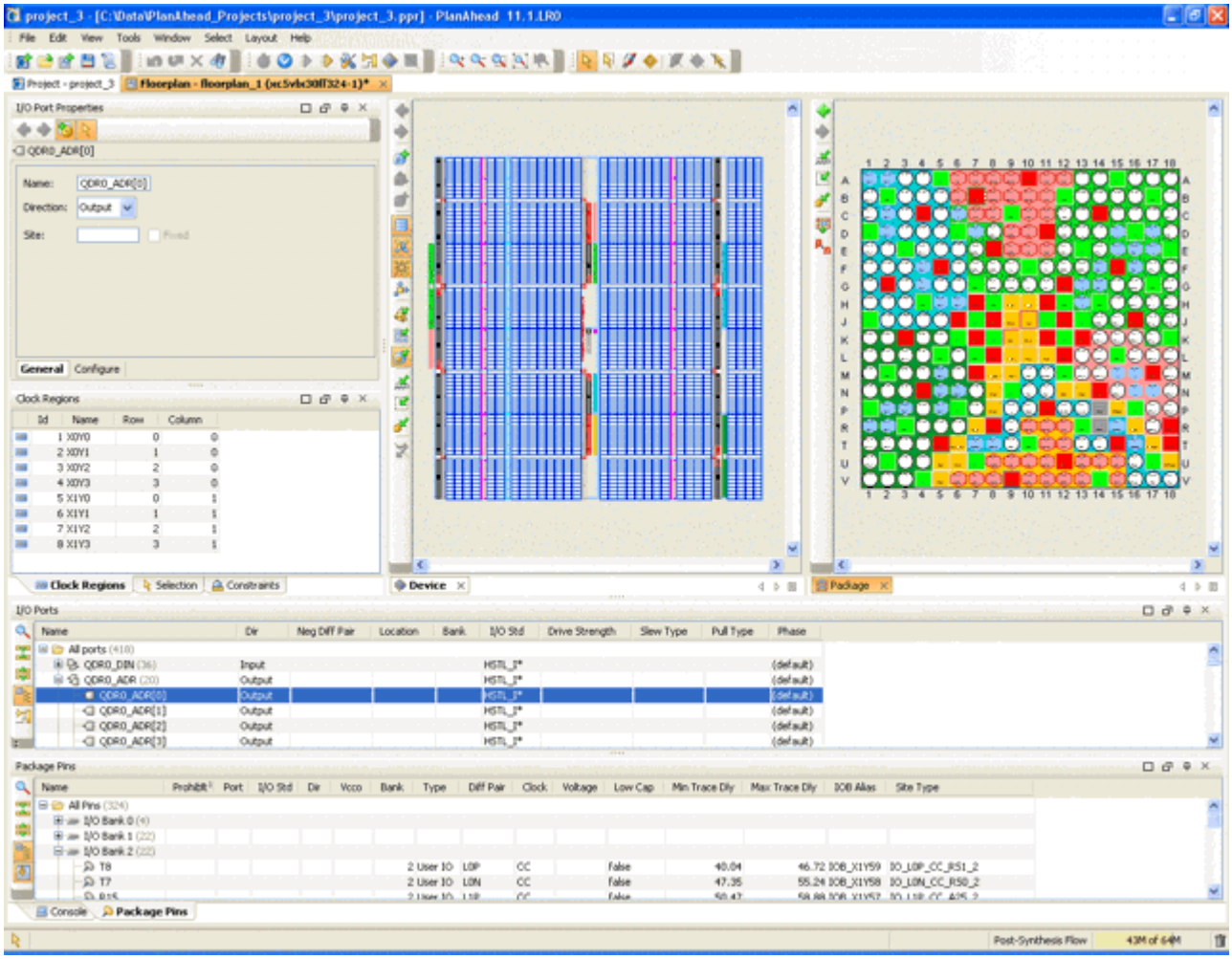
- Partgen utility. (For example, **`partgen -v xc5vlx30ff676`**)
- Download the package file for your target:
  - Virtex® families: Refer to Answer Record 20578 at: http://www.xilinx.com/support/answers/20578.htm
  - Spartan® families: Refer to Answer Record 21035 at: http://www.xilinx.com/support/answers/21035.htm

Create a Pinout using PinAhead

The PinAhead graphical tool is integrated into the PlanAhead™ software. As illustrated below, the PinAhead 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 PinAhead



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

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 the PinAhead I/O placer engine to automate I/O placement.

To check I/O placement against the FPGA pin assignment rules, PinAhead 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 Mentor schematic or spreadsheet entry tool.

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

Create a Pinout in I/O Designer

Using I/O Designer to create an FPGA pinout is similar to using PinAhead. In addition to the features described previously, I/O Designer adds a table view of the package along with advanced filtering options which help in isolating signals or pins of interest. In the physical device view, I/O Designer also shows the PCB floorplan and nets between devices allowing optimization of the connectivity during the I/O assignment process. With regards to the legality of pin assignment done within I/O Designer, Xilinx and Mentor Graphics work closely together to continuously ensure that the ISE Design Suite and I/O Designer have identical sets of DRCs so that an I/O assignment done within I/O Designer will be valid in the ISE Design Suite.

Here too, users do not need to completely define the I/O assignment within I/O Designer. You may read in partial assignments in the form of spreadsheets (CSV, Verilog, VHDL, or FPGAXchange) or Xilinx files (PAD, GYD or UCF) and then make adjustments or perform checks within I/O Designer.

Tip In I/O Designer remember to experiment with **Ctrl**, **Alt,** and **Shift** keys to assign multiple signals to multiple package pins at once.

Generate an Initial FPGA I/O 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 PinAhead. 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 PinAhead tool, see the I/O planning section of the *PlanAhead User Guide*.

Create a UCF with PinAhead

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

Create a UCF with I/O Designer

Within I/O Designer, once you have created a design database, imported or created a pin assignment, and the results are satisfactory, you can export this assignment as a Xilinx user constraint file (UCF) to run the FPGA place and route tool with this I/O placement. To do so, select **PR Constraints File** from the **Generate** drop-down menu. The file is saved at the location specified during the initial project creation.

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 Symbol 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

A schematic symbol is a graphic to which global properties are attached. Each symbol lists and locates I/Os on the graphic and may also tag I/Os with additional properties. Typically FPGAs have more I/Os than can be represented on a single schematic sheet, therefore FPGA symbols are often split into a 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 Xilinx recommends adding data sheet name, 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 easiest way to create and maintain schematic symbols is to use I/O Designer.

Create a Schematic Symbol in a Text Editor

To create a schematic symbol in a text editor please refer to Dx-PADS or DX-Expedition 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 Xilinx PAD files and automatically convert the data onto a symbol.

Create a Schematic Symbol with Dx-PADS or Dx-Expedition

Within DxDesigner use the Symbol Wizard guided process or use the DxBoardLink utility. Using DxBoardLink you can import signal names, pin number, and other properties directly from the Xilinx PAD file for FGPA designs or GYD files for CPLDs. In ISE software, the PAD file is generated at the end of the implementation step. On a placed database, you can also generate the PAD file using the ReportGen command line utility.

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

`reportgen` *ncd_file_name* **`-pad`**

Create a Schematic Symbol with I/O Designer

With I/O Designer you can create an FPGA symbol from a multitude of sources. You can also create the symbol starting from an empty database. Since entering details for each I/O on a large package can be quite cumbersome and error prone I/O Designer lets you import your signal names and already defined I/O properties from the following:

- ISE software files (UCF, PAD, GYD)
- Spreadsheet files (FPGAXchange, CSV)
- HDL files (Verilog, VHDL).

Not all these files are available at any given time in the design flow. Use the Database wizard and then Symbol wizard to guide you through the process of reading in available I/O data, then add and edit symbol properties to finally configure then fracture the symbol.

In short, the Symbol wizard guides you through the creation process by successively prompting you to enter the following:

- Basic information — Symbol, device and package names, etc.
- Symbol fracturing information — Fracturing scheme, max number of pins per page, etc.
- How to represent special pins — Config, JTAG, DCI, Power and Ground, etc.
- Symbol appearance — Shape, pin position, port dimension and spacing, symbol port label and position.

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. PADS and Expedition flows have dialog boxes that allow you to enter package physical dimensions such as the balls location and dimension. 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.

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.

In the Dx-PADS flow enter this data via a set of layout symbol attributes. In Dx-Expedition enter it via the parts database and editor utility. Please refer to the tool's documentation for details on how to establish this mapping.

Note When using I/O Designer, the mapping data including the necessary swapping information that you define within I/O Designer is automatically generated and maintained.

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

PADs or Expedition Series

In either PADS or Expedition there is no direct link to update the ISE constraint file. Typically the schematic engineer compiles a list of pinout changes in the form of a spreadsheet or meets 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 PinAhead can read (Verilog, VHDL, or CSV) and use PinAhead to generate the updated UCF file.

I/O Designer

Whenever I/O Designer detects a pinout modification such that the schematic and the FPGA user constraint file (UCF or CSV) are out of sync, a red flashing light appears in the bottom right-hand corner of the application. Click on it and select which change to propagate in which database to re-synchronize the design environments. This simple process removes many manual edits and email or oral discussions between members of separate teams and avoids cases where information gets lost or distorted.

Update the PCB Database with Pinout Changes Made in the 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.

PADs or Expedition Series

Within either the PADS or Expedition 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 meets with the schematic engineer.

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

Then the schematic engineer regenerates the symbol (using DxBoardLink) or manually enters the changes.

Tips Whenever a schematic symbol is regenerated there is a risk of this symbol loosing connectivity with 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 between schematic nets and FPGA symbols.

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 Mentor Graphics documentation for your layout tool.

I/O Designer

Whenever I/O Designer detects a modification of the FPGA constraint file, a flashing red light appears at the bottom right-hand corner of the application. The process to synchronize the databases becomes similar to the methodology detailed in previous paragraphs in that you simply click on the flashing light and select which change to propagate to which database and then click **Finish** to launch the database synchronization. This automation removes many manual edits, verification and email or oral discussions between members of separate teams. Another great benefit is that the maximum of connectivity in the schematic is preserved. Time consuming, manual and error prone steps are saved with this flow.

Update ISE Software Files 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 an 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.

PADs or Expedition Series

This is done in two steps. First propagate pin swaps from the layout tool to the schematics and symbols. Refer to Mentor Graphics documentation for instructions. The second step is to update ISE by synchronizing the UCF file with the schematic symbol data base as previously described in Update ISE Software Files with Pinout Changes Made in the Schematic Tool.

I/O Designer

When I/O Designer detects a modification in the layout tool database that affects an FPGA pinout, a red flashing light appears in the bottom right-hand corner of the application. The process to synchronize the databases is similar to the previous section in that you simply click on the flashing light and select which change to propagate to which database. Click **Finish** to launch the synchronization of the different design environments. This automation removes many manual edits, verification and email or oral discussions between members of separate teams and prevents cases where information gets lost or distorted, saving time in developing and debugging the system.