XAPP423 (v1.0) October 19, 2004

Creating Pin-Out Prior to Implementation with PACE

Author: Chris Zeh

Summary

This Application Note discusses the procedures and some commonly asked questions related to the creation of pin placement prior to implementation. The procedures and questions are tailored to several applications of memory interfaces, LVDS interfaces, and other applications. The procedures include the use of the PACE tool and several reference materials. Included are templates and examples of spreadsheets used in the creation of pin-outs.

Introduction

The PACE (Pin-out and Area Constraints Editor) tool helps you to create I/O pin assignments for your FPGA design. PACE has traditionally been employed after the initial HDL synthesis, where the design's primary inputs and outputs are derived from a complete netlist (either EDIF or NGC format). Now, you can also use PACE to create pin assignments before synthesis. In addition to allowing you to create pin assignments either before or after synthesis, PACE also assists in other areas of the pin assignment process. By first answering some basic questions about your design's particular needs, you can use this document to develop an overall pin assignment strategy.

Questions	Answers
How do I start the pin assignment process from my pin list?	Go to the Flexible Input Mechanisms for Pin Entry section.
What factors affect pin assignment on typical FPGA designs (general rules)?	Go to the General Considerations for Pin Assignment section.

Flexible Input Mechanisms for Pin Entry

The following pin entry procedure is used during the entire process of creating your pin placement for your design. Several input mechanisms can be used to create your pin placement. These include specifying the ports in the HDL, by using a spreadsheet (via CSV), or into PACE directly. If pin entry is done via a spreadsheet CSV file, or done through direct entry, PACE can be used to export a Verilog or VHDL (HDL) file, as an initial template for the top-level design. If pin entry is done via an HDL file, PACE can be used to write out a User Constraints File (UCF) or a spreadsheet CSV file.

Each of these ways is described in the following sections:

If	Then
You have an HDL design	Go to the Starting From Scratch section.
You have a spreadsheet CSV file	Go to the Importing From a Spreadsheet section.
You have an HDL top-level file with only I/Os	Go to the Extracting I/Os directly From HDL Source File section.
You have the I/Os listed on a piece of paper or a napkin	Go to the Directly Entering I/Os From Within PACE section.

In addition to the above methods to start pin entry prior to HDL synthesis, you can always use PACE with a design that has already been synthesized and translated through to the NGD netlist format.

Starting From Scratch

This pin entry method reads the top-level input and output ports from the HDL source file. The design does not necessarily need to have been synthesized. The example process outlined below shows you how to launch the PACE tool with the HDL file, specifying prohibits on special purpose pin/sites, and then place data, address, and control signals based upon the specific application suggestions. Then, you write out a CSV file and/or UCF file. The CSV file can be read into a spreadsheet reader or a board layout tool. The UCF file can then be used during implementation of the design.

1. Launch the PACE tool from ISE or command line (Figure 1).

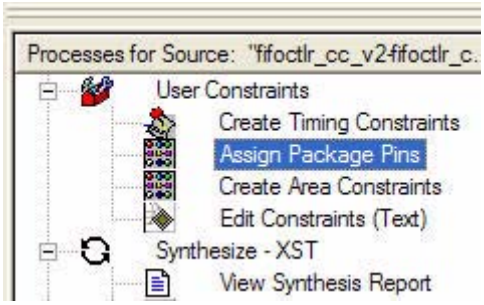

Figure 1: **Launching the PACE tool from ISE**

2. Open the HDL file in PACE (Existing HDL (File -> New) file option). If you are launching PACE from Project Navigator, this step is automatically done for you. Figure 2 shows the New Constraints dialog.

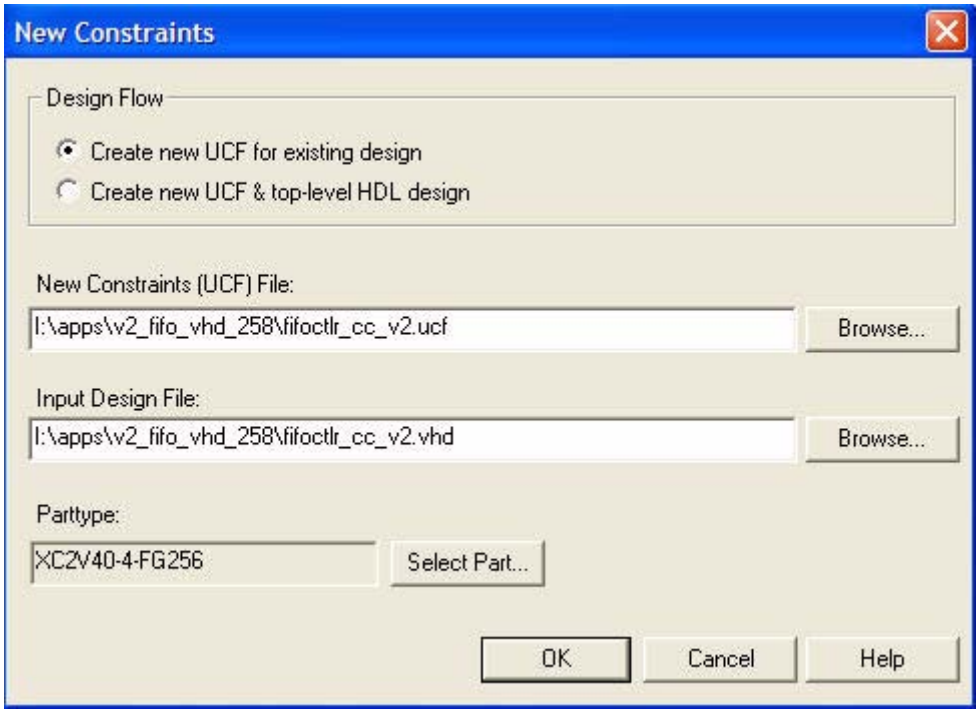

Figure 2: **The New Constraints dialog box**

3. Prohibit special purpose sites (JTAG, Configuration, pin compatible sites, and proposed probe sites). (See General Consideration for Pin Assignment for more details.)
4. Place data and address busses close to each other on the die, to share I/O standards and to help with timing. (See SSO Checking for more details.)
5. Place other signals (Figure 3) based upon the suggestions in the following sections.

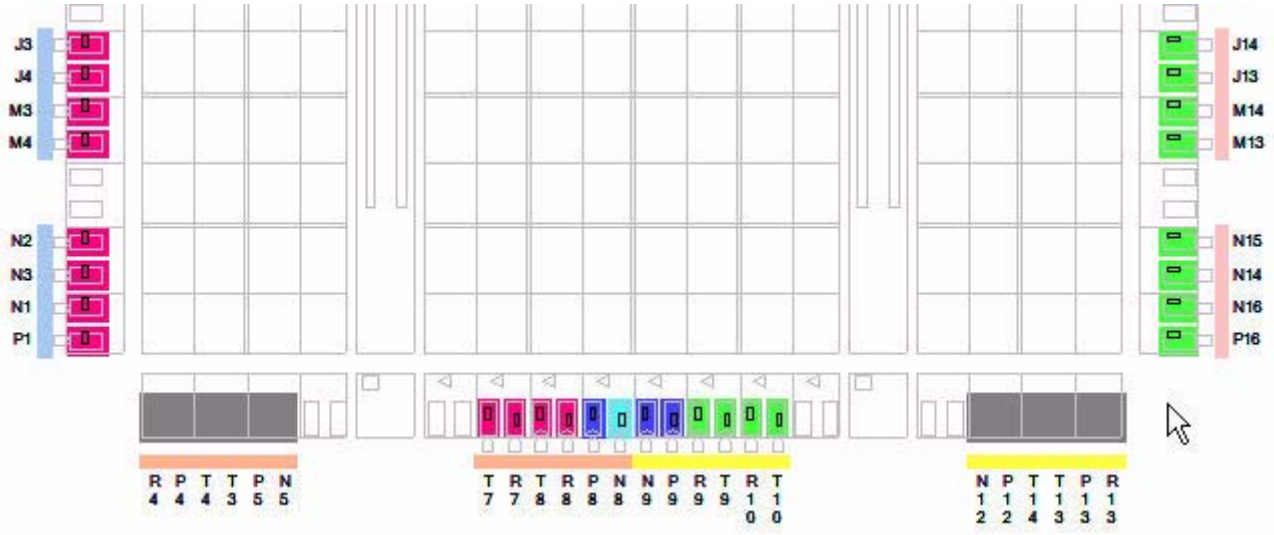

Figure 3: **Signal placement suggestions**

6. Save (File -> Save) the placed pins to the UCF file and/or Export (File -> Export) the placed pins to the CSV file.

Importing From a Spreadsheet

1. It is common for designers to use a spreadsheet program to initially capture the list of the design I/Os. The spreadsheet can be very simple with one column containing the signal names, and another specifying the directionality (input, output, bi-directional). The spreadsheet can also contain the location, the I/O standard, and other properties for each signal. Please refer to the PACE online help for further details. You can easily import that spreadsheet list into PACE, thereby eliminating errors due to the introduction of another manual entry process. The spreadsheet needs to be saved using the CSV (Comma Separated Value) file format. Examples of two spreadsheets are in the zip file as “top_pinouts1.csv” and “top_pinouts2.csv”. The zip file also contains a csv2ucf.pl Perl script that will convert the spreadsheet to UCF. The “top_pinouts2.csv” and “top_pinouts2.xls” files should be used as templates for this Perl script.

The actual design can be in any state of the design process, and all of the ports are listed in the CSV file. Microsoft Excel or various Board Layout tools can create the CSV file. The example process outlined below shows you how to launch the PACE tool and import the CSV file, apply prohibits on special purpose pins/sites, and place data, address, and control signals based upon the specific application suggestions. Finally, you write out an HDL file and/or UCF file. The HDL file can be either Verilog or VHDL and can then be used as a top-level template for the rest of your design. Each synthesis tool has a different default for bus delimiters. When saving the design, you are presented with a dialog that allows you to select the delimiter format for busses. The UCF file can then be used during implementation of the design.

2. Create a list of pin/pads for a given package with the "PARTGen"command (partgen –v 2v40fg256). Figure 4 provides an example of the data from PARTGen with the pad names, the pin location, the pin name, and the pin coordinates. This gives you the correlation between the package-level pads and the die-level pins. It is recommended that you do pin assignments based upon the die-level pins, to add in the placement of pins next to each other. Populate the spreadsheet with this information.

```
package 2v40fg256
pin     PAD1        C4      0   IO_L01N_0             X1Y15    0S
pin     PAD2        B4      0   IO_L01P_0             X1Y15    0M
pin     PAD3        D5      0   IO_L02N_0             X1Y15    1S
pin     PAD4        C5      0   IO_L02P_0             X1Y15    1M
pin     PAD5        B5      0   IO_L03N_0/VRP_0       X3Y15    2S
pin     PAD6        A5      0   IO_L03P_0/VRN_0       X3Y15    2M
pin     PAD7        B7      0   IO_L94N_0/VREF_0      X5Y15    3S
pin     PAD8        A7      0   IO_L94P_0             X5Y15    3M
pin     PAD9        D8      0   IO_L95N_0/GCLK7P      X7Y15    4S
pin     PAD10       C8      0   IO_L95P_0/GCLK6S      X7Y15    4M
pin     PAD11       B8      0   IO_L96N_0/GCLK5P      X7Y15    5S
pin     PAD12       A8      0   IO_L96P_0/GCLK4S      X7Y15    5M
pin     PAD13       A9      1   IO_L96N_1/GCLK3P      X9Y15    6S
pin     PAD14       B9      1   IO_L96P_1/GCLK2S      X9Y15    6M
pin     PAD15       C9      1   IO_L95N_1/GCLK1P      X9Y15    7S
pin     PAD16       D9      1   IO_L95P_1/GCLK0S      X9Y15    7M
pin     PAD17       A10     1   IO_L94N_1             X11Y15   8S
```

Figure 4: **Data from PARTGen**

3. Launch your favorite spreadsheet editor by opening the space and tab separated device.pkg file.
4. Avoid assigning every pin site in each Bank / I/O Standard area in the spreadsheet. Leave extra pins available in each Bank / I/O Standard for adding probes and debugging.
5. Avoid assigning pins to special purpose sites (JTAG, Configuration and pin compatible sites) in the spreadsheet. (See General Consideration for Pin Assignment for more details.)
6. Assign data and address busses close to each other, to share the I/O standards and to help with timing, in the spreadsheet. (See SSO Checking for more details.)

7. Assign the other signals based upon the suggestions in the following sections (Figure 5). The "top_pinouts1.xls" and "top_pinouts1.csv" are available in the zip file.

	A	B	C	D	E
1	package	2v40fg256			
2	Pad Number	Pin Number	Pin Name	Signal Name	Direction
3	PAD1	C4	IO_L01N_0	write_data_in<1>	Input
4	PAD2	B4	IO_L01P_0	write_data_in<3>	Input
5	PAD3	D5	IO_L02N_0	read_data_out<0>	Output
6	PAD4	C5	IO_L02P_0	read_data_out<2>	Output
7	PAD5	B5	IO_L03N_0/VRP_0		
8	PAD6	A5	IO_L03P_0/VRN_0		
9	PAD7	B7	IO_L94N_0/VREF_0		
10	PAD8	A7	IO_L94P_0		
11	PAD9	D8	IO_L95N_0/GCLK7P		
12	PAD10	C8	IO_L95P_0/GCLK6S		
13	PAD11	B8	IO_L96N_0/GCLK5P		
14	PAD12	A8	IO_L96P_0/GCLK4S		
15	PAD13	A9	IO_L96N_1/GCLK3P		
16	PAD14	B9	IO_L96P_1/GCLK2S		

Figure 5: **Signal assignment suggestions**

8. Launch the PACE tool from ISE or command line (see Figure 6).

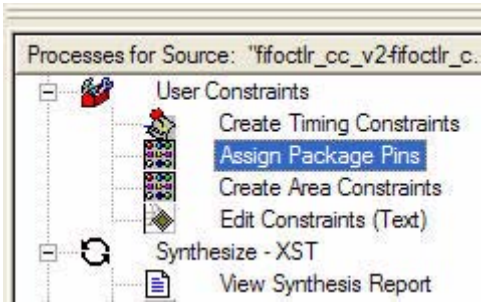

Figure 6: **Launching the PACE tool from ISE**

9. Open PACE (New HDL (File -> New) file option). Figure 7 shows the New Constraints dialog box.

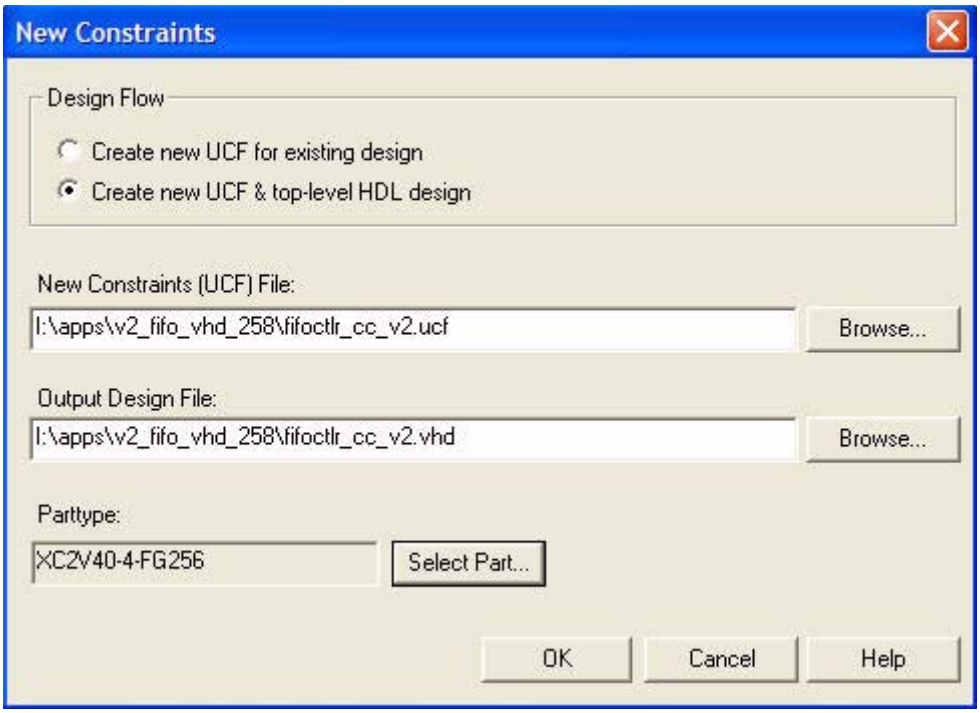

Figure 7: **The New Constraints dialog box**

10. Import the CSV file (File -> Import) and view your placement (Figure 8).

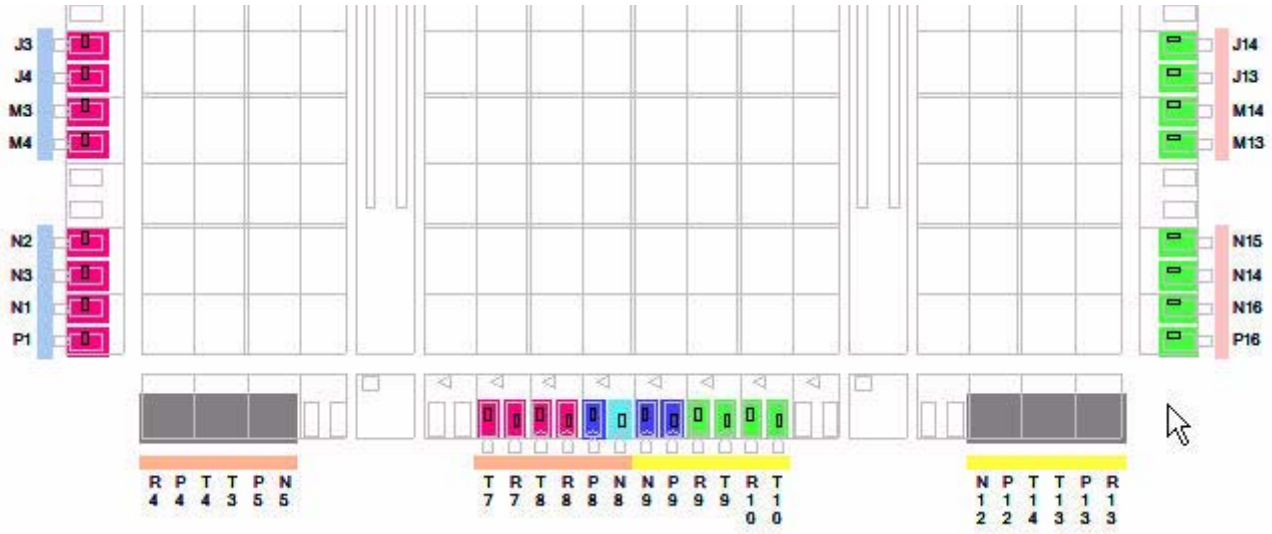

Figure 8: **Viewing pin placement**

11. Run a DRC on the pin placement (Tools -> DRC), and fix any placement errors/warnings you find. Figure 9 illustrates several SSO Warnings in the design. We recommend that you review the SSO Checking section.

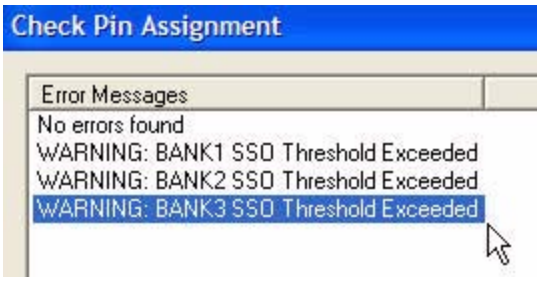

Figure 9: **Pin assignment warnings**

12. Save (File -> Save) the placed pins in the UCF file and in the HDL file.

Extracting I/Os Directly From HDL Source File

Another way to get your design I/Os into PACE is to have PACE directly extract the I/Os from your top-level Verilog or VHDL source. As PACE can directly read Verilog and VHDL, the source file does not need to be synthesized. The example process outlined below shows you how to launch the PACE tool with the HDL file, specify prohibits on special purpose pin/sites and place data, address, and control signals based upon the specific application suggestions. Then, you write out a CSV file and/or UCF file. The CSV file can be read into a spreadsheet reader or a board layout tool. The UCF file can then be used during implementation of the design.

1. Launch the PACE tool from ISE (Figure 10) or command line.

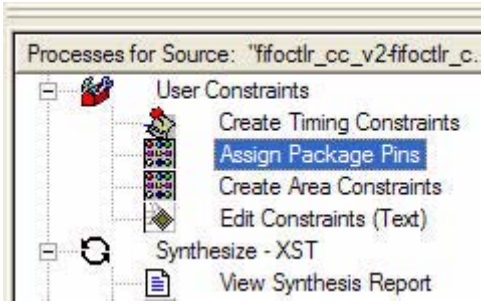

Figure 10: **Launch the PACE tool from ISE**

2. Open the HDL file and specify an UCF name in PACE (Existing HDL (File -> New) file option). If launching PACE from Project Navigator, this step is automatically done for you. Figure 11 shows the New Constraints dialog box.

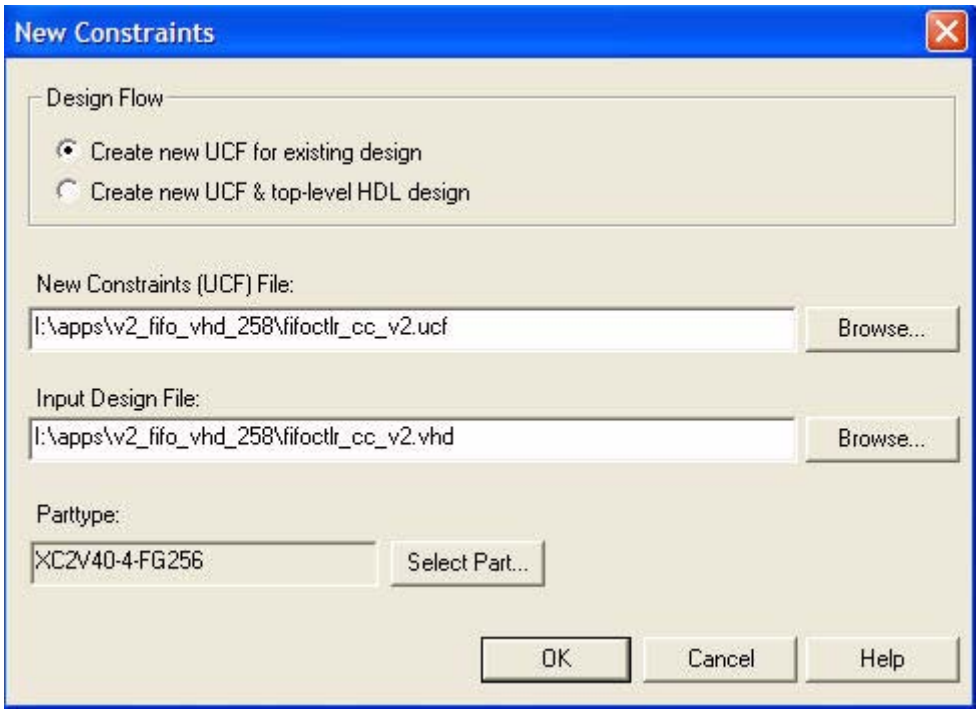

Figure 11: **The New Constraints dialog box**

3. Prohibit special purpose sites (JTAG, Configuration, pin compatible sites, and proposed probe sites). (See General Considerations for Pin Assignment for more details.)
4. Place data and address busses close to each other on the die, to share I/O standards and to help with timing. (See SSO Checking for more details.)
5. Place other signals based upon the suggestions in the following sections (Figure 12).

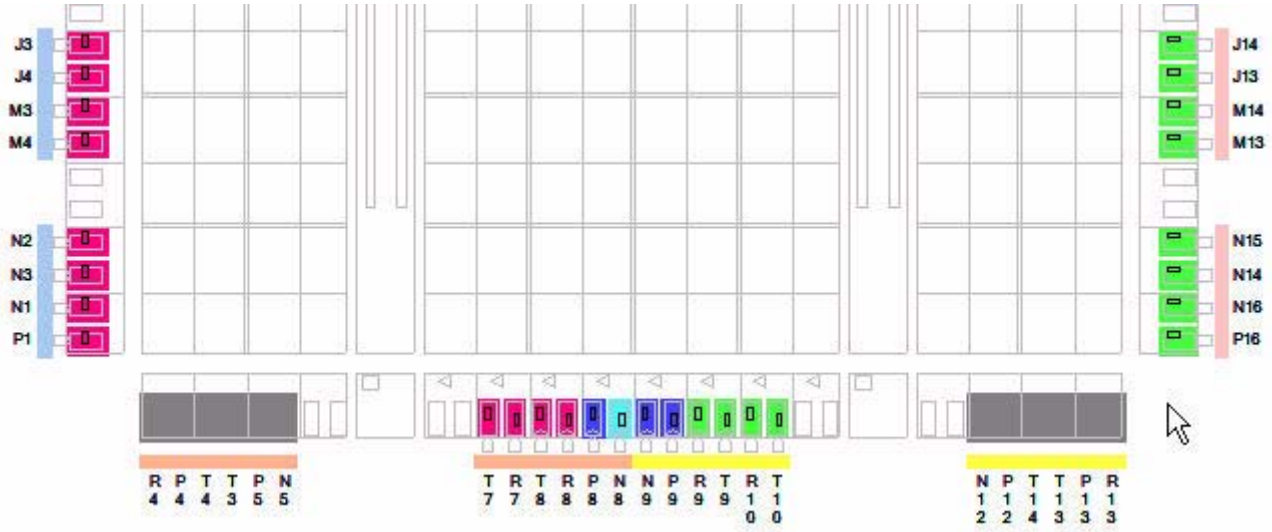

Figure 12: **Signal placement suggestions**

6. Save (File -> Save) the placed pins to the UCF file and/or Export (File -> Export) the placed pins to the CSV file.

Directly Entering I/Os From Within PACE

It is also possible to use PACE as the actual pin entry tool. PACE's Design Object List view can be used to directly enter pin names and directionality. When using PACE to directly enter your pin list, the Create Bus (IOBs -> Create Bus) feature will help you to create busses and assign properties using a single dialog.

The actual design is at the beginning stages of the design process. PACE can create a HDL, CSV, and/or UCF file based upon your list of I/Os. The example process outlined below shows how to launch the PACE tool, then illustrates how to directly enter the port signals into PACE. As before, you can then specify prohibits on special purpose pins/sites and place data, address, and control signals based upon the specific application suggestions. Then, you will write out an HDL, CSV, and/or UCF file. The HDL file can be either Verilog or VHDL and can then be used as a top-level template for the rest of your design. The CSV file can be read into a spreadsheet reader or a board layout tool. Each synthesis tool has a different default for bus delimiters. When saving the design, you will be presented with a dialogue that allows you to select the delimiter format for busses. The UCF file can then be used during implementation of the design.

1. Launch the PACE tool from ISE (Figure 13) or command line.

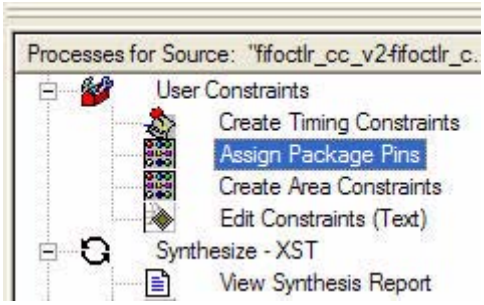

Figure 13: **Launch the PACE tool from ISE**

2. Open PACE (New HDL (File -> New) file option). Figure 14 shows the New Constraints dialog box.

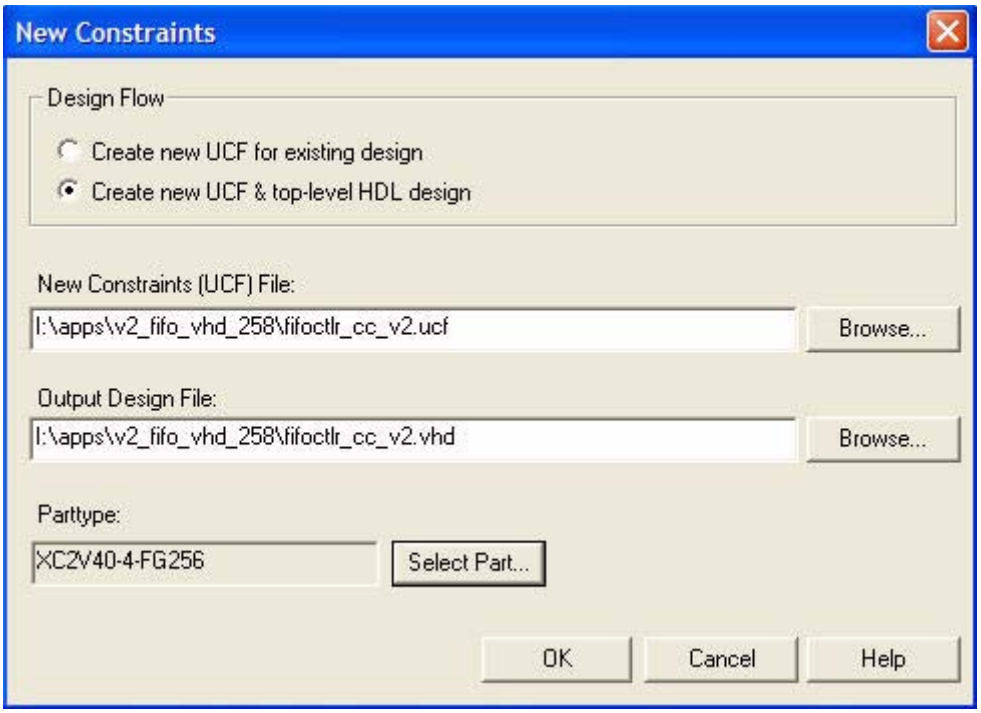

Figure 14: **The New Constraints dialog box**

3. Enter all of the port signals of the design in the Design Object List (DOL) (Figure 15). The IOB -> Create Bus dialog can be used to create a bus in the DOL.

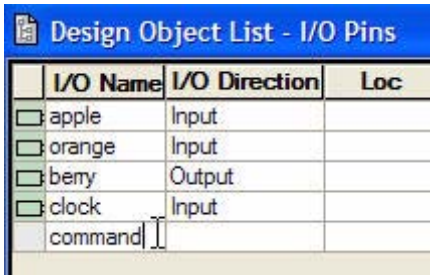

Figure 15: **The Design Object List**

4. Prohibit special purpose sites (JTAG, Configuration, pin compatible sites, and proposed probe sites). (See General Consideration for Pin Assignment for more details.)
5. Place data and address busses close to each other on the die, to share I/O standards and to help with timing. (See SSO Checking for more details.)
6. Place other signals based upon the suggestions in the following sections (Figure 16).

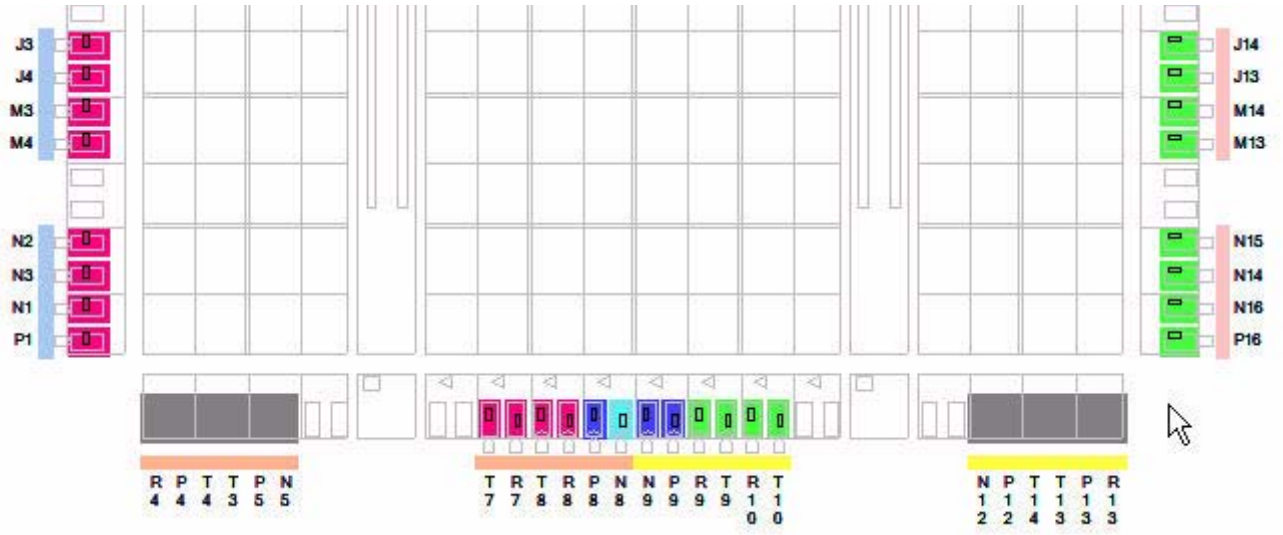

Figure 16: **Signal placement suggestions**

7. Save (File -> Save) the placed pins to the UCF and HDL file and/or Export (File -> Export) the placed pins to the CSV file.

General Considerations For Pin Assignment

PACE has numerous mechanisms to help you create pin assignments that are correct. Of course, in addition to the above methods to start pin entry prior to HDL synthesis, you can always use PACE with designs that have already been synthesized and translated through to the NGD netlist format.

In addition to the pin entry process, the following general considerations should be used as a foundation for your pin placement. The considerations include I/O Banking rules, differential pair placement, SSO checking, Package Flight Time Analysis, Clocking, and global logic placement. These considerations can be applied in the early stages of the design process. Each consideration might have several questions to help get you started in the process of doing your pin placement and assist you in determining your core requirements based upon your specific application and your design. These considerations are based upon various Application Notes and Architecture User Guides. Some general guidelines are listed below.

I/O Banking Rules

In today's market, designs keep getting faster and require more complex interfaces. These designs are also typically more complex and require larger I/O counts. The placement of ports in a high-speed design should be based upon the internal data flow of the design, the design functionality, and the Xilinx Architecture. The current implementation tools do not always place your ports in the best locations for best performance relative to other devices on the board. If you place the ports in a poor location, you will not get the best performance possible for your design.

By default, the Architecture and Package Pin views are color-coded to show the I/O Banks. This can be used as a visual indicator during pin assignment to ensure you are placing the I/Os into the intended target bank. All I/Os placed within a given bank must be compatible with each other. This means that the I/O standard assigned to all of the pins in a bank must be of compatible types. If you assign I/Os to a bank that has incompatible standards, PACE issues DRC warnings. (Note: You must run Tools -> DRC to see these errors/warnings.) For example, consider the case where one pin, A, is assigned to the I/O standard of LVCMOS25, and another

pin, B, is given the standard LVCMOS18. You will get DRC warnings because the two I/Os have different VCCO requirements.

The port placement should be based upon the I/O Banking rules for each Architecture that you are using. You can read the specific I/O Banking Rules that are described in each of the Architecture Data Sheets or use the PACE tool. The PACE tool can do a Design Rule Check (DRC) on the placed ports, and it will inform you if there are I/O Banking issues.

For the Virtex-II, Virtex-II Pro, and newer FPGA families, the I/Os are divided into eight banks, each one with one or more VCCO, VREF, and VRN/VRP pins. The VCCO and VREF are used for I/O standards that need an output or input reference voltage, respectfully. Each bank can have up to twelve VREF pins and should be tied to the same voltage reference. VRN/VRP pins are used for DCI termination; and the VRN should be pulled up, and the VRP should be pulled down. Only one DCI I/O standard can be used with either split or single termination style per bank.

You can assign all of the commonly used properties to the I/Os in your design in the fields of the Design Object List. These properties include I/O standard, output drive strength, output slew rate, termination type, and input delay element type. You can apply properties to the entire group in a single operation with the Group (Edit -> Group) function. Busses are automatically put into groups by PACE.

Questions:

A. How many inputs and output ports are in the design?

B. How many different I/O standards need to be supported?

C. How many ports for each I/O standard?

D. Are you using board termination or DCI?

E. Are there plans for future expansion of the design?

If you run out of general-purpose pins for your ports, then you can use the special or dual-purpose pins for your ports. Special or dual-purpose pins are the configuration and JTAG pins.

As a rule of thumb, if the size of the design is less than 100,000 gates, then it is recommended that you place the control signals on the top or bottom sides and run them vertically across the device. It is also recommended that you place the data and address busses on the left or right sides and run them horizontally across the device. If your design has more than 500,000 gates, use the same recommendations as the design with fewer than a hundred thousand gates, but place the high-fanout signals in the middle of the device and run them horizontally across the device.

Paying attention to signal integrity issues is more critical at higher design frequencies (e.g., if the frequency of the design is at or above 100 MHz). You can use HSTL, SSTL, and GTL I/O standards to decrease transition times at higher speeds, which can do the following:

- Lower current drive
- Lower voltage swing
- Reduce sensitivity to ringing
- Reduce noise

These standards use VREF (input reference voltage) and DCI termination. DCI can then be used to increase the chip-to-chip performance.

In addition to signal integrity, many designs have a clock that is forwarded out of one device at the same rate as the data is pushed out. This is called source-synchronous design, and the data timing is synchronized with the transmission clock. In a source-synchronous design, the clock can be either free running or a data strobe. If the clock is free running, it is usually produced by a DCM with a phase-shifted value to center align the clock in the data valid window. If the clock is a data strobe, it does not use a DCM to produce the clock signal, but

uses other low-skew resources of the device. It is recommended that you choose I/O placements that place the data capture elements close to the clock. This will help your design to meet setup/hold-timing requirements.

If you are using Dual-Data Rate (DDR) Flip-Flops in your design, it is recommended that you place adjacent I/Os in a four-pad IOI tile setup. Please refer to the Clocking Considerations for more details on the four-pad IOI tile setup. This will ensure that the two DDR sets will use the same clock. If one of the DDRs has a different clock than the others, the implementation tools will not be able to route the clock. The PACE tool can do (DRC) for this issue. Also, DDR Flip-Flops must be instantiated in the HDL in order for both Flip-Flops to be placed in the same IOB.

Differential Pair Placement

Differential I/Os (e.g., those with an I/O standard of LVDS or LVPECL) use two device pins, one for the “P-channel” and one for the “N-Channel”. When using PACE to place differential I/O pins, all you need to do is place (e.g., via drag and drop) one of the pins to an appropriate I/O site or any site, and PACE will automatically swap them if needed. PACE will automatically place the other pin to the appropriate matching site. The I/O standard must be instantiated into the design in order to have the auto-placement of differential pairs. LVDS is an Hdriver topology with a weak driver at 100-Ohm differential impedance. LVPECL is an open drain topology with a high-power driver at 100-Ohm differential impedance. If you enable the IOB -> Show Differential Pairs option, PACE uses red arcs to show you all the differential pin pairings for that device (Figure 17).

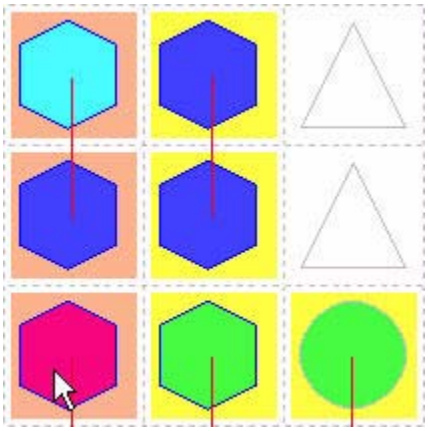

Figure 17: **Red arcs show differential pin pairings**

The correct placement of differential pairs is in concordance to the label for each pin on the package. The I/Os are labeled with IO_LXXY_#, where:

- O stands for the user I/O pin.
- LXXY stands for differential pair.
- XX stands for each unique pair in the bank.
- Y stands for the positive or negative side of the pair (P/N).
- # stands for the bank number of this differential pair.

The differential pair sets should be placed close to each other in each bank, to reduce the number of differential banks.

SSO Checking

Simultaneously Switching Outputs (SSO) checking can also be performed by PACE. This will factor in I/O density within a given bank, I/O standard, drive strength, and slew rate to determine if general SSO guidelines might be violated. Note that this check does not factor in actual design functionality. You should make sure that the number of outputs that can *actually* switch within a given bank do conform to the SSO rules for your device family (e.g., consult the Virtex-II Pro™ Platform FPGA User Guide.)

Questions:

A. How fast are your outputs switching at?

B. What is the total number of Outputs?

C. How many outputs can actually switch at the same time?

SSO is caused by ground bounce on the output drivers and the magnitude of the effect scales linearly. All manufacturers specify a SSO guideline for each type of driver, like a number of drivers per power/ground pairs.

The SSO limits are set for each bank. Even distribution of fast and strong drivers over the package can reduce ground bounce at the chip level. The SSO allowance is based upon the design-specific parameters, such as board-level inductance, input logic-low threshold, input undershoot voltage, and output loading capacitance. To determine the package SSO, you should do a Weighted Average SSO (WASSO) on each bank and the average over the package. The WASSO on each bank should not exceed 100%, and two adjacent banks should not exceed 105%. The package WASSO is based upon the average all banks WASSO. The WASSO value is based upon the Xilinx-assumed inductance, the total PCB inductance, the input undershoot, and input logic low threshold.

Placing a side bus on one edge of the device can cause uneven current distribution and ground bounce with the package. To reduce ground bounce, increase the load capacitance on the output signals.

Xilinx Application Note 689 has a very good guideline for how to calculate SSO and WASSO and a Microsoft Excel-based spreadsheet to calculate WASSO for your design. The spreadsheet asks for the PCB geometry, such as board thickness, via diameter, and breakout trace width and length, to calculate board inductance. It also calculates the average output capacitance based upon the smallest undershoot and logic-low threshold voltage for all input devices. The board inductance, and average output capacitance can then be used to determine the SSO allowance.

If your design and package combination fail the SSO guidelines, it is recommended that you move the design to a larger device or change the number of outputs and/or the I/O standard to reduce or match the SSO criteria.

Package Flight Time Analysis

The Show Fight Times command displays color-coded I/Os in the Package Pins and Device Architecture windows. PACE provides a visual feedback on flight time by color coding the entire package pins view with different colors for different ranges of flight times. These delays are included in Timing Analysis for Flip-Chip packages. You can place your bus signals in the same or smaller delay areas, to help minimize bus skew. Flight Time is the delay from the bonded pad to the package pin. As LVDS DDR frequencies exceed 600 MHz, flight time for the pins of a Flip-Chip package have significant effect on the performance of FPGA devices. This functionality allows you to determine the affect of flight time very early in your design flow.

Clocking Considerations

As designs are using faster interfaces and clock forwarding, you need to know the restrictions and recommendations for each architecture in the area of clocking.

Questions:

A. How many global clocks does the design have?

B. How many local clocks does the design have?

C. Which of those clocks are driven by a DCM?

D. What are the phase shifts for the DCM clocks?

E. For those clocks, what resources, banks, and clock regions do they need access to?

The goal of clock manipulation for a free running clock is to center align the clock in the data valid window by phase shifting the clock with the use of a DCM. The goal for a data strobe is very similar, but it does not use a DCM and the clock is center aligned with the use of delay on the clock line. It is recommended that you place the clock and data pins as close to each other as possible.

The clocks in the design should be placed at dedicated clock pins to help get access to the clock regions and reduce delay to the Global Buffer (BUFGMUXs) and DCMs.

If you have used all of the global clock pins, you can use local clock resources for the remaining clocks. PACE can also be used to locate the local clock pads and the corresponding data pads. The local clock locations are based upon the information in XAPP609. The local clocking structure is based upon Input/Output Interface (IOI) tile, which has Pad0 to Pad3 in each tile for Virtex-II and Pad0 to Pad2 for Spartan-3.

For memory applications with Virtex-II, place the DQS signal on Pad3. The DQ signals can be placed up to five IOI tiles above and six IOI tiles below the DQS signal. For Spartan-3, place the DQS signal on Pad1, and the DQ signals can be placed up to six IOI tiles above and below on the left and right sides only.

For a free-running clock application, the placement of the local clock is dependent upon the bank to which you want to place the clock. The placement of the corresponding data pins is also dependent upon the bank. Table 1 describes the local clock placement per side and the corresponding data pin placement for Virtex-II.

Table 1: **Local clock placement per side and corresponding data pin placement for Virtex-II (See XAPP609 for further details.)**

Side of Device	Pad in IOI Tile	Data Pins
Top/Bottom	Pad1	Up to Five IOI Tiles to the Left
Left	Pad2	Up to Six IOI Tiles Below and Up to Five IOI Tiles Above
Right	Pad1	Up to Six IOI Tiles Below and Up to Five IOI Tiles Above

Table 2 describes the local clock placement per side and the corresponding data pin placement for Spartan-3.

Table 2: **Local clock placement per side and corresponding data pin placement for Spartan-3**

Side of Device	Pad in IOI Tile	Data Pin
Bottom	Pad0	Up to Five IOI Titles to the Right or Left
Top	Pad0	Up to Six IOI Tiles to the Right or Left
Left	Pad0	Up to Six IOI Tiles Below and Up to Five IOI Tiles Above
Right	Pad0, Pad1, Pad2	Up to Twelve IOI Tiles Below and Up to Twelve IOI Tiles Above

If your design has seventeen or more local clocks, Xilinx recommends reducing the number of clocks in the design. One technique is to change the clock signal to a clock enable signal.

If package skew and clock skew are an issue, we recommend placing the clock on the left or right side of the device. This will help to minimize clock skew.

Use Area Constraints to limit the placement of the logic that is driven by a specific clock to a clock region of the device. If there are more than eight global clocks, then area constraints are more critical. PACE can be used to create area constraints for different levels of hierarchy in the design. Clock Analysis can be run to determine the number of clocks per region based upon the current constraints.

Global Logic Placement

You can use PACE to control the placement of “global” logic. Global logic includes BUFGs, BRAMs, MULTs, PPC405s, GTs, DLLs, and DCMs, and is displayed under the Global Logic folder in the Design Browser. Placement can only be done via drag-and-drop to the Architecture View.

Area Groups are the primary means by which you can place logic into specific regions of the device (e.g., within a particular clock region). Any block of design hierarchy (found in the Logic folder of the design browser) can be assigned to an Area Group. Two methods of creating Area Groups are supported:

- Drag-and-drop. Simply drag a block of hierarchy from the browser to the Architecture View. PACE will estimate the size that is required for that block by counting the design elements in that block, as well as compute minimum carry chain height.
- Select then Draw. Once a block is selected from the browser, you can use the Area > Add Rectangle command and use the mouse pointer to draw a rectangular region in the Architecture view.

To get optimum clock resource usage and performance, it is sometimes desirable to draw area groups that conform exactly to the device clock region boundaries. Using the Clock Regions (IOBs > Show Clock Regions) function, the Architecture View will have a color-coded overlay showing all of the device clock regions. You can use this as a template to create area group constraints by ensuring that your area group boundaries are wholly contained within clock region borders.

Questions:

A. How many DCMs are in the design?

B. What is the speed of the DCM outputs?

The Virtex-II/Pro families have up to twelve DCMs that can run up to 450 MHz. There are eight global buffers on both the bottom and the top, for a total of sixteen. Every two global buffers share inputs. There are also up to four clock regions per quadrant.

The Spartan-3 family has up to four DCMs that can run up to 325 MHz. There are four global buffers on both the top and bottom, for a total of eight. Every two global buffers share inputs. There are also up to four clock regions per quadrant.

The DCMs should be placed on the same side of the die as the global clock buffer and the clock pins. A DCM can drive only a total of four different loads. The PACE tool can be used to place clock pins, DCMs, and global buffers. The syntax for locking down the global buffers is “INST BUFGMUX_instance LOC = BUFGMUX0P;”. This constraint is automatically written to the UCF when you drag and drop a global buffer in PACE.

The clocks in the design should be placed at dedicated clock pins to help get access to the clock regions and reduce delay to the Global Buffer (BUFGMUXs) and DCMs. The BUFGMUXs can be configured as a Global Buffer, a 3-state Global Buffer, or a Muxed Global Buffer. Two BUFGMUXs share the same inputs. The implementation tools will not allow you to configure both BUFGMUXs as a Muxed Global Buffers for four clocks. Depending on your design and the BUFGMUX configuration, you might have to use only one of the BUFGMUXs in the pair. Place the BUFGMUXs so the inputs don’t conflict with each other and can easily drive a close DCM.

If your design has eight or less global clocks for Virtex-II and Spartan-3, use all of the Primary global buffers or all of the Secondary global buffers. If your design has between nine to sixteen global clocks for Virtex-II or nine or more for Spartan-3, use the following strategy. First divide up your clocks into the following categories based upon the loading and the frequency of each clock. The frequency cut-off is at 15 MHz. The categories are High Fan-Out/High Frequency, High Fan-Out/Low Frequency, Low Fan-Out/High Frequency, and Low Fan-Out/Low Frequency.

Once you have categorized the clocks, then place all of the High Fan-Out/High Frequency and High Fan-Out/Low Frequency clocks that require full access to all of the devices, then use all Primary global buffers or all of the Secondary global buffers. If two clocks do not require full access to all of the clock regions, they can be placed on the identical opposite sides of a primary and secondary pair. Then, place the Low Fan-out/High Frequency and Low Fan-Out/Low Frequency clocks in the remaining global buffers. Make sure that the non-full access clocks are not placed so that they conflict with other clocks for competing resources and clock regions.

PCB Layout

The routability of the traces to/from the Xilinx FPGA to other devices and the creation of test points are becoming more critical for the PCB layout.

Questions:

A. What does the overall system look like?

B. What interfaces are used to interact with other devices on the board?

C. What is the internal data flow for bus placement?

D. What voltage standards are going to be used for this device?

E. What is the device orientation on the board?

If a bus is connecting to a parallel interface connection on the board, Xilinx recommends using the same order and placement configuration on the particular side of the device.

Make sure that each via has its own power, and do not use shared vias for power. It is best to do blind and/or buried vias.

The PCB should be designed for high-speed signal integrity with a component pad size to closely match the width of the connecting transmission line. You should also minimize the size and number of vias to reduce the impedance mismatches. Therefore, you need to understand the effects that impedance and transmission lines have on your board and the design. Use SMB and parallel fixtures for connecting to test equipment, and if you are using a Serial/De-Serializer, then place it as close to the FPGA as possible.

As the devices get further apart, the PCB traces act like transmission lines, which need to be terminated with the use of DCI functions of the FPGA I/O. Use different I/O banks for the transmit and receive busses and place the control signals in the middle of the receive data busses or between the busses. The control, data, and clock trace lengths should be as short as possible to ensure that the signals arrive early and within a few tens of picoseconds to each other.

It is recommended that you do an IBIS and H-SPICE simulation to evaluate the signal integrity and switching characteristics of the signals in the design for both the FPGA and the interfacing devices. Then do a timing analysis and timing simulation on the design and the board. This should include the trace differences required by DQS and DQ for memory applications that are specified by the manufacture. The trace lengths of the DQS and DQ should be based upon the setup and hold times of the memory device.

As the frequency of the designs gets faster, above 100 MHz, signal integrity becomes more of an issue. The use of HSTL, SSTL, and GTL I/O standards can be used for higher speed interfaces by decreasing the transition times. These I/O standards are to be used in conjunction with VREF. These I/O standards used with DCI can produce the best signal integrity with no stub reflections.

Excessive overshoot on inputs can reverse-bias the clamping diodes in the IOBs and introduces large amounts of noise on the VCCO line. To combat this, decrease the drive strength of the interfaces and/or apply termination on input and output paths. In addition to overshoot, the I/O return current can also produce excessive noise. When a signal is transmitted out of the device and equal and opposite current flow occurs into the device, which is called I/O return current. To reduce the noise from return currents, place de-coupling capacitors on the board to keep the return path intact.

Each manufacturer specifies the maximum number of drivers per power and ground pair, for each driver type, and this specification is the SSO Guidelines. The driver type, along with the current strengths, and slew rates are based upon the SSO guidelines. The package SSO is very similar to the device SSO and is specified by the manufacturer. The distribution of fast/strong drivers is even across the package and will ensure that the device does not generate too much ground bounce. See the SSO Checking section for more details.

Once the pin-out is done, either by hand or via the PACE tool, we recommend giving the board layout engineers the placement of your pins. The board layout engineers may request that some of the I/Os be moved to a new location to make the board layout and interfaces easier.

Memory Interfaces

Interfacing to memory devices is becoming more time intensive for many customers. Clock forwarding is a must for interfacing to memory and this helps the Clock to Out delays, and the skew between address, data, and clock signals with the use of FDDR primitives. IBIS and HSPICE simulations are critical and should include worst-case duty cycle distortion. To limit the board skew, make sure the trace length and width are matched for similar signals. The layout of the data and clocks on the board is very critical

Questions:

A. Which devices interface to the Xilinx device?

B. What is the I/O standard needed for each interface?

C. Which side of the Xilinx device are the other devices located on?

D. What is the location of the other devices on the board?

The DQS and DQ pins should be placed in the same order and location that corresponds to the memory device. If the memory device is to the left of our design, then place the corresponding pins on the left side of the device. If you wish to use HSTL, then place the data and data strobes in banks zero, one, two, six, and seven. This is also true for RX and TX Quad-Data Rate interfaces.

To minimize Clock to Out delays and Pad to Setup delays, use both the input and output DDR functions in the IOBs. To ensure the minimum delays to get in and out of the device, place the Flip-Flops and pads in the proper placement for the specific interface. If the DQ is re-captured at a second stage of Flip-Flops, by another clock domain, place the second stage of Flip-Flops near the first stage.

The placement of DQS signal should utilize the local clock resources. The local clock resources are described in the Clock Consideration section.

If your design uses a core with specific location constraints on the pin placement, like PCI, PL3, or PL4 cores, do not use the same pin placement for other signals.

Configuration Control

Depending on the way the device is going to be configured, we recommend that you prohibit or block out the specific configuration pins. If you do run out of pins on the device, then you can re-use some of the configuration pins or dual-purpose pins as user I/Os.

Device Compatibility and Package Migration

When you are thinking about which package and device to use, you should think about a potential migration path for the design. If your design grows or shrinks, then you might want to see which device/package pairs are pin compatible. To make your pin placement more footprint compatible with other devices and/or packages, use the dual-purpose pins last.

The PACE tool can be used to specify prohibits on the pins that are not in common between your current package and the potential package that you might migrate to. The PACE tool actually places prohibit in the “No Connects” locations of the smaller device, which show up as gray boxes in the larger device.

Conclusion

In conclusion, we discussed several recommendations for the placement of your pins. We discussed the process of placing pins in the PACE tool and have provided suggestions for pin placement based upon other applications and areas of concern. The ZIP file has two examples of pin-outs, one (top_pinouts1.xls) for importing into PACE and the other (top_pinouts2.xls) for running the csv2ucf.pl Perl script.

Appendix

The following Xilinx Application Notes were used as references for this Application Note:

XAPP230, XAPP231, XAPP259, XAPP262, XAPP266, XAPP270, XAPP607, XAPP608, XAPP609, XAPP622, XAPP623, XAPP626, XAPP628, XAPP629, XAPP685, and XAPP689

The Virtex-II and Virtex-II Pro Users Guides were also used as references.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/19/04	1.0	Initial Xilinx release.