CPLD Lab

CPLD Lab

Introduction

This lab uses two 16-bit loadable counters to demonstrate the XC9500 CPLD's pin-locking capability. Since all devices in the XC9500 family are in-system programmable, these devices contain an enhanced pin-locking capability that avoids costly board rework. As described in the course material, the product term allocator controls how the product terms are assigned to each macrocell, while providing outstanding performance.

The purpose of this lab is to provide students the opportunity to implement a design for the XC9500 CPLD and explain some of the differences from the FPGA Flow Engine.

About the CNTSWAP project

The CNTSWAP project shows the effect of routing resources and function block fan-in on the XC9500s pin-locking capability. The project contains two 16 bit loadable address counters loaded from separate buses but with common clock and hold signals (see Figure 1). Note that all pin assignments have already been made in the design. The M1 software recommended these pin assignments.

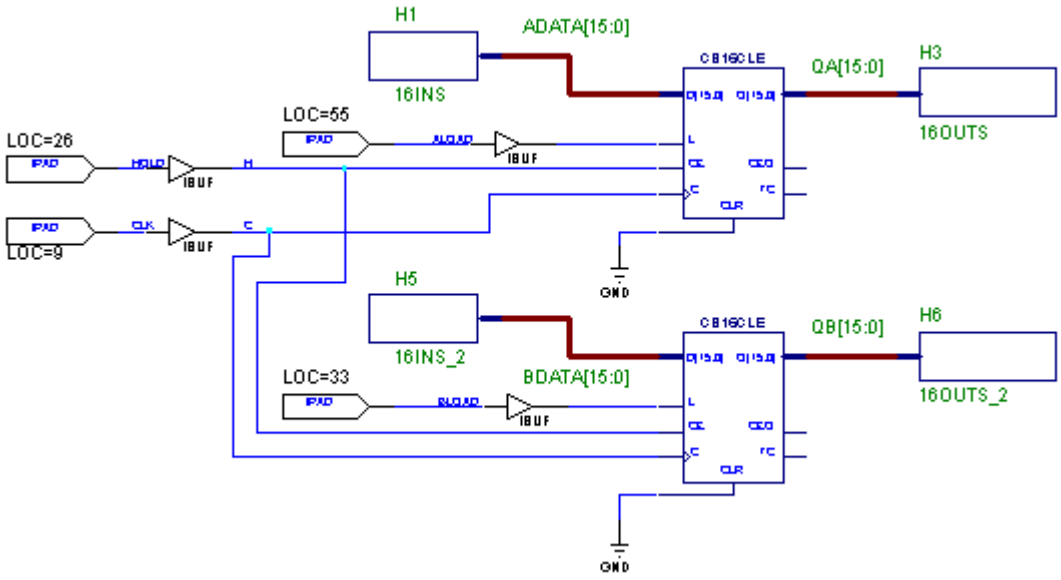

Figure 1. The CNTSWAP project in the c:\F15_labs\9500b4 directory.

Once this design had been implemented, the address data buses were switched and the project saved in the **`c:\F15_labs\9500aft`** directory (see Figure 2). After implementing this next version of the CNTSWAP project and observing the design's performance in the timing analyzer, it is apparent that switching fifty percent of the logic has

had no effect on the software's capability to find a route and that the design performance has not been degraded.

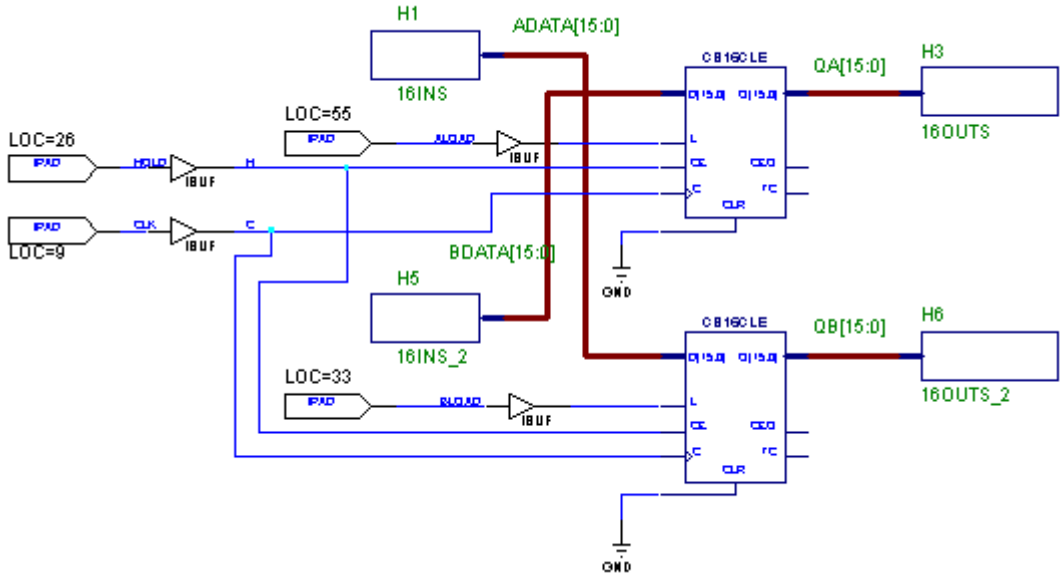

Figure 2. The CNTSWAP project in the c:\F15_labs\9500aft directory.

Pin Locking Issues with CPLDs

Typically, changes are made late in the design cycle, which can cause costly board rework. Historically, CPLDs could not re-route with all the pin assignments and still get the same performance. Some customers had to tolerate poorer performance to get their design to route. Routing congestion, poor fitter performance, or routing feedthroughs often caused the performance degradation. Routing feedthroughs occur when a signal cannot arrive at its destination without wasting another macrocell, and always requires extra delay.

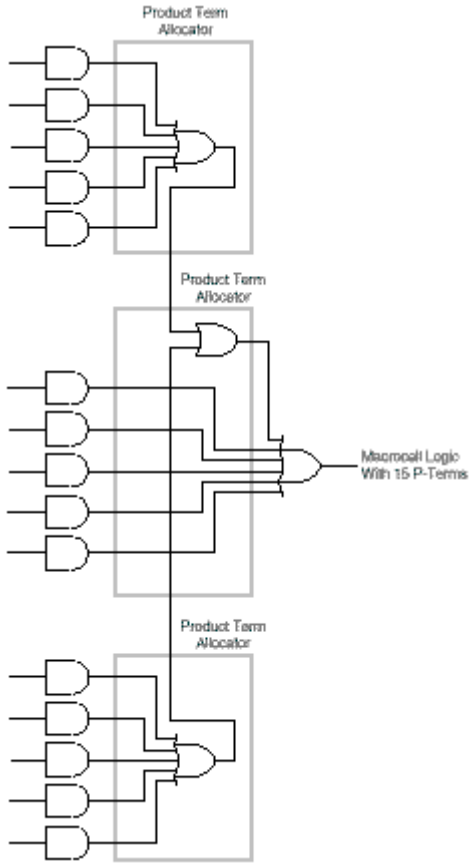

Figure 3. The Product Term Allocation of 15 Product Terms.

The product term allocator in the XC9500 architecture can re-assign other product terms within the function block to increase the logic capacity of a macrocell beyond five direct terms (see figure 3). The allocator is capable of borrowing product terms from any macrocell within the function block, regardless of direction, and can also skip macrocells if necessary. Although borrowing product terms adds an additional delay of approximately one nanosecond, having the flexibility greatly improves pin locking and performance.

Procedure

Opening an existing project within Foundation

1) Open the "**Foundation Project Manager**" by clicking on **Start→Programs→Foundation Series→Foundation Project Manager**.

2) In the "**Project Manager**" window, click on **File→Open Project**.

3) Go to the **c:\F15_labs\9500b4** directory, click on the "**CNTSWAP**" project, and click on the "**Open**" button.

4) In the "**Foundation Project Manager**" window, open the "**Schematic Editor**" by clicking on its icon.

Implementing the CNTSWAP project in the c:\F15_labs\9500b4 directory

5) After looking over the design, switch back to the "**Foundation Project Manager**". Click on the "**M1**" button to generate an EDIF netlist and to automatically create a new project in the **M1 Design Manager**.

6) After the translation is complete, M1 will show the design as the current project in the "**M1 Design Manager**" window.

7) In the "**M1 Design Manager**" window, click on the left most button on the horizontal scroll bar, or click on **Design→Implement**. The Design Manager will prompt for the correct device by bringing up the "**Implement**" window. Make sure that the XC95108-7-PC84 is selected.

8) Click on "**OK**", and click on the "**Run**" button within the "**Implement**" window. This will start the implementation process and bring up the "**Flow Engine**" window.

9) While the Design is being processed, monitor the implementation progress through the Flow Engine's Message window at the bottom of the "**Flow Engine**". After the Flow Engine has stopped running, click on the "**OK**" button.

10) To review the **Fitting Report**, click on **Utilities→Report Browser**. The information in the fitting report will be necessary to answer some of the questions in the next section.

Using the Timing Analyzer

11) To determine the maximum clock frequency, click on **Tools→Timing Analyzer**. Clicking on the "**Timing Analyzer**" icon on the Vertical Tool Bar will also start it.

12) In the "**Timing Analyzer**" window, click on the **Analyze→Advanced Design**, or click on the "**ADV ?** " icon on the horizontal tool bar. The Timing Analyzer will then open the **Performance Summary Report**.

13) The information in the **Performance Summary Report** will be necessary to answer some of the questions in the next section.

14) After completing questions 1 and 3 in the Questions section, close the **M1 Design Manager**.

Implementing the CNTSWAP project in the c:\F15_labs\9500aft directory

15) Implement the **CNTSWAP** project in the **`c:\F15_labs\9500aft`** directory by opening the project within Foundation and repeating the same steps described above. It is important to note the difference between the two projects, and to review the **Fitting Report** generated by the Flow Engine.

16) After reviewing the **Fitting Report**, review the **Design Performance** report generated by the timing analyzer.

17) The information in the **Fitting Report** and the **Design Performance Report** will be necessary to answer some of the questions in the next section.

Questions

1) What was the performance of the CNTSWAP project in the **`c:\F15_labs\9500b4`** directory?

2) What was the performance of the CNTSWAP project in the **`c:\F15_labs\9500aft`** directory?

3) How many product terms were used in Function Block #1 of the CNTSWAP project in the **`c:\F15_labs\9500b4`** directory?

4) How many product terms are available in a single function block in an XC95108?

5) What percentage of macrocells were used by CNTSWAP?

6) What percentage of I/O pins were used by CNTSWAP?

7) How many product terms were imported and exported in function block #1 for the CNTSWAP project?

8) Were the ALOAD, BLOAD, or HOLD signals routed on any global routing resources and how can this be determined?

9) Should using FAST slew rate on ALL the outputs for this design give cause for concern?

10) Which pin is associated with macrocell #5 in function block #1?

11) How can it be determined which input signal uses macrocell #5 in function block #1?

Answers

1) From the Design Performance report generated by the Timing Analyzer, the performance of the CNTSWAP project in the **`c:\F15_labs\9500b4`** directory should be between 75 and 100 Mhz.

2) The performance of the CNTSWAP project in the **`c:\F15_labs\9500aft`** directory should be the same as the project in the **`c:\F15_labs\9500b4`** directory. This was possible since the design does not use very much logic. If the design had been larger, the Product Term Allocator would have helped maintain the design performance.

3) To determine how many product terms were used in Function Block #1, review the Function Block Resource Summary in the Fitter Report. Both implementations should use between 15 and 18 product terms in function block #1.

4) The XC9500 devices have 90 product terms per function block.

5) From the Resource Summary, it was found that approximately 29% of the macrocells were used.

6) CNTSWAP uses approximately 98% of the I/O pins on the PC84 package. Overall, the CNTSWAP project uses very little logic, but almost all the I/O pins.

7) From the Function Block Resource Summary, none of the product terms in any of the function blocks needs to be imported or exported.

8) From the Inputs portion of the Resources Used by Successfully Mapped Logic Section, it was found that ALOAD, BLOAD, and HOLD use I/O pins.

9) Xilinx recommends that not all neighboring I/O pins use FAST slew rate, since this might cause ground bounce at the board level if the designer did not use proper high speed board design techniques. From the Logic portion of the Resources Used by Successfully Mapped Logic section, it is apparent that all the outputs are dispersed to different sides of the device, and therefore there should not be a problem with ground bounce.

10) From the Function Block Resource Summary, function block #1, macrocell #5 is associated with pin #3.

11) The Resources Used by Successfully Mapped Logic/Inputs Section, lists the input signals associated with each macrocell.

Some Tips on using the CPLD Flow Engine's Implementation Options

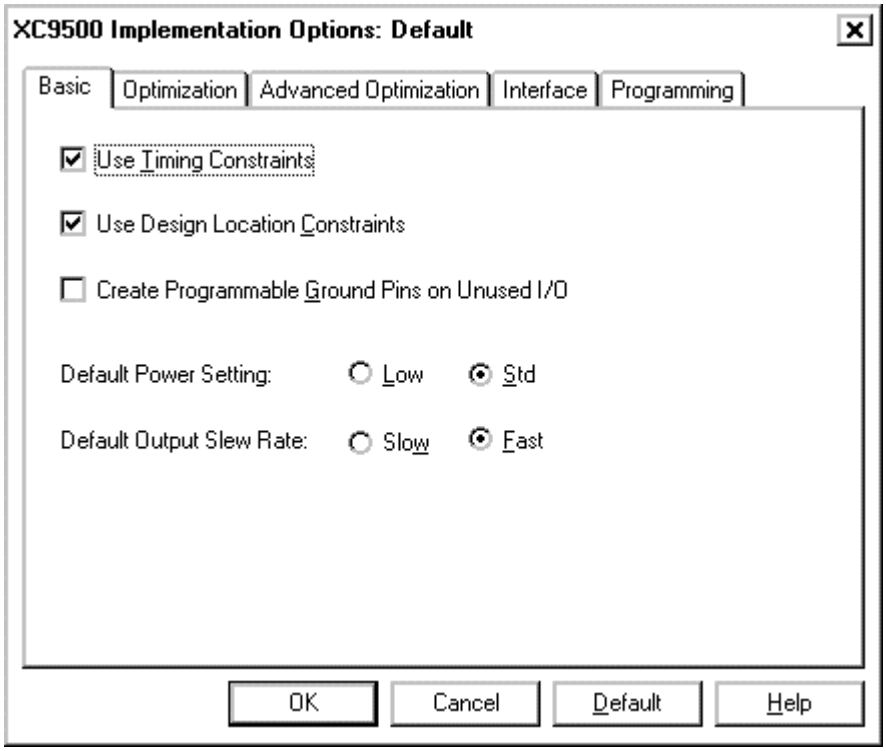

Figure 9. The Basic Implementation Options dialog box.

- Turn on the "**Create Programmable Ground Pins**" option. This configures all unused I/O pads as ground pins, which can reduce ground bounce.
- Set the "**Default Power Setting**" to Low only when an extra delay is affordable. If this option had been asserted for this project, an additional delay would have to be anticipated.
- Set the "**Default Output Slew Rate**" to Slow only when an extra delay is affordable. If this option had been asserted for this project, an additional delay would have to be anticipated.

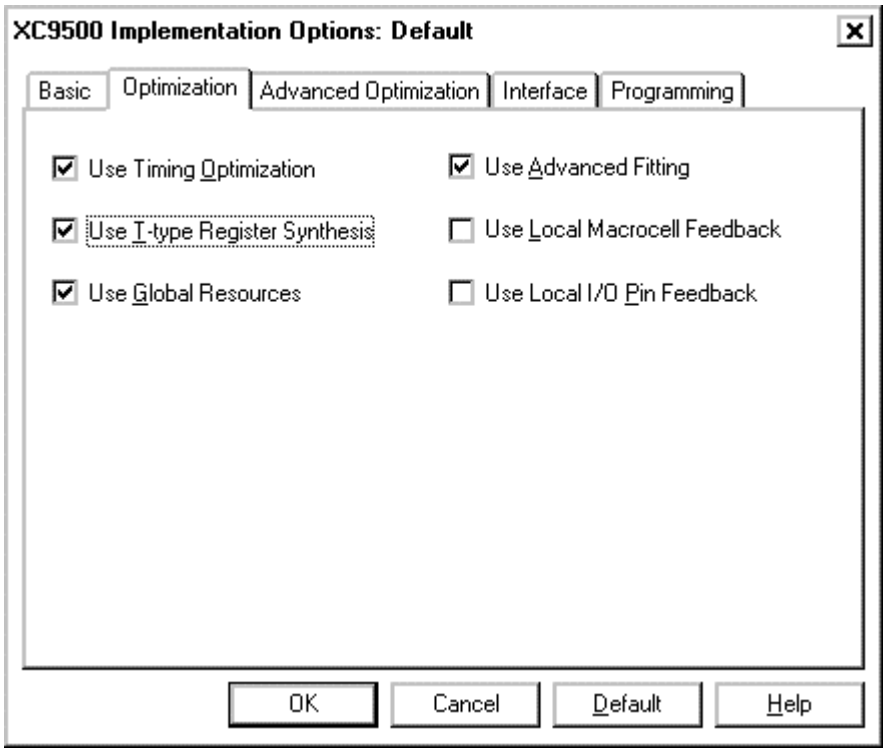

Figure 10. The Optimization Implementation Options dialog box

- Consider asserting the "**Use Local Macrocell Feedback**" and "**Use Local I/O Pin Feedback**" options only if the design needs faster performance. Before considering these options, review the Help information. Remember that the best way to meet performance expectations is to use timing constraints.

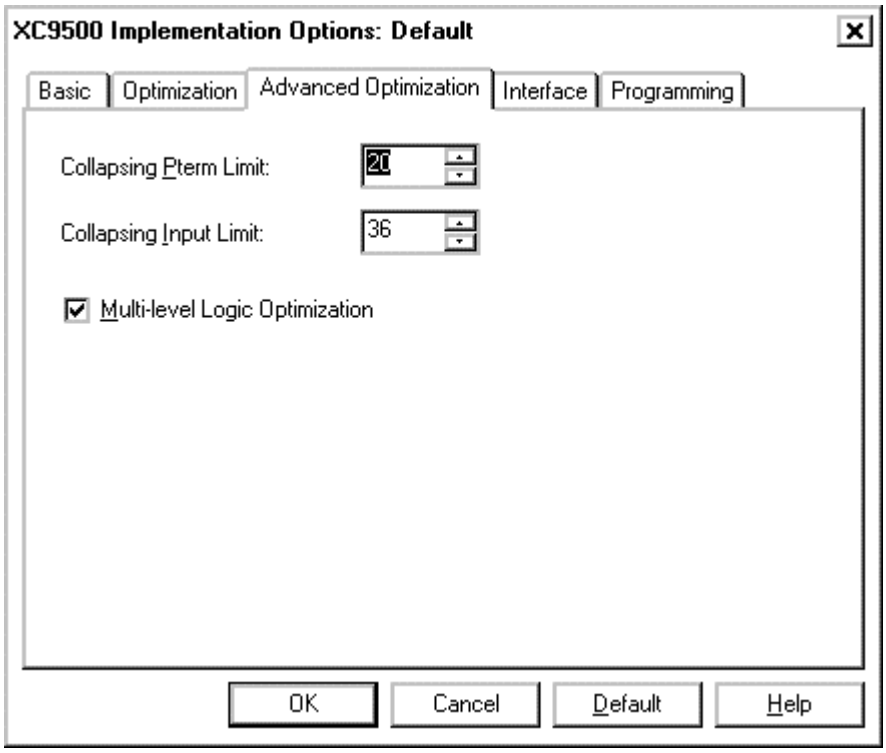

Figure 11. The Advanced Optimization Implementation Options dialog box.

- The "**Collapsing Pterm Limit**" option controls the degree to which a design netlist is flattened. Since the Product Term Allocator can borrow up to eighty-five product terms from macrocells in the same function block, it is sometimes useful to increase this limit since it will allow more logic gates to be collapsed together. The caveat to increasing this limit, is that if the project takes full advantage of this option, the design could become harder to fit. Xilinx recommends using the default of 20 p-terms, however, if some of the macrocells are borrowing nearly 10 p-terms, consider increasing this number to see if the performance needs can be met.
- The "**Collapsing Input Limit**" option controls the degree to which a design netlist is flattened. Since function blocks can only take up to thirty-six signals from the FastCONNECT Switch Matrix, decreasing this number limits the Flow Engines capability to flatten the projects' netlist. Increasing this option up to 36 might enable the Flow Engine to further flatten larger functions, and enable the timing constraints to be met.
- "**Multi-level Logic Optimization**" reduces the total number logic expressions in a design, and then minimizes each logic expression in order to meet user objectives. This option reduces the number of levels of logic, and minimize the number of macrocells and product terms necessary.

About the Fitter Report

The Fitter Report, which results from a CPLD implementation, contains a great deal of information, including:

- The Resource Summary section
- The Resources Used by Successfully Mapped Logic section
- The Function Block Resource Summary

The Resource Summary section

The frequently used information is placed at the beginning in the Resource Summary section. The Resource Summary contains the following information:

- The device chosen for implementation.
- The number of macrocells used and available.
- The number of product terms used and available.
- The number of pins used and available.
- A description of the pin types used in the project (i.e. global, input, or output).
- The number of macrocells in high and low power modes.

The Resources Used section

The Resources Used section of the Fitter Report contains information that specifically describes the resources necessary to generate each output signal. This chart has two sections, which contains the following information:

- The Logic section describes the logic necessary to generate an output signal from the CPLD.
- The Inputs section describes the location of each input signal.
- The following information is included in both sections:
  * The total number of product terms and signals necessary for the signals generation.
  * The specific function block and macrocell location where the signal was generated.
  * The power mode and slew rate of the each macrocell.
  * The pin number and type associated with each macrocell. There are five pin types available: I/O, (b) buried, GTS, GSR, and GCK. A buried pin is a macrocell that does not have a pin associated with it, or does not use its associated pin.

The Function Block Resource Summary section

The Function Block Resource Summary reviews the specific utilization of each function block. Some of the information contained in this chart is:

- The number of function block inputs used and remaining.
- The macrocells that are used and what signal they generate.
- The number of product terms used in each macrocell, and how many were borrowed using the Product Term Allocator.

The Function Block Resource Summary section also includes a graphical representation of the signals used by each macrocell in a particular function block. This chart contains the following information:

- The signals used by the entire function block and their corresponding number.
- A chart describing the fan-in for each macrocell.

Finally, at the end of the report there are Implemented Equations, Device Pin Out, and Compiler Options sections.

About the Timing Analyzer

After the Flow Engine finished compiling, the Timing Analyzer generated the "**Design Performance**" report. This report provides the maximum clock frequency that the design can properly function, the clock-to-output, and the clock-to-setup times.

Conclusion

In this lab, the XC9500 architecture was shown to have significant features to aid pin-locking. In this design, the XC9500 CPLD has the ability to maintain its pin assignments and performance, even when 50 percent of the logic is switched. The Fitting and Design Performance reports were reviewed, and it was shown how to analyze the actual placement of the logic in the CPLD.