ORCA® Series 4 Successful Place and Route

March 2002 Technical Note TN1018

Introduction

ORCA Series 4 Field Programmable Gate Arrays (FPGA) are designed with high performance and flexible routing structures for large, high speed applications.

However, the automatic ORCA Foundry software cannot predict all the specific requirements for a design. In order to make optimal use of a device, the designer should become familiar with a variety of timing constraints (called preferences when designing to ORCA devices) and Place And Route (PAR) techniques for providing the optimal PAR results. This document describes such rules, tips and techniques. Advanced techniques in floorplanning will not be discussed in this document. Instead they are covered in technical note number TN1010, *ORCA Design Floorplanning*.

ORCA Foundry Place and Route Software (PAR)

After a design has undergone the necessary translation to bring it into the physical design (.ncd file) format, it is ready for placement and routing. This phase is handled by the timing-driven PAR software program. Designers can invoke PAR from the ORCA Foundry Control Center or from the command line.

PAR performs the following:

- Takes a mapped physical design (.ncd file) and a preference file (.prf) as input files.
- Places and routes the design, attempting to meet the timing preferences in the input .prf file.
- Outputs a file which can then be processed by the ORCA Foundry design implementation tools.

Placement

The PAR process places the mapped physical design (.ncd file) in two stages: a constructive placement and an optimizing placement. PAR writes the physical design after each of these stages is complete.

During constructive placement, PAR places components into sites based on factors such as:

- Constraints specified in the input file (for example, certain components must be in certain locations).
- The length of connections.
- The available routing resources.
- Cost tables which assign random weighted values to each of the relevant factors. There are 100 possible cost tables.

Constructive placement continues until all components are placed. Optimizing placement is a fine-tuning of the results of the constructive placement.

Routing

Routing is also done in two stages: iterative routing and delay reduction routing (also called cleanup). PAR writes the physical design (.ncd file) only after iterations where the routing score has improved.

During iterative routing, the router performs an iterative procedure to converge on a solution that routes the design to completion or minimizes the number of unrouted nets.

During cleanup routing (also called delay reduction), the router takes the results of iterative routing and reroutes some connections to minimize the signal delays within the device. There are two types of cleanup routing that can be performed:

- A faster cost-based cleanup routing, which makes routing decisions by assigning weighted values to the factors (for example, the type of routing resources used) affecting delay times between sources and loads.
- A more CPU-intensive, delay-based cleanup routing, which makes routing decisions based on computed delay times between sources and loads on the routed nets.

Note that if PAR finds timing preferences in the preference file, timing-driven placement and routing is automatically invoked.

Timing Driven PAR Process

ORCA Foundry offers timing driven placement and routing through a Timing Wizard. The Timing Wizard is an integrated static timing analysis utility (i.e., it does not depend on input stimulus to the circuit). This means that placement and routing is executed according to timing constraints (preferences) that the designer specifies up front in the design process. PAR attempts to meet timing constraints in the preference file without exceeding the specified timing constraints.

To use timing-driven PAR, the designer simply writes timing preferences into a preference (.prf) file, which serves as input to the Timing Wizard. See chapter 5 of the *ORCA Foundry User's Guide* for more information about the PAR software.

General Strategy Guidelines

Preferences should be inserted at the front end of a design flow. This prevents designers from having to change PAR physical preferences as net names may change with every synthesis run.

The tips bulleted below are general recommendations.

- Analyze Trace results in the Timing Wizard Report (.twr) file carefully.
- Look at mapped frequency before you PAR a design to check for errors and warnings in the preference file and to check for logic depth. Logic depth is reported in .twr files as logic levels (components).
- Determine if design changes are required. A typical example design change is pipelining, or registering, the datapath. This technique may be the only way to achieve high internal frequencies if the designs logic levels are too deep.
- It is recommended to perform place and route early in the design phase with a preliminary preference file to gather information about the design.
- Tune up your preference file to include all I/O and internal timing paths as appropriate. The Translating Board Requirements into ORCA Preferences section of this document goes over an appropriate preference file example.
- Establish the pin-out in the preference file. Locating I/O can also be done in the HDL, as well as in sythesis constraint files.
- Push PAR when necessary by running multiple routing iterations and multiple placement iterations.
- Revise the preference file as appropriate, especially utilizing multicycle opportunities when possible.
- Floorplan the design if necessary (see technical note number TN1010, *ORCA Design Floorplanning*).
- Cycle steal as a last resort, remembering to run Trace hold timing checks. Please refer to the Cycle Stealing section of this document for more information on cycle stealing.

Typical Design Preferences

The full preference language includes many different design constraints from very global preferences to very specific preferences. To a new user this is a very large list to digest and utilize effectively. Listed here are the recommended preferences that should be applied to all designs. Please refer to technical note number TN1012, *Constraining ORCA Designs* for more information on preferences.

- **Block Asynchronous Paths:** Prevents the timing tools from analyzing any paths from input pads to registers or from input pads to output pads.
- **Block RAM Reads during Write:** If using PFU based RAM, this will prevent timing analysis on a RAM read during a write on the same address in a single clock period.
- **Frequency/Period <net>:** Each clock net in the design should contain a frequency or period preference.
- **Input Setup:** Each synchronous input should have an input_setup preference.
- **Clock-to-Out:** Each synchronous output should have a clock_to_out preference.
- **Block <net>:** All asynchronous reset nets in the design should be blocked.
- **Multicycle:** The multicycle preference allows the designer to relax a frequency/period constraint on selected paths.

Proper Preferences

Providing proper preferences is key to a successful design. If the constraints of a preference file are tighter than the system requirements, the design will end up being over-constrained. As a consequence, PAR run times will be considerably longer. In addition, over-constraining non-critical paths will force PAR to waste unnecessary processing power trying to meet these constraints, hence creating possible conflicts with real critical paths that ought to be optimized first.

On the other hand, if a preference file is under-constrained compared to real system requirements, real timing issues not previously seen during dynamic timing simulations and static timing analysis would be observed on a test board, or during production.

Common causes of over-constrained timing preferences include:

- Multicycle paths not specified.
- Multiple paths to/from I/Os with different specifications.
- Attempt to fool the PAR tool with tighter than necessary specifications.

Do note that over-constrained designs will also need a significantly larger amount of processing power and computing resources. As a result, it might be necessary to increase some of the allocated system resources (as in increasing your PC virtual memory paging size).

Common causes of under-constrained timing preferences include:

- I/O specifications not defined.
- Asynchronous logic without MAXDELAY preferences.
- Internally generated or unintentional clocks not specified in preference file.
- Blocking critical paths.

In general, to make sure that no critical paths were left out due to under-constraining, it is recommended to check for path coverage at the end of a Trace report file (.twr).

An example of such an output is shown in Figure 1.

Figure 1. Trace Report (.twr) Timing Summary Example

```
Timing summary:
---------------

Timing errors: 4096  Score: 25326584

Constraints cover 36575 paths, 6 nets, and 8635 connections (99.0% coverage)
```

This particular example shows a 99.0% coverage. The way to find unconstrained paths is to run Trace with the "Check Unconstrained Paths" checkbox selected. This will give a list of all of the signals that are not covered under timing analysis. In some designs, many of these signals are a common ground net that indeed does not need to be constrained. Designers should understand this point and use Trace (the ORCA Foundry static timing analysis tool) to check unconstrained paths to make sure they are not missing any design paths that are timing critical.

Also, note the timing score shown in Figure 1. The timing score shows the total amount of error (in picoseconds) for all timing preferences constraining the design. PAR attempts to minimize the timing score, PAR does not attempt to maximize frequency.

The above discussion can be summarized by the following single equality:

Quality of Preference File = Quality of PAR Results

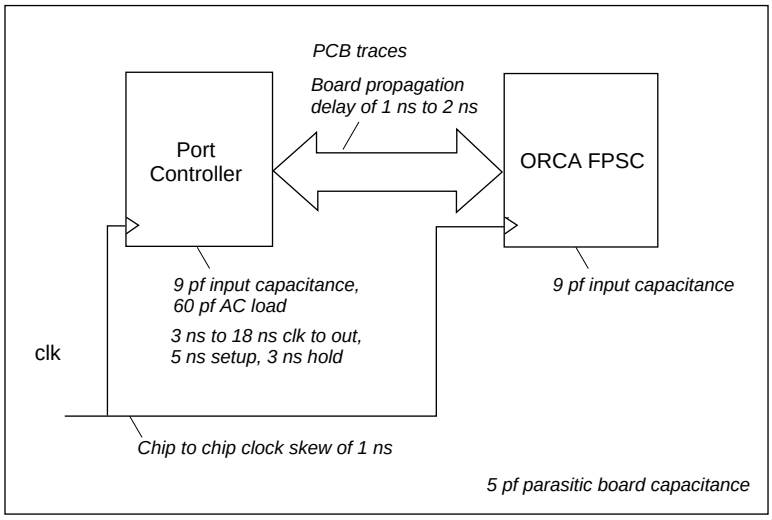
Translating Board Requirements into ORCA Preferences

Understanding the system board level timing and design constraints is the primary requirement for producing a complete preference file. As a result, the major requirements such as clock frequency, I/O timing and loads can be translated into the appropriate preference statements in a constraint file.

The following exercise will provide an example on how to extract preferences from system conditions.

Figure 2 shows an example system involving the interface between a port controller and an ORCA programmable device.

Figure 2. Interface Timing Example



In the system above, several parameters have already been provided:

- System clock frequency: period (P): 30 ns.
- Port controller maximum output propagation delay (PDMAXp): 18ns.
- Port controller minimum output propagation delay (PDMINp): 3 ns.
- Port controller input setup specification (TSp): 5 ns.
- Port controller input hold specification (THp): 3 ns.
- Max board propagation delay (PDMAXb): 6 ns.
- Min board propagation delay (PDMINb): 1 ns.
- Port controller to ORCA device clock skew and vice versa (Tskew): 1 ns.

- Board trace AC loading (Cbac): 60 pf.
- Board trace parasitic capacitance (Cb): 5 pf.
- Port controller input capacitance (Cp) :9 pf.
- ORCA device input capacitance (Co): 9 pf.

The above information was specified under the following environmental conditions:

- Maximum ambient temperature (Ta): 70 (C.
- Estimated ORCA Power Consumption (Q): 2 W.
- 680 PBGAM Package Thermal resistance (Φj) at 0 feet per minute (fpm) airflow: 13.4 °C/W.

The goal of this exercise is to compute the following ORCA device I/O constraints:

1. Input setup specification.
2. Input hold specification.
3. Maximum output propagation delay.
4. Minimum output propagation delay.
5. Output loading.
6. Temperature.

The only parameter which can be obtained from the above is the ORCA device junction temperature:

$$\begin{aligned} Tj &= \Phi j * Q - Ta \\ &= 13.4 * 2 + 70 \\ &= 96.8\ °C \end{aligned}$$

The required constraints can be computed as follows:

1. Input setup specification = P - PDMAXp - PDMAXb - Tskew
   = 30 - 18 - 2 - 1
   = 9 ns

2. Input hold specification = PDMINp + PDMINb - Tskew
   = 3 + 1 - 1
   = 3 ns

3. Output maximum propagation delay requirement
   = P - TSp - PDMAXb - Tskew
   = 30 - 5 - 6 - 1
   = 18 ns

4. Output minimum propagation delay requirement
   = Thp - PDMINb + Tskew
   = 3 - 1 + 1
   = 3 ns

5. Output loading = Cbac + Cb + Cp
   = 60 + 5 + 9
   = 74 pf

The preference file to use for this example is shown in Figure 3. For more preference language syntax and examples, please refer to chapter 4 of the *ORCA Foundry User's Guide* and technical note number TN1012, *Constraining ORCA Designs.*

Figure 3. Interface Timing Preference File Example

```
PERIOD PORT "clk" 30 NS ;
INPUT_SETUP "port_controller*" 9 NS HOLD 3 NS CLKNET "clk";
CLOCK_TO_OUT "port_controller*" 18 NS MIN 3 NS CLKNET "clk";
OUTPUT PORT "port_controller*" LOAD 74 PF ;
TEMPERATURE 96.8 C ;
```

Analyzing Timing Reports

This section describes two examples of actual Trace reports (.twr report file from Trace). The purpose is to analyze both examples and understand each section of the reports given the design paths constrained.

Example 1. Multicycle Between Two Different Clocks

In this first example, CLKA and CLKB were assigned 104 MHz and 66 MHz frequencies respectively.

In addition, a multicycle constraint was specified as per the preference file:

```
FREQUENCY  NET "CLKA" 104 MHZ ;
FREQUENCY NET "CLKB" 66 MHZ ;
MULTICYCLE "M2" START CLKNET "CLKA" END CLKNET "CLKB" 2.000000 X ;
```

See Figure 4 for the block diagram and waveform for this example. The resulting Trace report is shown in Figure 5.

Figure 4. Multicycle Clock Domains Block Diagram and Waveform

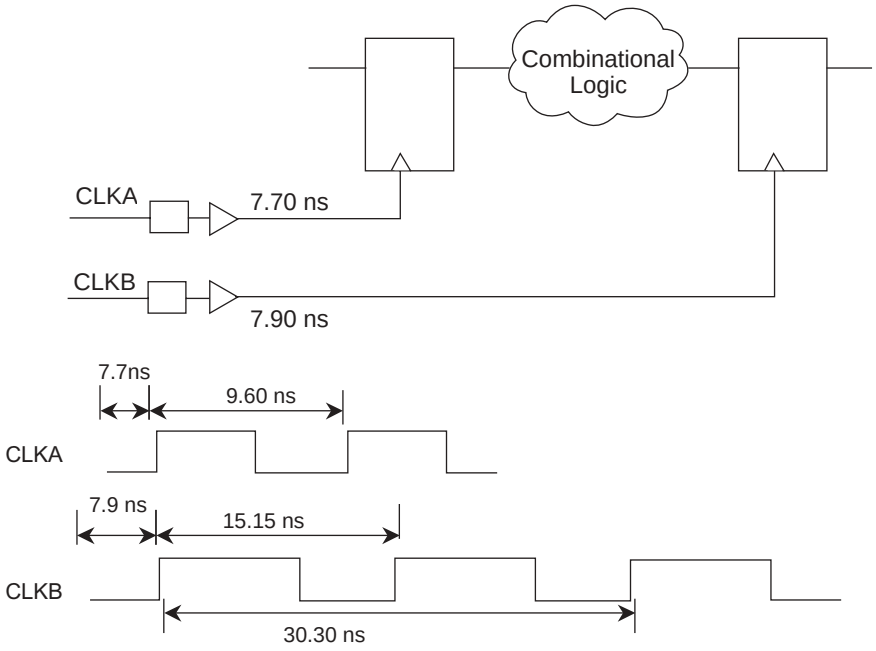


Figure 5. Trace Report for Multicycle Clock Domains Example

```
================================================================================
Preference: MULTICYCLE "M2" START CLKNET "CLKA"  END CLKNET "CLKB"  2.000000 X ;
            40 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
WARNING - trce: Clock skew between net 'CLKA' and net 'CLKB' not
          computed: nets may not be related
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 27.945ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         FF         Q              v_fifo_bank_1_stfifo0_wr_count_2  (from CLKA +)
   Destination:    FF         Data in        v_fifo_bank_1_stfifo0_wr_count_r_2  (to CLKB +)

   Delay:               2.456ns  (37.3% logic, 62.7% route), 1 logic levels.

 Constraint Details:

      2.456ns physical path delay PFU_155 to PFU_156 meets
     30.302ns delay constraint less
     -0.099ns DIN_SET requirement (totaling 30.401ns) by 27.945ns

 Physical Path Details:

   Name    Fanout   Delay (ns)          Site               Resource
REG_DEL     ---     0.917    R22C16.CLK0 to      R22C16.Q2 PFU_155 (from CLKA)
ROUTE         1     1.539      R22C16.Q2 to    R23C17.DIN2 v_fifo_bank_1_stfifo0_wr_countZ0Z_2 (to CLKB)
                  --------
                    2.456   (37.3% logic, 62.7% route), 1 logic levels.

 Clock Skew Details:

 Source Clock Path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192       AM17.PAD to      AM17.INDD ip_CLKA
ROUTE         1     2.989      AM17.INDD to   LLPPLL.CLKIN ip_CLKA_c
MCLK_DEL    ---     0.424   LLPPLL.CLKIN to    LLPPLL.MCLK v_io_ppl3_tx4_1_mtppll_rsp_rsppll_0_0
ROUTE       177     3.094    LLPPLL.MCLK to    R22C16.CLK0 CLKA
                  --------
                    7.699   (21.0% logic, 79.0% route), 2 logic levels.

 Destination Clock Path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192        C17.PAD to       C17.INDD ip_CLKB
ROUTE         1     3.091       C17.INDD to   ULPPLL.CLKIN ip_CLKB_c
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1_mtppll_mac_macpll_0_0
ROUTE       263     3.182    ULPPLL.MCLK to    R23C17.CLK0 CLKB
                  --------
                    7.889   (20.5% logic, 79.5% route), 2 logic levels.
```

In Figure 5, notice how the path is described in terms of "Logical Details."

This section shows both the source and destination registers using their unmapped names from the EDIF (Electronic Data Interchange Format) file. This is a feature that allows the user to recognize the type of logic being analyzed.

Based on the declared frequencies for both clocks, we already know the following:

- CLKA period = 9.6 ns.
- CLKB period = 15.15 ns.

- No relative phase information exists between both clocks. As a result, Trace does not factor in the skews on either clock.

As a consequence, we know that, ignoring everything else (clock skews, registers library setups, etc.), a single cycle positive edge to positive edge setup available from CLKA to CLKB is: 15.15ns (refer to waveforms in Figure 4). Hence, with 2X multicycle, the resulting setup would be twice that number, or:

Ts = 30.3 ns
(shows up as delay constraint under Constraint Details section of Trace report)

Having computed this , the available setup margin is known to be as follows:

M = (Ts - Td) - Ds

Where:

- Td = path delay from clock pin of source register to D pin of destination=2.456 ns. Shown in the Physical Path Details section of Trace report.
- Ds = destination cell library setup requirement= -0.099 ns. This matches DIN_SET under Constraint Details section of the .twr Trace report.

There is no phase relationship between CLKA and CLKB as indicated by the warnings in Figure 5. Hence, the following skews were correctly ignored:

- TSB = delay or skew on destination clock CLKB = 7.889 ns. Shown in the Clock Skews detail section of Trace report.
- TSA = delay or skew on source clock CLKA = 7.699 ns. Shown in the Clock Skews detail section of Trace report.

Hence:

- M = (30.3 - 2.46) - (-0.099) = 27.9 ns. This matches the number in the "PASSED" section at the top of the Trace report.

Example 2. CLOCK_TO_OUT with PLL Feedback

In this example, ip_macclk_c is assigned to 66 MHZ and the clock to out propagation delays are constrained in the preference file:

```
FREQUENCY NET "ip_macclk_c" 66 MHZ;
CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
```

See Figure 6 for the block diagram for this example. The resulting Trace report is shown in Figure 7.

Figure 6. CLOCK_TO_OUT with PLL

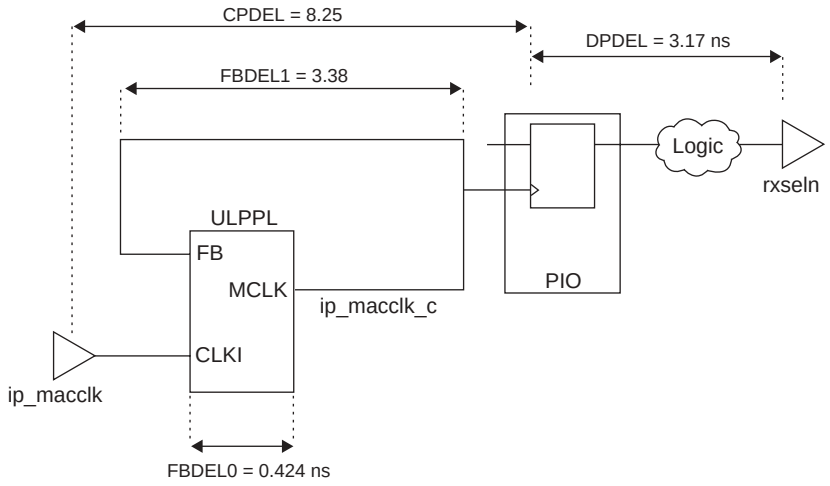


Figure 7. Trace Report for CLOCK_TO_OUT with PLL

```
================================================================================
Preference: CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
            2 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------

Passed:  The following path meets requirements by 0.681ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)
   Source:         IO-FF Out  Q              ppl3_rx5_1_rxselnio  (from macclk +)
   Destination:    Port       Pad            rxseln
   Data Path Delay:     3.164ns  (100.0% logic, 0.0% route), 1 logic levels.
   Clock Path Delay:    8.249ns  (19.6% logic, 80.4% route), 2 logic levels.

 Constraint Details:
      8.249ns delay ip_macclk to rxseln less
      5.094ns feedback compensation
      3.164ns delay rxseln to rxseln (totaling 6.319ns) meets
      7.000ns offset ip_macclk to rxseln by 0.681ns

 Physical Path Details:

      Clock path ip_macclk to rxseln:
   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192        C17.PAD to       C17.INDD ip_macclk
ROUTE         1     3.235       C17.INDD to   ULPPLL.CLKIN ip_macclk_c
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141     3.398    ULPPLL.MCLK to         F32.SC macclk
                  --------
                    8.249   (19.6% logic, 80.4% route), 2 logic levels.

      Data path rxseln to rxseln:
   Name    Fanout   Delay (ns)          Site               Resource
OUTREGF_DE  ---     3.164         F32.SC to        F32.PAD rxseln (from macclk)
                  --------
(100.0% logic, 0.0% route), 1 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141     3.380    ULPPLL.MCLK to      ULPPLL.FB macclk
                  --------
                    3.804   (11.1% logic, 88.9% route), 1 logic levels.

Report:    6.319ns is the minimum offset for this preference.
```

The different path measurements were obtained from the Trace report shown in Figure 7 as follows:

- DPDEL = Data Path Delay = 3.16 ns. Shown under Physical Path Details-> Data path in the timing report.
- FBDEL0 = Feedback cell delay across PLL = 0.42 ns, which is the first entry value under Feedback Path.
- FBDEL1 = Feedback routing delay from PLL output to PLL FB pin = 3.38 ns, which is the second entry value under Feedback Path.

The full feedback delay includes both FBDEL0 and FBDEL1 (0.42 + 3.38 = 3.80) under Feedback Path, in addition to any internal PLL delay added after the FB pin. Such a delay is a programmable attribute defined as FB_PDEL. This programmable value can be set to any of one of 4 values (DEL0, DEL1, DEL2 or DEL3; DEL0 being 0 delay) in either the HDL file input to synthesis, or in the graphical Editor for Programmable Integrated Circuits (EPIC) software tool included with ORCA Foundry.

Therefore, the total feedback delay would be:

$$FBDEL = FBDEL0 + FBDEL1 + FB\_PDEL = 3.80 + FB\_PDEL$$

Under "Constraint Details" of the report file, the feedback compensation (FBDEL) is shown to be 5.09 ns. Since this value is different from 3.804, we conclude that a non-zero value of FB_PDEL was applied (5.10 - 3.80 = 1.29 ns). This value corresponds to FB_PDEL = DEL2 in an OR4E4-2 device.

Now, let's verify the available margin on this CLOCK_TO_OUT preference:

$$\begin{aligned} M &= CKOUT - (CPDEL + DPDEL - FBDEL) \\ &= 7.000 - (8.249 + 3.164 - 5.094) = 0.681 \text{ ns} \end{aligned}$$

This value matches the one at the top of the report file ("Passed" section). It also matches the final value under "Constraints Details".

ORCA Foundry Control Center (OFCC) Controlled Place and Route

This section describes some techniques that can be used within the OFCC graphical user interface (GUI) to improve timing results from Trace on placed and routed designs.

Running Multiple Routing Passes

Improved timing results could be obtained by increasing the number of routing passes during the Routing phase of PAR.

The PAR options window in Figure 8 can be launched by selecting the drop-down menu:

Flow->Design Flow Options ->PAR.

In the example screenshot shown in Figure 8, the router will route the design for five routing iterations, or until all the timing preferences are met, whichever comes first. For example, PAR will stop after the second routing iteration if it hits a timing score of zero on the second routing iteration.

The high selection shown in Figure 8 on the Placement Effort level will result in longer PAR run times but may give better design timing results. A lower Placement Effort will result in shorter PAR run times but will likely give less than optimal design timing results.

Figure 8. PAR Options Window

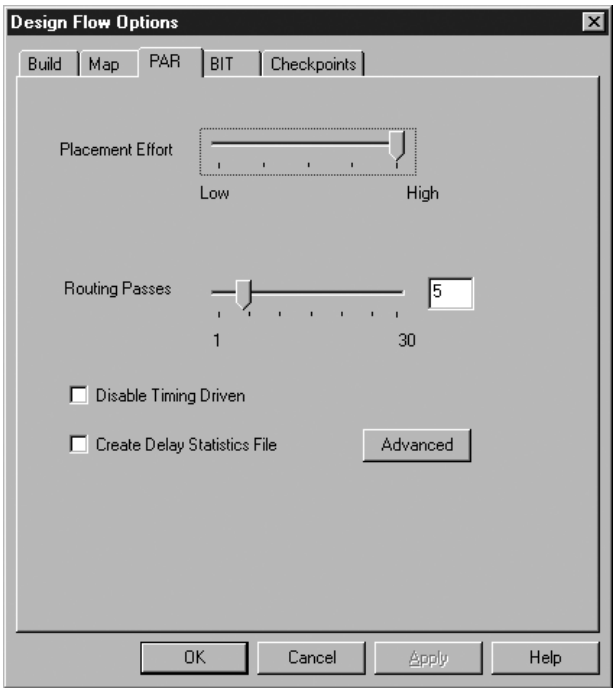


Figure 9. Example PAR report (.par) File, Routing Section

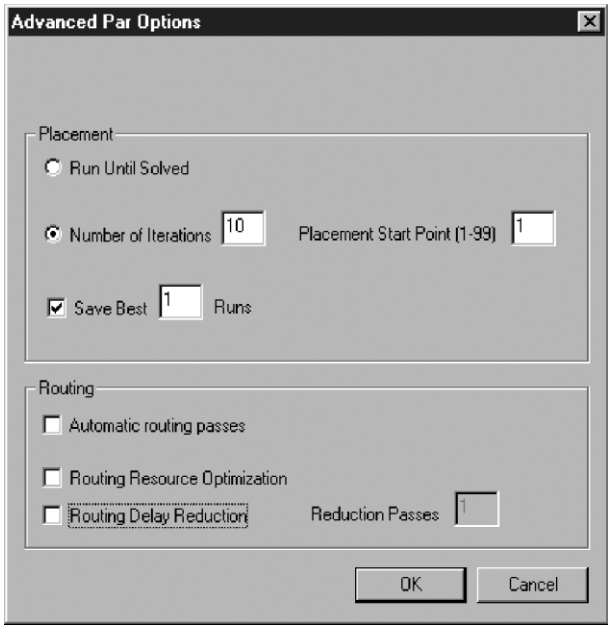
```
0 connections routed; 26590 unrouted.
Starting router resource preassignment
Completed router resource preassignment. Real time: 11 mins 31 secs
Starting iterative routing.
End of iteration 1
26590 successful; 0 unrouted; (151840) real time: 14 mins 29 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 2
26590 successful; 0 unrouted; (577) real time: 16 mins 23 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 3
26590 successful; 0 unrouted; (0) real time: 17 mins 39 secs
Dumping design to file
```

The place and route (.par) report file contains execution information about the PAR command run. The report also shows the steps taken as the program converges on a placement and routing solution. In the routing convergence example text in Figure 9, the number in parenthesis is the timing score after each iteration. In this example, timing was met after three routing iterations, as can be seen from the (0) timing score.

Using Multiple Placement Iterations (Cost Tables)

Using multiple placement iterations can be achieved by selecting the Advanced button in the pop-up window of Figure 9. The resulting window is shown in Figure 10.

Figure 10. Advanced PAR Options



As shown in the Figure 10, the number of iterations is set to 10 and the placement start point is set to iteration 1 (cost table 1). Only the best NCD file is to be saved as per the following line. Once PAR runs, the tool will loop back through the place and route flow until the number of iterations has reached 10. In this example, the NCD file with the best timing score would be saved. The tool keeps track of the timing and routing performance for every iteration in a file called the multiple par report (.par). Such a file is shown in Figure 11.

Figure 11. Multiple PAR Report (.par)

```
Level/          Number      Timing      Run         NCD
Cost [ncd]      Unrouted    Score       Time        Status
----------      --------    -------     -----       --------
5_4   *         0           0           01:58       Complete
5_6             0           25          02:01       Complete
5_2             0           102         01:45       Complete
5_7             0           158         02:15       Complete
5_3             0           186         01:54       Complete
5_10            0           318         02:39       Complete
5_1             0           470         01:51       Complete
5_8             0           562         02:25       Complete
5_5             0           732         02:00       Complete
5_9             0           844         02:27       Complete

* : Design saved.
```

Figure 11 indicates that:

- The "5_" under the Level/Cost column means that the Placement Effort level was set to 5. The Placement Effort level can range from 1 (lowest) to 5 (highest).
- 10 different iterations ran (10 cost tables).
- Timing scores are expressed in total picoseconds (ps) by which the design is missing constraints on all preferences.

- Iteration number 4 (cost table 4) achieved a 0 timing score and hence was the design saved. More than one .ncd file can be saved. This is user-controlled via the "Save Best Runs" value box shown in Figure 10.
- Each iteration routed completely.

Do note that, in Figure 10, if "Run Until Solved" is checked, the tool will run indefinitely through multiple iterations until a 0 timing score is reached. In a design that is known to have large timing violations, a 0 timing score will never be reached. As a consequence, the user must intervene and stop the flow at a given point in time.

In general, multiple placement iterations can help placement but can also use many CPU cycles. Multiple placement iterations should be used carefully due to system limitations and the uncertainty of results. It is better to fix the root cause of timing problems in the design stage.

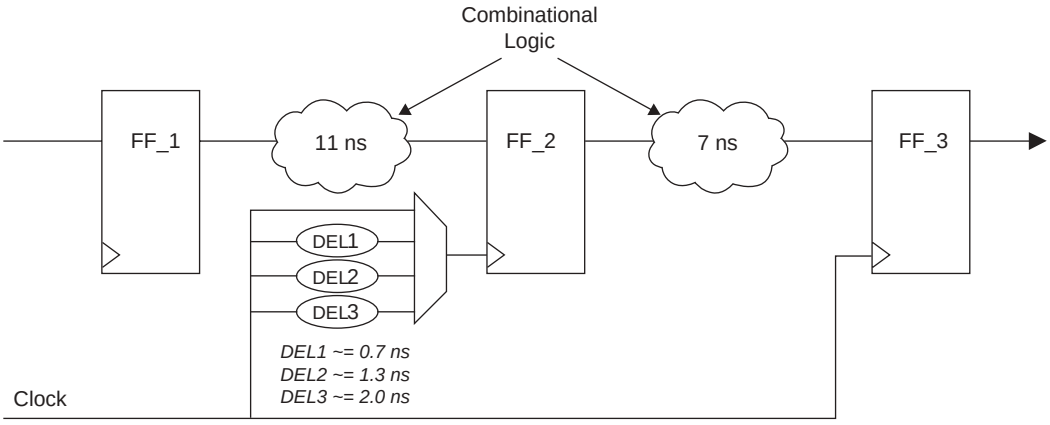
Cycle Stealing

Cycle stealing is the deliberate introduction of clock skew on a target flop to increase the setup margin. Every programmable flip-flop in the device has programmable delay elements before clock inputs for this purpose. The automated cycle stealing tool will attempt to meet setup constraints by introducing delays to as many target registers as needed to meet timing, in effect, steal register hold margins to meet register set-up timing. The following bullets summarize how cycle stealing is accomplished in ORCA devices.

- A 4-tap delay cell structure in front of the clock port of every flip-flop in the device (includes I/O flip-flops)
- Ability to steal clock cycle time from one easily-met path and give this time to a difficult-to-meet path

Cycle stealing is typically most useful in designs that are only missing timing on a few paths for one or two preferences. If the design is missing timing by over a few nanoseconds on any given path, cycle stealing will not be able to schedule skew in a way that will eliminate enough timing to make the critical preference. Cycle stealing run times can be shortened by using a preference file with only the failing preferences in it.

Figure 12. Cycle Stealing Example



Target Performance: 10 ns period (100 MHz)

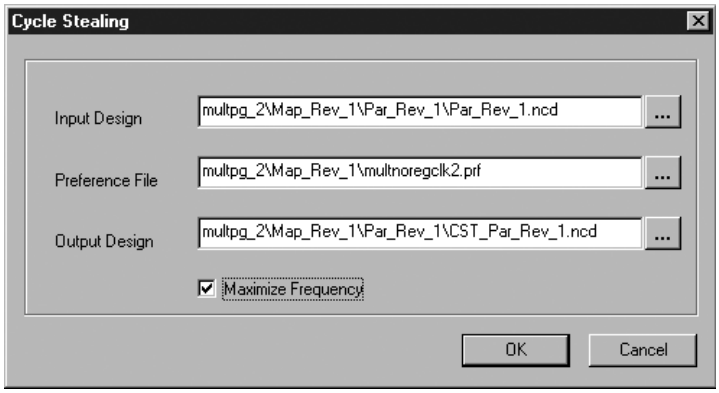
The example illustrated in Figure 12 shows two register-to-register transfers that both need to meet the 10 ns period constraint. By using delay cell DEL2 to delay the clock input on flip-flop FF_2, the first register transfer will make its period constraint with a new minimum period of ~9.7 ns and the second register transfer will make its period constraint by ~8.3 ns.

The D1, D2, and D3 delays shown in Figure 12 are typical values for a -2 speed grade ORCA device. For complete timing information, reference the software generated timing data sheet for Series 4 ORCA devices at %FOUNDRY%\or4e00\data\datasheet.htm.

To run cycle stealing, choose Tools -> Cycle Stealing from the main OFCC window.

As shown in Figure 13, the original .ncd and .prf files as well as the output .ncd file are typed into the corresponding entries. Checking "Maximize Frequency" will push the tool to improve the frequency beyond the input preference requirement. This is generally only useful for bench marking.

Figure 13. Cycle Stealing Window



Other important considerations on the praticality of using cycle stealing:

- Some circuits show big improvement, others have no gain. Cycle stealing results are very design-dependent.
- Cycle stealing uses minimum delay values which have not yet been validated at the system level.
- Automatic cycle stealing identifies skew and hold time issues. However, after cycle stealing is performed, designers are strongly recommended to run Trace twice, once with regular, maximum delay analysis, and again with minimum delays. The designer should then read over both resultant .twr timing reports to make sure there are no timing errors. The minimum delay analysis is done by checking the "Check Hold Times" checkbox in the Trace OFCC GUI window.
- Please reference chapter 6 of the *ORCA Foundry User's Guide* for more information about the Cycle Stealing (CST) software, including the output files that are created by the program.

Guided Map and PAR

To decrease PAR runtimes after minor changes to logical design, guided mapping uses a previously generated. ncd file to "guide" the mapping of the new logical design. Guided mapping can be performed from the Guided Design dialog box in OFCC. In general, guided MAP should only be used in conjunction with guided PAR.

To perform guided mapping in OFCC:

1. Click the Flow>Guided Design menu command.

The Guided Design dialog box appears. See the Guided Design Options Dialog Box in Figure 14.

2. In the Map section, type in the file name and path of the .ncd guide file in the Guide Design text box or use the browse button to navigate to the guide file.

3. Click OK.

The file designated as the guide file will remain designated as the guide file for the Map operation unless changed or deleted from the Guided Design text box.

4. Run a design flow.

The Map operation will use the guide file to generate the new design file. See the Running a Design Flow section in the ORCA Foundry Basics chapter of ORCA DOCS. See also the Notes on Guided Mapping and Synthesis Requirements for Guided Mapping subsections in the Guided Mapping section of ORCA DOCS.

To perform guided PAR in OFCC:

1. In the Par section of Figure 14 , type in the file name and path of the .ncd guide file in the Guide Design text box or use the browse button to navigate to the guide file.

2. Set the matching factor and click on the Verbose Guide Report check box to generate the .gpr file.

Both of the above are optional. The matching factor specifies the percentage of the same connectivity that guiding and guided objects must have. The .gpr file reports all the names of the objects in the new .ncd that does not have corresponding guiding objects in the guide file. See descriptions earlier in this section.

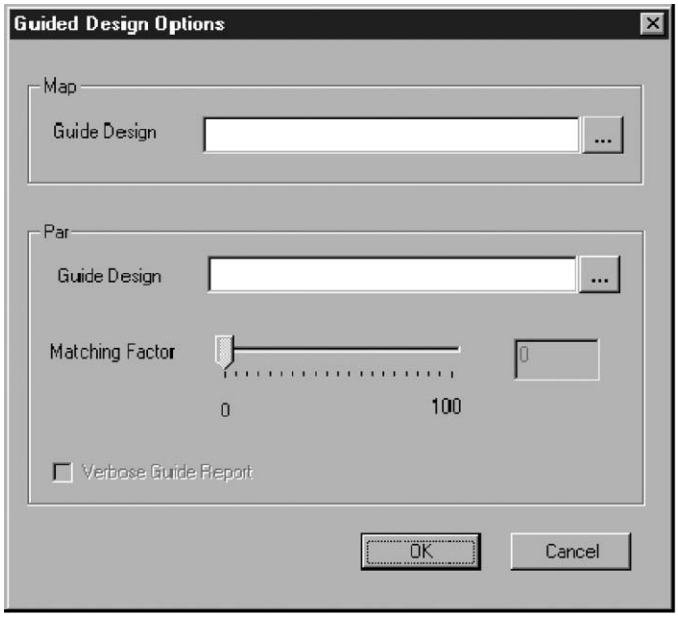
3. Click OK.

The file designated as the guide file will remain designated as the guide file for the PAR operation unless changed or deleted from the Guided Design text box.

4. Run a design flow.

The PAR operation will use the guide file to generate the new design file. See the Running a Design Flow section in the ORCA Foundry Basics chapter.

Figure 14. Guided Design Options



Notes on Guided Mapping

All guidance criteria is based on signal name matching. Topology of combinatorial logic is considered when Soft-wire LUTs (SWLs) exist in the guided file.

Register elements are mapped in two passes. In the first pass, register control signals are matched by name exactly. In the second pass, the control signals names are not matched. This methodology provides a greater chance of matching for registers since control signal names have a tendency to change from successive synthesis runs. Other matching considerations are as follows:

- For combinatorial logic, new SWLs are matched from SWLs extracted from the guide design.
- All unmatched logic are mapped through the regular mapping process.
- The performance of the guided mapped design can be no better than the original.
- A guide report, <design_name>.gpr, gives details of the success guided map had in matching with the guide file.

Notes on Guided PAR

To decrease PAR runtimes after minor changes to the physical design file (.ncd), guided PAR uses a previously placed and/or routed .ncd file to "guide" the placement and routing of the new .ncd file. Guided PAR can be performed from the Guided Design dialog box in OFCC.

For PAR to use a guide file for design, PAR first tries to find a guiding object (i.e., nets, components, and/or macros) in the guide file that corresponds to an object in the new .ncd file. A guiding object is an object in the guide file of the same name, type, and connectivity as an object in the new .ncd file. A guided object is an object in the new .ncd file that has a corresponding guiding object in the guide file.

After PAR compares the objects in each file, it places and routes each object of the new .ncd file based on the placement/routing of its guiding object. If PAR fails to find a guiding object for a component, for example, PAR will try to find one based on the connectivity. PAR appends the names of all objects which do not have a guiding object in the guide file to .gpr (Guided PAR Report) file. The matching factor option applies to nets and components only. When matching factor is 100 (the default), a guiding object must have exactly the same connectivity as the object it is guiding. When a matching factor is specified, the value specified is taken as the minimum percentage of the same connectivity that a guided object and its guiding object have.

After all guided objects are placed and routed, PAR locks down the locations of all guided components and macros and then proceeds with its normal operation. Guided PAR supports the following preferences: USE SPINE, USE PRIMARY, USE SECONDARY, USE LONGLINE, USE HALFLINE, LOCATE COMP, LOCATE MACRO, and hard-placed PGROUPs.

Conclusion

In general, different designs respond better to different strategies. The processes outlined in this application note may not be optimal for all cases. For a design's first place and route, run PAR at the low placer effort level and with a low number of routing iterations. There is no point in running 100 cost tables if the design's logic depth is too high. The techniques discussed within this document, like interpreting static timing reports and using proper preferences, will guide the user to better PAR results.

Technical Support Assistance

Hotline: 1-800-LATTICE (Domestic)
1-408-826-6002 (International)
e-mail: techsupport@latticesemi.com