Reduce Compile Times

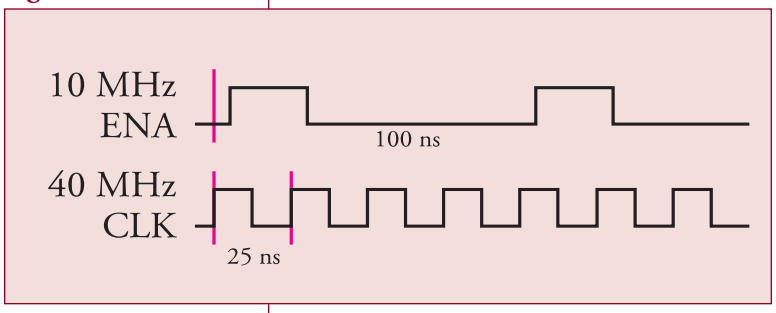
Using Timing Constraints in Foundation *Express*

by LAUREN WENZL ◆ Xilinx Boulder

Using multi-cycle timing constraints for specified paths can decrease place and route run times. Because the place and route tools must work harder to meet aggressive timing requirements, place and route run times can be optimized, if you apply tight timing constraints only to critical logic and apply relaxed timing constraints to your less-critical logic.

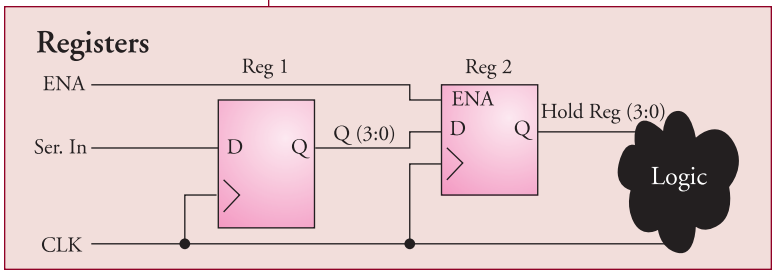
Foundation *Express* provides the ability to create multi-cycle timing constraints quickly and easily by creating timing groups and subsequent subpath timing groups. An example of this is illustrated by the following design, which eliminates clock skew by using derivative clock/enable signals from the system clock for non-critical logic. The technique shown in this example is a popular way to eliminate clock skew problems. Clock skew is avoided by using a single oscillator as apposed to using multiple oscillators. A secondary clock enable is created by dividing down the system clock, as opposed to introducing additional clock signals, which could result in skew between the multiple clocks.

Figure 1.



In this example, assume that an FPGA contains some high-speed interface logic that must run at 40 MHz and some low-speed interface and core logic that must run at 10 MHz. The FPGA has a 40-MHz system clock and uses it to generate a 10-MHz enable signal for internal distribution. The following figure shows how the 10-MHz enable might align with the system clock when the rising edge of the 40-MHz system clock is the active edge. The 40-MHz clock is distributed to the clock input of each FPGA flip-flop, while the enable signal is distributed to each FPGA flip-flop clock enable input. In this case, the primary clock period is 25 ns, but the enabled signal and the subsequent logic will be driven by the slower 10MHz clocked data.

Figure 2.
Simple Shift register



A simple shift register circuit shown in the following logic diagram illustrates how the multi-cycle timing constraint is assigned in Foundation Express.

Register **reg1** is a 4-bit serial-input, parallel-output register. Register **reg2** is a holding register that is loaded with the clock enable **ena**. The paths from the output of **reg1** to the output of **reg2** (net **q**) are multi-cycle paths because the output of **reg2** is based on the combination of the **CLK** and **ENA** signals. The register-to-register timing constraint is 25 ns (1/40 MHz), but the multi-cycle timing constraint which applies to the subsequent logic, is 100 ns (4 x 25 ns).

Once the system clock has been set, **reg1**, **reg2**, and the subsequent logic will automatically be grouped together and displayed in the constraint entry window as having a register-to-register or clock-to-clock timing of 25 ns. This can also be referred to as the rising clock_of_clock to rising clock_of_clock of 25 ns. To create a subpath group of the register-to-register path, click the right mouse button on the register-to-register path groups and select **New Subpath**, in the Paths Constraint table.

The **Create/Edit Timing Subpath** window appears. Use this window to construct

your own path group by selecting specific start-points and end-points. The newly created path group is called a subpath because it is a subset of another path group, in this case, the register-to-register paths in the design.

In this example, the outputs of **reg1** (bus **q**) are the start-points, and the inputs of **reg2** are the end-point for the subpath. A delay of 100 ns is assigned to the subpath. After you create a subpath 'slow_path' and apply the multi-cycle timing constraint, the subpath appears in the Paths constraint table as *slow path-from to slow path-to.*

Notice that an enabled flip-flop can be included in two different path groups: those that include clock-to-clock paths and those that include clock-to-enabled-clock paths. This implies that there are two TIMESPECs with overlapping constraints generated by Foundation *Express.* The constraint for register-to-register timing, which is 25 ns in this case, conflicts with the constraint for the slow path subpath timing, which is 100 ns. The Xilinx implementation software assigns different priorities to these two constraints, placing a higher priority on the more specific one. Because the subpath constraint is more specific than the clock-to-clock constraint, it takes precedence and the corresponding paths can be optimized for the slower speed.

Summary

By prudently specifying aggressive timing constraints you can avoid over-constraining your design, which could result in excessive runtimes in the implementation tools. In this example only the data out of register 1 requires a clocking constraint of 25 ns. The multi-cycle timing of the subpath required a much more relaxed constraint of 100 ns. Because the bulk of the design was constrained to 100 ns rather than 25 ns, the implementation of this design is achieved with a reduced run time. ◆

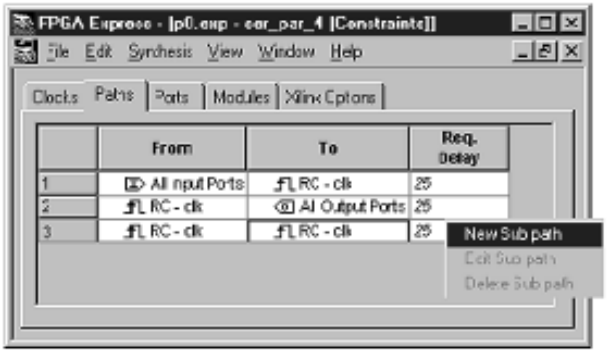


Figure 1:
In the Paths constraint table, to create a subpath group of the register-to-register paths, click the right mouse button on the register-to-register path groups and select New Subpath.

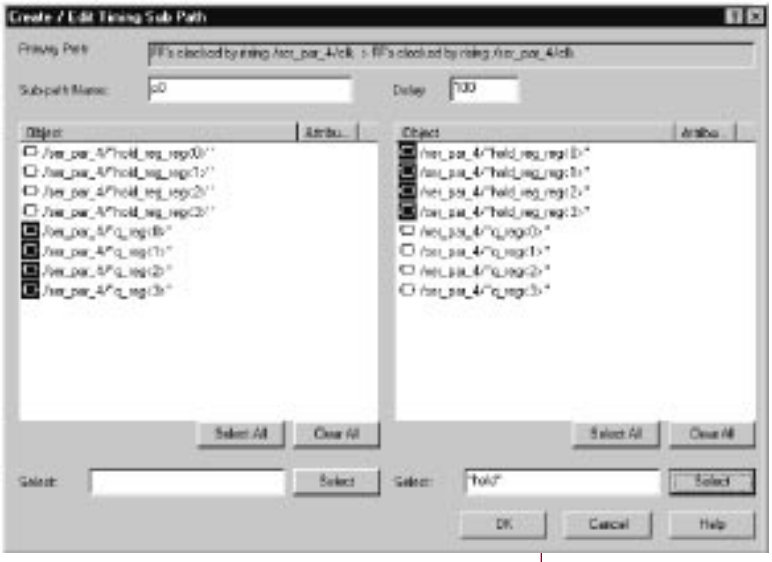


Figure 2:
The Create / Edit Timing Subpath window appears. Use this window to construct your own path group by selecting specific startpoints and endpoints. The newly created path group is called a subpath because it is a subset of another path group, in this case, the register-to-register paths in the design.

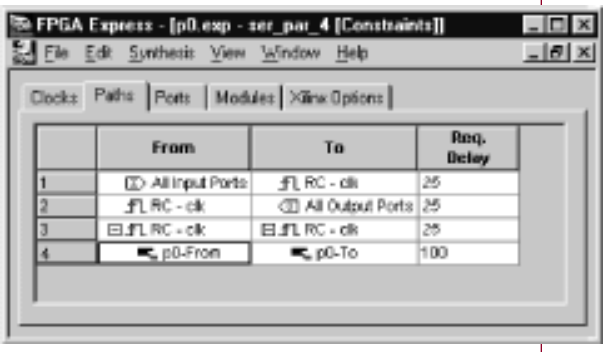


Figure 3:
In this example, the outputs of reg1 are the startpoints and the inputs of reg2 are the endpoint for the subpath. A delay of 100 ns is assigned to the subpath. After you create a subpath and apply the multicycle timing constraint, the subpath appears in the Paths constraint table.