WP388 (v1.0) March 1, 2011

Increased Productivity Using Team Design

By: Kate Kelley

Xilinx® FPGAs offer up to 2 million logic cells in capacity—and they continue to grow. Designs of this complexity usually require a team of developers, and often, a team leader, who is responsible for the synthesis and implementation of the entire design. To make matters more challenging, the developers can be located internationally, with different portions of the design developed in different locations, and even by different companies. The Xilinx Team Design flow introduced in ISE® Design Suite 13.1 focuses on solving these challenges.

Introduction

The Team Design flow has three major steps: the initial design setup, team member implementation, and assembly of all the team member modules into a final design. The initial design setup provides the framework for each team member to implement their portion of the design independently of other team members—but in context with the top-level design. At intervals during the design cycle, the entire design can be assembled by using the implementation results of each team member. See Figure 1.

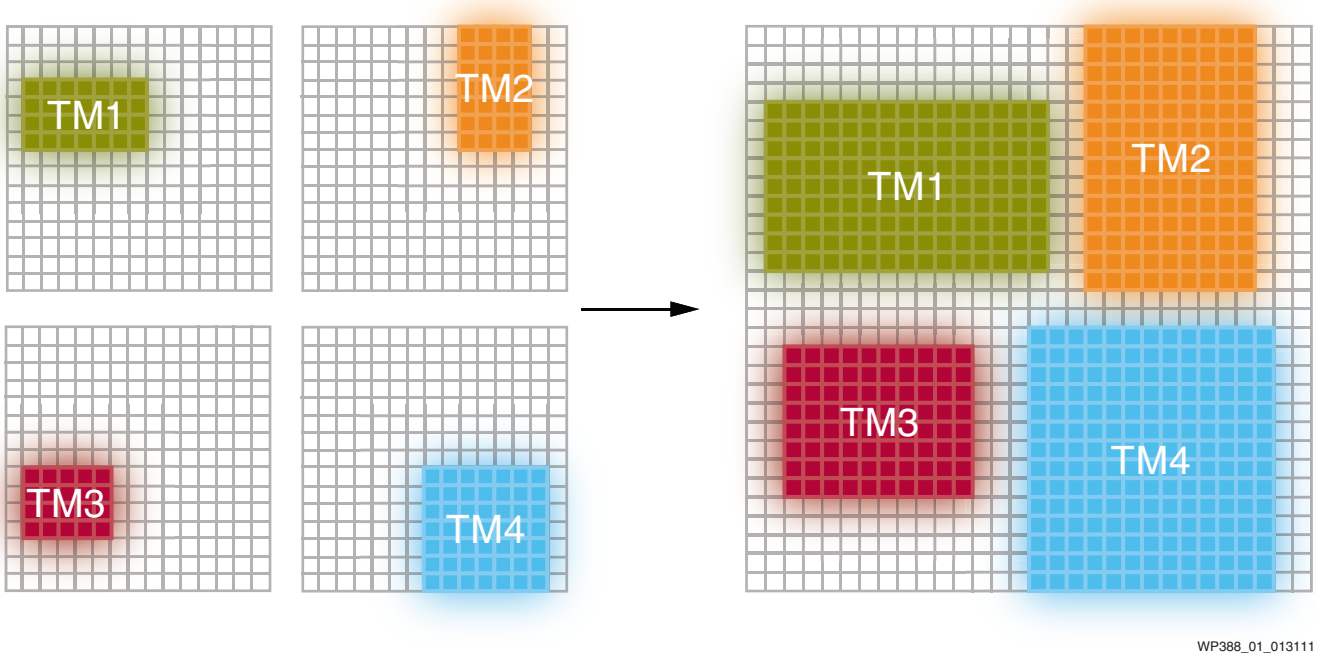

Figure 1: **Team Design Flow**

Initial Setup

The most important step of the flow is the initial planning stage. This step includes HDL design rules, design partitioning, synthesis setup, floorplanning, high-level timing requirements, directory structure for the team, initial implementation, and finally, the projects or work space required for each team member. The initial setup is typically done in the PlanAhead™ software. If desired, the project can be moved to a script-based flow.

HDL Design Rules

For the best quality of results (QoR) and utilization, certain design rules must be followed. The goal of these rules is to keep all the critical timing paths internal to each team member module for easier assembly. These rules are especially important if there are high QoR requirements between the modules:

- Inputs and outputs of modules should be registered with no combinatorial logic between the registers and module boundaries. This prevents a path that crosses a partition boundary from being critical because there is no logic optimization across boundaries. If it is not possible to register both inputs and outputs, it is better to register the outputs.

- Constants should not be used as inputs or outputs. Constants are not propagated across boundaries during synthesis. This can cause utilization and QoR issues if input constants are used to disable unused features of a design in a module. In the flat flow, unused logic is optimized away. This is not true for a hierarchical flow.
- All inputs and outputs must be used because the interface boundary is hard, unused inputs or outputs are not optimized away. This can lead to packing errors or higher utilization.

Design Partitioning

Not only is it critical to follow good design guidelines, but it is also important to determine the correct design hierarchy. These guidelines can be used to help with partitioning the design:

- Logic that needs to be optimized, implemented, and verified together should be in the same hierarchy. All designs have hierarchy, but to get the maximum performance, and because floorplanning is required, the hierarchy needs to be defined with the idea of the layout in the FPGA device. This might necessitate adding additional levels of hierarchy so that two critical modules can be synthesized and implemented together.
- Logic that needs to be packed together should be kept in the same level of hierarchy. This includes registers that need to be packed in larger components like block RAM and DSP. This guideline also affects I/O logic that needs to be packed together.
- An optimal number of team member modules should be selected. Because there are no optimizations across boundaries, too many team member modules can make floorplanning more challenging and affect performance. The optimal number of team modules is design dependent. For medium-sized designs, 4 to 10 might be ideal. Larger designs can have up to 20 team member modules.
- Team member modules should be pipelined if possible. Planning for extra cycles for signals between the modules can greatly simply the floorplan.

Synthesis Setup

Each team member module must be synthesized independently, creating a separate netlist to be implemented by the team member. This can be achieved with a bottom-up synthesis flow with a separate synthesis project file for each team member module. Alternatively, an incremental flow provided by Xilinx or third-party synthesis vendors can be used.

Floorplanning

Good design partitioning greatly simplifies floorplanning. Each team member module is required to be constrained to a specific location in the FPGA using an area group region. This constraint prevents any placement conflicts during the assembly stage. During floorplanning, the following should be considered:

- Clock regions: If possible, area groups should be aligned with the clock regions. This simplifies the clock planning, especially if there are many global and regional clocks in the design.
- Interfaces between modules: Modules with a high number of interfaces should be placed together.
- Pin placement: All device port locations should be defined.
- Global clock placement: All global clock components should be moved to a specific location for use by all the team members.

Global Timing Constraints

All clocks in the top level should be constrained. All I/Os should also have timing constraints. Any top-level timing exceptions, including multi-cycle or false path, can be defined in this step.

Directory Structure

The design directory structure must be defined so that each team member understands where the different files are located. The manner in which source control will be managed should also be defined.

Initial Implementation

The first implementation run should be carried out to verify the floorplan. For this run, the team member modules can be HDL, netlist, or a black box.

Team Member Projects

After the setup is finalized, each team member project or work space should be created. This can be done in the PlanAhead software or manually for a command line flow.

Team Member Synthesis and Implementation

After the initial setup is completed, each team member can synthesize and implement their portion of the design independently of other team member modules, but in context with the top-level design. The following flow choices are made during this step:

- Implementing or importing the top-level logic: This decision is design dependent and can change throughout the flow. Typically, the user implements the top-level design earlier in the design process and then might import it as the design starts to converge.
- Deciding if the other team member modules should be black boxes or whether existing logic should be imported: Existing logic can be imported to ensure that interface timing can be met. During the early phases of the design process, run time and memory requirements can be reduced by using black boxes.

Each team member can iterate on their block as needed and periodically export results for use by the team leader in an assembly run.

Design Assembly

At different stages in the design, the existing portions of the design can be assembled. This can be done at regularly scheduled intervals or when there has been a major update to one or more of the team member modules. When the assembly is done on a regular basis in the design stage, timing issues between modules can be found and fixed instead of finding major issues at the end of the design process. Assembly on a recurring basis also allows the team member to import the latest implemented version of other team member modules that are available after each assembly.

During assembly, the integrator (or team leader) imports any existing team member modules or uses black boxes if the design is not completed. The top-level logic,

including routes between the modules, can be imported from the initial setup stage or implemented. For best timing, any logic in the top level should be implemented. There should be no placement conflicts during import, but routing conflicts can occur. These can be resolved by backing off the preservation level on different modules. Although the default value of the preservation level is to preserve both the placement and routing, it can be changed to preserve only the placement, allowing routing changes. For even more flexibility, the placement can also be changed. As a last resort, any module can be re-implemented.

Conclusion

The Team Design flow allows multiple developers to work in parallel on one design. This has several advantages:

- Allows early implementation results on one to two finished modules.
  Engineers can start implementing their portion of the design without having to wait for the rest of the team.
- Facilitates the fixing of timing related issues.
  When a team member is working on meeting timing, they only have to implement their portion of the design. This limits the issues to the smaller portion of the design, reducing the run time and the overall number of issues.
- Provides run time reduction when making small changes to one module.
  After the design has converged, only the changed module needs to be implemented. The rest of the design can be preserved.

Related Documents

UG748, *Hierarchical Design Methodology Guide*

WP362, *Repeatable Results with Design Preservation*

Synopsys FPGA Synthesis User Guide

Synopsys FPGA Synthesis Reference Manual

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
03/01/11	1.0	Initial Xilinx release.

Notice of Disclaimer