

Implementing Incremental Changes Efficiently

Synplify Pro and Xilinx ISE software offer new SmartCompile methodologies.

by Angela Sutton
Sr. Marketing Manager
Synplicity, Inc.
sutton@synplicity.com

In May 2006, Synplicity and Xilinx formed the Ultra-High Capacity Task Force to develop improved methodologies for high-capacity designs. The first deliverable from this task force is a set of new incremental methodologies – SmartGuide™ technology and Partitions technology. These methodologies are available with Xilinx® ISE™ software version 9.1i and Synplicity's Synplify Pro v8.8/v8.8.1 and beyond. The two are often jointly referred to as "SmartCompile™" technology.

These methodologies are particularly useful for high-capacity FPGAs where the large size of the design results in long iteration times or for stages of the design cycle when you want to make small incremental changes to localized portions without upsetting other pieces of the design that already work. Table 1 summarizes the new methodologies; in this article, I'll provide details about how to use them.

Incremental Methodology	When to Use	How it Works	Advantage
SmartGuide	Small incremental RTL changes to non-critical paths	When you make a small change to your RTL and re-run synthesis. ISE software detects netlist changes and runs incremental PAR based on those netlist changes alone and timing goals.	Design preservation and consistency from one run to the next Saves PAR runtime (up to 70%) Easy to use – no need to create partitions or block-based constraints
Partition	Useful for block-based or team-designed methodologies. Allows you to get more stable results from one run to the next by locking down individual blocks.	Fully integrated block-based design methodology, synthesis through place and route. Blocks (“partitions”) defined at RTL level in Synplify Pro. Block definitions and subsequent block changes automatically communicated between Synplify and ISE software.	Teams can work on and tune specific RTL blocks in parallel Blocks that already work can be preserved

Table 1 – New Synplicity/Xilinx incremental methodologies

SmartGuide

SmartGuide minimizes changes in a design’s place and route (PAR) implementation compared with previous implementations used as a reference. Typically, you would synthesize the design to get an output netlist, which ISE software places and routes. You would then make changes to the RTL, rerun synthesis, and rerun ISE incremental place and route with SmartGuide enabled.

If you elect to use SmartGuide, ISE software place and route attempts to do the minimal changes based on what changed in the netlist, while still maintaining a legal netlist and meeting timing. This saves significant PAR runtime. In this methodology, there is no need to manually create partitions, nor are constraint or synthesis changes involved when using this capability (Figure 1).

How Does SmartGuide Work?

ISE software runs fully automatic incremental place and route on an entire design based on a netlist “name match” comparison (the current netlist output by Synplicity’s Synplify Pro tool versus the ini-

tial netlist). The Synplify Pro software’s ability to generate consistent and repeatable netlists has been key for this to work well. Synplify Pro software v8.8 employs many new techniques to maximize netlist consistency. These include:

- Repeatability of instance names from one synthesis run to the next
- Output netlist changes that are localized to the very specific paths where the RTL changed from one synthesis run to the next



Figure 1 – SmartGuide can halve your place and route runtime. Small changes to RTL/constraints result in small changes to the physical layout, especially when the changes were not on the critical path.

Xilinx results with Synplify Pro v8.8 software and ISE software version 9.1i showed that, for small RTL changes where the change was not on the critical path, place and route runtimes were halved and in many cases as much as 70% shorter. More than half of the test designs preserved 97% of placement and routing, leading to very consistent designs.

SmartGuide is best used when you need to make small changes to your RTL and those changes are not on a critical path. In such cases, SmartGuide can cut PAR time in half on average compared with running place and route again on the entire design.

Examples of a small changes include changing things that result in less than 10% of the netlist changing – a change in the value of an RTL constant, changing a logical OR to an AND, changing the logic condition on a “if” statement, or minor changes to a state machine. Also, this flow is recommended for roughly 10-15 consecutive incremental runs, after which you may want to rerun place and route on the entire design.

Partition

Partition is an incremental block-based design methodology – it allows place and route to run incrementally on blocks that are being modified or tuned while preserving other blocks that have not changed. This preservation can occur all the way down to the routing level. This approach results in shorter runtimes.

Blocks, also referred to as “partitions,” are defined at the RTL level before running synthesis. The block hierarchy is specified

ISE software and Synplify Pro have been integrated so that ISE software automatically determines which blocks/subblocks changed using partition date-stamp information.

as Synplify Pro compile points. Each time a block is re-synthesized it is date-stamped and updated by Synplify Pro in the output EDIF netlist and communicated to ISE software. ISE software automatically detects which blocks have changed by reading the EDIF time stamp.

To run the partition methodology (illustrated in Figure 2):

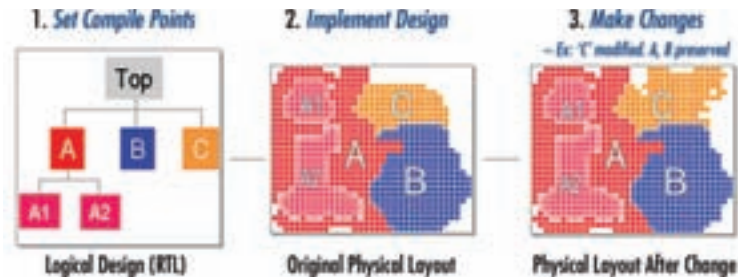


Figure 2 – The Synplify Pro/ISE Partition is a fully integrated (RTL through physical layout) block-based design methodology.

1. Set compile points. Define blocks/sub-blocks (partitions) as compile points in Synplify Pro. For example:

```
#
# Define Compile points in Synthesis .sdc file.
# You need to separately define time budgets
#
define_compile_point {v:work.or1200_cpu}
-type {locked, partition} -cpfile {}
define_compile_point{v:work.or1200_dc_top}
-type {locked, partition} -cpfile {}
```

You can specify your compile points in the GUI using the SCOPE editor or the Synplify Pro command line.

2. Implement the design (original PAR layout) and run synthesis. At the end of synthesis, Synplify Pro writes an extra property into the EDIF that includes the time stamp for when the compile point (block/partition) was last modified. For example:

```
(instance or1200_dc_top (viewRef verilog
(cellRef or1200_dc_top))
(property PARTITION (string "1169730682"))
)
```

In subsequent synthesis runs, the old time stamp will be inserted in the EDIF for blocks/partitions that did not change. A new time stamp is inserted in blocks/partitions that did change.

You can then run ISE place and route on the output from Synplify Pro software. ISE software automatically reads

the Synplify Pro compile points as partitions. ISE software also notes and stores the EDIF date stamp for each block/partition.

3. Make changes. Tune individual blocks (for example, RTL or constraints) in Synplify Pro and resynthesize, then rerun ISE place and route. ISE software and Synplify Pro have been integrated so that ISE software automatically determines which blocks/subblocks changed using partition date-stamp information. ISE runs place and route on the changed partitions and leaves other placed/routed blocks untouched (as long as timing can still be met).

You can control whether ISE software leaves placement alone on the unchanged blocks or both placement and routing using the partition property setting. Each partition has a customizable level of preservation with four values: “synthesis” (preserve the netlist), “placement” (preserve the netlist and placement), “routing” (preserve netlist, placement, and routing), and “inherit” (use in sub-blocks only when the preservation level of the sub-block must be the same as that of its parent block). The default setting is “routing,” for which the synthesis netlist, placement, and routing will be exactly preserved

from one run to the next (provided that the RTL within the compile point block is unchanged).

In Synplify Pro software, individual block results can be timing-driven. To get the best quality of results within a block, it is usually better to create timing constraints for each compile point individually.

If you change a top-level constraint in Synplify Pro software, only the impacted partitions are marked as changed (and therefore resynthesized).

When Synplify Pro’s partition methodology is used, you can create multiple instances of a module.

Conclusion

The latest generation of 65-nm FPGAs has enabled complex system-on-chip designs to be implemented on multimillion-gate-capacity FPGAs. As FPGA designs grow in capacity, it becomes increasingly important to deliver flows that are fast and flows that converge. Users are looking for better “divide-and-conquer” approaches to quickly tune their designs and assimilate small design changes.

Synplicity and Xilinx formed a joint Ultra-High Capacity Task Force to tackle this very challenge. SmartGuide and Partition are the first of several new solutions that Synplicity and Xilinx will be delivering to provide users with more efficient design methodologies. ●●