


New Tools Take the Pain Out of FPGA Synthesis



How to use RTL analysis, SDC constraints and guided synthesis to create better FPGA designs faster.

by Shakeel Jeeawoody

Director of Marketing
Blue Pearl Software Inc.
shakeel@bluepearlsoftware.com

Most FPGA designers are passionate about their work and thrive on the problem-solving and creative aspects of the profession. The job does, however, come with a fair amount of stresses and its share of monotonous tasks. Luckily, EDA companies and FPGA vendors are constantly developing new tools and methods that automate mundane tasks so design teams can focus on what they do best—being creative. Let's take a look at the evolution of FPGA tool flows and see how modern FPGA teams are using new tools for RTL analysis, constraint generation and guided synthesis to reduce design iterations.

If you are already an FPGA design specialist, you certainly have a bright future ahead of you, because an increasing number of designs that would have typically been implemented in an ASIC are now being implemented on FPGAs. Designing ASICs is becoming exponentially more expensive with each introduction of a new silicon process technology. Meanwhile, FPGA vendors implement each new generation of their devices on these new process technologies but do not burden customers with exorbitant costs.

The bad news, however, is that FPGA designs can be so complex that they require tool flows as advanced as ASIC flows and often demand the efforts of a design team rather than a single designer. As such, FPGA design teams need to seriously analyze their current tool sets as they embark on ECOs or new projects. The good news? There are numerous new-generation EDA tools available to help them. Designers can choose easy-to-install and easy-to-use products that employ standard data formats, which are simpler to integrate in a flow and run natively on their platform of choice, whether Windows or Linux.

THE EVOLUTION OF FPGA TOOL FLOWS

As FPGA designs have become more complicated over the years, the tool flows have evolved accordingly and become more ASIC-like. In the 1990s, FPGA flows (see Flow A in Figure 1) became RTL based and used synthesis and place-and-route tools, just like simple ASIC flows of the time. As designs got more complex, FPGA teams added timing analysis to their flows to help customers ensure that designs could perform at the required

frequency. Today’s FPGAs have become giant system platforms, so now it’s commonplace for design teams to employ RTL analysis to minimize design iterations and ensure their designs achieve performance goals.

Further, because today’s FPGA design projects can be so large and complex, designers need ways to better understand the scope and complexity of their design and to better control the tools in their flow to get their designs to market quickly. An emerging way that modern FPGA design teams do this is by using constraints throughout their design flows. Let’s take a look at one of the most popular constraint methodologies, the Synopsys Design Constraint (SDC) format, now supported in the new Xilinx® Vivado™ flow, and at how you can use SDC to benefit your design projects.

WHAT IS SDC?

SDC is a TCL-based format used to specify design intent, including the timing, power and area constraints for a design. There are several products that either read or write SDCs. Some sample SDC constraints include timing constraints, such as create_clock, cre-

ate_generated_clock, set_input_delay and set_output_delay; and timing exceptions, such as set_false_path, set_max_delay, set_min_delay and set_multicycle_path. These SDC constraints are typically applied to design objects like registers, clocks, ports, pins and nets.

It must be noted that even though SDC is a standard format, there are slight variations (among different tools) between generated and read SDCs. Understanding these minor variations and dealing with them in a timely manner helps to avoid surprises.

SDC SHOULD NOT IMPLY PAIN

One of the most popular uses of SDCs is to constrain synthesis. In general, designers have a feel for what areas of their design need to be constrained and thus start by writing SDCs for them. They would typically execute the flow described in Flow B and inevitably not close timing the first time around. It then becomes an iterative, manual, shooting-in-the-dark process of adding SDCs to close timing or get the design to work at the desired frequency. Many designers who have executed this process complain about spending

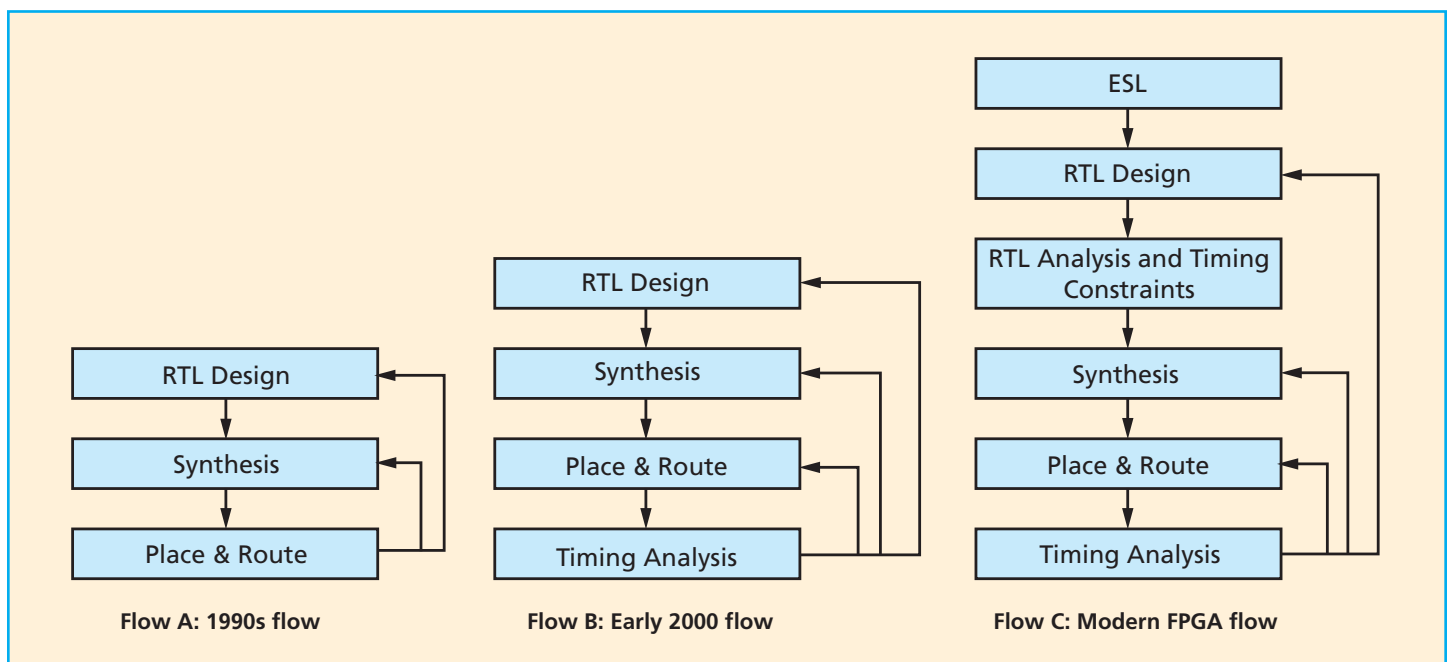


Figure 1 – FPGA tool flows have evolved over time to become more ASIC-like.

weeks iterating the design, often missing schedules in the process.

Another issue associated with the iterations is that several designers working on different blocks and possibly at different locations are contributing to the SDCs. This process more often than not becomes so complex that design teams need a methodology to verify the SDCs and remove conflicts in hierarchical names or pins as they assemble the design at the chip level. It is very important to have the right tools and methodologies for collaborative design to be effective.

The modern flow described in Flow C, which uses analysis, SDC constraints and high-level synthesis in addition to the tools in Flow B, makes great strides at solving both of these issues.

GUIDING SYNTHESIS

For a typical FPGA design, the synthesis solution space, being heuristic, has multiple local maxima and minima, irrespective of whether we are optimizing for area, speed or power. By using smart guidance, we can achieve an optimal solution rather than having the synthesis tool converge on an arbitrary local minimum. One of the most effective guidance techniques includes the usage of false paths and multicycle paths that keep the synthesis tool from spending precious optimization time on unnecessary elements.

However, finding all the false paths (FPs) and multicycle paths (MCPs) in a design is not a trivial task. You'll find some simple FPs and MCPs if you spend enough time looking for them, but complex ones involving state machines and counters (especially within multiple levels of hierarchy) are nearly impossible to ferret out. Fortunately for FPGA designers, innovative companies like Blue Pearl Software have tools to perform automatic FP and MCP generation that is complete, comprehensive and accu-

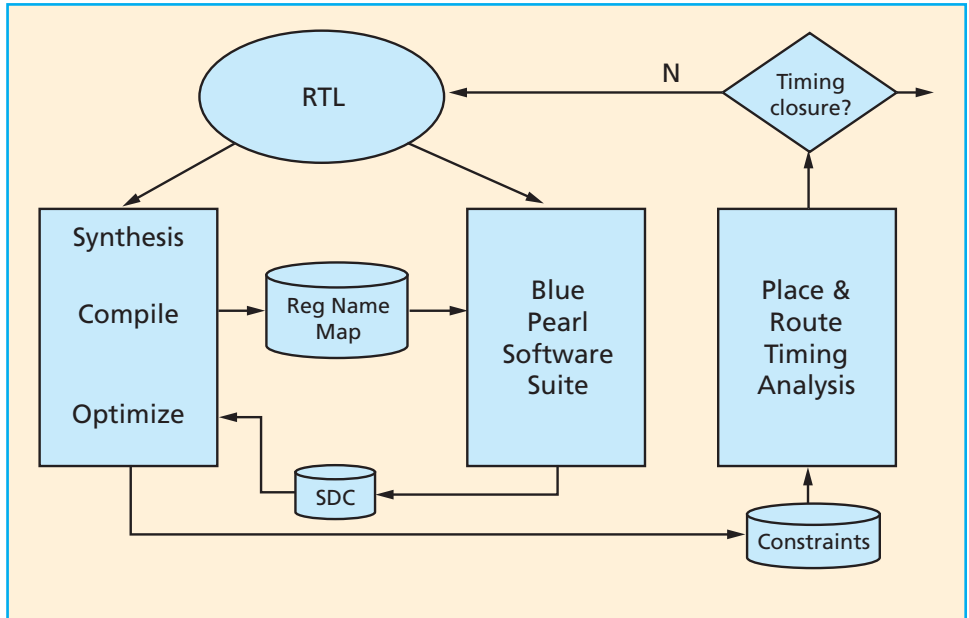


Figure 2 – How the Blue Pearl Software Suite and Vivado work together

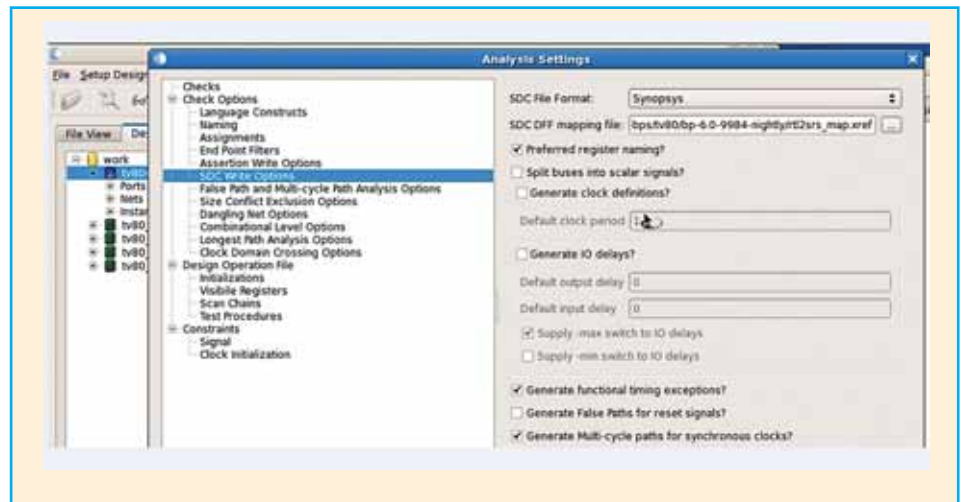


Figure 3 – The Blue Pearl software simplifies formatting issues.

rate. Furthermore, these tools provide different mechanisms for each FP and MCP—including schematic, assertion and audit trail—by which the user may verify its correctness.

Because FPGA and commercial EDA vendors have been increasingly working together and using common interfaces, design teams can integrate the Blue Pearl Software Suite into the flow they like the best. Since Xilinx's new Vivado Design Suite supports SDC, it is very simple to communicate design intent between the tools (Figure 2).

In addition to working with Xilinx and other FPGA vendors, Blue Pearl also closely collaborated with Synopsys. The two companies came together to investigate what needs to be done so that the synthesis tool can accept a maximum number of the automatically generated SDCs, without forcing the designer to perform any manual changes. Due to minor differences in different tools' usage of the SDC format, the team quickly identified naming convention as a major challenge for smooth interoperability.

The Blue Pearl Software Suite delivers a 20 percent QoR improvement in our sample design, which contains multiple IP cores including an R1200 in Verilog and AES encryption in VHDL.

The solution involved intercepting the name mapping that happens after the first phase (compilation) of synthesis, to use the names (see Figure 3) within the Blue Pearl Software Suite SDC generation tool, and then providing the proper SDC to the second phase (optimization) of the synthesis tool. This approach gives the FPGA designer an optimized solution without having to waste time with formatting issues.

For example, a nonoptimized constraint that reads:

```
set_false_path -from
[get_cells
{i_tv80_core.SP[*}}] -to
[get_cells
{i_tv80_core.i_reg.Regsl}]
```

might be optimized to read:

```
set_false_path -from
[get_cells
{i_tv80_core.SP[*}}] -to
[get_cells
{i_tv80_core.i_reg.Regsl_2[
7:0}}]
```

WHAT ABOUT THE TANGIBLE BENEFITS?

Even though the Blue Pearl Software Suite automates several tasks, designers will be pleased with its quality of results (QoR). Table 1 shows that using the automatically generated SDCs from the Blue Pearl Software Suite delivers a 20 percent QoR improvement for this sample design,

which contains multiple IP cores including an R1200 in Verilog and AES encryption in VHDL.

Run 1, without the Blue Pearl software, did not achieve timing closure; the designer could easily spend weeks iterating through the RTL design or tool constraints to meet the 60-MHz requirement. In Run 2, the Blue Pearl Software Suite generated the SDCs in minutes and the automatically generated SDCs were sufficient to guide the downstream tools to meet timing.

Clearly, for an FPGA designer, one way to reduce stress and simplify your life is to learn from others and add RTL analysis, SDC generation and guided synthesis tools to your tool box. For more information, visit www.bluepearlsoftware.com.

	Run 1	Run 2
Flow	Like Flow B above	Like Flow C above
Tools involved	Synopsys: Synplify Pro Xilinx: Virtex-5 , XC5VLX50, FF1153,-1	Synopsys: Synplify Pro Xilinx: Virtex-5 , XC5VLX50, FF1153,-1 Blue Pearl Software: Release 6.0
Design frequency	Set to 60 MHz globally from Synplify Pro	Set to 60 MHz globally from Synplify Pro
Blue Pearl SDC	NO	YES
Setup violation (after Place & Route)	-3.57 ns	NONE

Table 1 -- Comparing two runs shows an advantage in adding the Blue Pearl suite to the flow.