by Mahadevan Ramasame, Technical Marketing Engineer, Alliance Series, mahadeva@xilinx.com

The Basic Elements of HDL Simulation

This article introduces the basic facts and terminology of HDL simulation for FPGAs and CPLDs, to help you simulate your design more efficiently.

There are three stages in the FPGA design process in which you conduct simulation:

➤ **Register Transfer Level** - To verify the syntax and functionality without the timing information. The majority of the design development is done through repetitive RTL simulation until you get the required functionality. Errors identified early in the design cycle are inexpensive to fix compared to functional errors identified during silicon debug.

➤ **Gate-level Functional Simulation** – After the RTL simulation is error free, the HDL design is synthesized to gates. The post-synthesized gate-level simulation is a functional simulation with unit delay timing. The simulation can be used to identify initialization issues and to analyze don't care conditions. "The don't care space of a design may be larger than the functional space," says Michael Bohm, VP and Chief Scientist at Exemplar Logic. The post synthesis simulation generally uses the same testbench as functional simulation.

➤ **Gate-level Timing Simulation** – Gate-level timing simulation is a back-annotated timing simulation. Timing simulation is important in verifying the operation of your circuit after the worst case place and route delays are calculated for your design. The back annotation process produces a netlist of library components annotated in an SDF file with the appropriate block and net delays from the place and route process. The simulation will identify any race conditions and setup-and-hold violations based on the operating conditions for the specified functionality.

"The top-down design method refers to applying a single optimization at the top level of your design; the bottom-up design method refers to performing individual optimizations on sub-blocks of your design."

Design Techniques for Better Simulation Results

Design techniques are used in the process of applying optimizations to an FPGA design. The top-down design method refers to applying a single optimization at the top level of your design; the bottom-up design method refers to performing individual optimizations on sub-blocks of your design.

To improve the quality of the results of the simulation, use the following guidelines for design partitioning:

➤ Limit gate counts in sub-blocks; 10k to 50k gates.

➤ Limit clocks to one per block.

➤ Group similar logic together, such as state machines, data path logic, decoder logic, and ROMs.

➤ Partition state machines into separate blocks of hierarchy.

➤ Separate timing-critical blocks from non timing-critical blocks.

These features eliminate any ambiguity in your design by providing better quality simulation results. You can also add other features to the design such as breaking the asynchronous feedback loops, and design stitching to build the entire design after optimizations have been performed on individual subblocks. You can also unfold the netlist to perform two different optimizations (such as area or delay) on two different instances of common sub-blocks. These features can help improve the quality of simulation results.

Simulation Libraries

The following libraries are available for the Xilinx simulation flow:

➤ **UNISIM Library** - Used for functional simulation and contains default unit delays. This library includes all of the Xilinx Unified Library components that are inferred by most popular

for FPGAs and CPLDs

synthesis tools. The UNISIM library also includes components that are commonly instantiated such as I/O's and memory cells. You can instantiate the UNISIM library components in your design (VHDL or Verilog) and simulate them during the RTL simulation. The HDL code must refer to the compiled UNISIM library. The HDL simulator must map the logical library to the physical location of the compiled library.

➤ **LogiBLOX and Coregen Library** - LogiBLOX is a module generator used for schematic-based design entry of modules such as adders, counters, and large memory blocks. LogiBLOX can be used in the HDL flow to generate large blocks of memory for instantiation. LogiBLOX components are simulated with behavioral code. They are not intended to be synthesized, but they can be simulated. Coregen library models are high level VHDL behavioral or RTL models that are mapped to SIMPRIM structural models in the back-annotated netlist. The behavioral model is used for any post-synthesis simulation because synthesis processes these modules as a black box.

➤ **SIMPRIM Library** - Used for simulations at the following steps in the design flow:

- RTL simulations that include instantiated LogiBLOX modules.
- Post-implementation simulations.
- Timing simulation.

LIBRARY COMPILATION

The UNISIM libraries are used for RTL and post-synthesis simulations. Because industry standard simulators like ModelSim use pre-compiled libraries, Xilinx recommends compiling the UNISIM components that are instantiated in the current design. The UNISIM VHDL or Verilog Library can be compiled to any physical location. The order in which the VHDL source files for the UNISIM library must be compiled is listed in the Xilinx simulation design guide.

The LogiBLOX library is not a library of modules. It is a set of packages required by the LogiBLOX models that are created on-the-fly by the LogiBLOX tool. The source libraries for the LogiBLOX packages must be compiled into a library named LogiBLOX. These packages are available separately for VHDL and Verilog designs. The component model from the LogiBLOX GUI should be compiled into your working directory with your design.

The Simprim VHDL or Verilog Library can be compiled to any physical location and can be named Simprim.

VHDL Simulation

The Xilinx simulation flow supports the VHDL language standard IEEE-STD-1076-87 and the standard logic package IEEE-STD-1164-93. In VHDL designs, you must declare as ports any signals that are simulated or monitored from outside a module. Global GSR and GTS signals are used to initialize the simulation and require access ports if controlled from the testbench. The addition of these ports makes the pre and post implementations of your design different and your original testbench is no longer applicable to both versions of your design.

However, it's usually a good idea to get a pre-route VHDL description (used in gate-level functional simulation) that can be used for functional simulations with the GSR and GTS characteristics that match post-route results (gate-level timing simulation). This enables you to predict your design description accuracy at an earlier stage and reduces your design modification after place and route; therefore, this reduces your total design time. This is achieved by the addition of new library cells to simulate the GSR/GTS behavior.

Global Signal Methodology

To match the simulation behavior at all the three stages in the FPGA design, add a behavioral representation for GSR and Xilinx implementation directives. This directive is used to specify, to the place and route tools, the use of the special purpose GSR net that is pre-routed on the chip, and not to use the local asynchronous set/reset pins. Hence it utilizes the existing routing resources and significantly improves the performance. The

Continued on the following page

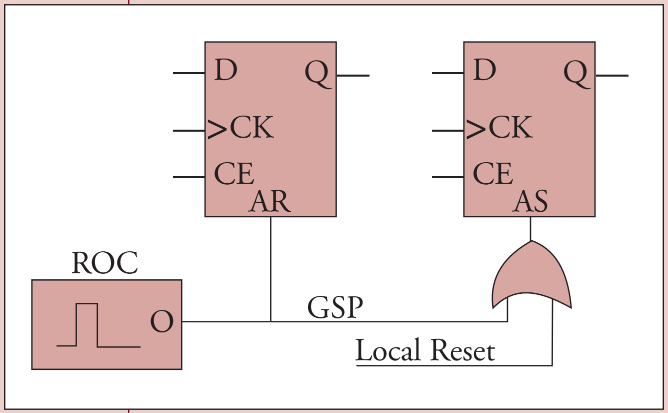
Continued from the previous page

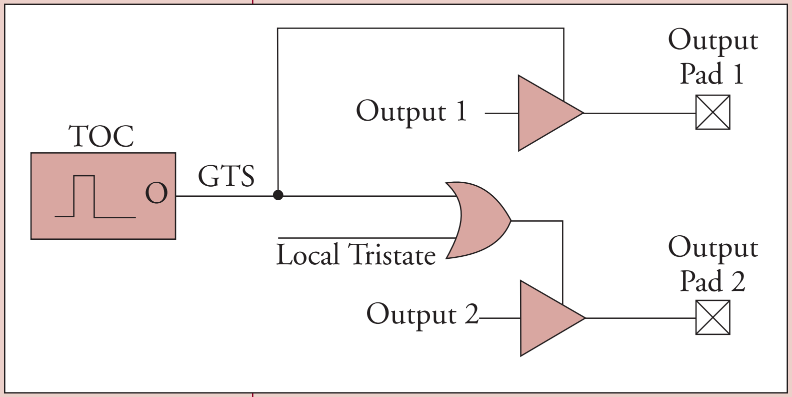
new library cells introduced have both the behavioral representation and the implementation directives. The new library cells are:

- ROC – Emulates the reset on configuration pulse.
- ROCBUF – Allows the test bench to drive the chip-generated reset on configuration without implementing an actual input pin on the chip.
- TOC – Emulates the chip-generated 3-state on configuration pulse.
- TOCBUF – Allows the test bench to drive the chip-generated 3-state on configuration without actually implementing the actual input on the chip.
- STARTBUF – A technology-independent version of the STARTUP block supported for simulation.

These five cells allow you to control the global reset and 3-state signal emulation, so you can get pre-route initialization simulations to match post-route simulations. The cells also drive implementation tools to add or delete pins and also help in the selection of nets for routing.

Model of ROC for Functional Simulation





Model of TOC for Functional Simulation

Models of the ROC and TOC cells, used for functional simulation, are given below.

Xilinx VHDL simulation supports the VITAL modeling standard IEEE-STD-1076.4 – 95 and Standard Delay Format version 2.1. This standard allows you to simulate your designs with any VITAL-compliant simulator and hence it accelerates your design compilation times, resulting in improved performance.

These features allow you to do high performance designs with shorter design cycles.

Verilog Simulation

The Xilinx simulation flow supports the Verilog language standard IEEE-STD-1364-95. The Verilog version of the UNISIM library may not need to be compiled, depending on the Verilog tool. Because there are a few cells with functional differences between Xilinx devices, a separate library is provided for each supported device. The libraries are in uppercase only and if needed, lower case libraries are provided in Xilinx/Cadence interface.

Unlike VHDL, Verilog can simulate internal signals and these signals are driven directly from the testbench without instantiating any specific component. The global set/reset net is present in your implemented design even if you do not instantiate the STARTUP block in your design. The function of STARTUP is to give you the option to control the global reset net from an external pin.

The general procedure for specifying the global set/reset during a post-synthesis Verilog UNISIM simulation involves defining the global reset signals with suitable Verilog macros. This is necessary because these global nets do not exist in the UNISIM libraries and as a result, the reset of UNISIM components is controlled by the detection of those macros. Also, the global set/reset signals need to be declared as either a **wire** or **reg** and the choice depends on whether the design contains a STARTUP component or not.

At the beginning of an FPGA design simulation, the global set/reset signal or the GR global reset signal must be toggled to emulate the power-on reset of the FPGA. This is to ensure that the flip-flops and latches in your simulation function correctly. The general procedure for specifying GTS is similar to that used for specifying the global set/reset signals, GSR and GR.

Testbenches

A testbench is a separate set of VHDL or Verilog code that connects to the inputs and outputs of a design. The testbench has two main purposes:

- It provides the stimulus and response information (clocks, set/reset, input data, and so on) that the design will encounter when it is implemented in an FPGA and installed into the final system.
- The testbench contains regression checking constructs, which allow design functionality to be tested throughout the FPGA HDL Simulation flow (RTL, Functional Gate, and Timing Gate).

A SAMPLE TESTBENCH MODEL:

```
library declarations ;
entity sample of testbench is
end ;
architecture test of testbench is
 instantiation of a component of the design ;
 signal declarations ;
begin
 port mapping of the component to the signals
 declared ;
process
     begin
                    clock period declaration ;
end process ;
process
     begin
                    Test vectors for design ;
end process ;
end test ;
configuration statement to configure
 architecture to the component
instantiation; (optional )
```

Use the steps for simulation in an industry standard simulator such as ModelSim:

- Create a working library.
- Compile the RTL/post-synthesis/place and route HDL design.
- Compile the testbench.
- Simulate the testbench and design. For place and route HDL design, simulate the testbench and design, with timing information.
- Run until the testbench stops.

You can create the testbench using a particular coding style for supplying the stimulus by referring to the synthesis or simulation vendors' documentation.

Conclusion

Xilinx offers a wide range of FPGA and CPLD solutions, including large density devices and low cost devices. You can successfully implement and correlate global initialization behavior of user-defined logic-, LogiBLOX-, and CORE Generator-based designs at all simulation phases, from RTL to back-annotated netlists. ◆

“Xilinx offers a wide range of FPGA and CPLD solutions, including large density devices and low cost devices. You can successfully implement and correlate global initialization behavior of user-defined logic-, LogiBLOX-, and CORE Generator-based designs at all simulation phases, from RTL to back-annotated netlists.”