FPGA System Simulation and Synthesis Using Synopsys VCS and FPGA Compiler II

This HDL design methodology can help you use the largest Virtex FPGAs with a minimum amount of time spent on synthesis, simula tion, and verification.

by Srikanth Vijayaraghavan, Applications Consultant, Synopsys,
raghavan@synopsys.com

Synopsys has combined its powerful logic synthesis technology with innovative architecture-specific optimization technology to address the needs of FPGA designers who are now adopting an HDL methodology. FPGA Compiler II delivers powerful architecture-specific synthesis capabilities with features such as behavioral re-timing and pipelining capability. Synopsys VCS (Verilog Compiled Simulator) provides a fully-featured implementation of the verilog language as defined in the IEEE Standard Hardware Description Language (IEEE Std 1364-1995).

VCS is specifically designed to simulate large, complex designs faster than any other Verilog-HDL simulator. VCS supports interfaces to a variety of other simulators and models, including (but not limited to) user PLI applications conforming to IEEE Std 1363-1995, delay calculators, SDF delay annotation, LMG Smatmodels, and the LMSI hardware modeler.

The combination of the FPGA Compiler II synthesis tool and the VCS simulator provides a simple and accurate design and verification flow that significantly reduces your total development time.

Detailed Design Flow

The steps involved in taking a design from an RTL description to a production FPGA are summarized in Figure 1. These steps are explained in detail using the RTL description of a flip-flop shown in Example 1. After synthesizing the RTL code for this flip-flop using FPGA Compiler II, you can automatically generate a functional Verilog simulation netlist as shown in Example 2. FPGA Compiler II is capable of synthesizing the design, either by flattening the design completely, or by preserving the complete hierarchy.

```
Module reg1 (clk,reset,din,dout);
input clk,reset;
input  din;
output dout;

wire clk,clk_enbl,reset;
wire  din;
reg  dout;

always@(negedge clk or posedge reset)
if (reset)
                        dout = 0;
else
                        dout = din;
endmodule
```

Example 1 - RTL description of a simple flip-flop.

```
// Synopsys FPGA Compiler II
// automatically generated file
// Author: raghavan
// Program:FPGA Compiler II
// Version:3.2.0.4206

module FDC_1 ( Q ,D ,C ,CLR );
    output Q ;
    input D ;
    input C ;
    input CLR ;
    wire synch_enable ;
    reg Q ;
    always@( negedge C or posedge CLR )
    begin
        if ( CLR ) Q = 1'b0;
        else Q = ( D );
    end
assign synch_enable = ( 1'b1 );
endmodule

module IBUF ( O ,I );
    output O ;
    input I ;
    assign O = ( I );
endmodule

module OBUF_S_12 ( O ,I );
    output O ;
    input I ;
    assign O = ( I );
endmodule

module reg1 ( clk ,reset ,din ,dout );
    input clk ;
    input reset ;
    input din ;
    output dout ;

    wire N_clk ;
    wire N_reset ;
    wire N_din ;
    wire N_dout ;

FDC_1 dout_reg (.CLR (N_reset),.Q (N_dout),.C (N_clk),.D (N_din));
IBUF C_clk (.I (clk),.O (N_clk));
IBUF C_reset (.I reset),.O(N_reset ) );
IBUF C_din (.I (din),.O (N_din));
OBUF_S_12
C_dout (.I (N_dout),.O (dout));

Endmodule
```

Example 2 - Post synthesis functional netlist generated by FPGA Compiler II.

The netlist can be simulated in VCS like any other Verilog file. FPGA Compiler II maintains the port names at the module level and therefore debugging through the hierarchy is very easy.

Assuming you have a testbench for the design module named test_reg1.v, a simple command line script for simulation in VCS will look as follows:

```
vcs -RI -Mupdate reg1.v test_reg1.v -o reg1.simv -l reg1.log
```

Where:

- **-RI** is for simulating and bringing up the XVCS Debugging debugging GUI automatically.
- **-Mupdate** is to enable incremental compile.
- **-o** is to provide a distinct name for the executable file; default name is simv.
- **-l** is to provide a distinct name for the log file produced during compile.

Once the functionality of the design is verified, the EDIF equivalent to this Verilog netlist is generated using FPGA Compiler II. The timing constraints entered in the FPGA Compiler II GUI can be exported into a spec file (.ncf), which is understood by the Xilinx software tools. The Xilinx software uses this EDIF netlist and the .ncf constraint file to place and route the design.

If the design routes successfully, you can write a Verilog simulation netlist in terms of gate cells, and also a delay file (.sdf, Standard Delay Format). A part of the delay file that corresponds to the gate level netlist is shown in Example 3. This .sdf file should be included during simulation to back annotate the original delays into the design.

The delay values need to be back annotated through the PLI interface in VCS. The .sdf file is called from the gate-level netlist through the “$sdf_annotate” utility. For performing a gate-

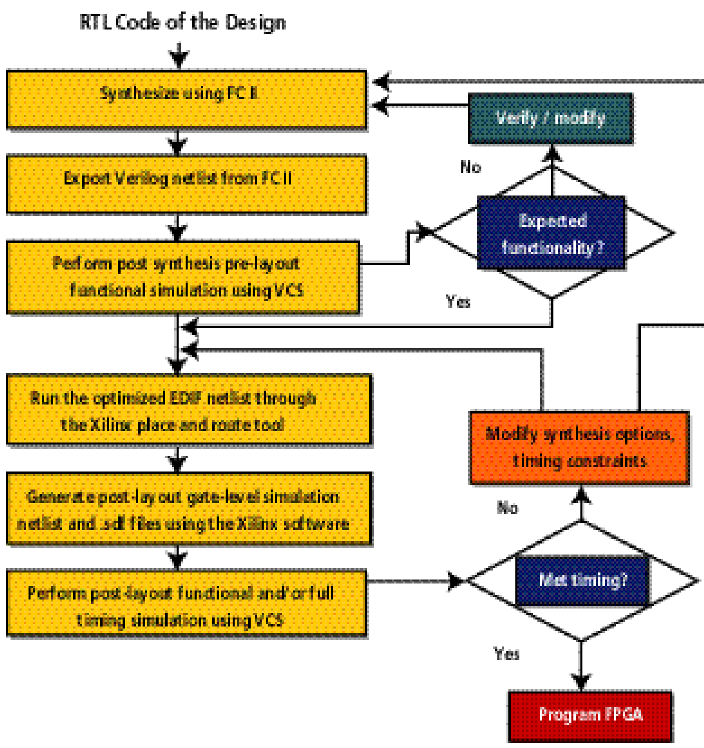
RTL Code of the Design
Synthesize using FC II
Export Verilog netlist from FC II
Perform post synthesis pre-layout functional simulation using VCS
Run the optimized EDIF netlist through the Xilinx place and route tool
Generate post-layout gate-level simulation netlist and .sdf files using the Xilinx software
Perform post-layout functional and/or full timing simulation using VCS
Verify / modify
No
Expected functionality?
Yes
Modify synthesis options, timing constraints
No
Met timing?
Yes
Program FPGA

Figure 1 - Detailed design f.low for Xilinx FPGA's

```
(TIMESCALE 1 ps)
  (CELL
    (CELLTYPE "X_FF")
      (INSTANCE dout_reg)
        (DELAY
          (ABSOLUTE
            (PORT IN (1948:1948:1948) (1948:1948:1948))
            (PORT CLK (1520:1520:1520) (1520:1520:1520))
            (PORT RST (0:0:0) (0:0:0))
            (IOPATH CLK OUT (887:887:887) (887:887:887))
            (IOPATH SET OUT (887:887:887) (887:887:887))
            (IOPATH RST OUT (887:887:887) (887:887:887))
          )
        )
(TIMINGCHECK
(SETUP (negedge RST) (posedge CLK) (479:479:479))
(SETUP (posedge IN) (posedge CLK) (195:195:195))
(SETUP (negedge IN) (posedge CLK) (195:195:195))
(HOLD (posedge IN) (posedge CLK) (0:0:0))
(HOLD (negedge IN) (posedge CLK) (0:0:0))
(WIDTH (negedge CLK) (3456:3456:3456))
(WIDTH (posedge CLK) (3456:3456:3456))
(WIDTH (posedge RST) (3456:3456:3456))
        )
  )
```

Example 3 - SDF file generated by Xilinx software.

level simulation, you need to first create a table file. This table file will list all the PLI tasks that need to be included. The "simprims" directory inside the Xilinx installation contains the Verilog description of all the library cells (.vmd extension). These descriptions contain the timing checks for the setup and hold time violations. Annotating the original delay values from the .sdf file during simulation performs these checks.

Now create a pli.tab file, with the following content:

```
$sdf_annotate call=sdf_annotate_call
     acc+=mp,prx:reg_gate+
```

Where: reg_gate is the name of the top level module

Now, to perform a gate level simulation, you can use a sample script as follows:

```
vcs -RI -Mupdate reg_gate.v test_reg.v
    -o gate_reg -l gate_reg.log \
  -y $XILINX/verilog/src/simprims \
        +libext+.vmd+ \
          -P pli.tab
```

Where:

- **-y** stands for the library directory.
- **+libext+.vmd+** stands for all the files with extension .vmd.
- **-P** stands for pli table file.

Conclusion

The Synopsys FPGA Compiler II and VCS provide an easy and intuitive HDL design methodology, a seamless design flow, and high level control within the design process. In addition, using FPGA Compiler II gives you the power to effectively use Xilinx Virtex devices with the highest quality results. Σ