Post Synthesis Verification for Virtex FPGAs

For large designs especially, verification throughout the design flow will save you time and effort.

Nij Dorairaj, Sr. CAD Engineer, Exemplar Logic, Inc., nij@exemplar.com

As FPGAs grow bigger and faster it becomes increasingly important to verify your designs; this saves you time and produces better designs. Verification after synthesis should be part of the design process because this ensures that you pass the correct netlist to the Xilinx place and route tools. Also, you can debug your designs faster, whether you are using an RTL netlist (RTL verification) or a synthesized netlist (post synthesis verification).

Post Synthesis Verification

A synthesis tool usually writes a netlist based on the UNISIM library. The cells in the library are modeled well for both simulation and synthesis. For example in the verilog UNISIM library, the LUTs (look up tables) are modeled using UDP (user defined primitives). This can speed up your design simulation considerably, and because most of the combinational logic will be mapped to LUTs, overall simulation will also be fast. In short, if the synthesis tool can write a Verilog/VHDL netlist with LUT cells and their INIT attributes, verification will be very accurate and fast.

Enhancements in Leonardo Spectrum

Traditionally, verification was performed on the Verilog or VHDL netlist generated by the Xilinx place and route tools. Thanks to new functionality in the latest release of LeonardoSpectrum from Exemplar Logic, gate-level verification is now supported prior to place and route, using the Xilinx UNISIM simulation library.

Procedure for Using Gate-level Simulation

A special variable in LeonardoSpectrum must be set, to turn on this feature:

```
xi_write_init_on_luts (default is FALSE)
```

This variable must to be set to "TRUE" before writing out the Verilog/VHDL netlist. The variable can either be set in the GUI, using Tools -> Variable Editor, or in the interactive shell you can type:

```
set xi_write_init_on_luts TRUE
```

Verilog Example

Here is a simple Verilog example, to demonstrate the feature. The RTL design is synthesized in LeonardoSpectrum and a Verilog netlist (with the above variable set to TRUE) is written out. The synthesized netlist is simulated using the MTI (Modeltech) Verilog simulation tool. The design output is the registered value of A and B and C. The register clock is four times slower than the main clock; it uses CLKDLL to divide the main clock by four. The output waveform is shown in Figure 1.

Notice the INIT properties being used to configure the LUT cells of the synthesized netlist.

RTL Design

```
// module defintion of CLKDLL
module CLKDLL ( CLKIN, CLKFB, RST, CLK0,CLK90,CLK180,CLK270,
         CLK2X,CLKDV, LOCKED
         ) ;
 parameter CLKDV_DIVIDE = "2.0" ;
 parameter DUTY_CYCLE_CORRECTION = "TRUE";
 input  CLKIN, CLKFB, RST ;
 output      CLK0,CLK90,CLK180,CLK270,CLK2X,CLKDV, LOCKED ;
endmodule // CLKDLL

module slow_clk_dll (CLKOUT, CLKIN, RST);
 input CLKIN, RST;
 output CLKOUT;
 wire IBUFG_OUT, BUFG_A_IN, BUFG_A_OUT, BUFG_B_IN, BUFG_B_OUT;
 IBUFG Ub_ibufg ( .I(CLKIN),.O(IBUFG_OUT) );
 CLKDLL  Ub_clkdll (  .CLKIN(IBUFG_OUT),.CLKFB(BUFG_A_OUT),.RST(RST),
           .CLK0(BUFG_A_IN),.CLKDV(BUFG_B_IN)  );
 // divide the input clock by 4
 defparam Ub_clkdll.CLKDV_DIVIDE = "4.0";
 BUFG Ub_bufgA ( .I (BUFG_A_IN),.O(BUFG_A_OUT) );
 BUFG Ub_bufgB ( .I (BUFG_B_IN),.O(BUFG_B_OUT) );
 assign CLKOUT = BUFG_B_OUT;
endmodule // slow_clk_dll

module simple (A,B , C, RST, OUT, CLK);
 input A, B, C, RST, CLK;
 output OUT;
 reg  OUT;
 wire SLOW_STEADY_CLK;
 slow_clk_dll  simple_clk  (SLOW_STEADY_CLK,CLK,RST);
 always @ (posedge SLOW_STEADY_CLK)
  begin
    if (RST)
     OUT = 1'b0;
    else
     OUT = A & B & C;
  end
endmodule // simple
```

Synthesized Netlist

```
module simple ( A, B, C, RST, OUT, CLK ) ;
  input A , B, C, RST, CLK;
  output OUT ;
  wire  SLOW_STEADY_CLK,simple_clk_BUFG_B_IN, simple_clk_BUFG_A_OUT,
    simple_clk_BUFG_A_IN, simple_clk_IBUFG_OUT, A_int,B_int,C_int,
    RST_int,OUT_dup0,nx53,nx54;
  wire [4:0] \$dummy ;
  IBUFG simple_clk_Ub_ibufg (.O (simple_clk_IBUFG_OUT),.I (CLK)) ;
  CLKDLL simple_clk_Ub_clkdll (.CLK0 (simple_clk_BUFG_A_IN),.CLK90 (
     \$dummy [0]),.CLK180 (\$dummy [1]),.CLK270 (\$dummy [2]),.CLK2X (
     \$dummy [3]),.CLKDV (simple_clk_BUFG_B_IN),.LOCKED (\$dummy [4]),.CLKIN (
     simple_clk_IBUFG_OUT),.CLKFB (simple_clk_BUFG_A_OUT),.RST (RST_int)
     ) ;
     defparam simple_clk_Ub_clkdll.CLKDV_DIVIDE = 4.0;
     defparam simple_clk_Ub_clkdll.DUTY_CYCLE_CORRECTION = "TRUE";
  BUFG simple_clk_Ub_bufgA (.O (simple_clk_BUFG_A_OUT),.I (
    simple_clk_BUFG_A_IN)) ;
  BUFG simple_clk_Ub_bufgB (.O (SLOW_STEADY_CLK),.I (simple_clk_BUFG_B_IN)) ;
  OBUF OUT_obuf (.O (OUT),.I (OUT_dup0)) ;
  IBUF RST_ibuf (.O (RST_int),.I (RST)) ;
  IBUF C_ibuf (.O (C_int),.I (C)) ;
  IBUF B_ibuf (.O (B_int),.I (B)) ;
  IBUF A_ibuf (.O (A_int),.I (A)) ;
  FDR reg_OUT (.Q (OUT_dup0),.D (nx54),.C (SLOW_STEADY_CLK),.R (nx53)) ;
  LUT2 ix46 (.O (nx53),.I0 (C_int),.I1 (RST_int)) ;
    defparam ix46.INIT = 4'hD;
  LUT2 ix47 (.O (nx54),.I0 (A_int),.I1 (B_int)) ;
    defparam ix47.INIT = 4'h8;
endmodule
```

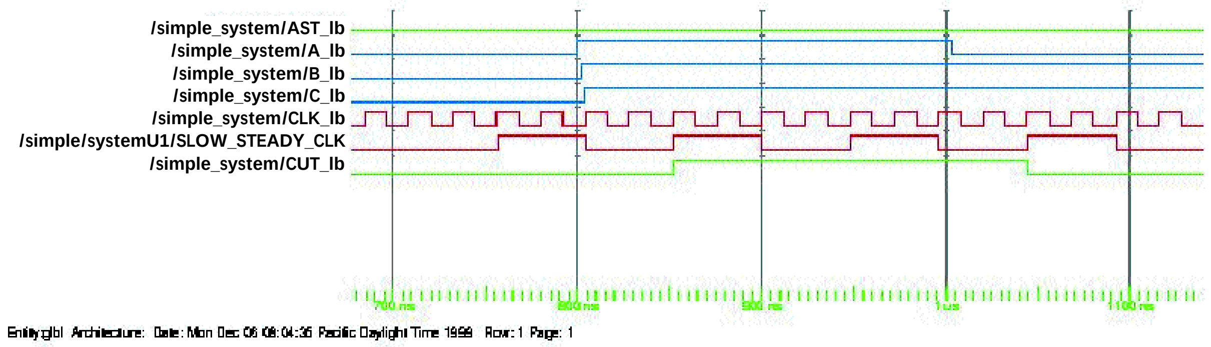

Figure 1 - Output waveform from Modeltech.

Conclusion

With LeonardoSpectrum you can verify the correctness of your netlist before you go to place and route, and get high quality results for your Virtex FPGA designs.

A more complete version of this article, with a test bench,is available at: http://www.exemplar.com/support/appnotes.html.

References

1. Virtex chapter in LeonardoSpectrum technology manual (leo_tech.pdf). 2. Verilog GSR/GTS Simulation Methodology in Xcell issue 33 (3rd quarter 1999).