Dipl.-Ing. T. Wurlitzer (a3wurl@et.htwk-leipzig.de)
Prof. Dr.-Ing. habil. W. Reinhold (reinhold@et.htwk-leipzig.de)

VHDL Post-Route Simulation mit XILINX-FPGA's

I. VHDL als Hardwarebeschreibungssprache ist im Vormarsch

Beim Schaltungsdesign von FPGA's wird der Einsatz von VHDL immer beliebter.

Die Vorteile liegen auf der Hand, wie z.B.:

- weltweit akzeptierter Standard zur Simulation und Design-Dokumentation
- verschiedene Abstraktionsebenen
- einfache Möglichkeiten zur Definition von Testumgebungen
- portabel für die Überführung in andere Technologien (z.B. ASIC)

Synthesewerkzeuge werden zunehmend - gerade im PC-Bereich - verfügbar und auch bezahlbar.

Leider ist es hier auch wie im übrigen Leben: Neben den Vorteilen bleiben dem Designer doch ein paar Probleme erhalten ...

So ist z.B. die Verwendung von herstellerspezifischen Blöcken für die Synthese der Schaltung sehr günstig - Place & Route Tools liefern optimale Ergebnisse - aber der VHDL-Simulator muß auch die zugehörigen Modelle kennen, sonst ist die Schaltung nicht simulierbar. Dies setzt aber noch zusätzliche Werkzeuge voraus (z.B. Synopsys Library Compiler für Workstations ...).

Genauso problematisch kann die Timing-Simulation plazierter und verdrahteter Designs werden, da dem VHDL-Simulator zuerst die hardwarespezifischen Informationen in aufbereiteter Form übergeben werden müssen.

In diesem Artikel soll eine mögliche Vorgehensweise für diese Post-Route Simulation anhand der XILINX FPGA's und dem VHDL Simulator V-System (Model Technology Inc.) erläutert werden. Mein Dank gilt dabei der Firma AVNET E2000, die als XILINX-Distributor die VITAL Packages zur Verfügung gestellt hat, die nun zusammen mit diesem Artikel (als PDF-Datei) unter **ftp://mrt.htwk-leipzig.de/pub/interrupt/97/vhdlback** vom Internet geladen werden können.

II. Der Schlüssel zum Erfolg - VITAL packages

Ziel der Post-Route Simulation ist die Designverifikation plazierter und verdrahteter Designs. Dabei sollen natürlich die gleichen Werkzeuge und die gleiche Testumgebung wie bei der Funktionalsimulation zum Einsatz kommen, um Abweichungen im Timing schnell erkennen zu können.

Das heißt, im VHDL-Simulator wird die (fast) gleiche Testbench benutzt. Das entworfene VHDL-Modell, das bei der Funktionalsimulation benutzt wurde, wird nun durch ein VHDL-Modell des plazierten und verdrahteten Designs ersetzt. Dazu sind noch einige nachfolgend beschriebene Ergänzungen im VHDL-Simulator sowie technologiespezifische Bibliotheken notwendig.

Damit sowohl Funktionalsimulation als auch Post-Route Simulation durchgeführt werden können, müssen die dazu notwendigen Dateien kopiert und entsprechend umbenannt werden. Darauf wird später noch ausführlich eingegangen.

Um technologiespezifische Bibliotheken schneller für VHDLverfügbar zu machen d.h. damit die plattformübergreifende Simulation von ASIC-Bibliotheken zu ermöglichen, wurde ein Konsortium, die „VHDL Initiative Towards ASIC Libraries" (VITAL) gegründet. Es wurde von diesem Konsortium eine allgemeine Modellspezifikation für Simulationszwecke erarbeitet, die sich an den IEEE Standards 1076 und 1164 orientiert und von verschiedenen EDA-Tools gleichermaßen genutzt werden kann.

Dazu muß der Hersteller eines FPGA's oder ASIC's Werkzeuge zur Verfügung stellen, die das Design gemeinsam mit den Timing-Informationen in eine VHDL-gemäße Beschreibung konvertieren, die dieser Modellspezifikation entspricht.

Die Spezifikation nutzt das Open Verilog International (OVI) Standard Delay Format (SDF) für die Timing Simulation. In dem SDF-File sind die Verzögerungszeiten des zugehörigen (gerouteten) Designs enthalten, so können z.B. minimale, typische oder maximale Werte für worst-case Untersuchungen genutzt werden.

Die VITAL packages enthalten weiterhin:

- VITAL_Timing VHDL package: enthält die Timing Prozeduren für die Auswahl, Überprüfung und Fehlerausgaben der Verzögerungszeiten
- VITAL_Primitives VHDL package: enthält die „Primitives" für die Funktionalbeschreibung mit Bool'schen Grundfunktionen, Wahrheitstabellen etc.

Der VHDL-Simulator muß das SDF-Format verarbeiten können und er muß die „VITAL packages" richtig simulieren können (gemäß VITAL Model Development Specification), d.h. er muß „VITAL-compliant" sein.

III. Die Installation der VITAL packages

Zuerst müssen die VITAL packages im VHDL-Simulator installiert werden, wenn sie nicht bereits vorinstalliert mitgeliefert werden. Dazu werden die folgenden Dateien in das VHDL-Source Verzeichnis des Simulators kopiert (z.B. im Unterverzeichnis vital95x) und anschließend im Simulator compiliert:

```
timing_p.vhd
timing_b.vhd
prmtvs_p.vhd
prmtvs_b.vhd
```

```
-- Compiling VITAL packages...
vmap IEEE e:\vhdl\IEEE
vcom -87 -work IEEE e:\vhdl\vhdl_src\vital95x\timing_p.vhd
vcom -87 -work IEEE e:\vhdl\vhdl_src\vital95x\timing_b.vhd
vcom -87 -work IEEE e:\vhdl\vhdl_src\vital95x\prmtvs_p.vhd
vcom -87 -work IEEE e:\vhdl\vhdl_src\vital95x\prmtvs_b.vhd
```

Das Compilieren kann wie in der Tabelle ausgeführt werden, außerdem sind die Anweisungen noch in der Datei compile.do enthalten und können so im Simulator als Makro ausgeführt werden.

Als nächstes sind die technologiespezifischen VITAL component packages zu installieren, die man vorher vom Hersteller beziehen muß. Das sind in dem Fall für die XILINX XC4000e und XC5200 Familien folgende Dateien:

```
x4ke_vpa.vhd
x4ke_vit.vhd
x4ke_vco.vhd
x4ke_vco.vhd
x5k_vtab.vhd
x5k_vit.vhd
x5k_vcom.vhd
```

die zweckmäßig in das Vital-Source Verzeichnis der Design Software - hier Synario von DATA/IO - kopiert werden (z.B. im Unterverzeichnis vitx-src) und anschließend im Simulator noch zu compilieren sind:

```
-- Compiling VITAL timing packages...
vlib e:\synario\mti_libs\xc4000e
vmap xc4000e e:\synario\mti_libs\xc4000e
vcom -87 -work xc4000e e:\synario\vitx-src\x4ke_vpa.vhd
vcom -87 -work xc4000e e:\synario\vitx-src\x4ke_vit.vhd
vcom -87 -work xc4000e e:\synario\vitx-src\x4ke_vco.vhd
vlib e:\synario\mti_libs\xc5200
vmap xc5200 e:\synario\mti_libs\xc5200
vcom -87 -work xc5200  e:\synario\vitx-src\x5k_vtab.vhd
vcom -87 -work xc5200  e:\synario\vitx-src\x5k_vit.vhd
vcom -87 -work xc5200  e:\synario\vitx-src\x5k_vcom.vhd
```

Das Compilieren kann wieder wie in der Tabelle ausgeführt werden oder auch als Makro (in der Datei compile.do) ausgeführt werden.

Zur Konvertierung der FPGA-Design-Daten in das VITAL-Format muß nun nur noch das Programm XNF2HDL.EXE in das bin-Verzeichnis der FPGA-Design Software kopiert werden.

Nun ist der Simulator soweit vorbereitet und wir können uns endlich wieder um das FPGA-Design kümmern.

IV. Die Backannotation des Designs

Die Vorgehensweise bei der Backannotation soll an Hand eines Beispieldesigns erläutert werden. Es handelt sich dabei um einen einstellbaren Zähler aus einem Synario-Demo Projekt (Projektname: zaehler), der als höchste Hierarchieebene die Komponente top enthält.

Zuerst muß das geroutete Design (zaehler.lca) mit dem Programm LCA2XNF in eine XILINX-Netzliste mit Timing-Informationen (zaehler.xpf) übersetzt werden, was durch die FPGA-Design Software automatisch erfolgen kann.

Danach kann man die erzeugte Datei nach vback.xnf umbenennen, um das Überschreiben von Originaldateien zu verhindern. Mit dem Programm XNF2HDL wird nun die Netzlistendatei in das VITAL kompatible VHDL-Format konvertiert. Ergebnis sind dann u.a. die Dateien vback_st.vhd und vback.sdf, die nun endlich vom VHDL-Simulator verarbeitet werden können und alle notwendigen Informationen über das geroutete Design enthalten.

Um den Ablauf für wiederholte Designzyklen zu automatisieren, wird top.vhd nach topr-h.vhd kopiert. In topr-h.vhd sind die IEEE-Library und VITAL packages einzufügen und dafür die nicht mehr Benötigten auszukommentieren. Die dazu gehörige Architetecture wird entfernt und das ganze Modell nach vback umbenannt.

```
-- file: topr-h.vhd
-- VHDL Model Created from ECS Schematic top.sch -- Jan 21
19:45:25 1997
-- modified for post route simulation

LIBRARY ieee;
--LIBRARY generics;
--LIBRARY xc_5000;
LIBRARY xc5200;

USE ieee.std_logic_1164.ALL;
--USE ieee.numeric_std.ALL;
--USE generics.components.ALL;
--USE xc_5000.components.ALL;
USE IEEE.VITAL_Primitives.all;
USE IEEE.VITAL_Timing.all;
USE xc5200.Vcomponents.all;
USE xc5200.Vtables.all;

entity vback is
      Port ( ...
           );
```

Diese Datei wird nun nach vback.vhd kopiert. Durch Anhängen der erzeugten Datei vback_st.vhd an vback.vhd wird sichergestellt, daß immer das aktuelle Design

simuliert wird. Für alle weiteren Simulationsläufe braucht nur die nachfolgende Batch-Datei ausgeführt werden.

```
copy *.xpf vback.xnf
xnf2hdl vback.xnf
copy topr-h.vhd vback.vhd
type vback_st.vhd >>vback.vhd
```

Nun muß noch die Datei, die die Testbench für die Funktionalsimulation enthält, nach simvhdbk.vhd kopiert werden. In simvhdbk.vhd muß nun statt der Komponente top die Komponente vback eingesetzt werden. Mit der alten Testbench kann die Funktionalsimulation weiterhin durchgeführt werden und mit der Testbench aus simvhdbk.vhd wird nun die Post-Route Simulation unter gleichen Testbedingungen durchgeführt.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
USE std.textio.all;

ENTITY testbench IS END;

ARCHITECTURE behavior OF testbench IS

-- declaration of component vback

COMPONENT vback Port (
...

);
END COMPONENT;

-- declare external signals to drive design

     SIGNAL ...
-- connect external signals to the ports of the chip

BEGIN
   DUT: vback PORT MAP(
     ...
   );
-- test signal stimulus (input waveforms)
...
END behavior;
```

Vback wird als DUT (Device Under Test) instantiiert. Nach dem Compilieren kann nun die Simulation erfolgen:

```
vsim -t ps -lib work -sdftyp DUT=vback.sdf testbench
```

DUT wird nun mit der erzeugten SDF-Datei gemappt und als Top-Level testbench simuliert.

So einfach kann die Post-Route Simulation sein ...

Dieser Beitrag entstand innerhalb eines Technologie Transfer Projektes für Mikroelektronik, das im Rahmen eines FUSE Application Experiments von der EU gefördert wird.