Chipentwurf
Übungsblatt 4

5. Januar 2018

1. Verbinden Sie sich über den Cisco-VPN-Client bzw. dem NX-Client mit dem IHP Server mit Ihren persönlichen Zugangsdaten. VPN-Server: `webvpn.ihp-ffo.de`
   Linux-Server: `uxsrvup` oder `172.16.104.1`

2. Kopieren Sie den Inhalt des Verzeichnisses `/data/share/2017-WS/calc` samt der versteckten Dateien in Ihr Home-Verzeichnis:
   ```
   cd /data/share/2017-WS/
   cp -r * .* ~
   ```
   Wechseln Sie mit einer Konsole in den Ordner `~/calc` und geben Sie `make dirs` zur Erstellung benötigter Verzeichnisse ein.

3. Kopieren Sie ihre Quelldateien in den Ordner `source`, sowie die Testbenches in den Ordner `tb`.

4. Der Taschenrechnerchip soll nun mittels des Synopsys Design Compiler synthetisiert werden. Starten sie das Tool im GUI-Modus über den Befehl `design_vision`. Alternativ lässt sich das Tool auch im Batch-Modus ausführen. Dazu geben Sie `dc_shell` in der Konsole ein. Sie erhalten dann eine Tcl-Shell, mit der Sie das Tool steuern. Mittels `man <command>` lässt sich die zugehörige Hilfe aufrufen.

   - Lesen Sie zunächst die Quelldateien des Designs ein. Öffnen Sie dazu den Dialog unter `File` → `Read...` und fügen Sie alle Dateien im Ordner `source` der Liste hinzu.
   - Definieren Sie die Top-Level-Entität (z.B. `calc`) über die Konsole:
     `current_design calc`
   - Schauen Sie sich das Schematic der Schaltung an (Button).
   - Definieren Sie die Taktfrequenz des Designs. Wechseln Sie dazu in die Symbolansicht des Designs ((Button )). Selektieren Sie dort den Takteingang (hier z.B. `clk`) und öffnen Sie den Dialog zur Definition des Clock-Constraints (`Attributes` → `Specify Clock...`). Tragen Sie als Name der Clock `clk` ein. Definieren Sie die Periode als 1 ns, sowie die steigende und die fallende Flanken, so dass ein symmetrisches Taktsignal erwartet wird. Sie können dies auch über den folgenden Befehl erreichen:
     `create_clock -name clk -period 1 -waveform {0 0.5} [find port clk]`

- Setzen Sie ein Constraint, um den Compiler anzuweisen eine möglichst kleine Schaltung zu generieren: `set_max_area 0`
- Synthetisieren Sie nun das Design: `compile`
- Überprüfen Sie das Timing: `report_timing`. Der Slack (Zeit des Eintreffens eines Signals relativ zur Taktflanke) darf von keinem Signal negativ sein, da dies bedeuten würde, dass die Taktflanke um die angegebene Zeit vor dem jeweiligen Signal eintrifft.
- Passen Sie das Clock-Constraint so an, dass die Schaltung das Constraint erfüllen kann und synthetisieren Sie diese erneut.

Anmerkung: In der Realität setzt man das Constraint entsprechend der Spezifikation des Designs. Sollte das Timing der Schaltung nicht passen, muss man das Design (!) und nicht das Constraint anpassen.

- Überprüfen Sie das Design (`check_design`), dessen Fläche (`report_area`), Leistungsaufnahme (`report_power`).
- Analysieren Sie das generierte Schematic der Schaltung.
- Exportieren Sie die Verilog-Netzliste:
  `write -format verilog -hierarchy -output ./syn/calc.v`
  das Zeitverhalten im SDF-Format:
  `write_sdf -version 3.0 ./syn/calc.sdf`
  und alle definierten Constraints im SDC-Format:
  `write_sdc ./syn/calc.sdc`
- Beenden Sie den Design Compiler.

Weitere Hinweise:

- Die Synthese der Schaltung kann komplett über ein Skript ausgeführt werden. Schauen Sie sich das `Makefile` und das Skript `./script/dc.tcl` an und nehmen Sie notwendige Änderungen vor.

5. Nun soll die generierte Netzliste mit Modelsim mit einer so genannten Back-Annotation simuliert werden. Dies bedeutet, dass das Zeitverhalten der Gatter bei der Simulation der Schaltung berücksichtigt wird.
   - Wechseln Sie in den Ordner `sim` und starten Sie das Werkzeug über die Konsole mit dem Befehlt `vsim`.
   - Erstellen Sie ein neues Projekt (`File` → `New` → `Project...`). Definieren Sie einen Namen z.B. `calc`.
   - Im darauffolgenden Dialog kann man Quelldateien dem Projekt hinzufügen (`Add Existing File`). Fügen Sie die generierte Netzliste `../syn/calc.ps.v` sowie die Test Bench `../tb/tb_calc.vhd` zum Projekt hinzu. Darüber hinaus wird ein Verhaltensmodell der durch die Netzliste verwendeten Gatter benötigt. Fügen Sie entsprechend die folgende Datei dem Projekt hinzu:
     `/home/ihpdk/sgb25/verilog/SESAME-LP2_IHP_0.25um.v`
   - Definieren Sie nun zunächst die Kompilierungsreihenfolge (`Compile` → `Compile Order`) als `SESAME-LP2_IHP_0.25um.v`, `calc.ps.v`, `tb_coffeebreak_synth.vhd` und kompilieren Sie die Dateien (`Compile` → `Compile All`).

- Simulieren Sie das Design (`Simulate` → `Start Simulation...`). Expandieren Sie im aufkommenden Dialog den Knoten `work` und suchen Sie dort den Eintrag mit dem Namen der Test Bench. Deaktivieren Sie die Checkbox `Enable Optimization` und setzen Sie die Resolution auf 1 ps. Gehen Sie auf den Reiter `SDF`. Fügen Sie mittels `Add...` die SDF-Datei `calc.ps.sdf` hinzu und geben Sie als Region `/tb_calc/uut` an. Damit wird spezifiziert, wo die Design-Hierarchie der SDF-Datei in der Hierarchy der Test Bench zu finden ist (vgl.: die `calc`-Instanz in der Test Bench heißt `uut`). Des Weiteren müssen Sie dort auch die Checkbox `Reduce SDF Errors to Warnings` aktivieren.
- Fügen Sie alle Signale des Designs der Waveform-View hinzu (rechte Maustaste auf den Simulationsbrowser → Add → To Wave) und simulieren Sie das Design. Geben Sie dazu den Befehl `run -all` in die Konsole ein.