Benutzerdokumentation - fhlow v1.4

Inhaltsverzeichnis

1 fhlow 1.4 **3**
1.1 Startskripte mit Python 3
1.2 Pfadkonfiguration mit Python 3

2 IP Cores **4**
2.1 Erstellen eines IP Cores 4
2.2 Integrieren von IP Cores 4

3 Beispiele **5**

1 fhlow 1.4

Im folgenden Kapitel wird auf die Optimierungen von fhlow eingegangen, die im Rahmen der dazugehörigen Bachelorarbeit entstanden sind. Aufgrund diverser Änderungen, insbesondere der Verwendung von IP Cores in Simulation und Synthese sowie die Adaptierung der Batch-Skripte und Makefiles zu Python-Skripte, werden alle wichtigen und für Benutzer und Benutzerinnen relevanten Neuerungen geschildert.

1.1 Startskripte mit Python

Die bekannte Programmiersprache Python erwies sich als eine zuverlässige Alternative zu Batch-Skripten und Makefiles. Aufgrund der Kompatibilität von Python unter allen großen und bekannten Betriebssystemen, konnte die betriebssystem-unabhängige Sprache Python unter einem UNIX System und unter Windows verifiziert werden.
Um diese Skripte auszuführen, muss das Programm Python Version 3 installiert werden. Unter der Webseite *https://www.python.org/* kann dieses heruntergeladen werden. Nach der Installation gibt es unter Windows die Möglichkeit, die Python-Skripte mit der *python.exe* automatisch auszuführen. Dazu muss mit einem Rechtsklick der Maus ein Python-Skript gewählt werden und der Dialog *Ausführen mit* gewählt werden. Nun ist lediglich die installierte *python.exe* zu wählen und die Skripte können wie ein Batch-Skript ausgeführt werden.
Unter einem UNIX System kann nach der Installation im Terminal, beispielsweise die Synthese, mit dem Befehl *python ChooseSynOption.py* gestartet werden.

Die zu wählende Option im Python-Skript erlaubt die Ausführung der jeweils dahinterstehenden Anwendung. Der Tastendruck muss nicht mit *Enter* bestätigt werden. Das Skript sichert sich zudem gegen eine falsche Parameterübergabe ab, indem nur die aufgelisteten Zahlen als Eingangsargument gültig sind.

1.2 Pfadkonfiguration mit Python

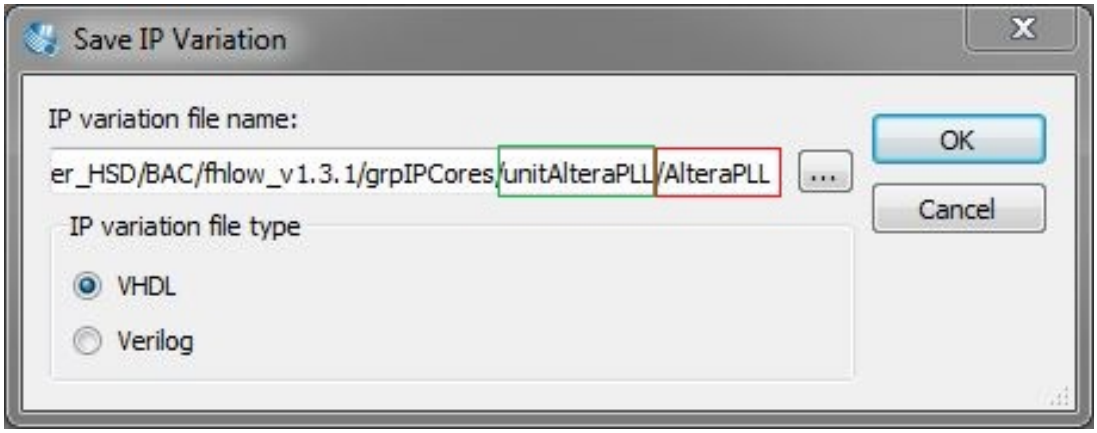
Da auch für die UNIX Systeme und Windows jeweils eigene Pfadkonfigurationen zu treffen waren, wurde dies in eine Python-Skript zusammengefasst. In der Datei *PathConfig.py* können sowohl die Variablen für den eigenen Benutzernamen gewählt werden (hierzu muss der Variable *User* der konfigurierte Name des Systems übergeben werden) oder eine Standardkonfiguration zum Setzen der Variablen in der Fachhochschule.
Des weiteren wird zwischen einem Windows System und einem UNIX System unterschieden. Die Variablen sind für das erfolgreich Ausführen der Anwendungen zwingend erforderlich!

2 IP Cores

Durch die gewinnende Bedeutung von IP Cores, insbesondere auf Developmentboards mit eine System-on-Chip, wurde die Integration von IP Cores hinzugefügt. Unter Altera Quartus lassen sich diese sowohl im MegaWizard als auch im Programm Qsys generieren.

2.1 Erstellen eines IP Cores

Um einen IP Core zu erstellen und erfolgreich in die Skriptumgebung zu integrieren, ist in den Ordner *grpIPCores* zu wechseln. In diesem Ordner befindet sich eine Python-Skript (*CreateIPCore.py*), welches gestartet werden muss. Durch die Ausführung öffnet sich Altera Quartus und der IP Core kann erstellt werden. Um den IP Core korrekt zu konfigurieren, ist auf den Dialog des Python-Skripts zu achten. Der Name, den das Modul erhält, muss in Altera Quartus ebenfalls verwendet werden! Im folgenden ist ein Beispiel der Integration einer Altera PLL zu sehen, welche mit Hilfe des MegaWizards erstellt wird.
Der automatisch erstellte Ordner *grün* ist umrandet, der von Benutzer oder Benutzerin hinzuzufügende Name ist *rot* markiert.



Nach der Erstellung des IP Cores, ist Altera Quartus zu schließen! Somit wird automatisch der IP Core in das korrekte Verzeichnis abgelegt. Zudem werden Änderungen am generierten Tcl-Skript vorgenommen, um die Kompatibilität mit allen Mentor Graphics Simulatoren zu unterstützen. Der IP Core kann nun erfolgreich in eine Konfigurationsdatei eines Einzelprojektes aufgenommen werden.

2.2 Integrieren von IP Cores

Um die IP Cores korrekt zu integrieren, ist die jeweilige Konfigurationsdatei für das Projekt zu öffnen. Unter *append IPCores* kann nun, wie aus dem oberen Beispiel, die AlteraPLL hinzugefügt werden. Diese Vorgehensweise ist identisch zum Hinzufügen anderen Units, Packages, etc.
Als Referenz können die Beispiele betrachtet werden. Da jedoch der grp-Ordner immer identisch bleibt, muss die Skriptumgebung nur den jeweiligen unit-Namen kennen.

3 Beispiele

Es wurden zur Demonstration ein paar Beispiele entwickelt, um das Einbinden genauer zu erörtern. Die Beispiele sind im Ordner *grpExamples* zu finden und sind wie im Folgenden numerisch aufgelistet.

Beispiel 1 *(unitExample1)*: Dieses enthält keinen IP Core und dient als Verifizierung der alten Funktionalität von fhlow.

Beispiel 2 *(unitExample2)*: In diesem ist ein IP Core mit Hilfe des MegaWizards erstellt worden. Die Altera PLL ist sowohl in Simulation als auch Synthese erfolgreich eingebunden worden.

Beispiel 3 *(unitExample3)*: Zur vorherigen PLL wurde hier noch zusätzlich ein IP Core mit Hilfe von Qsys generiert. Der IP Core ist ein Avalon SPI.

Beispiel 4 *(unitExample4)*: Einbinden eines AvalonJTAG2Memory IP Cores in Simulation und Synthese. Generiert wurde dieser mit Qsys.

Diese Beispiel sind lediglich zu Demonstrationszwecken, um die Änderungen an fhlow zu verifizieren. Die Projekte können zwar komplett simuliert und synthetisiert werden, jedoch dienen diese Implementierungen keinem realen Anwendungsfall und sollten somit unter keinen Umständen für ein eigenes FPGA Designs verwendet werden! Diese Beispiele und IP Cores können somit auch ohne Weiteres aus der Skriptumgebung entfernt werden, wenn diese nicht mehr benötigt werden.