Change-Library Python3 Skript

26. Januar 2018

Vorwort:

Es handelt sich hier um eine **BETA-Version**! Bitte speichert selbst die Dateien in einem separaten Backup Ordner! Das Programm besitzt bestimmt noch Bugs und falls jemand nicht die Linux Rechner in dem MAR Gebäude nutzt, kann ich nicht für ein fehlerfreies Verhalten garantieren.
Das Programm soll euch nur bei dem letzten Aufgabenblatt langweiliges Ein- und Auskommentieren von Dateien ersparen.
Zudem wollte ich mal Python ausprobieren und mir die Sprache anschauen, also seid nicht zu kritisch was den Quellcode angeht.

Was macht dieses Skript?

Wie in den letzten Aufgabenblätter beschrieben, müssen bei fehlerhaften Modulen diese mit Modulen aus **ARM_SIM_LIB** ersetzt werden. Hat man die fehlerhaften Module ersetzt und die Simulation läuft durch, so muss man für die Synthese **ARM_SIM_LIB** wieder auskommentieren.
Besonders das aufwendige Ein-/Auskommentieren von **ArmTypes** für **ArmArithInstructionCtrl** soll von diesem Skript automatisiert werden. Dafür wird in jede **.vhd**-Datei ganz oben eine Zeile eingefügt mit dem Inhalt **library ARM_SIM_LIB;**. Das ist notwendig, da in den anderen Dateien diese Bibliothek sonst gar nicht verwendet wird oder erst nach **work.ArmTypes**, was zu einem Fehler führen würde. Das doppelte Einbinden einer Bibliothek ist in VHDL erlaubt. Und falls man wieder das eigene **ArmArithInstructionCtrl** benutzt, wird die **library ARM_SIM_LIB;** Zeile am Kopf der Datei wieder entfernt.
Dieses Skript soll diesen Teil automatisieren. Dafür erwartet das Programm folgendes:

- Die Kommentare von **ARM_SIM_LIB** und **use.ARM_SIM_LIB.Arm[...]** die standardmäßig in den VHDL-Dateien vorgegeben sind, dürfen *nicht entfernt* worden sein. Falls diese entfernt worden sind, kann man sie einfach an der gleichen Stelle wieder aus den Vorgaben kopieren und einfügen.
- Die Dateien wurden *nicht unbenannt*.
- Alle Dateien liegen mit dem Skript und `easygui.py` in einem Ordner
  Falls dies nicht der Fall ist, dann kann man mit folgendem Befehl alle .vhd Dateien rekursiv in den Ordner `dest` kopieren:

`rsync -avm --include='*.vhd*' --filter 'hide,! */' . ./dest`
Dabei gilt es wieder zu beachten den Befehl *nicht* aus der pdf-Datei zu kopieren.

- Folgende Dateien sind mindestens vorhanden:
  - ArmCore
  - ArmTop
  - ArmDataPath
  - ArmControlPath

Was toleriert das Programm?

- Die oben genannten Kommentare dürfen anders eingerückt worden sein, diese werden flexibel mit Regex gesucht. Deswegen spielt auch die Groß-/ und Kleinschreibung von Modulen keine Rolle.
- Falls man überflüssigerweise die Module in work eingebunden hat, also zB work.ArmALU; so wird dieses Skript diese Zeilen ebenfalls ein-/ bzw. auskommentieren.

Folgendes ist erlaubt:

```
--ARM_SIM_LIB;
-- arm_sim_lib;
--  use ARM_SIM_LIB.ArmALU;
--- use arm_sim_lib.armALU; --Das Modul funkt eventuell nicht
```

Folgendes ist nicht erlaubt:

```
-- use ARM_SIM_LIB.ArmALU; Das Modul funkt eventuell nicht
```

Da nicht extra überprüft wird, ob nach dem Semikolon noch Text kommt, um diesen auszukommentieren. Es werden quasi nur die Minuszeichen am Anfang entfernt.
Das Skript kann mehrmals ausgeführt werden und es sollten keine Fehler auftreten, wenn man Module mehrfach ein-/ bzw. auskommentiert. Das Skript `ChangeLib_Win_Support.py` und `easygui.py` muss in den `src`-Ordner verschoben werden, bzw. dort wo sich die VHDL Dateien befinden. Auf den Rechner führt man das Skript mit folgendem Befehl aus:

```
python3 ChangeLib_Win_Support.py
```

Falls man dieses Skript auf dem eigenen Rechner ausführen möchte, muss man *python3* und *tkinter* installieren. Wobei *tkinter* bei der Standard Python3 Installation mitinstalliert werden sollte. Falls nicht, kann man dies mit:
`sudo apt-get install python3-tk` nachholen.

Führt man das Skript aus, so wird erst gefragt, ob die Dateien gespeichert werden sollten, siehe Abb. 1. Dies sollte man unbedingt bei dem ersten Mal machen!

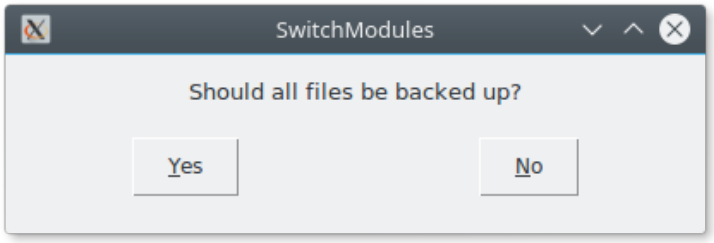

Abbildung 1: Backup Window

Danach wird man gefragt, ob man seine Module mit denen von der HWPTI Musterlösung tauschen möchte, siehe Abb. 2.

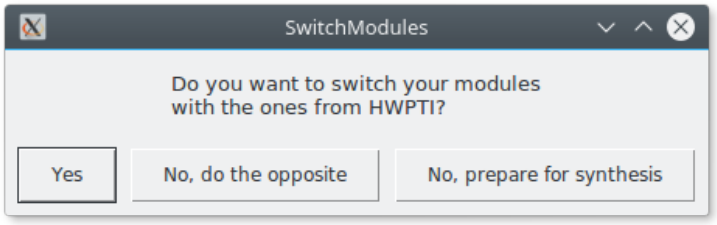

Abbildung 2: Switch Modules Window

Zur Auswahl stehen:

- „Yes“ - Also werden eure Module mit denen vom HWPTI ersetzt
- „No, do the opposite“ - Es werden die Module vom HWPTI mit euren ersetzt.
- „No, prepare for synthesis“ - Die Dateien werden für die Synthese vorbereitet.

Wählt man nun eins von den ersten beiden aus, so muss man sich für die Module entscheiden, die man tauschen möchte, siehe Abb. 3

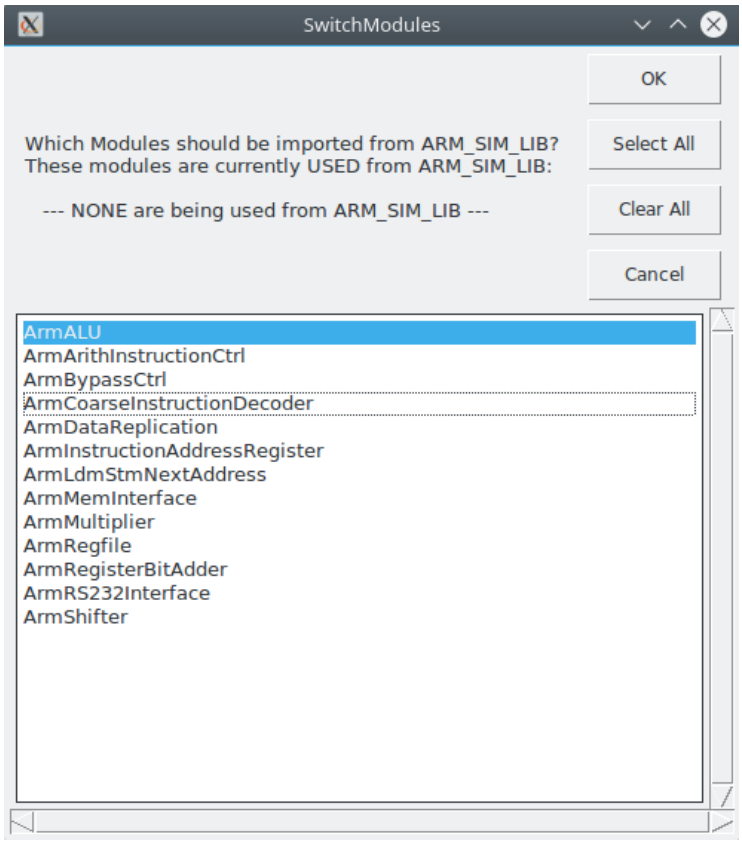

Abbildung 3: Choose Modules Window

Im Textfeld werden die Module aufgezählt, die momentan aus den HWPTI Musterlösungen genutzt werden, also von der Bibliothek ARM_SIM_LIB.

HINWEIS: ArmDataPath steht hierbei nicht zur Auswahl, da sie auch nicht als .ngc zur Verfügung steht und weil es nicht Sinn der Aufgabe ist, den kompletten Datenpfad aus der Musterlösung zu übernehmen. Desweiteren kann nicht ArmRegAddressTranslation aus der Musterlösung genutzt werden. Falls es Probleme beim Kompilieren von ArmControlPath gibt, so muss man nur ArmGlobalProbes neu kompilieren, dann sollte es funktionieren.

Hat man die Module ausgewählt, so muss man nur noch auf „OK“ klicken und die Module werden ausgetauscht.
Nachdem man alle Module ausgetauscht hat und die Simulation durchläuft, kann man im Auswahlmenü „No, prepare for synthesis“ auswählen und kann den Code zum Synthetisieren vorbereiten. Dabei werden alle ARM_SIM_LIB Zeilen auskommentiert und alle *work* Zeilen bleiben auskommentiert.