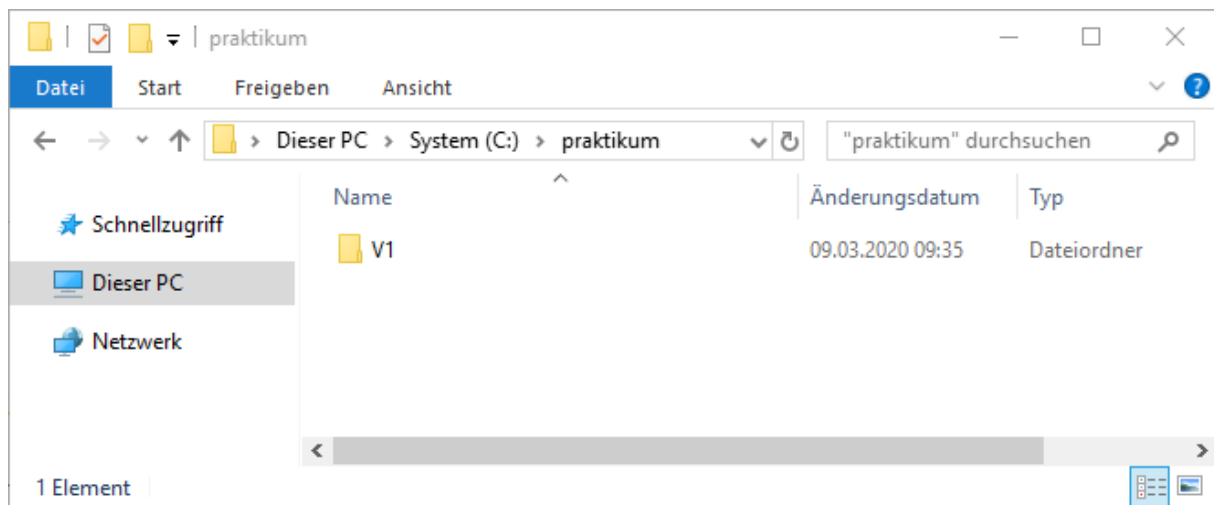


Simulation mit ModelSim

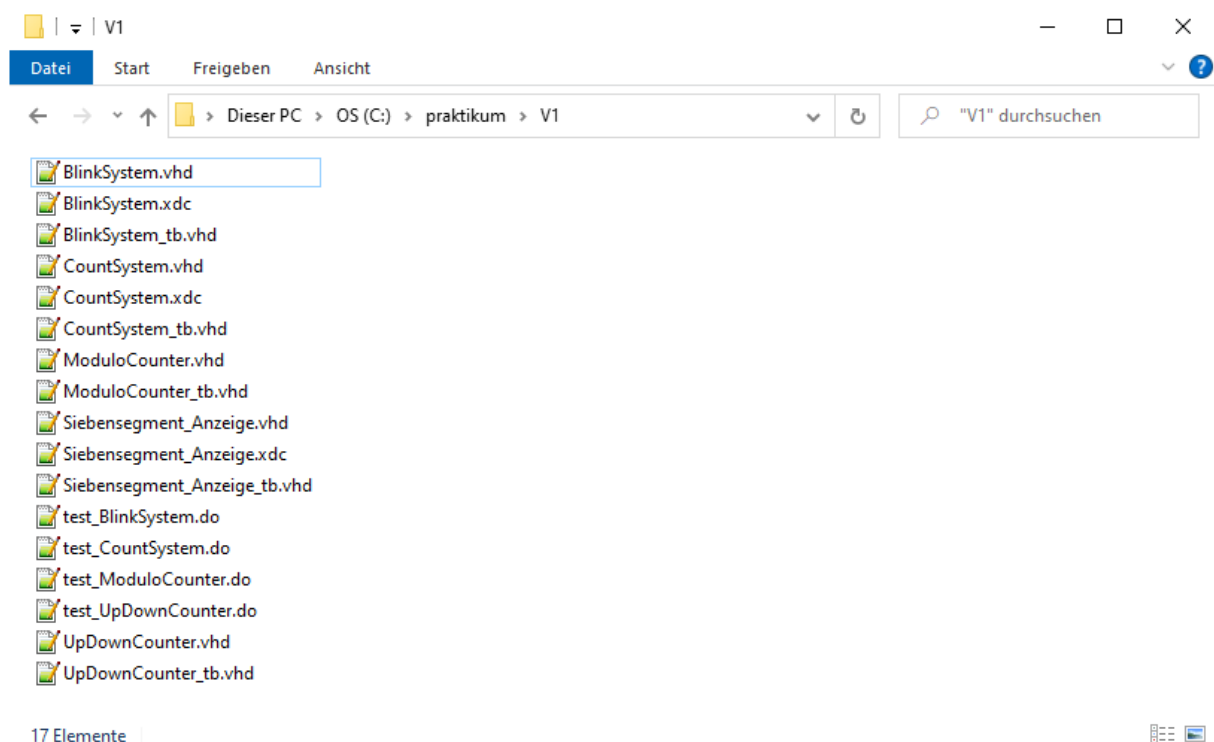
Rainer Höckmann, Hochschule Osnabrück

Bei der Erstellung von VHDL-Komponenten arbeitet man normalerweise mit einer größeren Anzahl von Quelldateien. Von den Simulations- und Synthese-Tools werden dazu noch viele Dateien automatisch erzeugt. Um hier nicht den Überblick zu verlieren, ist es sinnvoll, eine gewisse Struktur bei der Ablage von Dateien einzuhalten. Auf den Rechnern im Labor steht der Ordner `c:\praktikum` zur Verfügung. Verwenden Sie bei allen Verzeichnis- und Dateinamen keine Umlaute, keine Leerzeichen oder andere Sonderzeichen mit Ausnahme von Unterstrichen.

→ Legen Sie ein Arbeitsverzeichnis für den Versuch an



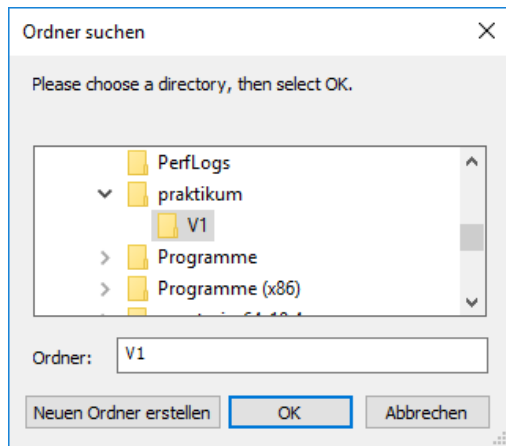
→ Kopieren Sie in dieses Verzeichnis die vorgegebenen und die von Ihnen vorbereiteten Dateien



→ Starten Sie nun den Simulator ModelSim

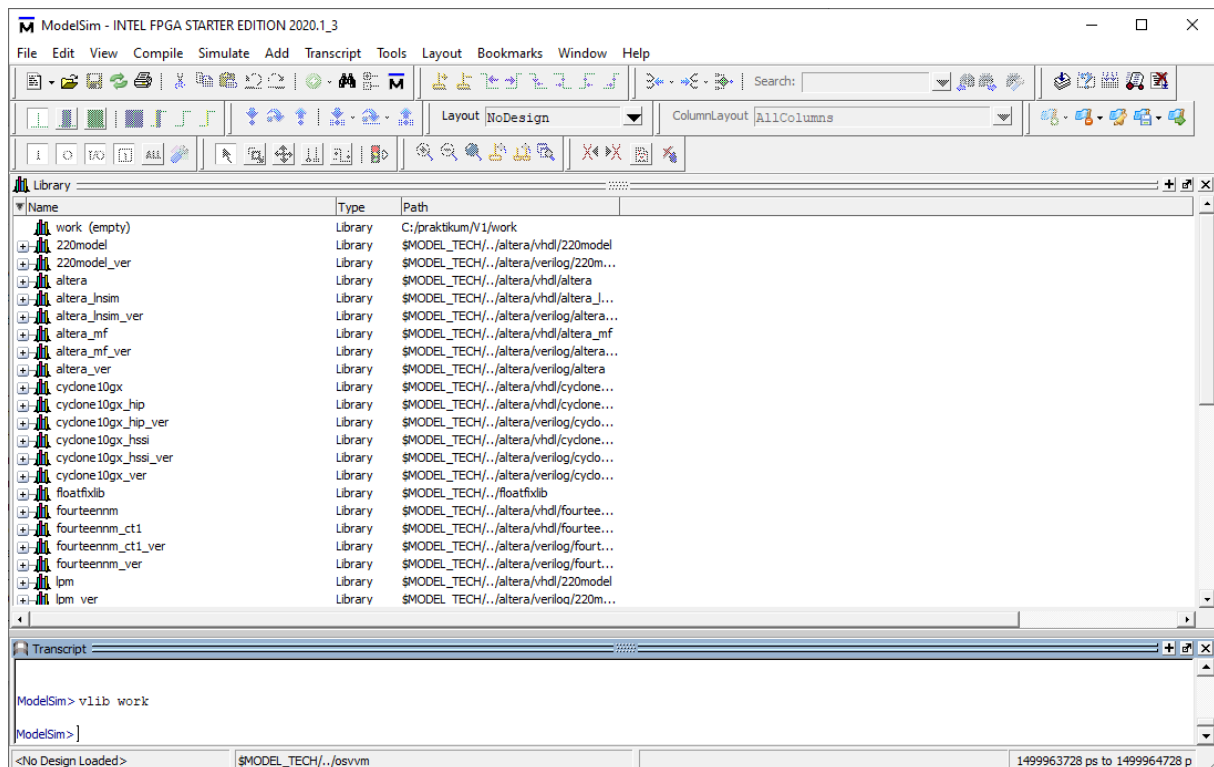
ModelSim verwaltet ein aktuelles Arbeitsverzeichnis. Der einfachste Weg, das Arbeitsverzeichnis zu wechseln führt über den Menüpunkt „File→Change Directory“.

→ Wechseln Sie in das zuvor von Ihnen erstellte Arbeitsverzeichnis



ModelSim lässt sich fast vollständig mit der Maus bedienen. Reproduzierbarer (und mit etwas Übung auch schneller) ist jedoch die Eingabe von Kommandos im „Transcript“-Fenster. Zur Ablage übersetzter Komponenten werden Bibliotheken verwendet. Wir werden im Praktikum eine Bibliothek mit dem Namen „work“ verwenden.

→ Erstellen Sie eine Bibliothek „work“ mit dem Kommando „vlib work“



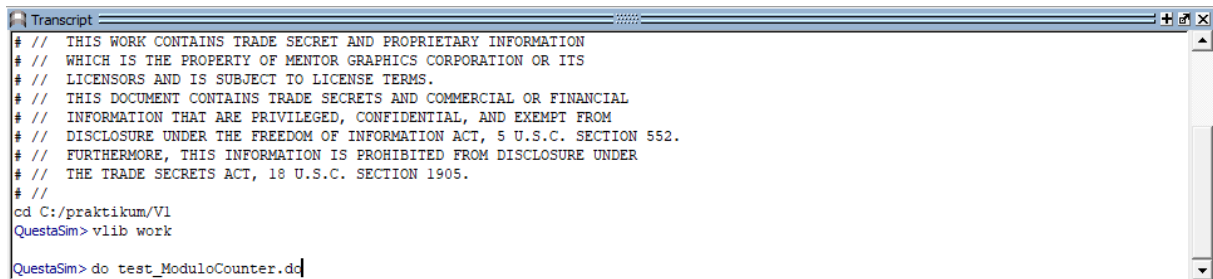
In dem hier dargestellten Workflow werden keine Projekte angelegt. Stattdessen verwenden wir Skriptdateien, um die Vorgänge zur Übersetzung und zur Simulation zu kontrollieren. Die Skriptdateien

werden in der Sprache TCL verfasst und bekommen typischerweise die Dateierendung „.do“ (Sie werden deshalb auch Do-Files genannt). Zeilenkommentare werden entweder durch „--“ oder „#“ eingeleitet.

Zur Bearbeitung der DO-Dateien (sowie der VHD-Dateien) empfehlen wir den Texteditor Notepad++.

Zum Aufruf einer Skript-Datei wird das Kommando „do“ (mit dem Namen des Skripts als Parameter) im Transcript-Fenster verwendet.

→ Starten Sie die Simulation mit der Skript-Datei.







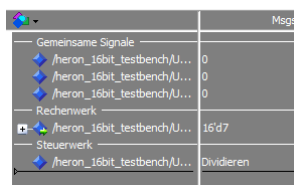
```
# // THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION
# // WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS
# // LICENSORS AND IS SUBJECT TO LICENSE TERMS.
# // THIS DOCUMENT CONTAINS TRADE SECRETS AND COMMERCIAL OR FINANCIAL
# // INFORMATION THAT ARE PRIVILEGED, CONFIDENTIAL, AND EXEMPT FROM
# // DISCLOSURE UNDER THE FREEDOM OF INFORMATION ACT, 5 U.S.C. SECTION 552.
# // FURTHERMORE, THIS INFORMATION IS PROHIBITED FROM DISCLOSURE UNDER
# // THE TRADE SECRETS ACT, 18 U.S.C. SECTION 1905.
# //
cd C:/praktikum/V1
QuestaSim> vlib work
QuestaSim> test_ModuloCounter.do
```

Mögliche Fehlermeldungen werden ebenfalls im Transcript-Fenster ausgegeben. Falls Syntax-Fehler gemeldet werden, sollten diese der Reihe nach behoben werden. Zur erneuten Ausführung eines Kommandos reicht es, den vorherigen Befehl mit den Cursor-Tasten auszuwählen und mit der Enter-Taste zu bestätigen.


→ Korrigieren Sie erkannte Fehler und führen Sie die Skript-Datei erneut aus





Die Simulation wird mit einer Testbench durchgeführt, welche die Eingangs-Ports auf bestimmte Werte setzt (Stimuli) und die Werte der Ausgangs-Ports überprüft (Verifikation).

Zur Fehlersuche wird das „Wave“-Fenster verwendet. Dieses kann durch Klick auf die Schaltfläche  abgedockt werden. Die Schwarze Lupe  kann man verwenden, um eine grobe Übersicht über die Simulationsergebnisse zu bekommen. Mit der linken Maustaste kann man einen Cursor platzieren und mit den Zoom-Lupen   hinein- und herauszoomen. Durch Druck auf die mittlere Maustaste kann ein Bereich zum Hineinzoomen markiert werden. Je nach Einstellung des Simulators werden die Signalnamen ungünstig mit ihren vollständigen Pfaden dargestellt.



| | Msgs |
|-----------------------------|------------|
| Gemeinsame Signale | |
| /heron_16bit_testbench/U... | 0 |
| /heron_16bit_testbench/U... | 0 |
| /heron_16bit_testbench/U... | 0 |
| Rechenwerk | |
| /heron_16bit_testbench/U... | 16d7 |
| Steuerwerk | |
| /heron_16bit_testbench/U... | Dividieren |

Dies kann temporär mit der Schaltfläche  korrigiert werden. (Sie können auch in der Skriptdatei das Kommando „configure wave -signalnamewidth 1“ ergänzen)

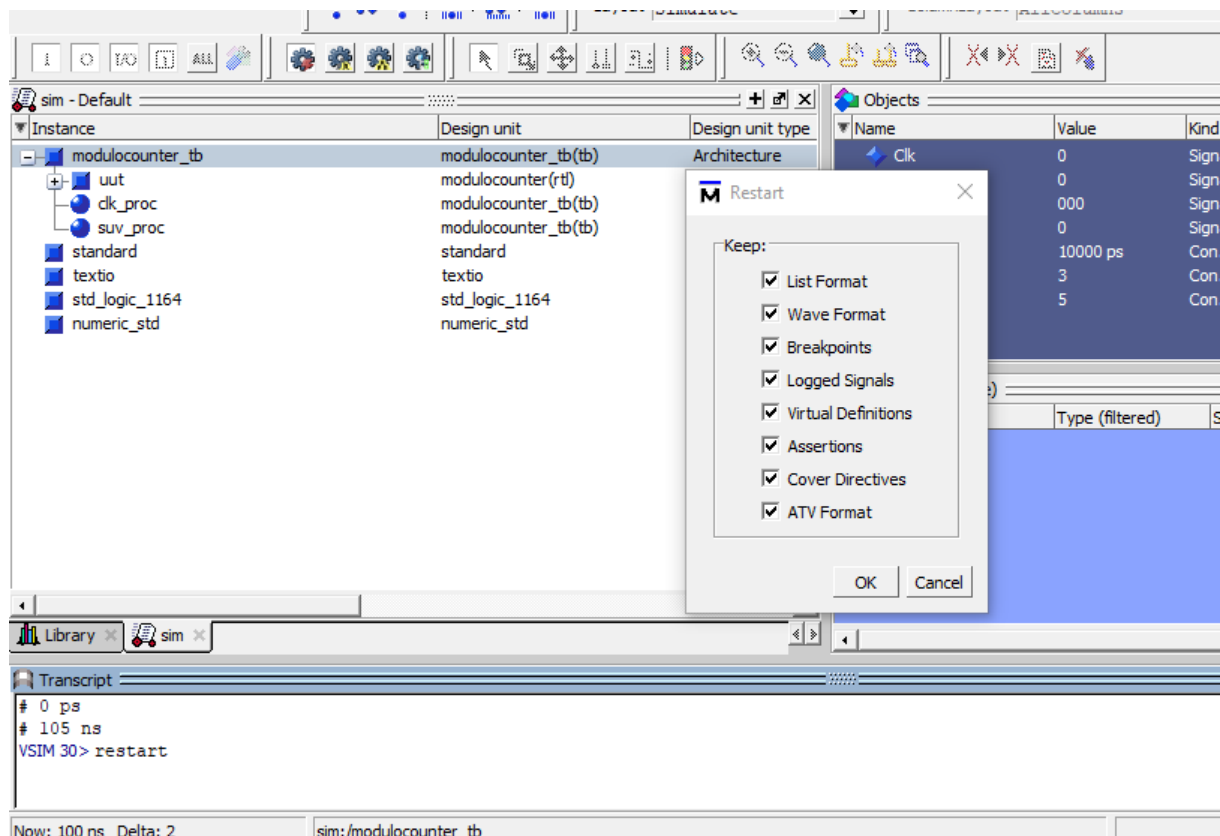
Zur Navigation innerhalb des Wave-Fensters kann man die Schaltflächen zum Aufsuchen von Signalflanken     verwenden. Bei Vektoren kann auch gezielt nach Werten gesucht werden. Dazu klicken Sie in der ersten Spalte des Wave-Fensters den Signalnamen an und tragen dann im

Search-Fenster den gesuchten Wert ein. Mit den Pfeilen unter den Fernglas-Icons können Sie dann den Wert suchen



Während der Fehlersuche möchte man manchmal auch Signale sehen, die nicht dauerhaft im Skript enthalten sein sollen. In diesem Fall kann man per Maus oder per Textkommando Signale hinzufügen und dann mit dem Kommando „restart“ die Simulationszeit auf „0“ zurücksetzen. Die daraufhin erscheinende Nachfrage kann in der Regel mit „OK“ bestätigt werden.

Zum erneuten Ausführen der Simulation wird dann das Kommando „run“ mit der gewünschten Zeitdauer als Parameter verwendet.



```

Transcript
# 0 ps
# 105 ns
VSIM 30> restart

Now: 100 ns Delta: 2    sim:/modulocounter_tb

Transcript
# ** Note: Alle Tests abgeschlossen
#   Time: 90 ns  Iteration: 1  Instance: /modulocounter_tb
# 0 ns
# 105 ns
VSIM 4> restart
# ** Note: (vsim-8009) Loading existing optimized design_opt
#
VSIM 5> run 100 ns

```