



Ausgabe: 02. November 2015

Abgaben	{		Theorie	entfällt
			Praxis	entfällt
			Rücksprache	entfällt

Hinweis zum Ausführen der Simulationen

Aufgrund der Lizenzknappheit werden wir ab diesem Aufgabenblatt ein *Makefile* für die Simulation in Kombination mit einem freien Betrachter verwenden. Zum schreiben der VHDL-Dateien kann jeder beliebige Editor (z.B. *gedit*, *vim*, *kate*, ...) verwendet werden.

Nachdem Sie ihren Quellcode geschrieben haben können Sie in einem *Terminal* in das Verzeichnis, welches das Makefile des aktuellen Vorgaben beinhaltet navigieren und das Kommando `make clean all` ausführen. Dieses Kommando kann erst ausgeführt werden, nachdem Sie in dem Terminal die notwendigen Tools zur Durchführung des Praktikums durch das Kommando `techgi2pr` aktiviert haben.

Das `make`-Kommando führt die Simulation aus und alle Meldungen, welche während des Simulationsprozesses auftreten, werden im Terminal angezeigt. Zum Betrachten der Signalverläufe können Sie die Datei `waveform.vcd` mit dem Betrachter *GtkWave* öffnen.

Geben Sie dazu in dem Terminal, in dem Sie den `make`-Befehl ausgeführt haben, das Kommando `gtkwave waveform.vcd` ein. Eine kurze Anleitung zur Nutzung von *GtkWave* können Sie auf der *ISIS*-Seite finden.

Werden in einem Praktikumstermin mehrere Testbenches verwendet, muss das *Makefile* an die jeweilige Testbench angepasst werden. Öffnen Sie dazu die Datei *Makefile*, welche sie im Wurzelverzeichnis der Vorgaben finden mit einem Texteditor und befolgen Sie die Anweisungen, welche Sie in den Kommentarzeilen des *Makefiles* (# am Beginn der Zeile) finden.

Hinweis zum Testen der Aufgaben 2 bis 4

Die Aufgaben 2 bis 4 verwenden die gleiche Testbench `logic_tb.vhd`, welche alle zu implementierenden Gatter testet. Sollten Sie die Testbench ausführen bevor Sie alle Gatter implementiert haben, werden Fehlermeldungen angezeigt. Vergewissern Sie sich in diesem Fall, dass die Fehlermeldungen nicht das Gatter, welches Sie bereits implementiert haben betrifft und betrachten Sie den Signalverlauf um sich von der Funktionalität des Gatters zu überzeugen.

Aufgabe 1: Gitlab-Anmeldung

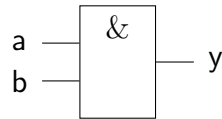
Diese Aufgabe ist von **allen Gruppenmitgliedern mit ihrem eigenen *TUBIT*-Login durchzuführen!**

Für die Durchführung der Abgaben wird in diesem Praktikum das *Gitlab* der Universität genutzt. Um die Gruppenzuordnungen dort abbilden zu können, müssen alle Teilnehmer des Praktikums bei dem *Gitlab* der Universität angemeldet sein.

Öffnen Sie die Seite https://gitlab.tubit.tu-berlin.de/users/sign_in und melden Sie sich mit ihren *TUBIT*-Login-Daten an! Dadurch wird ihr *Gitlab*-Konto automatisch angelegt,

weitere Informationen zu der Abgabe und der Benutzung des *Gitlab* werden in den folgenden Aufgabenblättern gegeben.

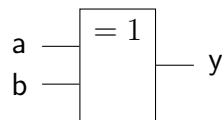
Aufgabe 2: AND2-Gatter



a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

1. Legen Sie eine neue Datei `and2.vhd` im Wurzelverzeichnis der Vorgaben für dieses Gatter an.
 2. Beschreiben Sie das Gatter in VHDL.
Implementieren Sie die entsprechend benannte `entity` und die `architecture "behavioral"`.
 3. Testen Sie Ihre Implementierung mit der Testbench `logic_tb.vhd` aus den Vorgaben.
-

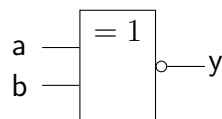
Aufgabe 3: XOR2-Gatter



a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

1. Legen Sie eine neue Datei `xor2.vhd` im Wurzelverzeichnis der Vorgaben für dieses Gatter an.
 2. Beschreiben Sie das Gatter in VHDL.
Implementieren Sie die entsprechend benannte `entity` und die `architecture "behavioral"`.
 3. Testen Sie Ihre Implementierung mit der Testbench `logic_tb.vhd` aus den Vorgaben.
-

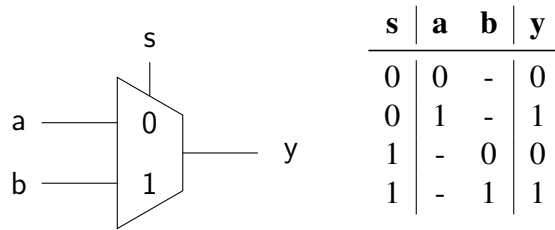
Aufgabe 4: XNOR2-Gatter



a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

1. Legen Sie eine neue Datei `xnor2.vhd` im Wurzelverzeichnis der Vorgaben für dieses Gatter an.
 2. Beschreiben Sie das Gatter in VHDL.
Implementieren Sie die entsprechend benannte `entity` und die `architecture "behavioral"`.
 3. Testen Sie Ihre Implementierung mit der Testbench `logic_tb.vhd` aus den Vorgaben.
-

Aufgabe 5: MUX2



Dieser Multiplexer schaltet abhängig von s die Eingänge a bzw. b nach y durch.

1. Stellen Sie mithilfe der Wertetabelle einen in VHDL gültigen Logik-Term für y auf.
Die in der oberen Wertetabelle mit - gekennzeichneten Eingänge sind für die Funktionalität nicht relevant. Die Funktionalität des Bauteils soll für jeden mögliche Eingangswert für den mit - gekennzeichneten Eingang sichergestellt sein.
2. Legen Sie eine neue Datei `mux2.vhd` im Wurzelverzeichnis der Vorgaben für den Multiplexer an.
3. Implementieren Sie die entsprechend benannte `entity` und die `architecture "behavioral"`.
4. Testen Sie Ihre Implementierung mit der entsprechenden Testbench aus den Vorgaben.

Literatur

- [1] Mentor Graphics Corporation. *ModelSim SE Reference Manual*, 6.4a edition.
- [2] Mentor Graphics Corporation. *ModelSim SE Tutorial*, 6.4a edition.
- [3] Mentor Graphics Corporation. *ModelSim SE User's Manual*, 6.4a edition.