



Ausgabe: 06. November 2017

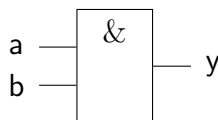
Abgaben	{		Theorie	entfällt
			Praxis	entfällt
			Rücksprache	entfällt

## Aufgabe 1: Logik-Gatter

In dieser Aufgabe sollen 3 grundlegende Gatter von Ihnen implementiert werden. Die Definition der Gatter-Funktionalität ist jeweils durch eine Wertetabelle vorgegeben.

Die Funktionalität aller drei Gatter kann durch die Testbench `logic_tb` getestet werden. Beachten Sie dabei, dass in den vorgegebenen Dateien für die drei Gatter sowohl ein *entity*- als auch ein *architecture*-Abschnitt vorhanden sein muss, bevor die Testbench ausgeführt werden kann. Sollten Sie erst ein Teil der Gatter implementiert haben, können Sie die Fehler, welche die Testbench zu den noch nicht implementierten Gattern anzeigt, ignorieren. Zum Ausführen der Testbench müssen Sie in einem Terminal in den Ordner navigieren, in den Sie die Dateien aus der Vorgabe zu dieser Aufgabe entpackt haben. Führen Sie dann in dem Terminal den Befehl `make clean all` aus. Das Betrachten der Signalverläufe erfolgt dann mit dem Kommando `make view_wave`.

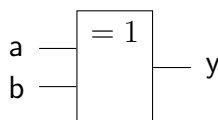
### 1. Das AND-Gatter



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

Implementieren Sie die Funktionalität des *AND*-Gatters in der Datei `and2.vhd`. Definieren Sie dafür eine neue *entity* mit dem Namen *and2* und eine *architecture* mit dem Namen *behavioral*.

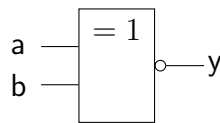
### 2. Das XOR-Gatter



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

Implementieren Sie die Funktionalität des *XOR*-Gatters in der Datei `xor2.vhd`. Definieren Sie dafür eine neue *entity* mit dem Namen *xor2* und eine *architecture* mit dem Namen *behavioral*.

### 3. Das XNOR-Gatter



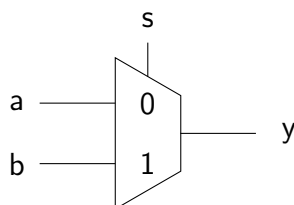
a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

Implementieren Sie die Funktionalität des *XNOR*-Gatters in der Datei `xnor2.vhd`. Definieren Sie dafür eine neue *entity* mit dem Namen `xnor2` und eine *architecture* mit dem Namen `behavioral`.

### 4. Simulation

Testen Sie Ihre Implementierung mit der Testbench `logic_tb` aus den Vorgaben.

## Aufgabe 2: MUX2



s	a	b	y
0	0	-	0
0	1	-	1
1	-	0	0
1	-	1	1

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 Bestimmung Ausgabewerts nicht relevant. Die Funktionalität des Bauteils soll daher für jeden mögliche Eingangswert an einem mit - gekennzeichneten Eingang sichergestellt sein.
2. Legen Sie für den Multiplexer eine neue Datei `mux2.vhd` im Wurzelverzeichnis der Vorgaben an.
3. Implementieren Sie die entsprechend benannte *entity* und die *architecture* "behavioral".
4. Testen Sie Ihre Implementierung mit der Testbench `mux_tb` aus den Vorgaben.

## Literatur

- [1] Jonas Tröger Max Uffke Drechsler. Gtkwavemod-repository. <https://gitlab.tubit.tu-berlin.de/jonas.e.troeger/GtkWave-Mod>.
- [2] Mentor Graphics Corporation. *ModelSim SE Reference Manual*, 6.4a edition.
- [3] Mentor Graphics Corporation. *ModelSim SE Tutorial*, 6.4a edition.
- [4] Mentor Graphics Corporation. *ModelSim SE User's Manual*, 6.4a edition.