2. Blatt	Fachgebiet Architektur eingebetteter Systeme **Rechnerorganisation Praktikum**	AES Embedded Systems Architecture WS2023/24

Ausgabe: 06. November 2023

Abgaben
- Theorie: entfällt
- Praxis: entfällt
- Rücksprache: entfällt

Aufgabe 1: NEQ4

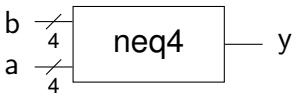

Abbildung 1: Entity neq4

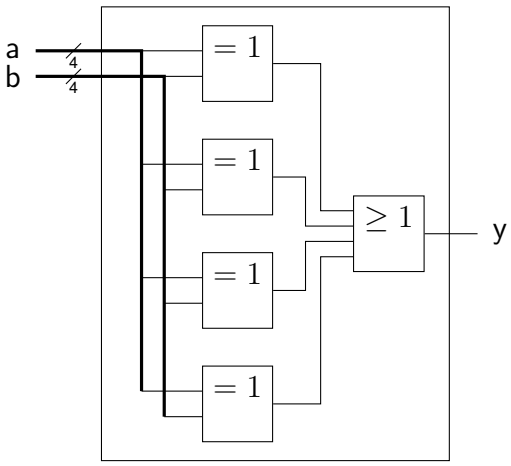

Abbildung 2: Architecture neq4

Name	Typ	in / out	Beschreibung
a	std_logic_vector(3 downto 0)	in	-
b	std_logic_vector(3 downto 0)	in	-
y	std_logic	out	Ergebnis des Vergleichs $a! = b$

In den folgenden Aufgaben sollen zu der oben gezeigten Antivalenzgatter-Schnittstelle mehrere Beschreibungsalternativen in Form von Architekturen auf *Logik-* und *Strukturebene* implementiert werden.
Dabei sollen alle Varianten des Antivalenzgatters identische Funktionalität aufweisen, d.h. nur für identische Eingangsvektoren `a` und `b` nimmt der Ausgang `y` einen *low* Pegel an, sonst ist dieser *high*.

1. Implementieren Sie zur Schnittstelle `neq4` die Architektur `logic` in der vorgegebenen Datei `neq4.vhd`. Die Architektur soll den Ausgang `y` des Antivalenzgatters mittels der in VHDL vordefinierten *Logik* (Elementarfunktionen) aus den Komponenten der Eingangsvektoren `a` und `b` berechnen.

2. Implementieren Sie die Architektur `netlist` des Antivalenzgatters `neq4` auf *struktureller Ebene* durch Instanziierung und Verdrahtung von geeigneten Elementargattern. Die Implementie-

rung erfolgt ebenfalls in der Datei `neq4.vhd`. Greifen Sie dabei auf die geschriebenen zweistelligen Logikgatter aus dem 0. bzw. 1. Aufgabenblatt zurück.

3. Testen Sie ihre Implementierung mit Hilfe der vorgegebenen Testbench `neq4_tb`. Rufen Sie dazu in dem Aufgabenordner der Vorgaben das Kommando `make clean all` in der Konsole auf. Die Testbench testet beide Implementierungen gleichzeitig. Sollten Sie den Test ausführen, während Sie nur eine Implementierung geschrieben haben, so können Sie die Fehler, welche die noch nicht geschriebene Implementierung betreffen, ignorieren. Die Datei `neq4.vhd` muss allerdings syntaktisch korrekt sein, damit die Simulation durchgeführt werden kann.
   Eine Betrachtung der Signalverläufe kann durch den Aufruf des Kommandos `make view_wave` erfolgen. Im Betrachter finden Sie die Fehlermeldungen im unteren, rechten Fenster. Durch einen Doppelklick auf den Fehler kann der Cursor im Signalfenster an die entsprechende Stelle verschoben werden.

Aufgabe 2: MUX2

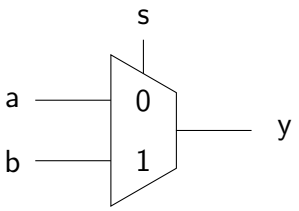

Abbildung 3: Entity mux2

s	a	b	y
0	0	–	0
0	1	–	1
1	–	0	0
1	–	1	1
U	–	–	X
X	–	–	X

Auf dem vorhergehenden Blatt haben Sie einen Multiplexer mit zwei Eingängen auf der Ebene von Logikgattern beschrieben. Nun sollen Sie dieselbe Schaltung noch einmal implementieren, dabei aber ein VHDL-Sprachelement nutzen, was dieses Verhalten auf einer höheren Abstraktionsebene beschreibt. Der Multiplexer soll außerdem eine Fehlerfortpflanzung erlauben, das heißt, dass im Falle das `s` nicht aus ‘0‘, ‘1‘ besteht, ein Fehler in Form von ‘X‘ an den Ausgang durchgereicht werden soll.

1. Implementieren Sie Ihren Multiplexer in der `architecture behavioral`, welche Sie in der vorgegebenen Datei `mux2.vhd` finden.
2. Verfizieren Sie Ihr Design mithilfe der vorgegebenen Testbench, indem Sie in dem Aufgabenordner das Kommando `make clean all` ausführen.

Literatur

[1] Jonas Tröger Max Uffke Drechsler. Gtkwavemod-repository. `https://git.tu-berlin.de/rorgpr-wise23/GTKWaveMod`.

[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.