georg.von-der-brueggen [☺] tu-dortmund.de
harun.teper [☺] tu-dortmund.de

Hardwarepraktikum
Wintersemester 21/22

# Übungsblatt 2 – 13 Punkte

**(Block 1 – insgesamt 60 Punkte)**

**Bearbeiten ab Samstag, 23. Oktober 2021.**
**Abgabe bis spätestens Freitag, 29. Oktober 2021, 23:59 Uhr.**

Hinweise: (1) VCD-Dateien müssen nicht mit abgegeben werden.
(2) Bitte gebt für jeden Aufgabenteil separate Quellcode-Dateien ab.
(3) Quellcode-Dateien müssen **kommentiert** und **kompilierbar** sein.

## 2.1 Modellierung von Signalwerten (IEEE 1164) (4 Punkte)

Spannungs- und Stromverläufe sind in der Realität kontinuierlich, das heißt es kann jeder beliebige Wert angenommen werden. In der Digitaltechnik beschränkt man sich auf diskrete Werte, im einfachsten Fall auf zwei, welche die logischen Zustände 0 und 1 repräsentieren. Um das interne Verhalten von Gattern zu verstehen, bedarf es noch einiger weiterer Zustände, welche im Standard IEEE 1164 definiert sind. Um diesen Standard in VHDL einzuhalten findet ihr am Anfang jeder VHDL-Datei folgende Codezeilen:

```
library IEEE;
use IEEE.std_logic_1164.all;
```

In IEEE 1164 werden die logischen Zustände der Signale definiert, welche die fundamentalen Einheiten in VHDL darstellen. Mit einem Signal vom Typ *std_logic*, welches in IEEE 1164 definiert ist, kann gerechnet werden. Dazu werden die Operatoren und Keywords durch die IEEE Library importiert. Solche Signale können unter anderem folgende Werte annehmen:

```
0, starker Zustand 0, starke Null
1, starker Zustand 1, starke Eins
L, schwacher Zustand 0, schwache Null
H, schwacher Zustand 1, schwache Eins
X, starker unbestimmter Zustand, unbestimmt
W, schwacher Zustand, schwach unbestimmt
Z, abgetrennter Zustand, abgetrennt
```

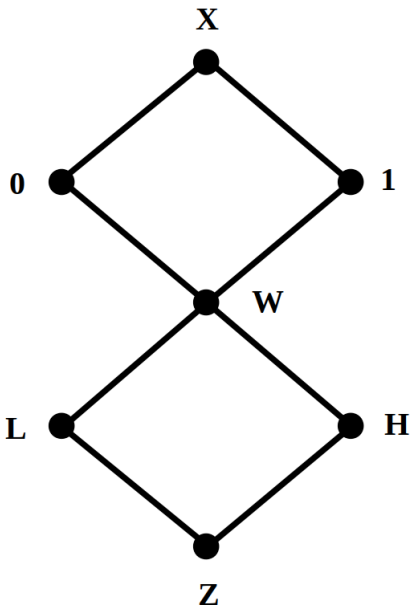
Erläuterungen dazu:

- 0 und 1 repräsentieren eine logische Null bzw. eine logische Eins. Schaltungstechnisch sind es direkte Anschlüsse an GND (ground bzw. Masse) bzw. Vcc (positive support voltage bzw. Versorgungsspannung) mit keinem oder nur sehr geringem Widerstand. Oft wird in der Praxis mit GND = 0 V und Vcc = 5 V gearbeitet.
- L und H repräsentieren ebenfalls eine logische Null bzw. eine logische Eins. Ein Beispiel dazu wäre wenn GND bzw. Vcc durch Widerstände (z.B. 100 Ohm) gedämpft werden.
- X und W repräsentieren Kurzschlüsse zwischen (0,1) bzw. (L,H). Sie stellen die unbestimmten logischen Zustände dar.
- Z ist der Zustand auf einer nicht angeschlossenen Leitung oder an einem Ausgang eines gesperrten Transistors.

Es stellt sich nun die Frage, was passiert, wenn zwei Zustände in einem Leiterknoten aufeinandertreffen. Die Namen der Zustände geben schon erste Hinweise darauf, welche Zustände sich im Zweifelsfall durchsetzen werden. Wir definieren eine Funktion Cond:

$$Cond : \{0,1,L,H,X,W,Z\}^2 \rightarrow \{0,1,L,H,X,W,Z\}$$

und betrachten zur Festlegung der Funktionswerte die Relation „stärker als“. Ein Teil dieser Relation kann graphisch durch ein Hasse-Diagramm dargestellt werden.




Eine Verbindung zwischen zwei Knoten gibt an, dass das Paar zur Relation gehört. Es bedeutet, dass das höher stehende Element stärker ist als das niedrigere. Bildet man nun die transitive Hülle, so erhält man die gesamte Relation.

Die Relation „stärker als“ induziert eine Halbordnung auf $\{0,1,L,H,X,W,Z\}$. Gemäß dieser Halbordnung ist nun $Cond(x,y)$ definiert als das Supremum $(x,y)$. Da wir es hier mit einer endlichen Menge zu tun haben, ist das Supremum also das kleinste Element, das gerade noch stärker ist als beide Argumente. Falls „x stärker als y“ gilt (entweder direkt aus dem Hasse-Diagramm oder aus der transitiven Hülle ableitbar), dann ist das Supremum gleich dem Maximum von x und y. Die Relation „stärker als“ ist transitiv.

**Aufgaben:**

a. (1 Punkt) Zeigt, dass $Cond(0,Z) = 0$ ist. Verwendet dazu folgende Notation: $x \geq y$ bedeutet „x ist stärker als y“.

b. (1 Punkt) Welche weiteren Zustände gibt es und wofür werden sie gebraucht ? Geht besonders auf die Anwendungsfälle der *Don't care* ein. Schaut dazu auch in der Datei $ghdl/libraries/ieee/std\_logic\_1164.vhdl$ nach (Link in [1]). Was ist in dieser Datei definiert?

c. (2 Punkte) In der Datei $signal\_levels.vhdl$ wurde eine einfache Testbench – eine Testumgebung für Komponenten in VHDL – implementiert. Wählt 8 verschiedene Tupel $(a,b)$ aus $\{0,1,L,H,X,W,Z\}$, wobei Signalwerte eines Tupels unterschiedlich sein müssen, und wendet die Operatoren $\{and,or\}$ auf die beiden Elemente im Tupel an (insgesamt also 16 Auswertungen). Erweitert dazu die Testbench so, dass die entsprechenden Ergebnisse als Signal ausgegeben werden. Kompiliert die Datei, lest die Signalwerte in GTKWave ab, und notiert die Ergebnisse (im Quellcode als Kommentar oder in einer separaten Datei).

   *Hinweis:* Hier sollen keine Logikgatter für die Operatoren $\{and,or\}$ erstellt werden. Überprüft die korrekte Ausgabe eurer Testbench indem ihr unter anderem das Tupel $(0,1)$ auswertet.

## 2.2 Logikgatter in VHDL (9 Punkte)

Im zweiten Teil dieser Übung werden wir erst das Wissen über Logikgatter auffrischen. Danach legt ihr die Bausteine in VHDL an und testet deren Funktionalität.

**Aufgaben:**

a. (2 Punkte) Zeichnet die Gatter für NOT, AND, NAND, OR, NOR, XOR, XNOR (IEC 60617-12 Norm) mit zwei Eingängen (falls möglich) und dazu die Wahrheitstabellen. Zum Auffrischen könnt ihr z.B. in den Folien von Rechnerstrukturen nachschlagen.

b. (2 Punkte) In der Datei *and_gate.vhdl* wurde ein AND-Gatter implementiert. Nutzt diese Implementierung als Vorlage, um die restlichen Gatter (in Aufgabenteil a) ebenfalls zu erstellen. Testet eure Implementierung mit allen Kombinationen der Eingabesignale 0 und 1 in einer Testbench und überprüft die Korrektheit der Gatter anhand der Wellenform in GTKWave. Nutzt dabei seperate Dateien für jedes Gatter und jede Testbench.

c. (1 Punkt) Erklärt warum Delays bei der Modellierung von digitalen Schaltungen berücksichtigt werden müssen. Geht dabei genauer auf die Unterschiede zwischen Transportverzögerung (transport delay) und Trägheitsverzögerung (inertial delay) ein.

d. (1 Punkt) Sucht euch nun ein Gatter außer NOT aus und verzögert dessen Ausgabe mit der Delay-Anweisung *transport* um 15 ns. Bei der Simulation der Verzögerung könnt ihr nun feststellen, dass euer Gatter auch bei beliebig kurzen Impulsen reagiert. Ändert das Gatter so ab, dass Impulse, die kürzer als 10 ns sind, ignoriert werden. Ändert die Testbench so, dass sie dieses Verhalten zeigt und kommentiert wo in der Testbench die unterdrückten/ignorierten Peaks sind. Nutzt separate Dateien für die Implementierung der Gatter mit den beiden verschiedenen Delay-Typen und separate Dateien für die Testbenches.

e. (1 Punkt) Erweitert danach das Gatter aus Aufgabenteil d) um einen weiteren Eingang (das Gatter soll also drei Eingänge haben) und testet alle Inputkombinationen. Implementiert das Gatter so, dass zu kurze Impulse ignoriert werden. Testet, ob zu kurze Signale korrekt ignoriert werden. Nutzt auch hier separate Dateien für die Implementierung des Gatters und der Testbench.

f. (2 Punkte) Arbeitet nun auch das zweite Kapitel im Skript durch und beantwortet danach die folgenden VHDL-Fragen. Erläutert kurz mit Kommentaren in eurem Quellcode aus Aufgabenteil e) oder in einem kurzen Text.

   1) Wofür werden die Keywords *entity* und *architecture* genutzt?
   2) Wofür wird *component* genutzt?
   3) Was macht *port map*?
   4) Wie werden Befehle in einem *process begin* Block ausgeführt und wie unterscheidet sich die Ausführung eines solchen Blocks wenn *process* nicht genutzt wird? (Stichworte: sequentiell/parallel)

## Literatur

[1] IEEE 1164 in GHDL.
`https://github.com/ghdl/ghdl/blob/master/libraries/ieee/std_logic_1164.vhdl`