georg.von-der-brueggen [☺] tu-dortmund.de
leonard.bereholschi [☺] tu-dortmund.de

Hardwarepraktikum
Wintersemester 25/26

Übungsblatt 2 – 14 Punkte

(Block A – insgesamt 58 Punkte)

Bearbeiten ab Samstag, 25. Oktober 2025.
Abgabe bis spätestens Freitag, 31. Oktober 2025, 23:59 Uhr.

Hinweis: Die bereitgestellten Dateien finden Sie in Blatt02.zip. Dort findet sich ebenfalls die Grundstruktur für die Abgabe sowie einige weitere Hinweise in der Datei hinweis.txt.
Im Hardware Praktikum sollen Sie den Aufbau von digitalen Schaltungen, vom einfachen Gatter bis hin zum Prozessor, kennenlernen. Dabei wird viel auf der Ebene der Binärwerte gearbeitet werden, die Computer in ihren Berechnungen verwenden. Diese werden intern durch anliegen bzw nicht anliegen von Spannung repräsentiert. Auf diesem Übungsblatt beschäftigen Sie sich zunächst mit diesen Signalwerten. Danach werden Sie die ersten einfachen Gatter erstellen die logische Operationen für diese Signalwerte definieren und die Basis für größere Schaltungen darstellen.

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 finden Sie 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
```

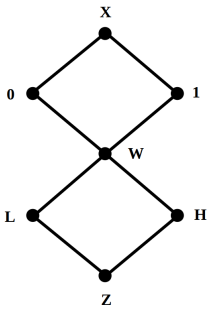
Erläuterungen dazu:

- 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) Zeigen Sie, dass $Cond(0,Z) = 0$ ist. Verwenden Sie 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? Gehen Sie besonders auf die Anwendungsfälle von *Don't care* ein. Schauen Sie 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 $signals\_tb.vhdl$ wurde eine einfache Testbench – eine Testumgebung für Komponenten in VHDL – implementiert. Wählen Sie 8 verschiedene Tupel (a,b) aus $\{0,1,L,H,X,W,Z\}$, wobei die Signalwerte eines Tupels unterschiedlich sein müssen, und wenden Sie die Operatoren $\{and,or\}$ auf die beiden Elemente im Tupel an (insgesamt also 16 Auswertungen). Erweitert Sie dazu die Testbench so, dass die entsprechenden Ergebnisse als Signal ausgegeben werden. Kompilieren Sie die Datei, lesen die Signalwerte in GTKWave ab, und notieren Sie die Ergebnisse (im Quellcode als Kommentar oder im pdf zur Abgabe).

   Hinweis: Hier sollen keine Logikgatter für die Operatoren $\{and,or\}$ erstellt werden. Überprüfen Sie die korrekte Ausgabe Ihrer Testbench indem Sie unter anderem das vorgegebene Tupel $(0,1)$ auswerten.

2.2 VHDL Aufbau (2 Punkte)

Arbeiten Sie das zweite Kapitel im Skript durch und beantworten Sie die folgenden VHDL-Fragen.

a. Wofür werden die Keywords *entity* und *architecture* genutzt?

b. Wofür wird *component* genutzt?

c. Was macht *port map*?

d. 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)

2.3 Logikgatter in VHDL (8 Punkte)

Im dritten Teil dieser Übung werden wir zunächst das Wissen über Logikgatter auffrischen. Danach legen Sie diese Bausteine in VHDL an und testen deren Funktionalität.

Aufgaben:

a. (1,5 Punkte) Geben Sie die Wahrheitstabellen für die Gatter AND, NAND, OR, NOR, XOR, XNOR mit zwei Eingängen an. (Hinweis: Sie können eine Wertetabelle mit 6 Ergebnisspalten für alle 6 Gatter verwenden.)

b. (2,5 Punkte) In der beigelegten Datei *and_gate.vhdl* wurde ein AND-Gatter implementiert. Implementieren Sie die restlichen 5 Gatter aus Aufgabenteil 2.3.a (NAND, OR, NOR, XOR, XNOR). Orientieren Sie sich dabei an der beigelegten Datei *and_gate.vhdl* als Vorlage für die generelle Struktur der Gatter. Nutzen Sie dabei auch für die Gatter mit Negierung (z.B. NAND) direkt die entsprechenden logischen Operationen und schalten Sie nicht, wie in der Beispielabgabe die Teil von Übungsblatt 1 ist, ein NOT-Gatter hinter ein anderes Gatter. Testen Sie Ihre Implementierung mit allen Kombinationen der Eingabesignale 0 und 1 in einer Testbench und überprüfen Sie die Korrektheit der Gatter anhand der Wellenform in GTKWave. Nutzen Sie dabei separate Dateien für jedes Gatter und jede Testbench.

c. (2 Punkt) Erklären Sie warum Delays bei der Modellierung von digitalen Schaltungen berücksichtigt werden müssen. Gehen Sie dabei genauer auf die Unterschiede zwischen Transportverzögerung (transport delay) und Trägheitsverzögerung (inertial delay) ein.

d. (1 Punkt) Verzögeren Sie die Ausgabe des AND-Gatters (bereitgestellt in *and_gate.vhdl*) mit der Delay-Anweisung *transport* um 15 ns. Erstellen Sie eine Testbench, welche die Verzögerung des Signals zeigt.

e. (1 Punkt) Bei der Simulation der Verzögerung in Aufgabenteil 2.3.d konnten Sie feststellen, dass Ihr Gatter auch bei beliebig kurzen Impulsen reagiert. Erstellen Sie eine Kopie des Gatters aus Aufgabenteil d und erweitern Sie es, sodass Impulse die kürzer als 10ns sind ignoriert werden. Erstellen Sie für dieses Gatter eine zusätzliche Testbench, welche das Verhalten zeigt. Kommentieren Sie außerdem in der Testbench, an welchen Stellen die Signalwechsel durch die Erweiterung unterdrückt werden.

Hinweis: Halten Sie sich bei der Abgabe im Moodle bitte an die folgenden Konventionen:
(1) Geben Sie den schriftlichen Teil der Abgaben bitte in einer zusammenhängenden pdf-Datei ab.
(2) Verwenden Sie zur Abgabe von Quellcode Containerdateien (zip). Am besten eine Datei mit mehreren Unterverzeichnissen. Oft kann hier die vorgebenen Verzeichnisstruktur der zur Verfügung gestellten Dateien übernommen werden. Verwenden Sie dabei die Bezeichnung *Gruppe"Nummer"_Blatt"Nummer".zip* (wahlweise mit Namenszusatz), z.B. für Gruppe 253, Blatt 2: *Gruppe253_Blatt2.zip* oder *Gruppe253_Blatt2.zip_(Ada_Lovelace_und_Konrad_Zuse).zip*
(3) Quellcode-Dateien müssen **kommentiert** und **kompilierbar** sein. Bitte kommentieren Sie den Code angemessen und ausreichend. Anderenfalls wird diese (Teil-) Aufgabe unter Umständen nicht bewertet.
(4) Wenn bei einer Aufgabe eine Quellcode-Datei (vhdl-Dateien, Scripte, etc) erstellt oder verändert werden soll, geben Sie bitte die entsprechende Datei zusammen mit allen anderen für die Ausführung benötigten Dateien ab (selbst wenn diese Dateien vorgeben sind und nicht verändert wurden oder in einer anderen (Teil-) Aufgabe erstellt wurden). Die abgegebene zip-Datei sollte außerdem das vorgegebene Python Skript sowie eine Datei *command.txt* enthalten die den Befehl angibt mit dem die Abgabe mit Hilfe des vorgegebenen Skriptes ausgeführt werden kann. Wenn z.B. eine Test-Datei für eine vorgegebene Datei geschrieben werden soll, geben Sie also bitte sowohl die Testdatei als auch die vorgegebene Datei ab sowie das Skript und die Datei *command.txt* für die entsprechende Aufgabe. VCD-Dateien und die *work_obj93.cf* Datei sollten ebenfalls mit abgegeben werden. Eine Beispielabgabe finden Sie bei den Dateien die für Übungsblatt 1 zur Verfügung gestellt werden.

Literatur

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