

### Restructure points

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

Hardwarepraktikum  
Wintersemester 25/26

## Übungsblatt 4 – 16 Punkte

(Block A – insgesamt 58 Punkte)

Bearbeiten ab Samstag, 8. November 2025.

Abgabe bis spätestens Freitag, 14. November 2025, 23:59 Uhr.

Beachtet bitte die Hinweise zu den Abgabekonventionen auf Blatt 2 bzw. im Moodle.

Falls Sie dies noch nicht getan haben, setzen Sie sich bitte mit Ihrer Übungsgruppenleitung wegen der Testattermine für den ersten Block in Verbindung.

Nach Gattern und Addierwerken werden Sie auf diesem Übungsblatt weitere wichtigen Bausteine für komplexe Systeme kennenlernen, die kombinatorischen Schaltungen. Die Ausgänge von kombinatorischen Schaltungen werden nur durch ihre Eingangssignale bestimmt. Sie arbeiten also *ohne Gedächtnis*. Zudem sind sie zyklusfrei, wodurch Änderungen an den Eingangssignalen sofort am Ausgang wirksam werden (wenn man von der Schaltverzögerung absieht). Sogar die komplexesten Systeme sind aus vielen einfachen kombinatorischen Schaltungen gebaut.

### 4.1 Multiplexer in VHDL (8 Punkte)

Einer der am häufigsten verwendeten Bausteine ist der *Multiplexer*, der genutzt wird um aus mehreren Eingangssignalen anhand eines Auswahlsignals ein Ausgabesignal zu wählen. Beispielsweise repräsentiert Abbildung 1(a) einen 4:1 Multiplexer, der das Ausgabesignal  $Y$  dadurch bestimmt, dass er einen der Eingänge  $I_0$ ,  $I_1$ ,  $I_2$  und  $I_3$ , anhand der Auswahlssignale  $S_1$  und  $S_0$ , auswählt. Ebenso wählt der 8:1 Multiplexer (Abbildung 1(b)) durch die Auswahlssignale  $S_2$ ,  $S_1$  und  $S_0$  ein Signal der Eingänge  $I_0$  bis  $I_7$  als Ausgabesignal  $Y$  aus.

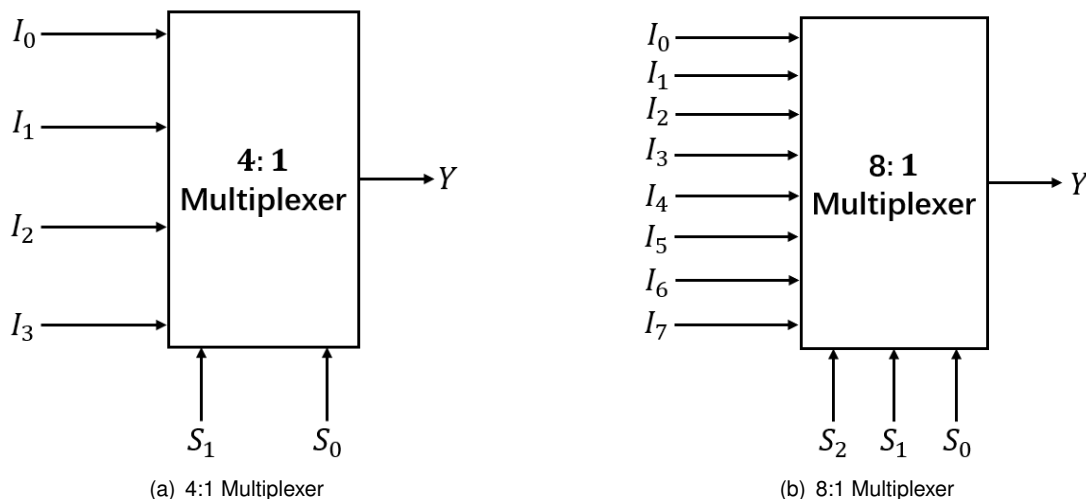


Abbildung 1: Schema eines 4:1 und eines 8:1 Multiplexers.

### Aufgaben:

Multiplexer können auch aus Gruppen von Eingangssignalen der Länge  $n$  auswählen. In dieser Aufgabe soll ein 4:1 Multiplexer implementiert werden, der vier 3-Bit Eingangssignale besitzt. In der Wahrheitstabelle 1 wurden die Eingänge dieses Multiplexers auf  $I_0(000)$ ,  $I_1(001)$ ,  $I_2(010)$  und  $I_3(011)$  gesetzt. Dabei bedeutet z.B.  $I_2(010)$ , dass am Eingang  $I_2$  der Wert 010 anliegt. Implementieren Sie diesen 4:1 Multiplexer gemäß der Wahrheitstabelle 1, sodass nur genau eines der Signale  $I_0$ ,  $I_1$ ,  $I_2$  und  $I_3$  gleichzeitig ausgewählt werden kann. Dies soll anhand der Auswahlssignale  $S_1S_0$  (zwei Bits) geschehen.

Auswahlsignal		Ausgabe
$S_1$	$S_0$	Y
0	0	$I_0(000)$
0	1	$I_1(001)$
1	0	$I_2(010)$
1	1	$I_3(011)$

Tabelle 1: Die Wahrheitstabelle eines 4:1 Mux.

Auswahlsignal			Ausgabe
$S_2$	$S_1$	$S_0$	Y
0	0	0	$I_0(000)$
0	0	1	$I_1(001)$
0	1	0	$I_2(010)$
0	1	1	$I_3(011)$
1	0	0	$I_4(100)$
1	0	1	$I_5(101)$
1	1	0	$I_6(110)$
1	1	1	$I_7(111)$

Tabelle 2: Die Wahrheitstabelle eines 8:1 Mux.

- a. (3 Punkte) Verwenden Sie die vorgegebenen logischen Gatter (AND, OR, NOT), um diesen 4:1 Multiplexer zu entwerfen. Diese Gatter unterscheiden sich von den auf Blatt 02 erstellten Gattern in der Anzahl der Eingänge für das AND-Gatter (3 Eingänge) und das OR-Gatter (4 Eingänge). Zeichnen Sie zuerst das **Schaltbild** des 4:1 Multiplexers und implementieren Sie danach den 4:1 Multiplexer in VHDL.  
Hinweis: Der 4:1 Mux hat vier 3-Bit Vektoren als Eingabe. Für das Schaltbild genügt es diese, wie in dem Schema in Abbildung 1, als einen Eingang darzustellen. Für die Implementierung werden dann Vektoren verwendet. Ein einfaches Beispiel in dem mehrere Gatter verwendet werden um einen 2-Bit Eingangsvektor zu negieren finden Sie im Verzeichnis 4.0\_example. Weitere Details finden Sie in Kapitel 4 des Bereitgestellten VHDL Skripts.
- b. (1 Punkte) Vervollständigen Sie die vorgegebene Testbench, in der die Fälle  $S_1S_0$  gleich 00, 10, 01 und 11 überprüft werden.

In den weiteren Aufgabenteilen geht es um einen 8:1 Multiplexer (Abbildung: 1(b)), mit 3-bit Eingangssignalen  $I_0(000)$ ,  $I_1(001)$ ,  $I_2(010)$ ,  $I_3(011)$ ,  $I_4(100)$ ,  $I_5(101)$ ,  $I_6(110)$  und  $I_7(111)$ . Implementieren Sie diesen 8:1 Multiplexer anhand der Wahrheitstabelle 2, sodass genau eines der 3-Bit Signale von  $I_0$  bis  $I_7$  ausgewählt werden kann. Dies soll anhand der Auswahlsignale  $S_2S_1S_0$  (drei Bits) geschehen.

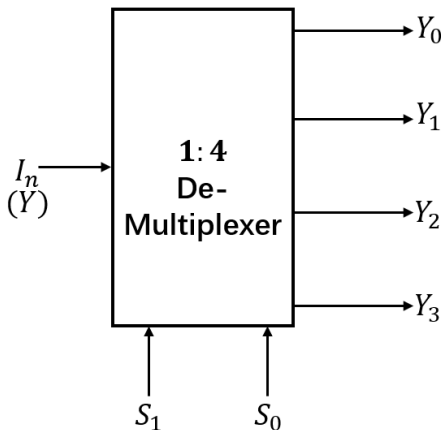
- c. (3 Punkte) Verwenden Sie zwei der 4:1 Multiplexer, die im vorherigen Aufgabenteil entworfen wurden, um diesen 8:1 Multiplexers zu entwerfen. Dabei können die bereitgestellten Gatter genutzt bzw. verändert werden und/oder die auf Übungsblatt 2 entworfenen Gatter verwendet werden. Zeichnen Sie zuerst das **Schaltbild** des 8:1 Multiplexers und implementieren Sie danach den 8:1 Multiplexer in VHDL. Dabei darf der 4:1 Multiplexer im Schaltbild als Komponente verwendet werden.
- d. (1 Punkte) Vervollständigen Sie die vorgegebene Testbench, in welcher die Fälle  $S_2S_1S_0$  gleich 000, 001, 010, 011, 100, 101, 110 und 111 überprüft werden sollen.

## 4.2 Demultiplexer in VHDL (4 Punkte)

Der Demultiplexer erfüllt die umgekehrte Aufgabe des Multiplexers. Während der Multiplexer dazu benutzt wird, eine Ausgabe aus vielen Eingaben auszuwählen, leitet der Demultiplexer ein einziges Eingangssignal auf eine von mehreren Ausgaben weiter. Beispielsweise repräsentiert Abbildung 2 einen 1:4 Demultiplexer, dessen Eingangssignal  $I_n$  eigentlich das Ausgangssignal Y des 4:1 Multiplexers aus Abbildung 1(a) ist. Die Aufgabe dieses 1:4 Demultiplexers ist es, die Eingabe  $I_n$  auf den korrekten Ausgang ( $Y_0$ ,  $Y_1$ ,  $Y_2$  oder  $Y_3$ ), abhängig von dem Auswahlsignal  $S_1S_0$ , weiterzuleiten.

### Aufgaben:

Nehmen Sie an, dass das Eingangssignal  $I_n$  des 1:4 Demultiplexers in Abbildung 2 ein 3-Bit Signal ist. Implementieren Sie einen 1:4 Demultiplexer anhand der Wahrheitstabelle 3, sodass  $I_n$ , aufgrund der Auswahlsignale  $S_1$  und  $S_0$ , auf den richtigen Ausgang weitergeleitet wird.



Eingabe	Auswahlsignal		Ausgabe			
	$S_1$	$S_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
$I_n$	0	0	$I_n$	0	0	0
$I_n$	0	1	0	$I_n$	0	0
$I_n$	1	0	0	0	$I_n$	0
$I_n$	1	1	0	0	0	$I_n$

Abbildung 2: Schema eines 1:4 Demultiplexer.

Tabelle 3: Die Wahrheitstabelle eines 1:4 Demux.

- (3 Punkte) Verwenden Sie logische Gatter (wahlweise hier bereit gestellte oder z.B. auf Blatt 2 selbst erstellte), um diesen 1:4 Demultiplexer zu entwerfen. Diese Gatter können wenn nötig angepasst werden (z.B. kann die Anzahl der Inputs verändert werden). Zeichnen Sie erst das **Schaltbild** des 1:4 Demultiplexers, den Sie entwerfen wollen und implementieren Sie den 1:4 Demultiplexer danach in VHDL.
- (1 Punkte) Testen Sie Ihre Implementierung in einer Testbench mit  $S_1S_0$  jeweils gleich 00, 10, 01 und 11.

### 4.3 Kodierer und Dekodierer in VHDL (4 Punkte)

Ein andere häufig genutzte kombinatorischen Schaltung ist der *Kodierer*. Während ein Multiplexer nur eines der Eingabesignale als Ausgangssignal auswählt, verarbeitet ein Kodierer - oft auch binärer Kodierer genannt - all seine Eingänge einzeln und wandelt sie in ein entsprechendes kodierte Ausgangssignal um. Abbildung 3 zeigt das Schaltbild einen 4:2 Kodierer, der ein 4-Bit Eingabesignal  $I_0I_1I_2I_3$  in ein kodierte 2-Bit Signal  $Y_1Y_0$  umwandelt wie in Wahrheitstabelle 4 dargestellt.

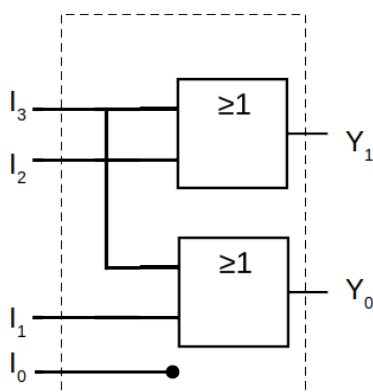


Abbildung 3: Schaltbild eines 4:2 Kodierer.

Eingabesignale				Ausgabesignale	
$I_3$	$I_2$	$I_1$	$I_0$	$Y_1$	$Y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Tabelle 4: Die Wahrheitstabelle eines 4:2 Kodierers.

Ein Problem eines solchen einfachen Binärkodierers ist, dass die Ausgabe mit hoher Wahrscheinlichkeit falsch ist, falls mehrere Eingangssignale den logischen Wert "1" besitzen. Wenn zum Beispiel  $I_2$  und  $I_1$  gleichzeitig den logischen Wert "1" haben, so generiert der Binärkodierer die Ausgabe "11", statt "01" oder "10".

Dieses Problem kann man vermeiden wenn verschiedenen Eingängen Prioritäten zugewiesen werden. Sobald mehrere Eingabesignale den logischen Wert "1" besitzen, bestimmt das System den Eingang mit der höchsten Priorität und gibt die entsprechende kodierte Ausgabe aus. Wenn an  $I_2$  eine logische "1" anliegt, so werden die Ausgabesignale  $Y_1Y_0$  nur

durch  $I_2$  bestimmt, unabhängig von den welche logischen Werten die an  $I_1$  und  $I_0$  anliegen. Diese Arbeitsweise ist in Tabelle 5 dargestellt.

Eingabesignale				Ausgabesignale	
$I_3$	$I_2$	$I_1$	$I_0$	$Y_1$	$Y_0$
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Tabelle 5: Die Wahrheitstabelle eines Prioritätskodierers.

### Aufgaben:

- (1 Punkte) Entwerfen Sie anhand der Wahrheitstabelle 5 einen 4:2 Prioritätskodierer indem Sie das Schaltbild zeichnen.
- (2 Punkte) Implementieren Sie den 4:2 Prioritätskodierer in VHDL und testen Sie Ihre Implementierung in einer Testbench mit  $I_3I_2I_1I_0$  jeweils gleich 0001, 0010, 0100 und 1000.
- (1 Punkte) Dekodierer, die das kodierte Signal zurück in seine originale Form übersetzen, implementieren die umgekehrte Funktion des Kodierers. Entwerfen Sie einen 2:4 Dekodierer nach der Wahrheitstabelle 6 indem Sie das Schaltbild zeichnen.

Eingabesignale		Ausgabesignale			
$I_1$	$I_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Tabelle 6: Die Wahrheitstabelle eines 2:4 Dekodierers.