Aufgabe 2: Motorsteuerung

G. Kemnitz, C. Giesemann, TU Clausthal, Institut für Informatik

13. Dezember 2017

Zusammenfassung

Schrittweiser Entwurf einer Schaltung für die Drehzahlsteuerung eines Gleichstrommotors über PWM und die Messung des Drehwinkels und der Winkelgeschwindigkeit.

1 Drehzahlsteuerung und Winkelmessung

Die Drehzahlsteuerung erfolgt über Pulsweitenmodulation (PWM). Gesteuert über zwei Signale (»En« und »Dir«) werden drei Schaltzustände unterschieden:

- Anlegen der Motorspannung in Vorwärtsrichtung,
- Kurzschluss des Motors und
- Anlegen der Motorspannung in Rückwärtsrichtung.

Die Drehzahl wird über die Breite der »En«-Pulse eingestellt (Abb. 1).

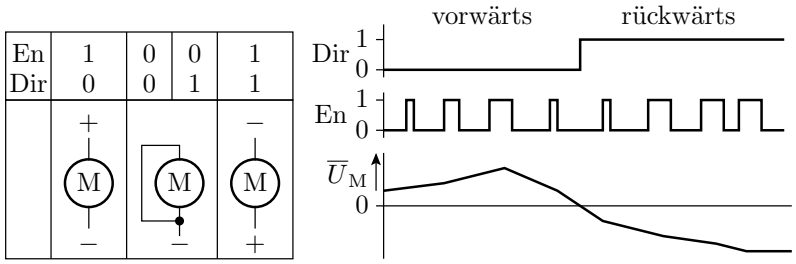
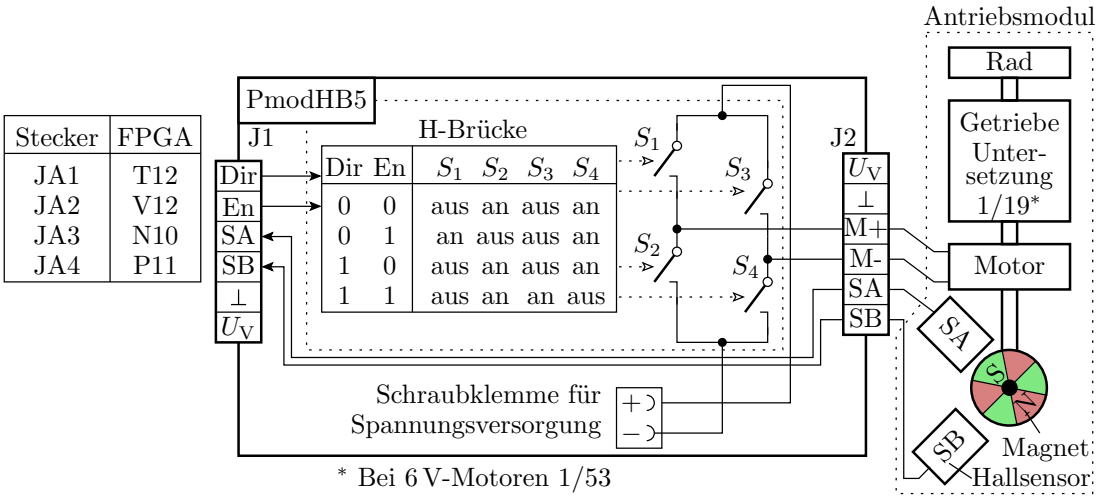


Abbildung 1: PWM-Steuerung eines Gleichstrommotors

Im Praktikum wird an die FPGA-Baugruppe an Stecker JA oben ein Gleichstom-Getriebemotor über ein H-Brückenmodul PmodHB5 angeschlossen. Das vom FPGA zu generierende Richtungssignal »Dir« legt die Drehrichtung fest und die relative Pulsbreite des gleichfalls vom FPGA zu generierenden Enable-Signals »En« die Drehgeschwindigkeit (Abb. 2).

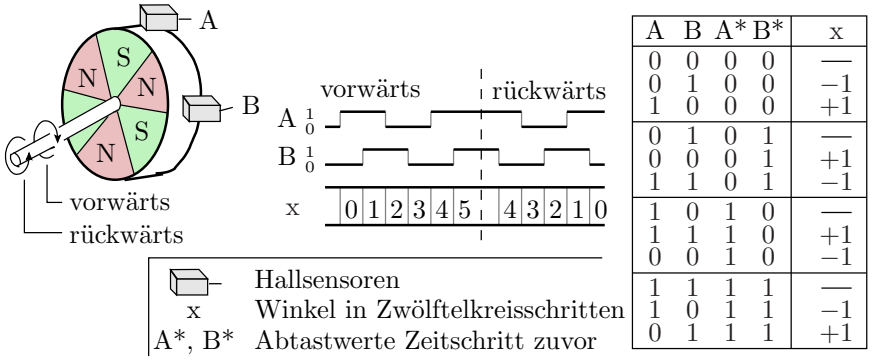
An der Achse des Motors ist ein runder Magnet befestigt, der zwei winkelversetzte Hallsensoren bei Rotation dreimal pro Umdrehung um ca. eine Viertelperiode phasenverschoben ein- und ausschaltet. Bei Vorwärtsrotation wechselt zuerst das Sensorsignal »SA« und dann das Sensorsignal »SB« und bei umgekehrter Drehrichtung zuerst »SB« und dann »SA«. Wie die Tabelle in Abb. 3 rechts zeigt, lässt sich aus den einmal und zweimal abgetasteten Sensorwerten für jeden Abtastschritt bestimmen, ob der Magnet ein Schritt vor- oder rückwärts rotiert ist, vorausgesetzt, dass zwischen zwei Änderungen mindestens einmal abgetastet wird. Zur Positionsbestimmung sind die



Stecker	FPGA
JA1	T12
JA2	V12
JA3	N10
JA4	P11

Abbildung 2: Versuchsaufbau für das Praktikum

postiven und negativen Increment-Schritte zu zählen. Eine Raddrehung entspricht 53 Motorumdrehungen und eine Motorumdrehung unterteilt sich $2 \cdot 6$ Increment-Schritte[1]. Der zurückgelegte Drehwinkel in Radumdrehungen ist die Zahl der Increment-Schritte geteilt durch $2 \cdot 6 \cdot 53$. Die Drehzahl sei im Weiteren die Anzahl der Radumdrehungen pro Minute.



A	B	A*	B*	x
0	0	0	0	—
0	1	0	0	−1
1	0	0	0	+1
0	1	0	1	—
0	0	0	1	+1
1	1	0	1	−1
1	0	1	0	—
1	1	1	0	+1
0	0	1	0	−1
1	1	1	1	—
1	0	1	1	−1
0	1	1	1	+1

Abbildung 3: Prinzip der Drehwinkelmessung

2 Simulationsmodell für das Senorverhalten

Das nachfolgende Simulationsmodell hat als Eingabe die Drehzahl des Motors pro Minute als ganze vorzeichenbehaftete Zahl und eine Boolean-Variable »run«, die auf »False« gesetzt die Simulation beendet. Ausgabe sind die Sensorsignale, wie sie der Motor bei der vorgegebenen Drehzahl erzeugt. Die Wartezeit zwischen zwei Sensorsignaländerungen ist

$$t_w = \frac{1\,\text{min}}{53 \cdot 2 \cdot 6 \cdot |u|} \tag{1}$$

[1] Wenn bei dem einen Sensor das Magnetfeld gerade wechselt befindet sich der andere in der Mitte zwischen zwei Magnetisierungsrichtungsänderungen.

(539 – Motorumdrehungen je Radumdrehung; $2 \cdot 6$ – Sensorschaltflanken je Motorumdrehung; $|u|$– Betrag der Anzahl der Radumdrehungen pro Minute). Der Algorithmus, nach dem die Sensorsignale weiterschalten, entspricht dem eines 2-Bit-Johnson-Zählers (vergl. EDS, Foliensatz 4, Abschn. 2.3). Bei Vorwärtsdrehung ist die Schieberichtung von »SA« nach »SB« und bei Rückwärtsdrehung von »SB« nach »SA«.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;

3  entity quadenc is
4   port(
5    um: in  integer range -128 to 127;      -- Drehzahl in Umdrehungen/min
6    run: in boolean;                        -- Simulation laeuft
7    SA, SB: buffer std_logic:= '0');
8   end entity;

9  architecture a of quadenc is
10 begin
11  process
12   variable tw: delay_length;               -- Wartezeit zur naechsten
13  begin                                     --  Sensorflanke
14   if    um>0 then SB <= SA; SA <= not SB;  -- Sensorfolge vorwaerts
15   elsif um<0 then SA <= SB; SB <= not SA;  -- Sensorfolge rueckwaerts
16   end if;
17   if not run then wait;                    -- Simulationsende
18   elsif um=0 then wait until um /= 0;      -- wenn Drehzahl null
19   end if;                                  -- warten, bis ungleich null
20   tw := 60 sec / (abs(um) * 53 * 6 * 2);   -- Wartezeit berechnen
21   wait for tw;                             -- Warten
22  end process;
23 end architecture;
```

Die zugehörige Testbench gibt für eine Folge von Testfällen die Drehzahl und die Dauer bis zur nächsten Drehzahländerung vor. Abb. 4 zeigt die von der Motorbaugruppe zu erwartenden Sensorausgaben. Aus diesen Sensorausgaben sollen später zu entwerfende Schaltungen Winkel und Drehzahl bestimmen.

```
1  library IEEE; use IEEE.STD_LOGIC_1164.ALL;

2  entity tb_quadenc is end entity;

3  architecture a of tb_quadenc is
4   signal um: integer range -128 to 127;   -- Drehzahl in Umdrehungen/min
5   signal SA, SB: std_logic;               -- Sensorsignale
6   signal run: boolean := True;
7  begin

8   qe: entity work.quadenc port map(um=>um, run=>run, SA=>SA, SB=>SB);
9   process begin
10   um <=   7; wait for 400 ms;
11   um <=  15; wait for 400 ms;
12   um <=   0; wait for 200 ms;
13   um <= -12; wait for 400 ms;
14   um <= -15; wait for 300 ms;
15   um <=  11; wait for 400 ms;
16   run <= False; wait;                     -- Simulation beenden
17  end process;

18 end architecture;
```

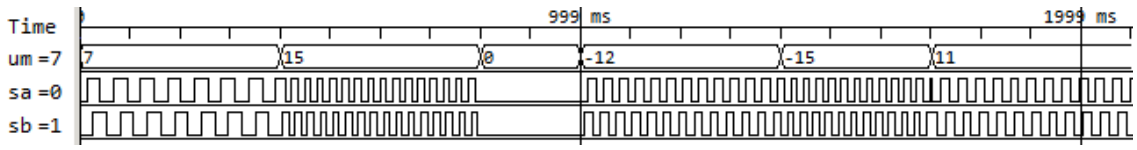


Abbildung 4: Simulationsergebnis

Aufgabe 2.1: Simulationsmodell kontrollieren

Wiederholen Sie die Simulation mit anderen Drehzahlen im Bereich von -128 bis 127. Kontrollieren Sie, das bei positiver Drehzahl stets der Wert an »SA« und bei negativen Geschwindigkeiten der Wert an »SB« zuerst wechselt. Wie simuliert das Modell den Sonderfall »Drehzahl null«?

3 Motor- und Steuerkennlinie

Der Zusammenhang zwischen der relative Pulsbreite und der Rotationsgeschwindigkeit ist nicht linear und hängt von zahlreichen Einflussfaktoren ab: (Mototyp, Motorlast, absolute Pulsbreite, ...). Abb. 5 zeigt die Motorkennlinie für einen Beispielmotor des verwendeten Typs im Leerlauf für die absoluten Pulsbreiten 2 ms (rot), 1 ms (grün) und 0,5 ms (blau).

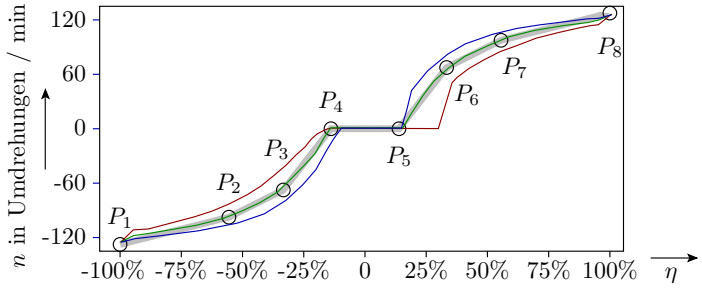


Abbildung 5: Motorkennlinie

Zur Linearisierung soll die Kennlinie für die absolute Pulsbreite 1 ms stückweise linear durch den dicken grauen Linienzug mit folgenen Punkten angenähert werden:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
η (relative Pulsbreite)	-100%	-55,6%	-33,3%	-13,9%	13,9%	33,3%	55,6%	100%
n (Umdrehungen / min)	-127,5	-97,5	-67,5	0	0	67,5	97,5	127,5

Daraus ergibt sich für den Betrag der Drehzahl abschätzungsweise:

$$|n|(\eta) = \begin{cases} 0 & |\eta| < 55{,}6\% \\ \frac{67{,}5}{33{,}3\%-13{,}9\%} \cdot (\eta - 13{,}9\%) & 13{,}9\% \le |\eta| < 33{,}3\% \\ \frac{97{,}5-67{,}5}{55{,}6\%-33{,}3\%} \cdot (\eta - 33{,}3\%) + 67{,}5 & 33{,}3\% \le |\eta| < 55{,}6\% \\ \frac{127{,}5-97{,}5}{100\%-55{,}6\%} \cdot (\eta - 55{,}6\%) + 97{,}5 & 55{,}6\% \le |\eta| \end{cases} \tag{2}$$

Aufgabe 2.2: Linearisierung der Steuerkennlinie

Bestimmen Sie die Steuerkennlinie $\eta(x)$ zur Umrechnung einer Byte-Eingabe x für eine vorzeichenbehaftete ganze Zahl in eine relative Pulsbreite η in Prozent so, dass beim Einsetzen von Gl. 2 die Steuergröße $x \in (-128, 127)$ linear auf eine Umdrehungsgeschwindigkeit von -128 bis 127 Umdrehungen pro min abgebildet wird. Ergänzen Sie zur Kontrolle in der nachfolgenden Tabelle die relativen Pulsbreiten η, die ihre Steuerkennline berechnet und die sich aus der berechneten Pulsbreite nach Gl. 2 ergebende Drehzahl $n(\eta)$.

x	0x80 (-128)	0xA0 (-96)	0xC0 (-64)	-0x4F (-32)	0	0x20 (32)	0x40 (64)	0x60 (96)	0x7F
n_{soll}	-128	-96	-64	-32	0	32	64	96	127
η									
$n(\eta)$									

4 Aufbau und Motortest

Stecken Sie wie in Abb. 6

- das H-Brückenmodul PmodHB5 an den Stecker JA oben,
- den Motor an die H-Brücke

schließen Sie die Versorgungsleitungen an

- schwarz (braun): GND-Schraubklemme PmodHB5 an GND-Stift auf der FPGA-Baugruppe,
- rot: VM-Schraubklemme PmodHB5 an »5VD«-Stift auf der FPGA-Baugruppe.

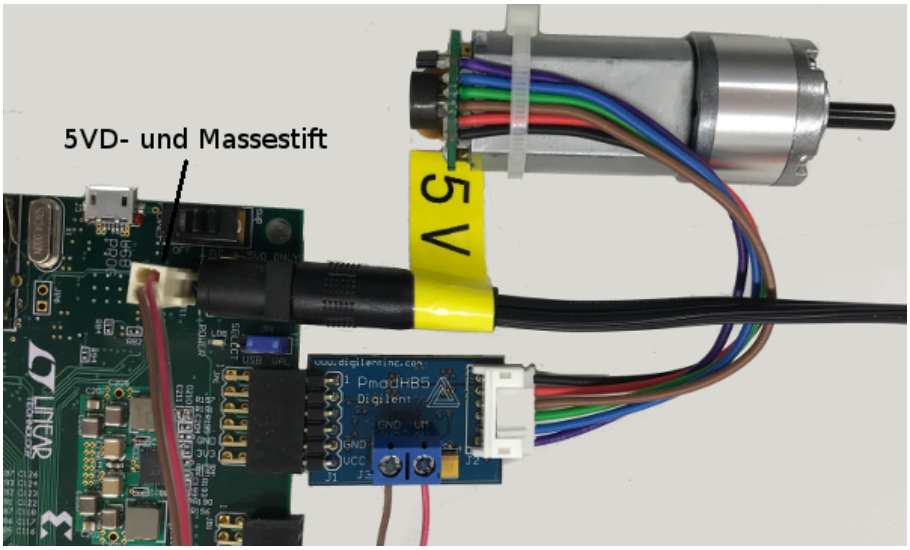


Abbildung 6: Foto des Versuchsaufbaus

Entwerfen Sie in VHDL eine Schaltung mit der Schnittstelle

```
entity pwm_ctrl is
 port( T: in std_logic;          -- 100 MHz Baugruppentakt
    I: in std_logic;             -- Init. (Taster)
    sw: in signed(7 downto 0); -- Schalter auf der Baugruppe
    EN, Dir: out std_logic);    -- Steuersignale für die H-Brücke
end entity;
```

die bei einem negativen Eingabewert »sw« die Drehrichtung »Dir« gleich '1' und sonst gleich '0' setzt und am Ausgang »En« ein PWM-Signal mit einer Periode von 1 ms und einer relativen Pulsbreite

$$\eta = \frac{|sw|}{128}$$

($|sw|$ – Betrag der Schaltereingabe) erzeugt. In der vorgegebene UCF-Datei »pwm_ctrl.ucf« sind die für die Teilaufgaben in diesem Abschnitt noch nicht benötigten Anschlüsse auskommentiert.

Aufgabe 2.3: Simulation der Motorsteuerung

Simulieren Sie die Schaltung für jeweils 2 ms (2 Pulse) mit den Schaltereingaben der nachfolgenden Tabelle und ergänzen Sie die Soll- und Ist-Breiten der Pulse:

Schaltereingabe	0x40 (+50%)	0x28 (+31,25%	0x90 (-87,5%)	0xC0 (-50%)	0xE0 (-25%)
Soll-Pulsbreit in µs					
ist-Pulsbreit in µs					

Aufgabe 2.4: Test der Motorsteuerung

Implementieren Sie die Schaltung und kontrollieren Sie, dass sich der Motor erst ab Schaltereingabe größer 14% (0x12) vorwärts und Schaltereingabe kleiner −14% (0xEE) rückwärts dreht.

Aufgabe 2.5: Linearisierung der Steuerkennlinie

Ergänzen Sie die Linearisierungsfunktion aus Aufgabe 2.2 und wiederholen Sie Simulation und Test.

5 Winkelmessung

Der Schaltungsaufbau aus Abb. 6 wird um ein PmodLED an Port JD ergänzt. Die Gesamtschaltung erhält zusätzlich eine 16-Bit LED-Ausgabe vom Typ »signed«. Für die Winkelmessung sind die beiden Sensorsignale etwa einmal je ms abzutasen. Aus den einfach und zweifach abgetasteten Sensorwerten ist entsprechen der Tabelle in Abb. 3 rechts zu bestimmen, ob sich der Magnet einen Schritt vor- oder rückwärts gedreht hat und entsprechend der 8-Bit-Zählwert, der auf die LEDs ausgegeben wird, hoch oder runter zu zählen.

Aufgabe 2.6: Simulation der Winkelmessung

Schreiben Sie die Winkelmessschaltung als eigenständige Entwurfseinheit mit einem 1 kHz-Takt und den Sensorsignalen als Eingänge sowie dem 16 Bit-Zählwert als Ausgabe. Simulieren Sie diese in dem vorgegebenen Testrahmen. Kontrollieren Sie, dass der Zähler bei jeder Sensorsignalflanke einer Schritt hoch bzw. runter zählt.

Aufgabe 2.7: Test der Winkelmessung

Binden Sie die Winkelmessschaltung aus der Aufgabe zuvor in die Motorsteuerschaltung aus Aufgabe 2.5 mit der linearisierten Steuerkennlinie ein und kontrollieren Sie, dass sich bei einer eingestellten Drehzahl von 60 Umdrehungen pro Minute der Zählwert um ca. $53 \cdot 12$ Zählschritte pro Sekunde zu- bzw. abnimmt, erkennbar daran, dass das Ausgabebit 7 etwa mit einer Frequenz von zwei Herz blinkt.

6 Drehzahlmessung

Für die Drehzahlmessung soll in der bisherige Entwurfseinheit die Winkelmessschaltung durch eine Drehzahlmesschaltung ersetzt werden. Die Teilschaltung für die Drehzahlmessung bekommt auch ein Takteingang, einen Initialisierungseingang, die Sensorsignale als Eingänge und die LEDs als Ausgänge. Der Messalgorithmus für die Geschwindigkeitsmessung ergibt sich durch Umstellung von Gl. 1 für die Wartezeitberechnung im Simulationsmodell »quadenc.vhd« nach dem Betrag der Anzahl der Radumdrehungen pro min:

$$|u| = n \cdot \frac{1\,\mathrm{min}}{53 \cdot 2 \cdot 6 \cdot t_\mathrm{w}}$$

($|u|$ –Betrag der Drehzahl in Radumdrehungen pro min; t_w – Wartezeit zwischen den Zählflanken; n – Anzahl der Zählflanken, auf die gewartet wird). Messen lassen sich die Wartezeit zwischen n Zählflanken, die Anzahl der Zählflanken innerhalb einer Wartezeit t_w oder beides. Eine einfache Lösung ist die Flankenzählung in einem Zeitfenster:

$$t_\mathrm{w} = 4 \cdot \frac{1\,\mathrm{min}}{53 \cdot 12} \approx 377\,\mathrm{ms}$$

Dann ist der Betrag der Drehzahl gleich ein viertel des Zählwertes:

$$|u| = \frac{n}{4}$$

Der Wertebereich der Drehzahl ist ca. -128 bis 127 Umdrehungen pro Sekunde. Im Experiment sollen die On-Bord-Leuchtdioden LD0 bis LD6 den Betrag und LD7 das Vorzeichen der Drehzahl anzeigen.

Aufgabe 2.8: Simulation der Drehzahlmessung

Schreiben Sie die Schaltung für die Drehzahlmessung als eigenständige VHDL-Entwurfseinheit mit einem 1 kHz-Takt und den Sensorsignalen als Eingabe sowie dem 7 Bit-Betrag und dem Vorzeichenbit als Ausgabe. Simulieren Sie die Schaltung in dem vorgegebenen Testrahmen und kontrollieren Sie, dass die vorgegebene Drehzahl mit der gemessenen übereinstimmt. Hinweis: Die Vorgabedrehzahl darf sich während der 377 ms dauernden Messungen nicht ändern.

Aufgabe 2.9: Test der Drehzahlmessung

Binden Sie die Drehzahlmessschaltung aus der Aufgabe zuvor in die Motorsteuerschaltung mit der linearisierten Steuerkennlinie aus Aufgabe 2.5 ein und kontrollieren Sie, dass die angezeigten Drehzahlen etwa den eingestellten Drehzahlen entsprechen. Tragen Sie hierzu in der nachfolgenden Tabelle für die vorgegebenen Schalterwerte die Soll-Drehzahlen nach Aufgabe 2.2 und die gemessenen Drehzahlen ein.

x	0x80 (-128)	0xA0 (-96)	0xC0 (-64)	-0x4F (-32)	0	0x20 (32)	0x40 (64)	0x60 (96)	0x7F (127)
n_soll									
$n\,(\eta)$									

Aufgabe 2.10: Zusatzaufgabe

Erweitern Sie die Schaltung so, dass umschaltbar über eine zusätzliche Eingabetaste wahlweise die Position in Zählschritten oder die Drehzahl in Umdrehungen pro Minute als Hexadezimalzahl im Zweierkompliment dargestellt wird. Alternativ könnte die Darstellung auch dezimal erfolgen mit einer der LEDs auf dem Board zur Visualisierung des Vorzeichens.

7 Abnahmekriterien

- Aufgabe 2.1: Geänderte VHDL-Testbench zur Kontrolle und ghw- sowie sav-Datei zur Visualisierung des Simulationsergebnisses.
- Aufgabe 2.2: Steuerkennlinie $\eta(x)$ als stückweise lineare Funktion und die ausgefüllte Tabelle.
- Aufgabe 2.3: VHDL-Datei zur Kontrolle der Beschreibung, ghw- plus sav-Datei zur Visualisierung des Simulationsergebnisses und die ausgefüllte Tabelle.
- Aufgabe 2.4: VHDL-Datei zur Kontrolle der Beschreibung und Bitdatei zur Programmierung der funktionierenden Schaltung.
- Aufgabe 2.5 VHDL-Datei zur Kontrolle der Beschreibung und Bitdatei zur Programmierung der funktionierenden Schaltung.
- Aufgabe 2.6: VHDL-Datei zur Kontrolle der Beschreibung und ghw- plus sav-Datei zur Visualisierung des Simulationsergebnisses.
- Aufgabe 2.7: VHDL-Datei zur Kontrolle der Beschreibung und Bitdatei zur Programmierung der funktionierenden Schaltung.
- Aufgabe 2.8: VHDL-Datei zur Kontrolle der Beschreibung und ghw- plus sav-Datei zur Visualisierung des Simulationsergebnisses.
- Aufgabe 2.9: VHDL-Datei zur Kontrolle der Beschreibung und Bitdatei zur Programmierung der funktionierenden Schaltung.
- Aufgabe 2.10: VHDL-Datei zur Kontrolle der Beschreibung und Bitdatei zur Programmierung der funktionierenden Schaltung.