



BAKKALAUREATSARBEIT

# **Messen, Steuern und Regeln mit FPGA**

FACHHOCHSCHULE TECHNIKUM WIEN  
STUDIENGANG ELEKTRONIK

Harald Sams  
Mayr-Melnhofstraße 4  
8700 Leoben  
Matr.Nr. TW194

FH-Prof. Dipl.-Ing. Dr. Martin Horauer

Wien, 23.01.2006

---

## EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich erteile hiermit der Fachhochschule Technikum Wien die wissenschaftlichen Nutzungsrechte an der vorliegenden Arbeit.

Wien, 23.01.2006

---

# Inhaltsverzeichnis

<b>1</b>	<b>Blockschaltbild des gesamten Systems</b>	<b>1</b>
<b>2</b>	<b>Der Inkrementalgeber (Encoder)</b>	<b>2</b>
2.1	<i>Bestimmen der Drehrichtung</i>	2
2.2	<i>Bestimmen der Drehzahl</i>	2
<b>3</b>	<b>Erstellung des Frequenzspektrums an die Drehzahl</b>	<b>3</b>
3.1	<i>Berechnung der minimalen Encoder – Frequenz</i>	3
3.2	<i>Messung der maximalen Encoder – Frequenz</i>	4
3.3	<i>Darstellung des Frequenzspektrums</i>	4
<b>4</b>	<b>Motortreiber (H-Brücke)</b>	<b>4</b>
<b>5</b>	<b>Pulsweiten – Modulation</b>	<b>5</b>
<b>6</b>	<b>Der Regelkreis</b>	<b>5</b>
6.1	<i>Wirkungsweise des Regelkreises</i>	6
6.2	<i>Blockschaltbild des Regelkreises</i>	6
6.3	<i>Hauptteile des Regelkreises</i>	6
6.4	<i>Die Regelstrecke</i>	6
<b>7</b>	<b>Der Regler</b>	<b>8</b>
7.1	<i>Wahl des Reglers</i>	8
7.2	<i>Der PI-Regler</i>	8
7.3	<i>Anpassung des Reglers an die Regelstrecke</i>	8
<b>8</b>	<b>Die Architektur</b>	<b>9</b>
8.1	<i>Beschreibung der gesamten Architektur</i>	9
8.2	<i>Blockschaltbild der gesamten Architektur</i>	9
8.3	<i>Modul: Encoder (Messeinrichtung)</i>	10
8.4	<i>Modul: Regler</i>	13
8.5	<i>Modul: PWM</i>	14
<b>9</b>	<b>Testen der Module</b>	<b>15</b>
9.1	<i>Testen des Encoder</i>	15
9.2	<i>Testen des Reglers</i>	17
9.3	<i>PWM</i>	18
<b>10</b>	<b>Literatur</b>	<b>19</b>

## **Aufgabenstellung**

Durch Aufteilung des Projektes in Teilaufgaben ist folgende Aufgabenstellung entstanden:

Es ist ein System zu erstellen das die Geschwindigkeit eines Gleichstrommotors (freie Wahl) misst, eine passende Regelung für dieses Objekt, in die auch eine Geschwindigkeitsvorgabe einfließt und eine Motoransteuerung mittels Pulsweitenmodulation (PWM). Ziel 1 des Projektes ist, das dieses Pendel ohne Fremdeinwirkung aufrecht stehen bleibt. Unser zweites Ziel ist es das sich das Pendel über eine Fernsteuerung nach vor und zurück bewegen lässt ohne umzukippen.

## **Kurzzusammenfassung**

Das komplette Schaltungsdesign wird mit der HW Beschreibungssprache VHDL realisiert und anschließend wird damit ein FPGA SPARTAN II von Xilinx konfiguriert. Um einen Regelkreis für das Inverse Pendel erstellen zu können muss zuerst eine Messeinrichtung für das Feststellen der Motorgeschwindigkeit und Drehrichtung entwickelt werden. Damit diese Messeinrichtung die Drehzahl des Motors messen kann, wird auf den DC-Motor ein Inkrementalgeber montiert. Diese Messeinrichtung liefert immer die aktuelle Drehzahl für das Regelsystem. Zunächst wird ein Ansteuerungsmodul für den Motortreiber entwickelt, das den berechneten Stellwert des Reglers in Form einer Pulsweitenmodulation ausgibt, um eine Drehzahländerung erzwingen zu können. Durch stetiges Messen der Motordrehzahl, ausregeln an die Soll-Drehzahl, und Ansteuern der Motortreiber kann ein funktionsfähiges Regelsystem realisiert werden.

## **Verwendete Werkzeuge**

- Xilinx ModelSim XEIII 6.0a
- Xilinx ISE 7.1i
- MathSoft MathCad 11

## **1 Blockschaltbild des gesamten Systems**

Dieses Blockschaltbild zeigt ein Mess-Regel-Steuerungssystem für DC-Motoren die über Motortreiber (H-Brücken) betrieben werden und über eine Messeinrichtung (Inkrementalgeber, Encoder) verfügen.

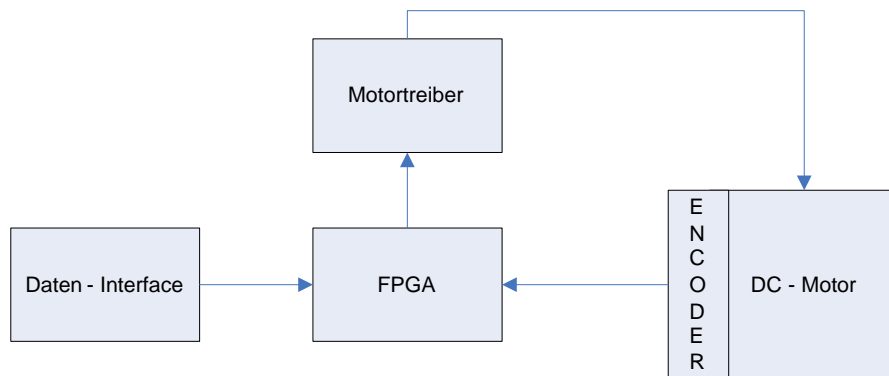


Abbildung 1: Blockschaltbild des gesamten Systems

## 2 Der Inkrementalgeber (Encoder)

Der Encoder IE2-64 von der Firma Faulhaber ist auf den dazugehörigen DC-Motor 2642W012CR, ebenfalls von der Firma Faulhaber, montiert. Dieser Encoder ist ein magnetischer Inkrementalgeber und liefert 64 Impulse pro Umdrehung. Dieser Encoder besitzt ein Interface von 4 Leitungen. Zwei Leitungen davon sind zur Spannungsversorgung des Encoders (+5V) und die beiden übrigen Leitungen sind digitale Outputs (Kanal A, B) über die die Encoder-Signale (A, B) ausgegeben werden. Weitere Details über den Inkrementalgeber finden sie im Datenblatt [4].

### 2.1 Bestimmen der Drehrichtung

Die beiden Encoder-Impulse (A, B) sind zu sich um  $90^\circ$  Phasenverschoben, siehe Abb. 2. Durch diese Phasenverschiebung kann festgestellt werden in welche Richtung sich der Motor dreht. Um eine Drehrichtung feststellen zu können muss der Zustand des Signals B, im Zeitpunkt einer fallenden Flanke des Signals A, abgefragt werden. Ist zu diesem Zeitpunkt das Signal B logisch '0' kann dieser Drehrichtung die Richtung „Links“ zugewiesen werden, andernfalls hat das Signal B den Zustand logisch '1' und definiert somit die Drehrichtung „Rechts“.

### 2.2 Bestimmen der Drehzahl

Um die aktuelle Drehzahl (Umdrehungen pro Sekunde „rps“) bestimmen zu können, muss:

- die Dauer der halben Periodendauer „tp“ des Impulses des Kanals A gemessen werden
- diese Dauer „tp“ muss mit dem Faktor 2 erweitert werden

- um die Frequenz „f“ zu erhalten muss der Kehrwert  $1/(2 \cdot t_p)$  gebildet werden
- $rps = (1/(t_p \cdot 2))/N$

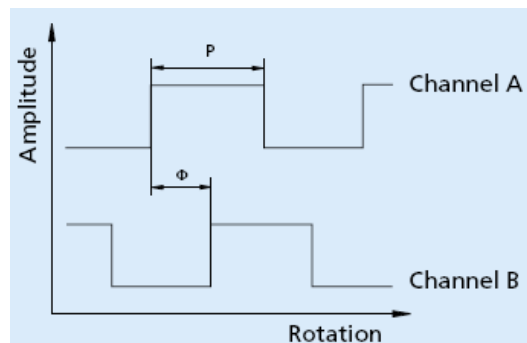


Abbildung 2: Signalverlauf der Encoder - Signale

### 3 Erstellung des Frequenzspektrums an die Drehzahl

Mit diesem Frequenzspektrum wird jeder gemessenen Encoder-Frequenz eine Drehzahl zugewiesen. Für die Erstellung eines Frequenzspektrums, das die Drehzahl des Motors widerspiegelt, muss zuerst die maximale und minimale Encoder-Frequenz gemessen bzw. errechnet werden. Die maximale Frequenz geht aus einer Messung hervor die abhängig vom gewählten Motor und der Encoder-Auflösung ist. Die minimale Frequenz wurde aus der gewünschten erkennbaren Mindest-Drehzahl, die frei wählbar ist, berechnet. Diese maximale und minimale Frequenz definieren die Grenzen der Drehzahl ( $n_{\max}$ ,  $n_{\min}$ ).

#### 3.1 Berechnung der minimalen Encoder – Frequenz

Die minimale Encoder-Frequenz kann selbst gewählt werden. Um möglichst kleine Drehzahlen zu erkennen, muss eine sehr kleine Encoder-Frequenz gewählt werden. In diesem Projekt wurde eine Drehzahl von  $rps = 1$  gewählt. Mit dieser Drehzahl kann nun die Halbperiodendauer bestimmt werden. Da aber der Encoder eine Auflösung von  $N = 64$  besitzt und nur die positive Halbperiode gemessen wird, muss pro Messung der positiven Halbperiode  $t_p = 1/(rps \cdot 64 \cdot 2)$  Sekunden gemessen werden, um eine Umdrehung pro Sekunde zu messen. Aus dieser Formel  $rps = (1/(t \cdot 2))/N$  heraus kann man für jede beliebige Drehzahl ( $rps$ ) die „positive Halbperiodendauer“ des Encoder – Signales berechnen.

$$rpm = (1/(t \cdot 2)) \cdot 60/N$$

$$rps = (1/(t \cdot 2))/N$$

t ... positive halbperiodendauer des Encoder – Signales

- 2 ... um ganze Periodendauer zu erhalten
- N ... Auflösung des Encoders

### 3.2 Messung der maximalen Encoder – Frequenz

An den Motor wurde eine Nennspannung von 14V angelegt um seine Nenndrehzahl zu erreichen. Durch die Nenndrehzahl stellt sich nun die maximale Encoder – Frequenz ein und kann mittels Hilfe eines Oszilloskops am Kanal A oder B gemessen werden. Dieser Messwert entspricht der maximal auftretbaren Frequenz.

### 3.3 Darstellung des Frequenzspektrums

Dieses Bild zeigt einen proportionalen Zusammenhang zwischen Frequenz des Encoder-Signales und der Drehzahl des Motors. Ebenso wird die maximale und minimale messbare Encoder-Frequenz dargestellt. Die Frequenz „fmax“ und „fmin“ bilden die Grenzen für die messbaren Encoder-Frequenzen. Wird eine Frequenz gemessen die höher oder gleich ist als die Maximale auftretbare (fmax), so wird dieser die maximale Drehzahl ( $n_{max}$ ) zugewiesen. Wird eine niedrigere Frequenz als die Minimale Frequenz (fmin) gemessen, so wird dieser die Drehzahl  $rps = 0$  zugewiesen. Allen gemessenen Frequenzen zwischen der maximalen und minimalen Grenze werden Drehzahlen von  $n_{min}$  bis  $n_{max}$  zugewiesen.

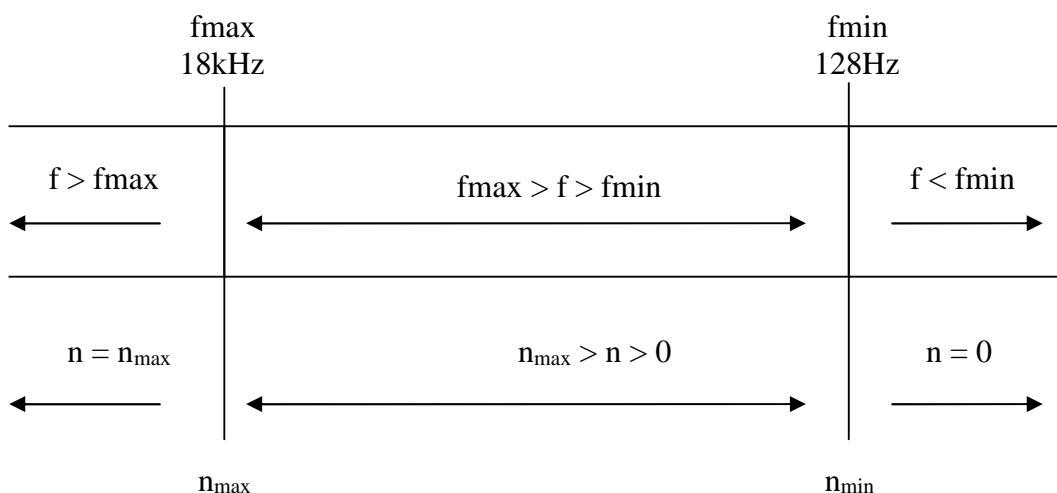


Abbildung 3: Frequenzspektrum

## 4 Motortreiber (H-Brücke)

Ausgewählt und verwendet wurde der Motortreiber (Serie 2632 CR) von der Firma Faulhaber. Dieser Motortreiber hat ein Signal-Interface von drei digitalen Inputs. Zwei Inputs (Direction A und B) werden zur Steuerung der

Ausgangsspannungsrichtung (Versorgungsspannung des DC-Motors) des Motortreibers verwendet. Durch unterschiedliche Kombination der zwei „Direction A und B“ Steuerleitungen, kann man den DC – Motor mit 3 verschiedenen Zuständen ansteuern, siehe Tabelle 1.

Motor - Zustand	Direction A	Direction B
Linkslauf	'1'	'0'
Rechtslauf	'0'	'1'
Fast - Stop	'1'	'1'
Keine Funktion	'0'	“0“

Tabelle 1: Ansteuerungskombinationen

An die dritte Steuerleitung wird ein selbst generiertes PWM - Signal angelegt. Die Ausgangsspannung des Motortreibers (Versorgungsspannung des DC-Motors) verhält sich proportional zum PWM-Signal. Die Frequenz des PWM-Signals darf maximal 20kHz betragen.

## 5 Pulsweiten – Modulation

Bei der Pulsweitenmodulation wird innerhalb einer bestimmten Schaltperiode „Ts“ das Verhältnis zwischen Ein- und Ausschaltzeit verändert. Wird die Einschaltzeit „tein“ vergrößert, so ergibt sich ein größerer arithmetischer Mittelwert der Ausgangsspannung und damit ein größerer Ausgangsstrom.

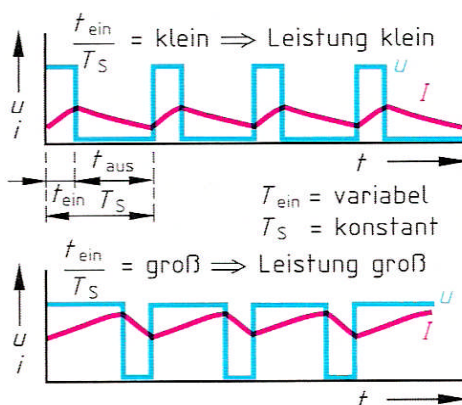


Abbildung 4: Pulsweitenmodulation

## 6 Der Regelkreis

Der Regelkreis dient dazu, eine vorgegebene physikalische Regelgröße „x“ (Drehzahl, Geschwindigkeit) auf einen gewünschten Wert (Sollwert bzw.

Führungsgröße  $w$ ) zu bringen und dort zu halten, unabhängig von auftretenden Störungen wie z.B.: Veränderung der Fahrbahn (Gefälle, Steigung), Veränderung der Reibung oder durch Fremdeinflüsse von Außen wie z.B.: Beschleunigung oder Abbremsung des Motors. Für weitere Informationen über die Wirkungsweise des Regelkreises siehe [1].

## 6.1 Wirkungsweise des Regelkreises

Dieser Regelkreis führt die folgenden Teilaufgaben fortlaufend aus:

- MESSEN
- VERGLEICHEN
- BERECHNEN
- STELLEN

## 6.2 Blockschaltbild des Regelkreises

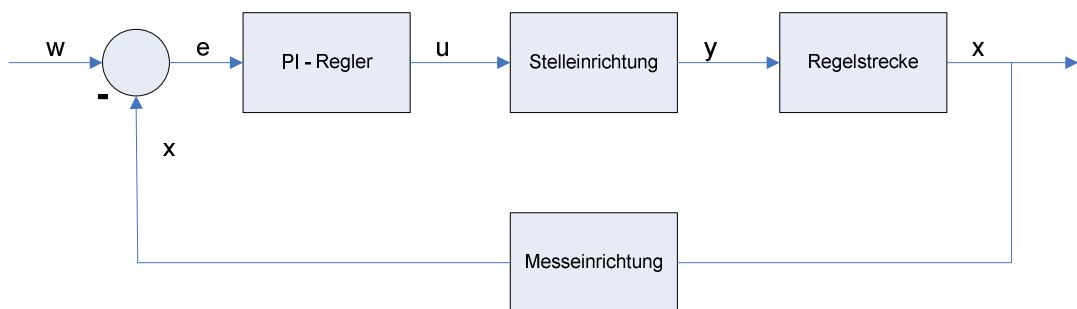


Abbildung 5: Blockschaltbild des Regelkreises

## 6.3 Hauptteile des Regelkreises

Das Blockschaltbild Abb. 5 zeigt den Aufbau des Regelsystems, dass aus 3 Hauptteilen Regler, Regelstrecke, Messeinrichtung besteht.

## 6.4 Die Regelstrecke

### Beurteilung der Regelstrecke

Regelstrecken werden nach ihrem Zeitverhalten beurteilt. Hierzu wird die Regelstrecke allein, ohne Regler betrachtet (offene Wirkungskette, keine Rückführung).

## Bestimmung der Anlaufkennlinie

Für die Anlaufkennlinie wurde die Funktion  $h(t)$  mit Ausgleich 1. Ordnung verwendet. Warum Ausgleich 1. Ordnung verwendet werden kann wird mit dem „95% Kriterium“ erklärt. Da in diesem Fall die Regelstrecke gleich der Anlaufkennlinie des DC – Motors entspricht muss somit die Anlaufkennlinie bestimmt werden. Die Drehzahl steigt nach einer e-Potenz an und nach einer bestimmten Zeit „ $3 \cdot T$ “ befindet sich der Motor im Sättigungsbereich. Nach „ $5 \cdot T$ “ hat er annähernd die Nenndrehzahl erreicht. Als Zeitkonstante „ $T$ “ wurde die mechanische Zeitkonstante „ $\tau$ “ aus dem Datenblatt [3] des DC – Motors entnommen und für die Übergangsfunktion  $h(t)$  verwendet. Anhand der Übergangsfunktion kann der regelungstechnische Typ der Regelstrecke sowie die typischen Kenngrößen ermittelt werden. Aus dem Verhalten des Motors kann nun eine Regelstrecke 1. Ordnung ausgewählt werden.

## Bedeutung und Bestimmung von T

Hier wird mit dem „95% Kriterium“ überprüft ob sich die Anlaufkennlinie des DC-Motors, sich wie eine Regelstrecke 1. Ordnung verhält.

$$K_S := 1 \quad t := 0, 0.0001.. 0.1 \quad \tau := 5.4 \cdot 10^{-3} \quad t_1 := \tau$$

$$\text{tang}_{\text{fakt}} := \frac{K_S}{\tau} \quad \text{tang}_{\text{fakt}} = 185.185 \quad h(t) := K_S \cdot \left( 1 - e^{-\frac{t}{\tau}} \right)$$

$$\text{tang}(t) := \text{tang}_{\text{fakt}} t \quad \text{gerade}(t) := K_S \quad \text{krit95}(t) := K_S \cdot 0.95$$

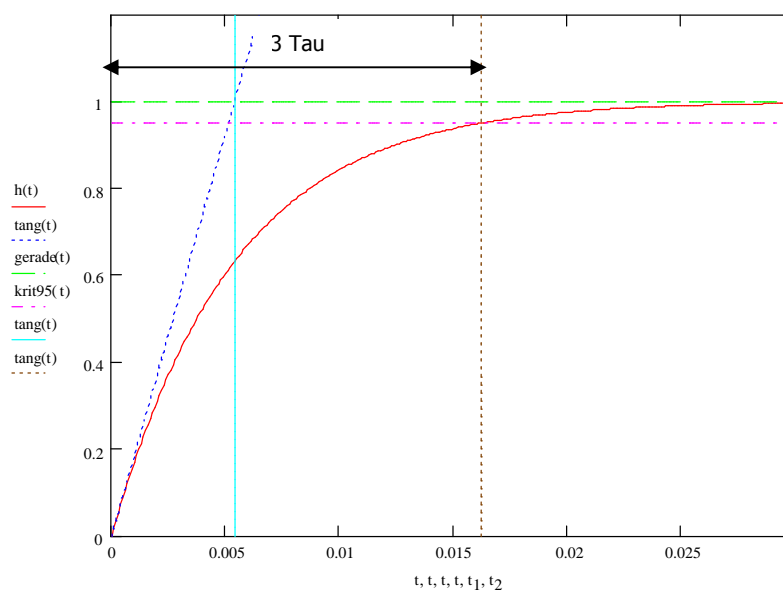


Abbildung 10: Überprüfung auf Ausgleich 1. Ordnung u. 95% Kriterium

## 7 Der Regler

Der Regler hat die Aufgabe, die gemessene Regelgröße (Istwert), mit dem Sollwert zu vergleichen und bei Abweichungen ( $x \neq w$  ergibt  $e \neq 0$ ) die Stellgröße so zu verändern, dass Soll- und Istwert der Regelgröße wieder übereinstimmen bzw. die Differenz minimal wird. Für weitere Informationen siehe [1].

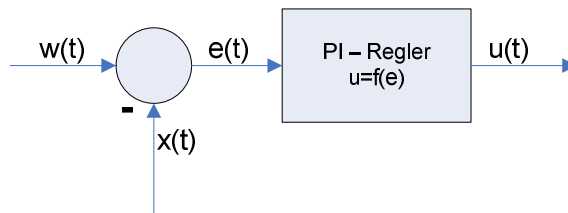


Abbildung 11: Blockschaltbild Regler

### 7.1 Wahl des Reglers

Für dieses Projekt wurde ein stetiger PI - Regler gewählt. Stetiger Regler, da die Ausgangsgröße des stetigen Reglers jeden beliebigen Wert ihres Stellbereiches (0% ... 100%) einnehmen kann. PI – Regler, weil bei einem reinen P-Regler eine ständige Regelabweichung auftritt und weil der PID–Regler zu schnell reagiert. Ungeeignet ist der PID-Regler für sehr schnelle Regelstrecken oder Regelstrecken mit pulsierenden Regelgrößen. Hier kann der D-Anteil zu Instabilitäten führen. Daher ist ein P- oder PID – Regler für ein „Inverses Pendel“ nicht geeignet.

### 7.2 Der PI–Regler

Ein PI-Regler besteht aus 2 Teilen, einem P-Anteil und einem I-Anteil. P steht für proportional wirkend und I steht für integral wirkend. Betrachtet man die dargestellte Reglergleichung, sind sofort beide Anteile erkennbar.

$$y(t) := K_R \cdot e(t) + K_I \int_0^t e(\tau) d\tau \qquad y(t) := K_R \cdot \left( e(t) + \frac{1}{T_n} \cdot \int_0^t e(\tau) d\tau \right)$$

### 7.3 Anpassung des Reglers an die Regelstrecke

Da weder Art der Regelstrecke noch Streckenparameter bekannt sind, so kann man die Reglerparameter mit dem Verfahren nach „Ziegler/Nichols“ trotzdem bestimmen. Zur Einstellung der Streckenparameter nach „Ziegler/Nichols“ siehe [1].

# 8 Die Architektur

## 8.1 Beschreibung der gesamten Architektur

Diese Architektur zeigt den Modularen Aufbau des Hardware-Designs im FPGA - Spartan II und besteht aus drei Modulen: Encoder-, Regler-, und PWM-Modul. Das Encoder Modul übernimmt die Frequenz-Messung des Encoder-Signals A und bestimmt die Drehrichtung mit Hilfe des Encoder-Signals B. Des Weiteren beinhaltet das Modul Regler die gesamte Regelung des Systems mittels PI-Regler. Durch ein externes Daten-Interface erhält der Regler die vorgegebene SOLL-Drehzahl (Drezahl in Takte). Das PWM-Modul gibt die Regelgröße des Reglers in Form eines PWM-Signals aus. Die Drehrichtung des Motors wird über die beiden Richtungs-Signale A und B über den Motortreiber gesteuert.

## 8.2 Blockschaltbild der gesamten Architektur

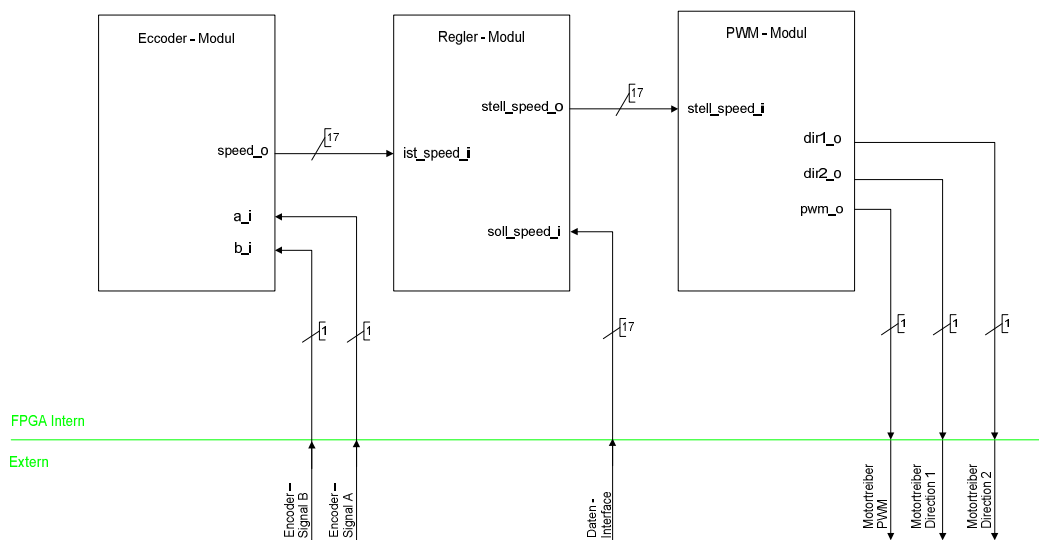


Abbildung 12: Blockschaltbild der gesamten Architektur

## 8.3 Modul: Encoder (Messeinrichtung)

### Flussdiagramm der Encoder-Architektur

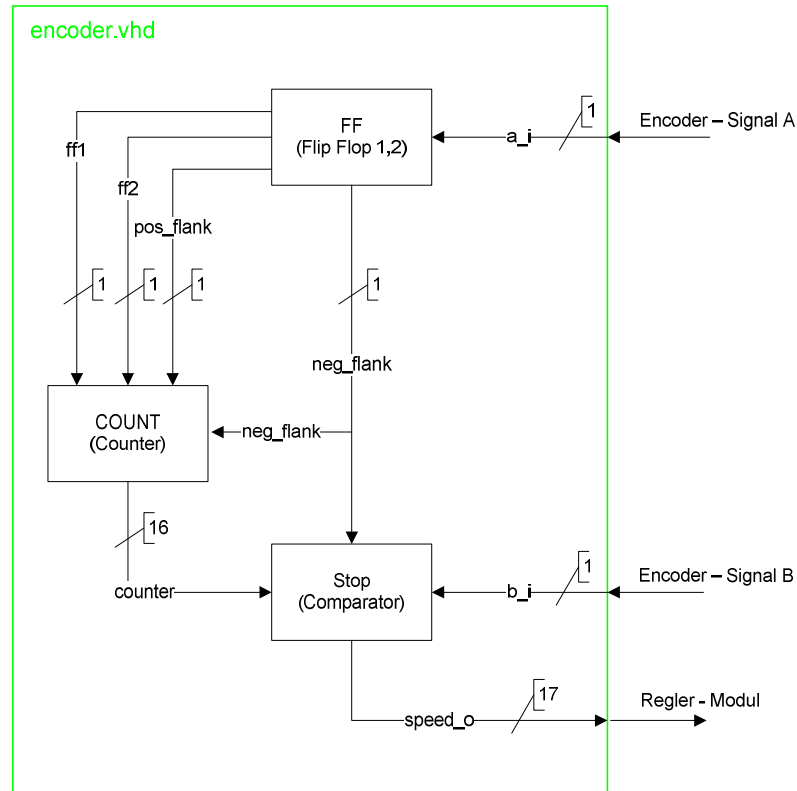


Abbildung 13: Flussdiagramm der Encoder-Architektur

### Beschreibung der Encoder-Architektur

#### **Process FF:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Prozess besteht aus zwei Flip Flops (FF1, FF2) und dienen zur Erkennung einer steigenden oder fallenden Flanke des Signals a\_in.

#### **Process STOP:** *Getakteter Prozess mit Asynchronen Reset*

Der Process STOP hat die Aufgabe bei auftreten einer fallenden Flanke den Wert des Signals „counter“ mit einen zusätzlichen Bit zu erweitern, das die Drehrichtung „LINKS“ oder „RECHTS“ des Motors angibt.

#### **Process COUNT:** *Getakteter Prozess mit Asynchronen Reset*

Hier wird die gemessene Encoder-Frequenz der Drehzahl gegenüber gestellt. Die Drehzahl ist in Form von Takten angegeben. Wird eine Messung gestartet beginnt der Counter „count\_max“ zu zählen. Dieser „count\_max“ bestimmt die Encoder-Frequenz von 18kHz, wobei der Drehzahl der Wert  $n = 65535$  zugewiesen wird. Ist die Encoder-Frequenz kleiner 18kHz so beginnt der

nächste Counter „counter“ zu zählen. Dieser Counter misst alle Frequenzen zwischen 18kHz und 126Hz. Dieser Bereich bestimmt alle möglichen Drehzahlen des Motors, von  $n = 65535$  (Maximum) bis  $n = 0$  (Minimum). Wird eine Encoder-Frequenz kleiner als 126Hz gemessen so wird diese Drehzahl als 0 definiert.

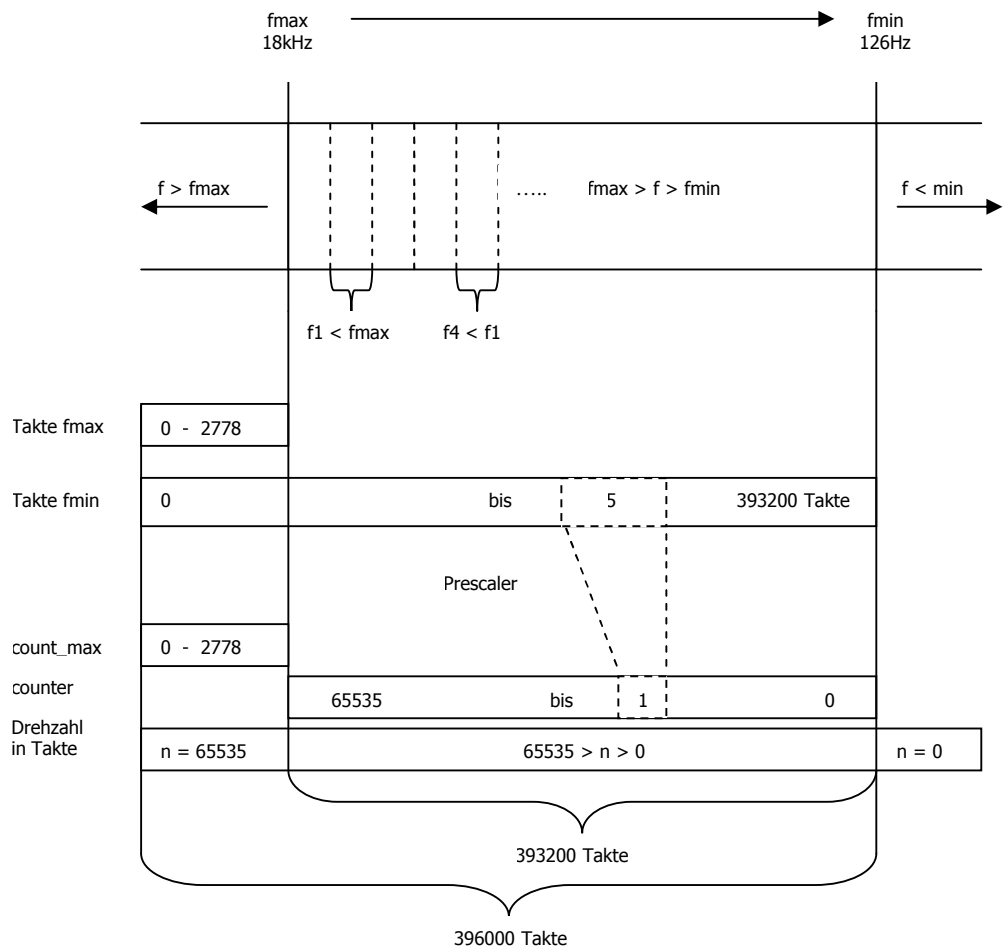


Abbildung 14: Darstellung Frequenzspektrum

## Berechnung von Prescaler, Taktefmax und Taktefmin

$$f_{\text{cpu}} := 48 \cdot 10^6 \text{ Hz} \quad T_{\text{cpu}} := 20 \cdot 10^{-9} \cdot \text{s} \quad \text{rps} = \text{s}^{-1} \quad \text{Auflösung} := 65535$$

### Berechnung Taktefmax (count\_max):

$$f_{\text{max}} := 18 \cdot 10^3 \text{ Hz}$$
$$T_{\text{fmax}} := \frac{1}{f_{\text{max}}} \quad T_{\text{fmax}} = 5.556 \times 10^{-5} \text{ s}$$

$$\text{Takte}_{\text{fmax}} := \frac{T_{\text{fmax}}}{T_{\text{cpu}}} \quad \text{Takte}_{\text{fmax}} = 2.778 \times 10^3$$

### Berechnung Prescaler(wait\_count), Taktefmin(counter):

$$f_{\text{min}} := 1 \text{ s}^{-1} \cdot 64 \cdot 2 \quad f_{\text{min}} = 128 \text{ Hz}$$

$$T_{\text{fmin}} := \frac{1}{f_{\text{min}}} \quad T_{\text{fmin}} = 7.813 \times 10^{-3} \text{ s}$$

$$\text{Takte}_{\text{fmin}} := \frac{T_{\text{fmin}}}{T_{\text{cpu}}} \quad \text{Takte}_{\text{fmin}} = 3.906 \times 10^5$$

$$\text{Prescaler} := \frac{\text{Takte}_{\text{fmin}}}{\text{Auflösung}} \quad \text{Prescaler} = 5.961$$

$$T_{\text{cpu}} := 20 \cdot 10^{-9} \cdot \text{s} \quad \text{Auflösung} := 65535 \quad \text{Prescaler} := 6$$

$$\text{Takte}_{\text{fmin}} := \text{Prescaler} \cdot \text{Auflösung} \quad \text{Takte}_{\text{fmin}} = 3.932 \times 10^5$$

$$\text{Takte}_{\text{fmin}} := \text{Takte}_{\text{fmin}} + \text{Takte}_{\text{fmax}} \quad \text{Takte}_{\text{fmin}} = 3.96 \times 10^5$$

$$T_{\text{fmin}} := \text{Takte}_{\text{fmin}} \cdot T_{\text{cpu}} \quad T_{\text{fmin}} = 7.92 \times 10^{-3} \text{ s}$$

$$f_{\text{min}} := \frac{1}{T_{\text{fmin}}} \quad f_{\text{min}} = 126.267 \text{ Hz}$$

## 8.4 Modul: Regler

### Flussdiagramm der Regler-Architektur

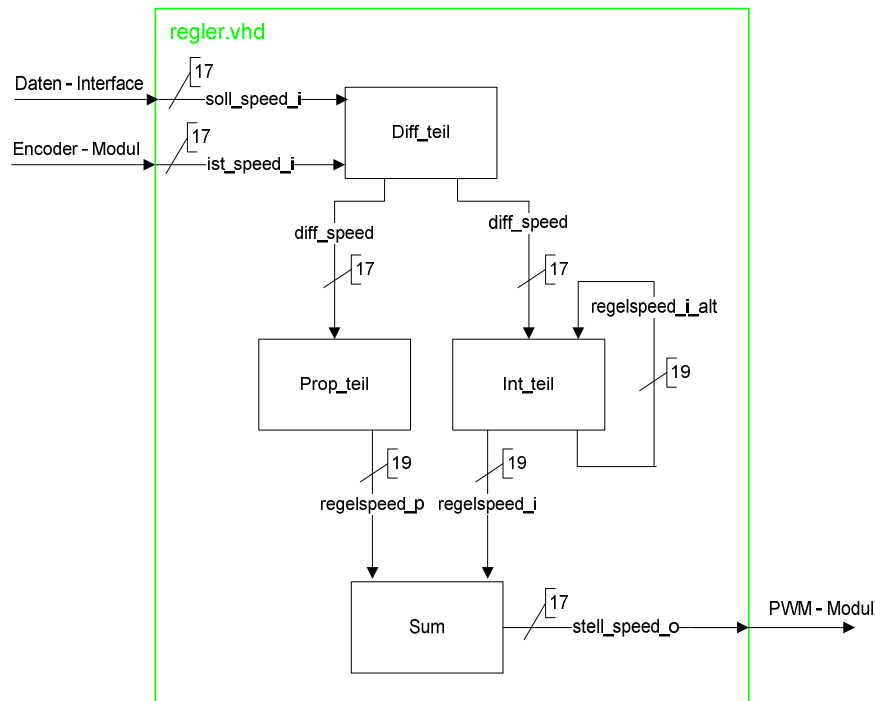


Abbildung 15: Flussdiagramm der Regler-Architektur

### Beschreibung der Regler-Architektur

#### Process diff\_teil: Getakteter Prozess mit Asynchronen Reset

Dieser Prozess dient zur Berechnung der Regelabweichung „e“. Durch Bildung der Differenz von SOLL-WERT und IST-WERT kann diese berechnet werden. Bei erfüllen der Bedingung von  $regel\_f = '1'$  beginnt der Regelzyklus.

#### Process prop\_teil: Getakteter Prozess mit Asynchronen Reset

In diesem Prozess wird der Proportionalteil des Reglers berechnet.

$$P := e \cdot K_P$$

#### Process int\_teil: Getakteter Prozess mit Asynchronen Reset

In diesem Abschnitt wird die Berechnung des Integralteiles der Regelabweichung „e“ durchgeführt. Der Faktor „t“ ist der Kehrwert der Regelzykluszeit. In unserem Fall kann dieser Faktor auf 1 gesetzt werden, da wir einen konstanten Regelzyklus haben. Zunächst wird der gesamte

Integralteil erstellt indem der aktuelle Integralteil und der gesamte Integralteil der vorherigen Berechnung addiert werden.

$$I := (e \cdot t \cdot K_I) \cdot I_{alt}$$

**Process sum:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Prozess hat die Aufgabe die Steuergröße „u“, durch Summe vom Proportional- und Integralteil, zu berechnen.

**Process regler\_loop:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Prozess wird verwendet um die Regelung (die oben genannten Prozesse) in einer konstanten Zeit, ständig zu wiederholen.

## 8.5 Modul: PWM

### Flussdiagramm der PWM-Architektur

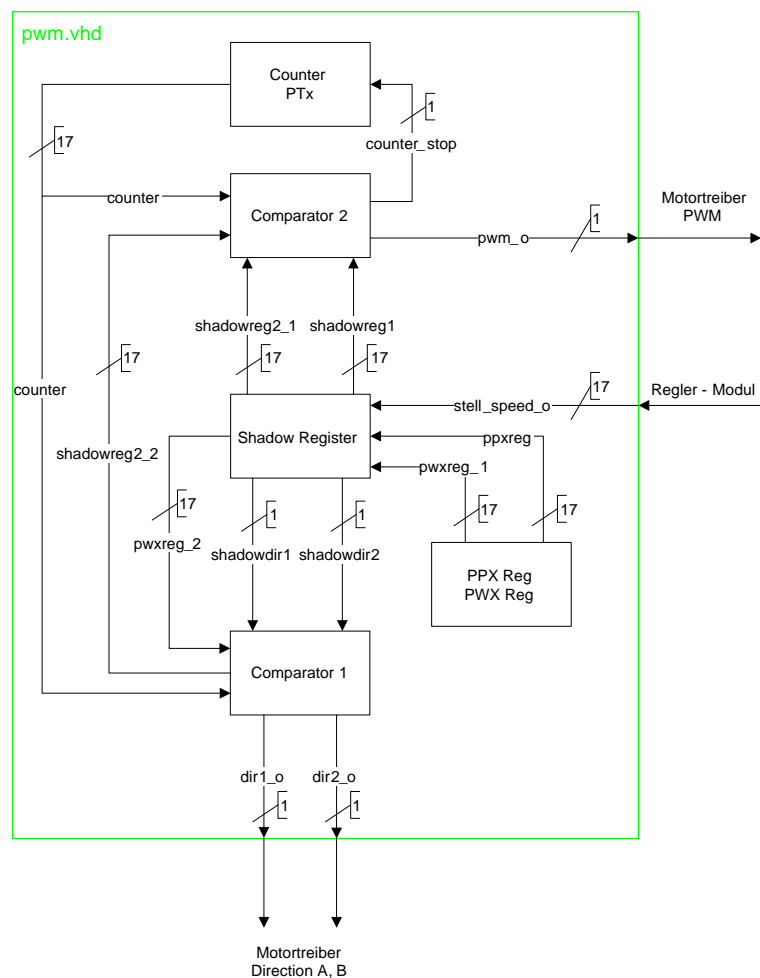


Abbildung 16: Flussdiagramm der PWM-Architektur

## Beschreibung der PWM-Architektur

### **Process counter:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Counter hat die Aufgabe die Dauer des PWM-Signals zu zählen.

### **Process shadow:** *Getakteter Prozess mit Asynchronen Reset*

Der Shadow-Prozess dient zum Zwischenspeichern der beiden Directionssignale für die Motordrehrichtung, der Regelgröße (`stell_speed_o`) die dem Duty-Cycle des PWM's proportional ist, den Startwert (`pwxreg_1`) des Duty-Cycles und dem Endwert (`ppxreg`) der die Frequenz des PWM's angibt. Diese Zwischenspeicherung der Register gewährleistet, dass das PWM-Signal während der Erzeugung nicht beeinträchtigt wird.

### **Process comparator1:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Comparator wird verwendet, um vor jeder neuen Generierung eines PWM's die Übernahme des Duty-Cycles und die Ausgabe der beiden Directionssignale zu gewährleisten. Durch diesen Comparator können während der Generierung eines PWM's die dafür benötigten Register (`shadowreg2_2`, `dir1_o`, `dir2_o`) nicht überschrieben werden. Die Übernahme der Werte erfolgt nur wenn der Counter den Zählstand 0 besitzt.

### **Process comparator2:** *Getakteter Prozess mit Asynchronen Reset*

Dieser Comparator ist für die Generierung des PWM-Signals verantwortlich.

## 9 Testen der Module

Das Testen der Module wird mit dem Programm ModelSim von Xilinx durchgeführt. Für jedes Modul wird eine eigene Testbench geschrieben die die Funktion des Moduls testet.

### 9.1 Testen des Encoder

Um die Maxima, Minima und die Richtungserkennung testen zu können muss das Encoder-Signal A und B mit den Frequenzen 18kHz ( $f_{max}$ ) und 122Hz ( $f_{min}$ ) generiert werden. Um die Richtungserkennung feststellen zu können wird eine Phasenverschiebung von  $90^\circ$  der Encoder-Signale vorgenommen. Bei der „Überprüfung auf  $f_{min}$ “ werden die Encoder-Signale nochmals um  $180^\circ$

phasenverschoben damit auch die gegengesetzte Drehrichtung überprüft werden kann.

## Überprüfen auf fmax

### Berechnung der Periodendauer für die Encoder-Signale

$$f_{\max} := 18 \cdot 10^3 \text{ Hz} \quad T_{f_{\max}} := \frac{1}{f_{\max}} \quad T_{f_{\max}} = 5.556 \times 10^{-5} \text{ s}$$

### Generierung der Signale a\_i und b\_i für fmax

Die Signale „a\_i“ und „b\_i“ werden nun, mit Hilfe der Testbench, mit der oben berechneten Periodendauer  $T_{f_{\max}}$ , einem Duty-Cycle von 50% und einer Phasenverschiebung von  $90^\circ$  generiert.

### Ergebnis der Simulation

Am Signal „speed\_o“ erkennt man den gemessenen Wert für das generierte Encoder-Signal. Die 16 niederwertigsten Bit geben die Drehzahl in Takte an. Diese 16 Bit stellen den Dezimalwert von 65535 dar, also maximale Drehzahl ( $f_{\max}$ ). Das höchstwertigste Bit ist in diesem Fall logisch '0' und gibt die Drehrichtung des Motors an.

## Überprüfen auf fmin

### Berechnung der Periodendauer für die Encoder-Signale

$$T_{\text{cpu}} := 20 \cdot 10^{-9} \cdot \text{s} \quad \text{Auflösung} := 65535 \quad \text{Prescaler} := 6$$

$$\text{Takte}_{f_{\min}} := \text{Prescaler} \cdot \text{Auflösung} \quad \text{Takte}_{f_{\min}} = 3.932 \times 10^5$$

$$\text{Takte}_{f_{\min}} := \text{Takte}_{f_{\min}} + \text{Takte}_{f_{\max}} \quad \text{Takte}_{f_{\min}} = 3.96 \times 10^5$$

$$T_{f_{\min}} := \text{Takte}_{f_{\min}} \cdot T_{\text{cpu}} \quad T_{f_{\min}} = 7.92 \times 10^{-3} \text{ s}$$

$$f_{\min} := \frac{1}{T_{f_{\min}}} \quad f_{\min} = 126.267 \text{ Hz}$$

### Generierung der Signale a\_i und b\_i für f\_min

Die Signale „a\_i“ und „b\_i“ werden nun, mit Hilfe der Testbench, mit der oben berechneten Periodendauer T\_fmin, einem Duty-Cycle von 50% und einer Phasenverschiebung von -90° generiert.

### Ergebnis der Simulation

Am Signal „speed\_o“ wird wieder erkannt, dass der Messwert der Encoder-Frequenz (f\_min) entspricht. Das höchstwertigste Bit ist in diesem Fall logisch '1' und gibt die gegengesetzte Drehrichtung des Motors an. Es wurde auch erkannt, dass ein kleiner Rundungsfehler mit drinnen ist, das heißt der Counter läuft 220ns bevor die negative Flanke kommt.

## 9.2 Testen des Reglers

Um die Maximale Regelabweichung zu testen muss z.B. ein negativer IST-WERT und ein positiver SOLL-WERT vorhanden sein. Es können nur Sprunghafte Änderungen für einen Durchlaufzyklus simuliert werden, da die Stellgröße in der Simulation keine Änderung des IST-WERTES hervorruft.

### Überprüfen der Maximal auftretbaren Regelabweichung

soll\_speed\_i := 65535      ist\_speed\_i := -65535      regelspeed\_i\_alt := 0  
KP := 1      KI := 1      t\_mul := 1

### Berechnung der Regelabweichung

diff\_speed := ist\_speed\_i - soll\_speed\_i      diff\_speed = -1.311 × 10<sup>5</sup>

diff\_speed wird auf Maximum (65535) zugeschnitten

diff\_speed := 65535

### Berechnung des Proportionalteiles

regelspeed\_p := diff\_speed · KP      regelspeed\_p = 6.553 × 10<sup>4</sup>

Anschließend wird „regelspeed\_p“ um 5 Bits nach rechts verschoben, das heißt „regelspeed\_p“ wird durch 32 dividiert.

regelspeed\_p :=  $\frac{\text{regelspeed\_p}}{32}$       regelspeed\_p = 2.048 × 10<sup>3</sup>

### Berechnung des Integralteiles

$$\text{regelspeed\_i} := \text{diff\_speed} \cdot \text{KI} \cdot \text{t\_mul} \quad \text{regelspeed\_i} = 6.553 \times 10^4$$

Anschließend wird „regelspeed\_i“ um 7 Bits nach rechts verschoben, das heißt „regelspeed\_i“ wird durch 128 dividiert.

$$\text{regelspeed\_i} := \frac{\text{regelspeed\_i}}{128} \quad \text{regelspeed\_i} = 511.992$$

$$\text{regelspeed\_i\_ges} := \text{regelspeed\_i} + \text{regelspeed\_i\_alt} \quad \text{regelspeed\_i\_ges} = 511.992$$

$$\text{regelspeed\_i\_alt} := \text{regelspeed\_i\_ges} \quad \text{regelspeed\_i\_alt} = 511.992$$

$$\text{regelspeed\_i} := \text{regelspeed\_i\_ges} \quad \text{regelspeed\_i} = 511.992$$

### Berechnen des Stellwertes

$$\text{stell\_speed\_o} := \text{regelspeed\_p} + \text{regelspeed\_i\_ges} \quad \text{stell\_speed\_o} = 2.56 \times 10^3$$

### Ergebnis der Simulation

Alle Ergebnisse stimmen soweit, bis auf kleine Rundungsfehler von 1-2.

## 9.3 PWM

Um einen Duty-Cycle von 0, 50, 100% testen zu können muss die Stellgröße vom Regler den Wert 0, 32767, 65535 besitzen.

### Ergebnis der Simulation

**Duty-Cycle:** 0% -> PWM-Signal ist für die ganze Periodendauer auf logisch “0“  
50% -> das Signal ist für die erste Hälfte der Periodendauer auf logisch “1“ und für die zweite Hälfte auf logisch “0“  
100% -> PWM-Signal ist für die gesamte Periodendauer auf logisch “1“

## 10 Literatur

- [1] Prof. Helmut Papp, Grundlagen Regelungstechnik, <http://www.vs-c.de>
- [2] Klaus Tkotz, Fachkunde Elektrotechnik, Europa-Lehrmittel, 1993
- [3] Faulhaber GmbH & Co. KG, Datenblatt - DC\_Motor\_2642W012CR
- [4] Faulhaber GmbH & Co. KG, Datenblatt - Encoder\_IE2\_512
- [5] ST-Microelectronics, Datenblatt - Motortreiber\_VNH3ASP30-E
- [6] Xilinx, Datenblatt – FPGA\_SPARTAN\_II, 2004