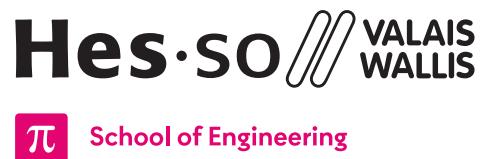


Cursor (Cur)

Vorlesung Digitales Design (DiD)



Orientierung: Informatik und Kommunikationswissenschaften (ISC)

Spezialisierung: Data Engineering (DE)

Kurs: Digitales Design (DiD)

Authoren: Silvan Zahno, Axel Amand, François Corthay, Christophe Bianchi

Datum: 01.12.2025

Version: v3.1



Inhalt

1 Einführung	3
2 Spezifikation	4
2.1 Funktionen	4
2.2 Schaltung	5
2.3 Szenario (Beispiel)	6
2.4 HDL-Designer Projekt	6
3 Komponenten	8
3.1 Schlitten	8
3.2 Motorsteuerungsschaltung	8
3.2.1 Gleichstrommotor	9
3.3 Encoder (Drehgeber)	11
3.4 Reed-Relais	12
3.5 FPGA-Platine	12
3.6 Knöpfe und LEDs	13
4 Implementierung der Motorenrampen	14
4.1 Grundkonzept der Rampen	14
4.2 Naives Rampenprofil	15
4.3 Probleme und Verbesserungen der naiven Lösung	17
4.4 Endgültige Rampe	18
4.5 Überprüfung der Annahmen	19
5 Bewertung	20
6 Erste Schritte	21
6.1 Tips	21
Glossar	22



1 | Einführung

Ziel dieses Projekts ist es, das erworbene Wissen anhand eines praktischen Beispiels am Ende des Semesters konkret anzuwenden. Es umfasst die Steuerung eines Gleichstrommotors, um einen Schlitten präzise entlang einer Gewindespindel zu vordefinierten Positionen zu bewegen. Dieses Positionierungssystem ist in der [Abbildung 1](#) dargestellt.

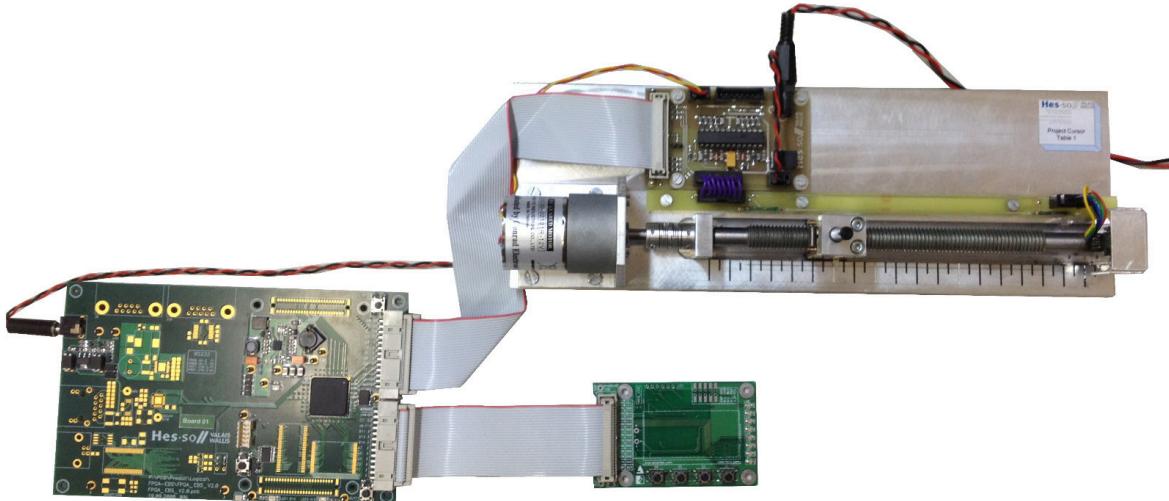


Abbildung 1 - Hardwareaufbau Cursor (EBS2)

Die Mindestspezifikationen (siehe [Abschnitt 2](#)) müssen erfüllt sein, aber die Studierenden werden ermutigt, zusätzliche Funktionen zu implementieren. Beispielsweise kann ein [Liquid Crystal Display \(LCD\)](#)-Bildschirm verwendet werden, um verschiedene Informationen anzuzeigen.



Durch die Implementierung zusätzlicher Funktionen können zusätzliche Punkte in der abschliessenden Bewertung erzielt werden.



2 | Spezifikation

2.1 Funktionen

Die Grundfunktionen sind wie folgt definiert:

- Wenn die Taste **restart** gedrückt wird, bewegt sich der Cursor zur Startposition, die durch ein Reed-Relais ([Abschnitt 3.4](#)) in der Nähe des Gleichstrommotors ([Abschnitt 3.2.1](#)) angezeigt wird.
- Wenn die Taste **Position₁** gedrückt wird, muss der Cursor zuerst gleichmässig zur Position 1 (p_1) beschleunigen, dann mit voller Geschwindigkeit vorwärts fahren und schliesslich gleichmässig abbremsen, um an Position 1 (p_1) anzuhalten. Dies kann von der Startposition oder von Position 2 aus erfolgen, siehe [Abbildung 2](#).
- Wenn die Taste **Position₂** gedrückt wird, muss der Cursor zuerst gleichmässig zur Position 2 (p_2) beschleunigen, dann mit voller Geschwindigkeit vorwärts fahren und schliesslich gleichmässig abbremsen, um an Position 2 (p_2) anzuhalten, siehe [Abbildung 2](#).

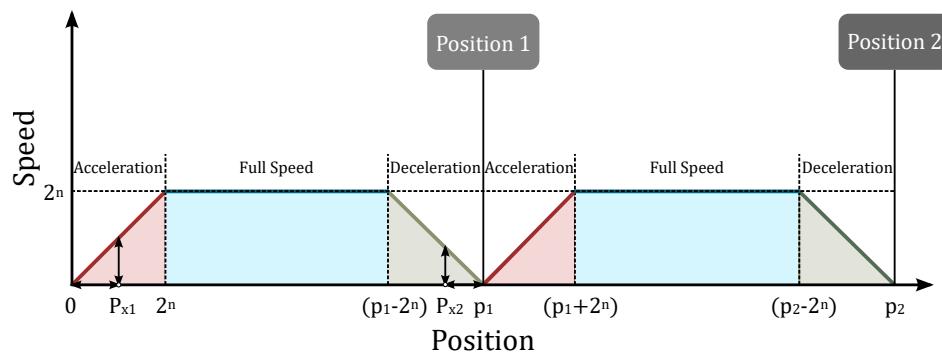


Abbildung 2 - Diagramm der Geschwindigkeit des Schlittens

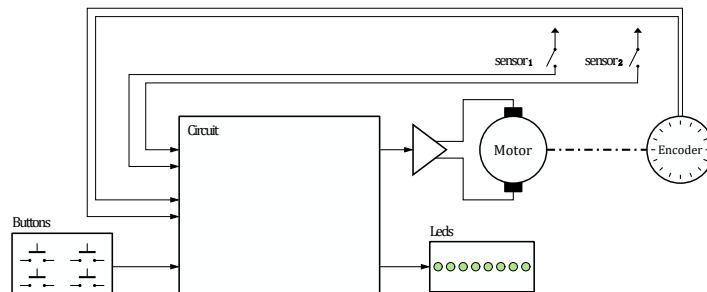


Abbildung 3 - Cursor Schaltung



2.2 Schaltung

Der Schaltungsaufbau ist wie folgt:

- Der Gleichstrommotor ([Abschnitt 3.2.1](#)) wird durch die drei Signale motor_{On} , side_1 , side_2 gesteuert:
 - Der Motor wird aktiviert, wenn das Signal motor_{On} auf 1 gesetzt ist und muss auf 0 zurückgesetzt werden, wenn der Motor nicht laufen soll.
 - Seine Geschwindigkeit wird durch eine **Pulse Width Modulation (PWM)** Modulation gesteuert, die entweder auf das Signal side_1 oder auf das Signal side_2 angewendet wird, je nach gewünschter Drehrichtung.
- Zwei Reed-Relais ([Abschnitt 3.4](#)) sensor_1 und sensor_2 sind an den Enden der Schiene [1] angebracht. Sie erkennen die Anwesenheit des Cursor-Schlittens, indem sie das Vorhandensein eines Magneten mit einem 1 anzeigen.
- Der Encoder ([Abschnitt 3.3](#)) wird verwendet, um die Position des Cursors zu verfolgen bzw. zu zählen. Seine drei Ausgänge, encoder_A , encoder_B und encoder_I , ermöglichen es, die Bewegungen der Spindel zu verfolgen. Die Ausgänge A und B wechseln zwischen 0 und 1 während der Bewegungen mit einer Phasenverschiebung von 90° zwischen ihnen, was es ermöglicht, die Bewegungsrichtung zu bestimmen. Der Ausgang I erzeugt einen 1-Impuls bei jeder vollständigen Umdrehung der Spindel.
- Drei Tasten werden verwendet, um das System zu steuern: restart , go_1 und go_2 . Eine zusätzliche Taste, button_4 , kann für optionale Funktionen verwendet werden. Wenn eine der Tasten gedrückt wird, wechselt das entsprechende Signal auf 1.
- Die Pins testOut können verwendet werden, um zusätzliche Informationen aus dem System auszugeben, z.B. um sie an die **Light Emitting Diodes (LEDs)** für Debugging- oder Visualisierungszwecke anzuschließen.
- Das Signal testMode wird auf 1 gesetzt während der Simulation. Dieses kann verwendet werden, um die Zähler zu verkürzen, die zur Signalerzeugung während der Tests verwendet werden, um diese zu beschleunigen.

Das leere Toplevel Design ([cursor-toplevel-empty.pdf](#)) zeigt alle Signale, die mit dem **Field Programmable Gate Array (FPGA)** Board verbunden sind [Abbildung 4](#).



Abbildung 4 - Leere Toplevel Schaltung



2.3 Szenario (Beispiel)

In [Abbildung 5](#) werden drei verschiedene Szenarien dargestellt. Zuerst wird die restart-Taste gedrückt und der Schlitten bewegt sich mit voller Geschwindigkeit zur Startposition (sensor₁). Die beiden anderen Szenarien go₂ und go₁ bewegen den Schlitten zur position₂ und zur position₁ respektive, ein variables **PWM** Signal wird auf die Signale side₂ und side₁ angewendet, um den Schlitten zu beschleunigen und zu verlangsamen.

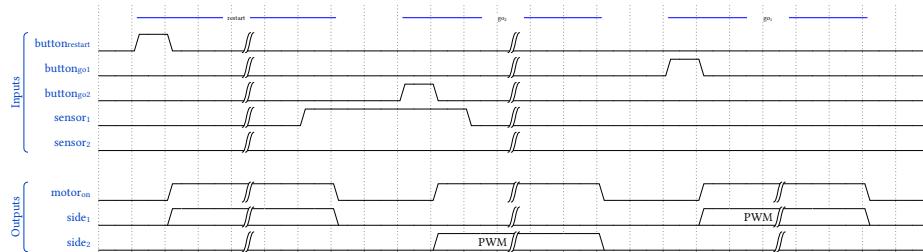


Abbildung 5 - Cursor Szenario



Die oben genannten Szenarien sind Beispiele, die von den Studierenden vervollständigt werden müssen.

2.4 HDL-Designer Projekt

Ein vordefiniertes HDL-Designer Projekt kann von [Cyberlearn](#) oder [Github](#) heruntergeladen oder geklont werden. Die Dateistruktur des Projekts sieht wie folgt aus:

```

did_cursor
+--Board/          # Project and files for programming the FPGA
|   +-concat/      # Complete VHDL file including PIN-UCF file
|   +-ise/          # Xilinx ISE project
+--Cursor/          # Library for the components of the student solution
+--Cursor_test/    # Library for the simulation testbenches
+--doc/             # Folder with additional documents relevant to the project
|   +-Board/        # All schematics of the hardware boards
|   +-Components/  # All data sheets of hardware components
+--img/              # Pictures
+--Libs/             # External libraries which can be used e.g. gates, io, sequential
+--Prefs/            # HDL-Designer settings
+--Scripts/          # HDL-Designer scripts
+--Simulation/       # Modelsim simulation files

```



Der Pfad des Projektordners darf keine Leerzeichen enthalten.



Im Projektordner **doc/** können viele wichtige Informationen gefunden werden. Datenblätter, Projektbewertung sowie Hilfsdokumente für HDL-Designer um nur einige zu nennen.



3 | Komponenten

Das System besteht aus 3 verschiedenen Hardwareplatinen, die in der [Abbildung 1](#) zu sehen sind.

- Ein Schlittenaufbau mit einer **Printed Circuit Board (PCB)**-Platine, die den Motor steuert und die Sensoren ausliest ([Abbildung 6](#))
- Eine **FPGA**-Entwicklungsplatine ([Abbildung 14](#) oder [Abbildung 15](#))
- Eine Steuerplatine mit 4 Tasten und 8 LEDs ([Abbildung 16](#))

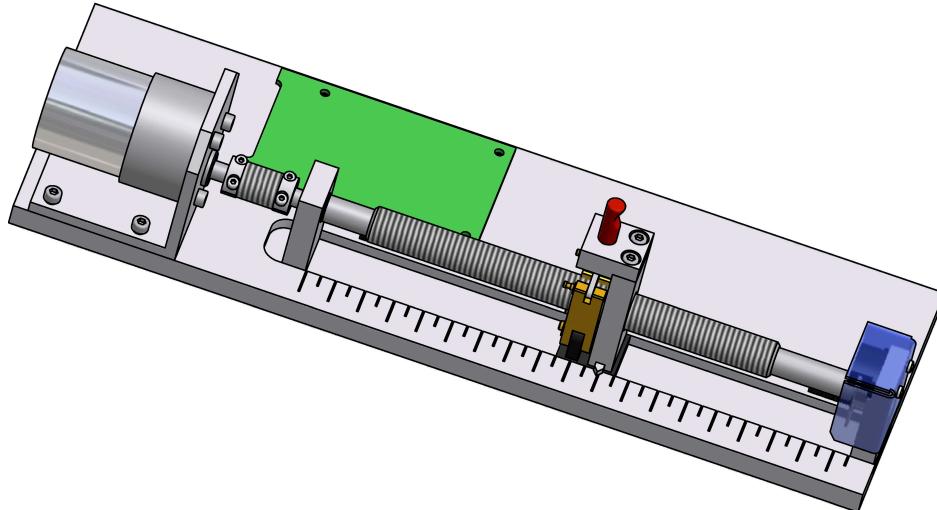


Abbildung 6 - Cursor Schlittenaufbau

3.1 Schlitten

Der Schlittenaufbau beinhaltet den Gleichstrommotor, die beiden Reed-Relais [Abschnitt 3.4](#) sowie den Schlitten und die Spindel. Der Spindelsteigung ist M12x1.75, was bedeutet, dass pro Umdrehung eine Strecke von 1.75mm zurückgelegt wird ([Abbildung 7](#)).

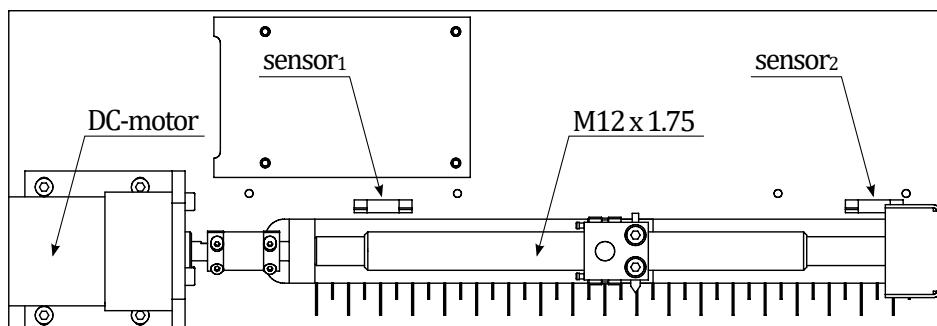


Abbildung 7 - Cursor Schlittenaufbau Detail

3.2 Motorsteuerungsschaltung

Der Gleichstrommotor des Wagens wird mit 12V versorgt. Die Stromversorgungsplatine besitzt eine H-Brücke, die durch digitale Signale gesteuert wird. Auf der Stromversorgungsplatine erzeugt ein 5V-Regler die richtige Spannung für die Versorgung der **FPGA**-Platine [2].



3.2.1 Gleichstrommotor

Der Gleichstrommotor wird von einem L6207 H-Brücken-Treiber [3] gesteuert, siehe Abbildung Abbildung 8. Die Maximale Schaltfrequenz der H-Brücke beträgt 100kHz. Dies sollte bei der Erstellung des PWM Signales in Betracht gezogen werden.

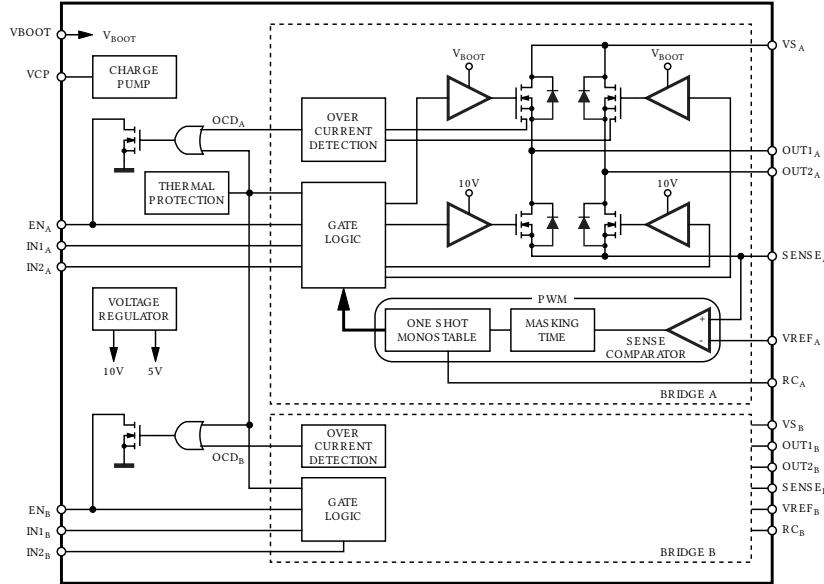


Abbildung 8 - H-Brücke L6207N Schaltung [3]

Um die Geschwindigkeit des Gleichstrommotors anzupassen, muss ein PWM-Signal an die side₁ oder side₂ Signale angelegt werden, wobei die maximale Frequenz 100kHz beträgt. Je länger die Spannung am Motor angelegt wird, desto schneller dreht er sich. Die Richtung des Motors wird gesteuert, indem entweder side₁ oder side₂ mit Spannung versorgt wird.

PWM - Implementierungsstrategie

Eine Strategie besteht darin, einen Sägezahnzähler ($0, 1, 2 \dots 2^N - 2, 2^N - 1, 0, 1, \dots$) zu erstellen und seinen Wert mit einem Schwellenwert zu vergleichen. Die Regel ist, „1“ auszugeben, wenn der Zählerwert kleiner als der Schwellenwert ist, und „0“ sonst. Durch Ändern des Schwellenwerts wird das Tastverhältnis des PWM-Signals verändert. Auf diese Weise wird die Frequenz des PWM-Signals durch die Geschwindigkeit des Zählers (Sägezahn) definiert, und das Tastverhältnis wird durch den Schwellenwert definiert.

In der Abbildung 9 dreht sich der Motor langsamer mit dem grünen Signal als mit dem blauen Signal und als mit dem roten Signal.

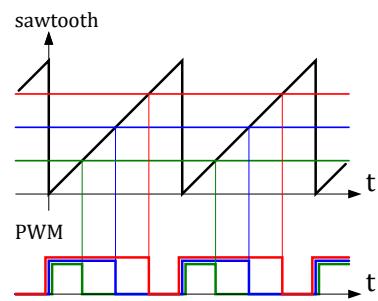


Abbildung 9 - PWM Signale



Je höher die Anzahl der Schritte (Bits) der **PWM**, desto präziser ist die Geschwindigkeitsregelung. Allerdings können zu viele Schritte aufgrund von Steuerungssystemen und Motoreigenschaften zu keinem bemerkenswerten Geschwindigkeitsunterschied führen. Eine gute Basis ist die Verwendung von 8 Bits für die **PWM**-Auflösung, was 256 Geschwindigkeitsstufen von 0 bis 255 ergibt.

Rampen - Implementierungsstrategien

Um Beschleunigungs- und Verzögerungsrampen zu erstellen, ist eine gängige Methode die Verwendung eines trapezförmigen Geschwindigkeitsprofils. Dieses Profil besteht aus drei Phasen: Beschleunigung, konstante Geschwindigkeit und Verzögerung, wie in [Abbildung 10](#) dargestellt.

Mehrere Strategien existieren, um solche Rampen zu erstellen. Eine - einfacher zu implementierende - Methode wird in [Abschnitt 4](#) beschrieben.

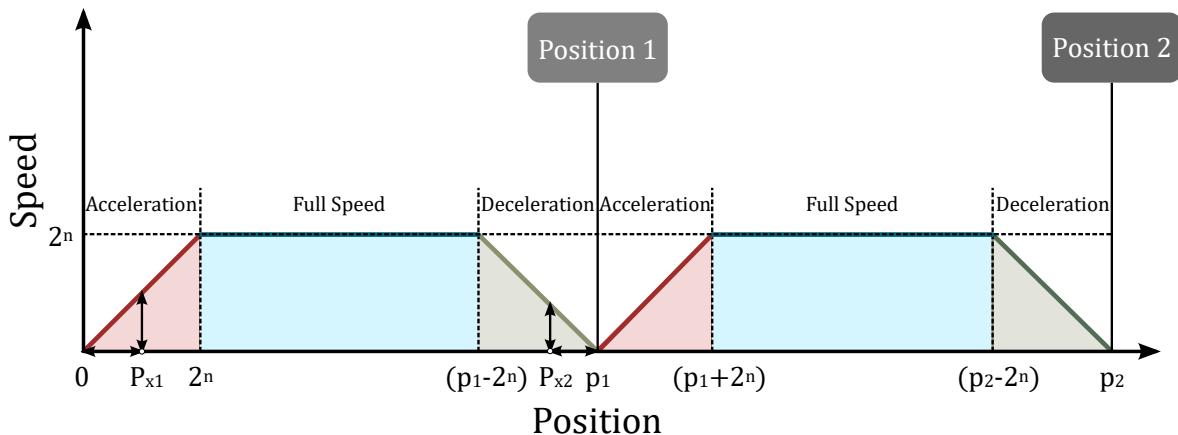


Abbildung 10 - Diagramm der Geschwindigkeit des Schlittens

3.3 Encoder (Drehgeber)

Der Winkel der Spindel kann mithilfe eines **inkrementalen Drehgebers** gemessen werden. Das Modell, das im Aufbau verwendet wird, ist ein AEDB-9140-A12 [4] ([Abbildung 11](#)) mit 2 Rückführungskanälen mit 500 **Counts per Revolution (CPR)** (Impulse pro Umdrehung) pro Kanal, dargestellt in der [Abbildung 12](#).



Abbildung 11 - Encoder AEDB-9140-A12 [4]

Encoder - Implementierungsstrategie

Es ist möglich, ihn auf zwei Arten zu verwenden:

- Durch die Verwendung eines einzigen Kanals, was eine Auflösung von $500 \frac{\text{Impulse}}{\text{Umdrehung}}$ ergibt, d.h. $1'000 \frac{\text{Flanken}}{\text{Umdrehung}}$, wenn sowohl steigende als auch fallende Flanken gezählt werden.
- Durch die Verwendung beider Kanäle als **QEI (Quadrature Encoder Interface)**, indem die steigenden und fallenden Flanken beider Kanäle gezählt werden, wodurch die Auflösung auf $2000 \frac{\text{Flanken}}{\text{Umdrehung}}$ erhöht wird. Diese Schnittstelle ermöglicht auch die Erkennung der Drehrichtung



des Motors und ist eine gängige Methode zur Steuerung von Gleichstrommotoren mit Rückführungs-Encodern in der Industrie.

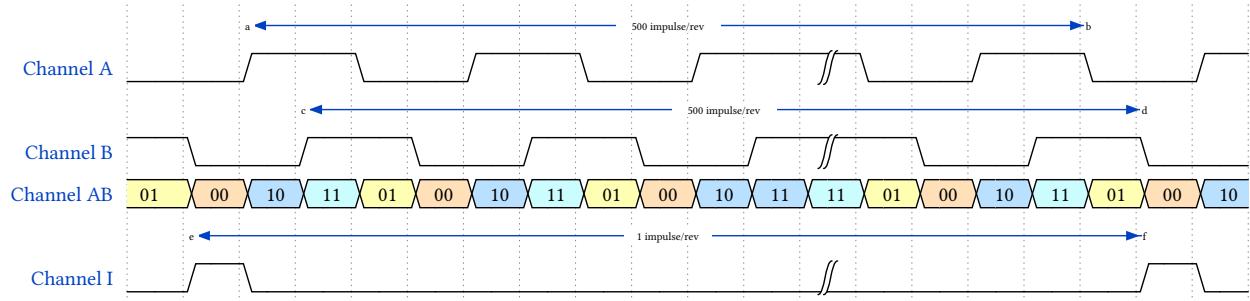


Abbildung 12 - Signale inkrementaler Drehgeber

3.4 Reed-Relais

Das Reed-Relais ist ein Schalter, der mit Hilfe von Elektromagneten umgeschaltet werden kann [1]. Wenn sich ein Magnet in der Nähe des Sensors befindet, schliesst sich der Kontakt (Abbildung 13). 2 Reed-Relais werden verwendet (sensor_1 und sensor_2), um die linke und rechte Grenze des Schlittens zu identifizieren.

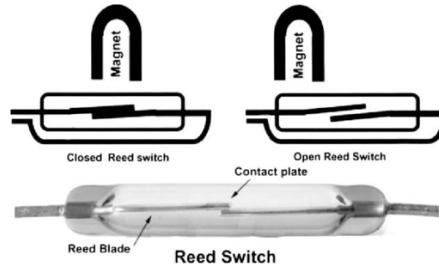


Abbildung 13 - Reed Schalter [5]

Die Signale sensor_1 und sensor_2 sind ,1‘, wenn der Kontakt geschlossen ist (Magnet in der Nähe), sonst ,0‘.

3.5 FPGA-Platine

Die Hauptplatine ist die FPGA-EBS 2 Laborentwicklungsplatine der Schule [6]. Diese beherbergt eine Xilinx Spartan xc3s500e FPGA [7], [8] und verfügt über viele verschiedene Schnittstellen (Universal Asynchronous Receiver Transmitter (UART), Universal Serial Bus (USB), Ethernet, etc.). Der benutzte Oszillator erstellt ein Taktsignal (**clock**) mit einer Frequenz von $f_{\text{clk}} = 66\text{MHz}$ [9].

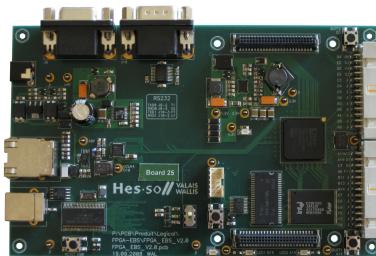
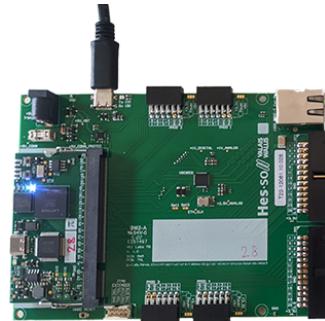


Abbildung 14 - FPGA Platine [6]

Auf der EBS3-Karte erzeugt der verwendete Oszillator ein Taktsignal (**clock**) mit einer Frequenz von $f_{\text{clk}} = 100\text{MHz}$, die durch PLL auf $f_{\text{clk}} = 60\text{MHz}$ reduziert wird.

Abbildung 15 - **FPGA Platine** [10]

Die Simulators sind standardmäßig für die EBS3 boards eingestellt. Um sie zu ändern, öffnen Sie einen Block von testbench **xxx_tb** und doppelklicken Sie auf die **Pre-User**-Deklarationen (oben links auf der Seite), um die Variable **clockFrequency** auf den gewünschten clock-Wert zu ändern.

3.6 Knöpfe und LEDs

Die Platine mit den Knöpfen und **LEDs** [11] wird an die **FPGA** Platine angeschlossen. Sie hat 4 Tasten und 8 **LEDs**, die im Design verwendet werden können. Falls gewünscht kann diese Platine mit einer **LCD** Anzeige ausgestattet werden [12], [13].

Abbildung 16 - **Knöpfe-LEDs-LCD** Platine [11]



4 | Implementierung der Motorenrampen

Dieses Kapitel ist ein Vorschlag zur Implementierung von Beschleunigungsrampen. Es gibt andere Methoden, die ebenfalls verwendet werden können.

Um Beschleunigungs- und Verzögerungsrampen zu erstellen, ist eine gängige Methode die Verwendung eines trapezförmigen Geschwindigkeitsprofils. Dieses Profil besteht aus drei Phasen: Beschleunigung, konstante Geschwindigkeit und Verzögerung, wie in Abbildung 17 dargestellt.

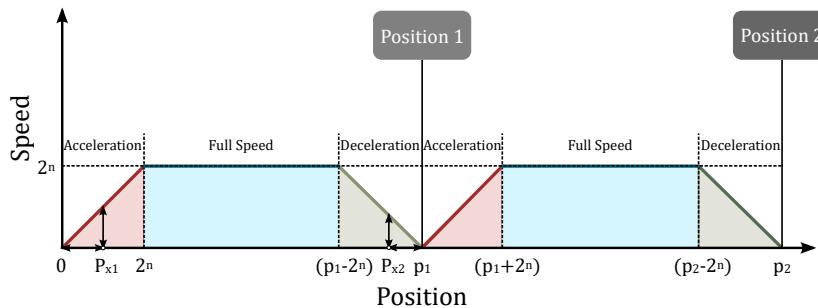


Abbildung 17 - Diagramm der Geschwindigkeit des Schlittens

Sowohl Beschleunigungs- als auch Verzögerungsrampen verwenden dasselbe Profil.

4.1 Grundkonzept der Rampen

Typischerweise basieren Geschwindigkeitsprofile auf der Zeit. Diese Methode wird häufig in industriellen Anwendungen verwendet.

Solche Systeme benötigen jedoch eine vollständige Rückkopplungsschleife, um die tatsächliche Motorgeschwindigkeit in Abhängigkeit von der Zeit genau zu messen, sowie eine Steuerungsschleife, um eine präzise Positionierung zu ermöglichen. Der Motor könnte aufgrund von Laständerungen, Reibung, Verschleiß, Alterung... langsamer werden. Außerdem wäre es schwierig zu definieren, wann die Verzögerungsphase beginnen soll, ohne das System vollständig zu charakterisieren.

Um dem entgegenzuwirken, basiert dieser Rampenvorschlag stattdessen auf der Position.

Aufgrund der Natur dieses Konzepts reguliert sich das System automatisch (kann jedoch Laständerungen nicht ausgleichen).



4.2 Naives Rampenprofil

Um ein vollständiges Profil zu erhalten, müssen sich die Phasen der Beschleunigung + Verzögerung innerhalb der kleinsten verfügbaren Entfernung zwischen zwei Zielpositionen abschließen. Hier beträgt diese Entfernung 4cm von Position p1 (80mm) bis p2 (120mm).

Die Rampen sollten also etwa 1cm (um sichtbar genug zu sein) bis unter 2cm (um Zeit für die volle Geschwindigkeit zu lassen) verwenden, um sich zu vervollständigen.

Die naive Rampe ist unter [Abbildung 18](#) dargestellt:

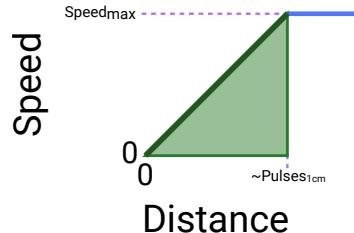


Abbildung 18 - Naive Beschleunigungsrampe

Um diese Rampe zu erstellen, wird eine lineare Gleichung verwendet, die die Geschwindigkeit als Funktion der Position darstellt:

$$v_x = m * x + b$$

mit

$$m = \frac{\Delta y}{\Delta x} = \frac{\text{speed}_{\max} - 0}{\sim \text{pulses}_{\text{for_1_cm}} - 0}$$

$$b = 0$$

Um alle Berechnungen zu vereinfachen und keine Hardware-Multiplikatoren und -Teiler zu benötigen, werden nur Potenzen von zwei als Argumente der Funktion verwendet:

- speed_{\max} entspricht der PWM-Auflösung, d.h. 2^{pwmBitNb} mit $\text{pwmBitNb} = 8$ standardmäßig
- $\sim \text{pulses}_{\text{for_1_cm}}$ hängt davon ab, wie die Impulse gezählt werden:
 - Verwendung eines einzelnen Encoders, d.h. $500 \frac{\text{pulses}}{\text{revolution}}$:
 - $\sim \text{pulses}_{\text{for_1_cm}} = \frac{1 \text{ cm}}{3.5 \frac{\text{mm}}{\text{pulse}}} = 2'857 \text{ pulses} \approx 2^{12} = 4'096 \text{ pulses} \rightarrow 4'096 \text{ pulses} * 3.5 \mu\text{m} = 1.425 \text{ cm}$
 - Verwendung eines einzelnen Encoders bei Zählung beider Flanken, d.h. $1'000 \frac{\text{pulses}}{\text{revolution}}$:
 - $\sim \text{pulses}_{\text{for_1_cm}} = \frac{1 \text{ cm}}{1.75 \frac{\text{mm}}{\text{pulse}}} = 5'714 \text{ pulses} \approx 2^{13} = 8'192 \rightarrow 8'192 \text{ pulses} * 1.75 \mu\text{m} = 1.425 \text{ cm}$
 - Verwendung beider Encodersignale, d.h. $2'000 \frac{\text{pulses}}{\text{revolution}}$ (QEI-Modus):
 - $\sim \text{pulses}_{\text{for_1_cm}} = \frac{1 \text{ cm}}{875 \frac{\text{mm}}{\text{pulse}}} = 11'428 \text{ pulses} \approx 2^{14} = 16'384 \rightarrow 16'384 \text{ pulses} * 875 \text{ nm} = 1.425 \text{ cm}$

Der Rest dieses Vorschlags basiert auf der QEI-Methode, der genauesten der drei.

Mit diesen definierten Variablen sind die Steigungsparameter:

$$m = \frac{\text{speed}_{\max} - 0}{\sim \text{pulses}_{\text{for_1_cm}} - 0} = \frac{2^8}{2^{14}} = \frac{1}{2^6} = \frac{1}{64}$$

$$b = 0$$



Das bedeutet, die Steigung würde die Form annehmen: $v_x = \frac{x}{2^6}$.

Die Division durch eine Potenz von zwei kann durch eine Rechtsverschiebung vereinfacht werden:
 $v_x = x >> 6$



Da der Steigungsfaktor m kleiner als 1 ist und die Schaltung nur Ganzzahlen verarbeitet, wird die Geschwindigkeitskurve nur alle 2^N Impulse ansteigen. Das bedeutet, die Geschwindigkeit wird in Stufen ansteigen, nicht glatt. Dies ist für diese Anwendung akzeptabel. Siehe [Abbildung 19](#) zur Veranschaulichung.

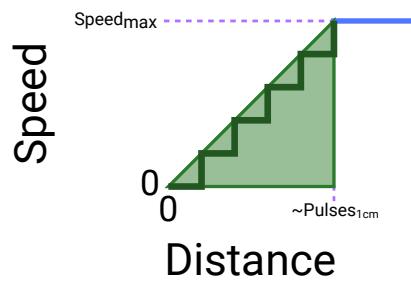


Abbildung 19 - Naive Beschleunigungsrampe - stufiger Effekt



4.3 Probleme und Verbesserungen der naiven Lösung

Ein Problem besteht weiterhin bei diesem Ansatz: Zu Beginn der Bewegung beträgt die Geschwindigkeit 0. Der Motor wird nicht angetrieben und beginnt sich nicht zu bewegen. Wenn die Gleichung von der Zeit abhängen würde, würde die Kurve langsam forschreiten und den Motor starten. Hier ändert sich die Position nicht, wenn sich der Motor nicht bewegt, sodass die Geschwindigkeit bei 0 bleibt, solange X 0 ist.

Um diese Probleme zu vermeiden, wird der Gleichung ein minimaler Offset hinzugefügt, wie in [Abbildung 20](#) gezeigt:

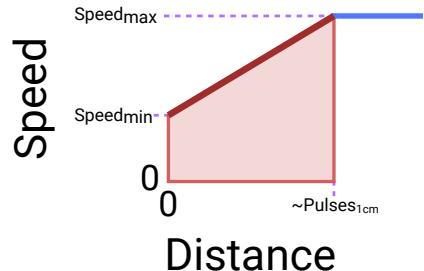


Abbildung 20 - Modifizierte Beschleunigungsrampe

Die Gleichung wird dann:

$$v_x = m * x + b$$

mit

$$m = \frac{\Delta y}{\Delta x} = \frac{\text{speed}_{\text{max}} - \text{speed}_{\text{min}}}{\sim \text{pulses}_{\text{for_1_cm}} - 0}$$

$$b = \text{speed}_{\text{min}}$$

$\text{speed}_{\text{min}}$ hängt davon ab, „wie viel Leistung“ benötigt wird, damit sich der Motor bewegt, beeinflusst durch: die verwendeten Motoren, deren Verschleiß, mechanische Einschränkungen, PWM-Implementierung, Antriebselektronik... was Experimente erfordert.

Für das Beispiel wird $\text{speed}_{\text{min}}$ von $2^6 = 64$ verwendet.

Mit diesen definierten Variablen sind die Steigungsparameter:

$$m = \frac{\text{speed}_{\text{max}} - \text{speed}_{\text{min}}}{\sim \text{pulses}_{\text{for_1_cm}} - 0} = \frac{2^8 - 2^6}{2^{14}} = \frac{192}{16384}$$

Ein solcher m -Wert ist ohne Multiplikatoren/Teiler in der Hardware nicht leicht implementierbar. Es ist möglich, diesen Faktor durch Potenzen von zwei zu approximieren. Um es einfach zu halten, wird die zuvor berechnete gleiche Steigung verwendet. Es wird einfach ein Offset hinzugefügt, um niedrige Geschwindigkeitswerte bei niedrigen X-Positionen auszugleichen.



Solche Approximationen erfordern, dass die Basisannahmen immer neu berechnet werden, sobald alle Parameter festgelegt sind. In diesem Fall, da die Steigung approximiert ist, muss sichergestellt werden, dass die Beschleunigungsrampe nicht „zu schnell“ wird (siehe [Gleichung 1](#)).



4.4 Endgültige Rampe

Die endgültige Gleichung lautet nun:

$$v_x = m * x + b = \frac{x}{2^6} + 2^6 = (x >> 6) + 64$$

Diese Gleichung sollte nur für die Beschleunigungsphase angewendet und dann auf den maximalen PWM-Wert begrenzt werden, sobald dieser erreicht ist.

Beispiel

Angenommen, der Schlitten befindet sich an Position $X = 0$ und soll Punkt P1 erreichen:

- die Beschleunigungsphase erfolgt zwischen $X_{\text{Wagen}} - X_0$, bis die PWM 255 erreicht
- gefolgt von der Phase mit voller Geschwindigkeit, bis $X_{\text{P1}} - X_{\text{Wagen}}$ einen PWM-Wert unter 255 ergibt
- die Verzögerungsphase beginnt, bis die Zielposition erreicht ist, zu welchem Zeitpunkt der Motor gestoppt werden muss



4.5 Überprüfung der Annahmen

Da die vorherige Steigung beibehalten wurde, wird der Motor vor Erreichen der vordefinierten Entfernung mit voller Geschwindigkeit laufen, wie in Abbildung 21 gezeigt (blaue Kurve):

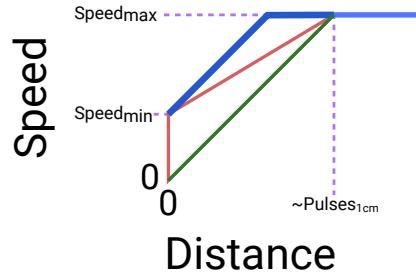


Abbildung 21 - Endgültige Beschleunigungsrampe

Dieser Wert kann wie folgt berechnet werden:

$$x = \frac{v_x - b}{m} = \frac{v_x - b}{\frac{\text{speed}_{\text{max}}}{\sim \text{pulses}_{\text{for } 1 \text{ cm}}}} = \frac{(v_x - b) * (\sim \text{pulses}_{\text{for } 1 \text{ cm}})}{\text{speed}_{\text{max}}} = \frac{(2^8 - 2^6) * 2^{14}}{2^8} = \frac{3'145'728}{256} = 12'288 \text{ pulses}$$

$$\rightarrow 12'288 * 875 \text{ nm} = 1.075 \text{ cm}$$

Gleichung 1 - Endbeschleunigungsstrecke

Der minimale Abstand zwischen zwei Punkten beträgt 4cm (p1 - p2), die Beschleunigungs- + Verzögerungskurven nehmen insgesamt 2.15cm ein, was 1.85cm bei voller Geschwindigkeit übrig lässt. Die Kurven müssen nie „abgeschnitten“ werden, um die Zielpositionen zu erreichen.

Beachten Sie, dass wenn die $\text{speed}_{\text{min}}$ aufgrund der Motoreigenschaften höher eingestellt werden müsste, die Steigung m entsprechend reduziert werden müsste, um weiterhin den Anforderungen an die Rampenlänge von 1cm bis 2cm gerecht zu werden.



Aus zeitlicher Sicht wird die Kurve, da sie von der Position abhängt, nicht linear aussehen. Da niedrige X-Positionen niedrige Motorgeschwindigkeiten haben und daher länger brauchen, um sich zu bewegen als höhere X-Werte, wird die Kurve eher exponentiell wie in Abbildung 22 aussehen.



Abbildung 22 - Motorische Rampe in Abhängigkeit der Zeit



5 | Bewertung

Im Ordner **doc/** zeigt die Datei **evaluation-bewertung-cursor.pdf** das detaillierte Bewertungsschema, [Tabelle 1](#).

Die Schlussnote beinhaltet den Bericht, den Code sowie eine Präsentation eurerseits des Systems.

Evaluierte Aspekte	Punkte
Bericht	55
Einleitung	3
Spezifikation	5
Entwurf	20
Verifizierung und Validation	10
Integration	9
Schlussfolgerung	3
Formale Aspekte des Berichtes	5
Funktionalität der Schaltung	30
Qualität der Lösung	10
Präsentation	10
Total	105

Tabelle 1 - Bewertungsraster



Das Bewertungsraster gibt bereits Hinweise über die Struktur des Berichtes. Für einen guten Bericht konsultieren Sie das Dokument „Wie verfasst man einen Projektbericht?“ [\[14\]](#).



6 | Erste Schritte

Um mit dem Projekt zu beginnen, kann folgendermassen vorgehen werden:

- Lest die obigen Spezifikationen und Informationen genau durch.
- Schaut euch die Hardware und testet das vorprogrammierte Programm.
- Stöbert durch die Dokumente im Ordner **doc/** eures Projektes.
- Entwickelt ein detailliertes Blockdiagramm. Die Signale und deren Funktionen solltet Ihr erklären können.
- Implementierung und Simulation der verschiedenen Blöcken.
- Testen der Lösung auf der Platine und finden etwaiger Fehler .

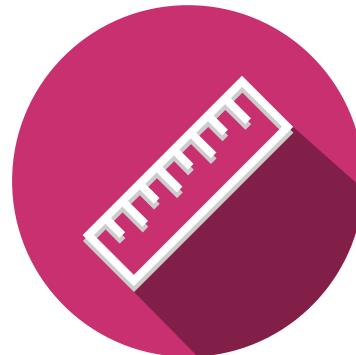
6.1 Tips

Anbei noch einige zusätzlichen Tips um Probleme und Zeitverlust zu vermeiden:

- Teilt das Problem in verschiedene Blöcke auf, benutzt hierzu das leere Toplevel Dokument (**cursor-toplevel-empty.pdf**). Es ist ein ausgeglichener Mix zwischen Anzahl Komponenten und Komponentengrösse empfohlen.
- Analysiert die verschiedenen Ein- sowie Ausgangssignale, hierzu sollten teilweise die Datenblätter zu Hilfe genommen werden.
- Beachtet bei der Erstellung des Systems das DiD Kapitel „Methodologie für die Entwicklung von digitalen Schaltungen (MET)“ [15].
- Es wird empfohlen das System in zwei Schritten zu realisieren.



Vergesst nicht Spass zu haben.





Glossar

CPR – Counts per Revolution [11](#)

FPGA – Field Programmable Gate Array [5, 8, 12, 13](#)

LCD – Liquid Crystal Display [3, 13](#)

LED – Light Emitting Diode [5, 8, 13](#)

PCB – Printed Circuit Board [8](#)

PWM – Pulse Width Modulation [5, 6, 9, 10, 11](#)

UART – Universal Asynchronous Receiver Transmitter [12](#)

USB – Universal Serial Bus [12](#)



Literatur

- [1] „Reed Relay“. 5. Dezember 2020. Zugegriffen: 24. November 2021. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Reed_relay&oldid=992433034
- [2] Olivier Walpen, „Schematic: Cursor Chariot Power Circuit“. 2009.
- [3] STMicroelectronics, „Datasheet: DMOS Dual Full Bridge Driver with PWM Current Controller“. 2003.
- [4] Agilent Technologies, „Datasheet Agilent AEDB-9140 Series Three Channel Optical Incremental Encoder Modules with Codewheel, 100 CPR to 500 CPR“. 2005.
- [5] „Magnetic-Reed-Switch-Above-Closed-and-open-reed-switch-in-response-to-magnet-placement.Png (850×345)“. Zugegriffen: 24. November 2021. [Online]. Verfügbar unter: <https://www.researchgate.net/profile/Sidakpal-Panaich-2/publication/51169357/figure/fig1/AS:394204346896388@1470997048549/Magnetic-reed-switch-Above-Closed-and-open-reed-switch-in-response-to-magnet-placement.png>
- [6] Silvan Zahno, „Schematic: FPGA-EBS v2.2“. 2014.
- [7] Xilinx, „Spartan-3 FPGA Family“. Zugegriffen: 20. November 2021. [Online]. Verfügbar unter: <https://www.xilinx.com/products/silicon-devices/fpga/spartan-3.html>
- [8] Xilinx, „Datasheet Spartan-3E FPGA Family“. 2008.
- [9] CTS, „Datasheet CTS Model CB3 & CB3LV HCMOS/TTL Clock Oscillator“. 2006.
- [10] A. Amand und S. Zahno, „FPGA-EBS3 Electornic Technical Documentation“. 2022.
- [11] Silvan Zahno, „Schematic: Parallelport HEB LCD V2“. 2014.
- [12] Sitronix, „Datasheet Sitronix ST7565R 65x1232 Dot Matrix LCD Controller/Driver“. 2006.
- [13] Electronic Assembly, „Datasheet: DOGM Graphics Series 132x32 Dots“. 2005.
- [14] Christophe Bianchi, François Corthay, und Silvan Zahno, „Wie Verfasst Man Einen Projektbericht?“. 2021.
- [15] François Corthay, Silvan Zahno, und Christophe Bianchi, „Méthodologie de Conception de Circuits Numériques“. 2021.