Hochschule **Rosenheim**
University of Applied Sciences

Plattformunabhängige modellbasierte Entwicklung von hochdynamischen Antriebsregelungen

Julia Höllthaler | Hochschule Rosenheim

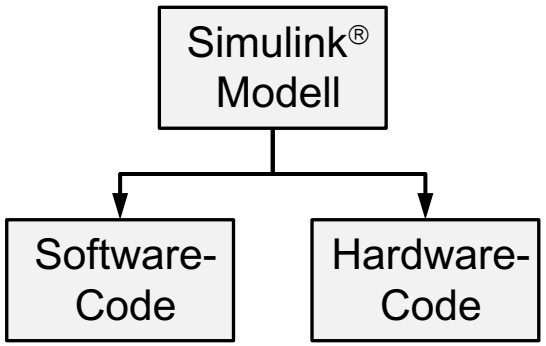
MATLAB Expo 2017

Warum modellbasiert entwickeln?

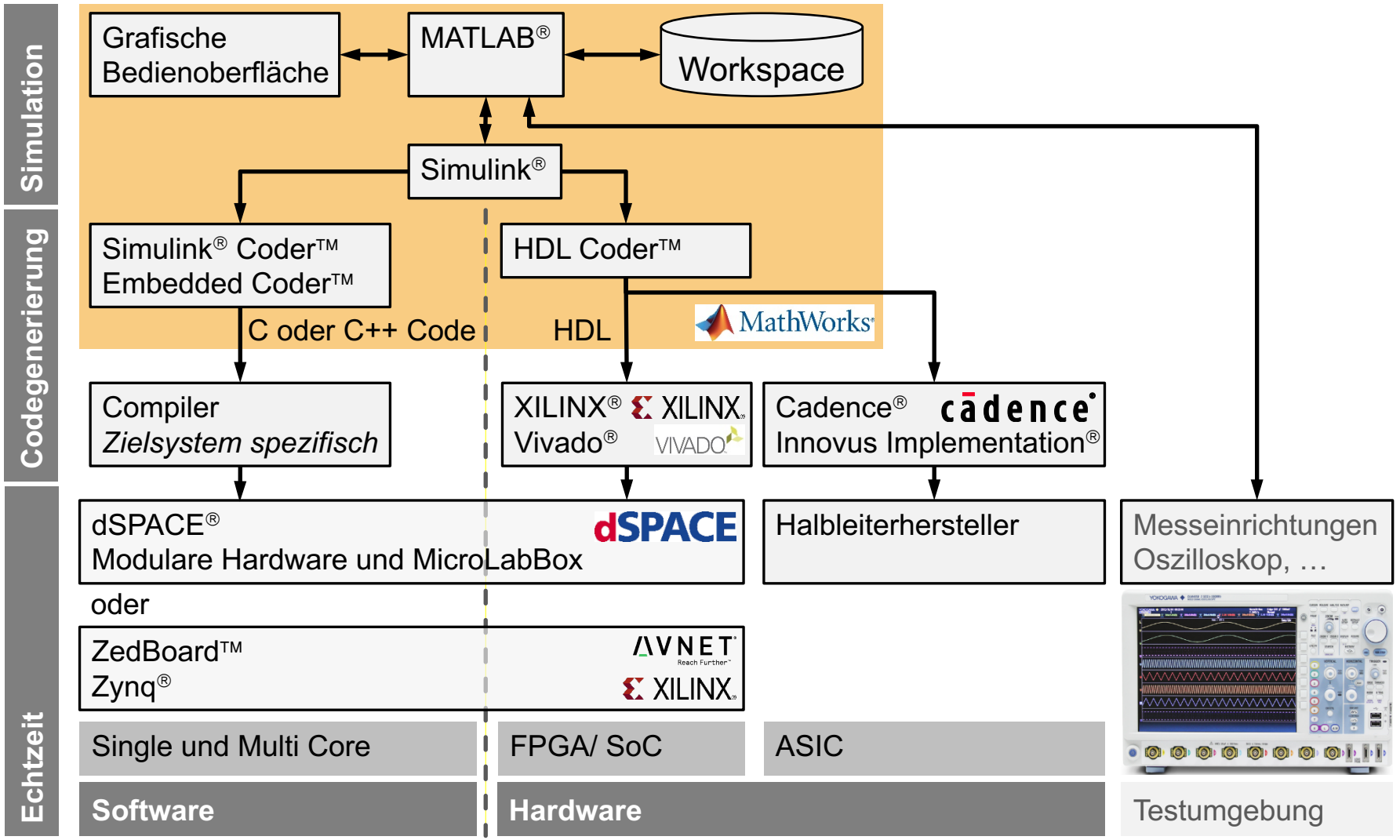
- Modelle besser verständlich als Code
- Kürzere Entwicklungszeiten

Zielsetzung

- Automatische Generierung von C Code und VHDL Code aus einem einzigen Modell
- Einfacher und schneller Wechsel von Software zu Hardware
- Ausarbeitung eines durchgängigen und automatisierten Workflows von Modellbildung und Simulation bis zur Verifikation am realen Prüfstand und Seriencodeerzeugung



Einordnung



HDL Hardware Description Language | **FPGA** Field Programmable Gate Array I **SoC** System on Chip I **ASIC** Application Specific Integrated Circuit

Unterschiede und Einschränkungen

C Code / Embedded Coder™

- Datentyp: Double, (Single, Fixed Point)
- Berechnungen meist in physikalischen Größen
- Niedrige Abtastfrequenzen
  (typisch bei Servoantrieben: 16kHz)
  → Keine Speicherelemente notwendig
  → Keine aufwendigen Anforderungen an Taktzeiten zu berücksichtigen
- Umfangreiche Auswahl von Simulink® Blöcken

VHDL Code / HDL Coder™

- Datentyp: Fixed Point
- Skalierung sinnvoll
- Hohe Abtastfrequenzen
  (typisch bei Servoantrieben: 100kHz bis 200kHz)
  → Speicherelemente für ein synchrones Design notwendig (über Input/Output Pipelines)
  → Funktionen wie Division oder Tangens nicht trivial umsetzbar
- Nur vom HDL Coder™ unterstützte Simulink® Blöcke verwendbar

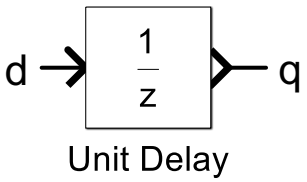
Konsequenzen für eine plattformunabhängige Entwicklung

- Modellanpassungen für automatische HDL Codegenerierung nötig
- Properties der einzelnen Blöcke müssen plattformabhängig geändert werden

Beispiel der automatischen VHDL Codegenerierung anhand eines D-Flipflops

❶ Simulink® Modell

- Beispielhaft einzelner Block
- Unit Delay in Simulink® entspricht einfachem D-Flipflop
- Oft gebraucht für synchrones Design (Speicherelement)



Unit Delay

❷ HDL Coder™

❸ VHDL Code

- Strukturierte Darstellung
- Nachvollziehbare Umsetzung
- Gut verständlich
- Auch eigenhändige Überprüfung möglich

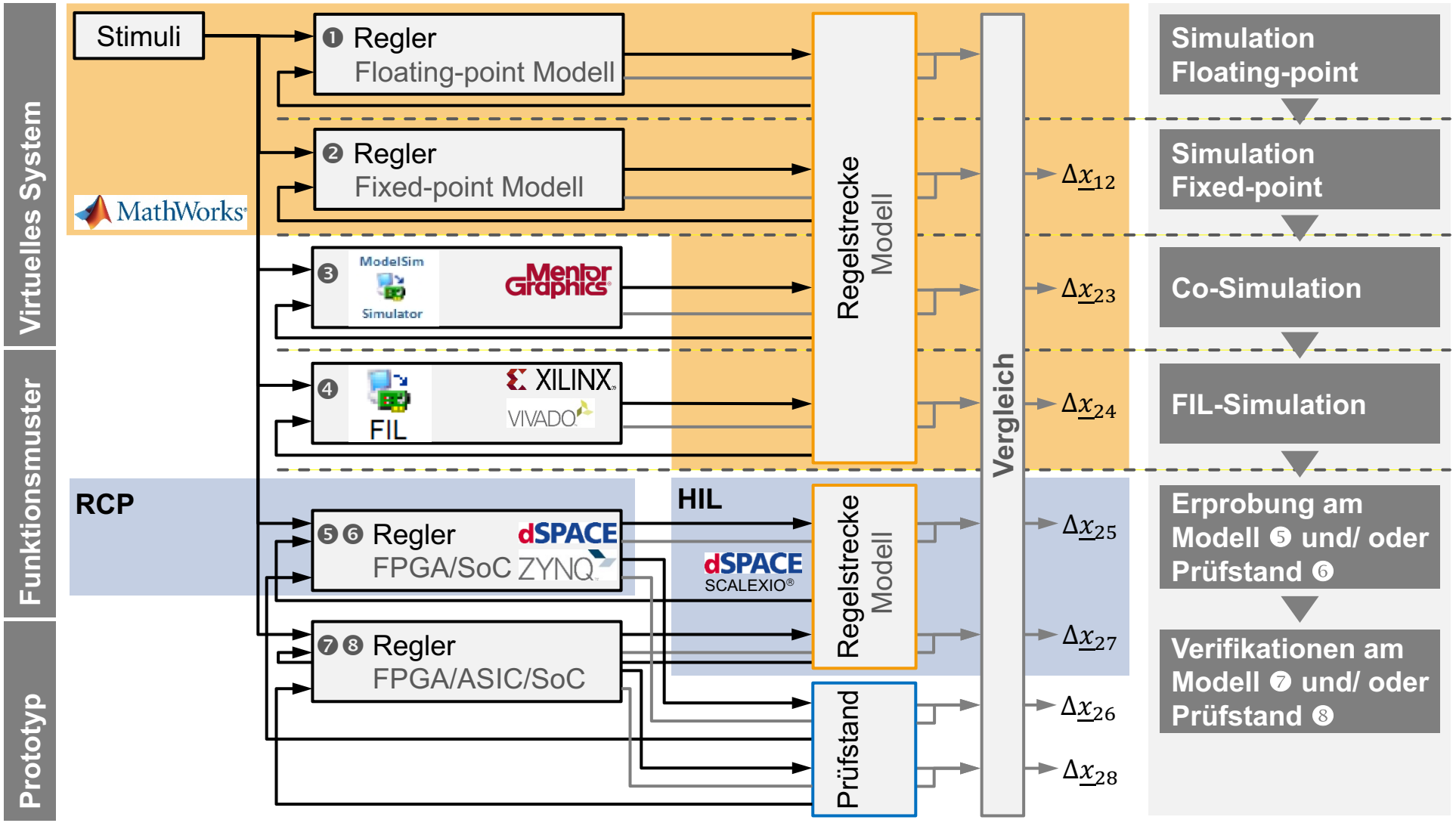
Vom HDL Coder™ erzeugter VHDL Code

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY Subsystem IS
  PORT( clk                               :   IN    std_logic;
        reset                             :   IN    std_logic;
        enb                               :   IN    std_logic;
        d                                 :   IN    std_logic;
        q                                 :   OUT   std_logic
        );
END Subsystem;

ARCHITECTURE rtl OF Subsystem IS
 BEGIN
  -- <S2>/Unit Delay
  Unit_Delay_process : PROCESS (clk, reset)
  BEGIN
    IF reset = '1' THEN
      q <= '0';
    ELSIF clk'EVENT AND clk = '1' THEN
      IF enb = '1' THEN
        q <= d;
      END IF;
    END IF;
  END PROCESS Unit_Delay_process;
END rtl;
```

Arbeitsablauf von der Modellbildung bis zur Verifikation am Prüfstand

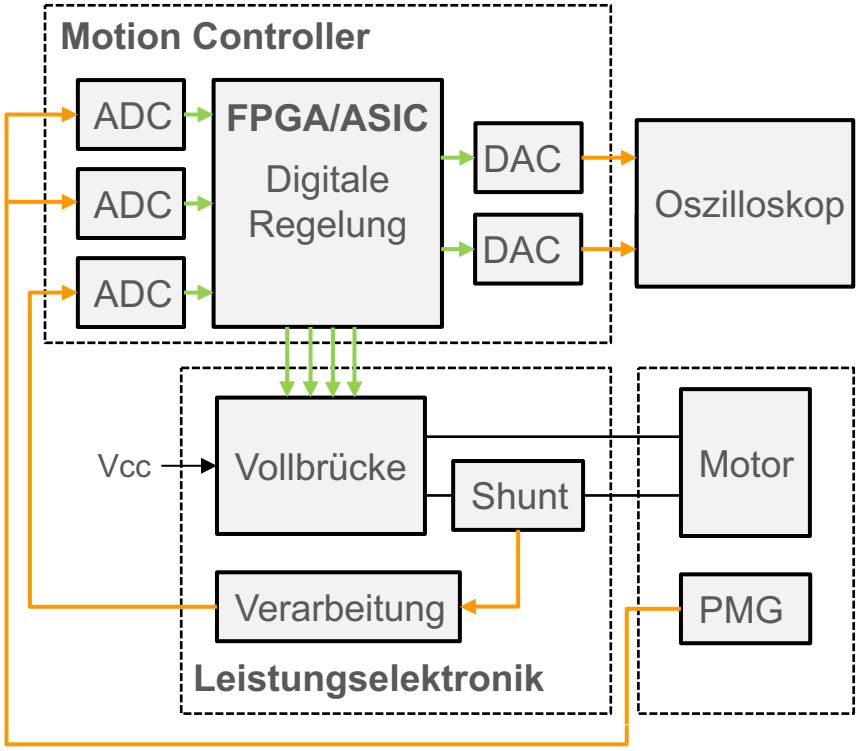


RCP Rapid Control Prototyping I **FIL** FPGA in the Loop I **HIL** Hardware in the Loop
FPGA Field Programmable Gate Array I **ASIC** Application Specific Integrated Circuit I **SoC** System on Chip

Positionsregelung eines Tauchspulenmotors
Übersicht

Hochschule **Rosenheim**
University of Applied Sciences

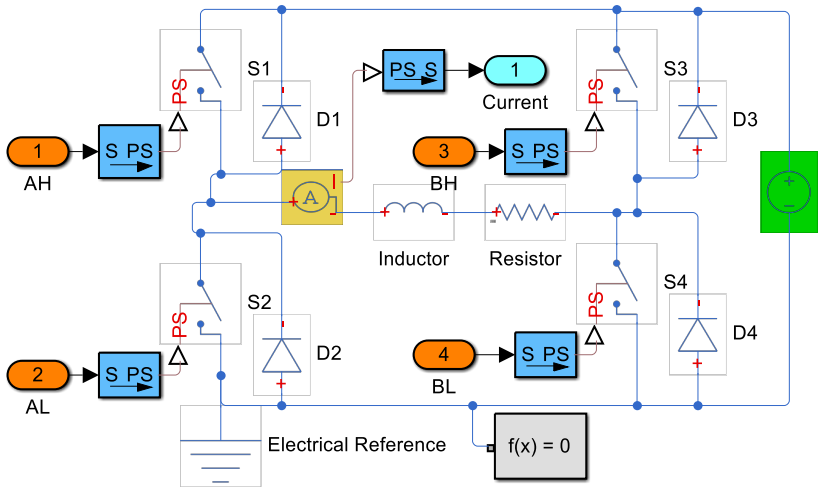
Schematischer Aufbau



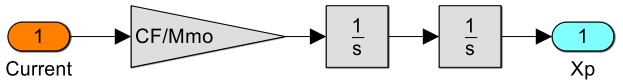
— analoge Signale
— digitale Signale
PMG: Positionsmessgerät

Simulationsmodell Regelstrecke

- **Leistungselektronik und Motor**
  Physikalische Modellierung in Simscape™ Electronics™

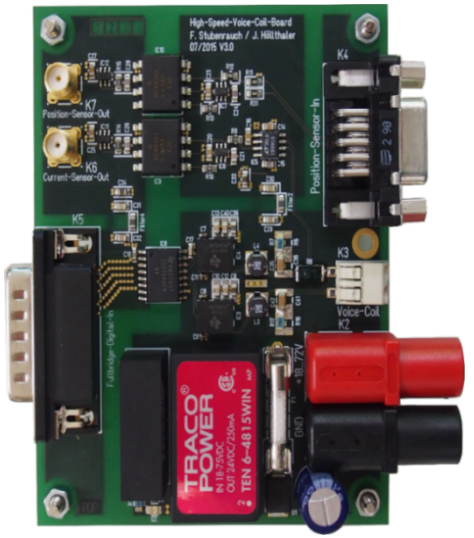


- **Mechanik (Ausschnitt)**
  Blockorientierte Modellierung in Simulink®

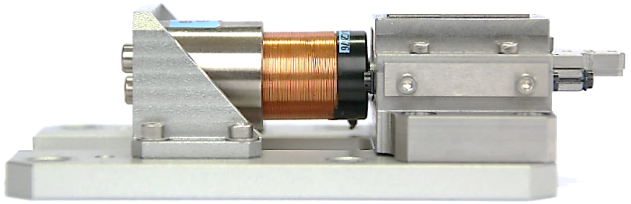


Prüfstand

- **Leistungselektronik**
  100kHz bis 1MHz

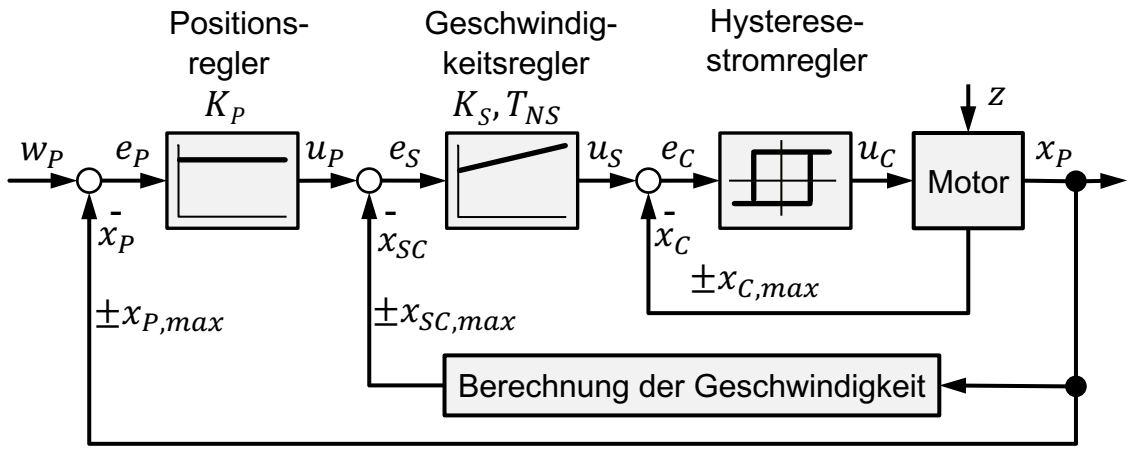


- **Motor und Mechanik**

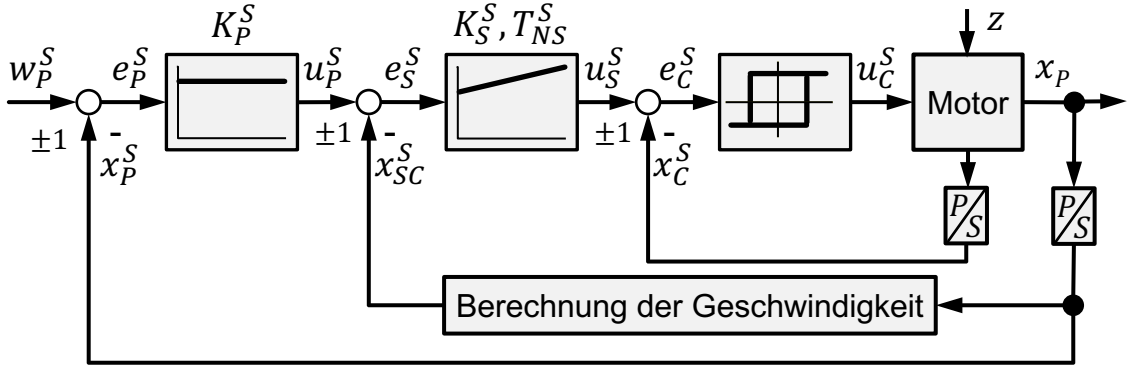


Skalierung des Modells

Physikalisches Modell



Skaliertes Modell



P/S Umrechnung physikalische in skalierte Werte

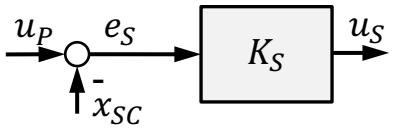
Skalierung

- Für eine flächen- und kosteneffiziente FPGA Umsetzung ist eine möglichst enge Wertebereichsbegrenzung der Größen notwendig.
- Durch eine Skalierung auf die physikalischen Maximalwerte des Antriebssystems ergeben sich Wertebereiche von ±1.
- Mit dieser Skalierung kann die Auflösung über die Bitbreite einfach verändert werden.
- Ein Überlauf am Ausgang eines Multiplikationsblockes kann durch diese Skalierung auftreten und muss durch Sättigung verhindert werden. Dies ist unkritisch, da physikalisch nicht mehr als der Maximalwert möglich ist.

	Istwert	Sollwert	Stellgröße	Regelabweichung
Position	x_P	w_P	u_P	e_P
Geschwindigkeit	x_{SC}	w_S	u_S	e_S
Strom	x_C	w_C	u_C	e_C

Werteskalierung (Proportionalverstärkung Geschwindigkeitsregler)

Physikalisches Modell



$$u_S = K_S \cdot e_S$$

Skalierung auf Maximalwerte

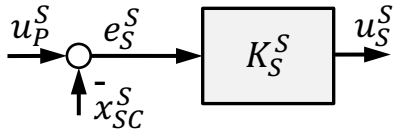
$$u_S^S = \frac{u_S}{x_{C,max}}; \quad e_S^S = \frac{e_S}{x_{SC,max}}$$

Strom x_C
Geschwindigkeit x_{SC}

Berechnung der skalierten Proportionalverstärkung

$$u_S^S \cdot x_{C,max} = K_S \cdot e_S^S \cdot x_{SC,max}$$

Skaliertes Modell

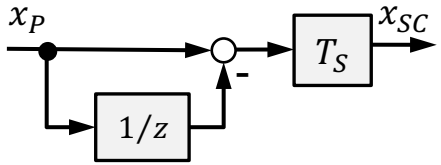


$$u_S^S = \underbrace{\frac{x_{SC,max}}{x_{C,max}} \cdot K_S}_{K_S^S} \cdot e_S^S$$

$$u_S^S = K_S^S \cdot e_S^S$$

Zeitskalierung (Geschwindigkeitsberechnung aus der Position)

Physikalisches Modell



$$x_{SC} = \frac{x_P[k] - x_P[k-1]}{T_S}$$

Skalierung auf Maximalwerte

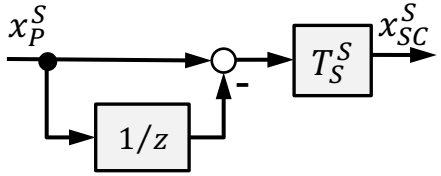
$$x_P^S = \frac{x_P}{x_{P,max}}; \quad x_{SC}^S = \frac{x_{SC}}{x_{SC,max}}$$

Position x_P
Geschwindigkeit x_{SC}

Berechnung der skalierten Geschwindigkeit

$$x_{SC}^S \cdot x_{SC,max} = \frac{(x_P^S[k] - x_P^S[k-1]) \cdot x_{P,max}}{T_S}$$

Skaliertes Modell



$$x_{SC}^S = \underbrace{\frac{x_{P,max}}{x_{SC,max}} \cdot \frac{1}{T_S}}_{T_S^S} \cdot (x_P^S[k] - x_P^S[k-1])$$

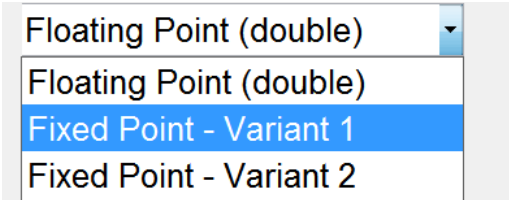
$$x_{SC}^S = \frac{x_P^S[k] - x_P^S[k-1]}{T_S^S}$$

Festlegung verschiedener Datentypen in einem Modell

❶ Datentypdefinition für alle Block Outputs (Zentral)

```
Controller.OutputDataType.Multiply.GainSpeed = ...
 {'double'; ...
  'fixdt(1,25,24)'; ...
  'Inherit: Inherit via back propagation'};
```

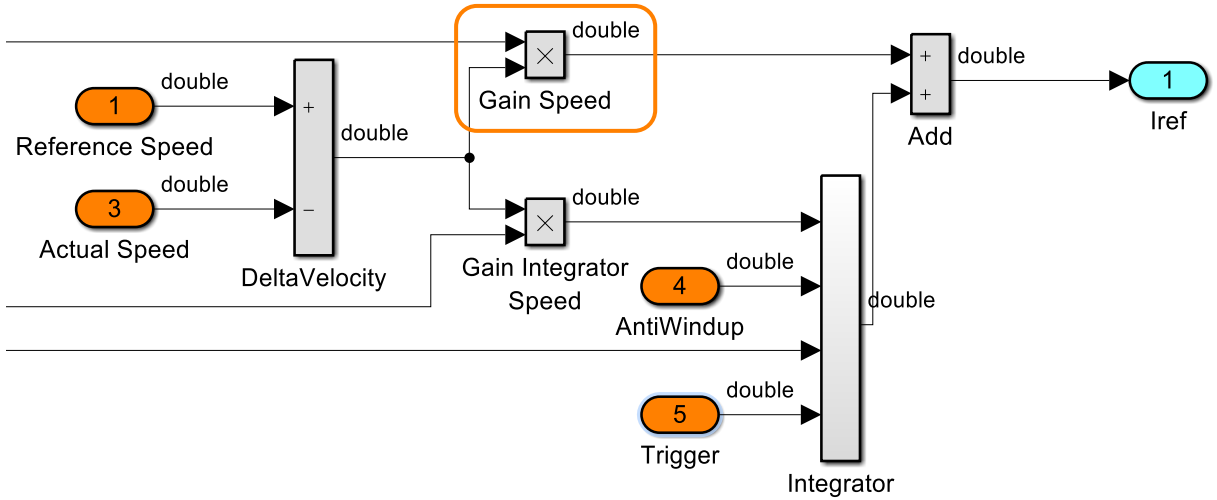
❷ Auswahl des Datentyp-Modells in GUI



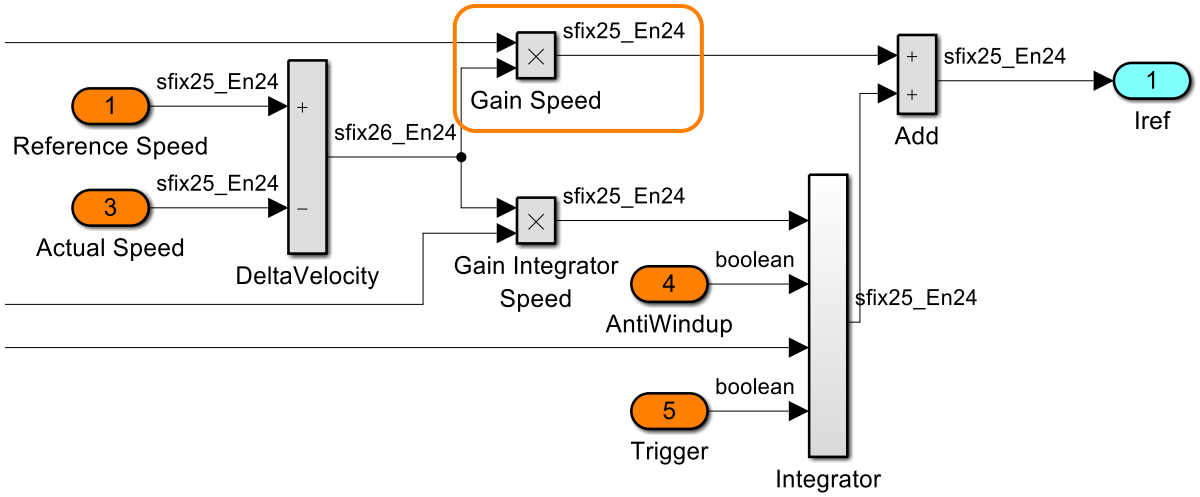
❸ Programmatische Festlegung des Datentyps im Modell für alle Blöcke

```
set_param([Subsystem.BlockName 'Cascade Controller/Speed Controller/Gain Speed'],...
'OutDataTypeStr',char(Controller.OutputDataType.Multiply.GainSpeed(DataTypeNumber)),...
'SaturateOnIntegerOverflow','On');
```

Datentyp-Modell mit Floating-Point Datentypen (Ausschnitt)



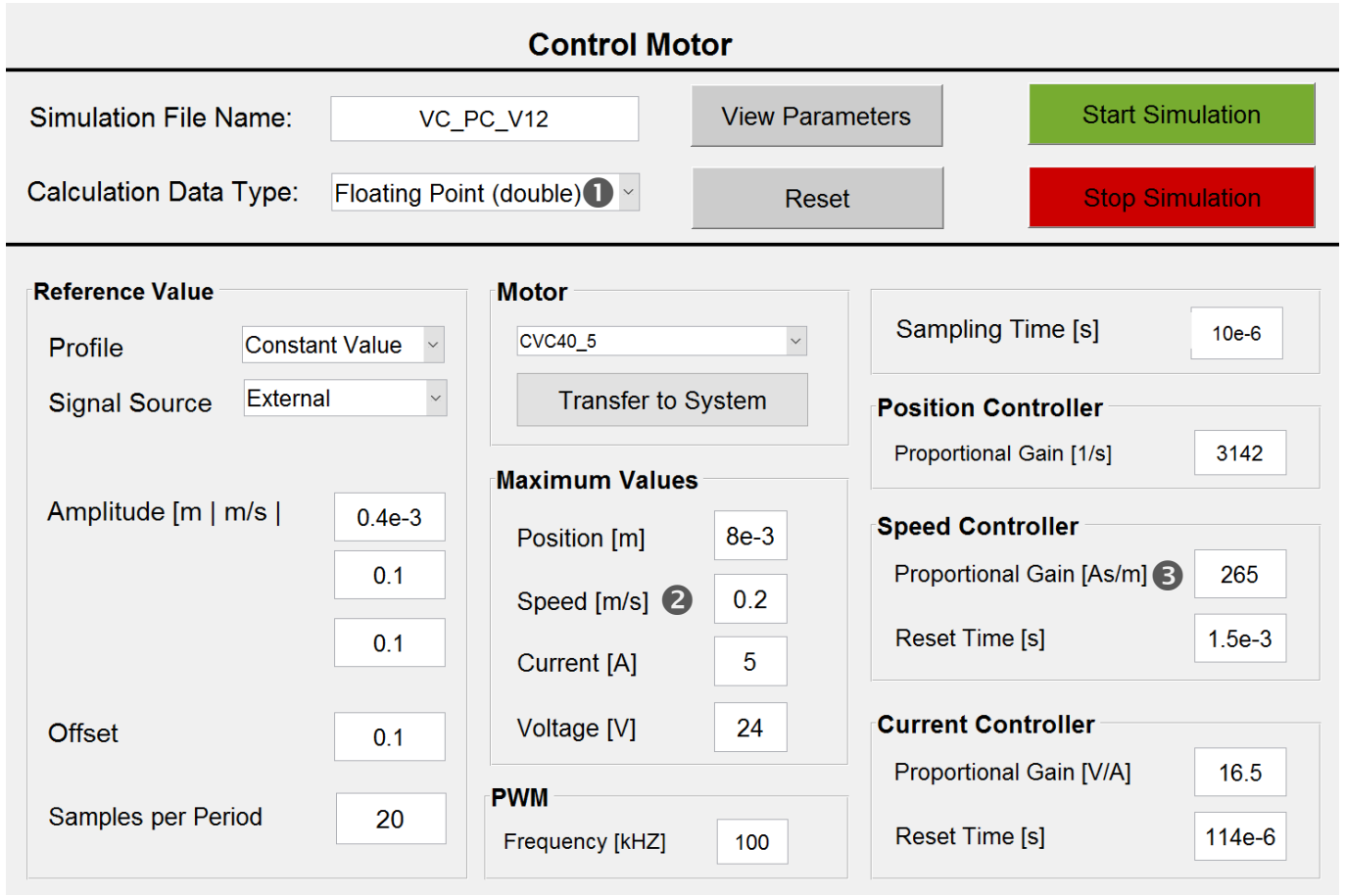
Datentyp-Modell mit Fixed-Point Datentypen (Ausschnitt)



Auszug aus der grafischen Bedienoberfläche zur Modellerstellung und Parametrierung

Hochschule **Rosenheim**
University of Applied Sciences

Entwicklung der grafischen Bedienoberfläche mit MATLAB® GUIDE



❶ **Auswahl des Datentyps**

❷ **Eingabe der Maximalwerte des jeweiligen Antriebssystems**

- Maximale Geschwindigkeit $x_{sc,max} = 0.2\,\frac{m}{s}$
- Maximaler Strom $x_{c,max} = 5A$
- …

❸ **Festlegung der Reglerparameter**

- Eingabe der physikalischen Werte
- Automatische Berechnung der skalierten Werte in der Callback Function der grafischen Bedienoberfläche

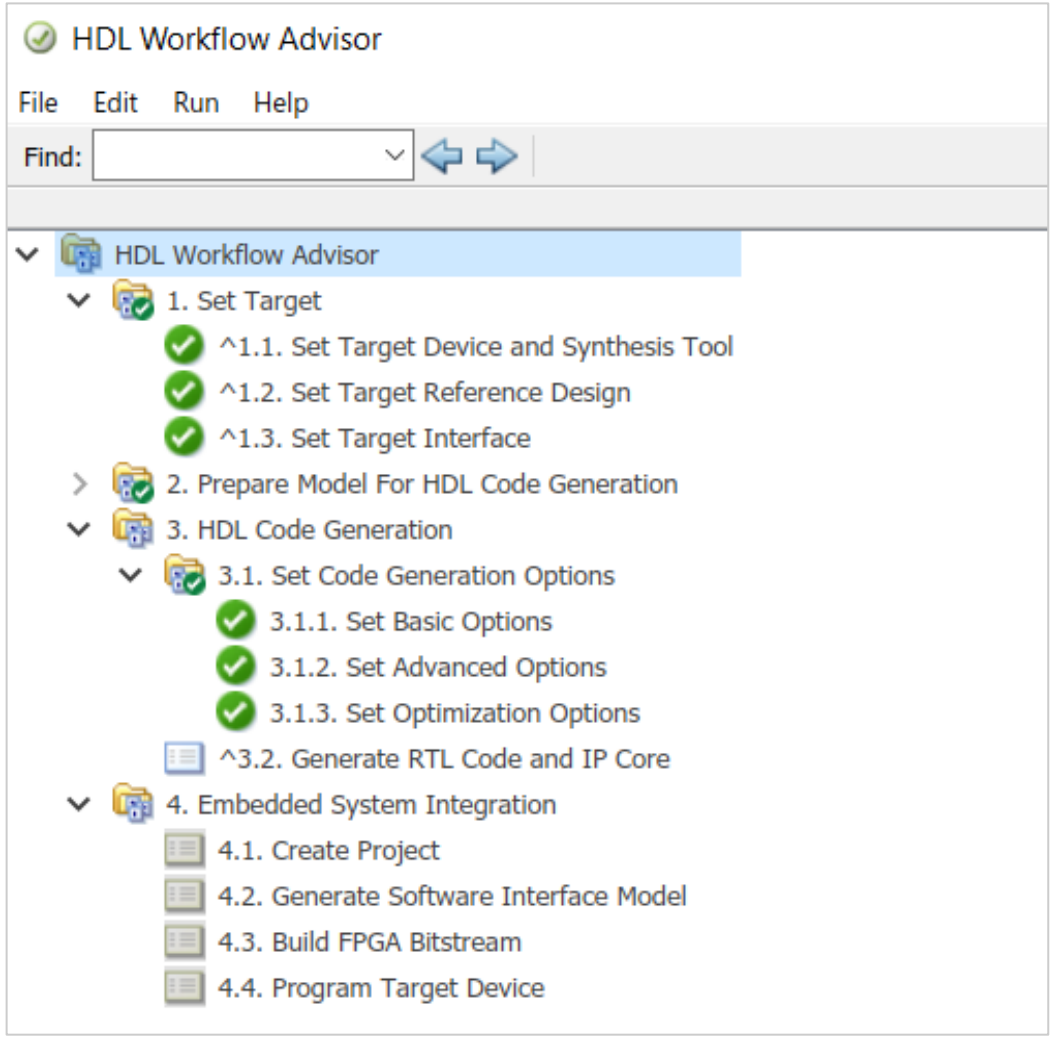
  z.B. Proportionalverstärkung Geschwindigkeitsregler:

$$u_S^S = \frac{x_{SC,max}}{x_{C,max}} \cdot K_S \cdot e_S^S = K_S^S \cdot e_S^S$$

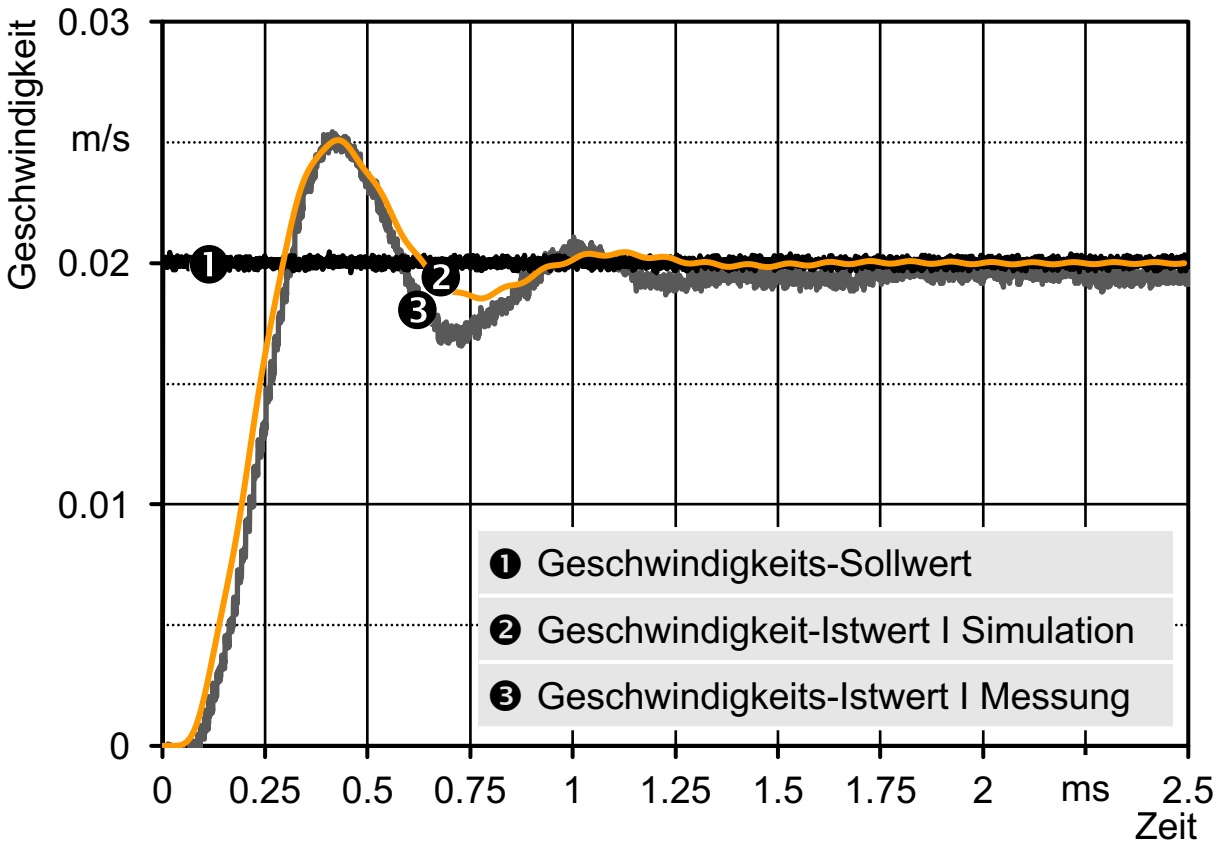
$$K_S^S = \frac{0{,}2\frac{m}{s}}{5A} \cdot K_S = 0{,}04\,\frac{m}{As} \cdot 265\,\frac{As}{m} = 10{,}6$$

→ Übergabe von K_S^S an das FPGA-Modell

Automatische Code-Erzeugung mit dem HDL Workflow Advisor



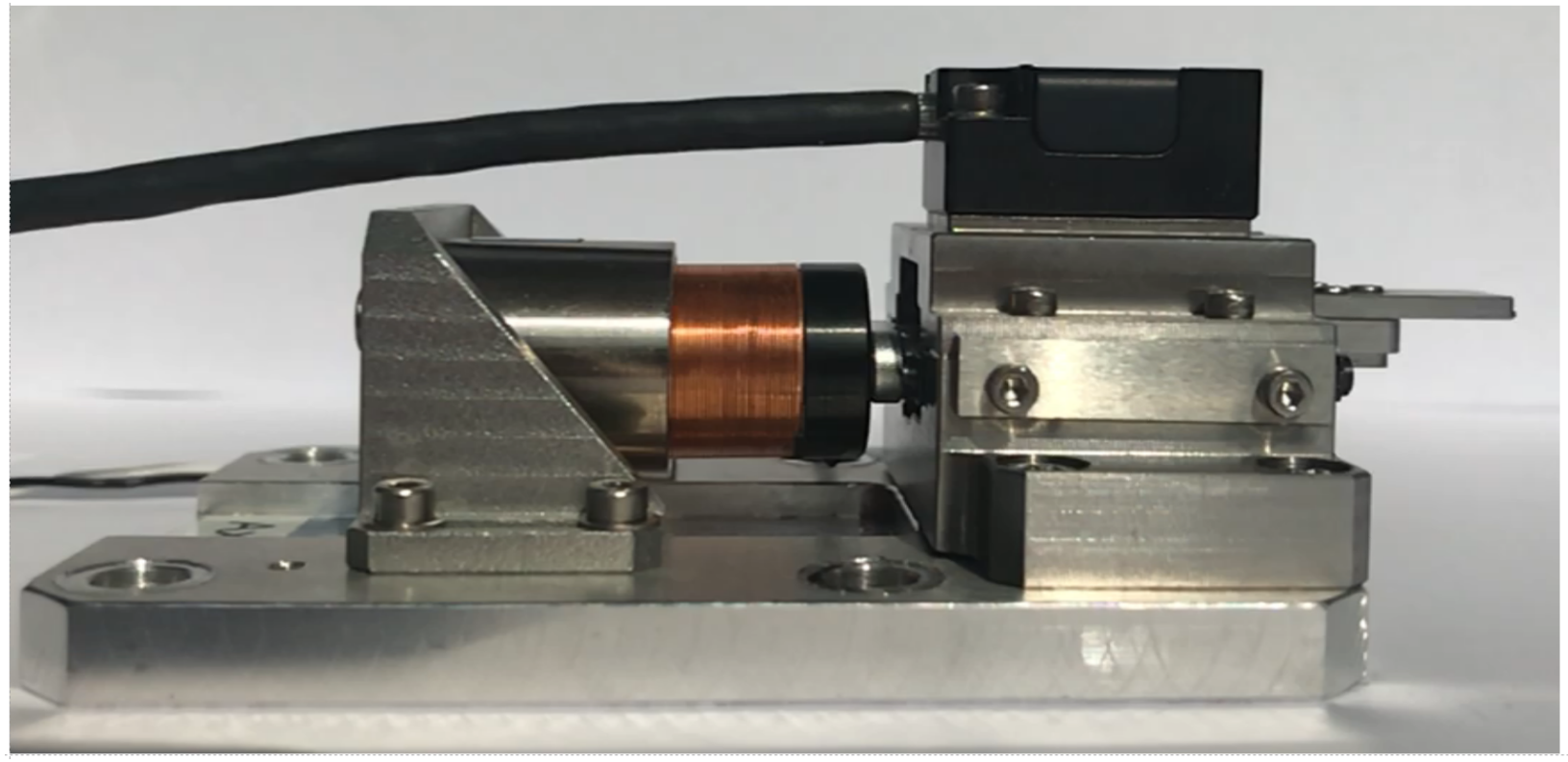
Ergebnisse



➔ Simulationsmodell stimmt sehr gut mit dem realen Verhalten überein

Verbesserungen durch FPGA Implementierung

- Erhöhung der PWM-Frequenz auf ≥ 200kHz
- Deutliche Erhöhung der Bandbreite des Stromregelkreises von ca. 1kHz auf 10kHz
  → Steigerung der Bandbreite des Positions- und Geschwindigkeitsregelkreises
- Verbesserung der Gleichlaufkonstanz
- Verbesserung der Positionsstabilität



Aktueller Stand

- Automatische Generierung von C Code und VHDL Code aus einem einzigen Modell ist möglich
- Bestätigung der Funktionsfähigkeit des Workflows

Ausblick

- Automatisierung der hardwarespezifischen Modellergänzungen für die Schnittstellen zum Prozess
- Prüfen der Verwendbarkeit des generierten HDL Codes für die ASIC Entwicklung