Praxisgerechte Umsetzung der Unschärferelation: Steuern von Schrittmotoren mittels FPGA-basierter SPS ohne zusätzlichen Motion-Controller

Alfons Austerhoff
07.03.2016
H. Zander GmbH & Co. KG

Steuern von Schrittmotoren mit FPGA-basierter SPS

ZANDER AACHEN

SAFETY

AUTOMATION

ENGINEERING

AUTOMATION

Grundlegendes zu Schrittmotoren

Sonderform der Synchronmaschine

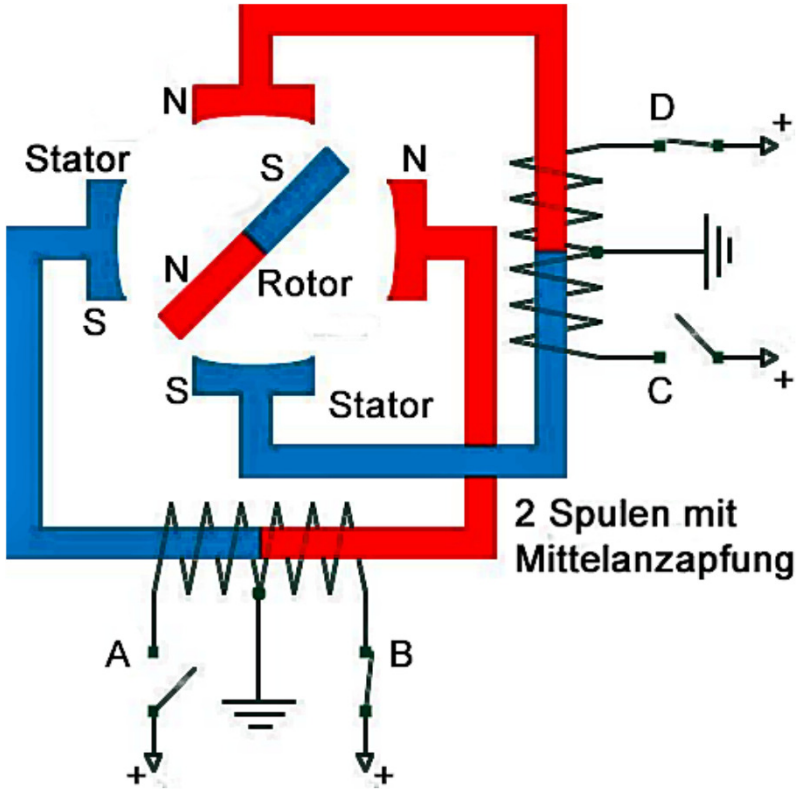
→ Rotor bewegt sich synchron zum Stator-Drehfeld, aber quasi „digital“, d.h. in Schritten mit definierter Schrittweite.

Vorteile:

- einfache Ansteuerung
- genaues Positionieren ohne Rückkopplung/Messung

Steuern von Schrittmotoren mit FPGA-basierter SPS

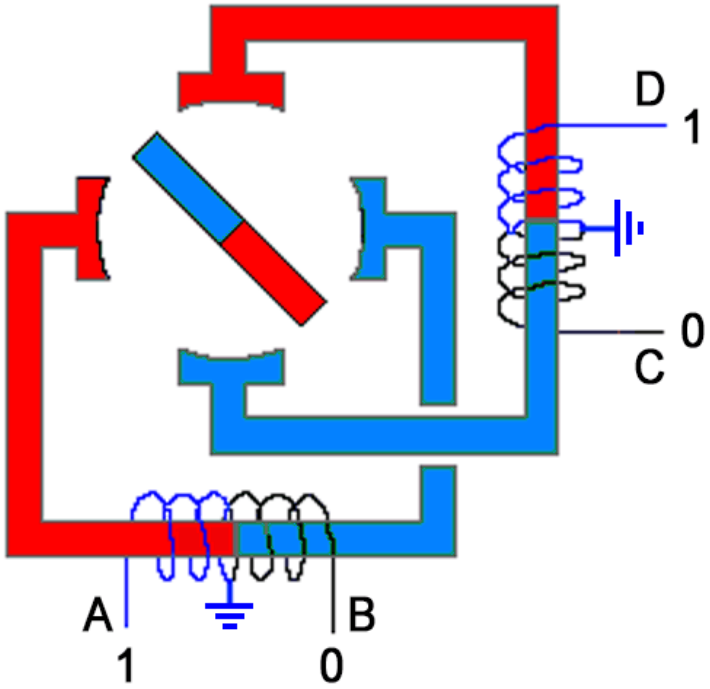
Aufbau und Funktionsprinzip (unipolarer Schrittmotor):



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

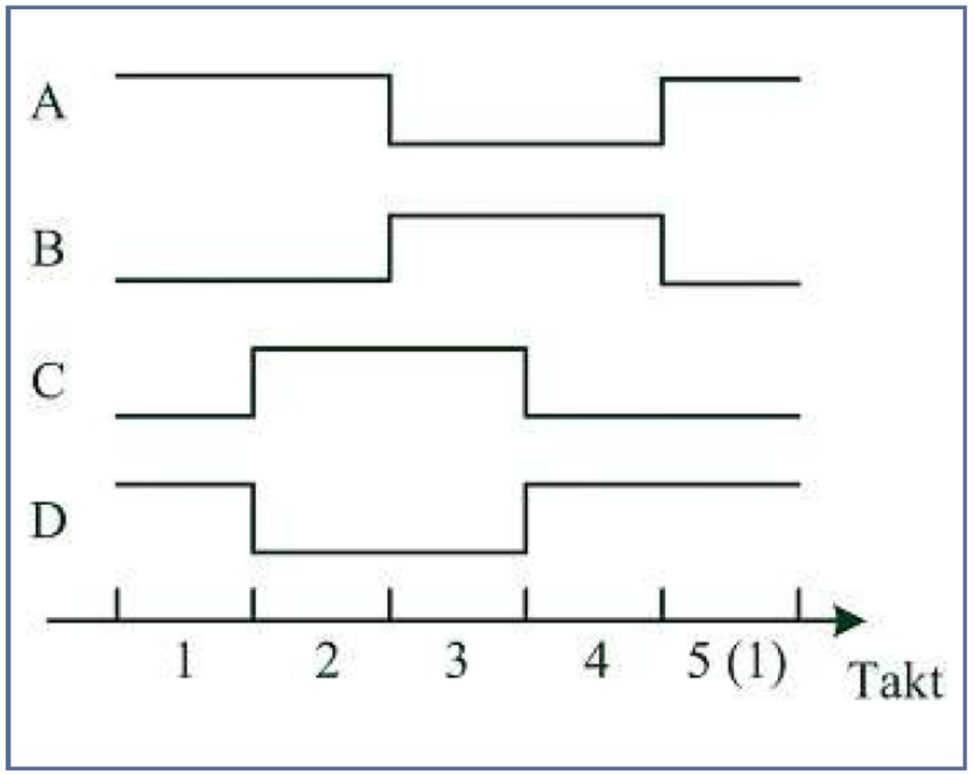
Ansteuerung der Spulen:



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

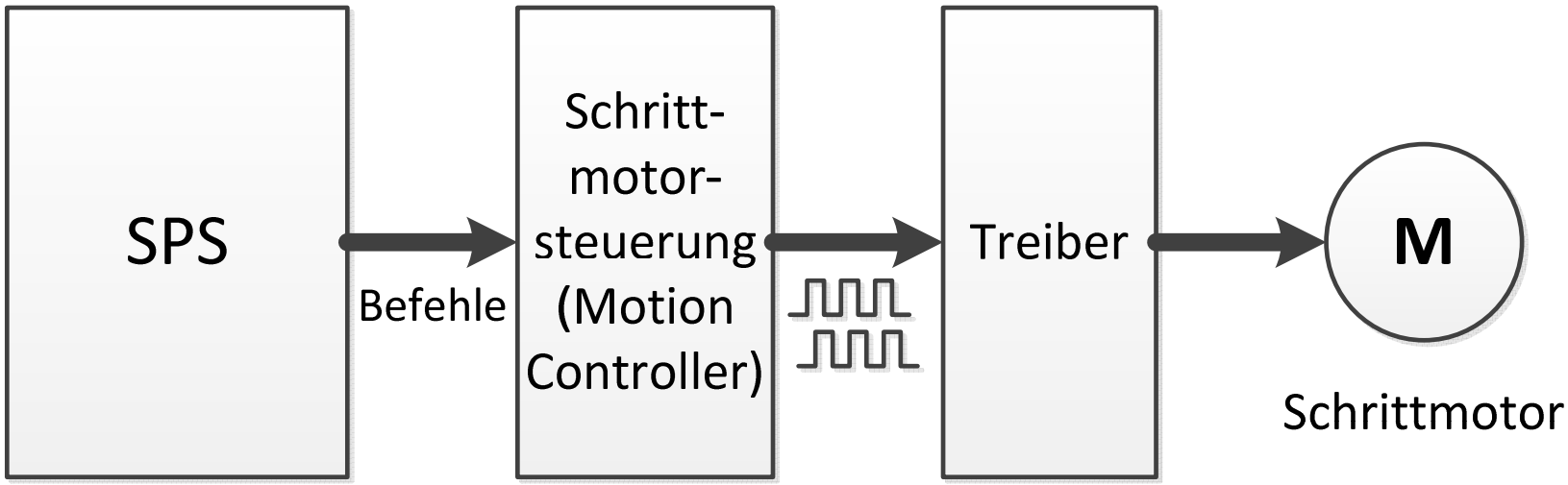
Ansteuerung der Spulen:



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

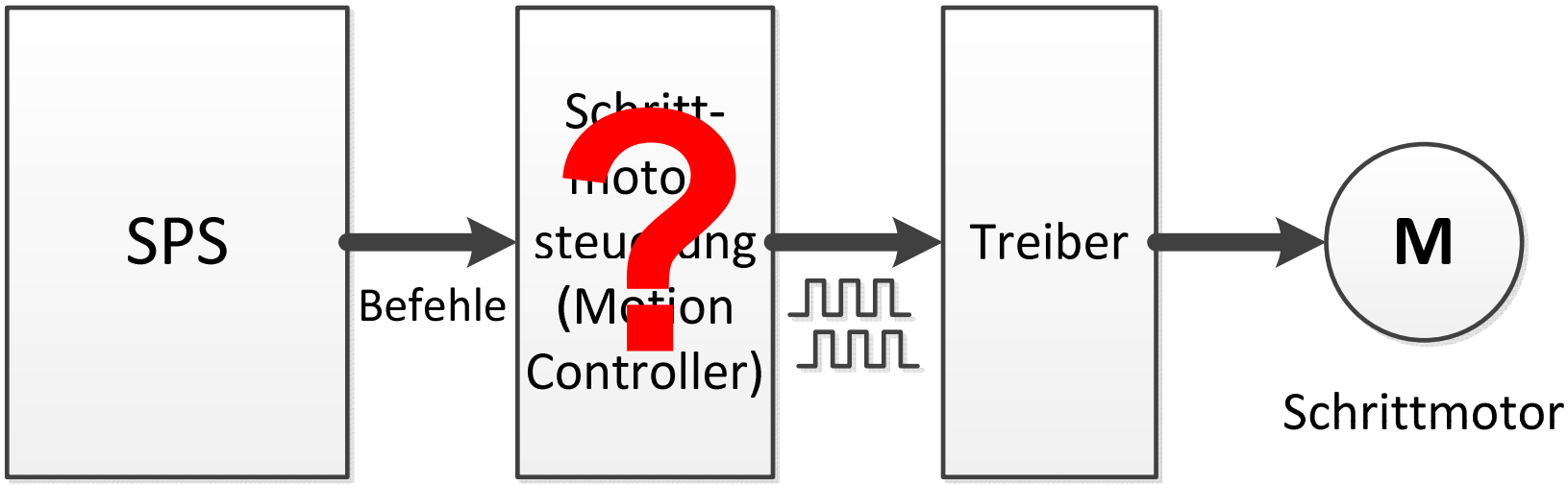
Steuerung mit konventioneller SPS:



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

Steuerung mit konventioneller SPS:
Benötigt man zwingend den Motion Controller?



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

Steuerung mit konventioneller SPS:
Benötigt man zwingend den Motion Controller?

Vorteile, wenn man auf den Motion Controller verzichten könnte:

- Es muss nur die SPS programmiert werden (sonst zusätzlich Konfiguration/Parametrierung des Motion Controllers)
- Individuelle Beschleunigungs- und Abbremsrampen (ungleichmäßige Beschleunigung in verschiedenen Drehzahlbereichen)
- Einsparen von Verdrahtungsaufwand

AUTOMATION

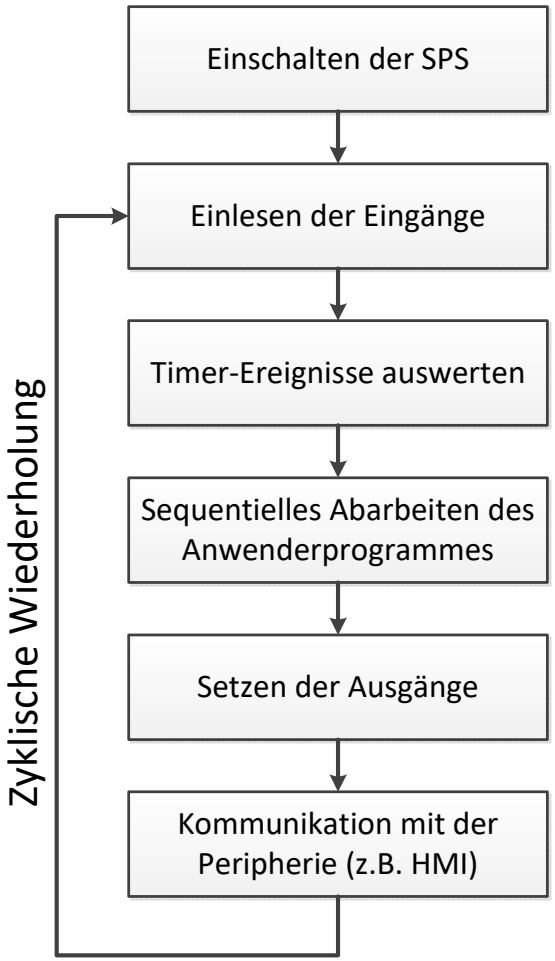
Steuern von Schrittmotoren mit FPGA-basierter SPS

Steuerung mit konventioneller SPS:

Zyklische Abarbeitung des Anwenderprogramms

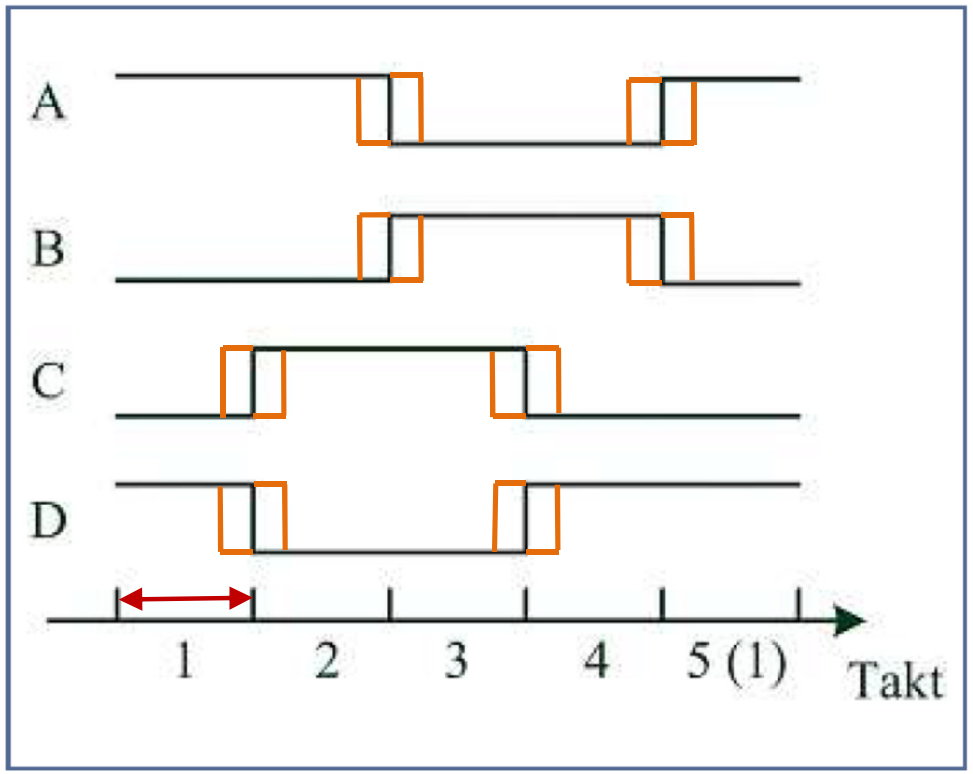
Zykluszeit (abhängig von der Größe des Anwenderprogramms)

Jitter (= unpräzise Reaktionszeiten)



AUTOMATION

Ansteuerung der Spulen:



Jitter

Minimale Taktzeit = Zykluszeit

Steuern von Schrittmotoren mit FPGA-basierter SPS

Steuerung mit konventioneller SPS:

Beispiel: Zykluszeit = 10ms

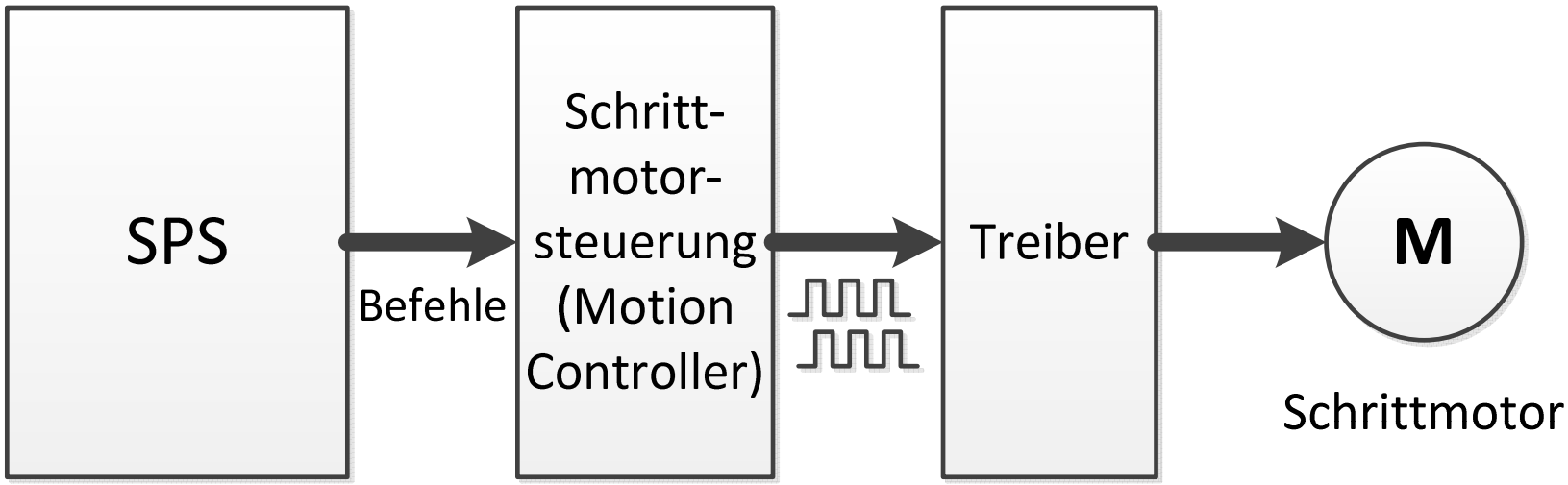
Schrittmotor mit 200 Schritten pro Umdrehung
→ 2 Sekunden für 1 Umdrehung

Maximal erreichbare Geschwindigkeit des Motors:
30 U/min

AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

Steuerung mit konventioneller SPS:
Benötigt man zwingend den Motion Controller?

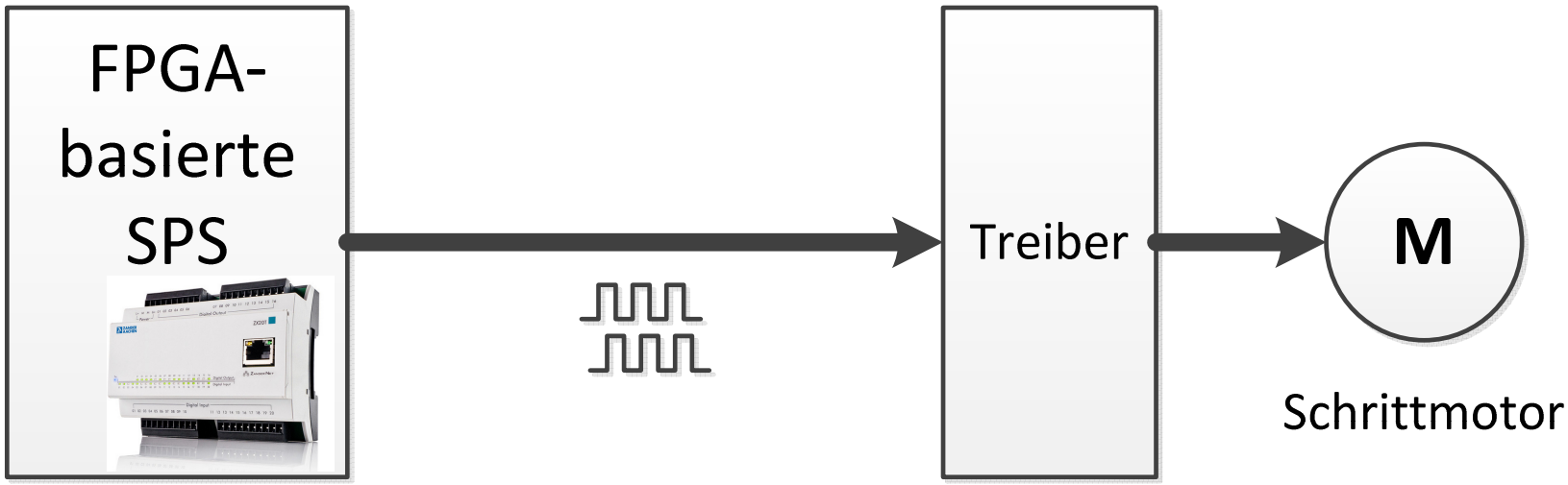


Antwort: Ja!

AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

Alternative: FPGA-basierte SPS



AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

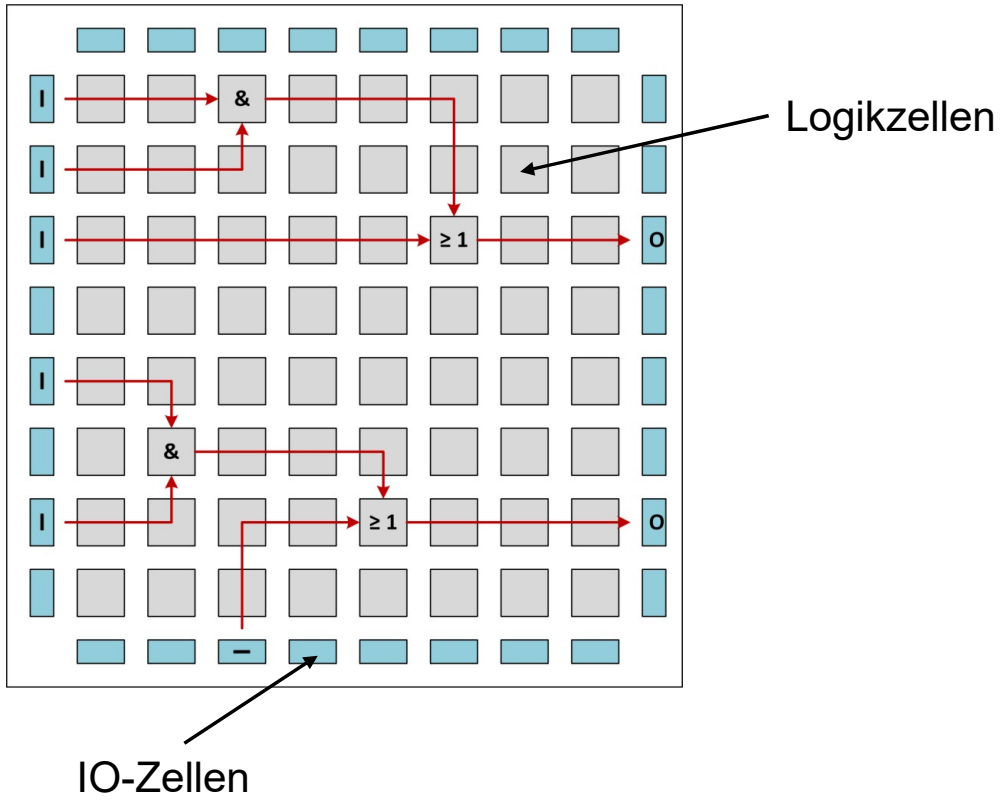
FPGA (vereinfacht dargestellt)

Anwenderprogramm
wird in Hardware umgesetzt

Echte Parallelverarbeitung

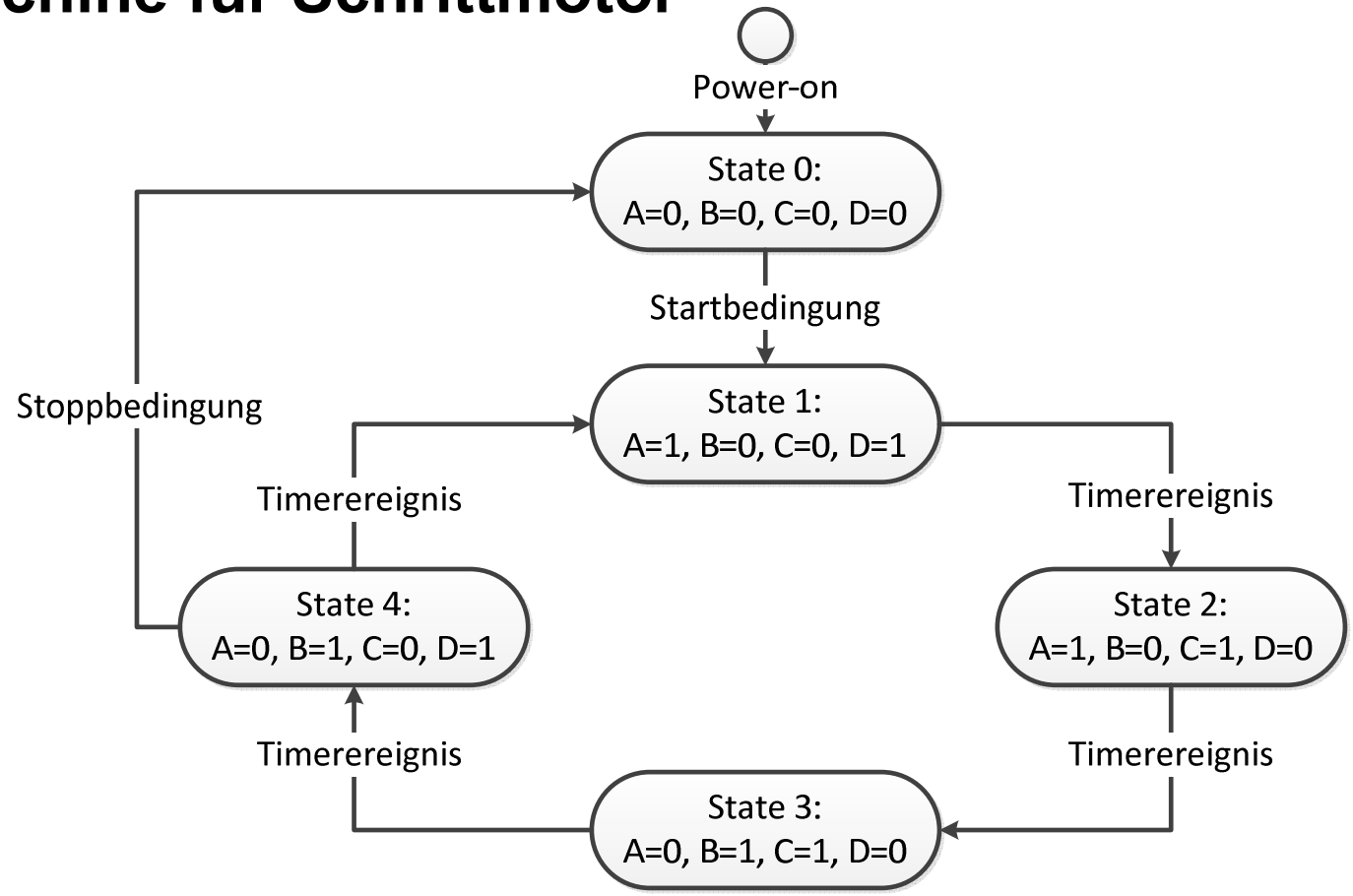
Extrem kurze Reaktions-
zeiten auf Ereignisse

Kein Jitter: Absolut konstante
Reaktionszeiten



AUTOMATION

Statemachine für Schrittmotor



Realisierung der FPGA-Konfiguration

Konfiguration von FPGA üblicherweise mit Hardwarebeschreibungssprachen wie VHDL oder Verilog

Problem: SPS-Programmierer haben diese Sprachen in der Regel nicht erlernt.

Lösung: Konfiguration eines FPGA mit SPS-Programmiersprache nach IEC 61131-3.

Programmiersystem EX_PRESS 5

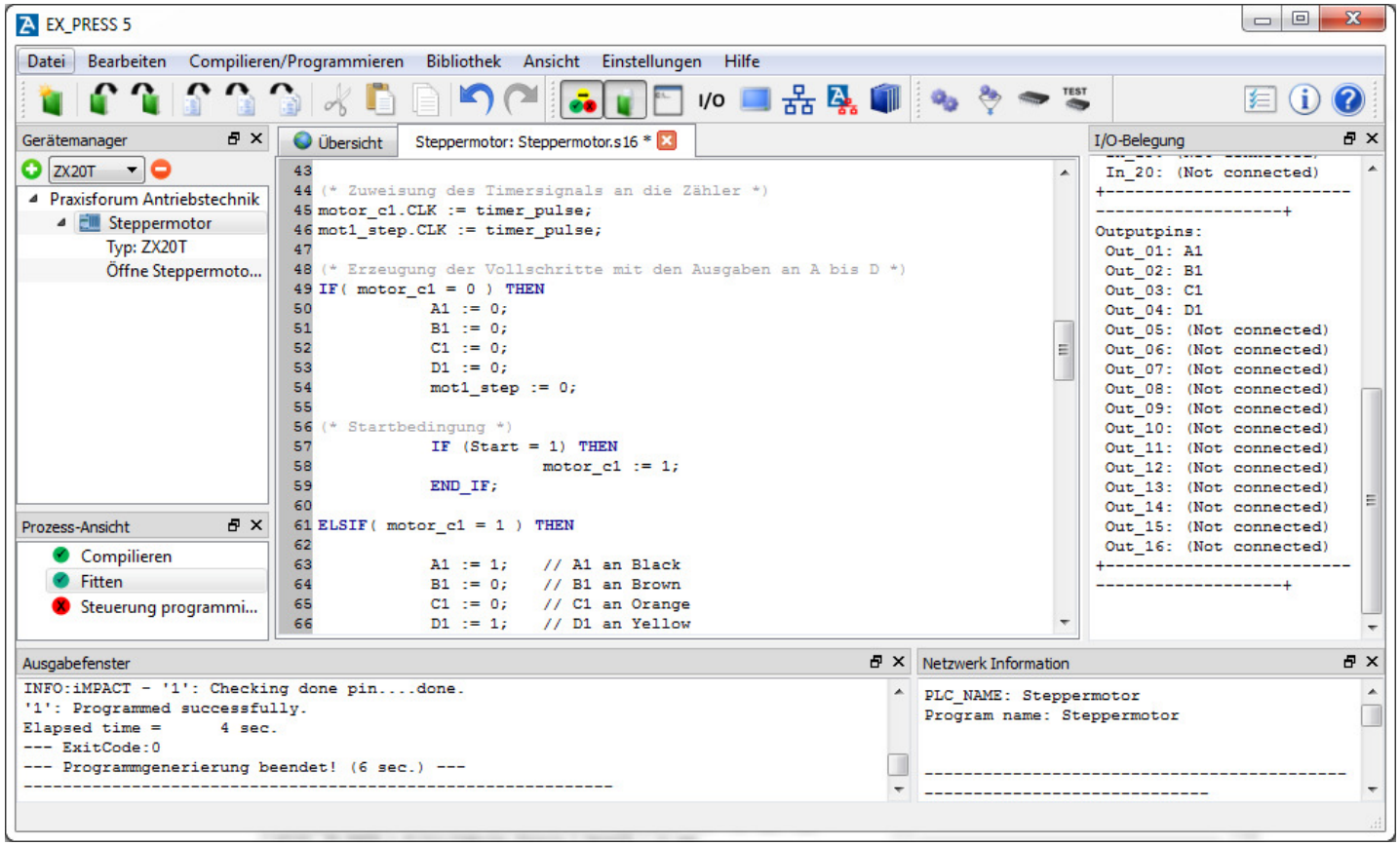
Umsetzer „Strukturierter Text“ in Hardwarebeschreibung

Der SPS-Programmierer kann wie gewohnt seine Anwendung erstellen, muss aber folgendes beachten:

- Alle Anweisungen werden parallel verarbeitet.
- Single-Assignment: Einer Variablen darf nur an einer Stelle im Programm ein Wert zugewiesen werden.
- Bei einer Wertzuweisung an eine nicht getaktete Variable dürfen keine zyklischen Abhängigkeiten geschaffen werden (z.B. A := NOT A;)

Steuern von Schrittmotoren mit FPGA-basierter SPS

Programmiersystem EX_PRESS 5



AUTOMATION

Das ST-Programm, Deklaration der Variablen:

```
VAR_INPUT
        Start AT In_01;         (* Start-Taster *)
END_VAR;

VAR_OUTPUT
        A1 AT Out_01;           (* Anschluesse fuer den Schrittmotor *)
        B1 AT Out_02;
        C1 AT Out_03;
        D1 AT Out_04;
END_VAR;
```

Das ST-Programm, Deklaration der Variablen:

```
VAR
        motor_c1: USINT := 0;   (* Zähler für Statemachine *)
        mot1_step: UINT := 0;   (* Schrittzaehler fuer jeweils einen Vollschritt *)
END_VAR;


VAR_TIMER
        timer_pulse;            (* Timer für die Pulse an den Ausgängen *)
END_VAR;
```

Das ST-Programm, Anweisungsteil:

```
Start.TDB := 20ms;     (* Entprellung des Start-Tasters *)



(* Zuweisung des Timersignals an die Zähler *)

motor_c1.CLK  := timer_pulse;
mot1_step.CLK := timer_pulse;
```

AUTOMATION

Das ST-Programm, Anweisungsteil:

```
(* Statemachine: Erzeugung der Vollschritte mit den Ausgaben an A1 bis D1 *)

(* State 0 *)
IF( motor_c1 = 0 ) THEN
        A1 := 0;
        B1 := 0;
        C1 := 0;
        D1 := 0;
        mot1_step := 0;


        (* Startbedingung *)
        IF (Start = 1) THEN
                motor_c1 := 1;
        END_IF;
```

Das ST-Programm, Anweisungsteil:

```
(* State 1 *)
ELSIF( motor_c1 = 1 ) THEN
        A1 := 1; // A1 an Black
        B1 := 0; // B1 an Brown
        C1 := 0; // C1 an Orange
        D1 := 1; // D1 an Yellow
        motor_c1 := 2; // Start des nächsten State
        mot1_step := mot1_step + 1;


(* State 2 *)
        ELSIF( motor_c1 = 2 ) THEN
        A1 := 1;
        B1 := 0;
        C1 := 1;
        D1 := 0;
        motor_c1 := 3;
        mot1_step := mot1_step + 1;
```

Das ST-Programm, Anweisungsteil:

```
(* State 3 *)
ELSIF( motor_c1 = 3 ) THEN
        A1 := 0;
        B1 := 1;
        C1 := 1;
        D1 := 0;
        motor_c1 := 4;
        mot1_step := mot1_step + 1;
```

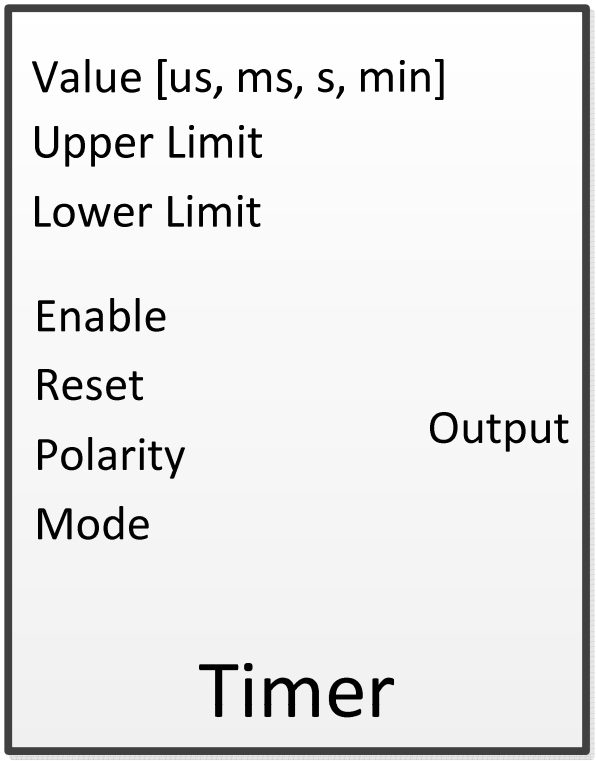
Das ST-Programm, Anweisungsteil:

```
(* State 4 *)
ELSIF( motor_c1 = 4 ) THEN
        A1 := 0;
        B1 := 1;
        C1 := 0;
        D1 := 1;

(* Bedingung für Fortsetzen bzw. Beenden der Motorbewegung *)
        IF( mot1_step < 600 ) THEN
                motor_c1 := 1;
                mot1_step := mot1_step + 1;
        ELSE            (* Stoppbedingung erfüllt *)
                motor_c1 := 0;
                mot1_step := 0;
        END_IF;

END_IF;
```

Timer


Value [us, ms, s, min]
Upper Limit
Lower Limit
Enable
Reset
Polarity
Mode
Output
Timer

Das ST-Programm, Anweisungsteil:

```
(* Die Konfiguration des Timers *)

timer_pulse := 5000 us;

timer_pulse.ENABLE := 1;
timer_pulse.RESET  := 0;
timer_pulse.POL    := HIGH;
timer_pulse.MODE   := CONT_CONF;


timer_pulse.UPPER_LIMIT := 5000 us;
timer_pulse.LOWER_LIMIT := 1000 us;
```

Das ST-Programm, Anweisungsteil:

```
(* Dynamische Konfiguration des Motor-Timers für Beschleunigungs- und Abbremsrampe  *)
IF( mot1_step < 1 ) THEN
        timer_pulse.VAL := 4500;
ELSIF( mot1_step < 2 ) THEN
        timer_pulse.VAL := 4000;
ELSIF( mot1_step < 3 ) THEN
        timer_pulse.VAL := 3800;

ELSIF( mot1_step < 598 ) THEN
        timer_pulse.VAL := 3700;       (* Maximum Speed *)

ELSIF( mot1_step < 599 ) THEN
        timer_pulse.VAL := 3800;
ELSIF( mot1_step < 600 ) THEN
        timer_pulse.VAL := 4000;
ELSE
        timer_pulse.VAL := 4500;
END_IF;
```

Mehrere Motoren: Kein Problem

Die Applikation kann einfach auf zwei, drei oder vier Motoren erweitert werden:

Quellcode kopieren und die Variablen umbenennen, Ausgangsklemmen für die weiteren Motoren zuordnen, fertig.

Jeder Motor erhält seine eigene, unabhängige „Hardware“ innerhalb des FPGA.

Steuern von Schrittmotoren mit FPGA-basierter SPS

Hier spielt die Musik!

AUTOMATION

Steuern von Schrittmotoren mit FPGA-basierter SPS

Hier spielt die Musik!

Ausbau zur Applikation „Trommeln“:

Programmierung des Rhythmus‘, den die Schrittmotoren zur Bewegung der Schlägel ausführen:

Realisierung mit einem Zähler, der über einen weiteren Timer gesteuert wird (übergeordnete Statemachine). Abhängig vom Zählerstand werden die Motorenbewegungen aktiviert bzw. deaktiviert.

AUTOMATION

Reale Einsatzgebiete

Von einer SPS gesteuerte Schrittmotoren sind in vielen Einsatzgebieten vorteilhaft, insbesondere, wenn es um präzise Positionierung geht.

Beispiele:

- Werkzeugmaschinen
- Roboter
- Laser-Cutter
- Schneidplotter

…

Vereinfachung der Programmerstellung mit Funktionsblöcken

Die SPS-Programmiersprache „Strukturierter Text“ unterstützt auch den Aufruf von Funktionsblöcken.

Erstellung von wiederverwendbaren Funktionsblöcken: Dort werden die Taktfolgen für die Ausgänge sowie Beschleunigungs- und Abbremsrampen programmiert.

Dann Aufruf:

```
<Funktionsblock-Name> (<Anzahl Schritte>, <Drehrichtung>);
```

Zusammenfassung

Das Ziel, Schrittmotoren ohne zusätzlichen Motion-Controller zu steuern, wurde erreicht. Dies wird möglich durch die Fähigkeit des in der SPS implementierten FPGA, die programmierten Steuerungsaufgaben echt parallel und damit zeitlich sehr präzise auszuführen.

Das Praxisbeispiel hat gezeigt, dass man - mit geeigneten Programmierwerkzeugen - kein „FPGA-Spezialist“ sein muss, um eine FPGA-basierte SPS zu programmieren.

Steuern von Schrittmotoren mit FPGA-basierter SPS

Für weitere Informationen nehmen Sie bitte Kontakt auf:

H. ZANDER GmbH & Co. KG
- Dipl.-Ing. Alfons Austerhoff -
Am Gut Wolf 15
52070 Aachen
GERMANY
Tel: +49 (0)241 910501-19
Fax: +49 (0)241 910501-38
E-Mail: a.austerhoff@zander-aachen.de

AUTOMATION