Wann sollte man einen FPGA anwenden ?

Prof. Dr.-Ing. V. Pfeiffer

Vortrag

GI Regionalgruppe Rhein-Main

11. Mai 2017

Stand 11. 09 2017

Kürzel: /WANNFPGA/
Datei: VOLv\DOC\FPGA\ Wann_sollte_man_einen_FPGA_anwenden.doc

Dieses Dokument ist für zweiseitigen Ausdruck vorgesehen

Inhalt

1. WAS IST EIN FPGA ? 1

2. FPGA MIT MIKROPROZESSOREN 3

3. PROGRAMMIERUNG 4

3.1. Ablauf 4

3.2. VHDL 5
3.2.1. historische Entwicklung 5
3.2.2. Beispielprogramm 6
3.2.3. Wirkungsplan ► VHDL 7

3.3. Schematic 8

4. NOTIERUNG UND REALISIERUNG DER ALGORITHMEN 9

4.1. Lebenszyklus 9

4.2. Entwurf 10

5. REALISIERUNGSNEUTRALE BESCHREIBUNGSMITTEL 13

5.1. Überblick 13

5.2. Wertetabelle 14

5.3. Zustandstabelle 17

5.4. Automatengraph 22

5.5. Programmablaufgraph 28

5.6. Vergleich 36

6. UMSETZUNGEN 37

6.1. Wertetabelle 37

6.2. Zustandstabelle 39

6.3. Zeitliches Verhalten 42
6.3.1. Zustandsregister 45
6.3.2. Voreinstell- und Haltezeit 49
6.3.3. Master-Slave-Zustandsregister 53
6.3.4. VHDL-Programm Master-Slave-Zustandsregister 56
6.3.5. Ausgangsregister 57

7. SIMULATION ? 58

8. INBETRIEBNAHME 58

9. WARUM SOLLTE ICH EINEN KONFIGURIERBAREN CHIP EINSETZEN ? 60

9.1. **Vorteile gegenüber klassischen (TTL-) Lösungen** **60**

9.2. **Vorteile gegenüber Mikroprozessoren** **60**

9.3. **Probleme** **60**

10. **ANHANG** **61**

11. **QUELLEN** **65**

12. **BEGRIFFE** **67**

13. **QUELLEN ZU DEN WISSENGEBIETEN CPLD, FPGA** **70**

14. **CPLD- FPGA-BAUEINHEITEN UND ZUBEHÖR** **81**

1. Was ist ein FPGA ?

FPGA-Schaltkreise (Field Programmable Gate Array) und
CPLD-Schaltkreise (Complex Programmable Logic Device)

- sind intern verbindungsprogrammierbare Einheiten.
- Sie gehören zu der Gruppe der konfigurierbaren Chips.

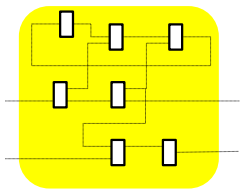


Bild 1: Intern verbindungsprogrammierbare Einheit

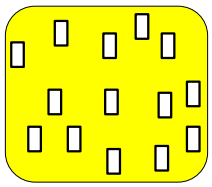


Bild 2: Intern verbindungsprogrammierbare Einheit, Elemente nicht verbunden

Auslieferungszustand

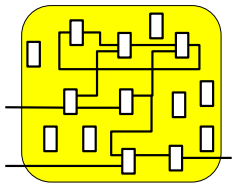


Bild 3: Intern verbindungsprogrammierbare Einheit, Elemente verbunden

durch Anwender programmiert

Die Verfahren für das Herstellen oder Trennen der internen Verbindungen (Tabelle 24, Seite 61) sind sehr unterschiedlich. Von dem Verfahren hängt es ab, ob die Verbindungen spannungsausfallsicher sind oder nicht.

CPLD	FPGA
einige Tausend Gatter	Millionen Gatter
"Verbindungen" spannungsausfallsicher (meistens)	"Verbindungen" nicht spannungsausfallsicher (meistens)

Tabelle 1: Vergleich CPLD / FPGA

Für die Komplexität der zu verbindenden Elemente gibt es ein großes Spektrum.
Mit der Komplexität wird über den notwendigen Flächenbedarf auf dem Chip und die mögliche Ausnutzung des Chips entschieden (Tabelle 2, Seite 2).

verbindbare Elemente

einfache Gatter	komplexe Elemente *(Bild 4)*
• alle Gatter komplett nutzbar • hoher Flächenbedarf (Chipfläche) für Verbindungen	• geringer Flächenbedarf (Chipfläche) für Verbindungen • nicht alle Teile der Elemente immer nutzbar

Tabelle 2: Komplexität der zu verbindenden Element

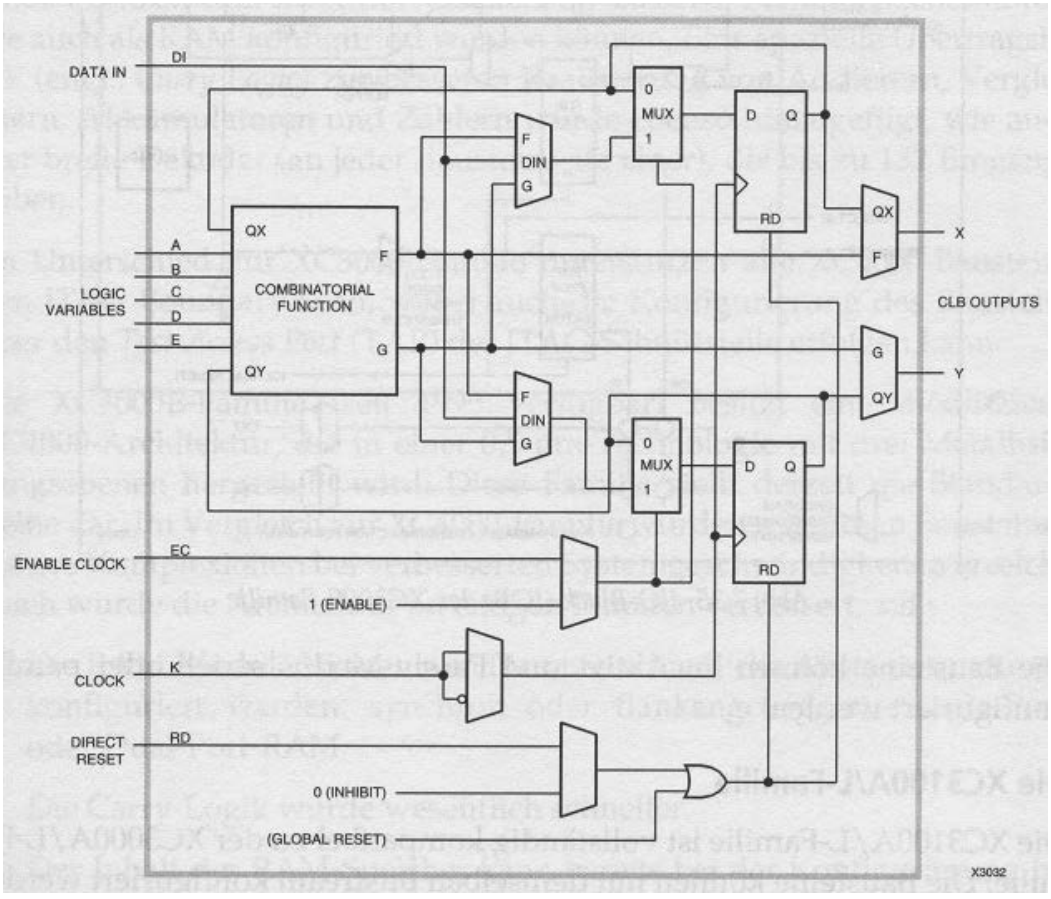


Bild 4: Logikblock (CLB) der XC3000-Familie /FPGA_KOCHBUCH/

2. FPGA mit Mikroprozessoren

Diese FPGAs enthalten neben dem intern verbindungsprogrammierbaren Bereich auch einem Bereich mit Mikroprozessoren (MP) (Bild 6, Seite 3)

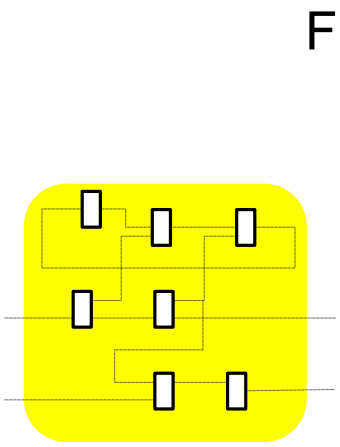


Bild 5: "klassischer FPGA", intern verbindungsprogrammierbar

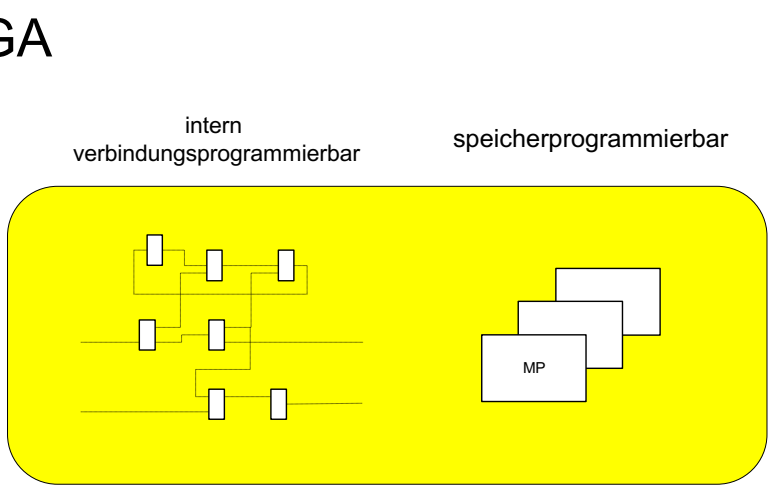


Bild 6: FPGA, intern verbindungsprogrammierbar + Mikroprozessoren (MP)

Programmierung	interne Verbindungsprogrammierung	Speicherprogrammierung
mögliche Realisierung Algorithmus	• parallel • seriell	• seriell

Bild 7: Eigenschaften der Bereiche des FPGA

parallele Realisierung Algorithmus

Für die gleichzeitige Realisierung eines durch die Beschreibungsmittel beschriebenen Zustandsübergangs in allen (parallelen) Automaten benötigt die Signale verarbeitende Einheit genau einen Zustandsübergang.

serielle Realisierung Algorithmus

Für die gleichzeitige Realisierung eines durch die Beschreibungsmittel beschriebenen Zustandsübergangs in allen (parallelen) Automaten benötigt die Signale verarbeitende Einheit nicht immer nur einen Zustandsübergang..

Beispiel siehe Bild 67, Seite 62

Damit ergeben sich neuen Möglichkeiten für die Implementierung der Algorithmen:

- Bewährte Lösungen können in den Mikroprozessoren realisiert werden.
- Neue Lösungen, auch parallele Realisierungen, können in dem verbindungsprogrammierbaren Teil realisiert werden.

Zunehmend befinden sich in den FPGAs Standard-Schnittstellen (Bild 68, Seite 62)

3. Programmierung

3.1. Ablauf

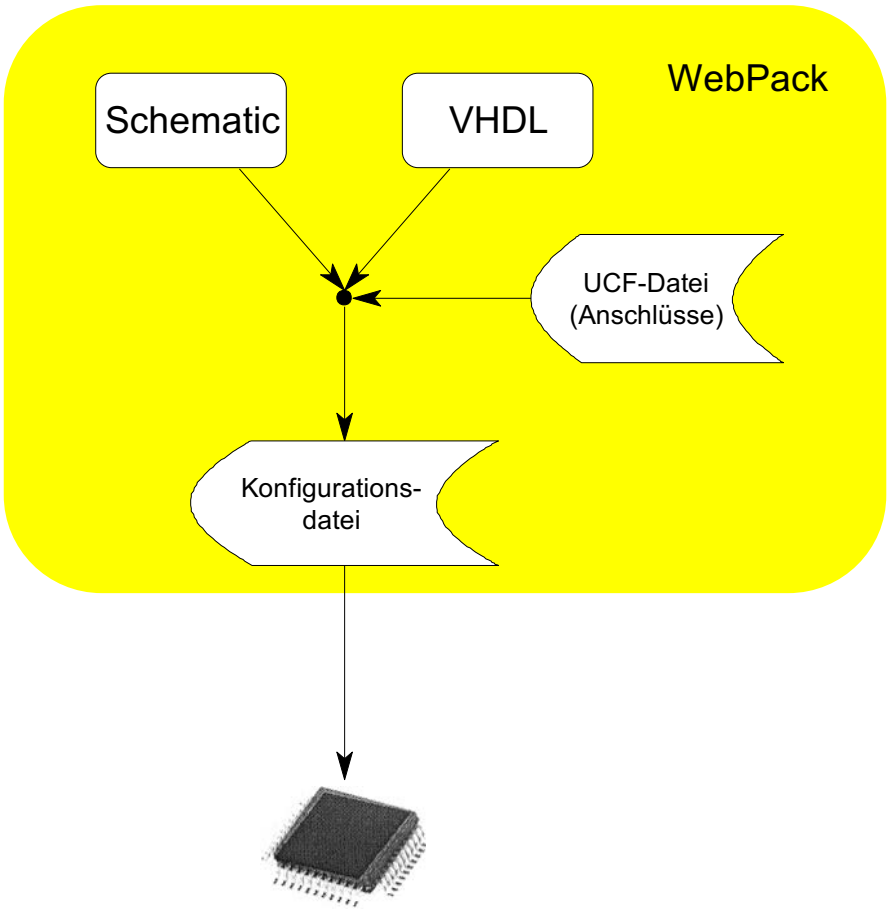


Bild 8: Ablauf Programmierung FPGA (WebPACK)

3.2. VHDL

3.2.1. historische Entwicklung

VHDL VHSIC Hardware description Language
VHSIC Very High Speed Integrated Circuit

Beschreibungsmittel zur Beschreibung von Verhalten und Struktur

Signale verarbeitender Einheiten

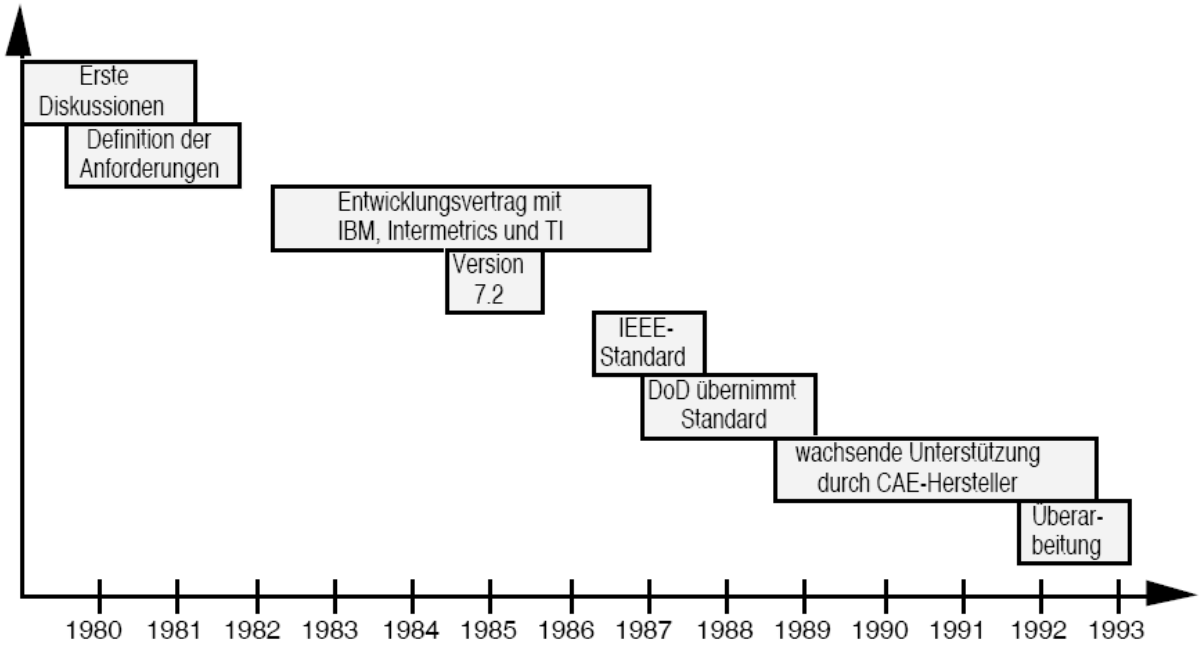


Bild 9: Historische Entwicklung VHDL /VHDL_DESIGN/

ursprünglich

Beschreibung des Verhaltens und der Struktur einer vorhandenen Hardware

heute

Beschreibung des Verhaltens und der Struktur einer zu realisierenden Hardware

3.2.2. Beispielprogramm

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity INTRO_BTN_LED_VHDL_XC2_XL_DIO1_SRC is
 Port ( IN_BTN : in std_logic;
   OUT_LED : out std_logic);
end INTRO_BTN_LED_VHDL_XC2_XL_DIO1_SRC;

architecture Behavioral of INTRO_BTN_LED_VHDL_XC2_XL_DIO1_SRC is

begin

 OUT_LED <= IN_BTN;

end Behavioral;
```

Programm 1: Beispielprogramm

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

benutzte verfügbare Bibliotheken

```
entity INTRO_BTN_LED_VHDL_XC2_XL_DIO1_SRC is
```

Einheit *Name* die beschrieben wird

```
Port ( IN_BTN : in std_logic;
  OUT_LED : out std_logic);
```

Port (Ein- /	Ausgänge)
out:	Ausgang
in:	Eingang
std_logic	Standardlogik,
BOOL	

```
architecture Behavioral of
INTRO_BTN_LED_VHDL_XC2_XL_DIO1_SRC is

begin

 OUT_LED <= IN_BTN;

end Behavioral;
```

Architektur
(hier nur)
Verhaltensbeschreibung

Der Datentyp "Aufzählung" ist verfügbar.

Damit ist keine binäre Zustandscodierung mit allen Folgeprobleme wie u. a. nicht beschriebene Zuständen notwendig.

3.2.3. Wirkungsplan ► VHDL

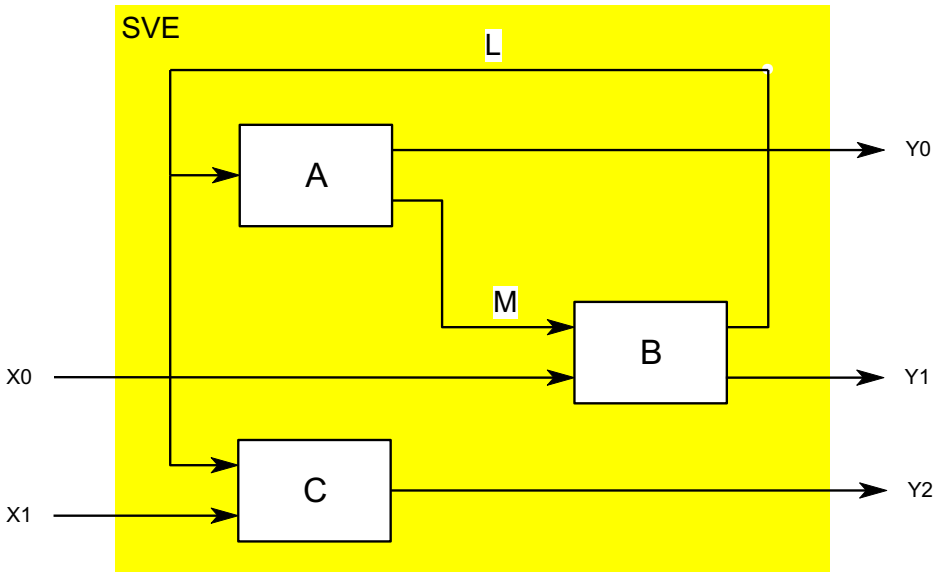


Bild 10: Wirkungsplan

```
entity SVE is                          -- SVE die umgesetzt werden soll
     Port ( X1 : in std_logic;         -- Ein und Ausgänge der SVE
      X0 : in std_logic;
      Y2 : out std_logic;
      Y1 : out std_logic;
      Y0 : out std_logic);
end SVE;

architecture Behavioral of SVE is      -- (Verhaltens-) und Struktur-
                                       -- Beschreibung SVE

     signal L : std_logic;             -- interne Variablen
     signal M : std_logic;             -- Datenaustausch zwischen Blöcken

begin
 Block_A process (L)                   -- Verhaltensbeschreibung Block A
 begin                                 -- (sensitive list) Eingangsvariablen
 …
 end process;

 Block_B process (M, X0)               -- Verhaltensbeschreibung Block A
 begin                                 -- (sensitive list) Eingangsvariablen
 …
 end process;

 Block_C process (L, X1)               -- Verhaltensbeschreibung Block A
 begin                                 -- (sensitive list) Eingangsvariablen
 …
 end process;
end Behavioral;
```

Programm 2: Prinzipielle Umsetzung des Wirkungsplans nach Bild 10

3.3. Schematic

Logikplan

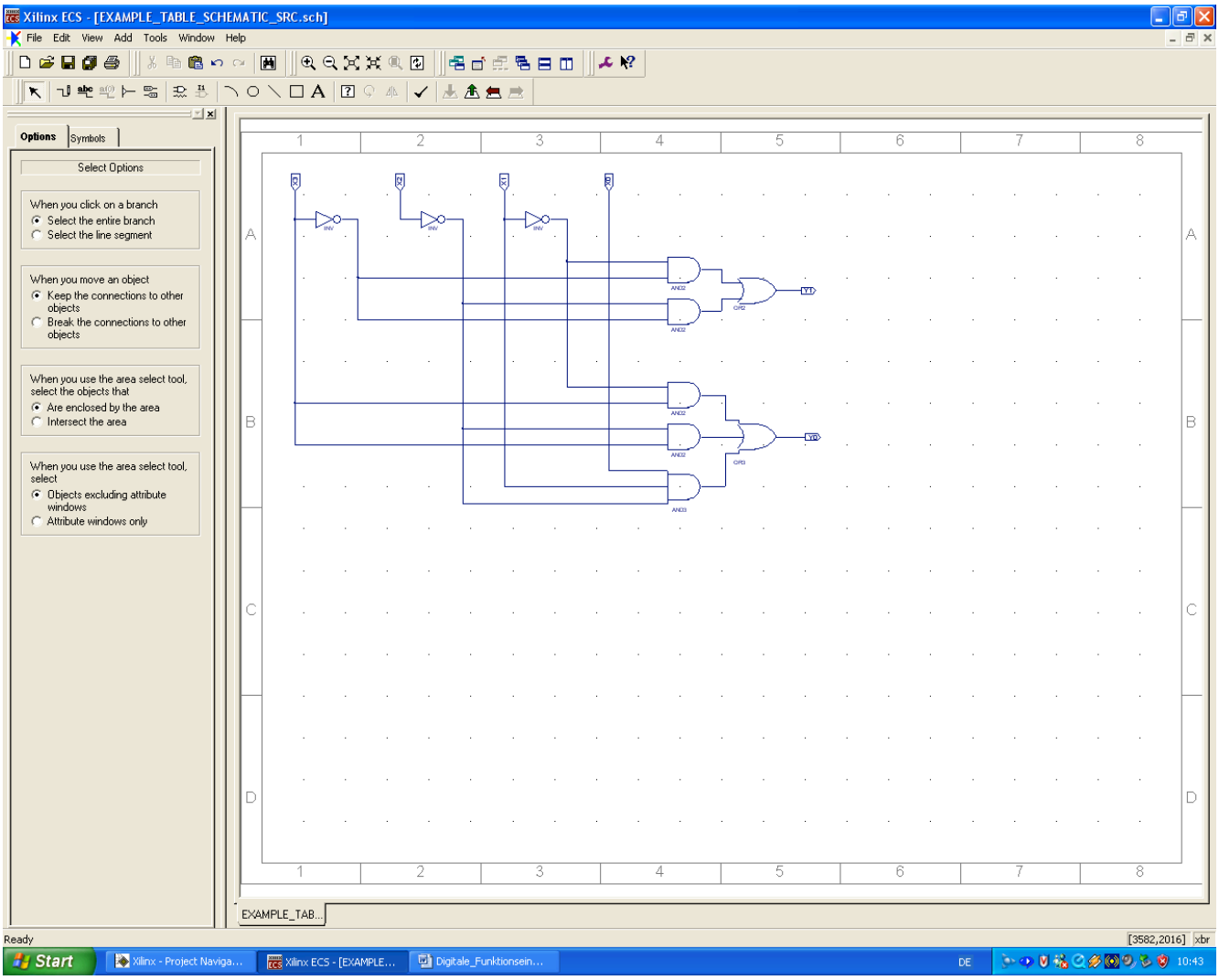


Bild 11: Schematic(Logikplan)

4. Notierung und Realisierung der Algorithmen

4.1. Lebenszyklus

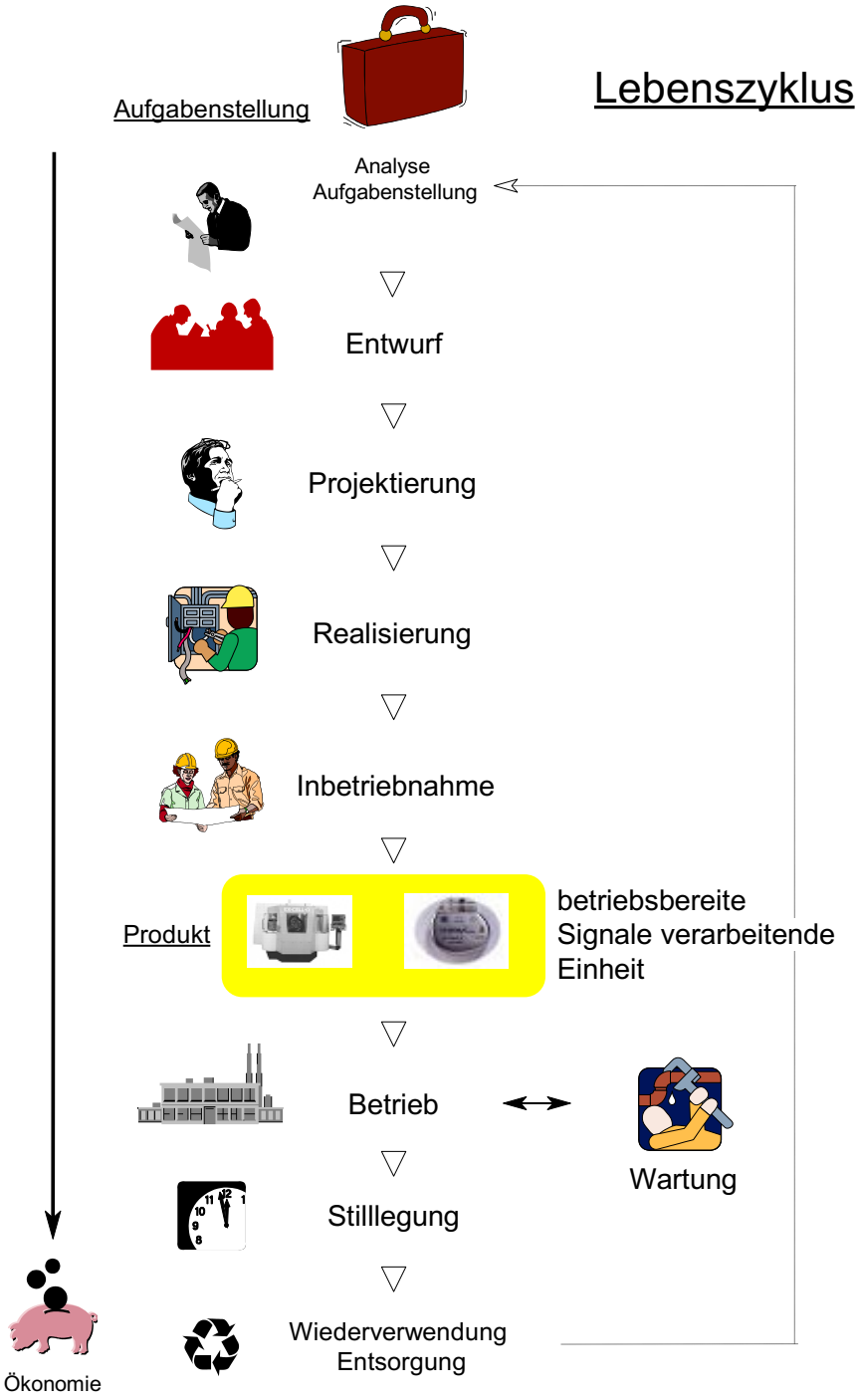


Bild 12: Phasen des Lebenszyklus einer Signale verarbeitenden Einheit

Von den Phasen des Lebenszyklus einer Signale verarbeitenden Einheit werden nachfolgend die Phasen Entwurf und Projektierung betrachtet.

4.2. Entwurf

In dieser Phase werden betrachtet:

- Informationsfluss SVE / Umgebung
- Variablendefinition
- Strukturbeschreibung
- Verhaltensbeschreibung
- Eigenschaften SVE

Begonnen wird mit einer Informationen verarbeitenden Einheit, die zunächst unscharf gegenüber der Umgebung abgegrenzt ist.

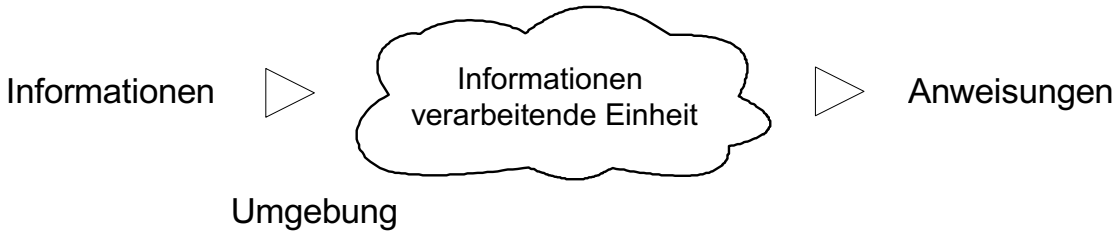


Bild 13: Informationen verarbeitende Einheit mit Informationen / Anweisungen

Danach folgt die Zuordnung der Informationen zu den Werten der Eingangsvariablen und der Anweisungen zu den Werten der Ausgangsvariablen. Es entsteht eine Signale verarbeitende Einheit.

Bild 14: Zuordnung der Informationen zu den Werten der Eingangsvariablen und der Anweisungen zu den Werten der Ausgangsvariablen

Die Signale verarbeitende Einheit wird genauer abgegrenzt und ihre Umgebung kann genauer differenziert werden.

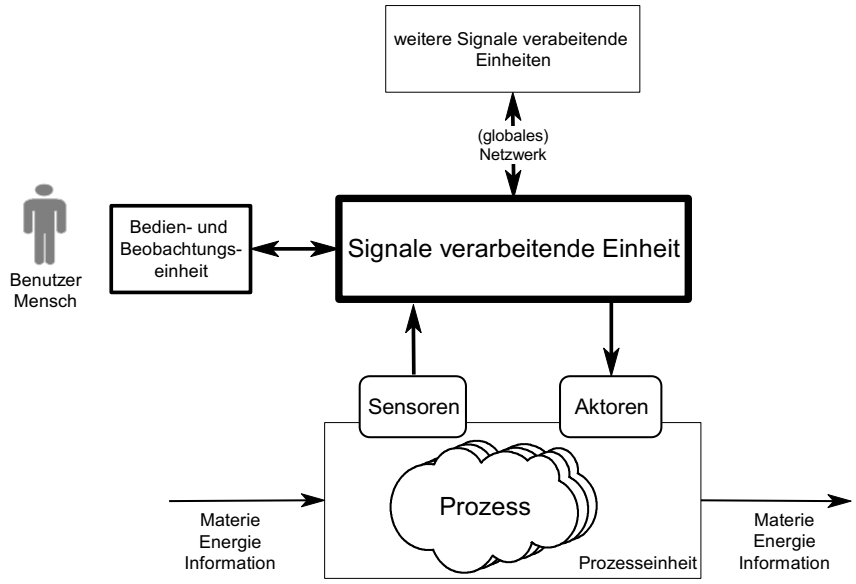


Bild 15: Signale verarbeitende Einheit / Umgebung

Danach werden beginnend mit Signale verarbeitenden Einheit und danach mit den Blöcken bezogen auf diese folgende Fragen gestellt und entsprechende Handlungen ausgeführt:

- Gibt es eine fertige Lösung in form eines Bausteins ?

  bevorzugte Lösung

- Soll eine Verhaltensbeschreibung erstellt werden ?

  Wenn Ja, dann Auswahl eines Beschreibungsmittels

- Soll eine weitere Zerlegung durchgeführt werden ?

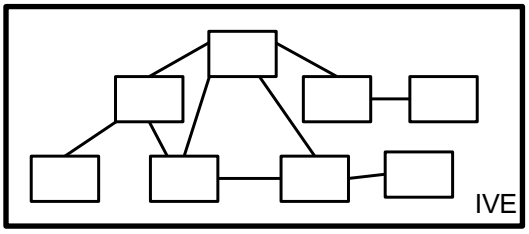


Bild 4-16: Zerlegung SVE in Blöcke

Vorteile	Nachteile
■ Es entstehen kleinere, überschaubare Einheiten ■ Die kleineren Einheiten können teilweise mit fertigen Lösungen realisiert werden	■ Das Gesamtverhalten wird nicht beschrieben Durch die Zusammenschaltung kann das Gesamtverhalten „überraschend anders“ sein ■ Im Falle der Realisierung mit Rechnerstrukturen kann zusätzliche Rechenleistung für den Datenaustausch zwischen den Blöcken notwendig sein

Bild 17: Vor- und Nachteile Zerlegung in Blöcke

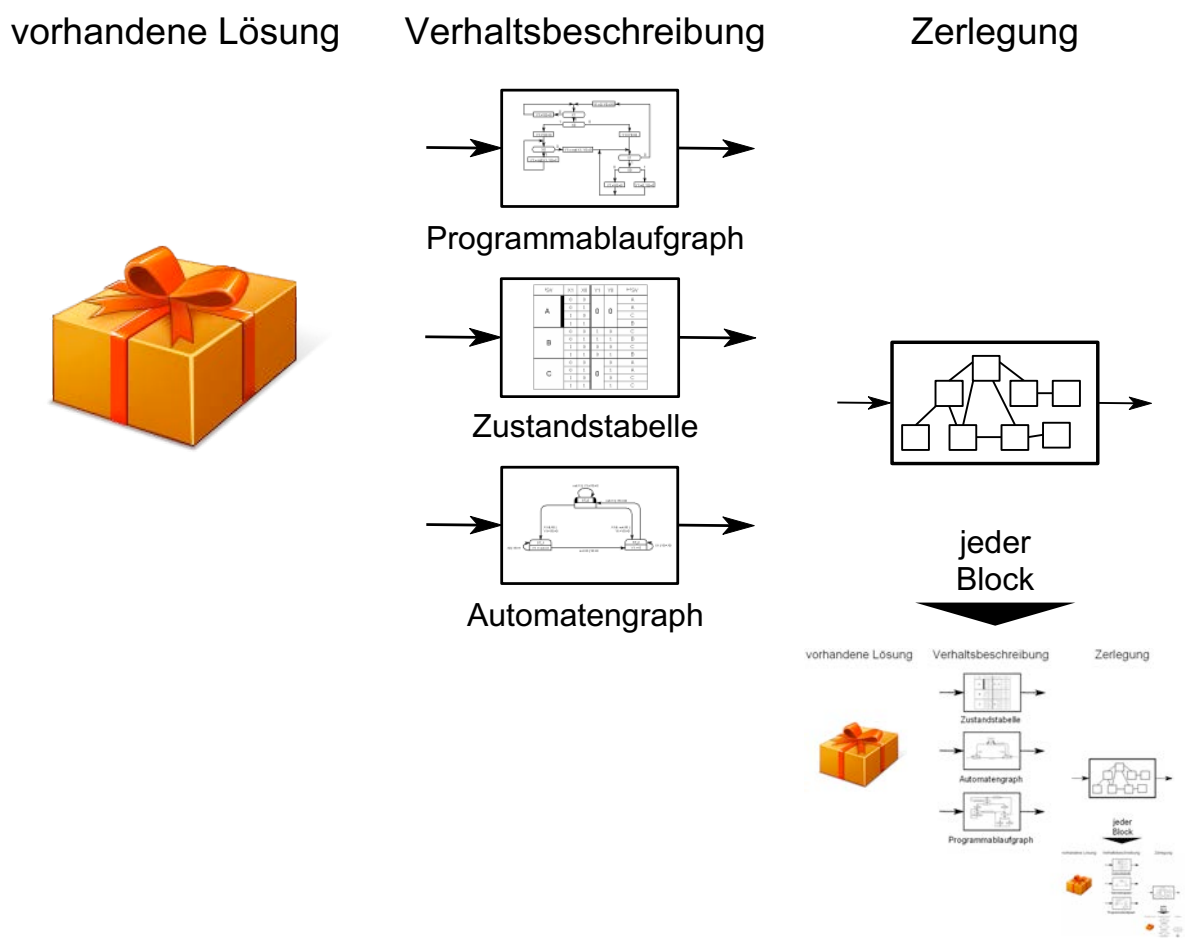


Bild 18: Mögliche Vorgehensweisen Signale verarbeite Einheit / Blöcke

5. Realisierungsneutrale Beschreibungsmittel

5.1. Überblick

- ausgehend von einem automatentheoretisch interpretierbare Beschreibungsmittel kann eine Signale verarbeitende Einheit nach einem Regelwerk realisiert werden

  automatisierbar !

- Korrekte Signale verarbeitende Einheit

  Die Signale verarbeitende Einheit verhält sich genau so, wie mit dem Beschreibungsmittel beschrieben

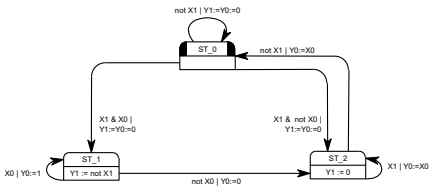
- Beschreibungen sind auch mit Nicht-Fachleuten diskutierbar

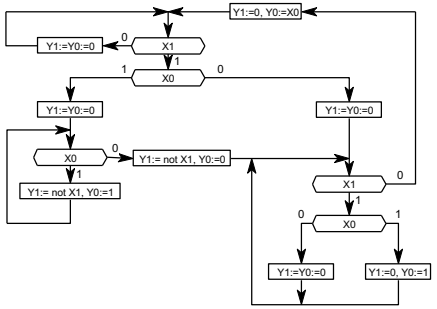
Wertetabelle	Zustandstabelle	Automatengraph	Programmablaufgraph

X2	X1	X0	Y1	Y0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Automat mit 1 Zustand

SV	X1	X0	Y1	Y0	$^{t+1}$SV
ST_0	0	0	0	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_1
ST_1	0	0	1	0	ST_2
	0	1	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0	0	ST_0
	0	1		1	ST_0
	1	0		0	ST_2
	1	1		1	ST_2





5.2. Wertetabelle

- **Beschreibungsmittel zur Beschreibung des Verhaltens**
- **Automat mit 1 Zustand**
- **In den Zellen der Ausgangsvariablen sind auch boolesche Ausdrücke möglich**

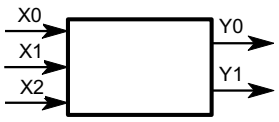


Bild 19: Einheit, deren Verhalten beschrieben werden soll

Wirkungsplan !

X2	X1	X0	Y1	Y0	
0	0	0	0	0	*0*
0	0	1	0	1	*1*
0	1	0	0	1	*2*
0	1	1	0	1	*3*
1	0	0	0	1	*4*
1	0	1	0	1	*5*
1	1	0	0	1	*6*
1	1	1	1	1	*7*

Tabelle 3: Beschreibungsmittel Wertetabelle

X2	X1	X0	Y1	Y0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	÷	0	1
1	1	0	0	0
1	1	1	1	1

Tabelle 5-4: Wertetabelle mit gleichgültigen Werten

Gesetz 1: Verhalten bei unvollständiger und widersprüchlicher Beschreibung

Wenn die Beschreibung nicht vollständig oder nicht widerspruchsfrei ist, hängt das Verhalten von der Realisierung ab.

Definition 1: Vollständigkeit und Widerspruchsfreiheit Wertetabelle

Eine Wertetabelle ist vollständig und widerspruchsfrei, wenn für jede Belegung der Eingangsvariablen genau eine Zeile existiert und für jede Ausgangsvariable in jeder dieser Zeilen genau ein Wert angegeben ist.

- boolesche Eingangsvariablen ➔ 8 Belegungen, 8 Zeilen
- in jeder Zeile ein Wert für Y1 und Y0 definiert

X2	X1	X0	Y1	Y0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabelle 5: Wertetabelle vollständig und widerspruchsfrei

X2	X1	X0	Y1	Y0
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabelle 6: Wertetabelle, Widerspruch Belegungen

X2	X1	X0	Y1	Y0	Y1
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	0

Tabelle 7: Wertetabelle, Widerspruch Ausgangsvariablen

X2	X1	X0	Y1	Y0	
0	0	1	0	1	*0*
0	1	0	0	1	*1*
0	1	1	0	1	*2*
1	0	0	0	1	*3*
1	0	1	0	1	*4*
1	1	0	0	1	*5*
1	1	1	1	1	*6*

Tabelle 8: Wertetabelle, Unvollständigkeit Belegung

X2	X1	X0	Y1	Y0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0		1
1	1	1	1	1

Tabelle 9: Wertetabelle, Unvollständigkeit Ausgangsvariablen

5.3. Zustandstabelle

- **Beschreibungsmittel zur Beschreibung des Verhaltens**
- **In den Zellen der Ausgangsvariablen sind auch boolesche Ausdrücke möglich**

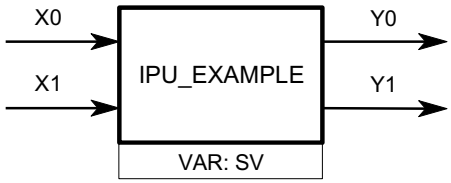


Bild 20: Einheit, deren Verhalten beschrieben werden soll

Name	Datentyp	Quelle	Beschreibung
X0	BOOL	Umgebung	Eingangsvariable
X1	BOOL	Umgebung	Eingangsvariable
Y0	BOOL	IPU_EXAMPLE	Ausgangsvariable
Y1	BOOL	IPU_EXAMPLE	Ausgangsvariable
SV	ENUM	IPU_EXAMPLE intern	Zustandsvariable, Literale: ST_0, ST_1, ST_2

Name	Datentyp	Quelle	Beschreibung
X0	BOOL	Umgebung	Eingangsvariable
X1	BOOL	Umgebung	Eingangsvariable
Y0	BOOL	IPU_EXAMPLE	Ausgangsvariable
Y1	BOOL	IPU_EXAMPLE	Ausgangsvariable
SV	ENUM	IPU_EXAMPLE intern	Zustandsvariable, Literale: ST_0, ST_1, ST_2

Tabelle 10: Beispiel-Algorithmus, Variablendefinition

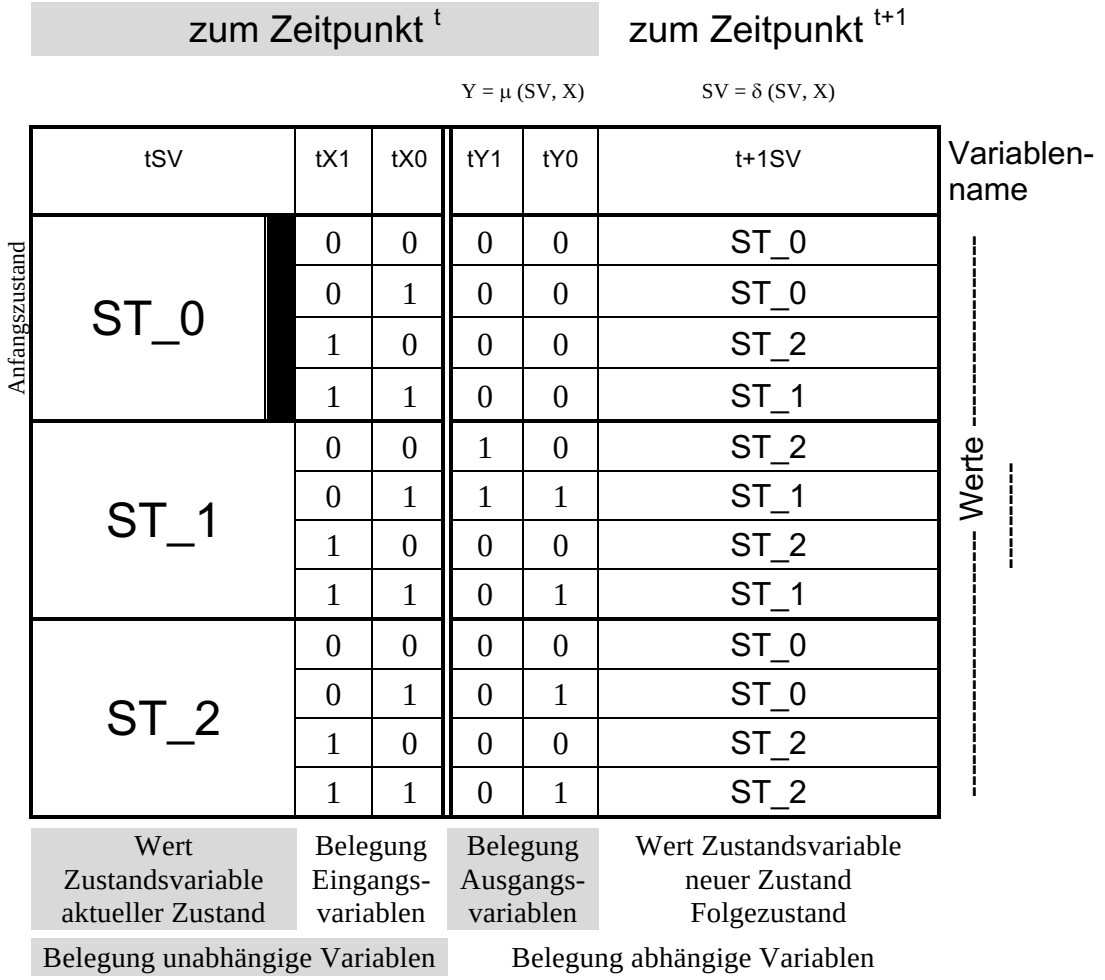


Tabelle 11: Beschreibungsmittel Zustandstabelle

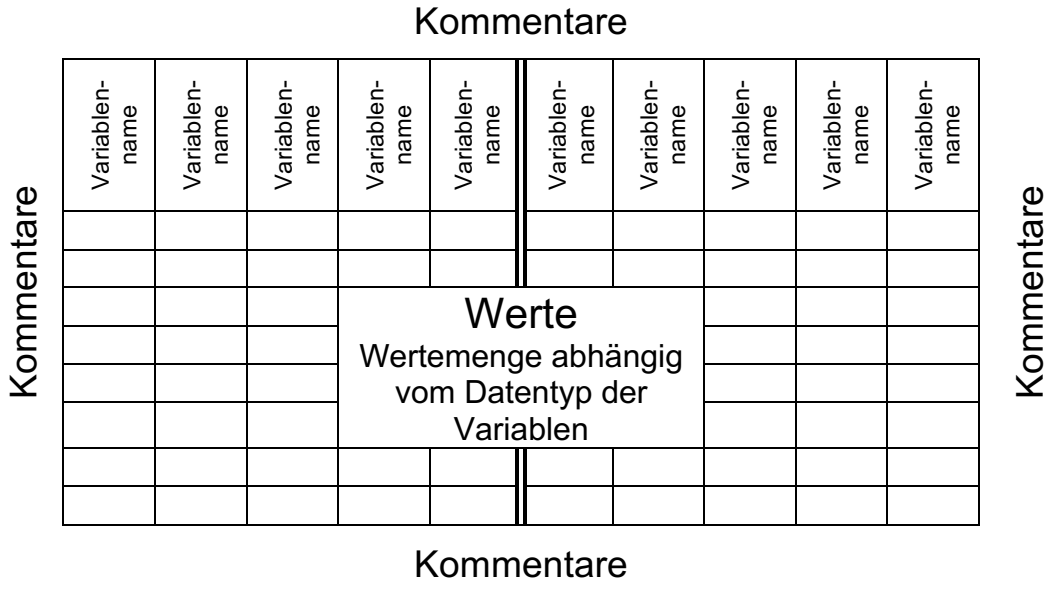


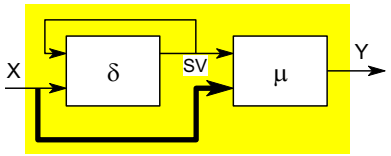
Tabelle 12: Zustandstabelle, Elemente

SV	X1	X0	Y1	Y0	$^{t+1}$SV
ST_0	0	÷	0	0	ST_0
	1	0			ST_2
	1	1			ST_1
ST_1	0	0	1	0	ST_2
	0	1	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0	0	ST_0
	0	1		1	ST_0
	1	0		0	ST_2
	1	1		1	ST_2

Tabelle 13: Beispiel-Zustandstabelle mit gleichgültigem Wert einer Eingangsvariablen

SV	X1	X0	Y1	Y0	$^{t+1}$SV
ST_0	0	0	0	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_1
	0	÷	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0	0	ST_0
	0	1		1	ST_0
	others			0	ST_2

Tabelle 14: Zustandstabelle, gleichgültiger Wert, übrige Belegungen

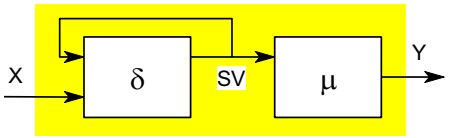


SV	X1	X0	Y1	Y0	$^{t+1}$SV
ST_0	0	0	0	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_1
ST_1	0	0	1	0	ST_2
	0	1	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0		ST_0
	0	1	1		ST_0
	1	0	0		ST_2
	1	1	1		ST_2

----Mealy----

Die Belegung der Ausgangsvariablen Y ist von den Werten der Zustandsvariablen SV (Zustände) und der Belegung der Eingangsvariablen X abhängig.

Tabelle 15: Beispiel-Zustandstabelle, Mealy-Automat



SV	X1	X0	Y1	Y0	$^{t+1}$SV
ST_0	0	0	0	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_1
ST_1	0	0	1	1	ST_2
	0	1			ST_1
	1	0			ST_2
	1	1			ST_1
ST_2	0	0	1	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_2

Die Belegung der Ausgangsvariablen Y ist nur von den Werten der Zustandsvariablen SV (Zustände) abhängig.

Tabelle 16: Beispiel-Zustandstabelle, Moore-Automat

Definition 2: Vollständigkeit, Widerspruchsfreiheit Zustandstabelle

- **Die Kopfzeile enthält** den Namen jeder Eingangs- und Ausgangsvariablen, des aktuellen Zustandes und des Folgezustandes genau einmal.
- Für alle Zustände genau eine Zeile pro Belegung der Eingangsvariablen vorhanden
- In jeder dieser Zeilen sind genau ein Folgezustand und genau ein Wert pro Ausgangsvariable notiert

SV	X1	X0	Y1	Y0	$^{t+1}$SV	
ST_0	0	0	0	0	ST_0	(0)
	0	1			ST_0	(1)
	1	1			ST_1	(2)
ST_1	0	0	0	0	ST_2	(3)
	0	1	0	1	ST_1	(4)
	1	0	0	0	ST_2	(5)
	1	0	1	1	ST_1	(6)
	1	1	1	1	ST_1	(7)
ST_2	0	0	0	0	ST_0	(8)
	0	1		1	ST_0	(9)
	1	0			ST_2	(10)
	1	1		1	ST_2	(11)

Unvollständigkeit:

- Im Zustand SV = ST_0 fehlt die Belegung der Eingangsvariablen X1 = 1, X0 = 0
- Im Zustand SV = ST_2 fehlt in der Zeile (10) der Wert für Y0

Widerspruch

Im Zustand SV = ST1 (Zeile (5), (6)) sind für die Belegung X1 = 1, X0 = 0 zwei sich widersprechende Belegungen der Ausgangsvariablen und Folgezustände angegeben.

Tabelle 17: Unvollständige und widersprüchliche Zustandstabelle

5.4. Automatengraph

- **Beschreibungsmittel zur Beschreibung des Verhaltens**

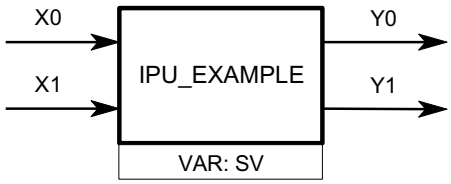
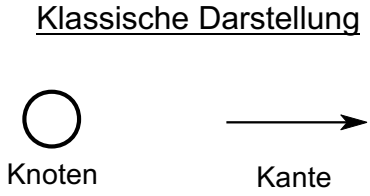


Bild 21: Einheit, deren Verhalten beschrieben werden soll

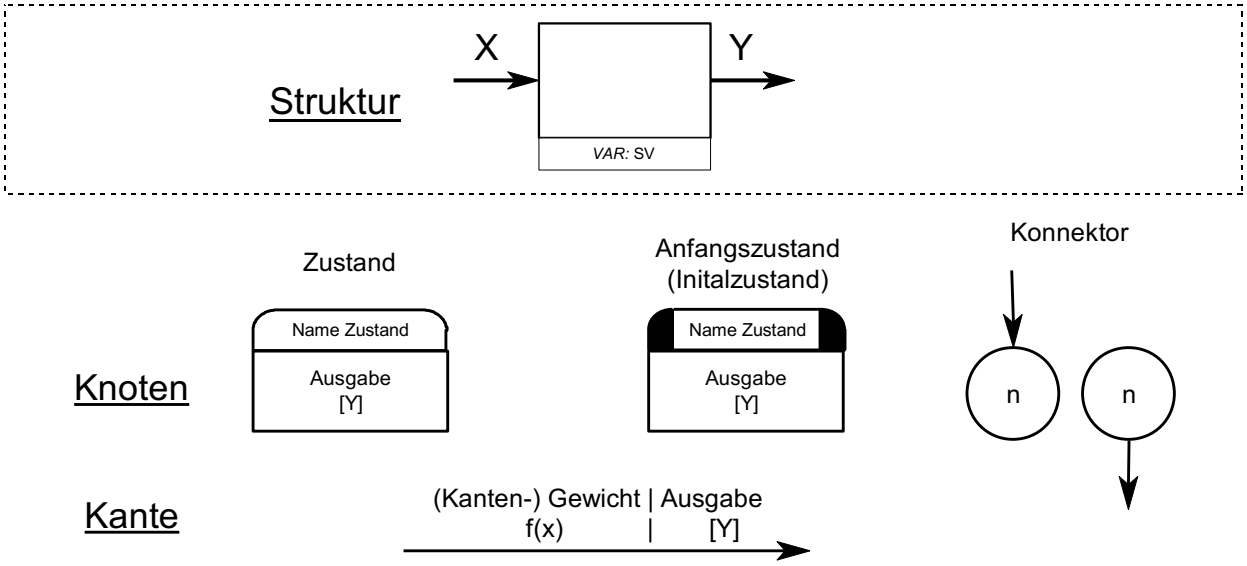
Name	Datentyp	Quelle	Beschreibung
X0	BOOL	Umgebung	Eingangsvariable
X1	BOOL	Umgebung	Eingangsvariable
Y0	BOOL	IPU_EXAMPLE	Ausgangsvariable
Y1	BOOL	IPU_EXAMPLE	Ausgangsvariable
SV	ENUM	IPU_EXAMPLE intern	Zustandsvariable, Literale: ST_0, ST_1, ST_2

Tabelle 18: Beispiel-Algorithmus, Variablendefinition



„es passt Nichts rein“

Bild 22: Klassische Darstellung der Elemente



Zustand: Wert der Zustandsvariablen ZV (in diesem Knoten, Zustand)

Ausgabe: Belegung der Ausgangsvariablen, die in diesem Zustand erzeugt werden soll
- Belegung der Ausgangsvariablen [Y] kann abhängig von der Belegung der Eingangsvariablen [X] sein
- Die Werte der Ausgangvariablen können alternativ im Knoten oder an den Kanten notiert werden

(Kanten-) Gewicht: boolescher Ausdruck, Kante wird durchlaufen, wenn Ausdruck erfüllt (Zustandsübergang)

| Trennzeichen zwischen Kantengewicht und Ausgabe

; Kommentar

Bild 23: Elemente des Automatengraphen

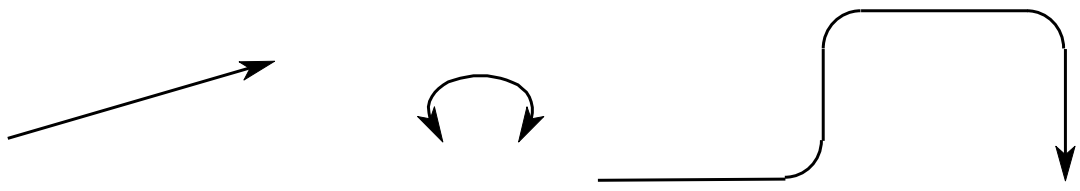


Bild 24: Varianten Kanten

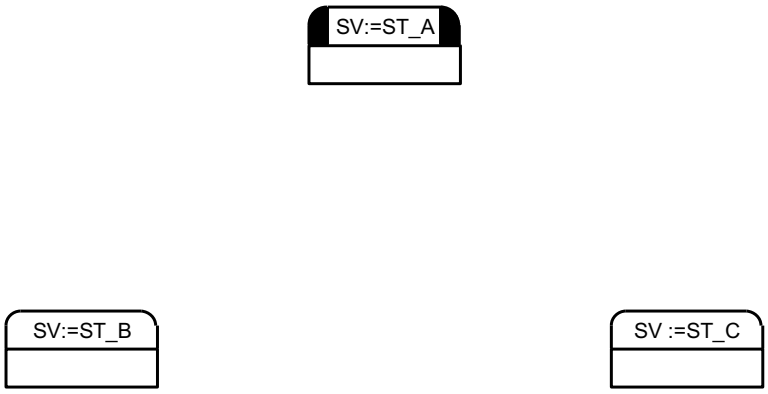
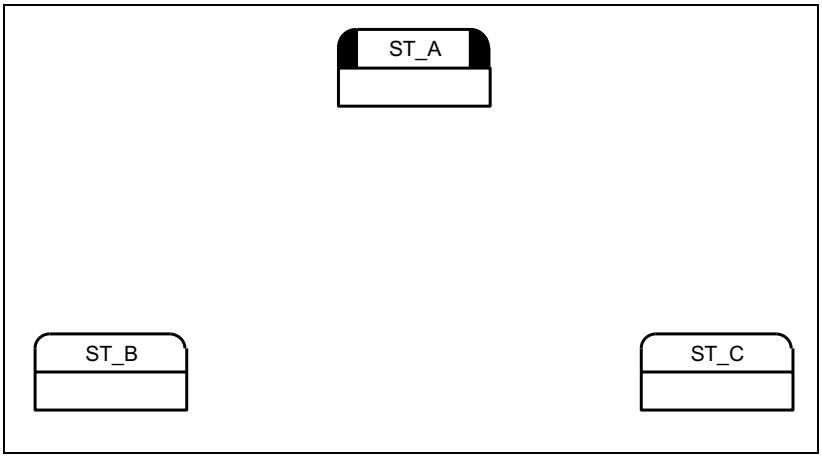


Bild 25: Ausführliche Darstellung der Zustände



bevorzugte Variante !

Bild 26: Kurzdarstellung Zustände

Notierung der Belegung der booleschen Ausgangsvariablen

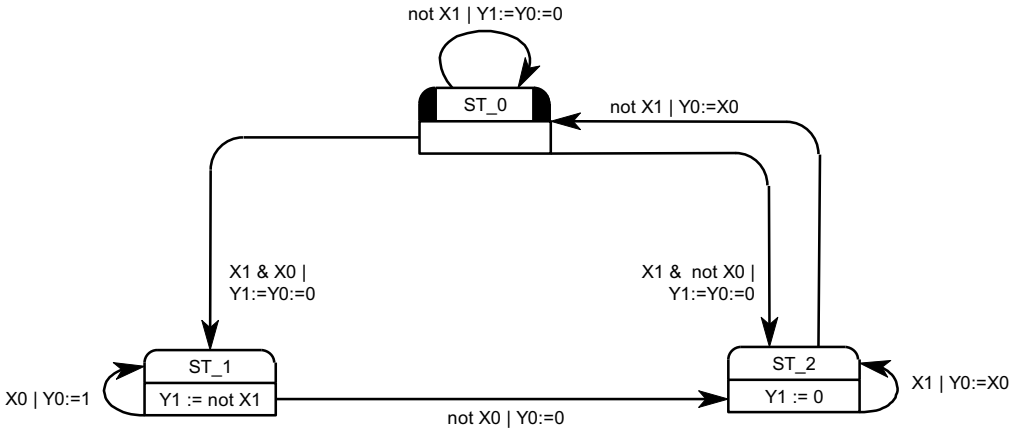


Bild 27: Beispiel-Automatengraph, Ausgangsvariable alle Werte notiert

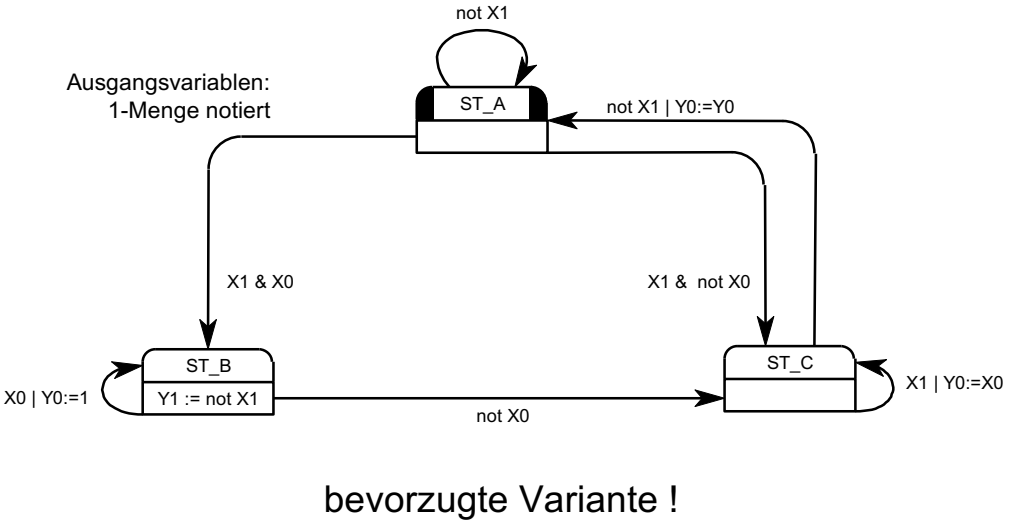


Bild 28: Beispiel-Automatengraph, Ausgangsvariablen: 1-Menge notiert

Gesetz 2: ***Eigenschleife***

Kantengewicht Eigenschleife eines Zustands: Negation der ODER-Verknüpfung der Kantengewichte der (übrigen) wegführenden Kanten

Die Eigenschleifen können auch weggelassen werden und die zugehörigen Ausgaben in den Knoten notiert werden

↯ *Definition 3: Vollständigkeit, Widerspruchsfreiheit abstrakter Automat*

Ein abstrakter Automat ist vollständig und widerspruchsfrei beschrieben, wenn
- in allen Zuständen
- für alle Belegungen der Eingangsvariablen
  - ein Folgezustand und
  - eine Belegung für alle Ausgangsvariablen definiert sind

angewendet auf Automatengraph

↯ *Definition 4: Vollständigkeit, Widerspruchsfreiheit Automatengraph*

Zustandsübergang
- Vollständigkeit
  Die Beschreibung mit dem Automatengraphen ist vollständig bezüglich der Zustandsübergänge, wenn in jeden Zustand die ODER-Verknüpfung der wegführenden Kanten (einschließlich Eigenschleife) 1 ergibt

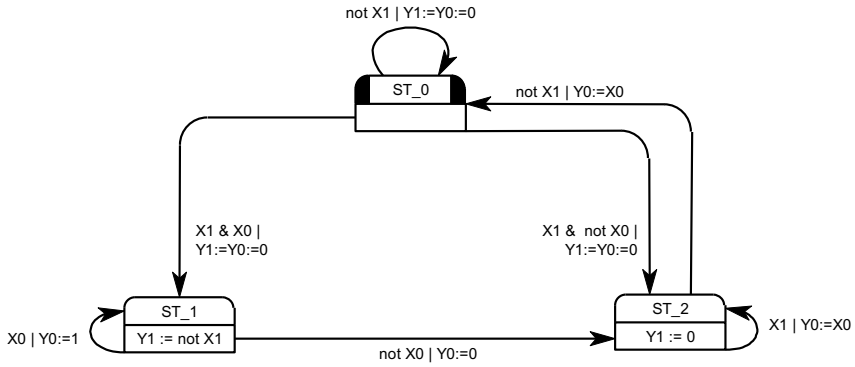
- Widerspruchsfreiheit
  Die Beschreibung mit dem Automatengraphen ist widerspruchsfrei bezüglich der Zustandsübergänge, wenn in jeden Zustand die paarweise UND-Verknüpfung der wegführenden Kanten (einschließlich Eigenschleife) 0 ergibt

Ausgabe
In jedem Zustand wird jeder Ausgangsvariablen genau ein Wert zugewiesen.

Anmerkungen:

Wenn die Eigenschleife nicht notiert wird, ist die Beschreibung immer vollständig bezüglich der Zustandsübergänge.

Wenn nicht alle Werte der Ausgangsvariablen notiert werden (z. B. nur 1-Menge), dann ist der Automatengraph immer vollständig bezüglich der Ausgangsvariablen. Es muss dann nur gezeigt werden, dass jeder Ausgangsvariablen höchstens ein Wert zugewiesen wird.



Vollständigkeit Zustandsübergänge

Zustand ST_0

(x1 & x0) or (x1 & not x0) or(not x1)
ST_0 ➔ ST_1 ST_0 ➔ ST_2 ST_0 ➔ ST_0

x1 (x0 or not x0) or not x1
x1 (1) or not x1
x1 or not x1

= 1

Gl. 1

Zustand ST_1

(not x0) or (x0)
ST_1 ➔ ST_2 ST_1 ➔ ST_1

= 1

Gl. 2

Zustand ST_2

(not x1) or (x1)
ST_2 ➔ ST_0 ST_2 ➔ ST_2

= 1

Gl. 5-3

Widerspruchsfreiheit Zustandsübergänge

Zustand ST_0

1. Paar

(x1 & x0) & (x1 & not x0)
ST_0 ➔ ST_1 ST_0 ➔ ST_2

= 0

Gl. 5-4

2. Paar

(x1 & x0) & (not x1)
ST_0 ➔ ST_1 ST_0 ➔ ST_0

= 0

Gl. 5-5

3. Paar

(not x1) & (x1 & not x0)
ST_0 ➔ ST_0 ST_0 ➔ ST_2

= 0

Gl. 5-6

Zustand ST_1

(x0) & (not x0)
ST_1 ➔ ST_1 ST_1 ➔ ST_2

= 0

Gl. 5-7

Zustand ST_2

(x1) & (not x1)
ST_2 ➔ ST_2 ST_2 ➔ ST_0

= 0

Gl. 5-8

Ausgabe

In jedem Zustand wird jeder Ausgangsvariablen genau ein Wert zugewiesen (in Knoten oder an Kanten).

5.5. Programmablaufgraph

Der Programmablaufgraph (PAG) ist ein mit Einschränkungen versehener Programmablaufplan /DIN 66001/. Aufgrund der Einschränkungen ist eine automatentheoretische Interpretation möglich.

Es werden betrachtet:
- Folge Belegung Eingangsvariablen
- Folge Belegung Ausgangsvariablen

„Zustände“ werden (zunächst) nicht betrachtet !

Elemente

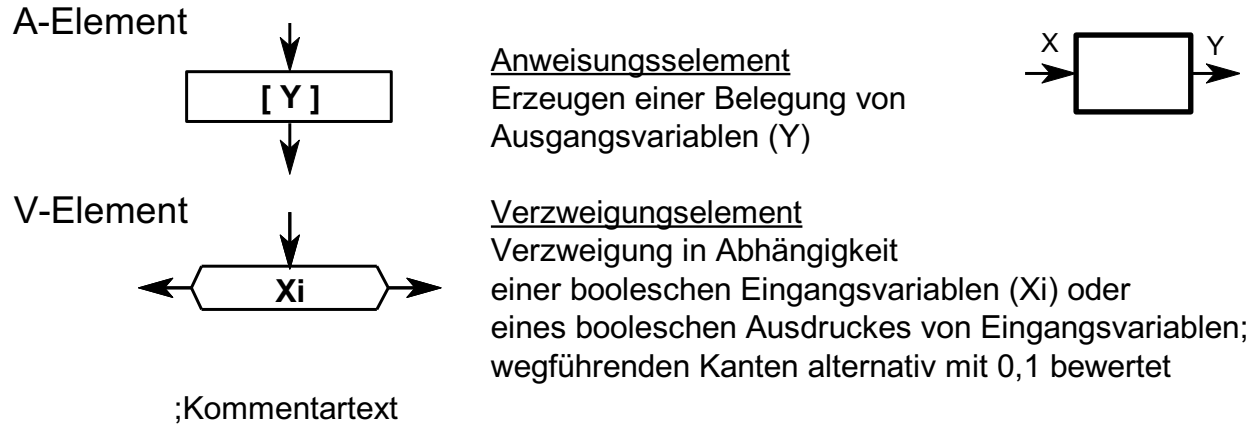


Bild 29: Grundelemente Programmablaufgraph

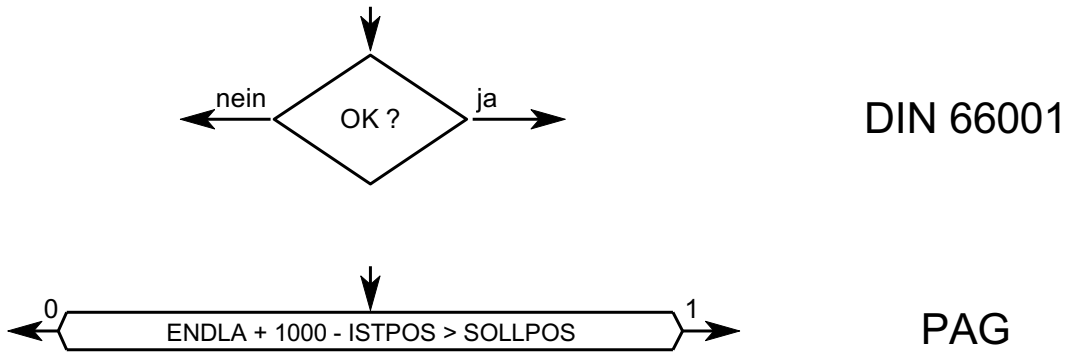


Bild 30: Modifikation Verzweigungselement

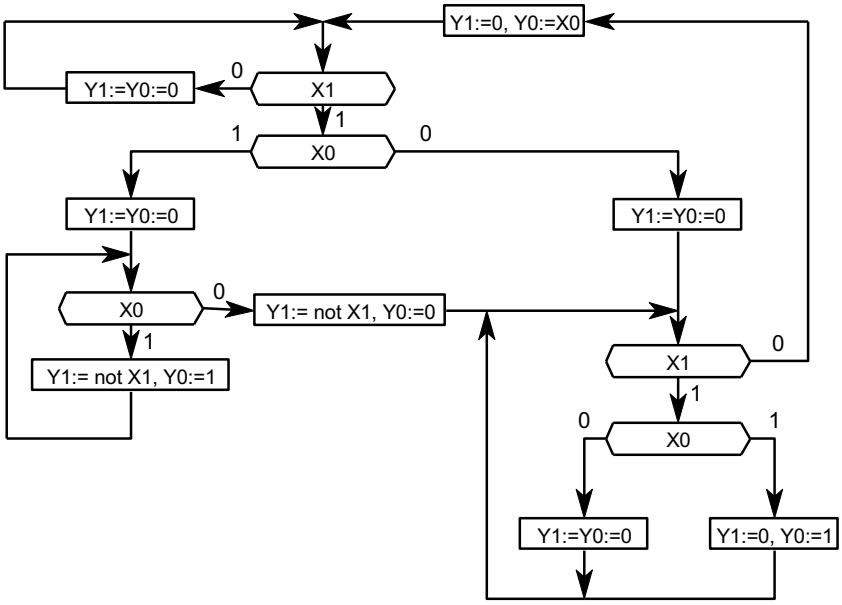


Bild 31: Beispiel-Programmablaufgraph

Knotenbewertungen

Bild 32: Beispiele Knotenbewertungen

hinführende Kanten, wegführende Kanten

Die Kanten werden bevorzugt von "oben" an das Element herangeführt .
Das Heranführen von "links" oder "rechts" ist auch möglich.
Von "unten" sollten die Kanten möglichst nicht herangeführt werden.

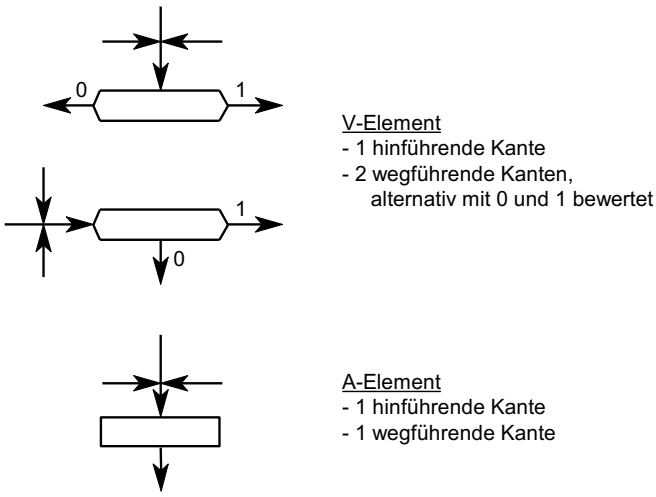


Bild 33: Elemente PAG und Anzahl der Kanten

Die wegführenden Kanten der Verzweigungselemente müssen alternativ bewertet sein.

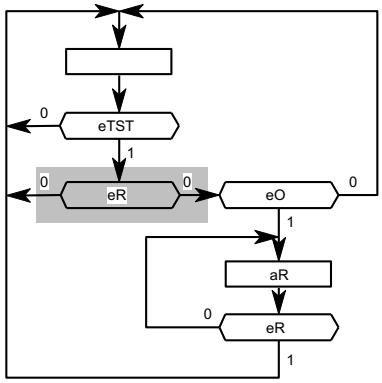


Bild 34: Fehlerhafter PAG, Kanten gleich bewertet

Zulässige Folgen

Vor dem Auftreffen auf ein Element können Kanten zusammengeführt werden).
Verzweigungen sind nur mit Verzweigungselementen möglich.

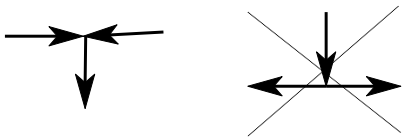


Bild 35: Zulässiges Zusammenführen von Kanten, unzulässiges Verzweigen von Kanten

Es darf keine Schleifen ohne Anweisungselement geben. Ohne Anweisungselement wird die Belegung der Ausgangsvariablen nicht definiert

Hinweis:
In Programmen für "Rechner" ist es möglich, dass innerhalb einer Schleife die Belegung der Ausgangsvariablen nicht definiert wird. Das ist unproblematisch weil speicherndes Verhalten bezüglich der Ausgangsvariablen vorausgesetzt wird. Der "alte Wert" der Variablen wird weiterhin ausgegeben.

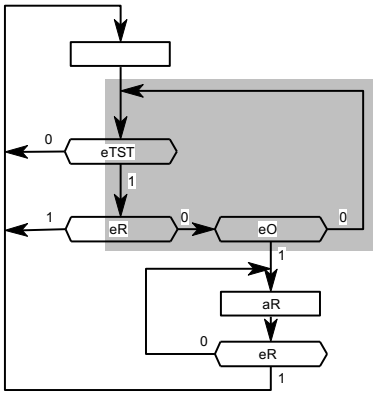


Bild 36: Fehlerhafter PAG, Schleife ohne Anweisungselement

Notierung Belegung Ausgangsvariablen

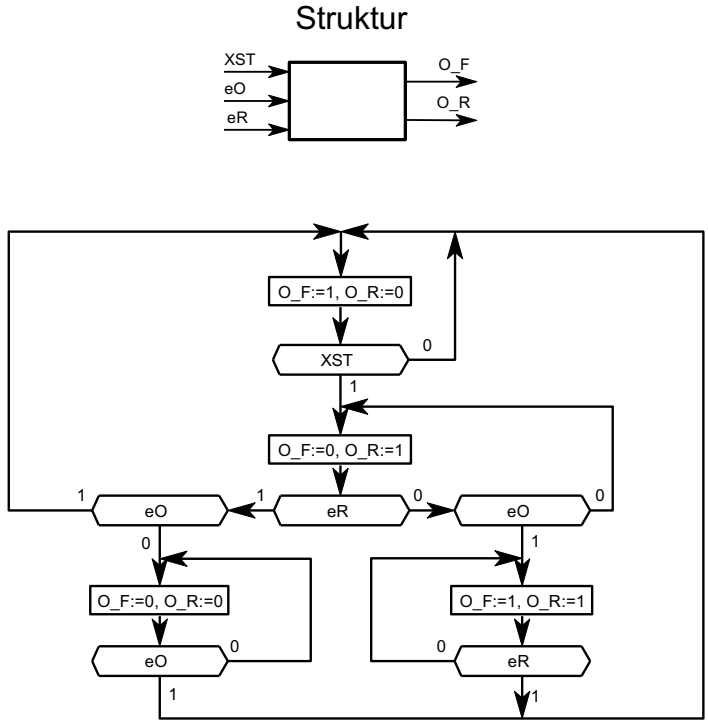


Bild 37: Notierung komplette Belegung in A-Elementen (0- und 1-Menge)

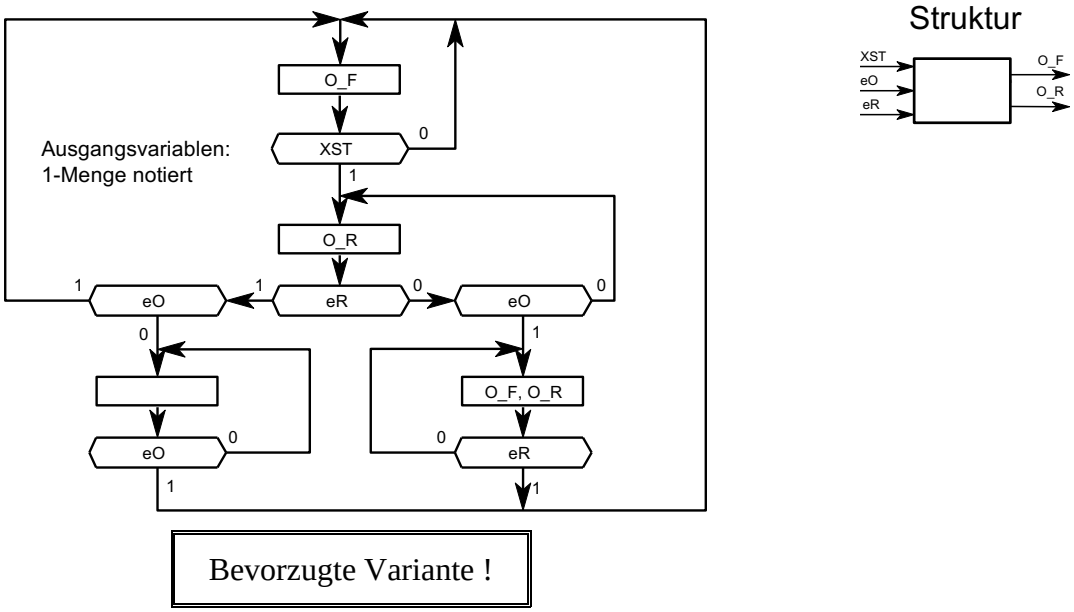
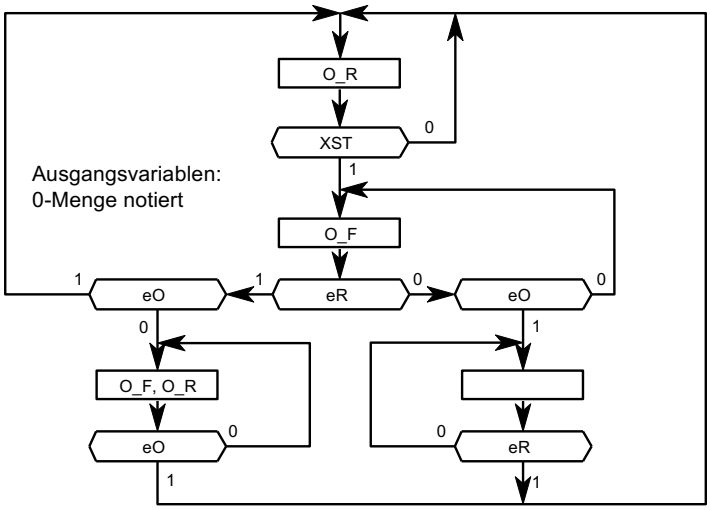


Bild 38: Notierung 1-Werte (Datentyp BOOL) in A-Elementen



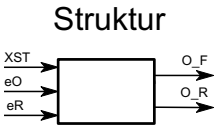
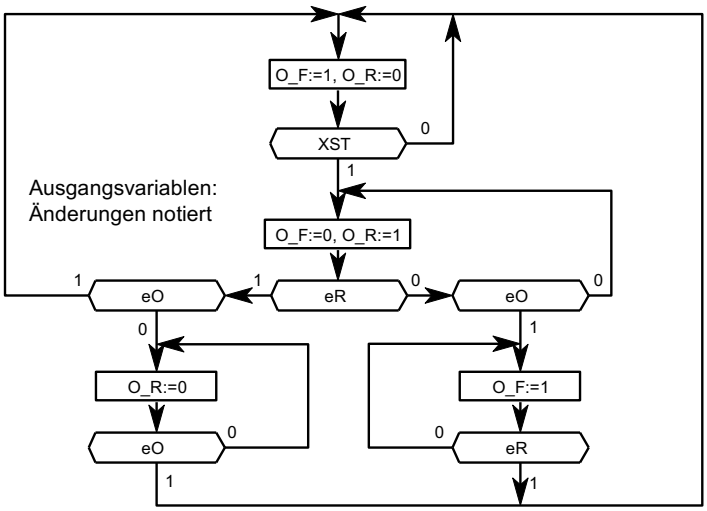


Bild 39: Notierung 0-Werte (Datentyp BOOL) in A-Elementen



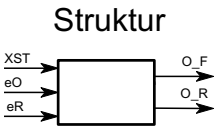
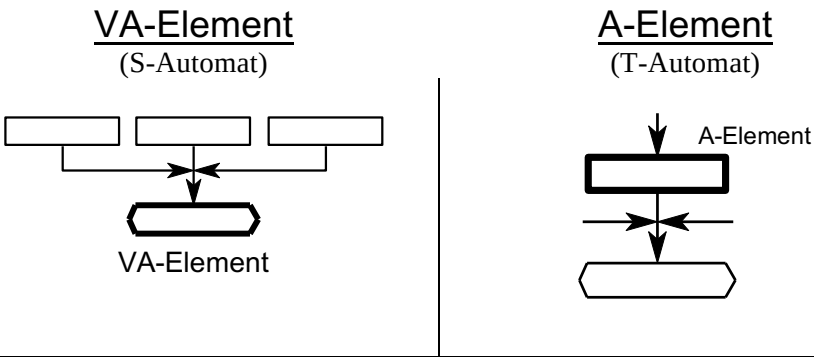


Bild 40: Notierung Änderungen (Datentyp BOOL) in A-Elementen)

Element, dem ein Zustand zugeordnet wird



	kleiner	größer
Zustandszahl		
syntaktische Regeln	• keine Schleife ohne A-Element • VA-Elemente ausschließlich A-Elemente als Vorgänger	• keine Schleife ohne A-Element

VA-Element
V- Element, welches ausschließlich Anweisungselemente (A-Elemente) als Vorgänger hat

Hinweis:
Wenn der PAG dieser Forderung nicht genügt, muss er um entsprechende Anweisungs-elemente ergänzt werden

Tabelle 19: Zustandsinterpretation Programmablaufgraph

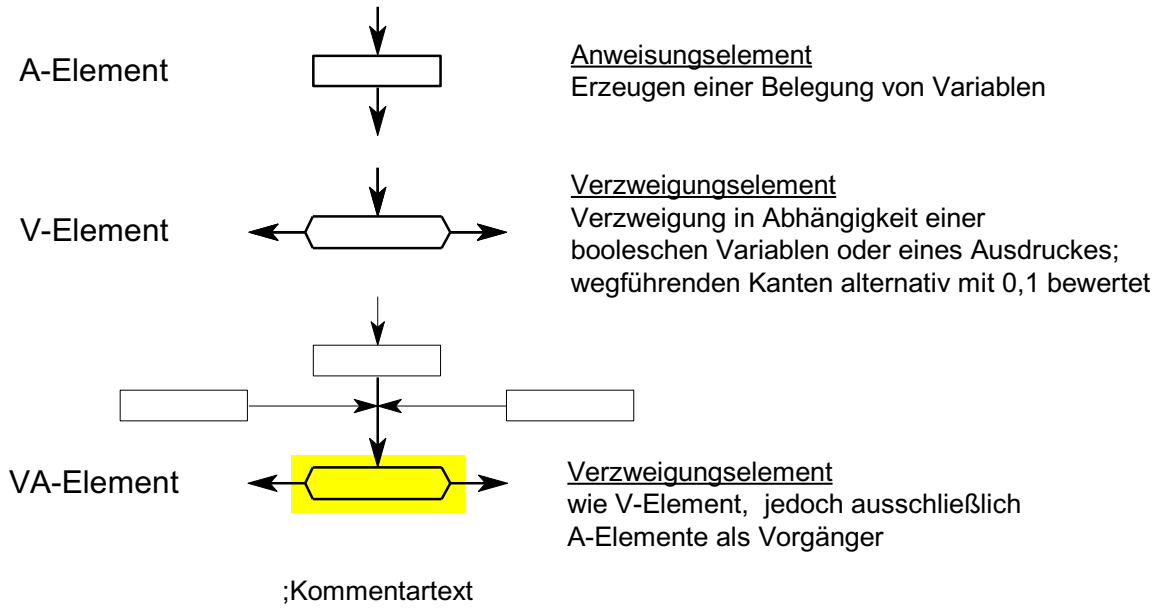
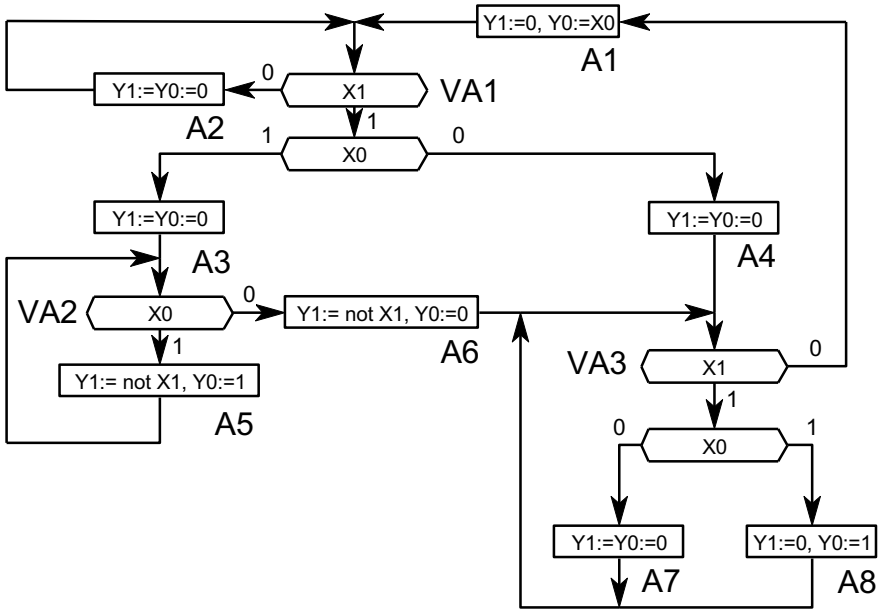


Bild 41: Elemente Programmablaufgraph, VA-Element



VA-Elemente: 3
A-Elemente: 8

Bild 42: Anzahl der VA- und A-Elemente

Gesetz 3: Anzahl VA-Elemente / A-Elemente

Anzahl VA-Elemente ≤ Anzahl A-Elemente

5.6. Vergleich

Zustandstabelle	Automatengraph	Programmablaufgraph
• zu jeder Belegung der Eingangsvariablen muss eine Festlegung getroffen werden • kein als Prüfvorschrift genutzt werden	• etwas aufwendigere Überprüfung Vollständigkeit Widerspruchsfreiheit	• zu Beginn keine "Zustände" erforderlich • Ablaufplan allgemein akzeptiert

6. Umsetzungen

6.1. Wertetabelle

X2	X1	X0	Y1	Y0	
0	0	0	0	0	*0*
0	0	1	0	1	*1*
0	1	0	0	1	*2*
0	1	1	0	1	*3*
1	0	0	0	1	*4*
1	0	1	0	1	*5*
1	1	0	0	1	*6*
1	1	1	1	1	*7*

Tabelle 20: Wertetabelle

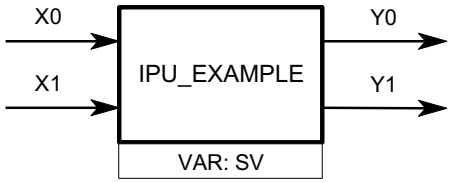
Zwei Prozesse.

- TYPE_CHG_PROC: Typwandlung Bit ➔ Vector
- TABLE_PROC Umsetzung Wertetabelle

Hinweise:
Das Programm nicht komplett, sondern nur stückweise eingeben und die Syntaktische Richtigkeit prüfen

```
-- Umsetzung der Beispiel-Wertetabelle mit CASE-Konstruktion
-- Bearbeiter: V. Pfeiffer
-- erstellt: 10. 12. 2013
-- geändert:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity EXAMPLE_TABLE_CASE_VHDL_SRC is
 Port ( X0 : in std_logic;
   X1 : in std_logic;
   X2 : in std_logic;
   X3 : in std_logic;
   Y0 : out std_logic;
   Y1 : out std_logic);
end EXAMPLE_TABLE_CASE_VHDL_SRC;

architecture Behavioral of EXAMPLE_TABLE_CASE_VHDL_SRC is

 signal IN_X: std_logic_vector (3 downto 0);

begin

 TYPE_CHG_PROC: process (X3, X2, X1, X0)
 begin
 IN_X (3) <= X3;
 IN_X (2) <= X2;
 IN_X (1) <= X1;
 IN_X (0) <= X0;
 end process;

 TABLE_PROC: process (IN_X)
 begin
 CASE IN_X is
 when "0000" => Y1 <= '1'; Y0 <= '0'; -- 0
 when "0001" => Y1 <= '1'; Y0 <= '0'; -- 1
 when "0010" => Y1 <= '1'; Y0 <= '0'; -- 2
 when "0011" => Y1 <= '1'; Y0 <= '1'; -- 3
 when "0100" => Y1 <= '1'; Y0 <= '0'; -- 4
 when "0101" => Y1 <= '1'; Y0 <= '0'; -- 5
 when "0110" => Y1 <= '0'; Y0 <= '0'; -- 6
 when "0111" => Y1 <= '0'; Y0 <= '0'; -- 7
 when "1000" => Y1 <= '0'; Y0 <= '1'; -- 8
 when "1001" => Y1 <= '0'; Y0 <= '1'; -- 9
 when "1010" => Y1 <= '0'; Y0 <= '1'; --10
 when "1011" => Y1 <= '0'; Y0 <= '1'; --11
 when "1100" => Y1 <= '0'; Y0 <= '1'; --12
 when "1101" => Y1 <= '0'; Y0 <= '1'; --13
 when "1110" => Y1 <= '0'; Y0 <= '0'; --14
 when "1111" => Y1 <= '0'; Y0 <= '0'; --15
 when others => Y1 <= '0'; Y0 <= '0';
 end case;

 end process;

end Behavioral;
```

6.2. Zustandstabelle



Name	Datentyp	Quelle	Beschreibung
X0	BOOL	Umgebung	Eingangsvariable
X1	BOOL	Umgebung	Eingangsvariable
Y0	BOOL	IPU_EXAMPLE	Ausgangsvariable
Y1	BOOL	IPU_EXAMPLE	Ausgangsvariable
SV	ENUM	IPU_EXAMPLE intern	Zustandsvariable, Literale: ST_0, ST_1, ST_2

Tabelle 21: Variablendefinition

SV	X1	X0	Y1	Y0	t^{+1}SV
ST_0	0	0	0	0	ST_0
	0	1			ST_0
	1	0			ST_2
	1	1			ST_1
ST_1	0	0	1	0	ST_2
	0	1	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0	0	ST_0
	0	1		1	ST_0
	1	0		0	ST_2
	1	1		1	ST_2

Mealy-Automat

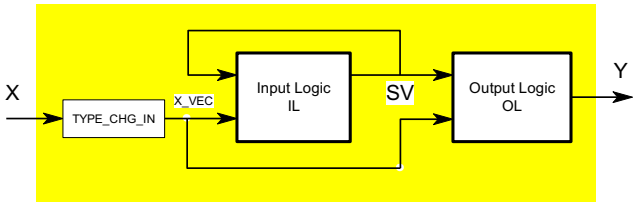


Bild 43; Struktur, Mealy-Automat

```
-- Umsetzung der Beispiel-Zustandstabelle mit CASE-Konstruktionen
-- Ausgangspunkt Mealy-Automat
-- Bearbeiter: V. Pfeiffer
-- erstellt: 15. 1. 2013
-- geändert:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity EXAMPLE_STATE_TABLE_CASE_VHDL_0_SRC is
 Port ( X0 : in std_logic; -- Eingangsvariable
  X1 : in std_logic; -- Eingangsvariable
  Y0 : out std_logic; -- Ausgangsvariable
  Y1 : out std_logic); -- Ausgangsvariable
end EXAMPLE_STATE_TABLE_CASE_VHDL_0_SRC;

architecture Behavioral of EXAMPLE_STATE_TABLE_CASE_VHDL_0_SRC is

 type TYPE_STATE is (ST_0, ST_1, ST_2);  -- Zustände

 signal SV:  TYPE_STATE;    -- Zustandsvariable
 signal X_VEC: std_logic_vector (1 downto 0); -- Vektor Eingangsvariablen

begin

 TYPE_CHG_IN_PROC: process (X1, X0)   -- X1, X0 in X_VEC
 begin          -- Typwandlung
 X_VEC(1) <= X1;
 X_VEC(0) <= X0;
 end process;



 IL_PROC: process (SV, X_VEC) -- Eingangslogik Input Logic IL
 begin
 case SV is
 when ST_0 =>
 case X_VEC is
  when "00" => SV <= ST_0;
  when "01" => SV <= ST_0;
  when "10" => SV <= ST_2;
  when "11" => SV <= ST_1;
  when others => SV <= ST_0;
 end case;
 when ST_1 =>
 case X_VEC is
  when "00" => SV <= ST_2;
  when "01" => SV <= ST_1;
  when "10" => SV <= ST_2;
  when "11" => SV <= ST_1;
  when others => SV <= ST_0;
 end case;
 when ST_2 =>
 case X_VEC is
  when "00" => SV <= ST_0;
  when "01" => SV <= ST_0;
  when "10" => SV <= ST_2;
  when "11" => SV <= ST_2;
  when others => SV <= ST_0;
 end case;
 when others => SV <= ST_0;
 end case;
 end process;

 OL_PROC: process (SV, X_VEC) -- Ausgangslogik Ouput Logic OL
 begin
 case SV is
 when ST_0 =>
 case X_VEC is
  when "00" => Y1 <= '0'; Y0 <= '0';
  when "01" => Y1 <= '0'; Y0 <= '0';
  when "10" => Y1 <= '0'; Y0 <= '0';
```

```
  when "11" => Y1 <= '0'; Y0 <= '0';
  when others => Y1 <= '0'; Y0 <= '0';
 end case;
 when ST_1 =>
 case X_VEC is
  when "00" => Y1 <= '1'; Y0 <= '0';
  when "01" => Y1 <= '1'; Y0 <= '1';
  when "10" => Y1 <= '0'; Y0 <= '0';
  when "11" => Y1 <= '0'; Y0 <= '1';
  when others => Y1 <= '0'; Y0 <= '0';
 end case;
 when ST_2 =>
 case X_VEC is
  when "00" => Y1 <= '0'; Y0 <= '0';
  when "01" => Y1 <= '0'; Y0 <= '1';
  when "10" => Y1 <= '0'; Y0 <= '0';
  when "11" => Y1 <= '0'; Y0 <= '1';
  when others => Y1 <= '0'; Y0 <= '0';
 end case;
 when others => Y1 <= '0'; Y0 <= '0';
 end case;
 end process;
end Behavioral;
```

Programm 6-1: Umsetzung Zustandstabelle, CASE-Konstruktion Mealy-Automat

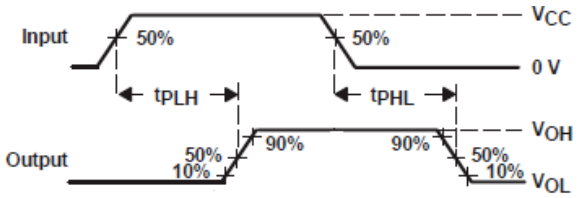
Block im Automatenmodell	Prozess im VHDL-Programm
Input Logic IL *(Eingangslogik)*	IL_PROC
Output Logic OL *(Ausgangslogik)*	OL_PROC
Type_CHG_IN *(Typwandlung)*	Type_CHG_IN

Tabelle 22: Zuordnung Blöcke / Prozesse

6.3. Zeitliches Verhalten

Bei den bisherigen Betrachtungen wurde davon ausgegangen, dass die Informationsverarbeitung innerhalb einer vernachlässigbaren Zeit durchgeführt wird.

In Gattern treten jedoch Verzögerungen auf (Bild 44). Eine Signaländerung am Eingang (Input) erscheint erst verzögert am Ausgang (Output). Diese Verzögerung wird als Gatterlaufzeit, Gatterverzögerungszeit bezeichnet. Vom Hersteller wird garantiert, dass ein vorgegebnes Maximum nicht überschritten wird.



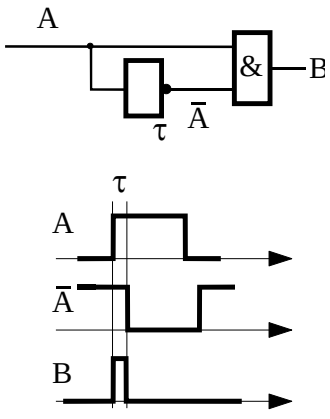
tpLH = tpHL ≤ 20 … 115 ns (abhängig von Betriebsbedingungen)

Bild 44: Verzögerung im Schaltkreis SN74HC00 /74HC00_DS/ (schematische Darstellung)

Die Schaltung nach realisiert die Funktion:

$$B := A \,\&\, \overline{A} := 0 \qquad \textbf{Gl. 6-1}$$

Aufgrund der Verzögerungszeiten der Gatter wird diese Funktion nicht immer realisiert:
Am Ausgang B tritt kurzzeitig der logische Wert „1“ auf.
Das kann zu Fehlverhalten der Schaltung führen. Besonders kritisch wird es, wenn ein isolierter Zustand angenommen wird.



τ Verzögerungszeit des Gatters

Bild 45: Beispiel für dynamische Effekte (Prinzip-Darstellung)

(Stör-) Impuls: Hazard
Glitch

Ab einer bestimmten Komplexität kann der entstehende Impuls auch praktisch nachgewiesen werden (Bild 47).

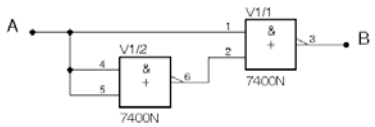


Bild 46: Impuls nicht nachweisbar

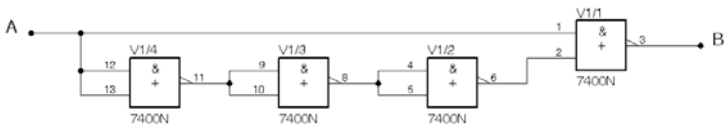


Bild 47: Impuls nachweisbar

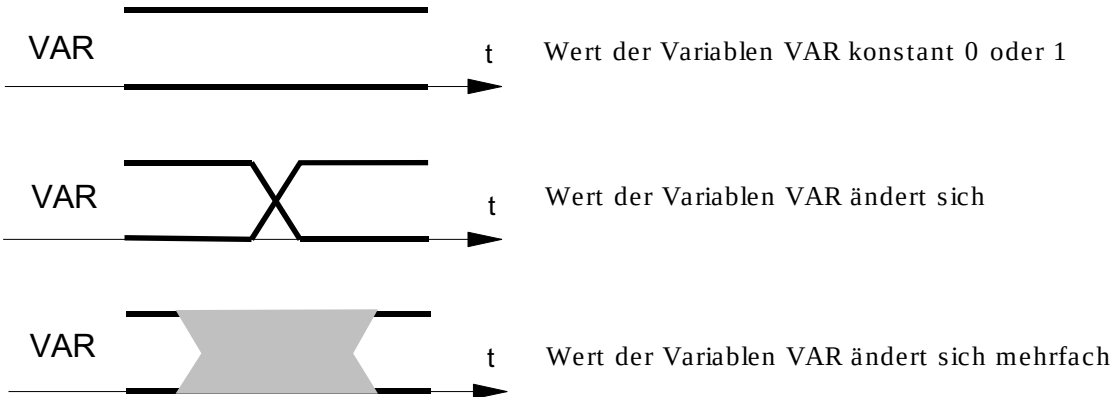


Bild 48: Darstellung der Signalverläufe

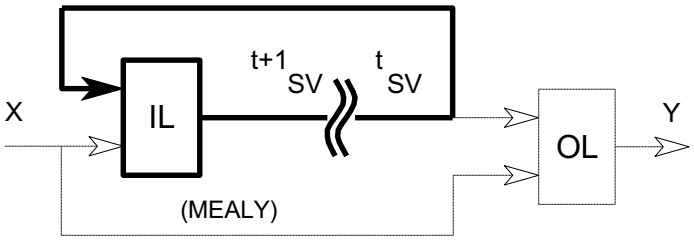


Bild 49: Rückführung

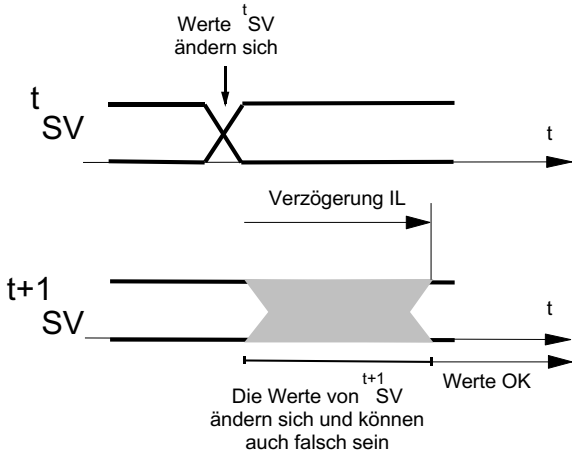


Bild 50: Signalverläufe

Die falschen Werte / Belegungen werden in der Eingangslogik verarbeitet:

➔ falsches Verhalten der Signale verarbeitenden Einheit

Auf diese Problematik weist bei einigen Schaltungen Webpack auch hin.

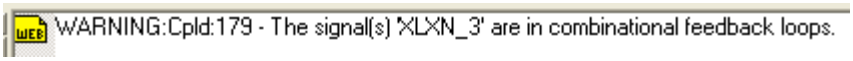


Bild 51: Warnung Webpack

Ob Probleme auftreten, hängt wie bereits festgestellt, von der Komplexität der Schaltung ab. Einfache (überschaubare) Schaltungen, wie z. B. Bild 52, funktionieren problemlos.

Im Falle komplexeren Schaltungen ist dies schon fraglich.

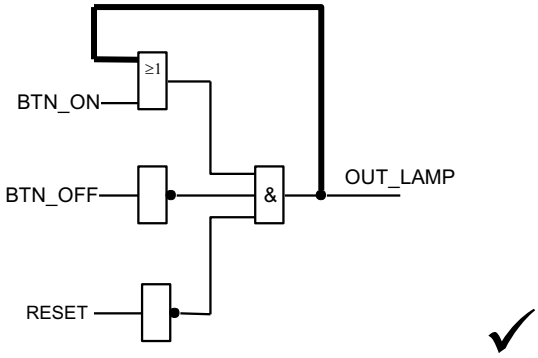


Bild 52: Selbsthaltschaltung

Durch das WebPACK wird ausgehend von den VHDL-Programmen für die Eingangslogik eine Schaltung ähnlich Bild 53 erzeugt. Die Werte der Zustandsvariablen SV werden mit den booleschen Variablen bSV1 und bSV0 codiert.

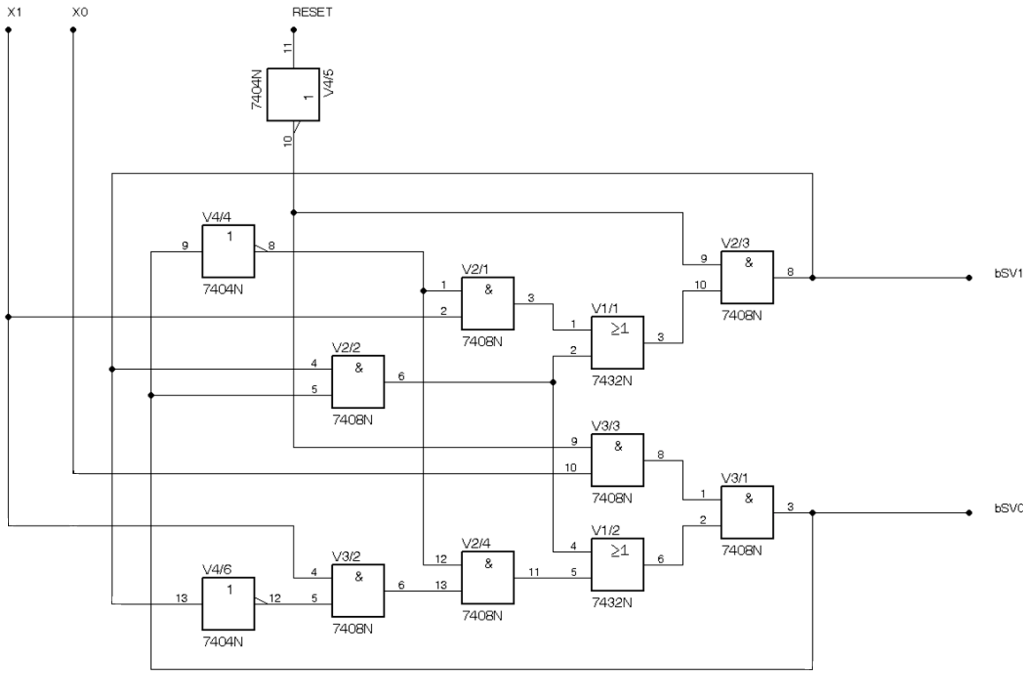


Bild 53: Mögliche Schaltung Eingangslogik

6.3.1. Zustandsregister

Aufgrund der Verzögerungen der Gatter können für die Variablen bSV1 und bSV0 falsche Belegungen auftreten. Diese „falschen Belegungen“ dürfen nicht weiterverarbeitet werden.

Das kann erreicht werden, wenn man zusätzlich zu der bereits vorhanden Zustandsvariablen SV (mit dem aktuellen Wert) eine Neuwert-Variable n_SV einführt und den Transport der Werte von der Neuwert-Variablen zur Variablen mit dem aktuellen Wert mit einem Ereignis (event) steuert.

Im Falle der Programmierbaren Chips (FPGA, CPLD) kann als „event“ die Änderung der Taktvariablen CLK (Systemtakt) interpretiert werden. Für die nachfolgenden Darstellungen wird vereinbart, dass mit 0/1-Flanke von CLK der Wert von n_SV in das Zustandsregister SREG übernommen wird.

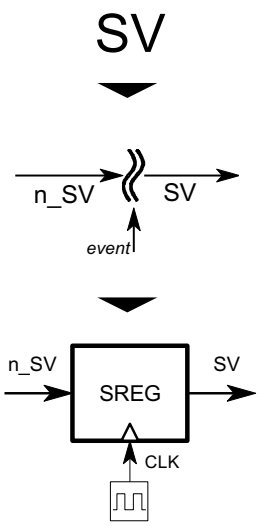


Bild 54: Neuwertsteuerung mit Zustandsregister

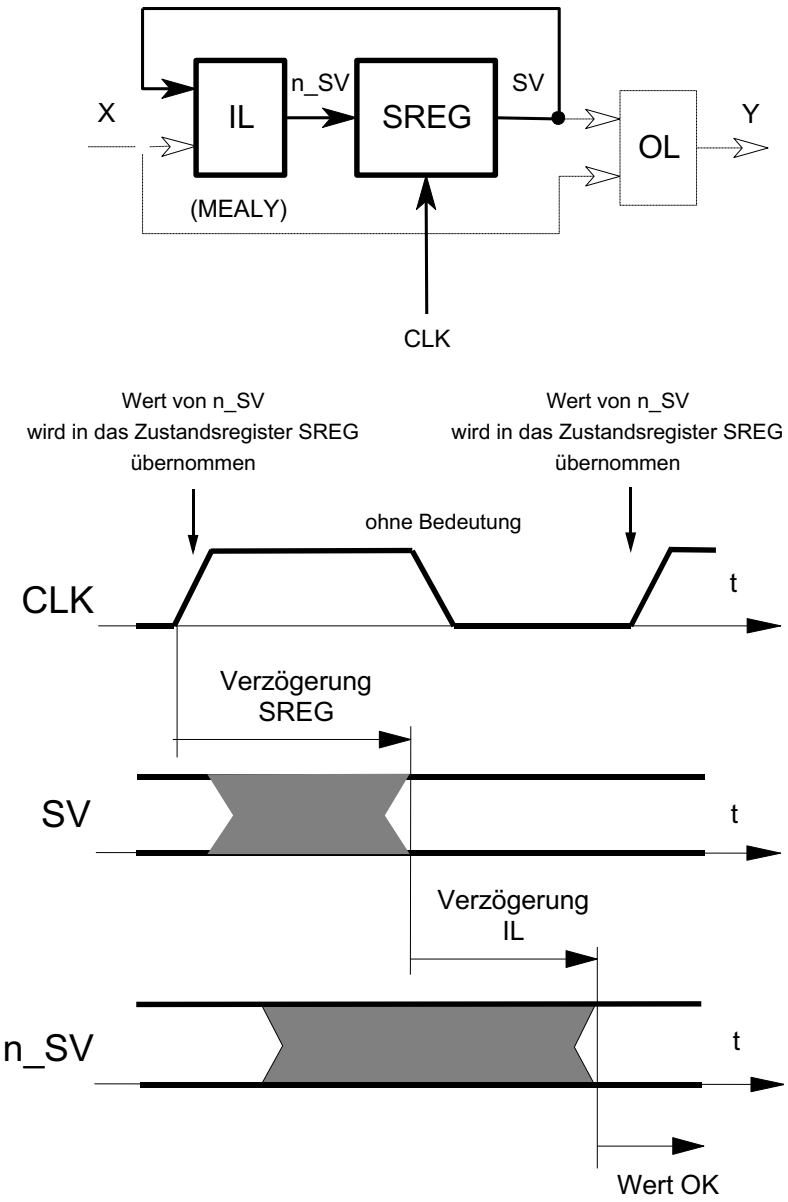


Bild 55: Zeitliches Verhalten mit Zustandsregister

Damit die Informationsverarbeitung „ständig“ (zyklisch) erfolgt, muss sich die Taktvariable zyklisch ändern. Dieses sich ändernde Signal wird von einem Taktgenerator erzeugt (Bild 56).

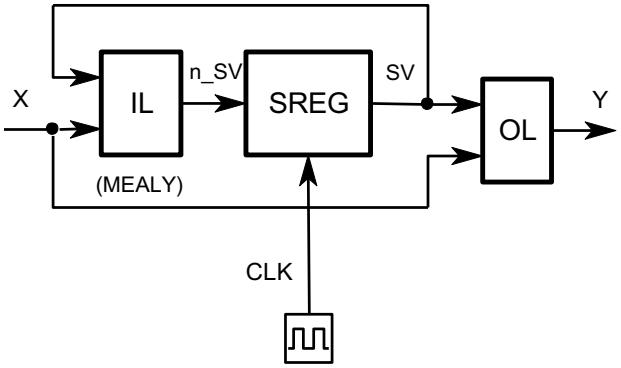


Bild 56: Erzeugung der Werte der Taktvariablen CLK mit Taktgenerator

Die minimale Periodendauer für die Frequenz der Taktvariablen tv gilt (Bild 57):

Periodendauer CLK ≥ Verzögerung SREG + Verzögerung IL

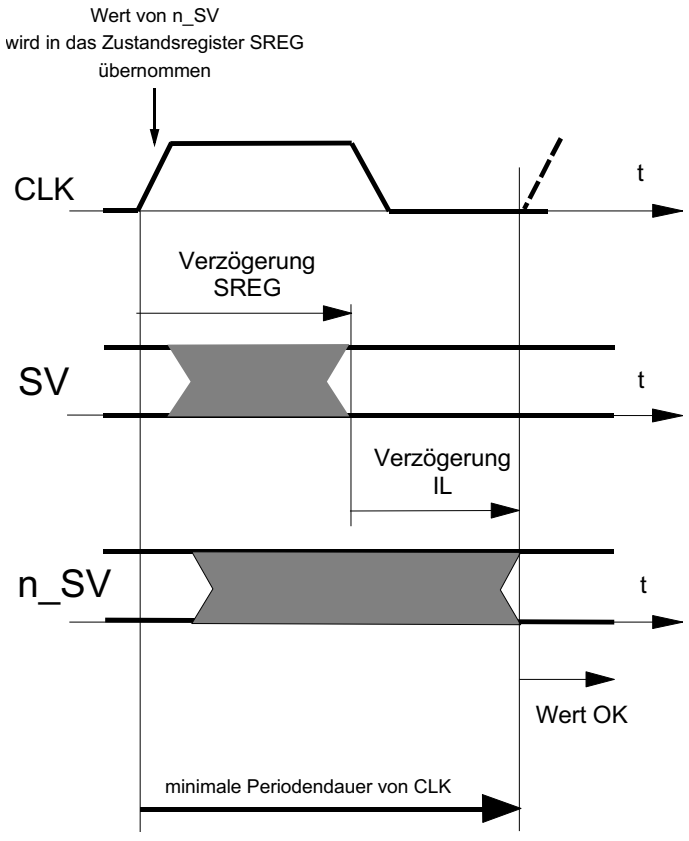
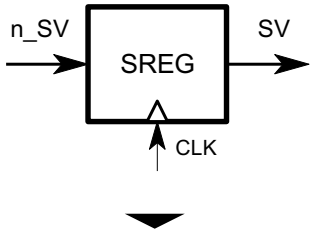


Bild 57: minimale Periodendauer von CLK

Das Zustandsregister wird mit dem Prozess SREG_PROC realisiert.
In diesem Prozess wird auch der Übergang in den Initialzustand für RESET = 1. realisiert.
Mit der 0/1-Flanke von CLK geht der Automat in den Folgezustand.




```
SREG_PROC: process (n_SV, CLK, RESET) -- Zustandsregister
 begin     -- State Register SREG
 if (RESET = '1')   -- RESET =1 :
 then SV <= ST_0;   -- Initialzustand
 else
 if rising_edge (clk) then SV <= n_SV; -- wenn CLK 0/1-Änderung:
 end if;    -- SV := n_SV;
 end if;
 end process;
```

Programm 6-2: VHDL-Programm, Prozess Zustandsregister SREG

Die Taktfrequenz (Frequenz von CLK) muss so gewählt werden, dass die Periodendauer größer als die Verzögerungen von SREG und IL/OL ist.

$$1 / f_CLK > tv_IL_OL + tv_SREG$$

$$f_CLK < 1 / (tv_IL_OL + tv_SREG)$$

Gl. 6-2

f_CLK	Taktfrequenz, Systemtakt
tv_IL_OL	Verzögerung Eingangslogik / Ausgangslogik
tv_SREG	Zustandsregister

.

6.3.2. Voreinstell- und Haltezeit

Das Entwicklungssystem für die Programmierbaren Chips generiert eine Schaltung ähnlich Bild 58.
Das Zustandsregister wird hier mit den beiden D-Flip-Flops V2/1 und V2/2 realisiert.

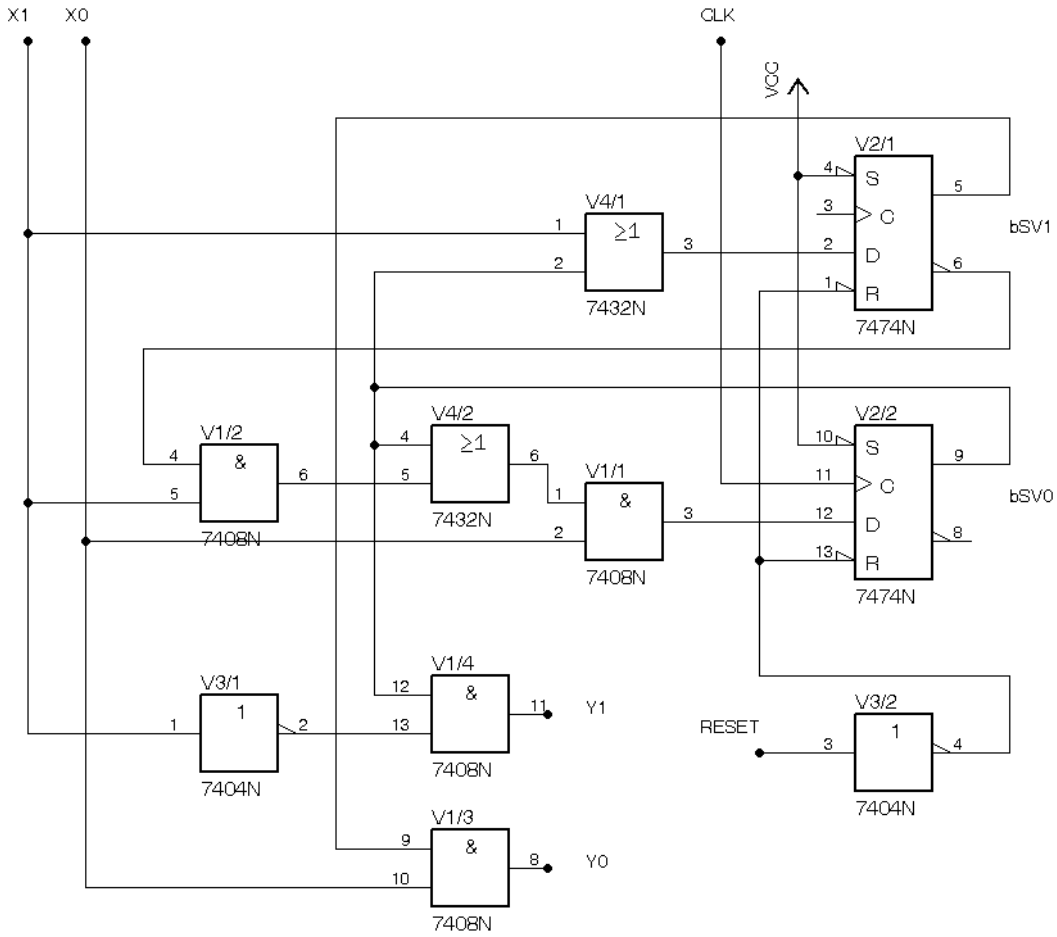


Bild 58: Generierte Schaltung mit Zustandsregister

Die D-Flip-Flops übernehmen mit der 0/1-Flanke der Taktvariablen CLK die Belegung an den D-Eingängen. Mit dieser Flanke schalten die Flip-Flops - es wird ein Zustandsübergang realisiert.

Während der Übernahme der Signale in das Zustandsregister SREG (Flip-Flops) muss die Belegung der Variablen konstant gehalten werden (Bild 59). Da sich die Eingangsvariablen zu jedem Zeitpunkt ändern können ist dies nicht gewährleistet

Es besteht die Gefahr, dass falsche Zustandsübergänge realisiert werden.
Das kann verhindert werden, wenn Eingangsregister eingeführt wird.

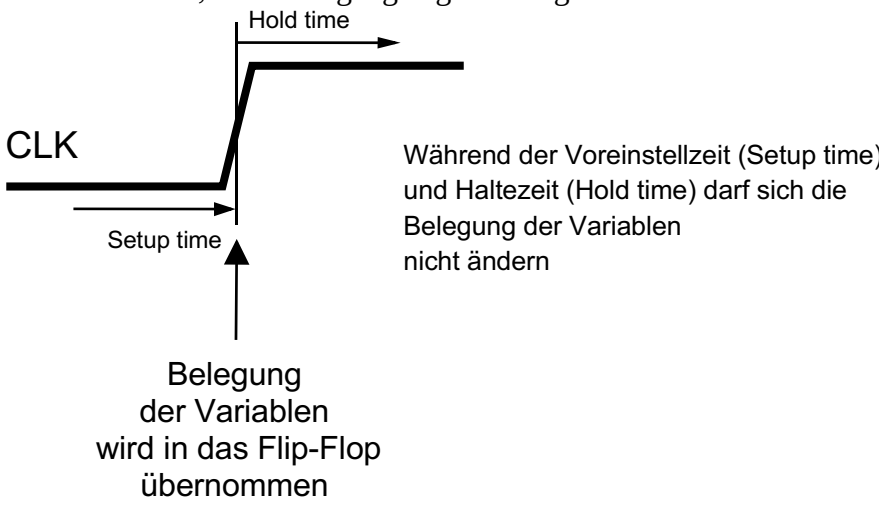


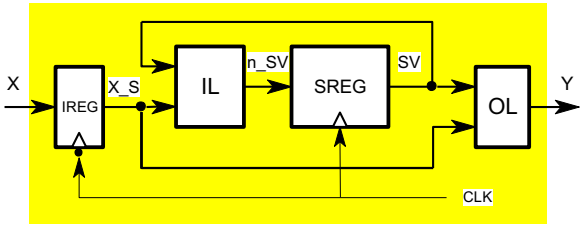
Bild 59: Voreinstell- und Haltezeit

			V_{CC}	T_A = 25°C		SN54HC74		SN74HC74		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
f_{clock}	Clock frequency		2 V	0	6	0	4.2	0	5	MHz
			4.5 V	0	31	0	21	0	25	
			6 V	0	36	0	25	0	29	
t_w	Pulse duration	$\overline{PRE}$ or $\overline{CLR}$ low	2 V	100		150		125		ns
			4.5 V	20		30		25		
			6 V	17		25		21		
		CLK high or low	2 V	80		120		100		
			4.5 V	16		24		20		
			6 V	14		20		17		
t_{su}	Setup time before CLK↑	Data	2 V	100		150		125		ns
			4.5 V	20		30		25		
			6 V	17		25		21		
		$\overline{PRE}$ or $\overline{CLR}$ inactive	2 V	25		40		30		
			4.5 V	5		8		6		
			6 V	4		7		5		
t_h	Hold time, data after CLK↑		2 V	0		0		0		ns
			4.5 V	0		0		0		
			6 V	0		0		0		

Setup time: Voreinstellzeit
Hold time: Haltezeit

*Tabelle 23: Dynamische Kennwerte SN*HC74 /74HC74_DS/*

Die Einhaltung der Voreinstell- und Haltezeit kann durch ein Eingangsregister gewährleistet werden.



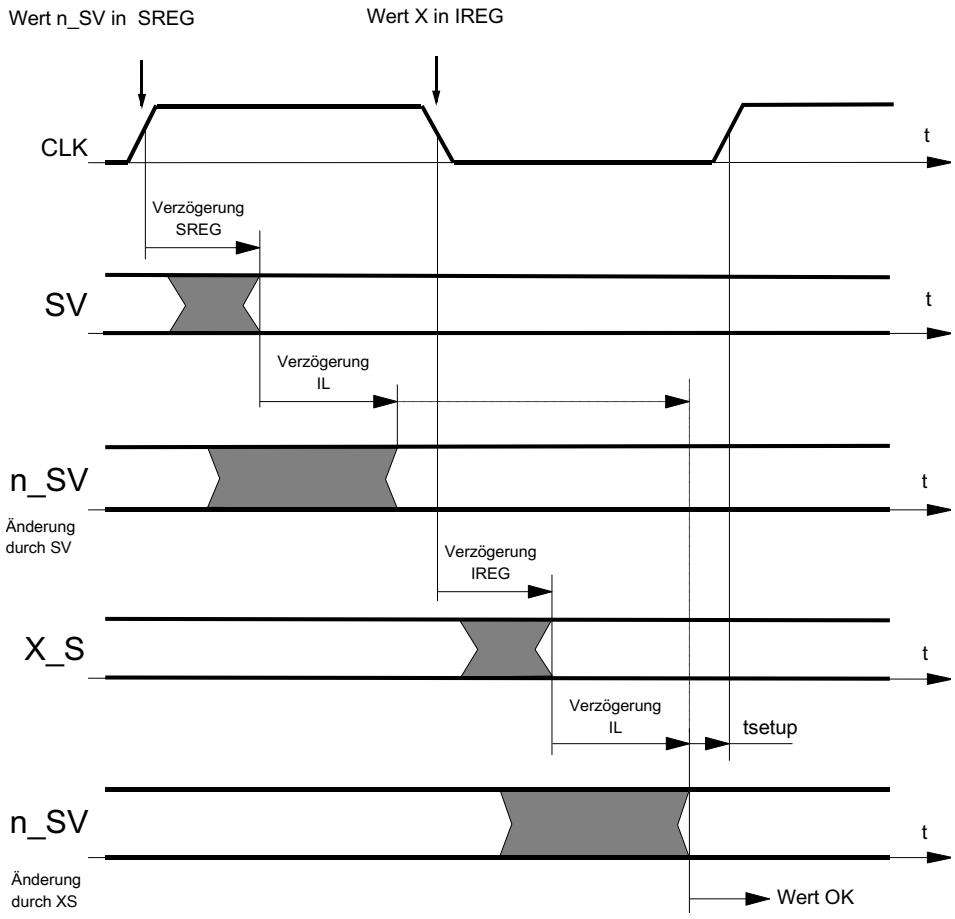


Bild 60: zeitlicher Ablauf Zustandsregister, Eingangsregister

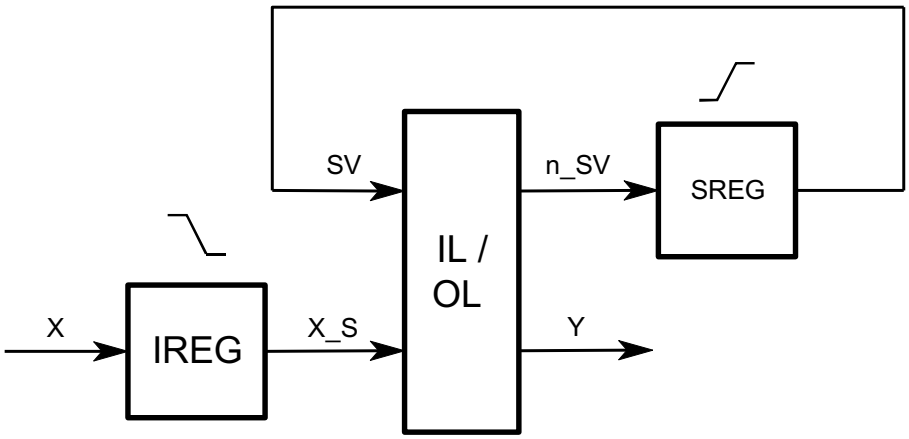


Bild 61: Taktregime Zustandsregister, Eingansregister

Die Taktfrequenz (Frequenz von CLK) muss so gewählt werden, dass die halbe Periodendauer größer als die Verzögerung der Kombination aus Eingangs- und Ausgangslogik ist.

$$1 / f_CLK * 2 > tv_IL_OL$$

$$f_CLK < 1 / tv_IL_OL * 2$$

Gl. 3

F_CLK: Taktfrequenz, Systemtakt; tv_IL_OL: Verzögerung Eingangslogik / Ausgangslogik

Wird ein Mealy oder Moore-Automat als Automatenmodell benutzt, dann könnte eine höhere Taktfrequenz möglich sein. Die Verzögerung der Eingangslogik ist wahrscheinlich geringer als die der Kombination aus Eingangs- und Ausgangsregister.

6.3.3. Master-Slave-Zustandsregister

Auf der Taktleitung (Variable CLK) im Bild 62 treten in Programmierbaren Chips nicht vernachlässigbare Verzögerungen auf.
Das kann dazu führen, dass die Flip-Flops teilweise die neuen Belegungen der booleschen Zustandsvariablen (andere Flip-Flops „haben bereits geschaltet“) mit der 0/1-Flanke verarbeiten. Das kann zu falschen Zustandsübergängen führen.

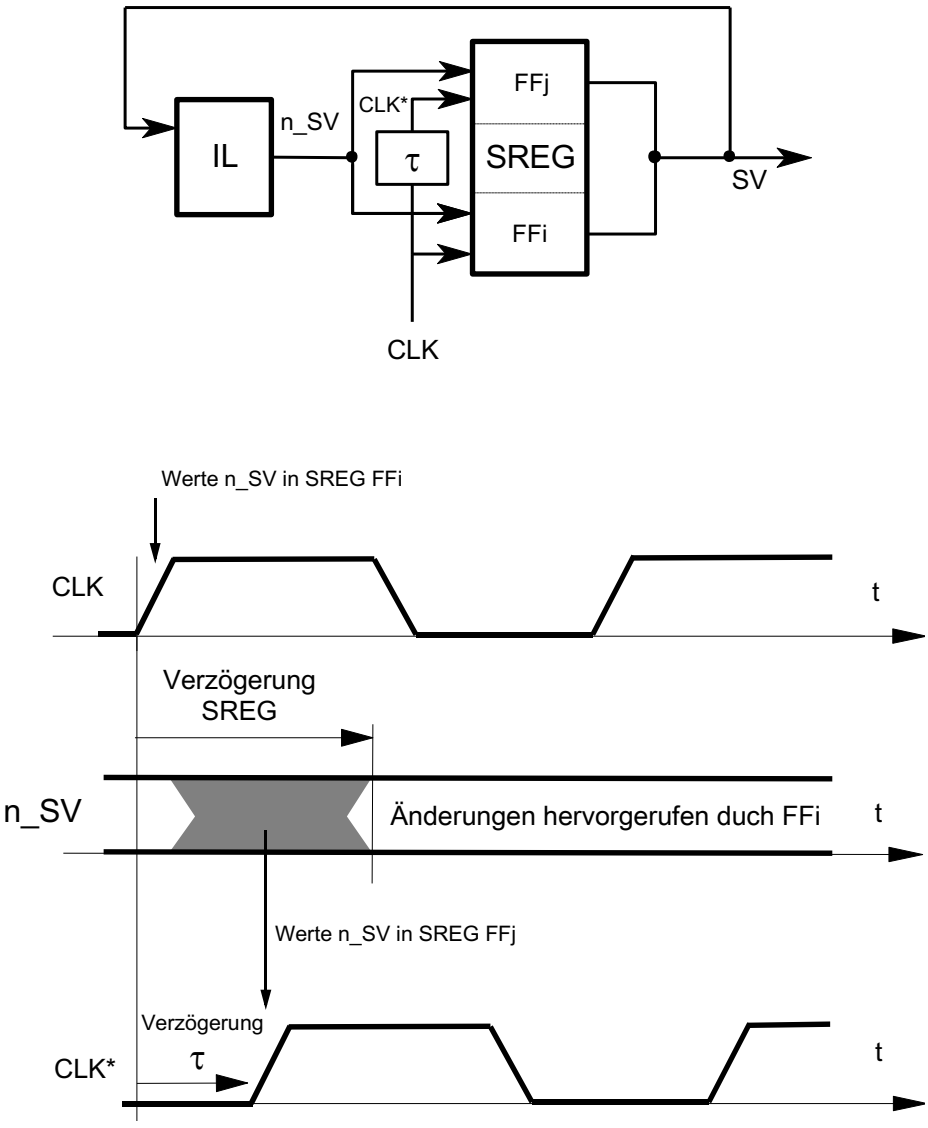


Bild 62: Verzögerungen auf der Taktleitung

Deshalb wird das Zustandsregister durch einen Master- und einen Slave-Teil realisiert (Master-Slave-Zustandsregister).

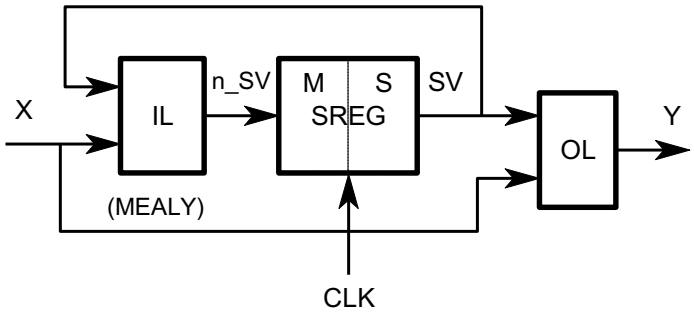


Bild 63: Master-Slave-Zustandsregister

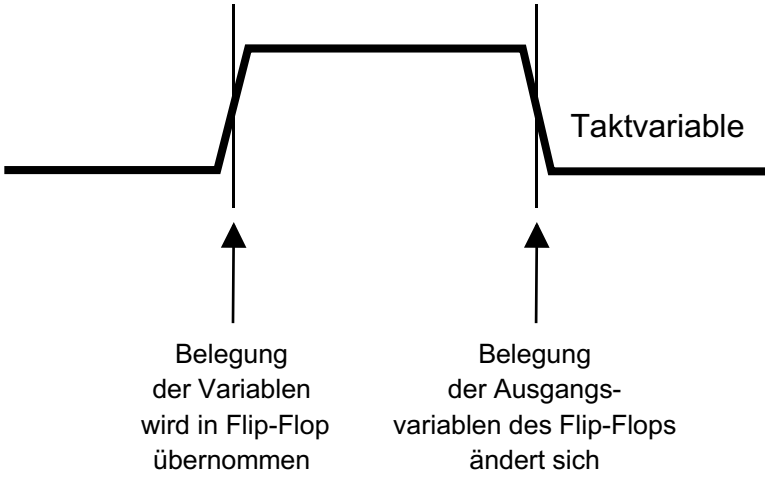


Bild 64: Arbeitsweise Master-Slave-Zustandsregister

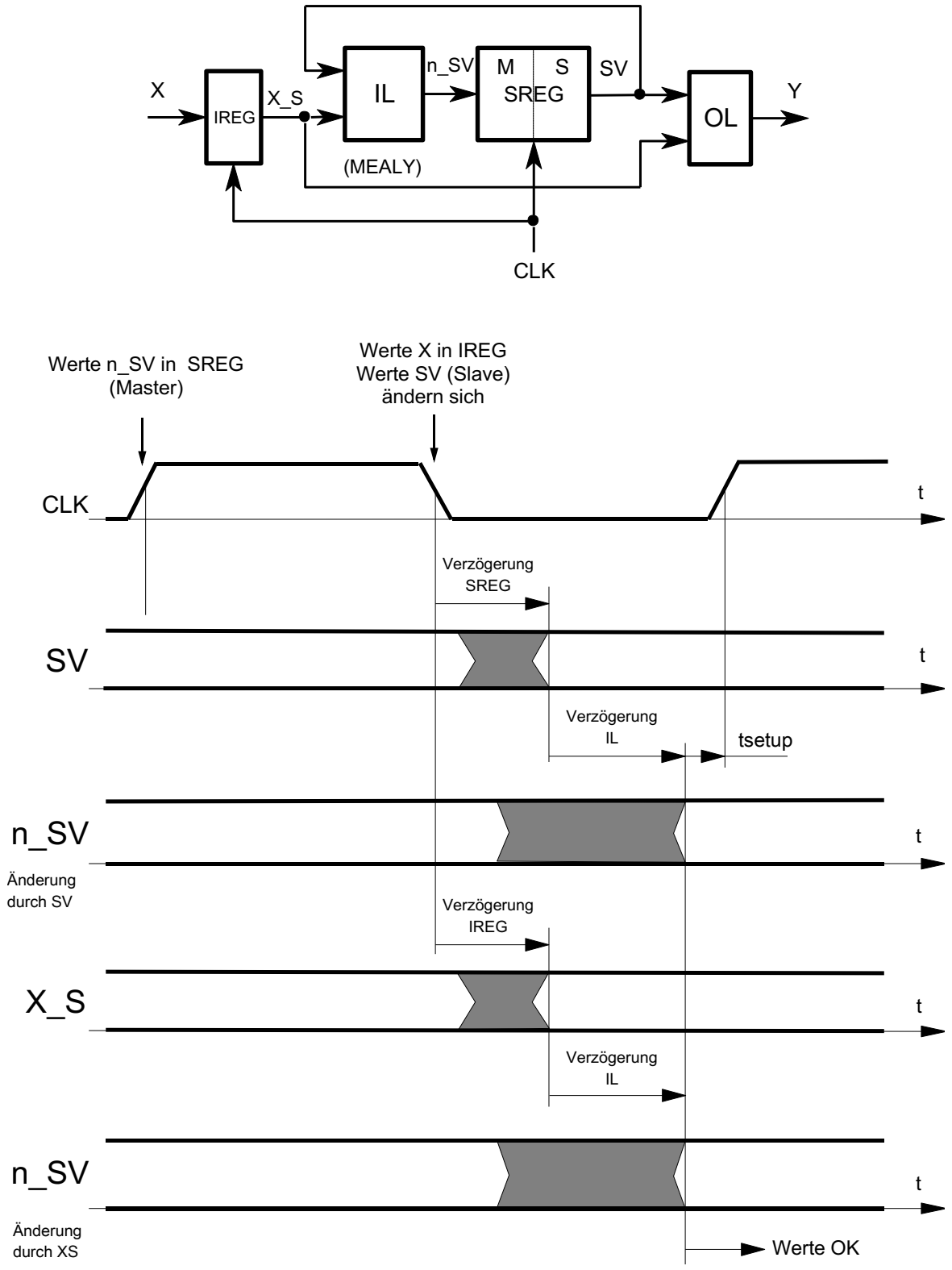
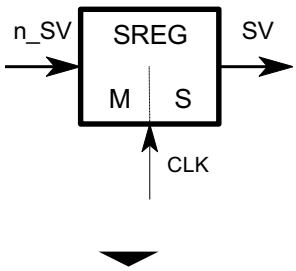


Bild 65: Zeitliche Verläufe Master-Slave-Zustandsregister, Eingangsregister

6.3.4. VHDL-Programm Master-Slave-Zustandsregister

Der Prozess SREG_PROC wird modifiziert.

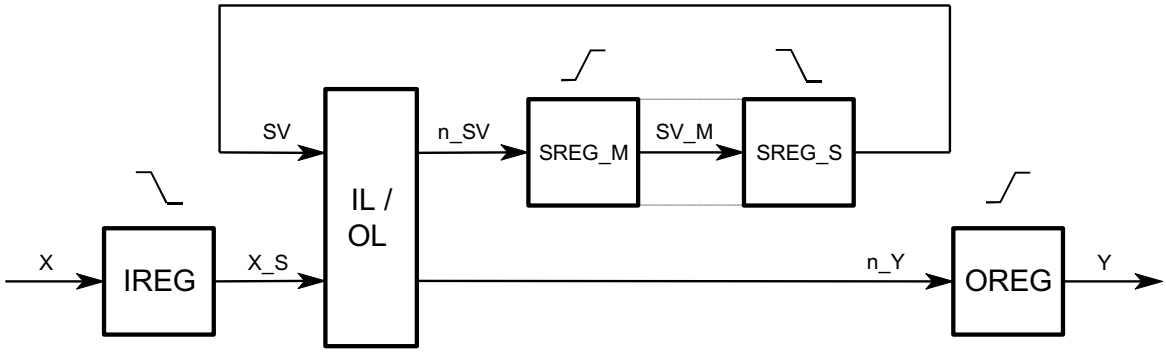



```
SREG_PROC: process (n_SV, CLK, RESET) -- Master-Slave-Zustandsregister

 variable  SV_Master: TYPE_STATE;  -- Zustandsvariable, Master

 begin

 -- State Register SREG, Master
 if (RESET = '1')        -- RESET =1 :
 then SV_Master := ST_0;      -- Initialzustand
 else
 if rising_edge (clk) then SV_Master := n_SV; -- wenn CLK 0/1-Änderung:
 end if;          -- SV_Master := n_SV;
 end if;

 -- State Register SREG, Slave
 if (RESET = '1')        -- RESET =1 :
 then SV <= ST_0;        -- Initialzustand
 else
 if falling_edge (clk) then SV <= SV_Master; -- wenn CLK 0/1-Änderung:
 end if;          -- SV := SV_Master;
 end if;

 end process;
```

Programm 3: VHDL-Programm, Master-Slave-Zustandsregister

6.3.5. Ausgangsregister

Am Ausgang (Y) können ebenfalls falsche Belegungen auftreten. Damit eine angeschlossene Einheit sich nicht falsch verhält, kann dieser die Taktvariable CLK zur Verfügung gestellt und die Einheit damit synchronisiert werden.
Eine andere Möglichkeit ist Verwendung eines Ausgangsregisters.



IREG	Input Register Eingangsregister
IL / OL	Input / Output Logic Eingangs- / Ausgangslogik
SREG_M	State Register Master Zustandsregister Master
SREG_S	State Register Slave Zustandsregister Slave
OREG	Output Register Ausgangsregister

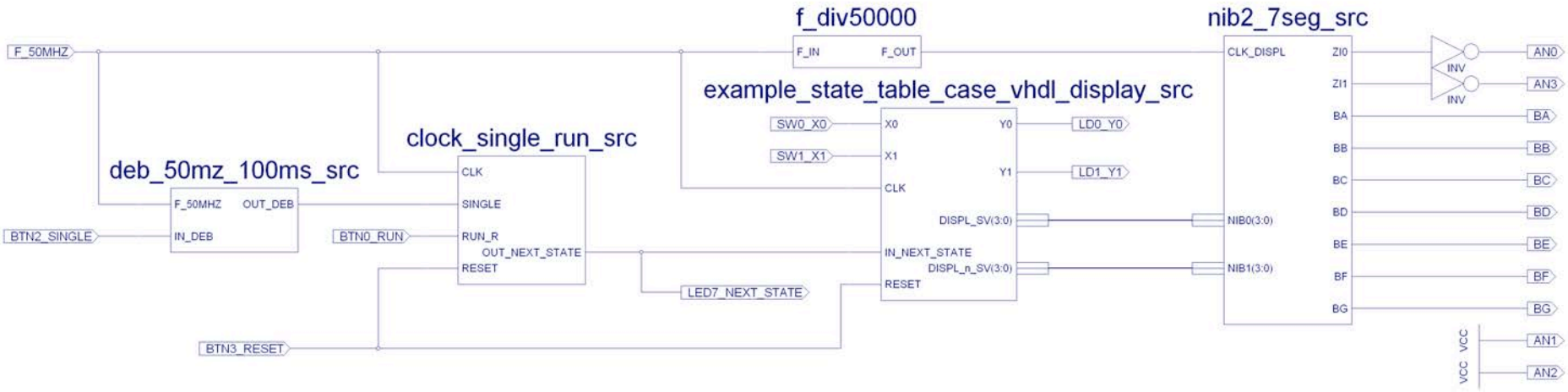
X	Input Variables Eingangsvariablen
X_S	Input Variables stored Eingangsvariablen gespeichert
SV	State Variables (State Variables Slave) Zustandsvariablen (Zustandsvariablen Slave)
n_SV	State Variables new Value Zustandsvariablen neuer Wert
SV_M	State Variables Master Zustandsvariablen Master
Y	Output Variables Ausgangsvariablen
n_Y	Output Variables new Value Ausgangsvariablen neuer Wert

⎍	information processing by rising edge of clock variable Informationsverarbeitung mit aufsteigender Flanke der Taktvariablen
⎎	information processing by falling edge of clock variable Informationsverarbeitung mit fallender Flanke der Taktvariablen

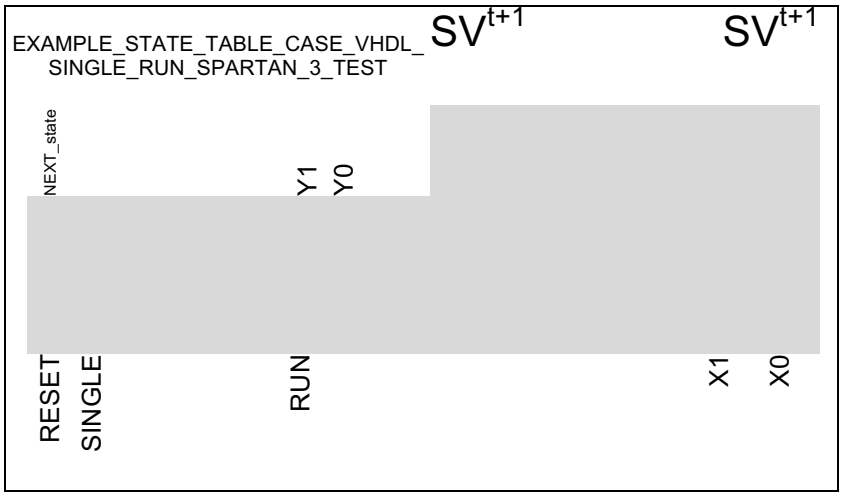
Bild 66: Taktregime, Huffman-Automat mit Eingangsregister, Zustandsregister /Master-Slave und Ausgangsregister

7. Simulation ?

8. Inbetriebnahme



SV	X1	X0	Y1	Y0	^{t}SV
ST_0	0	0	0	0	ST_0
	0	1	0	0	ST_0
	1	0	0	0	ST_2
	1	1	0	0	ST_1
ST_1	0	0	1	0	ST_2
	0	1	1	1	ST_1
	1	0	0	0	ST_2
	1	1	0	1	ST_1
ST_2	0	0	0	0	ST_0
	0	1	0	1	ST_0
	1	0	0	0	ST_2
	1	1	0	1	ST_2



9. Warum sollte ich einen Konfigurierbaren Chip einsetzen ?

9.1. Vorteile gegenüber klassischen (TTL-) Lösungen

- auch sehr komplexe Algorithmen mit einen Chip realisierbar
- Vervielfältigung der Bausteine beim Anwender möglich
- Datentyp "Aufzählung" verfügbar

  Damit ist keine binäre Zustandscodierung mit allen Folgeprobleme wie u. a. nicht beschriebene Zustände notwendig.

9.2. Vorteile gegenüber Mikroprozessoren

- parallele Realisierung der Algorithmen möglich
- Zustandsübergang in allen Automaten mit einer Flanke des Systemtaktes möglich
- Die Blöcke werden verbindungsprogrammiert parallel realisiert
- Datenaustausch zwischen den Blöcke ist verbindungsprogrammiert realisiert
  Es wird keine Rechenleistung benötigt.
- kein Betriebssystem erforderlich
  Der Zugang von Dritten ist damit erschwert
- Hardwaremäßig getrennte Programmierschnittstelle
  Nur über diese Schnittstelle können die implementierten Algorithmen geändert werden

9.3. Probleme

- Die meisten Entwickler können nur in "Mikroprozessoren denken"
- Die parallele Realisierung von Algorithmen wird durch die Herstellern der konfigurierbaren Chips praktisch nicht unterstützt

10. Anhang

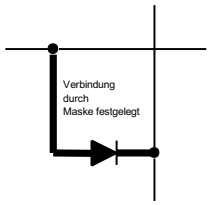
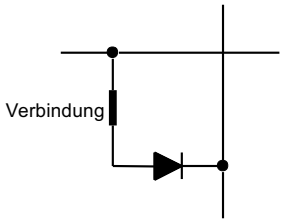
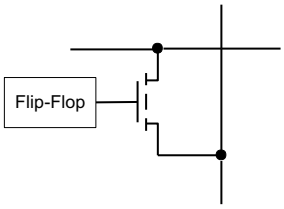
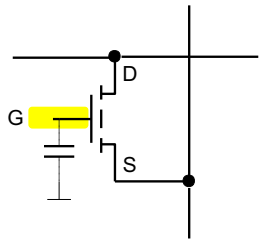
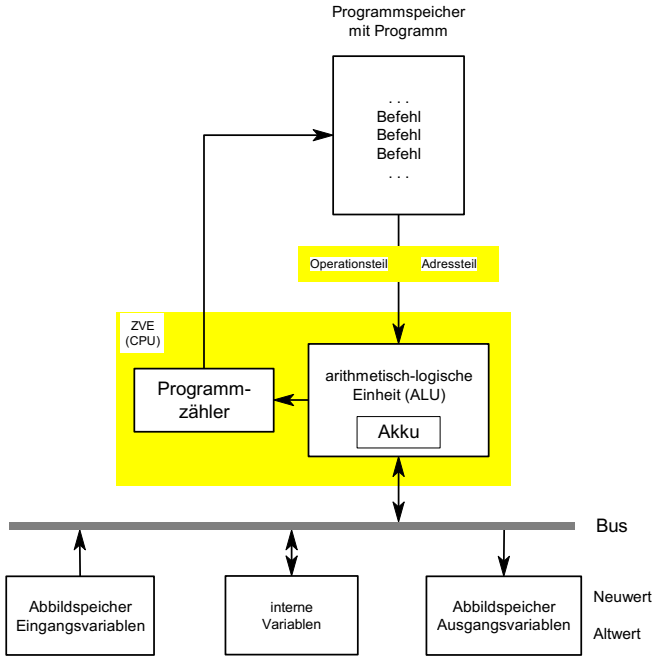
Technik		Anwendung (Beispiele)
	<u>Maskenprogrammierung</u> Die Elemente werden beim Hersteller des Schaltkreises nach Vorgaben des Kunden verbunden. Die Verbindungen bleibt bei Spannungsausfall erhalten	ASIC
	<u>Fuse-Technik, Anti-Fuse-Technik</u> Einmalige Programmierung beim Anwender Die Verbindungen bleibt bei Spannungsausfall erhalten	PLD PLA, PAL
	<u>RAM-Zellen</u> Programmierung beim Anwender (beliebig oft) Umprogrammierung während des Betriebs möglich Die Signale zu den Verbindungen gehen bei Spannungsausfall verloren	FPGA
	<u>(E) EPROM-Zelle</u> Flash Programmierung beim Anwender (<10^6) Die Verbindungen bleibt bei Spannungsausfall erhalten	CPLD

Tabelle 24: Verfahren für die interne Verbindungsprogrammierung



Beispiel:

Eingangsvariable & interne_Variable =: Ausgangsvariable

SPS- Programm:

Operationsteil OT	*Adressteil AT*	*Kommentar*
LD	Eingangsvariable	Wert Eingangsvariable (aus Eingangsabbildspeicher) in Akku
AND	interne_Variable	Wert interne Variable in SVE, Wert Akku := Wert Akku & Wert interne Variable
ST	Ausgangsvariable	Wert Akku in Ausgangsabbildspeicher (Neuwert)

Bild 67:Beispiel für serielle Realisierung eines Algorithmus

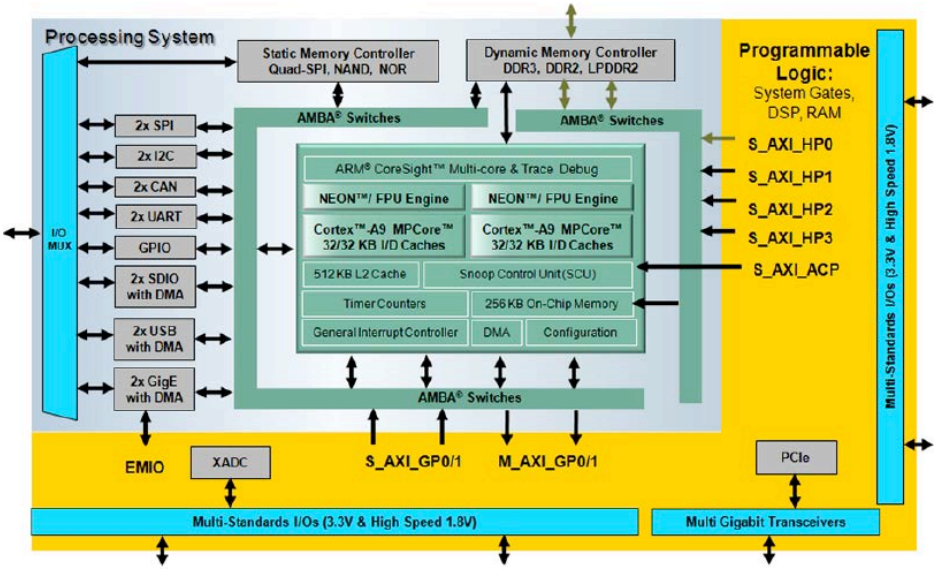
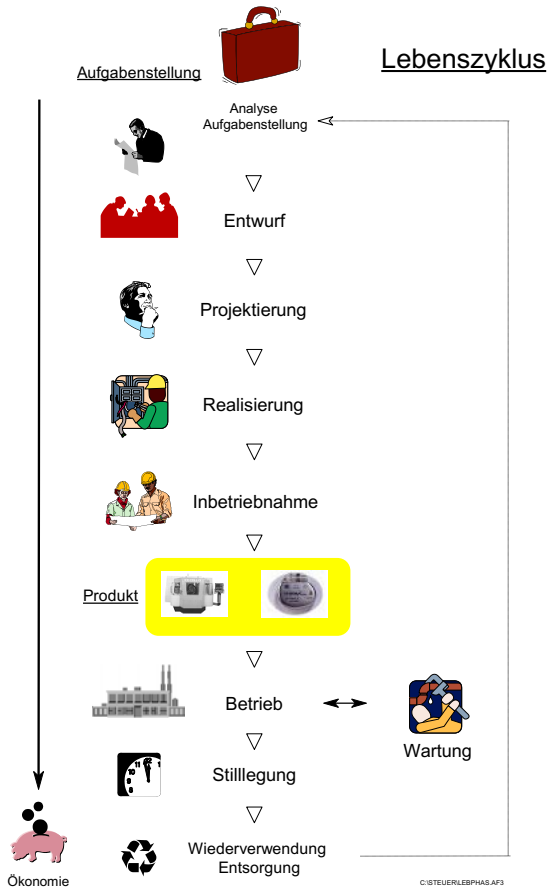


Bild 68: Beispiel für FPGA mit Mikroprozessoren /ZYNQ7000_01/



Analyse Aufgabenstellung

- Anforderungen

Entwurf

- Informationsfluss SVE / Umgebung
- Variablendefinition
- Strukturbeschreibung
- Verhaltensbeschreibung
- Eigenschaften SVE

Projektierung

- Realisierungsvariante
- Vorschriften für die nachfolgenden Phasen
- Eigenschaften der realisierten SVE

Realisierung

- System realisiert und geprüft

Inbetriebnahme *Verifikation SVE, System*

- betriebsbereites System

Betrieb

- Realisierung Anforderung

Stilllegung

- zerlegbares System

Wiederverwendung Entsorgung

- entsorgte Teile, wieder verwendbare Teile des Systems

Anmerkung: In allen Phasen werden Festlegungen getroffen

Tabelle 10-2: Ergebnisse Phasen

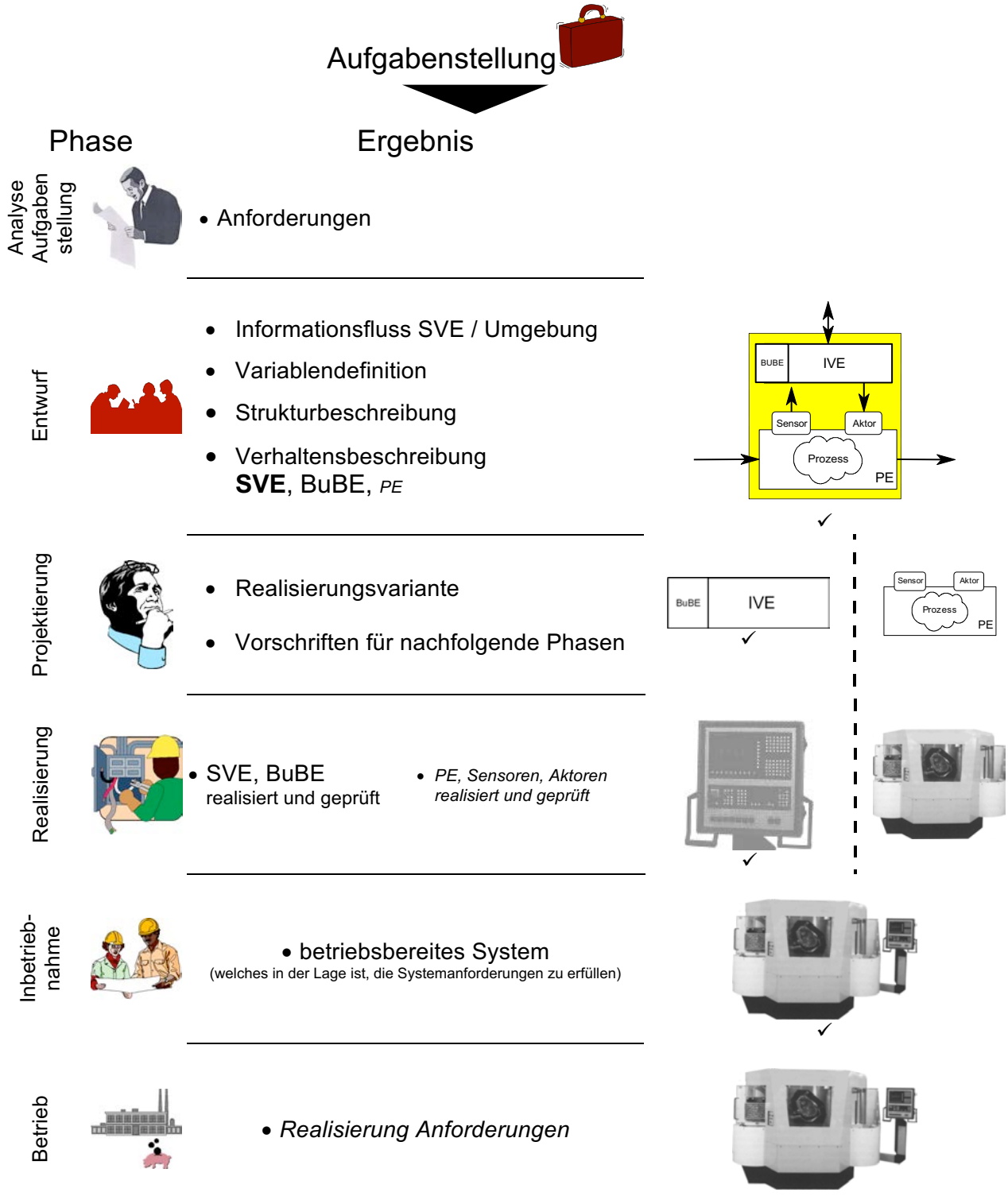


Bild 69: Ergebnis Phasen, Beispiel Werkzeugmaschine

11. Quellen

/74HC00_DS/ SN54HC00, SN74HC00
4 NAND-Gatter mit je 2 Eingängen
Institution: Firma Texas Instruments
www.ti.com
Sprache: englisch

/74HC74_DS/ SN54HC74, SN74HC74
2 D-Flip-Flops, SET und RESET-Eingang, flankengetriggert (0/1-Flanke)
Institution: Firma Texas Instruments
www.ti.com
Sprache: englisch

/FPGA_KOCHBUCH/ Markus Wannemacher

Das FPGA-Kochbuch
Verlag: International Thomson Publishing
ISBN: 3-8266-2712-1
www.aufzu.de/fpga/kochbuch

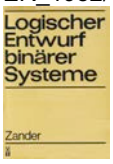
/GLOSSWANNFPGA/ Volker Pfeiffer
Glossar
Glossar zu Wann sollte man einen FPGA anwenden ?

/KILLENB_1977/ Harald Killenberg, Michael Krapp, Klaus Flurschütz
Struktureller Entwurf industrieller Steuerungen, insbesondere von Werkzeugmaschinen, mit Hilfe von Programmablaufgraphen
Entwurfsrichtlinie.
Institution: Werkzeugmaschinenkombinat "7. Oktober", Berlin 1977

/VHDL_DESIGN/ Gunther Lehmann, Bernhard Wunder, Manfred Selz
Schaltungsdesign mit VHDL
Verlag: Franzis' Verlag, Poing, 1994
ISBN: 3-7723-6163-3
Sprache: englisch

/WUTTKE_2003/ Heinz-Dietrich Wuttke, Karsten Henke

Schaltsysteme
Eine automatenorientierte Einführung
Verlag: Pearson Education Deutschland GmbH
ISBN: 3-8273-7035-3

/ZANDER_1982/ Hans - Joachim Zander

Logischer Entwurf binärer Systeme
Verlag: VEB Verlag Technik, Berlin, 1982

/ZYNQ7000_01/ Zynq-7000 All Programmable SoC
Technical Reference Manual
Institution: Firma XILINX
www.xilinx.com
Sprache: englisch

12. Begriffe

Algorithmus
{m} ist die Vorschrift, nach der beginnend mit einem Anfangszustand ausgehend von einer Folge an den Eingängen einer Einheit eine Folge an den Ausgängen dieser Einheit erzeugt wird

Algorithmus, Ebene Belegungen von Variablen
{m} ist die Vorschrift, nach der beginnend mit einem Anfangszustand ausgehend von einer Folge von Belegungen der Eingangsvariablen einer Einheit eine Folge von Belegungen der Ausgangsvariablen dieser Einheit erzeugt wird.
Den Werten der Eingangsvariablen können Informationen aus der Umgebung und den Werten der Ausgangsvariablen können Anweisungen für die Umgebung zugeordnet sein

Algorithmus, Ebene Informationen / Anweisungen
{m} ist die Vorschrift, nach der beginnend mit einem Anfangszustand ausgehend von einer Folge von Informationen aus der Umgebung an den Eingängen einer Einheit eine Folge von Anweisungen an die Umgebung an den Ausgängen dieser Einheit erzeugt wird.

Ausgangsvariable
{f} >>>Variable einer >>>Einheit, deren Werte in der Einheit erzeugt werden und an die Umgebung weiterggeben werden. Den Werten können Anweisungen an die Umgebung zugeordnet sein.

Belegung
{f} Zuordnung von Werten zu einer Variablenmenge

Bezeichner
{m} Zeichenkette, die einem Sprachelement einer Programmiersprache zugeordnet ist, dieses bezeichnet.

Complex Programmable Logic Device
Abk.: CPLD
{m} intern >>> verbindungsprogrammierbare Einheit in Form eines integrierten Schaltkreises. Gegenüber einem >>> FPGA hat ein CPLD meistens weniger Elemente und eine geringere Komplexität. Gehört zur Gruppe der >>> konfigurierbaren Chips.
wörtlich: komplexe programmierbare logische Einheit

CPLD
>>>Complex Programmable Logic Device

Eingangsvariable
{f} >>>Variable einer >>>Einheit, deren Werte in der Umgebung erzeugt werden und durch die Einheit ausgewertet werden können. Den Werten können Informationen aus der Umgebung zugeordnet sein

Einheit
{f} Eine Einheit ist der Teil der unter einer gegebenen Betrachtungsweise als Ganzes betrachtet wird und damit abgrenzbar gegenüber der Umgebung ist.

Field Programmable Gate Array mit Mikroprozessoren
>>>Field Programmable Gate Array with Microprocessors

Field Programmable Gate Array with Microprocessors
enthält neben dem >>> intern verbindungsprogrammierbaren Bereich auch Mikroprozessoren

Field Programmable Gate Array
>>> intern verbindungsprogrammierbare Einheit in Form eines integrierten Schaltkreises. Gehört zur Gruppe der >>> konfigurierbaren Chips.
wörtlich:
im Feld, vor Ort programmierbares Gatterfeld

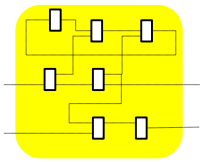
FPGA mit Mikroprozessoren
>>>Field Programmable Gate Array with Microprocessors

FPGA
>>>Field Programmable Gate Array

intern verbindungsprogrammierbar
die Auswahl der >>> Algorithmen aus der Menge der möglichen Algorithmen wird durch Herstellen oder Trennen von internen Verbindungen realisiert

intern verbindungsprogrammierbare Einheit

{f} >>> Einheit, mit der Eigenschaft, dass die Auswahl der >>> Algorithmen aus der Menge der möglichen Algorithmen durch Herstellen oder Trennen von internen Verbindungen realisiert wird.
siehe auch: Konfigurationsdatei



interne Verbindungsprogrammierung

{f} Die Auswahl der >>> Algorithmen aus der >>> Menge der möglichen Algorithmen wird durch Herstellen oder Trennen von Verbindungen innerhalb einer Einheit realisiert.

Konfigurationsdatei

{f} enthält die Vorschriften für die Verbindungen einer >>> intern verbindungsprogrammierbaren Einheit. Die Konfigurationsdatei wird mit einem Entwicklungssystem erzeugt und über eine Schnittstelle (z. B. JTAG-Schnittstelle) in die intern verbindungsprogrammierbare Einheit eingespielt.
siehe auch: WebPACK

konfigurierbarer Chip

Abk.: CCHIP
{m} Schaltkreis, der eine intern verbindungsprogrammierbare Einheit auf seinem Chip enthält

korrekte Signale verarbeitende Einheit

{f} Eine >>>Signale verarbeitende Einheit, die die Algorithmen genau so realisiert wie mit den Beschreibungsmitteln beschrieben, spezifiziert.

Mikroprozessor

{m} Ein Mikroprozessor ist die auf einem Chip realisierte zentrale Verarbeitungseinheit eines Rechners

Parallele Realisierung Algorithmus

Für die gleichzeitige Realisierung eines durch die Beschreibungsmittel beschriebenen Zustandsübergangs in allen (parallelen) Automaten benötigt die Signale verarbeitende Einheit genau einen Zustandsübergang.

programmierbare Einheit

{f} >>> Einheit, mit der Eigenschaft, dass bezüglich der Menge der >>> Algorithmen, die diese Einheit realisieren kann, eine Auswahl möglich ist

Programmierung

{f} Auswahl von >>> Algorithmen aus der >>> Menge der möglichen Algorithmen
siehe auch: Verbindungsprogrammierung, Speicherprogrammierung

Serielle Realisierung Algorithmus

Für die gleichzeitige Realisierung eines durch die Beschreibungsmittel beschriebenen Zustandsübergangs in allen (parallelen) Automaten benötigt die Signale verarbeitende Einheit nicht immer nur einen Zustandsübergang..

Signale verarbeitende Einheit

{f} Eine >>>Einheit die beginnend mit einem Anfangszustand ausgehend von einer Folge von Belegungen der Eingangsvariablen X eine Folge von Belegungen der Ausgangsvariablen Y erzeugt.

Speicherprogrammierung

{f} Die Auswahl der >>> Algorithmen aus der >>> Menge der möglichen Algorithmen wird durch die Belegungen eines Speichers realisiert.

SVE

>>>Signale verarbeitende Einheit

Umgebung

{f} alles außerhalb einer Einheit

Variable

{f} ist bestimmt durch
- Namen (>>>Bezeichner, >>>Zeichenkette)
- >>>Datentyp

Verbindungsprogrammierung

{f} Die Auswahl der >>> Algorithmen aus der >>> Menge der möglichen Algorithmen wird durch Herstellen oder Trennen von Verbindungen realisiert. Diese Art der Programmierung wird auch als Konfigurieren bezeichnet.

WebPACK

Entwicklungssystem der Firma XILINX für konfigurierbare Chips

13. Quellen zu den Wissengebieten CPLD, FPGA

/ADEPT_USER/

Adept Application User's Manual
Firma Digilent

/AUER_1994/

von Adolf Auer
Programmierbare Logik-IC
Eigenschaften, Anwendung und Programmierung von PLD und FPGA
Hüthig Fachverlage 1994
ISBN 3-7785-2276-0

/AUER_1995A/

Adolf Auer, D. J. Rudolf
FPGA
Feldprogrammierbare Gate Arrays
Hüthig Fachverlage 1995
ISBN 3-7785-2359-7

/BASYS2_100K_DS/

Basys2 100K - XILINX Spartan-3E FPGA

/BERMBACH/

R. Bermbach
Labor Datentechnik I
Laborversuch FPGA
Ostfalia Hochschule für angewandte Wissenschaften
Fakultät Elektrotechnik
Labor Datentechnik
Prof. Dr.-Ing. Rainer Bermbach

/BLANK_1992/

Hans Joachim Blank
Logikbausteine – Grundlagen, Programmierung und Anwendung
PLDs, PALs GALs und FPGAs mit dem PC programmieren
Markt & Technik Verlag AG 1992

/CHEAP_FPGA_BOARDS/

Cheap FPGA Development Boards
joel.id.au/fpga/cheapfpgadevelopmentboards

/CMOD295_SCH/

Digilent C-Mod C295 board schematic
Digilent Inc.
Bemerkung: alternativ mit XC2C64 oder XC9572XL bestückt

/CMODA7_DOC/

Cmod A7: Breadboardable Artix-7 (35T) FPGA Module
Firma Digilent

/CMODA7_REF/

Cmod A7 Reference Manual
Firma Digilent

/CMODA7_SCH/

Cmod A7 Schematic
Firma Digilent

/CONF_COMP/

Christian Siemers
Configurable Computing
V1.01 SS2005
Institut für Informatik der Technischen Universität Clausthal

/COOL_RUN_I2C/

Cool Runner CPLD I²C Bus Controller Implementation
Firma Xilinx
Sprache: englisch

/COOL_RUN_SER/

CoolRunner-II Serial Peripheral Interface Master
Firma Xilinx
Sprache: englisch

/COOLRUNNER_II_CPLD_DS/

XC2C256 CoolRunner-II CPLD
Firma Xilinx

/CPLD_01/

CPLD Fitting, Tips and Tricks
Firma XILINX

/CPLD_02/

CPLD Altera
Firma Mouser

/DE1_DOC/

Altera DE1 Board
Firma Altera

/DE1SOCBOARD_DOC/

DE1-SoC Board
Firma Terasic Technologies
www.terrasic.com

/DEV_NANO_SOC_A9_SBC_DS/

DEV NANO SOC A9 SBC
Firma Terasic Technologies

/DSP_FPGA/

Steve Zack, Suhel Dhanani
DSP Co-Processing in FPGAs: Embedding High-Performance, Low-Cost DSP Functions

/EFM01_DS/

EFM01
The Embedded FPGA Module EFM 01 is a very small low-cost module with SPARTAN-3E FPGA and USB 2.0 Interface. 50 I/O balls of the FPGA are available on standard 0,1 inch headers. It offers multiple configuration options and can also be used without the USB interface
CESYS

/EK_10M08E144ES_P/

MAX 10 FPGA (10M08S, 144-EQFP) Evaluation Kit
User Guide
Firma Altera

/EP1C3T144BOARDCABLEPROG/

FPGA Cyclone Altera FPGA EP1C3T144 Learning Board + USB Blaster JTAG programmer
Firma Altera

/FPGA_03/

Frank Riemenschneider
UltraLite-Chips minimieren Gehäusegröße und Leistungsaufnahme
Das geschrumpfte FPGA
Elektronik
www.elektroniknet.de

/FPGA_04/

N. Hemsoth, T. P. Morgan Co-Authors, Editors
FPGA FRONTIERS: New Applications in Reconfigurable Computing
Next Platform Press

/FPGA_05/

I. Schoppa, C. Kielmann
Ressourcenschoner
Mikroprogrammierte Steuerwerke in FPGAs

/FPGA_06/
Zynq UltraScale + MPSoC
Product Tables and Product Selection Guide
Firma XILINX

/FPGA_07/
A. P. Taylor
Zynq-SoC mit ARM-A9-Cores
Zweigleisige Kraftpakete
Elektronik 20 / 2015, Seite 24

/FPGA_08/
Xilinx Delivers a Comprehensive All Programmable Cost-Optimized Portfolio
Bemerkung: enthält auch Preise der Entwicklungs-Kits

/FPGA_BEG/
C. Elias
FPGAs für Einsteiger
c't Hacks 1/2014, Seite 40

/FPGA_MAKER/
C. Elias
FPGAs für Maker
Eine praktische Einführung in programmierbare Logik
dpunkt.verlag, Heidelberg
ISBN 978-3-86490-173-7

/FPGADEV_01/
Quartus Prime Software Download and Installation Quick Start Guide
Firma Infineon

/FPGAMP_02/
Udo Weik
FPGAs in Theorie und Praxis
Teil 2: Soft-Prozessorkerne für FPGAs - Wie sie funktionieren und was sie leisten
Die Komplexität heutiger FPGA-Bausteine hat mittlerweile ein Niveau erreicht, das die Integration von Prozessor- oder Controllerfunktionen in programmierbare Logikbausteine immer populärer werden lässt. dabei bieten die führenden FPGA-Hersteller passende Soft-Prozessorkerne für ihre eigenen Bausteine an und erleichtern so den Einstieg. Was die Prozessorkerne von Altera, Lattice und Xilinx können, lesen Sie in diesem Vergleich.
Elektronik 21 / 2006
www.elektroniknet.de

/FUNK_1998/

Carsten Funk
Entwicklung von I/O-Karten für ein PC-gestütztes Meßsystem mit zugehörigem Testprogramm
Diplomarbeit Fachhochschule Frankfurt am Main - University of Applied Sciences, Fachbereich Feinwerktechnik 1998

/GOP_XC2C64_DOC/

GOP_XC2C64 Mini CPLD
Breakout-Modul mit CPLD XC2C64 und Taktgenerator, XC2C64: 64 Macrozellen
Firma OHO-Elektronik
www.oh-elektronik.de

/GOP_XC3S200_DS/

M. Randelzhofer
GOP XC3S200 USER'S MANUAL
0.91
Firma OHO-Elektronik
www.oh-elektronik.de

/GOP_XC9572XL_DS/

M. Randelzhofer
GOP_XC9572XL
User's Manual
V 0.9
Firma OHO-Elektronik
www.oh-elektronik.de

/GOP_XCR3064XL_DS/

M. Randelzhofer
GOP_XCR3064XL
User's Manual
V 0.9
Firma OHO-Elektronik
www.oh-elektronik.de

/GRUND_HW_SYST_1/

K. Waldschmidt
Praktikum Grundlagen von Hardwaresystemen Sommersemester 2007
Versuch 1: VHDL-Grundlagen
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/GRUND_HW_SYST_2/

K. Waldschmidt
Praktikum Grundlagen von Hardwaresystemen Sommersemester 2007
Versuch 2: Multiplexer, Carry-Ripple- und Carry-Lookahead-Addierer
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/GRUND_HW_SYST_3/

K. Waldschmidt
Praktikum von Hardwaresystemen Sommersemester 2007
Versuch 3: Flipflops, synchrone sequentielle Schaltungen, Einführung in XILINX Webpack ISE und die FPGA-Platine
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/GRUND_HW_SYST_4/

K. Waldschmidt
Praktikum von Hardwaresystemen Sommersemester 2007
Versuch 4: Einführung in die Praktikums-CPU, Programmzähler und Assembler
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/GRUND_HW_SYST_5/

K. Waldschmidt
Praktikum von Hardwaresystemen Sommersemester 2007
Versuch 5: Ansteuerung eines VGA-Monitors und Grafikprogrammierung
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/GRUND_HW_SYST_6/

K. Waldschmidt
Praktikum von Hardwaresystemen Sommersemester 2007
Versuch 6: Computergrafik und Sprites
Johann Wolfgang Goethe-Universität Frankfurt am Main
Fachbereich 12: Informatik und Mathematik Professur für Technische Informatik 26. Juni 2007

/HERTWIG/

Andre Hertwig, Rainer Brück
Entwurf digitaler Systeme
Von den Grundlagen zum Prozessorentwurf mit FPGAs
Carl Hanser Verlag 2000
ISBN 3-446-21406-2

/HEUSINGER_1994/

P. Heusinger, K. Ronge, G. Stock
PLDs und FPGAs in der Praxis: komplexe Logikbausteine erfolgreich programmieren
Franzis-Verlag GmbH 1994
ISBN 3-7723-6075-0

/ICE_FAM/

iCE40™ LP/HX/LM Family Handbook
Firma Lattice

/ICESTICK/

ICEstick Evaluation Kit
User's Guide
Firma Lattice

/ISE_QUICK_START/

ISE Quick Start Tutorial
Firma Xilinx
Sprache: englisch

/ISPLSI_2032E_DS/

ispLSI 2032E
In-System Programmable SuperFAST High Density PLD
Firma Lattice

/JTAG_ADAPT_CABLE_XC2C64A_DOC/

/JTAG_USB_10PIN_CABLE_ALTERA_DS/

USB Blaster Programmer Mini Altera FPGA CPLD Cable

/MULTICOPTER_01/

Quadrokopter mit FPGA und ARM
http://www.youtube.com/watch?v=vx6_it0OVB4#t=343
http://www.xilinx.com/publications/prod_mktg/ZynqAuto_ProdBrf.pdf

/NEXYS_III_MANUAL/

NEXYS3 Board Reference Manual
Firma DIGILENT
Sprache: englisch

/NEXYS_III_SCH/

Nexys III Schaltplan
Firma DIGILENT
Sprache: englisch

/PCI_FPGA/

Oliver Schmidt
Design eines PCI-Target-Controllers und
Implementierungsvorbereitung für einen XILINX-FPGA

/PLD_HANDBOOK_2003/

Karen Parnell, Nick Mehta
Programmable Logic Design Quick Start Handbook
Firma Xilinx

/PLD_LAB/

Versuch PLD
Anleitung zum Bedienen der Entwurfssoftware und Beschreibung des verwendeten CPLD-Trainings-Boards mit Schaltungsbeispielen
HTWK Leipzig FBEIT

/PROGKAB_USB/

V. Pfeiffer
Programmierkabel mit USB-Anschluss
Firma Digilent Inc.

/PROLOGDES/

Programmable Logic Design
Quick Start Guide
Firma Xilinx
Sprache: englisch

/RATSCHINSKI_1999/

Frank Ratschinski
Entwicklung eines CAN-Moduls für ein modulares Meßsystem
Diplomarbeit Fachhochschule Frankfurt am Main, Fachbereich Feinwerktechnik 1999

/SPARTAN_3_FAMILY/

Spartan-3 FPGA Family
Data Sheet
Firma XILINX

/SPARTAN_3_GEN/

Spartan-3 Generation FPGA User Guide
Spartan-3A DSP, Spartan-3AN, Spartan-3A, Spartan-3E, and Spartan-3 FPGA Platforms
Firma Xilinx
Bemerkung: umfassende Darstellung
Sprache: englisch

/SPARTAN_3_USER/

Spartan-3 Starter Kit Board User Guide
Firma Xilinx
Bemerkung: enthält auch die Schaltungen
Sprache: englisch

/SPARTAN_3A_DS /

Spartan-3A FPGA Family: Data Sheet
Firma Xilinx

/SPARTAN_3E_SCH/

Digilent 3E Starter Board
Firma Digilent Inc. 2004
Bemerkung: Schaltung Kit Spartan-3E
Sprache: englisch

/SPARTAN_3E_USER/

Spartan-3E Starter Kit Board User Guide
Firma Xilinx
Sprache: englisch

/SPARTAN_PINOUT/

Spartan-3 FPGA Family: Pin out Description
Firma Xilinx
Sprache: englisch

/WANNFPGA/

V. Pfeiffer
Wann sollte man einen FPGA anwenden ?
Vortrag GI Regionalgruppe Rhein-Main 11. Mai 2017

/XC2_XL_DEV_SCH/

DIGILAB XC2-XL
Firma Digilent Inc. 2003
Bemerkung: Schaltung Kit XC2-XL
Sprache: englisch

/XC2C64A_BOARD_DOC/

Firma canton electronics

/XC2C64A_BRK01_DS/

Digilent C-Mod™ Boards Reference Manual
Breakout-Modul mit CPLD XC2C64
Firma Digilent

/XC2C64A_BRK02_DS/

CoolRunner-II CPLD breakout board
Firma DANGEROUS PROTOTYPES

/XC2C64A_DS/

XC2C64A CoolRunner-II CPLD
Firma XILINX
www.xilinx.com

/XC2XL_DEV_DS/

Digilent XC2-XL System Board Reference Manual
Firma Digilent
Bemerkung: enthält XC2C256, XC9572XL; Beschreibung des Kits XC2-XL
Hinweis !
Die darin enthaltenen Schaltungen sind teilweise falsch. Die "richtigen" Schaltungen sind in /XC2_XL_BOARD_SCH/ enthalten
Sprache: englisch

/XC9500_FAM/

XC9500 In-System Programmable CPLD Family
Firma Xilinx

/XC9536_DEMO/

XC9536 ISP Demo Board
Firma Xilinx

/XC9536_DS/

XC9536 In-System Programmable CPLD
In-System Programmable CPLD
Firma XILINX

/XC9572_2006_DS/

XC9572 In-System Programmable CPLD
Firma XILINX

/XC9572_UEB/

J. Humer, B. Geiger
CPLD - Übungsboard 2 für XC9572/108

/XC9572XL_DS/

XC9572XL High Performance CPLD
Firma XILINX

/XCR3064XL_DS/

XCR3064XL 64 Macrocell CPLD
Firma OHO-Elektronik
www.oh-elektronik.de

/XILINX_01/

Xilinx delivers a Comprehensive All Programmable Cost-Optimized Portfolio
Firma XILINX

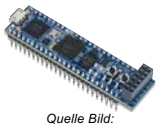
/ZYNQ_7000/

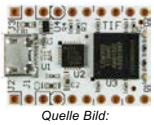
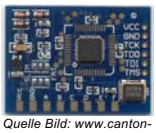
ZYNQ-7000 All PROGRAMMABLE SoC
Firma XILINX
www.xilinx.com

/ZYNQ7000_01/

Zynq-7000 All Programmable SoC
Technical Reference Manual
Firma XILINX
www.xilinx.com
Sprache: englisch

14. CPLD- FPGA-Baueinheiten und Zubehör

Gegenstand	Kurzbeschreibung	Bild
Basys2 100K - XILINX Spartan-3E FPGA	/BASYS2/ *Hersteller:* Firma Digilent *Dokument:* BASYS2_100K_DS *Lieferant:* TRENZ *Bestellnr.:* 23455 *Preis:* 122,55 € + MWST	 *Quelle Bild: www.digilentinc.com*
Cmod A7	/CMOD_A7/ Breadboardable Artix-7 (35T) FPGA Module *Hersteller:* Firma Digilent Inc. *Lieferant:* TRENZ *Preis:* 85,17 € oMWST.	 *Quelle Bild: www.digilentinc.com*
CORE3S250E	/CORE3S250E/ Waveshare CORE3S250E Xilinx Spartan 3E XC3S250E FPGA Development Core Board *Hersteller:* Firma Waveshare *Lieferant:* Eckstein *Bestellnr.:* WS06040	 *Quelle Bild: www.waveshare.com*
EFM01	/EFM01/ *Hersteller:* Firma CESYS *Dokument:* EFM01_DS *Preis:* 124,- €	 *Quelle Bild: CESYS.de*
FPGA CYCLONE Altera FPGA EP1C3T144 Learning Board + USB Blaster JTAG Programmer	/EP1C3T144USBLASTER/	 *Quelle Bild: www.ebay.de*
Altera MAX II EPM240 CPLD Development Board	/EPM240_CPLD_BOARD/ On board EPM240T100C5 chip, JTAG Interface, 50 MHz active crystals. 5 V supply with a switch *Hersteller:* Firma Alter	 *Quelle Bild: DX.COM*

Gegenstand	Kurzbeschreibung	Bild
	/JTAG_ADAPT_CABLE_XC2C64A/ *Hersteller:* Firma canton electronics *Dokument:* JTAG_ADAPT_CABLE_XC2C64A_DOC *Lieferant:* canton electronics	 *Quelle Bild: www.canton-electronics.com*
MAX® II Micro Kit	/MAX_II_MICRO_KIT/ *Dokument:* MAX_II_MICRO_KIT_DS	 *Quelle Bild: terrasic.com*
Spartan-3	/SPARTAN_3/ *Hersteller:* Firma Digilent *Dokument:* SPARTAN_3_USER	
Spartan-3E	/SPARTAN_3E/ *Hersteller:* Firma Digilent *Dokument:* SPARTAN_3E_USER	 *Quelle Bild: www.digilentinc.com*
TIF4000	/TIF4000/ Tiny FPGA Board Lattice *Lieferant:* Watterott electronic *Bestellnr.:* 20110932	 *Quelle Bild: www.Watterott.com*
	/XC2C64A_BOARD/ *Hersteller:* Firma canton electronics *Dokument:* XC2C64A_BOARD_DOC *Lieferant:* canton electronics	 *Quelle Bild: www.canton-electronics.com*
C-MOD-C2	/XC2C64A_BRK01/ Breakout-Modul mit CPLD XC2C64 *Hersteller:* Firma Digilent *Dokument:* XC2C64A_BRK01_DS *Lieferant:* TRENZ *Bestellnr.:* 23565	
XC2C64A CoolRunner-II CPLD Dev Board	/XC2C64A_BRK02/ XC2C64A CoolRunner-II CPLD Dev Board *Hersteller:* Firma DANGEROUS PROTOTYPES *Dokument:* XC2C64A_BRK02_DS *Lieferant:* Watterott electronic *Preis:* 12,75 €	

Gegenstand	Kurzbeschreibung	Bild
GOP_XC2C64	/XC2C64A_BRK03/ Breakout-Modul mit CPLD XC2C64 und Taktgenerator, XC2C64: 64 Macrozellen *Hersteller:* Firma OHO-Elektronik *Dokument:* GOP_XC2C64_DOC *Lieferant:* TRENZ *Bestellnr.:* TE0252-00 *Preis:* 19,90 € + MWST	 *Quelle Bild: www.trenz-electronic.de*
XC2-XL System Board	/XC2XL_DEV/ *Hersteller:* Firma Digilent *Dokument:* XC2XL_DEV_DS	 *Quelle Bild: PFEIFFER*
XC3S500E-DIL	/XC3S500E_DIL/ DIL Modul mit XC3S500E Spartan-3E FPGA (nacktes Modul) *Lieferant:* TRENZ *Bestellnr.:* TE0260-00B	 *Quelle Bild: www.trenz-electronic.de*
GOP_XC9572XL Mini CPLD	/XC9572XL_BRK/ GOP_XC9572XL Mini CPLD, 72 Macrozellen, 1700 Gatter, DIL Footprint 24 Pin, 3,5V … 5.5V, 49.152 MHz *Hersteller:* Firma OHO-Elektronik *Dokument:* GOP_XC9572XL_DS *Lieferant:* TRENZ *Bestellnr.:* TE0250-00 *Preis:* 23,68 €	 *Quelle Bild: www.trenz-electronic.de*
GOP_XCR3064XL Mini CPLD	/XCR3064XL_BRK/ GOP_XCR3064XL Mini CPLD 64 Macrozellen, 1500 Gatter, 192 MHz *Lieferant:* TRENZ *Bestellnr.:* TE0251-00 *Preis:* 19,90 € + MWST	 *Quelle Bild: www.trenz-electronic.de*