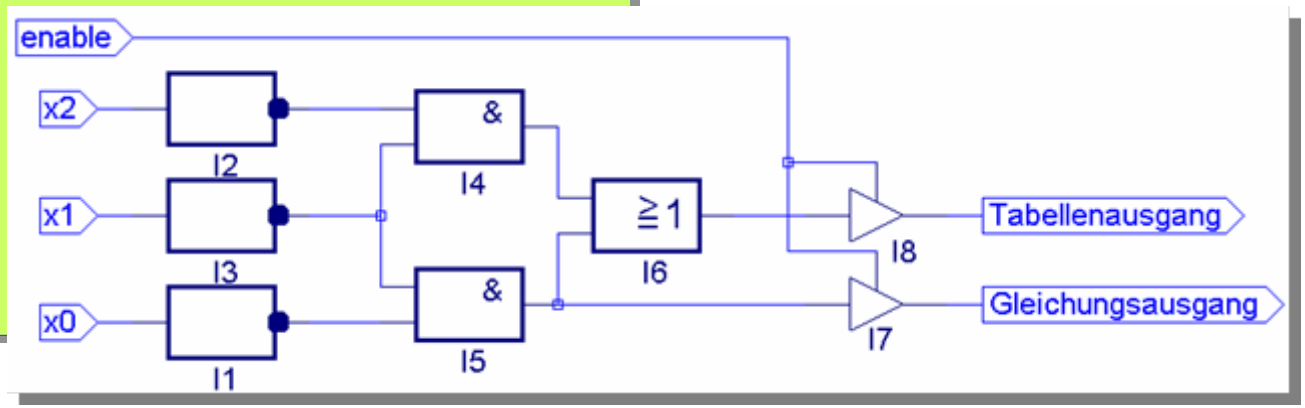
Vorbemerkung

```
MODULE phdl_0
TITLE 'Kombinatorische Schaltung; Gleichung / Tabelle'

DECLARATIONS
x0,x1,x2                PIN;
enable                  PIN;
Tabellenausgang         PIN ISTYPE 'com';
Gleichungsausgang       PIN ISTYPE 'com';

EQUATIONS
Gleichungsausgang.oe = enable;

Tabellenausgang.oe     = enable;
Gleichungsausgang      = !x2 & !x1 & !x0  #  !x2 & !x1 & x0  #  x2 & !x1 & !x0;

TRUTH_TABLE        ( [x2,x1,x0]->[Tabellenausgang])
                     [0,0,0]->[1];
                     [0,0,1]->[1];
                     [0,1,0]->[0];
                     [0,1,1]->[0];
                     [1,0,0]->[1];
                     [1,0,1]->[0];
                     [1,1,0]->[0];
                     [1,1,1]->[0];
END
```

Philips Hardware Description Language (PHDL) ist die Beschreibungssprache der XPLA™-Entwicklungsumgebung für die Philips-CPLD´s. Ein Derivat der ABEL-HardwareDescription Language



Bestandteile der PHDL- Syntax

```
MODULE phdl_0
TITLE 'Kombinatorische Schaltung; Gleichung / Tabelle'

DECLARATIONS
x0,x1,x2              PIN;
enable                PIN;
Tabellenausgang       PIN ISTYPE 'com';  " Definition des
                                         "Tabellenausgangs
Gleichungsausgang     PIN ISTYPE 'com';

EQUATIONS
Gleichungsausgang.oe = enable;

Tabellenausgang.oe      = enable;
Gleichungsausgang = !x2 & !x1 & !x0  # !x2 & !x1 & x0  # x2 & !x1 & !x0;

TRUTH_TABLE    ( [x2,x1,x0]->[Tabellenausgang])
                      [0,0,0]->[1];
                      [0,0,1]->[1];
                      [0,1,0]->[0];
                      [0,1,1]->[0];
                      [1,0,0]->[1];
                      [1,0,1]->[0];
                      [1,1,0]->[0];
                      [1,1,1]->[0];
END
```

Bezeichner

- maximale Länge 31 Zeichen
- keine reservierten Wörter
- müssen mit Buchstaben oder Unterstrich "_" beginnen
- Buchstaben, Zahlen, Unterstrich erlaubt
- Groß-/ Kleinschreibung wird unterschieden (nicht bei reservierten Wörtern)

Kommentar

- ´´ Kommentar bis Zeilenende
- ´´ Kommentar bis ´´
- // Kommentar bis Zeilenende
- // Kommentar bis //
- /* Kommentar bis */

Operatoren

!	Negation
&	UND
#	ODER
$	Antivalenz
!$	Äquivalenz
.	.
.	.

Dateiaufbau des PHDL-Files

```
MODULE phdl_0
TITLE 'Kombinatorische Schaltung; Gleichung / Tabelle'

DECLARATIONS
x0,x1,x2                PIN;
enable                  PIN;
Tabellenausgang         PIN ISTYPE 'com';   " Definition des
                                            "Tabellenausgangs
Gleichungsausgang       PIN ISTYPE 'com';
TRUTH_TABLE    ( [x2,x1,x0]->[Tabellenausgang] )
                        [0,0,0]->[1];
                        [0,0,1]->[1];
                        [0,1,0]->[0];
                        [0,1,1]->[0];
                        [1,0,0]->[1];
                        [1,0,1]->[0];
                        [1,1,0]->[0];
                        [1,1,1]->[0];

EQUATIONS
Gleichungsausgang.oe = enable;
Tabellenausgang.oe   = enable;
Gleichungsausgang    = !x2 & !x1 & !x0  #  !x2 & !x1 & x0  #  x2 & !x1 & !x0;

END
```

Dateikopf

Module Satement → MODULE <Dateiname>
Titel → 'Erläuterungen zum Design (Text)'
Property → Compiler- und Fitteranweisungen

Deklarationen

DECLARATIONS

Definition von → Konstanten, Variablen, Macros, I/O-Pins, Tabellen, Marken, ...

Gleichungen

EQUATIONS

Definition von → logischen Gleichungen und Zustandsübergangs-gleichungen

Dateiende

End Satement → END

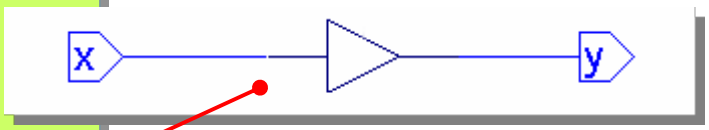
Deklarationen

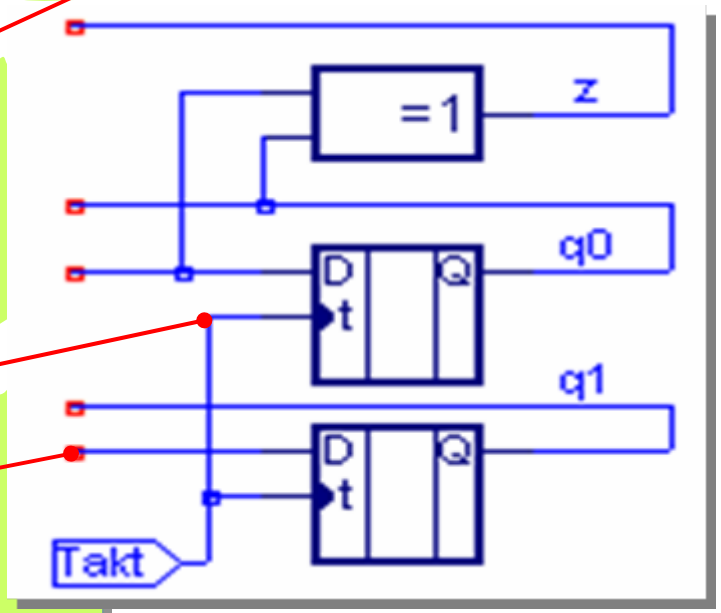
```
DECLARATIONS

LOW,HIGH    =     0,1;              " Definition von Konstanten
Adresse  =    ^hFFA0;


x0,x1       PIN;                    " Definition von Variablen
enable      PIN;

Tabellenausgang         PIN ISTYPE 'com';
Gleichungsausgang       PIN ISTYPE 'com';

Tabelleneingang  = [x2,x1,x0];   " Definition von Signalgruppen


TRUTH_TABLE  ([Tabelleneingang]->[Tabellenausgang] )
                        [0,0]->[1];          " Definition von Wahrheitstabellen
                        [0,1]->[1];
                        [1,0]->[0];
                        [1,1]->[0];

Teil MACRO(a,b,c)                            " Definition von Macro‘s
          { c = !a&b # a&!b  }  ;

END
```

Ein- / Ausgangsvariablen

<Signalname> PIN <Pinnummer> ISTYPE <Attribut>, <Attribut>, <Attribut>;

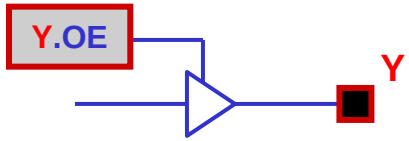
```
DECLARATIONS

x, Takt    PIN;                  " Definition der Eingangsvariablen

y          PIN ISTYPE 'com';     " Definition einer kombinatorischen
                                 "Ausgangsvariablen

q0, q1     NODE ISTYPE 'reg_d';  " Definition der Zustands-
                                 "variablen q0, q1

z          NODE                  " Definition der internen
                                 "Variablen z

EQUATIONS

y = x ;
q0.CLK  = Takt;
q1.CLK  = Takt;
q0.D    =  ...
q1.D    =  ...

END
```



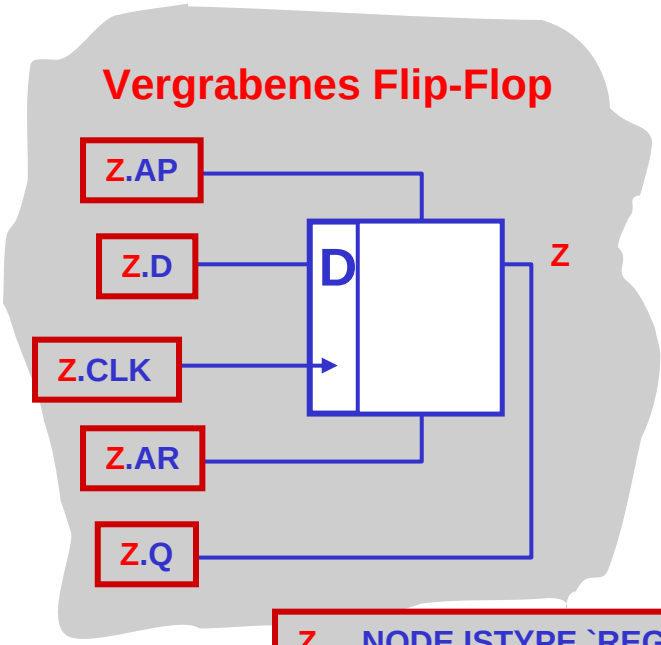


Variablenkennzeichnung

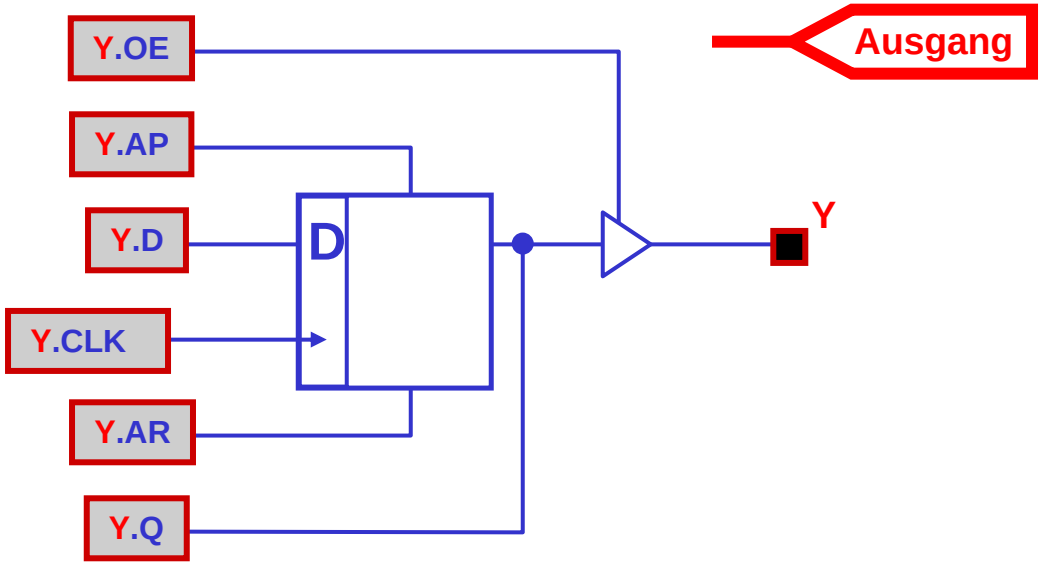

```
X0, X1   PIN;
```




```
Y    PIN ISTYPE `COM`;
```



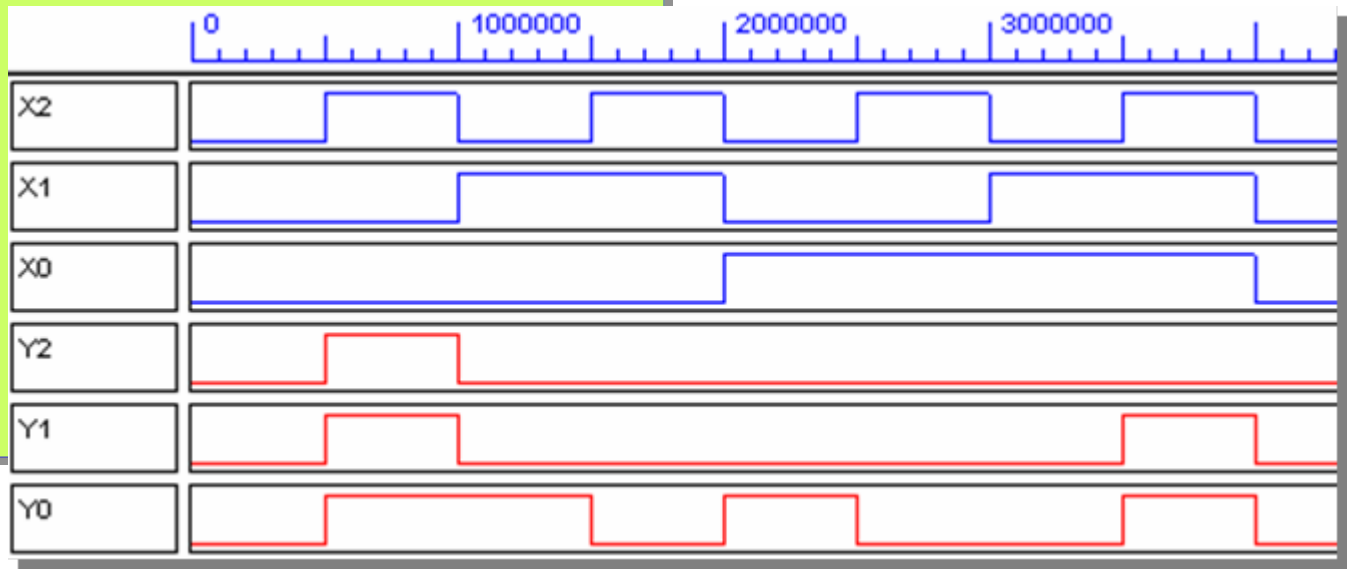

```
Z    NODE ISTYPE `REG`
```




```
Y    PIN ISTYPE `REG`;
```

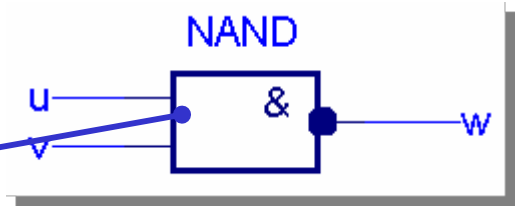
Tabelle

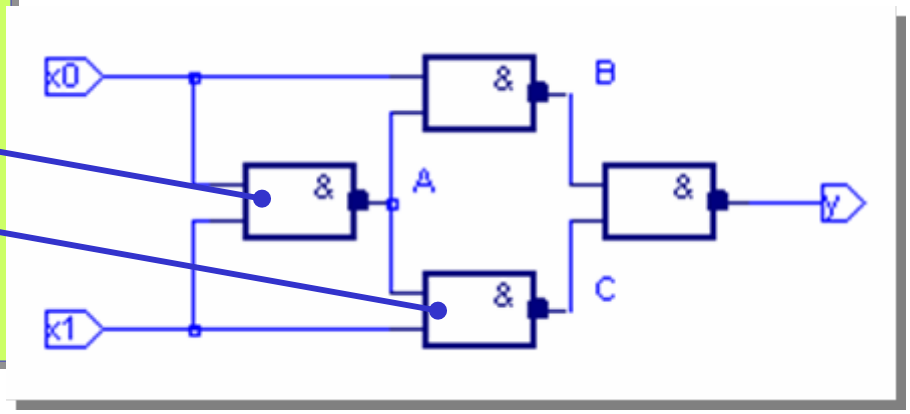
```
MODULE SCH_TAB
TITLE   'Umsetzung einer Schaltbelegungstabelle'

DECLARATIONS

x2,x1,x0  PIN;                  " Definition der Eingangsvariablen
y2,y1,y0  PIN  ISTYPE 'com';    " Definition einer kombinatorischen
                                "Ausgangsvariablen

Eingang  = [x2,x1,x0] ;         " Definition der Signalgruppen
Ausgang = [y2,y1,y0] ;

TRUTH_TABLE
        (Eingang -> Ausgang)

            [0,0,0] -> [0,0,0];
            [0,0,1] -> [0,0,1];
            [0,1,0] -> [0,0,1];
            [0,1,1] -> [0,0,0];
            [1,0,0] -> [1,1,1];
            [1,0,1] -> [0,0,0];
            [1,1,0] -> [0,0,0];
            [1,1,1] -> [0,1,1];

END
```



Macrodeklaration

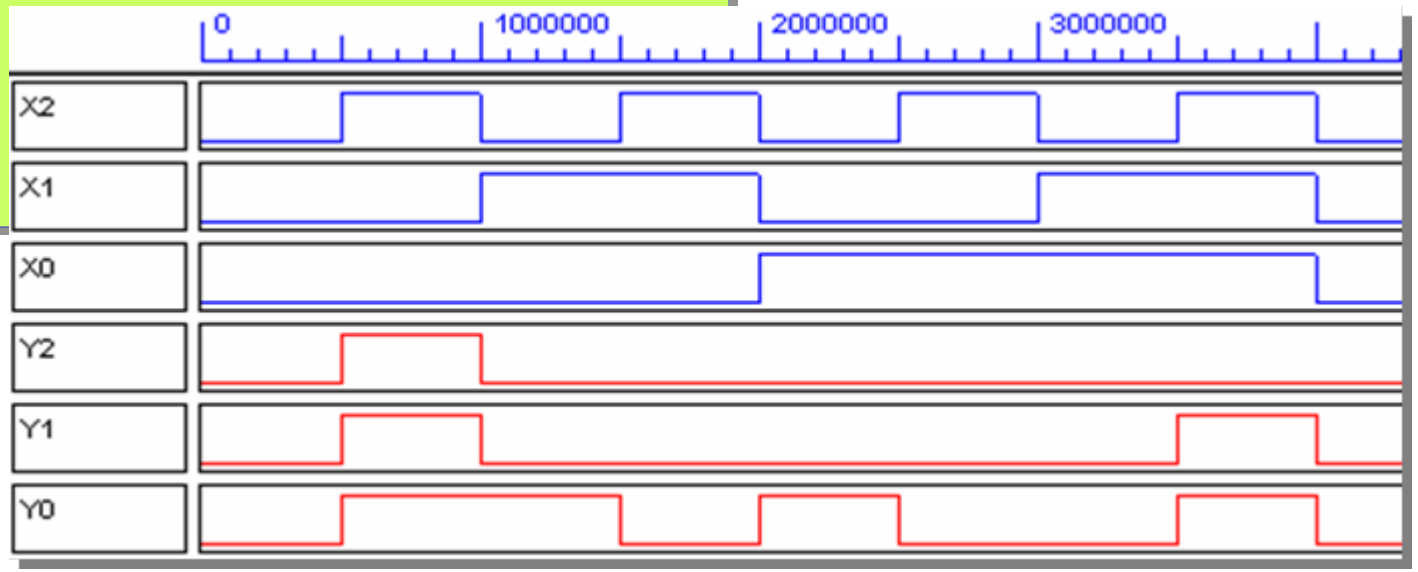
```
MODULE  Makro              " !! Macro ist ein
                           "    reserviertes Wort

TITLE   'Verwendung von Macros'

DECLARATIONS

x0,x1       PIN;
y           Pin Istype 'com';
A,B,C       node;

NAND MACRO(u,v,w)          " Definition des Macro
            { ?w = !(?u&?v) };

EQUATIONS

NAND (x0,x1,A);            " Benutzung des Macro
NAND (x0,A,B );            " Benutzung des Macro
NAND (x1,A,C );            " Benutzung des Macro
NAND ( B,C,y );            " Benutzung des Macro


END
```





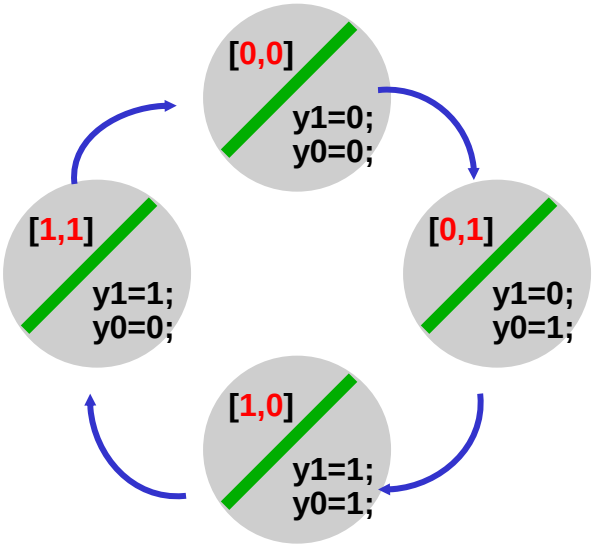
Logische Gleichungen

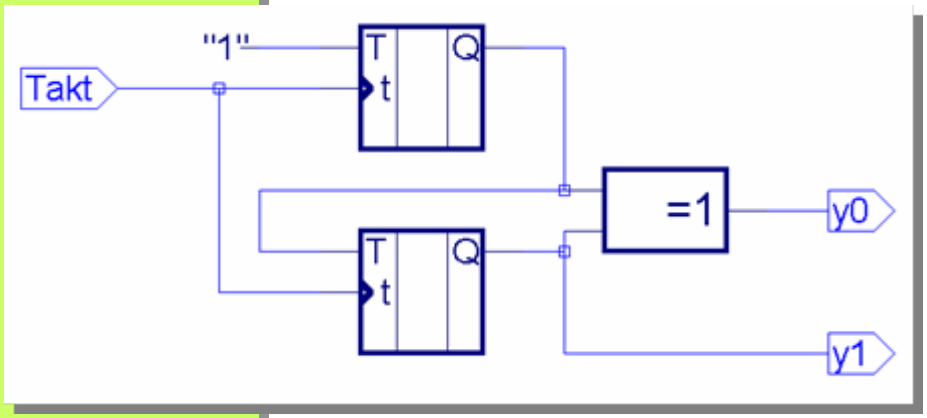
```
MODULE  Log_Gl
TITLE   'Umsetzung eines Gleichungsbündels'

DECLARATIONS

x2,x1,x0   PIN;                    // Definition der Eingangsvariablen
y2,y1,y0   PIN ISTYPE 'com';       // Definition der Ausgangsvariablen

EQUATIONS

y0   =  x2 & !x1  # x2 & x0 # !x1 & x0 # !x2 & x1 & !x0 ;

y1   =  x2 & x1  $  x2 & x0  $  x2 ;

y2   =  x2 & !x1 & !x0 ;


END
```



Zähler

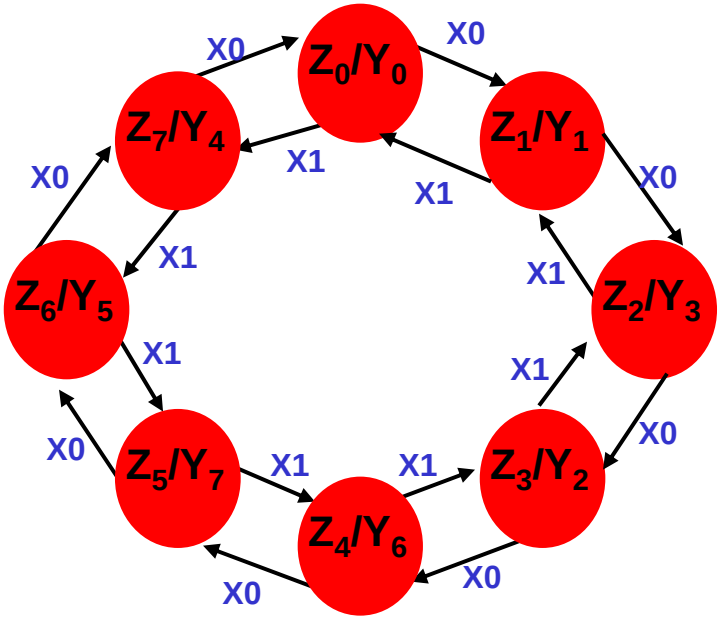
```
MODULE  Automat1

TITLE   'Umsetzung einer sequenziellen Schaltung'

DECLARATIONS

Takt   PIN;                     // Definition der Eingangsvariablen
y1,y0    PIN ISTYPE 'com';      // Definition der Ausgangsvariablen
q1,q0  NODE ISTYPE 'reg';       // Definition der Zustandsvariablen

EQUATIONS

q1.clk = Takt;
q0.clk = Takt;

STATE_DIAGRAM [q1,q0]

        STATE [0,0]:    y1=0;  y0=0;
                        GOTO  [0,1];
        STATE [0,1]:    y1=0;   y0=1;
                        GOTO  [1,0];
        STATE [1,0]:    y1=1;   y0=1;
                        GOTO  [1,1];
        STATE [1,1]:    y1=1;   y0=0;
                        GOTO  [0,0];

END
```

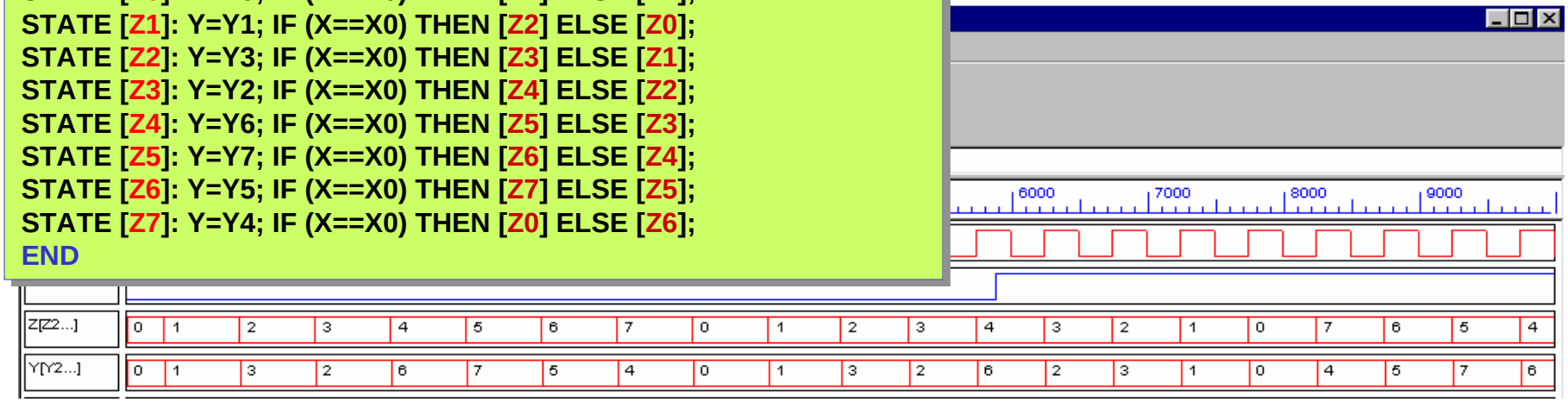




Mooreautomat

```
MODULE  Moore
TITLE   'Vor- Rückzähler mit Graycodeausgang'
DECLARATIONS
x, Takt     PIN;
y2..y0      PIN ISTYPE 'com, retain';
z2..z0  NODE ISTYPE 'reg, retain';
X  = [x];       Z = [z2,z1,z0];  Y =[y2,y1,y0];
X0 = [0];       Z0 = [0,0,0];   Y0 =[0,0,0];  Z4 = [1,0,0];   Y4 =[1,0,0];
X1 = [1];       Z1 = [0,0,1];   Y1 =[0,0,1];  Z5 = [1,0,1];   Y5 =[1,0,1];
                Z2 = [0,1,0];   Y2 =[0,1,0];  Z6 = [1,1,0];   Y6 =[1,1,0];
                Z3 = [0,1,1];   Y3 =[0,1,1];  Z7 = [1,1,1];   Y7 =[1,1,1];
EQUATIONS
z2.c= Takt;
z1.c= Takt;
z0.c= Takt;
STATE_DIAGRAM Z
STATE [Z0]: Y=Y0; IF (X==X0) THEN [Z1] ELSE [Z7];
STATE [Z1]: Y=Y1; IF (X==X0) THEN [Z2] ELSE [Z0];
STATE [Z2]: Y=Y3; IF (X==X0) THEN [Z3] ELSE [Z1];
STATE [Z3]: Y=Y2; IF (X==X0) THEN [Z4] ELSE [Z2];
STATE [Z4]: Y=Y6; IF (X==X0) THEN [Z5] ELSE [Z3];
STATE [Z5]: Y=Y7; IF (X==X0) THEN [Z6] ELSE [Z4];
STATE [Z6]: Y=Y5; IF (X==X0) THEN [Z7] ELSE [Z5];
STATE [Z7]: Y=Y4; IF (X==X0) THEN [Z0] ELSE [Z6];
END
```



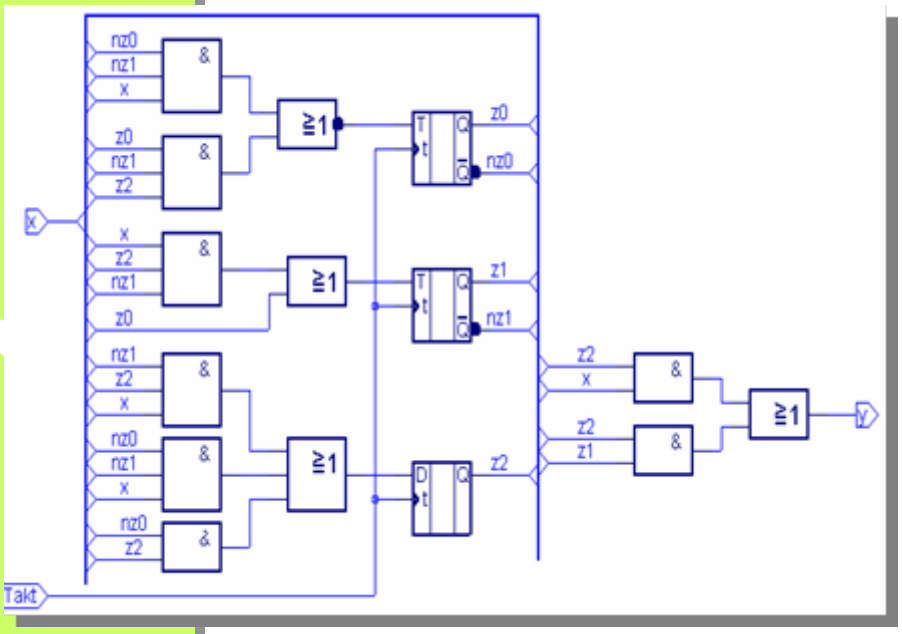


Mealyautomat

```
MODULE  Mealy
TITLE   'Design eines Mealyautomaten'
DECLARATIONS
x, Takt     PIN;                    //Definition der Eingangsvariablen
z2..z0      NODE ISTYPE 'reg';      //Definition der Zustandsvariablen
y           PIN  ISTYPE 'com';      //Definition der Zustandsvariablen
X = [x]; X0 = [0]; X1 = [1];                    //Eingangscodierung
Z = [z2..z0]; Z0 = [0,0,0]; Z1 = [0,0,1];       //Zustandscodierung
     Z2 = [0,1,0]; Z3 = [0,1,1]; Z4 = [1,0,0];
     Z5 = [1,0,1]; Z6 = [1,1,0]; Z7 = [1,1,1];
Y = [y]; Y0 = [0]; Y1 = [1];                    //Ausgangscodierung
EQUATIONS
z0.clk = Takt; z1.clk = Takt; z2.clk = Takt;   //Taktbeschalt.
STATE_DIAGRAM Z
STATE [Z0]:             IF(X==X0) THEN [Z1] WITH Y=Y0;
                        IF(X==X1) THEN [Z4] WITH Y=Y0;

STATE [Z1]:             IF(X==X0) THEN [Z2] WITH Y=Y0;
                        IF(X==X1) THEN [Z2] WITH Y=Y0;

STATE [Z6]:             IF(X==X0) THEN [Z7] WITH Y=Y1;
                        IF(X==X1) THEN [Z7] WITH Y=Y1;

STATE [Z7]:             IF(X==X0) THEN [Z0] WITH Y=Y1;
                        IF(X==X1) THEN [Z0] WITH Y=Y1;
END
```

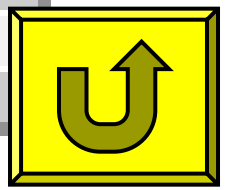
Dekadenprüfung

δ/λ	X0	X1
Z0	Z1/Y0	Z4/Y0
Z1	Z2/Y0	Z2/Y0
Z6	Z7/Y1	Z7/Y1
Z7	Z0/Y1	Z0/Y1



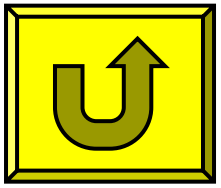
Dot Extensions

Erweiterung	Beschreibung	unter-stützt	emu-liert
.AP .ASET .PR	asynchron setzen	x	
.AR .ACLR .RE	asynchron rücksetzen	x	
.CE	Takt freigeben		x
.CLK	Takteingang für Register	x	
.CLR .SR	synchron rücksetzen	x	
.COM	kombinatorische Rückführung	x	
.D	Dateneingang D-Flip-Flop	x	
.FB .Q	Registerrückführung	x	
.J	Dateneingang JK-Flip-Flop		x
.K	Dateneingang JK-Flip-Flop		x
.PIN	Rückführung vom Pin		x
.OE	Output Enable (tristate Steuerung)	x	
.R	Dateneingang RS-Flip-Flop	x	
.S	Dateneingang RS-Flip-Flop		x
.SET .SP	synchron setzen	x	
.T	Dateneingang T-Flip-Flop	x	



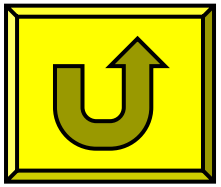
Reservierte Wörter

Als Bezeichner nicht erlaubte Worte !							
bidirect	bin	break	busi	buso	call	case	data
datafile	date	dats	datv	dec	declarations	decv	devbus
defsis	defstate	defvar	device	do	else	encv	end
endcase	equations	etc	f	for	functional_block	with	gt
goto	hex	hp	ident	if	ifv	include	incr
input	interface	istype	it	le	list	macro	module
ne	node	notdet	oct	output	p	pc	pco
pe	pin	property	pt	pv	reload	repeat	ret
s	save	sdc	sp	st	stab	state	state_diagram
su	suns	sub	tap2q	tar2q	tbuf	tclk	then
time	title	toe	tpd0	tpd1	tpd2	trac	trd
undef	until	when	while	with			



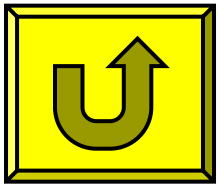
Zahlenformate

Zahlenformate			
Zahl	Zahlen-basis	Sym-bol	Beispiel
Binär	2	^2	^b10010
Oktal	8	^o	^o24304
Dezimal	10		124
Dezimal	10	^d	^d124
Hex.	16	^h	^h4A2



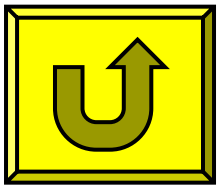
Operatoren

Operatoren			
Prio-rität	Operator	Beschreibung	Beispiel
1	-	Zweierkomplement	- A
1	!	Bitweise Invertierung	! A
2	&	Und	A & B
2	<<	links schieben	A << 1
2	>>	rechts schieben	A >> 1
2	*	Multiplikation	A * B
2	/	Division	A / B
3	%	Modulo Division	A % B
3	+	Addition	A + B
3	-	Subtraktion	A - B
3	#	Oder	A # B
3	$	Antivalenz	A $ B
3	!$	Äquivalenz	A !$ B
4	==	gleich	A == B
4	!=	ungleich	A != B
4	<	kleiner	A < B
4	<=	kleiner gleich	A <= B
4	>	größer	A > B
4	>=	größer gleich	A >= B



Attribute

Attribut <attr>	Beschreibung	
buffer	Nutzung von nicht invertiertem Ausgang	nicht bei NODE
collapse	Signal wird bei der Logiksynthese aufgelöst	
com	kombinatorischer Ausgang	
invert	Nutzung von invertiertem Ausgang	nicht bei NODE
keep	Signal wird bei der Logiksynthese nicht aufgelöst	
reg	Registerausgang; getaktet	
reg_d	D-Flip-Flop; getaktet	
reg_g	D-Flip-Flop; transparent; getaktet	zustandsgesteuert
reg_jk	JK-Flip-Flop; getaktet	
reg_sr	SR-Flip-Flop; getaktet	
reg_t	T-Flip-Flop; getaktet	
retain	keine Reduktion bei der Logiksynthese; redundante Terme bleiben erhalten	



Praktikum Digitale Systeme / Automaten

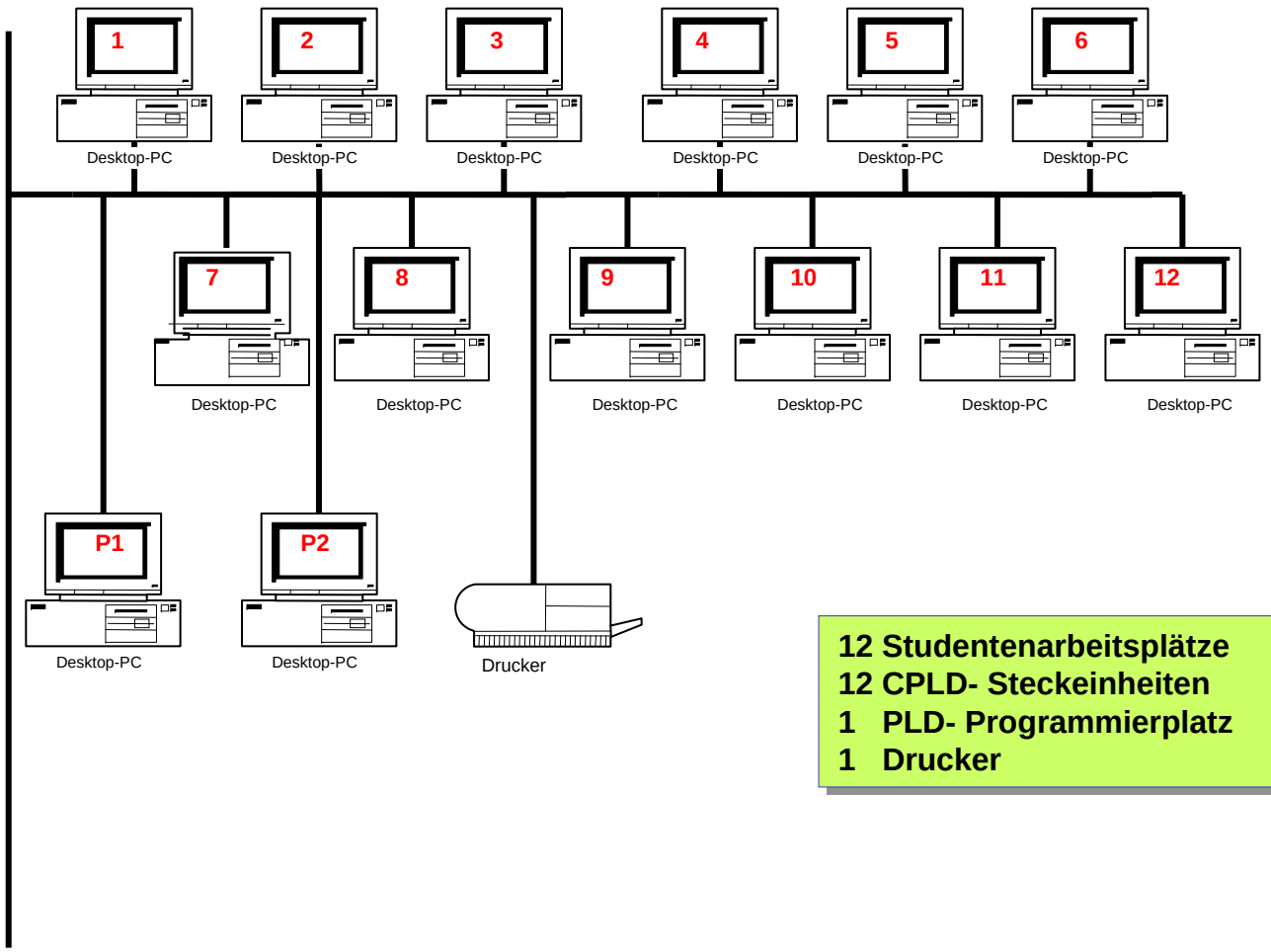
Raum 106

Prof. H.Pfahlbusch
Dr.J.Krupke

Tel: **1280**
: **1286**

Hardwareausstattung

Hochschulnetz

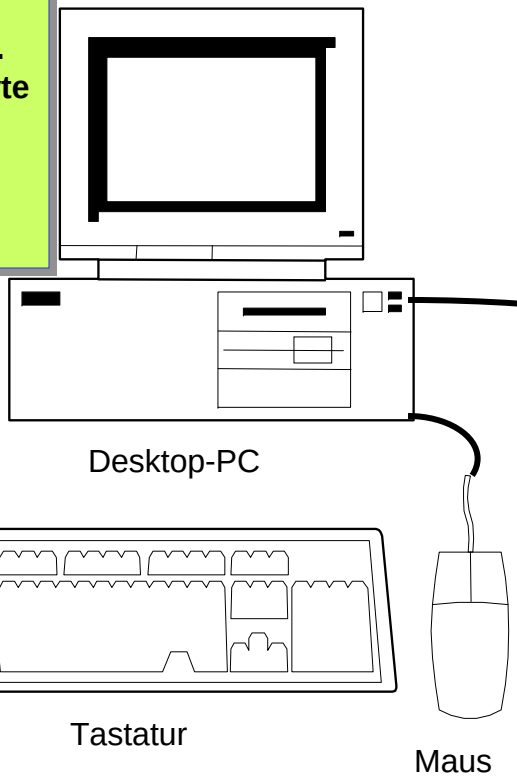


12 Studentenarbeitsplätze
12 CPLD- Steckeinheiten
1 PLD- Programmierplatz
1 Drucker

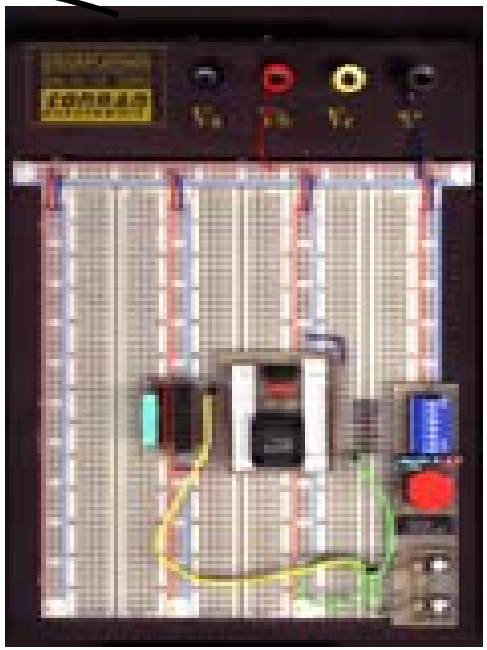
Hochschulnetz

Ausstattung der Studentenarbeitsplätze

PC:
Pentium 1,2 Ghz.
Festplatte 40 GByte
RAM 512 MByte
DVD ROM
ZIP 250
17“ TFT- Monitor

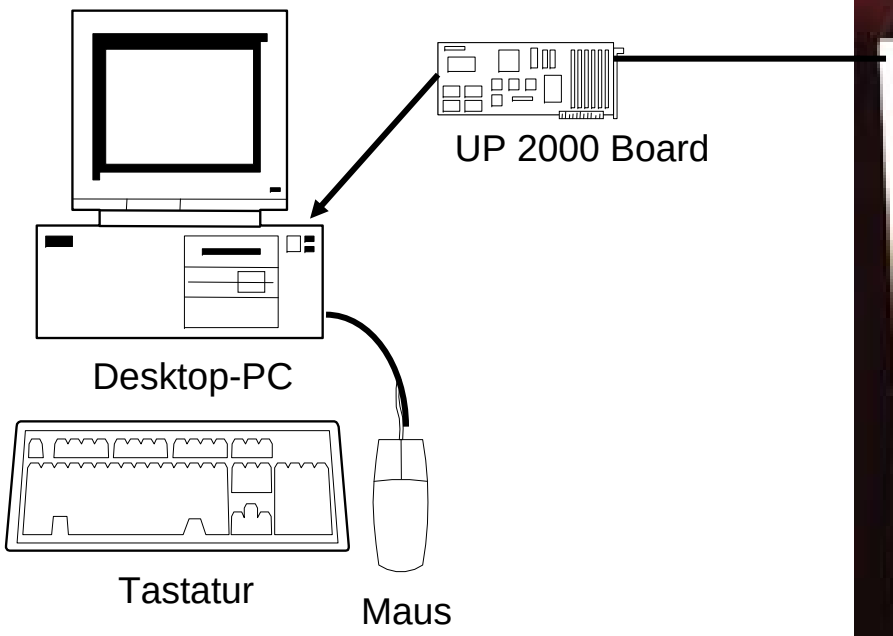


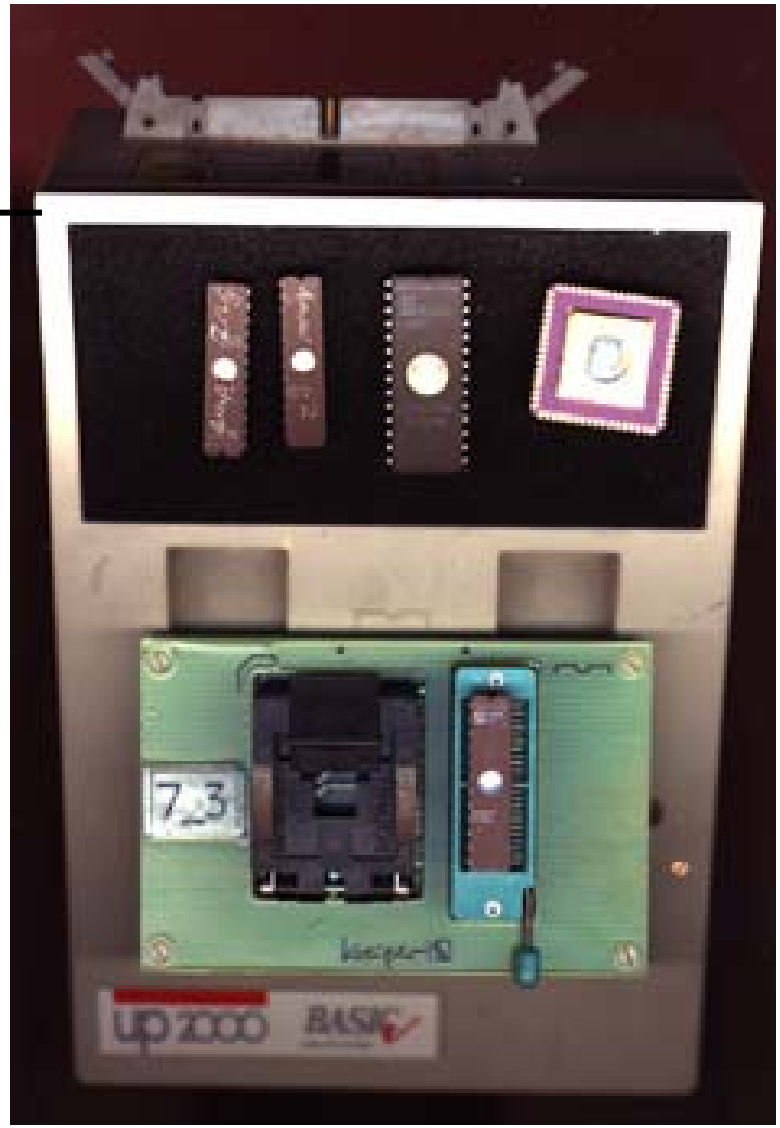
Experimentierboard



Software : Windows NT4.0
XPLA-Professional

Ausstattung der Programmierarbeitsplätze





PC:		
486	33	Mhz.
Festplatte	1	GByte
RAM	16	MByte
Software:	DOS/Win3.11	
	UP 2000	