Digitaltechnik Wintersemester 2021/2022 Hilfsblatt

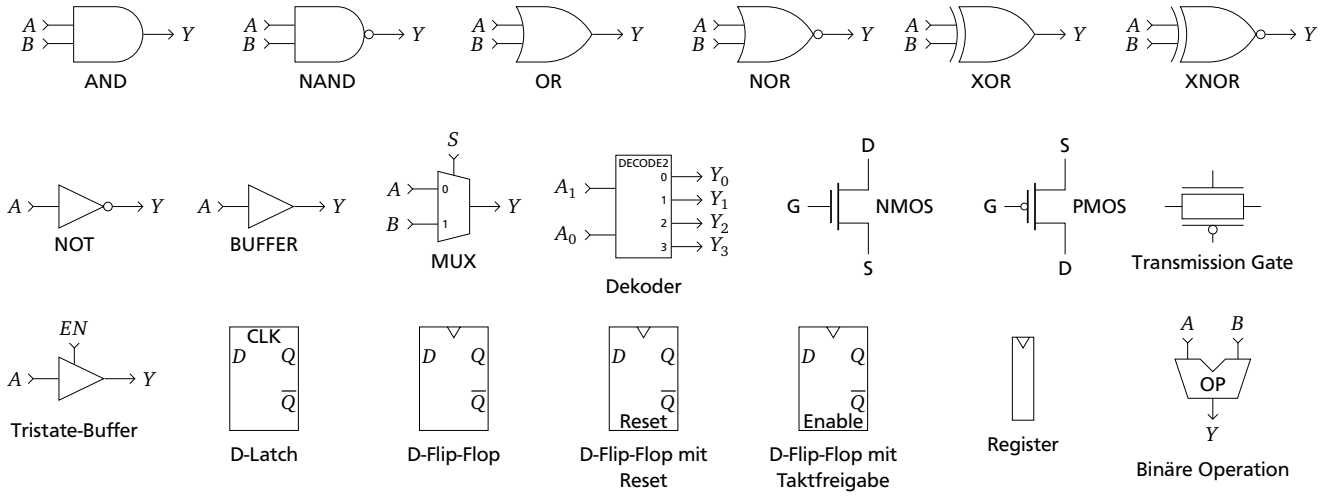
TECHNISCHE UNIVERSITÄT DARMSTADT

Prof. Dr.-Ing. Thomas Schneider, M.Sc. Daniel Günther, M.Sc. Amos Treiber

Einheitenvorsätze

Bezeichnung	Kürzel	Wert	Bezeichnung	Kürzel	Wert	Bezeichnung	Kürzel	Wert
Milli	m	10^{-3}	Kilo	k	10^{3}	Kibi	Ki	2^{10}
Mikro	µ	10^{-6}	Mega	M	10^{6}	Mebi	Mi	2^{20}
Nano	n	10^{-9}	Giga	G	10^{9}	Gibi	Gi	2^{30}
Piko	p	10^{-12}	Tera	T	10^{12}	Tebi	Ti	2^{40}

Schaltsymbole



Axiome und Theoreme der boole'schen Algebra

	Axiom		Dual	Bedeutung
A1	$B \neq 1 \Rightarrow B = 0$	A1'	$B \neq 0 \Rightarrow B = 1$	Dualität
A2	$\overline{0} = 1$	A2'	$\overline{1} = 0$	Negieren
A3	$0 \cdot 0 = 0$	A3'	$1 + 1 = 1$	Und / Oder
A4	$1 \cdot 1 = 1$	A4'	$0 + 0 = 0$	Und / Oder
A5	$0 \cdot 1 = 1 \cdot 0 = 0$	A5'	$1 + 0 = 0 + 1 = 1$	Und / Oder

	Theorem		Dual	Bedeutung
T1	$A \cdot 1 = A$	T1'	$A + 0 = A$	Neutralität
T2	$A \cdot 0 = 0$	T2'	$A + 1 = 1$	Extremum
T3	$A \cdot A = A$	T3'	$A + A = A$	Idempotenz
T4	$\overline{\overline{A}} = A$			Involution
T5	$A \cdot \overline{A} = 0$	T5'	$A + \overline{A} = 1$	Komplement

	Theorem		Dual	Bedeutung
T6	$A\,B = B\,A$	T6'	$A + B = B + A$	Kommutativität
T7	$A\,(B\,C) = (A\,B)\,C$	T7'	$A + (B + C) = (A + B) + C$	Assoziativität
T8	$A\,(B + C) = (A\,B) + (A\,C)$	T8'	$A + (B\,C) = (A + B)\,(A + C)$	Distributivität
T9	$A\,(A + B) = A$	T9'	$A + (A\,B) = A$	Absorption
T10	$(A\,B) + (A\,\overline{B}) = A$	T10'	$(A + B)\,(A + \overline{B}) = A$	Zusammenfassen
T11	$(A\,B) + (\overline{A}\,C) + (B\,C) = (A\,B) + (\overline{A}\,C)$	T11'	$(A + B)\,(\overline{A} + C)\,(B + C) = (A + B)\,(\overline{A} + C)$	Konsensus
T12	$\overline{A\,B\,C\ldots} = \overline{A} + \overline{B} + \overline{C}\ldots$	T12'	$\overline{A + B + C\ldots} = \overline{A}\,\overline{B}\,\overline{C}\ldots$	De Morgan

SystemVerilog Syntax (Auszug)

Modul Deklaration

```
module modul_ID
 #(parameter param_ID = wert)
  (input  datentyp /*[n:m]*/ in_port_ID,
   output datentyp /*[n:m]*/ out_port_ID);

  // lokale Signale
  datentyp /*[n:m]*/ signal_ID /*[k:l]*/;

  // parallele Anweisungen
  assign /* #delay */ signal = ausdruck;
  always sequentielle_anweisung
  submodule #(parameter_map) instanz (port_map);

  // generische Anweisungen
  genvar id;
  generate
    if  (bedingung) begin
      // lokale Signale, parallele Anweisungen
    end
    for (init; cond; step) begin
      // lokale Signale, parallele Anweisungen
    end
  endgenerate
endmodule
```

Sequentielle Anweisungen

```
// Zuweisung
signal  = ausdruck; // blockierend
signal <= ausdruck; // nicht-blockierend

// verzögerte Anweisungen
#delay anweisung
@(ausdruck) anweisung
@(posedge ausdruck) anweisung
@(negedge ausdruck) anweisung
@* anweisung

// bedingte Anweisungen
if (bedingung) anweisung1 else anweisung2
case (ausdruck)
  wert1  : anweisung1
  wert2  : anweisung2
  default: anweisung3
endcase

// wiederholte Anweisung
for (init; cond; step) anweisung

// kombinierte Anweisung
begin anweisung1 anweisung2 ... end
```

Vertikale Gruppierung nach Präzedenz, beginnend mit der höchsten

Operator	Bedeutung
[]	Zugriff auf Vektorelement
~	bitweise NOT
!	logisches NOT
-	unäre Negation
&	unäre Reduktion mit AND
|	unäre Reduktion mit OR
^	unäre Reduktion mit XOR
~&	unäre Reduktion mit NAND
~|	unäre Reduktion mit NOR
~^	unäre Reduktion mit XNOR
**	Exponentialfunktion
*	Multiplikation
/	Division
%	Modulo
+ -	Addition, Subtraktion
<< >>	logischer Shift
<<< >>>	arithmetischer Shift
<	kleiner als
<=	kleiner oder gleich
>	größer als
>=	größer oder gleich
==	gleich
!=	ungleich
===	bitweise gleich
!==	bitweise ungleich
& ~&	bitweise AND, NAND
^ ~^	bitweise XOR, XNOR
| ~|	bitweise OR, NOR
&&	logisches AND
||	logisches OR
?:	ternärer Operator
{}	Konkatenation

Numerische Literale

```
// Bitbreite 'Basis Ziffernfolge
64'h0123456789abcd // hexadezimal
27'd0123456789     // dezimal
24'o01234567       // oktal
 4'bxz01           // binär (vierwertig)
   // x - unbekannt/ungültig
   // z - hochomig
```

Elementare Datentypen

```
bit     // zweiwertige Logik
logic   // vierwertige Logik
byte    // 8 bit signed
integer // 32 bit signed
longint // 64 bit signed
time    // 64 bit signed for Zeitwerte
real    // Gleitkomma-Werte
```

System Funktionen

```
// Basis und Genauigkeit der Simulationszeit setzen
'timescale base / precision;
$time           // aktuelle Systemzeit (als int)
$realtime       // aktuelle Systemzeit (als real)

$clog2(num)     // Logarithmus zur Basis 2
$dumpfile(pfad);// VCD Ausgabedatei setzen
$dumpvars;      // (alle) Signale beobachten
$finish;        // Simulation beenden
$display(format, ausdrücke); // Meldung ausgeben
      // %b binary format
      // %c ASCII character format
      // %d decimal format
      // %h hex format
      // %o octal format
      // %s string format
      // %t time format
```