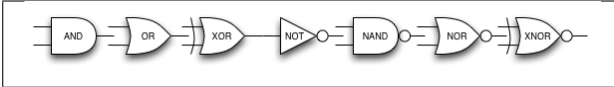


Logische Verknüpfungen

Für N eingänge hat man 2^N Eingangskombinationen.
Elementare Logische Funktionen: NOT, AND, OR, NAND, NOR, XOR, XNOR



Zahlensysteme

Dezimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- LSB: Least Significant Bit - z.B.: 2^0
- Nibble: Gruppe von 4 Bit
- Byte: Gruppe von 8 Bit (2Nibble)
- Word: Gruppe von mehr als 8 Bit (Meisstens 16Bit)
- DWord: Double Word: oft eine Gruppe von 32 Bit

Bsp: 1011 1100 0010 (BIN) = BC2 (HEX)
101 111 000 010 (BIN) = 5702 (OCTAL)

Divisionsmethode:

47 b10 = 101111 b2

47 : 2 = 23 r1

23 : 2 = 11 r1

11 : 2 = 5 r1

5 : 2 = 2 r1

2 : 2 = 1 r0

1 : 2 = 0 r1

LSB

MSB

Schaltalgebra

Funktion	NOR	NAND
NOT	$x \nabla x$	x NAND x
OR	$(x \nabla y) \nabla (x \nabla y)$	(x NAND x) NAND (y NAND y)
AND	$(x \nabla x) \nabla (y \nabla y)$	(x NAND y) NAND (x NAND y)

Vereinfachungen

- Kommutativgesetze
 - A & B = B & A
 - A # B = B # A
- Assoziativgesetze
 - (A & B) & C = A & (B & C)
 - (A # B) # C = A # (B # C)
- Distributivgesetze
 - (A # B) & C = (A & C) # (B & C)
 - (A & B) # C = (A # C) & (B # C)
- Vereinfachungen
 - A # (A & B) = A
 - A & (A # B) = A
 - A # (!A & B) = A # B
 - A & (!A # B) = A & B

Disjunktive Normalform

- OR Verknüpfung von AND Blöcken für K=1
- Jeder AND-Block ist ein MINTERM
- Die DNF K ist eine OR-Verknüpfung aller guten MINTERME (gut = Wahrheitstabelle 1)

Für die Darstellung mit NAND anstelle von OR:
Das DeMorgan Theorem anwenden: $K = \neg(\neg K)$ und dann weiter vereinfachen.

Konjunktive Normalform

- AND Verknüpfung von OR Blöcken
- Herstellen durch DNF von K=0, dann DeMorgan Theorem anwenden
- Jeder OR-Block ist ein MAXTERM, der einer Zeile in der Wahrheitstabelle entspricht, negiert, wenn in der Wahrheitstabelle =1, direkt falls WT=0.

Multiplexer:
Art von Drehschalter, umschalten zwischen verschiedenen Eingängen

Vorzeichenlose und Vorzeichenbehaftete Zahlen

Typ	min	-2	-1	0	1	2	max
Unsigned	-	-	-	0000	0001	0010	1111 (15)
One's Complement	1000 (-7)	1101	1110	0000, 1111	0001	0010	0111 (7)
Two's Complement	1000 (-8)	1110	1111	0000	0001	0010	0111 (7)
Sign Magnitude	1111 (-7)	1010	1001	0000, 1000	0001	0010	0111 (7)

CF: Carry Flag: Übertrag beim Addieren
OF: Overflow Flag: Über oder Unterlaufen

Addition und Subtraktion

Operanden		Addition			Subtraktion		
op1	op2	op1+op2	carry	overflow	op1-op2	borrow	overflow
6C	97	03	1	0			1
76	33	A9	1	1	43	0	0

Addition:

0	1	1	0	1	1	0	0			0	1	1	1	0	1	1	0		
1	0	0	1	0	1	1	1	Carry	Overflow	0	0	1	1	0	0	1	1	Carry	Overflow
1	1	1	1	1	-	-	-	1	1xor1= 0	1	1	1	-	1	1	-	-	0	0xor1= 0
0	0	0	0	0	0	1	1			1	0	1	0	1	0	0	1		

Kombinatorische Schaltungen vs Sequenzielle Schaltungen

- Komb: Reine Schaltnetze, werden nur durch Input beeinflusst, Zeiteinfluss nur von int. Delays
- Seq: Haben “Speicher”, Zustände werden auch von den vorhergehenden Inputs beeinflusst (Zustand des Speichers), Zeitlicher Ablauf spielt eine grosse Rolle

D-Latch (transp. FF)

- D = Input
- C = Control (solange C=1, wird der Input durchgeschaltet)
- Q !Q = Outputs
- Rückkopplung ist gefährlich, gespeicherter Zustand zufällig

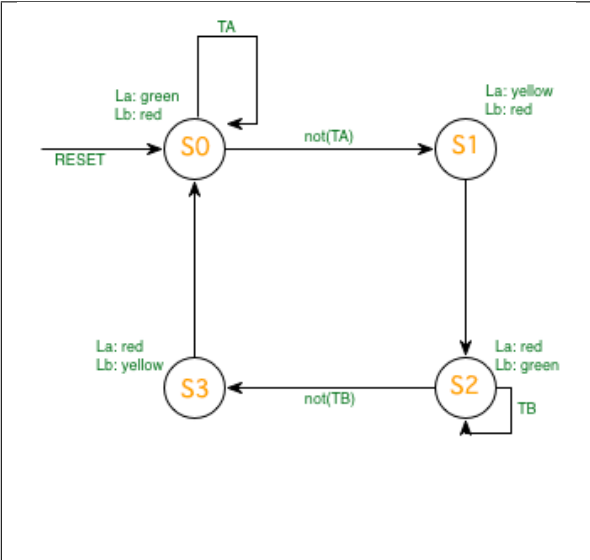
D-Flip-Flop (nichttransp. FF)

- Pos/Neg Flanken gesteuert
- gleiche Funktionsweise wie D-Latch
- Rückkopplung halbiert den Takt
- Set 1 und Reset 0 Funktion -> Synchron mit Takt oder Asynchron sofort

Bausteine

- Register: aus FF mit write-enable: zur Datenspeicherung
- asynchrone Schaltungen: D-FF ohne Clock, Staffelung, arbeiten so schnell wie möglich

Finite State Machine



Current State	CS Enc.	In TA	In TB	Next State
s0	00	0	x	s1
s0	00	1	x	s0
s1	01	x	x	s2
s2	10	x	0	s3
s2	10	x	1	s2
s3	11	x	x	s0

OutPut Table
-> Cur.S. to Output
-> State Encoding

C

Typen und Print Ausgabe

char	1B	−128 bis 127	Zahl	Interpretation	ph	interpret
int	4B	−2 ³¹ bis 2 ³¹	037	oktal Zahl	%d, %i	int
float	4B	−3.4 × 10 ³⁸ bis 3.4 × 10 ³⁸	0x23	hexdezimal Zahl	%u	unsigend int
double	8B	−1.79 × 10 ³⁰⁸ bis 1.79 × 10 ³⁰⁸	3.215f	Float	%c	char
unsigned char	1B	0 bis 255	0 / other	false / true	%s	char* (String bis \0)
unsigned int	4B	0 bis 2 ³² − 1	stdin	Tastatur	%f	double, float
short (int)	2B	−32768 bis 32767	stdout	Konsole		
unsigned short (int)	2B	0 bis 65535	stderr	Konsole		
long (int)	8B	−2 ⁶³ bis 2 ⁶³ − 1				
unsigned long (int)	8B	0 bis 2 ⁶⁴ − 1				
long double	12B	−1.2 × 10 ⁴⁹³² bis 1.2 × 10 ⁴⁹³²				

Funktionen

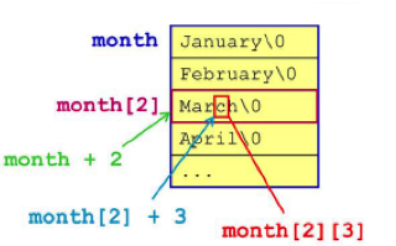
Erlaubte Parameter

- grundlegende Datentypen (Tabelle oben)
- Strukturen (struct)
- Arrays
- Pointer

Erlaubte Rückgabewerttypen

- grundlegende Datentypen (Tabelle oben)
- Strukturen (struct)
- Pointer

ACHTUNG: Arrays können nicht zurückgegeben werden, jedoch Pointer auf Datenbereiche mit Arrays (char*, char**)



Pointer

int *p	Pointer auf int Objekt
double *d[20]	d= Array von 20 Pointer auf double-Objekte
double (*d)[20]	d= Pointer auf ein double-Array mit 20 Objekten
char **ppc	Pointer auf einen Pointer vom Typ char
double *cdp	Pointer auf double, *cdp=5.0, cdp++ möglich
double *const cdp	Konstanter Pointer, *cdp=5.0 möglich
const double *cdp	Pointer auf Konstante, *cdp++ möglich
const double *const cdp	Konstanter Pointer auf Konstante, keine Aktion

Beispiel Code

:-)

#include <stdio.h>	
#include <string.h>	/*Enthält: int strlen(const char s[]); welche die länge des Strings zurückgibt(ohne \0)*/
int max(int a, int b);	
void otherFunction(void);	/*void soll geschrieben werden, damit Parameter checking durchgeführt wird*/
int checkStatic(int a);	
#define MAX_LENGTH 1000	/*Definieren einer Konstante*/
typedef enum {Mo, Di...} Wochentage	/*Montag = 0, Dienstag = 1 etc*/
typedef enum [false, true] Bool;	
typedef struct {	
int x;	
int y;	
int z;	
} Point3D	
int min = 0;	/*Definition globale Variable */
int main(void) {	
int i = 5, a=4, b=6, len;	
double d;	
extern int min;	/*Deklaration globale Variable */
Bool flag = true;	
int data[100];	/*Array Deklaration*/
int a[5] = {4,7,12,77,2};	/*Array Deklaration + Initialisierung*/
int a[] = {4,7,12,77,2};	/*Alternative Array Deklaration + Initialisierung */
const int c[] = {10,11,12,13,14};	/*Konstante Array Deklaration, Werte können NICHT mehr verändert werden*/
int a[2][3] = {{1,2,3},{4,5,6}};	/*Array Deklaration mit 2 Dimensionen (2 Zeilen, 3 Spalten)*/
char hello[] = „hello“;	/*Deklaration eines Strings (Array von Chars)(Grösse +1, da letztes Element: \0)*/
char test[] = „test“;	
a[3] = 4;	/*Zugriff auf Array Element*/
a[9] = 222;	/*Achtung! Es wird kein Fehler zur Laufzeit angezeigt!!*/
len = strlen(hello);	/*len = 5 */
strcat(hello, test);	/*test wird an hello angefügt: char hello[] = „hellotest“*/
strcpy(char dest[], const char source[]);	/*Kopieren eines Strings (char*) */
strcmp(const char s1[], const char s2[]);	/*Vergleicht zwei Strings, gibt zurück: <0, wenn s1 kl., >0 wenn s2 kl., 0 wenn gl.*/
int j;	
int *jp;	
jp = &j;	/* Zuweisung der Adresse von j; jp zeigt jetzt auf j*/
jp = 3;	/ jp wird dereferenziert und dem Objekt wird 3 zugewiesen; j ist jetzt also 3*/
void *vp;	/* Pointer von Typ void, dieser kann auf irgendetwas zeigen*/
jp = NULL;	/* Pointer zeigt explizit „auf nichts“ */
int h[3] = {2,4,6};	
int *pa;	
pa = h;	/* Pointer zeigt nun auf den Array a, genauer auf das erste Element a[0]*/
pa = 7;	/ a[0] ist nun 7*/
*(pa + 3) = 8;	/*a[3] ist nun 8 {pa+i → pa zeigt auf i-te Element) *(pa+3) ist äquiv. zu pa[3]*/
pa = 8;	/ a[0] ist nun 8*/
pa++;	/* Pointer zeigt nun auf das nächste Element im Array*/
char a[] = „hello, Winterthur“;	
char *pa = „hello, Switzerland“;	
[a = pa;]	/* Nicht möglich → Kompilierfehler*/
pa = a;	/*OK, Pointer zeigt nun auf „hello, Winterthur“ */
char* pmonth[12] = {„Jan“, „Feb“, ... }	/*Pointer auf Array, Anstatt 2-Dimensionaler Array*/
pmonth[1];	/*Greift auf February zu*/
*(pmonth[1]+3);	/*Greift auf das ‚r‘ in February zu*/
pmonth[1][3];	/*Greift ebenfalls auf das ‚r‘ in February zu*/
Point3D pt = {2, 4, 6};	
(void)printf(„A=(%d, %d, %d)\n“	
.pt.x,pt.y,pt.z);	

d = i/3;	/* d= 1.0 */
d = (double) i/3;	/*d = 1.66667*/
i = max(a,b);	/*i = b = 6*/
otherFunction();	
Wochtage w1 = Mittwoch;	
(void)printf(„Hello World in C\n“);	
i=scanf(„%d%d%d“, &day,&month,&year);	
(void)printf(“%d”, day);	
for (i = 1; i<=max; i++) {	/*Deklaration int i = 1 geht in c nicht, i muss vorher schon deklariert werden */
/*do something*/	
}	
switch(n) {	
case 1: result = 1;	
break	
case 2: result = 2;	
break	
default: result =3;	
break	
}	
exit(0);	
}	
int max(int a, int b) {	
if(a<b) {	
return b;	
}	
return a;	
}	
void otherFunction(void) {	
extern int max;	/*Deklaration glob. Variable */
int i = 8;	/*Entsteht kein Konflikt mit i aus main */
otherFunction(void);	/* Rekursion, wie in Java*/
}	
int checkStatic(int a) {	
static int max = 0;	/*Deklaration statischer Variable*/
if (a > max) {	/* Statische Variable */
max = a;	/* lokale Variable */
}	
return max;	
}	