LU02g - Logische Operationen

Siehe auch http://www.schoe-berlin.de/Zertifizierung/Grundlagen/Binare_Logik/binare_logik.html

Einleitung

In vielen Bereichen der Informatik werden logische Operationen auf der Ebene von Bits durchgeführt. Bei der binären Logik handelt es sich um Rechenoperationen, die auf binärer Ebene stattfinden. Zum Beispiel wird beim Verschlüsseln von Daten häufig mit XOR (exklusives oder) gearbeitet.

In modernen Programmiersprachen wird das Bit als kleinste Informationseinheit fast nur noch für logische Operationen genutzt. Dennoch muss jeder Informatiker diese Operationen verstehen.

Operationen mit Bits

Wie Sie wissen, kann ein Bit nur zwei Zustände annehmen.

Bit 0	Strom aus	false	falsch
Bit 1	Strom an	true	wahr

In der binären Logik verwenden wir in der Regel die Bezeichnungen **true** bzw. Bit '1' und **false** bzw. Bit '0'.

Die logischen Operationen in diesem Kapitel können einerseits auf einzelne Bits angewandt werden. Dies geschieht vor allem bei der Auswertung von Bedingungen.

Andererseits können auch Ketten von Bits (z.B. 0100 1101 AND 1100 0011) mit logischen Operationen verknüpft werden. Bei solchen Bitketten wird die logische Operation einzeln für jede binäre Stelle angewandt.

Und (AND)

Die logische Operation **AND** vergleicht zwei binär codierte Werte miteinander. Das Resultat dieser Operation ist nur dann **true**, wenn auch die beiden zu vergleichenden Werte **true** sind.

Die folgende Tabelle zeigt das Resultat, wenn zwei einzelne Bits (Wert 1 und Wert 2) mit AND verknüpft werden.

Wert 1	**Wert 2**	**Resultat**
false	false	false
false	true	false
true	false	false
true	true	true

Sie können neben einzelnen Bits auch ganze Bitketten mit AND verküpfen:

Wert 1	0	1	0	0	1	1	0	1
Operation	AND							
Wert 2	1	1	0	0	0	0	1	1
Resultat	0	1	0	0	0	0	0	1

Die logische Operation AND vergleicht die Bits Stelle für Stelle.

- 1. Stelle: 0 AND 1 ⇒ 0
- 2. Stelle: 1 AND 1 ⇒ 1
- usw.

Oder (OR)

Die logische Operation **OR** vergleicht zwei binär codierte Werte miteinander. Das Resultat dieser Operation ist **true**, wenn mindestens einer der zu vergleichenden Werte **true** ist.

Die folgende Tabelle zeigt das Resultat, wenn zwei einzelne Bits (Wert 1 und Wert 2) mit OR verknüpft werden.

Wert 1	**Wert 2**	**Resultat**
false	false	false
false	true	true
true	false	true
true	true	true

Ganze Bitketten mit OR verküpfen:

Wert 1	0	1	0	0	1	1	0	1
Operation	OR							
Wert 2	1	1	0	0	0	0	1	1
Resultat	1	1	0	0	1	1	1	1

Entweder oder (XOR)

Die logische Operation **XOR** vergleicht zwei binär codierte Werte miteinander. Das Resultat dieser Operation ist **true**, wenn **genau** einer der zu vergleichenden Werte **true** ist.

Die folgende Tabelle zeigt das Resultat, wenn zwei einzelne Bits (Wert 1 und Wert 2) mit XOR verknüpft werden.

Wert 1	**Wert 2**	**Resultat**
false	false	false
false	true	true
true	false	true
true	true	false

Auch hier können Sie ganze Bitketten mit XOR verküpfen:

Wert 1	0	1	0	0	1	1	0	1
Operation	XOR							
Wert 2	1	1	0	0	0	0	1	1
Resultat	1	0	0	0	1	1	1	0

Nicht (NOT)

Die logische Operation **NOT** invertiert einen binär codierten Wert.

Wert 1	**Resultat**
false	true
true	false

In einem Programm könnte man anstelle von

```
if (wert1 == true) {
  wert1 = false;
} else {
  wert1 = true;
}
```

einfach schreiben:

```
wert1 = !wert1;
```

Bitketten

Was mit einzelnen Bits möglich ist, können Sie auch mit ganzen Bitketten (eine Abfolge von Bits) machen. Zum Beispiel: `0100 0110 XOR 1101 1100 = 1001 1010`.

Vorgehen

Am einfachsten schreiben Sie die beiden Bitketten untereinander.

```
    0100 0110
XOR 1101 1100
```

Verknpüfen Sie nun jeweils die beiden vordersten Bits die untereinander stehen. Notieren Sie das Ergebnis unterhalb. Zum Beispiel:

```
     0... ....
XOR  1... ....
     ---------
     1... ....
```

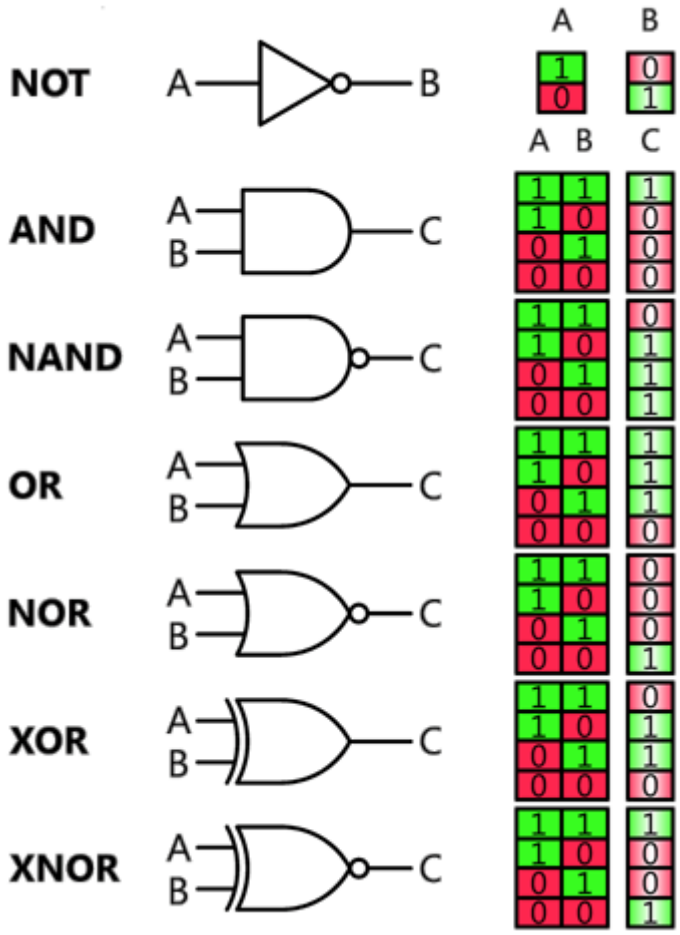
Wiederholen Sie dies für 2. Stelle, die 3. Stelle, ... Zum Schluss erhalten Sie:

```
    0100 0110
XOR 1101 1100
    ---------
    1001 1010
```

Logikgatter

Ein kleiner Exkurs für alle die lieber eine grafische Darstellung haben. Beim Prozessor (CPU) eines Computers besteht das Rechenwerk aus Milliarden von Logikgattern. Jedes Logikgatter führt auf binärer Ebene eine logische Operation mit zwei Bits aus.

Diese Grafik zeigt die übliche Darstellung von Logikgattern in einem Schaltschema:



Quelle: http://elektro.turanis.de/html/prj087/gatter.png

Erklärung

- Links sind die Eingänge. Diese Eingänge führen Strom (1) oder keinen Strom (0).
- Rechts ist der Ausgang bzw. das Resultat. Ob der Ausgang Strom führt, hängt davon ab
  - welche Eingänge Strom führen,

- welches Logikgatter die Eingänge mit dem Ausgang verbindet.

Denkaufgabe

In Schaltungen gibt es zwei bisher nicht besprochene logische Operationen: **NAND** und **NOR**. Andererseits findet sich oft kein **XOR**.

- Welchen zwei der logischen Operatoren AND, OR, XOR, NOT müssen Sie kombinieren, um **NAND** zu erhalten?
- Welchen zwei der logischen Operatoren AND, OR, XOR, NOT müssen Sie kombinieren, um **NOR** zu erhalten?
- Wie müssten Sie die Logikgatter kombinieren, um ein **XOR** zu erhalten?

Lösung

---

m114-A1F, m114-A1E