FACHHOCHSCHULE
TECHNIKUM WIEN

Fachbereich: Embedded Systems

Themengebiet: Computerarchitektur

Kombinatorische Integer-Arithmetik

Version 2.2, August 2007

Peter Balog

Inhaltsverzeichnis

0. Übersicht

0.1. Lehrziele

Ziel des Studienbriefes *Kombinatorische Integer-Arithmetik* ist der Aufbau von Wissen über die Funktionsweise und die Realisierungsmöglichkeiten von elementaren Integer-Arithmetik-Systemen, wie sie in der Computertechnik Anwendung finden.

- Darstellung und Verarbeitung ganzer Zahlen
- Vergleich von ganzen Zahlen
- Addition und Subtraktion
- Fehlerbehandlung

0.2. Lehrinhalt

Ausgehend von der Darstellung von vorzeichenlosen und vorzeichenbehafteten ganzen Zahlen werden Systeme zum Vergleich von Zahlen, zur Addition und zur Subtraktion spezifiziert und realisiert. Besonderes Augenmerk wird auf die Fehlerbehandlung und die Qualifikation des Rechenergebnisses gelegt.

0.3. Voraussetzungen

Elementare Grundkenntnisse der Informatik (bit, Bit, Byte, polyadische Zahlensysteme, binäre und hexadezimale Zahlendarstellung, etc.).

0.4. Anmerkungen

Führen Sie diverse Syntheseaufgaben mit *Espresso-Web* (*Berkeley Espresso Tool*) durch.

1. Integer (Darstellung ganzer Zahlen)

Das generelle Problem bei der digitalen bzw. rechnergestützten Zahlendarstellung liegt darin, dass ein unendlicher Zahlenbereich auf einen endlichen abgebildet werden muss. Aus den, aus der „normalen“ Algebra bekannten Zahlengeraden werden Zahlenkreise. Die damit verbundenen Probleme sind: Fehlerbehandlung bei Bereichsüberlauf und Bereichserweiterung.

- Eine N-Bit vorzeichenlose ganze Zahl (*unsigned integer*) hat den Zahlenbereich:

  $$ZB = \left[0,..,2^N - 1\right]$$

- Eine N-Bit vorzeichenbehaftete ganze Zahl in Zweierkomplementdarstellung (*signed integer in 2’c*) hat den Zahlenbereich:

  $$ZB = \left[-2^{N-1},..,0,..,2^{N-1} - 1\right]$$

Beispiele:

N=4	N=8
$ZB_{vz.los} = \left[0,..,15\right]$	$ZB_{vz.los} = \left[0,..,255\right]$
$ZB_{vz.beh.} = [-8,..,0,..,+7]$	$ZB_{vz.beh.} = [-128,..,0,..,+127]$

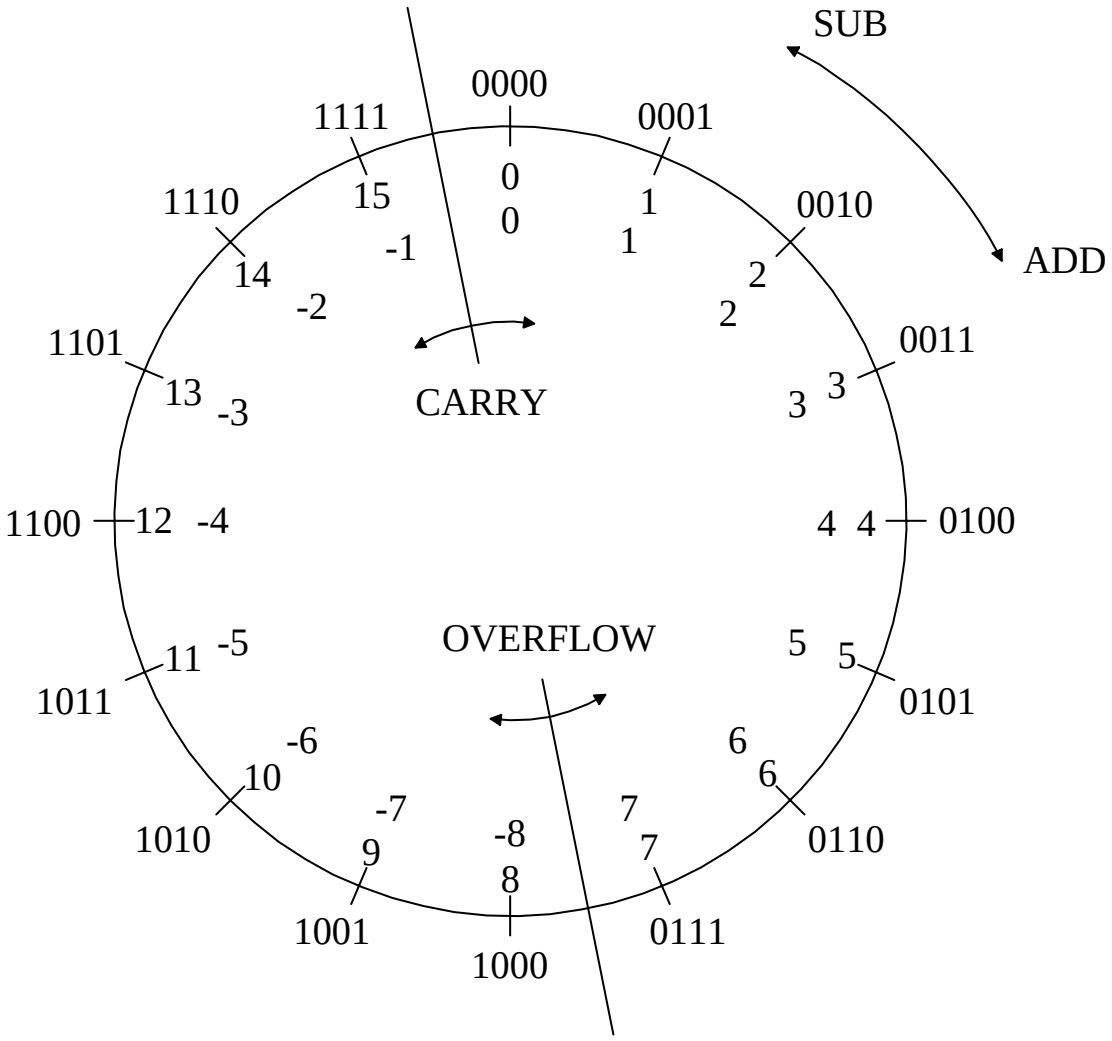
- Vorzeichenwechsel von 2’c-Zahlen:

  $$-Z_{2'c} = \overline{Z_{2'c}} + 1$$

  d.h. die zu negierende 2‘c-Zahl wird bitweise invertiert und dann wird 1 dazuaddiert. Die betragsmäßig größte negative Zahl kann nicht negiert werden, da kein positiver Pedant existiert! (vgl. dazu Bereichsüberlauf weiter unten)

1.1. Der Zahlenkreis

Am Beispiel einer 4-stelligen ganzen Zahl (N=4) wird die Zuordnung der vorzeichenlosen (Mitte) und der vorzeichenbehafteten (innen) Zahlen auf die Bitmuster (außen) gezeigt. Wenn wir von vorzeichenbehafteten Zahlen sprechen beziehen wir uns immer auf die Zweierkomplementdarstellung.



Die Richtung für die Addition ist im Uhrzeigersinn, die Richtung für die Subtraktion ist der Gegenuhrzeigersinn. Die Darstellung der Werte der vorzeichenlosen Zahlen entspricht den Dezimaläquivalenten der Bitmuster.

Wird der Zahlenkreis für vorzeichenlose Zahlen zwischen 0 und 15 überschritten, so liegt ein Fehler vor, der durch den Übertrag (Carry) angezeigt wird.

Vorzeichenbehaftete Zahlen werden so auf die Bitmuster verteilt, dass die Addition und und die Subtraktion wie für vorzeichenlose Zahlen funktionieren. Dass dem Bitmuster 1000 der Wert -8 (und nicht +8) zugeordnet wird ist willkürlich. Jedoch ist diese Willkür dahingehend sinnvoll, dass man so einer vorzeichenbehafteten Zahl auf den ersten Blick ansieht, welches Vorzeichen sie besitzt. Das höchstwertigste Bit wird deshalb auch als Vorzeichenbit (Sign-Bit) bezeichnet. Ist das Sign-Bit=1, dann handelt es sich um eine negative Zahl.

Wird der Zahlenkreis für vorzeichenbehaftete Zahlen zwischen -8 und +7 überschritten, so liegt ein Fehler vor, der als Überlauf (Overflow) bezeichnet wird.

1.2. Bereichsüberlauf, Fehlersignalisierung

Die Fehlersignalisierung nach einer Arithmetikfunktion ist davon abhängig, ob die Bitmuster, die eine ganze Zahl darstellen, vorzeichenlos oder vorzeichenbehaftet interpretiert werden. Für vorzeichenlose Zahlen steht dafür das **Carry-Flag** (CF) und für vorzeichenbehaftete Zahlen das **Overflow-Flag** (OF) zur Verfügung.

Wir betrachten für die folgenden Überlegungen N-Bit breite Operationen, d.h. die Operanden (A, B) und die Ergebnisse (S, D) sind N-Bit breit:

- Addition: $S = A + B$
- Subtraktion: $D = A - B$

Für den vorzeichenlosen Fall (A,B,S,D sind vorzeichenlos) wird ein Fehler durch das Carry-Flag (CF) wie folgt dargestellt:

Addition

$$CF = \begin{cases} 1 \Leftrightarrow S < A \\ 0 \Leftrightarrow S \geq A \end{cases}$$

Subtraktion

$$CF = \begin{cases} 1 \Leftrightarrow D > A \\ 0 \Leftrightarrow D \leq A \end{cases}$$

Die Addition war dann und nur dann fehlerfrei, wenn die Summe größer oder gleich einem Operanden ist. Beispiele für 4-Bit breite Zahlen:

4+5=9, 7+0=7, etc.

Ist jedoch die Summe nach der Addition kleiner als die Operanden, dann liegt ein Fehler vor. Beispiele für 4-Bit breite Zahlen:

9+7=0, 4+15=3, etc.

Während es für die Fehlerbehandlung nach einer Addition egal ist, mit welchem der beiden Operanden die Summe verglichen wird, muss nach der Subtraktion die Differenz mit dem Minuenden (Differenz = Minuend – Subtrahend) verglichen werden. Nach einer korrekten Subtraktion muss natürlich die Differenz kleiner-gleich dem Minuenden sein.

Für den vorzeichenbehafteten Fall (A,B,S,D sind 2'c Zahlen) wird ein Fehler durch das Overflow-Flag (OF) dargestellt.

Aufgabe 1:

Spezifizieren Sie für die Addition und für die Subtraktion die jeweilige OF-Funktion mit Hilfe einer Wahrheitstabelle. (Lösungsansatz siehe Seite 25)

1.3. Zahlenbereichserweiterung

Ist der Zahlenbereich zu klein um die Ergebnisse darstellen zu können, muss er erweitert werden, indem man die Stellenzahl N der beteiligten Variablen auf M erhöht:

$$N \rightarrow M \text{ mit } M > N$$

$$Z_N \rightarrow Z_M : \begin{cases} Z_N = \left[Z_{N,N-1}, .., Z_{N,0} \right] \\ Z_M = \left[Z_{M,M-1}, .., Z_{M,N}, Z_{M,N-1}, .., Z_{M,0} \right] \end{cases}$$

Die M-N Erweiterungsstellen bilden dabei die höherwertigen Bits des erweiterten, nun M-stelligen, Operanden. Die Erweiterungsmethode ist natürlich davon abhängig, ob die Zahlen vorzeichenlos oder vorzeichenbehaftet (2'c) sind.

- Vorzeichenlose Erweiterung (*zero extension*):

$$Z_{M,i} = \begin{cases} Z_{N,i} & \forall i: 0 \le i \le N-1 \\ 0 & \forall i: N \le i \le M-1 \end{cases}$$

  d.h. die zusätzlichen höherwertigen M-N Stellen werden mit 0 initialisiert.

- Vorzeichenrichtige Erweiterung (*sign extension*):

$$Z_{M,i} = \begin{cases} Z_{N,i} & \forall i: 0 \le i \le N-1 \\ Z_{N,N-1} & \forall i: N \le i \le M-1 \end{cases}$$

  d.h. die zusätzlichen höherwertigen M-N Stellen werden mit dem Wert des Vorzeichenbits (höchstwertiges Bit $Z_{N,N-1}$ der zu erweiternden Zahl) der ursprünglichen, N-stelligen Zahl, initialisiert.

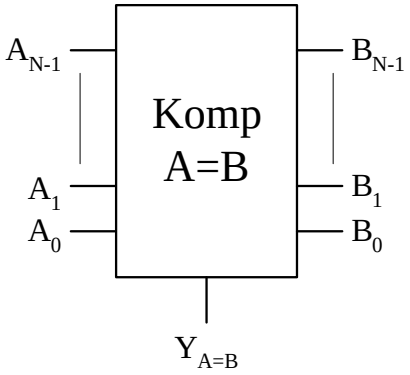
2. Komparatoren

Komparatoren sind digitale Systeme die zwei Zahlen miteinander vergleichen. Ganz allgemein liefern Komparatoren Ergebnisse wie gleich, ungleich, grösser, grösser-gleich, etc.

2.1. Komparator auf Gleichheit

Der einfachste Fall eines Komparators ist der „Komparator auf Gleichheit“. Sein Ergebnis ist natürlich unabhängig von der Zahlendarstellung. Sind die beiden Bitmuster gleich, dann liefert er am Ausgang den Wert 1.

Schaltsymbol



Interface

N A-Eingänge ($A_{N-1},..,A_0$)

N B-Eingänge ($B_{N-1},..,B_0$)

1 Ausgang ($Y_{A=B}$)

Funktionsbeschreibung (verbal)

Der einfache Komparator vergleicht zwei N-stellige Eingangsvektoren (Bitmuster, Zahlen) auf Gleichheit.

Funktionsbeschreibung (formalisiert)

$$Y_{A=B} = \begin{cases} 1 \Leftrightarrow \forall i: 0 \le i \le N-1: A_i = B_i \\ 0 \Leftrightarrow \exists i: 0 \le i \le N-1: A_i \ne B_i \end{cases}$$

Anmerkung:

Der **All-Quantor** ($\forall x : \textit{Bereich für } x : \textit{Bedingung}$) liefert logisch 1 wenn **für alle x** aus dem spezifizierten Bereich die Bedingung erfüllt ist.

Der **Existenz-Quantor** ($\exists x : \textit{Bereich für } x : \textit{Bedingung}$) liefert logisch 1 wenn die Bedingung für **zumindest ein x** aus dem spezifizierten Bereich erfüllt ist.

Logische Gleichung

1. Möglichkeit: $$Y_{A=B} = \overline{\sum_{i=0}^{N-1} \left(A_i \oplus B_i \right)}$$

2. Möglichkeit: $$Y_{A=B} = \prod_{i=0}^{N-1} \left(A_i \equiv B_i \right) = \prod_{i=0}^{N-1} \left(A_i \overline{\oplus} B_i \right)$$

Aufgabe 2:

Wie hängen die 1. und die 2. Möglichkeit der logischen Gleichungen zusammen. (Lösung siehe Seite 26)

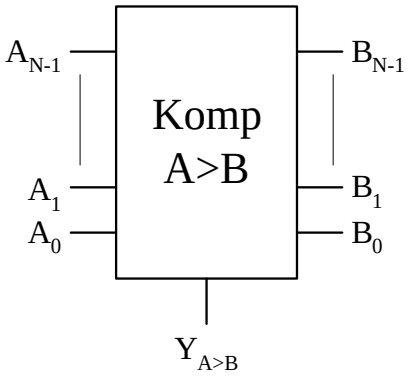
Aufgabe 3:

Warum ist das Ergebnis eines Komparators auf Gleichheit (oder Ungleichheit) unabhängig davon, ob die verglichenen Zahlen vorzeichenlos oder vorzeichenbehaftet sind? (Lösung siehe Seite 26)

2.2. Komparator A>B

Bei einem Vergleich auf „größer“ ist auf die Zahlendarstellung zu achten. Die beiden zu vergleichenden Zahlen A und B sind entweder beide vorzeichenlos oder beide vorzeichenbehaftet! Generell dürfen bei arithmetischen Systemen i.A. die Zahlendarstellung nicht gemischt werden.

Schaltsymbol



Interface

N A-Eingänge (A_{N-1},..,A_0)

N B-Eingänge (B_{N-1},..,B_0)

1 Ausgang ($Y_{A>B}$)

Funktionsbeschreibung (verbal)

Der erweiterte Komparator vergleicht zwei N-stellige ganze Zahlen auf „größer“. Dabei muss man beachten, ob die Zahlen vorzeichenlos oder vorzeichenbehaftet (2’c) interpretiert werden sollen.

Funktionsbeschreibung (formalisiert)

$$Y_{A>B} = \begin{cases} 1 \Leftrightarrow A > B \\ 0 \Leftrightarrow A \leq B \end{cases}$$

Aufgabe 4:

Entwerfen Sie einen 2-Bit „A>B“-Komparator, der sowohl vorzeichenlos ($US_{A>B}$) als auch vorzeichenbehaftet ($S_{A>B}$) vergleicht. (Lösung siehe Seite 26)

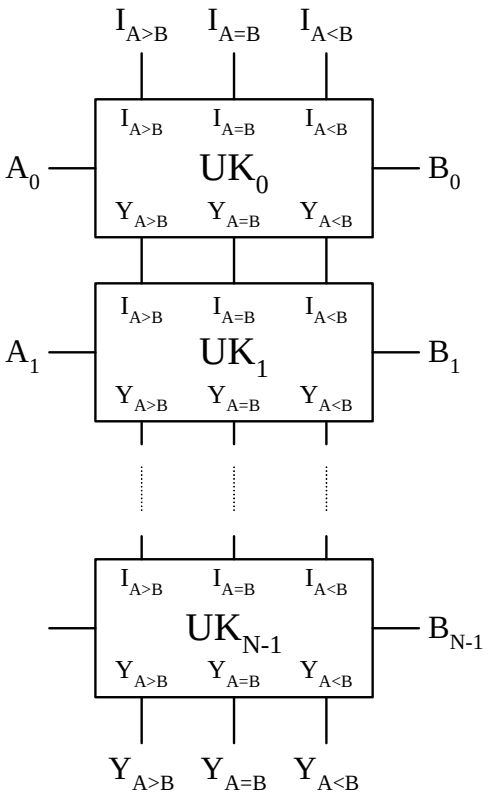
#	A_1	A_0	B_1	B_0	A_{US}	B_{US}	$US_{A>B}$	A_S	B_S	$S_{A>B}$
0	0	0	0	0	0	0		0	0	
1	0	0	0	1	0	1		0	1	
2	0	0	1	0	0	2		0	-2	
3	0	0	1	1	0	3		0	-1	
4	0	1	0	0	1	0		1	0	
5	0	1	0	1	1	1		1	1	
6	0	1	1	0	1	2		1	-2	
7	0	1	1	1	1	3		1	-1	
8	1	0	0	0	2	0		-2	0	
9	1	0	0	1	2	1		-2	1	
10	1	0	1	0	2	2		-2	-2	
11	1	0	1	1	2	3		-2	-1	
12	1	1	0	0	3	0		-1	0	
13	1	1	0	1	3	1		-1	1	
14	1	1	1	0	3	2		-1	-2	
15	1	1	1	1	3	3		-1	-1	

Ausgehend von der gegebenen Wahrheitstabelle tragen Sie die Funktionswerte für $US_{A>B}$ und $S_{A>B}$ ein und synthetisieren die Gleichungen in DNFmin. A_{US} und B_{US} sind die Werte der Eingangsverktoren [A1,A0] und [B1,B0] vorzeichenlos und A_S und B_S entsprechend vorzeichenbehaftet interpretiert und dezimal dargestellt.

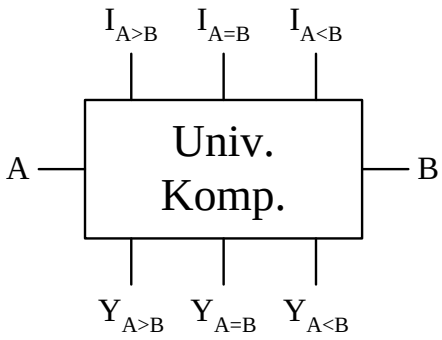
2.3. Der kaskadierbare Universalkomparator

Um Komparatoren für Zahlen mit beliebiger Breite realisieren zu können soll nun versucht werden einen kaskadierbaren 1-Bit Universal-Komparator, zunächst für vorzeichenlose Zahlen, zu entwerfen, der Ergebnisse für „gleich“, „größer“ und „kleiner“ liefert.

Schaltung des Gesamtsystems



Schaltsymbol und Interface



2 Daten-Eingänge (A,B)

3 Kaskadierungs-Eingänge ($I_{A>B}$,$I_{A=B}$,$I_{A<B}$)

3 Ausgänge ($Y_{A>B}$,$Y_{A=B}$,$Y_{A<B}$)

Funktionsbeschreibung (verbal)

Die i-te Stufe vergleicht die beiden Eingänge A_i und B_i und setzt die Y-Ausgänge in Abhängigkeit dieses Vergleichs und der Information auf den Kaskadierungseingängen ($I_{A>B}$,$I_{A=B}$,$I_{A<B}$), die ja das Ergebnis der vorhergehenden Stufen darstellen.

Funktionsbeschreibung (formalisiert)

$$Y_{i,A=B} = \left(A_i \equiv B_i\right) \cdot I_{i,A=B}$$

$$Y_{i,A>B} =$$

$$Y_{i,A<B} =$$

Aufgabe 5:

Spezifizieren Sie die fehlenden Gleichungen in der formalisierten Funktionsbeschreibung. Auf welche Werte müssen die Kaskadierungseingänge der 0-ten Stufe gelegt werden? (Lösung siehe Seite 27)

$I_{0,A=B} =$ $I_{0,A>B} =$ $I_{0,A<B} =$

Aufgabe 6:

Entwerfen Sie die Wahrheitstabelle für den vorzeichenlosen, kaskadierbaren 1-Bit Universalkomparator. (Lösung siehe Seite 28)

Aufgabe 7:

Synthetisieren Sie den 1-Bit Universalkomparator aus der Wahrheitstabelle (Ergebnis aus Aufgabe 6) in eine DNFmin. (Lösung siehe Seite 29)

Aufgabe 8:

Synthetisieren Sie die Schaltung des vorzeichenlosen, kaskadierbaren 1-Bit Universalkomparators direkt aus der (formalisierten) Funktionsbeschreibung. (Lösung siehe Seite 31)

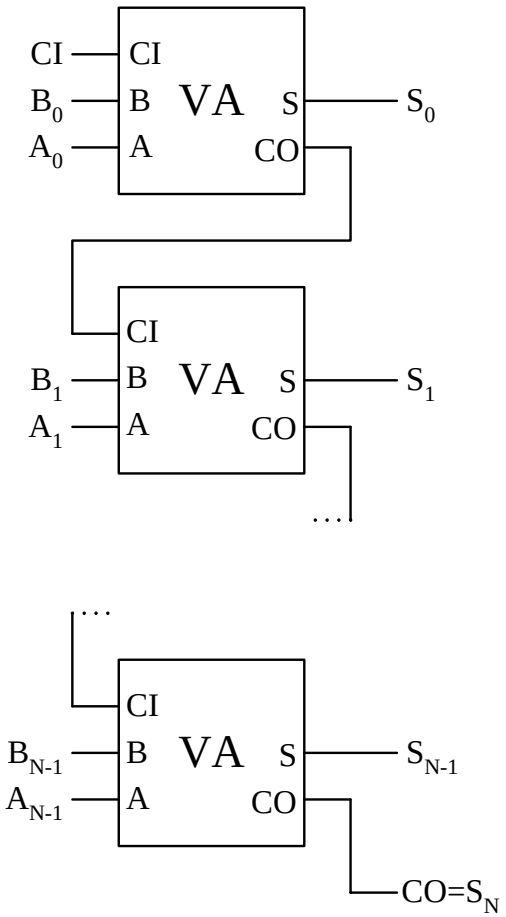
3. Addier- und Subtrahiersysteme

Die Addierer und Subtrahierer werden ausgehend von vorzeichenlosen Zahlen als 1-Bit kaskadierbare Systeme entworfen.

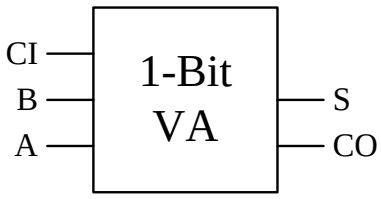
3.1. 1-Bit Volladdierer

Der 1-Bit Volladdierer ist ein kaskadierbarer 1-Bit Addierer für vorzeichenlose ganze Zahlen. Die Kaskadierung erfolgt über CI (*carry in*) und CO (*carry out*).

Schaltung des Gesamtsystems



Schaltsymbol, Interface und Funktion



3 Eingänge (A,B,CIN)

2 Ausgänge (S,COUT)

$$[CO, S] = va([CI, B, A])$$
$$[CO, S] = A + B + CI$$

mit: + ... Addition

CI ... Carry In, CO ... Carry Out

Wahrheitstabelle

CI	B	A	CO	S	[CO,S]
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	2
1	0	0	0	1	1
1	0	1	1	0	2
1	1	0	1	0	2
1	1	1	1	1	3

Der Ausgangsvektor [CO,S] kann als 2-stellige Zahl gelesen werden und repräsentiert so automatisch das korrekte Additionsergebnis (in obiger Wahrheitstabelle dezimal dargestellt) des 1-Bit Volladdierers.

Aufgabe 9:

Synthetisieren Sie die beiden Logikfunktionen für S und CO in eine DNFmin. (Lösung siehe Seite 31)

S=vas(CI,B,A)

A \ CI,B	00	01	11	10
0	0	2	6	4
1	1	3	7	5

S =

CO=vaco(CI,B,A)

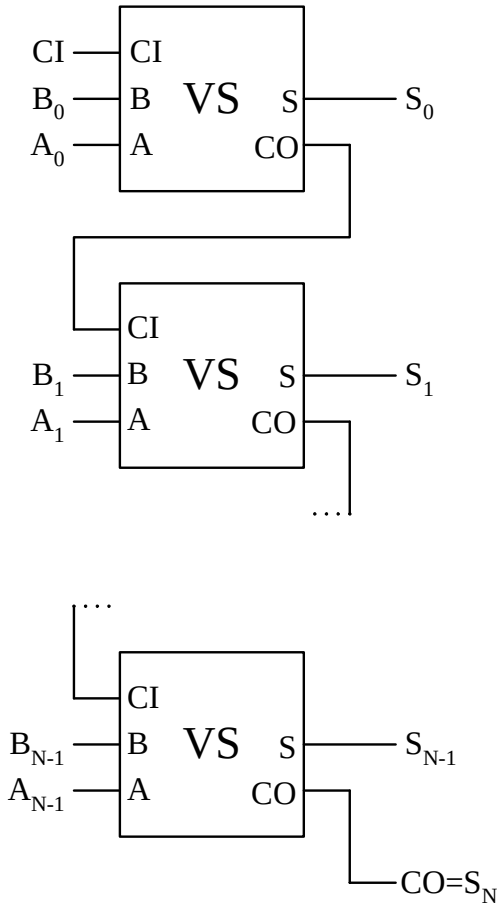
A \ CI,B	00	01	11	10
0	0	2	6	4
1	1	3	7	5

CO =

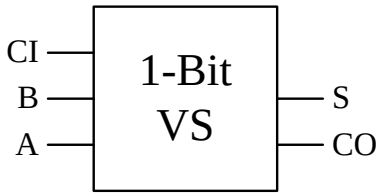
3.2. 1-Bit Vollsubtrahierer

Der 1-Bit Vollsubtrahierer ist ein kaskadierbarer 1-Bit Subtrahierer für vorzeichenlose ganze Zahlen.

Schaltung des Gesamtsystems



Schaltsymbol, Interface und Funktion



3 Eingänge (A,B,CIN)

2 Ausgänge (D,COUT)

$$[CO, D] = vs([CI, B, A])$$

$$[CO, D] = B - (A + CI)$$

mit: + ... Addition

CI ... Carry In, CO ... Carry Out

Wahrheitstabelle

CI	B	A	(A+CI)	B-(A+CI)	CO	D
0	0	0	0	0 - 0	0	0
0	0	1	1	0 - 1	1	1
0	1	0	0	1 - 0	0	1
0	1	1	1	1 - 1	0	0
1	0	0	1	0 - 1	1	1
1	0	1	2	0 - 2	1	0
1	1	0	1	1 - 1	0	0
1	1	1	2	1 - 2	1	1

Damit die Wahrheitstabelle leichter lesbar wird, wurden die Spalten für die Zwischenrechnung (A+CI) und B-(A+CI) eingeführt und dezimal dargestellt.

Aufgabe 10:

Synthetisieren Sie die beiden Logikfunktionen für D und CO in eine DNFmin. (Lösung siehe Seite 31)

D=vsd(CI,B,A)

A \ CI,B	00	01	11	10
0	0	2	6	4
1	1	3	7	5

D =

CO=vsco(CI,B,A)

A \ CI,B	00	01	11	10
0	0	2	6	4
1	1	3	7	5

CO =

Aufgabe 11:

Auf welchen Wert muss CI_0 gelegt werden? (Lösung siehe Seite 31)
Beim Addierer: CI_0= beim Subtrahierer: CI_0=

Aufgabe 12:

Wie hängen die Summenfunktion **vas(CI,B,A)** des Addierers und die Differenzfunktion **vsd(CI,B,A)** des Subtrahierers zusammen? (Lösung siehe Seite 31)

Aufgabe 13:

Wie hängen die *Carry-Out*-Funktion **vaco(CI,B,A)** des Addierers und die *Carry-Out*-Funktion **vsco(CI,B,A)** des Subtrahierers zusammen? (Lösung siehe Seite 32)

Aufgabe 14:

Kann ein vorzeichenloser N-Bit Addierer für vorzeichenbehaftete Zahlen (2'c) eingesetzt werden? (Lösung siehe Seite 32)

Aufgabe 15:

Entwickeln Sie eine 1-Bit Addier-/Subtrahierzelle für vorzeichenlose und vorzeichenbehaftete Zahlen. (Lösung siehe Seite 32)

4. Flags (Condition Codes)

Es steht eine N-Bit Arithmetikeinheit zur Verfügung die sowohl vorzeichenlose als auch vorzeichenbehafte Zahlen addieren und subtrahieren kann. Neben dem N-Bit Ergebnis liefert diese Arithmetikeinheit auch noch **4 Flags**, oft auch als *Condition-Codes* bezeichnet, die das Ergebnis qualifizieren.

Carry-Flag (CF)	Das CF zeigt an, ob das vorzeichenlos betrachtete Ergebnis richtig (CF=0) oder falsch (CF=1) ist. Weiters wird das CF für die Kaskadierung von Addierern und Subtrahierern verwendet.
Overflow-Flag (OF)	Das OF zeigt an, ob das vorzeichenbehaftet betrachtete Ergebnis richtig (OF=0) oder falsch (OF=1) ist.
Zero-Flag (ZF)	Das ZF zeigt an, ob alle Ergebnisbits 0 sind (ZF=1) oder nicht (ZF=0).
Sign-Flag (SF)	Das SF zeigt an, ob das Ergebnis vorzeichenbehaftet betrachtet negativ (SF=1) oder positiv (SF=0) ist.

Aufgabe 16:

Wie wird das Sign-Flag realisiert? (Lösung siehe Seite 32)

Aufgabe 17:

Wie wird das Zero-Flag realisiert? (Lösung siehe Seite 32)

Beispiele:

Es steht eine 4-Bit Addier-/Subtrahier-Einheit zur Verfügung, welche neben dem 4-Bit Ergebnis auch die Flags CF, OF, ZF und SF liefert.

						vz.los	vz.beh.	Flags			
B		0	0	1	1	3	3	C	O	Z	S
A	+	0	1	0	0	4	4	0	0	0	0
ERG		0	1	1	1	7	7				

						vz.los	vz.beh.	Flags			
B		0	1	1	0	6	6	C	O	Z	S
A	+	1	0	1	0	10	-6	1	0	1	0
ERG		0	0	0	0	0	0				

						vz.los	vz.beh.	Flags			
B		0	0	1	0	2	2	C	O	Z	S
A	-	0	1	0	1	5	5	1	0	0	1
ERG		1	1	0	1	13	-3				

5. Lehrzielorientierte Fragen

1. Was ist die Idee der 2‘c Darstellung für vorzeichenbehaftete Zahlen?
2. Wie erfolgt die Zahlenbereichserweiterung für Integer und wovon ist sie abhängig?
3. Wie hängen der All-Quantor und der Existenz-Quantor zusammen?
4. Welche Funktionen hat das Carry-Bit bei einem kombinierten Additions- und Subtraktionsssystem?
5. Ist die Berechnung des Overflowflags von der Bitbreite der Operanden abhängig?
6. Ist die Berechnung des Signflags von der Bitbreite der Operanden abhängig?
7. Ist die Berechnung des Zeroflogs von der Bitbreite der Operanden abhängig?
8. Was ist die Aussage des Signflags für vorzeichenlose Operationen?
9. Was ist die Aussage des Overflowflags für vorzeichenlose Operationen?
10. Was ist die Aussage des Carryflags für vorzeichenbehaftete Operationen?

5.1. Antworten auf die lehrzielorientierten Fragen

1. Die Additions/Subtraktions Vorschrift wird von dem vorzeichenlosen Zahlenbereich in den vorzeichenbehafteten Zahlenbereich „hinübergerettet“.
2. Höherwertige Stellen hinzufügen. Abhängig ob vorzeichenlos oder vorzeichenbehaftet. Bei vorzeichenlos mit 0 auffüllen und bei vorzeichenbehaftet mit dem Wert des höchstwertigen Bits.
3. Liefert der All-Quantor den Wert 0 so existiert (mindestens) ein Wert für den die inverse Bedingung des All-Quantors erfüllt ist,- das klingt nach DeMorgan !
4. Fehleranzeige (Bereichsüberlauf) für vorzeichenlose Zahlen und zur Kaskadierung.
5. Nein,- nur von den höchstwertigen Bits der Operanden und des Ergebnisses.
6. Nein
7. Ja
8. Keine Aussage.
9. Keine Aussage.
10. Keine Aussage.

6. Lösungen

Lösungsansatz zu Aufgabe 1:

Um das OF zu ermitteln benötigt man lediglich die Sign-Bits der beiden Operanden A und B sowie das Sign-Bit des Ergebnisses. Die OF-Berechnung ist unabhängig von der Bitbreite des Addierers bzw. Subtrahierers.

#	A_{N-1}	B_{N-1}	S_{N-1}	OF	A + B = S
0	0	0	0	0	pos+pos=pos: korrekt
1	0	0	1	1	pos+pos=neg: falsch
2	0	1	0	0	usw.
3	0	1	1	0	
4	1	0	0	0	
5	1	0	1	0	
6	1	1	0	1	
7	1	1	1	0	

#	A_{N-1}	B_{N-1}	D_{N-1}	OF	A – B = D
0	0	0	0	0	pos-pos=pos: möglich, korrekt
1	0	0	1	0	pos-pos=neg: möglich, korrekt
2	0	1	0	0	usw.
3	0	1	1	1	
4	1	0	0	1	
5	1	0	1	0	
6	1	1	0	0	
7	1	1	1	0	

Lösung zu Aufgabe 2:

Über das DeMoransche Gesetz. Die 1. Möglichkeit ist eine NOR-Verknüpfung über XOR-Verknüpfungen. Nach Anwendung der DeMorgan´schen Regel wird (i) aus der NOR-Verknüpfung eine UND-Verknüpfung und (ii) aus den XOR-Verknüpfungen werden ÄQUIVALENZ-Verknüpfungen (vgl. dazu auch Studienbrief *Grundlagen digitaler Systeme*).

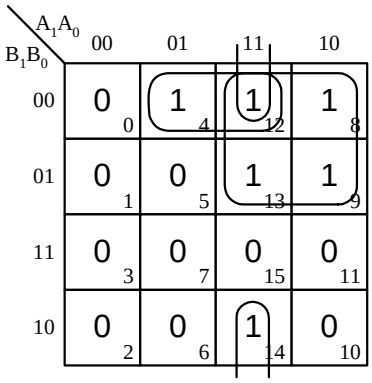
Lösung zu Aufgabe 3:

Weil unabhängig von der Zahlendarstellung der Vergleich auf Gleichheit / Ungleichheit auf einen Vergleich von Bitmuster zurückgeführt werden kann.

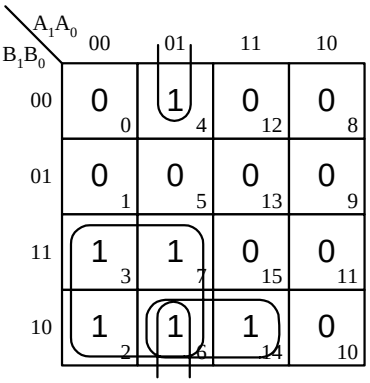
Lösung zu Aufgabe 4:

#	A_1	A_0	B_1	B_0	A_{US}	B_{US}	$US_{A>B}$	A_S	B_S	$S_{A>B}$
0	0	0	0	0	0	0	**0**	0	0	**0**
1	0	0	0	1	0	1	**0**	0	1	**0**
2	0	0	1	0	0	2	**0**	0	-2	**1**
3	0	0	1	1	0	3	**0**	0	-1	**1**
4	0	1	0	0	1	0	**1**	1	0	**1**
5	0	1	0	1	1	1	**0**	1	1	**0**
6	0	1	1	0	1	2	**0**	1	-2	**1**
7	0	1	1	1	1	3	**0**	1	-1	**1**
8	1	0	0	0	2	0	**1**	-2	0	**0**
9	1	0	0	1	2	1	**1**	-2	1	**0**
10	1	0	1	0	2	2	**0**	-2	-2	**0**
11	1	0	1	1	2	3	**0**	-2	-1	**0**
12	1	1	0	0	3	0	**1**	-1	0	**0**
13	1	1	0	1	3	1	**1**	-1	1	**0**
14	1	1	1	0	3	2	**1**	-1	-2	**1**
15	1	1	1	1	3	3	**0**	-1	-1	**0**

KV-Diagramm für $US_{A>B}$



KV-Diagramm für $S_{A>B}$



$$US_{A>B} = A_1 \cdot \overline{B_1} + A_0 \cdot \overline{B_1} \cdot \overline{B_0} + A_1 \cdot A_0 \cdot \overline{B_0}$$

$$S_{A>B} = \overline{A_1} \cdot B_1 + A_0 \cdot B_1 \cdot \overline{B_0} + \overline{A_1} \cdot A_0 \cdot \overline{B_0}$$

Lösung zu Aufgabe 5:

$$Y_{i,A=B} = \left(A_i \equiv B_i\right) \cdot I_{i,A=B}$$

Der $Y_{A=B}$ Ausgang einer Stufe darf nur dann gesetzt werden, wenn (i) die zu vergleichenden Bits gleich sind und (ii) auch alle niederwertigeren Stufen ein Gleichheitsergebnis lieferten. Die Information der niederwertigen Stufen kumuliert in den Ausgängen der nächst niederwertigeren Stufe, welche die Kaskadierungseingänge der gerade betrachteten Stufe ansteuern.

$$Y_{i,A>B} = \left(A_i > B_i\right) + \left(A_i \equiv B_i\right) \cdot I_{i,A>B}$$

Der $Y_{A>B}$ Ausgang einer Stufe liefert 1, wenn das A-Bit größer dem B-Bit ist unabhängig von den Kaskadierungseingängen. Sind die beiden Bits A und B jedoch gleich, dann hängt das Ergebnis vom Ergebnis der vorherigen Stufe ab. Analoges gilt für den Fall $Y_{A<B}$:

$$Y_{i,A<B} = \left(A_i < B_i\right) + \left(A_i \equiv B_i\right) \cdot I_{i,A<B}$$

Die Eingänge der Stufe 0 müssen auf den Gleichheitsfall gesetzt werden:

$$I_{0,A=B} = 1 \qquad I_{0,A>B} = 0 \qquad I_{0,A<B} = 0$$

Lösung zu Aufgabe 6:

#	$I_{A>B}$	$I_{A=B}$	$I_{A<B}$	A	B	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$	Kommentar
0	0	0	0	0	0				Kask.eing. nicht zulässig
1	0	0	0	0	1				Kask.eing. nicht zulässig
2	0	0	0	1	0				Kask.eing. nicht zulässig
3	0	0	0	1	1				Kask.eing. nicht zulässig
4	0	0	1	0	0	0	0	1	
5	0	0	1	0	1	0	0	1	
6	0	0	1	1	0	1	0	0	
7	0	0	1	1	1	0	0	1	
8	0	1	0	0	0	0	1	0	
9	0	1	0	0	1	0	0	1	
10	0	1	0	1	0	1	0	0	
11	0	1	0	1	1	0	1	0	
12	0	1	1	0	0				Kask.eing. nicht zulässig
13	0	1	1	0	1				Kask.eing. nicht zulässig
14	0	1	1	1	0				Kask.eing. nicht zulässig
15	0	1	1	1	1				Kask.eing. nicht zulässig
16	1	0	0	0	0	1	0	0	
17	1	0	0	0	1	0	0	1	
18	1	0	0	1	0	1	0	0	
19	1	0	0	1	1	1	0	0	
20	1	0	1	0	0				Kask.eing. nicht zulässig
21	1	0	1	0	1				Kask.eing. nicht zulässig
22	1	0	1	1	0				Kask.eing. nicht zulässig
23	1	0	1	1	1				Kask.eing. nicht zulässig
24	1	1	0	0	0				Kask.eing. nicht zulässig
25	1	1	0	0	1				Kask.eing. nicht zulässig
26	1	1	0	1	0				Kask.eing. nicht zulässig
27	1	1	0	1	1				Kask.eing. nicht zulässig
28	1	1	1	0	0				Kask.eing. nicht zulässig
29	1	1	1	0	1				Kask.eing. nicht zulässig
30	1	1	1	1	0				Kask.eing. nicht zulässig
31	1	1	1	1	1				Kask.eing. nicht zulässig

Wenn die Kaskadierungseingänge ungültig sind, dann werden die Ausgänge 0 gesetzt, was für die nächst höhere Stufe wiederum einen ungültigen Wert darstellt. Geht man davon aus, dass die Stufe 0 ordnungsgemäß angesteuert wird, dann kann nie ein ungültiger Zustand auftreten. Unter dieser Voraussetzung können die Ausgänge auch auf X (don’t care) gesetzt werden.

Lösung zu Aufgabe 7:

In der Wahrheitstabelle werden alle Ausgänge für ungültige Eingangssituationen auf 0 gesetzt. Da die zu synthetisierenden Funktionen selten den Wert 1 liefern, geht man der Einfachheit halber von der KDNF aus und minimiert rechnerisch.

Das ON-Set für $Y_{A=B}$ besteht aus den Mintermen m_8 und m_{11} die nicht zusammengefasst werden können:

$$Y_{i,A=B} = \overline{I_{A>B}} \cdot I_{A=B} \cdot \overline{I_{A<B}} \cdot \overline{A} \cdot \overline{B} + \overline{I_{A>B}} \cdot I_{A=B} \cdot \overline{I_{A<B}} \cdot A \cdot B$$

Das ON-Set für $Y_{A>B}$ besteht aus den Mintermen m_6, m_{10}, m_{16}, m_{18} und m_{19}. Die Minterme m_6 und m_{10} haben keine Nachbarn, die Minterme m_{16} und m_{18} sowie m_{18} und m_{19} bilden zwei Zweierblöcke:

$$Y_{i,A>B} = \overline{I_{A>B}} \cdot \overline{I_{A=B}} \cdot I_{A<B} \cdot A \cdot \overline{B} + \overline{I_{A>B}} \cdot I_{A=B} \cdot \overline{I_{A<B}} \cdot A \cdot \overline{B} + I_{A>B} \cdot \overline{I_{A=B}} \cdot \overline{I_{A<B}} \cdot \overline{B} + I_{A>B} \cdot \overline{I_{A=B}} \cdot \overline{I_{A<B}} \cdot A$$

Das ON-Set für $Y_{A<B}$ besteht aus den Mintermen m_4, m_5, m_7, m_9 und m_{17}. Die Minterme m_9 und m_{17} haben keine Nachbarn, die Minterme m_4 und m_5 sowie m_5 und m_7 bilden zwei Zweierblöcke:

$$Y_{i,A<B} = \overline{I_{A>B}} \cdot \overline{I_{A=B}} \cdot I_{A<B} \cdot \overline{A} + \overline{I_{A>B}} \cdot \overline{I_{A=B}} \cdot I_{A<B} \cdot B + \overline{I_{A>B}} \cdot I_{A=B} \cdot \overline{I_{A<B}} \cdot \overline{A} \cdot B + I_{A>B} \cdot \overline{I_{A=B}} \cdot \overline{I_{A<B}} \cdot \overline{A} \cdot B$$

Lösung zu Aufgabe 8:

Synthese von $Y_{i,A=B} = \left(A_i \equiv B_i\right) \cdot I_{i,A=B}$:

Für die Äquivalenz (XNOR Funktion) wird die entsprechende DNF eingesetzt:

$$Y_{i,A=B} = \left(A_i \cdot B_i + \overline{A_i} \cdot \overline{B_i}\right) \cdot I_{i,A=B} = A_i \cdot B_i \cdot I_{i,A=B} + \overline{A_i} \cdot \overline{B_i} \cdot I_{i,A=B}$$

Synthese von $Y_{i,A>B} = (A_i > B_i) + (A_i \equiv B_i) \cdot I_{i,A>B}$:

Die Wahrheitstabelle für die Funktion A>B lautet:

A	B	A>B
0	0	0
0	1	0
1	0	1
1	1	0

Den Ausdruck A>B durch die DNF aus der Wahrheitstabelle ersetzen:

$Y_{i,A>B} = \left(A_i \cdot \overline{B_i}\right) + A_i \cdot B_i \cdot I_{i,A>B} + \overline{A_i} \cdot \overline{B_i} \cdot I_{i,A>B}$ gefolgt von 2 Absorptionen:

$Y_{i,A>B} = A_i \cdot \overline{B_i} + A_i \cdot I_{i,A>B} + \overline{B_i} \cdot I_{i,A>B}$

Synthese von $Y_{i,A<B} = (A_i < B_i) + (A_i \equiv B_i) \cdot I_{i,A<B}$:

Die Wahrheitstabelle für die Funktion A<B lautet:

A	B	A<B
0	0	0
0	1	1
1	0	0
1	1	0

Den Ausdruck A<B durch die DNF aus der Wahrheitstabelle ersetzen:

$Y_{i,A<B} = \left(\overline{A_i} \cdot B_i\right) + A_i \cdot B_i \cdot I_{i,A<B} + \overline{A_i} \cdot \overline{B_i} \cdot I_{i,A<B}$ gefolgt von 2 Absorptionen:

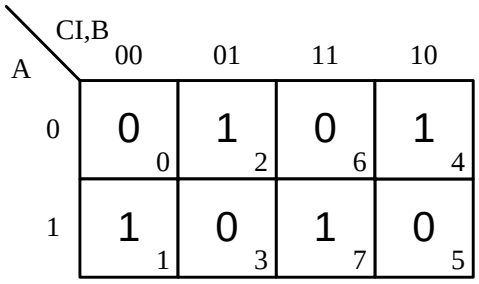
$Y_{i,A<B} = \overline{A_i} \cdot B_i + B_i \cdot I_{i,A<B} + \overline{A_i} \cdot I_{i,A<B}$

Anmerkung:

Versuchen Sie zu erklären warum sich die Ergebnisse von den Ergebnissen der Lösung zu Aufgabe 7 unterscheiden.

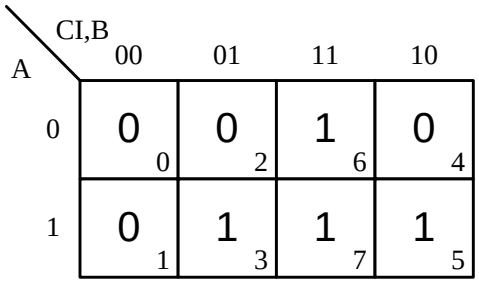
Lösung zu Aufgabe 9:

S=vas(CI,B,A)



$$S = A \oplus B \oplus CI$$

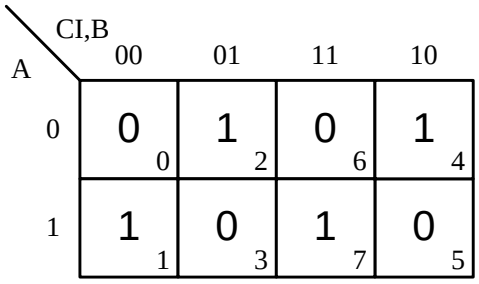
CO=vaco(CI,B,A)



$$CO = A \cdot B + A \cdot CI + B \cdot CI$$

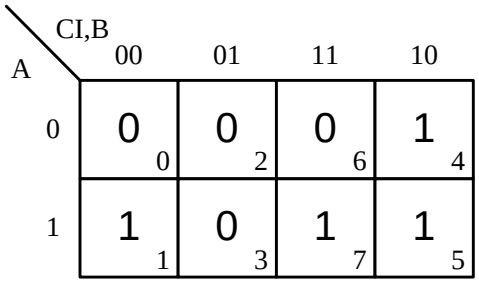
Lösung zu Aufgabe 10:

D=vsd(CI,B,A)



$$D = A \oplus B \oplus CI$$

CO=vsco(CI,B,A)



$$CO = A \cdot \overline{B} + A \cdot CI + \overline{B} \cdot CI$$

Lösung zu Aufgabe 11:

Beim Addierer: CI_0= **0** beim Subtrahierer: CI_0= **0**

Lösung zu Aufgabe 12:

Die beiden Funktionen sind ident.

Sollte Sie dieses Ergebnis wundern, dann stellen Sie sich einen 1-Bit Zahlenkreis vor und addieren Sie zu 0 den Wert 1 dazu und anschließend subtrahieren Sie von 0 die Zahl 1...

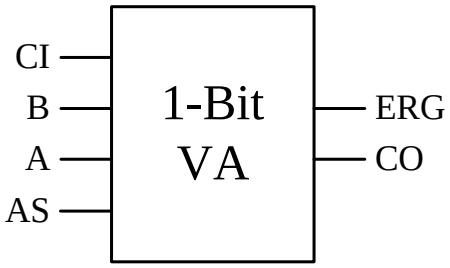
Lösung zu Aufgabe 13:

$$vsco(CI, B, A) = vaco(CI, \overline{B}, A)$$

Lösung zu Aufgabe 14:

Ja.

Lösung zu Aufgabe 15:



Additions / Subtraktions Steuerleitung AS:

AS=0 bedeutet Addition

AS=1 bedeutet Subtraktion

$$ERG = A \oplus B \oplus CI$$

$CO = A \cdot B_{INT} + A \cdot CI + B_{INT} \cdot CI$ wobei gilt:

$B_{INT} = AS \oplus B$ d.h. mit der Steuerleitung AS wird der Eingang B bei der Subtraktion invertiert zu B_{INT}. Die XOR-Funktion wird hier als sogenannter „gesteuerter Inverter“ verwendet.

Lösung zu Aufgabe 16:

Kopie des höchstwertigsten Ergebnisbits.

Lösung zu Aufgabe 17:

NOR-Verknüpfung über alle Ergebnisbits.