1.3.F Einführung und Zahlensysteme/Rechnen mit Dualzahlen – Ergänzungen und Bilder

1.3.F.1 1-Bit-Addierer (Halbaddierer)

Tabelle 1 zeigt die Wahrheitstabelle eines Addierers für ein Bit. Das Ergebnis besteht aus den Stellen E (Ergebnis) und Co (Übertrag, *carry output*). Aus Gründen, die noch folgen, nennt man diesen Addierer einen *Halbaddierer*. Abbildung 1 zeigt eine möglich Realisierung.

A	B	Co	E	
0	0	0	0	$0 + 0 = 00$
0	1	0	1	$0 + 1 = 01$
1	0	0	1	$1 + 0 = 00$
1	1	1	0	$1 + 1 = 10_{(2)}$

Tabelle 1: Wahrheitstabelle des Halbaddierers

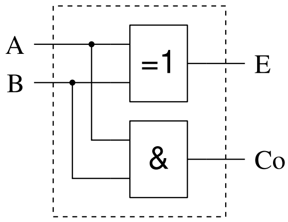

Abbildung 1: Mögliche Realisierung des Halbaddierers

1.3.F.2 4-Bit-Addierer, erste Version

Mehrstellige Zahlen kann man Bit für Bit addieren, hier am Beispiel zweier 4-Bit-Zahlen :

$$\begin{array}{ccccc|l} & 1 & 0 & 1 & 0 & \text{Zahl A} \\ + & 0 & 1 & 0 & 1 & \text{Zahl B} \\ \hline & 1 & 1 & 1 & 1 & \text{Ergebnis E} \end{array}$$

Abbildung 2 zeigt den Entwurf eines 4-Bit-Addierer aus vier Halbaddierern. Aber bei manchen

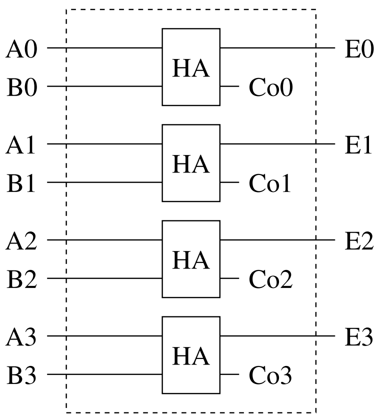

Abbildung 2: 4-Bit-Addierer aus vier Halbaddierern (Entwurf)

Berechnungen gibt es ein Problem:

	0	0	1	0	Zahl A
+	0	0	1	1	Zahl B
	0	0	**10**	1	Ergebnis E

Das Ergebnis in der dritten Spalte ist 2; als Dualzahl schreibt man $10_{(2)}$, man braucht dort also zwei Stellen. Es entsteht also ein *Übertrag*, den man als weiteren Summand in die nächsthöhere Spalte (also nach rechts) überträgt:

	0	0	1	0	Zahl A
+	0	0	1	1	Zahl B
		1			Übertrag
	0	0	**0**	1	Ergebnis E (falsch)

Das Ergebnis ist noch falsch, denn hier brauchte man einen 1-Bit-Addierer, der einen dritten Eingang, den Übertrags-Eingang (Ci=*carry input*) hat. Mit ihm sähe die Rechnung so aus:

	0	0	1	0	Zahl A
+	0	0	1	1	Zahl B
		1			Übertrag
	0	1	**0**	1	Ergebnis E (richtig)

1.3.F.3 1-Bit-Addierer mit Übertrags-Eingang (Volladdierer)

Den 1-Bit-Addierer mit Übertrags-Eingang (also mit drei Eingängen) nennt man *Volladdierer*. Tabelle 2 zeigt die Wahrheitstabelle.

A	B	Ci	Co	E	
0	0	0	0	0	$0+0+0=00$
0	0	1	0	1	$0+0+1=01$
0	1	0	0	1	$0+1+0=00$
0	1	1	1	0	$0+1+1=10_{(2)}$
1	0	0	0	1	$1+0+0=01$
1	0	1	1	0	$1+0+1=10_{(2)}$
1	1	0	1	0	$1+1+0=10_{(2)}$
1	1	1	1	1	$1+1+1=11_{(2)}$

Tabelle 2: Wahrheitstabelle des Volladdierers

In Abbildung 3 sieht man eine mögliche Realisierung des Volladdierers. In Abbildung 4 kann man sehen, wie ein Volladdierer aus zwei Halbaddierern gebaut werden kann.

1.3.F.4 4-Bit-Addierer, zweite Version

Mit diesen Volladdierern kann nun tatsächlich ein mehrstelliger Addierer aufgebaut werden (Abbildung 5).

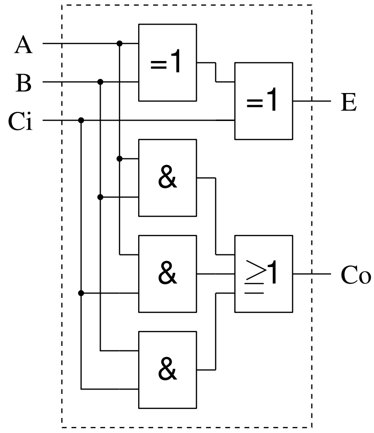

Abbildung 3: Mögliche Realisierung des Volladdierers

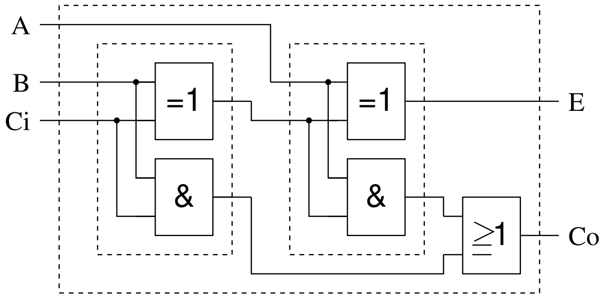

Abbildung 4: Andere Realisierung des Volladdierers

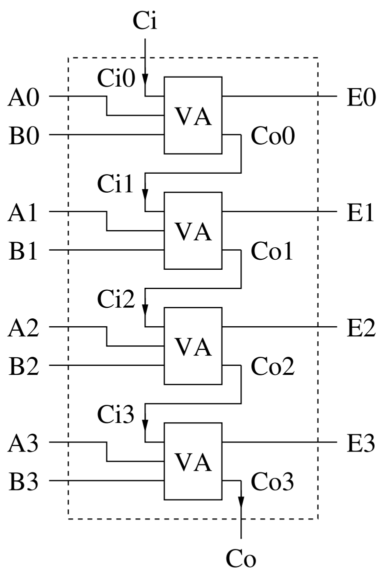

Abbildung 5: 4-Bit-Addierer aus vier Volladdierern