Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Prof. Dr.-Ing. Dr. h. c. J. Becker

becker@kit.edu

Karlsruher Institut für Technologie (**KIT**)

Institut für Technik der Informationsverarbeitung (ITIV)

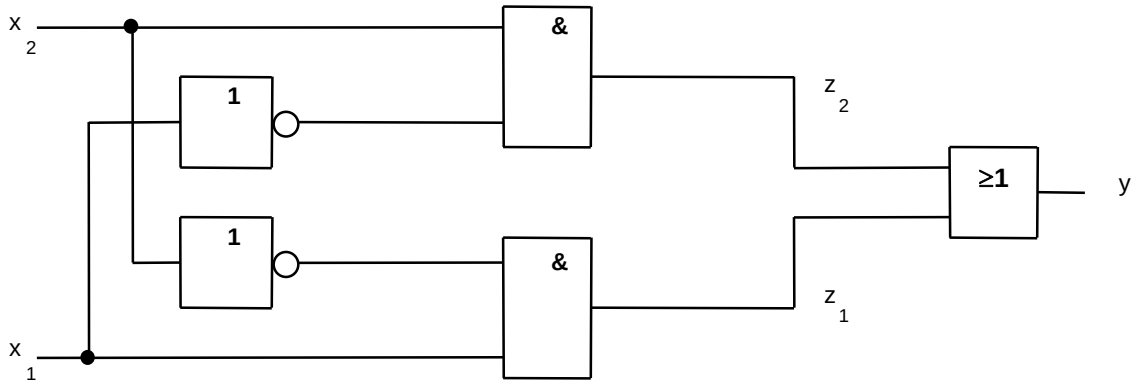
Digitaltechnik

Bausteine der Digitaltechnik
- Schaltnetze (Teil 1) -

Schaltnetze

- in der Regel bestehen **logische Ausdrücke** aus mehr als einem Operator
- aus den logischen Ausdrücken kann unmittelbar eine **Konstruktionsvorschrift** für **Digitalschaltungen** erstellt werden
  → Strukturvorschrift bzw. **Strukturausdruck**

Beispiel: $y = (x_2 \,\&\, \overline{x_1}) \vee (\overline{x_2} \,\&\, x_1) = [z_2 \vee z_1]$



- Als **Stufenzahl** bezeichnet man die **Tiefe** der **Schaltung**, der maximalen Anzahl von Schaltgliedern von den Eingängen zum Ausgang (*Inverter* werden dabei nicht gezählt)

Schaltnetze

- ***Schaltnetze*** sind Schaltungen, die **unmittelbar** einem ***Strukturausdruck*** entsprechen
- Dabei **hängen** die **Ausgänge** nur von den Signalen an den **Eingängen ab** (Schaltzeiten der Schaltglieder (= Gatter) und Signallaufzeiten werden zunächst vernachlässigt)

- **Definition (nach DIN IEC 748 Teil 2):**
  Ein **Schaltnetz** ist eine Digitalschaltung, in der es für **jede mögliche Kombination** von digitalen Signalen an den Eingängen eine – **und nur eine** – **Kombination** von digitalen Signalen an den Ausgängen gibt:

$$X \Rightarrow \boxed{F} \Rightarrow Y \qquad Y^t = F(X^t)$$

- Dabei ist der hochgestellte **Index t** lediglich ein Hinweis darauf, dass die **Größe Y zeitlich unmittelbar** aus der **Größe X** gebildet wird.

Entwurf von Schaltnetzen

Vorgehensweise:

1. **Schritt:** Umsetzung der verbalen **Aufgabenstellung** in eine **formale Form**, z.B. in eine **Funktionstabelle**;
   (dazu gehört die Festlegung der Wertezuordnung für die Binärvariablen)

2. **Schritt:** **Bildung** einer **Normalform**; Bildung einer vollständigen Blocküberdeckung

3. **Schritt:** **Bildung** einer **Minimalform** (bspw. nach S-Diagramm oder Nelson/Petrick)

4. **Schritt:** **Umformung** in das gewählte **Basissystem**

5. **Schritt:** **Umformen** in den **Strukturausdruck**

6. **Schritt:** **Umsetzen** in das entsprechende **Schaltnetz**
   (unter Umständen sind nicht alle Schritte zum Entwurf notwendig)

Schaltnetzbeispiel: Addiererbaustein

Einführung:

Um eine praktische Handhabung von arithmetischen Operationen zu erhalten, benötigen wir eine Reihe von Schaltungen, die uns die Grundoperationen wie ***Addition***, ***Subtraktion*** oder ***Multiplikation*** zur Verfügung stellen

Im Folgenden werden zu diesem Zweck eine Reihe von Schaltungen vorgestellt, die uns diese Funktionalität bieten:

Addierer sind Schaltungen, die uns ermöglichen
digital eine **Addition** durchzuführen

Hierzu existieren eine Reihe von **Addiererstrukturen**, die sich in ihren Eigenschaften wie **Laufzeiten** oder **Kosten** (Gatteranzahl) unterscheiden

Schaltnetzbeispiel: Addiererbaustein

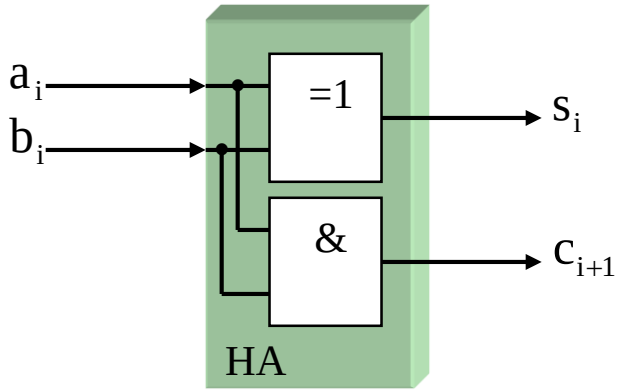
Halbaddierer:

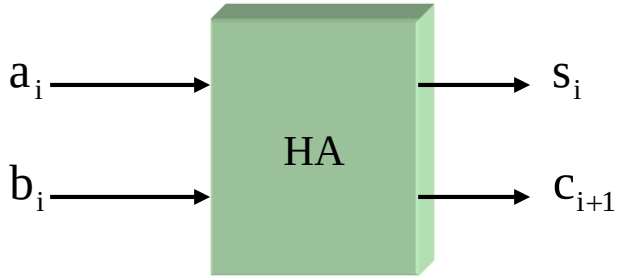
- summieren die beiden **Eingangsbits a_i und b_i** und legen die **Summe** auf den Ausgang **s_i**
- zusätzlich wird ein **Übertragsbit c_{i+1}** erzeugt

$$s_i = a_i \oplus b_i$$

$$c_{i+1} = a_i \cdot b_i$$

a_i	b_i	s_i	c_{i+1}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





Schaltnetzbeispiel: Addiererbaustein

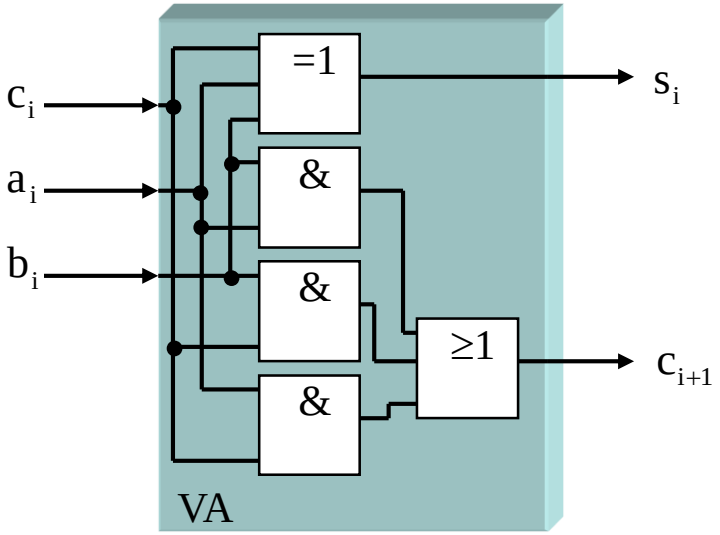
Volladdierer:

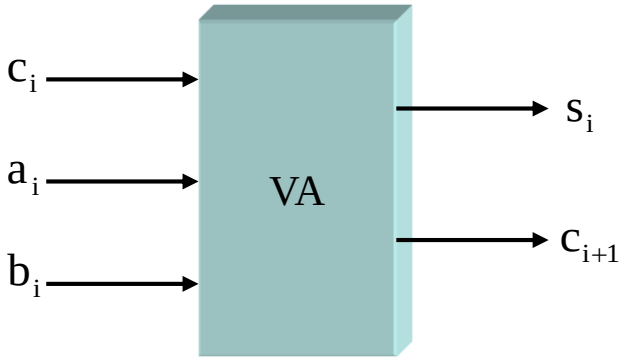
- besitzen zusätzlich einen **Übertragseingang** und sind somit in der Lage, **vorhergehende Stellen** in die Berechnung **einzubeziehen**

$$c_{i+1} = a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i$$

$$s_i = a_i \oplus b_i \oplus c_i$$

a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

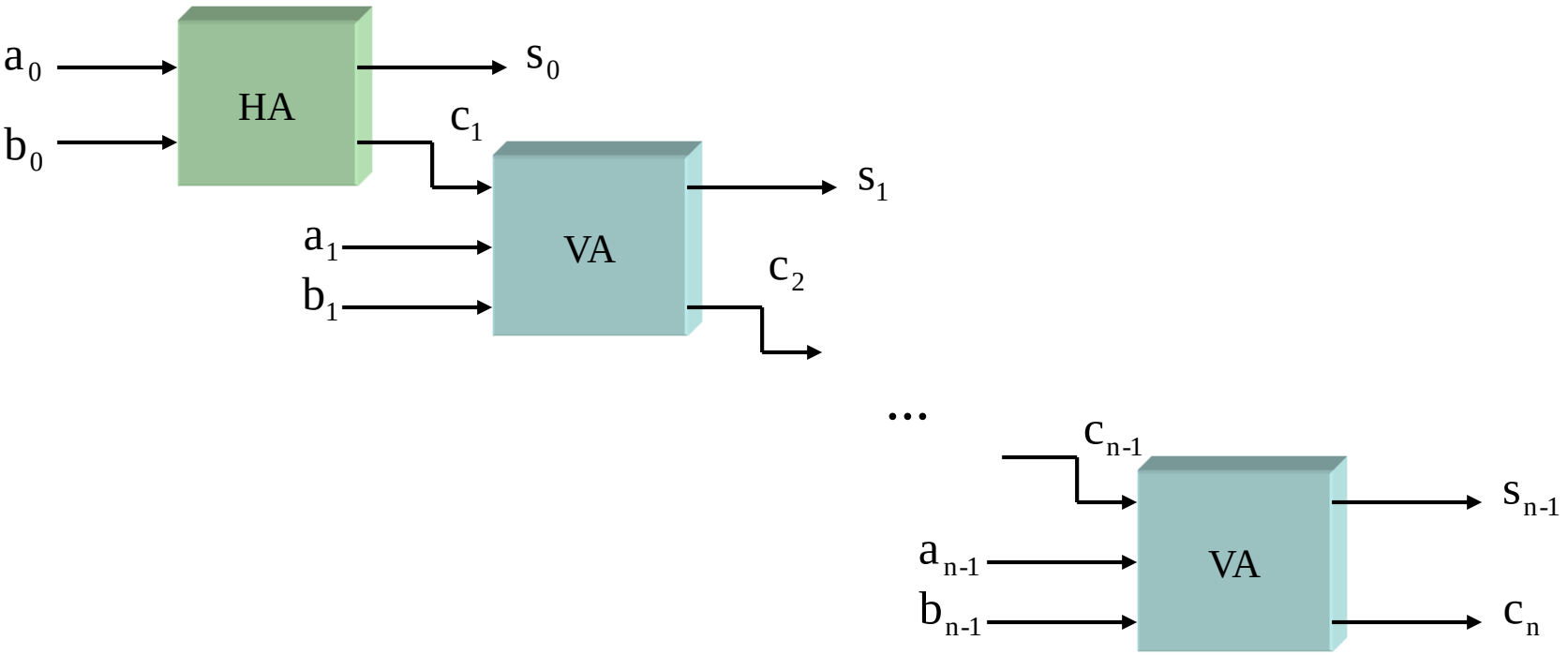




Schaltnetzbeispiel: Addiererbaustein

n-Bit Ripple-Carry-Addierer:

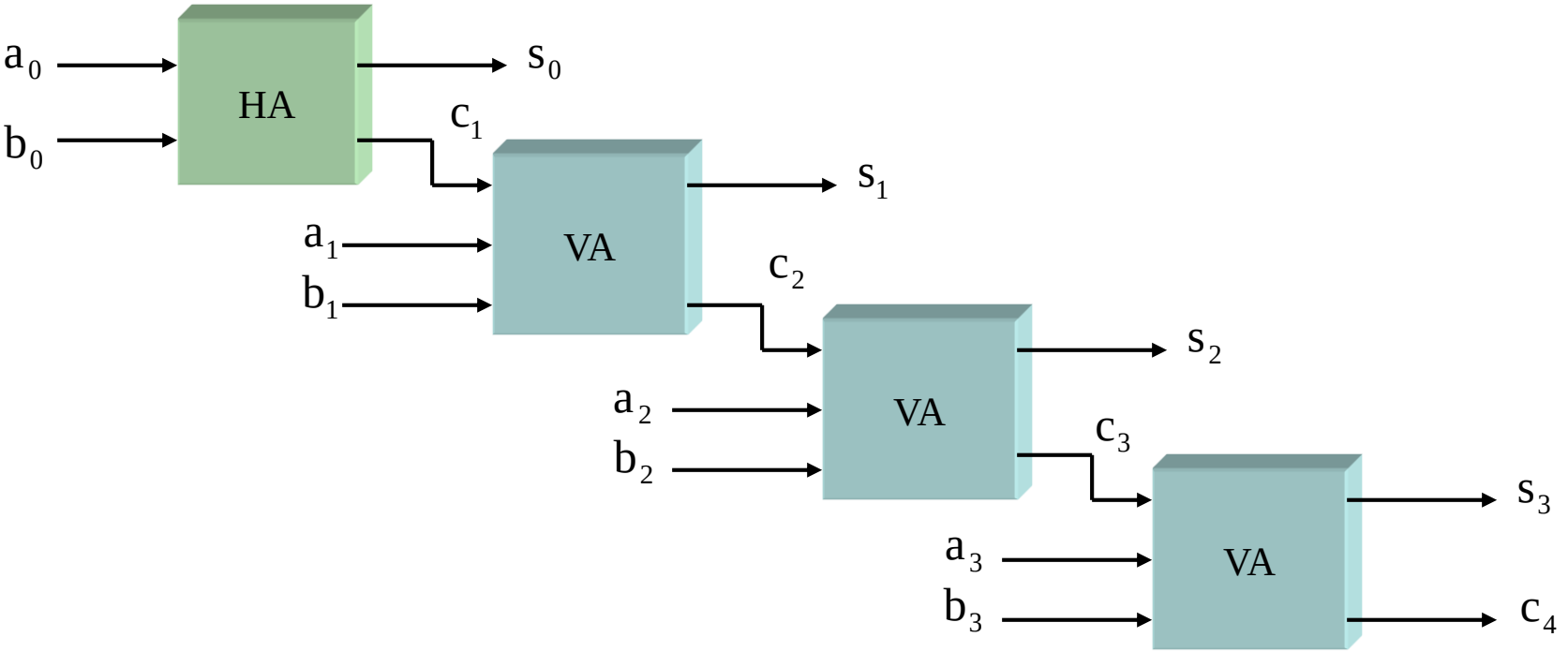
- **1. Baustein** ist ein **HA**, die darauffolgenden **VAs**



Schaltnetzbeispiel: Addiererbaustein

Beispiel: 4-Bit Ripple-Carry-Addierer

- Nachteil: ein sehr **langer kritischer Pfad**!!!
- die **Laufzeit** beträgt **2n Gatter** für die Berechnung von c_n



Schaltnetzbeispiel: Addiererbaustein

Carry-Look-Ahead-Addierer:

- die Berechnung der **Überträge** geschieht **parallel**
- wir führen **neue Signale** ein: **p_i (Propagate)** und **g_i (Generate)**

$$s_i = a_i \oplus b_i \oplus c_i = p_i \oplus c_i$$

$$\begin{aligned} c_{i+1} &= a_i \cdot b_i + a_i \cdot c_i + b_i \cdot c_i \\ &= a_i \cdot b_i + c_i (a_i + b_i) \\ &= a_i \cdot b_i + c_i (a_i \oplus b_i) \\ &= g_i + c_i p_i \end{aligned}$$

$$\text{mit} \quad p_i = a_i \oplus b_i \quad \text{und} \quad g_i = a_i \cdot b_i$$

Schaltnetzbeispiel: Addiererbaustein

Carry-Look-Ahead-Addierer:

- daraus ergeben sich folgende Gleichungen:

$$c_1 = g_0 + c_0 p_0$$

$$c_2 = g_1 + c_1 p_1 = g_1 + g_0 p_1 + c_0 p_0 p_1$$

$$c_3 = g_2 + c_2 p_2 = g_2 + g_1 p_2 + g_0 p_1 p_2 + c_0 p_0 p_1 p_2$$

...

$$c_n = g_{n-1} + c_{n-1} p_{n-1} = g_{n-1} + g_{n-2} p_{n-1} + g_{n-3} p_{n-2} p_{n-1} + \cdots + g_o p_1 \cdots p_{n-1} + c_0 p_0 \cdots p_{n-1}$$

damit lassen sich die **Übertragswerte** mit **3 „Stufen“** berechnen.

Schaltnetzbeispiel: Addiererbaustein

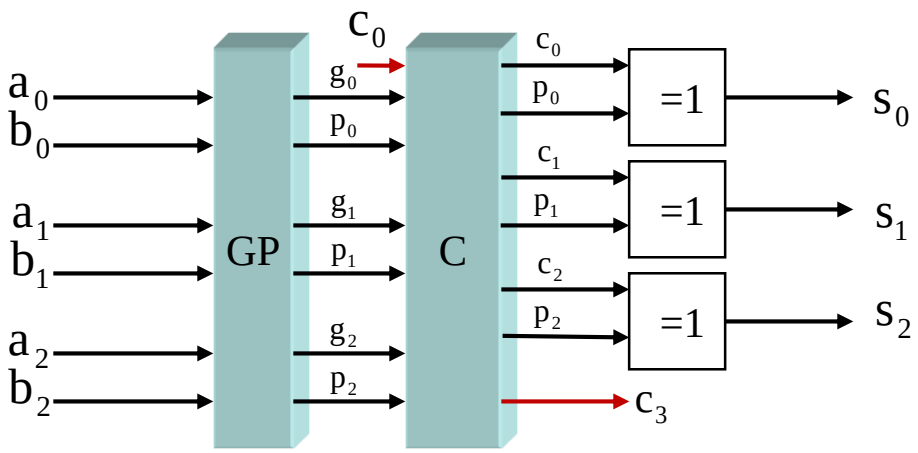
Beispiel: **3-Bit Carry-Look-Ahead-Addierer**

$$s_i = a_i \oplus b_i \oplus c_i = p_i \oplus c_i$$

$$c_1 = g_0 + c_0 p_0$$

$$c_2 = g_1 + c_1 p_1 = g_1 + g_0 p_1 + c_0 p_0 p_1$$

$$c_3 = g_2 + c_2 p_2 = g_2 + g_1 p_2 + g_0 p_1 p_2 + c_0 p_0 p_1 p_2$$



Schaltnetzbeispiel: Addiererbaustein

Carry-Look-Ahead-Addierer:

- **Problem:** beim **n-Bit Addierer** werden Gatter mit **n+1 Eingängen**, sowie **interne Signalverzweigungen** häufig benötigt
  → **Doppelberechnung** von **gleichen Termen**!

- **Abhilfe:** **Erweiterung des PG-Schemas auf Bitgruppen**, z.B. 4 Bit

$$c_4 = g_3 + g_2 p_3 + g_1 p_2 p_3 + g_0 p_1 p_2 p_3 + p_0 p_1 p_2 p_3 c_0$$

$$c_4 = G_{3:0} + P_{3:0} c_0$$

$$G_{3:0} = g_3 + g_2 p_3 + g_1 p_2 p_3 + g_0 p_1 p_2 p_3$$

$$P_{3:0} = p_0 p_1 p_2 p_3$$

Schaltnetzbeispiel: Addiererbaustein

Carry-Look-Ahead-Addierer:

- weiterhin gilt:

$$c_4 = G_{3:0} + P_{3:0} c_0$$

$$c_8 = G_{7:4} + P_{7:4} c_4$$

$$c_{12} = G_{11:8} + P_{11:8} c_8$$

$$c_{16} = G_{15:12} + P_{15:12} c_{12}$$

- durch **Substitution** erhält man:

$$c_4 = G_{3:0} + P_{3:0} c_0$$

$$c_8 = G_{7:4} + P_{7:4} G_{3:0} + P_{7:4} P_{3:0} c_0$$

$$c_{12} = G_{11:8} + P_{11:8} G_{7:4} + P_{11:8} P_{7:4} G_{3:0} + P_{11:8} P_{7:4} P_{3:0} c_0$$

$$c_{16} = G_{15:12} + P_{15:12} G_{11:8} + P_{15:12} P_{11:8} G_{7:4} + P_{15:12} P_{11:8} P_{7:4} G_{3:0} + P_{15:12} P_{11:8} P_{7:4} P_{3:0} c_0$$

Schaltnetzbeispiel: Subtrahiererbaustein

- Die Subtraktion lässt sich auf die Addition des **K2-Komplements** zurückführen

- Dies bedeutet die **bitweise Negation** und Addition einer 1

- Für einen **kombinierten Addierer/Subtrahierer** lässt sich die bitweise Negation über **XOR** steuern

Schaltnetzbeispiel: Subtrahiererbaustein

Beispiel: 4-Bit Addierer/Subtrahierer

- Addierer mit **k=0**, Subtrahierer mit **k=1**