Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

GTI – ÜBUNG 12

KOMPARATOR UND ADDIERER

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Entwickeln Sie eine digitale Schaltung, die zwei Bits a und b miteinander vergleicht. Die Schaltung besitzt drei Ausgänge: ’<’ ist logisch ’1’, genau dann, wenn a<b. Entsprechend ist ’>’ genau dann ’1’, wenn a > b. Der Ausgang ’=’ soll genau dann eine ’1’ anzeigen, wenn a und b gleich sind.
- Achten Sie darauf, dass stets genau einer der drei Ausgänge aktiv ist.
- Hinweis: Beginnen Sie mit einer Funktionstabelle.

Aufgabe 1 – Komparator

- Funktionstabelle

Verlässliche Grundlage der digitalen Schaltungsbildung ist das Aufstellen einer Funktionstabelle, welche die abstrakte Beschreibung in funktional, logischer Form vereinfacht.

a	b	a < b	a = b	a > b
0	0	0	1	0
1	0	0	0	1
0	1	1	0	0
1	1	0	1	0

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Schaltfunktionen

a	b	a < b	a = b	a > b
0	0	0	1	0
1	0	0	0	1
0	1	1	0	0
1	1	0	1	0

$$f_< = \bar{a}b$$

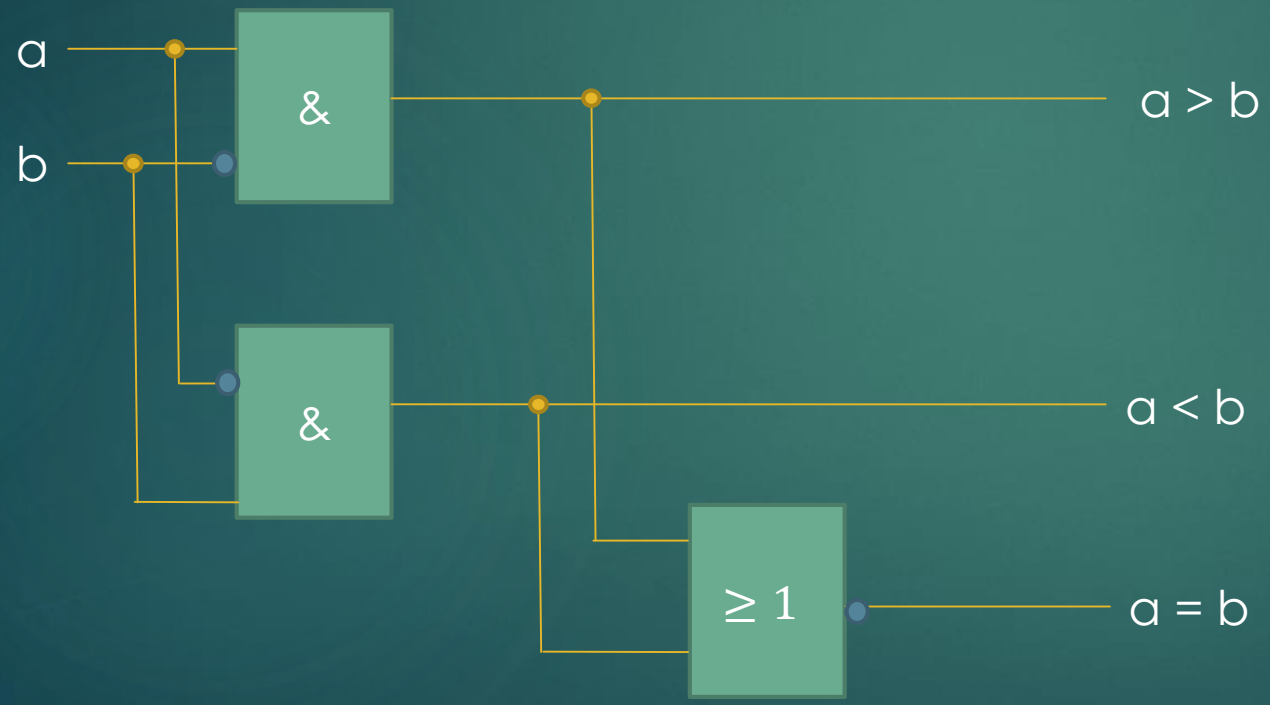
$$f_= = \bar{a}\bar{b} + ab = \overline{a \otimes b} = a\ xnor\ b$$

$$f_> = a\bar{b}$$

Aufgabe 1 – Komparator

- Gatterumsetzung

$$f_< = \bar{a}b \qquad f_= = \bar{a}\bar{b} + ab = \overline{a \otimes b} = a\ xnor\ b \qquad f_> = a\bar{b}$$



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Entwerfen Sie nun eine digitale Schaltung, deren Eingänge die Ausgänge $K_0 = (<_0, >_0, =_0)$ und $K_1 = (<_1, >_1, =_1)$ zweier Komparatoren sind und die diese *lexikographisch* auf die Ausgabe (<, >, =) reduziert. Dabei soll K_0 nieder- und K_1 höherwertig sein.
- Hinweis: wie würde man das schriftlich rechnen?

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

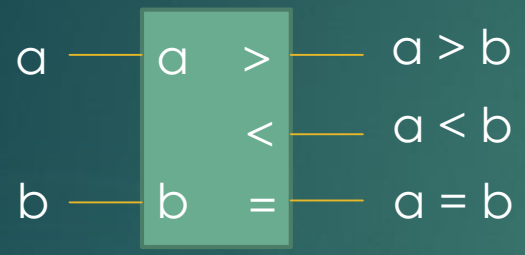
Blackbox:

- Um wiederkehrende Bauelement nicht jedes Mal auf Grund auf neu implementieren zu müssen, fügt man eine abstrahierte Blackbox des Bauelementes in die Schaltung ein
- Diese definiert sich nur noch über ihre Ein- und Ausgänge. Das Verhalten ist aus der Schaltung nicht mehr direkt ersichtlich.
- Beispiele dieser Abstrahierung:
  - Flipflops
  - (De-) Multiplexer
  - Komparator
  - Schieberegister

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Blackbox des Komparators:



$$> (K_1, K_0) = (>_1) + (=_1) \cdot (>_0)$$
$$< (K_1, K_0) = (<_1) + (=_1) \cdot (<_0)$$
$$= (K_1, K_0) = (=_1) \cdot (=_0)$$

Verhalten bei einer Zweibitzahl:

A = a_1a_0 und B = b_1b_0

Der Wert der einzelnen Bits sinkt von links nach rechts, wobei jedes Bit vom Betrag her größer ist als alle seine rechten Nachbarn zusammen. Ist ein Bit von A kleiner als von B, so ist A auch insgesamt kleiner als B, wenn die Bits links von diesen identisch bzw. auch kleiner sind.

Aufgabe 1 – Komparator

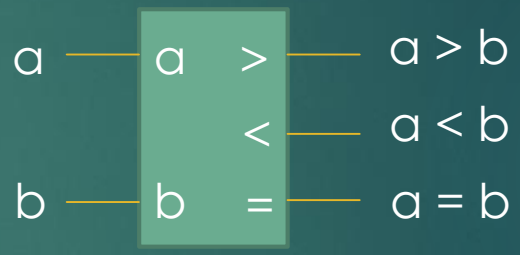
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

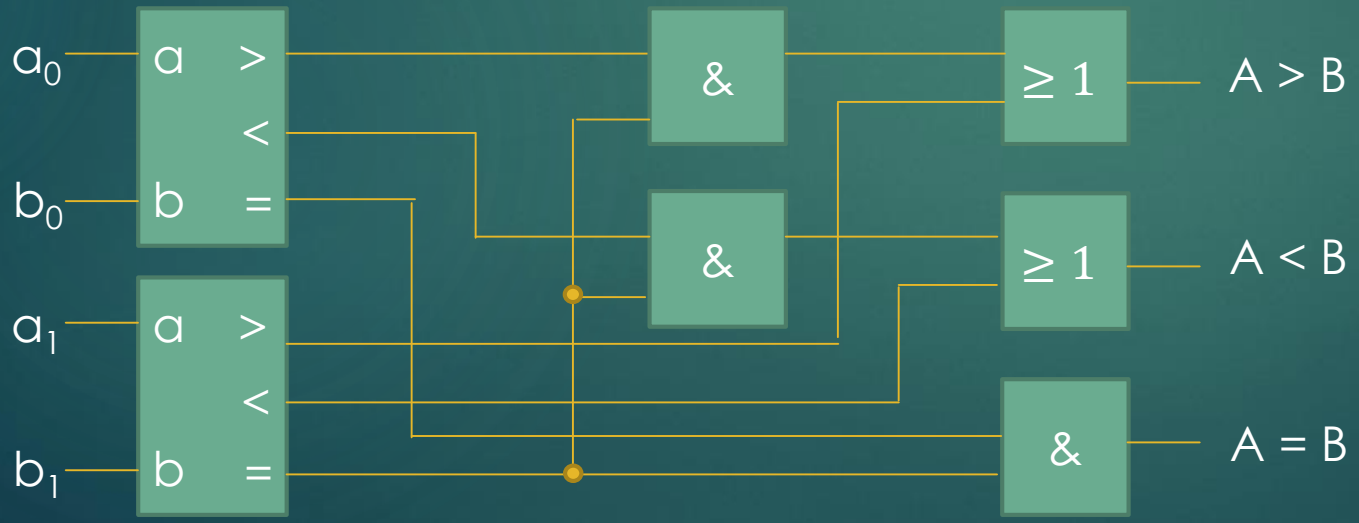
Verhalten bei einer Zweibitzahl:

$A = a_1a_0$ und $B = b_1b_0$

Also (beginnend von links):

a > b:	$(a_1 > b_1)$	oder	$(a_1 = b_1)$ und $(a_0 > b_0)$
a = b:	$(a_1 = b_1)$	und	$(a_0 = b_0)$
a < b:	$(a_1 < b_1)$	oder	$(a_1 = b_1)$ und $(a_0 < b_0)$

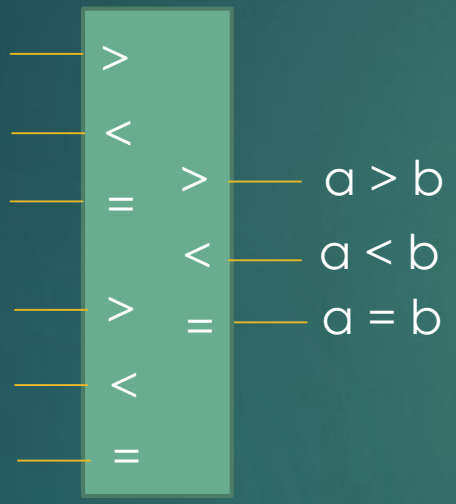




Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Blackbox des 2-Bit-Komparators:



Verhalten bei einer Vierbitzahl:

$A = a_3 \ldots a_0$ und $B = b_3 \ldots b_0$

Wir benutzen zwei 2-Bit-Komparatoren (a_{32}, b_{32}) und (a_{10}, b_{10}), da auch hier von links nach rechts verglichen werden muss. Somit gleicht die Auswertung der des 1-Bit-Komparators.

Aufgabe 1 – Komparator

Beschreibung

- Die Schaltung aus a) und b) wird nun als ’Black-Box’ mit den Eingängen a und b und den Ausgängen ’<’, ’>’ und ’=’ betrachtet.
- Entwerfen Sie ausschließlich mit diesen Komponenten einen Komparator für vorzeichenlose 4-Bit-Binärzahlen. Welche Möglichkeiten gibt es, die Komponenten zusammenzuschalten, und welche Auswirkungen hat dies auf die benötigte Fläche und den kritischen Pfad?
- Hinweis: wie würde man das schriftlich rechnen?

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 1 – Komparator

Verhalten bei einer Vierbitzahl:

$A = a_3 \ldots a_0$ und $B = b_3 \ldots b_0$

a > b: $(a_{32} > b_{32})$ oder

$(a_{32} = b_{32})\ (a_{10} > b_{10})$

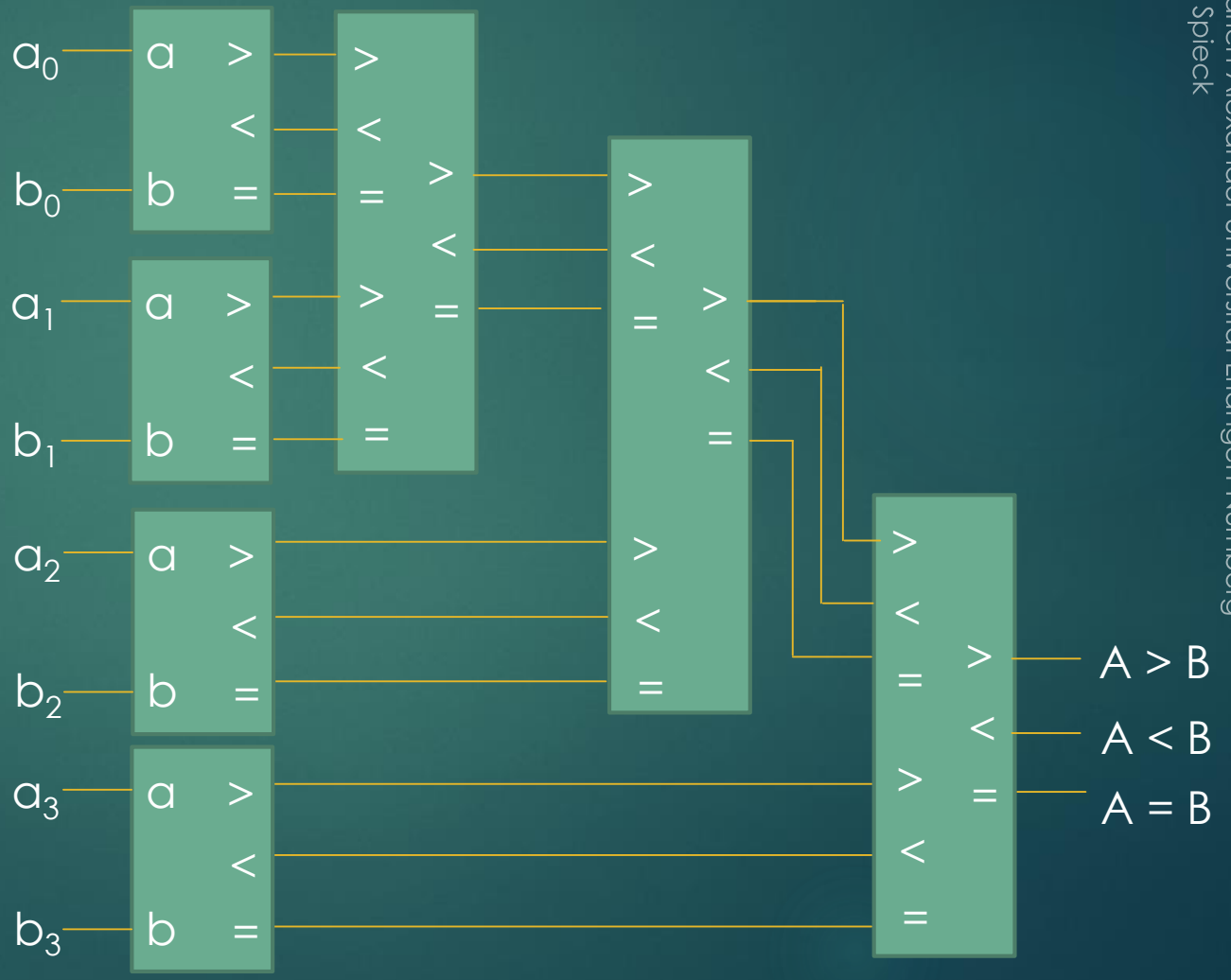
a = b: $(a_{32} = b_{32})$ und

$(a_{10} = b_{10})$

a < b: $(a_{32} < b_{32})$ oder

$(a_{32} = b_{32})\ (a_{10} < b_{10})$

Lineare Lösung



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Verhalten bei einer Vierbitzahl:

$A = a_3 \ldots a_0$ und $B = b_3 \ldots b_0$

a > b: $(a_{32} > b_{32})$ oder

$(a_{32} = b_{32})\ (a_{10} > b_{10})$

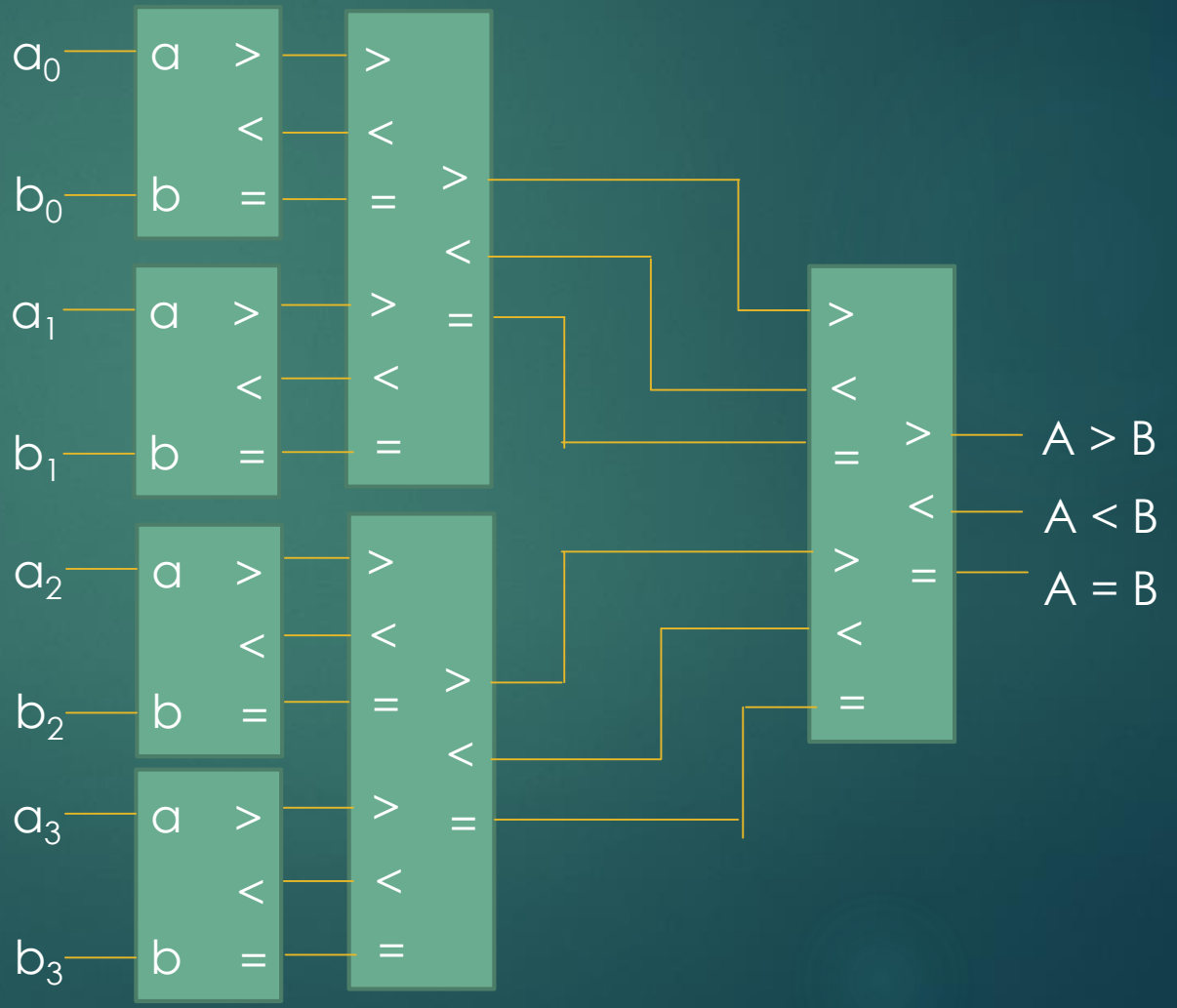
a = b: $(a_{32} = b_{32})$ und

$(a_{10} = b_{10})$

a < b: $(a_{32} < b_{32})$ oder

$(a_{32} = b_{32})\ (a_{10} < b_{10})$

Baumartige Lösung



Aufgabe 1 – Komparator

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Welche Möglichkeiten gibt es, die Komponenten zusammenzuschalten, und welche Auswirkungen hat dies auf die benötigte Fläche und den kritischen Pfad?

Lineares Zusammenschalten:

- Allgemein n-1 Stufen
- Kritischer Pfad steigt linear O(n) an

Baumartiges Zusammenschalten:

- Allgemein $\lceil \log_2 n \rceil$ Stufen
- Kritischer Pfad steigt logarithmisch O(log n) an (VORTEIL!)

Die Fläche der beiden Lösungen ist hingegen identisch!

Aufgabe 2 – Addierer/Subtrahierer

Beschreibung

- Realisieren Sie einen Halbaddierer mit Hilfe von NAND-Gattern.
- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.
- Bestimmen Sie jeweils die Anzahl der verwendeten Gatter und die Anzahl der Gatter des kritischen Pfades.

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Halbaddierer mit Hilfe von NAND-Gattern.

Halbaddierer: a + b = s mit c

- S: Summe der beiden Operanden
- C: Carry, d.h. der Übertrag

a		b		S	c
0	+	0	=	0	0
0	+	1	=	1	0
1	+	0	=	1	0
1	+	1	=	0	1

$s = a \otimes b$

$c = ab$

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Nandisierung der Funktionen

$s = a \otimes b = a\bar{b} + \bar{a}b$ | Doppelnegation

$= \overline{\overline{a\bar{b}1} + \overline{\bar{a}1b}}$ | De Morgan

$= \overline{\overline{a\bar{b}1}\ \overline{\bar{a}1b}}$

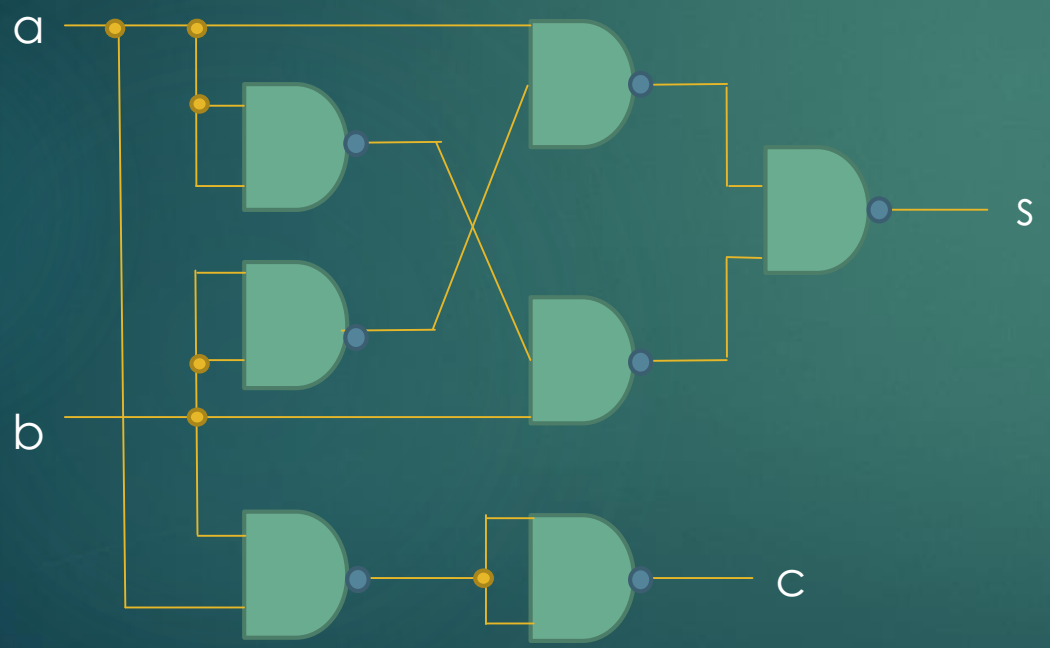
$c = ab = \overline{\overline{ab1}}$

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Gatterumsetzung

$$s = \overline{\overline{\overline{ab1}\ \overline{a1b}}} = \overline{\overline{abb}\ \overline{aab}} \qquad c = \overline{\overline{ab1}} = \overline{\overline{ab}\ \overline{ab}}$$



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

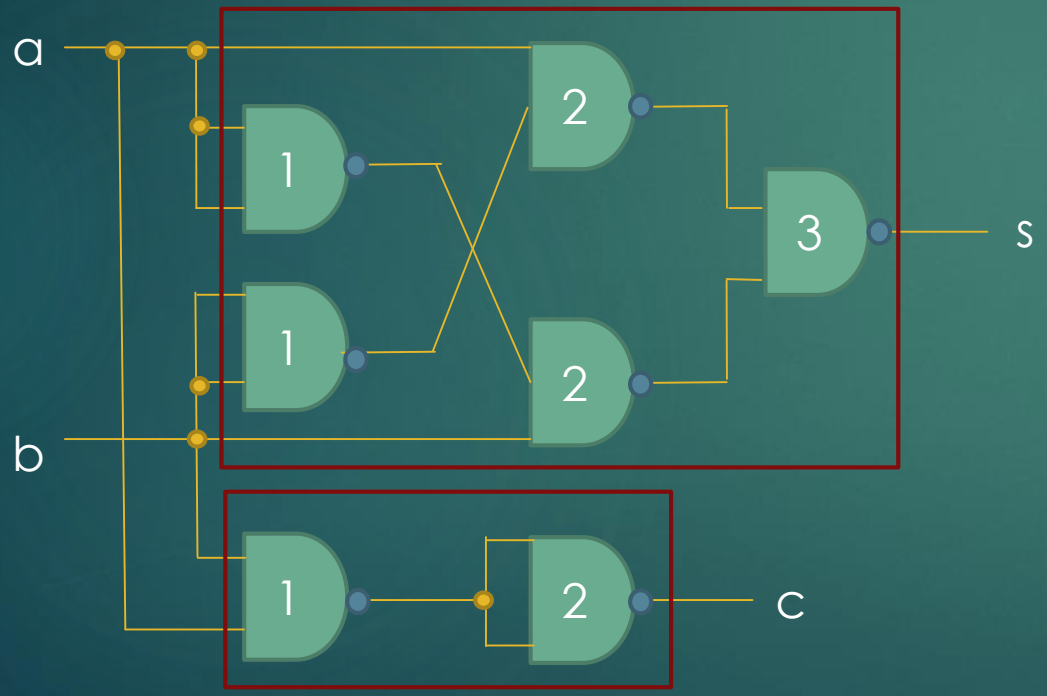
Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Gatteranzahl, Kritischer Pfad

Momentan: 7 Gatter

kritischer Pfad: 3 (Summe), 2 (Carry)



Kritischer Pfad:

Größtmöglicher Anzahl Gatter auf einem Pfad bis zu dem bestimmten Ausgang.

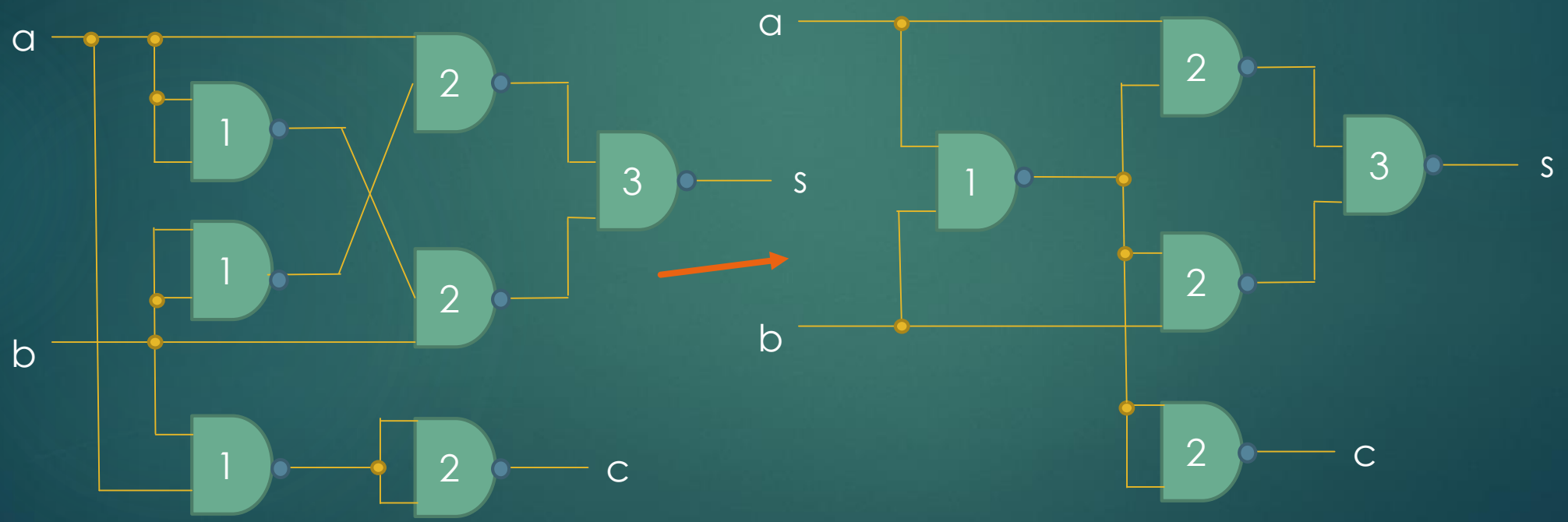
So beträgt der kritischer Pfad der Summe 3, da genau drei Gatter durchlaufen werden müssen, bis man zum Ausgang gelangt.

Der kritische Pfad ist ein Maß der Zeit, welche die Schaltung zur Auswertung der Funktion benötigt.

Aufgabe 2 – Addierer/Subtrahierer

- Minimierung

Wir haben Summe und Carry getrennt verschaltet. Wenn wir die beiden Schaltungen verschmelzen, können wir uns zwei Gatter sparen.



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.

Wir nutzen dazu die Implementierung des Halbaddierers.

Dessen Blackbox, die wir hier leider nicht benutzen dürfen, sieht folgendermaßen aus:

Ein Volladdierer berücksichtigt im Gegensatz dazu ein Carrybit in der Eingabe.
Dessen Schaltbild sieht dann folgendermaßen aus:

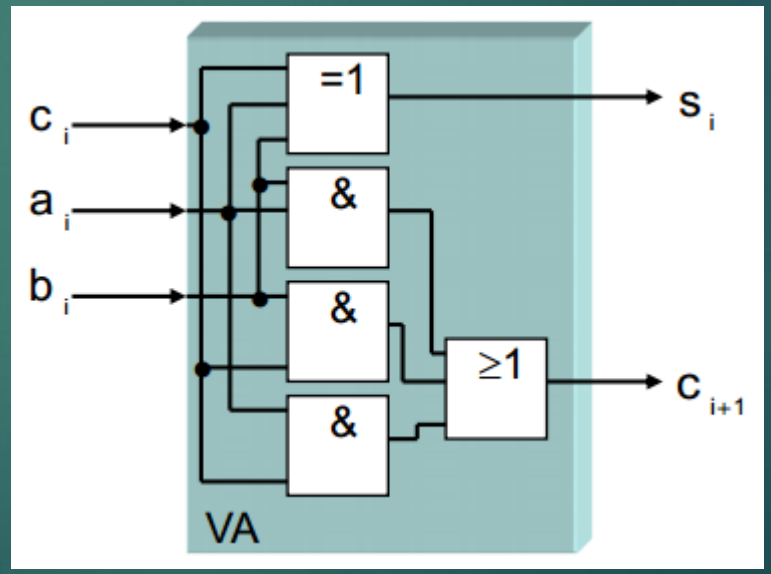
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.

Überlegen wir uns das gewünschte Verhalten (ohne NANDs):

a		b		c_{in}		S	c_{out}
0	+	0	+	0	=	0	0
0	+	1	+	0	=	1	0
1	+	0	+	0	=	1	0
1	+	1	+	0	=	0	1
0	+	0	+	1	=	1	0
0	+	1	+	1	=	0	1
1	+	0	+	1	=	0	1
1	+	1	+	1	=	1	1

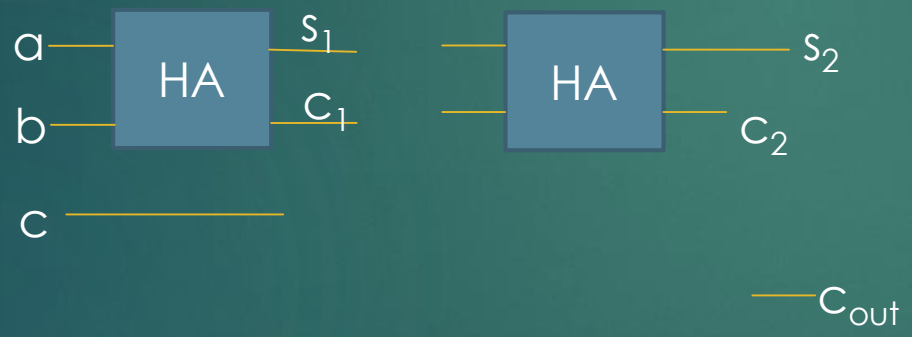


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.

Wir bilden das Problem des Volladdierers auf zwei Halbaddierer ab. Nun müssen wir uns überlegen, wie wir die Halbaddierer verkabeln müssen.



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.

Summenbildung: $((a+b) + c_{in})$ = Summe, d.h. $s_1 + c_{in} = s_2$ (mit c_2)

S_1	c_1	c_{in}		S_2	c_2	c_{out}
0	0	0	=	0	0	0
1	0	0	=	1	0	0
0	1	0	=	0	0	1
0	0	1	=	1	0	0
1	0	1	=	0	1	1
0	1	1	=	1	0	1

H1 H2

Das Carry-Bit ist 1, wenn:

a, im ersten Halbaddierer 1 ist oder

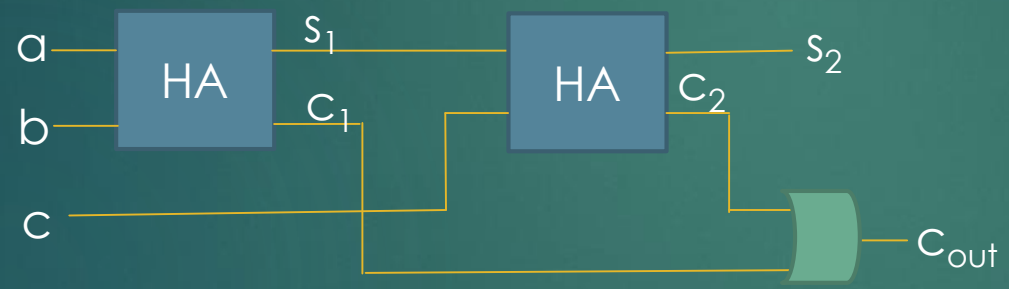
b, im zweiten Halbaddierer 1 ist

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Realisieren Sie einen Volladdierer ebenfalls mit Hilfe von NAND-Gattern.

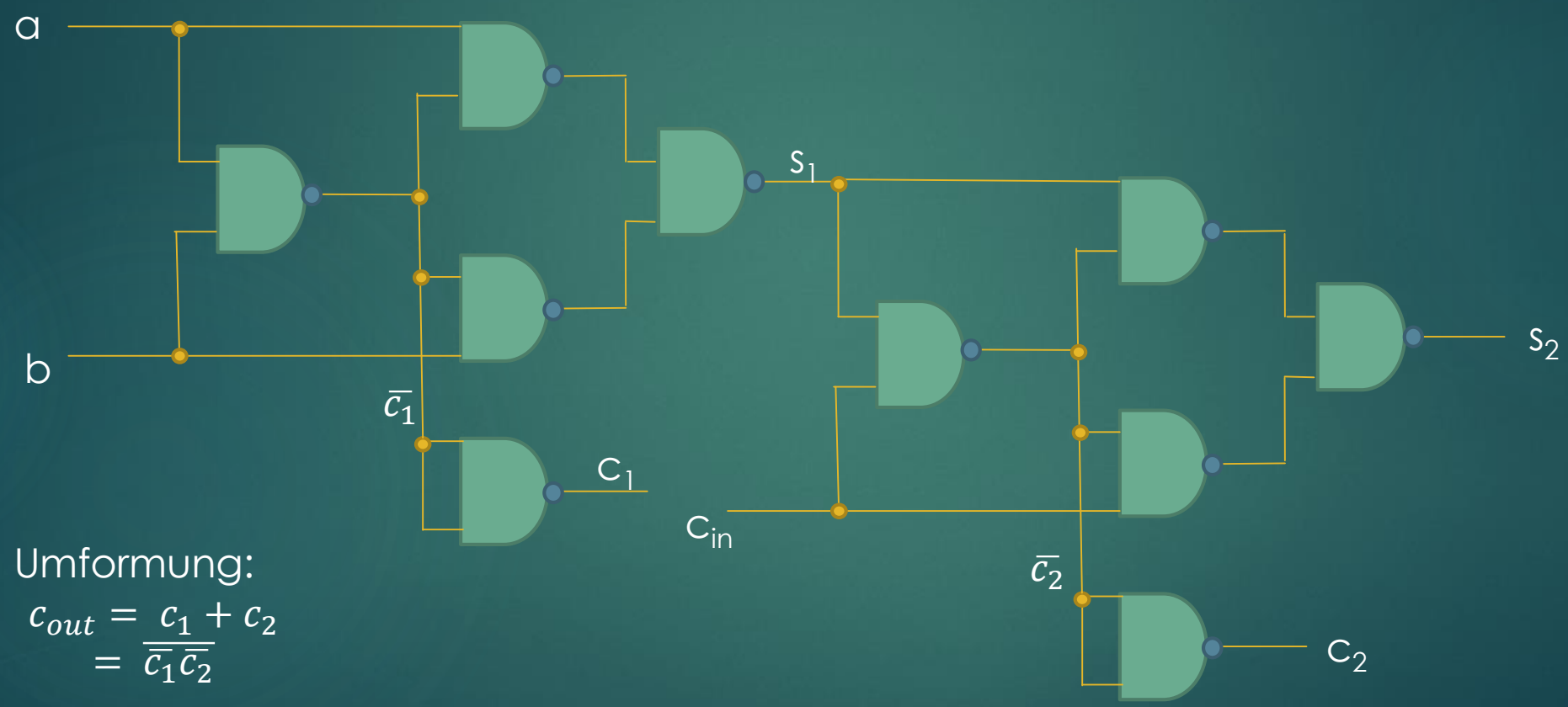
$((a+b) + c_{in}) = \text{Summe}$ $\quad c_1 + c_2 = c_{out}$



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- NAND-Realisierung



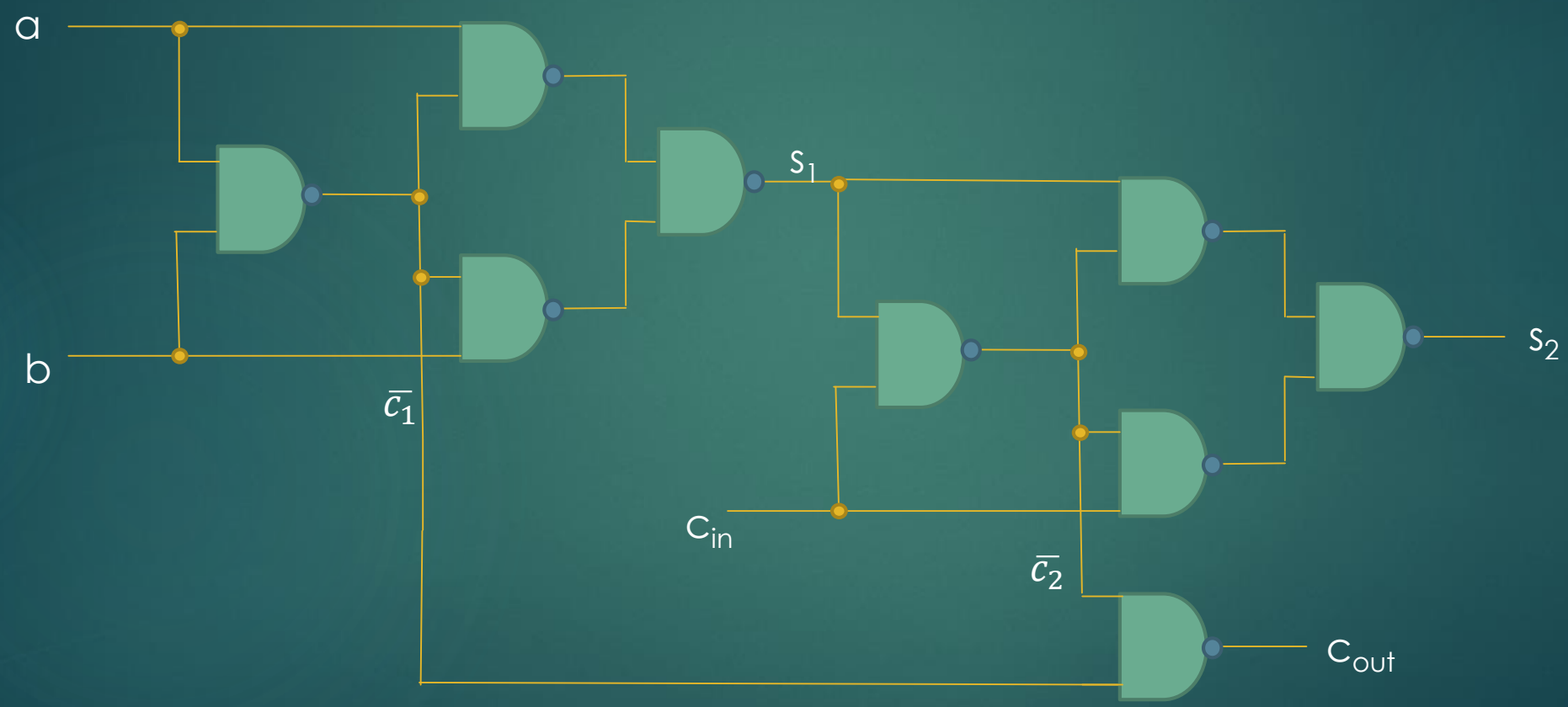
Umformung:

$$c_{out} = c_1 + c_2 = \overline{\overline{c_1}\,\overline{c_2}}$$

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

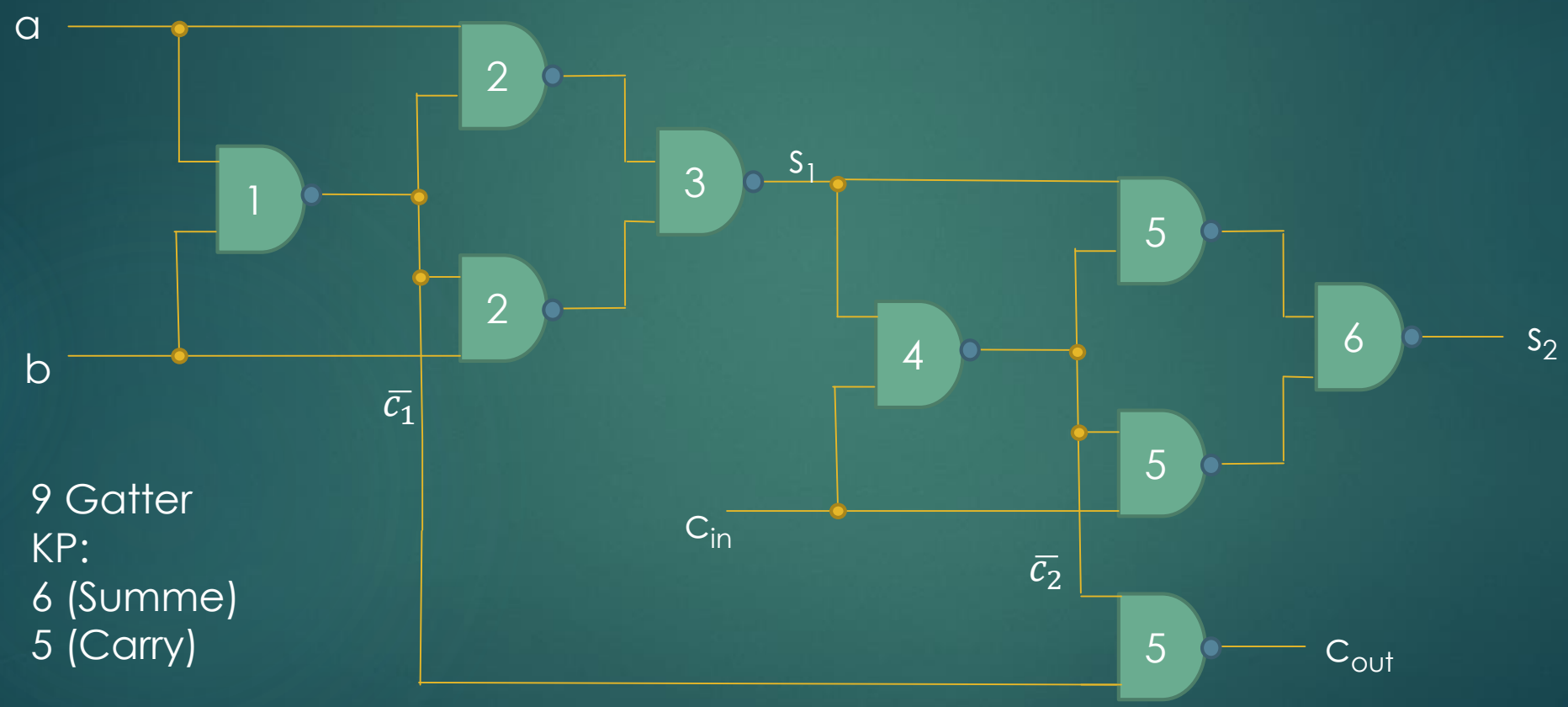
- NAND-Realisierung



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- NAND-Realisierung



9 Gatter
KP:
6 (Summe)
5 (Carry)

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Denkpause

Zwei Züge rasen mit 50 km/h auf einem Gleis direkt aufeinander zu. Sie sind zum Zeitpunkt 0 genau 100 km auseinander. Eine Biene sitzt auf einem der beiden Züge und fliegt zum Zeitpunkt 0 auf direkten geraden Weg los, um den anderen Zug zu erreichen. Da die Biene dem „The grass is always greener on the other side“-Syndrom unterliegt, fliegt sie beim Erreichen des anderen Zugs direkt wieder los, um auf den ursprünglichen Zug zu landen. Das wiederholt sie danach ewig so weiter, fliegt also immer von einem Zug zum anderen.

Wenn die Biene 90 km/h fliegt, wie viele Kilometer legt sie dann insgesamt zurück?

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Denkpause

Es dauert genau 1 Stunde bis beide Züge aufeinandertreffen (50 km/h *2).

Die Biene fliegt konstant eine Stunde mit 90 km/h, also insgesamt 90 km!

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

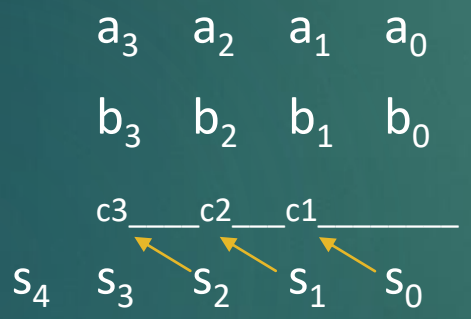
- Erstellen Sie aus den Volladdiererzellen aus a) einen Carry-Ripple-Addierer für 4-Bit breite Operanden ($a_3 \ldots a_0 + b_3 \ldots b_0 + c_{in} = c_{out}\ s_3 \ldots s_0$).
- Wie viele Gatter enthält nun der kritische Pfad des gesamten Schaltnetzes?

Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Carry-Ripple-Addierer

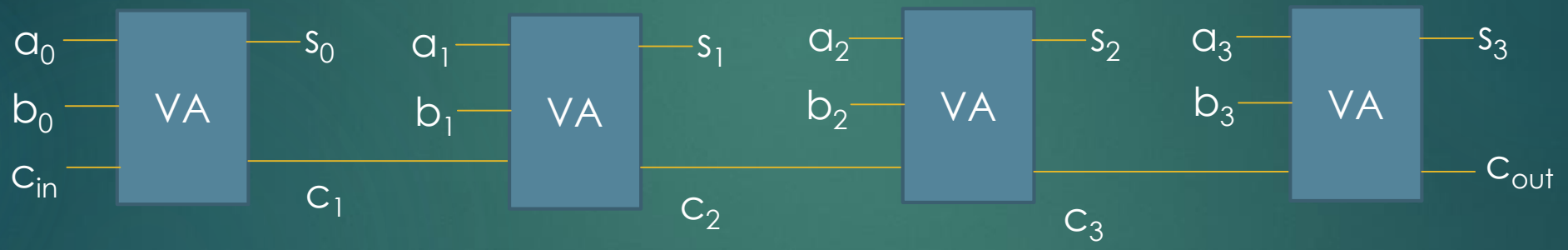
- Carry: Übertrag, Ripple: Rieseln
- Mehrere kaskadierte Volladdierer, bei denen der Übertrag von einem Volladdierer zum nächsten weitergereicht wird
- Umsetzen des Schulprinzips zum Addieren zweier Zahlen A und B:



- Bei S_4 handelt es sich um den Carry-Out
- Gerechnet wird also von rechts nach links

Aufgabe 2 – Addierer/Subtrahierer

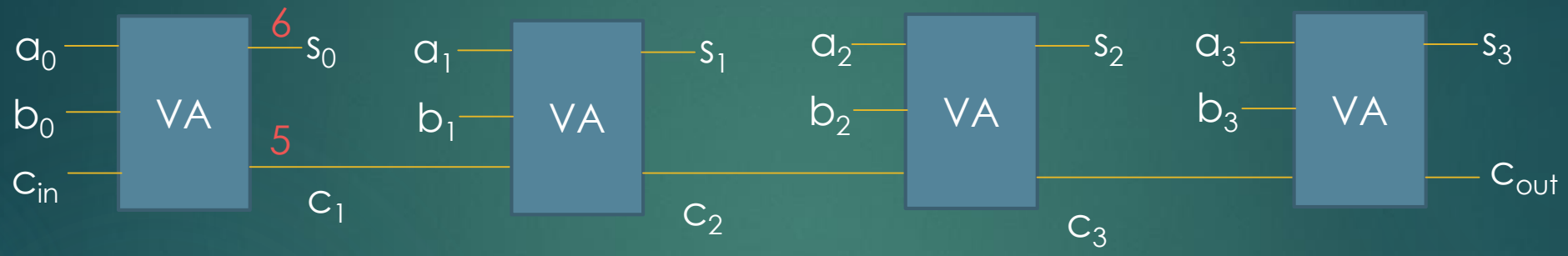
- Erstellen Sie aus den Volladdiererzellen aus a) einen Carry-Ripple-Addierer für 4-Bit breite Operanden ($a_3 \ldots a_0 + b_3 \ldots b_0 + c_{in} = c_{out}\ s_3 \ldots s_0$).



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Wie viele Gatter enthält nun der kritische Pfad des gesamten Schaltnetzes?



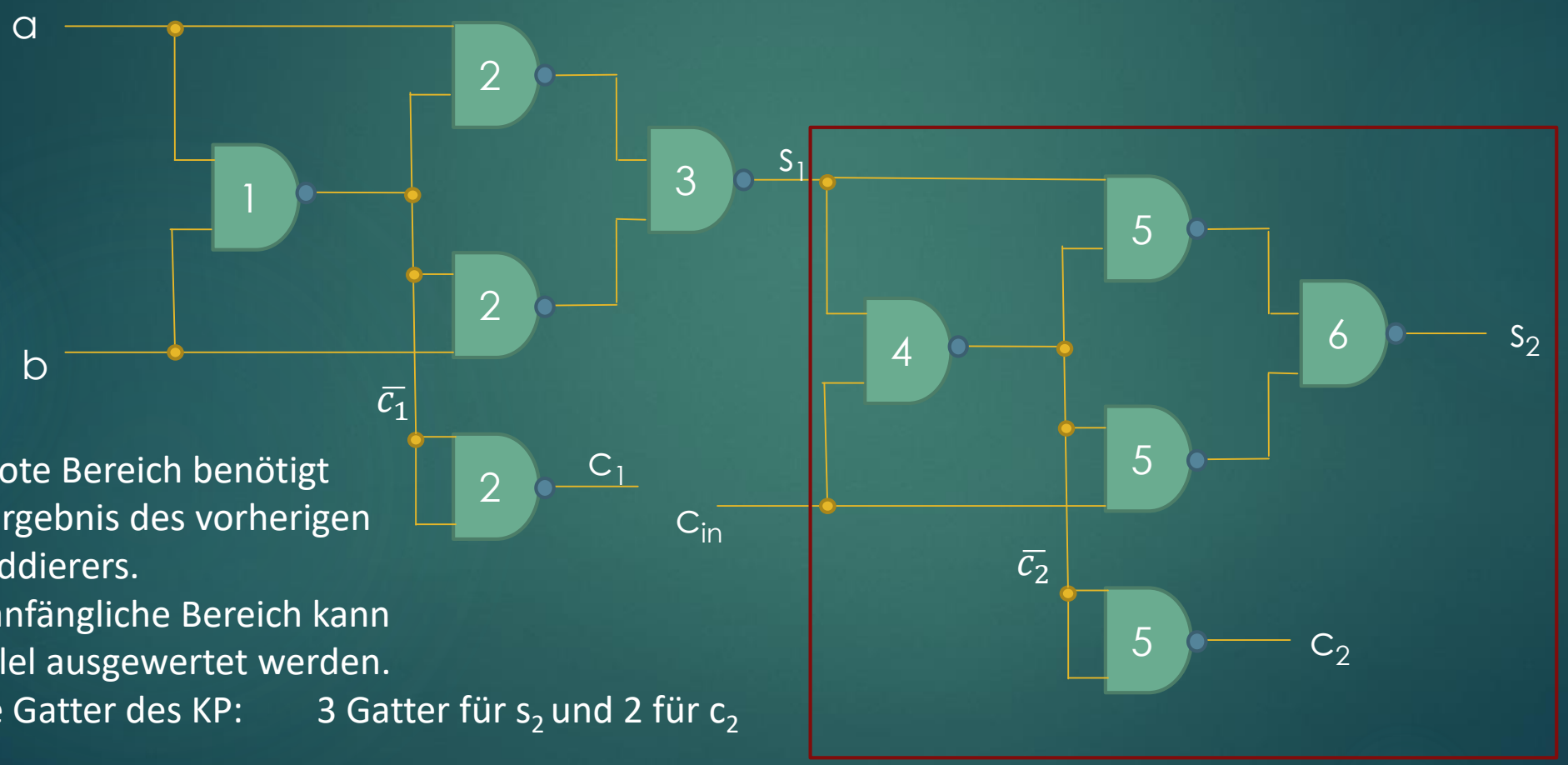
Zur Erinnerung der kritische Pfad eines VA: 6 (Summe) 5 (Carry)

1: s_0 ist nach 6 Schritten ausgerechnet, c_1 somit nach 5

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

Übersicht über den Volladdierer:

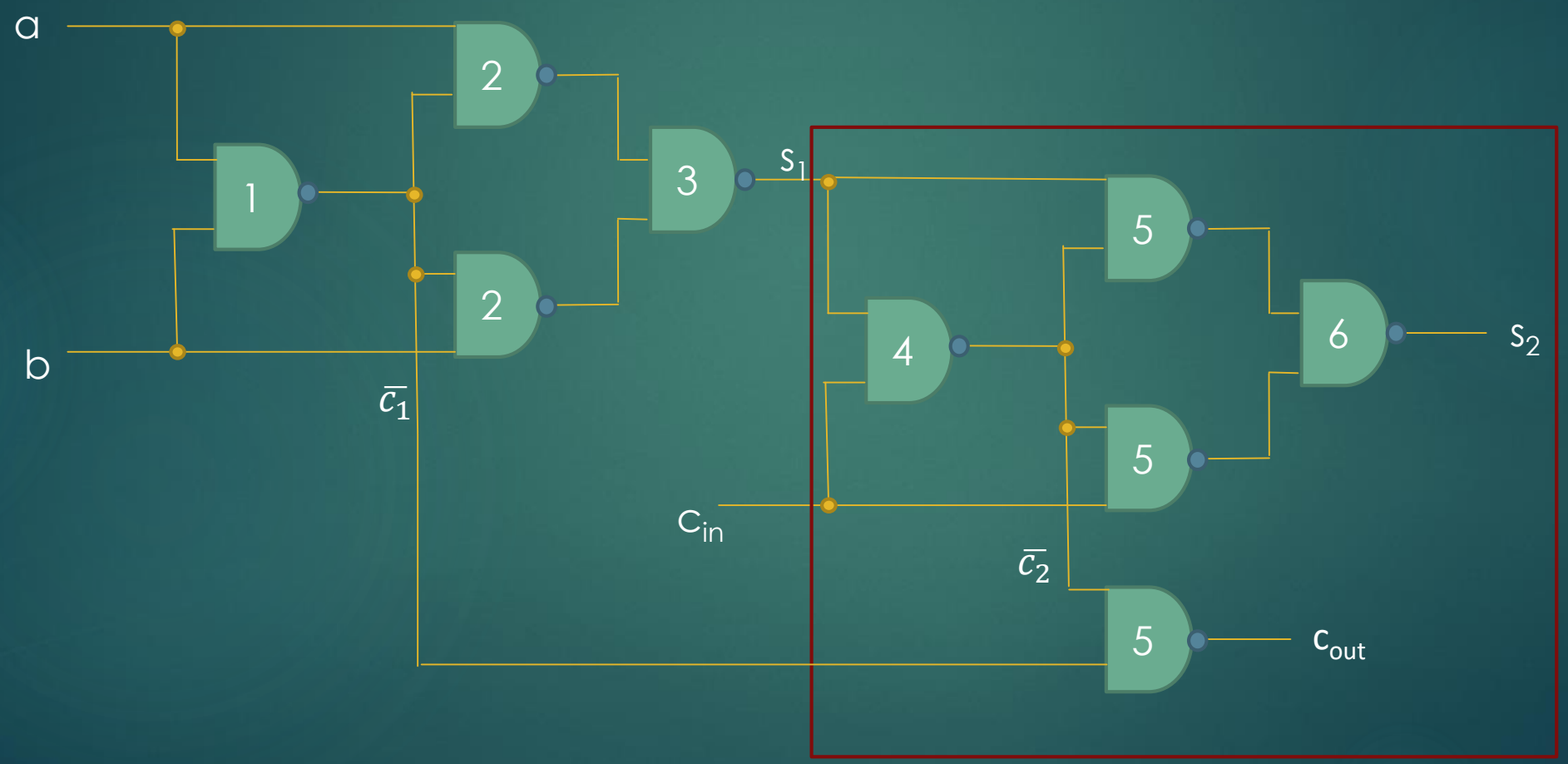


Der rote Bereich benötigt das Ergebnis des vorherigen Volladdierers.
Der anfängliche Bereich kann parallel ausgewertet werden.
Neue Gatter des KP: 3 Gatter für s_2 und 2 für c_2

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

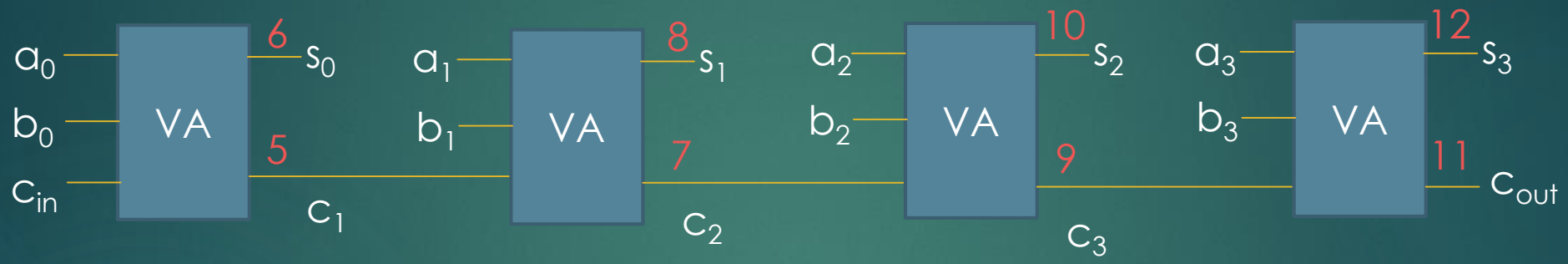
Berücksichtigung c_{out}



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Wie viele Gatter enthält nun der kritische Pfad des gesamten Schaltnetzes?



Zur Erinnerung der kritische Pfad eines VA: 6 (Summe) 5 (Carry)

1: s_0 ist nach 6 Schritten ausgerechnet, c_1 somit nach 5

2: $5 + 3 = 8$ für s_1 $5 + 2 = 7$ für c_2

3: $7 + 3 = 10$ für s_2 $7 + 2 = 9$ für c_3

4: $9 + 3 = 12$ für s_3 $9 + 2 = 11$ für c_4

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Beschreibung

- Der Addierer aus b) soll nun als Block betrachtet werden und so erweitert werden, dass er durch ein zusätzliches Steuerbit Sub auch subtrahieren kann (durch Umwandlung von B in das 1er-Komplement). Dazu stehen Ihnen beliebige Gatter zur Verfügung. Beachten Sie:

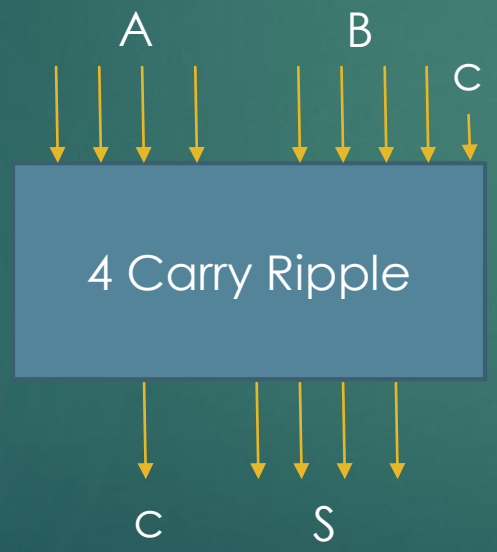
- Erzeugung negativer Zahlen (Komplementierer)
- Behandlung des Übertrags
- Erkennung eines Überlaufs (Overflow)

Aufgabe 2 – Addierer/Subtrahierer

Aufgaben:

- zusätzliches Steuerbit Sub
- Subtraktion durch Umwandlung von B in das 1er-Komplement

Blockschaltbild:



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Wandeln in das Einerkomplement:

Unser Steuerbit Sub entscheidet, ob B in das Einerkomplement gewandelt wird.

Erinnerung: A – B = A + (-B)

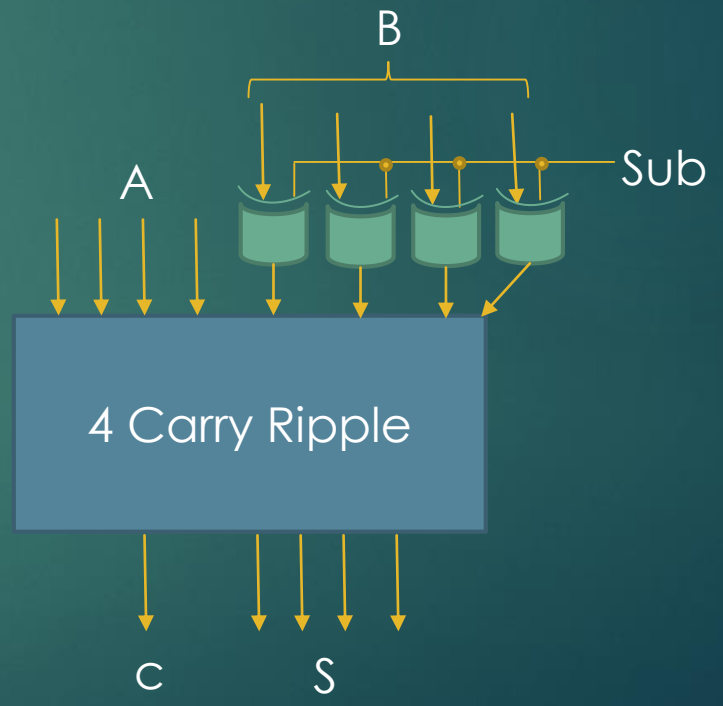
Wir legen fest:

Sub-Bit 1: Subtraktion

Sub-Bit 0: Addition

Sub	B	B_{neu}
0	0	0
0	1	1
1	0	1
1	1	0

Diese Schalttabelle entspricht einem XOR!



Aufgabe 2 – Addierer/Subtrahierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Behandlung des Übertrags:

Einerkomplement: Null wird sowohl durch 0000, als auch 1111 dargestellt.

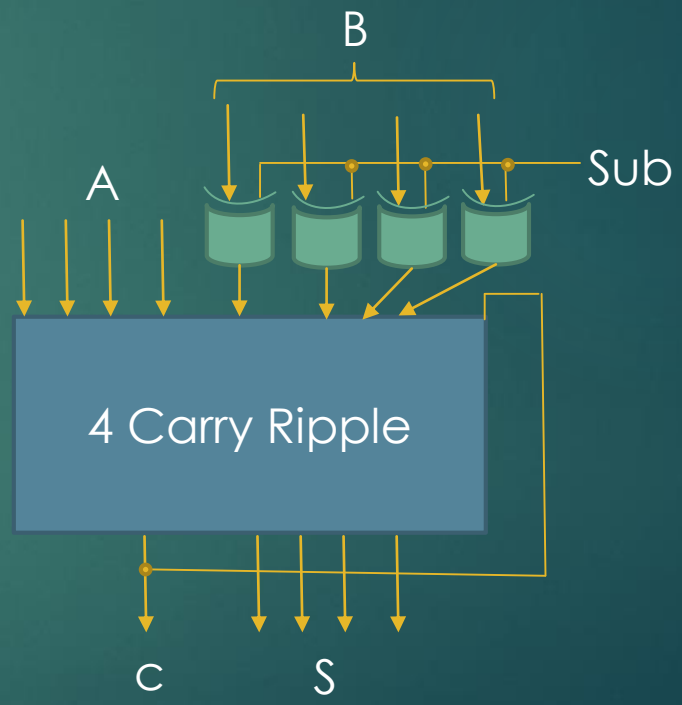
Beispiel der Subtraktion: 4 (0100) – 3 (0011)

Einerkomplement (3): 0011 → 1100

Addition: 0100 + 1100 = 0000 C1

Damit das Ergebnis stimmt muss das Carry wieder addiert werden 0000 + 1 = 0001. Dies geschieht einen Takt später, indem es in den CRA zurückgeleitet wird

Name: End-Around Carry, EAC



Weiteres Beispiel: 3 (0011) – 1 (0001) = 0011 + 1110 = 0001 C1 0001 +1 = 0010

Aufgabe 2 – Addierer/Subtrahierer

- Behandlung des Überlaufs:

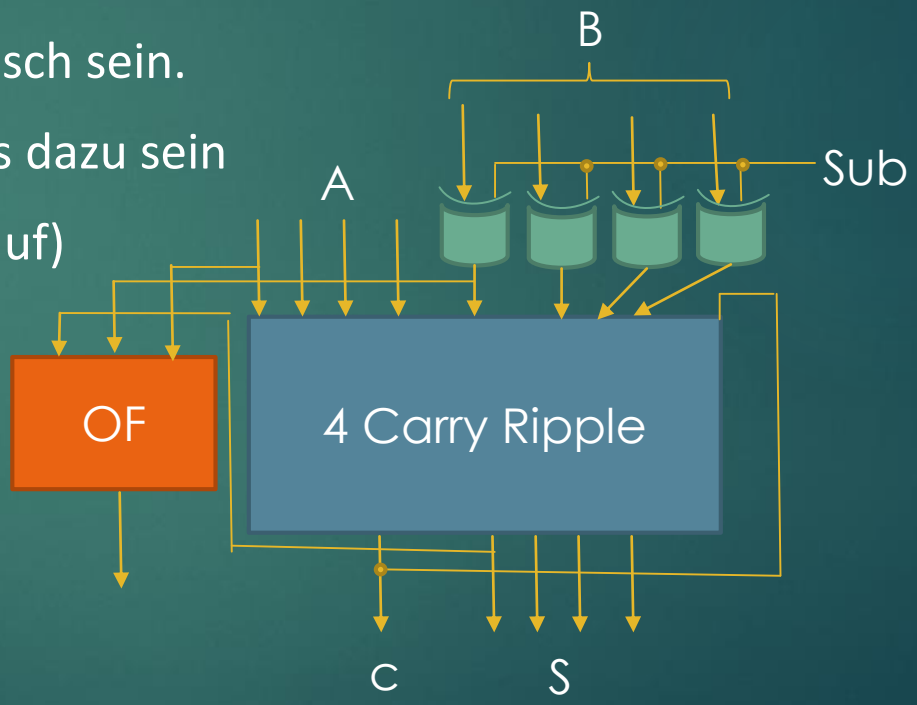
Wie erkennt man einen Überlauf (Overflow) bzw. Unterlauf:

1. Das Vorzeichen von A und B muss identisch sein.
2. Zusätzlich muss das Vorzeichenbit invers dazu sein

Beispiel: 0|100 + 0|100 = 1|000 (Überlauf)

Wie benötigen also das MSB von A und B, sowie das MSB vom Ergebnis S.

MSB = Most Significant Bit (ganz links)



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

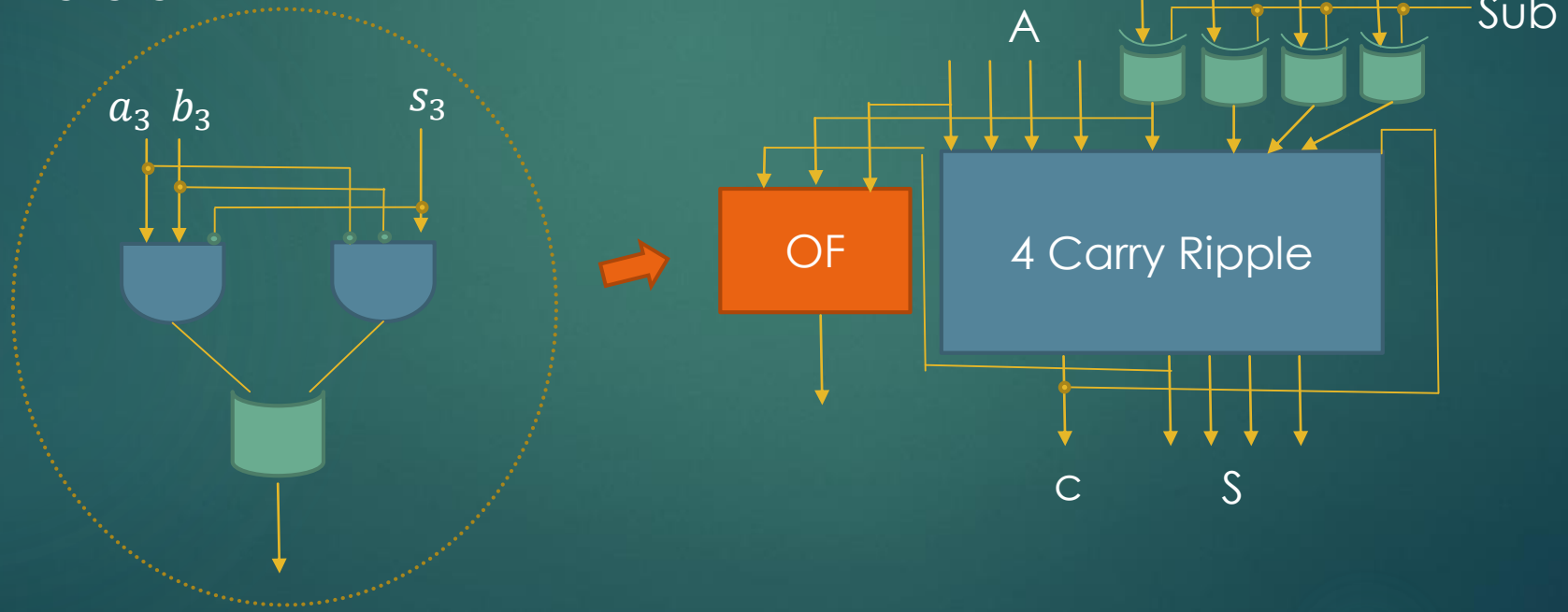
Aufgabe 2 – Addierer/Subtrahierer

- Behandlung des Überlaufs:

Overflow: Vorzeichenbit identisch; MSB von S dazu invers

Bedingungen:

$a_3 b_3 \overline{s_3} + \overline{a_3}\,\overline{b_3} s_3$



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

Beschreibung

- Führen Sie den Schritt c) für eine Recheneinheit durch, die B in Abhängigkeit von Sub in das Zweierkomplement umwandelt.

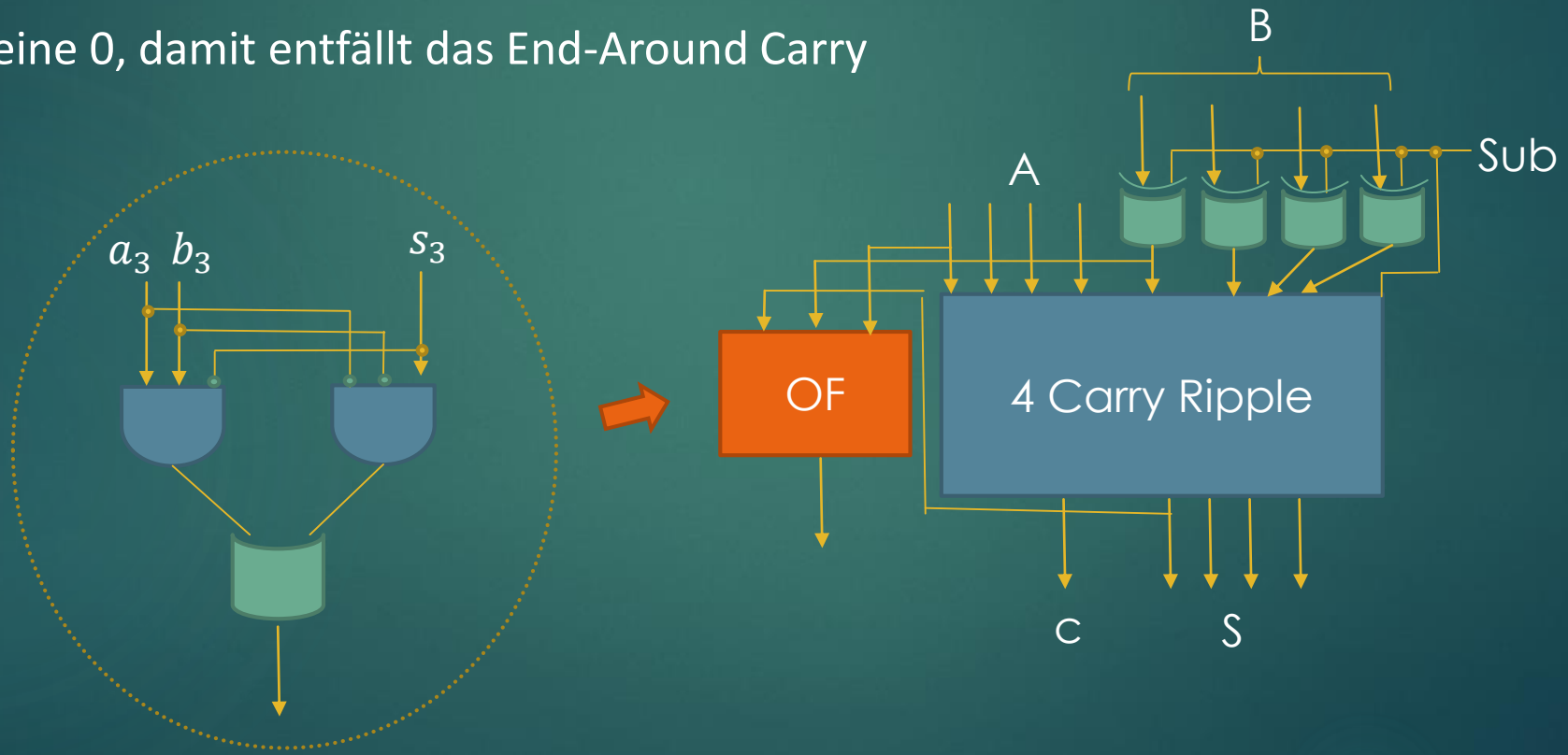
Hinweis: Nur eine kleine Änderung!

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 2 – Addierer/Subtrahierer

- Unterschied des Zweierkomplements

1. Zweierkomplement = Einerkomplement + 1, dazu nutzen wir den Carry Eingang. Das Sub-Bit ist sowieso 1, wenn subtrahiert wird, was wir ausnutzen
2. Nur eine 0, damit entfällt das End-Around Carry



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Beschreibung

In dieser Aufgabe soll ein Schaltnetz für die Addition von vier Summanden (u, v, w, x) erstellt werden, das beispielsweise für die Addition von Teilprodukten eines Multiplizierers benutzt werden könnte.

			u_3	u_2	u_1	u_0
+			v_3	v_2	v_1	v_0
+			w_3	w_2	w_1	w_0
+			x_3	x_2	x_1	x_0
	s_5	s_4	s_3	s_2	s_1	s_0

Aufgabe 3 – MO - Addierer

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

- Beschreibung

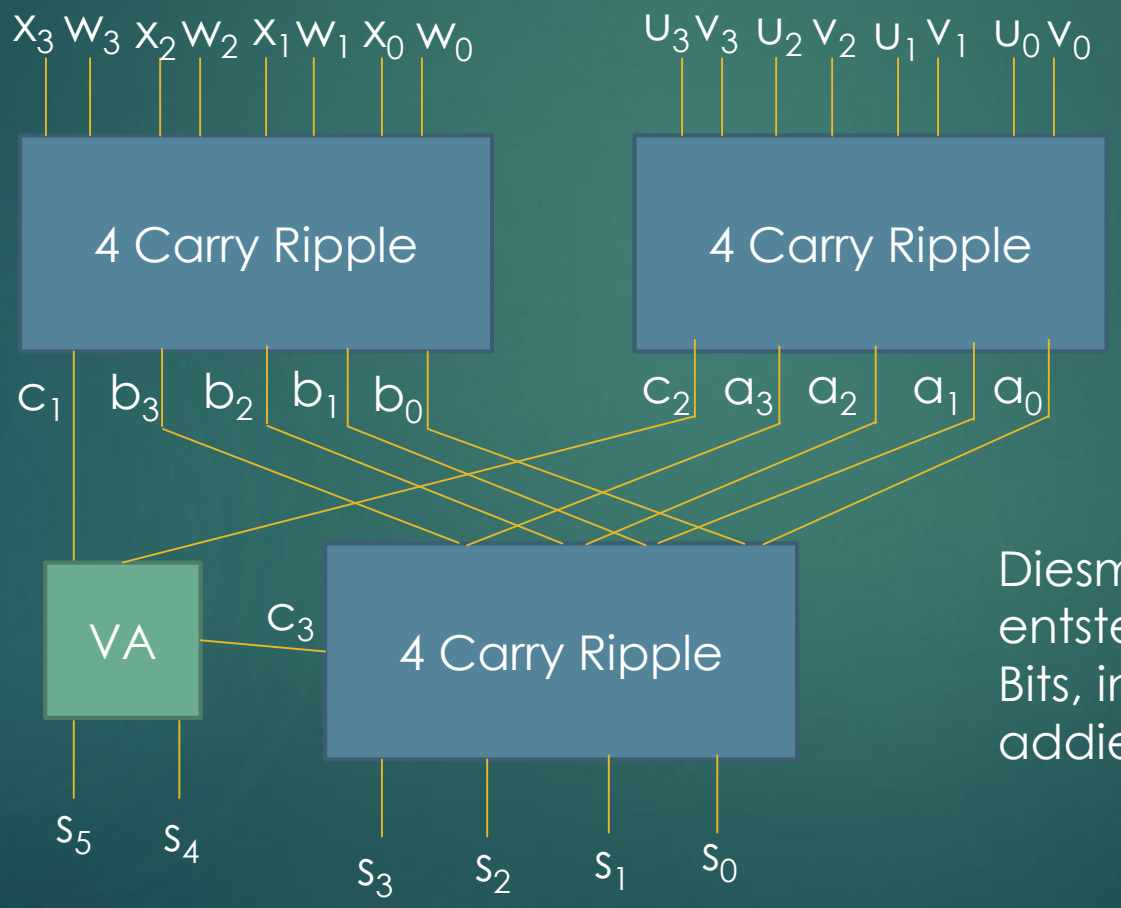
Eine mögliche Lösung besteht darin, zuerst u + v und w + x mit Hilfe je eines Carry-Ripple Addierers zu berechnen. Die Teilsummen werden dann baumartig zu dem Endergebnis zusammenaddiert.

Wie viele NAND-Gatterverzögerungszeiten umfasst dann der kritische Pfad minimal, wenn nur die in Aufgabe 2a) entworfenen Volladdierer verwendet werden dürfen?

			u_3	u_2	u_1	u_0
+			v_3	v_2	v_1	v_0
+			w_3	w_2	w_1	w_0
+			x_3	x_2	x_1	x_0
	s_5	s_4	s_3	s_2	s_1	s_0

Aufgabe 3 – MO - Addierer

- U + v = a x + w = b a + b = s



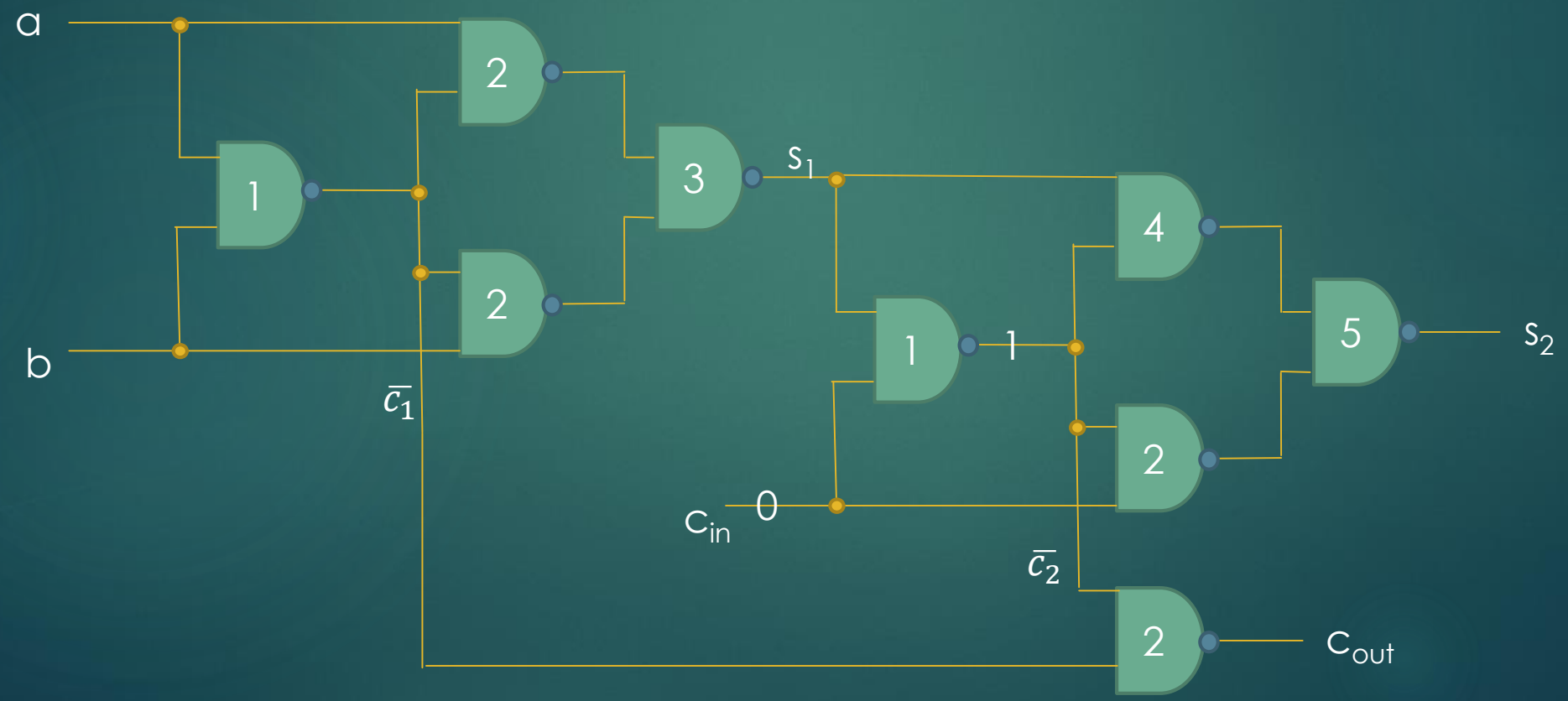
Diesmal achten wir auf den entstehende weitere Ergebnis-Bits, indem wir die Carries auch addieren

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

- Kritischer Pfad

Durch Einleiten einer Null in den ersten Volladdierer (da noch kein Carry) verkürzt sich der kritische Pfad auf 5 bzw. 2. Da das NAND sofort den richtigen Wert annimmt, der sich in späteren Takten nicht mehr ändert, da eine einzige 0 ausreicht!

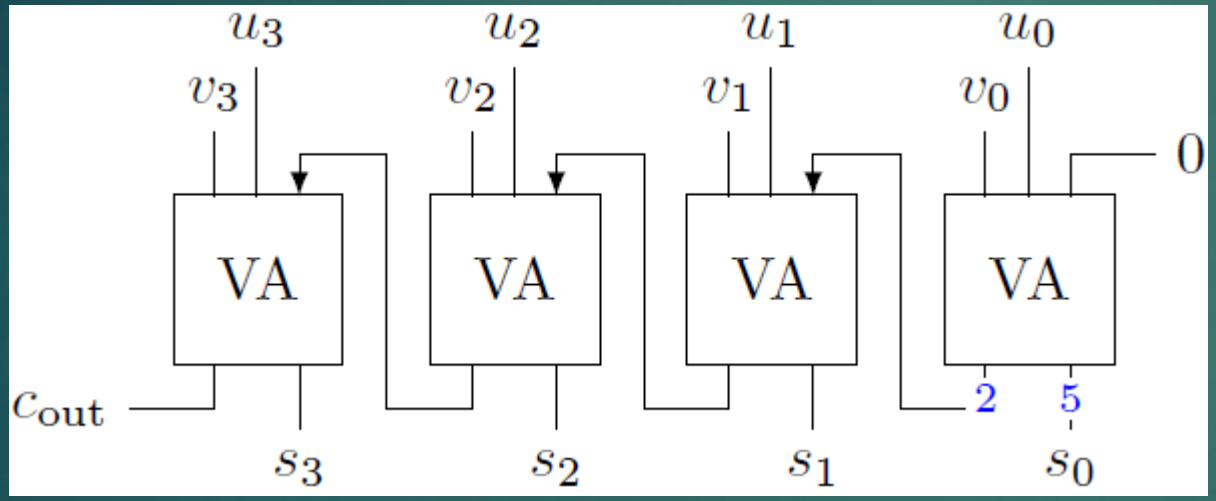


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

- 4 Carry Ripple

Eingeleitete Null führt zur Ausgang 2 und 5.

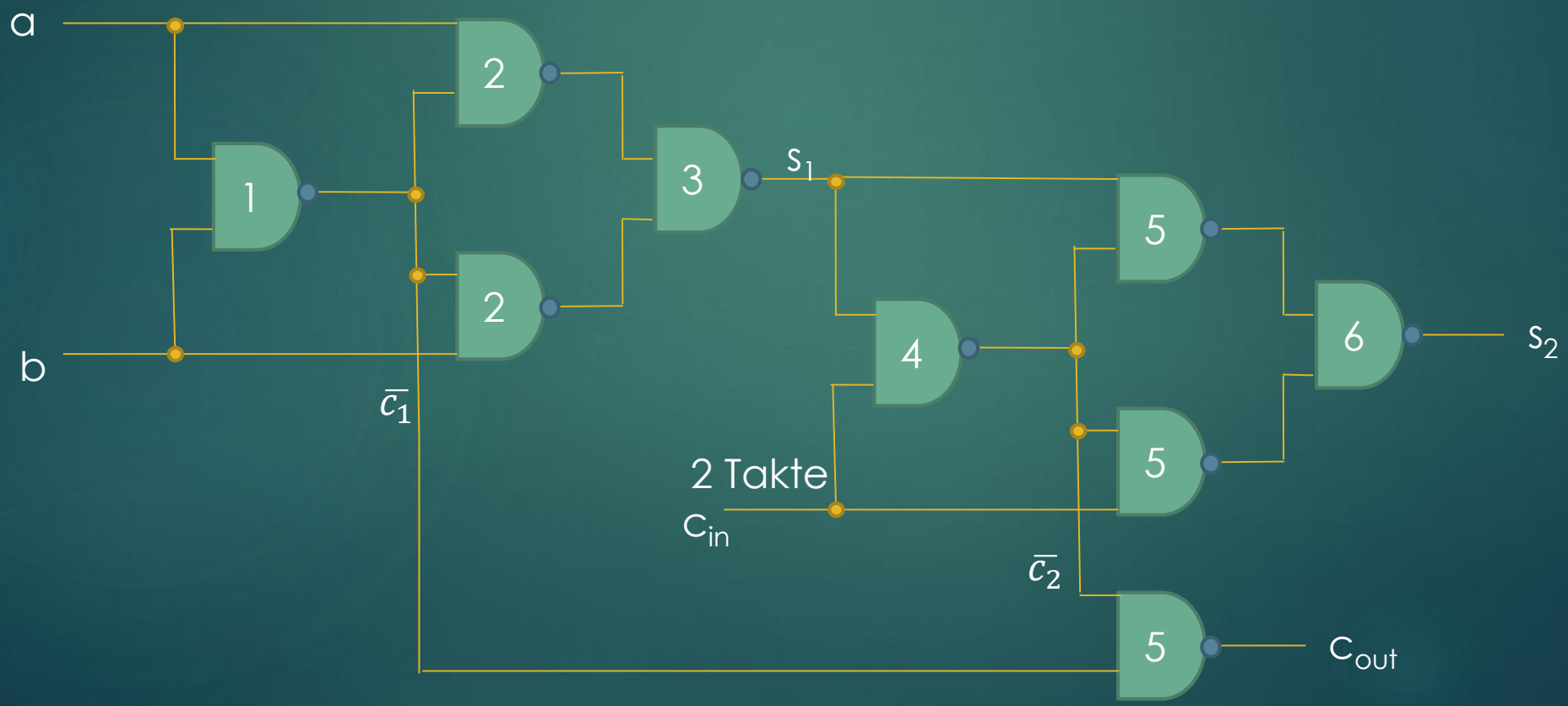


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

- Kritischer Pfad

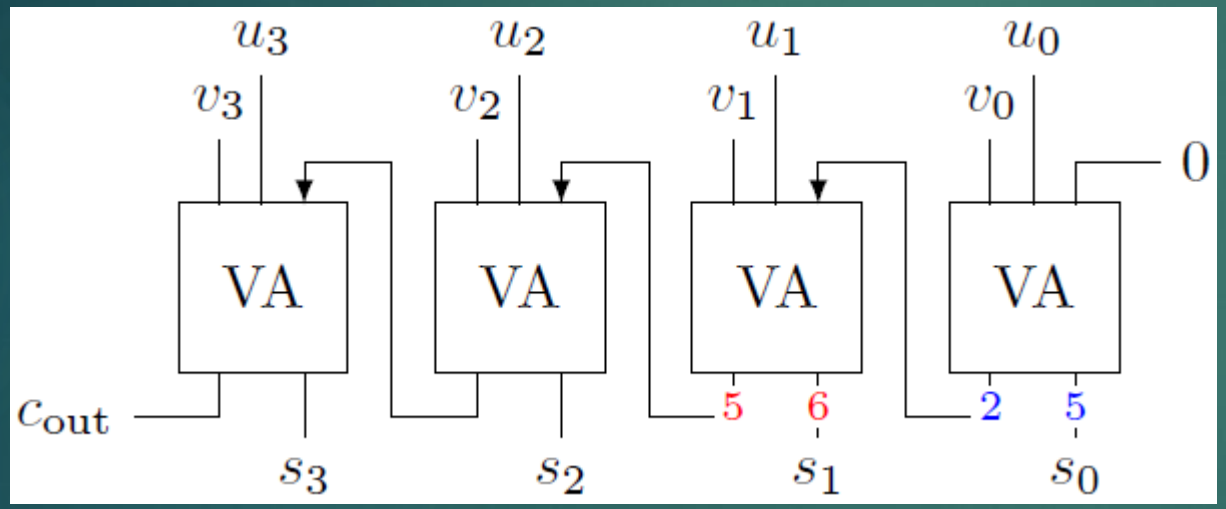
Im zweiten Schritt hilft uns die schnelle Auswertung von c_{in} nichts, da das NAND auf die Auswertung von s_1 warten muss, damit sicher ist, dass der richtige Wert am NAND anliegt



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

- 4 Carry Ripple

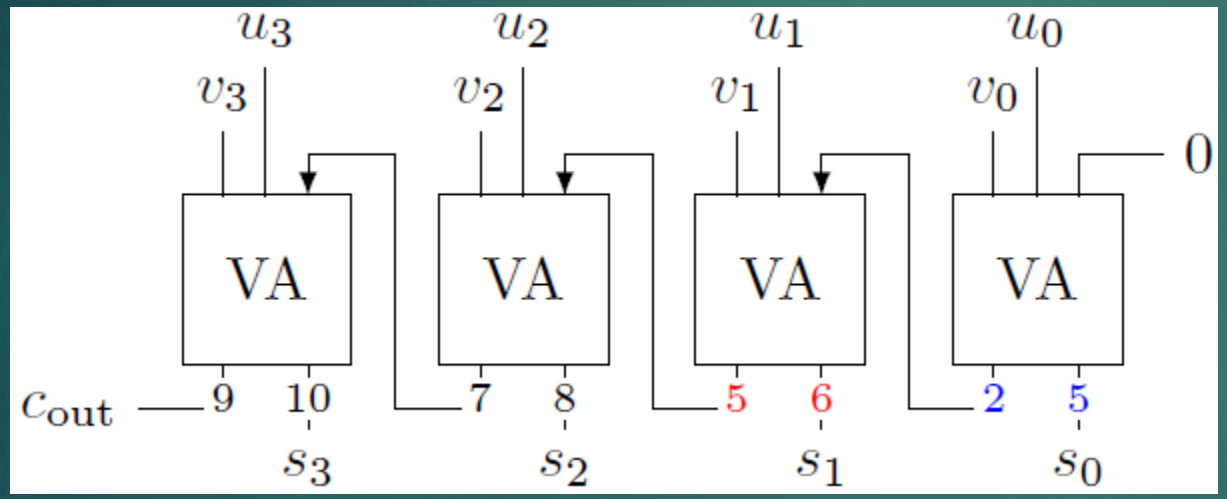


Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

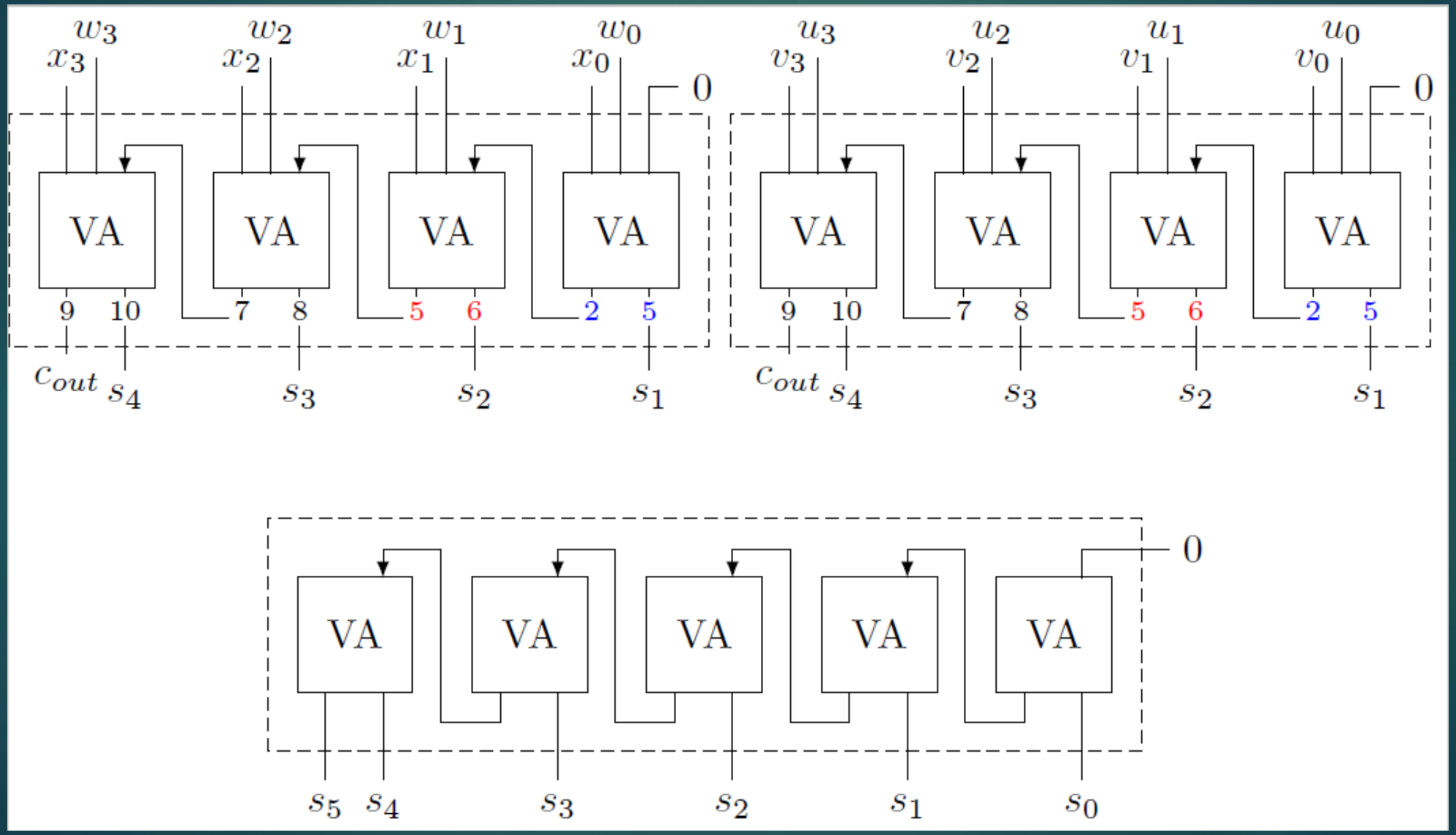
- 4 Carry Ripple

Im nächsten Schritt muss der Volladdierer 5 Takte auf c_{in} warten, bevor die Werteänderung korrekt ist. Wir brauchen ab jetzt +2/+3 Takte für c und s.



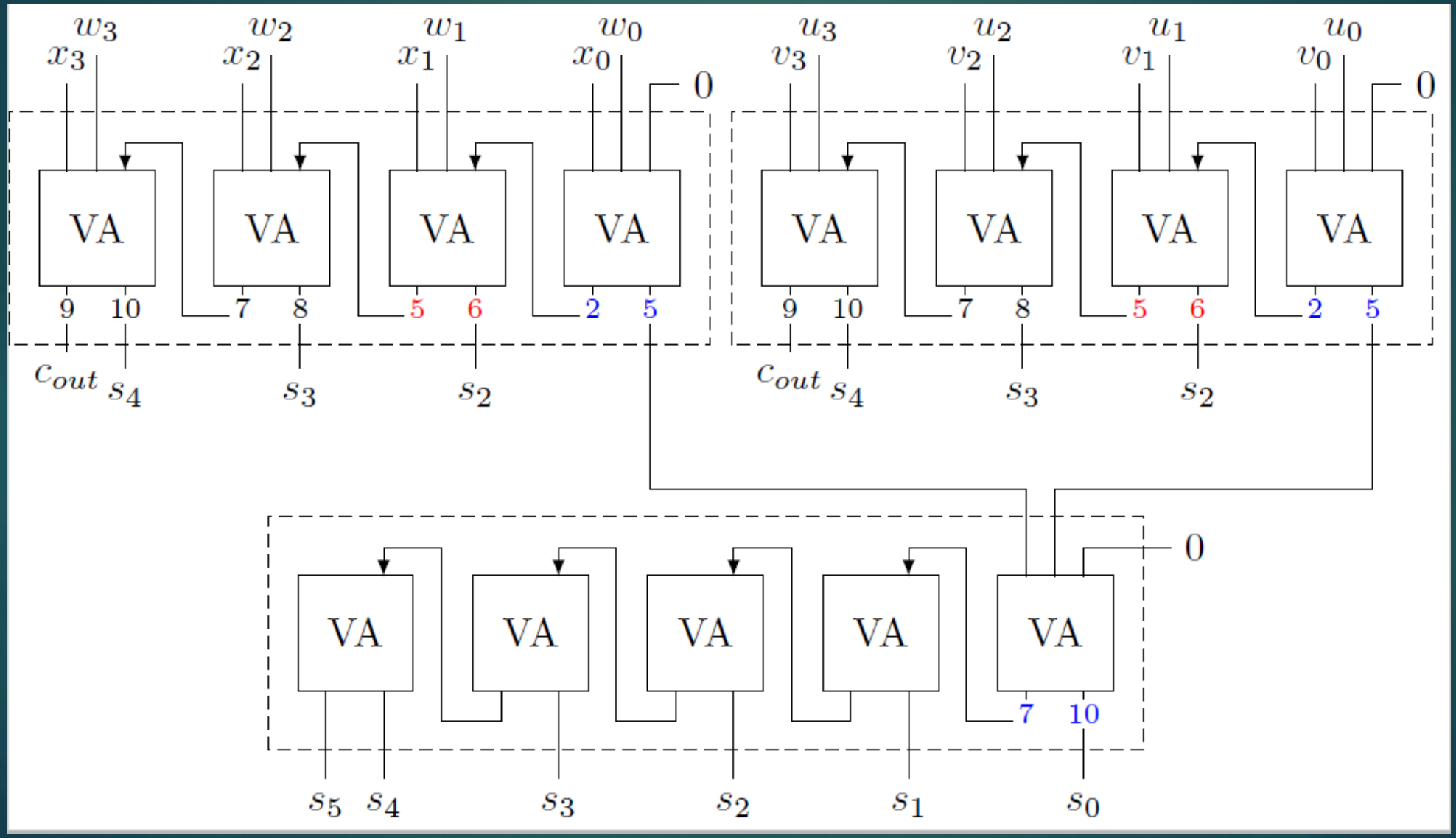
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer



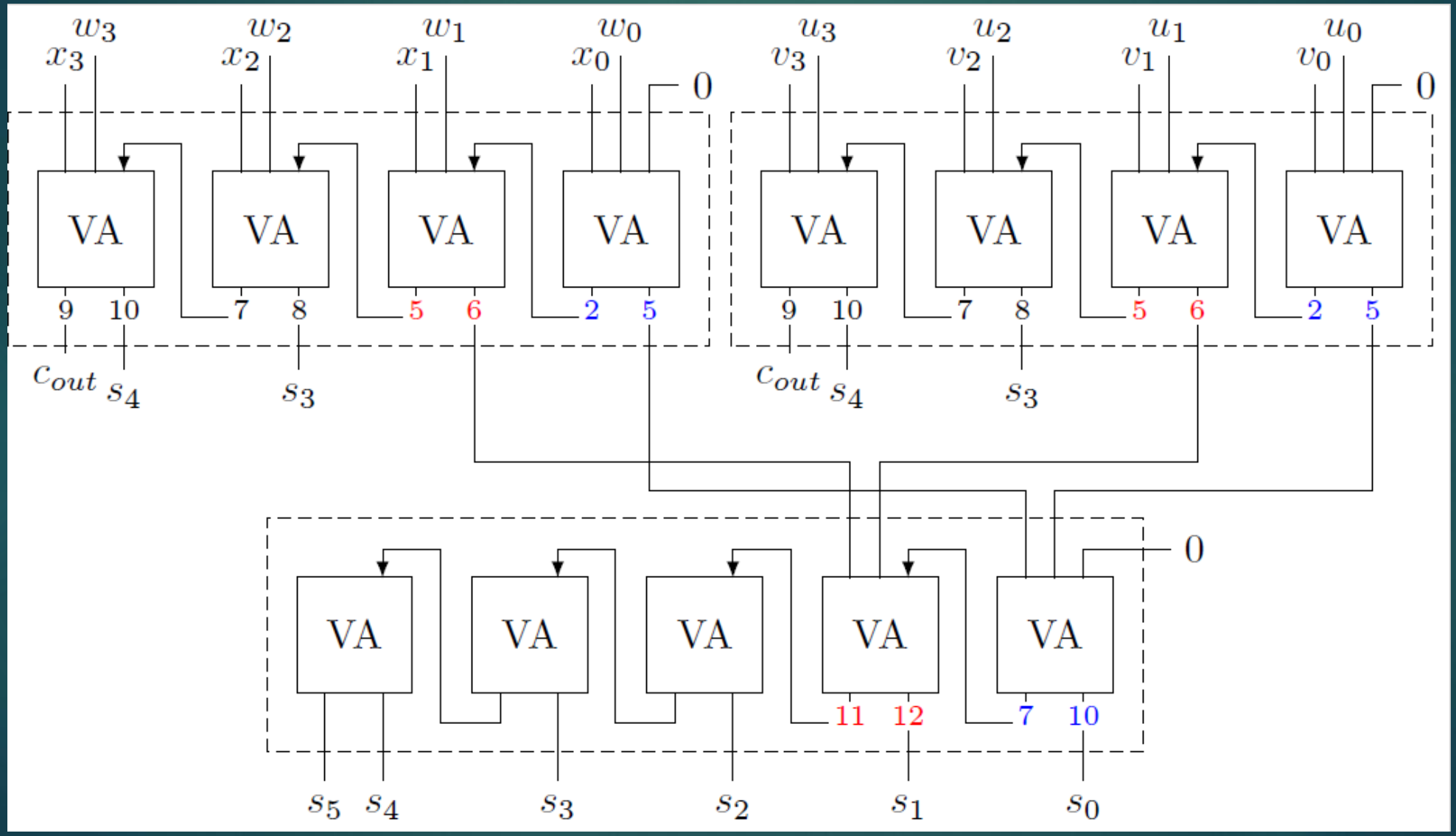
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer



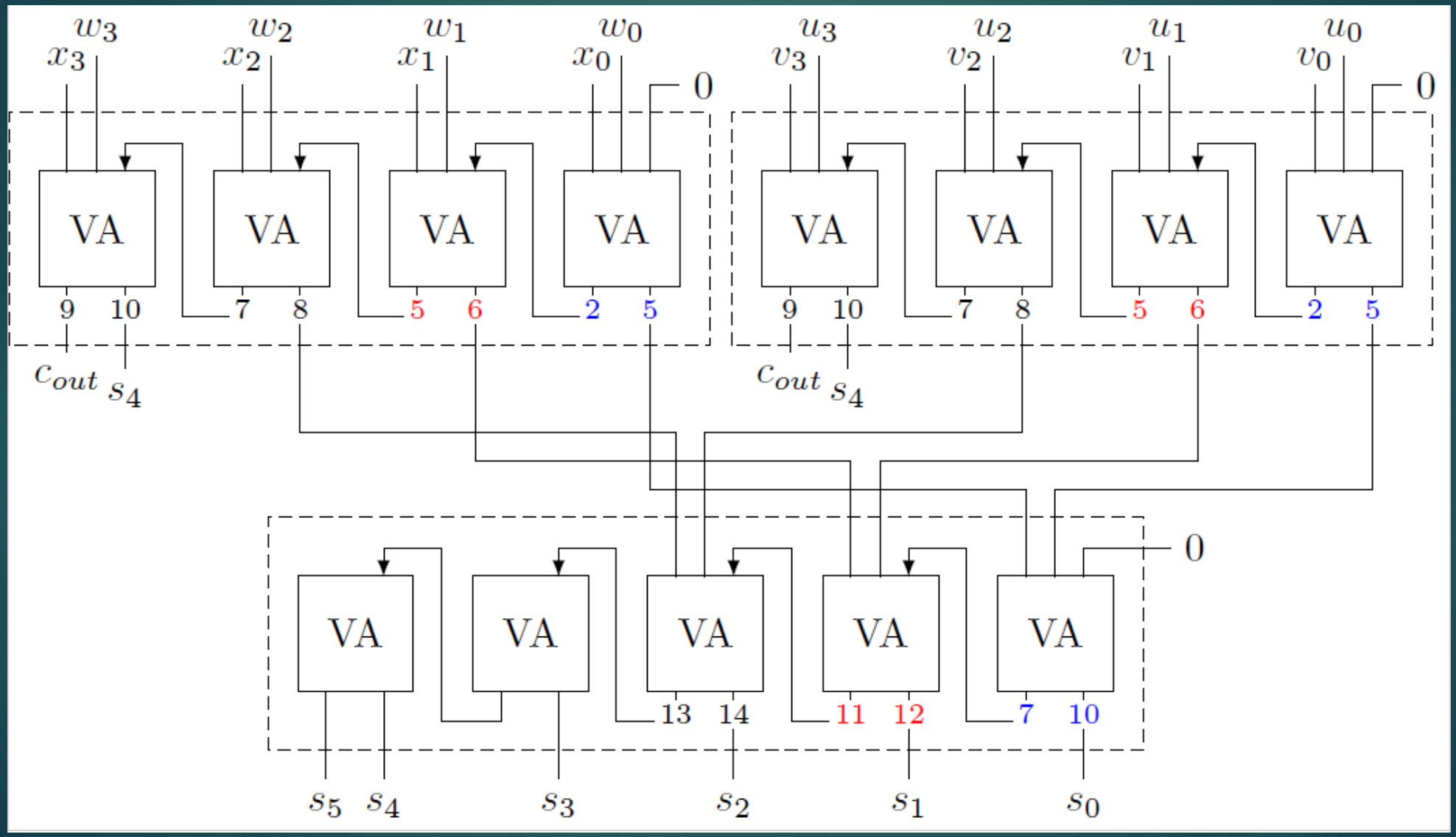
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer



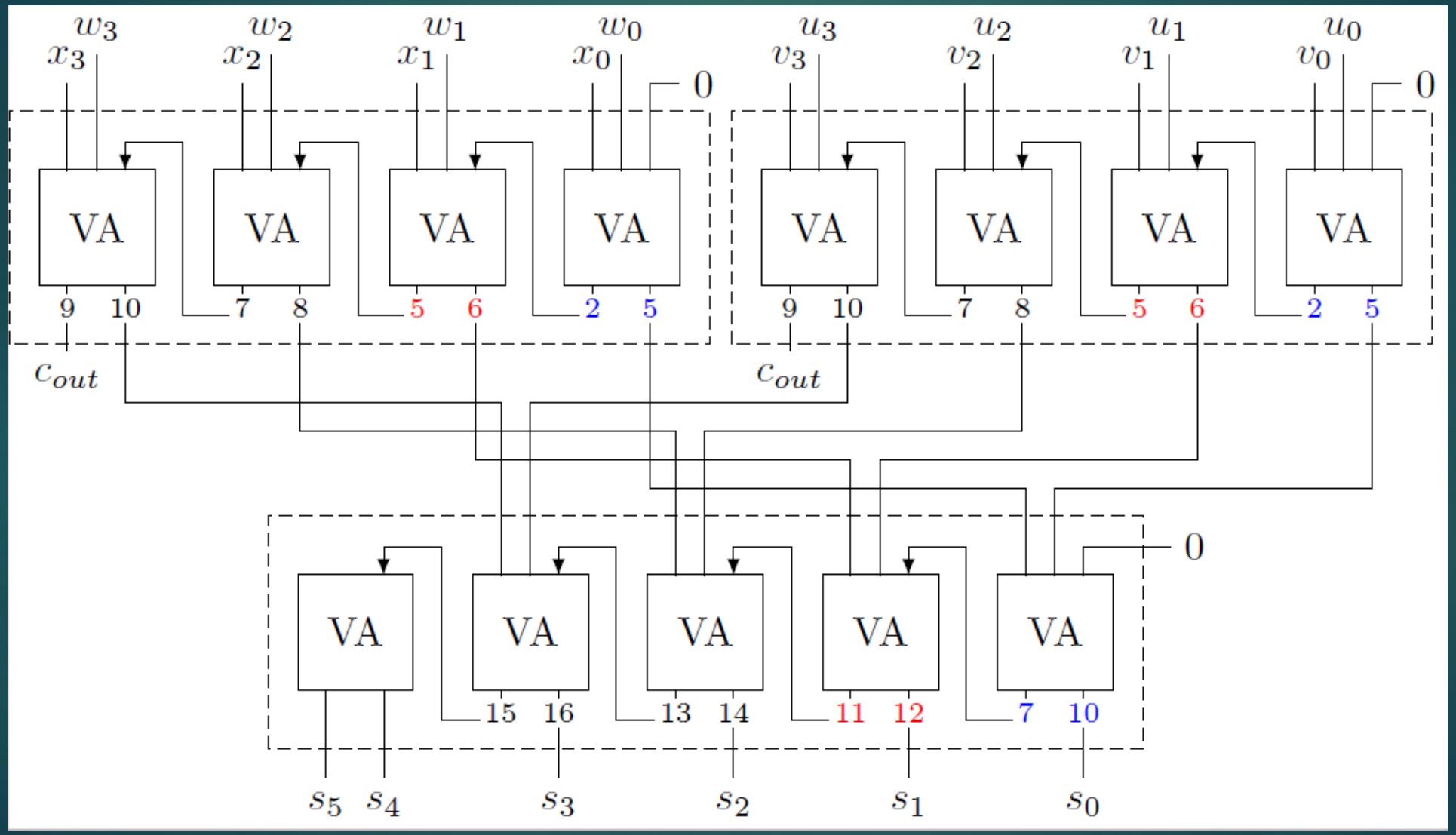
Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

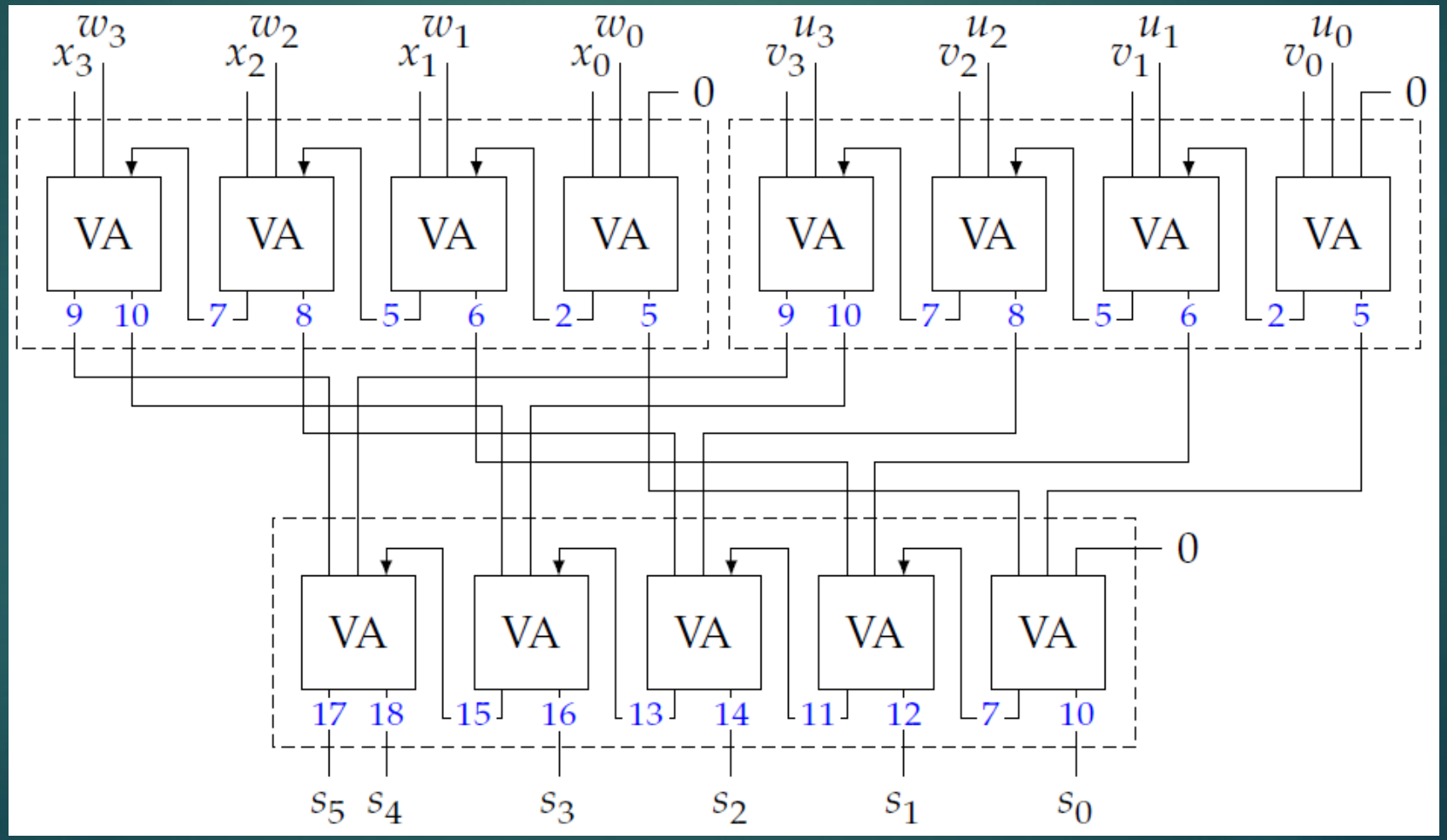
Aufgabe 3 – MO - Addierer



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Aufgabe 3 – MO - Addierer

- U + v = a x + w = b a + b = s



Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck

Vielen Dank für die schmeichelnde Aufmerksamkeit

‘Без труда ничего не даётся.‘

oder auch

‚Любишь кататься, люби и саночки возить‘

Friedrich-Alexander Universität Erlangen-Nürnberg
Jan Spieck