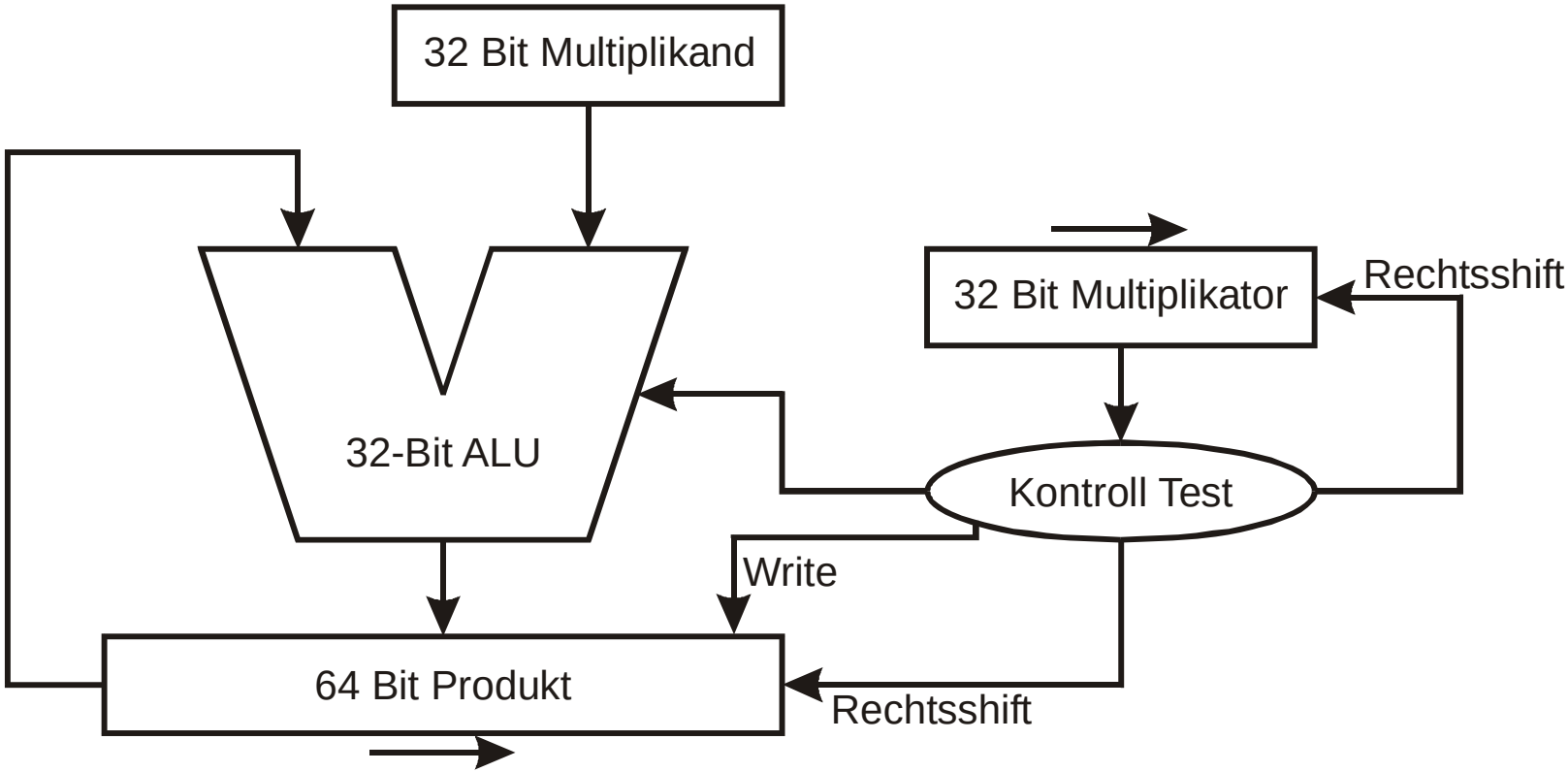
Computerarithmetik (15b)

Dazugehöriges Beispiel:

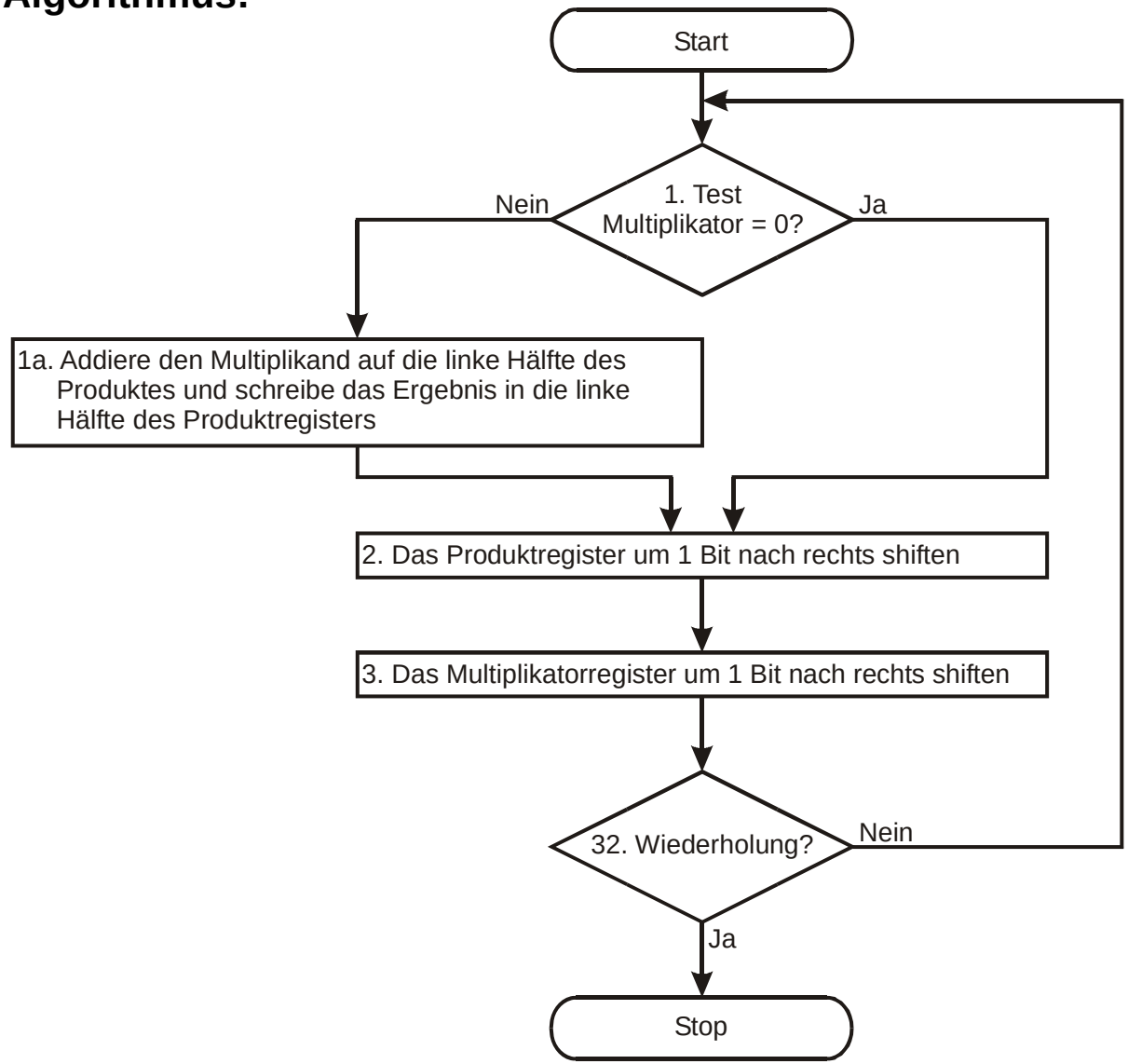
Schleife	Schritt	Multiplikator	Multiplikand	Produkt
0	Anfangswerte	001**1**	0000 0010	0000 0000
1	1a: **1** -> Prod. = Prod. + Mcand	0011	0000 0010	0000 0010
	2: Shifte Multiplikand nach links	0011	0000 0100	0000 0010
	3: Shifte Multiplikator nach rechts	000**1**	0000 0100	0000 0010
2	1a: **1** -> Prod. = Prod. + Mcand	0001	0000 0100	0000 0110
	2: Shifte Multiplikand nach links	0001	0000 1000	0000 0110
	3: Shifte Multiplikator nach rechts	000**0**	0000 1000	0000 0110
3	1: **0** -> Keine Operation nötig	0000	0000 1000	0000 0110
	2: Shifte Multiplikand nach links	0000	0001 0000	0000 0110
	3: Shifte Multiplikator nach rechts	000**0**	0001 0000	0000 0110
4	1: **0** -> Keine Operation nötig	0000	0001 0000	0000 0110
	2: Shifte Multiplikand nach links	0000	0010 0000	0000 0110
	3: Shifte Multiplikator nach rechts	0000	0010 0000	0000 0110

Computerarithmetik (16)

2. Version einer Multiplikationshardware:



Dazugehöriger Algorithmus:



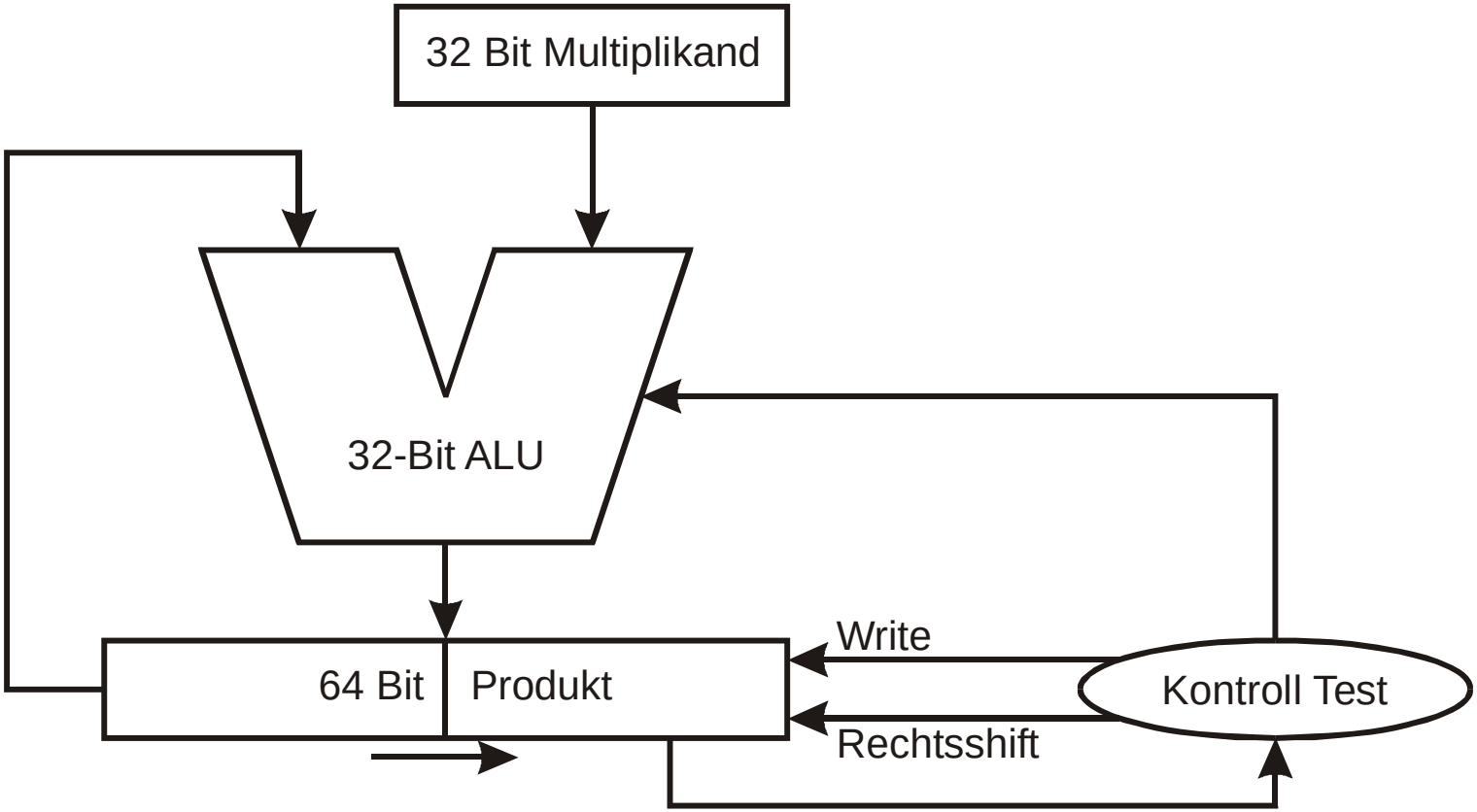
Computerarithmetik (17)

Dazugehöriges Beispiel:

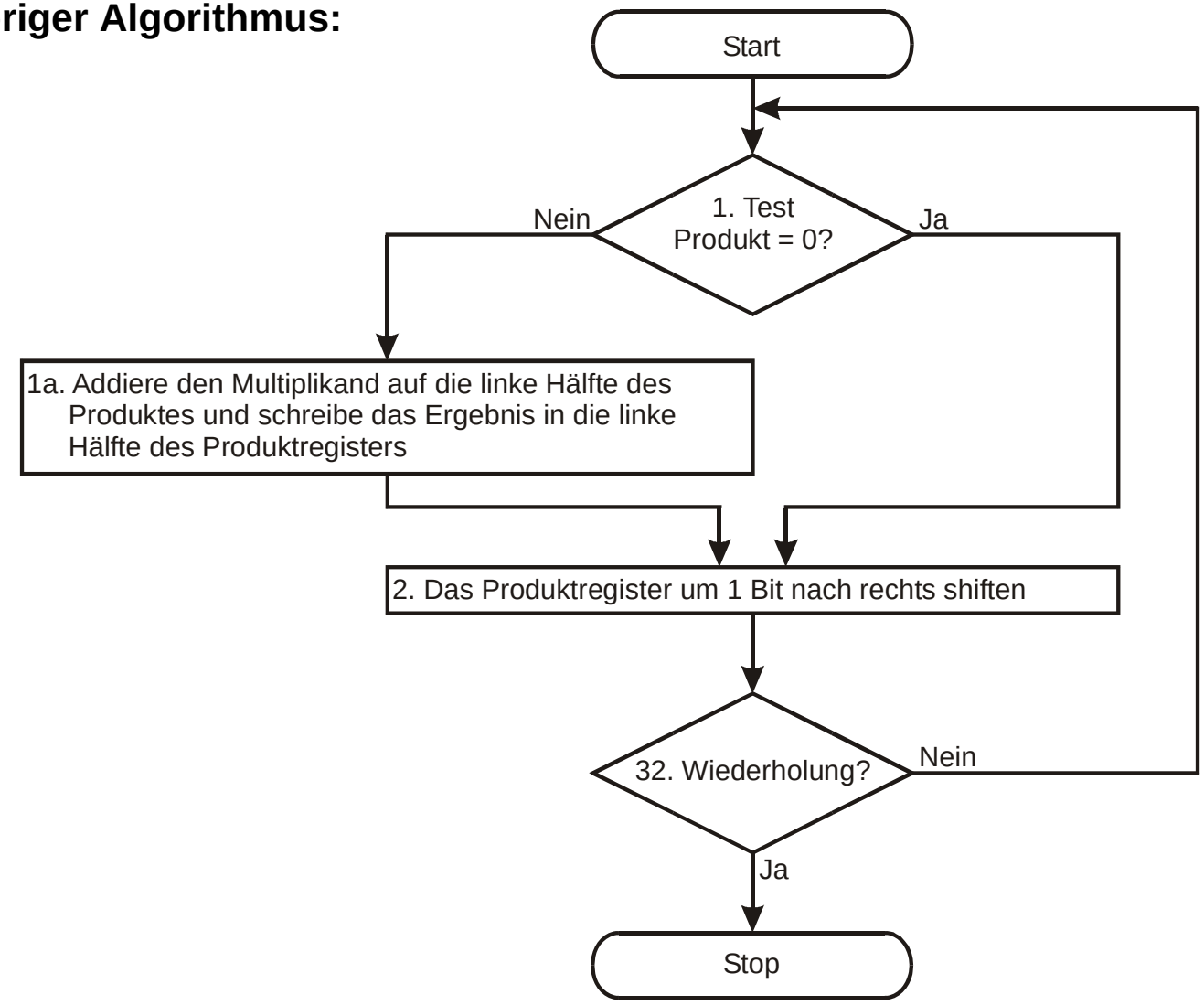
Schleife	Schritt	Multiplikator	Multiplikand	Produkt
0	Anfangswerte	001**1**	0010	0000 0000
1	1a: 1 -> Prod. = Prod. + Mcand	0011	0010	0010 0000
	2: Shifte Produkt nach rechts	0011	0010	0001 0000
	3: Shifte Multiplikator nach rechts	000**1**	0010	0001 0000
2	1a: 1 -> Prod. = Prod. + Mcand	0001	0010	0011 0000
	2: Shifte Produkt nach rechts	0001	0010	0001 1000
	3: Shifte Multiplikator nach rechts	000**0**	0010	0001 1000
3	1: 0 -> Keine Operation nötig	0000	0010	0001 1000
	2: Shifte Produkt nach rechts	0000	0010	0000 1100
	3: Shifte Multiplikator nach rechts	000**0**	0010	0000 1100
4	1: 0 -> Keine Operation nötig	0000	0010	0000 1100
	2: Shifte Produkt nach rechts	0000	0010	0000 0110
	3: Shifte Multiplikator nach rechts	0000	0010	0000 0110

Computerarithmetik (18)

3. Version einer Multiplikationshardware:



Dazugehöriger Algorithmus:



Computerarithmetik (18b)

Dazugehöriges Beispiel:

Schleife	Schritt	Multiplikand	Produkt
0	Anfangswerte	0010	0000 0011
1	1a: 1 -> Prod. = Prod. + Mcand	0010	0010 0011
	2: Shifte Produkt nach rechts	0010	0001 0001
2	1a: 1 -> Prod. = Prod. + Mcand	0010	0011 0001
	2: Shifte Produkt nach rechts	0010	0001 1000
3	1: 0 -> Keine Operation nötig	0010	0001 1000
	2: Shifte Produkt nach rechts	0010	0000 1100
4	1: 0 -> Keine Operation nötig	0010	0000 1100
	2: Shifte Produkt nach rechts	0010	0000 0110

Computerarithmetik (19)

Signed multiplication (Multiplikation mit Vorzeichen)

Beispiel für normale Multiplikation:

```
                  0 1 0 1 1 0 1
                  0 0+1+1+1+1 0
-------------------------------
                  0 0 0 0 0 0 0
                0 1 0 1 1 0 1
              0 1 0 1 1 0 1
            0 1 0 1 1 0 1
          0 1 0 1 1 0 1
        0 1 0 1 1 0 1
      0 0 0 0 0 0 0
    0 0 0 0 0 0 0
-------------------------------
0 0 0 1 0 1 0 1 0 0 0 1 1 0
```

Das gleiche Beispiel mit **Booth-Algorithmus:**

```
                  0 1 0 1 1 0 1
                  0+1 0 0 0-1 0
-------------------------------
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 0 1 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 1 1 0 0
0 0 0 0 0 0 0 0
-------------------------------
0 0 0 1 0 1 0 1 0 0 0 1 1 0
```

Beobachtung:

- Es sind wesentlich weniger Teiladditionen notwendig.
- Es können positive und negative Faktoren verarbeitet werden.

Computerarithmetik (19a)

Der Booth-Algorithmus

Idee:

- Darstellung des Multiplikators durch eine größere Zahl abzüglich einer Differenz
- Größere Zahl wird so gewählt, dass möglichst wenig Einsen, d.h. 2er Potenz
- Größere Zahl und Differenz zum Multiplikator sind neue Multiplikatoren

7	0	1	1	1
8	1	0	0	0
-1	0	0	0	-1
	+1	0	0	-1

Multiplikator wird von rechts beginnend umcodiert:

	⌐				⌐			
0	0	1	1	1	1	0	0	(0)
0	+1	0	0	0	−1	0	0	

- Bei Wechsel von 0 auf 1 schreibe -1 unter die 1
- Bei Wechsel von 1 auf 0 schreibe +1 unter die 0
- Erfolgt kein Wechsel, so schreibe 0, d.h. es erfolgt keine Addition in diesem Schritt

Computerarithmetik (20a)

Beispiel 2 x (-3):

Schleife	Schritt	Multiplikand	Produkt
0	Anfangswerte	0010	0000 1101 0
1	1c: 10 -> Prod. = Prod. + Mcand	0010	1110 1101 0
	2: Shifte Produkt nach rechts	0010	1111 0110 1
2	1b: 01 -> Prod. = Prod. + Mcand	0010	0001 0110 1
	2: Shifte Produkt nach rechts	0010	0000 1011 0
3	1c: 10 -> Prod. = Prod. + Mcand	0010	1110 1011 0
	2: Shifte Produkt nach rechts	0010	1111 0101 1
4	1d: 11 -> keine Operation nötig	0010	1111 0101 1
	2: Shifte Produkt nach rechts	0010	1111 1010 1

Computerarithmetik (20b)

Schnelle Multiplikation

A) Beschleunigung der Berechnung durch Verringerung der Anzahl der Summanden

Bit Pair Recoding

Ausgangspunkt: Codierung nach Booth-Algorithmus
Bit-Paare von 2 Einsen mit unterschiedlichem Vorzeichen können wie folgt umcodiert werden (bei gleichem Vorzeichen findet nach Definition keine Aufsummierung statt):

+2n –n = +n

+1 -1
0 +1

-2n +n = -n

-1 +1
0 -1

Ergebnis:

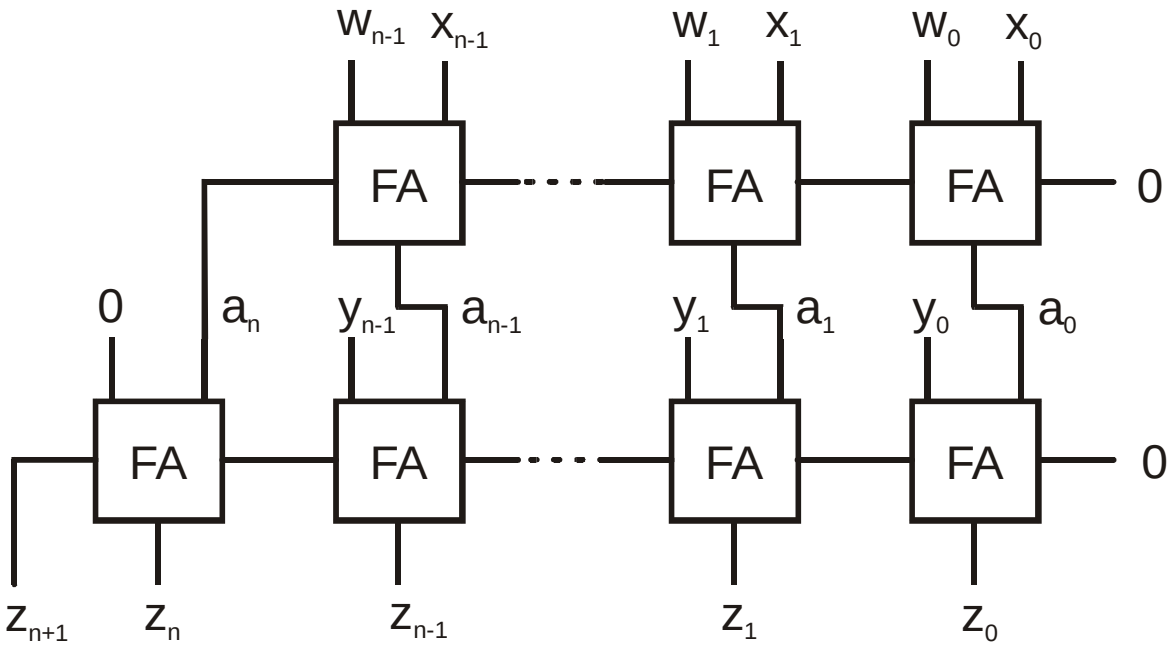
1 Stelle in einem Bitpaar ist stets 0 -----> somit Halbierung der maximalen Summandenanzahl

Computerarithmetik (20c)

Schnelle Multiplikation

B) Beschleunigung der Berechnung durch modifizierte Addition

Carry- Ripple- Adder (klassisch) bzw. Lookahead::



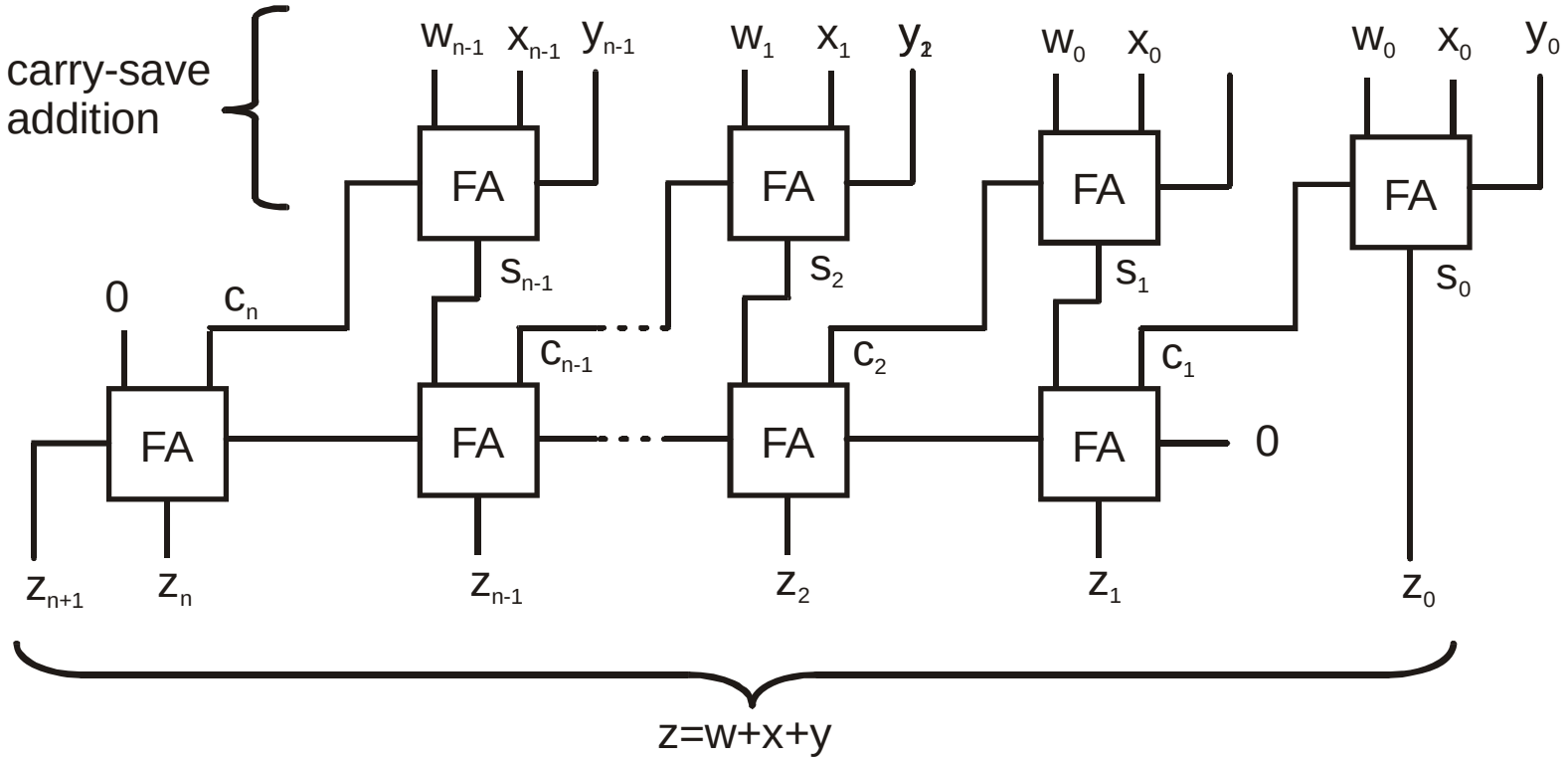
Nachteil: resultierendes Carry-Bit wird im selben Additionsvorgang auch als Eingangswert benötigt
---> längere Rechenzeit durch Warten

Computerarithmetik (20d)

Schnelle Multiplikation

B) Beschleunigung der Berechnung durch schnelle Addition

Beispiel für eine Implementierung mit CSA (Carry Save - Addition):



Vorteil: resultierendes Carry-Bit wird nur bei letzter Addition auch als Eingabewert benötigt
---> stark verkürzte Rechenzeit
---> Multiplikation bzgl. Geschwindigkeit auf Größenordnung einer Addition reduziert

Computerarithmetik (30)

Zusammenfassung:

- Computerarithmetik ist endlich und kann folglich **nicht** übereinstimmen mit der natürlichen Arithmetik

- Selbst der IEEE 754 - Standard für die floating point - Darstellung, wie jede andere auch, ist fast immer eine **Approximation** der realen Zahlen.

- Rechnersysteme müssen dafür sorgen, den daraus resultierenden Unterschied zwischen Computerarithmetik und Arithmetik in der realen Welt möglichst zu minimieren.

- Massive Performanzsteigerung durch folgende Techniken:
  - Carry-lookahead - Techniken für Addierer mit hoher Performanz
  - Für schnelle Multiplizierer:
    - Booth-Algorithmus und Methode des Bit-pairing zur Reduzierung der Anzahl der benötigten Additionen für die Erzeugung des Produkts
    - Carry-save - Addition zur nochmaligen substantiellen Reduzierung auf letztlich zwei zu addierende Summanden nach der carry-lookahead - Technik.

- Informatiker sollten sich dieser Zusammenhänge bewußt sein.

Rückblick

Gliederung :

- Aufbau und Grobstruktur eines Rechners
- Aufbau kombinatorischer Schaltnetze auf der Basis von Gattern
- Analyse durch Boolesche (Schalt-) Algebra
- Aufbau und Test von sequentiellen Schaltnetzen (Schaltwerken) auf der Basis von Flip-Flops
- Codes
- Computerarithmetik
  - Zahlendarstellung
  - IEEE Standard
  - Binäre Arithmetik

Rechnersysteme im SS 09

Inhalt:

- Wie werden Programme in einem Rechner ausgeführt aus der Sicht von Ebene 2 (Assembler-Ebene)
  - welche M□öglichkeiten gibt es, Befehlsfolgen vom Hauptspeicher in die CPU zu bringen und wie werden sie dann ausgeführt
  - welche Adressierungstechniken gibt es für Hauptspeicherzellen und CPU-Register

- Struktur und Funktionsweise der CPU
  - Adresspfad
  - Datenpfad
  - Kontrolleinheit (Befehlsdekodierung)
  - RISC-Konzept

- Struktur und Funktionsweise des Hauptspeichers
  - Aufbau eines Speichers
  - Entwurf einer Speicherhierarchie
    - Cache
    - virtuelles Speicherkonzept

- Parallelrechner
  - Kommunikationsmodelle
  - Verbindungsnetzwerke
  - Taxonomie