Digitaltechnik & Rechnersysteme

Spezielle Schaltnetze, Arithmetik

Martin Kumm

HOCHSCHULE FULDA
UNIVERSITY OF APPLIED SCIENCES

WiSe 2025/2026

Probeklausuren online

Teil 1 der 1. Probeklausur ist nun online

Dieser kann als Selbsttest verwendet werden (Musterlösung ist online).

Bitte nicht selbst betrügen: Erst nach der eigenen Lösung in die Musterlösung schauen!

Eigene Aufgabe im 7. Übungsblatt

Im 7. Übungsblatt sollen Sie eine »Eigene Aufgabe« erstellen

Hier sollen Sie eine eigene kombinatorische Problemstellung überlegen welche mit den eingeführten *Tools* (Wahrheitstabelle, KV-Diagramm, etc.) bearbeitet und über einen alternativen Rechenweg überprüft werden soll.

Die Lösung soll per Moodle geteilt werden (extra Übungsmaterial)

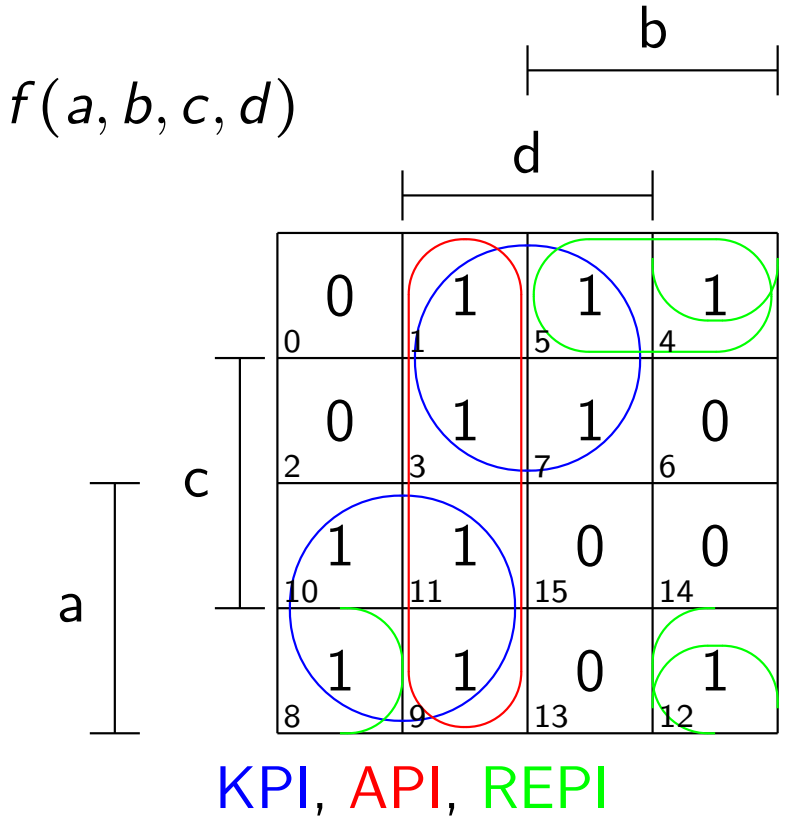
Wenn genügend Aufgaben zusammenkommen werde ich eine davon in der Klausur abfragen (nach Ankündigung!) 😉

Was bisher geschah...

- Definitionen: Primimplikant, KPI, API, REPI
  - KPI: Enthält mindestens eine 1 die von keinem anderen Primimplikanten überdeckt wird
  - API: Alle 1en werden von KPIs überdeckt
  - REPI: Alle anderen
- Minimierung mit KV-Diagrammen
  - KPI werden immer benötigt
  - API werden nie benötigt
  - Aus REPIs muss eine geschickte Auswahl getroffen werden
- Don't cares
  - Vereinfachungsmöglichkeit im KV-Diagramm wenn Ausgabewert für bestimmte Eingabe egal (*don't care*) ist

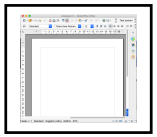
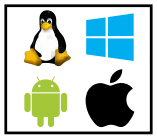
Beispiel aus letzter Vorlesungsaufgabe

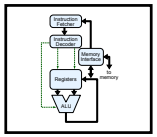
Markieren Sie alle Primimplikanten.

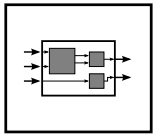
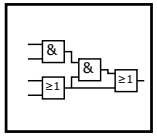


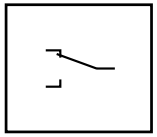
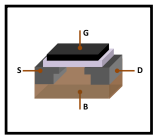
Alle KPIs sind nötig sowie einer der beiden REPIs.

Die Macht der Abstraktion



User-Software

Betriebssystem

Computer

Prozessor

Funktionseinheiten ← Thema jetzt

Logik-Gatter

Schalter

Transistor

Inhalte

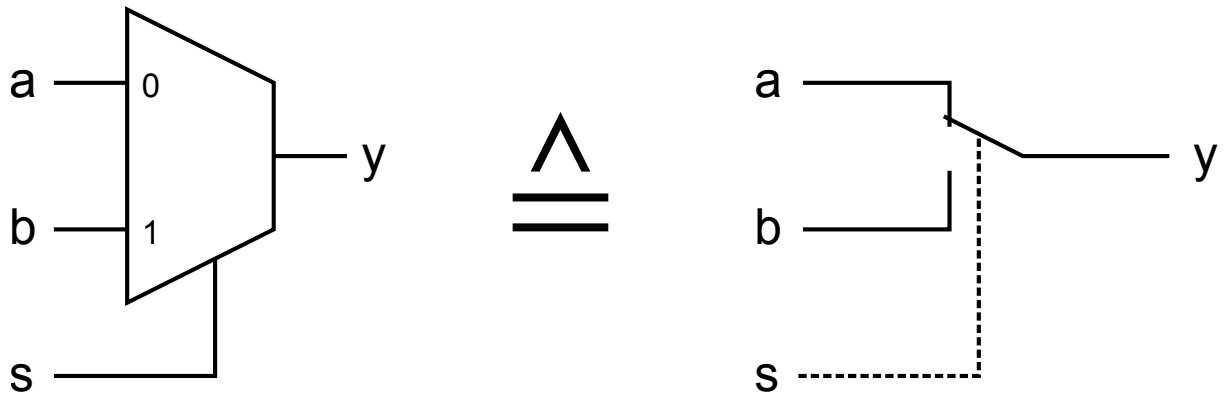
AI Angewandte Informatik

Multiplexer (MUX)

Ein $2:1$ Multiplexer (MUX) kann 2 Eingänge auf einen Ausgang schalten

Gesteuert wird dies über einen Steuer-Eingang (*Select*)

Funktionsweise: Wenn der Select-Eingang $s = 0$, wird Eingang 0 durchgeschaltet, wenn $s = 1$, wird Eingang 1 durchgeschaltet.

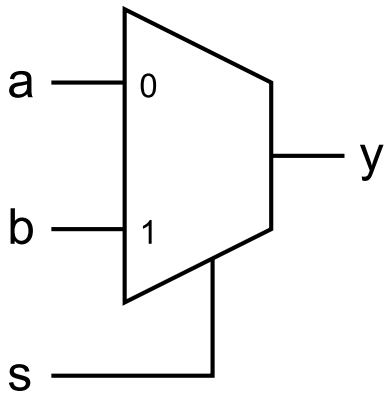


Anwendungsgebiete: Programmierbare (Rechner-)verbindungen

Multiplexer (MUX)

Funktionsweise: Wenn der Select-Eingang $s = 0$, wird Eingang 0 durchgeschaltet, wenn $s = 1$, wird Eingang 1 durchgeschaltet.

Schaltsymbol:



Wahrheitstabelle:

s	a	b	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

KDNF: $y = \overline{s}\, a\overline{b} + \overline{s}\, ab + s\overline{a}\, b + sab$

Multiplexer

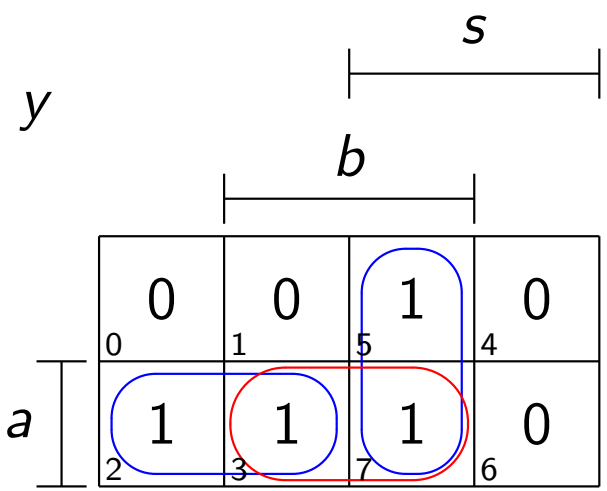
Vereinfachung:

$$
\begin{aligned}
y &= \overline{s}\, a\overline{b} + \overline{s}\, ab + s\overline{a}\, b + sab \\
&= \overline{s}\, a(\overline{b} + b) + sb(\overline{a} + a) \\
&= \overline{s}\, a \underbrace{(\overline{b} + b)}_{=1} + sb \underbrace{(\overline{a} + a)}_{=1} \\
&= \overline{s}\, a + sb
\end{aligned}
$$

Multiplexer

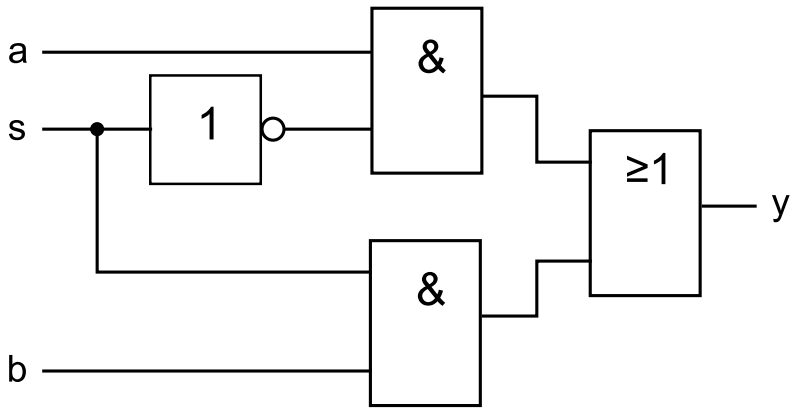
s	a	b	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

KV-Diagramm:

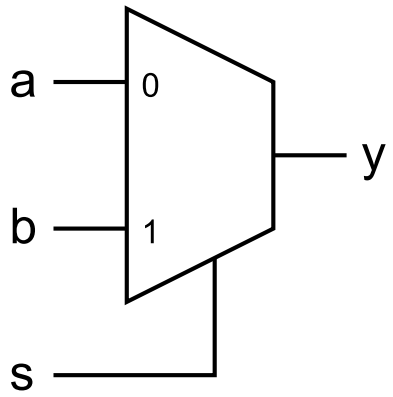


$$y = \overline{s}\, a + s b$$

2 : 1 Multiplexer Symbol



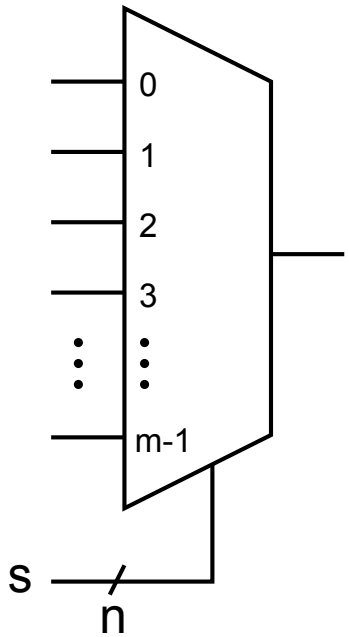
Realisierung



Symbol

Allgemeiner Multiplexer

Bei $m = 2^n$ Eingängen werden n Steuerleitungen für die Auswahl benötigt.



Die Ein/Ausgabe kann hier auch mehr Bits/Datenwörter umfassen.

Decoder

Schaltnetz, das n Eingänge auf 2^n Ausgänge abbildet.

Schaltet für jede Eingangskombination genau einen Ausgang auf 1 (auch *one hot* code genannt)

Allgemein können Ausgangsseitig weniger als 2^n Ausgänge vorgesehen sein: $m \leq 2^n$ bei n-zu-m-Decodern

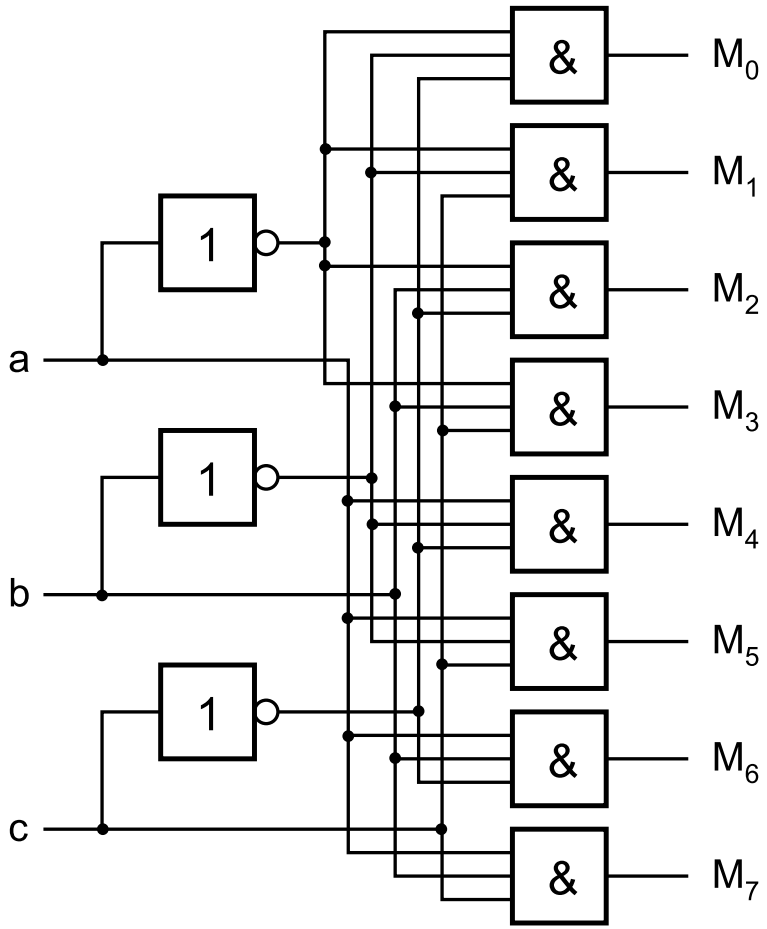
Beispiel: 3 : 8 Decoder

Wahrheitstabelle zum 3 : 8 Decoder:

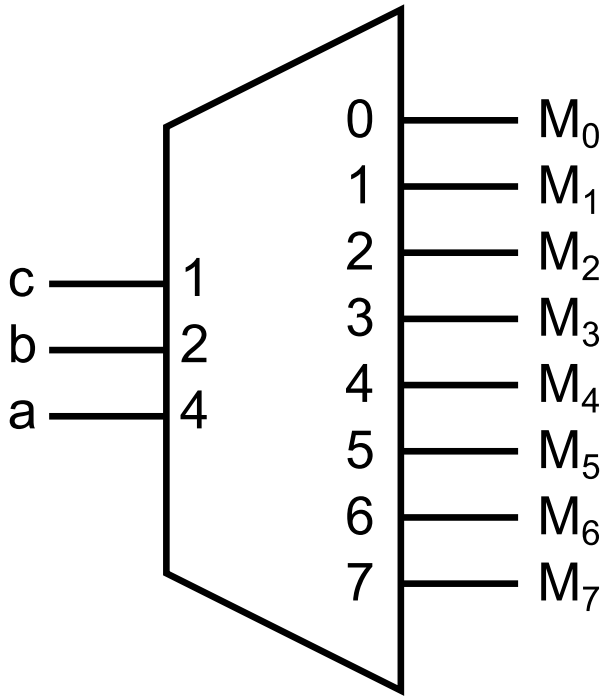
Index	a	b	c	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

Ausgang i entspricht Minterm M_i.

3 : 8 Decoder Schaltbild & Symbol



Realisierung



Symbol

Grundlegende Addition

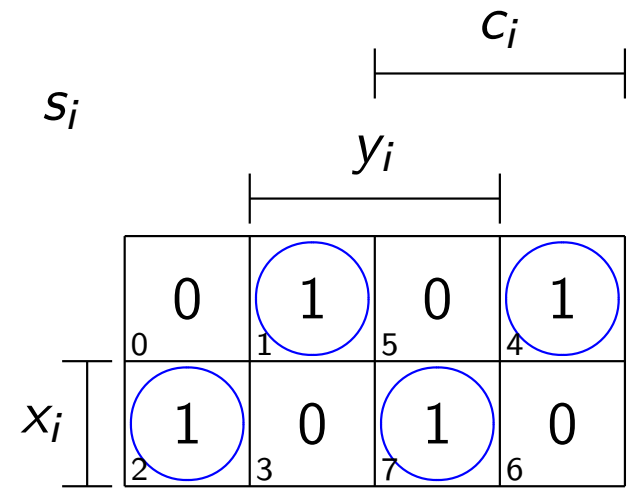
- Betrachten wir die Addition zweier ganzer Zahlen $X = 1010111100_2$ und $Y = 1010010_2$ $(700_{10} + 82_{10})$:

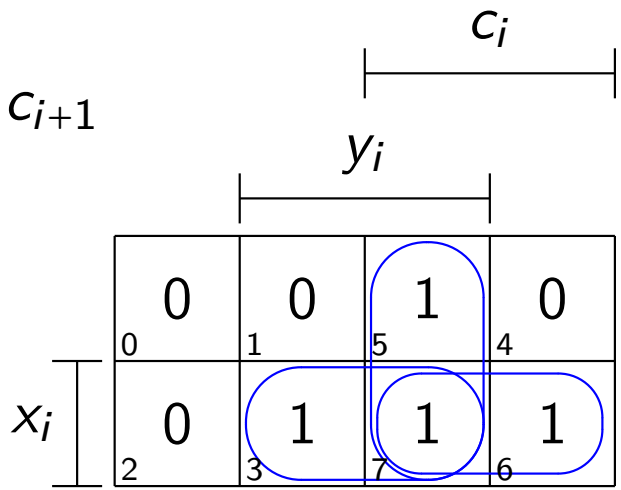
$$\begin{array}{rr} X & 1010111100_2 \\ +Y & +0001010010_2 \\ \hline = S & = 1100001110_2 \end{array}$$

- Das Bit s_i (i-te Bit von S) hängt ab von x_i, y_i und vom aktuellen Übertragsbit c_i (Carry)
- Das nächste Übertragsbit c_{i+1} hängt ebenfalls ab von x_i, y_i und vom aktuellen Übertragsbit c_i
- Beides sind Boolesche Funktionen mit je drei Eingängen
- Die technische Funktionseinheit dieser Funktionen wird Volladdierer (VA) (engl. *full adder* (FA)) genannt

Volladdierer

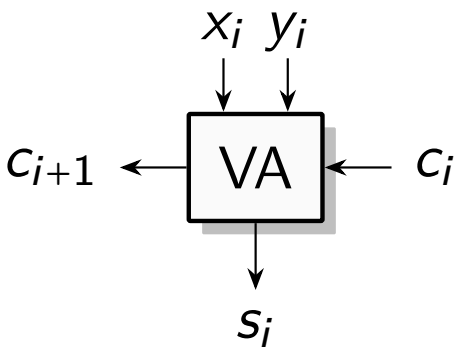
c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1





Volladdierer

Der Volladdierer (VA) abstrahiert die beiden Funktionen in einem Element



$$s_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

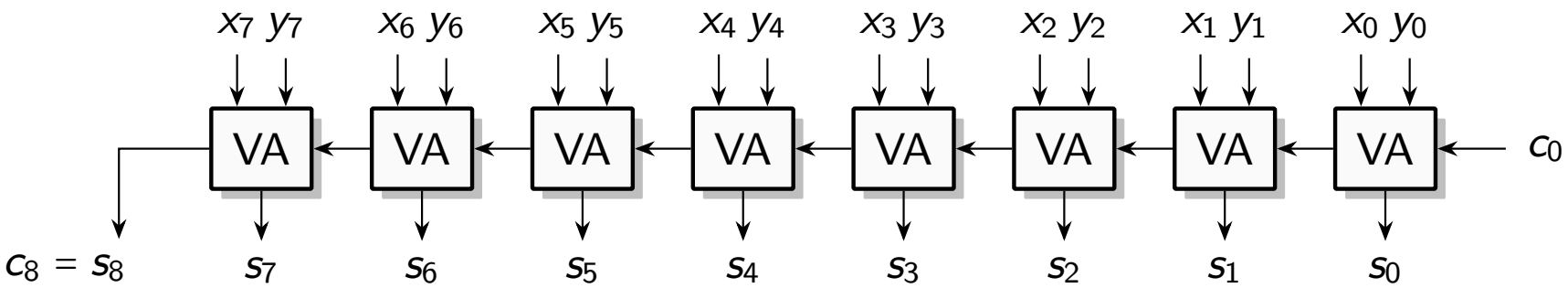
Ohne Carry-In ($c_i = 0$) wird aus dem VA ein Halbaddierer (HA):

$$s = x_i \oplus y_i \text{ und } c_o = x_i y_i.$$

Ripple-Carry Addierer

Ripple-Carry Addierer

- Durch die Weiterschaltung der Überträge zu höherwertigen VAs gelangt man zum Ripple-Carry-Addierer (RCA).
- Direkte Umsetzung der »Papier und Bleistift« Methode.
- Der VA für das LSB (rechts) kann durch einen HA ersetzt werden, wenn kein Carry-Eingang nötig ist.



Grundlegende Subtraktion

Wie funktioniert die Subtraktion?

Die Subtraktion wird über eine Addition mit dem Zweierkomplement realisiert: $D = X - Y = X + (-Y)$

Beispiel: $700_{10} - 82_{10}$ ($X = 01010111100_2$, $Y = 1010010_2$):

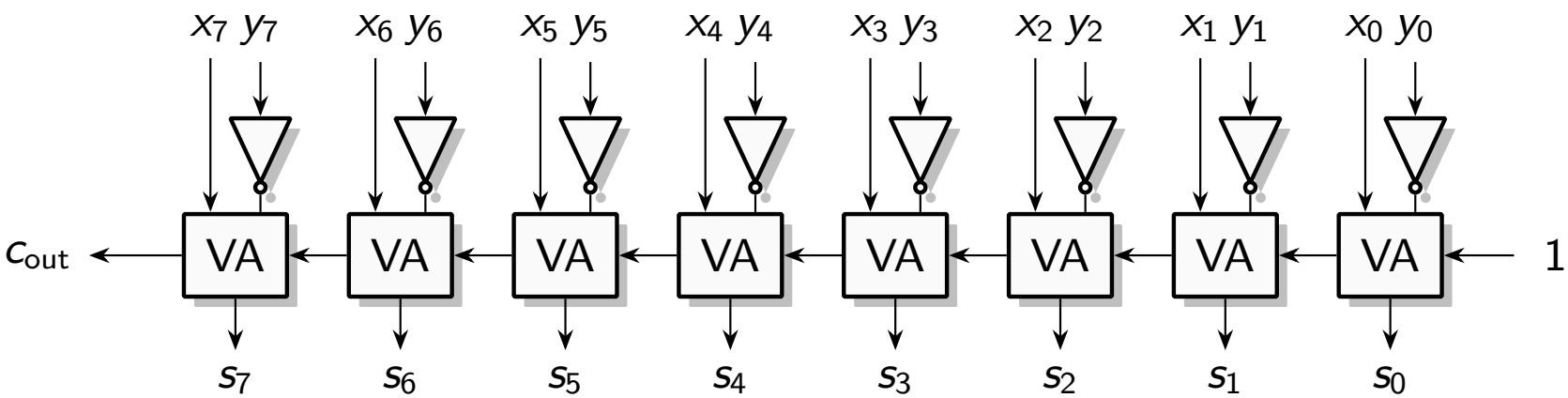
$$\begin{array}{rr} \overline{Y} & 11110101101 \\ +1 & 1 \\ \hline -Y = & 11110101110 \end{array}$$

$$\begin{array}{rr} X & 01010111100 \\ +(-Y) & +11110101110 \\ \hline = D & = 1\,|\,01001101010 \end{array}$$

- Überträge werden im Zweierkomplement ignoriert!
- Alle Zahlen (auch Ergebnis) müssen gleiche Wortbreite haben!
- Alle Zahlen (auch Ergebnis) müssen auch darstellbar sein!

Ripple-Carry Addierer/Subtrahierer

Für die Subtraktion wird das Komplement gebildet und über das Carry-In eine Eins hinzuaddiert.



Ein umschaltbarer Addierer/Subtrahierer lässt sich über XOR-Gatter realisieren (als steuerbarer Inverter).

Moderne Addition

Der Ripple Carry Addierer ist die einfachste Addiererschaltung, leider auch die langsamste.

Durch die Weiterreichung der Überträge an die nächste Stelle müssen in einem n-Bit Addierer insgesamt n Volladdierer durchlaufen werden.

In modernen Addierern wird daher versucht Überträge »vorausschauend« zu berechnen (sog. *Carry-Look-Ahead*)

⇒ Bei Interesse sehen wir uns wieder im Modul »Computerarithmetik« im Master AI! 🙂

Multiplikation mit binären Zahlen

Die Multiplikation wird auf bitweise Multiplikation mit anschließender bitverschobener Summation reduziert:

$$P = X \times Y = \underbrace{\sum_{i=0}^{n-1} x_i 2^i}_{=X} \times Y$$

Beispiel $n = 4$: $P = x_0 2^0 Y + x_1 2^1 Y + x_2 2^2 Y + x_3 2^3 Y$

Beispiel: $0111_2 \times 1101_2 = 7_{10} \times 13_{10} = 91_{10} (= 1011011_2)$

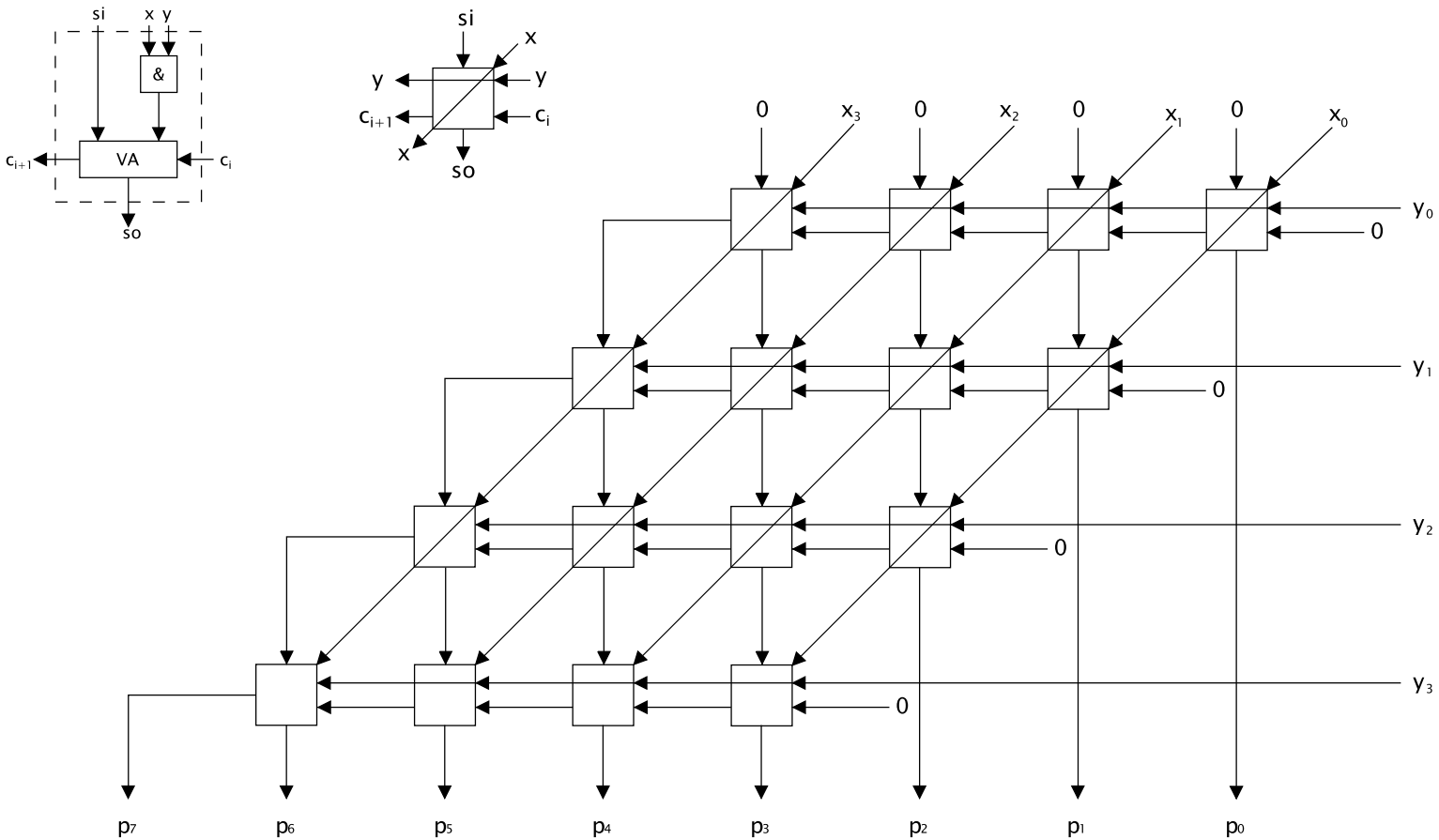
$$\begin{aligned} x_0 2^0 Y = 1 \times 2^0 \times 13_{10} = & \quad 1101_2 \\ x_1 2^1 Y = 1 \times 2^1 \times 13_{10} = & \quad +11010_2 \\ x_2 2^2 Y = 1 \times 2^2 \times 13_{10} = & \quad +110100_2 \\ x_3 2^3 Y = 0 \times 2^3 \times 13_{10} = & \quad +000000_2 \\ \hline & = 1011011_2 \end{aligned}$$

Vorlesungsaufgabe

Berechnen Sie das Ergebnis der Binären Multiplikation aus:

$9_{10} \times 5_{10} = 1001_2 \times 101_2 (= 45_{10} = 101101_2)$

Ripple Carry Array Multiplizierer



Moderne Multiplizierer

Der Ripple Carry Array Multiplizierer ist einer der einfachsten Multiplizierer aber leider auch der langsamste

Viele Logikstufen müssen hier verarbeitet werden um das Ergebnis zu erzeugen

In modernen Multiplizierern werden die Teilprodukte als auch deren Summation weitestgehend parallel berechnet

⇒ Bei Interesse sehen wir uns wieder im Modul »Computerarithmetik« im Master AI! 🙂

Arithmetisch-logische Einheit I

Die Arithmetisch-logische Einheit (engl. *Arithmetic Logic Unit*, ALU) ist die Recheneinheit einer CPU

Sie fasst die arithmetischen und logischen Operationen in einer Einheit zusammen

Arithmetische Operationen sind u.A.

- Addition, Subtraktion
- Vergleichsoperation
- Multiplikation
- Division

Logische Operationen sind u.A.

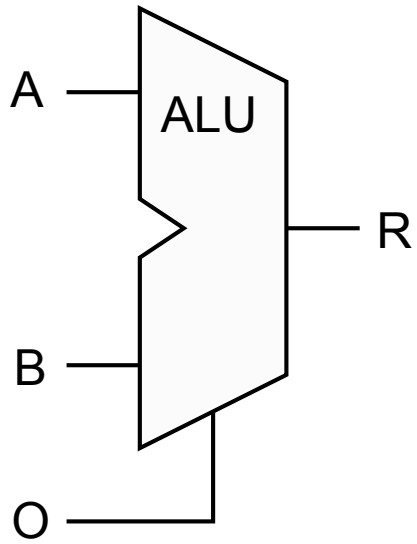
- UND, ODER, NICHT, XOR (Bitweise für ein ganzes Wort
- Bitverschiebungen nach rechts oder links

Arithmetisch-logische Einheit II

Die ALU ist rein kombinatorisch.

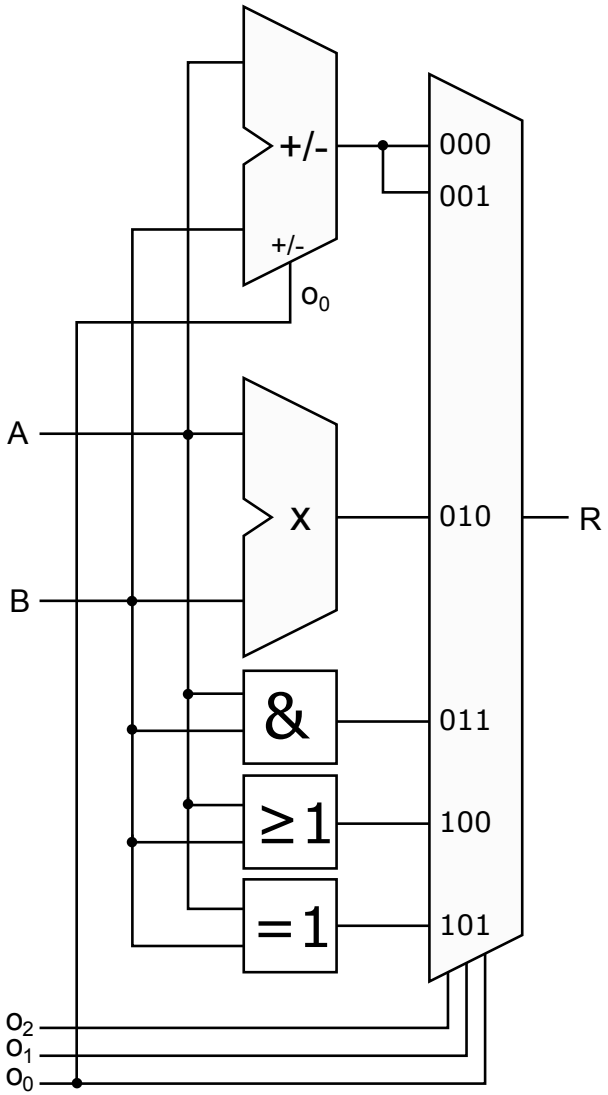
Die ALU hat i.d.R

- zwei Operanden-Worte (A und B, je n Bit)
- ein Ergebnis-Wort (R, n Bit)
- ein Steuereingang zur Auswahl der Operation (O, m Bit)



Ggf. existieren noch weitere Ein-/Ausgaben wie z.B. Überträge

Aufbau einer ALU



o_2	o_1	o_0	Operation	R
0	0	0	+	$A + B$
0	0	1	−	$A - B$
0	1	0	×	$A \times B$
0	1	1	UND	$A \wedge B$
1	0	0	ODER	$A \vee B$
1	0	1	XOR	$A \oplus B$
1	1	–	keine	–