

# Tut 14 (Tutor) 2nd Edition

Monday, 4. February 2019 20:59

## 0. ORGANISATORISCHES

### 3. Block-Test

• Zeit bis So. 10.02.19 23:59

### Klausur

27.02.19 (Mi) 08:00, Anmeldung  
10.02. - 20.02 über MOSES &&  
GISPOS ??

- Seit dem 04.02. (Mo) ist die Probeklausur verfügbar
- Bearbeiten bis nächste Woche (empfohlen), Q&A nächste Woche. Mehr auf Freitagrunde

## 1. DIE ALU

- Letzte Woche: Schaltungen konstruiert, um Aufgaben & Funktionen zu erfüllen - z.B. Primzahlerkennung
- Computerprozessor setzt sich aus Logikgatter zusammen
- Rechenwerk, Steuerwerk, Speicher...
- Rechenwerk (ALU) führt Berechnungen aus, u.a. Addition

### 1.1. Halbaddierer

- Wie sieht so ein Addierwerk aus?
- Addieren eines Bits

#### ALU: Halbaddierer

##### 1. 1 Bit Addition

0	0	1	1
+0	+1	+0	+1
0	1	1	10
			↑
			c s

##### 2. WT

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

##### 3. Log. Ausdrücke

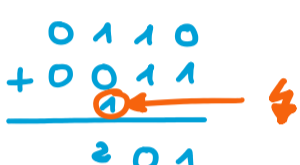
$$s = x \oplus y$$

$$c = x \cdot y$$

##### 4. Schaltplan



- Ein Halbaddierer setzt sich einfach aus einem XOR & AND zusammen!
- Mehrere Bits addieren



- Funktioniert nicht, weil wir nur zwei Eingangsbits haben
- Doof: wie kann das Carry in berücksichtigt werden?

### 1.2 Der Volladdierer

- Zielsetzung durch Wertetabelle

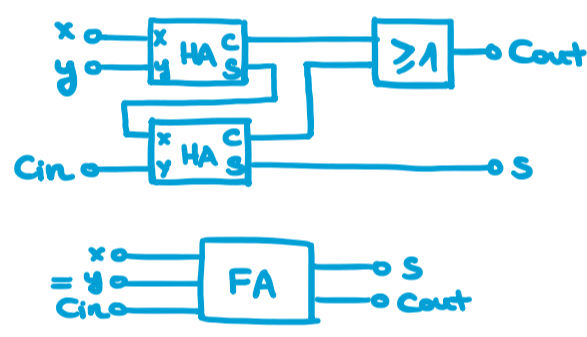
#### ALU: Volladdierer

- Ziel: Carry in berücksichtigen!

#### Wertetabelle

Cin	x	y	S	Cout
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

... lässt sich aus 2 HA und 1 OR bauen!



- Zusätzlicher Eintrag für den alten Übertrag (Carry-In)
- Wird aus zwei HA und einem OR-Gatter zusammengesetzt
- Logische Ausdrücke:
  - $s = (x \oplus y) \oplus Cin$
  - $Cout = x \cdot y + (Cin \cdot (x \oplus y))$

- Nun können wir zwei 4 Bit zahlen bzw. durch Kaskodierung von 4 Volladdierer addieren

- Bild: was ist C<sub>in</sub>?

→ = 0V, auf GND gelegt, da eh kein Carry in!

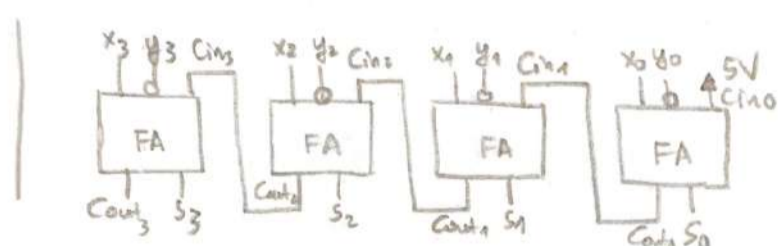
- Ende des reg. Stoffes

### 1.3 ALU-Entwurf

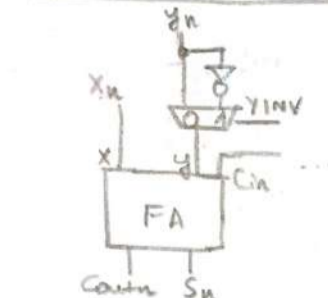
#### Erweiterung d. 4-Bit Addierers:

- Subtraktion  $x-y = x+(-y)$  in 2K
- (-y): invertieren und +1 addieren, um 2K-Darst. zu erhalten

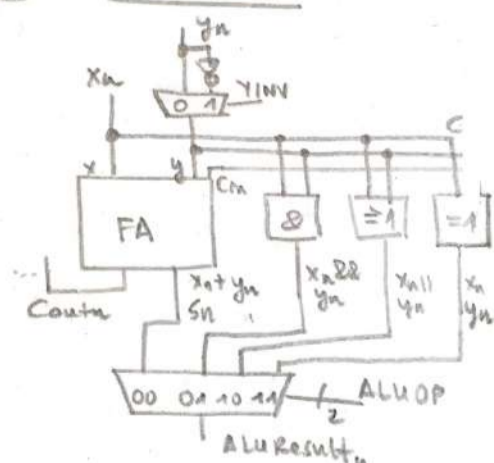
#### Subtrahierwerk, 4 Bit



#### Mod: Multiplexer

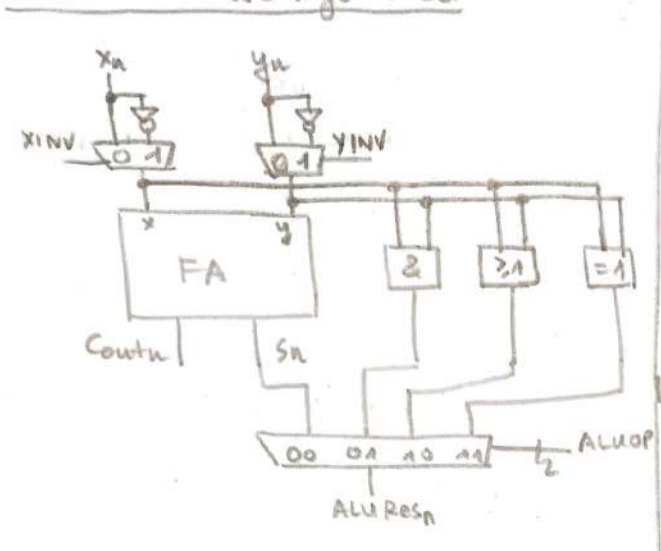


#### Mod: ALU



- ALU kann jetzt add, sub, and, or, xor. Fehlen noch nand, nor, xnor → XinV einführen!

#### Mod: vollwertige ALU



#### \* MULTI-PLEXER

d	x	y	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

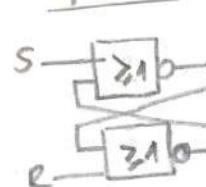
$$s = d \cdot y + \bar{d} \cdot x$$

- Selektive Datenleitung



- entweder x oder y wird durchgeleitet

#### Speicher



- RS-FF (lucks)
- D-FF: Speichert wenn Enable=1
- JK-FF: Im Verb. Zustand: T
- T-FF: Toggle bei 1 in

FF teuer & Stromhungrig; flüchtig

→ NPL: HDD, USB-Stick / SSD (Flash), Disketten

Hierarchie: Register → Cache (L1-L3), RAM, Latte CD, Diskette