

# Tutorat 10

Betrag Zweierkomplementzahl, Basiszelle ALU, NOR RS-Flipflop, D-Flip-Flop

Gruppe 9

---

*Präsentator:*  
Jürgen Mattheis  
([juergmatth@gmail.com](mailto:juergmatth@gmail.com))

*Vorlesung von:*  
Prof. Dr. Scholl

*Übungsgruppenbetreuung:*  
Tobias Seufert

*12. Juli 2023*

Universität Freiburg, Lehrstuhl für Rechnerarchitektur

# Gliederung

Aufgabe 1

Aufgabe 2

Aufgabe 3

Aufgabe 4

Appendix

# Aufgabe 1

# Aufgabe 1 I

## Betrag Zweierkomplementzahl

### Aufgabe 1.1

Entwickle einen Schaltkreis zu:

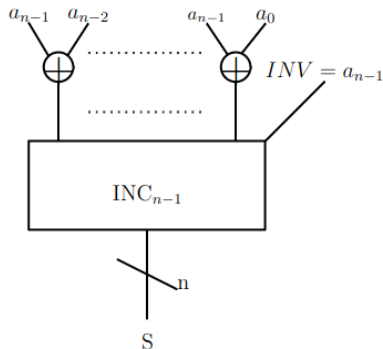
$$abs_n : \mathbb{B}^n \rightarrow \mathbb{B}^n, (a_{n-1}, \dots, a_0) \mapsto (s_{n-1}, \dots, s_0)$$

$$\langle s_{n-1}, \dots, s_0 \rangle = |[a_{n-1}, \dots, a_0]|$$

# Aufgabe 1 II

## Betrag Zweierkomplementzahl

### Lösung 1.1



# Aufgabe 1 III

## Betrag Zweierkomplementzahl

### Lösung 1.1

$$\begin{aligned} \text{cost}(abs_n) &= \text{cost}(INC_n - 1) + (n - 1) \cdot \text{cost}(XOR) = (n - 1) \cdot \text{cost}(HA) + (n - 1) \\ &= 3(n - 1) \end{aligned}$$

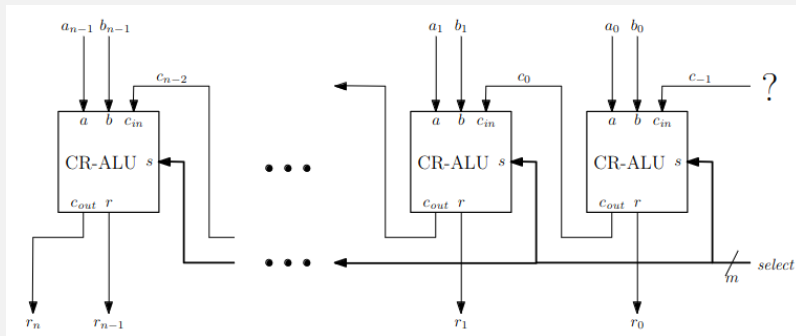
Es gibt **keinen** Überlauf!  $|[10\dots 0]| = \langle 10\dots 0 \rangle$

# Aufgabe 2

# Aufgabe 2 I

## Basiszelle Carry-Ripple-ALU

### Aufgabe 2.1

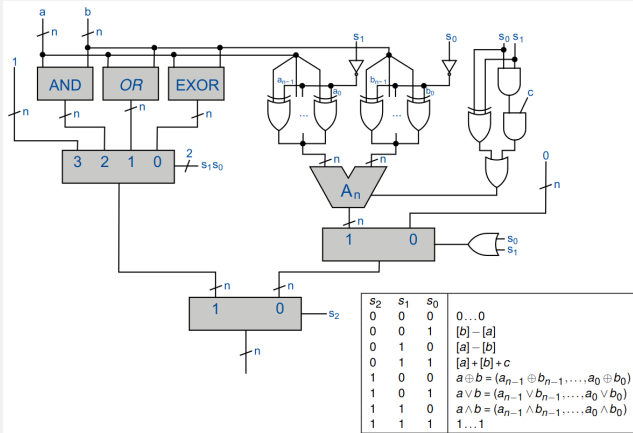




# Aufgabe 2 II

Regionale Campus Digital ALU

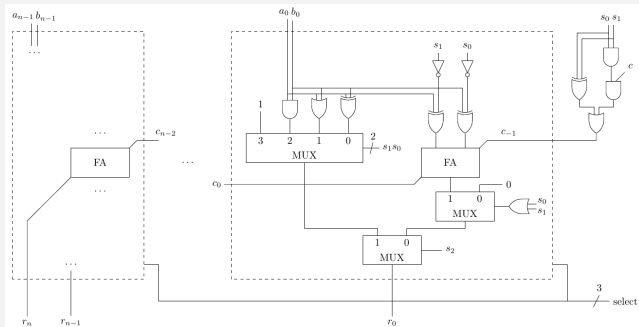
## Voraussetzungen 2.1



# Aufgabe 2 III

## Basiszelle Carry-Ripple-ALU

### Lösung 2.1



# Aufgabe 2 IV

## Basiszelle Carry-Ripple-ALU

### Lösung 2.1



*CR-ALU ist wie auf vorheriger Folie bei  $n = 1$  mit folgenden Änderungen:*

- ▶  *$A_n$  ist ein Volladdierer*
- ▶ *Es gibt einen weiteren Ausgang an  $A_n$ , um das Carry für die nächste Zelle zu übergeben*
- ▶  *$c$  wird direkt in den Volladdierer übergeben*
- ▶ *Die wegfallende Schaltung erzeugt  $c_{-1}$*

# Aufgabe 3

# Aufgabe 3 I

## NOR RS-Flipflop

### Lösung 3.1



► Es gibt *fünf stabile Belegungen*:

1.  $a = 0, b = 0, c = 0, d = 1$
2.  $a = 0, b = 0, c = 1, d = 0$
3.  $a = 0, b = 1, c = 1, d = 0$
4.  $a = 1, b = 0, c = 0, d = 1$
5.  $a = 1, b = 1, c = 0, d = 0$

### Lösung 3.2



- bei  $a = b = 0$  wird der aktuelle *Wert gehalten*
- bei  $a = 0, b = 1$  wird  $c$  auf 1 und  $d$  auf 0 gesetzt
- bei  $a = 1, b = 0$  wird  $c$  auf 0 und  $d$  auf 1 gesetzt

### Lösung 3.3



- $a$  und  $b$  sind *active-high*, da sie durch das Heben auf 1 aktiviert werden

# Aufgabe 3 II

## NOR RS-Flipflop

### Lösung 3.4



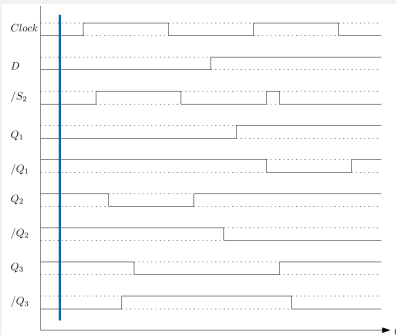
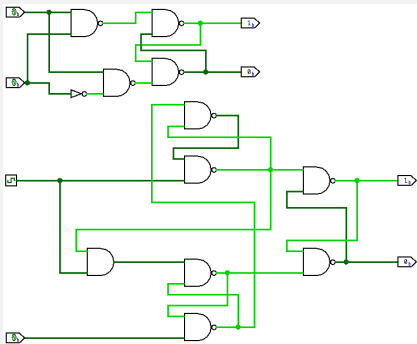
- ▶ die Belegung  $a = 1, b = 1$  ergibt keinen Sinn, da
  - ▶ es bei gleichzeitigem Absenken von  $a$  und  $b$  zu *Flackern* kommen kann
  - ▶ da es für die diese Eingangsbelegung *nur einen stabilen Zustand* gibt. Daher kann *nur ein Wert* „gespeichert“ werden

# Aufgabe 4

## Aufgabe 4

## D-Flip-Flop

## Lösung 4.1

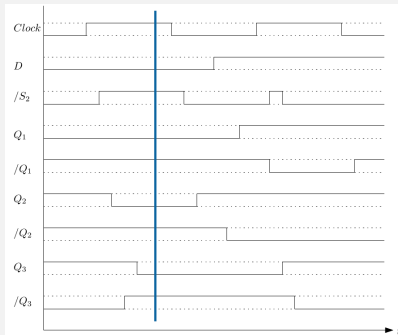
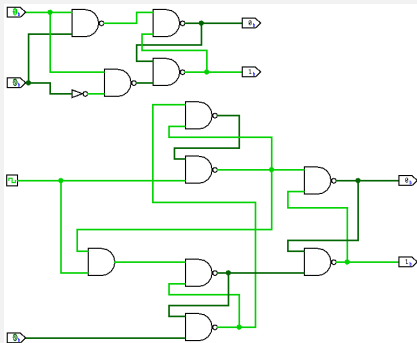




# Aufgabe 4

## D-Flip-Flop

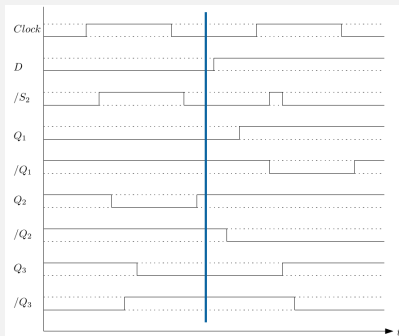
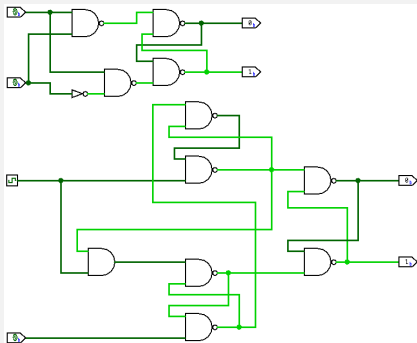
### Lösung 4.1



# Aufgabe 4

## D-Flip-Flop

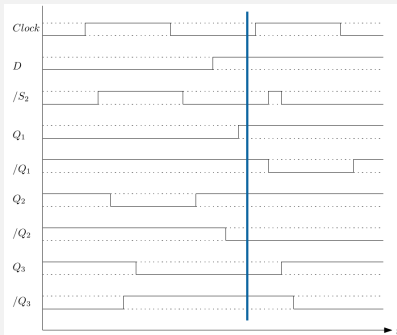
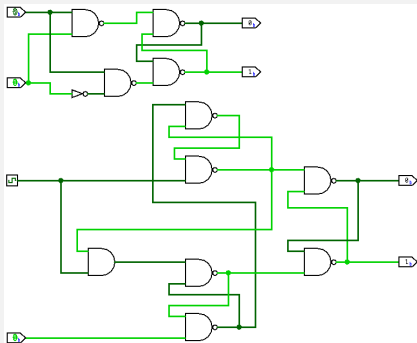
### Lösung 4.1



# Aufgabe 4

## D-Flip-Flop

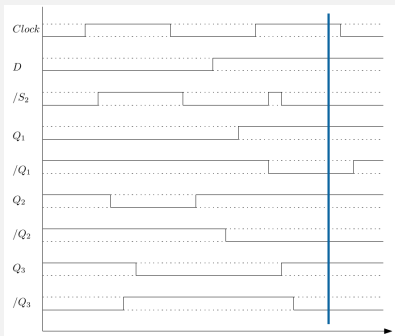
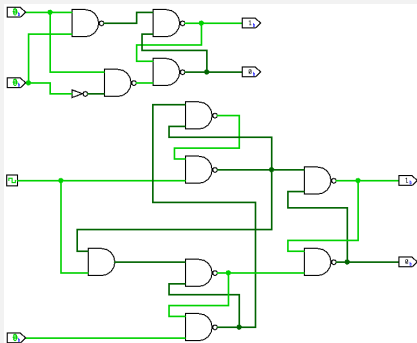
### Lösung 4.1



# Aufgabe 4

## D-Flip-Flop

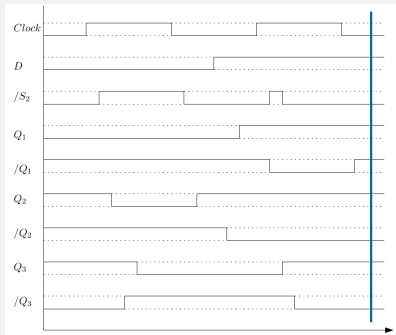
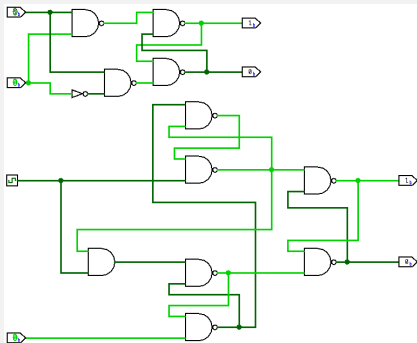
### Lösung 4.1



# Aufgabe 4

## D-Flip-Flop

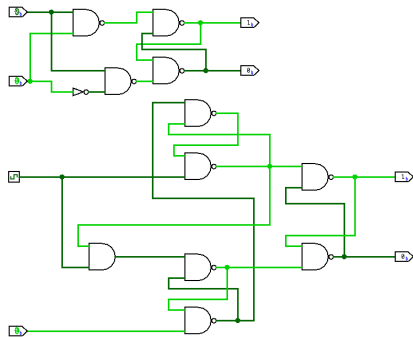
### Lösung 4.1



# Appendix

# Appendix

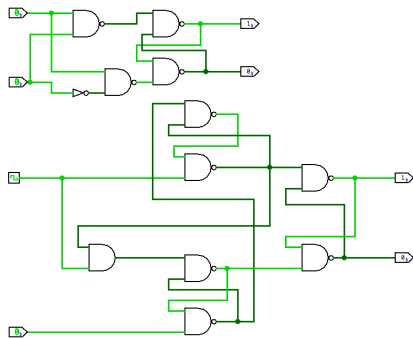
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ oberer RS-Flip-Flop ist im **Metastabilen Zustand** und hält dadurch den oberen Eingang des **rechten RS-Flip-Flop** auf 1
  - ▶ unterer RS-Flip-Flop ist im **Set Zustand** und hält dadurch den unteren Eingang des **rechten RS-Flip-Flop** auf 1
- ⇒ **rechter RS-Flip-Flop** ist dadurch im **Wert-Speichern Zustand**

# Appendix

## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop

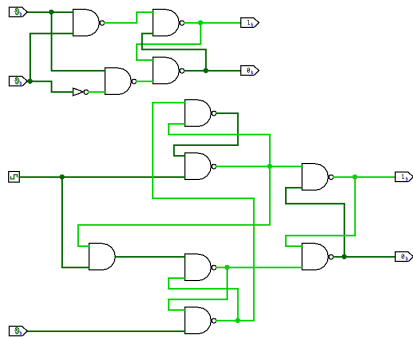


- ▶ bei  $d = 1$ 
    - ▶ oberer RS-Flip-Flop geht bei clock auf 1 in den Set Zustand über und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 0
    - ▶ unterer RS-Flip-Flop geht bei clock auf 1 in den Set Zustand über und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 1
- ⇒ rechter RS-Flip-Flop ist dadurch im Set Zustand



# Appendix

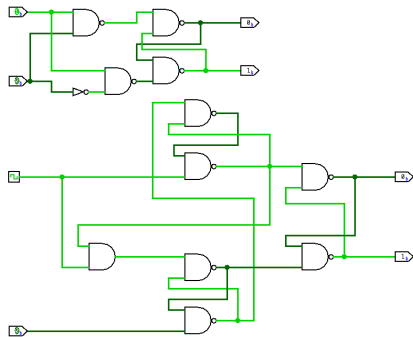
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ oberer RS-Flip-Flop ist im Reset Zustand und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 1
  - ▶ unterer RS-Flip-Flop ist im Metastabilen Zustand und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 1
- ⇒ rechter RS-Flip-Flop ist dadurch im Wert-Speichern Zustand

# Appendix

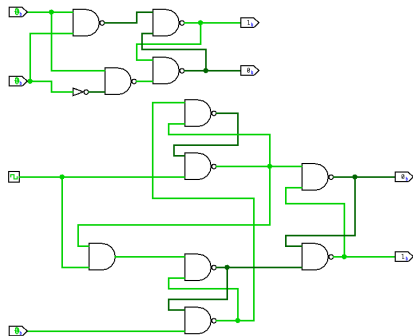
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ bei  $d = 0$ 
    - ▶ oberer RS-Flip-Flop geht bei clock auf 1 in den Wert-Speichern Zustand über und hält dadurch den oberen Eingang des rechten RS-Flip-Flop auf 1. Der obere RS-Flip-Flop muss vorher in am unteren Eingang den Wert 1 gehalten haben, da der obere RS-Flip-Flop wenn die Clock auf 0 ist dafür sorgen muss, dass der rechte RS-Flip-Flop im Wert-Speichern Zustand ist, indem er eine 1 hält
    - ▶ unterer RS-Flip-Flop geht bei clock auf 1 in den Reset Zustand über und hält dadurch den unteren Eingang des rechten RS-Flip-Flop auf 0
- ⇒ rechter RS-Flip-Flop ist dadurch im Reset Zustand

# Appendix

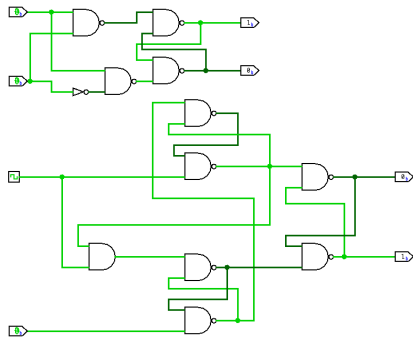
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ der **D-Flip-Flop** verhält sich gleich wie ein **D-Latch** (siehe Aufgabe 4) allerdings mit dem **Unterschied**, dass beim D-Flip-Flop der zu speichernde Wert in D beim Anheben der Clock von 0 auf 1 feststehen muss und im Nachhinein **nicht geändert** werden kann solange die clock 1 ist und auch nicht während die Clock 0 ist
  - ▶ das wird erreicht indem die Werte an den beiden Eingängen des rechten RS-Flip-Flop durch den oberen und unteren RS-Flip-Flop gehalten werden wenn die Clock 1 ist
  - ▶ und indem der Wert 1 an einem der beiden Eingänge des rechten RS-Flip-Flop durch den oberen oder unteren RS-Flip-Flop passend gehalten wird wenn die Clock 0 ist

# Appendix

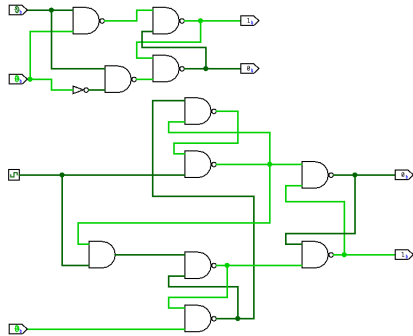
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ der **D-Flip-Flop** verhält sich gleich wie ein **D-Latch** (siehe Aufgabe 4) allerdings mit dem **Unterschied**, dass beim D-Flip-Flop der zu speichernde Wert in D beim Anheben der Clock von 0 auf 1 feststehen muss und im Nachhinein **nicht geändert** werden kann solange die clock 1 ist und auch nicht während die Clock 0 ist
  - ▶ das Ändern von **D** ändert die Zustände des oberen und unteren RS-Flip-Flop zwar, aber niemals so, dass sich dadurch die an den Eingängen des rechten RS-Flip-Flop gehaltenen Werte ändern
  - ▶ das Ändern von der **Clock** kann nur einen der beiden Eingänge des rechten RS-Flip-Flop zwischen 0 und 1 wechseln lassen

# Appendix

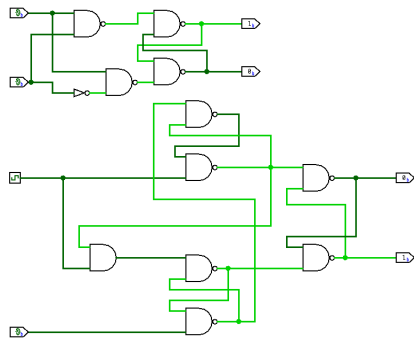
## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- ▶ beim D-Latch kann der gespeicherte Wert beliebig durch **Ändern** des Wertes an D geändert werden solange die **Clock 1** ist
  - ▶ nur während die **Clock 0** ist wird der Wert gehalten und kann **nicht geändert** werden
  - ▶ damit der D-Latch sinnvoll verwendet werden kann müsste der Wert von D die **ganze Zeit** über gehalten werden, während die Clock 1 ist + **Setup-Zeit** und **Hold-Zeit**, da jede Änderung während die **Clock 1** ist den gespeicherten Wert ändern würde
- ▶ beim D-Flip-Flop wird der gespeicherte Wert sowohl während die **Clock 1 oder 0** ist gehalten und kann nur geändert werden, wenn die Clock von 0 auf high wechselt
  - ▶ beim Wechsel der Clock auf 0 geht der **rechte RS-Flip-Flop** in den Zustand **Wert-Speichern** über

# Appendix

## Unterschiede und Gemeinsamkeiten bei D-Latch und D-Flip-Flop



- D sorgt dafür, dass der obere und untere RS-Flip-Flop jeweils im „richtigen“ Zustand sind, so dass ein Flankenwechsel der Clock auf 1 den rechten RS-Flip-Flop richtig einstellt