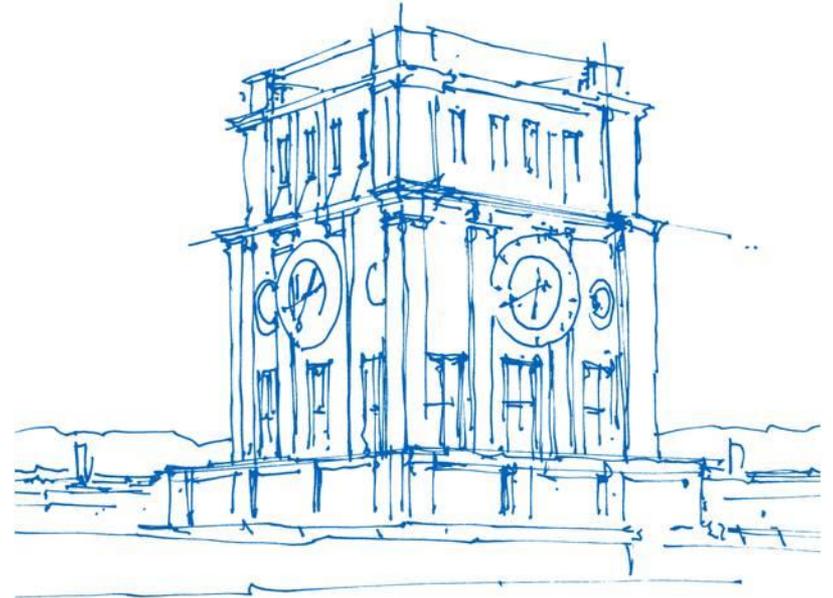


# Grundlagenpraktikum: Rechnerarchitektur

WiSe 2024/25

~ *Danial Arbabi*

[danial.arbabi@tum.de](mailto:danial.arbabi@tum.de)



*Uhrenturm der TUM*

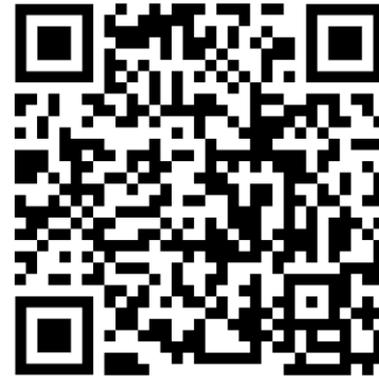
# Zulip-Gruppen

MI-1400-Z-RH



<https://zulip.in.tum.de/#narrow/stream/2619-GRA24W---Tutorium-Mi-1400-Z-RH>

MI-1600-L



<https://zulip.in.tum.de/#narrow/stream/2620-GRA24W---Tutorium-Mi-1600-L>

## Tutoriums-Website



<https://home.in.tum.de/~arb>

oder

<https://arb.tum.sexy>

*Disclaimer:*

*Dies sind keine offiziellen  
Materialien, somit besteht keine  
Garantie auf Korrektheit und  
Vollständigkeit.*

*Falls euch Fehler auffallen, bitte  
gerne melden.*

# Wiederholung

# Module in SystemC

Stichworte: Konstruktor, Sensitivity-Lists, behaviour

```
1 SC_MODULE(M1) {  
2     sc_signal<bool> x;  
3     sc_signal<bool> y;  
4     sc_signal<bool> output;  
5  
6     SC_CTOR(M1) {  
7         SC_THREAD(behaviour);  
8         sensitive << x << y;  
9     }  
10  
11     void behaviour() {  
12         while (true) {  
13             output = x.read() | (!x.read() & y.read());  
14             wait();  
15         }  
16     }  
17 };
```

# Module in SystemC

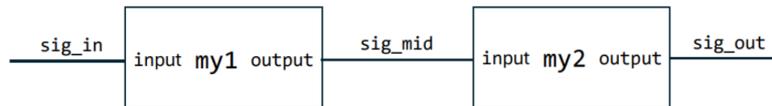
Stichworte: Ports (input, output)

```
1 SC_MODULE(MyModule) {  
2     sc_in<bool> input;  
3     sc_out<bool> output;  
4  
5     SC_CTOR(MyModule) {  
6         SC_THREAD(behaviour);  
7     }  
8  
9     void behaviour() {  
10        while (true) {  
11            output->write(!input->read());  
12            wait();  
13        }  
14    }  
15 };
```

- Lesen vom Input
- Schreiben auf den Output

# Module in SystemC

## Kommunikation zwischen Modulen (Kabel verbinden)



```
1 MyModule my1("my1");
2 MyModule my2("my2");
3 sc_signal<bool> sig_in, sig_mid, sig_out;
4 // Die Tatsächlichen Signale werden außerhalb der Module
  erstellt.
5
6 my1.input.bind(sig_in);
7 my1.output.bind(sig_mid);
8 // port.bind(signal) weist einem Port ein Signal zu.
9
10 my2.input(sig_mid);
11 my2.output(sig_out);
12 // Alternative Schreibweise: port(signal).
```

- Schreiben / Lesen von Signalen
- Verbinden dieser Signale mit den Ports

# XOR GATTER

## Tutoriumsaufgabe T6-2

1. Implementiere ein Modul für ein XOR Gatter mit dem Namen XOR\_GATE und einem einfachen Konstruktor.
2. Das Modul soll eine kombinatorische Schaltung darstellen, zwei Signale (a, b) für den Input verwenden und ein Signal (out) für den Output.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

# RISC-V Multiplexer

## Tutoriumsaufgabe T6-3

1. Zwei Aufgabe:
  1. MULTIPLEXER\_BOOLEAN
  2. MULTIPLEXER\_I32
2. Was ist ein Multiplexer?
  1. Wie ein Switch-Case Statement
  2. Entscheidet welches Eingangssignal raus kommt

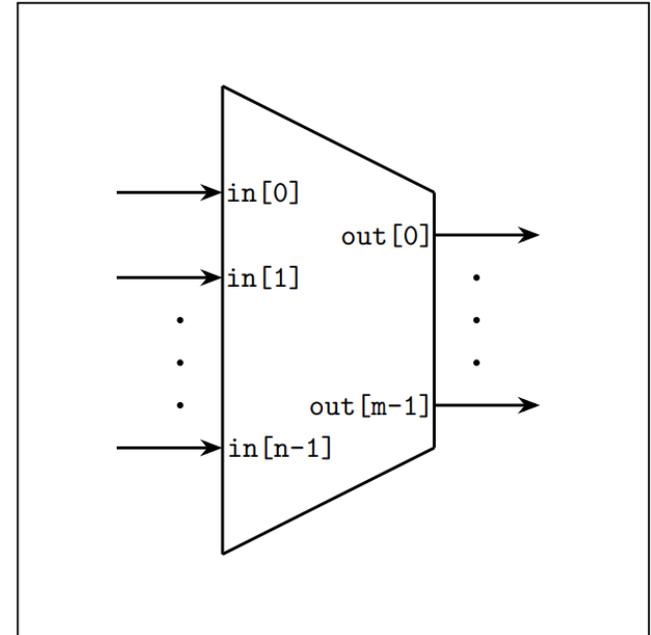
### SPEZIFIKATION: MULTIPLEXER\_BOOLEAN

#### *Inputs*

- `in`: vector of boolean flags
- `select`: 8-bit register

#### *Outputs*

- `out`: vector of boolean flags



# RISC-V Multiplexer

## Tutoriumsaufgabe T6-3

1. select: 8-bit unsigned Integer
  1. Falls *select* größer als #Inputs → alle Outputs 0
  2. Sonst entscheidet select welches Inputsignal an die Outputs kommt
2. Definiere eigenen Konstruktor:
  1. Neue Params: *int fanIn*, *int fanOut*  
→ Bestimmen #Input/Output Ports

**Erlaubte *Magie*:** Input wählen, Outputs kopieren, Vector-Größe prüfen.

# RISC-V Multiplexer

## Tutoriumsaufgabe T6-3

### SPEZIFIKATION: MULTIPLEXER\_I32

---

#### *Inputs*

- `in`: vector of 32-bit values
- `select`: 8-bit register

#### *Outputs*

- `out`: vector of 32-bit values