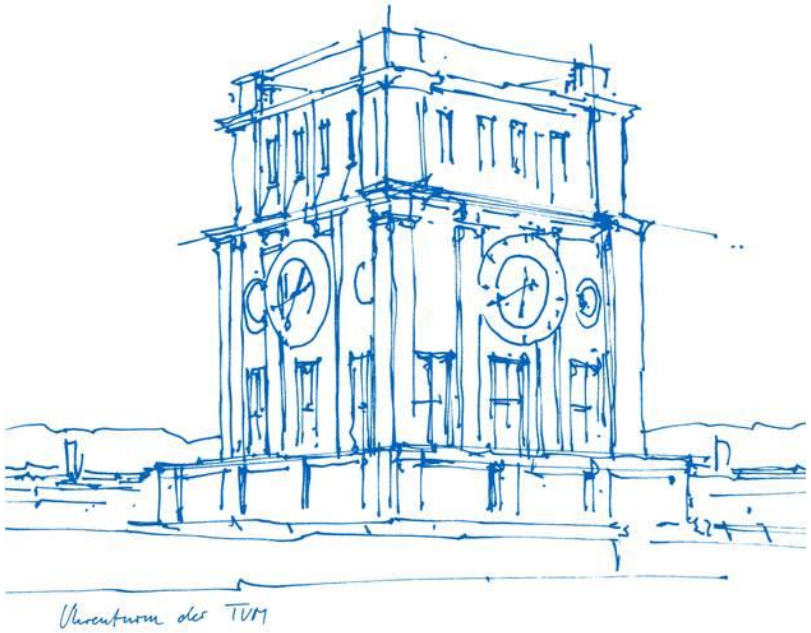
School of Computation, Information and Technology
Technische Universität München

Grundlagenpraktikum: Rechnerarchitektur

SoSe 2025

~ *Danial Arbabi*
danial.arbabi@tum.de



Zulip-Gruppen

GRP 01: Montag 10:00

MI 03.13.10

#GRA/S - Tutorial-GRP-01

GRP 03: Montag 14:00

MI 01.06.20

#GRA/S - Tutorial-GRP-03

Tutoriums-Website

https://home.cit.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.

Organisatorisches

- Nächste Woche letzte Tutoraufgabe + Fragestunde
- Hausaufgaben und Übungen machen
- Mit Teampartnerinnen und Teampartnern vor Projektstart schonmal treffen
- Fragen JETZT stellen

Trace File

- *Trace*: Gibt an, zu welchem Zeitpunkt sich der Wert eines Signals ändert.
- *Trace File*: Sammlung von Traces verschiedener Signale.

```
1  int sc_main(int argc, char* argv[]) {
2      sc_signal<bool> my_signal;
3      sc_signal<int> my_count;
4      ...
5
6      sc_trace_file* file = sc_create_vcd_trace_file("trace");
7      sc_trace(file, my_signal, "my_signal");
8      sc_trace(file, my_count, "my_count");
9      sc_start(10, SC_SEC);
10     sc_close_vcd_trace_file(file);
11
12     ...
13 }
```

Trace File

Anleitung und Beispiel

1. Erstellen der Tracefile (`Pfad bzw. Name vom User über Commandline übergeben`) und zuweisen auf eine Pointervariable `trace_file`

```
sc_trace_file* trace_file = sc_create_vcd_trace_file(argv[1]);
```

2. Beobachten einer Variablen `signal` in `trace_file` und nenne sie „signal“

```
sc_trace(trace_file, signal, „signal“);
```

3. Starten der Simulation für `time` Sekunden

```
sc_start(time, SC_SEC);
```

4. Schließen der `trace_file`

```
sc_close_vcd_trace_file(trace_file);
```

Trace File

Aufbau

- *Value Change Dump*: Format für Traces, häufig bei Systemdesign Tools.

```
$date                                  // Erstellungsdatum des Traces
     May 31, 2024        18:00:00
$end

$version                             // Version und Datum von SystemC
 SystemC 2.3.4-Accellera --- May 11 2024 10:27:04
$end

$timescale                 // Zeiteinheit aller Zeitangaben im Trace
     1 ps
$end
```

Trace File

Aufbau

- *Value Change Dump*: Format für Traces, häufig bei Systemdesign Tools.

```
$scope module SystemC $end
$var wire    1  aaaaa  my_signal       $end
  // 1 Bit, Name "my_signal", ID "aaaaa"
$var wire   32  aaaab  my_count [31:0]  $end
  // 32 Bit, Name "my_count", ID "aaaab"
$upscope $end
$enddefinitions  $end

$comment
All initial values are dumped below at time 0 sec = 0 timescale
    units.
$end
```

Trace File

Aufbau

- *Value Change Dump*: Format für Traces, häufig bei Systemdesign Tools.

```
1 $dumpvars                                  // Startwerte aller Signale
2 1aaaaa                                        // aaaaa (my_signal) = 1
3 b0 aaaab                                  // aaaab (my_count) = b0 = 0
4 $end
```

Trace File

Aufbau

- *Value Change Dump*: Format für Traces, häufig bei Systemdesign Tools.

```
#2000000000000                          // Zeitpunkt 2 Sekunden
0aaaaa                                  // aaaaa (my_signal) = 0
b1 aaaab                          // aaaab (my_count) = b1 = 1
```

```
#4000000000000                          // Zeitpunkt 4 Sekunden
1aaaaa                                  // aaaaa (my_signal) = 1
b10 aaaab                        // aaaab (my_count) = b10 = 2
```

```
#6000000000000                          // Zeitpunkt 6 Sekunden
b11 aaaab                        // aaaab (my_count) = b11 = 3
                          // my_signal hat sich nicht verändert
```

Trace File

Bedenken

- Performance-Analyse durch Messung von Delays: genau genug?
- Es gibt weiterhin "real-world" Faktoren, die Performance beeinflussen können
  - *Signal I/O*: Jeder *read/write* benötigt gewisse Zeit.
  - *Taktfrequenz*: Taktfrequenzen von 1 ps sind selten möglich. Overhead kann uns zwingen, auf tiefere Frequenzen auszuweichen.
  - *Abstraktion & Rechenzeit*: Manche Operationen sind in C/C++ nur eine Zeile Code, aber im Schaltkreis sehr komplex. Kann viel Overhead bringen.
- SystemC bietet uns Möglichkeit zur Abstraktion von Systemdesign.
  - Hilft uns, Schaltkreise zu entwerfen.
  - Macht aber genaue Analysen schwieriger.

Trace File

I/O-Analyse

```
1  SC_MODULE(MY_MODULE) {
2      int reads;
3      int writes;
4      ...
5
6      void behaviour() {
7          while (true) {
8              wait();
9              bool result = a->read() ^ b->read();
10             reads += 2;
11             out->write(result);
12             writes++;
13         }
14     }
15 };
```

Trace File

I/O-Analyse

- Resultate der Zählung bieten nützliche Statistiken:
  - Gesamtzahl I/O Operationen
  - I/O Operationen pro Sekunde
  - I/O Operationen pro Clock Zyklus
- Bieten Möglichkeit zum Vergleich verschiedener Lösungen

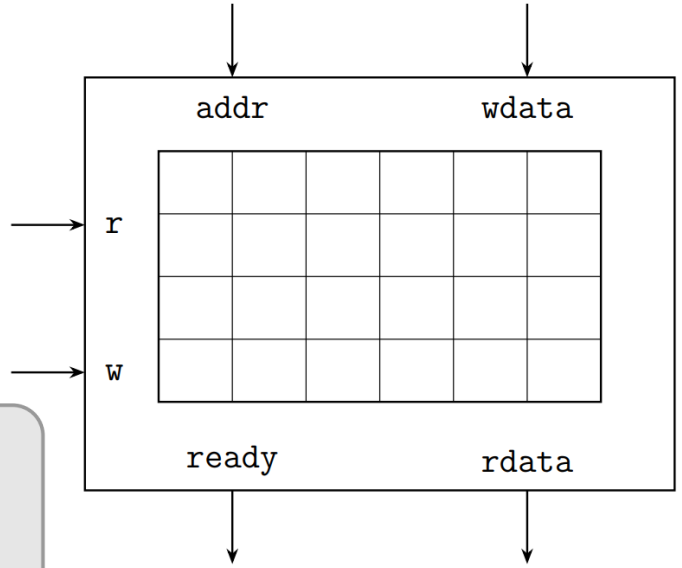
- Aber: Wie genau ist Resultat?
  - Read/Write dauert nicht unbedingt immer gleich lang.
  - Abstrahierte Teile eines Moduls können in "real-world" auch Reads/Writes beinhalten.
  - I/O Kosten hängen auch von Signaltyp ab: `sc_signal<bool>` benötigt weniger Writes/Reads als `sc_signal<int>`.
  - Umgekehrt: 10-mal so viele Writes/Reads bedeutet nicht gleich 10-mal langsamer. I/O kann auch parallel stattfinden.

RISC-V Main Memory

T 8.2

1. Welchen Datentypen zum Speichern verwenden?
2. Wie setzen wir das Lesen um?
3. Wie setzen wir das Schreiben um?

Tipp: Verwende Hilfsfunktionen!



SPEZIFIKATION

Inputs	*Outputs*
☐ clk: bool (clock input)	☐ ready: bool
☐ addr: uint32_t	☐ rdata: uint32_t
☐ wdata: uint32_t	
☐ r: bool	
☐ w: bool	