

# FAKULTÄT FÜR INFORMATIK

## Computer?

Das passiert in der Hardware,  
wenn ich einen Befehl eingebe.

## Was werden wir heute machen?

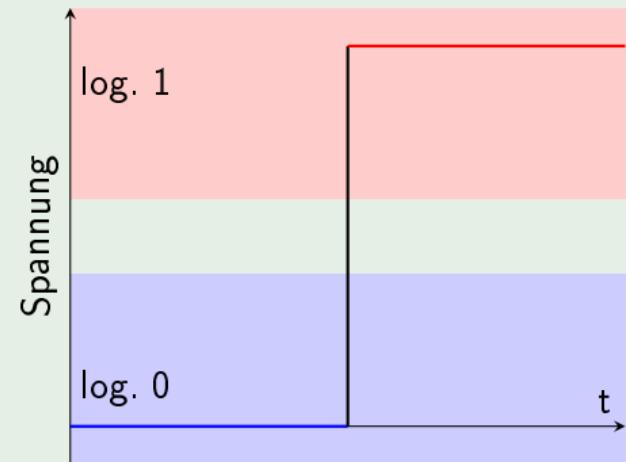
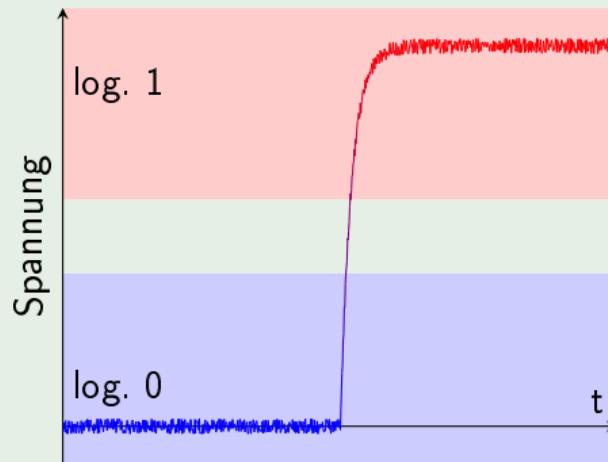
- ▶ Einen Einblick in eines der wichtigsten Werkzeuge eines Technikers geben: die Abstraktion.
- ▶ Dabei anhand von Beispielen aufzeigen, dass Abstraktionen Grenzen haben und spannendes Verhalten verstecken kann.
- ▶ Einen ersten Eindruck in das Entstehen eines Computerchips und die dabei auftretenden Probleme geben.

## Abstraktionen sind überall!

- ▶ Eine Abstraktion ist ein Modell.
- ▶ Dieses Modell versteckt komplexes Verhalten durch Weglassen von Informationen/Details.
- ▶ Dadurch ist ein Sachverhalt leichter zu verstehen, als durch die Summe der zugrundeliegenden Modelle.
- ▶ Abstraktionen haben aber auch ihre Grenzen!

Was bedeutet logisch 1 bzw. logisch 0?

Abstraktion des gemessenen Signals



## Handy

Wir können damit Anrufe t tigen und vieles mehr.

- ▶ Von der 趕bertragung der Daten zwischen Handy und Mast bemerken wir nichts. Allerdings ist hier komplizierte Hochfrequenztechnik im Einsatz.
- ▶ Die 趕bertragung der Sprache passiert einfach. Warum und wie ist beim Telefonieren eigentlich egal, oder?
- ▶ Wie das Handy eine Verbindung zu einem Gesprchsteilnehmer aufbaut ist uns als Anwender auch egal: Wir whlen nur eine Nummer.
- ▶ Wie reagiert das Handy auf unsere Eingabe? Was passiert bei einem Tastendruck?

## Handy

Wir können damit Anrufe t tigen und vieles mehr.

- ▶ Von der 趕bertragung der Daten zwischen Handy und Mast bemerken wir nichts. Allerdings ist hier komplizierte Hochfrequenztechnik im Einsatz.
- ▶ Die 趕bertragung der Sprache passiert einfach. Warum und wie ist beim Telefonieren eigentlich egal, oder?
- ▶ Wie das Handy eine Verbindung zu einem Gesprchsteilnehmer aufbaut ist uns als Anwender auch egal: Wir whlen nur eine Nummer.
- ▶ Wie reagiert das Handy auf unsere Eingabe? Was passiert bei einem Tastendruck?

Wir sehen schon mehrere Abstraktionen:

- ▶ Was ist die “bermittelte Sprache” eigentlich genau?
- ▶ Wie/wovon wird das Handy gesteuert?

Die Frage ist doch:

Muss man wissen was in der Hardware passiert?

Machen wir ein kleines Hardwareprojekt und schauen nach!

Dabei werden wir

- ▶ sehen dass doch viel mehr hinter einem kleinen Gadget steckt als man denken mag.
- ▶ einen ersten Eindruck gewinnen, wie ein Computer aufgebaut ist und was denn so alles passiert wenn man damit interagiert.
- ▶ uns punktuell anschauen welche Kompetenzen man im Informatikstudium, am Beispiel der Technischen Informatik, erwirbt.

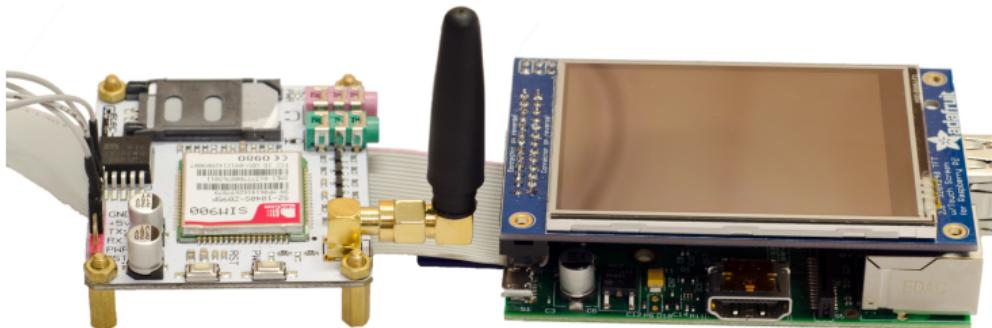


Abbildung: Ein Handy?

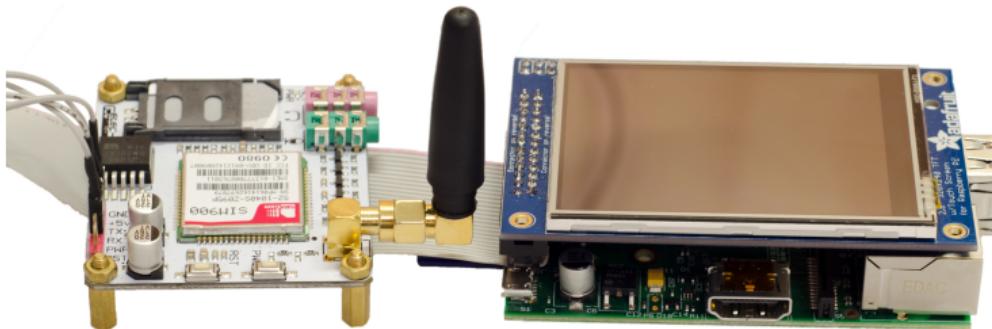


Abbildung: Ein Handy?

Was brauchen wir dafür?



Abbildung: PiTFT by Adafruit CC BY-SA 3.0

- ▶ 2.8" TFT + resistiver Touchscreen.
- ▶ 320×240 Pixel mit je 16 Bit Farbinformation.
- ▶ ansprechbar über SPI Interface.

- ▶ Jedes Computersystem hat in irgendeiner Form eine Interaktion mit dem Benutzer.
- ▶ Das kann im einfachsten Fall Taster + LED sein.
- ▶ In unserem Fall ist ein Touchscreen im Einsatz.

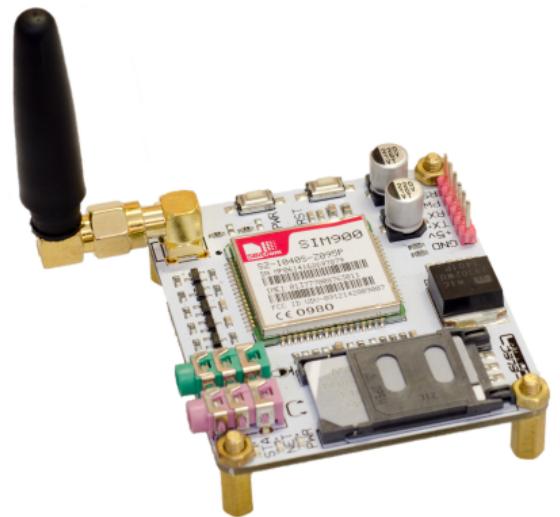


Abbildung: EFCom Pro GPRS/GSM Module

- ▶ SIM900 Quad-Band GSM/GPRS Module.
- ▶ ansprechbar über serielle Schnittstelle und AT-Commands.

Computer kommunizieren und interagieren typischerweise

- ▶ mit dem Benutzer  
per “User Interface”
- ▶ *mit anderen Computern*  
“Vernetzung”, “verteilte Systeme”
- ▶ mit der Umwelt  
per Sensor/Aktor, “Embedded Systems”

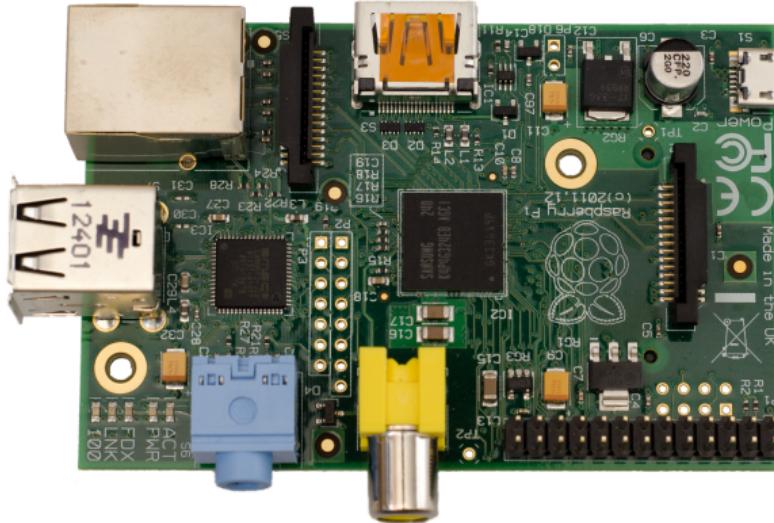


Abbildung: Raspberry Pi B

- ▶ Broadcom BCM2835 @ 700 MHz ARM11 core, 256 MB RAM.
- ▶ 26 Pin Header für Erweiterungen ...
- ▶ bietet SPI und serielle Schnittstelle.

- ▶ Macht die “Intelligenz” im System aus.
- ▶ Ermöglicht die Programmierung entsprechend der Anwendung (Handy, Alarmmelder, Anrufbeantworter, . . . ).
- ▶ Führt passende Ansteuerung von Touchscreen und Modem durch.
- ▶ Macht aus der Summe der Teile ein Handy.

Jeder Computer braucht eine Energiequelle;  
in unserem Fall soll sie portabel sein.



Abbildung: Akkupack

- ▶ Akkupack mit integrierter Ladeelektronik.
- ▶ 2200 mAh

| Preis (€) | Komponente            |
|-----------|-----------------------|
| 35        | Raspberry Pi          |
| 35        | Adafruit PiTFT        |
| 40        | EFCom GPRS/GSM Module |
| 15        | Akku                  |
| 5         | Kleinteile            |
| 130       | Gesamt                |

## Live-Demo

## Quiz:

Wie nennt man die kleinste Informationseinheit, die ein Computer verarbeitet?

1. PIN
2. BIT
3. LED
4. PUK

## Quiz:

Wie nennt man die kleinste Informationseinheit, die ein Computer verarbeitet?

1. PIN
2. BIT
3. LED
4. PUK

0681 108 23 819

... den Rechner?

- ▶ Ein Rechner umfasst Prozessor, Speicher und Peripherie.
- ▶ Der Prozessor versteht Befehle und kann daher programmiert werden.
- ▶ Nicht jeder Prozessor ist gleich, je nach Hersteller versteht er andere Befehle.
- ▶ Wie der Prozessor auf Hardware Ebene genau funktioniert, werden wir uns morgen genauer ansehen.
- ▶ Für unser Projekt ist uns das egal, durch die Verwendung einer Programmiersprache abstrahieren wir das alles weg und arbeiten prozessorunabhängig (soweit es geht).

... den Rechner?

- ▶ Ein Rechner umfasst Prozessor, Speicher und Peripherie.
- ▶ Der Prozessor versteht Befehle und kann daher programmiert werden.
- ▶ Nicht jeder Prozessor ist gleich, je nach Hersteller versteht er andere Befehle.
- ▶ Wie der Prozessor auf Hardware Ebene genau funktioniert, werden wir uns morgen genauer ansehen.
- ▶ Für unser Projekt ist uns das egal, durch die Verwendung einer **Programmiersprache** abstrahieren wir das alles weg und **arbeiten prozessorunabhängig** (soweit es geht).

## . . . den Touchscreen?

- ▶ Wie wird der Druck des Fingers erkannt?
- ▶ Durch Änderung der Kapazität an einer bestimmten Koordinate
- ▶ oder durch einen Spannungsabfall.
- ▶ Aber: Wie genau, ist uns eigentlich egal (Abstraktion!).
- ▶ Wir bekommen nur die Information wo gedrückt wurde.

## . . . den Touchscreen?

- ▶ Wie wird der Druck des Fingers erkannt?
- ▶ Durch Änderung der Kapazität an einer bestimmten Koordinate
- ▶ oder durch einen Spannungsabfall.
- ▶ Aber: Wie genau, ist uns eigentlich egal (Abstraktion!).
- ▶ **Wir bekommen nur die Information wo gedrückt wurde.**

## . . . das Display?

- ▶ Wie kommt ein Zeichen auf das Display?
- ▶ Spannung steuert, wie ein Flüssigkristall die Polarisationsrichtung von Licht beeinflusst.
- ▶ Wir sagen nur, was wo am Display sein soll.

## . . . das Display?

- ▶ Wie kommt ein Zeichen auf das Display?
- ▶ Spannung steuert, wie ein Flüssigkristall die Polarisationsrichtung von Licht beeinflusst.
- ▶ Wir sagen nur, was wo am Display sein soll.

## . . . das Modem?

- ▶ Wie funktioniert eigentlich die Funkübertragung?
- ▶ Über kontrollierte (modulierte) elektromagnetische Wellenausbreitung.
- ▶ Wie wandelt das Modem digitale Daten in elektromagnetische Wellen um?
- ▶ Wieder abstrahieren wir diese “Details” weg: wir übergeben dem Modem Daten zum Senden.

## . . . das Modem?

- ▶ Wie funktioniert eigentlich die Funkübertragung?
- ▶ Über kontrollierte (modulierte) elektromagnetische Wellenausbreitung.
- ▶ Wie wandelt das Modem digitale Daten in elektromagnetische Wellen um?
- ▶ Wieder abstrahieren wir diese “Details” weg: **wir übergeben dem Modem Daten zum Senden.**

## . . . das Audiointerface?

- ▶ Wie kommt die Sprache eigentlich von A nach B?
- ▶ Ein Mikrofon wandelt die Schallwellen in ein elektrisches Signal um.
- ▶ Und wie erhalten wir aus diesem elektrischen Signal einen Datenwert, der weiter verarbeitet werden kann?
- ▶ Dazu gibt es eigene Analog/Digital Konverter, die nach verschiedenen Prinzipien arbeiten.
- ▶ Doch auch diese Details nehmen uns darunter liegende Ebenen (HW Module, Treiber) ab, wir betrachten die Signale einfach als digitale Daten, die wir verarbeiten.

## . . . das Audiointerface?

- ▶ Wie kommt die Sprache eigentlich von A nach B?
- ▶ Ein Mikrofon wandelt die Schallwellen in ein elektrisches Signal um.
- ▶ Und wie erhalten wir aus diesem elektrischen Signal einen Datenwert, der weiter verarbeitet werden kann?
- ▶ Dazu gibt es eigene Analog/Digital Konverter, die nach verschiedenen Prinzipien arbeiten.
- ▶ Doch auch diese Details nehmen uns darunter liegende Ebenen (HW Module, Treiber) ab, **wir betrachten die Signale einfach als digitale Daten, die wir verarbeiten.**

# Der Zauber der Abstraktion

Wir konzentrieren uns auf das Wesentliche, lassen Details weg.  
Das erlaubt uns, effizient zu sein, den Überblick zu behalten.

*Wie produktiv wäre ein Programmierer, müsste er sich stets über den Verlauf der elektrischen Felder in den Transistoren der Ziel-Hardware Gedanken machen?*

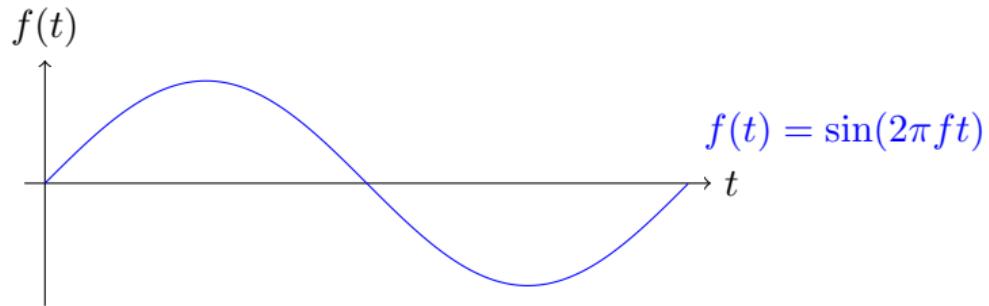
Wir konnten mit fertigen Komponenten und deren abstrakter Funktionsbeschreibung immerhin ein ganzes Handy zusammenbauen!

# Wozu also ein Studium?

Abstraktes verstehen!

## Das Audiointerface liefert digitale Daten

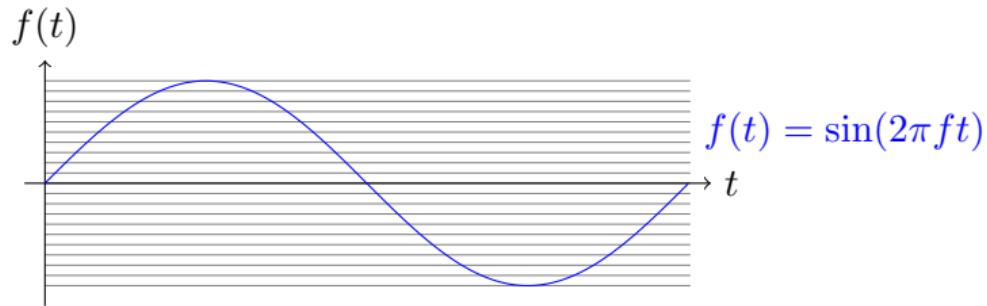
- ▶ Was steht eigentlich hinter dieser Abstraktion?
- ▶ Gibt es da auch Grenzen und Probleme?
- ▶ Sollte man da auch mehr wissen?



Um das Signal digital zu verarbeiten, müssen wir

- ▶ Amplitude
- ▶ und Zeit diskretisieren (“Abtastpunkte” statt Verlauf)

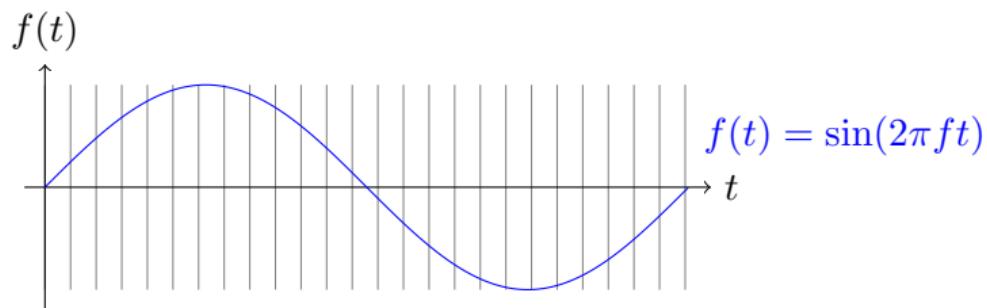
Wie viele Abtastpunkte ( $\Rightarrow$  Speicherplatz) brauchen wir?



Um das Signal digital zu verarbeiten, müssen wir

- ▶ Amplitude
- ▶ und Zeit diskretisieren (“Abtastpunkte” statt Verlauf)

Wie viele Abtastpunkte ( $\Rightarrow$  Speicherplatz) brauchen wir?

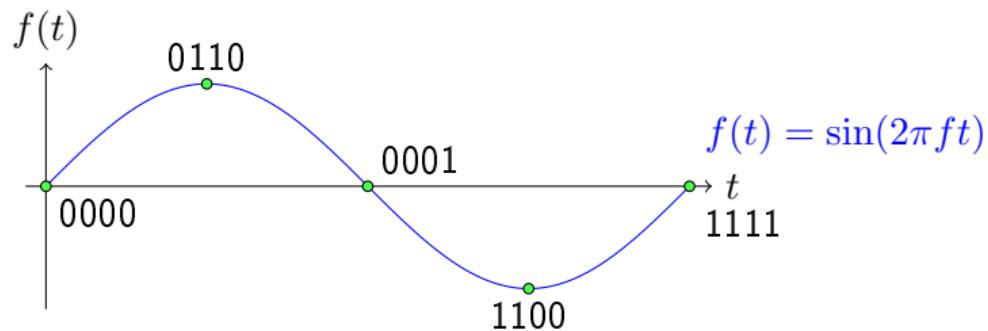


Um das Signal digital zu verarbeiten, müssen wir

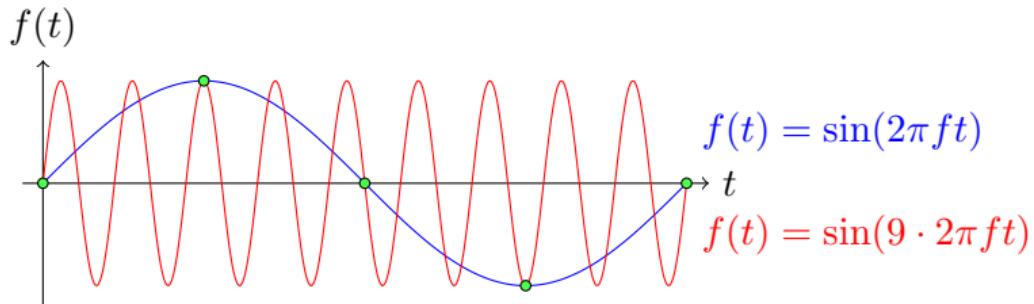
- ▶ Amplitude
- ▶ und Zeit diskretisieren (“Abtastpunkte” statt Verlauf)

Wie viele Abtastpunkte ( $\Rightarrow$  Speicherplatz) brauchen wir?

## Ein Signal abtasten

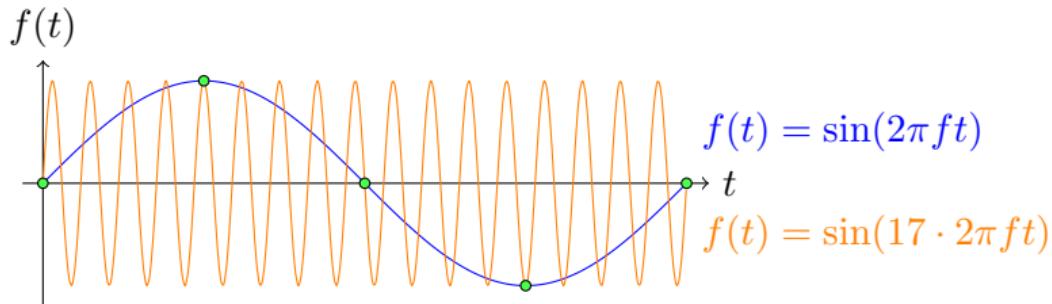


Kann ein Signal eindeutig rekonstruiert werden?



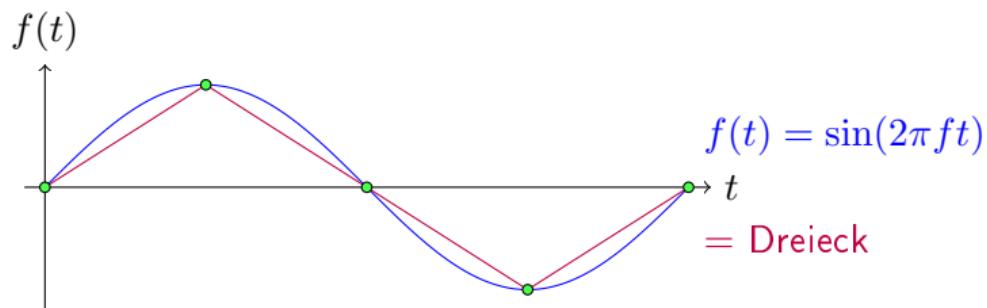
Kann ein Signal eindeutig rekonstruiert werden?

Es gibt unendlich viele Möglichkeiten ein abgetastetes Signal (falsch) zu rekonstruieren!



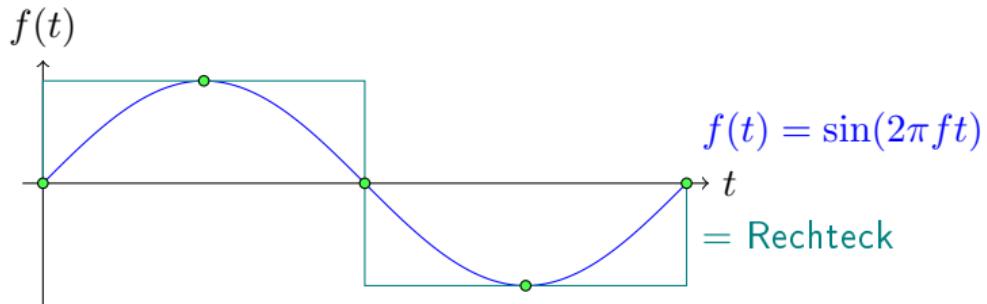
Kann ein Signal eindeutig rekonstruiert werden?

Es gibt unendlich viele Möglichkeiten ein abgetastetes Signal (falsch) zu rekonstruieren!



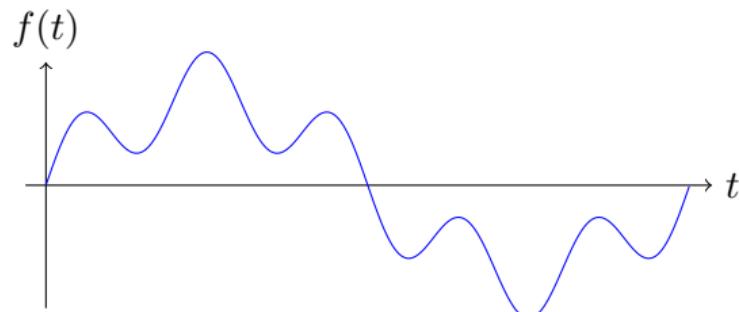
Kann ein Signal eindeutig rekonstruiert werden?

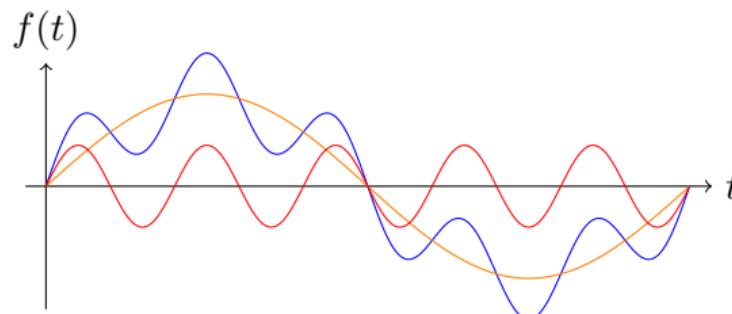
Es gibt unendlich viele Möglichkeiten ein abgetastetes Signal (falsch) zu rekonstruieren!



Kann ein Signal eindeutig rekonstruiert werden?

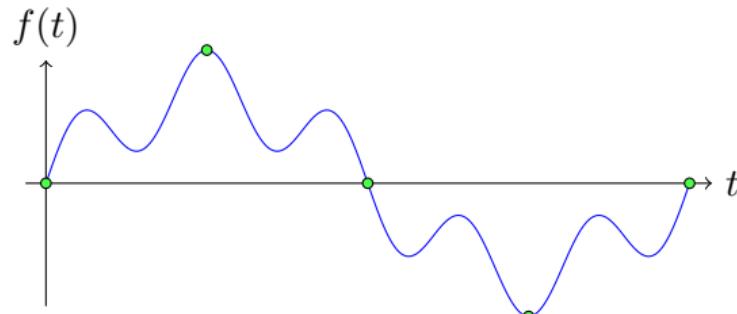
Es gibt unendlich viele Möglichkeiten ein abgetastetes Signal (falsch) zu rekonstruieren!



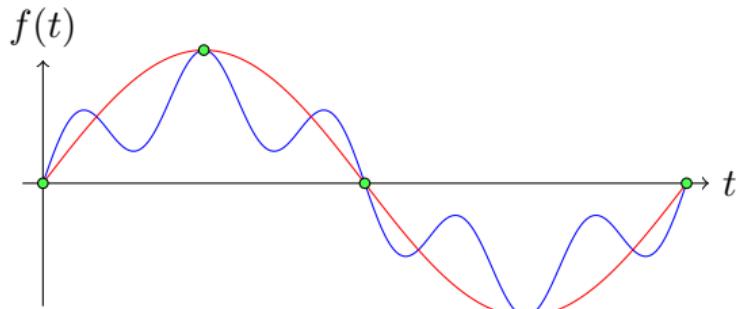


## Fourieranalyse:

- ▶ Man kann jedes Signal als Summe von Sinussignalen darstellen.

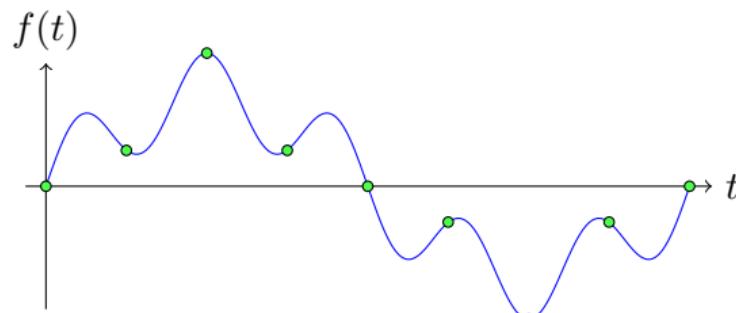


Wie viele Abtastpunkte brauchen wir mindestens?



Wie viele Abtastpunkte brauchen wir mindestens?

Abtasttheorem: Wir brauchen zumindest 2 Abtastwerte pro Periode des höchstfrequenten Sinus.



### Vorgangsweise

- Wir legen die höchste relevante Frequenz fest (Hörbereich ... ).
- Wir filtern alle höheren Frequenzen weg (Tiefpass):  
Dadurch können wir bei der Rekonstruktion alle höheren Frequenzen ausschließen.

Warum also Hintergründe verstehen, “unter die Haube blicken”?

- ▶ Interesse / Neugier?
- ▶ Um es ein bisschen besser machen zu können?

Warum also Hintergründe verstehen, “unter die Haube blicken”?

- ▶ Interesse / Neugier?
  - ▶ Um es ein bisschen besser machen zu können?
- 
- ▶ Weil es auch jemanden geben muss, der die Abstraktion erstellt.
  - ▶ Weil man Abstraktionen nur verwenden sollte, wenn man auch ihre Grenzen kennt!

Warum also Hintergründe verstehen, “unter die Haube blicken”?

- ▶ Interesse / Neugier?
  - ▶ Um es ein bisschen besser machen zu können?
- 
- ▶ Weil es auch jemanden geben muss, der die Abstraktion erstellt.
  - ▶ **Weil man Abstraktionen nur verwenden sollte, wenn man auch ihre Grenzen kennt!**

# Grenzen der Abstraktion

Warum also Hintergründe verstehen, “unter die Haube blicken”?

- ▶ Interesse / Neugier?
  - ▶ Um es ein bisschen besser machen zu können?
- 
- ▶ Weil es auch jemanden geben muss, der die Abstraktion erstellt.
  - ▶ Weil man Abstraktionen nur verwenden sollte, wenn man auch ihre Grenzen kennt!

Aber:

Trifft man denn wirklich auf so komplizierte Spezialfälle wo Abstraktionen versagen?

## Der einsame Hirte

hat bei einer Spielzeit von 4:25 in unkomprimierter Form bei

- ▶ Abtastung mit 44100 Hz: 41 MB

## Der einsame Hirte

hat bei einer Spielzeit von 4:25 in unkomprimierter Form bei

- ▶ Abtastung mit 44 100 Hz: 41 MB
- ▶ Abtastung mit 22 050 Hz: 21 MB

## Der einsame Hirte

hat bei einer Spielzeit von 4:25 in unkomprimierter Form bei

- ▶ Abtastung mit 44 100 Hz: 41 MB
- ▶ Abtastung mit 22 050 Hz: 21 MB
- ▶ Abtastung mit 2 205 Hz: 2.1 MB

## Der einsame Hirte

hat bei einer Spielzeit von 4:25 in unkomprimierter Form bei

- ▶ Abtastung mit 44100 Hz: 41 MB
- ▶ Abtastung mit 22050 Hz: 21 MB
- ▶ Abtastung mit 2205 Hz: 2.1 MB
- ▶ Abtastung mit 2205 Hz ohne Filter: 2.1 MB

## Der einsame Hirte

hat bei einer Spielzeit von 4:25 in unkomprimierter Form bei

- ▶ Abtastung mit 44 100 Hz: 41 MB
- ▶ Abtastung mit 22 050 Hz: 21 MB
- ▶ Abtastung mit 2 205 Hz: 2.1 MB
- ▶ Abtastung mit 2 205 Hz ohne Filter: 2.1 MB

## Die Abtastfrequenz macht's aus!

Auch hier macht es Sinn die Theorie zu kennen!

... hat die denkbar einfachste Funktion:

Signal wird unverändert übertragen.

- ▶ Ist das auch nur eine Abstraktion?
- ▶ Gibt es da auch Grenzen?
- ▶ Warum gibt es so viele verschiedene Kabel und Anschlüsse?

... hat die denkbar einfachste Funktion:

Signal wird unverändert übertragen.

- ▶ Ist das auch nur eine Abstraktion?
- ▶ Gibt es da auch Grenzen?
- ▶ Warum gibt es so viele verschiedene Kabel und Anschlüsse?

## Experiment

Wir analysieren ein Kabel!

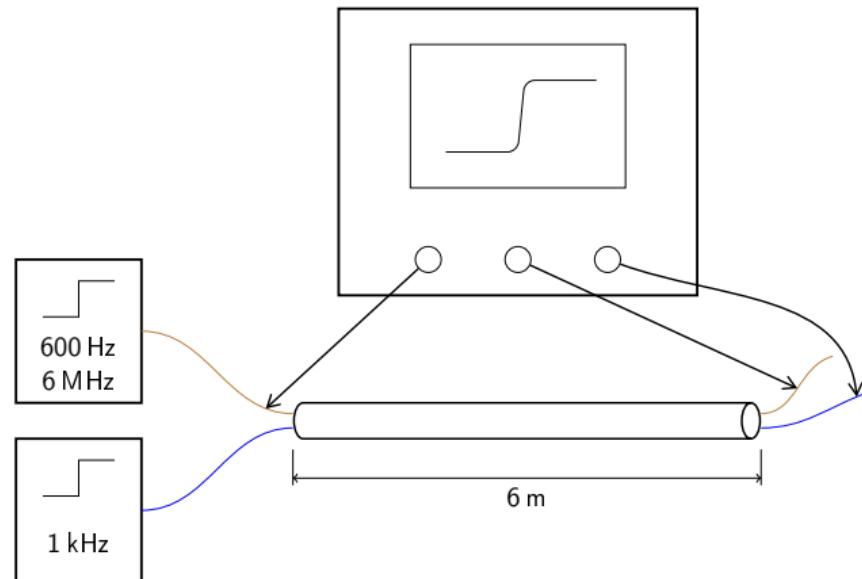


Abbildung: Versuchsaufbau

## Live-Demo

## Erkenntnisse

- ▶ Bei genauerer Betrachtung verhält sich das Kabel wie ein Tiefpass-Filter.
- ▶ Bei noch genauerer Betrachtung gibt es Signalverzögerung und Reflexionen.
- ▶ Es gibt trotz Isolation eine Kopplung zur Nachbarader.
- ▶ Wenn das Kabel lang und die Frequenz hoch ist, werden diese Effekte signifikant.

Schlechte Nachricht für die oberflächlichen Bastler

Wenn's dumm läuft ist nicht einmal ein Kabel ein Kabel.

## Schlechte Nachricht für die oberflächlichen Bastler

Wenn's dumm läuft ist nicht einmal ein Kabel ein Kabel.

## Gute Nachrichten für die tiefer Interessierten

- ▶ Die Grenzen der Abstraktion "Kabel" sind bestens erforscht.
- ▶ Wer die Grenzen kennt, kann die Abstraktion bis dorthin sicher verwenden.
- ▶ Auch für "jenseits der Grenzen" gibt es relativ einfache Methoden.

# Was steckt hinter der Abstraktion “Computer”?

- ▶ Das ist natürlich etwas umfangreicher,
- ▶ daher werden wir uns morgen damit ausführlich beschäftigen.
- ▶ Aber jetzt wir wollen uns einmal ansehen, ...

# Woraus besteht eigentlich ein Computerchip?

# "All computers are just carefully organized sand"

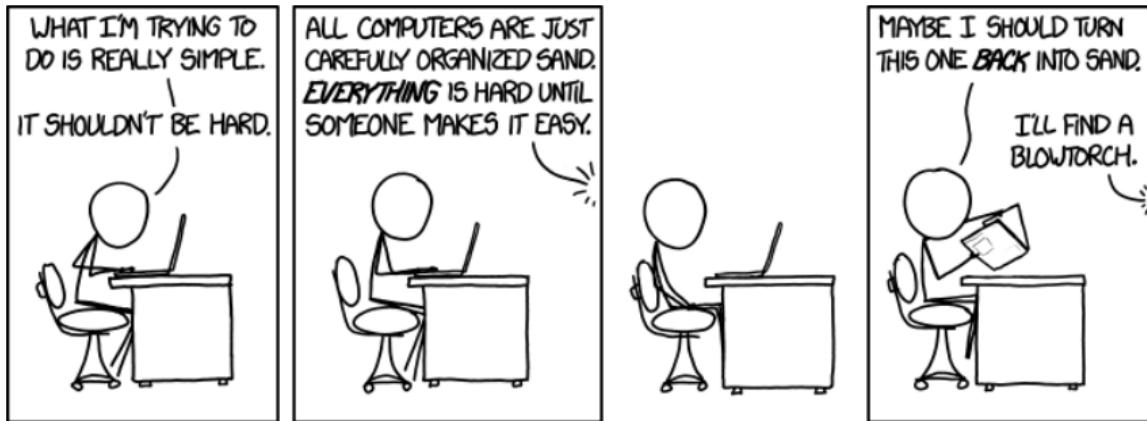


Abbildung: "Shouldn't Be Hard" by Randall Munroe CC-BY-NC-2.5, <http://xkcd.com/1349>

# "All computers are just carefully organized sand"

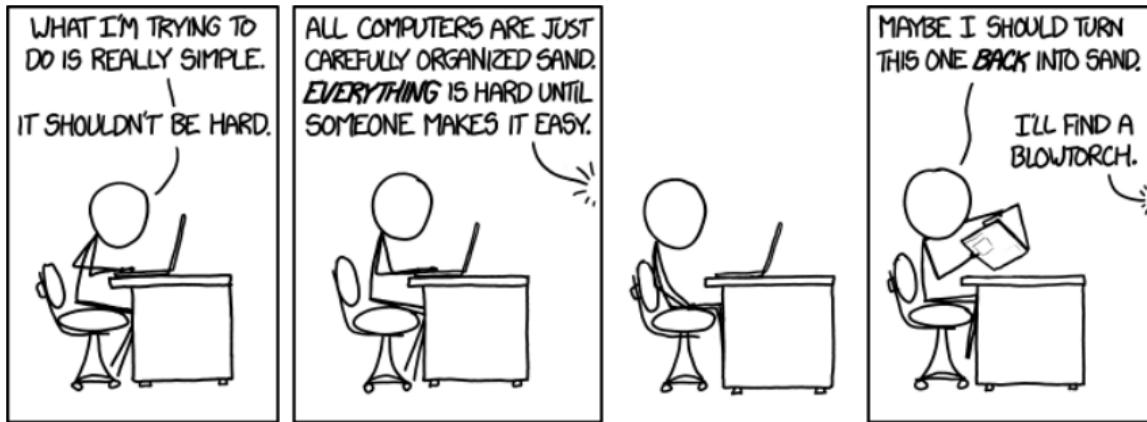


Abbildung: "Shouldn't Be Hard" by Randall Munroe CC-BY-NC-2.5, <http://xkcd.com/1349>

*(six hours later) ARGH. How are these stupid microchips so durable?! All I want is to undo a massive industrial process with household tools!*

“All computers are just carefully organized sand”



Abbildung: “Siliziumquelle” SiO<sub>2</sub>

“The Desert” by John O’Nolan CC-BY-2.0, <http://flic.kr/p/aEJ8Rk>

# Vom Sandkorn zum Chip



- ▶ Durchmesser 30–40 cm
- ▶ Länge 2 m
- ▶ Gewicht > 100 kg
- ▶ Reinheitsgrad > 99.999 999 9 %

**Abbildung:** Hochreiner Silizium-Einkristall.  
By Oleg Alexandrov CC-BY-SA-3.0  
via Wikimedia Commons.

# Vom Sandkorn zum Chip



Abbildung: Ein Wafer © Intel

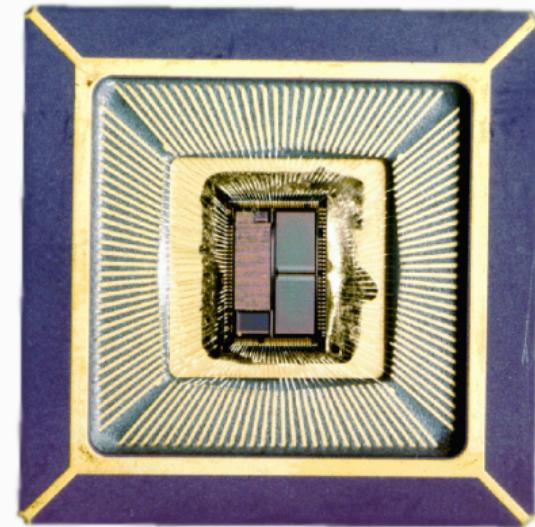


Abbildung: Chip im Package

# Vom Sandkorn zum Chip

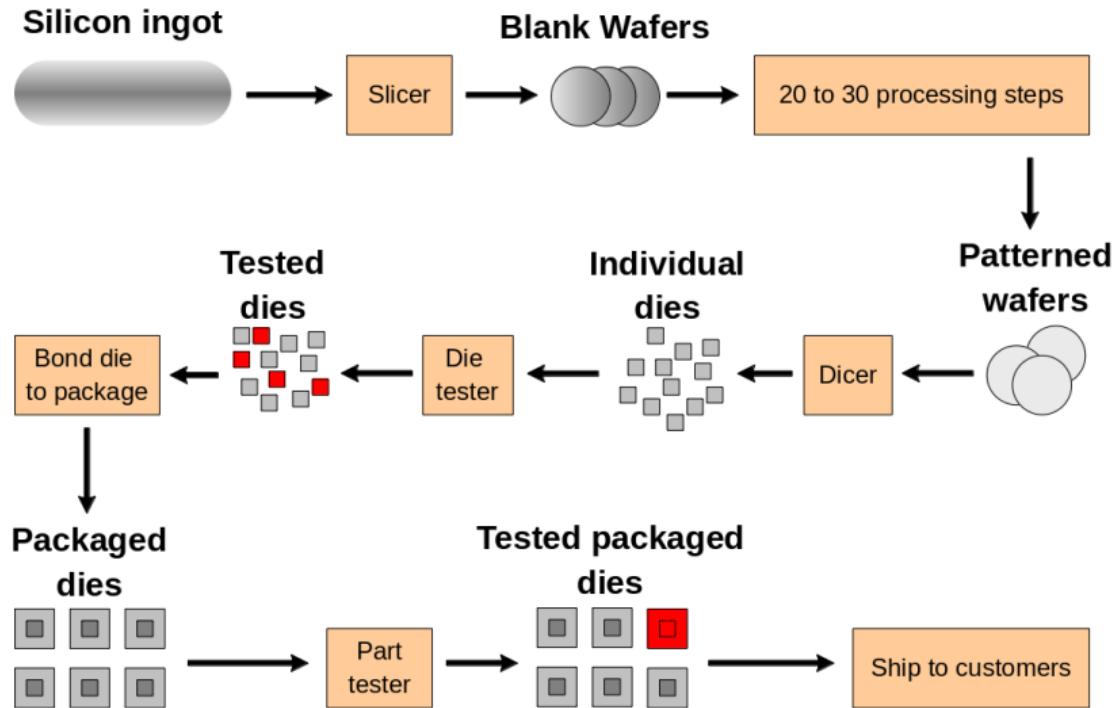


Abbildung: Industrielle Chipfertigung

## Was macht man nun mit den ganzen Transistoren?

- ▶ Standard Cell Designer entwerfen logische Grundbausteine (AND, OR, NAND, ...) aus Transistoren in konkreter Technologie, z.B. 14 nm.
- ▶ Bibliothek von solchen “Standard Cells” mit unterschiedlichen Vor-/Nachteilen: Geschwindigkeit, Größe, Stromverbrauch, ...

## Was macht man mit den Standard Cells?

- ▶ Designer benutzen Standard Cell Bibliothek um Funktionsblock nach ihren Vorstellungen zu realisieren.
- ▶ Hier gibt es viel Tool-Support.
- ▶ Schaltungsdesigner arbeiten meist mit “Beschreibungssprachen”, ähnlich wie Programmierer.
- ▶ Ein komplexer Chip kann hunderte Millionen Transistoren beinhalten.

# Grenzen der Chiptechnologie

Stellen Sie sich vor:

Ein 14 nm Transistor wäre so groß wie ein Fingernagel . . .

Stellen Sie sich vor:

Ein 14 nm Transistor wäre so groß wie ein Fingernagel ...

... dann wäre ein menschliches Haar so dick dass es auf dem Podium keinen Platz mehr hätte.

Anders gesagt:

Ein rotes Blutkörperchen ist etwa 500x größer als ein 14 nm Transistor!

## Intel 4004

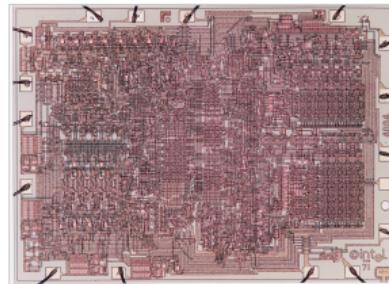
- ▶ Einführung 1971
- ▶ 4-Bit (vier!) Architektur
- ▶ 1 Kern
- ▶ 740 kHz
- ▶ 2300 Transistoren
- ▶ 16 Pins
- ▶  $144 \text{ mm}^2$  Die Fläche
- ▶  $10 \mu\text{m}$  Prozess

## Intel Core i7 2600 (Sandy Bridge)

- ▶ Einführung 2011
- ▶ 64-Bit Architektur
- ▶ 4 Kerne
- ▶ 3.4 GHz
- ▶ 1.16 Milliarden Transistoren
- ▶ 1155 Pins
- ▶  $216 \text{ mm}^2$  Die Fläche
- ▶ 32 nm Prozess

# Chiptechnologie — ein Generationenvergleich

1971: Intel 4004



2011: Intel i7

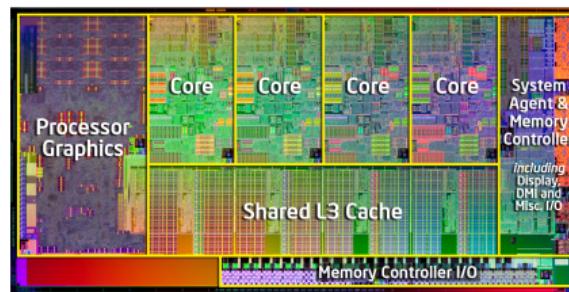


Abbildung: Die Designs in ihrer Technologie gefertigt © Intel

# Chiptechnologie — ein Generationenvergleich

2011: Intel 4004

144 mm<sup>2</sup> ⇒ 0.0003 mm<sup>2</sup>

2011: Intel i7

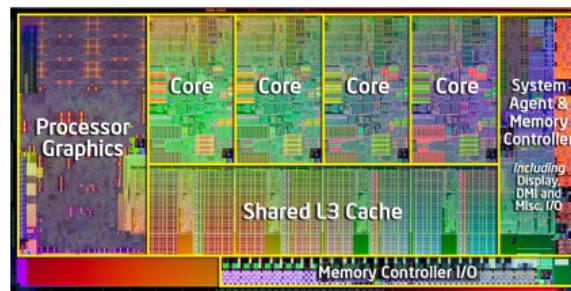


Abbildung: Das 40 Jahre altes Design in aktueller Technologie gefertigt © Intel

# Chiptechnologie — ein Generationenvergleich

2011: Intel 4004

144 mm<sup>2</sup> ⇒ 0.0003 mm<sup>2</sup>



2011: Intel i7

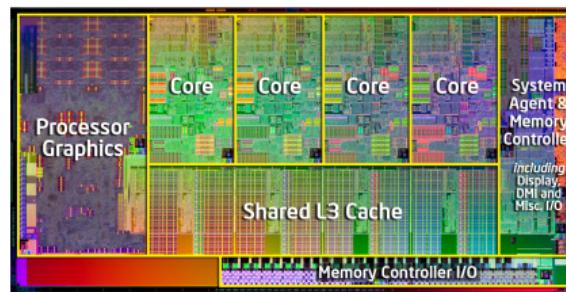


Abbildung: Das 40 Jahre altes Design in aktueller Technologie gefertigt © Intel



**Abbildung:** Mailüfterl (1958); 8000 “Transistoren”, 20 km Schaltdraht  
By Florian Staudacher CC-BY-3.0 via Wikimedia Commons.

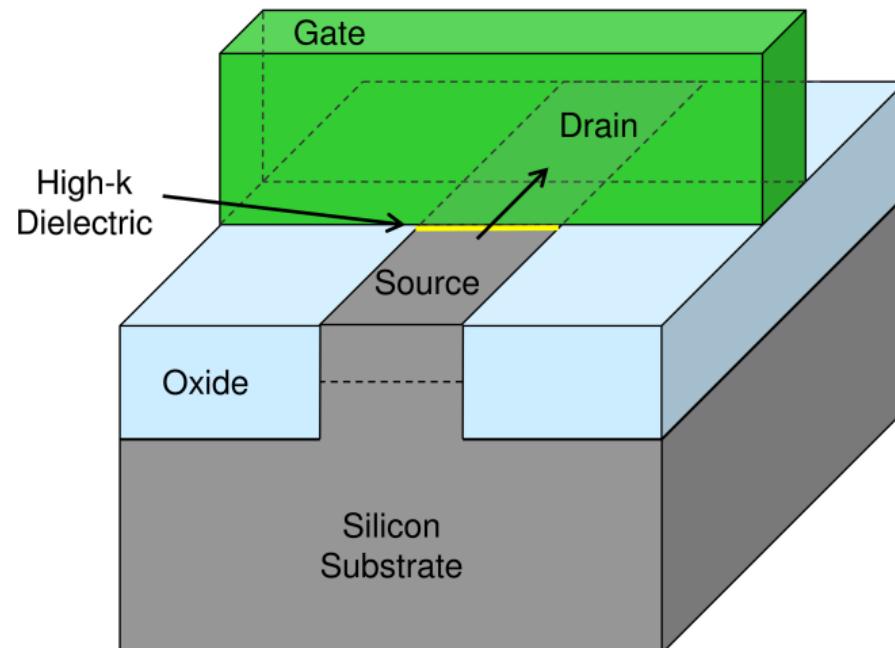


Abbildung: Traditionelles FET-Transistordesign © Intel

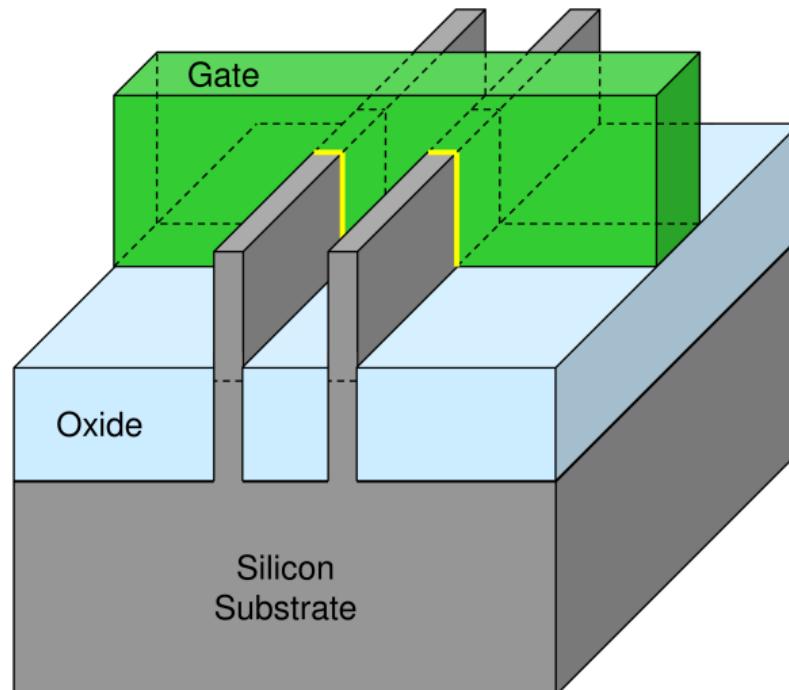


Abbildung: 14 nm FinFET-Transistordesign © Intel

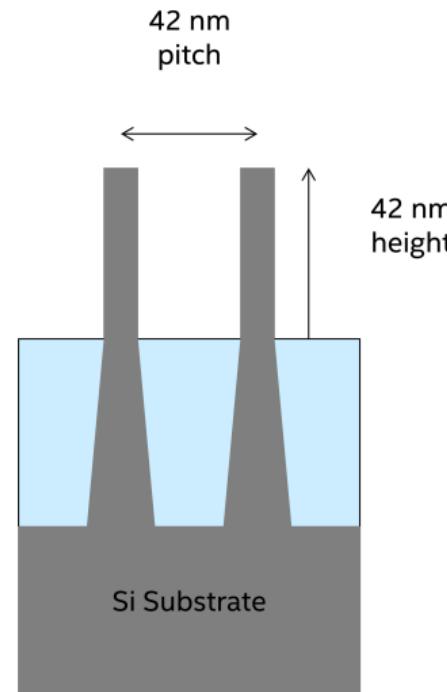


Abbildung: 14 nm FinFET-Transistor Querschnitt © Intel

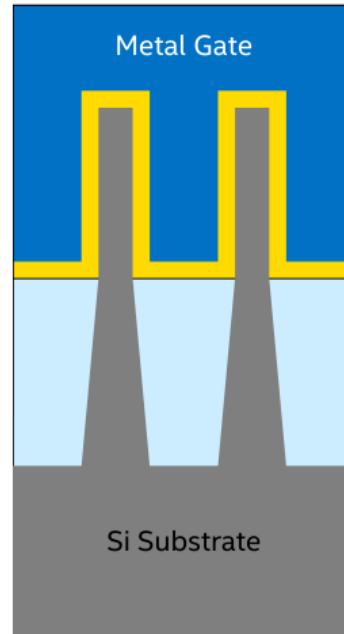


Abbildung: 14 nm FinFET-Transistor Querschnitt © Intel

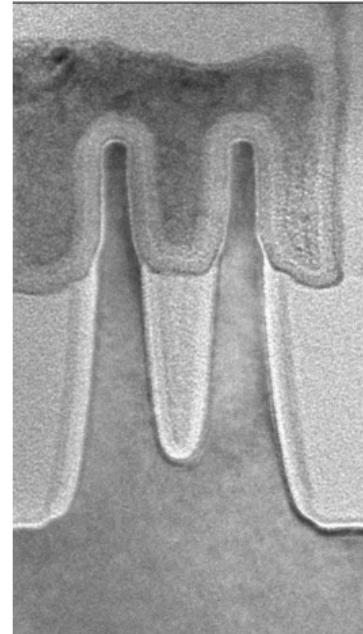


Abbildung: 14 nm FinFET-Transistor im Chip © Intel

Wo ist der 30 GHz Prozessor?

Oder: Warum stagnieren in letzter Zeit die Taktraten?

Bestimmt nicht einfach nur der Taktgeber die Taktrate?

- ▶ Wer bestimmt eigentlich, mit welcher Taktrate mein Rechner läuft?
- ▶ Wieso holen Übertakter noch enorme Steigerungen raus?

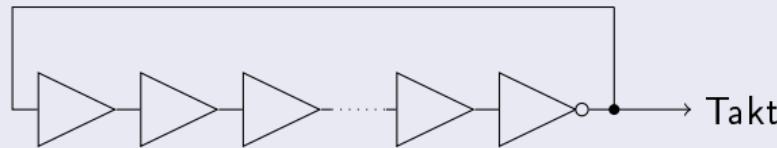
Bestimmt nicht einfach nur der Taktgeber die Taktrate?

- ▶ Wer bestimmt eigentlich, mit welcher Taktrate mein Rechner läuft?
- ▶ Wieso holen Übertakter noch enorme Steigerungen raus?

Probieren wir es aus!

Bauen wir eine Schaltung ohne Taktgeber und beobachten, was passiert.

## Ringoszillator



- ▶ Die Frequenz des erzeugten Taktes ist nur durch die Laufzeit einer langen Kette von Gattern bestimmt.

Wir bauen einen “Ringoszillator”

# Live-Demo

# Wir bauen einen “Ringoszillator” — Erkenntnisse

Wir haben gesehen dass:

- ▶ ... die Frequenz des Ringoszillators temperaturabhängig ist!
- ▶ Also muss auch die Gatterlaufzeit temperaturabhängig sein!
- ▶ Eine konstante Taktperiode ist also wieder eine Abstraktion.

# Wir bauen einen “Ringoszillator” — Erkenntnisse

Wir haben gesehen dass:

- ▶ ... die Frequenz des Ringoszillators temperaturabhängig ist!
- ▶ Also muss auch die Gatterlaufzeit temperaturabhängig sein!
- ▶ Eine konstante Taktperiode ist also wieder eine Abstraktion.

Warum brauchen wir diese Abstraktion?

- ▶ Die Signallaufzeiten am Chip sind abhängig von Temperatur und Spannung.
- ▶ Darum will sich ein Programmierer nicht kümmern müssen,
- ▶ daher werden diese Abhängigkeiten verborgen:
- ▶ Wir wählen eine konstante Taktperiode, die größer ist als die Laufzeit unter den schlechtesten Bedingungen.

Aber:

- ▶ Woher kommen diese Laufzeiten eigentlich?
- ▶ Sind elektrische Signale nicht unendlich schnell?

## Grenze 1: Lichtgeschwindigkeit

- ▶  $30 \text{ GHz} = 30 \cdot 10^9$  Taktperioden pro Sekunde,
- ▶ Lichtgeschwindigkeit  $c_0 \approx 3 \cdot 10^8 \frac{\text{m}}{\text{s}}$  (im Vakuum)  
Signale breiten sich am Chip mit  $\approx \frac{2}{3} c_0$  aus.

## Grenze 1: Lichtgeschwindigkeit

- ▶  $30 \text{ GHz} = 30 \cdot 10^9$  Taktperioden pro Sekunde,
- ▶ Lichtgeschwindigkeit  $c_0 \approx 3 \cdot 10^8 \frac{\text{m}}{\text{s}}$  (im Vakuum)  
Signale breiten sich am Chip mit  $\approx \frac{2}{3} c_0$  aus.
- ▶ In einer Taktperiode legt das Licht also nur 1 cm zurück!

### Grenze 1: Lichtgeschwindigkeit

- ▶  $30 \text{ GHz} = 30 \cdot 10^9$  Taktperioden pro Sekunde,
- ▶ Lichtgeschwindigkeit  $c_0 \approx 3 \cdot 10^8 \frac{\text{m}}{\text{s}}$  (im Vakuum)  
Signale breiten sich am Chip mit  $\approx \frac{2}{3} c_0$  aus.
- ▶ In einer Taktperiode legt das Licht also nur 1 cm zurück!
- ▶ Bei einem Chip mit 2 cm Kantenlänge braucht das Taktsignal also mehr als 2 Takte quer durch den Chip.
- ▶ Das widerspricht der Abstraktion, dass Taktflanken überall am Chip "synchron" stattfinden.

## Grenze 2: Lade-/Entladevorgänge

- ▶ Gatter sind aus Transistoren aufgebaut.
- ▶ Idealerweise sind Transistoren Schalter (Abstraktion ...)
- ▶ Bei genauerer Betrachtung verhalten sich Transistoren (und Leitungen) wie Kondensatoren und Widerstände.
- ▶ Laden bzw. Entladen von Kondensatoren über einen Widerstand benötigt Zeit (vgl.: über Schlauch Gefäß füllen).
- ▶ Diese Zeit begrenzt die erreichbare Geschwindigkeit des Chips.

Mehr Rechenleistung bei gleicher Geschwindigkeit?

Kann man die Physik überlisten?

## Lösung 1: Multi-Core und vernetzte Rechner

- ▶ Moderne Desktop Computer: 4–16 Prozessoren (Cores) verbaut in einem Chip, Tendenz steigend.
- ▶ Moderne Grafikkarten: über 1000 “Cores”.
- ▶ Parallele Programme erfordern neue Programmiertechniken;  
**große Herausforderung für die Informatik!**

## Lösung 2: Spezialisiertes Design

- ▶ Bisher betrachtet: “general-purpose” Prozessoren die für verschiedenste Einsatzbereiche geeignet sind.
- ▶ Programmierbarkeit und Vielseitigkeit kosten Effizienz.
- ▶ Spezialisierte Designs (ASICs) erreichen eine Effizienz die mit general-purpose Lösungen nicht erreichbar ist.
- ▶ Höherer Entwicklungsaufwand, aber unverzichtbar für Anwendungsbereiche, in denen general-purpose Lösungen die benötigten Anforderungen (Performance, Verlustleistung, Fehlertoleranz, ...) nicht erfüllen.

# Bachelor TI

Ein paar Highlights . . .

## Hardwarenah programmieren?

- ▶ Programmkonstruktion
- ▶ Microcontroller und Betriebssysteme
- ▶ Rechnerstrukturen und Betriebssysteme

## Hardware entwerfen?

- ▶ Grundlagen Digitaler Systeme
- ▶ Elektrotechnische Grundlagen
- ▶ Digital Design

## Signale und Grundlagen?

- ▶ Signale und Systeme
- ▶ Modellbildung in der Physik
- ▶ Regelungstechnik

## Fehlertoleranz?

- ▶ Dezentrale Automation
- ▶ Programm- und Systemverifikation
- ▶ Zuverlässige Echtzeitsysteme

Wir haben erlebt, dass . . .

- ▶ Abstraktion ein mächtiges Werkzeug ist,
- ▶ dessen Grenzen man allerdings stets genau kennen sollte.
- ▶ Diese Kompetenz vermittelt ein Studium.
- ▶ Erlernen von Grundlagen ist essenziell
- ▶ . . . und macht Spaß!