Formale Verifikation von Hardware

Dr. Felix Klein

15. Mai 2025 – **SIC!**

QBayLogic.
The key to FPGA-ASIC design

Hardwareentwickelung: Grundlagen

Software:

```
def calc(a, b):
  c = a * a
  r = c + a * b
  return foo(r)
```

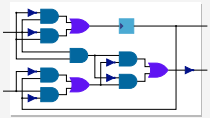
KOMPILIERUNG

Maschinencode

Hardware:

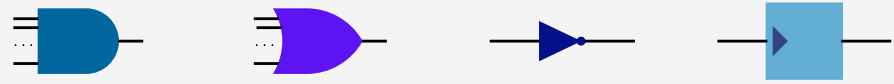
```
calc a b =
  let c = a * a
      r = c + a * b
   in foo r
```

SYNTHESE

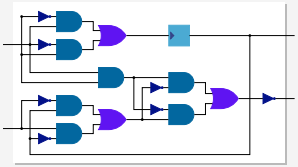


Schaltkreis

Hardwareentwickelung: Grundlagen



Hardwareentwickelung: Grundlagen



(AS)IC
(application-specific)
Integrated Circuit

FPGA
Field Programmable
Gate Array

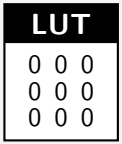
Hardwareentwickelung: Grundlagen

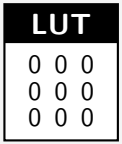
UND Gatter

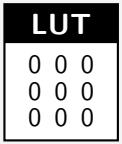
ODER Gatter

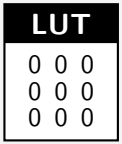
Inverter

Latch

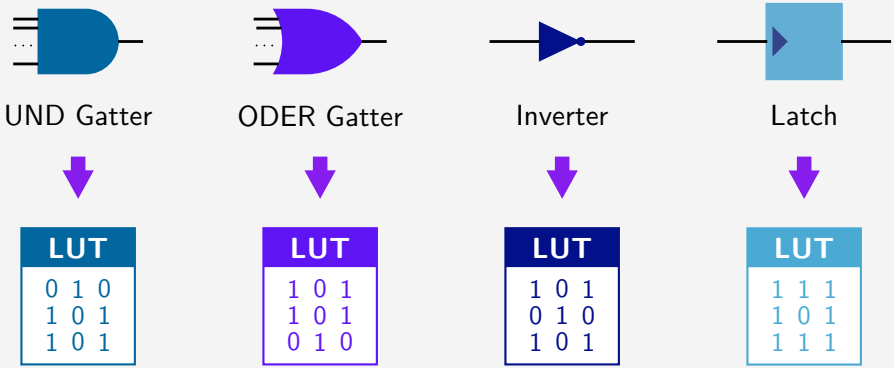




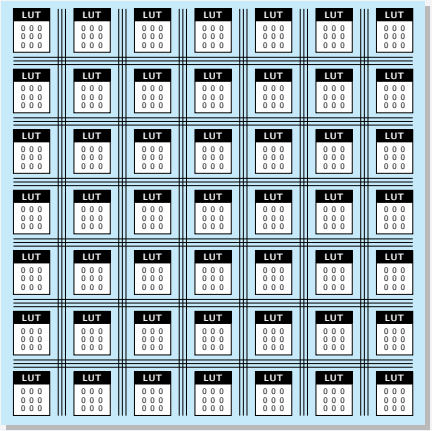




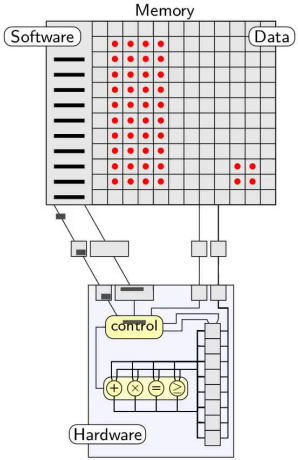
Hardwareentwickelung: Grundlagen



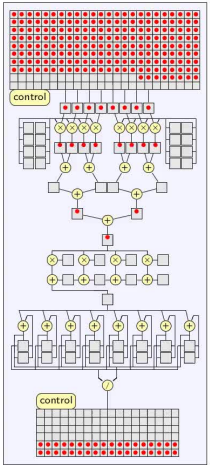
Hardwareentwickelung: Grundlagen



FPGA



Normal processor
Software controls computation process

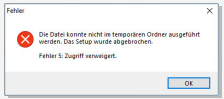


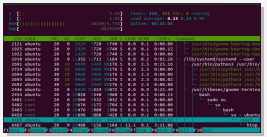
FPGA
Computation integrated in hardware

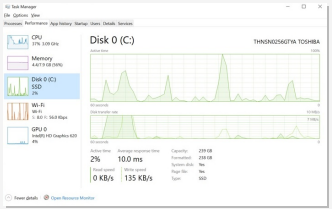
Software ↔ Hardware: Fehler finden *(Debugging)*

Software

- Aufteilung in einzelne Prozesse
- Betriebssystem bietet Werkzeuge um Prozesse
  - zu beobachten (Resourcenverbrauch: Auslastung, Speicher)
  - zu steuern (pausieren, Schritt-für-Schritt Ausführung)
  - zu manipulieren (Debugger Schnittstelle)
- i.d.R. existiert genügend Speicher zur Datenerfassung
- Logging / Fehlermeldungen

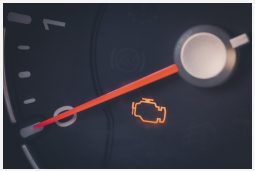






Software ↔ Hardware: Fehler finden *(Debugging)*

- hohe Parallelisierung & Datenverarbeitungsraten
- i.d.R. zu wenig Speicher um alle Daten (ausreichend schnell) zu erfassen
- Simulation
  - langsam / kostenintensiv
  - Separation in Teilkomponenten erforderlich
- In-circuit Debugging / In-circuit Testing (ICT)
  - Integrated Logic Analyzer (ILA)
  - Debug Interfaces: SWD, JTAG
- Oszilloskop



$\Longrightarrow$ **hoher Aufwand, hohe Kosten**

Software ↔ Hardware: Fehler beheben *(Updates)*

- Austausch des fehlerhaften Program(teil)s durch eine neuere Version
- Gesonderter Update Prozess
  - heutzutage meist automatisiert übers Internet
  - erfordert Systemneustart oder zumindest Neustart abhängiger Prozesse
- Besondere Herausforderung: **Eingebettete Systeme**
  - Aktualisierungsprozess oft komplexer & kritischer
  - Gefahr eines Geräte-Bricks
  - IoT / Industrie 4.0

Software ↔ Hardware: Fehler beheben *(Updates)*

- **ICs / ASICs**
  - erneuter Tape-out
  - ggf. im Umlauf befindliche Chips austauschen
- **FPGAs**
  - neuen Bitstream einspielen
  - ähnlicher Aufwand wie bei einem Firmware Update

⟹ **hoher Aufwand, hohe Kosten, erhöhte Risiken**

Software ↔ Hardware: Fehler vermeiden *(Verifikation)*

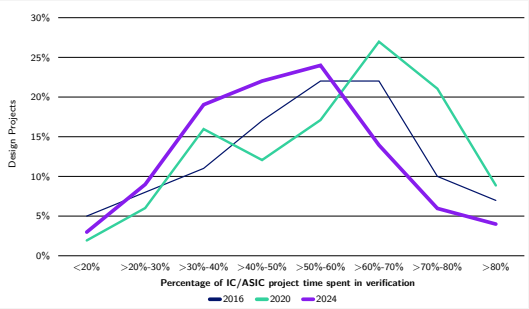
- potentiell unendlicher Zustandsraum
- Semantik des Maschinenmodells muss berücksichtigt werden

  Speicher / Caches / Multi-CPU / Befehlssatzarchitektur / Betriebssystem
- erfordert den Einsatz verschiedener Abstraktionsebenen
  - oft nur modellbasierte Betrachtungen möglich
  - Annahmen an die Funktionsweise der zugrunde liegenden Hardware
- **Herausforderungen**
  - Kompositionalität
  - Skalierbarkeit

Software $\leftrightarrow$ Hardware: Fehler vermeiden *(Verifikation)*

- prinzipiell endlicher Zustandsraum *(aber dennoch groß)*
- i.d.R. präzise Semantik der einzelnen Komponenten
- Digital Design: effiziente Abstraktion der zugrunde liegenden Physik
- **Herausforderungen**
  - Kompositionalität
  - Skalierbarkeit

$\Longrightarrow$ **einfacher, kosteneffizienter, erhöhter Bedarf**

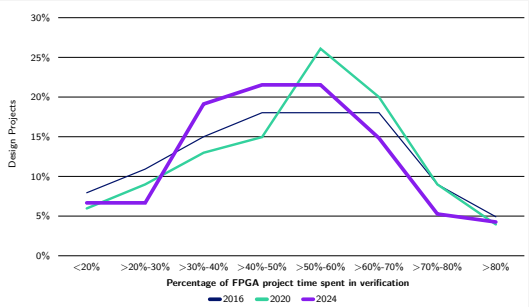


Quelle: *2024 Wilson Research Group IC/ASIC Functional Verification Trend Report*, Siemens

Software ↔ Hardware: Fehler vermeiden *(Verifikation)*

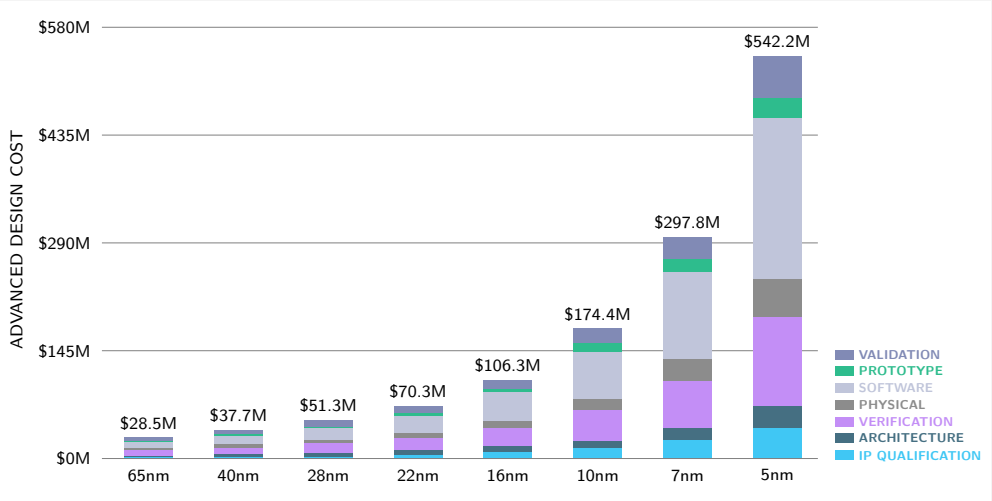
- prinzipiell endlicher Zustandsraum *(aber dennoch groß)*
- i.d.R. präzise Semantik der einzelnen Komponenten
- Digital Design: effiziente Abstraktion der zugrunde liegenden Physik
- **Herausforderungen**
  - Kompositionalität
  - Skalierbarkeit

$\Longrightarrow$ **einfacher, kosteneffizienter, erhöhter Bedarf**



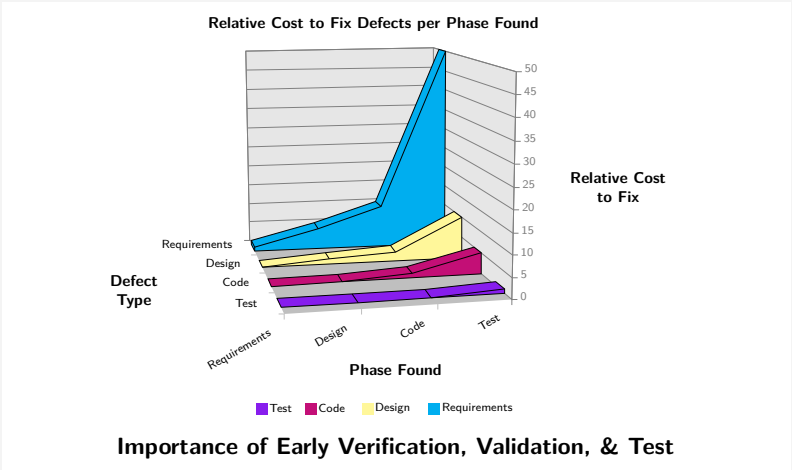
Quelle: *2024 Wilson Research Group FPGA Functional Verification Trend Report*, Siemens

Welche Kosten entstehen durch fehlerhafte Funktionalität



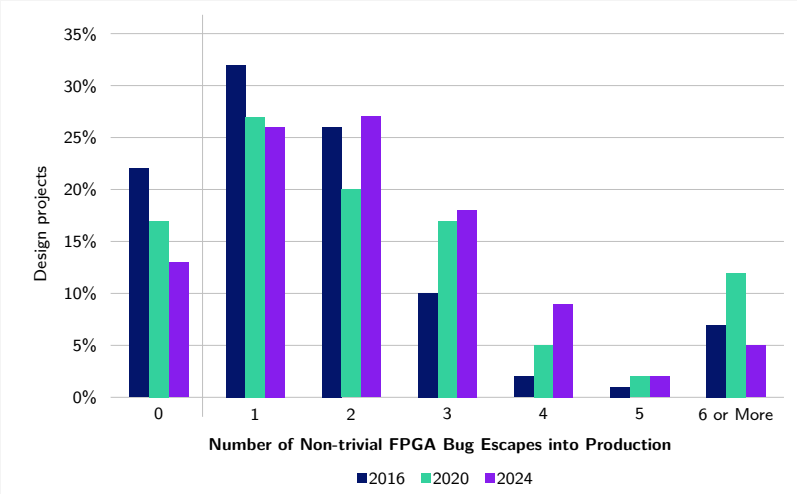
Quelle (2021): *Veloce Hardware-Assisted Verification: Complete, Unified and Progressive*, Siemens

Welche Kosten entstehen durch fehlerhafte Funktionalität



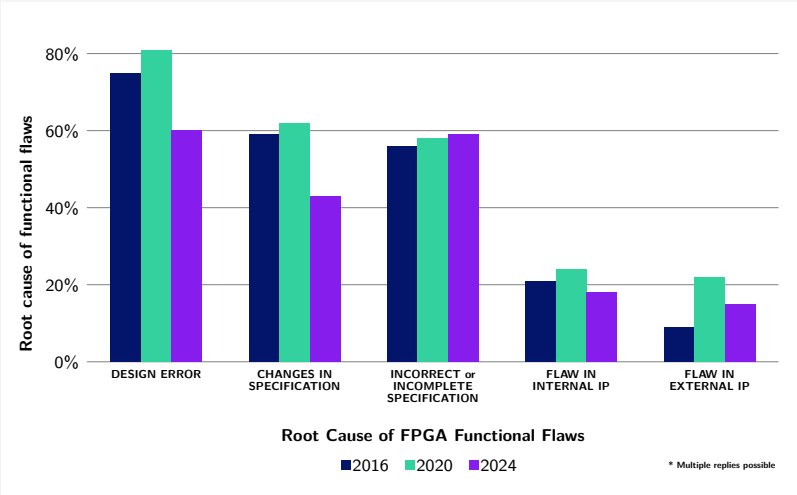
Quelle (2004): Return of Investment for Independent Verification & Validation, NASA

Welche Kosten entstehen durch fehlerhafte Funktionalität



Quelle: *2024 Wilson Research Group FPGA Functional Verification Trend Report*, Siemens

Welche Kosten entstehen durch fehlerhafte Funktionalität



Quelle: *2024 Wilson Research Group FPGA Functional Verification Trend Report*, Siemens

Welche Kosten entstehen durch fehlerhafte Funktionalität

Referenzbeispiel *(2024)*

Intel's *CPU Degradierung*
der 13./14. Generation

Welche Kosten entstehen durch Sicherheitslücken

Referenzbeispiel *(2017)*

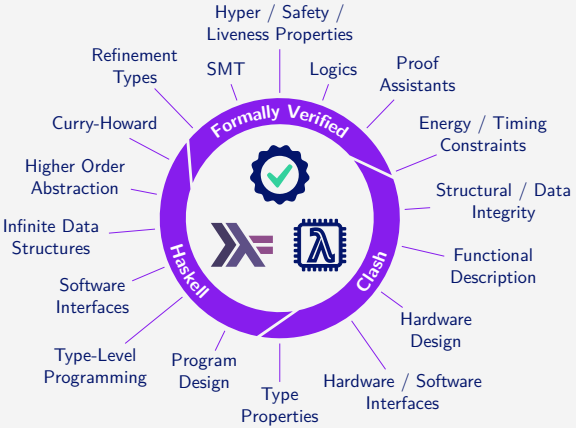
ROCA Sicherheitslücke in der Infineon RSA Bibliothek

CVE-2017-15361

(ROCA: Return of the Coppersmith Attack)

Clash Formal

Ecosystem formally verifiable IT



Beweisassistenten

Agda ROCQ LEAN

SMT / Refinement Types

Liquid Haskell

Spezifikationsverfahren

Sail Sail RISC-V CHERI

https://clash-formal.org