

AER

Mikroelektronik Moors Law + 16-24 monate
IC - Integrated Circuit Verdopplung
! Pins, Speicher zugriffszeit

Hardware - Entwurf
complex
time-to-Market TTM
re-spin teuer erneute Fabrikation
↓

Abstraktion Entwurf \Rightarrow technische Realisierung
Synthese nicht immer ideal

Entwurfsmethoden
Zerlegung
hierarchisch
regulär (Varielfältigung Komponente)

Verifikation

Testbench Ziel: abstrakt erweiterbar wiederwendbar
modular
Generator \rightarrow Scoreboard \rightarrow Vergleicher - Reporter
↓
Driver \rightarrow DUT \rightarrow Monitor
Prüf- Post-Synthese

abstrakte Realisierung

Verhaltensebene
effiziente Umsetzung Systemebene grobe Aufteilung von Struktur, Zeit, Daten, Kommunikation möglich
RTL - Register Transfer Ebene synchron, getaktet
RTL-Simulation Logik-/Gatter-Ebene Zeitverhalten abschätzbar
Analysimulation Transistor-/Schaltkreisebene mit Schaltzeiten
Layoutebene Geometrie Transistoren, Leitungen
genaues Zeitverhalten ↓ Zellen = Basisgatter

Tape-Out
Halbleiterhersteller komplexes wie 46 Empfänger
manuell bis Layout

RTL-Modell

\downarrow HDL-Compiler

unoptimierte Zwischendarstellung
Design Compiler (Zeit Fläche Ausfallrisiko)
Lokaloptimierung & Design Constraints
Technologymapping Technologie, Bibliothek
optimiertes Gattermodell

Detaillierung
Komposition Top-down -
Bottom-up - Entwurf
Meet-in-the-Middle -

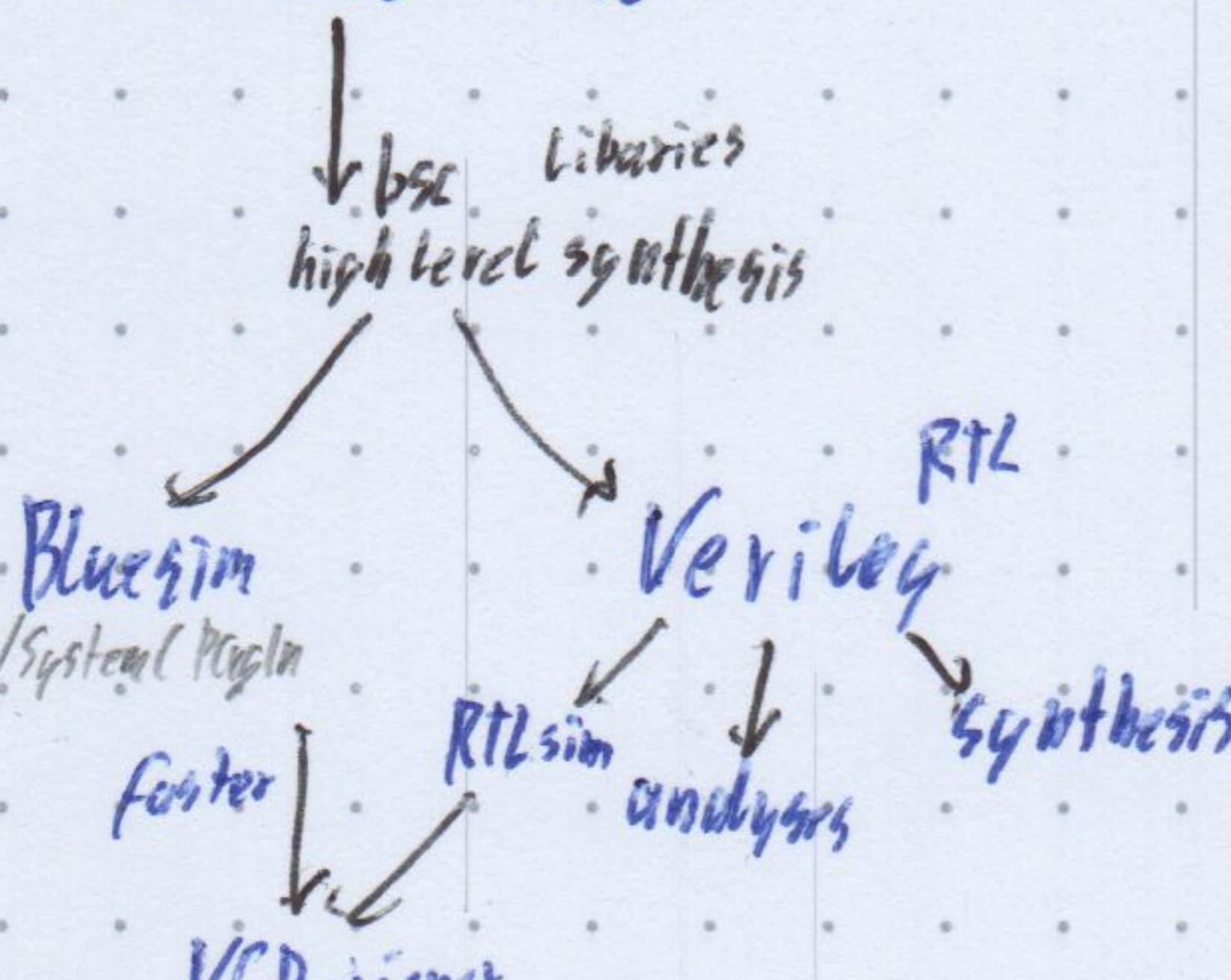
Functional Coverage Wie viel der Spezifikation verifiziert

riesiger Inputraum manuell aufwendig
Constrained Random Stimulus
zu fällige repräsentative Testfälle

Bluespec System Verilog

- Verhaltensbeschreibung
- Strukturbeschreibung
- atomare Regeln, Methoden
- basiert auf termverarbeitungssystemen
- angelehnt an Haskell

SOURCE CODE



Zerlegungshierarchie von Modulen

Blätter: primitive Zustandselemente
Reg
FIFO
Schnittstellen
(Methoden)
Regeln
können
Methoden
Aufrufen
Ricke
mk Name
mk Name (Ifc-Name); // Parum?
Value
" methods.
endinterface kann Zustand Schaltung nicht ändern
method mk Name (Ifc-Name); // Parum?

// Modulinstanziierung

Type #(param) name <- mk Name (param);
Reg #(type)
Int #(A,B)
Bool
mkReg (initialValue)
mkRegU

// functions - nur syntactic sugar

function Type|Action|ActionValue#(type) name (Type param);

z.B.

return

action

// Befehle

endaction;

endifunction

function

return

action

// Befehle

endaction;

endifunction

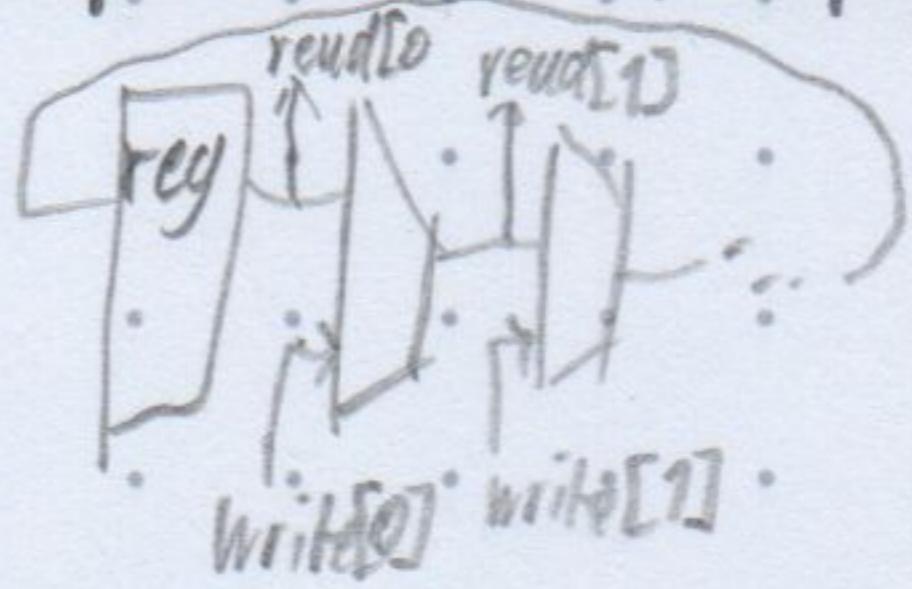
(Reg) - Concurrent Register

Lephemeral Historie innerhalb Takt

Array an Reg Schnittstellen.

Reg H(1). name[N] \leftarrow mk(Reg(N, initialV));
name[0].-read < name[0].-write < name[1].-read

Impl. z.B. mit Multiplexern



- + wohl definiert
- nur wenn Funktionalität noch nicht vorhanden

Beeinflussung Ablaufplan

Dringlichkeit # Frühzeitigkeit
Reihenfolge Regelbedingung Regelkörper
urgency cond body



(* descending_urgency = "r1, r2")

(* execution_order = "r1, r2")
 $\Rightarrow r1 < r2$

BSV Attribut: (* attribute = "...")
üblicherweise vor betreffender regel/methode

(* preempts = "r1, r2")
r1 unterdrückt r2

(* mutually_exclusive = "r1, r2")

Zusicherung an Compiler
Regelbedingungen vor r1 und r2 schließen sich gegenseitig aus

BSV zu Verilog

Grundsätzlich modul. zu modul
aber aus Effizienz oft inline'd aufgelöst

(* synthesize *) vor module nur ausgewählte Module

bsc ... -g mkM1 -g mkM2 bleiben erhalten

Einschränkung
Schnittstelle nach außen
nur aus Bits, Skalaren,
Bit-Vektoren

Schnittstellen

Formale Parameter \Rightarrow input Ports method attribute
Ergebnis \Rightarrow output Ports method

falls (Ausführungsbereitschaft \Rightarrow output Port RDY_methodname
nötig Ausführen von action[bit] \Rightarrow EN-methodname

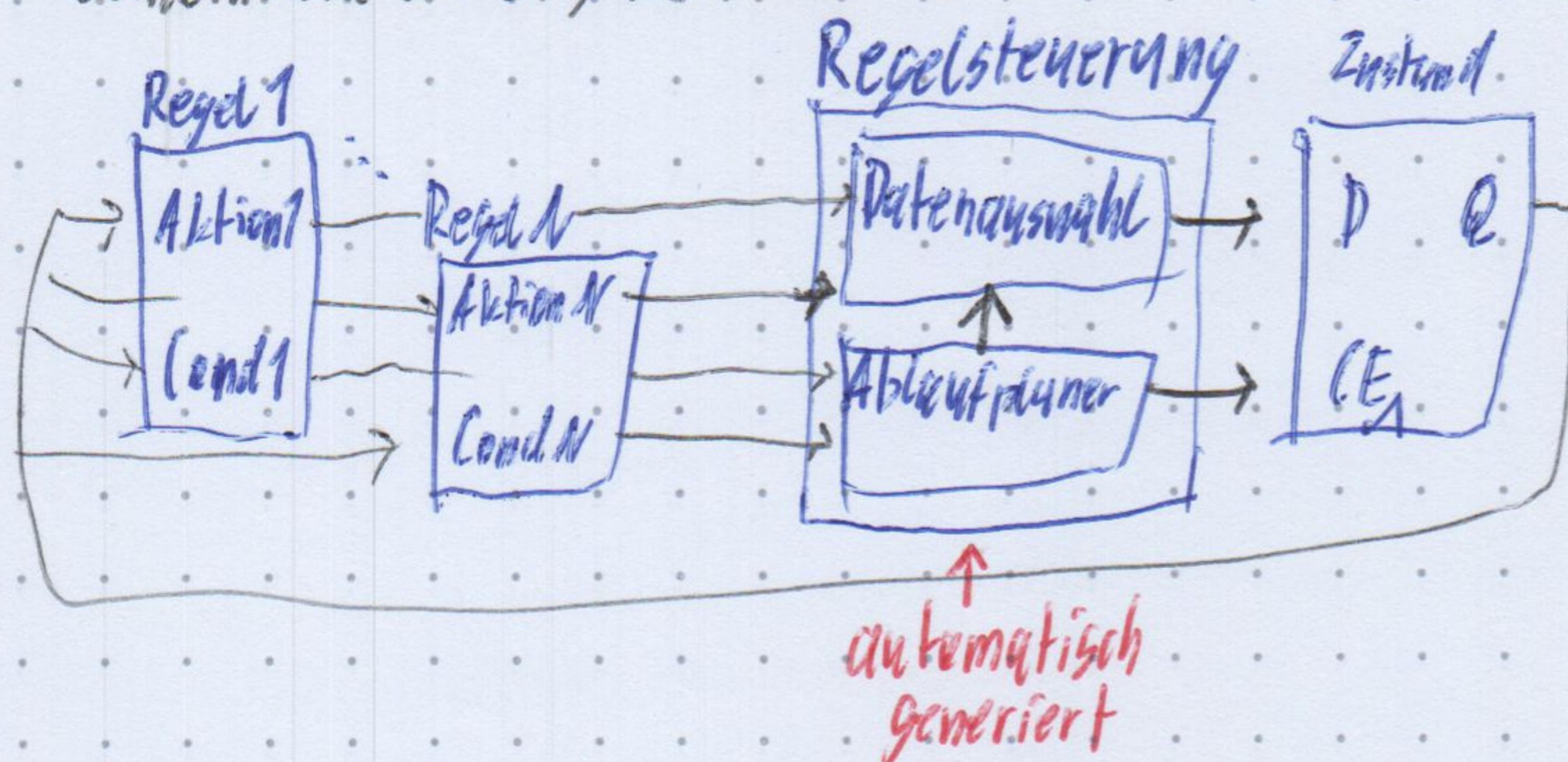
Compiler

bsc -verilog -g top -g dut ... -u myfile.bsv
andere inline'd

Linken

bsc -verilog -e top
Ergebnis a.out

Gemeinsame Nutzung von Hardware



Clock- und Reset-Signale

automatisch erzeugt

im Verilog-Testrahmen aus Bluespec Bibliothek
nur config möglich, standardmäßig 1 Takt, 1 Reset

Regeln in for-Schleifen erlaubt
Enthalt Z.

kleinere Chips \Rightarrow weniger kaputt \Rightarrow + Myield
Chip Herstellungsfehler \rightarrow Redundanzen durch Firmware gesteuert
teurer reparieren

Entwicklung p. a.

- +55% Chip Komplexität (zahl transistoren, Speichergröße)
- +33% Packungsdichte
- +33% Taktfrequenz $-!$ Wärme \Rightarrow Parallelle Systeme
- +12% Chip-Fläche
- +12% mehr Pins \Rightarrow Flaschenhals
- 6% Speicherzugriffszeit

(verb Fabrikosten) für maler ältere Technologien
 $\sim 2 \times$ p. a. Fertigungskosten \rightarrow Multi-Projekt-Chips

- Software V teuer, weil FPGAs
Verification V wenige Kunden
- Validation
- Physical
- IP Qualification Eingekauft
Architecture V Lizenzen (teilweise verschlüsselt)
Prototype

Kostensenkung durch Massenfertigung

Härdware - Entwurfstechniken

viele Möglichkeiten
Zeitdruck Time-to-Market TT.M
Fehler → Millionenenschaden

Abstraktion

Verhaltensebene

Was soll passieren?
Realisierung offen

Vorlsg.
 $out = a * b;$

auch rekursiv

reguläre Zerlegung - gezielte Vervielfältigung von Komponenten

Entwurf:

Top-down ↔ Meet-in-the-Middle ↔ Bottom-up

Detaillierungsprozess → Vereinigt

setzt Teilsysteme als

Blockbaus mit

Untermodulen idealer Funktionalität

verans.

von Wissen über Gesamtfunc.

Kompositionssprozess
von Primitiven
aus Bibliothek

Systemebene

Komponenten / grobe Aufteilung
Zeit, Daten Kommunikation

RTL Register-Transfer- Ebene

Pipelines / Automaten netze

Logik- oder Gatter- Ebene

generische Gatter

Digital
Analog

Transistor- / Schaltkreisebene

Schaltpläne: Transistoren, Widerstände, ...

Layout - Ebene

Geometrie der Transistoren, Leitungen
als Polygone

maßstabs
getreue
Abbildung
tape-out
an Halbleiter-
hersteller

Chip

Synthese

RTL-Simulation RTL-Modell - Prä-Synthesen-Simulation

HDL-Compiler

unoptimiert Zwischendarstellung

Design
Compiler

Logikoptimierung

Technologie-Mapping

Optimiertes Gattermodell

keine Synthese
mehr z.

Platzieren/ Verdrahten

Back-Annotation

Gattermodell
mit Leitungslaufzeiten

Zeit (geschätzt)

Timing-Analyse

Fläche geschätzt

ohne Verdichtung

Technologie
Bibliothek

Post-Synthesen
-Simulation

Elektrische Leistungsanforderungen

Simulation auf Logik-Ebene

bestimmung Umschaltfrequenz

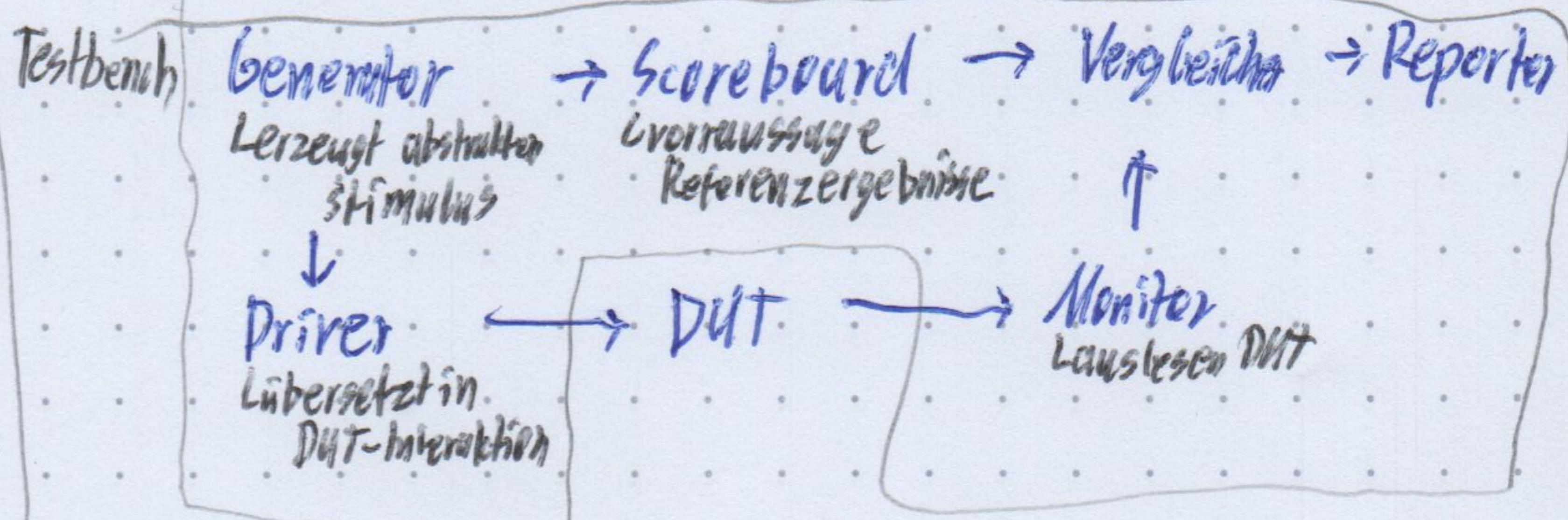
hoch selten (Toggle frequency)

mit gleicher Testbank

Vergleich miteinander

anderes Zeitverhalten

Verifikation ≈ sicherheitskritisch?



modular

erweiterbar

abstrakt

wiederverwendbar

andere Designs

Abstraktions-
Ebene

Umsetzung durch
Frameworks

Was testen?

Functional Coverage unabh. Spezifikation
Code zur Zuordnung - Coverpoints/groups

+ quantifiziert Fortschritt

Code Coverage

misst keine vorgegebene Funktionalität

manuelle Tests (directed testing)

laufwändig
Entwickler overfitting

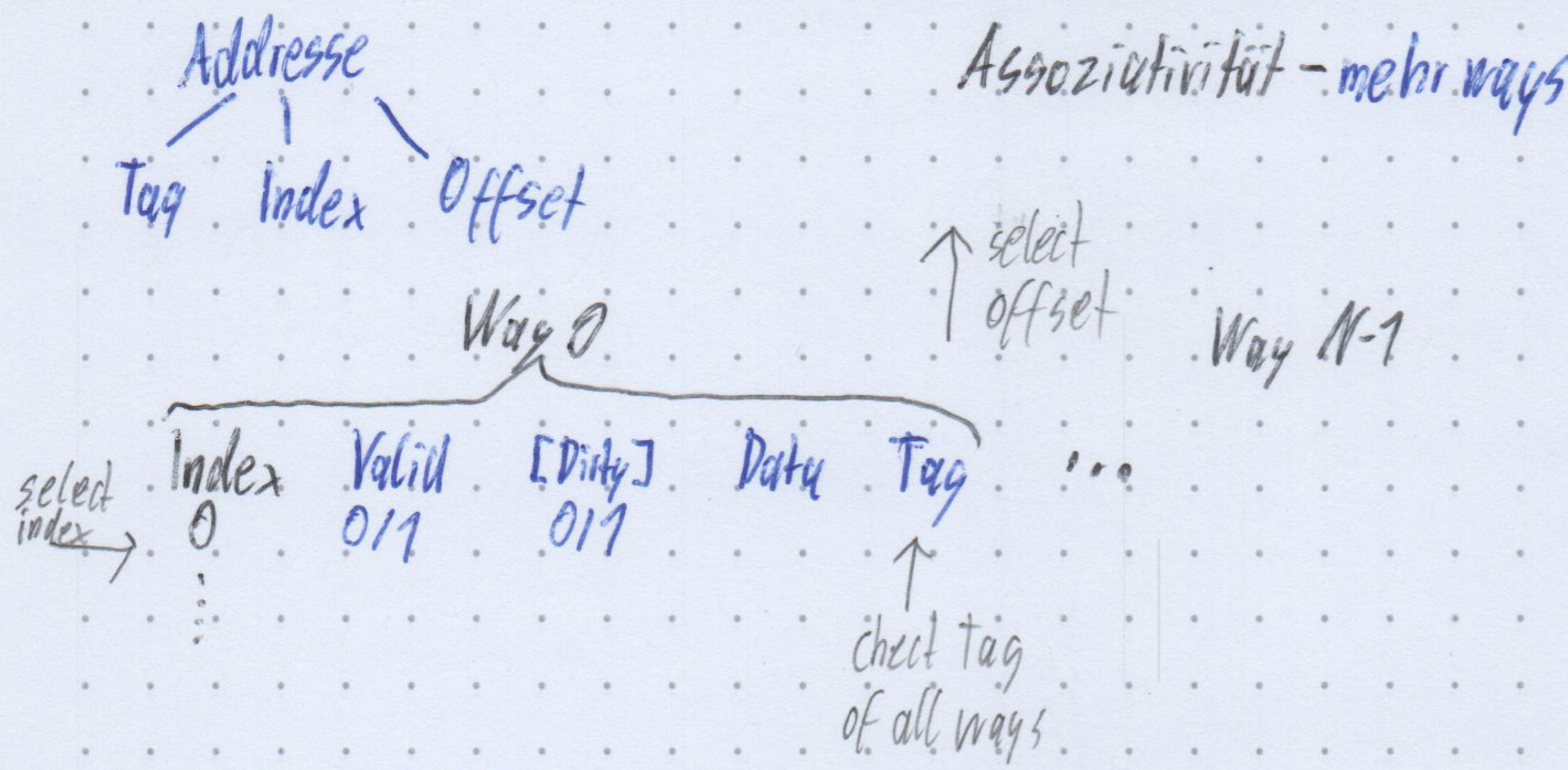
Constrained Random Stimulus

Einschränkungen → gezielte Tests

→ auch mehrere getrennt

- Post-Layout-Simulation

Cache



Types

N-way Set-Associative

- ↳ N-ways
- ↳ > 1 sets

Direct Mapped

- ↳ 1 way
- ↳ 1 set

Fully Associative

- ↳ > 1 ways
- ↳ 1 set

unvermeidbar/
Prefetching (=)

größerer Cache (=)

höhere
Assoziativität (=)

Hit	<ul style="list-style-type: none"> ↳ Daten in Cache ↳ direkte Antwort
Miss	<ul style="list-style-type: none"> ↳ nicht in Cache ↳ Miss Penalty ↳ zusätzlich benötigte Zeit zum Daten holen
Compulsory	<ul style="list-style-type: none"> ↳ Erster Datenzugriff
Capacity	<ul style="list-style-type: none"> ↳ Cache zu klein, verdrängt
Conflict	<ul style="list-style-type: none"> ↳ nur Satz voll

Ergebnis Cache-Zugriff

Eriction

↳ Verdrängung Daten zum Lochen gerade aus
Hauptspeicher geholt

Policy bei > 1 ways

↳ LRU - Least Recently Used

↳ LFU - " Frequently

Schreibzugriffe

erst lesen

aktualisierung Cache
Hauptspeicher nach Write Policy

needed but
may only at end

- ↳ Write Through
- ↳ direkt aktualisieren
- ↳ Write Back
- ↳ set dirty Flag
- ↳ Aktualisierung, wenn nötig
- ↳ später

System on Chip \leftrightarrow System on Board

clock
memory
logic, math, decisions, accelerators
interfaces

三

Massen \leftrightarrow angepasst für wenig
billig | teuer Stück

Rekonfigurierbare System on Chip

e.g. XILINX ZYNQ 7000 R50C

Processor Memory
 [controller]

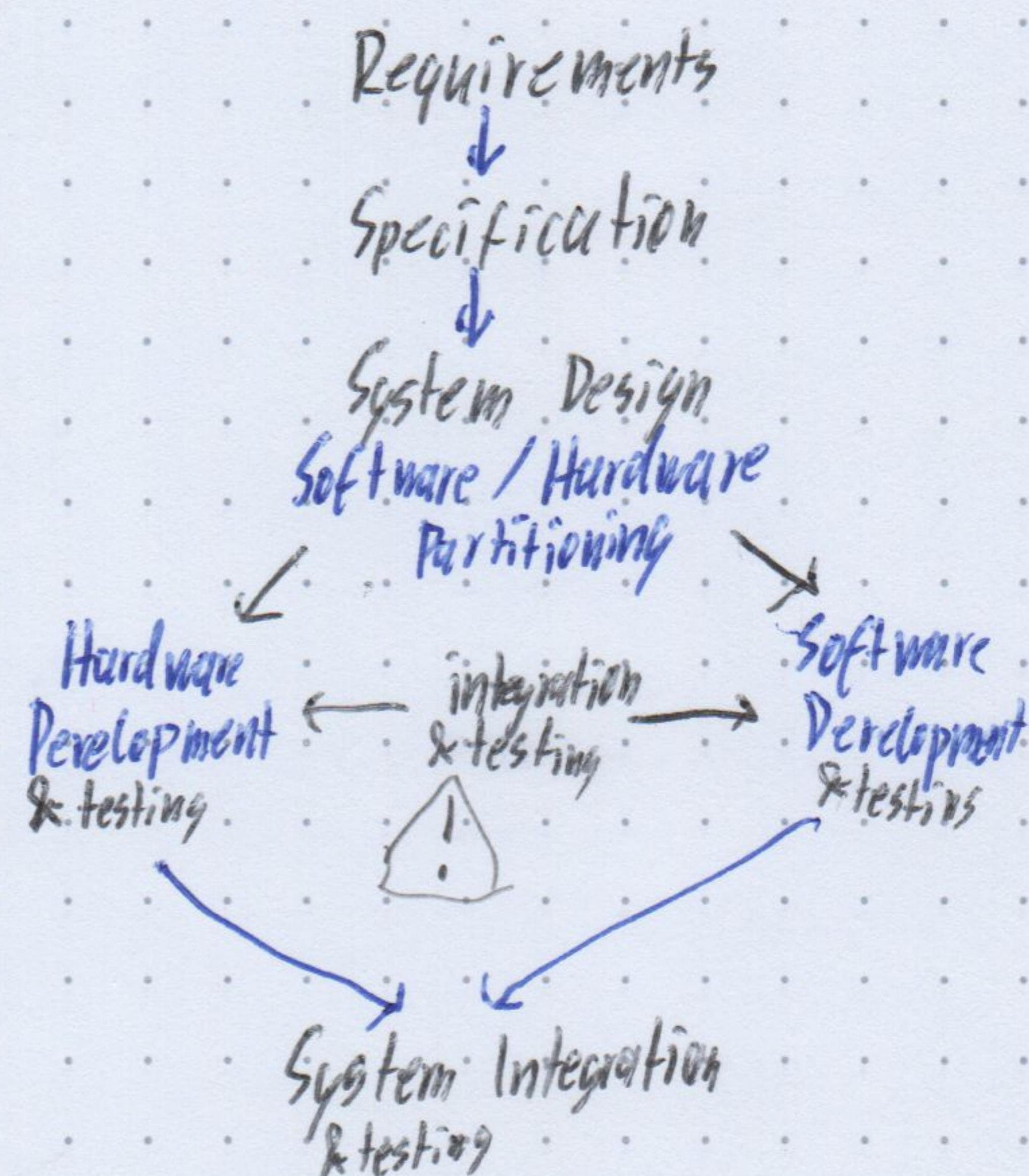
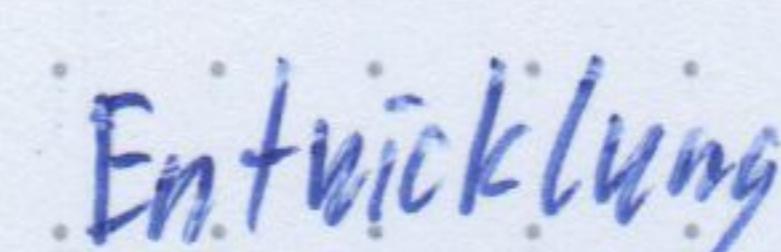
Processing System

System Bus

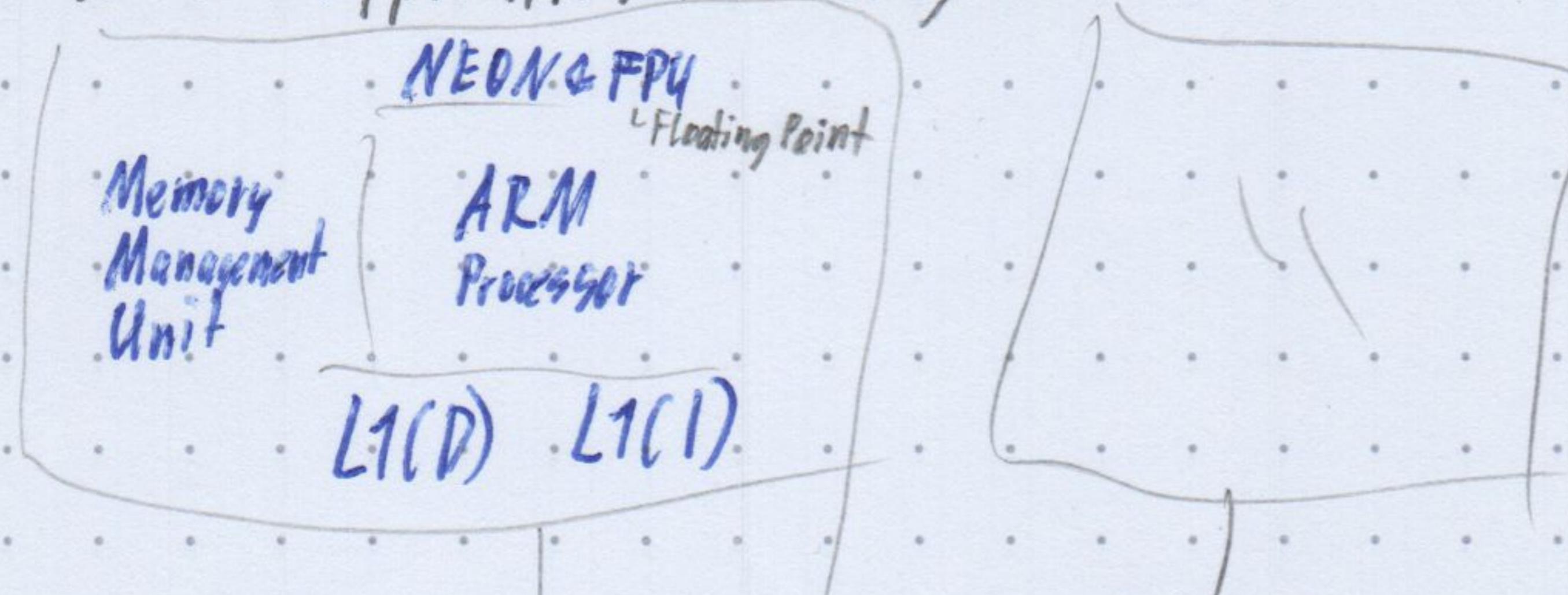
Bridge

Peripheral Bus

Peripheral Peripheral Peripheral



APU : Application Processing Unit



Snoop (control Unit (SCU)) - sorgt für Cache-Kohärenz

ARM Prozessorkerne IPC - Instructions Per Cycle
7 Phasen: F D R I E M W

Fetch Instruction mit Sprungvorhersage cache's

- Decode Instruktionen
- + 2 pro Fakt
- Rename Registers zum Auflösen von Abhängigkeiten
virtuelle \rightarrow physical
- Instruktionen zugeteilt / issue / dispatch
- Umsortierung
- 2. Welle auf den Ausführungsseinheiten

2,111 → bis zu 4 an Ausführungseinheiten

Execute

4 Einheiten:

ALU/Mul

ALU

FPU/NEON

Address

1-3 Takte

bei shift,rotate

SIMD - vector operationen

ints, floats

128bit breit mit 2...16 Words

auch auf skalaren Daten genutzt

Speichersystem

Address Übersetzung virtual-physical.

Micro TLB - 1 Takt Latenz

TLB in MMU - variable Latenz

└ Translation Lookaside Buffer

Memory Management Unit

APU On-Chip Memory (OCM)

- geringe Bandbreite

+ geringe, stabile Latenz

Prefetching

Beobachtet ≤ 8 Datenströme von Prozessor

≤ 4 aktive Anfragen an DRAM

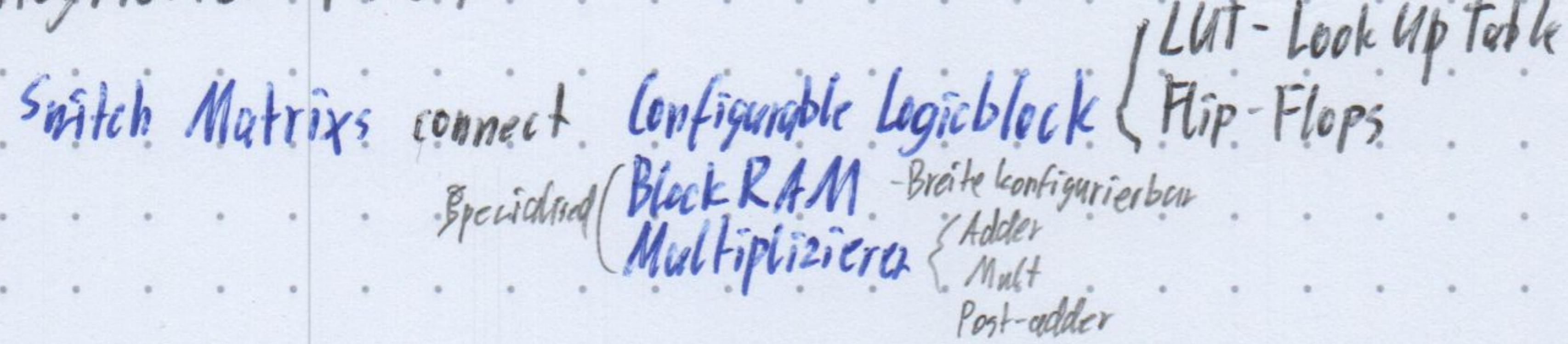
Store-to-Load Forwarding

└ Wenn parallele Anfragen auf gleiche Adresse gehen

Lesezugriff muss nicht auf Schreibzugriff warten

Axi Interface

Integriertes FPGA



Partiell dynamische Rekonfiguration
└ Zur Laufzeit
nur betroffene PL Teile Rest läuft weiter

→ in rekonfigurierbarer Logik

Soft ↔ Hard Cores

└ langsam → lieber Kombi

- Chipfläche

+ spezifisch

Hard Cores, Rekonfigurierbare Logik

Partitionierung

PS - Software

PL - Logic

ZynQ PL keine Speicherschnittstelle

⇒ PL-PS Schnittstellen

Varianten des ARM AMBA AXI Protokolls
Signale für optionale Übertragungsrichtung getrennt

HP-high performance → ACP-accelerator Ports
coherency protocol

über SCU auf L2 cache

+ schnell if cacheid

- Bei zu viel Verdrängt cachelines
verlangsamt CPU

AXI4

- Chip-Fläche
+ mächtig
supports memory-mapped I/O
burst transfers

AXI4-Lite

einfacher
+ memory-mapped I/O
- no burst transfers

geteilt PL-PS

Interconnects

L2 Cache

Speicherkontroller

PL, PS

Master-fähige Peripherie gerät
Teilen Übertragungs durchsatz

AXI4-Stream

- nur Datenströme (kein memory-mapped I/O)
+ unbegrenzt lange bursts
- unidirectional

getrennte Übertragungskanäle

Addressen/Kommandos und Daten AxI-stream verwendet
nur 1 Datenkanal

Master initiiert Slave reagiert

Lese/Schreibzugriff Statusrückmeldung
Lesekanal
extra Rückkanal beim schreiben

mehrere Kommunikationspartner

über Interconnect

agiert gegenüber partnern als Master
oder Slave

wesentliche Signale V@posedge

Prefixe

AW - Address Write

W - Write

B - Response Write

AR - Address Read

R - Read

ACLK

ARESETn

jeweils

Valid

READY

ADDR/DATA

[31:0] [n-1:0]

und

RRESP [1:0]

msc

Handshake:

valid & ready gesetzt ⇒ Daten übernommen

Burst

überträgt nur initiale Adresse, Rest folgeadresse

ARLEN, AWLEN bestimmen # beats / data

Sklare beendet durch setzen RLAST

IP Blöcke und High-Level Synthese

Hardwareentwurfsteiner

IP-Intellectual property blocks/cores

für Standardaufgaben

Standardisierte Schnittstellenbeschreibung

IP XACT

z.B. AXI Interconnect

Dokumentation

Simulationsmodelle

Beispiele

Tools z.B. Xilinx Vivado IP Package

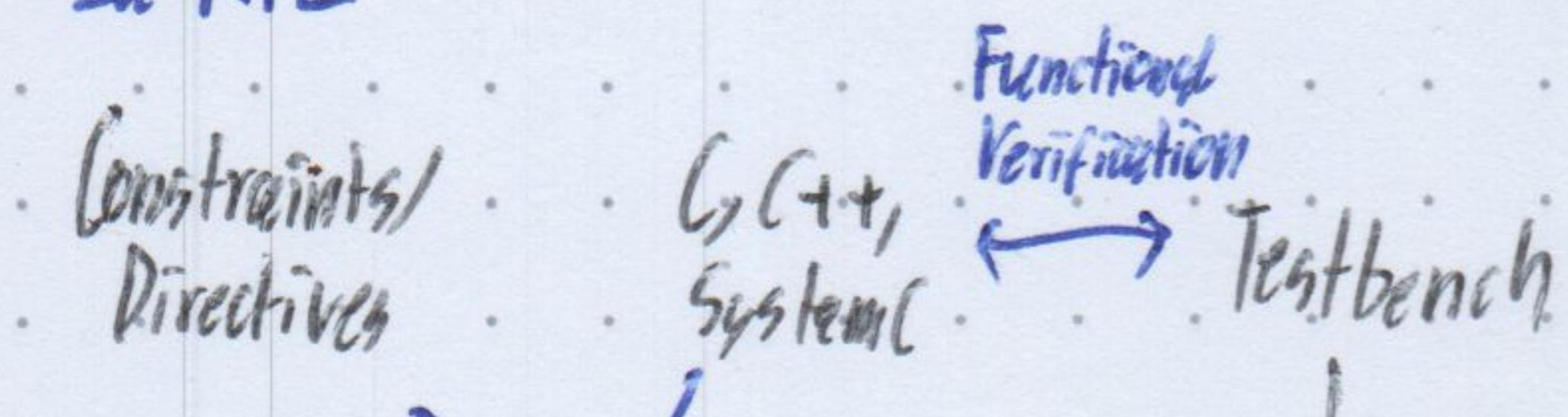
IP Integrator

Blöcke konfigurierbar

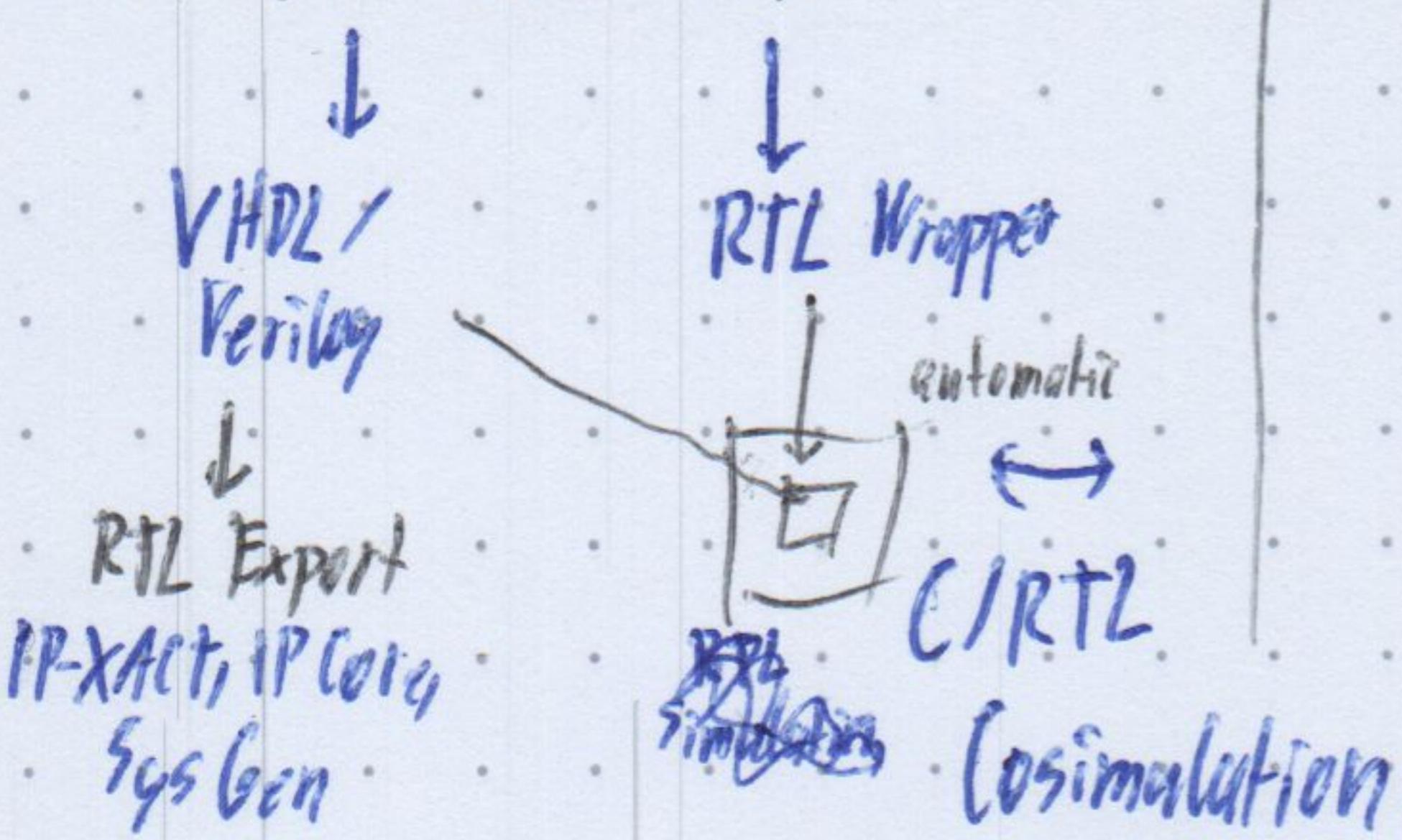
Connection automation zwischen Blöcken

High Level Synthese HLS / Hochsprachensynthese

zu RTL



HLS



Steuersignale auf Blockebene

Befehle

ap_rst - üblicherweise bei Systemreset

ap_start

Statusabfragen

ap_idle

ap_done

ap_ready - z.B. bei Pipelining,
auch wenn Block noch
beschäftigt

Algorithmensynthese User Directives, Technology Library

Scheduling

Op ~ Zeitschritte

Binding

scheduled Op ~ Hardware Resources

Implementierungsroutine

Ressource sharing

„soviel wie möglich“ → wiederholen

Operator Chaining

in einen Takt, mehr Hardware, Delay klein gestellt

spezialisierte Hardare

stark limitiert zur Verfügung

Interface-Synthese mit restlichen RTL Schaltungen

Erfordert Entwicklerhintergrund

z.B. #pragma HLS INTERFACE ap_type port=attribute name=

ap_none

keine weiteren Signale

Annahmen:

Input, liegt rechtzeitig an
& während Ausführung

ap_memory

1 Element großer Eingang, numeriert q0

Address ausgang

- address

Clock Ensemble für spezif.

- ce

ap_rld

Output <name>

valid Output <name>_rld

Annahme

Output nach variabler Zeit

Moore's Law Observation → imperative, death 2017 - abgefeiert von

transistors doubles 2 years
computing power 18 months

multi-core, parallel, distributed

specialized hardware

no clear winner, heterogeneous architectures

Pa Herson's Walls if circumvent 2, hit 3.

Brick Wall =

Power Wall

Lumped Circuit Abstraction to capacitors

frequency needs currents limited by power
heat dissipation

+ Memory Wall

program = filter input → output
side effects

access latency

transfer speed

+ Instruction-Level Parallelism Wall

non-contentious hardware resource requirements
more hardware ⇒ frequency drop

no silver bullet

+ flexibility

Commodity ISA

μController - limited scope
z.B. Arduino

SoC - desktop and server class

Low-Power CPU - desktop class

+ standard ISA

⇒ extensive tool support
language

COTS - commercial

of the
shelf

performance
power
development costs

NRE - non
recurring
costs

/ developers

time dev. iterations

scaleability

Specialized ISA

General Purpose GPU

Digital Signal Processor

Many Core

- only

- vendor tools

specialized languages

~ months, years

amortize 5-10 years

problem change little,
stay relevant

ASIC - application specific integrated circuit

- full custom HDL

⇒ long development performance
unflexible low power

~ months

already used in clusters such. amazon cloud

firms bought by intel, amd

Reconfigurable Technology

Programmable logic device

Field programmable gate array

- full custom

only vendor tools

+ flexible

shorter development than ASICs
but slower and less efficient

FPGA Workflow

hardware bring-up

board

select ↔ build

large variation

on-chip - FPGA itself

area, slice composition
addressing diff. applications

architectural family

ressources = hard blocks

BRAM, DSP, ...

speed grade
other:
energy
space hardening

off-chip - periphery

connectivity Host/outside world

e.g. PCIe, Network, shared memory

memories, caches

secondary peripherals

Displays, Video, Sensors

major issues

Portability

hardware choice

influences everything else

change \Rightarrow repetition

Scalability

base design

base design

with default connections

controls pin-connections

on logical level

where pin connected?

pin constraints

adapt from vendor

↔ build for custom chip

X time consuming

logic design

develop custom logic

usually in HDL

in theory target independent

but scalability & portability issues

simulate

behavioral on RTL level

\approx unit tests \Rightarrow confidence in correctness

Hardware synthesis

Place and route

15 min to days

NP-complete

\Rightarrow heuristic approach with iterative backtracking
simulated annealing
analytic methods

place: op = hardware resources

route: wiring; check time constraints

Synthesis

to netlist

[primitive instances
nets]

X time consuming

Exercise on real Hardware

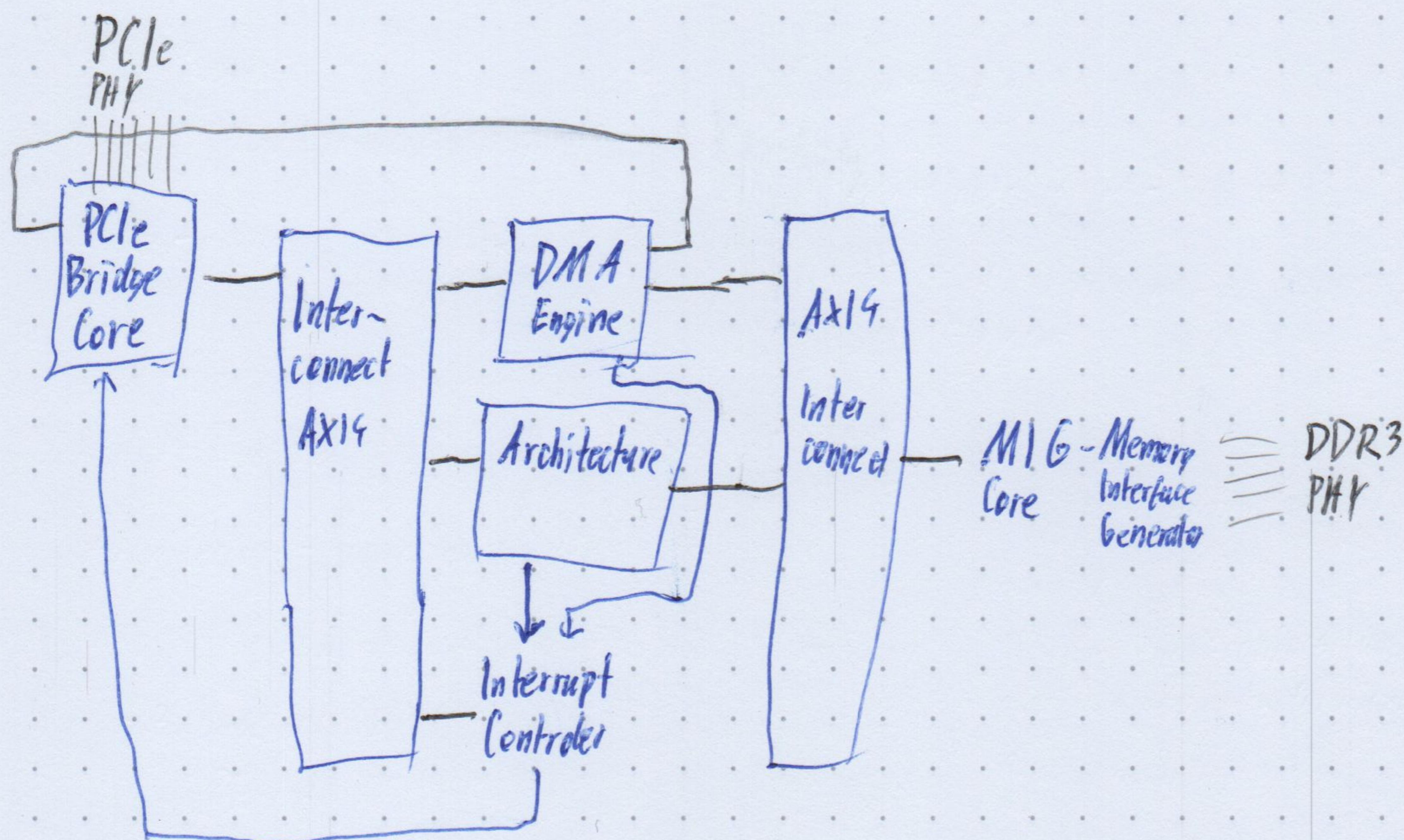
only here uncertainties resolved
e.g. OS interactions

X time consuming

Platform

Tasks:

- control connection
- memory access
- signaling interface to host
- [instantiate hardware infrastructure]



Software Stack

Device driver - Ring 0

onchip physical address space → global/bus address space
virtuell?

API

C - full access for corner cases

Rust

(++ Initialization `tupasco()`;

Device Memory Management

`tupasco::alloc`

`free`

`copy-to`

`copy-from`

`Tupasco(tupasco())`; high-level task management

`auto wrapped = makeWrappedPointer(in, size);` // for host memory region, auto copy to and from device

`auto in-buff = makeInOnly(wrapped);` // dont copy back; make `OutOnly` analog

`int sum = -1`

`RetVal<int>= ret-val(&sum);` // only for single value

`auto job = tupasco.launch(PE-ID, ret-val, in-buff);` // optional arguments: ~ simple values
job(); // awaits
return sum;