# Hardware Modelling

## Overview

*ECS Group, TU Wien*

# Outline

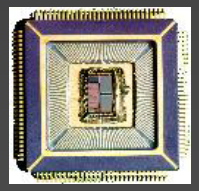
- Difference: Hardware vs. Software
- Introduction to Hardware Modelling
  - Specification
  - Realisation
  - Verification

# Hardware vs. Software: Design Flow

## C-Code

```
Function counter(int a, int b)
{
  Begin
    if a=b then
       count:= 0
    else
```

Development tools

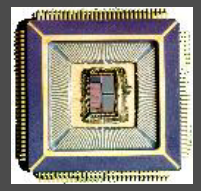
Binary code 

Memory is loaded, hardware is not changed

## VHDL-Code

```
Architecture rtl of Counter is

Begin
  if a=b then
     count<= 0
  else
```

Development tools

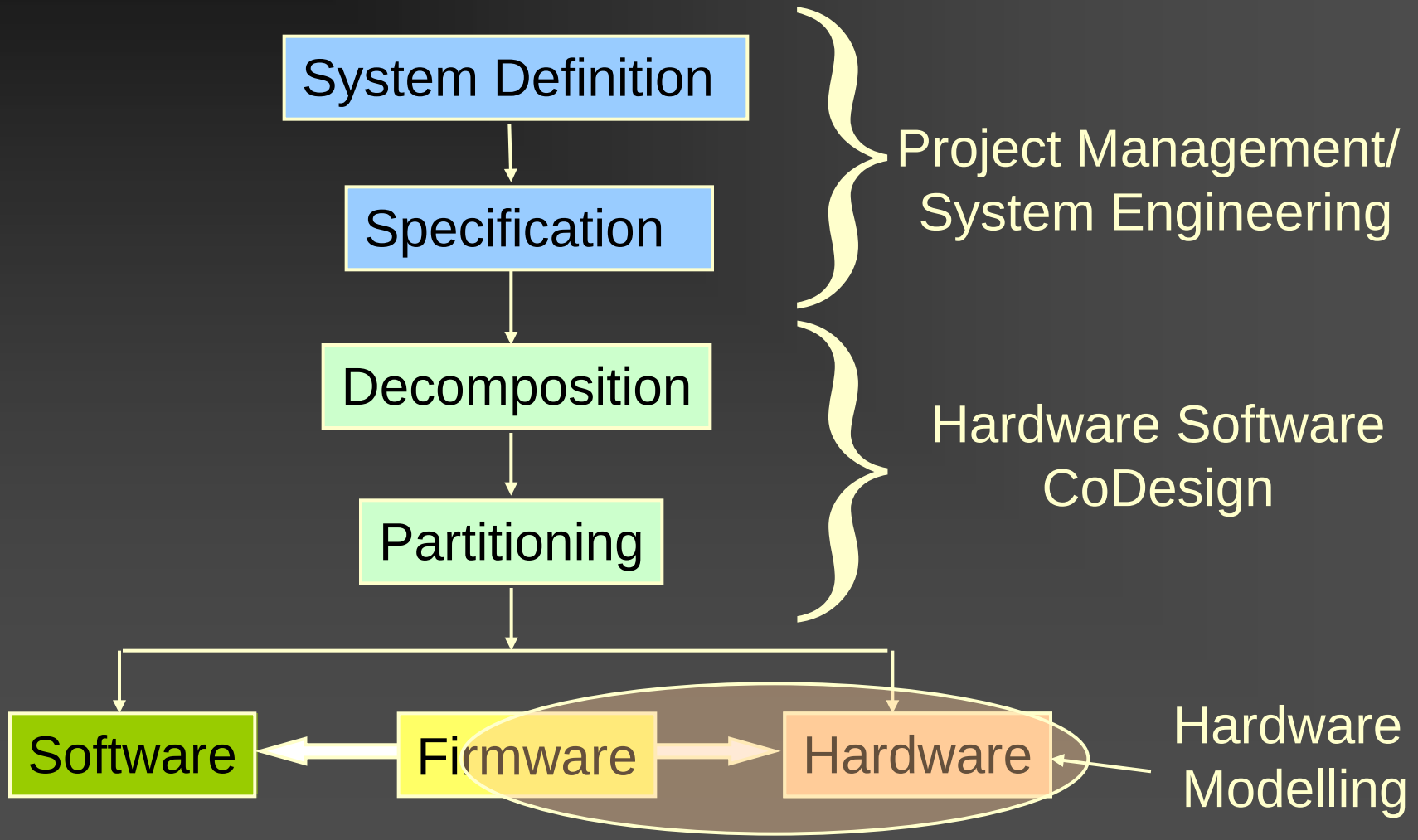
edif - file 

Gates and connections are configured no software is running on the device

# Hardware vs. Software: Advantages and Drawbacks

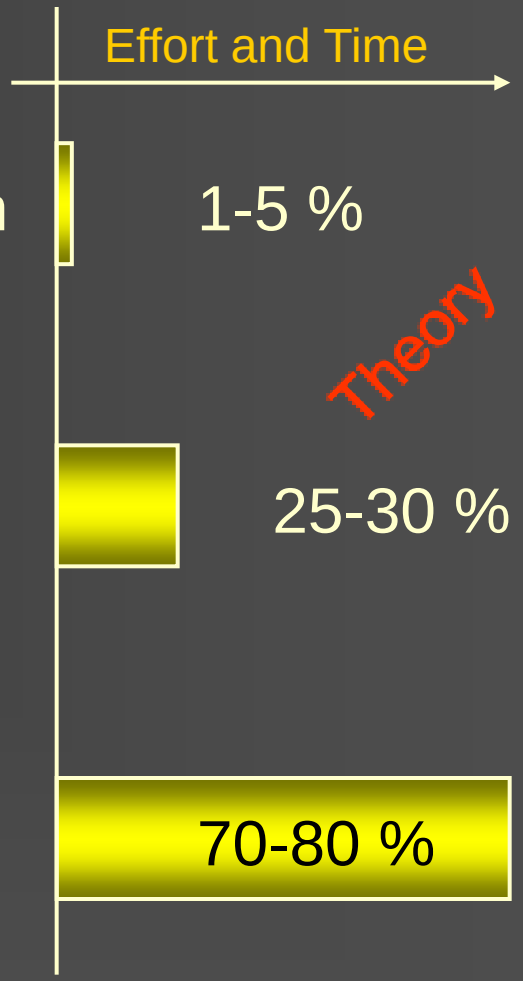
| | Software | Hardware |
|---|---|---|
| Complexity | +++ | --- |
| Performance | --- | +++ |
| Flexibility | +++ | + |
| Effort | ++ | ++ |
| Customization | + | +++ |

# Embedded Systems Design Flow



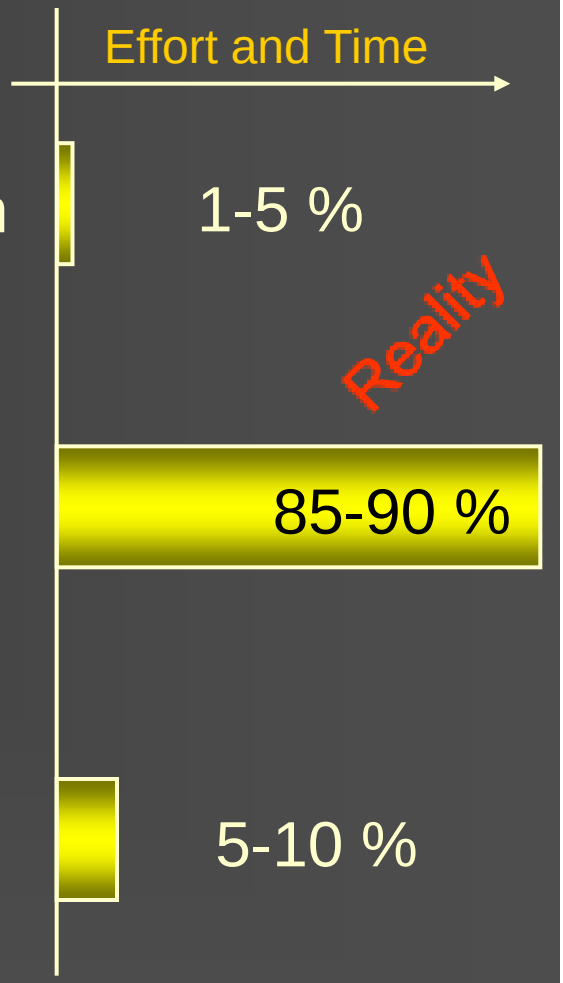
# Hardware Modeling

- Specification
  - Functional specification
  - High-level requirements description
  - Detailed design description
- Realisation
  - Hardware description
  - Hardware implementation
- Verification
  - Review
  - Formal verification
  - Functional verification



# Hardware Modeling

- Specification
  - Functional specification
  - High-level requirements description
  - Detailed design description
- Realisation
  - Hardware description
  - Hardware implementation
- Verification
  - Review
  - Formal verification
  - Functional verification



# Hardware Modeling

- Specification
  - Functional specification
  - High-level design description
  - Detailed design description
- Realisation
  - Hardware description
  - Hardware implementation
- Verification
  - Review
  - Formal verification
  - Functional verification

# Specification (1)

- Functional description
  - Operational principle
  - User interface
  - Transaction level description
  - …

⇒ Project proposal / project description

# Specification (2)

- High-level design description
  - Requirement specification
  - Decomposition in submodules
  - Interface definition
    - Physical Interfaces (Module ports, pins, etc).
    - Logical Interfaces (data types, e.g.)
    - Behavioral Interfaces ( reactions on a stimulus)
    - ⇒ independent implementation of submodules possible
  - Testcase specification

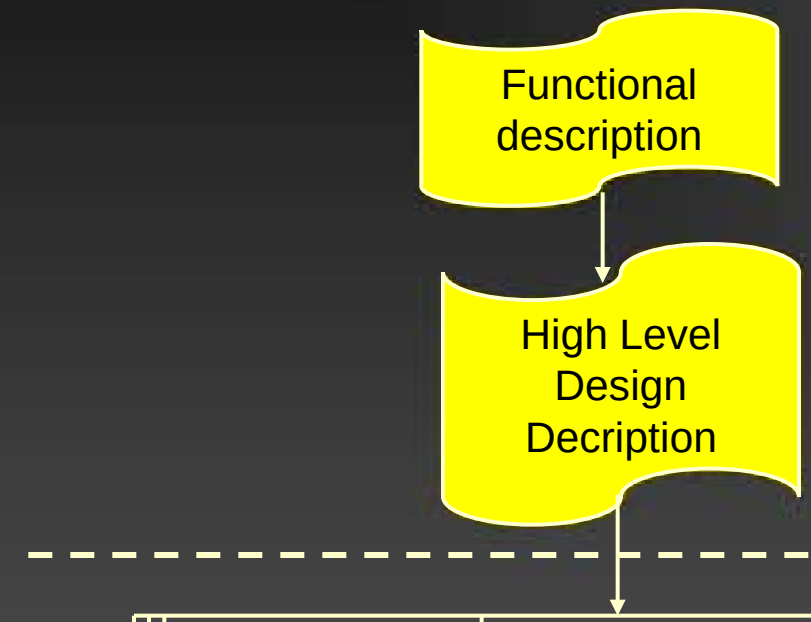
⇒ Reference document for all implementations

# Specification (3)

- Detailed Design Description
  - Refinement of the submodules
  - Detailed description of structure and functionality
  - References to requirements
    - Traceable decisions
  - One possible implementation of a submodule

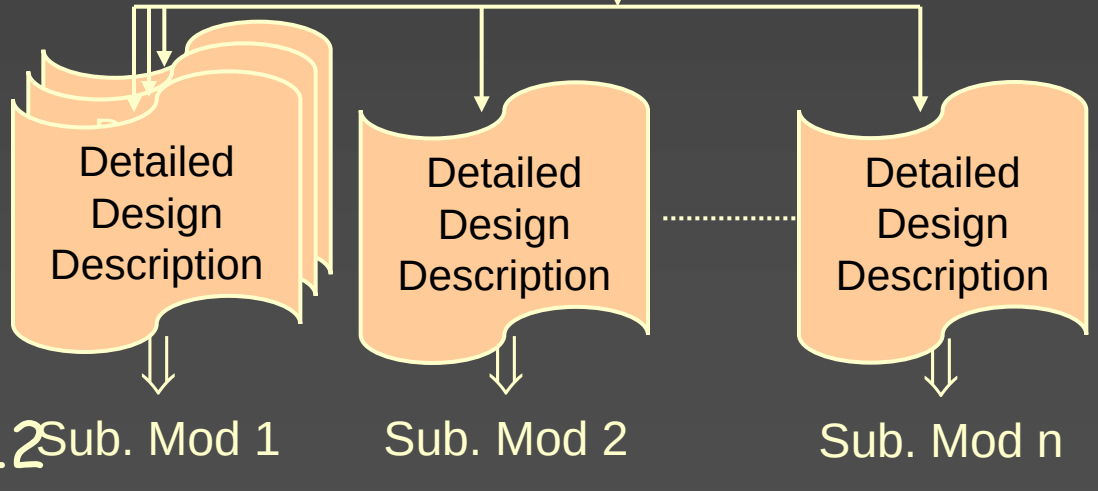
$\Rightarrow$ Technical manual for implementation

# Specification - Summary




⇒ Incomplete specification

**Common reference**
⇒ Complete specification
⇒ Submod & IF definition




⇒ One possible implementation
⇒ Requirements must be fulfilled

# Hardware Modeling

- Specification
  - Functional specification
  - High level requirement description
  - Detailed design description
- Realisation
  - Hardware description
  - Hardware implementation
- Verification
  - Review
  - Formal verification
  - Functional verification

# Realisation

- Hardware Description
  - Design entry
  - Levels of abstraction
- Hardware Implementation
  - Synthesis
  - Technology mapping
  - Place and Route

# Realisation

- Hardware Description
  - Design entry
  - Levels of abstraction
- Hardware Implementation
  - Synthesis
  - Technology mapping
  - Place and Route

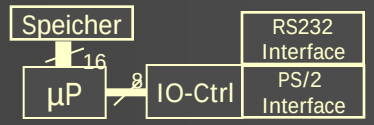
# Hardware Description: Design Entry(1)

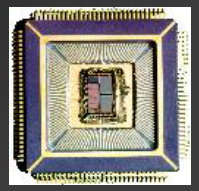
Design Appraoch

## Behaviour

```
while input
    Read „Schilling“
    Calulate Euro
    Display  „Euro“
```

## Structure



## Geometry


⇒ different views describe the same chip → Y-diagram

# Hardware Description: Design Entry (2)

## Schematic Entry

- Gates (AND, OR, INV, FF,… )
- Modules (ALU, Register, Decoder,…)
- IP Core (ProcessorCore, USB Interface)

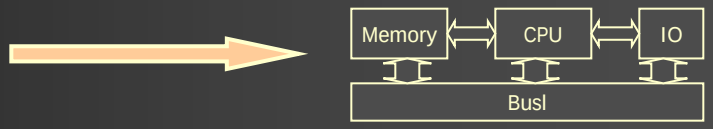
## Text-based Entry

- Boolean equations
- VHDL, Verilog
- SystemC, SystemVerilog

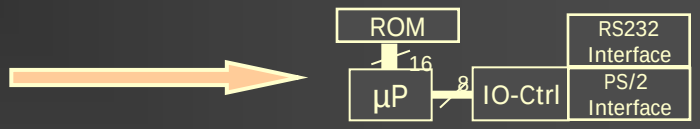
# Hardware Description: Design Entry (3)

## Levels of abstraction

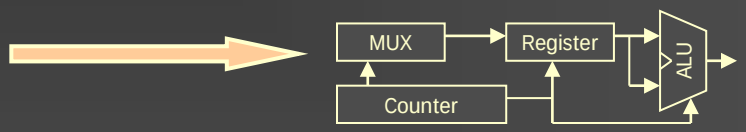
System level



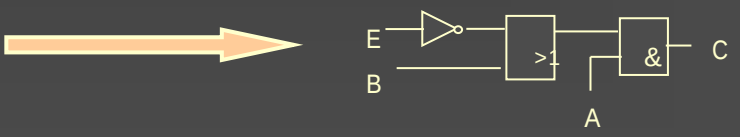

Algorithmic level



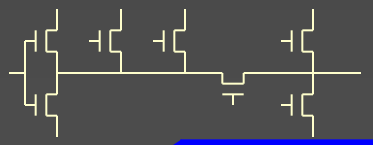

Register Transfer level




Gate level




Transistor level



# Realisation

- Hardware Description
  - Design entry
  - Level of abstraction

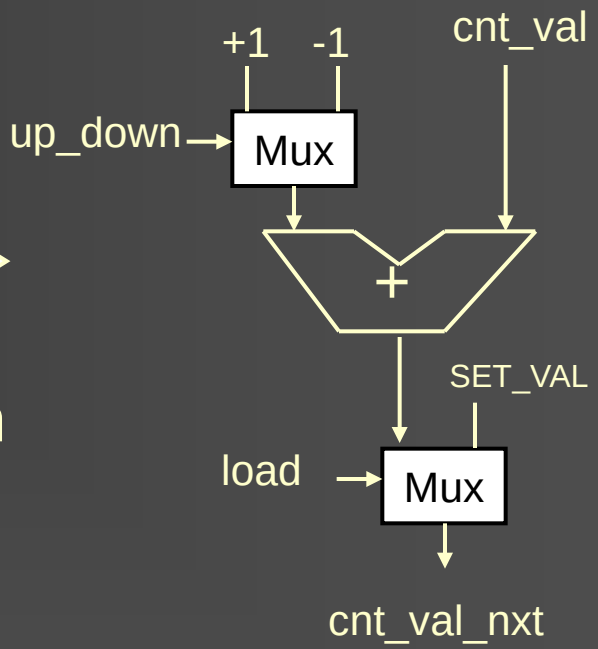
- Hardware Implementation
  - Synthesis
  - Technology mapping
  - Place and Route

# Hardware Implementation - Synthesis

Mapping of a hardware description to hardware components

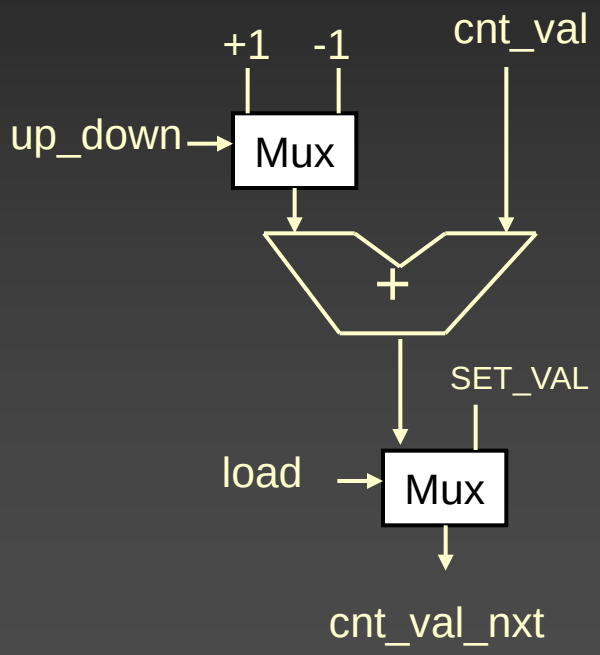
```
cnt_ntx: process (load,up_down, cnt_val)
begin

cnt_val_nxt <= cnt_val;

if load ='1' then
   cnt_val_nxt <= SET_VAL;
elsif up_down = '1'  then
   cnt_val_nxt <= cnt_val + 1;
 else
    cnt_val_nxt <= cnt_val - 1;
 end if;

end process cnt_ntx;
```

Transformation

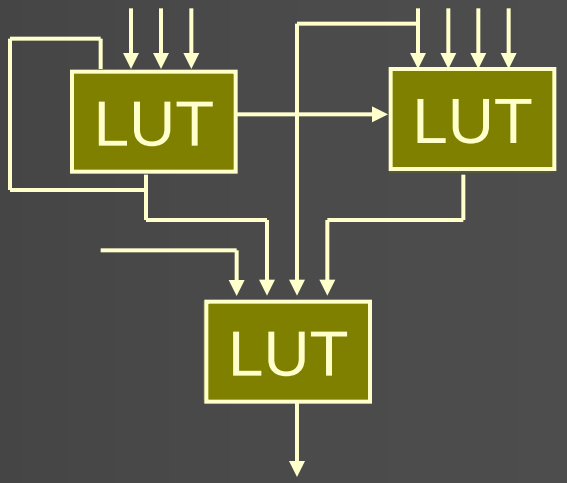


# Hardware Implementation - Technology mapping

Mapping of hardware components to the selected library

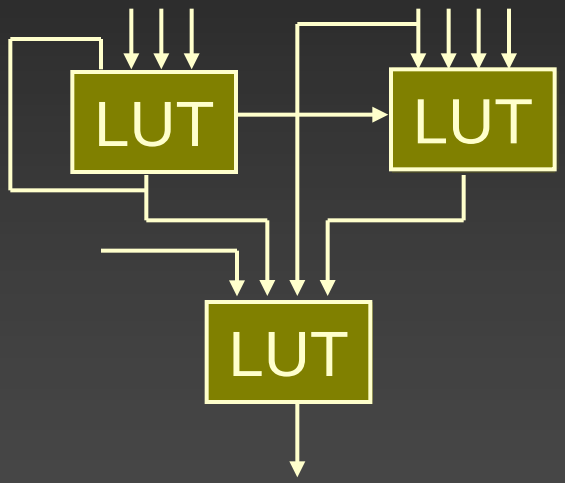



Technology Mapping

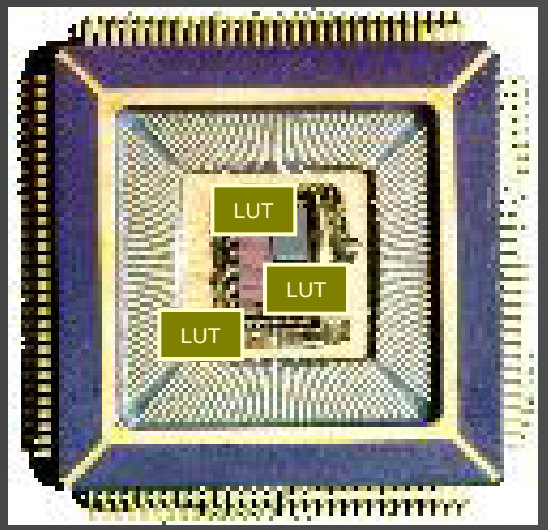


# Hardware Implementation – Place and Route

Mapping of library elements to the real chip




Place and Routing



# Hardware Modelling

- Specification
  - Functional specification
  - High level requirement description
  - Detailed design description
- Realisation
  - Hardware description
  - Hardware implementation
- Verification
  - Review
  - Formal verification
  - Functional verification

# Verification

- Review
  - High level of abstraction
- Formal verification
  - Equivalence checking
  - Model checking
- Functional Verification
  - Simulations
  - Prototype

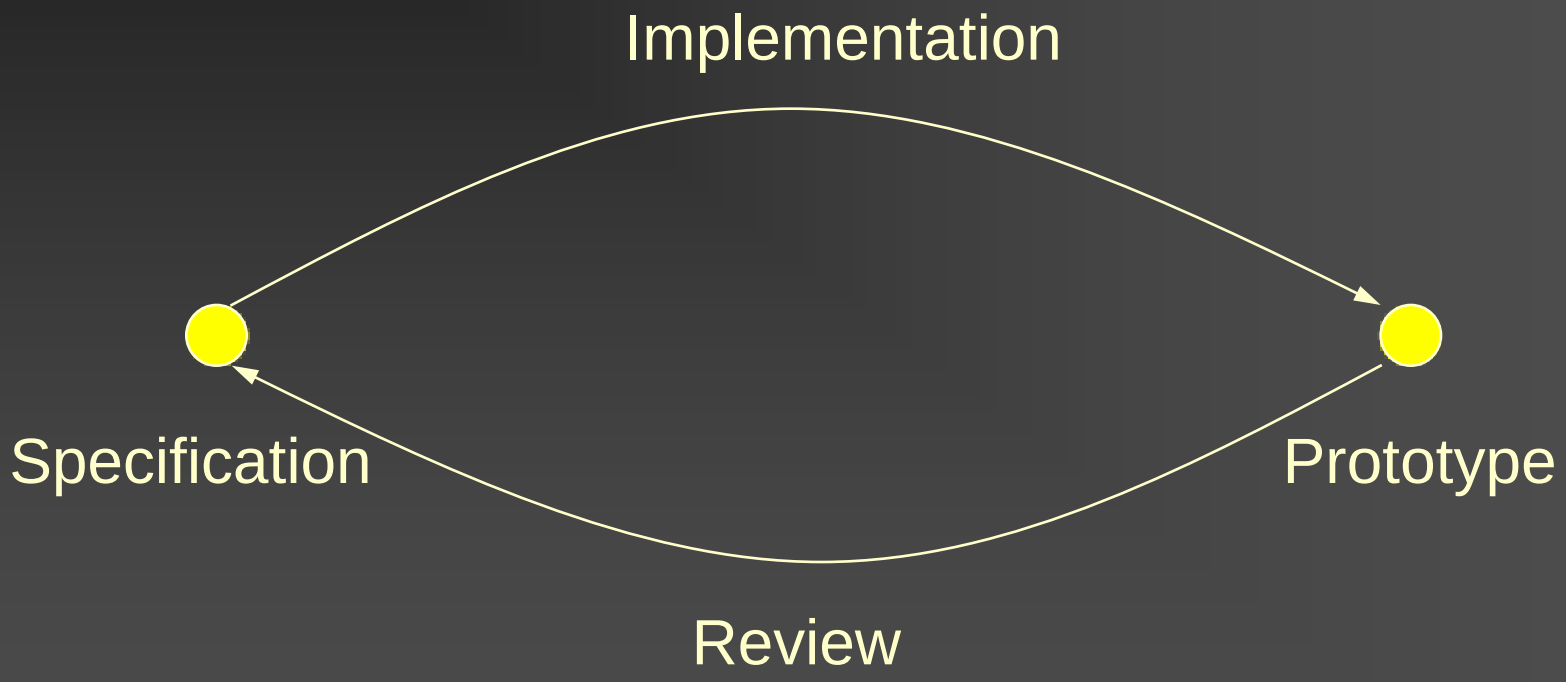
# Verification

- Review
  - High levels of abstraction
- Formal verification
  - Equivalence Checking
  - Model checking
- Functional Verification
  - Simulations
  - Prototype

# Review (1)

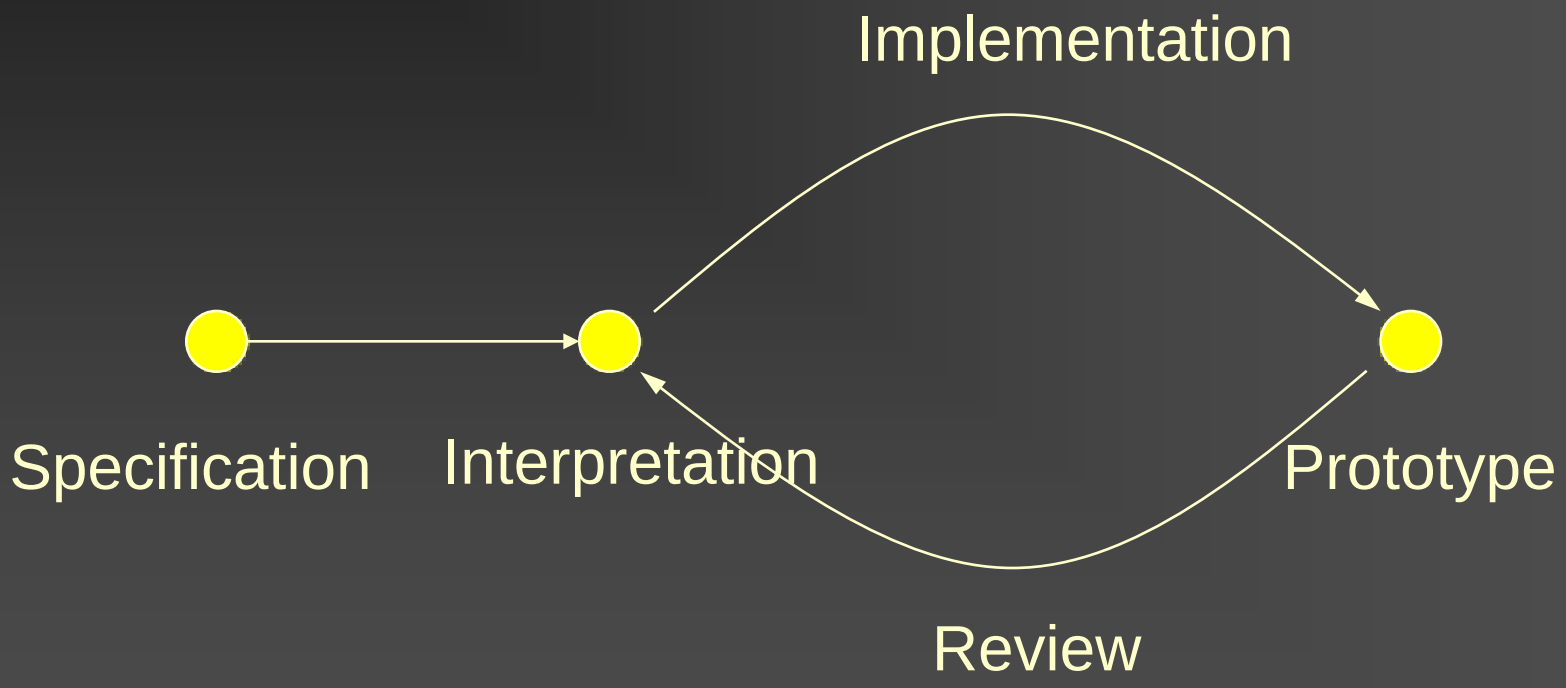
- Check for correctness
- High level of abstraction
- Usually not automated
- Code review by colleagues

# Review (2)



# Review (3)

## The human factor



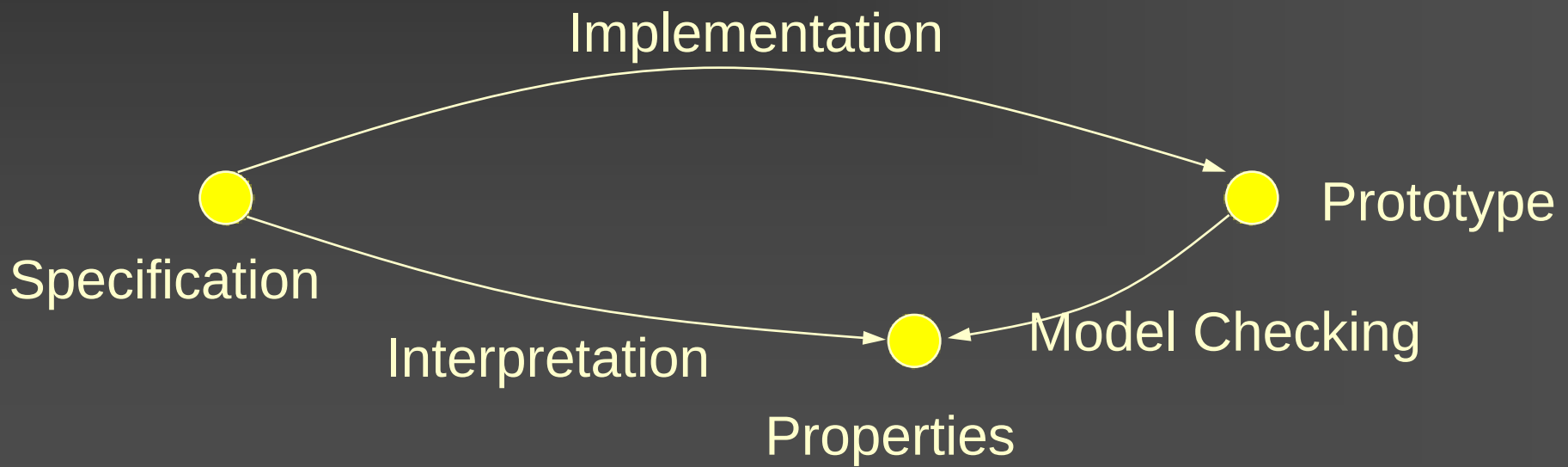

⇒ designer reviews design against its own interpretation, not against the specification

# Verification

- Review
  - High levels of abstraction
- Formal verification
  - Model checking
  - Equivalence Checking
- Functional Verification
  - Simulations
  - Prototype

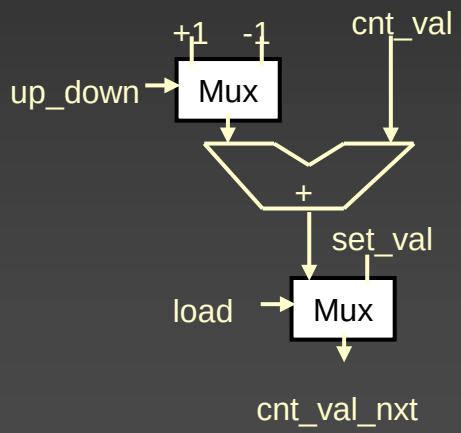
# Formal Verification (1)

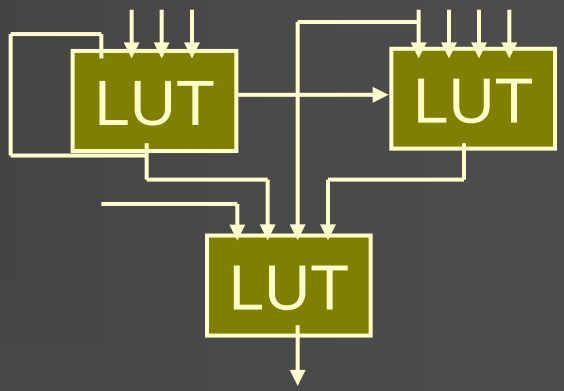
- Model Checking
  - Identifies generic problems
  - Violation of user defined rules



# Formal Verification (2)

- Equivalence check
  - Compares to two models




generic implementation

technology specific gates

⇒ same functionality, but different models

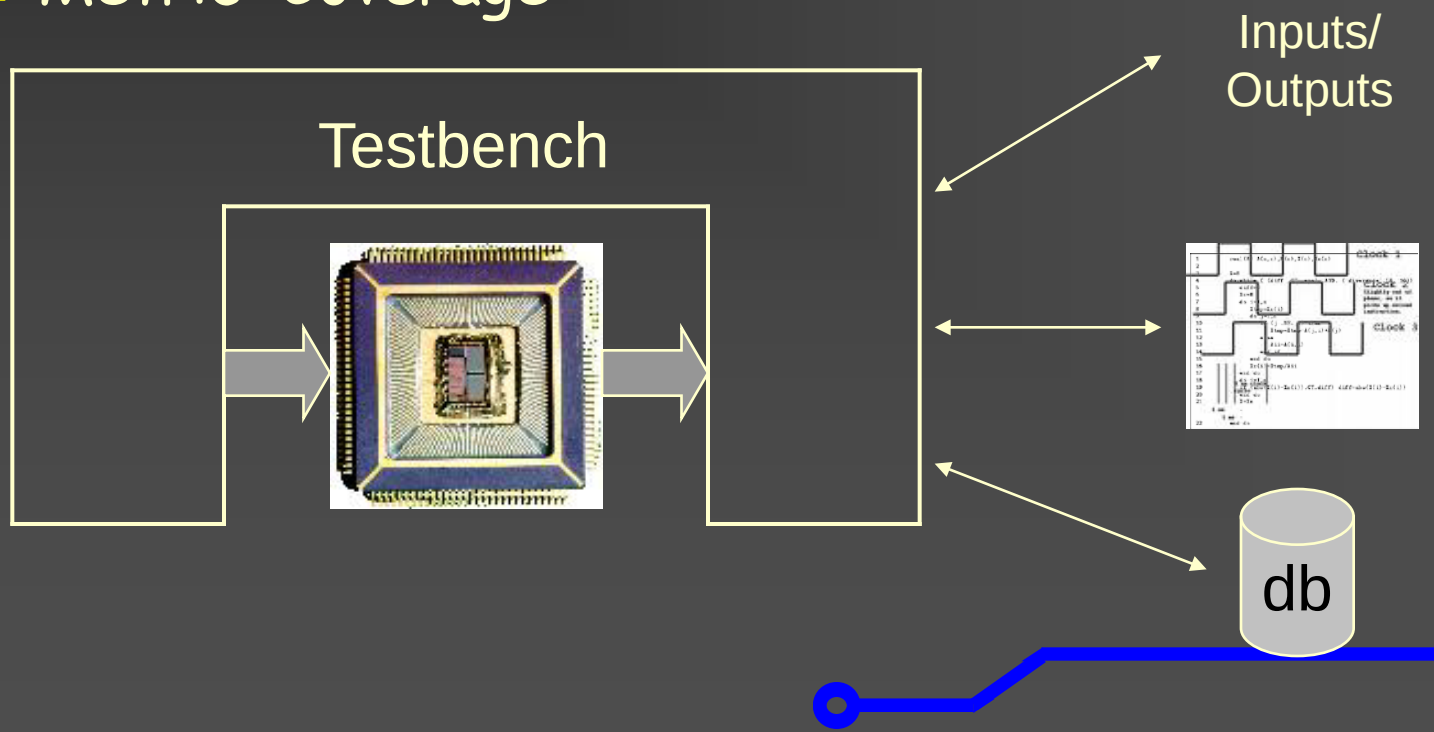
# Verification

- Review
  - High levels of abstraction
- Formal verification
  - Equivalence Checking
  - Model checking
- Functional Verification
  - Simulations
  - Prototype

# Simulation (1)

Testbench

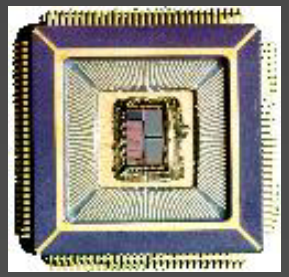
- Generate stimuli, emulate testcases
- Record / check response
- Metric: Coverage



# Simulation (2)

- Different abstraction levels
  - Behavioural simulation
  - Functional simulation
  - Prelayout simulation
  - Postlayout simulation

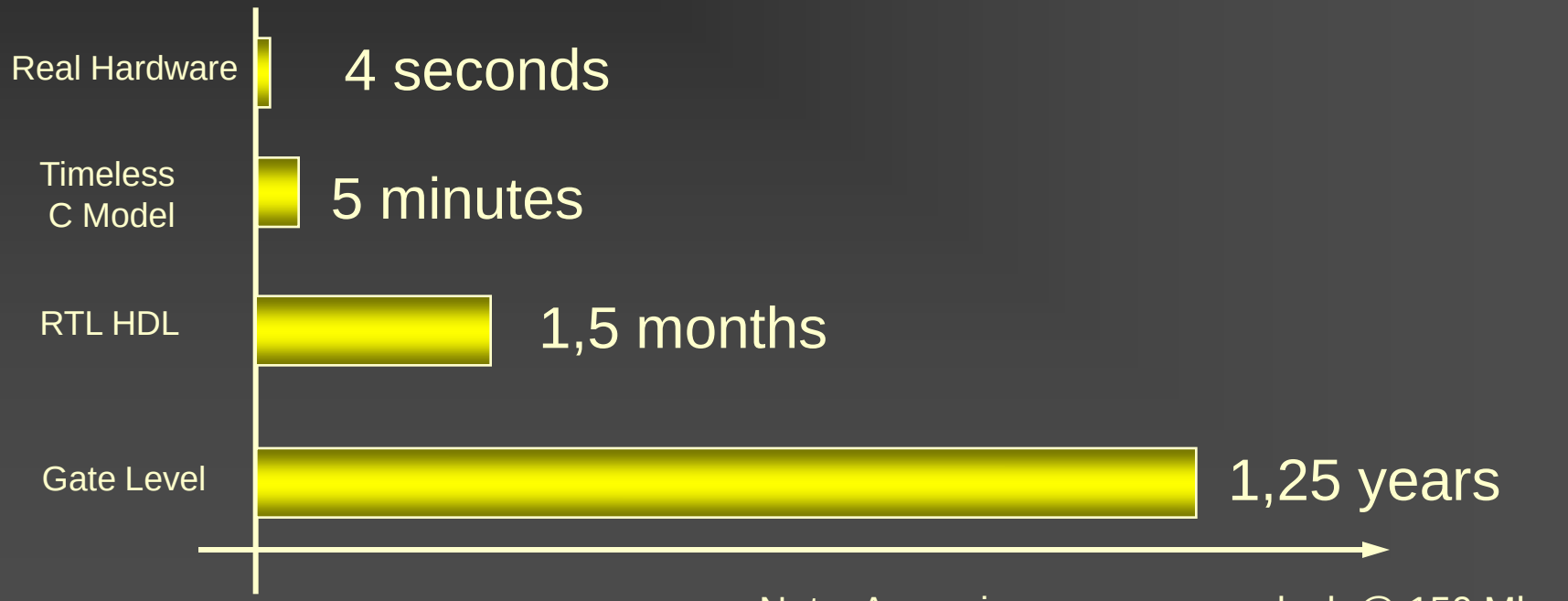
Refinement



# Simulation (3)

Trade-off speed vs. accurancy

- Simulation time for a SoC




Note: Assuming a processor clock @ 150 Mhz

# Content of this course

- Hardware Specification
  - Functional specification
  - High Level Requirements
  - Detailed Design Description

  → • Exercise

- Realisation
  - Hardware Description

    → • VHDL
  - Hardware Implementation

    → • FPGA Design Flow

- Verification
  - Review
  - Formal verification
  - Functional verification

    → • Testbench
    • Coverage