High-Level Decision Diagrams based Verification with PSL Assertions

(Previous: *Automated Reasoning for Hardware Verification*, A.Sudnitsõn)

→ Maksim JENIHHIN
Jaan RAIK
Alexander SUDNITSÕN

1918
TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Outline

- Introduction
- Formal verification
- Simulation-based verification
- Assertion based verification (PSL, the paper)

Estonia

- ~ 45,000 km2
- ~ *1,300,000* inhabitants
- capital: Tallinn (~400,000)

- Summer:
  avr: 20º C / 70º F
  max: (35º C / 95º F)
  day ~ 18 hours

- Winter:
  avr: -10º C / 15º F
  min: (-35º C / -30º F)
  day ~ 6 hours

Tallinn University of Technology

TTÜ1918

- The only technical university in Estonia
- 12,000 students
- 25% are at IT faculty
- IT Faculty consists of

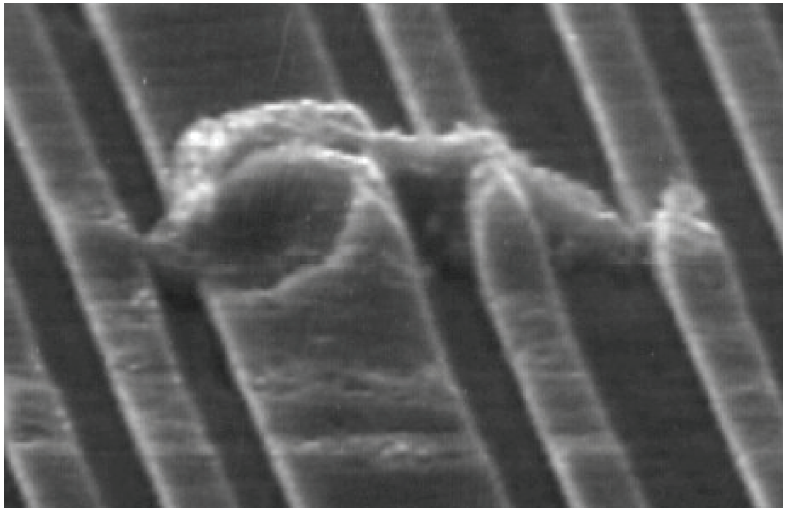
  6 Departments:
  - » Computer Engineering
  - » Computer Science
  - » Electronics
  - » Computer Control
  - » Informatics and Radio
  - » Communication Engineering

Dept. of Computer Engineering

- 32 employees: 5 professors, 6 associate professors, 10 researchers, 10 PhD students
- Participation in R&D projects:
  - 8 EU projects since 1993
  - Bilateral research projects with Sweden, Poland, Germany
- Around 80 papers published annually
- Research topics:
  - Hardware design (SoC, NoC, FPGA)
  - Hardware verification (static / dynamic)
  - Hardware testing (BIST, Boundary Scan,etc)

What is verification?

- **Verification** is checking if the circuit was designed correctly
- **Validation** is similar to verification but it is performed on physical prototype
- **Testing** is checking every manufactured circuit for its correctness (absence of manufacturing defects)



The scope

- Digital circuits (i.e ASIC, SoC)
  - » not software!
  - » not analog, RF, mixed-signal!
- Functional verification of a model (i.e. .vhdl)

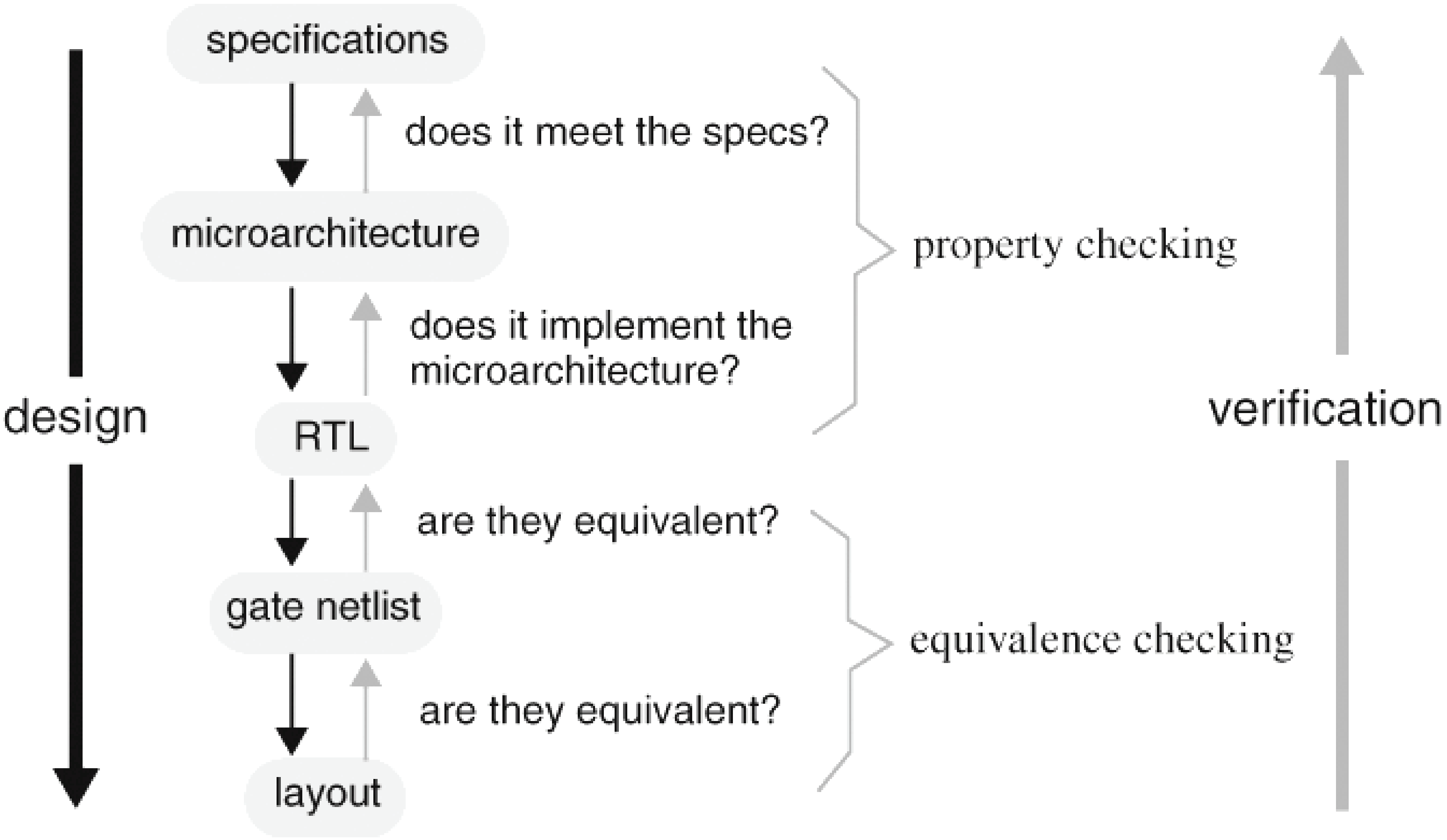
Motivation

- Verification takes roughly 70-85% of design costs
- Some companies have 2-4 verification engineers for every design engineer
- A need to increase verification effectiveness
  - Design-for-Verifiability (DFV)
  - Assertion-based Verification (ABV)

International Technology Roadmap for Semiconductors report

http://www.itrs.net

Introduction to hardware verification 1



specifications

microarchitecture

RTL

gate netlist

layout

does it meet the specs?

does it implement the microarchitecture?

are they equivalent?

are they equivalent?

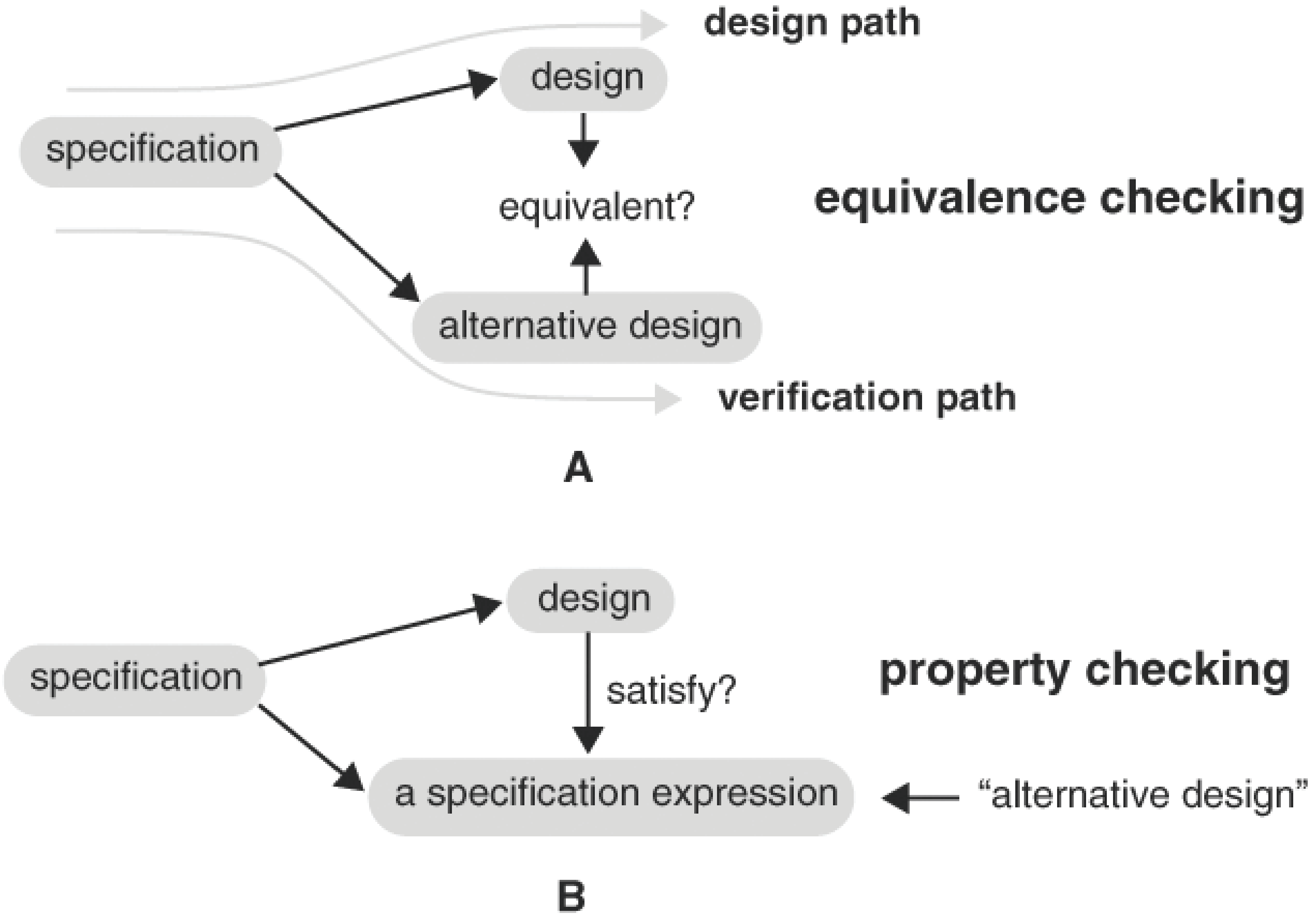
property checking

equivalence checking

design

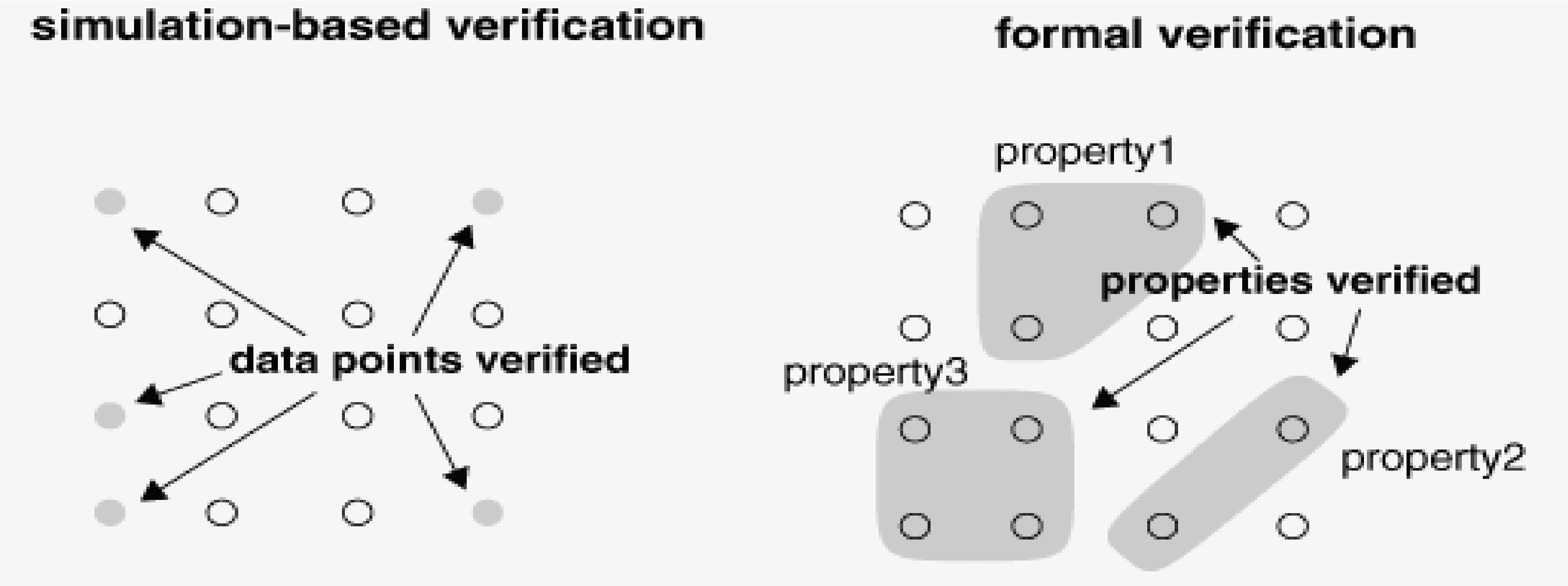
verification

Introduction to hardware verification 2



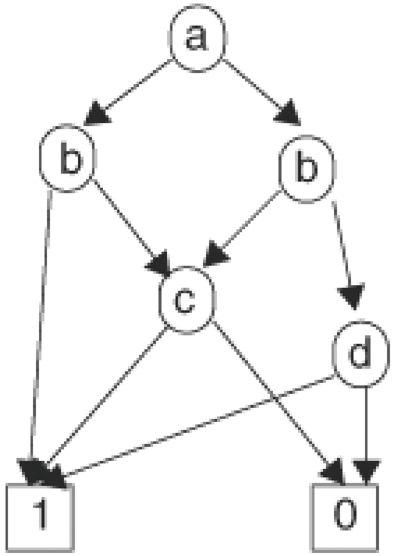
Formal vs. Simulation-based verification

- Formal verification (*static*)
  - Intelligent (mathematical) proof of correctness
  - Constrained application
- Simulation-based (dynamic)
  - Simulation of input vectors (random or deterministic)
  - The most commonly used

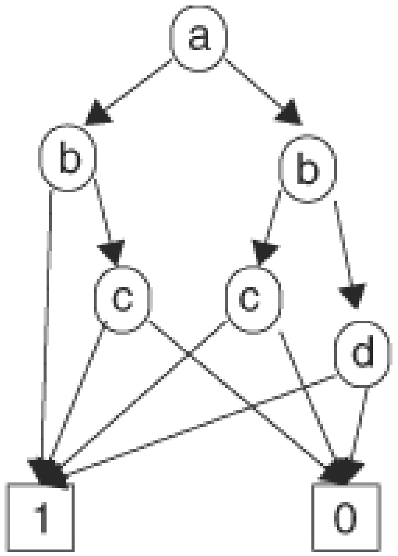


Formal verification: Equivalence checking 1

- Decision Diagrams canonical form
- The idea:
  - Construct DD for the two circuits to be compared
  - Manipulate them to prove that they are equivalent



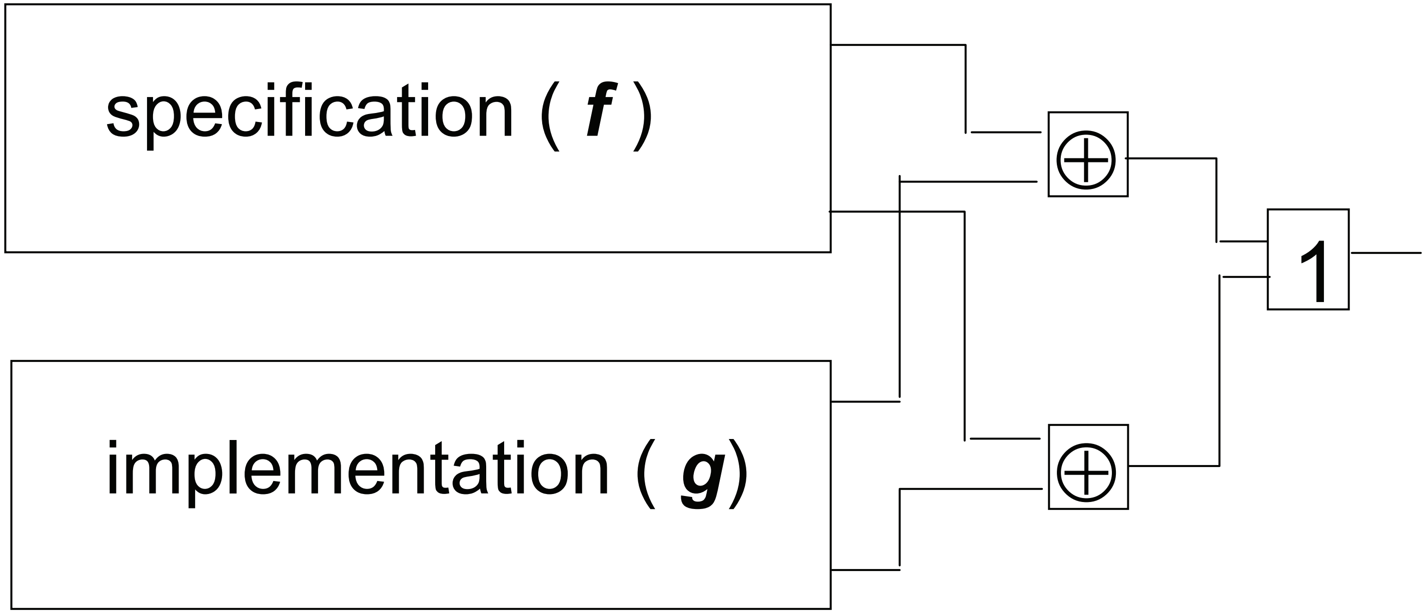
A



B

Formal verification: Equivalence checking 2

- $d = f \oplus g$
  - Assign the variables of boolean formula to evaluate it to TRUE (*satisfiable, **f** and **g** are not equivalent*)
  - OR prove that it evaluates to FALSE for all possible assignments (*not satisfiable, **f** and **g** are equivalent*)



SAT

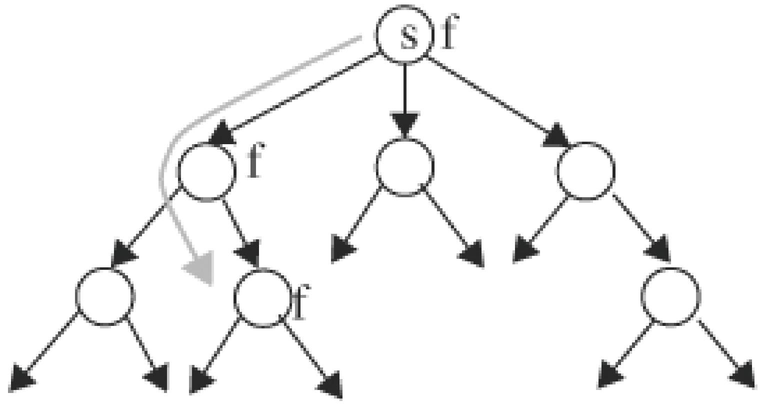
$$f(a,b,c) = (a+b+c)(\bar{a}+b+\bar{c})(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+\bar{c})(\bar{a}+\bar{b}+c)$$

- SAT (Boolean satisfiability problem)
- 2-SAT is solvable in polynomial time
- 3-SAT is NP-complete
  - » n-SAT can be reduced to 3-SAT in polynomial time

- Is *f(a,b,c)* solvable? (=1)
- The solution is:
  - » a = 1, b = 0, c = 0
- The worst case is to try 2^n options

Formal verification: Model checking

- Model checking proves or disproves that a property (part of specification) holds for the circuit
- Exhaustively searches the entire state space
  - » In real live the space can be constrained
- Typically the properties are described in CTL (Computational Tree Logic)
  - » [**AG**(P→((**EX.**Q)^(**EX**¬Q)))]

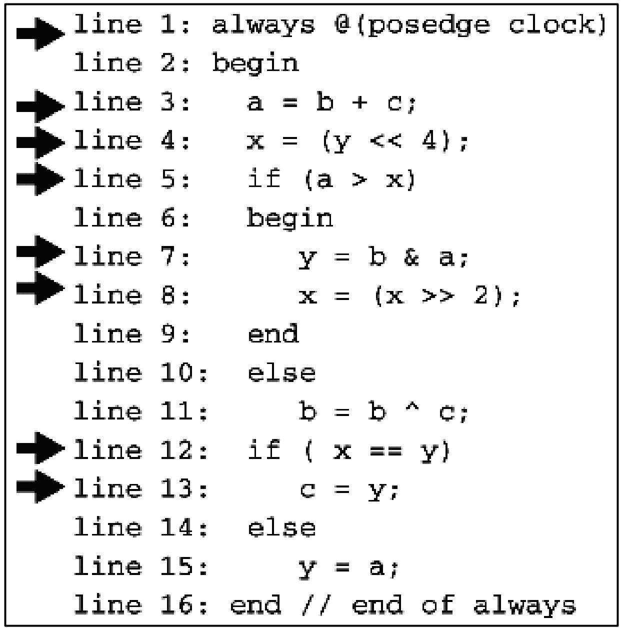
EG(*f*)	There is a path along which *f* holds at every state.	

Simulation-based verification

- <u>Coverage metrics</u>
- Usually it is not feasible to simulate all possible input combinations
- It is necessary to measure how much functionality given stimuli (input data) covers
- 3 types of coverage metrics in hardware verification:
  - » Code coverage
    - » (next slide)
  - » Parameter coverage
    - » Depends on implementation, used for parameters
  - » Functional coverage
    - » Depends not on implementation but on specifications
    - » Difficult to measure

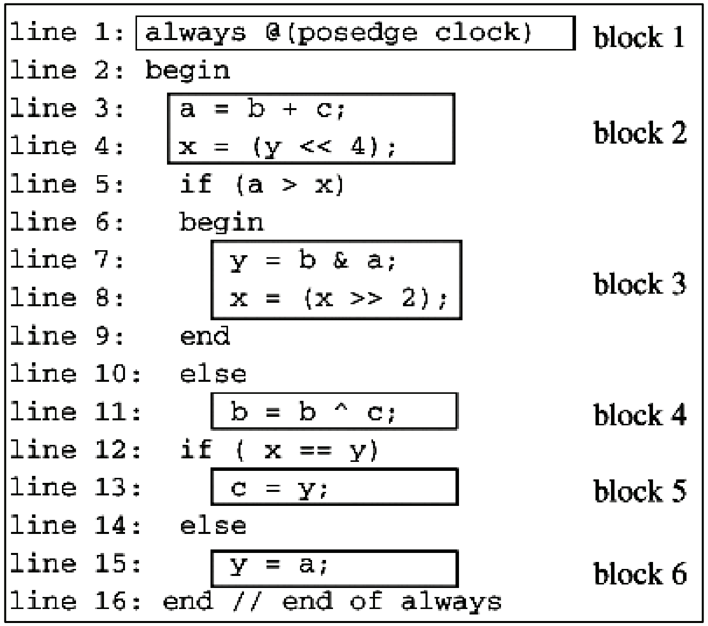
Code coverage metric

- How good code entities are stimulated by simulations
- Depends on implementation
  - It is possible to have 100% code coverage on completely wrong implementation
- Easy to calculate



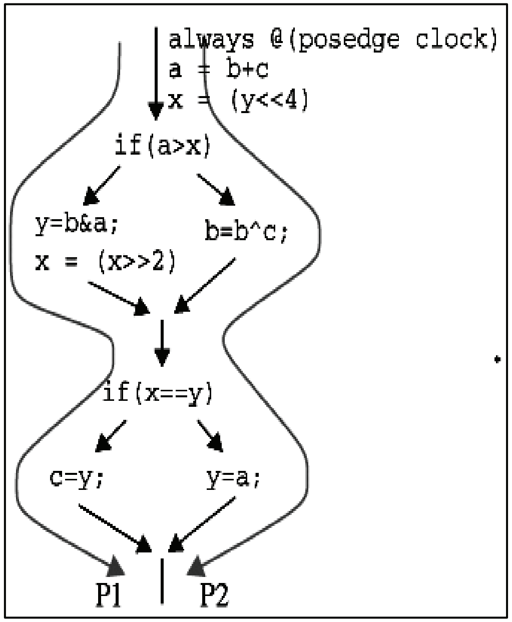
```
line 1: always @(posedge clock)
line 2: begin
line 3:   a = b + c;
line 4:   x = (y << 4);
line 5:   if (a > x)
line 6:   begin
line 7:      y = b & a;
line 8:      x = (x >> 2);
line 9:   end
line 10:  else
line 11:     b = b ^ c;
line 12:  if ( x == y)
line 13:     c = y;
line 14:  else
line 15:     y = a;
line 16: end // end of always
```

Statement coverage



```
line 1: always @(posedge clock)     block 1
line 2: begin
line 3:   a = b + c;
line 4:   x = (y << 4);             block 2
line 5:   if (a > x)
line 6:   begin
line 7:      y = b & a;
line 8:      x = (x >> 2);          block 3
line 9:   end
line 10:  else
line 11:     b = b ^ c;             block 4
line 12:  if ( x == y)
line 13:     c = y;                 block 5
line 14:  else
line 15:     y = a;                 block 6
line 16: end // end of always
```

Block coverage



Path coverage

Assertion-based Verification

	Property Checking	*Equivalence checking*
Dynamic verificatio	<u>**Assertion monitors**</u>	Code coverage analysis
Static Verification	Model checking	Equivalence checking

- ABV benefits:
  - » **Dynamic** – better observability

    detecting bugs earlier and closer to their origin
  - » **Static** – better controllability

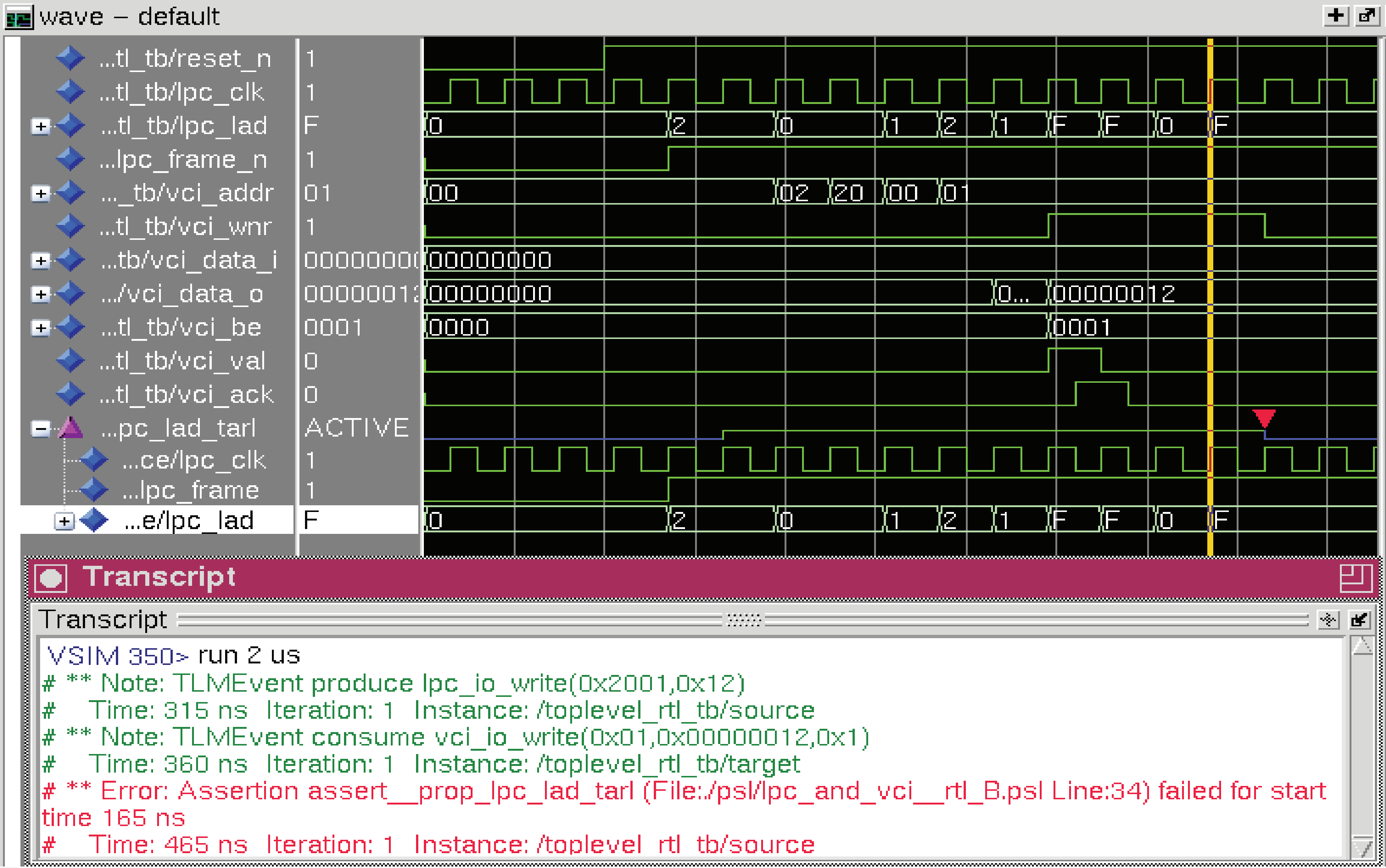
    direct verification to the area of interest

Assertion-based Verification

- Completeness problem
  - Who/what and when should specify assertions?
  - When is it enough?

- In practice design engineer writes them for VHS (Verification Hot Spots). Such spot:
  - contains a great number of sequential states;
  - deeply hidden in the design, making it difficult to control from the inputs
  - has many interactions with other state machines and external agents
  - has a combination of these properties

QuestaSIM from MentorGraphics



```
VSIM 350> run 2 us
# ** Note: TLMEvent produce lpc_io_write(0x2001,0x12)
#    Time: 315 ns  Iteration: 1  Instance: /toplevel_rtl_tb/source
# ** Note: TLMEvent consume vci_io_write(0x01,0x00000012,0x1)
#    Time: 360 ns  Iteration: 1  Instance: /toplevel_rtl_tb/target
# ** Error: Assertion assert__prop_lpc_lad_tarl (File:./psl/lpc_and_vci__rtl_B.psl Line:34) failed for start
time 165 ns
#    Time: 465 ns  Iteration: 1  Instance: /toplevel_rtl_tb/source
```

PSL

- PSL = Property Specification Language
  - » Based on IBM’s Sugar, developed by Accellera
  - » IEEE 1850 Standard in 2005
- Flavors:

  VHDL, Verilog, SystemVerilog, GDL, SystemC
- 4 layers:

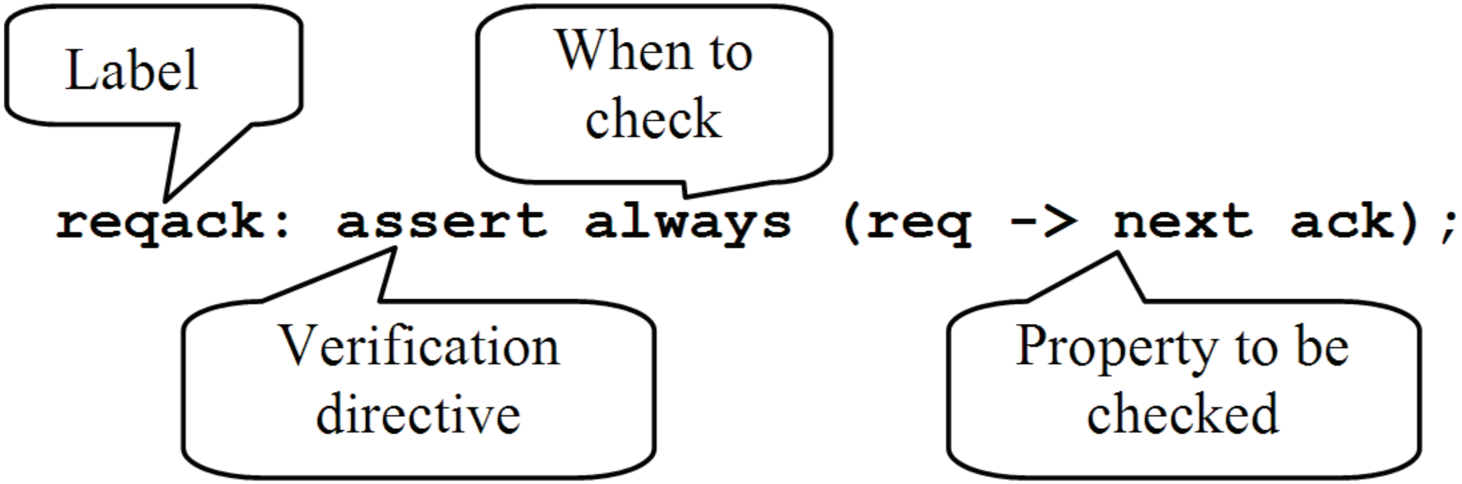
  Boolean layer –boolean expressions in HLD: (*a&&(b||c)*)

  Temporal later – sequences of boolean expressions over multiple clock cycles, supports SERE: (*{A[*3];B}|->{C}*)

  Verification layer - directives for verification tool telling what to do with specified properties

  Modelling layer – models environment

PSL (cont.)



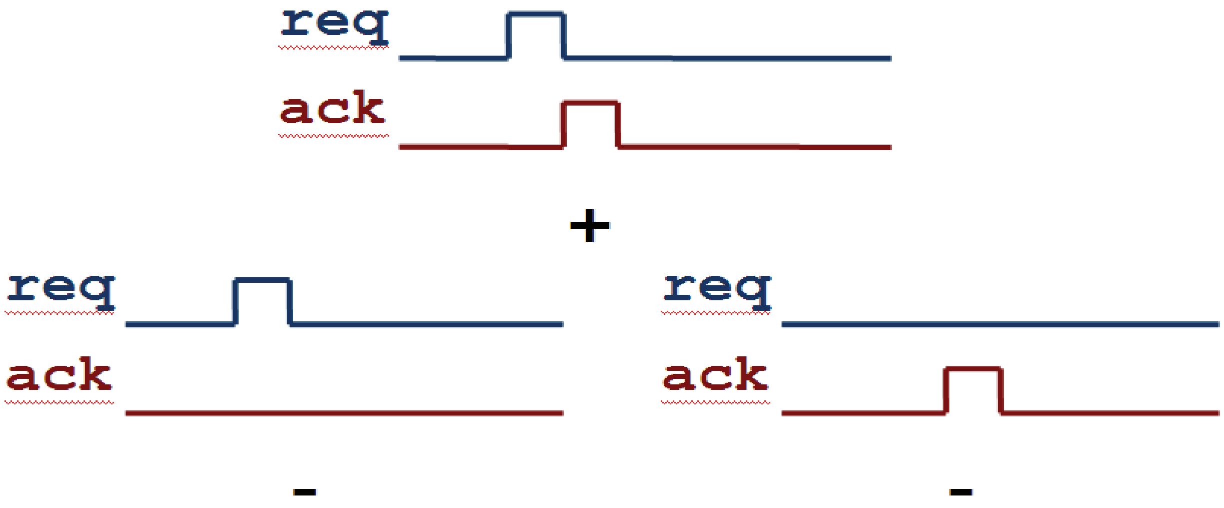
Label

When to check

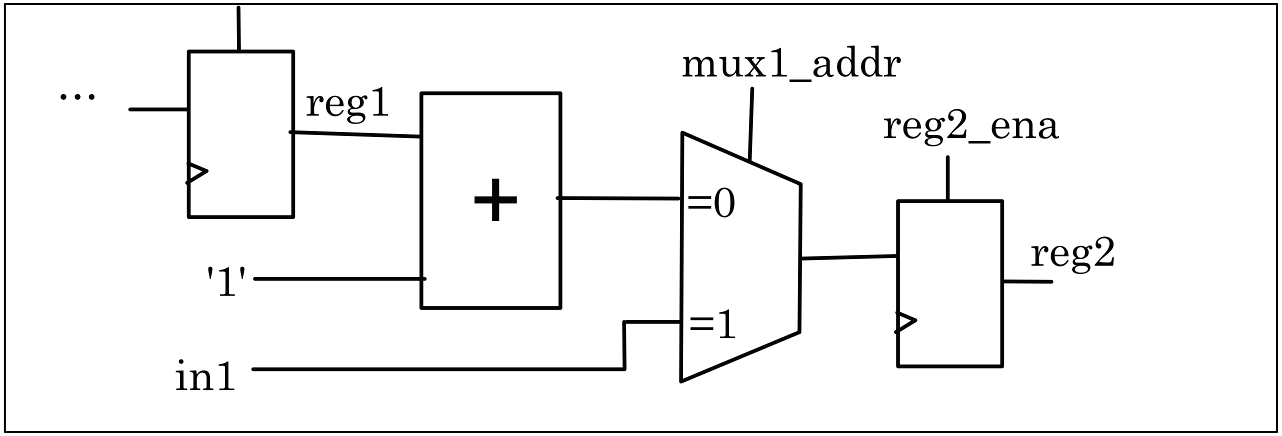
```
reqack: assert always (req -> next ack);
```

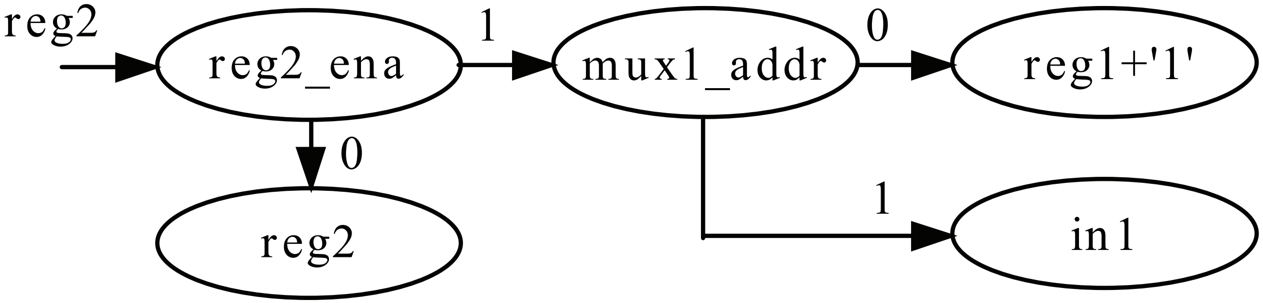
Verification directive

Property to be checked



High-Level Decision Diagrams (HLDD)





- Proposed and developed in TUT
- HLDDs are proved to speed-up simulation
  - » By up to factor 10 compared to commercial simulators

European Commission

6th Framework Programme Research Project

- ST Microelectronics (coordinator)
- Aerielogic and TransEDA
- Universities form

  Tallinn (Estonia)

  Linköping (Sweden)

  Southampton (UK)

  Verona (Italy)

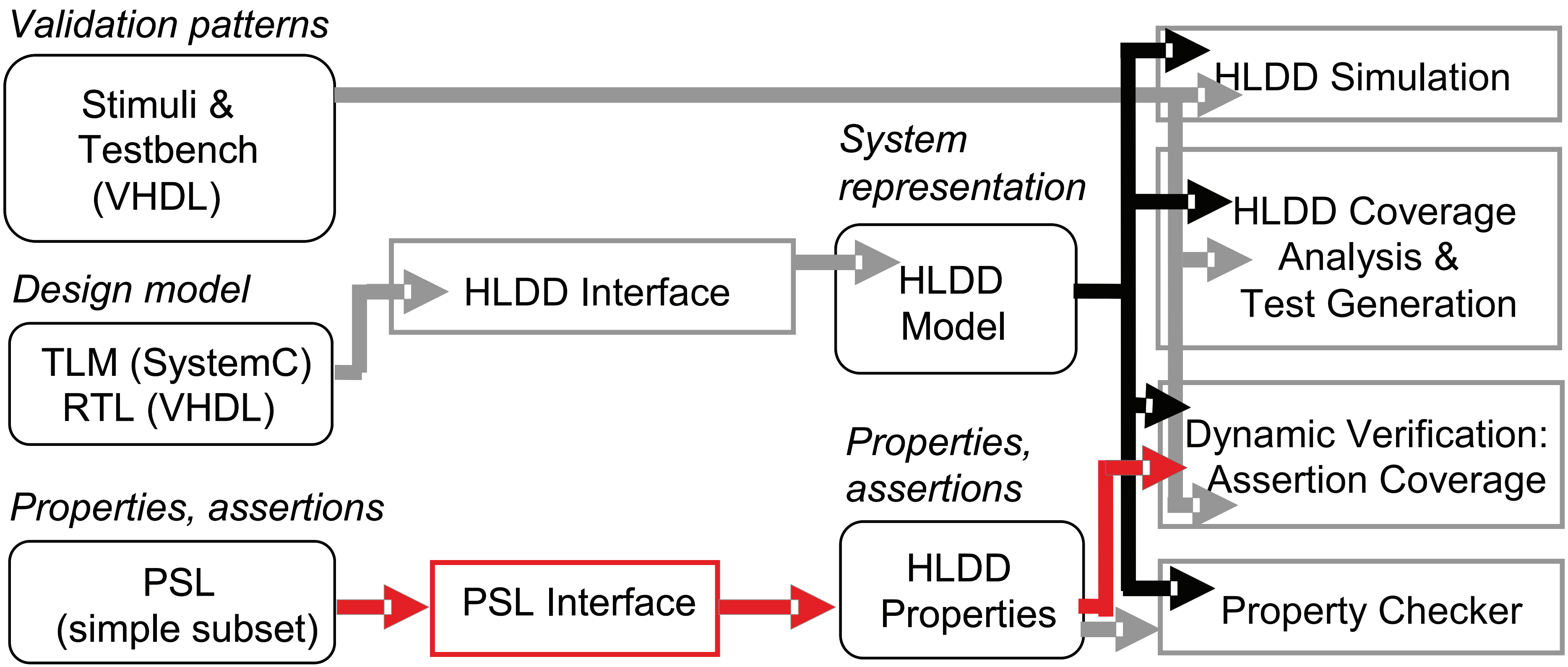
2006 - 2008

VERTIGO

Verification and Validation of Embedded System Design Workbench

Work-in-progress research at TUT

- HLDD-based verification flow


Validation patterns

Stimuli & Testbench (VHDL)

Design model

TLM (SystemC) RTL (VHDL)

Properties, assertions

PSL (simple subset)

HLDD Interface

PSL Interface

System representation

HLDD Model

Properties, assertions

HLDD Properties

HLDD Simulation

HLDD Coverage Analysis & Test Generation

Dynamic Verification: Assertion Coverage

Property Checker

PSL integration to HLDD

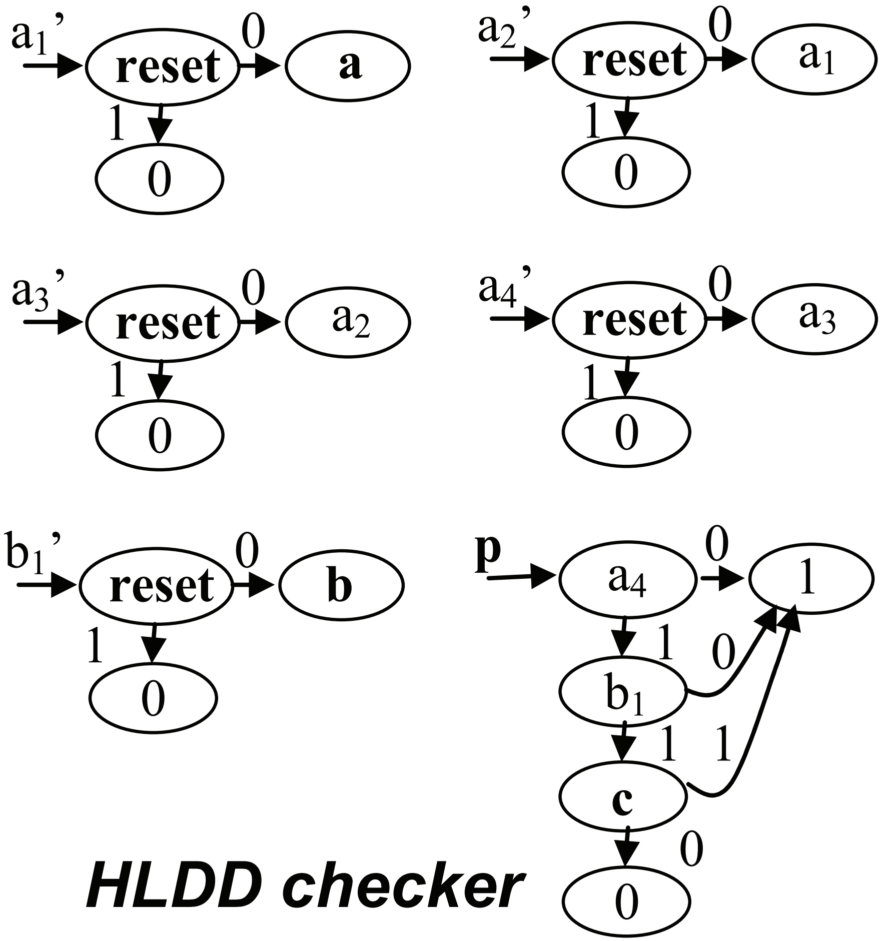
- Assertions automatic translation is a complex process employing:
  - » Nondeterministic Finite Automaton
  - » Deterministic Finite Automaton
- Only subset of PSL properties is translatable
- An option to TUT internal solution is external translator like FoCs form IBM

Translation using FoCs from IBM

```
p: assert always ({a; [*2] ;b} |=> {c});
```

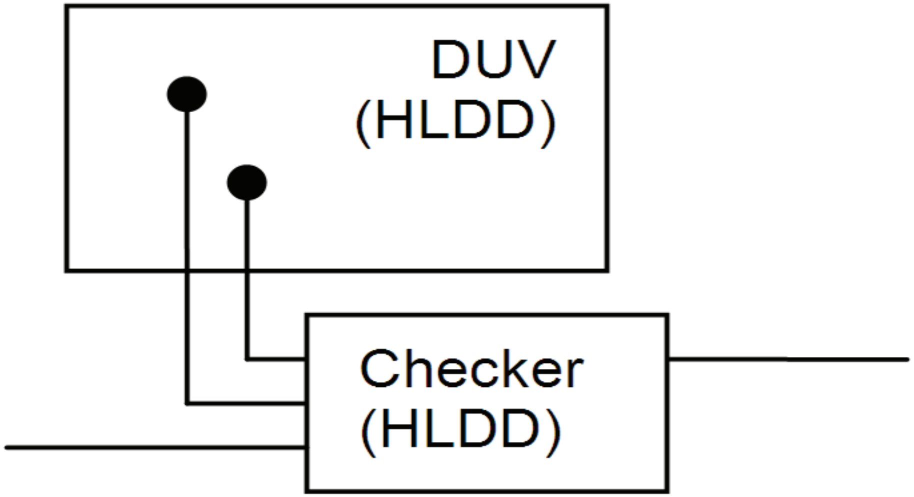
```
PROCESS (clk)
BEGIN
   IF ( ( clk = '1' ) ) THEN
      focs_ok <=
         ( focs_vout(4) OR NOT( c ) ) ;
   ELSE
      focs_ok <= '1' ;
   END IF;
END PROCESS;
PROCESS
...
VARIABLE focs_vout : std_logic_vector(4 DOWNTO 0);
BEGIN
   WAIT UNTIL (clk'EVENT AND clk = '1');
   ...
   focs_vout(4 DOWNTO 0) := reverse( ( ( ( ( ( (
      focs_v(0) AND a ) ) & ( ( focs_v(1) AND '1' )
      ) ) & ( ( focs_v(2) AND '1' ) ) ) & ( (
      focs_v(3) AND b ) ) ) & ( ( focs_v(4) AND
      NOT( c ) ) ) ) ) );
   ...
END PROCESS;
```

FoCs .vhdl checker



HLDD checker

Conclusions



- This presentation has given a brief overview of ***hardware verification***
- Work-in-progress of Ph.D. research was presented
  - » PSL assertions are used as HLDD simulation checkers in verification flow

Bibliography and contact

- Book:

  Hardware Design Verification: Simulation and Formal Method-Based Approaches, *William K. Lam, Sun Microsystems 2005*

- Papers:

  High-Level Decision Diagrams (HLDD) and DECIDER by **Jaan Raik** and **Prof. Raimund Ubar** from IEEE*Explore*

- Contact:

  papers on PSL and assertions

  Maksim Jenihhin – maksim@pld.ttu.ee

  Tallinn University of Technology, ESTONIA

The “Thank You” slide

Thank You!