Digital systems' modeling and verification

IAS0440, 6.0 Credits, Exam

MS Teams team "*Digital systems' modeling and verification*"

Lectures: *Jaan Raik, Peeter Ellervee*

Labs: *René Pihlak*

ICT-524, +372 55 13141

jaan.raik@taltech.ee

Course outline and materials in Moodle

Week 1. Introductory Lecture. Modeling and Verification

Slides: Verification intro

Introduction to verification; Motivation and rationale; Types of verification; Course structure.

Slides: Modeling and simulation intro

Introduction to modeling and simulation; Motivation and rationale; Modeling and simulation concepts; Course structure.

Remote login to lab computer

Week 2-3. Verilog/SystemVerilog Basics (2 Lectures, 1 Lab)

Slides: Verilog/SystemVerilog basics

Introduction to Verilog/SystemVerilog hardware descriptiion languages; Main modeling concepts.

Slides: SystemVerilog Lab

Lab resources: multiplexer

Lab resources: Sequence detector

Lab resources: GCD design ideas

Week 4-5. Simulation, Co-Simulation (1 Lecture, 3 Labs)

Slides: Simulation/co-simulation

Simulation and co-simulation; Semantics and scheduling.

LAB: Self-learning. Possible to ask questions in class/lab.

Week 6. Equivalence Checking (1 Lecture, 1 Exercise)

Slides: Equivalence checking

The concept of equivalence checking; Binary decision diagrams; Satisfiability aka SAT.

Exercises 1-2. SSBDDs, SAT

Week 8 & 10. Verification Coverage (1 Lecture, 3 Labs)

Slides: Verification coverage

Coverage metrics; Functional coverage; Parameter coverage; Code coverage.

Week 11. SystemVerilog Constraints (1 Lecture, 1 Lab)

Slides: SystemVerilog constraints

SystemVerilog constraints.

Week 12. Property Checking (1 Lecture)

Slides: Property checking

Verification properties; Automata; Temporal logic (LTL, CTL); Model-checking;

Week 13-14. Assertions (1 Lecture, 3 Labs)

Slides: Assertions

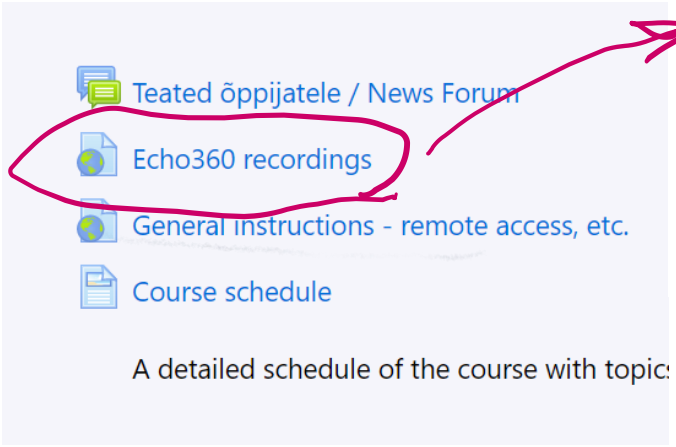
Verification assertions; SystemVerilogAssertions; Sequences and operators.

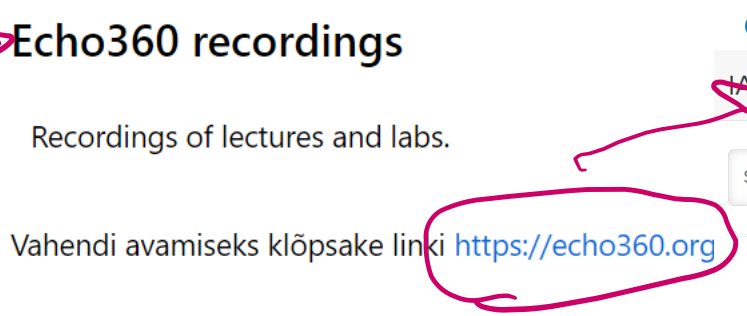
Week 15. Advanced topics: automated debug, pitfalls in modeling (1 Lecture)

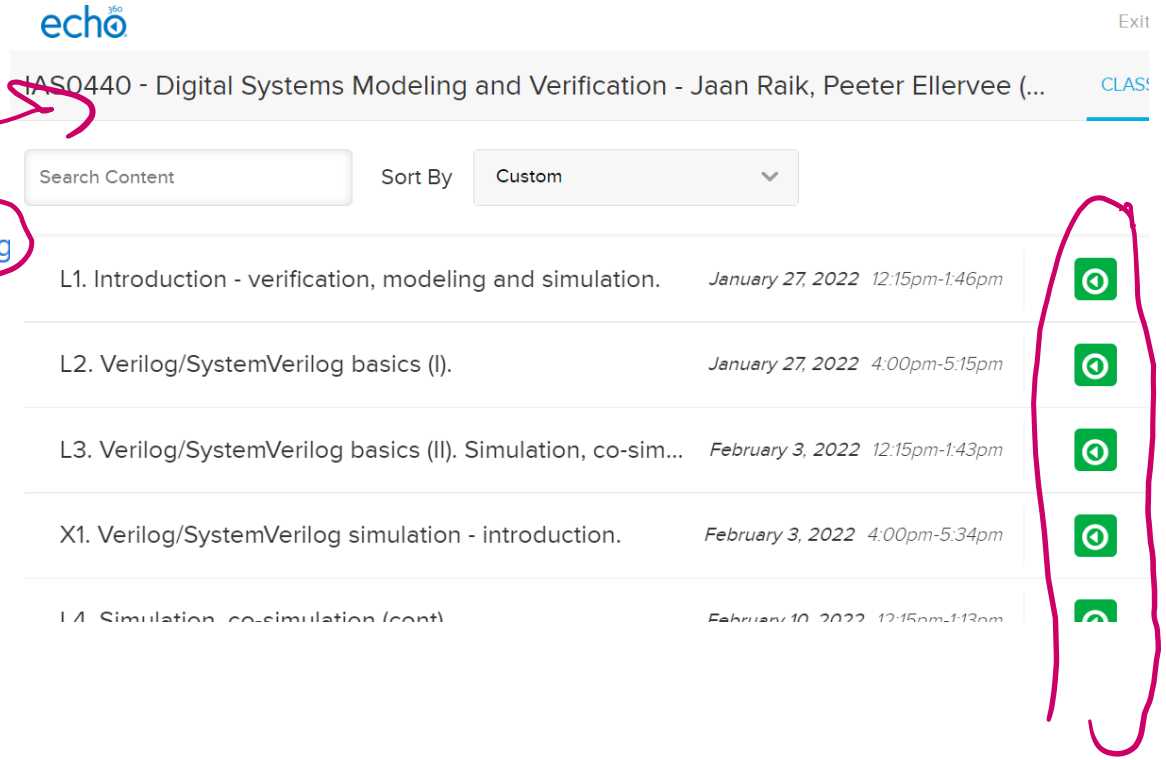
PPT

Week 16. Finalization of coursework (Consultation and lab)

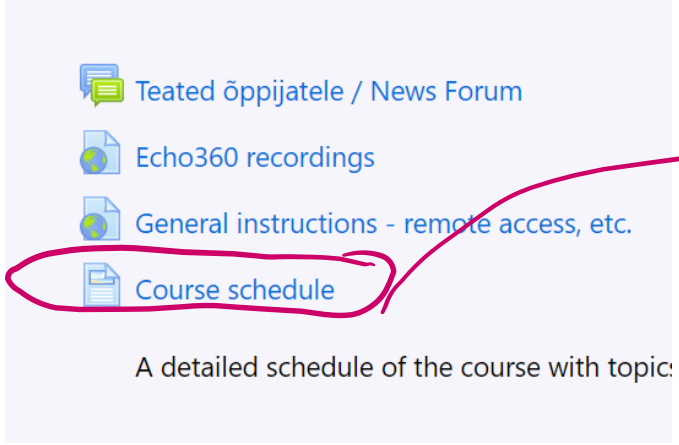
Lecture/lab recordings & schedule







Lecture/lab recordings & schedule



Course schedule

Verification coverage (I)Lab. Verification CoverageSystemVerilog constraintsLab: AssertionsLab: AssertionsLab: Verification CoverageExercise: Equivalence checking

Week	Lecture/Practice	Practice
#1 (31.01.23)	Introduction. Modeling & Verification • Topics • Schedule • Labs overview • Report requirements • Grading system	—
#2 (7.02.23)	Verilog/SystemVerilog Basics (I)	—
#3 (14.02.23)	Verilog/SystemVerilog Basics (II)	Lab: simulation intro
#4 (21.02.23)	Simulation, Co-Simulation	Lab: SystemVerilog, GCD
#5 (28.02.23)	Lab: GCD (cont.)	Lab: GCD (cont.)
#6 (7.03.23)	Equivalence checking	Exercise: Equivalence checking
#7 (14.03.23)	No lecture	No Lab

Digital systems' modeling and verification: course progress

1. Introduction to modeling and verification
2. Hardware description language SystemVerilog
3. Co-modeling/simulation of digital systems, testbenches
4. Equivalence checking, BDDs and SAT
5. Verification coverage
6. SystemVerilog constraints
7. Property checking
8. SystemVerilog Assertions

Lab 1: 10%

Ex 1: req

Lab 2: 15%

Lab 3: 10%

Lab 4: 15%

Exam: 50%

Digital systems' verification

Course book:

Hardware Design Verification: Simulation and Formal Method-Based Approaches

William K. Lam, Sun Microsystems

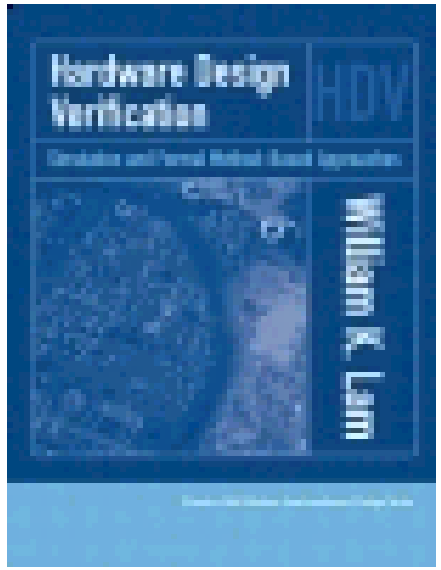
.....

Publisher: **Prentice Hall PTR**

Pub Date: **March 03, 2005**

ISBN: **0-13-143347-4**

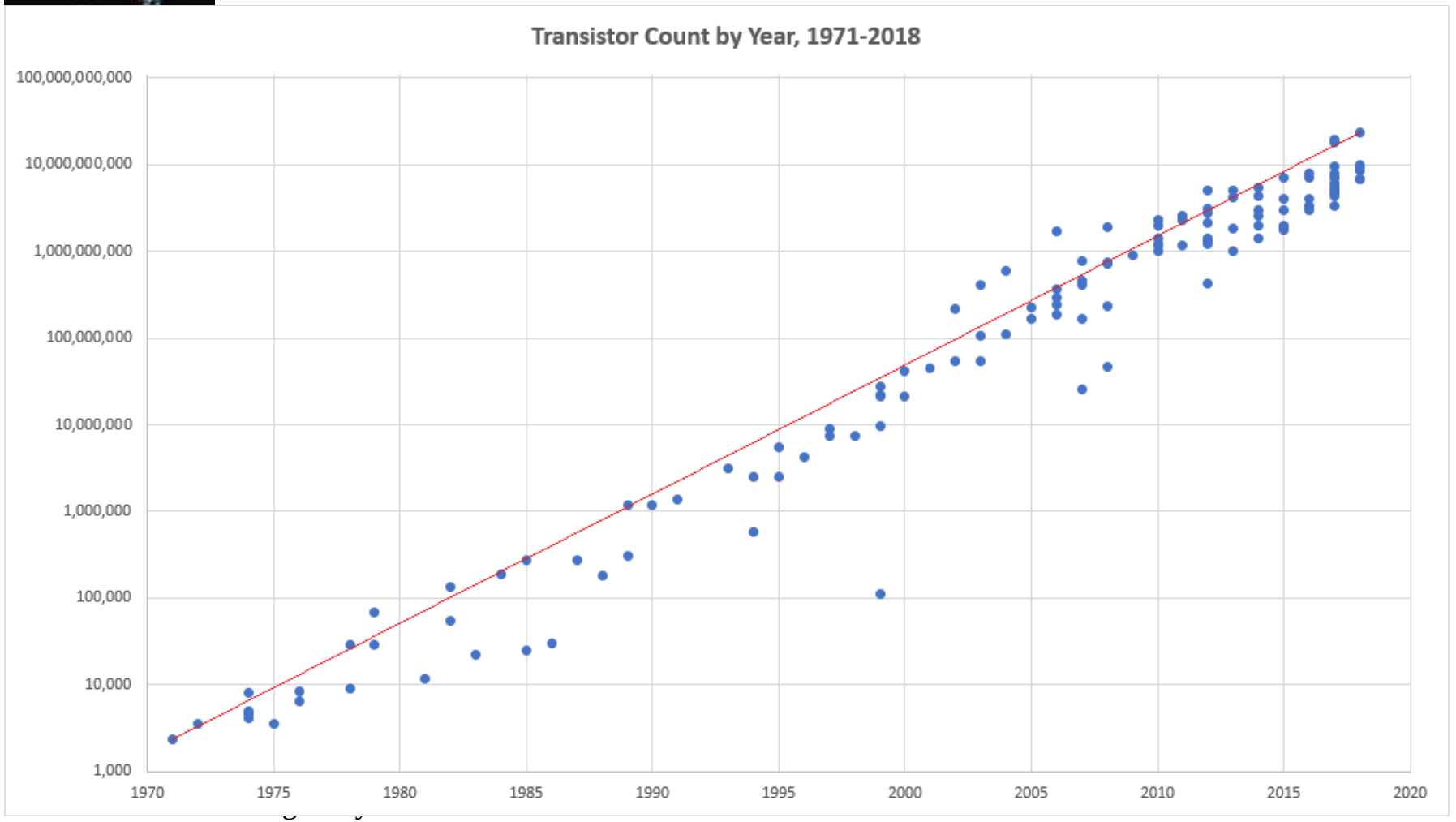
Pages: **624**



Embedded Systems

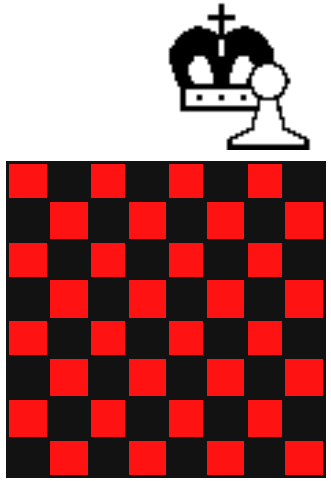
Moore's law (1965)

"Essential parameters of digital devices double each 18 months."

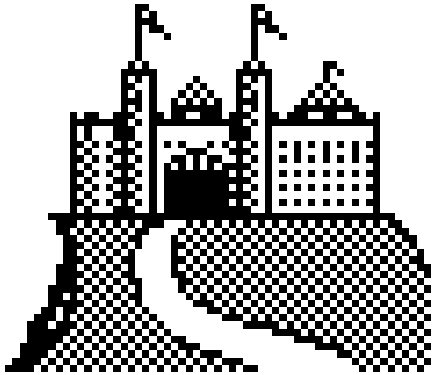


Moore's law (1965)

	Year of Introduction	Transistors
4004	1971	2,250
8008	1972	2,500
8080	1974	5,000
8086	1978	29,000
286	1982	120,000
Intel386™ processor	1985	275,000
Intel486™ processor	1989	1,180,000
Intel® Pentium® processor	1993	3,100,000
Intel® Pentium® II processor	1997	7,500,000
Intel® Pentium® III processor	1999	24,000,000
Intel® Pentium® 4 processor	2000	42,000,000
Intel® Itanium® processor	2002	220,000,000
Intel® Itanium® 2 processor	2003	410,000,000
AMD Epyc Rome	2019	**32,000,000,000**



???

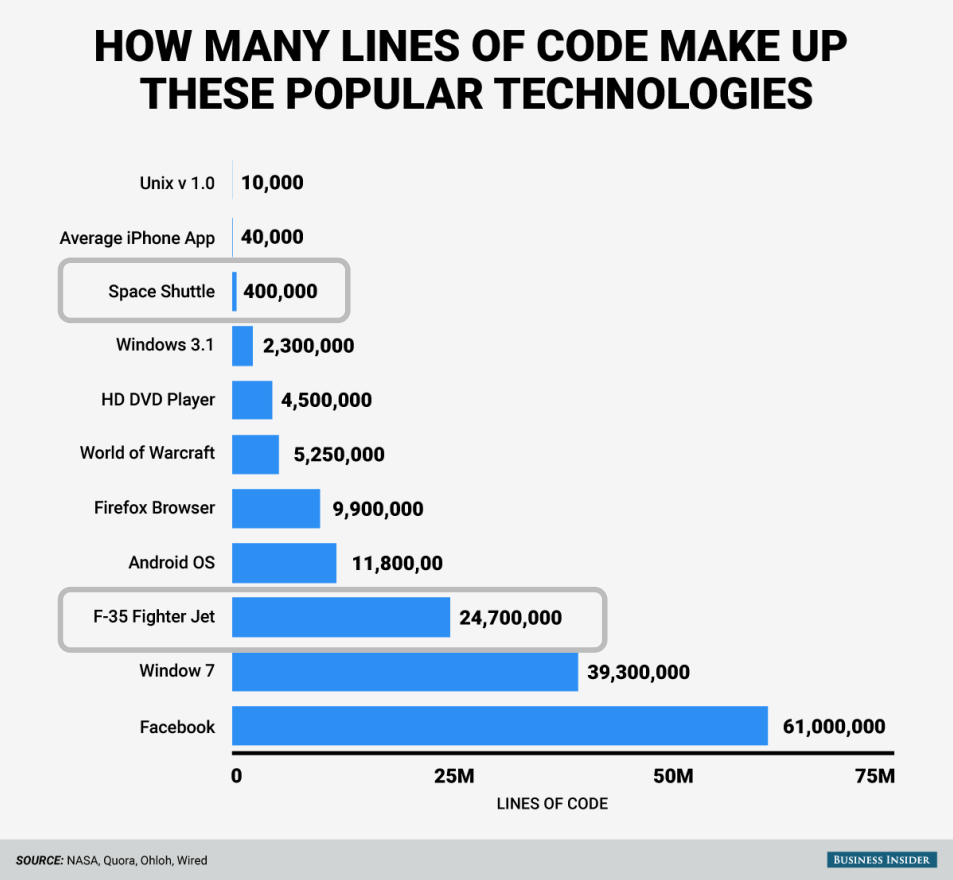


32. position ~ 4 billion grains

64. position ~10^{19} grains

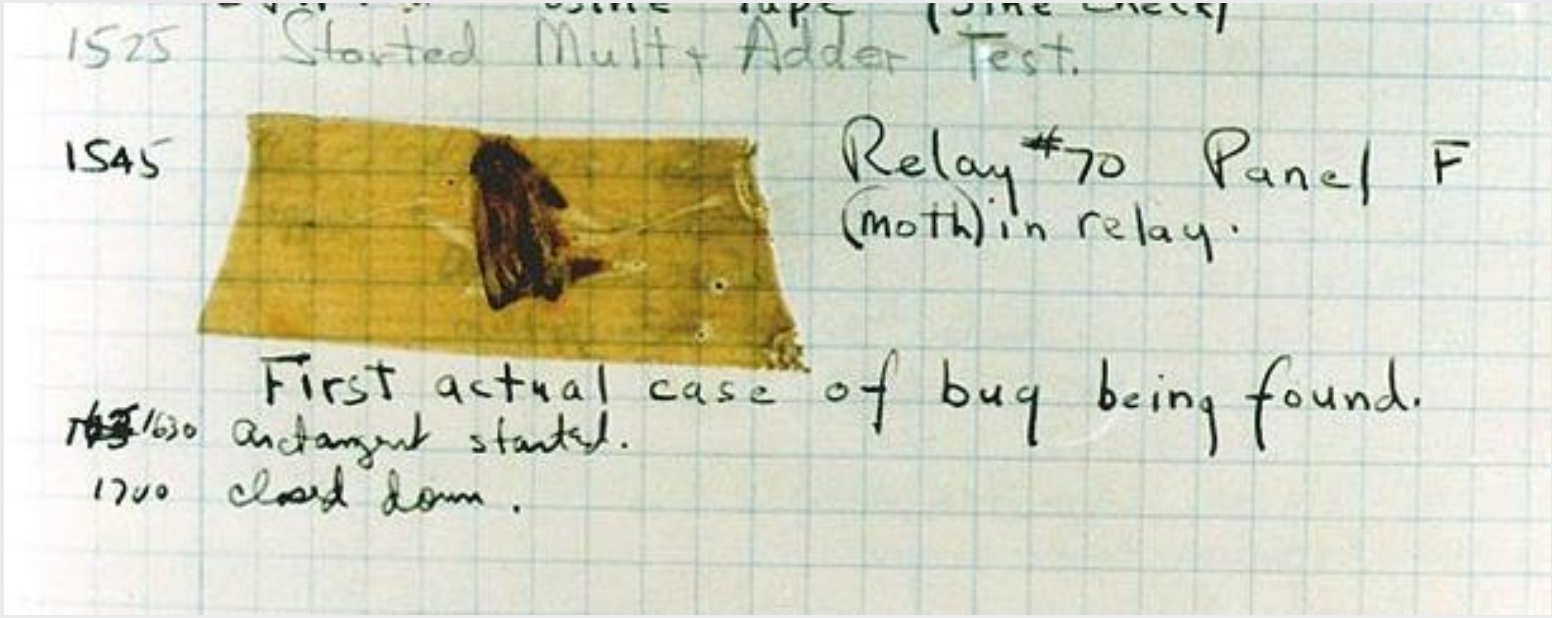
HW/SW Systems

- ✓ Complexity in Lines of Code



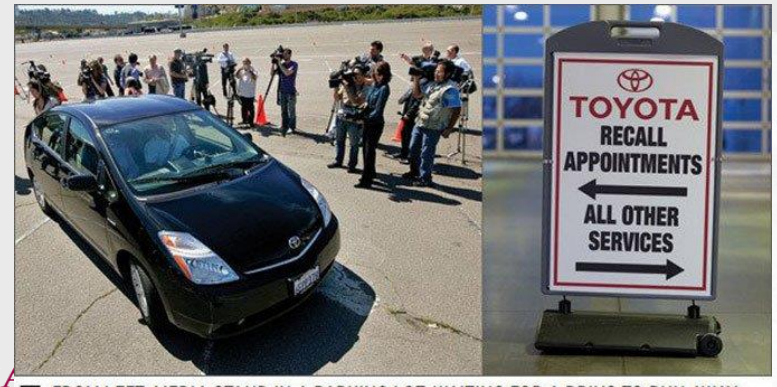
The first bug found

- ✓ Harvard Mark II system, 1945



When things go wrong...

- ✓ **1985-1987 - Therac-25 medical accelerator.** At least 5 patients die.
- ✓ **1993 - Intel Pentium floating point divide.** Costs te company $475 million.
- ✓ **June 4, 1996 - Ariane 5 Flight 501.** Rocket disintegrates 40 seconds after launch.
- ✓ **February 2010 - Toyota Prius failure.** Toyota loses its market share.



FROM LEFT: MEDIA STAND IN A PARKING LOT, WAITING FOR A PRIUS TO RUN AWAY; RECALLS KEPT TOYOTA SERVICE BAYS PLENTY BUSY

Why is verification important

Some figures:

- 2-4 verification engineers per designer
- Verification takes 70-85 % of total cost of chip design
- Why has verification become that important?

Rapid growth of digital technology

☺ 25-30 % annually decreasing cost per function

☺ 15 percent annual growth of the market for integrated circuits

• But …

☹ The cost of developing a digital chip keeps on growing.

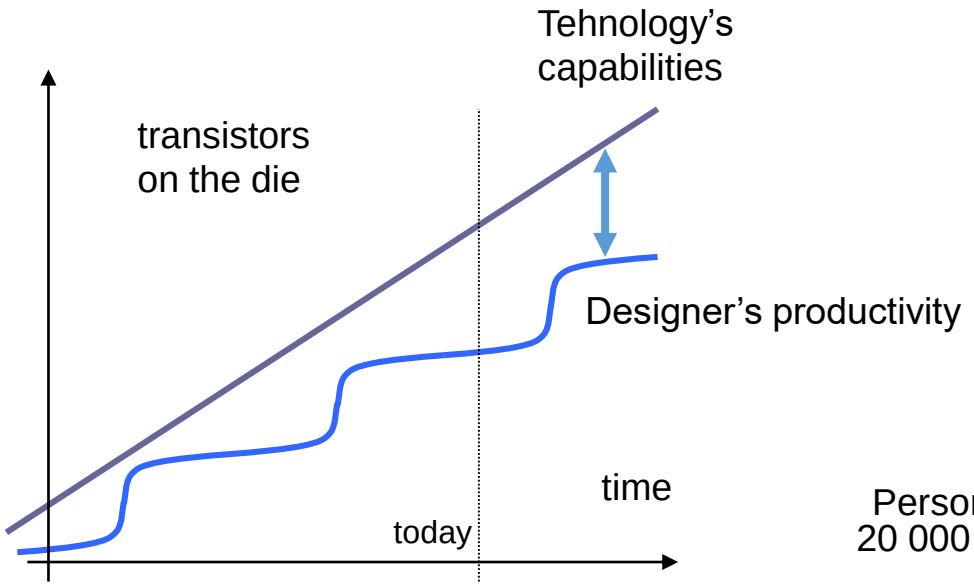
• In 1981, development of a leading-edge CPU cost 1 M$

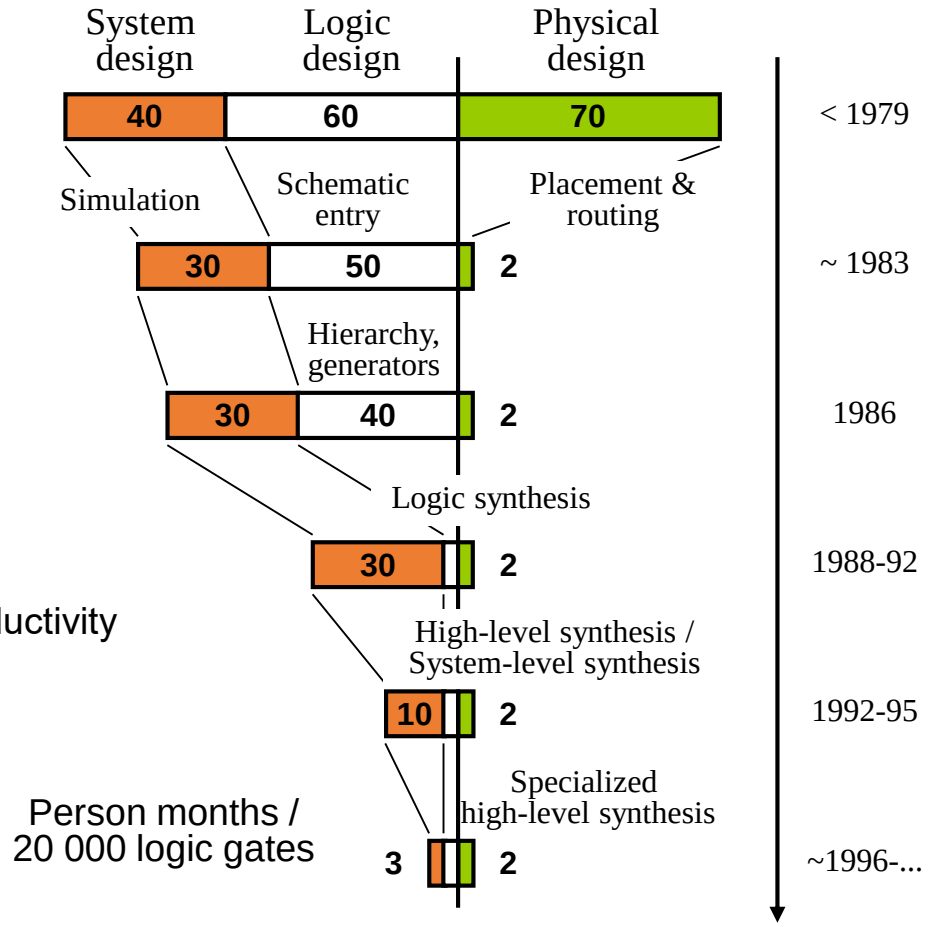
• …today it costs more than 300 M$!!!

• Why do costs increase ???

Design automation

- productivity gap
  - 58% versus 21% annually





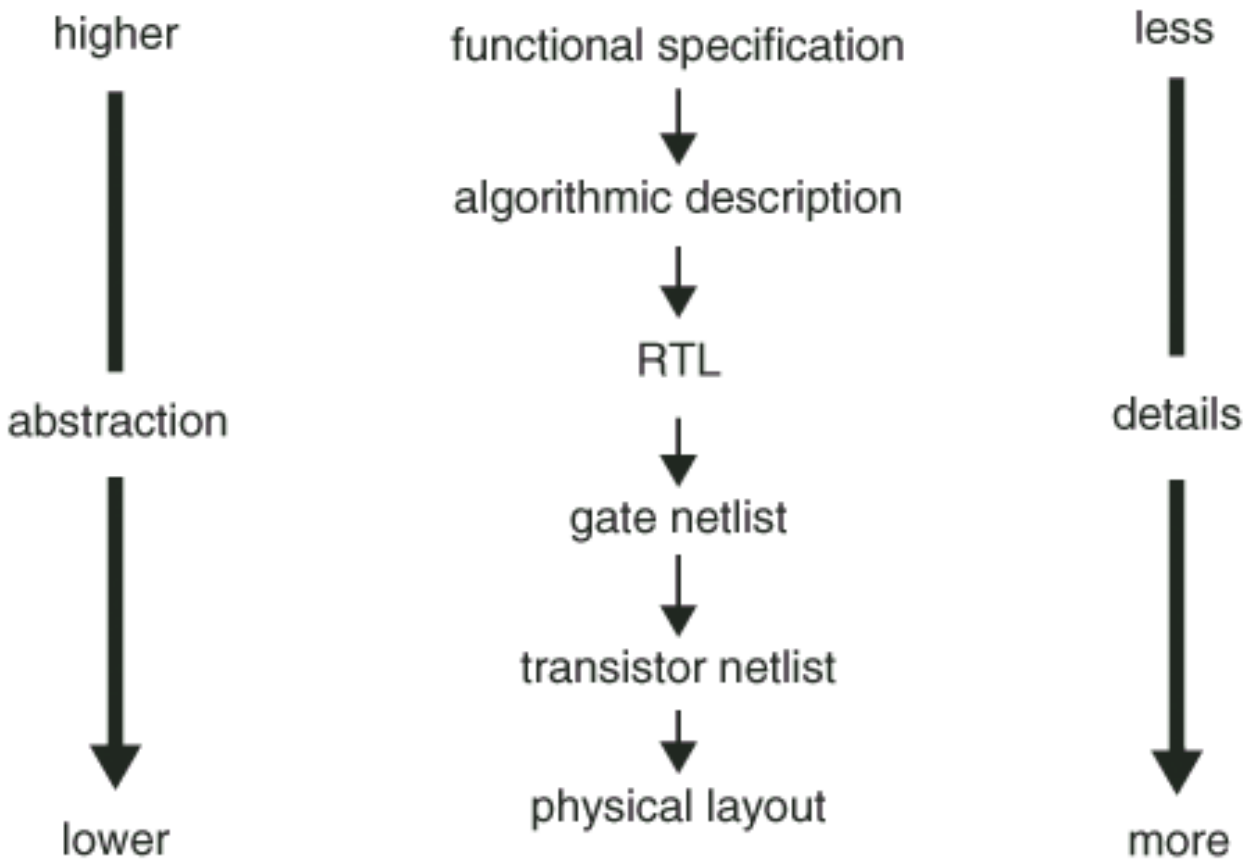
Verification versus test

- The goal of *verification* is to check if a system is **designed** correctly.
- *Validation* is similar to verification but we check on a **prototype device**, not a model.
- *By (manufacturing) Test* we understand checking every instance of a produced **chip** against manufacruring defects.

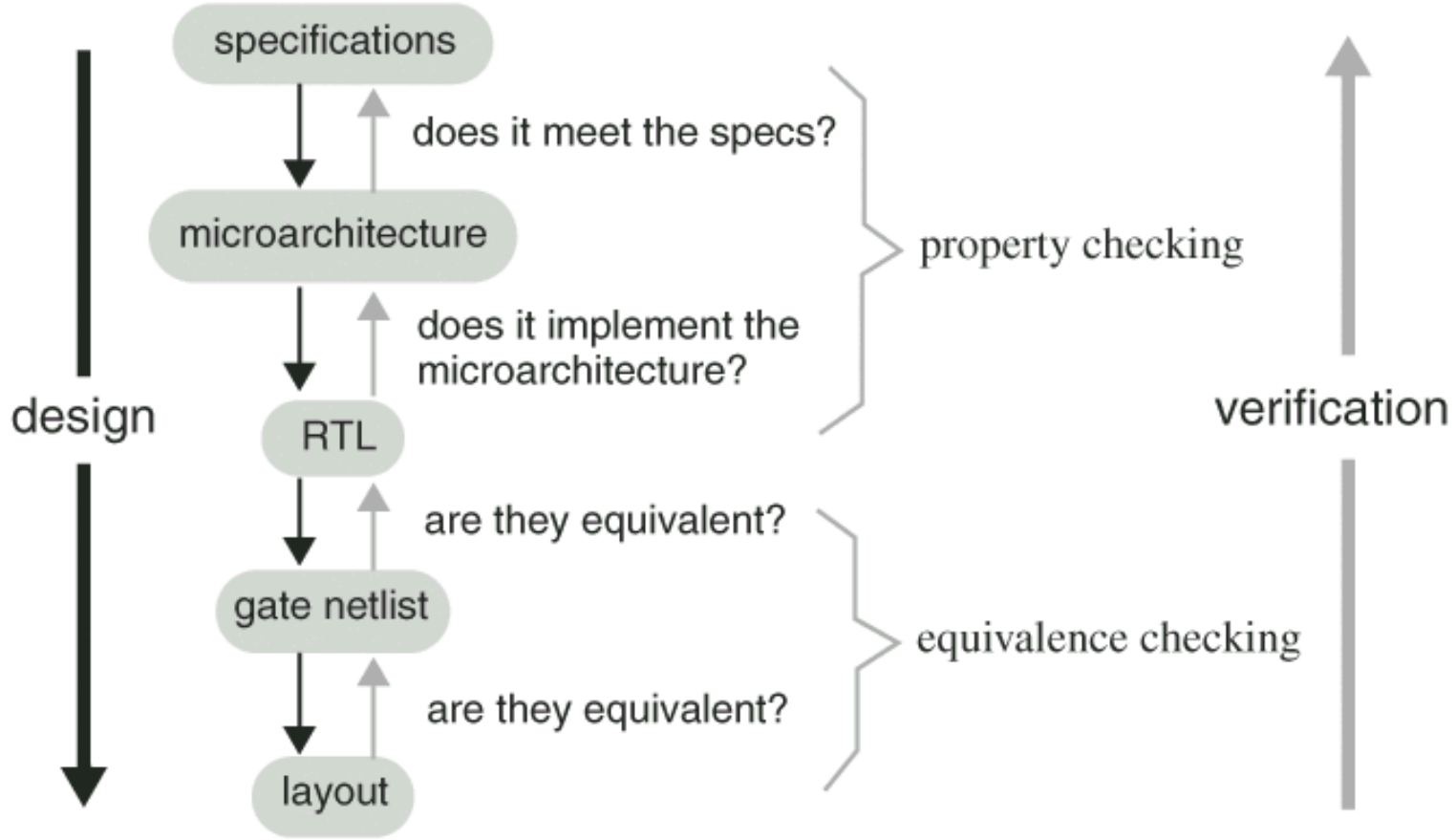
Types of verification

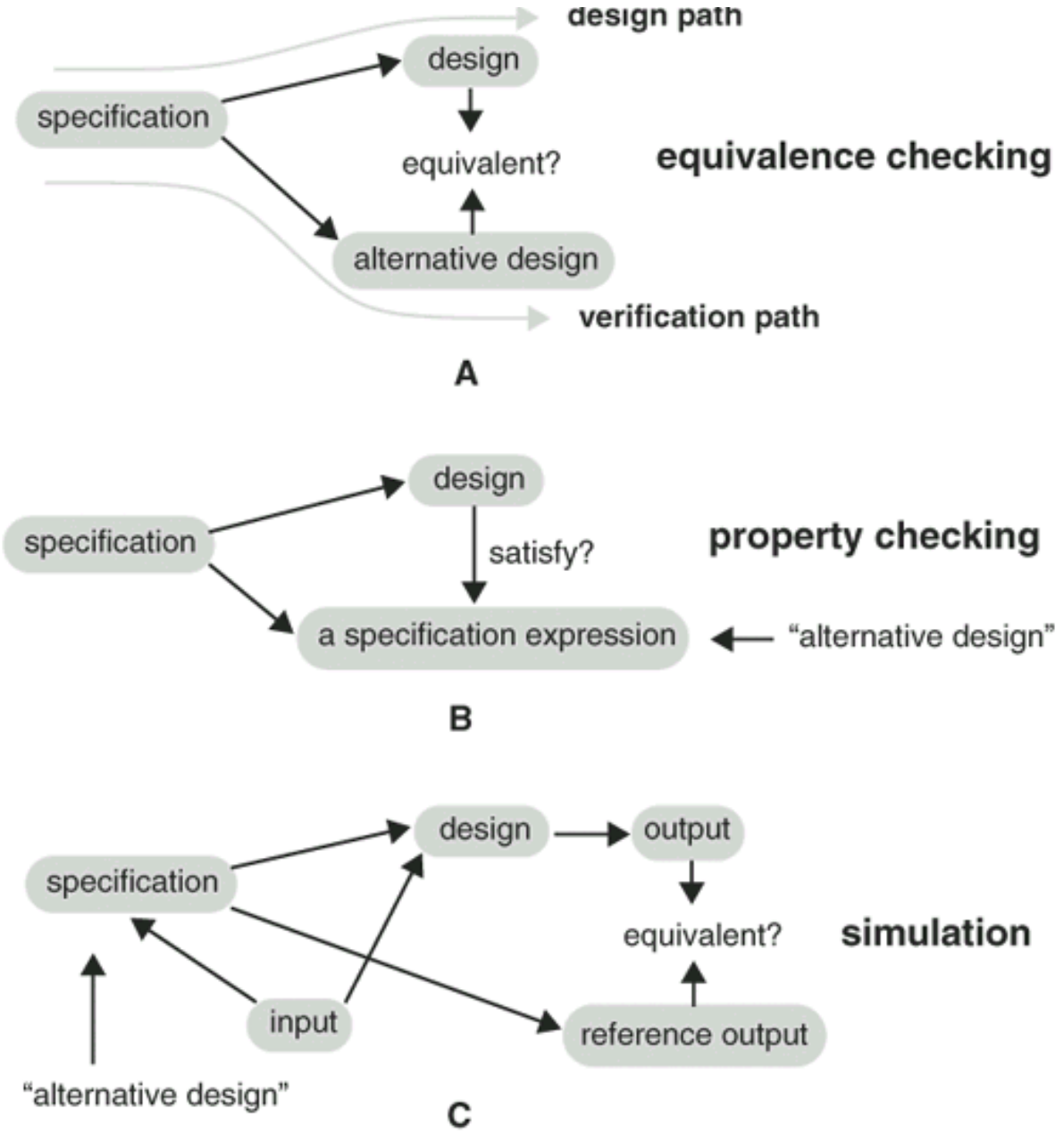
- Mixed-signal, analog, RF, ...
- We consider only digital
- Functional, timing, layout, electrical etc. verification
- Here, we consider functional only

Design abstraction levels and verification



Verification flow

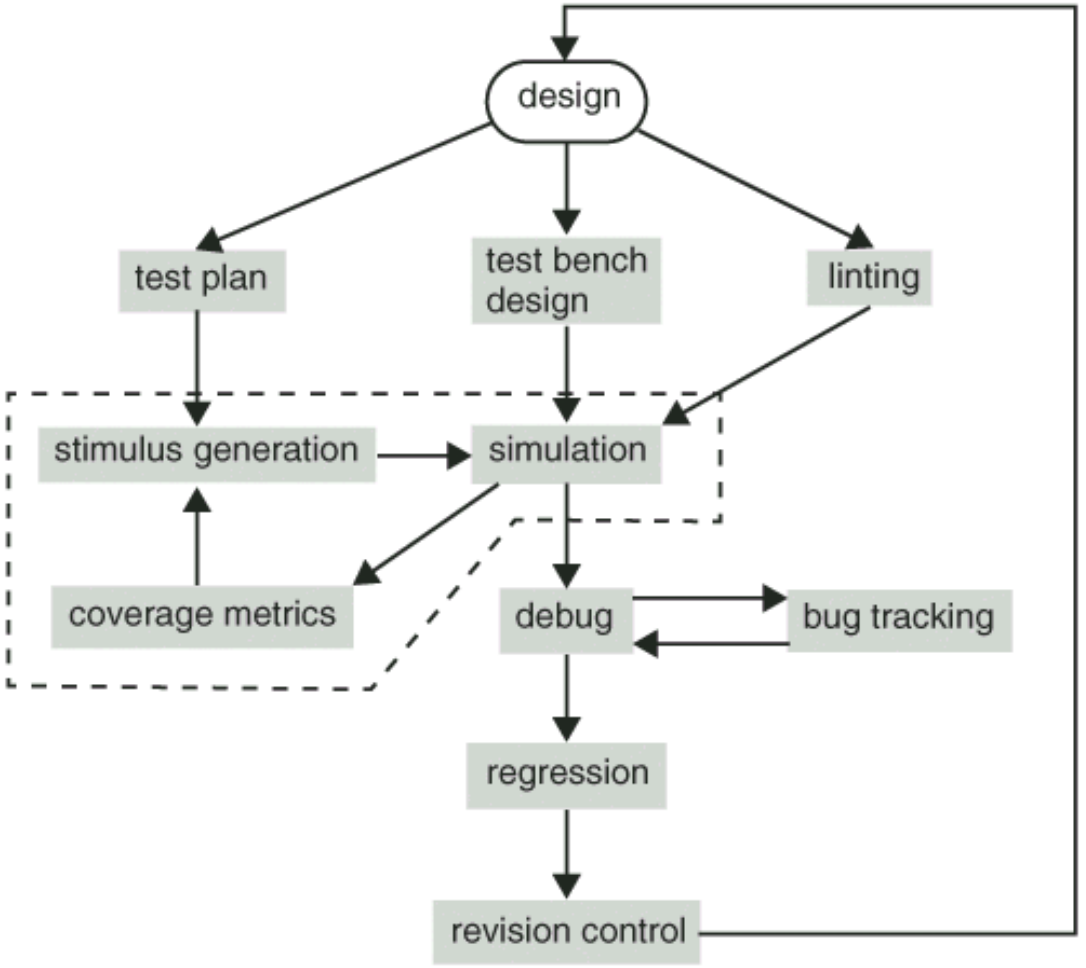



design path
design
specification
equivalent?
equivalence checking
alternative design
verification path
A
design
specification
satisfy?
property checking
a specification expression
"alternative design"
B
design
output
specification
equivalent?
simulation
input
reference output
"alternative design"
C

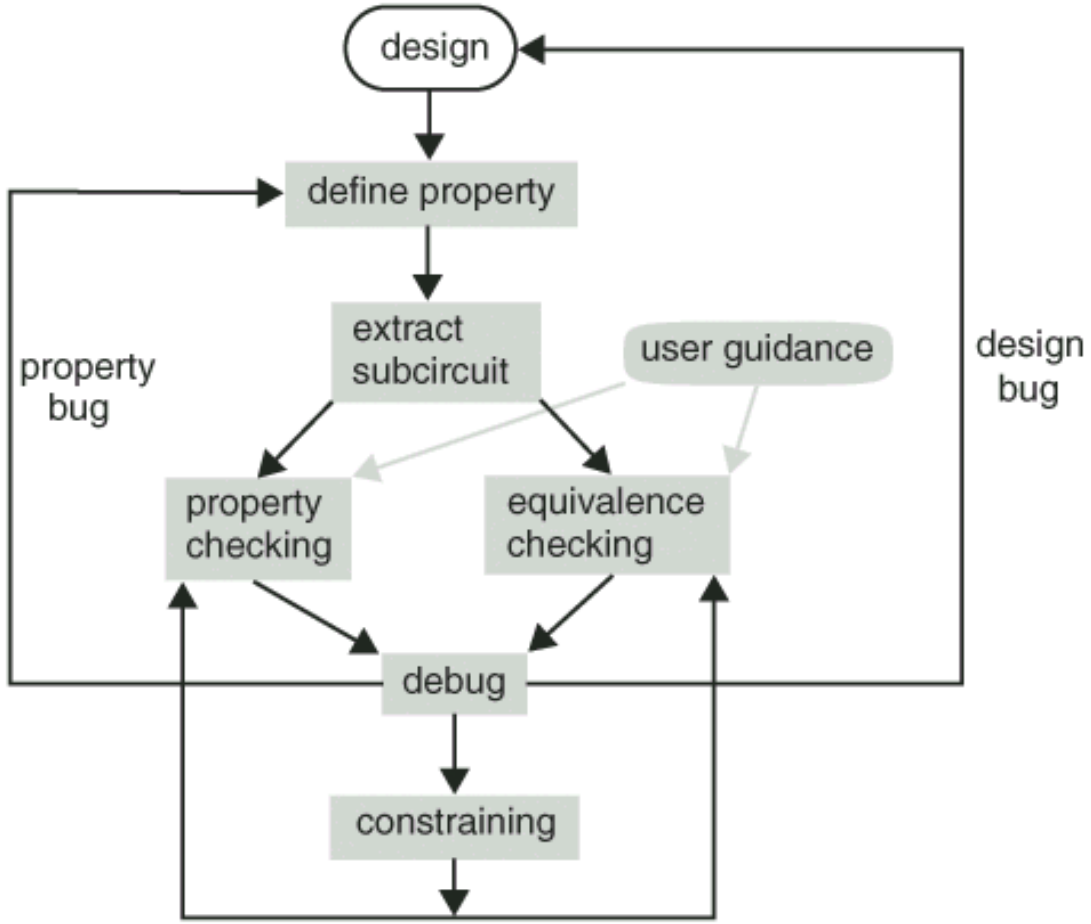
Problems during verification

- Errors in spec, implementation, language
- No way to detect bugs in the spec, because reference object is missing. Thus: *verification by redundancy*.
- Problem: How to measure verification quality i.e. coverage? (except in equivalence checking)

Simulation-based verification



Formal verification



Equivalence checking

- It is necessary to match the variable names of the designs under comparison!
- We implement canonical forms ...
- ... or look for an input vector that would distinguish the output responses of two designs. SAT methods. Similar to test pattern generation.

Equivalence checking

- Comparison of pre- and post-scan schematics
- Comparison of RTL versus transistor layout
- Verification of minor changes in the design

Property checking

- The goal is to find an input assignment violating the assumed property.
- If such an assignment (*counter-example*!) exists then we can simulate it to obtain signal waveforms for debugging.
- If there exists no counter-example then we have proven that the implementation matches the property.

Property checking

Problems:

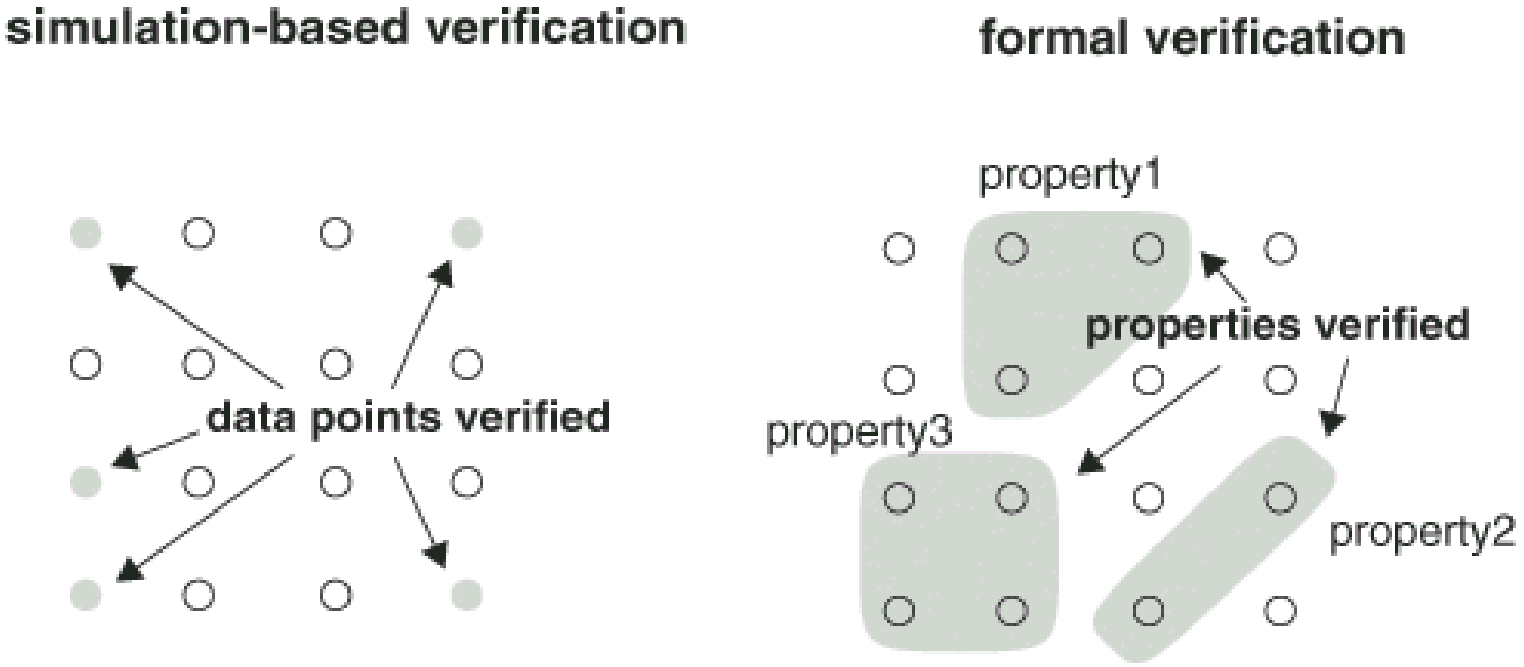
- How to extract only this portion of the design that is related to the property to be checked. Currently done manually...
- Selection of the properties to be checked is tricky...
- Bugs may occur in the implementation, in properties or in configuration (i.e. environment).
- Roughly 70 % of total effort for setting up the configuration.

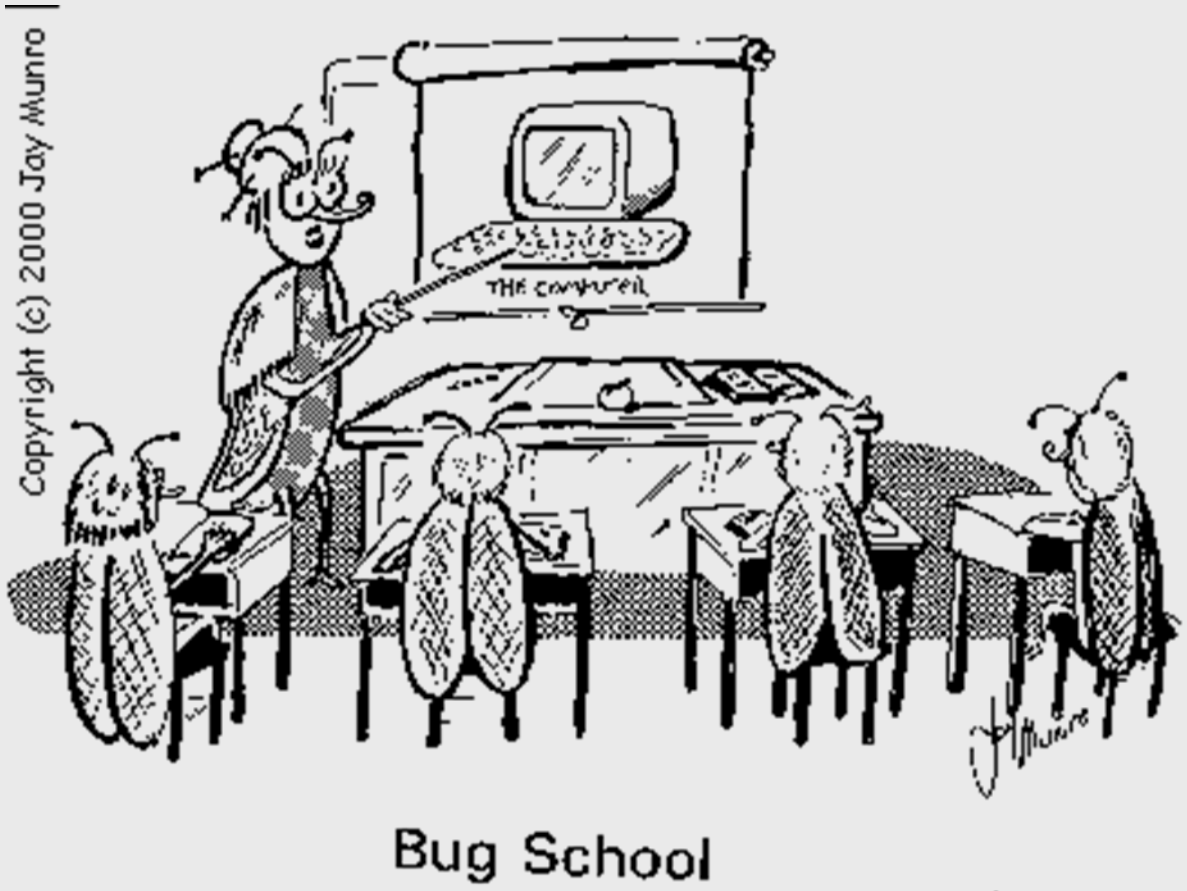
Property checking

Theorem provers

- Not too much automated but...
- ... can handle larger designs, require less memory.
- Use higher order logic. Thus, can check more complex properties.

Simulation-based vs formal





Bug School