Introduction to **Design Verification COMS31700**

Kerstin Eder

Design Automation and Verification

University of Bristol — Department of COMPUTER SCIENCE

Welcome to COMS31700

- Lecturer and Unit Director
  - Kerstin EDER
  - Department of Computer Science
  - Room 3.25 MVB
  - Kerstin.Eder@bristol.ac.uk
  - Office hours:
    - Talk to me or arrange a meeting after any of our lectures.
    - Alternatively, just come to my office.
    - Email may not get you a timely response, sorry.
- Lecture notes, exercises and assignments are/will be on Blackboard and also available at uobdv.github.io/Design-Verification/
- Comments and feedback are always welcome

2

COMS31700 Unit Details

- Lectures (weeks 2 – 12, but not week 8)
  - Please check your timetable, at the moment:
  - Tuesday (1h) 12:00 – 12:50 in QB 1.68
  - Tuesday (2h) 15:00 – 16:50 in MVB 1.11
- Practical Work (weeks 2 – 12)
  - You are expected to invest at least 2h per week into practical work.
  - On demand
    - DV scheduled lab, MVB 2.11, Thursdays 15:00 for 50 min
    - DV help session on demand, Fridays 13:00-13:50, MVB 1.11/QB1.68 (W4)
  - Teaching Assistant: Dave McEwan
- Assessment
  - 2 assignments (25% due in week 5, 25% due in week 12)
  - individual feedback session and assignment review seminar
    - Option to obtain "feed forward" *up to 5 working days before the deadline*
  - 1 exam (50% in January)

3

Literature and Study Resources

- **Writing Testbenches: Functional Verification of HDL Models** by Janick Bergeron. Second Edition, Kluwer, 2003.
- **Comprehensive Functional Verification** by Bruce Wile, John Goss and Wolfgang Roesner. Elsevier, 2005.
- verificationacademy.com
- In addition:
  - Lecture slides and on-line tutorials on COMS31700 web page
  - On-line documentation of ModelSim/Questa Simulator and SpecMan Elite
  - Watch the unit web page for further supplementary literature.

[**Credits:** Parts of the lecture notes contain material from the book "Comprehensive Functional Verification" by Bruce Wile etal, the book "Writing Testbenches: Functional Verification of HDL Models" by Janick Bergeron, the book "The Verilog Hardware Description Language" by Donald Thomas and from lecture slides developed at IBM (by Avi Ziv and Jaron Wolfstal), the University of Pittsburgh, Penn State University, North Carolina State University and Ohio State University. The HDL for the assignments has been developed at IBM.]

4

What is this unit about?

Aim: To familiarise you with the state of the art in Design Verification, and to give you the **technical background** plus some of the **practical skills** expected from a professional Design Verification Engineer.

- Pre-/Co-requisites: programming experience

On successful completion of this unit, you will be able to:

- understand the complexities and limits of verification;
- carry out functional verification and determine its effectiveness;
- set appropriate verification goals, select suitable verification methods and assess the associated risks;
- compile a verification plan that fits into the flow of a design project.

5

Unit Outline

Lecture Topics

- Introduction: What is Verification? What is a Testbench?
- Verification Flow and Tools including basic Verilog HDL coding
- Verification cycle, methodology and plan including coverage
- Simulation-based Verification: Stimulus Generation and Checking
- Assertion-based Verification (ABV)
- Advanced Testbench Design Methodology with SpecMan Elite
- (Functional Formal Verification and Property Checking)

Labs

- Exercise 1: Evita Verilog interactive tutorial (do @ home asap)
- Exercise 2: Introduction to the ModelSim/Questa Simulator
- A1, weeks 2-5: Verification of calculator design with ModelSim/Questa
- Exercise 3: How to collect Code Coverage with ModelSim/Questa
- Exercise 4: Introduction to SpecMan Elite
- A2, weeks 7-12: Advanced testbench design with SpecMan Elite

6

What is Design Verification?

What is Design Verification?

"Design Verification is the process used to gain confidence in the correctness of a design w.r.t. the requirements and specification."

Types of verification:

- Functional verification
- Timing verification
- ...
- What about performance?

8

Verification vs Validation

- Verification:
  - Confirms that a system has a given input / output behaviour, sometimes called the **transfer function** of a system.
- Validation:
  - Confirms that the system's transfer function results in the intended system behaviour when the system is employed in its target environment, e.g. as a component of an embedded system.
- Validation is sometimes used when verification is meant.

9

Why is Verification important?

- Verification is the single biggest lever to effect the triple constraints:
  - **Quality**
    - A high quality track record preserves revenue and reputation.
    - Ideally a team can establish a "right-first-time" track record.
  - **Cost**
    - Fewer revisions through the development/fabrication process means lower costs.
    - Respinning a chip costs hundreds of thousands of £/$/€
      - + the associated "lost opportunity" costs.
  - **Timing/Schedule**
    - Fewer revisions through the development/fabrication process means faster time-to-market.
    - Respinning a chip costs 6-8 weeks at least
      - + the associated "lost opportunity" costs.

10

All about Bugs

Types of bugs

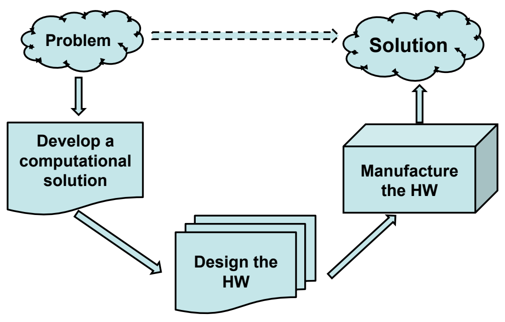
How are bugs introduced?

How can bugs be found?

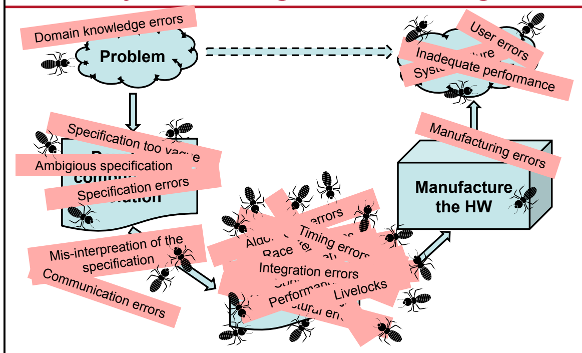
Why do Designs have Bugs?

12

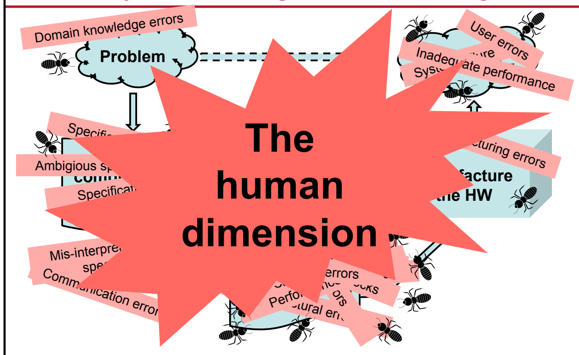
Why do Designs have Bugs?



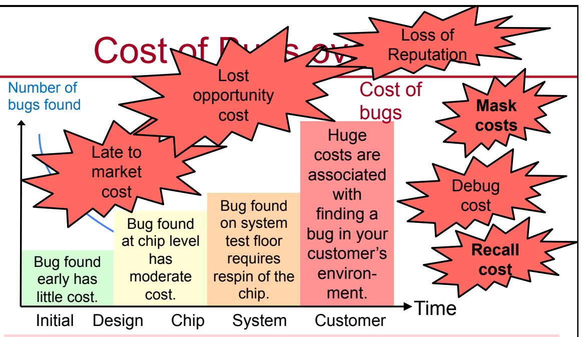
Why do Designs have Bugs?



Why do Designs have Bugs?



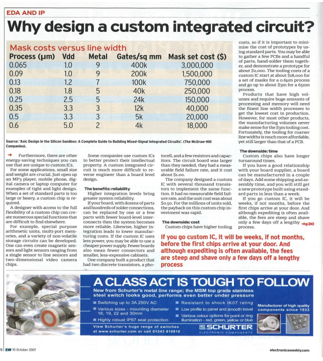
Cost of



The longer a bug goes undetected,
the more expensive it is!

Remember the Intel Pentium FDIV bug!
http://en.wikipedia.org/wiki/Pentium_FDIV_bug

Mask costs (Electronics Weekly, 10 October 2007)



EDA AND IP

Why design a custom integrated circuit?

Mask costs versus line width

Process (µm)	Vdd	Metal	Gates/sq mm	Mask set cost ($)
0.065	1.0	9	400k	3,000,000
0.09	1.0	9	200k	1,500,000
0.13	1.2	7	100k	750,000
0.18	1.8	5	40k	250,000
0.25	2.5	5	24k	150,000
0.35	3.3	3	12k	40,000
0.5	3.3	3	5k	20,000
0.6	5.0	2	4k	18,000

Source: 'Asic Design in the Silicon Sandbox: A Complete Guide to Building Mixed-Signal Integrated Circuits'. (The McGraw-Hill Companies).

→ Furthermore, there are other energy-saving techniques you can use that are unique to custom ICs.

For some applications, small size and weight are crucial. Just open up an MP3 player, mobile phone, digital camera or laptop computer for examples of tight and light design. When a set of standard parts is too large or heavy, a custom chip is required.

A designer with access to the full flexibility of a custom chip can create numerous special functions that are difficult to find elsewhere.

For example, special purpose arithmetic units, multi-port memories, and a variety of non-volatile storage circuits can be developed. One can even create magnetic sensors and light sensors ranging from a single sensor to line sensors and two-dimensional video camera chips.

Some companies use custom ICs to better protect their intellectual property. A custom integrated circuit is much more difficult to reverse engineer than a board level design.

The benefits: reliability

Higher integration levels bring greater system reliability.

If your board, with dozens of parts and hundreds of solder connections, can be replaced by one or a few parts with fewer board-level interconnects then the system becomes more reliable. Likewise, higher integration leads to lower manufacturing costs. If the custom IC uses less power, you may be able to use a cheaper power supply. Fewer boards also mean fewer connectors and smaller, less-expensive cabinets.

One company built a product that had two discrete transistors, a photocell, and a few resistors and capacitors. The circuit board was larger than they needed, they had a measurable field failure rate, and it cost about $1.00.

The company designed a custom IC with several thousand transistors to implement the same function. It had no measurable field failure rate, and the unit cost was about $0.50. For the millions of units sold, the payback on this custom chip investment was rapid.

The downside: cost

Custom chips have higher tooling costs, so if it is important to minimise the cost of prototypes by using standard parts. You may be able to gather a few PCBs and a handful of parts, hand-solder them together, and demonstrate a prototype for about $2,000. The tooling costs of a custom IC start at about $18,000 for a set of masks for a 0.6µm process and go up to about $3m for a 65nm process.

Products that have high volumes and require huge amounts of processing and memory will need the finest line width processes to get the lowest cost in production. However, for most other products, the manufacturing volumes never make sense for the $3m tooling cost. Fortunately, the tooling for coarser line widths is much more affordable, yet still larger than that of a PCB.

The downside: time

Custom chips also have longer turnaround times.

If you have a good relationship with your board supplier, a board can be manufactured in a couple of days. Add some shipping and assembly time, and you will still get a new prototype built using standard parts in less than a week.

If you go custom IC, it will be weeks, if not months, before the first chips arrive at your door. And although expediting is often available, the fees are steep and shave only a few days off a lengthy process. →p30

If you go custom IC, it will be weeks, if not months, before the first chips arrive at your door. And although expediting is often available, the fees are steep and shave only a few days off a lengthy process

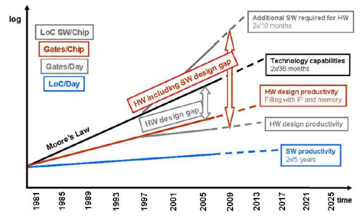
A CLASS ACT IS TOUGH TO FOLLOW
New from Schurter's metal line range; the MSM top grade stainless steel switch looks good, performs even better under pressure
- Switching up to 3A 250V AC
- Various sizes - mounting diameter 16, 19, 22 and 30mm
- Highly robust IP67 seal protection
- Resistant to shock IK07 rating
- Low profile to panel and smooth travel
- Various colour options for point or ring illumination - red, green, yellow or blue

Manufacturer of high quality components since 1933
View Schurter's huge range of switches at www.schurter.com or call 01243 810810
SCHURTER ELECTRONIC COMPONENTS

28 EW 10 October 2007 electronicsweekly.com

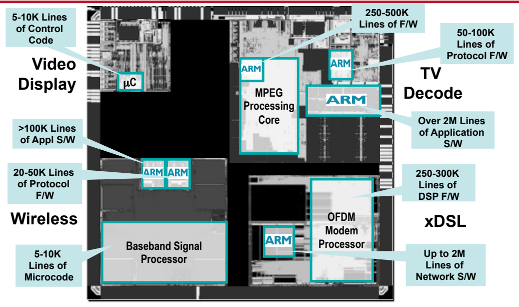
Increasing Design Complexity vs tight TTM Constraints

ITRS Edition 2009, Design Chapter (http://www.itrs.net/ and http://www.itrs2.net/)
– Hardware and Software Design Gaps versus Time



Getting it right (first time) is more and more difficult:
- rapidly increasing design complexity
- tight "time-to-market" constraints

Increasing Design Complexity

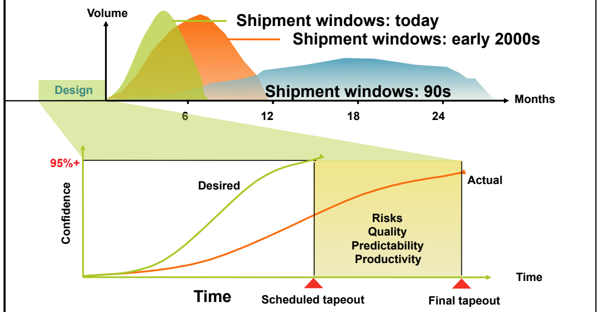

5-10K Lines of Control Code
Video Display
µC
250-500K Lines of F/W
ARM
ARM
MPEG Processing Core
ARM
50-100K Lines of Protocol F/W
TV Decode
>100K Lines of Appl S/W
Over 2M Lines of Application S/W
20-50K Lines of Protocol F/W
ARM ARM
250-300K Lines of DSP F/W
Wireless
xDSL
OFDM Modem Processor
ARM
5-10K Lines of Microcode
Baseband Signal Processor
Up to 2M Lines of Network S/W

Multiple Power Domains, Security, Virtualisation
Nearly five million lines of code to enable Media gateway

Increasing Design Complexity

- From mobile phone to smart phone

Shorter Time-To-Market Windows



Increasing Verification Productivity

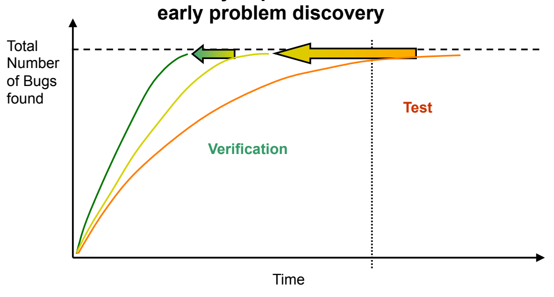
Need to minimise verification time e.g. by using:

- **Parallelism:** Add more resources
- **Abstraction:**
  - Higher level of abstraction (i.e. C vs Assembly)
  - This often means a reduction of control!
- **Automation:**
  - Tools to automate standard processes
  - Requires standard processes/methodology.
  - Usually a variety of functions, interfaces, protocols, and transformations must be verified.
  - Not all (verification) processes can be automated.

Productivity improvements drive early problem discovery!

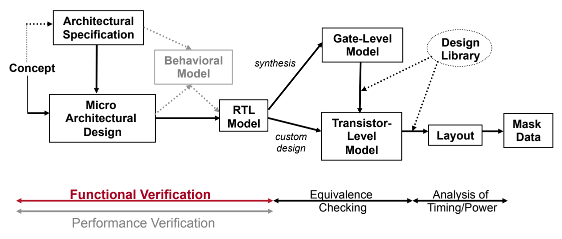
Increasing Verification Productivity

Productivity improvements drive early problem discovery



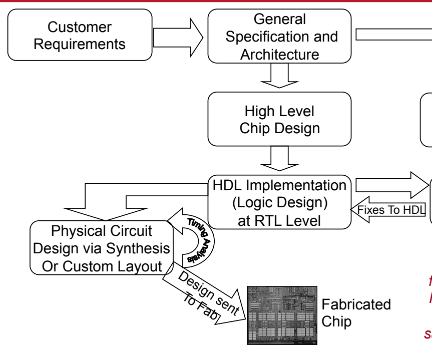
Verification in the IC Design Process

The IC Design Process



Functional verification aims to demonstrate that the functional intent of a design is preserved in its implementation.

Chip Design Process



***Result:** Description with all details for fabrication (tape-out). In practice, all "steps" start (almost) at the same time - they run in parallel!*

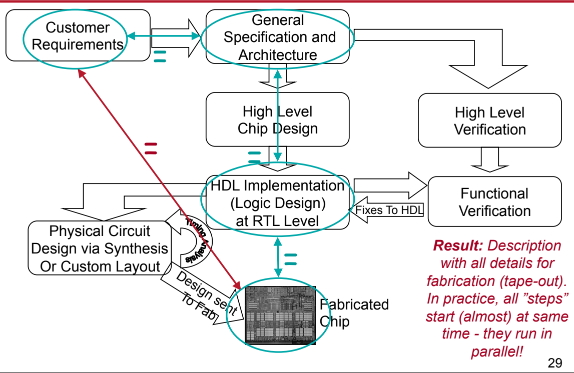
Role of Verification in IC Design

IC design process is complex:

- **Engineers need to balance conflict of interest:**
  - Tight time-to-market constraints vs. increasing design complexity
- **Aim:** "Right-first-time" design, "correct-by-construction"
- More and more time-consuming to obtain acceptable level of confidence in correctness of design!
- **design time << verification time**
  - Remember: Verification does not create value!
    - But it preserves revenue and reputation!
  - Up to 70% of design effort can go into verification.
  - 80% of all written code is often in the verification environment.
  - Properly staffed design teams have dedicated verification engineers.
  - In some cases verification engineers outnumber designers 2:1.

What are you going to verify?

Chip Design Process



***Result:** Description with all details for fabrication (tape-out). In practice, all "steps" start (almost) at same time - they run in parallel!*

How do Designers know whether a circuit is correct?

Concept
↓ → *VERIFY: What you specified is what you envisioned*
Specification
↓ → *VERIFY: What you designed is what you specified*
HDL Design (RTL)
↓ → *VERIFY: What you taped out is what the RTL describes*
Tape-out
↓ → *TEST: What was manufactured is what you taped out*
Silicon

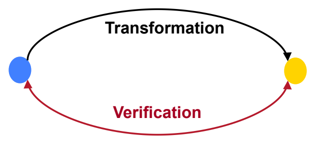
There is skill, science and methodology behind verification.

Reconvergence Models [Bergeron]

Conceptual representation of the verification process

- Most important question:

What are you verifying?

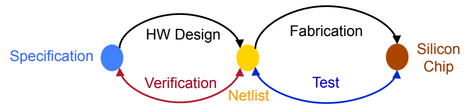


- Purpose of verification is to ensure that the result of some transformation is as intended or as expected.

31

Verification vs. Test

- **Often confused in the context of HW design!**
  - Purpose of test is to show design was **manufactured** properly.
  - Verification is done to ensure that **design meets its functional intent prior to manufacture!**



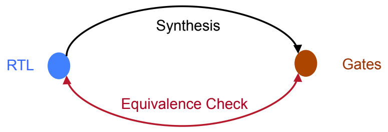
- One test method is scanning: **"Design for Test"** methodology
  - Link all internal registers together into a chain.
  - Chain accessible from chip pins.
  - Allows control/observation of internal state.
  - Impacts area of design, but keeps testing cost down.
- Why not **"Design for Verification"?** [Hot topic of research!]
- @DT, consider: What is design supposed to do? How will it be verified?

32

Formal: Equivalence Checking

Compares two models to check for equivalence.

- Proves mathematically that both are *logically equivalent*.
  - Commonly used on **lower levels** of design process.
- Example: RTL to Gates (Post Synthesis)

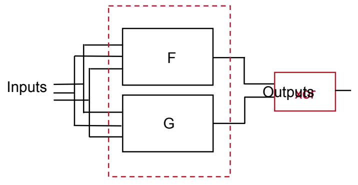


Why do equivalence checking when EDA tools exist for synthesis?

- See "*HDL Chip Design - A Practical Guide for Designing, Synthesising, and Simulating ASICs and FPGAs using VHDL or Verilog*" book by Douglas Smith page 136 and compare MUX spec with what they claim will be synthesised!

33

Equivalence Checking



- Conceptually, we are asking the question:

  "Is there an input vector such that the output of the XOR gate can be 1"?

34

Cost of Verification

Necessary Evil

- Always takes too long and costs too much.
- As number of bugs found decreases, cost and time of finding remaining ones increases.

So when is verification done? **(Will investigate this later!)**

- Remember: Verification does not generate revenue!

Yet indispensable

- To create revenue, design must be functionally correct and provide benefits to customer.
- Proper functional verification demonstrates **trustworthiness** of the design.
- Right-first-time designs demonstrate **professionalism** and "increase" reputation of design team.

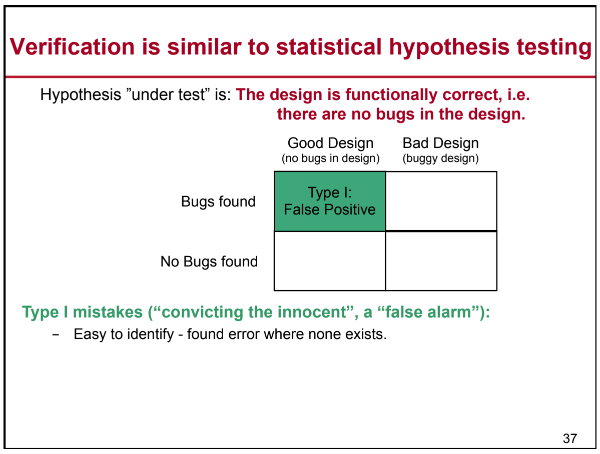
35

Verification is similar to statistical hypothesis testing

Hypothesis "under test" is: **The design is functionally correct, i.e. there are no bugs in the design.**

	Good Design (no bugs in design)	Bad Design (buggy design)
Bugs found		
No Bugs found		

36



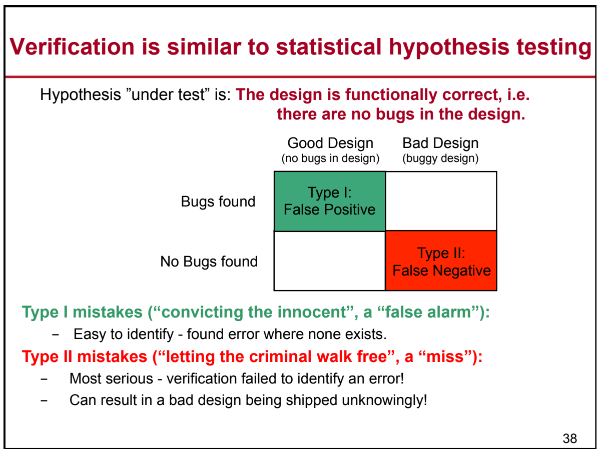
Verification is similar to statistical hypothesis testing

Hypothesis "under test" is: **The design is functionally correct, i.e. there are no bugs in the design.**

	Good Design (no bugs in design)	Bad Design (buggy design)
Bugs found	Type I: False Positive	
No Bugs found		

Type I mistakes ("convicting the innocent", a "false alarm"):
- Easy to identify - found error where none exists.

37



Verification is similar to statistical hypothesis testing

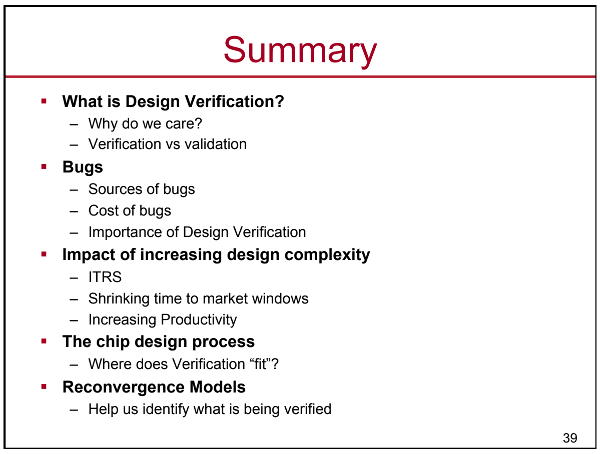
Hypothesis "under test" is: **The design is functionally correct, i.e. there are no bugs in the design.**

	Good Design (no bugs in design)	Bad Design (buggy design)
Bugs found	Type I: False Positive	
No Bugs found		Type II: False Negative

Type I mistakes ("convicting the innocent", a "false alarm"):
- Easy to identify - found error where none exists.

Type II mistakes ("letting the criminal walk free", a "miss"):
- Most serious - verification failed to identify an error!
- Can result in a bad design being shipped unknowingly!

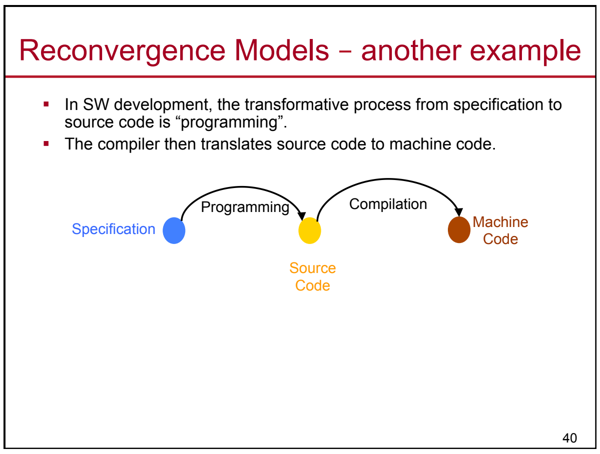
38



Summary

- **What is Design Verification?**
  - Why do we care?
  - Verification vs validation
- **Bugs**
  - Sources of bugs
  - Cost of bugs
  - Importance of Design Verification
- **Impact of increasing design complexity**
  - ITRS
  - Shrinking time to market windows
  - Increasing Productivity
- **The chip design process**
  - Where does Verification "fit"?
- **Reconvergence Models**
  - Help us identify what is being verified

39

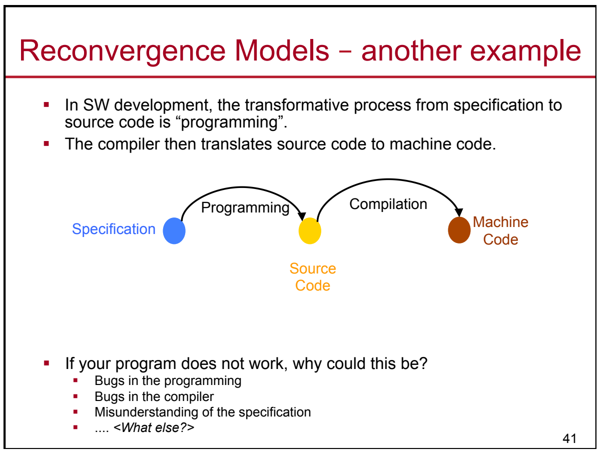


Reconvergence Models – another example

- In SW development, the transformative process from specification to source code is "programming".
- The compiler then translates source code to machine code.

$$\begin{tikzcd}\text{Specification} \arrow[r, bend left, "\text{Programming}"] & \text{Source Code} \arrow[r, bend left, "\text{Compilation}"] & \text{Machine Code}\end{tikzcd}$$

40



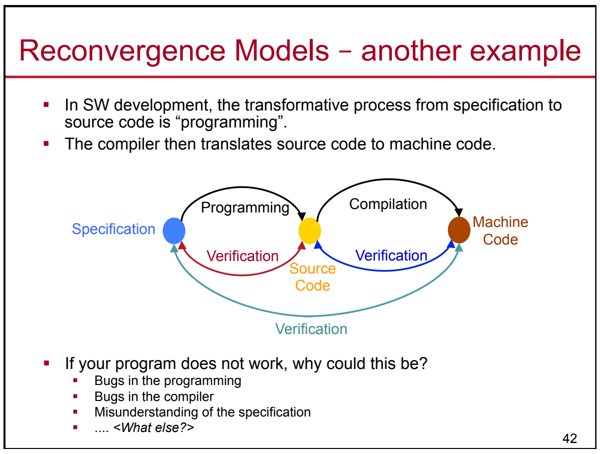
Reconvergence Models – another example

- In SW development, the transformative process from specification to source code is "programming".
- The compiler then translates source code to machine code.

$$\begin{tikzcd}\text{Specification} \arrow[r, bend left, "\text{Programming}"] & \text{Source Code} \arrow[r, bend left, "\text{Compilation}"] & \text{Machine Code}\end{tikzcd}$$

- If your program does not work, why could this be?
  - Bugs in the programming
  - Bugs in the compiler
  - Misunderstanding of the specification
  - .... *<What else?>*

41



Reconvergence Models – another example

- In SW development, the transformative process from specification to source code is "programming".
- The compiler then translates source code to machine code.

$$\begin{tikzcd}\text{Specification} \arrow[r, bend left, "\text{Programming}"] & \text{Source Code} \arrow[l, bend left, "\text{Verification}"] \arrow[r, bend left, "\text{Compilation}"] & \text{Machine Code} \arrow[l, bend left, "\text{Verification}"] \arrow[ll, bend left=50, "\text{Verification}"]\end{tikzcd}$$

- If your program does not work, why could this be?
  - Bugs in the programming
  - Bugs in the compiler
  - Misunderstanding of the specification
  - .... *<What else?>*

42