Introduction to **Design Verification** **COMS30026**

Kerstin Eder

Trustworthy Systems Lab

University of BRISTOL

Department of COMPUTER SCIENCE

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?

Annex A
(informative)

The Role of Testing in Verification and Validation

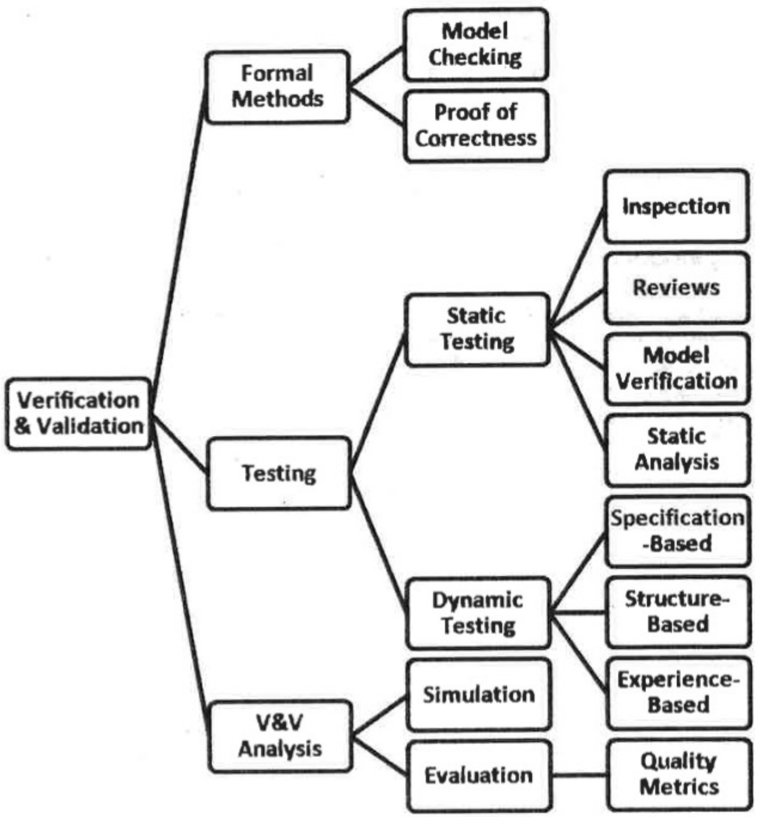


Figure 11 — Hierarchy of Verification and Validation activities

Figure 11 defines the complete nature of verification and validation (V&V) activities. V&V can be done on system, hardware, and software products. These activities and planning are defined and refined in IEEE 1012 and ISO/IEC 12207. Much of V&V is accomplished by testing. The ISO/IEC 29119 standard addresses the Dynamic and Static software testing (directly or via reference), thus covering parts of this verification and validation model. ISO/IEC 29119 is not intended to address all the elements of the V&V model, but it is important for a tester to understand where they fit within this model.

DRAFT INTERNATIONAL STANDARD ISO/IEC DIS 29119-1

ISO/IEC JTC **1** Secretariat: **ANSI**

Voting begins on **2012-10-09** Voting terminates on **2013-01-09**

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Software and systems engineering — Software testing —

Part 1:
Concepts and definitions

Ingénierie du logiciel et des systèmes — Essais du logiciel —
Partie 1: Concepts et définitions

ICS 35.080

To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.

Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

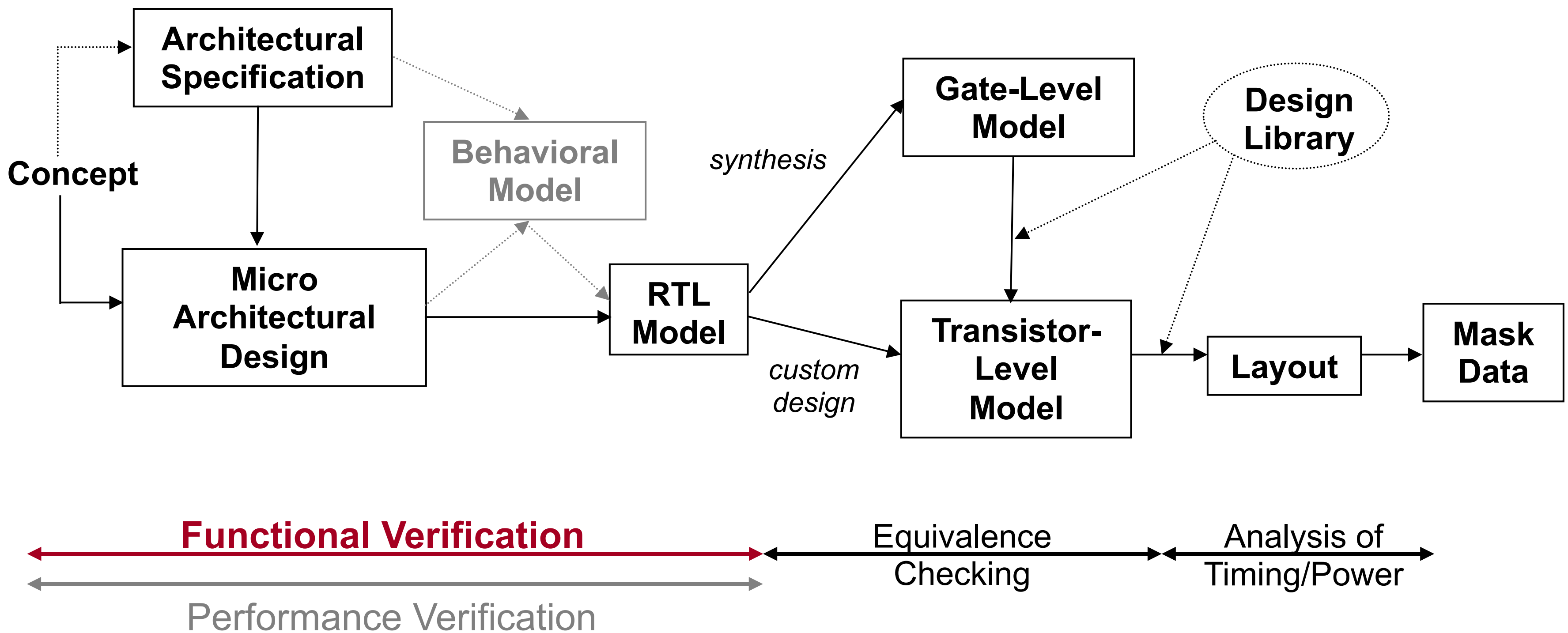
RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

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

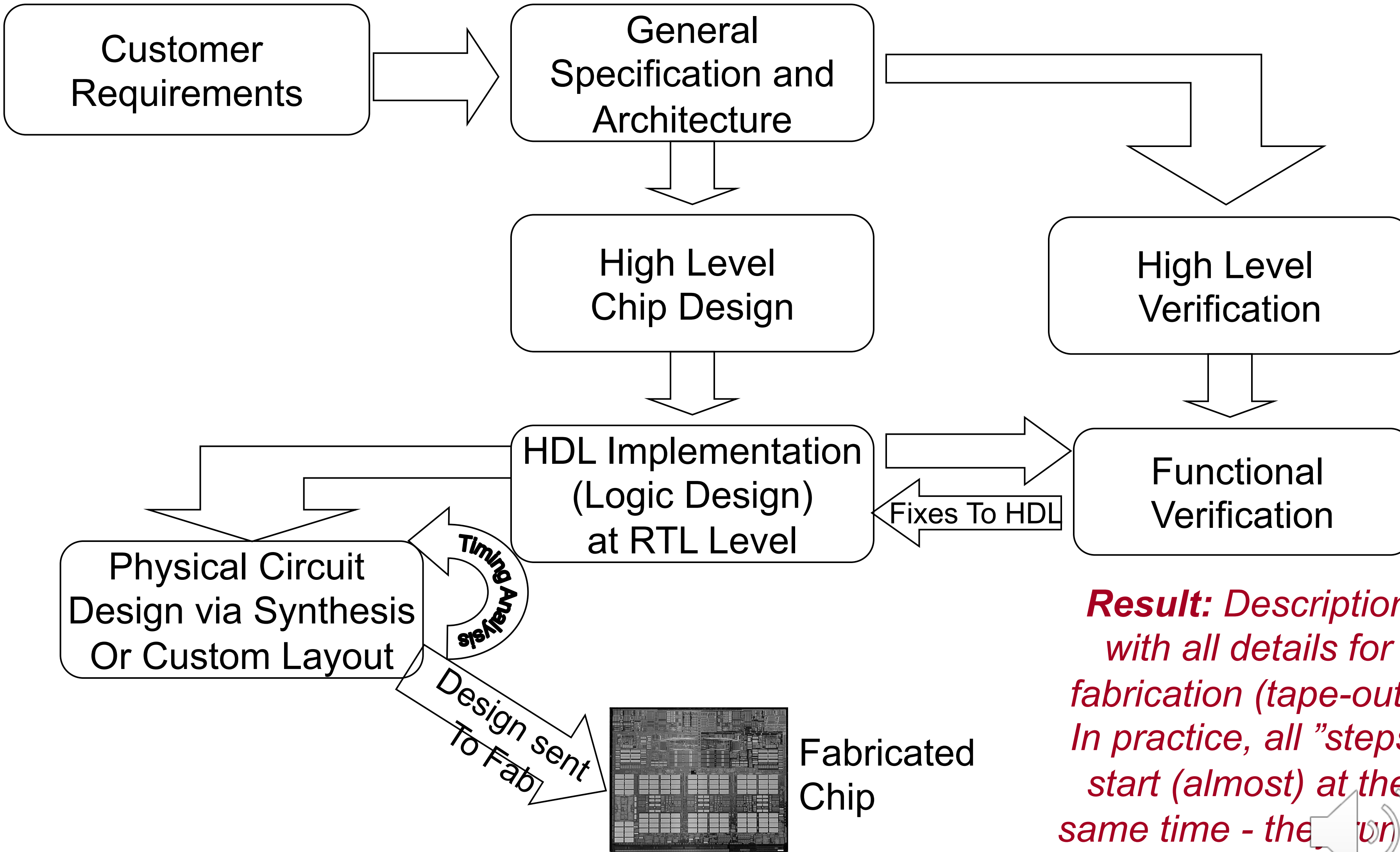
Verification in the IC Design Process

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!*

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.

All about Bugs

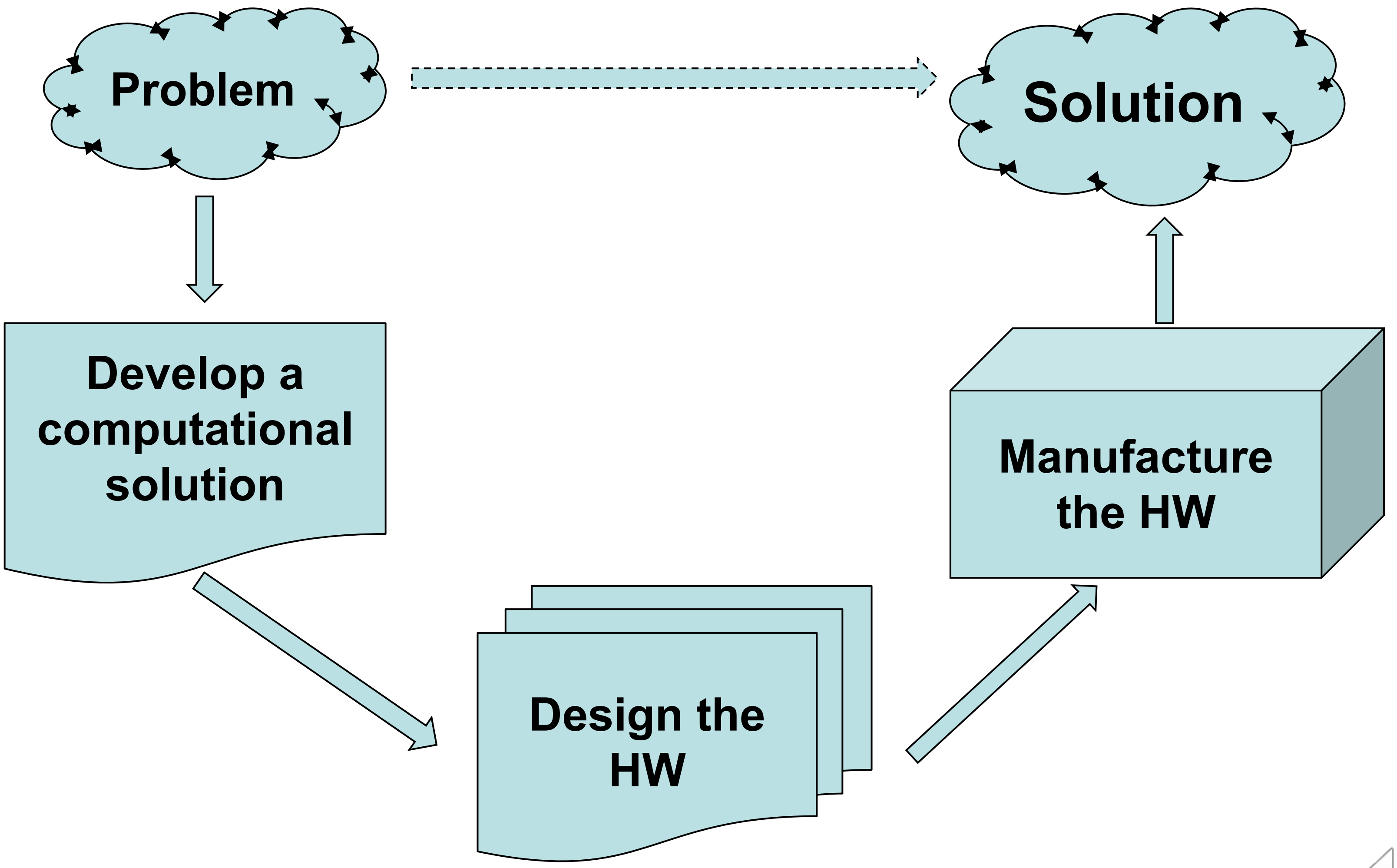
Types of bugs

How are bugs introduced?

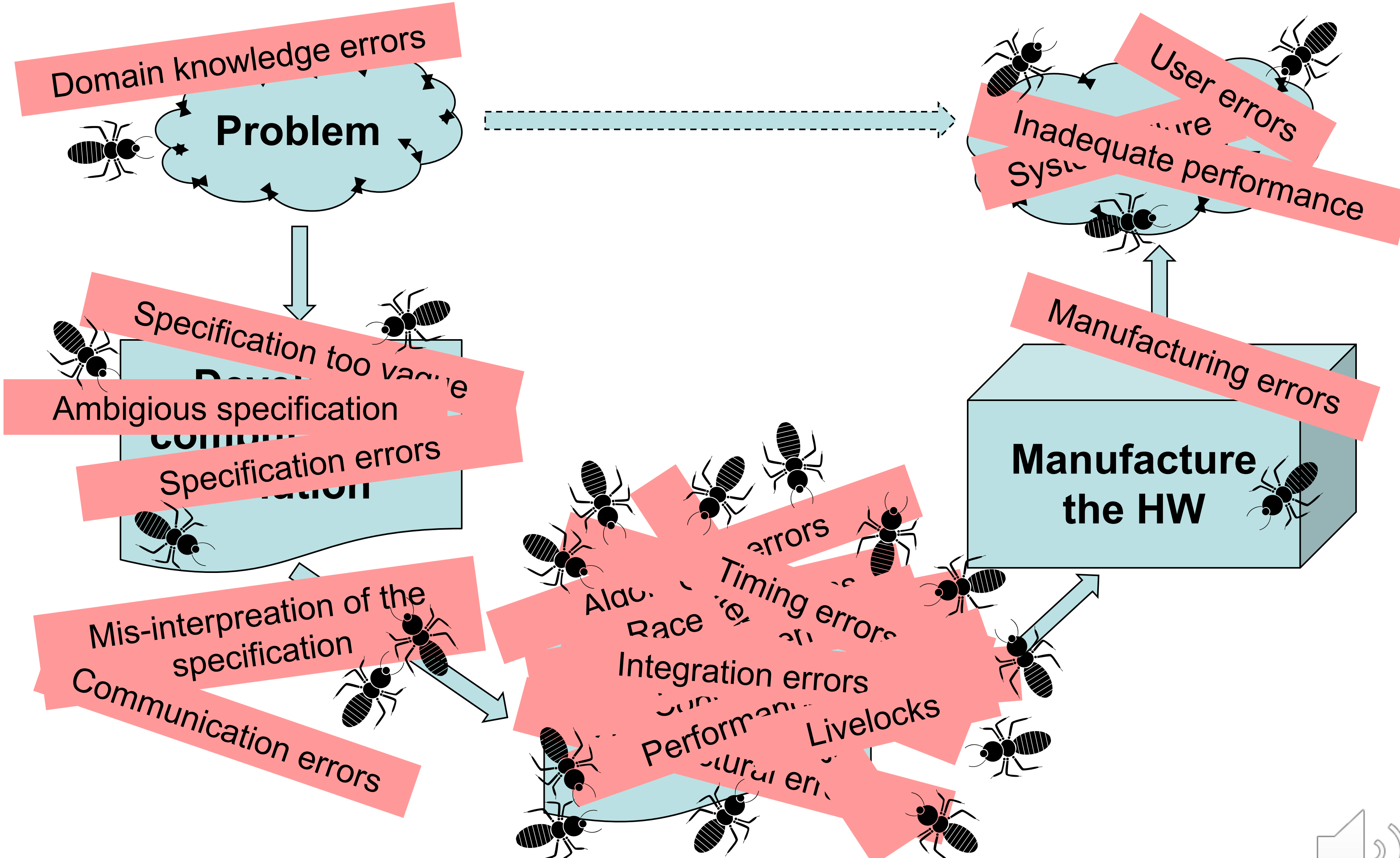
How can bugs be found?

Why do Designs have Bugs?

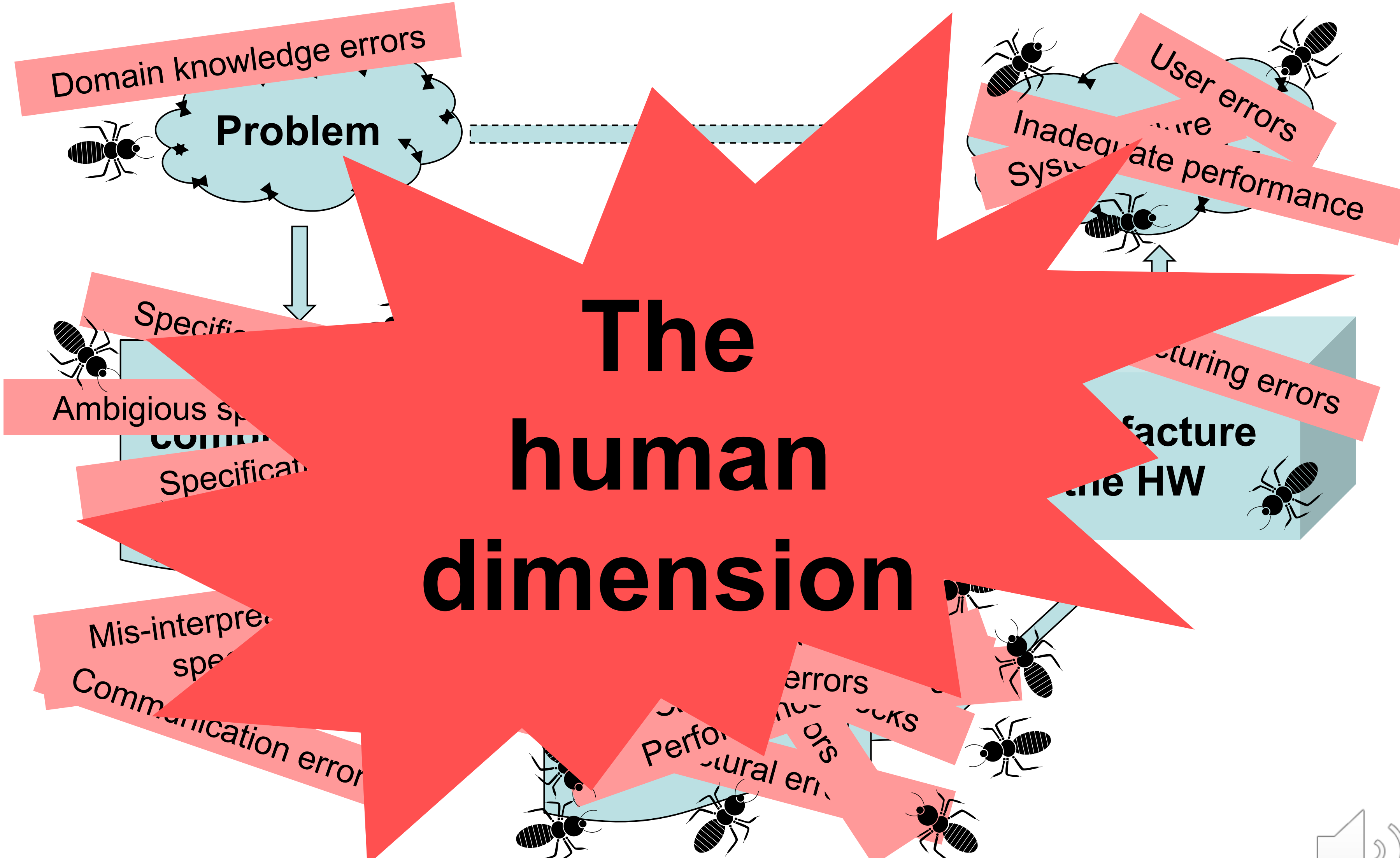
Why do Designs have Bugs?



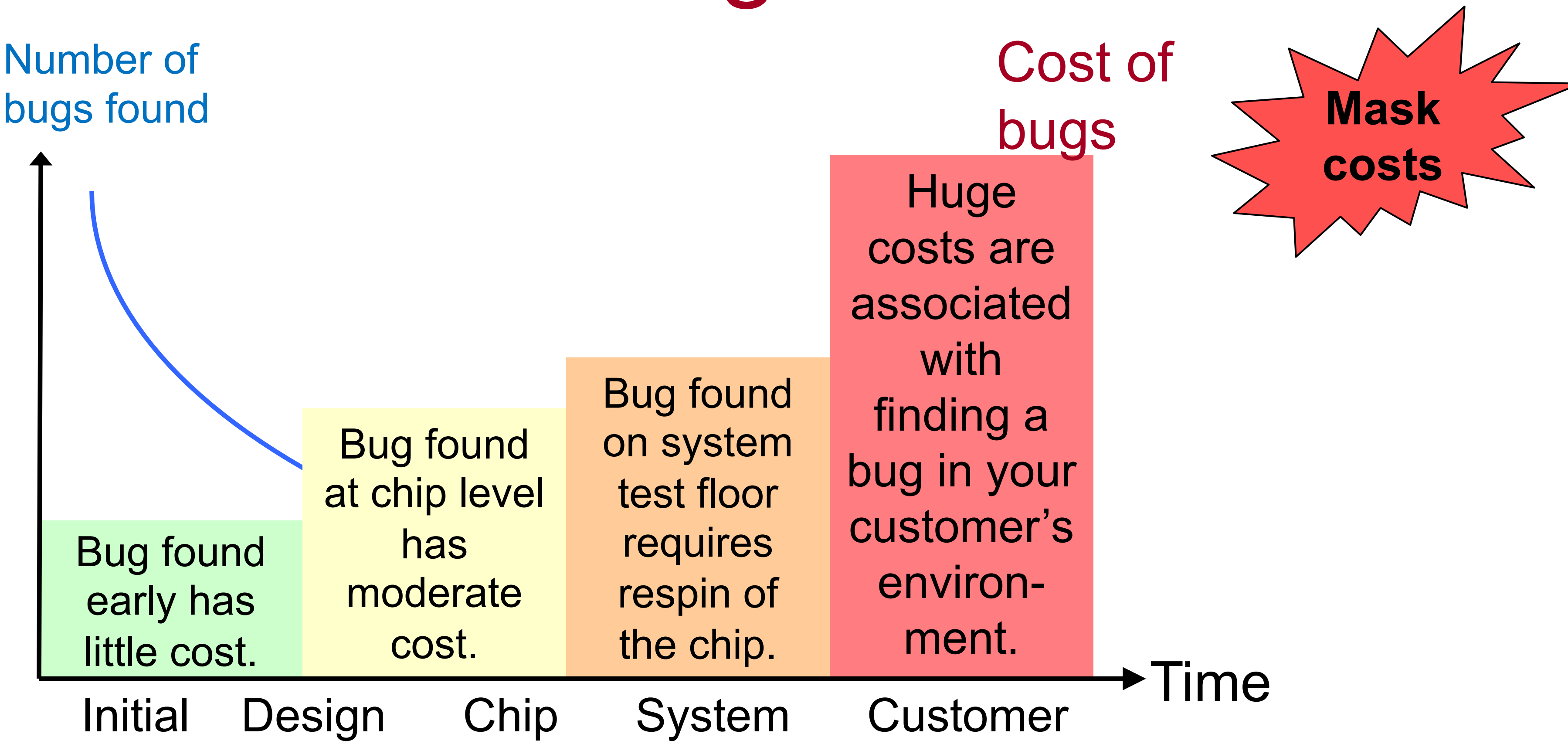
Why do Designs have Bugs?



Why do Designs have Bugs?



Cost of Bugs over Time



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.

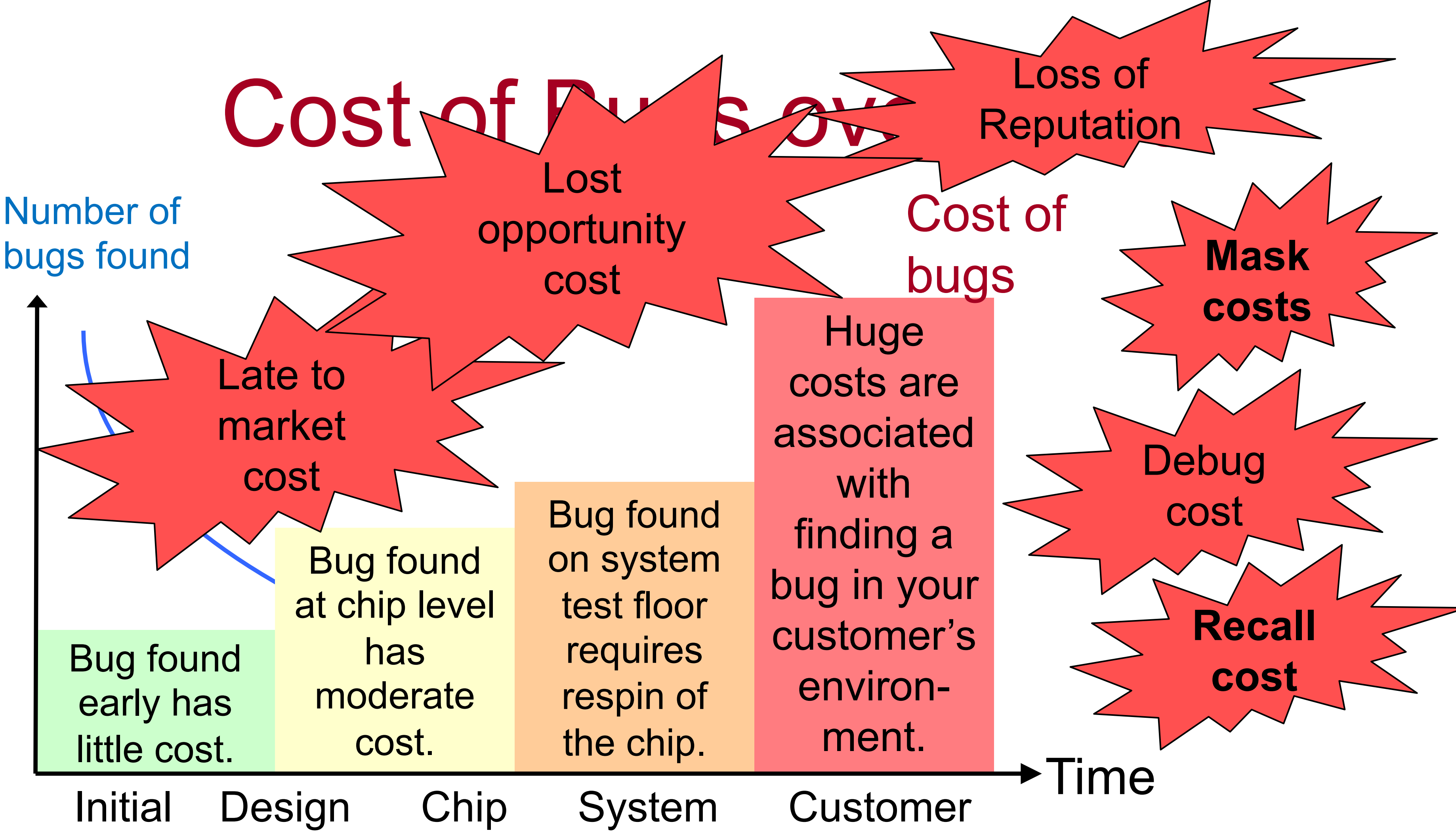
The downside: time

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

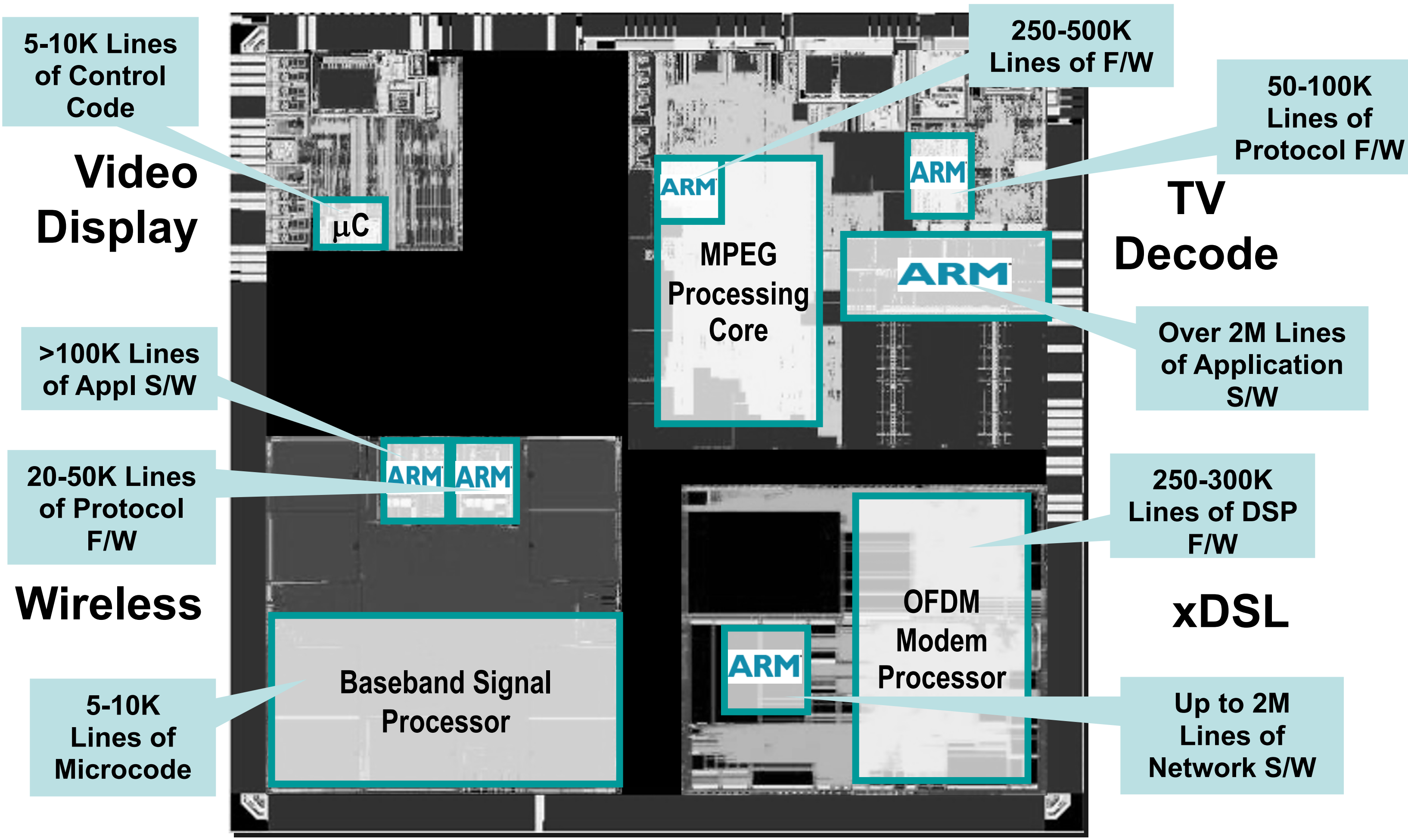


The longer a bug goes undetected, the more expensive it is!

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

Increasing Design Complexity

Increasing Design Complexity

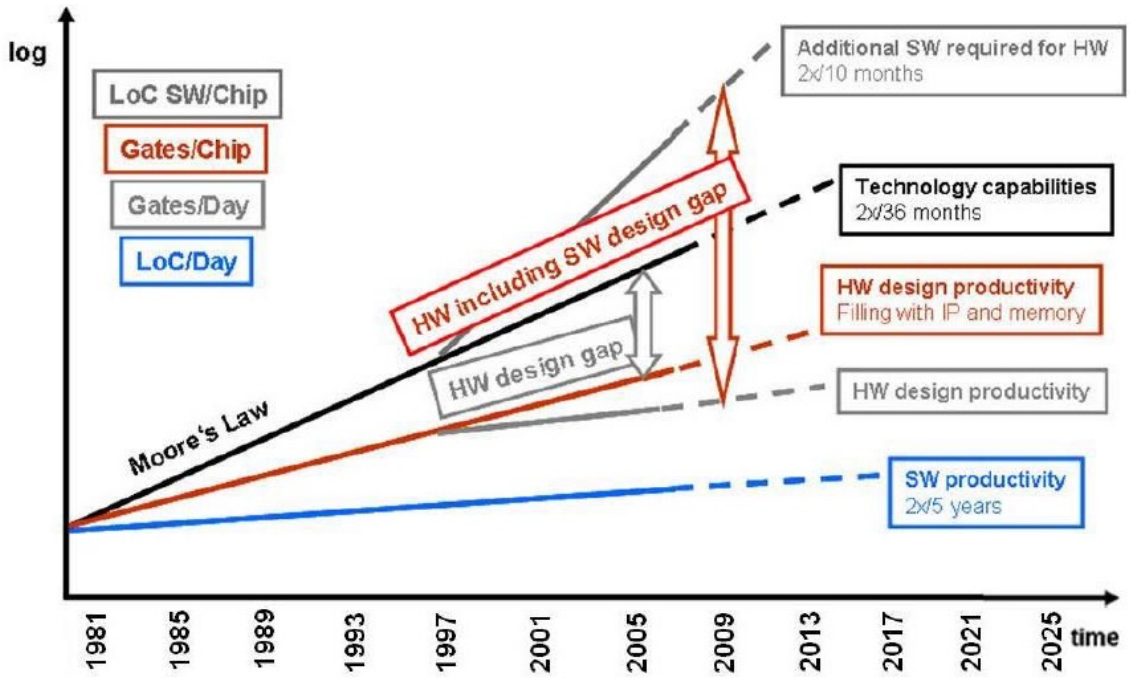


Multiple Power Domains, Security, Virtualisation
Nearly five million lines of code to enable Media gateway

Increasing Design Complexity: Moor’s Law

ITRS Edition 2009, Design Chapter (http://www.itrs.net/ and http://www.itrs2.net/)

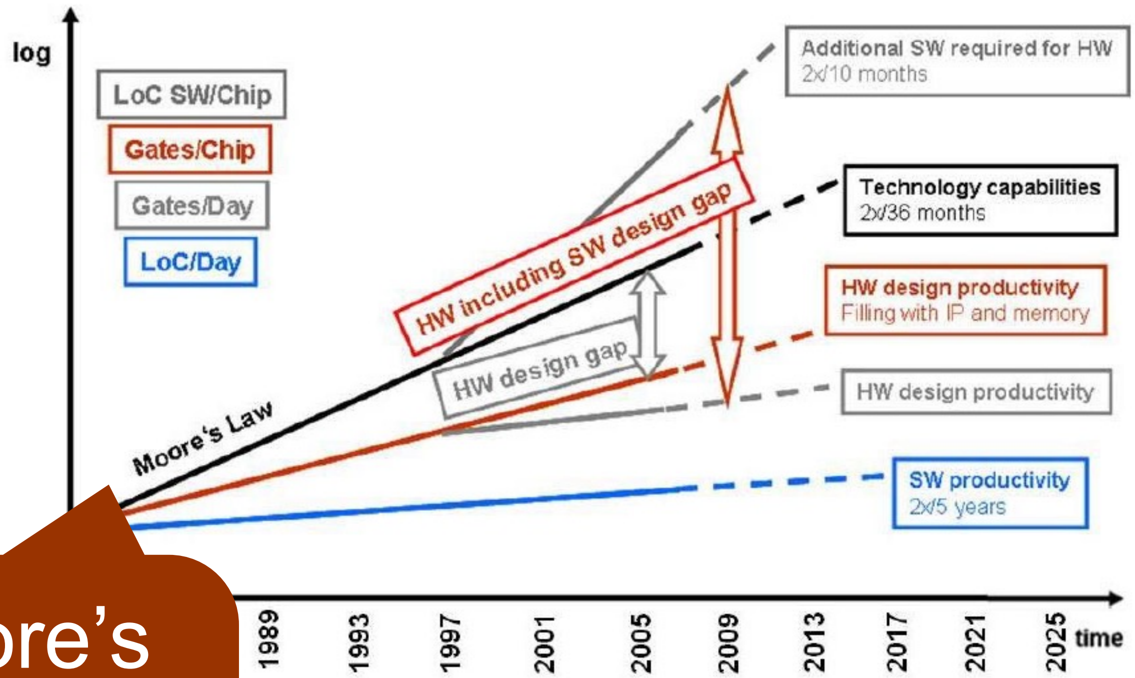
– Hardware and Software Design Gaps versus Time



Increasing Design Complexity: Moor’s Law

ITRS Edition 2009, Design Chapter (http://www.itrs.net/ and http://www.itrs2.net/)

– Hardware and Software Design Gaps versus Time

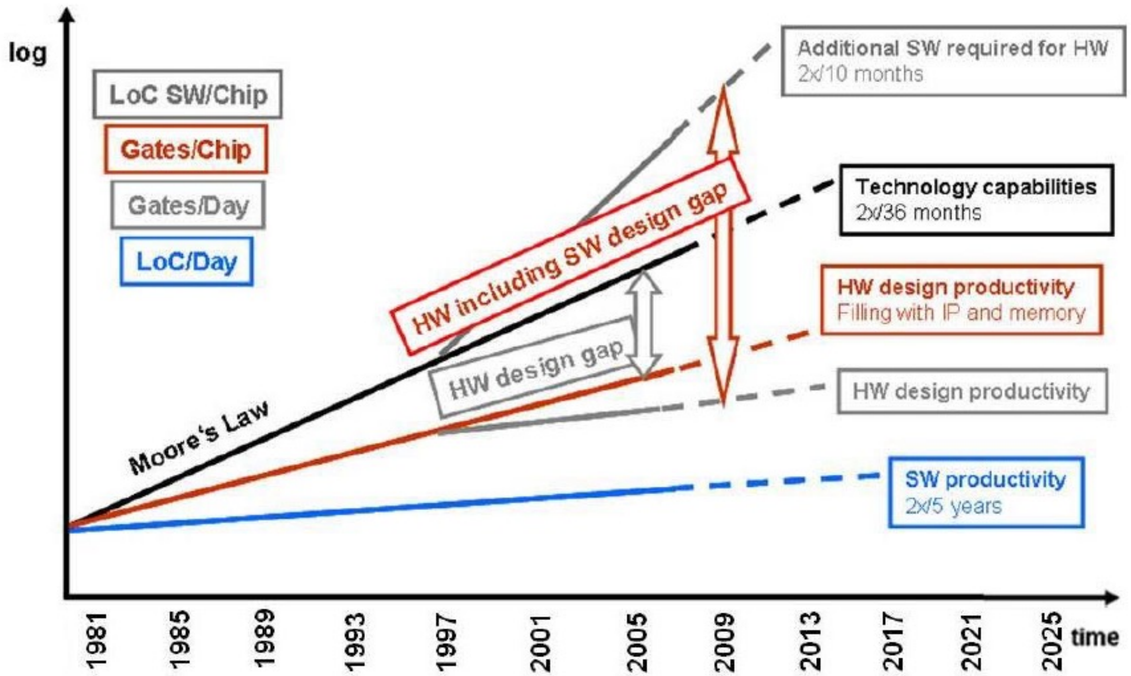


Is Moore’s Law a law of physics?

Increasing Design Complexity: Moor's Law

ITRS Edition 2009, Design Chapter (http://www.itrs.net/ and http://www.itrs2.net/)

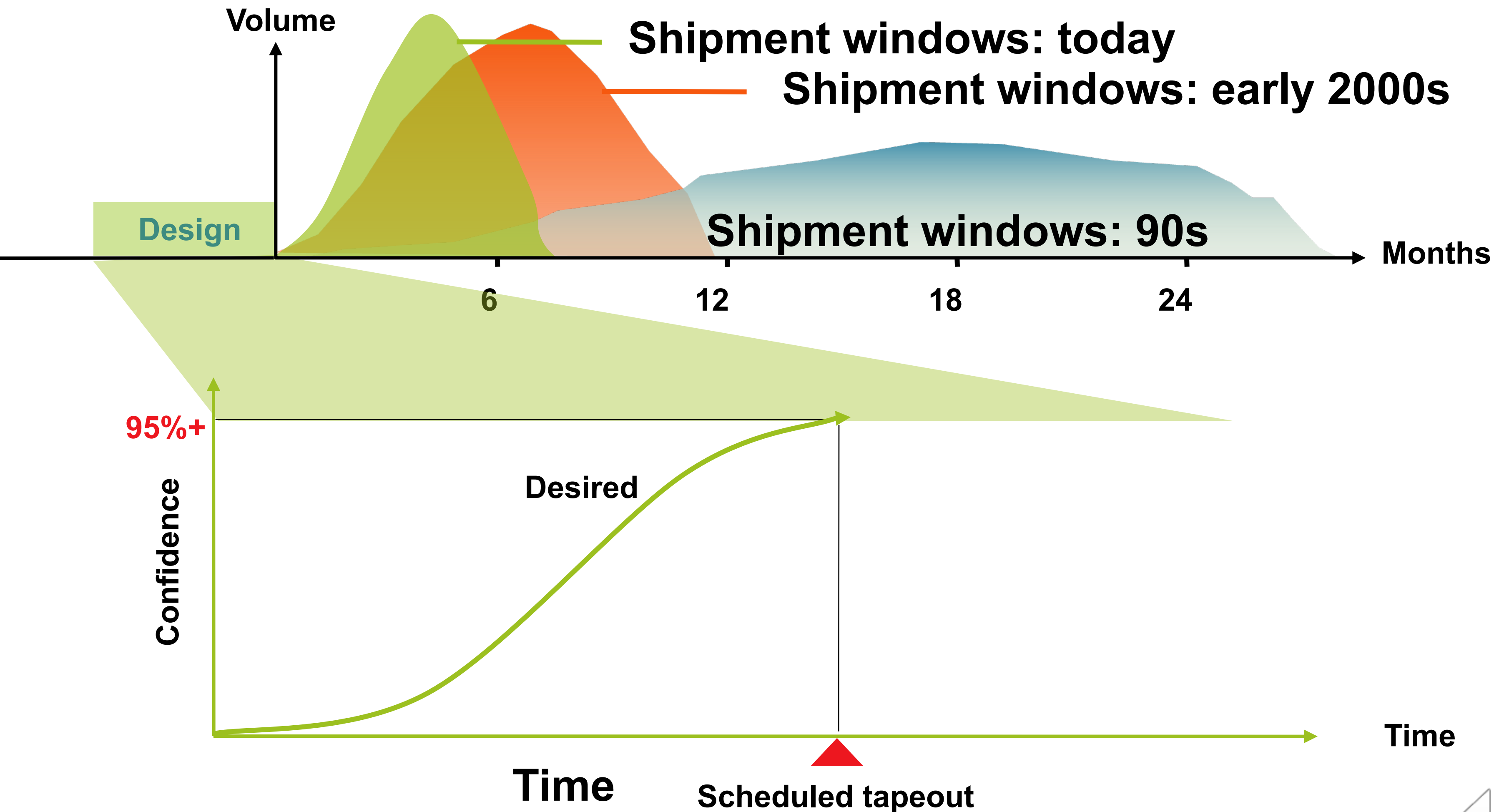
– Hardware and Software Design Gaps versus Time



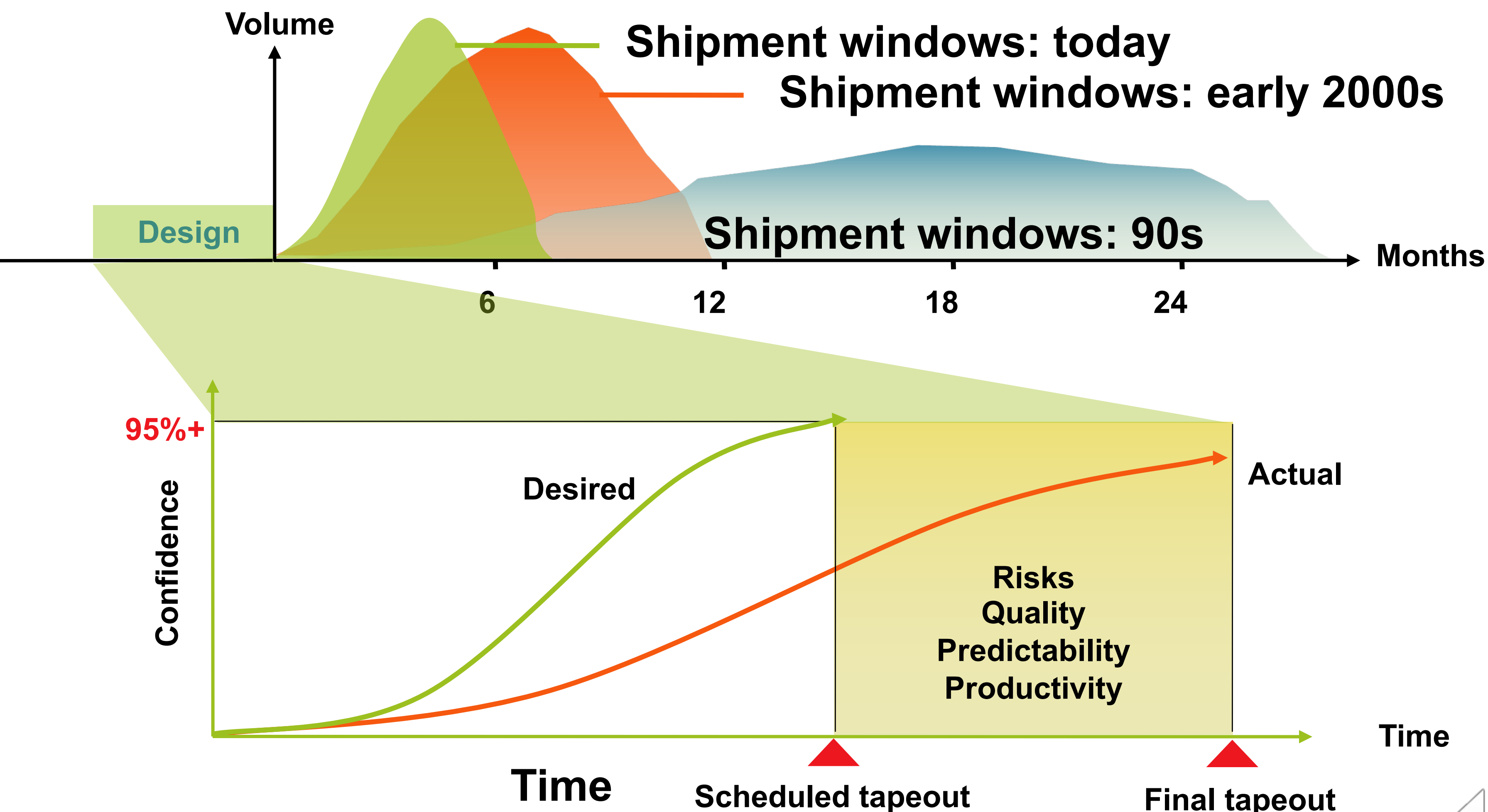
Getting it right (first time) is more and more difficult:

– rapidly increasing design complexity
– tight "time-to-market" constraints

Shorter Time-To-Market Windows



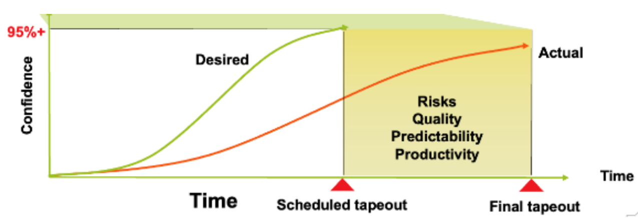
Shorter Time-To-Market Windows



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**
  - Up to 70% of design effort can go into verification.
  - 80% of all written code is often in the verification environment.
  - Remember: Verification does not create value!
    - But it preserves revenue and reputation!
  - Properly staffed design teams have dedicated verification engineers.
  - In some cases verification engineers outnumber designers 2:1.



Increasing Verification Productivity

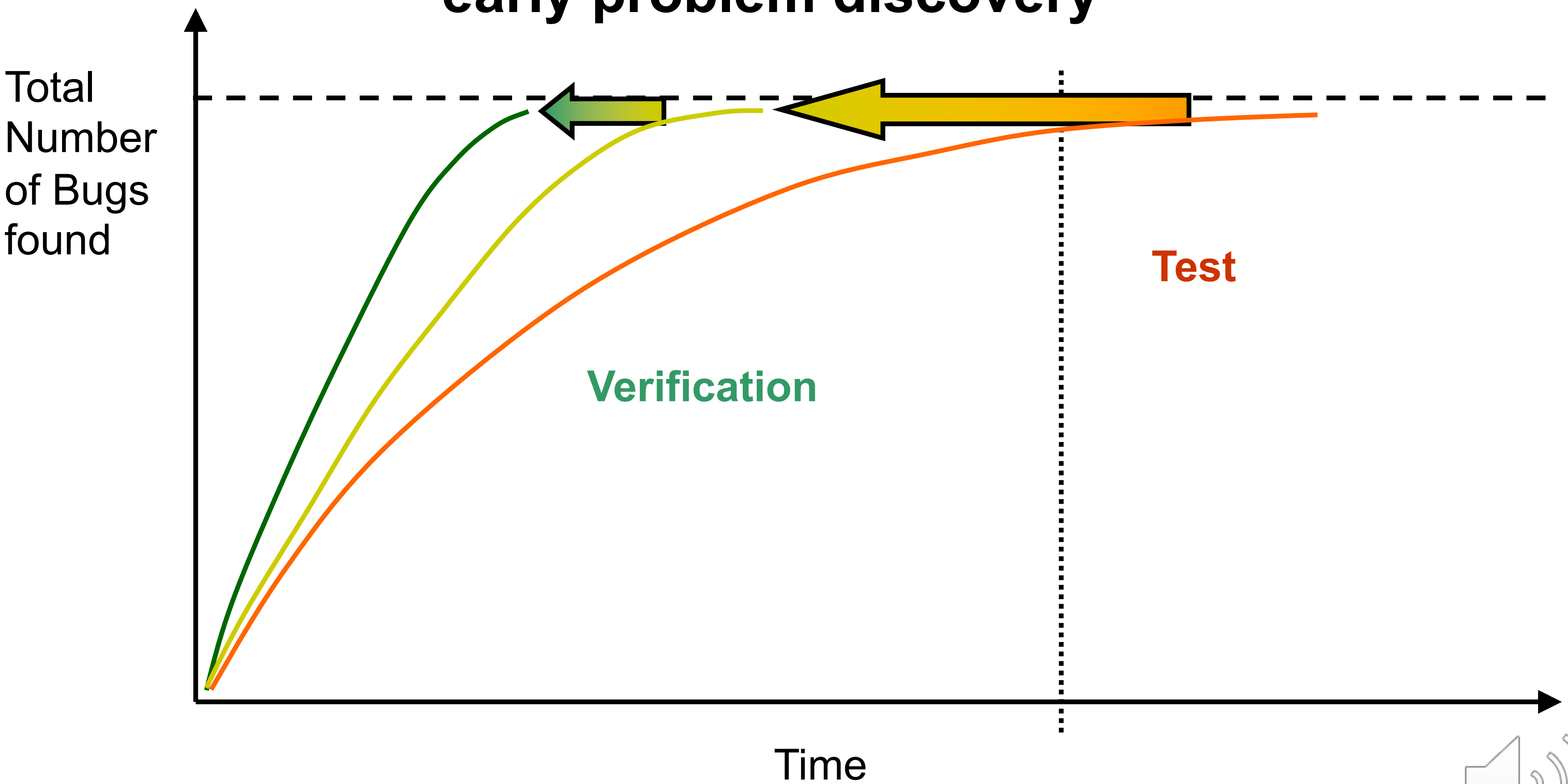
Need to minimise verification time e.g. by using:

- **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



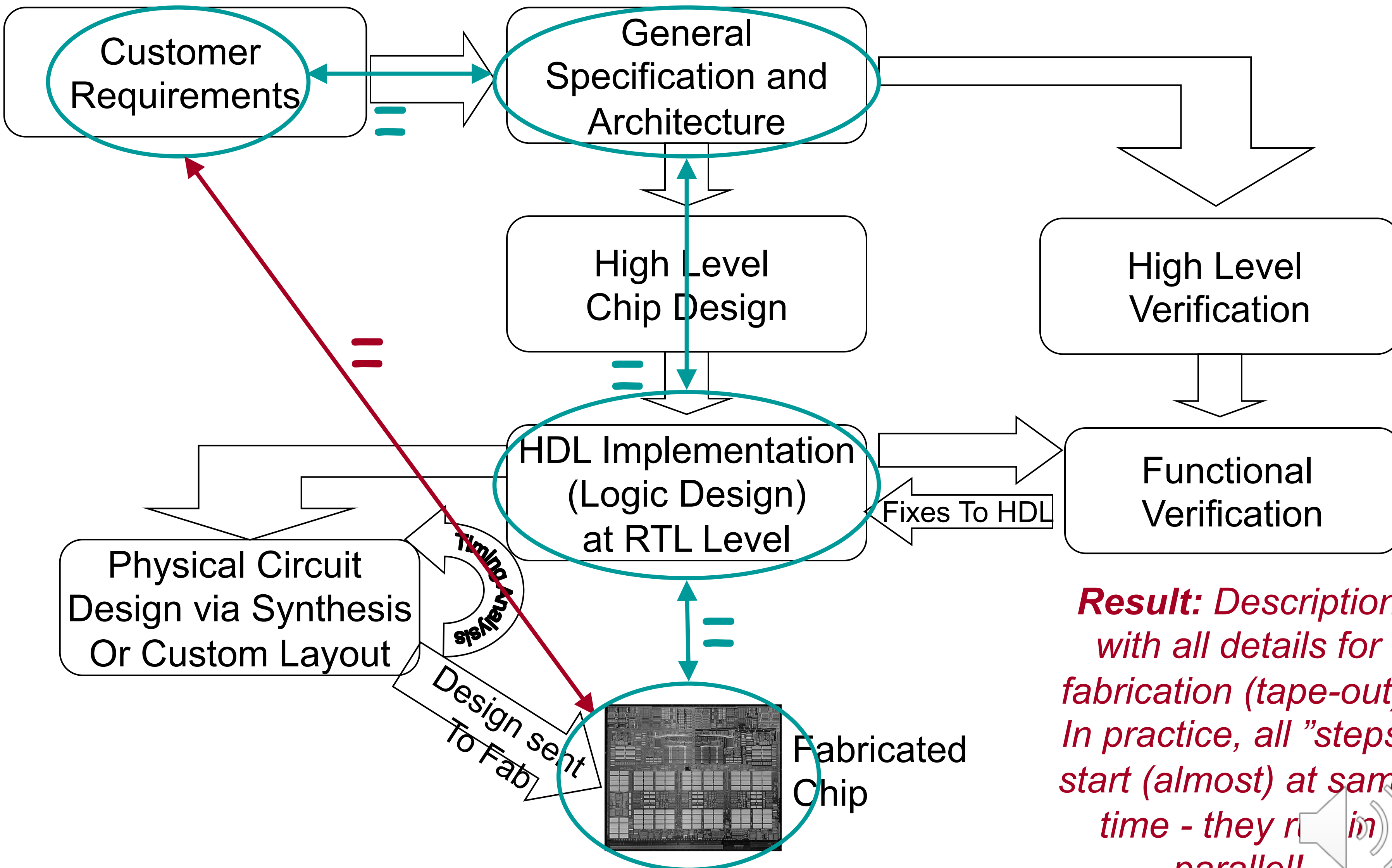
Summary so far …

- **What is Design Verification?**
  – Why do we care?
  – Verification vs validation
- **Bugs**
  – Sources of bugs
  – Cost of bugs
  – Importance of Design Verification
- **The chip design process**
  – Where does Verification “fit”?
  – Intel Fab Tour:
    - https://www.youtube.com/watch?v=2ehSCWoaOqQ
    - https://www.youtube.com/watch?v=JBYHwRXmEhY
    - https://www.youtube.com/watch?v=BtFdraQWVtM
- **Impact of increasing design complexity**
  – ITRS
  – Shrinking time to market windows
  – Need to increase productivity

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

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 r[...]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

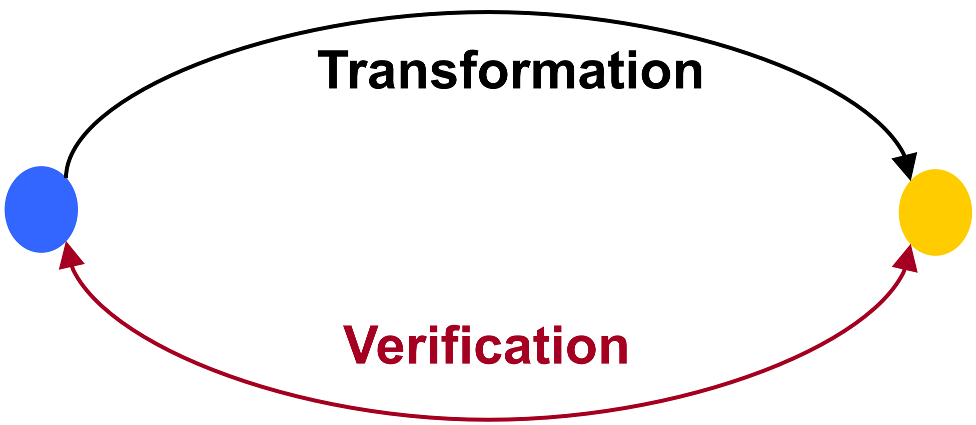
There is skill, science and methodology behind verification.

Reconvergence Models [Bergeron]

Conceptual representation of the verification process

- Most important question:

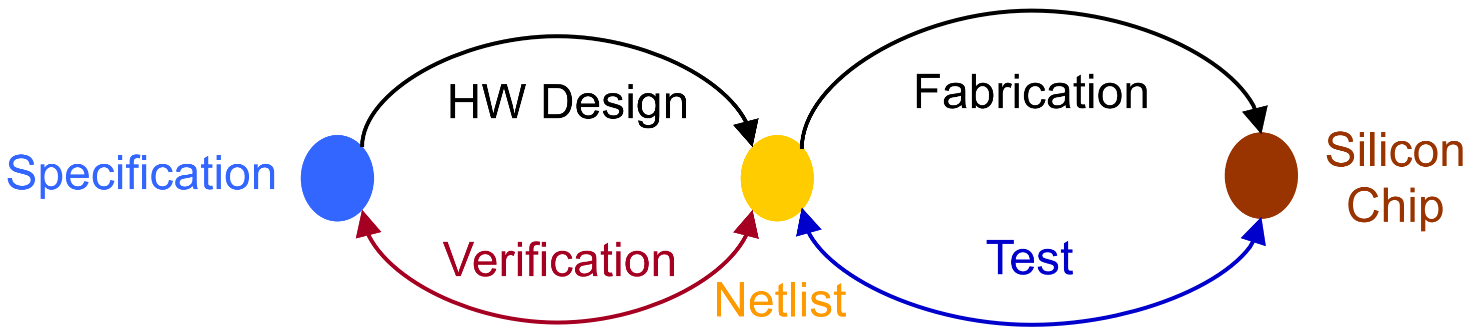
What are you verifying?



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

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!**

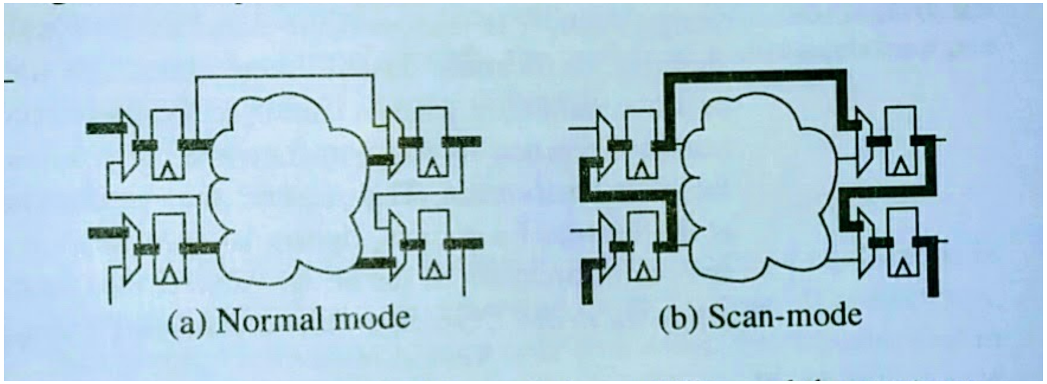


Design for Test

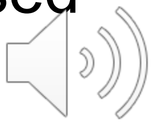
- One method employed during the test phase to facilitate testing is scanning
  - Link all internal registers together into a chain.
  - Chain accessible from chip pins.
  - Allows control/observation of internal state.
  - Impacts area of design, but keeps testing cost down.

Figure from the book "Writing Testbenches" by J. Bergeron

- This results in a **"Design for Test"** methodology



- Why not **"Design for Verification"?** [Hot topic of research!]
  - @ design time, consider: What is the design supposed to do? How will this be verified?

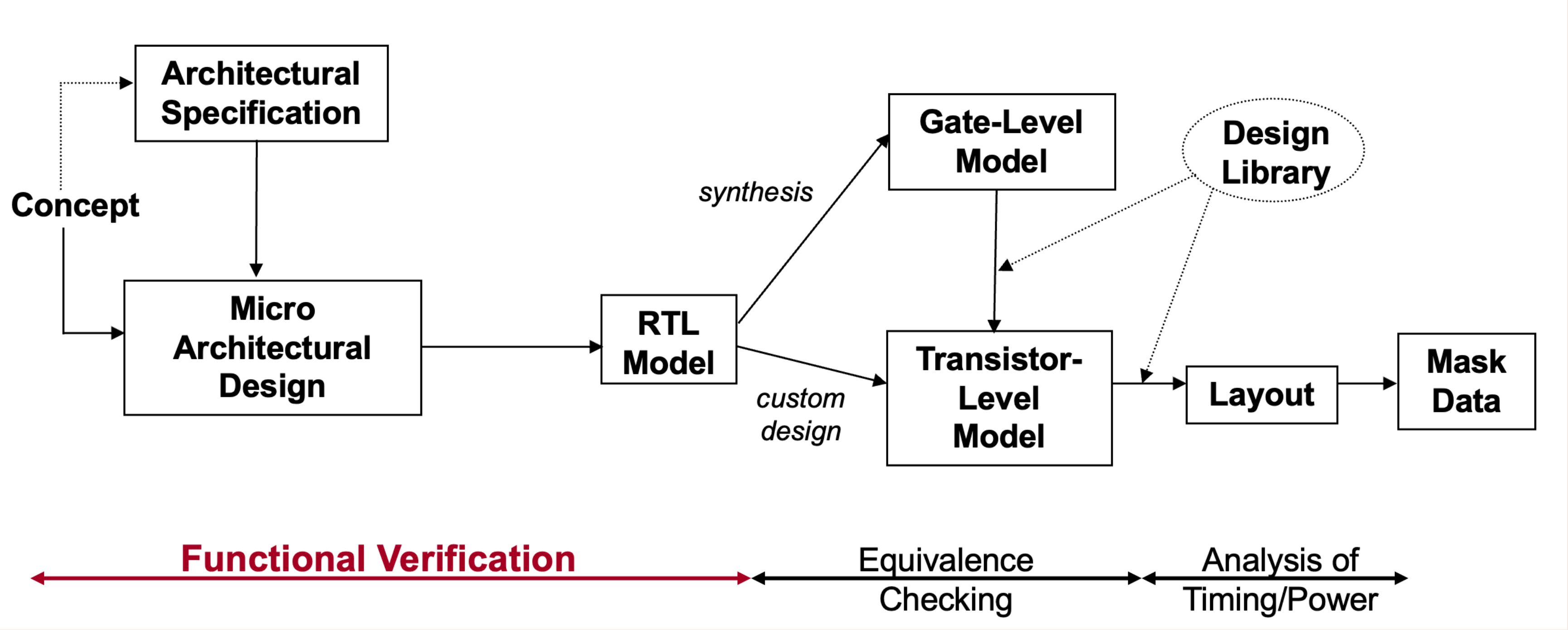


Formal: Equivalence Checking

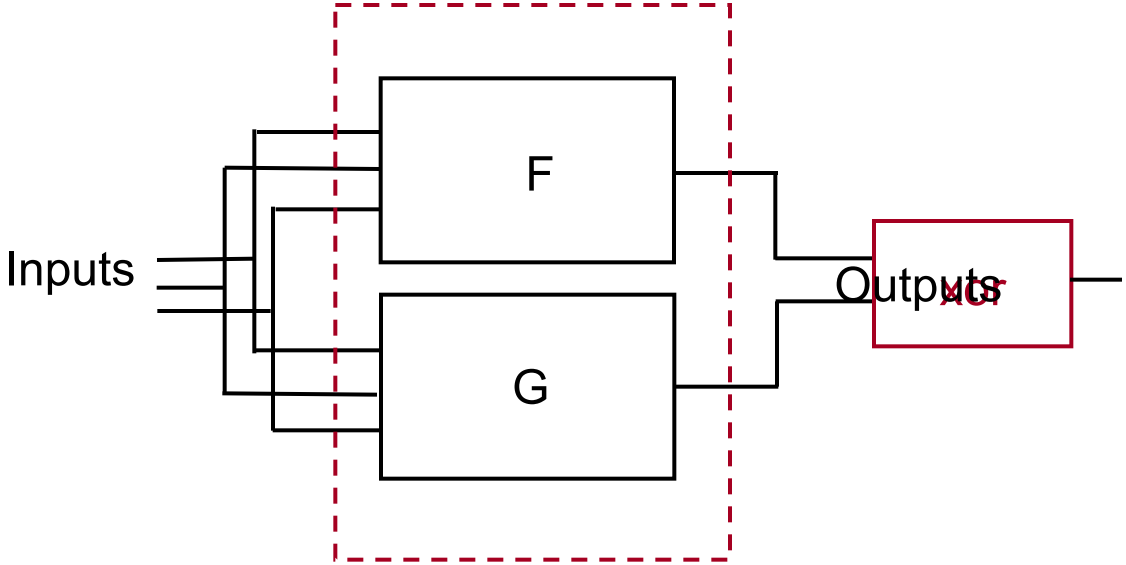
Compares two models to check for equivalence.

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

The IC Design Process



Equivalence Checking



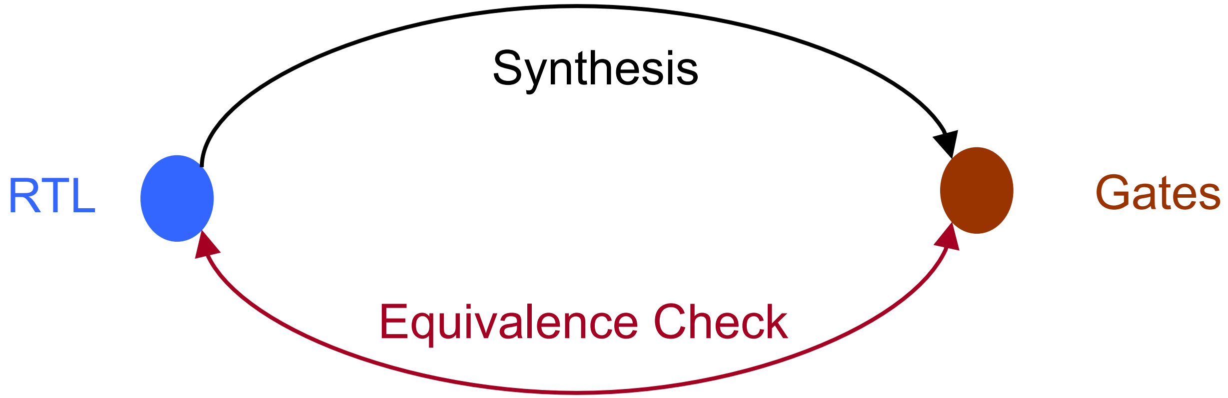
Conceptually, we are asking the question:

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

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!

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.

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		

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.

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!

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

Reconvergence models are the starting point for any verification activity!
When asked to verify something, first draw a reconvergence model and see whether you've got all you need to perform verification!

Next

- Recordings of lectures (about 2h - 3h per week)

  Week 1:

  ✓ Introduction to Design Verification

  ✓ Verification Hierarchy

  ✓ Driving & Checking

  – uobdv.github.io/Design-Verification/
  shows a **weekly schedule of topics** to watch BEFORE the next session, ideally

  – Recordings are available from Blackboard unit page

- Tasks for you this week:

  – **Attend the lab session** on Wednesday to set up access to the EDA tools

  – **Paper review "*The limits of correctness*"**

Paper review

Brian Cantwell Smith. 1985. The limits of correctness. SIGCAS Comput. Soc. 14,15, 1,2,3,4 (Jan 1 1985), 18–26. DOI: https://doi.org/10.1145/379486.379512

THE LIMITS OF CORRECTNESS†

Brian Cantwell Smith*

- *Identify the main lines of argument*
- *Why does the author question the notion of "correctness"?*
- *What are the two or three key take-away messages for you?*

Over the last ten years, the Defense Department has spent many millions of dollars on a new computer technology called "program verification" - a branch of computer science whose business, in its own terms, is to "prove programs correct". Pro-

And my answer, to give away the punch-line , is no. For fundamental reasons - reasons that anyone can understand - there are inherent limitations to what can be proven about computers and computer programs. Although program verification is an important new technology, useful, like so many

Opportunities

Tuesday 17th October 2022 I 12:00-13:00 BST

AI/ML in Verification

https://www.eventbrite.co.uk/e/dvclub-europe-aiml-in-verification-tickets-722622893527?aff=oddtdtcreator

eventbrite | Search events | Find Events | Create Events | Help Center

Just Added!

Tuesday, October 17

DVClub Europe: AI/ML in Verification

This DVClub consider how we can save time and effort whilst improving time-to-market through the application of AI/ML to verification

By **Tessolve Semiconductor Ltd**
395 followers
Follow

Remote Access 1
Free
Reserve a spot

Date and time

Tuesday, October 17 · 12 - 1pm BST

IC INTELLECTUAL CAPITAL RESOURCES

global technology recruitment partner since 1999

2021 end of year UK SALARY REVIEW

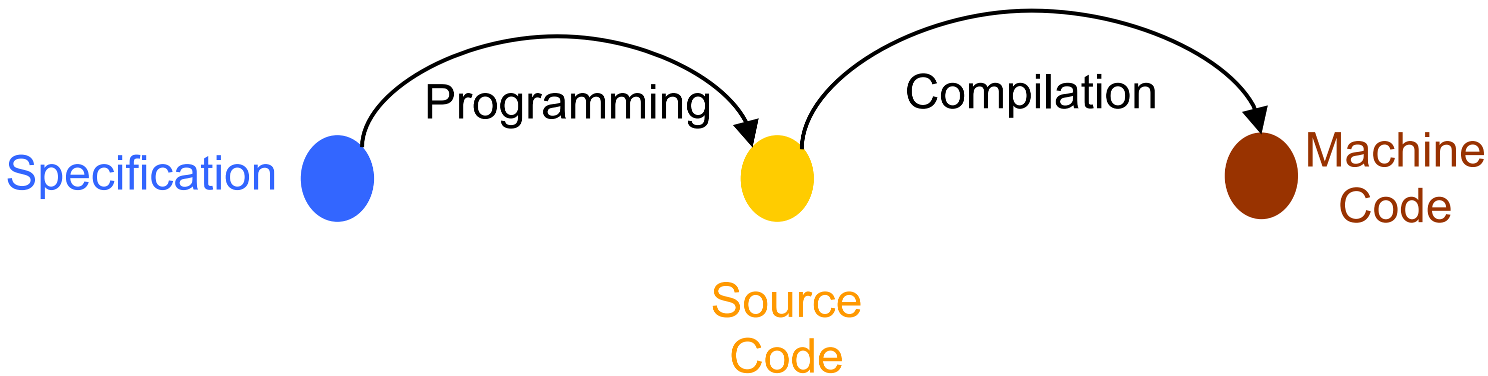
SOFTWARE | SEMICONDUCTOR | ELECTRONICS | SALES & MARKETING | SUPPLY CHAIN

W: ic-resources.com
T: +44 (0)118 988 1150
E: enquiries@ic-resources.com

Experience		Graduate	3 years	5 years	10 years	12+ years
Digtial IC Design	*Perm (p.a)*	£34,000	£42,000	£52,000	£65,000	£75,000+
	Cont (p.h)	-	*£42*	*£48*	*£50*	*£52+*
Digital IC Verification		£34,000	£42,000	£55,000	£65,000	£80,000+
		-	*£42*	*£48*	*£52*	*£55+*
Physical Design		£34,000	£42,000	£52,000	£65,000	£75,000+
		-	*£40*	*£46*	*£50*	*£52+*
FPGA Design		£31,000	£40,000	£47,000	£60,000	£70,000+
		-	*£40*	*£48*	*£50*	*£52+*
Analog/Mixed Signal IC Design		£34,000	£42,000	£52,000	£65,000	£75,000+
		-	*£42*	*£48*	*£52*	*£55+*
RF IC Design		£37,000	£45,000	£57,000	£70,000	£85,000+
		-	*£42*	*£48*	*£52*	*£55+*
Analog / RF Layout		£30,000	£38,000	£41,000	£52,000	£60,000+
		-	*£40*	*£45*	*£50*	*£50+*
IC Test		£32,000	£38,000	£40,000	£45,000	£60,000+
		-	*£45*	*£50*	*£55*	*£60+*
IC Process		£32,000	£38,000	£40,000	£45,000	£60,000
		-	-	-	-	-

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.

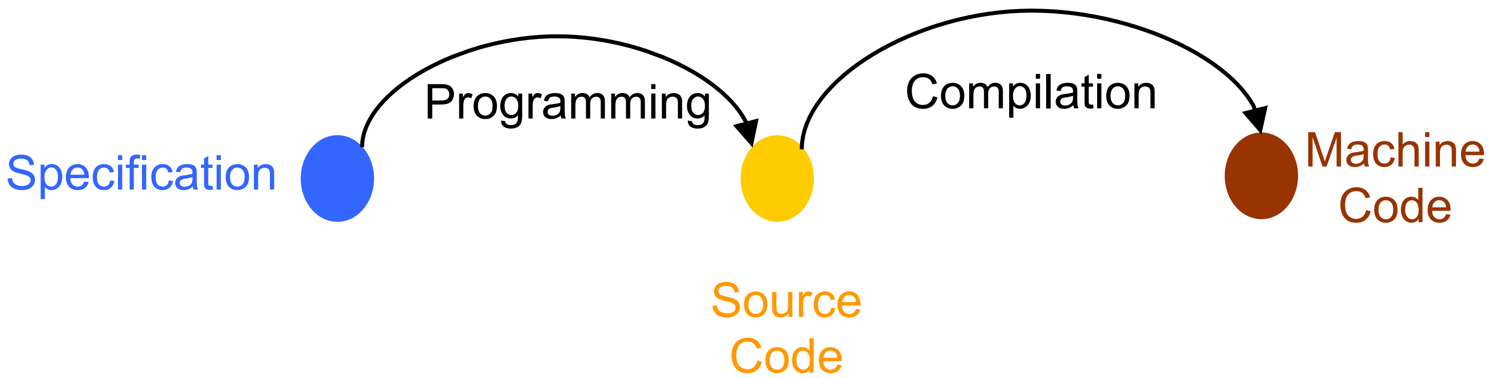


This slide is intentionally left blank for you to take some time to attempt the task on the reconvergence model on programming ☺

Please do not proceed until you've tried – you'll learn more if you try.

Reconvergence Models – another example

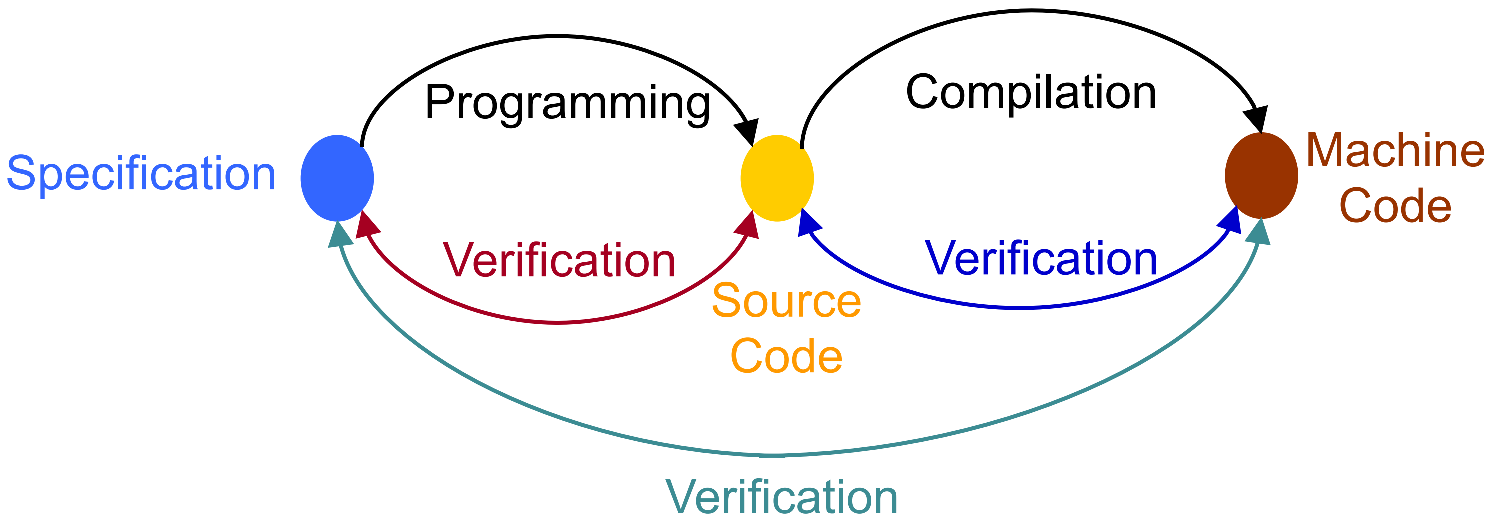
- In SW development, the transformative process from specification to source code is “programming”.
- The compiler then translates source code to machine code.



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

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.



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