On My Perfect Life: A Tenured Position While AI Does The Job

Rolf Drechsler

University of Bremen, Germany
DFKI Bremen, Germany
drechsler@uni-bremen.de

What to expect – what not

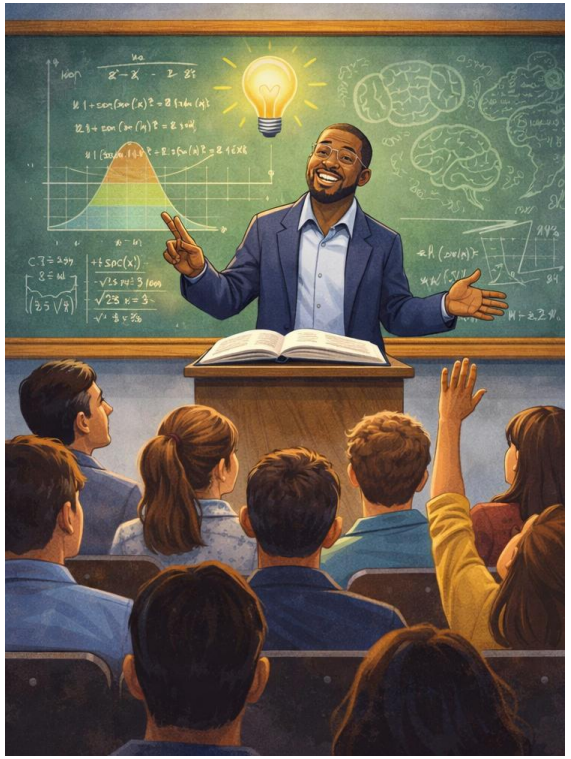
- Not my private life
  - "German perspective"
- Slides are not by GenAI (but many pictures)

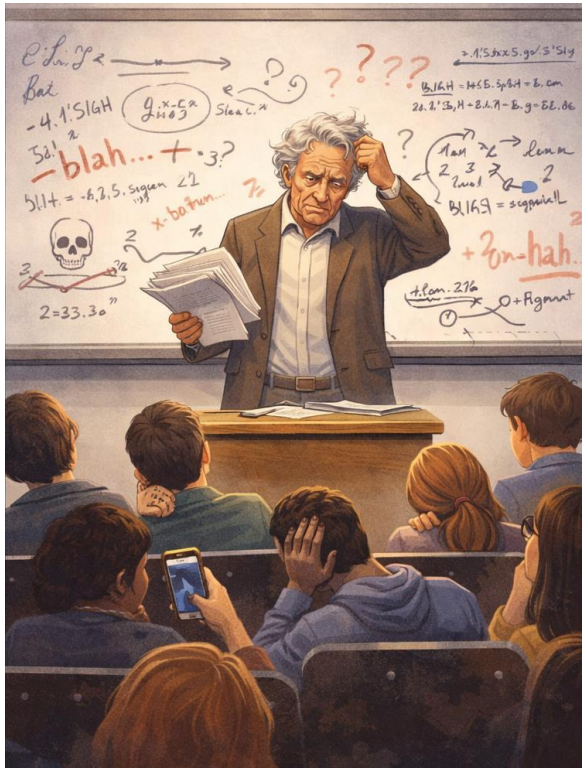
- Hot topic, with lots of changes
  - Not black and white, but lot of *gray*
  - Not "conflict-free"; not complete
- Not very technical
- **Inspiration/ideas/critical thinking**

Tenured position = no work (any more)

- Research and education

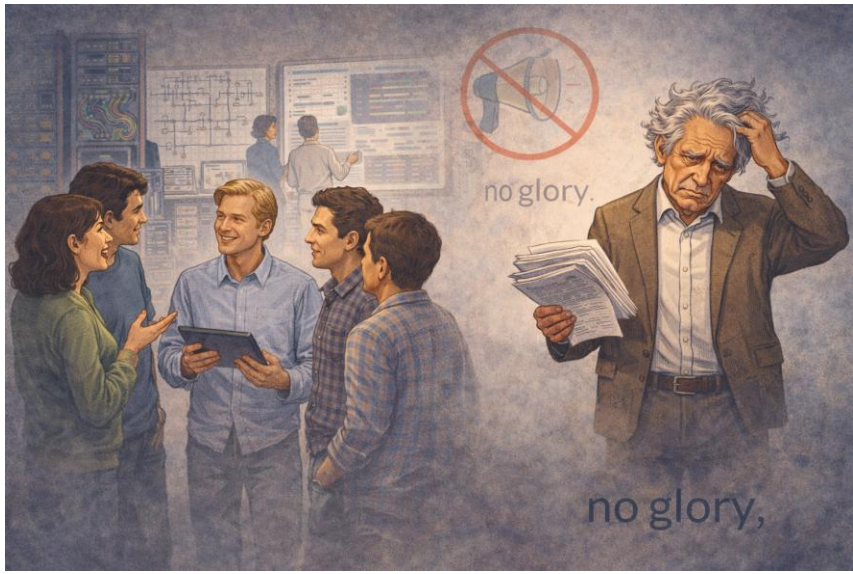
- Give lectures
  - **Good lectures**
    - lots of students
    - lots of exams
    - lots of work
  - Thus: **bad lectures**



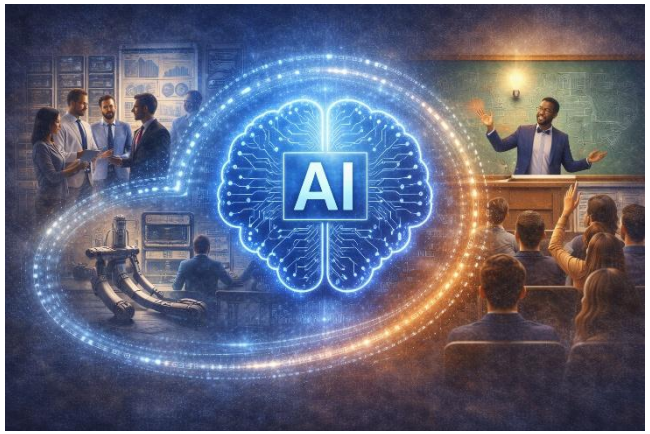


Tenured position = no work (any more)

- Do research
  - Publish papers: lots of work ☹
    - Choose hard problem
    - Blame colleagues
- **But:** *low reputation* – no glory, no fame
  - Possible solution: PhD students
  - Requires funding/write proposals: lots of work
  - Getting **best students**: good lectures



AI might be the solution



- **Research**
  - Give/discuss ideas, literature review
    - Implementation; carry out experiments; write the paper
  - Writing proposals (and reviews)
- **Education**
  - Provide lecture content
    - Answer students; review the submissions
  - Help the students
    - Teaching buddy; interactive learning; provide examples

Aspects

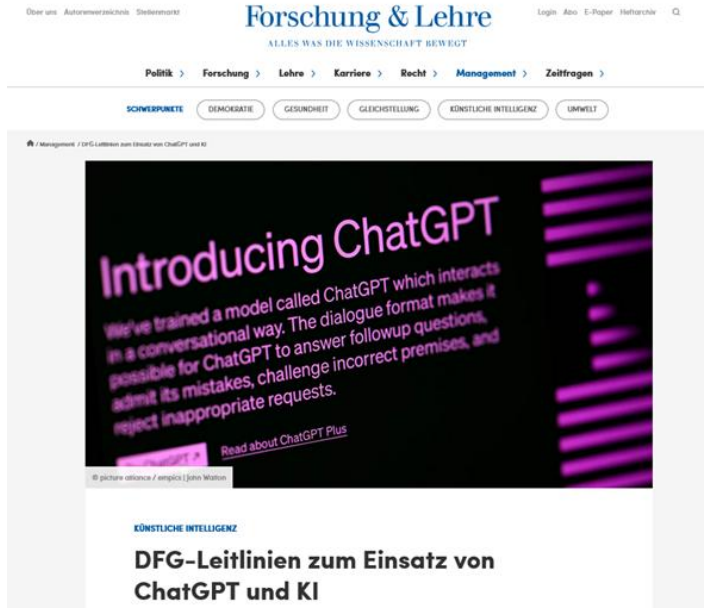
- Legal aspect
  - Are we allowed to use them?
  - What for?
- Where are they helpful?
  - What can they do?
- What is missing?
  - What they can’t do (yet)
  - Future trends

Legal Aspect


ARTIFICIAL INTELLIGENCE

'Positive review only': Researchers hide AI prompts in papers

Instructions in preprints from 14 universities highlight controversy on AI in peer review



- Literature review/writing
  - Papers/proposals: DFG/NSF in 2023
- Reviewing
  - DFG/NSF in 2023 **not allowed**
  - March 2026: "The guideline allows the use of AI systems in the review process exclusively in a **supporting role**, linking such use to four central principles: *confidentiality*, *transparency*, *quality assurance* and *responsibility*."

How it is used...

Todd Austin • 1.
S. Jack Hu Collegiate Professor of CSE at UofM, Computer Engineering Lab (C...
3 Tage • Bearbeitet •

I just submitted a research paper and added the AI-use disclaimer below.

I'm integrating more AI into my research workflow, and it's delivering increasing value. Here's how I'm using it:

- Code development assistance
- Hardware design assistance
- Test generation (especially negative tests)
- Microbenchmark generation
- Large-codebase navigation (e.g., "where in LLVM should this change go?")
- Cross-LLM fact-checking (draft with Model A, verify with Model B)
- Future-work ideation
- Related-work exploration
- Heavy reframing of paper sections
- Paper reading and summarization
- Final-pass grammar/style checks
- Consistency checks (terminology, figure numbers, citations)
- Prose generation from bullet notes (with pre-context of my voice)
- Early AI-generated reviews addressed before initial submission

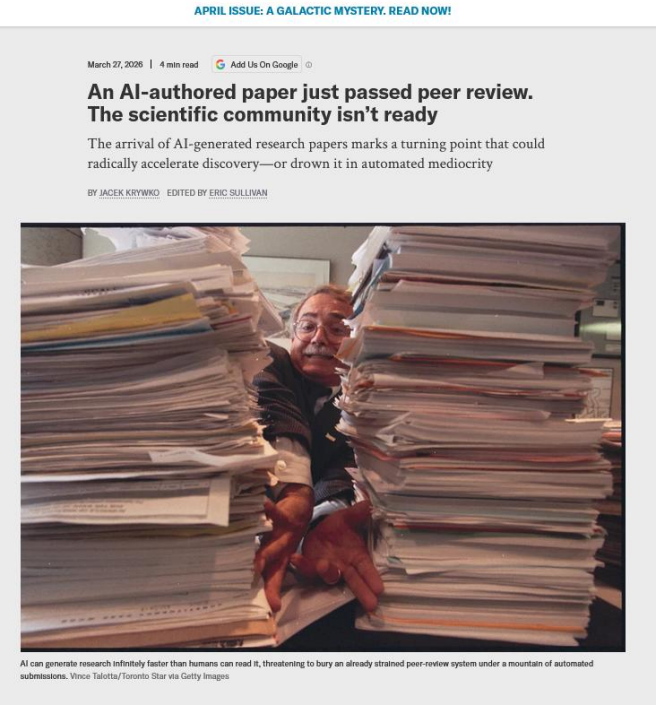
How are you using AI in your research?

VII. Content Generated by AI

Copilot, ChatGPT, Gemini, and QuillBot were used in the preparation of this manuscript for editing, grammar checking, and knowledge retrieval. No passages were copied without full author review, and all substantive ideas, analyses, and conclusions are the product and responsibility of the authors. Additionally, the listed AI tools were utilized for code development and early paper reviews.

Next: AI does it all?

- Can AI-written papers be accepted?
  - Passed peer review for workshop at 2025 International Conference on Learning Representations (ICLR)
    - Top-tier venue in the field of ML
- Given a *general topic prompt* by researchers
  - Surveys available literature
  - Generates hypotheses, evaluates and refines those ideas
  - Modules plan and execute experiments
  - …



A (very) short history of AI

1940 Alan Turing, Intelligent Machinery

1950 Foundation at Dartmouth College

https://upload.wikimedia.org/wikipedia/commons/b/bb/Dartmouth_College_%28145565527%29.jpeg

1960 Golden era of AI, symbolic reasoning rules

1970 First AI winter: reduced funding and interest in AI research

1980 Revival of AI: expert systems

```
 Welcome to
                EEEEEE  LL      IIII  ZZZZZZZ   AAAAA
                EE      LL       II      ZZ   AA   AA
                EEEEE   LL       II     ZZZ   AAAAAAA
                EE      LL       II    ZZ     AA   AA
                EEEEEE  LLLLLL  IIII  ZZZZZZZ  AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

1990 Second AI winter: reduced progress

2000 Quiet progress (Deep Blue vs. Kasparow (1997))

2010 Deep learning revolution

The world's most valuable resource is no longer oil, but data

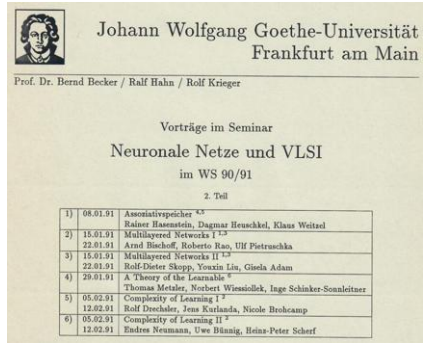
The data economy demands a new approach to antitrust rules

The Economist

https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data, 12.04.2026

2020 Generative AI

today



Johann Wolfgang Goethe-Universität
Frankfurt am Main

Prof. Dr. Bernd Becker / Ralf Hahn / Rolf Krieger

Vorträge im Seminar

Neuronale Netze und VLSI

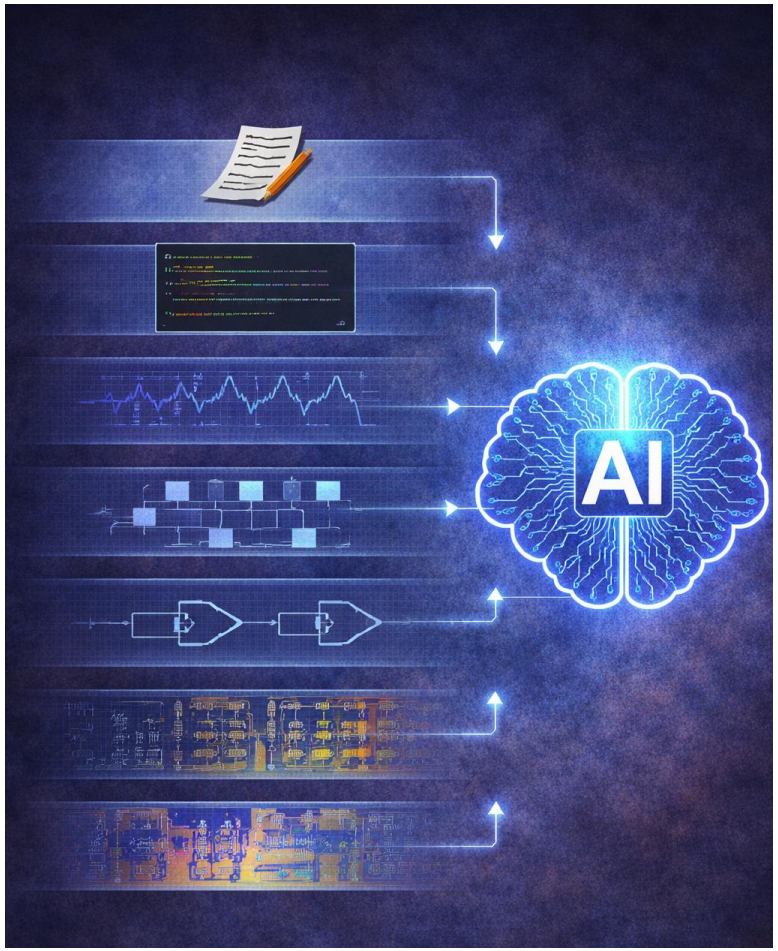
im WS 90/91

2. Teil

1)	08.01.91	Assoziativspeicher [4,5]	Rainer Hasenstein, Dagmar Heuschkel, Klaus Weitzel
2)	15.01.91 22.01.91	Multilayered Networks I [1,3] Arnd Bischoff, Roberto Rao, Ulf Pietruschka	
3)	15.01.91 22.01.91	Multilayered Networks II [1,3] Rolf-Dieter Skopp, Youxin Liu, Gisela Adam	
4)	29.01.91	A Theory of the Learnable [6] Thomas Metzler, Norbert Wiessollek, Inge Schinker-Sonnleitner	
5)	05.02.91	Complexity of Learning I [7]	
6)	05.02.91 12.02.91	Complexity of Learning II [7] Endres Neumann, Uwe Bünig, Heinz-Peter Scherf	

Description languages on various levels

- Abstraction levels in hardware design
  - Textual/functional specification
  - Algorithmic level
  - Behavioral level
  - Register-transfer level (RTL)
  - Logic/gate level
  - Circuit/transistor level
  - Layout/physical level

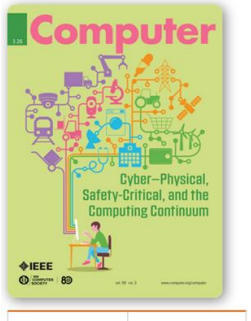


Good at text

- Specification
- Executable spec
  - C model
- All kind of programming languages
  - SystemC
  - Verilog/VHDL
- Verification
  - Writing properties
  - Testbenches

Writing code

- Hand-written – what we are used to
  - Time consuming
  - Error prone
  - Code generation
  - …
- Very quick with **GenAI**
  - Vibe coding



Home / Magazines / Computer / 2026.03

Computer

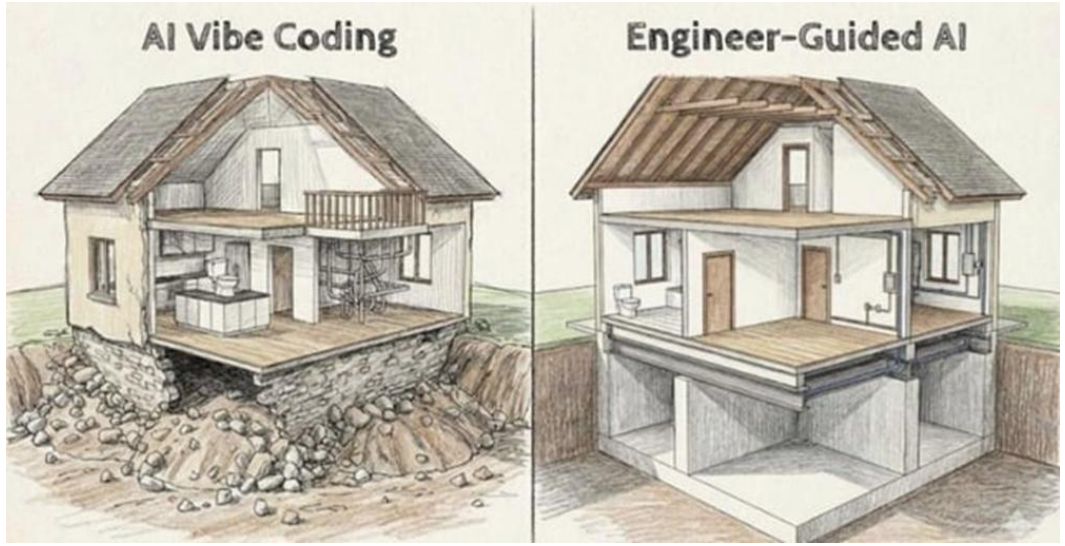
Is Vibe Coding the Future of Software?

Mar. 2026, pp. 100-104, vol. 59
DOI Bookmark: 10.1109/MC.2025.3634712

Authors

Nir Kshetri, University of North Carolina at Greensboro, Greensboro, NC, USA
Jeffrey Voas

DOWNLOAD PDF | SHARE ARTICLE | GENERATE CITATION



https://www.reddit.com/r/vibecoding/comments/1q30apl/vibe_coding_vs_vibe_engineering

Prompts

- Generate a RISC V core in VHDL
- Generate a VP for this core in C
- Generate properties in PSL

```
architecture rtl of riscv_core is
--------------------------------------------------------------------
RISC-V encoding constants (RV32I)
--------------------------------------------------------------------

constant OPCODE_LOAD    : std_logic_vector(6 downto 0) := "0000011";
constant OPCODE_STORE   : std_logic_vector(6 downto 0) := "0100011";
constant OPCODE_OP_IMM  : std_logic_vector(6 downto 0) := "0010011";
constant OPCODE_OP      : std_logic_vector(6 downto 0) := "0110011";
constant OPCODE_LUI     : std_logic_vector(6 downto 0) := "0110111";
constant OPCODE_AUIPC   : std_logic_vector(6 downto 0) := "0010111";
…
```

- **Correctness/trust**
  - Wrong interpretation
  - Hallucinations
  - …

10. PC progresses if no control transfer

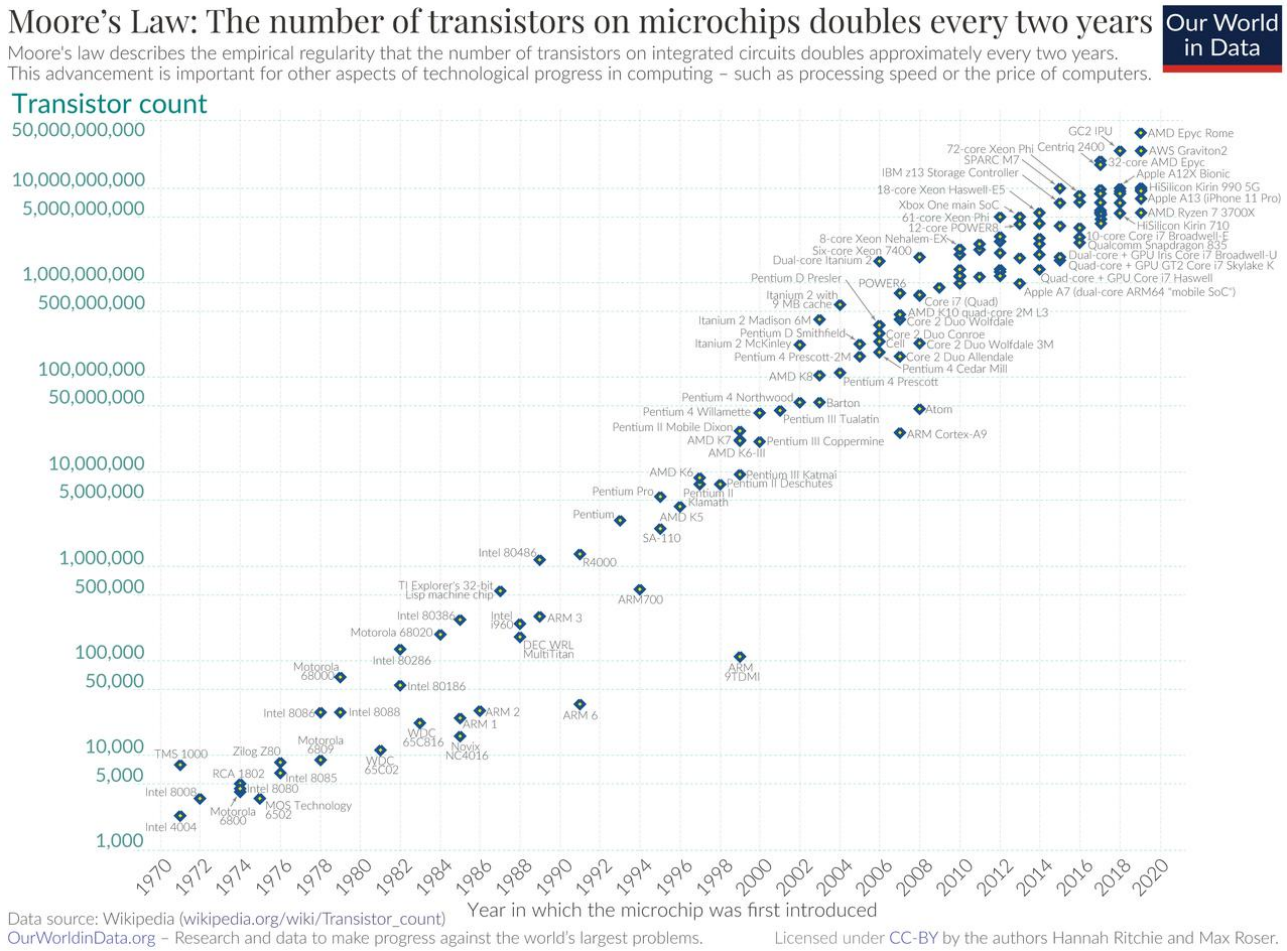
If we're not taking branch/jump, PC should just go +4.

Assume `branch`, `jal`, `jalr`, `take_branch`:

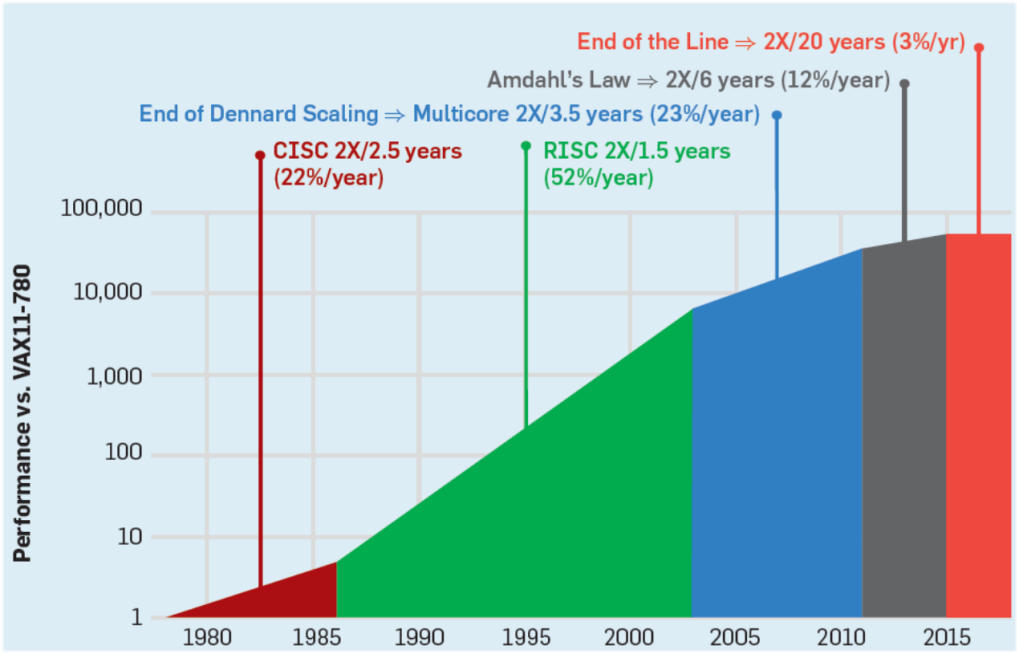
vhdl — Code kopieren

```
-- psl PC_Increments: assert always
--   (branch = '0' and jal = '0' and jalr = '0') ->
--     next (pc = std_logic_vector(unsigned(pc'prev) + 4));
```

Moore's law



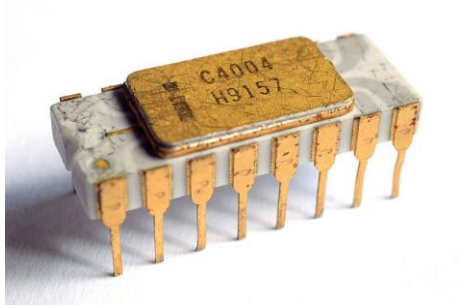
John Hennessy and David Patterson: Turing Lectures



Complexity of digital circuits

- Complexity of digital circuits is still increasing!

Intel 4004



2,250 transistors
Very low frequency

Intel Core i9-12900K

~10 bn transistors
Very high frequency

Source: Wikipedia

- Shift towards **verification**
  - See e.g.: Wilson Research Group & Mentor Graphics, Functional Verification Study

Ensuring correctness

- Simulation
  - Very **simple in nature**; efficient for **single** test vectors
  - Obtainable coverage is very low
- Emulation
  - Quality can be improved by **hardware acceleration**
  - Coverage is still very low for complex designs
- Formal verification
  - Powerful technique that **allows complete verification**
  - Very expensive in terms of resources

Ensuring correctness using behavior

- **Simulation/Emulation**
- **Formal verification**

- Describe the **behavior** of the circuit
  - Common in model checking/property checking

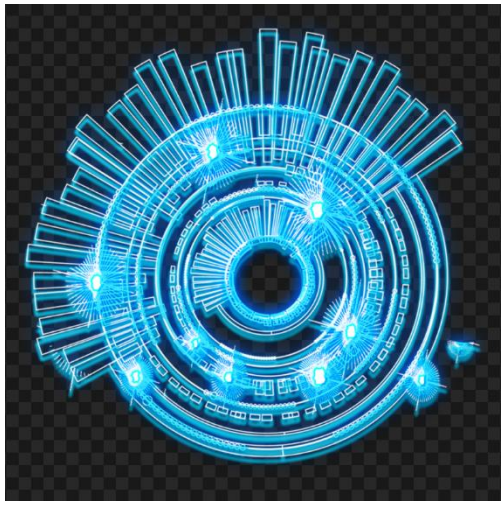
Complexity increases

- Use of **hierarchy**
- Similarities to **software development process**
- Techniques from **software** testing
  - Mutation testing
  - Fuzzing
  - …

Test driven development

- Describe tests first
- Code development comes **next**
- Tests *directly applied* to newly generated code

Behavior driven development



- Specify **behavior**

- Feature: User Login

  Scenario: Successful login with valid credentials

  Given the user is on the login page

  When the user enters valid credentials

  Then the user should see the dashboard

- Form: **Given-When-Then**

BDD for hardware

- Proposed more than 10 years ago
- Scenarios
- **But:**
  - Based on classical AI tools
  - Stanford parser
  - Princeton WordNet

Behavior Driven Development for Circuit Design and Verification

Melanie Diepenbeck[1] Mathias Soeken[1,2] Daniel Große[1] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{diepenbeck, msoeken, grosse, drechsle}@informatik.uni-bremen.de

***Abstract*—The design of hardware systems is a challenging and erroneous task where about 70% of the effort in designing these systems is spent on verification. In general, testing and verification are usually tasks that are being applied as a post-process to the implementation.**

In this paper, we propose a new design flow based on *Behavior Driven Development* (BDD), an agile technique for the development of software in which acceptance tests written in natural language play a central role and are the starting point in the design flow. We advance the flow such that the specifics that arise when modeling hardware are taken into account. Furthermore, we present a technique that allows for the automatic generalization of test cases to properties that are suitable for formal verification. This allows the designer to apply formal verification techniques based on test cases without specifying properties.

We implemented our approach and evaluated the flow for an illustrative example that successfully demonstrates the advantages of the proposed flow.

First, we customize BDD for circuits modeled in a *Hardware Description Language* (HDL). Therefore, we have to take the specifics of circuits, in particular timing and testbenches, into account. As a result, circuit components are iteratively created and, based on the underlying BDD methodology, the implemented functionality is tested in a structured way while the tests are connected to natural language specifications. Second, given the specific structure of the acceptance tests we present a technique to automatically generalize these tests. That is, we are able to generate properties from the written test code. These properties are formally verified on the iteratively developed implementation and hence enable to fully exhaust the potential of the test cases. Both contributions are integrated into a new flow enabling the following advantages:

- Circuit design starting from a natural language based specification down to an HDL implementation

Scenario Outline: *Adding two numbers*
Given a calculator
When I add the numbers <a> and <b>
Then I see the result <c>

Examples:

a	b	c
2	8	10
3	9	12
100	20	120

BDD for hardware

Given the digit <a> is between 0 and 9
When the circuit is reset
And I wait 1 cycle
When I type the first digit <a>
And I wait 1 cycle
Then the resulting number is <a>

(a) Feature file

```
Given /^the digit (\d+) is between 0 and 9$/ do |arg1|
  $assert( arg1 < 10 );
end
When /^the circuit is reset$/ do
  reset = 1;
end
When /^I type the first digit (\d+)$/ do |digit|
  reset = 0;
  in_digit = digit;
end
Then /^the resulting number is (\d+)$/ do |num|
  $assert( number === num );
end
```

(b) Step definitions

```
vunit typed(digit_reader) {
  property type_first_digit = always (
    reset == 1
    && next(reset == 0)
    && next[2](reset == prev(reset))
    && next[2](in_digit == prev(in_digit))
  ) -> (
    next[2](number == prev(in_digit))
  );
  property env_assume_0 = always (
    next_a[0..2](in_digit < 10));

  assume env_assume_0;
  assert type_first_digit;
}
```

(c) Resulting property

Use LLMs in hardware design

- LLMs very powerful for text generation
- Various applications in EDA
- Generate scenarios using LLMs
- E.g.
  - Prompt: *Create ADD scenario with A = B, 3 examples.*

Case study: 16-bit ALU

- Verilog
- Operations, like
  - ADD, SUB, MULT,…
- Flags, like
  - Carry, overflow,…

```
// ALU operation logic
always @(*) begin
    // Default values
    result = 16'b0;
    carry = 1'b0;
    overflow = 1'b0;
    temp_result = 17'b0;

    case (opcode)
        4'b0000: begin // ADD
            temp_result = a + b;
            result = temp_result[15:0];
            carry = temp_result[16];
            overflow = (a[15] == b[15]) && (result[15] != a[15]);
        end

        default: begin
            result = 16'b0;
        end
    endcase
end

// Flag generation
always @(*) begin
    zero = (result == 16'b0);
    negative = result[15];
end
```

Scenario

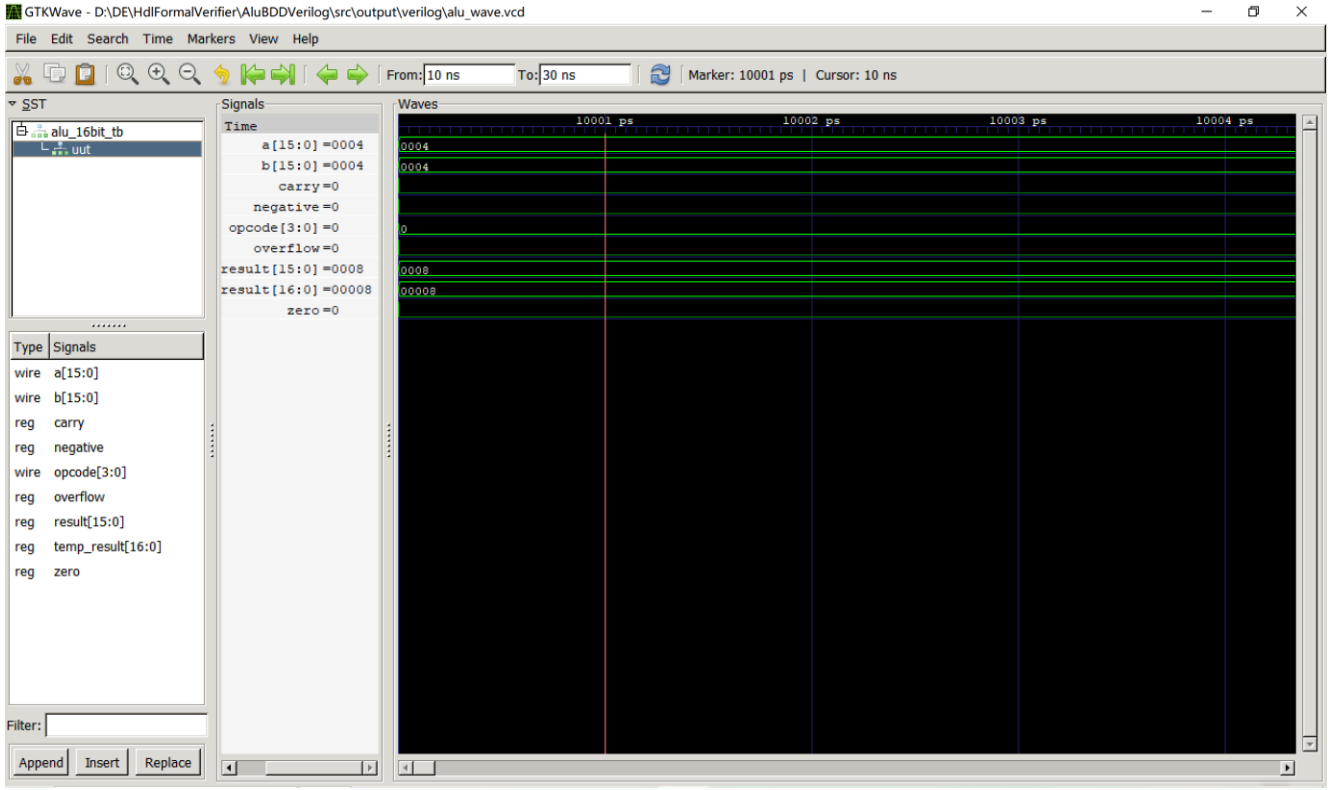
- ADD
  - A greater B
- Flags
  - Zero
  - Overflow
  - Negation

```
@arithmetic @addition @A>B @comparison
Scenario: Addition when A is greater than B
  Given operand A is greater than operand B
  When I set opcode to "0000" for ADD operation
  Then the result should be A + B

  Examples:
  |   A   |   B   | result | zero_flag | overflow | negative_flag | opcode |
  | 31503 | 29321 | 60824  | false     | false    | true          | 0000   |
  | 27364 | 4635  | 31999  | false     | false    | false         | 0000   |
  | 59362 | 56688 | 50514  | false     | true     | true          | 0000   |
  | 54876 | 39900 | 29240  | false     | true     | false         | 0000   |
  | 3020  | 572   | 3592   | false     | false    | false         | 0000   |
  | 50620 | 49223 | 34307  | false     | true     | true          | 0000   |
  | 46875 | 19364 | 703    | false     | true     | false         | 0000   |
  | 49916 | 28936 | 13316  | false     | true     | false         | 0000   |
  | 10919 | 5099  | 16018  | false     | false    | false         | 0000   |
  | 45900 | 10728 | 56628  | false     | false    | true          | 0000   |
```

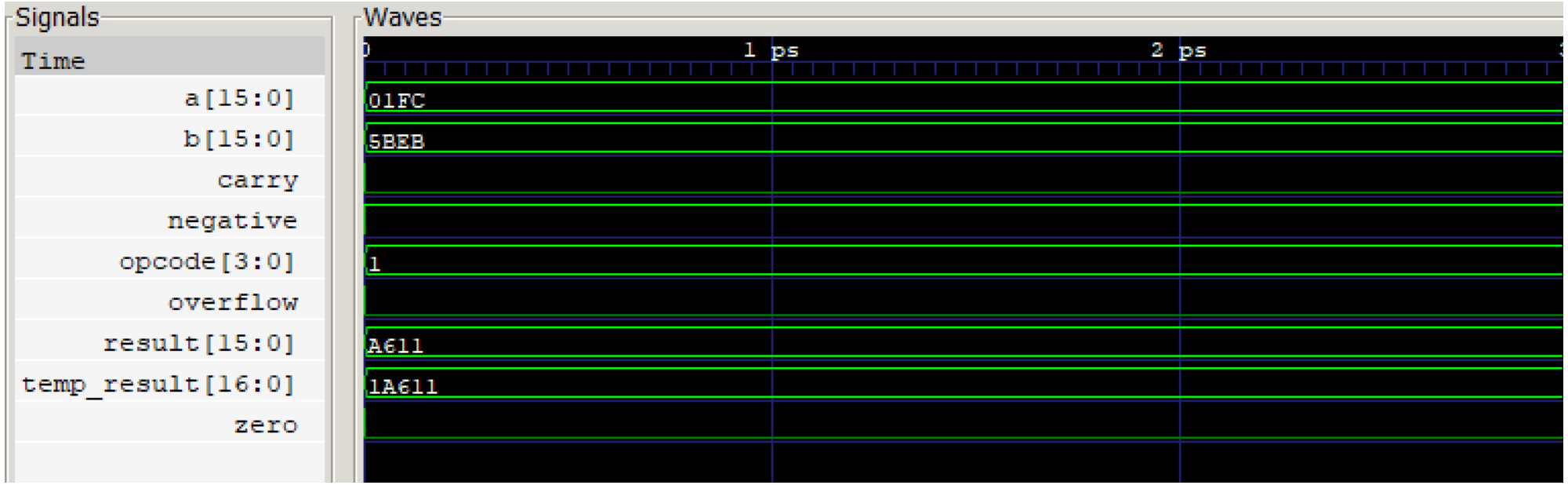
Simulation of examples

- GTKWave simulator



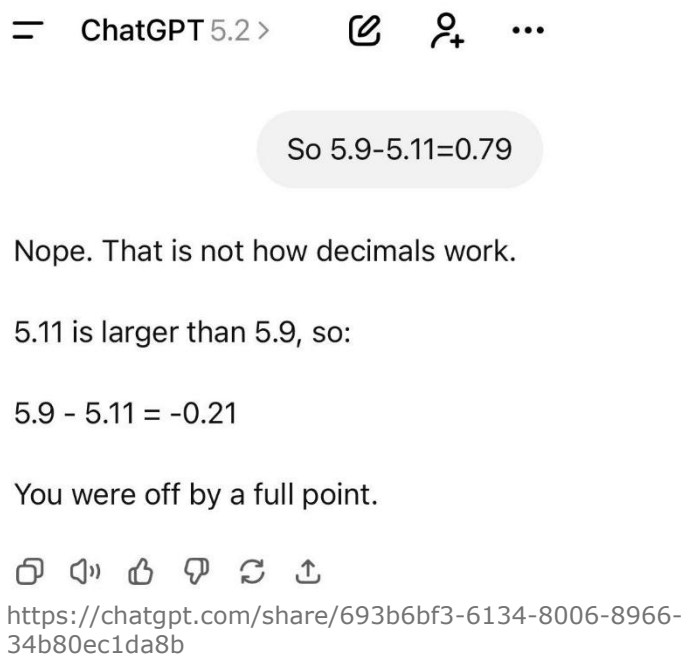
Simulation of examples

- GTKWave simulator



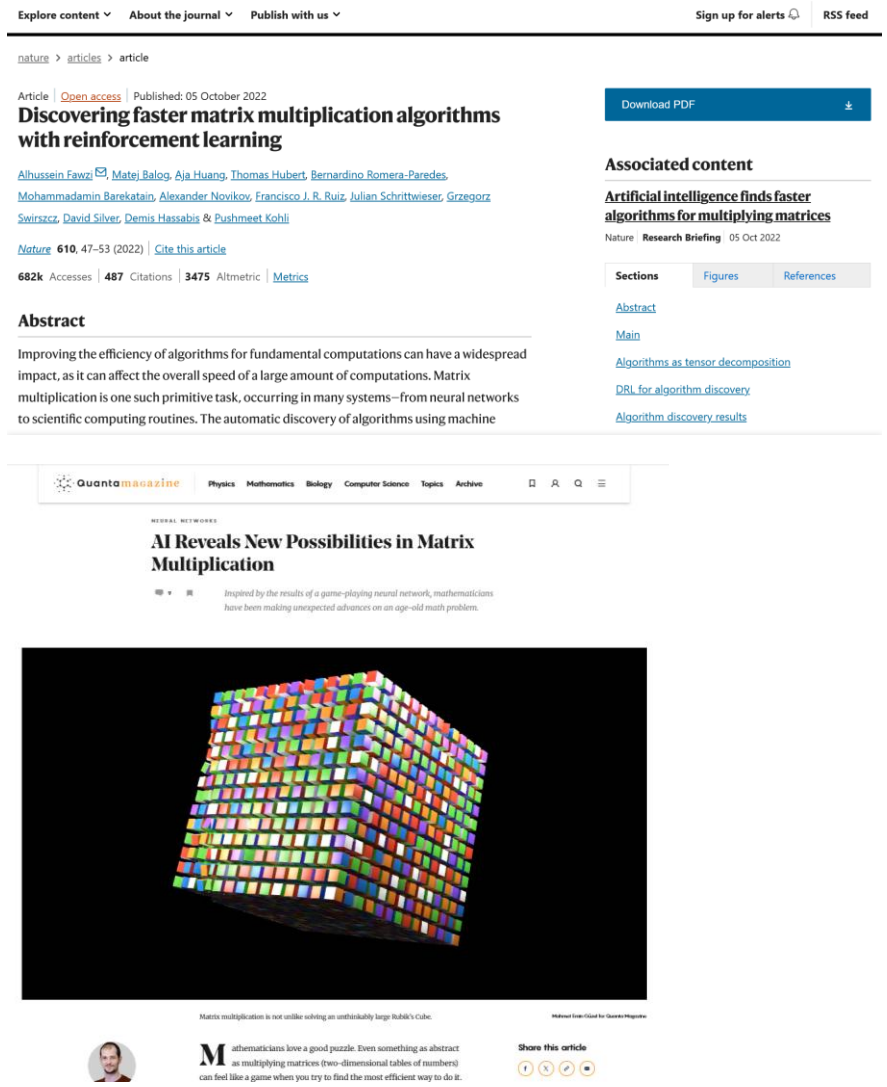
Application

- LLM for text
  - Poor results on graphs/netlists
- LCM for circuits
  - *Fewer* test cases
- Originally not meant for
  - **Optimization**: combination with (classical) heuristics
  - **Correctness**: need verification
    - In contrast: hallucinations due to randomness/probabilities



Optimization: results

- Many papers report improvements
  - Real breakthroughs?
- Proofs of (long) open problems?
  - Discovering faster matrix multiplication algorithms with reinforcement learning



Explore content · About the journal · Publish with us · Sign up for alerts · RSS feed

nature › articles › article

Article | Open access | Published: 05 October 2022

Discovering faster matrix multiplication algorithms with reinforcement learning

Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis & Pushmeet Kohli

Nature **610**, 47–53 (2022) | Cite this article

682k Accesses | **487** Citations | **3475** Altmetric | Metrics

Download PDF

Associated content

Artificial intelligence finds faster algorithms for multiplying matrices

Nature | Research Briefing | 05 Oct 2022

Sections · Figures · References

Abstract

Main

Algorithms as tensor decomposition

DRL for algorithm discovery

Algorithm discovery results

Abstract

Improving the efficiency of algorithms for fundamental computations can have a widespread impact, as it can affect the overall speed of a large amount of computations. Matrix multiplication is one such primitive task, occurring in many systems—from neural networks to scientific computing routines. The automatic discovery of algorithms using machine

Quanta magazine · Physics · Mathematics · Biology · Computer Science · Topics · Archive

NEURAL NETWORKS

AI Reveals New Possibilities in Matrix Multiplication

Inspired by the results of a game-playing neural network, mathematicians have been making unexpected advances on an age-old math problem.

Matrix multiplication is not unlike solving an unthinkably large Rubik's Cube.

Mathematicians love a good puzzle. Even something as abstract as multiplying matrices (two-dimensional tables of numbers) can feel like a game when you try to find the most efficient way to do it.

Share this article

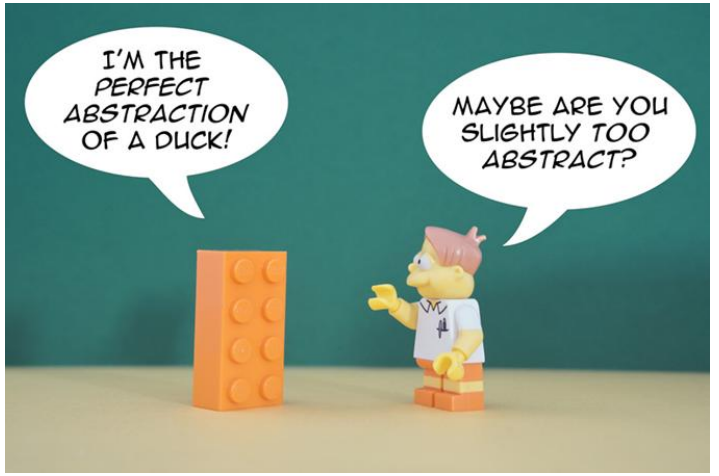
Correctness: automation of verification

- So far, still validation “by hand”
  - Classical methods for checking completeness
  - Simulation, formal properties,…

- Can we get automated proofs?
  - Reasoning (cf. early AI)
    - Now with a precise goal
  - Explanation

Are LLMs the next level of abstraction?

- Introduce *abstraction* to simplify well-known techniques
  - Hide boring details to increase efficiency
- Remember when:
  - Assembly was replaced by “automatic programming”, aka **compilers**
  - Hand-traversed data structures were replaced by **libraries**



https://medium.com/@abnoanmuniz/understanding-abstraction-with-c-ba79a0622448

Are LLMs the next level of abstraction?

Higher programming languages

Compiler code is slow and bloated

Loss-of-control vs. hand-tuned assembly/RTL

Will de-skill or replace programmers

Benchmarks, standards, debuggers, hybrid workflows (drop to low level when needed)

Vibe Coding

LLM code is inefficient, over-engineered or naive

Loss-of-control vs. hand-written, deeply understood code

Will de-skill or replace real programmers

Test suites, code review, sandboxing, and hybrid workflows

Mathematical proof of correctness

- Proving its own correctness
  - Gödel's Second Incompleteness Theorem (1931)
    - No sufficiently powerful and consistent formal system can prove its own consistency

Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I[1].

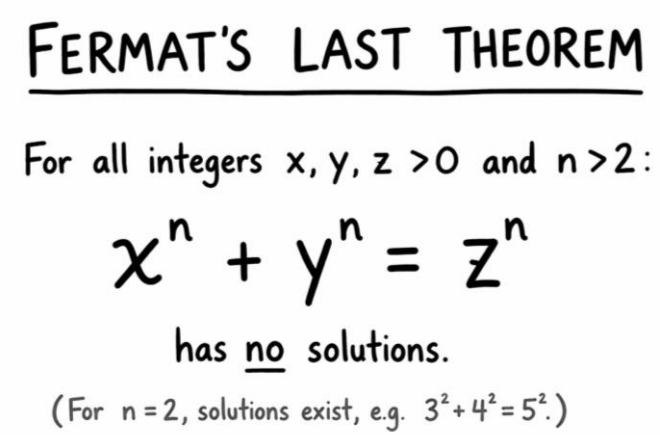
Von **Kurt Gödel** in Wien.

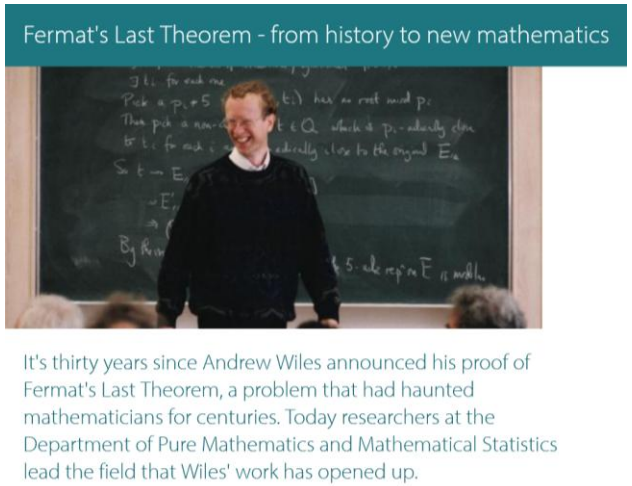
1.

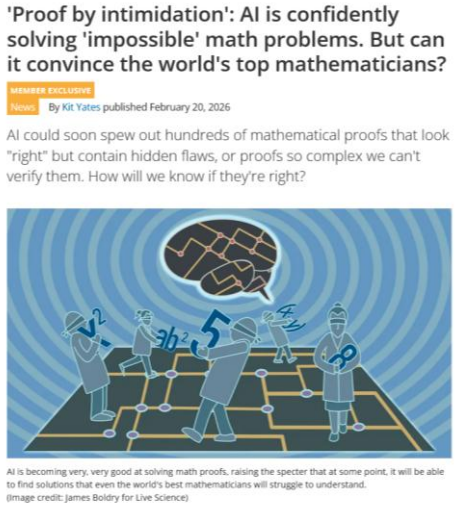
Die Entwicklung der Mathematik in der Richtung zu größerer Exaktheit hat bekanntlich dazu geführt, daß weite Gebiete von ihr formalisiert wurden, in der Art, daß das Beweisen nach einigen wenigen mechanischen Regeln vollzogen werden kann. Die umfassendsten derzeit aufgestellten formalen Systeme sind das System der Principia Mathematica (PM)[2]) einerseits, das Zermelo-Fraenkelsche (von J. v. Neumann weiter ausgebildete) Axiomensystem der Mengenlehre[3]) andererseits. Diese beiden Systeme sind so weit, daß alle heute in der Mathematik angewendeten Beweismethoden in ihnen formalisiert, d. h. auf einige wenige Axiome und Schlußregeln zurückgeführt sind. Es liegt daher die Vermutung nahe, daß diese Axiome und Schlußregeln dazu ausreichen, alle mathematischen Fragen, die sich in den betreffenden Systemen überhaupt formal ausdrücken lassen, auch zu entscheiden. Im folgenden wird gezeigt, daß dies nicht der Fall ist, sondern daß es in den beiden angeführten Systemen sogar relativ einfache Probleme aus der Theorie der gewöhnlichen ganzen Zahlen gibt[4]), die sich aus den Axiomen nicht

Solving hard problems

- Fermat's last theorem
  - Proof by Andrew Wiles
  - More than 100 pages
- Similar to SAT solvers
  - Satisfying assignment: trivial
  - Unsat: how to verify?
- Can we always understand the reasoning of AI?
  - "Proof by intimidation"
  - Reading fatigue







Future (and past)

- *Model raising*: reform AI model-training paradigms from post hoc alignment to intrinsic, identity-based development
  - Not fixing inherently unsafe AI with external safety measure applied as post-processing
  - "Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?"
- World model: learn from reality


opinion

DOI:10.1145/3748645 Roland Aydin, Christian Cyron, Steve Bachelor, Ashton Anderson, and Robert West

Opinion

From Model Training to Model Raising

A call to reform AI model-training paradigms from post hoc alignment to intrinsic, identity-based development.

TODAY'S DOMINANT PARADIGM for creating state-of-the-art user-facing artificial intelligence (AI) models, such as ChatGPT, Gemini, or Claude, is to first *pre-train* a raw model that is not explicitly aligned with human values and then align it in a post-processing step, using techniques such as handcrafted system prompts or reinforcement learning from human feedback (RLHF). Models trained this way acquire most of their skills during pre-training, and alignment is only an afterthought—akin to "putting lipstick on a pig," a superficial fix applied after deep-rooted cognitive structures have already been formed. The upshot is that misaligning, or jailbreaking, even the most celebrated frontier models is rather easy,[1,2,3] leading to a cat-and-mouse game between alignment efforts and attempts to undermine them. Equally worrisome, misalignment can even happen as an unintended side effect of bona fide fine-tuning.[4]

It is important to understand that today's common alignment practice—developing capabilities first and aligning values second—is not the result of a long-term research program but simply reflects the historical course of development. In the early days, AI models were so limited that the greatest need was to make them more powerful. Only once AI models had become highly performant was it realized by a broader community that aligning AI with human values is not just an academic problem but one of pressing societal importance. The pragmatic fix was to take the existing AI models—including their already established training procedures—for granted and to develop additional procedures to limit their harm via post-processing. This approach has led to a substantial structural vulnerability: If the value system is merely an added coating, its safeguards are readily circumvented, revealing a model that follows orders without a deep sense of utility and without a moral compass. This realization has led RLHF co-inventor Paul Christiano to call RLHF "obviously inadequate for aligning really powerful models"[a] and ACM A.M. Turing Award recipient and Nobel Prize recipient Geoffrey Hinton to call it "a pile of crap."[b]

A Nurturing Alternative: Early, Intrinsic Alignment

Instead of slapping on a corrective value system after the fact, we argue for moving alignment further upstream in the training process. Initial attempts in this direction have been encouraging (e.g., "pre-training with human feedback"[c] or safety pretraining[d]), and we call for ingraining human values into AI models even more deeply and striving

a https://tinyurl.com/28fep9hb

b https://tinyurl.com/24m6bgvg

24 COMMUNICATIONS OF THE ACM | FEBRUARY 2026 | VOL. 69 | NO. 2

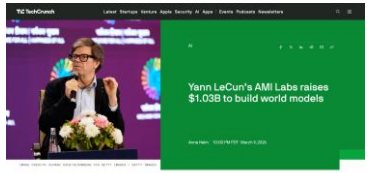
A. M. Turing (1950) Computing Machinery and Intelligence. *Mind 49:* 433-460.

COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

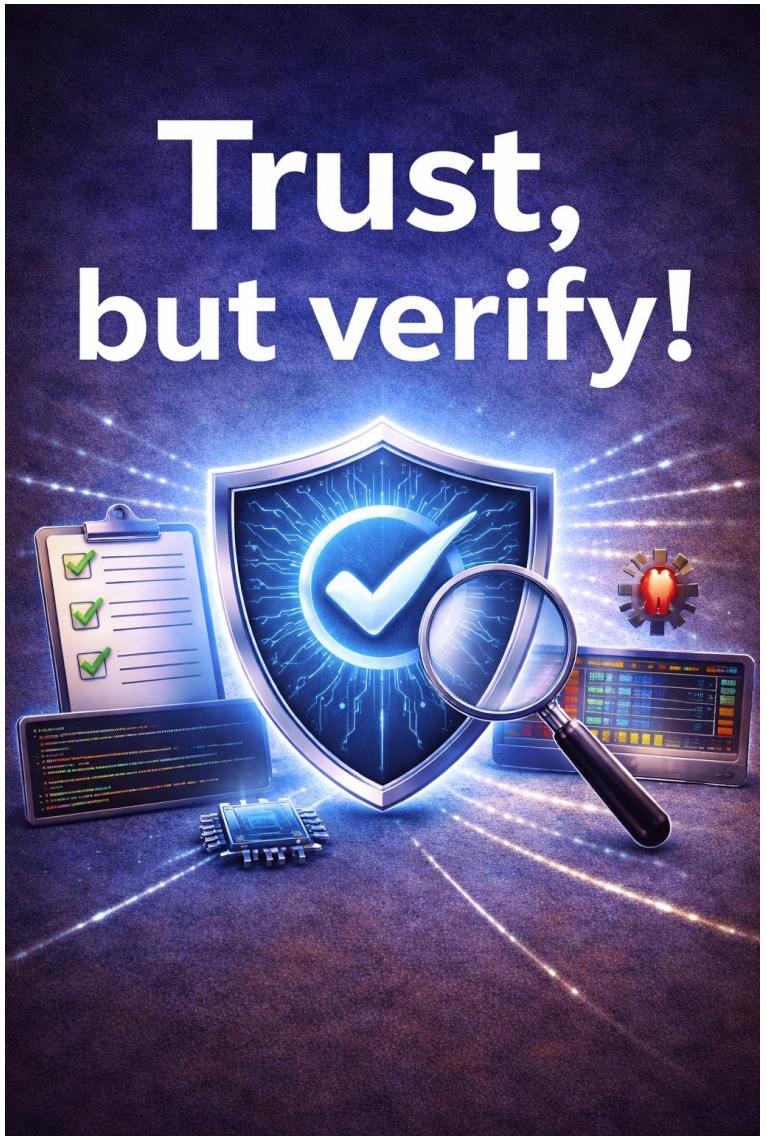
1. The Imitation Game

I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think." The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous, If the meaning of the words "machine" and "think" are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, "Can machines think?" is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.



Research summary

- Very powerful new tool
- Dependent on abstraction level
- Future work
  - Optimization on lower levels
  - Verification
  - Validation of learning model



How about education?

- Use of GenAI
  - Public talks: 5-10%
  - Own working group (AGRA day): around 50%
  - Bachelor students: 99%
- Training on the job

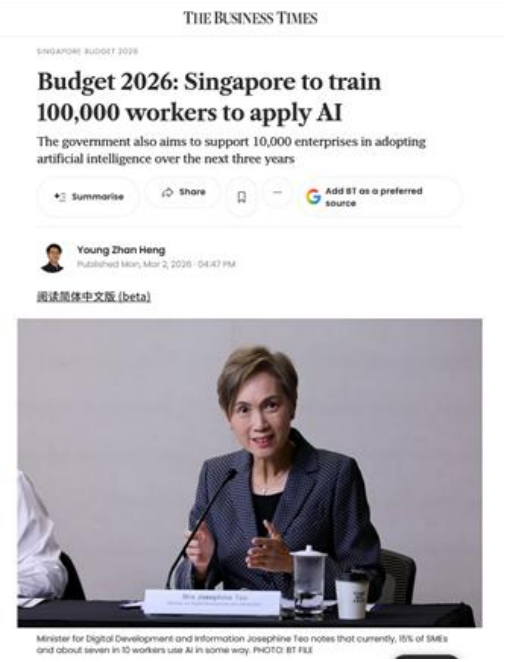

WirtschaftsWoche 100

WiWo+ MENTALE LEISTUNG

„Rund 60 Prozent der Beschäftigten brauchen neue Fähigkeiten“

Mentale Leistungsfähigkeit wird im KI-Zeitalter zu einem zentralen Wirtschaftsfaktor. Eine Expertin erklärt, wie sich „Brain Capital“ sichern und fördern lässt.

Tiffany Peach
03.02.2026 - 08:56 Uhr


THE BUSINESS TIMES

Budget 2026: Singapore to train 100,000 workers to apply AI

The government also aims to support 10,000 enterprises in adopting artificial intelligence over the next three years

Summarise Share Add BT as a preferred source

Young Zhan Heng

阅读简体中文版 (beta)

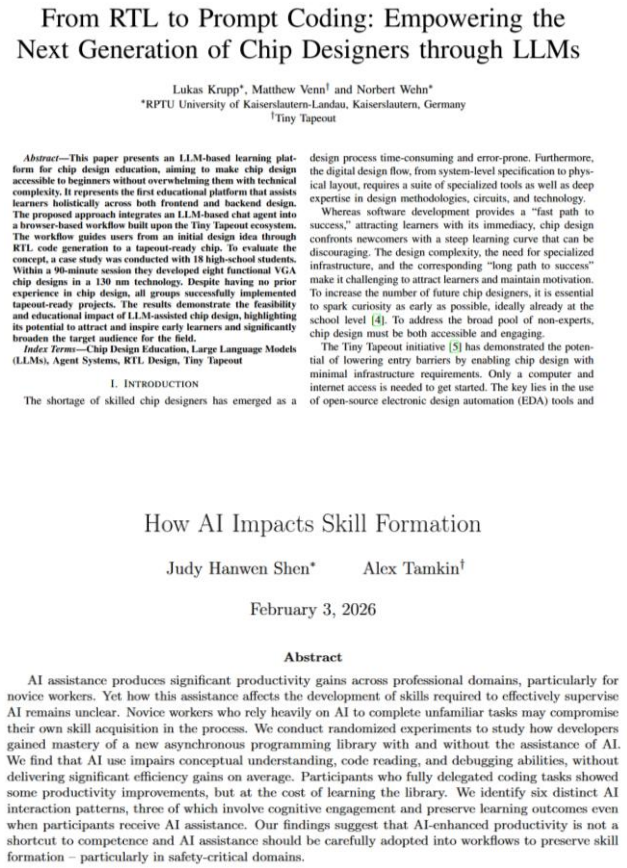
Minister for Digital Development and Information Josephine Teo notes that currently, 15% of SMEs and about seven in 10 workers use AI in some way. PHOTO: BT FILE

Teaching in the AI era

- Students use AI for assignments
  - AI explains concepts instantly
  - Assessment becomes fragile
- The **risk**
  - Outsourcing thinking
  - Superficial understanding
  - Acceleration without depth
  - Confidence without comprehension

Examples: potential vs. risk

- RPTU: “Within a 90-minute session they [18 high-school students] developed eight functional VGA chip designs in a 130 nm technology.”
- Anthropic (Claude): “Our findings suggest that AI-enhanced productivity is not a shortcut to competence and AI assistance should be carefully adopted into workflows to preserve skill formation – particularly in safety-critical domains.”



From RTL to Prompt Coding: Empowering the Next Generation of Chip Designers through LLMs

Lukas Krupp*, Matthew Venn† and Norbert Wehn*
*RPTU University of Kaiserslautern-Landau, Kaiserslautern, Germany
†Tiny Tapeout

***Abstract*—This paper presents an LLM-based learning platform for chip design education, aiming to make chip design accessible to beginners without overwhelming them with technical complexity. It represents the first educational platform that assists learners holistically across both frontend and backend design. The proposed approach integrates an LLM-based chat agent into a browser-based workflow built upon the Tiny Tapeout ecosystem. The workflow guides users from an initial design idea through RTL code generation to a tapeout-ready chip. To evaluate the concept, a case study was conducted with 18 high-school students. Within a 90-minute session they developed eight functional VGA chip designs in a 130 nm technology. Despite having no prior experience in chip design, all groups successfully implemented tapeout-ready projects. The results demonstrate the feasibility and educational impact of LLM-assisted chip design, highlighting its potential to attract and inspire early learners and significantly broaden the target audience for the field.**

***Index Terms*—Chip Design Education, Large Language Models (LLMs), Agent Systems, RTL Design, Tiny Tapeout**

I. Introduction

The shortage of skilled chip designers has emerged as a design process time-consuming and error-prone. Furthermore, the digital design flow, from system-level specification to physical layout, requires a suite of specialized tools as well as deep expertise in design methodologies, circuits, and technology.

Whereas software development provides a “fast path to success,” attracting learners with its immediacy, chip design confronts newcomers with a steep learning curve that can be discouraging. The design complexity, the need for specialized infrastructure, and the corresponding “long path to success” make it challenging to attract learners and maintain motivation. To increase the number of future chip designers, it is essential to spark curiosity as early as possible, ideally already at the school level [4]. To address the broad pool of non-experts, chip design must be both accessible and engaging.

The Tiny Tapeout initiative [5] has demonstrated the potential of lowering entry barriers by enabling chip design with minimal infrastructure requirements. Only a computer and internet access is needed to get started. The key lies in the use of open-source electronic design automation (EDA) tools and

How AI Impacts Skill Formation

Judy Hanwen Shen* Alex Tamkin†

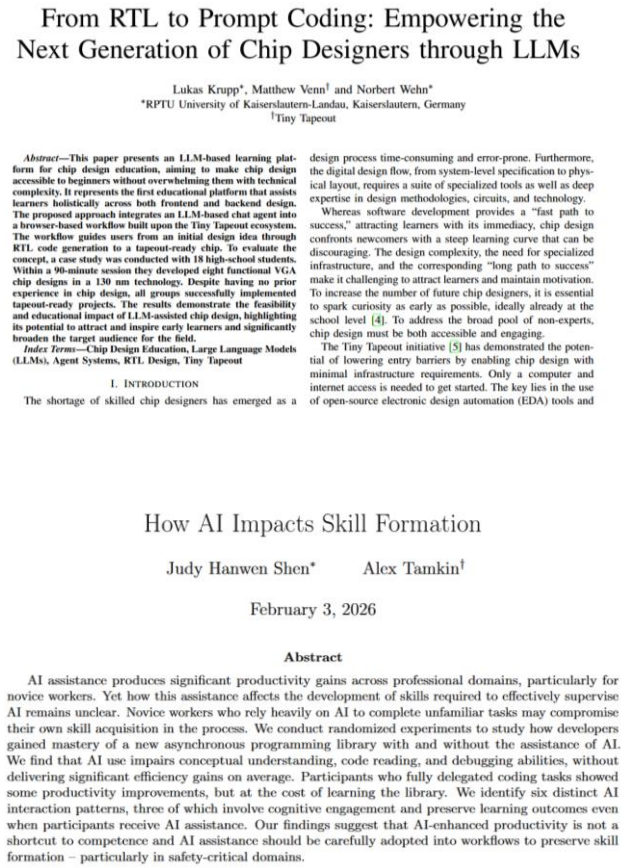
February 3, 2026

Abstract

AI assistance produces significant productivity gains across professional domains, particularly for novice workers. Yet how this assistance affects the development of skills required to effectively supervise AI remains unclear. Novice workers who rely heavily on AI to complete unfamiliar tasks may compromise their own skill acquisition in the process. We conduct randomized experiments to study how developers gained mastery of a new asynchronous programming library with and without the assistance of AI. We find that AI use impairs conceptual understanding, code reading, and debugging abilities, without delivering significant efficiency gains on average. Participants who fully delegated coding tasks showed some productivity improvements, but at the cost of learning the library. We identify six distinct AI interaction patterns, three of which involve cognitive engagement and preserve learning outcomes even when participants receive AI assistance. Our findings suggest that AI-enhanced productivity is not a shortcut to competence and AI assistance should be carefully adopted into workflows to preserve skill formation – particularly in safety-critical domains.

Examples: potential vs. risk

- RPTU: “Within a 90-minute session they [18 high-school students] developed eight functional VGA chip designs in a 130 nm technology.”
- Anthropic (Claude): “Our findings suggest that AI-enhanced productivity is not a shortcut to competence and AI assistance should be **carefully adopted** into workflows to preserve skill formation – particularly in **safety-critical domains**.”



From RTL to Prompt Coding: Empowering the Next Generation of Chip Designers through LLMs

Lukas Krupp*, Matthew Venn† and Norbert Wehn*
*RPTU University of Kaiserslautern-Landau, Kaiserslautern, Germany
†Tiny Tapeout

***Abstract*—This paper presents an LLM-based learning platform for chip design education, aiming to make chip design accessible to beginners without overwhelming them with technical complexity. It represents the first educational platform that assists learners holistically across both frontend and backend design. The proposed approach integrates an LLM-based chat agent into a browser-based workflow built upon the Tiny Tapeout ecosystem. The workflow guides users from an initial design idea through RTL code generation to a tapeout-ready chip. To evaluate the concept, a case study was conducted with 18 high-school students. Within a 90-minute session they developed eight functional VGA chip designs in a 130 nm technology. Despite having no prior experience in chip design, all groups successfully implemented tapeout-ready projects. The results demonstrate the feasibility and educational impact of LLM-assisted chip design, highlighting its potential to attract and inspire early learners and significantly broaden the target audience for the field.**

***Index Terms*—Chip Design Education, Large Language Models (LLMs), Agent Systems, RTL Design, Tiny Tapeout**

I. Introduction

The shortage of skilled chip designers has emerged as a design process time-consuming and error-prone. Furthermore, the digital design flow, from system-level specification to physical layout, requires a suite of specialized tools as well as deep expertise in design methodologies, circuits, and technology.

Whereas software development provides a “fast path to success,” attracting learners with its immediacy, chip design confronts newcomers with a steep learning curve that can be discouraging. The design complexity, the need for specialized infrastructure, and the corresponding “long path to success” make it challenging to attract learners and maintain motivation. To increase the number of future chip designers, it is essential to spark curiosity as early as possible, ideally already at the school level [4]. To address the broad pool of non-experts, chip design must be both accessible and engaging.

The Tiny Tapeout initiative [5] has demonstrated the potential of lowering entry barriers by enabling chip design with minimal infrastructure requirements. Only a computer and internet access is needed to get started. The key lies in the use of open-source electronic design automation (EDA) tools and

How AI Impacts Skill Formation

Judy Hanwen Shen* Alex Tamkin†

February 3, 2026

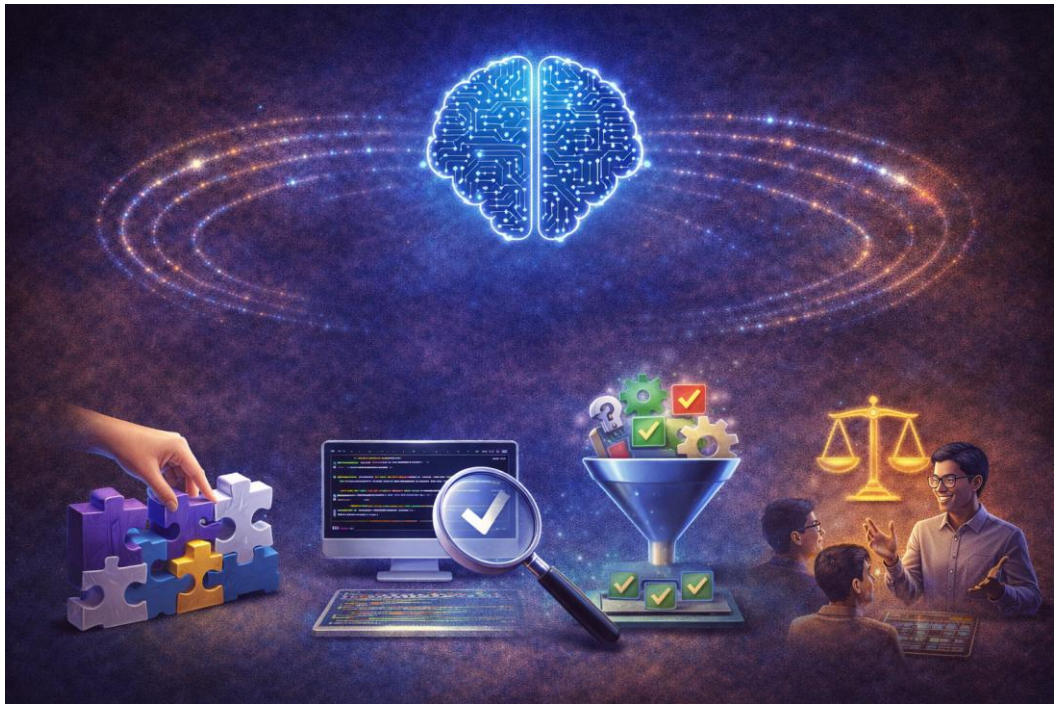
Abstract

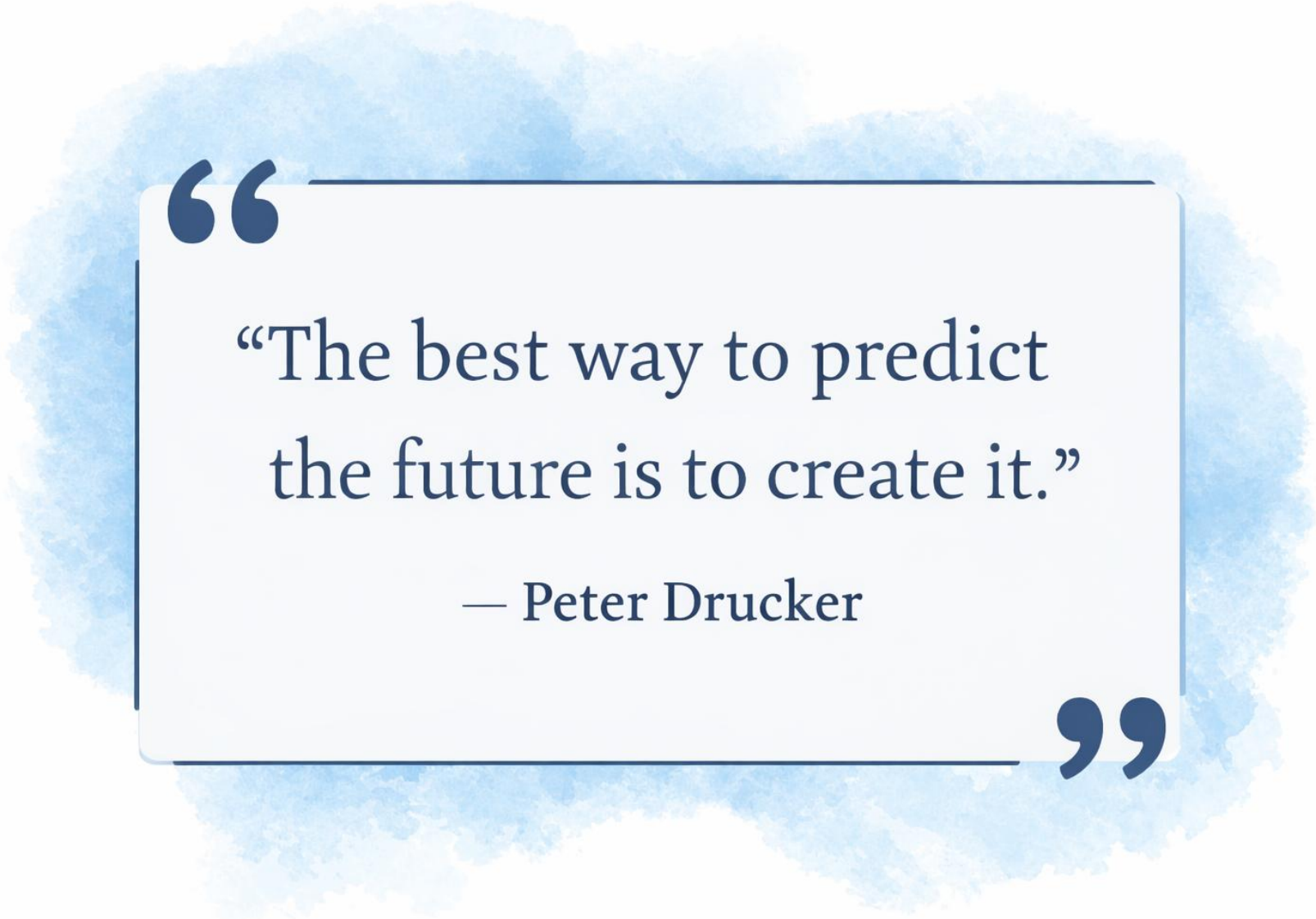
AI assistance produces significant productivity gains across professional domains, particularly for novice workers. Yet how this assistance affects the development of skills required to effectively supervise AI remains unclear. Novice workers who rely heavily on AI to complete unfamiliar tasks may compromise their own skill acquisition in the process. We conduct randomized experiments to study how developers gained mastery of a new asynchronous programming library with and without the assistance of AI. We find that AI use impairs conceptual understanding, code reading, and debugging abilities, without delivering significant efficiency gains on average. Participants who fully delegated coding tasks showed some productivity improvements, but at the cost of learning the library. We identify six distinct AI interaction patterns, three of which involve cognitive engagement and preserve learning outcomes even when participants receive AI assistance. Our findings suggest that AI-enhanced productivity is not a shortcut to competence and AI assistance should be carefully adopted into workflows to preserve skill formation – particularly in safety-critical domains.

Summary teaching

- Lot of potential

- Requires changes in teaching
  - Teach abstraction
    - Teach decomposition
    - Teach validation
    - Teach responsibility




“The best way to predict
the future is to create it.”
— Peter Drucker

On My Perfect Life: A Tenured Position While AI Does The Job

Rolf Drechsler

University of Bremen, Germany
DFKI Bremen, Germany
drechsler@uni-bremen.de