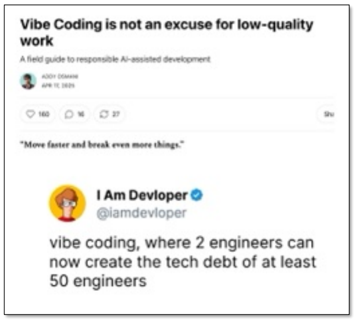
Christof Teuscher

ECE 410/510: Hardware for AI and ML

Codefest #5: Going down to physical/transistors (or CLBs) level

Portland State University
Department of Electrical and Computer Engineering (ECE)
www.teuscher-lab.com
teuscher@pdx.edu



Vibe Coding is not an excuse for low-quality work

A field guide to responsible AI-assisted development

ADDY OSMANI
APR 17, 2025

160 16 27

"Move faster and break even more things."

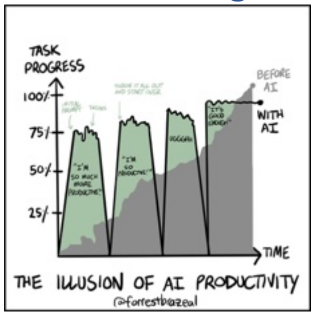
I Am Devloper @iamdevloper

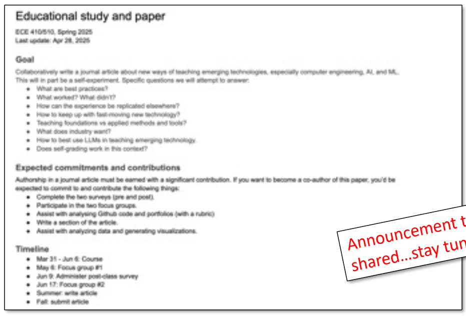
vibe coding, where 2 engineers can now create the tech debt of at least 50 engineers

https://addyo.substack.com/p/vibe-coding-is-not-an-excuse-for

---

Vibe coding





Educational study and paper

ECE 410/510, Spring 2025
Last update: Apr 28, 2025

Goal

Collaboratively write a journal article about new ways of teaching emerging technologies, especially computer engineering, AI, and ML. This will in part be a self-experiment. Specific questions we will attempt to answer:

- What are best practices?
- What worked? What didn't?
- How can the experience be replicated elsewhere?
- How to keep up with fast-moving new technology?
- Teaching foundations vs applied methods and tools?
- What does industry want?
- How to best use LLMs in teaching emerging technology
- Does self-grading work in this context?

Expected commitments and contributions

Authorship in a journal article must be earned with a significant contribution. If you want to become a co-author of this paper, you'd be expected to commit to and contribute the following things:

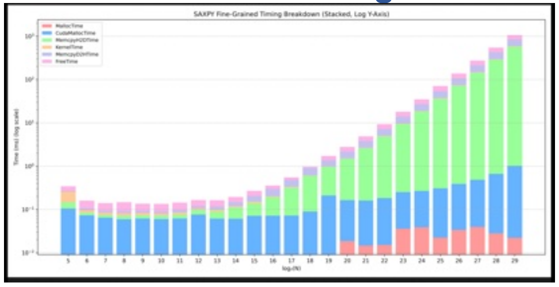
- Complete the two surveys (pre and post).
- Participate in the two focus groups.
- Assist with analysing Github code and portfolios (with a rubric)
- Write a section of the article.
- Assist with analyzing data and generating visualizations.

Timeline

- Mar 31 - Jun 6: Course
- May 6: Focus group #1
- Jun 9: Administer post-class survey
- Jun 17: Focus group #2
- Summer: write article
- Fall: submit article

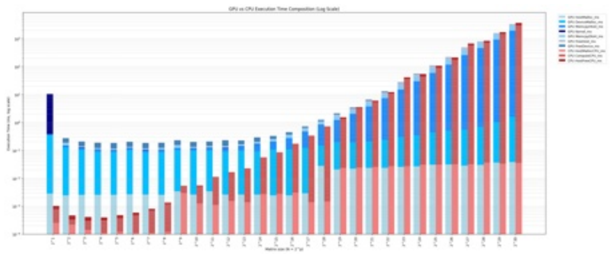
Announcement to be shared…stay tuned

Week 4 challenges



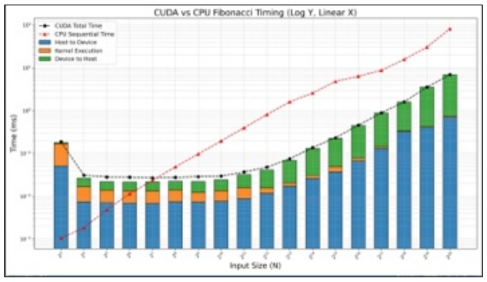
Source: Stephen Weeks

Week 4 challenges



Source: Eric Zhou

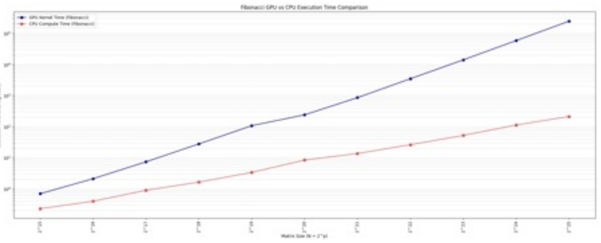
Week 4 challenges



Source: Stephen Weeks

teuscher Lab teuscher-lab.com Portland State UNIVERSITY

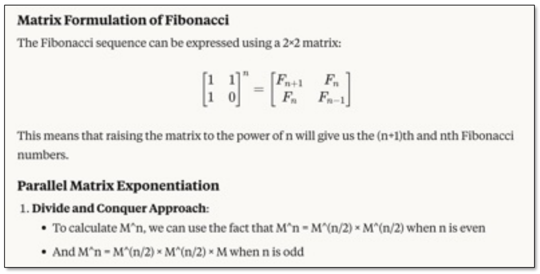
Week 4 challenges



Source: Eric Zhou

teuscher Lab teuscher-lab.com Portland State UNIVERSITY

Fibonacci



Matrix Formulation of Fibonacci

The Fibonacci sequence can be expressed using a 2×2 matrix:

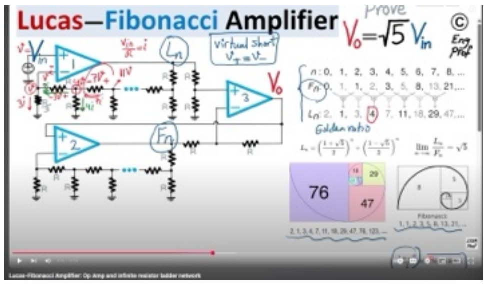
$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$

This means that raising the matrix to the power of n will give us the (n+1)th and nth Fibonacci numbers.

Parallel Matrix Exponentiation

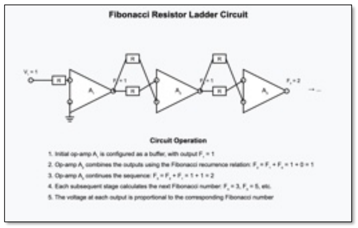
1. **Divide and Conquer Approach:**
   - To calculate M^n, we can use the fact that M^n = M^(n/2) × M^(n/2) when n is even
   - And M^n = M^(n/2) × M^(n/2) × M when n is odd

teuscher Lab teuscher-lab.com Portland State UNIVERSITY


Lucas—Fibonacci Amplifier
$V_o = \sqrt{5}\, V_{in}$
Prove
Virtual Short $V_+ = V_-$
L_n
F_n
n: 0, 1, 2, 3, 4, 5, 6, 7, 8, ...
F_n: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...
L_n: 2, 1, 3, 4, 7, 11, 18, 29, 47, ...
Golden ratio
$\lim_{n\to\infty} \frac{L_n}{F_n} = \sqrt{5}$
76
47
29
18
Lucas-Fibonacci Amplifier: Op Amp and infinite resistor ladder network

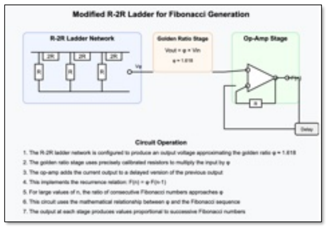
https://www.youtube.com/watch?v=jd_tWsTeLMY

teuscher Lab teuscher-lab.com Portland State UNIVERSITY



Circuit Operation

1. Initial op-amp A_1 is configured as a buffer, with output $F_1 = 1$
2. Op-amp A_2 combines the outputs using the Fibonacci recurrence relation: $F_2 = F_1 + F_0 = 1 + 0 = 1$
3. Op-amp A_3 continues the sequence: $F_3 = F_2 + F_1 = 1 + 1 = 2$
4. Each subsequent stage calculates the next Fibonacci number: $F_4 = 3$, $F_5 = 5$, etc.
5. The voltage at each output is proportional to the corresponding Fibonacci number



Circuit Operation

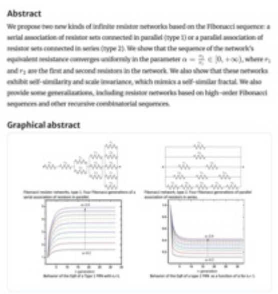
1. The R-2R ladder network is configured to produce an output voltage approximating the golden ratio φ ≈ 1.618
2. The golden ratio stage uses precisely calibrated resistors to multiply the input by φ
3. The op-amp adds the current output to a delayed version of the previous output
4. This implements the recurrence relation: F(n) = φ·F(n-1)
5. For large values of n, the ratio of consecutive Fibonacci numbers approaches φ
6. This circuit uses the mathematical relationship between φ and the Fibonacci sequence
7. The output at each stage produces values proportional to successive Fibonacci numbers

Thanks Claude…

teuscher Lab teuscher-lab.com Portland State UNIVERSITY

Fractality in resistive circuits: the Fibonacci resistor networks

The European Physical Journal B



Abstract

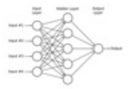
We propose two new kinds of infinite resistor networks based on the Fibonacci sequence: a serial association of resistor sets connected in parallel (type 1) or a parallel association of resistor sets connected in series (type 2). We show that the sequence of the network's equivalent resistance converges uniformly in the parameter $\alpha = \frac{r_2}{r_1} \in [0, +\infty)$, where r_1 and r_2 are the first and second resistors in the network. We also show that these networks exhibit self-similarity and scale invariance, which mimics a self-similar fractal. We also provide some generalizations, including resistor networks based on high-order Fibonacci sequences and other recursive combinatorial sequences.

Graphical abstract

teuscher Lab teuscher-lab.com Portland State UNIVERSITY

Challenge #16: Benchmarking SAXPY with PyTorch

Learning goals:
- Compare the performance of a simple feed-forward neural network (as seen in class) accelerated with CUDA vs accelerated with PyTorch.



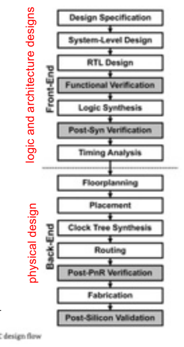
Tasks:
1. (Vibe) Code a CUDA-accelerated version of a simple multi-layer feedforward, e.g., 4 inputs, 5 hidden neurons, 1 output, fully connected (as seen in class).
2. (Vibe) Code the same network by using PyTorch.
3. Benchmark both implementations and compare. What can you conclude?
4. If you want to go further, increase the depth and the width of the network and compare its execution time for various sizes. What's the outcome? Can you beat PyTorch with CUDA? Or vice versa?

Challenge #17: Sorting on a systolic array

Learning goals:
- Learn how to implement Bubble sort on a systolic array.
- Evaluate its performance as a function of the problem size.

Tasks:
1. Design a systolic array that can do Bubble sort. What dimension does the array need to have?
2. (Vibe) Code a software version in your favorite language and test it.
3. Visualize the execution times for various sorting sizes. E.g., 10, 100, 1000, 10000, etc.

ASIC design flow

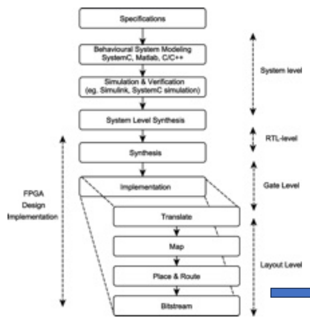


Python
Verilog/SystemVerilog/VHDL

Max. frequency, # transistors, etc.

Mishra, Ashutosh; Cha, Jaekwang; Park, Hyunbin; Kim, Shiho. Artificial Intelligence and Hardware Accelerators.

FPGA design flow


Specifications
Behavioural System Modeling SystemC, Matlab, C/C++
Simulation & Verification (eg. Simulink, SystemC simulation)
System Level Synthesis
Synthesis
Implementation
Translate
Map
Place & Route
Bitstream
System level
RTL-level
Gate Level
Layout Level
FPGA Design Implementation

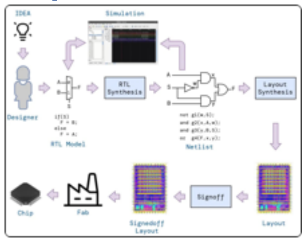
Max. frequency, # CLBs, etc

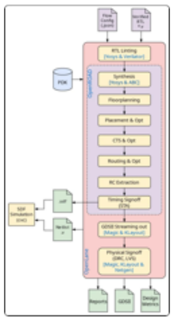
AMD Vivado™ Design Suite: Standard & Enterprise Edition

Features

https://www.amd.com/en/products/software/adaptive-socs-and-fpgas/vivado/vivado-buy.html

OpenLane 2 tools





https://efabless.com/openlane
https://github.com/efabless/openlane2
https://openlane2.readthedocs.io/en/latest/getting_started/newcomers/index.html

Main goals for today

1. Check out the OpenLane 2 tools.
2. Go to the physical level with a sample design (vibe-generated) or your own initial HW design.
3. Ready yourself for design iterations & rapid prototyping.

https://www.pacific-research.com/iterative-product-development

Getting started with OpenLane 2

To check out an example of an OpenLane 2-based flow right in your **browser**, try the Google Colab™ notebook at https://colab.research.google.com/github/efabless/openlane2/blob/main/notebook.ipynb

To set up OpenLane 2 on your **computer**, check out the Getting Started guide at the following link: https://openlane2.readthedocs.io/en/latest/getting_started/index.html

Workflow/toolchain (1)

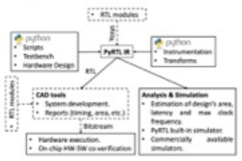
PyMTL (Mamba):

- https://pymtl.github.io
- User group: https://groups.google.com/g/pymtl-users
- An open-source hardware modeling, generation, simulation, and verification framework.
- MyHDL allows a subset of Python code to be translated to Verilog or VHDL. It offers co-simulation options where native Python code runs alongside a compiled simulation model of your hardware.
- Hardware-software co-simulation using PyMTL3:
  - Create your hardware model in PyMTL3
  - Develop software that will interact with the hardware
  - Set up the co-simulation environment
  - Run and analyze results

Workflow/toolchain (2)

PyRTL:

- https://ucsbarchlab.github.io/PyRTL
- PyRTL is an open-source Python-based hardware development toolkit.
- PyRTL provides a minimal set of hardware primitives, expressed as a Python class, which can then be extended with other classes and libraries as appropriate.
- Allows for fast design iteration in a variety of domains, including cryptography and machine learning.
- Paper:
  - Agile Hardware Development and Instrumentation with PyRTL
  - https://doi.org/10.1109/MM.2020.2997704



Workflow/toolchain (3)

Cocotb:

- https://www.cocotb.org
- cocotb is an open source coroutine-based co-simulation testbench environment for verifying VHDL and SystemVerilog RTL using Python.

MyHDL:

- https://www.myhdl.org
- MyHDL turns Python into a hardware description and verification language, providing hardware engineers with the power of the Python ecosystem.
- MyHDL designs can be converted to Verilog or VHDL.

pyUVM:

- https://github.com/pyuvm/pyuvm
- pyuvm is the Universal Verification Methodology (UVM) implemented in Python instead of SystemVerilog. pyuvm uses cocotb to interact with simulators and schedule simulation events.

yosys:

- https://github.com/YosysHQ/yosys
- Yosys is a framework for RTL synthesis and more. It currently has extensive Verilog-2005 support and provides a basic set of synthesis algorithms for various application domains.