Christof Teuscher

ECE 410/510: Hardware for AI and ML

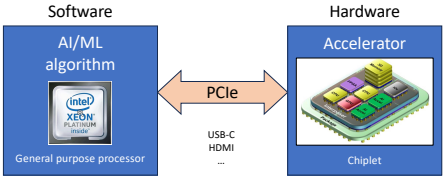
Codefest #2: Bootstrapping the final project

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

teuscher Lab teuscher-lab.com

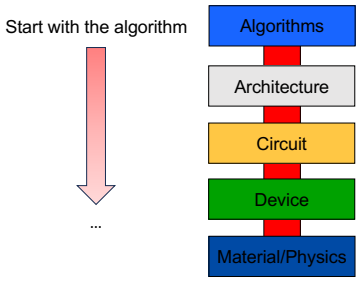
Portland State UNIVERSITY

Project vision



Design, test, and benchmark a co-processor chiplet that accelerates parts of some AI/ML code. Start with a blank slate for your design.
[Alternative]: design a stand-alone chiplet.

Compute stack



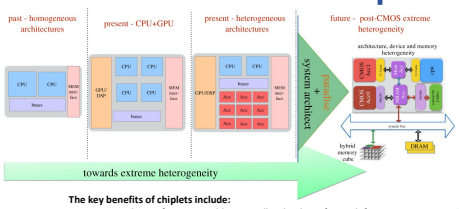
Example AI/ML applications

- **Computer Vision**
  - Image classification and object detection
  - Facial recognition (with privacy considerations)
  - Medical image analysis
  - Quality control in manufacturing
  - Autonomous vehicles and robotics
- **Natural Language Processing**
  - Machine translation
  - Sentiment analysis
  - Text summarization
  - Chatbots and virtual assistants
  - Information extraction from documents
- **Recommendation Systems**
  - E-commerce product recommendations
  - Content recommendations (streaming services, news)
  - Advertisement targeting
  - Social media feed curation
- **Predictive Analytics**
  - Financial forecasting and risk assessment
  - Predictive maintenance for equipment
  - Supply chain optimization
  - Healthcare outcome prediction
  - Customer churn prediction
- **Anomaly Detection**
  - Fraud detection in finance
  - Network security monitoring
  - Manufacturing defect detection
  - Health monitoring systems
- **Time Series Analysis**
  - Stock market prediction
  - Energy load forecasting
  - Weather forecasting
  - Sales forecasting
- **Reinforcement Learning**
  - Game playing agents
  - Robotics control
  - Resource management optimization
  - Autonomous systems
- **Speech Recognition and Synthesis**
  - Voice assistants
  - Transcription services
  - Accessibility tools
  - Voice-based authentication
- **Generative AI**
  - Text generation (creative writing, code, content)
  - Image, video, and audio generation
  - Design assistance (architecture, fashion, graphics)
  - Synthetic data generation

What is a chiplet?

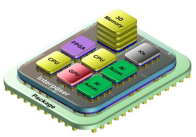
- A chiplet is a **small, specialized piece of silicon that can be combined with other chiplets to form a complete integrated circuit (IC) or system-on-chip (SoC).**
- Instead of manufacturing one large monolithic chip, the chiplet approach breaks down complex processor designs into smaller functional blocks that can be manufactured separately and then integrated together using advanced packaging technologies.
- Chiplets are connected using **high-speed, short-range communication interfaces** that enable them to work together as if they were on a single piece of silicon.
- This **modular approach** has been widely adopted by companies like AMD, Intel, and TSMC to create more efficient and powerful processors while overcoming the limitations of traditional monolithic chip design

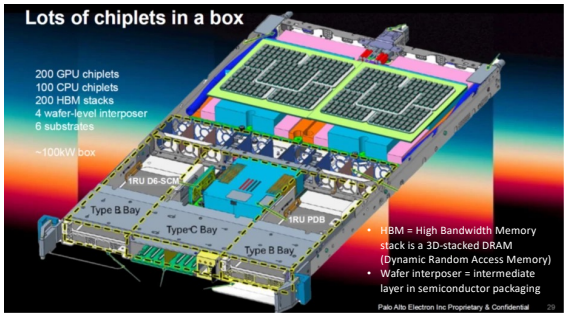
What is a chiplet?



The key benefits of chiplets include:

- Improved manufacturing yields, as smaller dies have fewer defects
- Better scalability for different performance and power requirements
- The ability to mix and match process technologies optimized for specific functions
- Reduced costs by reusing proven IP blocks across different products





https://semiengineering.com/chiplets-where-are-we-today

Communication interfaces

- **Advanced Interface Bus** (AIB): Developed by Intel, AIB is an open-source, die-to-die interconnect technology that enables high-bandwidth, low-power communication between chiplets. It supports both parallel and serial communication modes.
- **Universal Chiplet Interconnect Express** (UCIe): An industry-standard specification backed by major companies like AMD, Intel, Arm, and TSMC. UCIe aims to establish an open ecosystem for chiplet integration with standardized physical interfaces, protocols, and power management.
- **High Bandwidth Memory (HBM) Interface**: While primarily for connecting memory chiplets, this interface uses through-silicon vias (TSVs) and microbumps to achieve very high bandwidth connections.
- **Bunch of Wires (BoW):** A simpler, more cost-effective interface for medium-bandwidth connections between chiplets, developed through the Open Compute Project.
- TSMC's **Integrated Fan-Out (InFO)** and **Chip-on-Wafer-on-Substrate (CoWoS):** These are packaging technologies that include specific interconnect interfaces for connecting multiple chiplets.

Unclear of IP is available for any of these…

How to pick a project

Dos

- Pick something exciting, at the forefront of technology
- Pick something that challenges you
- Failure should be an option
- Something new that you want to learn
- Start with a blank slate, but it's OK to draw inspiration from existing architectures.

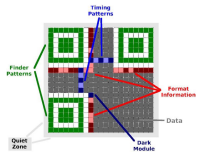
Dont's

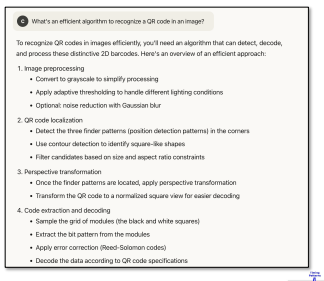
- No relying on current accelerators, e.g., GPUs
- No traditional CPUs.
- No cache algorithms, no branch predictors, etc.!
- No teamwork allowed. You must have your own project. If several want to work on the same problem, you'd be competing with each other (in a friendly way!).

Example

QR code recognizer for an edge device.

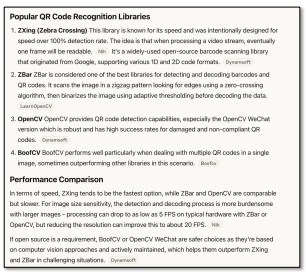
- Must be fast and ultra low-power.




What's an efficient algorithm to recognize a QR code in an image?

To recognize QR codes in images efficiently, you'll need an algorithm that can detect, decode, and process these distinctive 2D barcodes. Here's an overview of an efficient approach:

1. Image preprocessing
   - Convert to grayscale to simplify processing
   - Apply adaptive thresholding to handle different lighting conditions
   - Optional: noise reduction with Gaussian blur
2. QR code localization
   - Detect the three finder patterns (position detection patterns) in the corners
   - Use contour detection to identify square-like shapes
   - Filter candidates based on size and aspect ratio constraints
3. Perspective transformation
   - Once the finder patterns are located, apply perspective transformation
   - Transform the QR code to a normalized square view for easier decoding
4. Code extraction and decoding
   - Sample the grid of modules (the black and white squares)
   - Extract the bit pattern from the modules
   - Apply error correction (Reed-Solomon codes)
   - Decode the data according to QR code specifications


Popular QR Code Recognition Libraries

1. **ZXing (Zebra Crossing)** This library is known for its speed and was intentionally designed for speed over 100% detection rate. The idea is that when processing a video stream, eventually one frame will be readable. Nin It's a widely-used open-source barcode scanning library that originated from Google, supporting various 1D and 2D code formats. Dynamsoft
2. **ZBar** ZBar is considered one of the best libraries for detecting and decoding barcodes and QR codes. It scans the image in a zigzag pattern looking for edges using a zero-crossing algorithm, then binarizes the image using adaptive thresholding before decoding the data. LearnOpenCV
3. **OpenCV** OpenCV provides QR code detection capabilities, especially the OpenCV WeChat version which is robust and has high success rates for damaged and non-compliant QR codes. Dynamsoft
4. **BoofCV** BoofCV performs well particularly when dealing with multiple QR codes in a single image, sometimes outperforming other libraries in this scenario. BoofCv

Performance Comparison

In terms of speed, ZXing tends to be the fastest option, while ZBar and OpenCV are comparable but slower. For image size sensitivity, the detection and decoding process is more burdensome with larger images - processing can drop to as low as 5 FPS on typical hardware with ZBar or OpenCV, but reducing the resolution can improve this to about 20 FPS. Nin

If open source is a requirement, BoofCV or OpenCV WeChat are safer choices as they're based on computer vision approaches and actively maintained, which helps them outperform ZXing and ZBar in challenging situations. Dynamsoft

Heilmeier questions

1. **What are you trying to do**? Articulate your objectives using absolutely no jargon.
2. **How is it done today**, and what are the limits of current practice?
3. **What is new in your approach** and why do you think it will be successful?
4. Who cares? If you are successful, what difference will it make?
5. What are the risks?
6. How much will it cost?
7. How long will it take?
8. What are the mid-term and final "exams" to check for success?

https://www.darpa.mil/about/heilmeier-catechism

Week	Monday	Wednesday (Codefest)
2	HW/AI/ML overview + codesign overview	Start main project: pick workload, start analysis, benchmark, …
3	GPU architecture and programming for AI	Drafting a HW architecture, creating a model
4	Deep neural networks on GPUs	Coding HW description
5	Transformers on GPUs	First simulation + refinement
6	In-memory computation	Improving initial design
7	Neuromorphic chips: TrueNorth, Loihi, Akida	Simulation + refinement
8	Neuromorphic computing with mem-devices	Synthesizing design + benchmarking
9	Hardware accelerators for embedded systems	Final improvements
10	Emerging technologies and future directions	Final tests, validation, verification, benchmarking

Codefest goals

- **Rapid prototyping**, solve problems, and create complex designs using “vibe coding.”
- Learn how to use **LLMs** efficiently to create complex design we couldn’t create without.
- Use LLMs to learn and to gain a deeper understanding of the subject matter.
- Collaborate, share and learn from each other.
- Present, document, communicate.
- **Motivate you to go home and brings things to the next level.**