ECE382M.20: System-on-Chip (SoC) Design

Lecture 0 – Class Overview

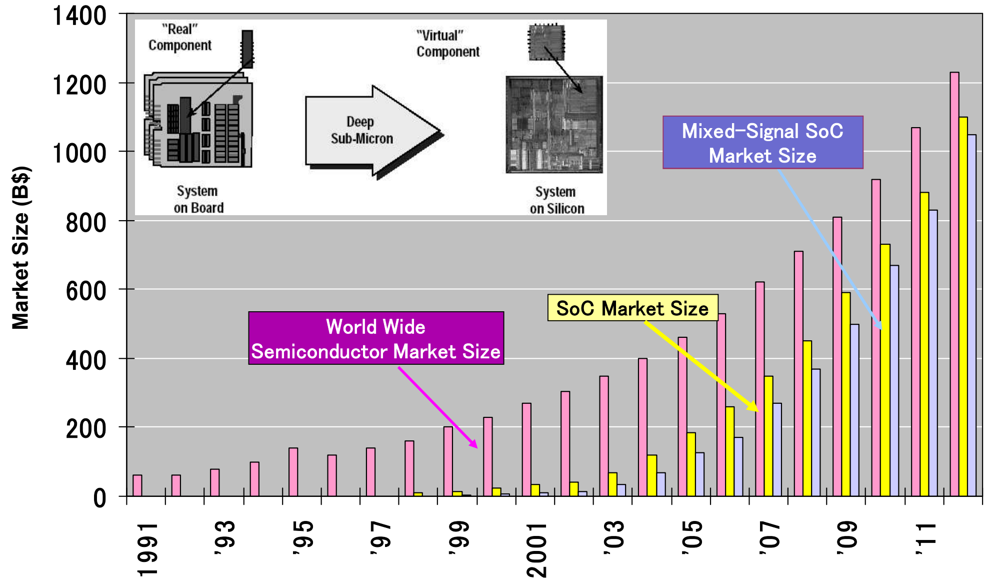
Andreas Gerstlauer
Electrical and Computer Engineering
The University of Texas at Austin
gerstl@ece.utexas.edu

The University of Texas at Austin
Chandra Department of Electrical and Computer Engineering
Cockrell School of Engineering

Lecture 0: Outline

- **Introduction**
  - Systems-on-Chip (SoCs)
  - Design flow
- **Course information**
  - Overview, goals, topics
  - Administration
  - Labs and project
- **Class project**
  - LLM inference
  - SoC implementation

System-on-Chip (SoC) Era

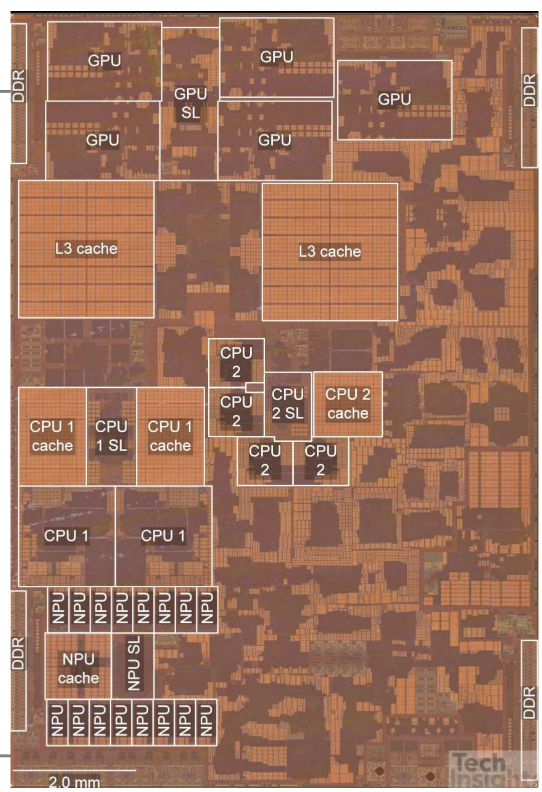


Source: SONY Corp & Market Estimates

Apple A15 SoC

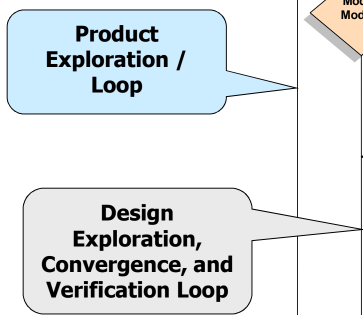
- **6-core ARMv8**
  - 2 big (Avalanche)
  - 4 little (Blizzard)
- **5-core GPU**
  - Proprietary
- **Co-processor**
  - M-class ARM
- **16-core neural engine**
  - Neural Processing Units (NPUs)
- **Accelerators**
  - Image processor (ISP)
  - Lots others…
- **15 billion transistors (2021)**

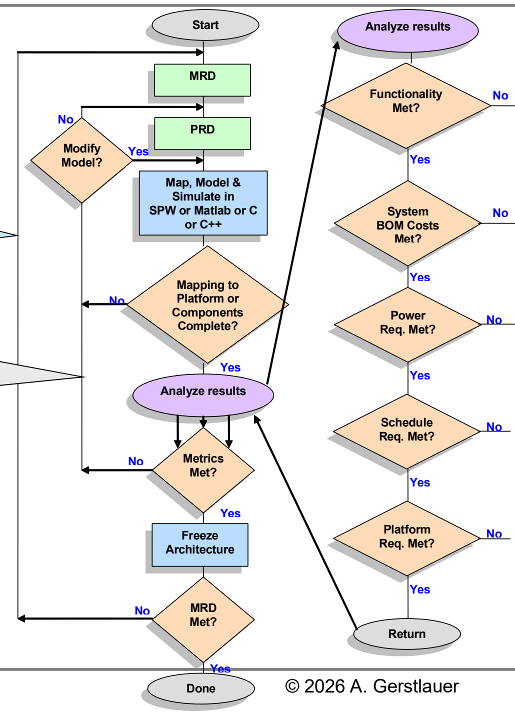
Source: AnandTech, Angstronomics & TechInsights



SoC Design Flow

MRD: Marketing Requirements Doc.
PRD: Product Requirements Doc.

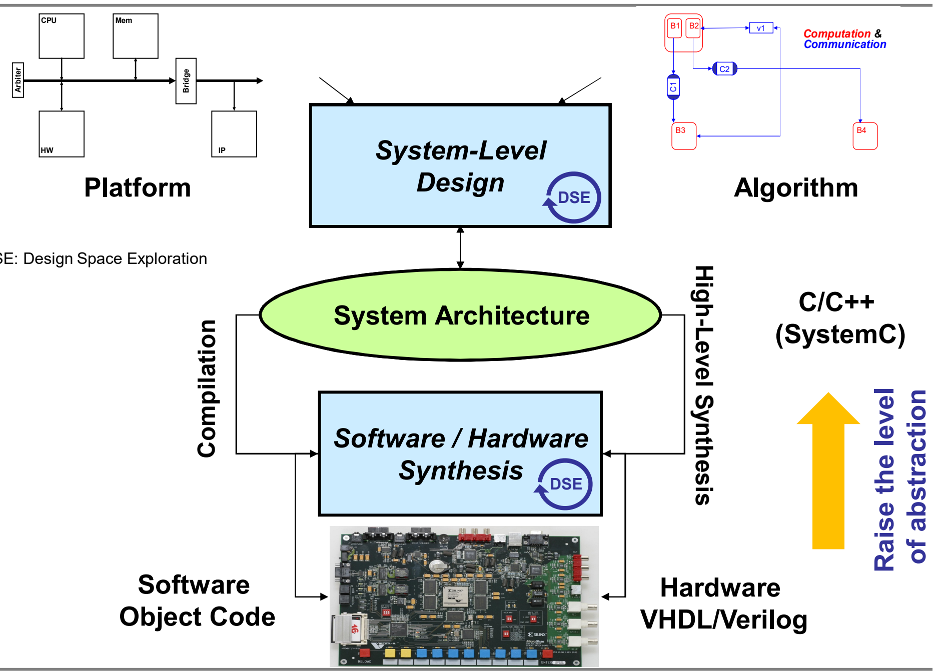




- **Traditionally done at RTL**

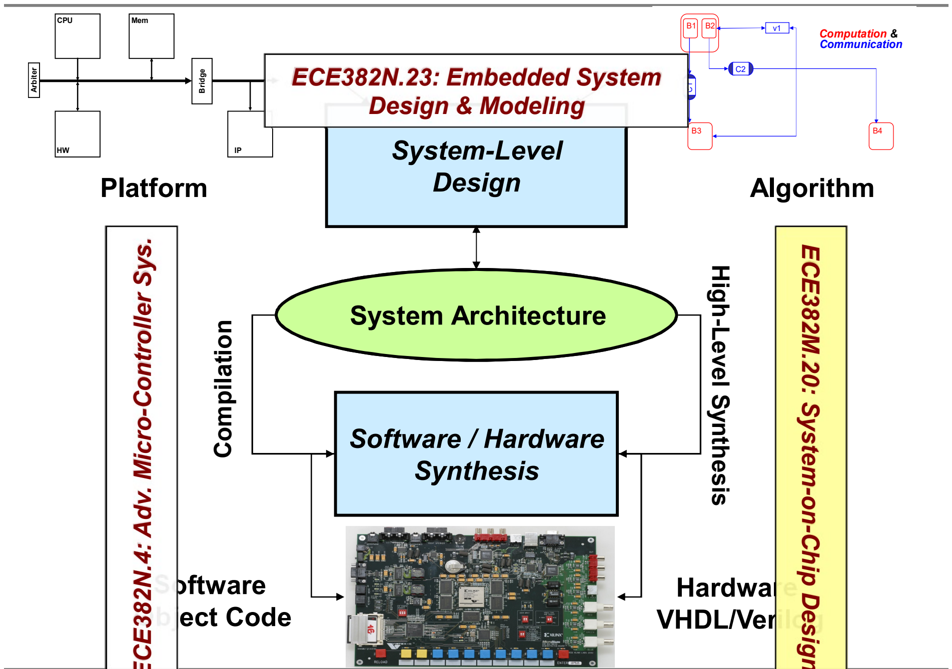
ECE382M.20:SoC Design, Lecture 0 © 2026 A. Gerstlauer 5

System-Level Design

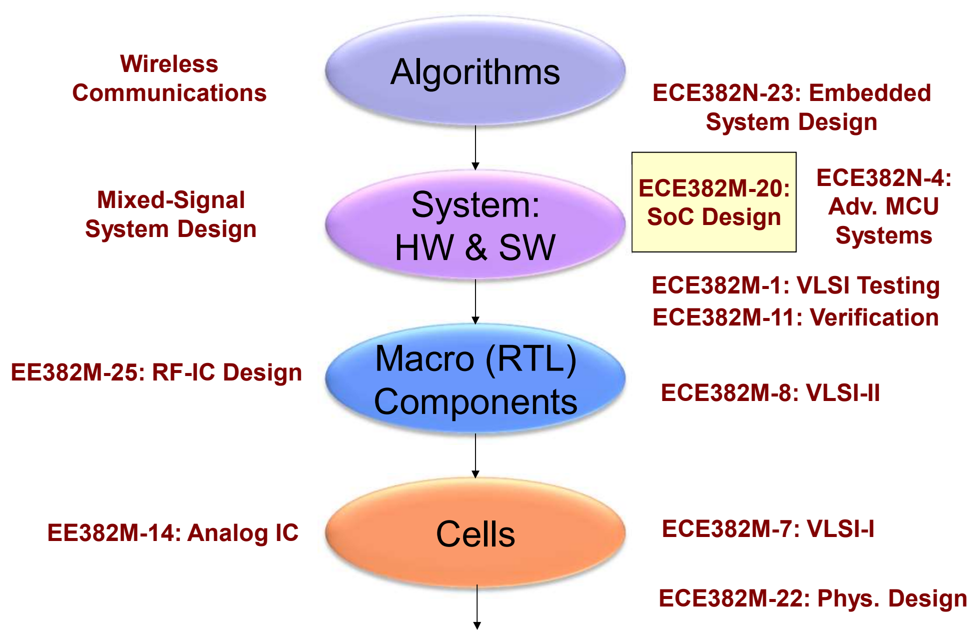


ECE382M.20:SoC Design, Lecture 0 © 2026 A. Gerstlauer 6

UT ECE Courses



Integrated Circuits and Systems at UT



Course Overview

- **Provide an understanding of the concepts, issues, and process of designing highly integrated SoCs**
  - Systematic hardware/software co-design & co-verification
  - Industry-standard/advanced SoC design flow
  - Manual SoC architecture definition + component synthesis

➢ **Class labs and project: LLM edge inference SoC**
  - Large Language Model (LLM) edge inference
    - Hardware/software co-design
  - State-of-the-art synthesis and verification tools
    - Virtual platform prototyping and simulation in SystemC/TLM
    - High-level hardware synthesis from C++ to RTL
  - ARM-based FPGA prototyping platform
    - ARM processor, I/O devices, memory components, hardware accelerators

Course Goals

- **Course is designed to learn about:**
  - Early functional and nonfunctional performance analysis to support design decisions.
  - Analysis and optimization of hardware/software tradeoffs, algorithms, and architectures based on requirements and implementation constraints.
  - Analyze tradeoffs and explore architecture and mico-architecture design spaces to develop and synthesize custom hardware accelerators.
  - Hardware, software, and interface synthesis.
  - Interface design.
  - Co-simulation to validate system functionality.
  - Examples of applications and systems developed using a co-design approach.
  - Intellectual property, reuse, and verification issues.

Course Topics and Prerequisites

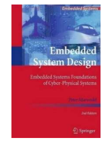
- **Covered in class**
  - System-level and SoC design methodologies and tools;
  - HW/SW co-design: analysis, partitioning, scheduling, hardware acceleration;
  - High-Level Synthesis (HLS): allocation, scheduling, binding algorithms for C-to-RTL synthesis;
  - Virtual prototyping: virtual platform models, electronic system-level languages, and HW/SW co-simulation;
  - SoC integration: SoC communication architectures, accelerator interfacing, verification and test;
  - FPGA prototyping of HW/SW systems.
- **Prerequisites**
  - Working knowledge of C/C++ software development
  - Digital hardware design and HDLs (VHDL/Verilog)
  - Embedded system design and HW/SW interfacing basics
  - Basic computer architecture knowledge

Course Administration

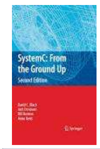
- **Instructors**
  - Instructor: Andreas Gerstlauer <gerstl@ece.utexas.edu>
    - Office hours: TTh 11am-noon, EER 5.882
  - TAs: James Boyle <james.boyle@utexas.edu>
    Daniel Peralta Velazquez <dperaltav@utexas.edu>
    - Office hours: M & Th 4-6pm, EER 0.810
- **Information**
  - Web: http://users.ece.utexas.edu/~gerstl/ece382m_f26
    - Lecture notes, homework and lab exercises
  - Canvas
    - Announcements, assignments, grades
  - Ed Discussion
    - All non-personal class discussions
- **Dates (tentative)**
  - Exam: November 5, in class
  - Project design reviews: Dec. 1 & 3, in class
  - Final project presentations: Dec. 8 (finals week)

Textbooks

- **No required textbook**
- **Suggested references**
  - P. Marwedel, *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things*, 4th edition, Springer, 2021
    - Available online under open access or through UT library



  - D. Black, J. Donovan, B. Bunton, A. Keist, *SystemC: From the Ground Up*, 2nd edition, Springer, 2010
    - Electronic version available from the UT library



  - S. Pasricha, N. Dutt, *On-Chip Communication Architectures (System-on-Chip Interconnect)*, Morgan Kaufman, 2008
    - On reserve in the library

  - G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994
    - Out of print, but can be ordered; on reserve in the library

Course Policies

- **Grading:**
  - Homework: 10%
  - Lab assignments: 20%
  - Exam: 30%
  - Project: 40%
- **Late penalties:**
  - 20% per day (24 hours)
- **Academic dishonesty**
  - Homeworks
    - Ok to discuss questions and problems with others
    - But turn in own, independent solution
  - Labs and project
    - In teams, one report and presentation
    - Collaboration encouraged and desired
  - Plagiarism
    - Use of any outside source of information without quoting or referencing is cheating
    - AI use is allowed, but needs to be disclosed

Labs (tentative)

- **Lab 1: Application analysis (3 weeks, due Sep 21)**
  - Profiling of code on Arm board
    - Identify time consuming bottlenecks to optimize
    - Identify, partition and isolate code for hardware acceleration
  - Software exploration
    - Model complexity, performance vs. prediction quality tradeoffs
    - Quantization, software optimizations, etc.
- **Lab 2: Accelerator design (3 weeks, due Oct 12)**
  - High-level synthesis of accelerator from C++ to RTL
    - Accelerator design space exploration using Xilinx Vitis HLS tool
  - Verification of synthesis results
    - Testbench around standalone SystemC and RTL hardware module
- **Lab 3: System architecture (3 weeks, due Nov 2)**
  - Hardware acceleration and system architecture
    - Hardware/software interface, communication architecture and drivers
  - Virtual platform prototyping and modeling
    - Interface hardware accelerator as SystemC module
    - Co-simulation w/ software and firmware on ARM-Linux CPU

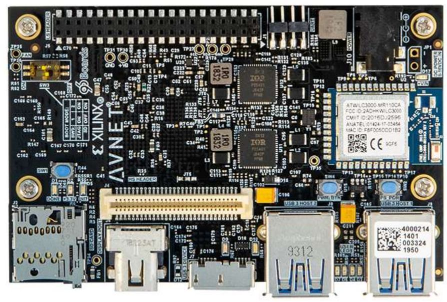
Class Project

- **Implementation on prototyping board**
  - Synthesizing hardware accelerator(s) into the FPGA
    - Synthesis and download using Xilinx software
  - C/C++ program on the ARM host processor
    - Cross-compilation or development under Ubuntu Linux on the ARM
  - Hardware/software interfacing and communication
    - Software drivers and hardware bus interfaces
  - Analysis and validation of product metrics
    - Estimate timing, area and power consumption

➢ **Low-power SoC implementation of LLM inference**
  - Reference design of LLM inference ASIC
    - FPGA-based implementation as prototype of ASIC design
  - To be incorporated into edge and IoT devices
    - Satisfy market and product requirements

Prototyping Board

- **Ultra96 V2 by Avnet**
  - Xilinx Arm+FPGA programmable SoC
  - 2GB DDR4 RAM
  - microSD card socket
  - Wifi/Bluetooth
    (no wired Ethernet!)
  - 1x USB3 Micro-B upstream
  - 2x USB3 Type A downstream
  - DisplayPort
  - USB UART/JTAG port
    via daughter board

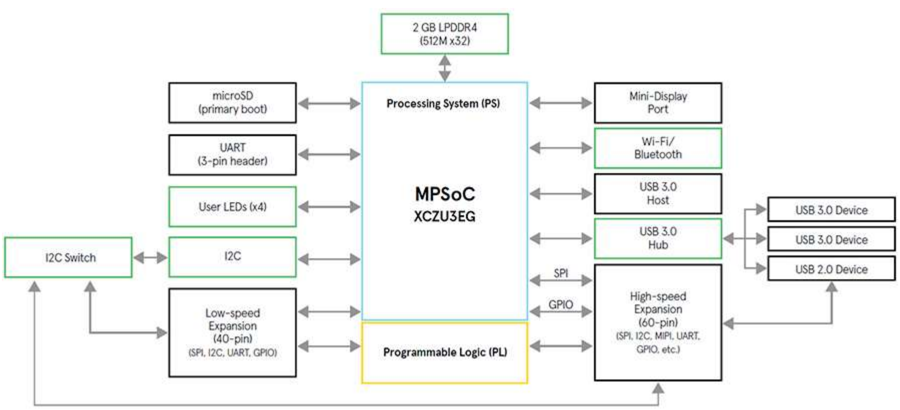


- **Based on Xilinx Zynq UltraScale+ MPSoC**
  - Quad-core Arm A53 + dual-core Arm R5 + Arm Mali GPU
  - Xilinx UltraScale FPGA fabric (16nm FinFET)

➢ **We will target SoC supporting one Arm A53 @ 300Mhz**

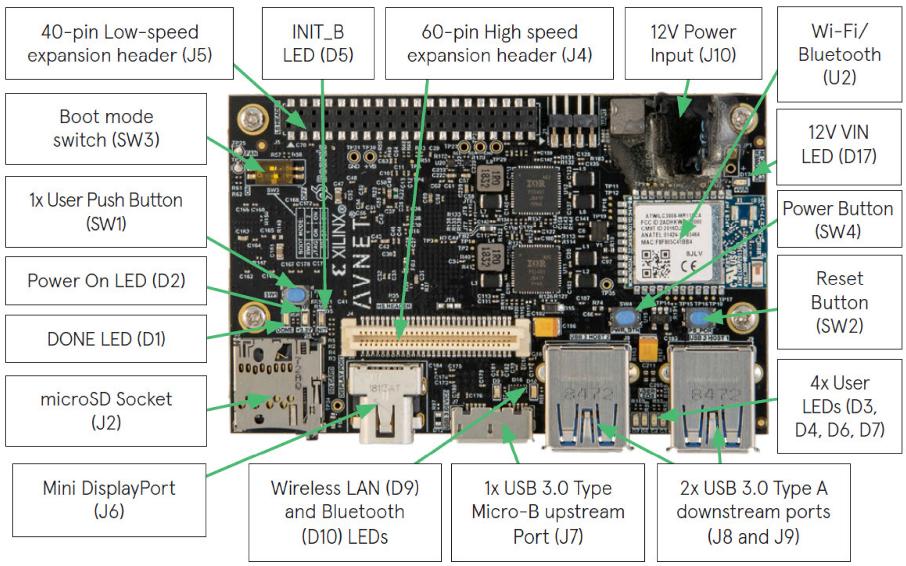
ECE382M.20:SoC Design, Lecture 0 © 2026 A. Gerstlauer 17

Ultra96 Block Diagram

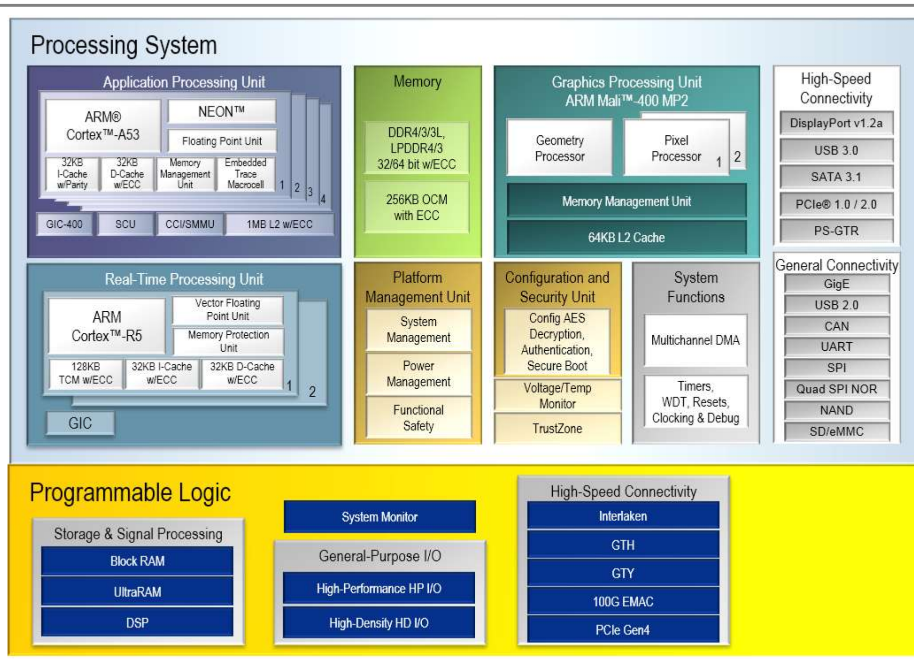


ECE382M.20:SoC Design, Lecture 0 © 2026 A. Gerstlauer 18

Ultra96 Board Layout



Xilinx Zynq UltraScale+ MPSoC ZU3EG



Ultra96 Software Environment

- **We provide SD card running Ubuntu 22.04 Linux OS**
  - You will have root privileges
  - Two partitions
    - `/media/boot` – bootloader and Linux kernel
    - `/` - Ubuntu root filesystem
- **PYNQ image (https://www.pynq.io)**
  - Python support for FPGA programming
  - Overlay driver for partial dynamic reconfiguration (PDR)
  - `peek` / `poke` Linux commands for memory-mapped I/O
- **Setup tailscale (https://tailscale.com) for remote access**

Board Checkout

- **One board per team**
  - We have 24 boards, 3 students per team
  - Pick up board from TA and sign contract
- **Board rules**
  - You will be responsible for board replacement if broken
    - These boards run $250
  - No food or drinks anywhere near the board (liquid damage)
  - Maintain ESD rules (grounding)
  - Be very careful with any commands run as root
    - Deleting files (you are responsible for your own backups)
    - Accessing memory-mapped locations can easily brick the board