

Course Intro



open-source

/əʊp(ə)n'sɔ:s/

adjective **COMPUTING**

denoting software for which the original source code is made freely available and may be redistributed and modified.

Chip Design

Electronic Design Automation (EDA):
Software flows used in chip design

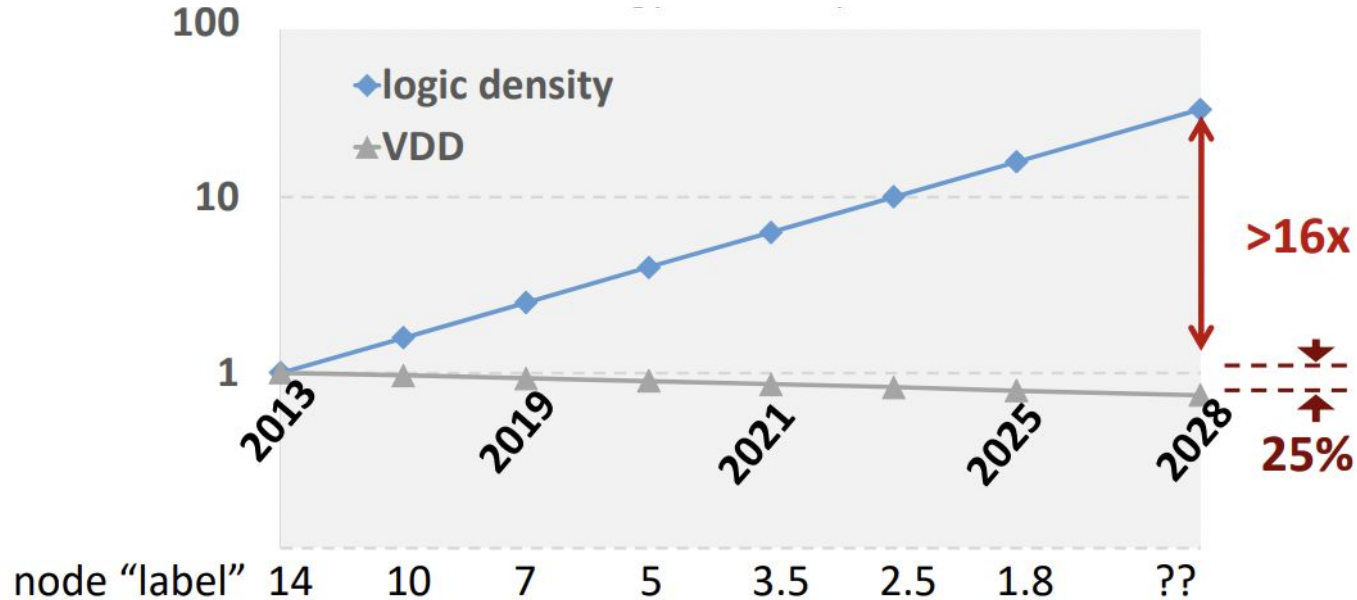
Why custom chip design?

Why should I care about custom hardware?



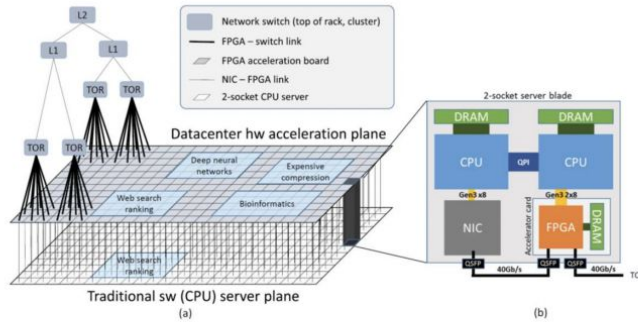
- End of Moore's Law and Dennard Scaling
 - Can't just keep making processors bigger, need to focus on specialization
 - More importance to power-efficiency these days, especially with edge-computing
- Software people: "Today's cloud is tomorrow's edge"
 - Hardware needs to keep up

End of Dennard Scaling



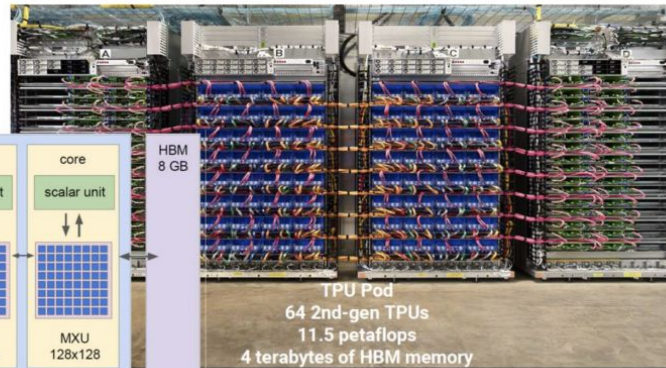
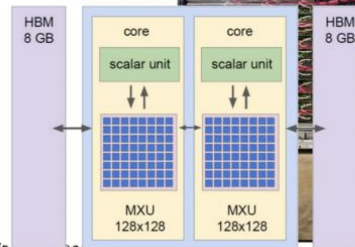
**Under fixed power ceiling, more ops/second
only achievable if less Joules/op?**

Custom Architectures



Microsoft Catapult
[MICRO 2016,
Caulfield, et al.]

Google TPU
[Hotchips, 2017,
Jeff Dean]



Custom Architectures: Big-Little + NPU

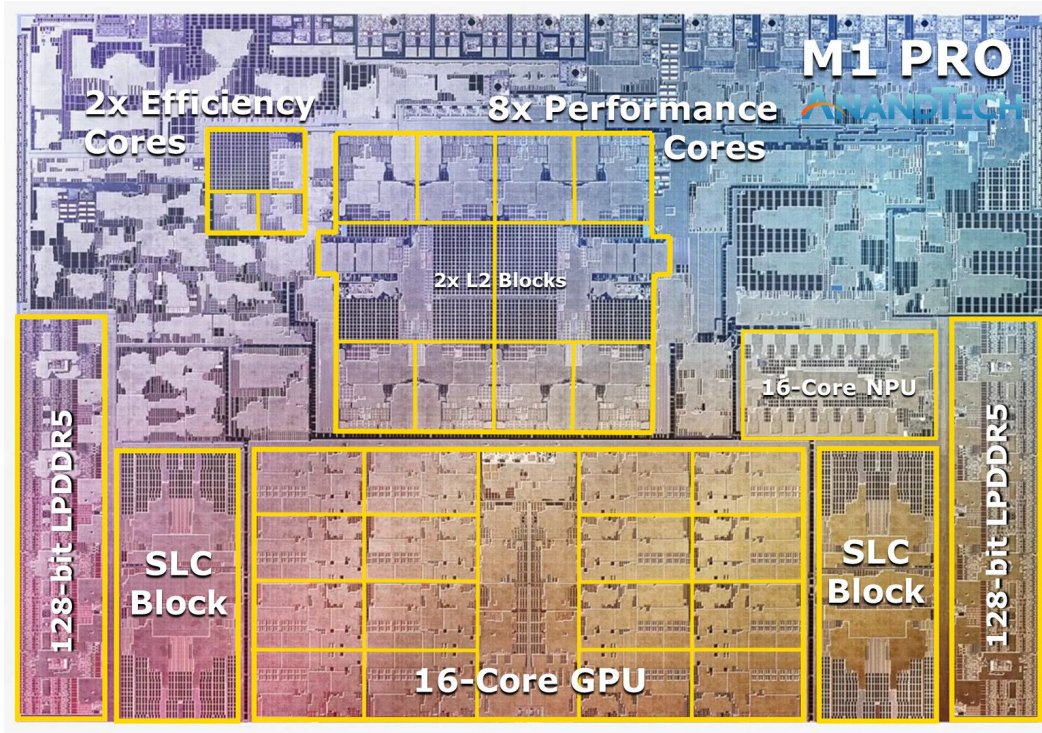
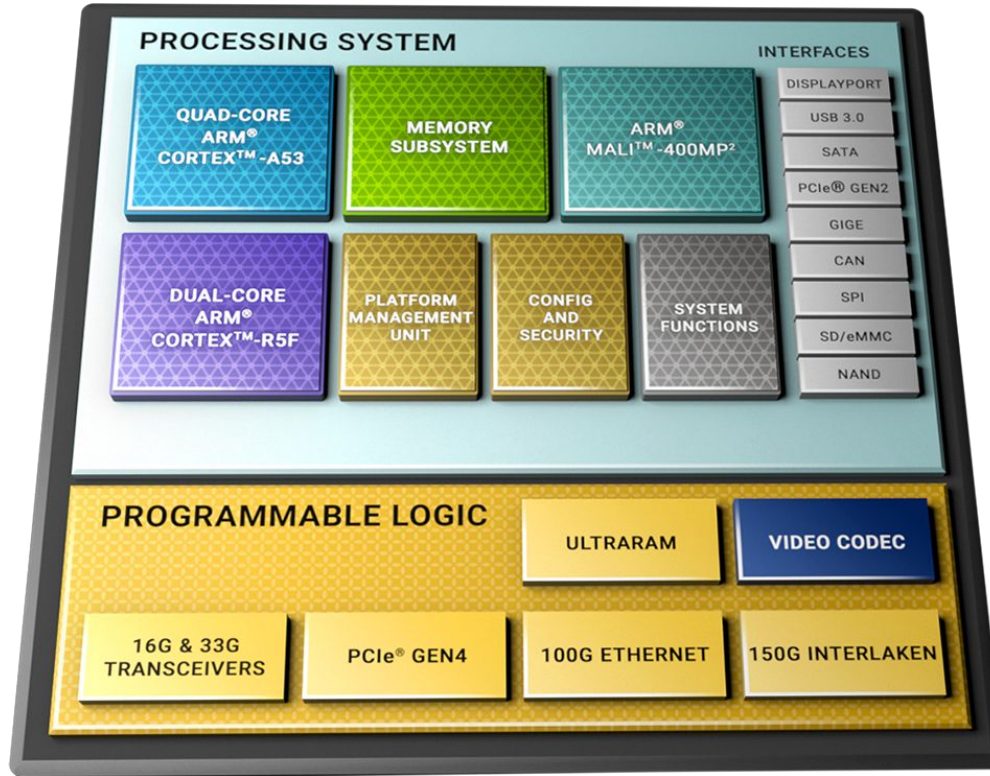


image credit: AnandTech

Custom Architectures: Even inside FPGAs



Future of Hardware is All about Efficiency

Performance/Watt and Ops/Joule

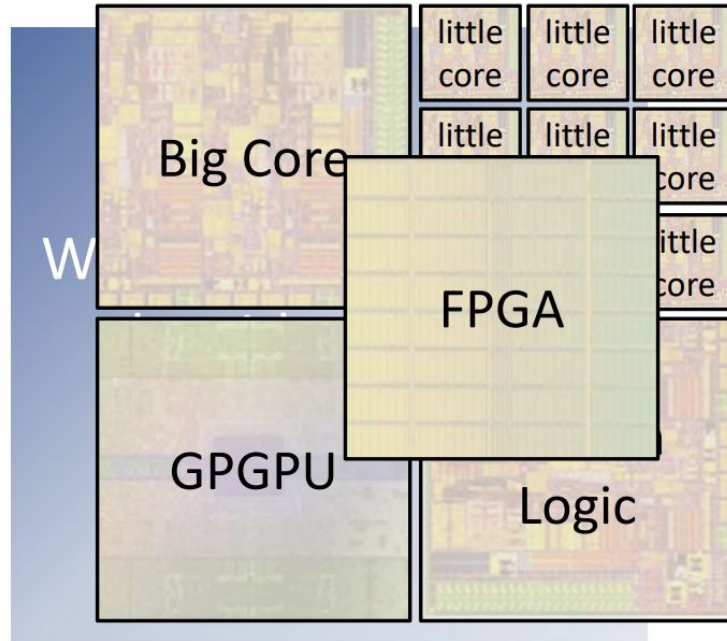


image credit: James C. Hoe

Great CompArch classes here at CMU:

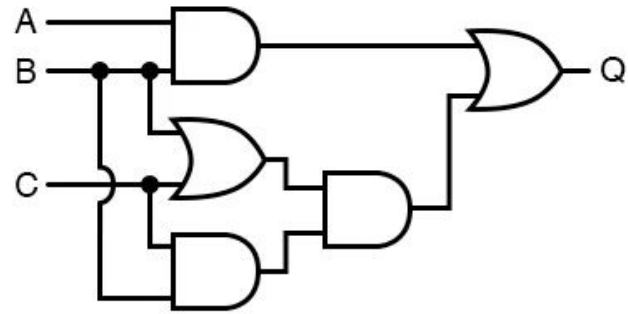
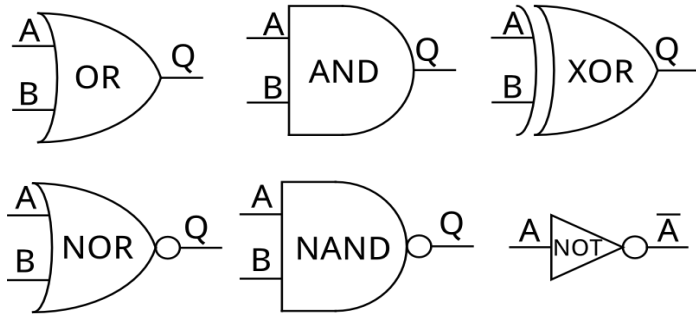
- 18-344
- 18-447
- 18-643
- 18-740

Background on Chip Design

**How can we actually build
specialized hardware?**

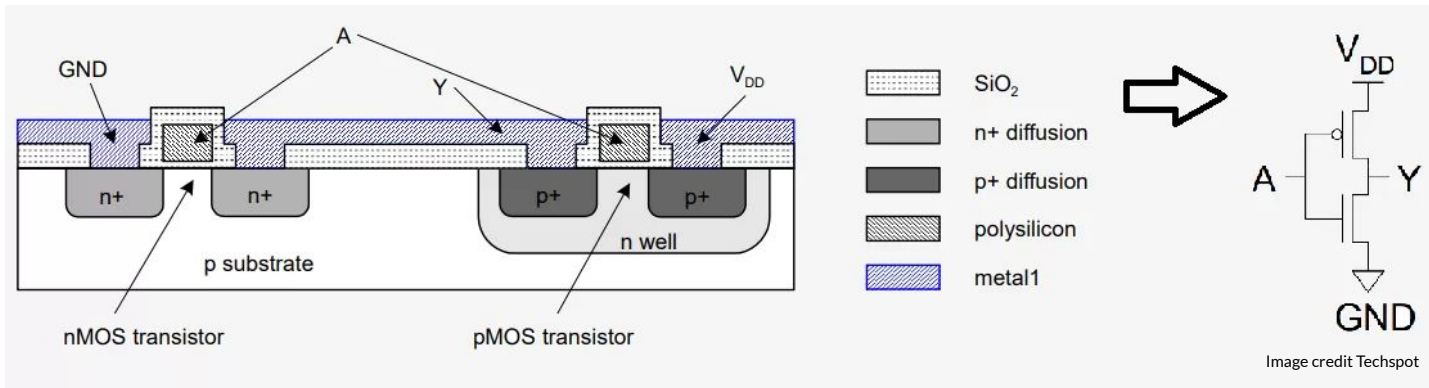
Digital Chip Design

- Digital chip design: everything starts at the level of digital logic
- From there, different ways to “implement” it into hardware
 - Discrete Logic (think 18-100 breadboard logic)
 - Field-Programmable Gate Array (FPGA)
 - Application Specific Integrated Circuit (ASIC)



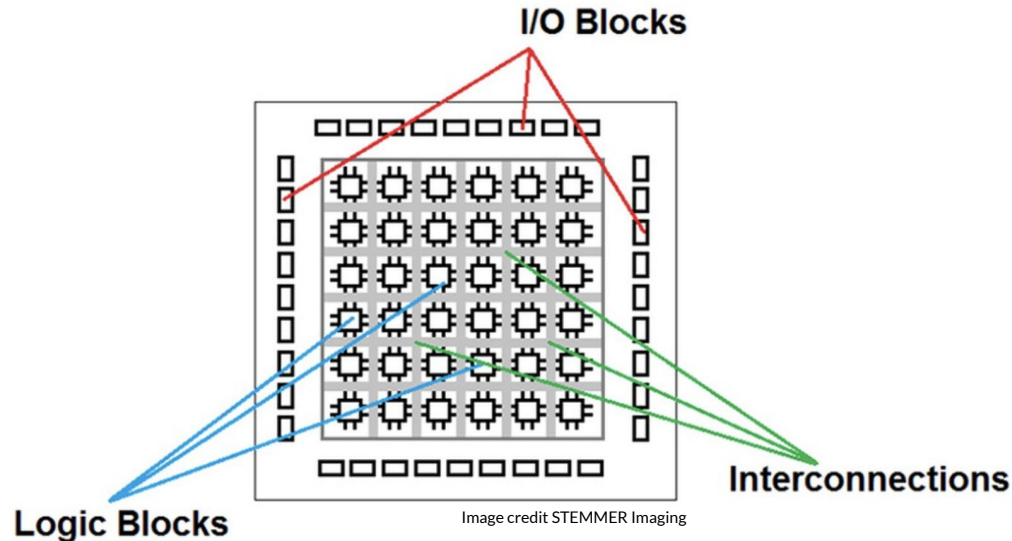
ASIC

- Application-Specific Integrated Circuit
- We will use the term ASIC or “chip” to refer to any kind of custom integrated circuit
 - Software people will usually break this down further into CPU, GPU, TPU, DSP, specialized accelerator, etc etc.



FPGA

- Field-Programmable Gate Array — consists of blobs of logic which can be reconfigured in how they are connected together



FPGA vs Custom ASIC



| Criterion | FPGA | Custom ASIC |
|----------------------------|------------------------------|-------------------|
| Cost (in small quantities) | \$5 – \$5K | \$10K – \$1M |
| Cost (in large quantities) | \$5 – \$5K | ~\$0.10 – \$1K |
| Iteration Time | few minutes – few hours | 2 – 6 months |
| Logic Density | Low – Medium | High |
| Operation Frequency | tens – hundreds of Megahertz | several Gigahertz |
| Power Efficiency | Poor | Good |
| Flexibility | ? | ? |

Uses of FPGAs



- Prototyping and emulation before fabricating a chip
- Video-processing, test equipment, Software-Defined Radio
- Embedded (power-efficient) machine learning
- Virtual-reality, augmented-reality headsets
- High-frequency trading (financial markets)
- Digital-signal processing (cars, robots, satellites)

Why Open-Source?



- Strong communities form around large open-source projects
 - Bugs get fixed way more quickly and easily
- Ability to modify to add features for your desired use-case
 - Upstream your improvements back to the community
- Significantly-reduced barrier to entry (because tools are free)
- Consistency, portability across platforms
- Huge growth over the past few years
- Less control from large corporations

Major players in the FPGA/ASIC world



- Xilinx (owned by AMD) — Makes a variety of medium-large sized FPGAs and development boards
- Intel-Altera — Makes a variety of (mostly-large) FPGAs, mostly focused on datacenter applications
- Lattice — Makes mostly small-medium FPGAs for embedded and machine-vision applications
- Smaller players: Gowin, QuickLogic, Efinix, etc
- Bringing ASICs to the masses: Efabless, TinyTapeout, ChipFlow

Major players in the EDA software world



- Synopsys — Makes synthesis, design, simulation, and verification tools (such as VCS)
- Cadence — Also makes synthesis, design, simulation, verification, and PCB design tools (such as Xcelium)
- Siemens (formerly Mentor) - best known for ModelSim
- YosysHQ - founded in early 2021 by those who developed most of the big open-source EDA tooling (formerly Symbiotic EDA)

Parts of a Hardware Design Flow

Logic Design (using an HDL)




- **Big Names:** Verilog, SystemVerilog, VHDL
- Languages like Chisel, Spinal, Migen, Amaranth are more abstract and “generative”
- PipelineC and DFiantHDL aim to be more dataflow-oriented
- People used to design logic out of logic gates by hand
- Nowadays, write “code” in Hardware Description Languages and synthesize it
- Lots of fragmentation in feature support among tools
- “High Level Synthesis” tools aim to take imperative code and convert it to logic, not very practical yet

Templated SoC Generation




- FuseSoC (Verilog/Python)
 - LiteX (Migen/Python)
 - Rocket Chip/Chipyard (Chisel)
 - SiFive Core Designer (Chisel)
 - Xilinx Vivado Block Editor
- Often times you want “standard” components in your design like a memory bus, I/O interface, or even a small CPU
 - SoC generators allow you to configure desired peripherals in a high-level language and then automatically combine them with your logic design at compile-time
 - Often built on top of generative HDL languages

Simulation & Testbenching



- Icarus Verilog
 - Verilator (compiles to C++)
 - CocoTB (testbench in Python)
 - CXXRTL (based on Yosys)
 - GHDL (for VHDL)
 - Built-in simulators in Chisel, Amaranth, etc.
- Because it is hard to debug in hardware, we simulate the logic designs with realistic inputs to check for bugs
 - Open-source simulation tools for SystemVerilog somewhat lacking, but improving rapidly (Verilator, etc.)
 - Writing testbenches is just like writing software, so it can make sense to use a language like Python or C++ for testbenching

Synthesis, Place & Route (FPGA)



- Yosys + NextPNR
 - Yosys + SymbiFlow/F4PGA
 - Verilog-to-Routing
 - Xilinx Vivado
 - Intel Quartus
 - Lattice Diamond
- Synthesis = going from code written in an RTL like Verilog to a list of LUTs, flops, and the other primitives of an FPGA
 - Place-and-Route = finding an efficient way to choose onto which specific primitives to map each piece of logic
 - PnR is usually the longest step, it has to optimize until the logic meets speed requirements

Synthesis, Place & Route (ASIC)



- Yosys + OpenLane/OpenROAD
 - Yosys + Qflow (outdated)
 - Yosys + Alliance/Coriolis
 - Magic VLSI (analog-focus)
 - Cadence toolchain
 - Synopsys toolchain
- Fabs are hesitant to release info about process nodes without NDA, which hinders open-source progress
 - Major milestone in late 2020 with open release of 130nm Skywater process node and Google-sponsored tapeouts
 - **2 more open-source process nodes released in mid 2022**
 - Could see a massive change in the way chips are designed, if this trend continues

Other Interesting Things



- Formal Verification (SymbiYosys) — Rigorously prove certain properties about your design
- Coverage Analysis (MCY) — Determine whether your test cases are sufficiently representative
- RISC-V — Open-source CPU instruction set
- Self-reproducing processors — make an FPGA program itself by running Yosys on a CPU running on the FPGA
- Waveforms, VCD files, and advanced debugging tools