CIS 5710
Computer Organization & Design

Unit 1: Introduction

Slides developed by Joseph Devietti, Benedict Brown, C.J. Taylor, Milo Martin & Amir Roth
at the University of Pennsylvania
with sources that included University of Wisconsin slides
by Mark Hill, Guri Sohi, Jim Smith, and David Wood.

Today’s Agenda

- Course logistics
- A brief history of computer architecture
- Lab Demo

Course Logistics

Course Staff

- Instructor
  - Joe Devietti
- TAs
  - Aaraddhya Thakore
  - Sumiran Bhasin
  - Shashank Khemka
  - Arjun Makwana
  - Pranoti Dhamal
  - Zeyu Jason
  - Nathan Sobotka
  - Sara Strenger
  - Audrey Yang
  - Priya Shah

Course Number

- 371/501 => 471/571 => 4710/5710
- No difference in content between 4710 and 5710
  - I'll just say "5710" for short

PhD students: WPE-1 exam

- CIS 5710 is a “coursework” WPE-1 course
  - must obtain a sufficiently high course grade
  - declare your WPE1 status with Britton

Course Resources

- Course website
  - http://www.cis.upenn.edu/~cis5710
  - slides (publicly accessible)
- Github
  - documentation and starter code for the labs
- Canvas
  - lecture videos available via **Class Recordings** tab **by request**
- Gradescope
  - submit labs
- ~~Piazza~~ **Ed Discussions**
  - private and anonymous-to-students posts/questions
  - As a general rule, **don't use email**
- Office Hours
  - in-person and Zoom
  - Google Calendar with OH time slots and links

Course Resources

- biglab.seas.upenn.edu
  - build & run your lab code
- Textbooks (none required)
  - Patterson & Hennessy, *Computer Organization and Design,* **4th or 5th edition**
    - free via Penn library, link on course homepage
  - Reese & Thornton, *Intro to Logic Synthesis using Verilog HDL*
    - PDF on Canvas in the **Files** tab
  - *Digital Design: A Systems Approach* by Dally and Harting

The Verilog Labs

- “Build your own processor” (pipelined 16-bit CPU for LC4)
- Use Verilog HDL (hardware description language)
  - Programming language compiles to gates/wires not insns
- Implement and test on real hardware
  - FPGA (field-programmable gate array)

+ Instructive: learn by doing

+ Satisfying: “look, I built my own processor”

Lab Logistics

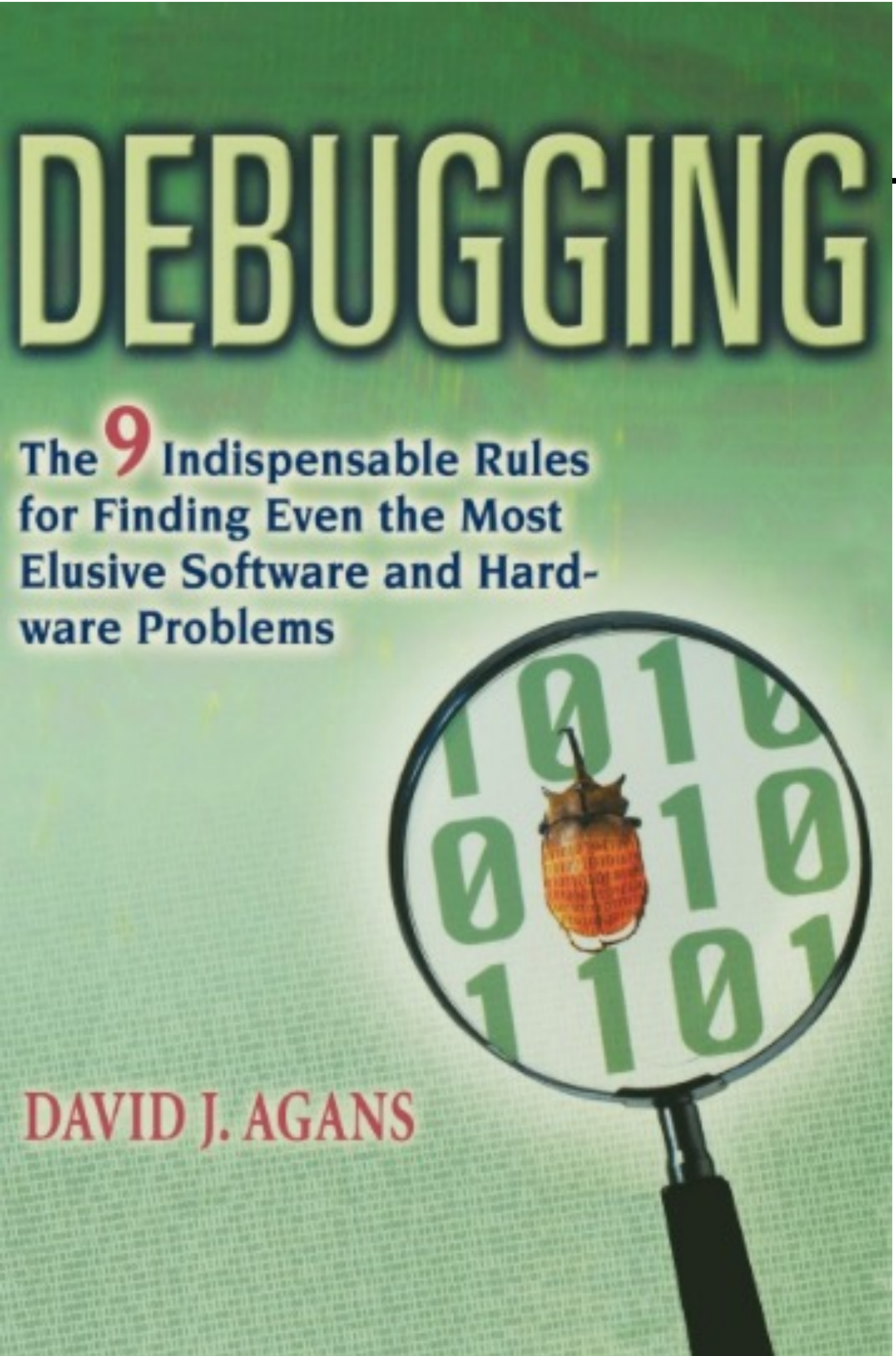
- Tools
  - Icarus Verilog, Xilinx Vivado hardware compilers
    - Run from biglab.seas.upenn.edu
  - ZedBoard FPGA boards
    - Live in Towne lockers, details coming
- ZedBoard demos
  - Most labs have an **optional** demo component that runs on the ZedBoard
  - Development and simulation should be done before final testing on the board

Labs

- **Labs 2-5 done in groups of two**
  - start forming groups now
  - join an **existing** Canvas group with your partner
- Lab descriptions
  - Lab 1: Verilog debugging
  - Lab 2: arithmetic & logical unit
  - Lab 3: single-cycle LC4 & register file
  - Lab 4: pipelined LC4
  - Lab 5: pipelined +superscalar LC4
- Labs are **cumulative** and increasingly complex
- Each lab broken down into "milestone" deadlines
  - Roughly one per week
- **Threshold grading**: grades are [0%,90%] or 100%

Midterm & Final Exam

- In-class midterm exam
  - Monday 27 February
- Final exam
  - Date, time & location TBD


DEBUGGING
The 9 Indispensable Rules for Finding Even the Most Elusive Software and Hardware Problems
DAVID J. AGANS

In many ways this is a class about **debugging**

http://debuggingrules.com

Grading

- Tentative grade contributions:
  - Midterm: 15%
  - Final: 25%
  - Labs: 60%
- Historical grade distribution: median grade is B+

Late Policy

- Assignments usually due Friday at 6:00pm US Eastern time. Deadline is enforced by Gradescope.
- Submit as often as you like; your **most recent** submission is what counts.
- **Any** assignment can be submitted **up to 48 hours late, for 75% credit**
    - No need to give an excuse, just turn it in late
    - Assignments are cumulative - you have to get things to work!

Academic Misconduct

- Cheating will **not** be tolerated

- General rule:
  - Anything with your name on it must be **YOUR OWN** work
  - You **MUST** scrupulously credit all sources of help

- See the course website for additional policies

- Penn’s Code of Conduct
  - http://www.vpul.upenn.edu/osl/acadint.html

Computer Architecture Courses @ Penn

CIS 5710 vs CIS 2400

- In CIS 2400 you learned how a processor worked, in CIS 5710 we will tell you how to make it work **well**.

CIS 2400

CIS 5710

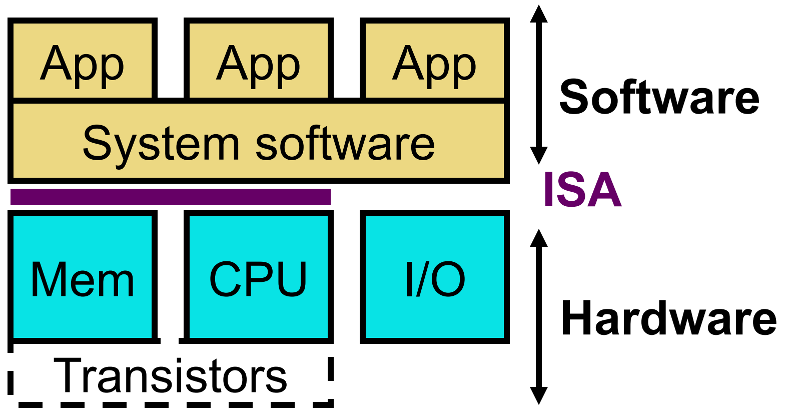
CIS 2400 vs. CIS 5710

- CIS 2400
  - Focus on one toy ISA: LC4
  - Focus on functionality: “just get something that works”
  - Instructive, learn to crawl before you can walk
  - Not representative of real machines: 240 hardware is circa 1975
- CIS 5710
  - Less emphasis on any particular ISA during lectures
  - Focus on quantitative aspects: **performance**, cost, power, etc.
  - Representative of ~1980s hardware
    - also modern low-power processors, e.g., inside a Fitbit

Pervasive Idea: Abstraction and Layering

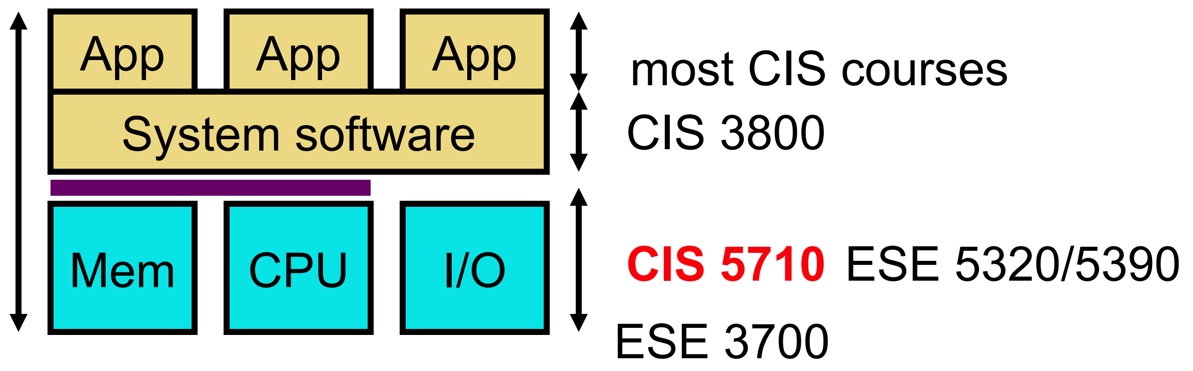
- **Abstraction**: only way of dealing with complex systems
  - Divide world into objects, each with an…
    - **Interface**: knobs, behaviors, knobs $\rightarrow$ behaviors
    - **Implementation**: “black box” (ignorance+apathy)
  - Only specialists deal with implementation, rest of us with interface
  - Example: only mechanics know how cars work
- **Layering**: abstraction discipline makes life even simpler
  - Divide objects in system into layers, layer n objects…
    - Implemented using interfaces of layer $n - 1$
    - (mostly) Don’t need to know interfaces of layer $n - 2$
- **Inertia**: a dark side of layering
  - Layer interfaces become entrenched over time (“standards”)
  - Very difficult to change even if benefit is clear
- **Opacity**: hard to reason about performance across layers

Abstraction, Layering, and Computers



- Computers are complex, built in layers
  - Several **software** layers: assembler, compiler, OS, applications
  - **Instruction set architecture (ISA)**
  - Several **hardware** layers: transistors, gates, CPU/Memory/IO
- 99% of users don't know hardware layers implementation
- 90% of users don't know implementation of any layer
  - That's okay, world still works just fine
  - But sometimes it is helpful to understand what's "under the hood"

Beyond CIS 5710



- CIS 3800: Operating Systems
  - A closer look at system level software
- **CIS 5710: Computer Organization and Design**
  - **A closer look at hardware layers**
- ESE 3700: Circuit-Level Modeling, Design, and Optimization for Digital Systems
  - Diving into gate-level abstractions
- ESE 5320/5390: Accelerator Design
  - HW+SW: design an application-specific hardware accelerator

A Brief History of Computer Architecture

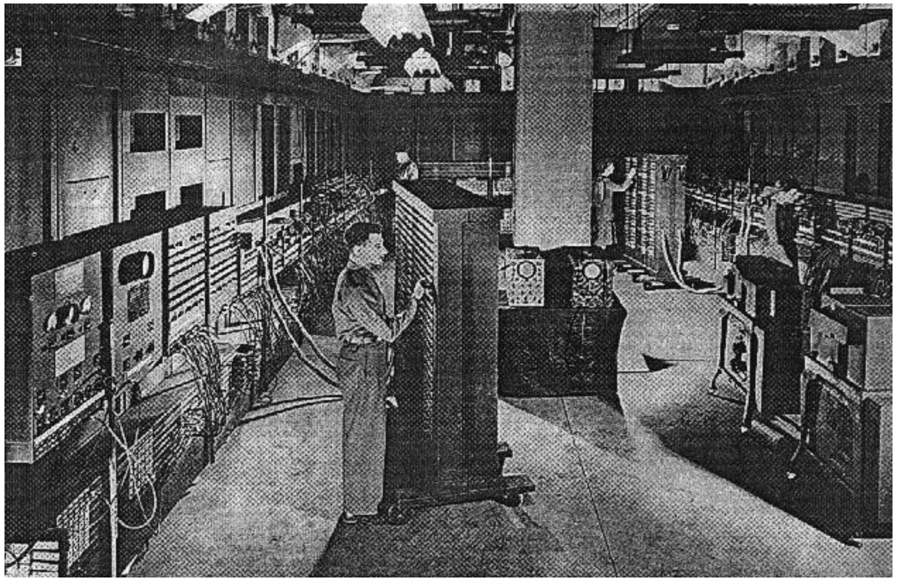
Why Study Hardware?

- **Understand where computers are going**
  - Future capabilities drive the (computing) world
  - Real world-impact: no computer architecture → no computers!
- **Understand high-level design concepts**
  - The best system designers understand all the levels
    - Hardware, compiler, operating system, applications
- **Understand computer performance**
  - Writing well-tuned (fast) software requires knowledge of hardware
- **Write better software**
  - The best software designers also understand hardware
  - Understand the underlying hardware and its limitations
- **Design hardware**
  - Intel, AMD, IBM, ARM, Qualcomm, Apple, Oracle, NVIDIA, Samsung, …

Penn Legacy

- **ENIAC**: electronic numerical integrator and calculator
  - First operational general-purpose stored-program computer
  - Designed and built here by Eckert and Mauchly
  - See it in Moore 100!

- **First seminars on computer design**
  - Moore School Lectures, 1946
  - "*Theory and Techniques for Design of Electronic Digital Computers*"



Computer Architecture

- **Computer architecture**
  - Definition of ISA to facilitate implementation of software layers
  - The hardware/software interface
- **Computer micro-architecture**
  - Design processor, memory, I/O to implement ISA
  - Efficiently implementing the interface
- CIS 5710 is mostly about processor micro-architecture
  - "architecture" is also a vacuous term for "the design of things"
    - software architect, network architect, …

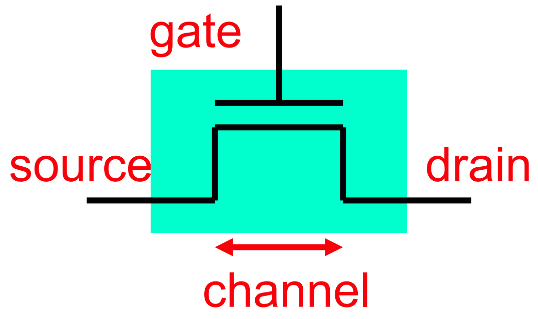
Application Specific Designs

- This class is about **general-purpose CPUs**
  - Processor that can do anything, run a full OS, etc.
  - E.g., Intel Atom/Core/Xeon, AMD Ryzen/EPYC, ARM M/A series
- In contrast to **application-specific chips**
  - Or **ASICs** (Application specific integrated circuits)
    - Also application-domain specific processors
  - Implement critical domain-specific functionality in hardware
    - Examples: video encoding, 3D graphics, machine learning
  - General rules
    - - Hardware is less flexible than software
    - + Hardware more effective (speed, power, cost) than software
    - + Domain specific more "parallel" than general purpose
      - But mainstream processors are quite parallel as well

Technology Trends

Technology that Drives

- Basic element
  - Solid-state **transistor** (i.e., electrical switch)
  - Building block of **integrated circuits (ICs)**

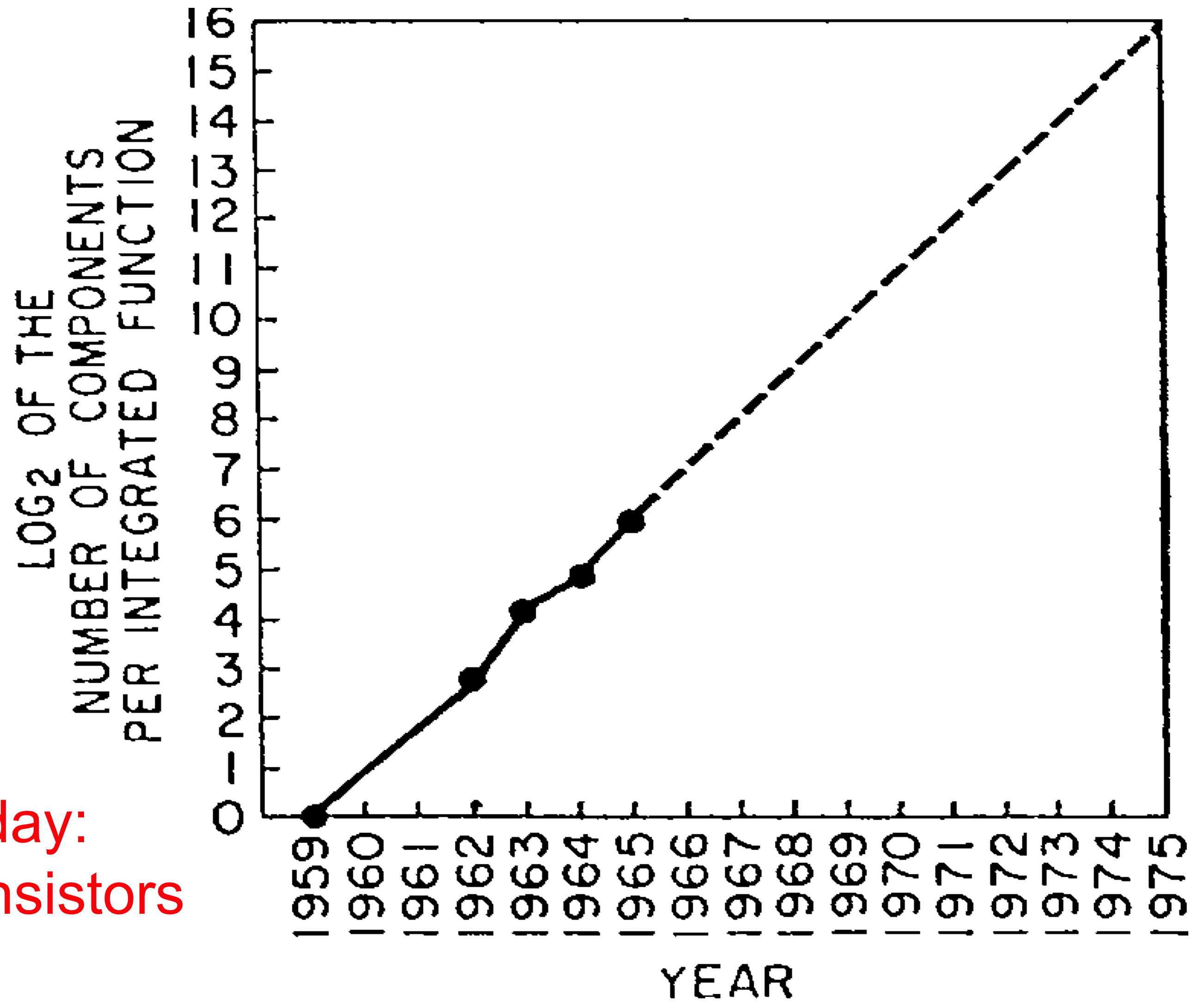


- What's so great about ICs? Everything
  - + High performance, high reliability, low cost, low power
  - + Lever of mass production

- Several kinds of integrated circuit families
  - **SRAM/logic**: optimized for speed (used for processors)
  - **DRAM**: optimized for density, cost, power (used for memory)
  - **Flash**: optimized for density, cost (used for storage)
  - Increasing opportunities for integrating multiple technologies

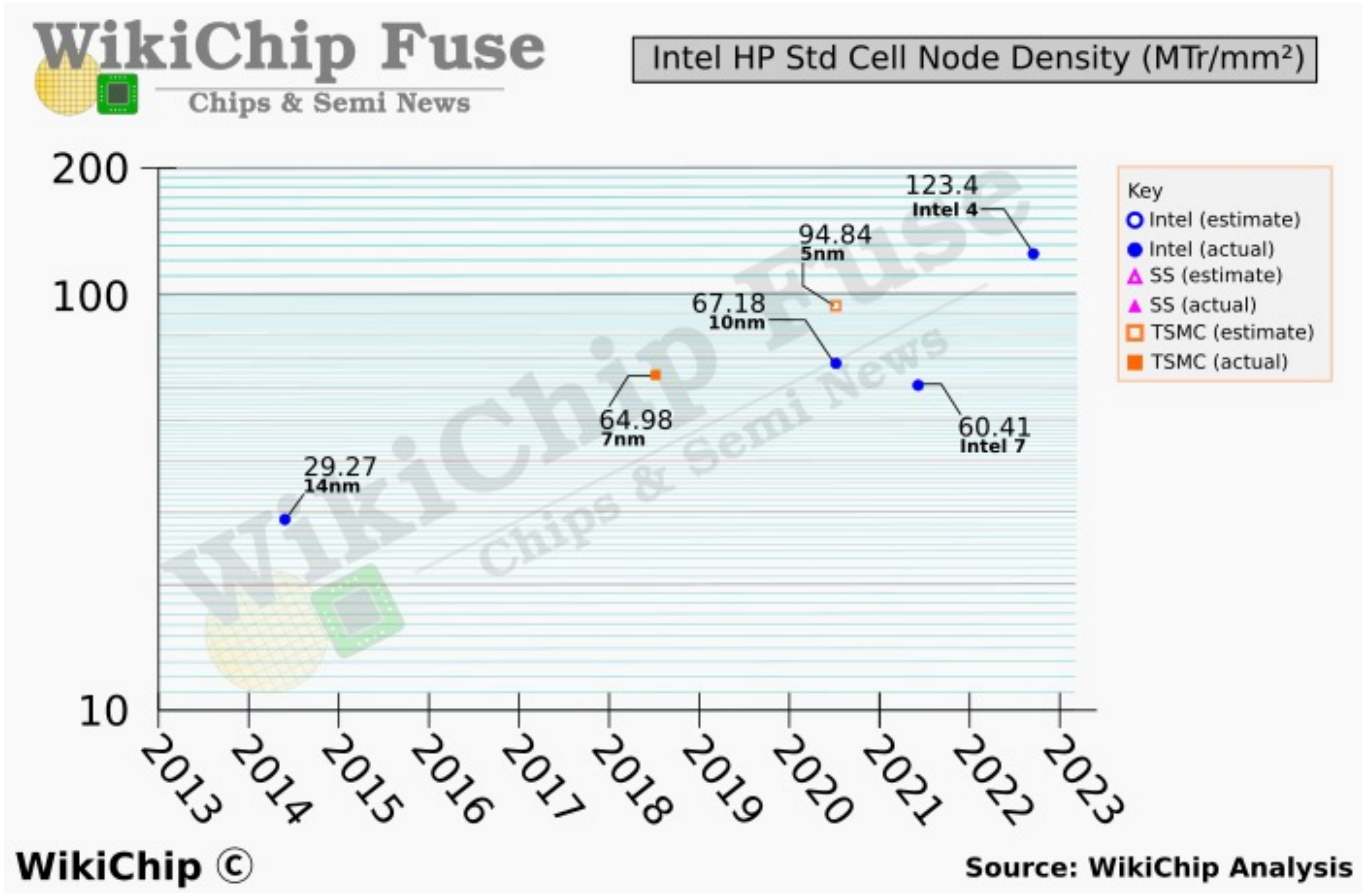
- Non-transistor storage and inter-connection technologies
  - Magnetic disks, optical storage, ethernet, fiber optics, wireless

Moore’s Law - 1965



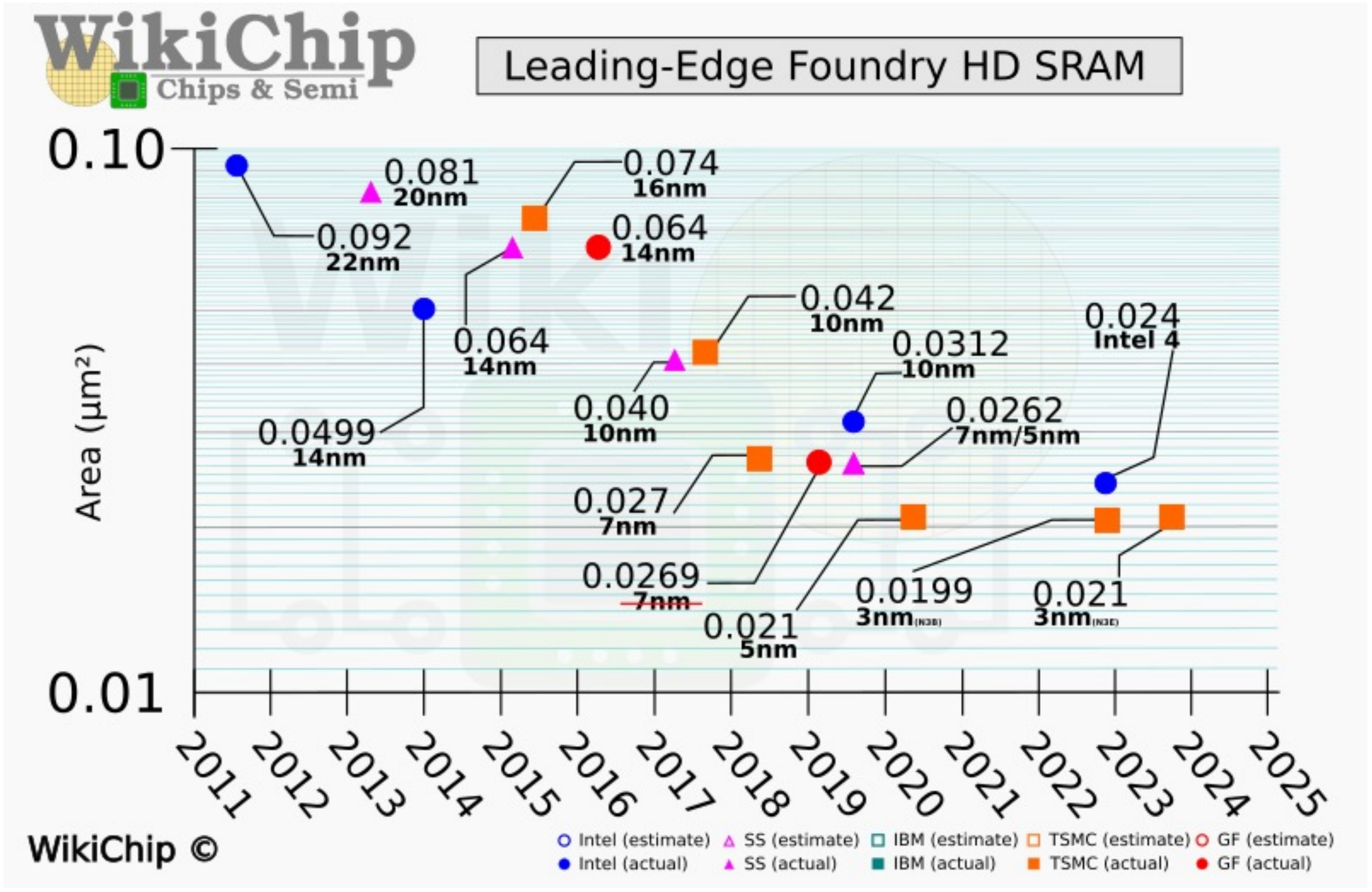
Today:
2^{35} transistors

Moore’s Law today



data c/o WikiChip

Moore's Law today



data c/o WikiChip

Revolution I: The Microprocessor

- **Microprocessor revolution**
  - One significant technology threshold was crossed in 1970s
  - Enough transistors (~25K) to put a 16-bit processor on one chip
  - Huge performance advantages: fewer slow chip-crossings
  - Even bigger cost advantages: one "stamped-out" component

- Microprocessors have allowed new market segments
  - Desktops, CD/DVD players, laptops, game consoles, set-top boxes, mobile phones, digital camera, mp3 players, GPS, automotive

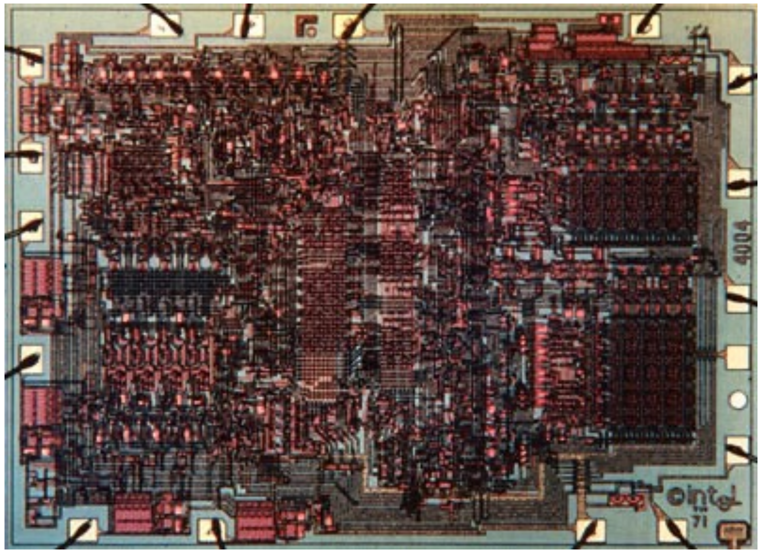
- And replaced incumbents in existing segments
  - Microprocessor-based system replaced supercomputers, "mainframes", "minicomputers", "desktops", etc.

First Microprocessor

- Intel 4004 (1971)
  - Application: calculators
  - Technology: 10,000 nm

  - 2300 transistors
  - 13 mm^2
  - 108 KHz
  - 12 Volts

  - 4-bit data
  - Single-cycle datapath



Revolution II: Implicit Parallelism

- Then to **extract implicit instruction-level parallelism**
  - Hardware provides parallel resources, figures out how to use them
  - Software is oblivious

- Initially using pipelining …
  - Which also enabled increased clock frequency
- … caches …
  - Which became necessary as processor clock frequency increased
- … and integrated floating-point
- Then deeper pipelines and branch speculation
- Then multiple instructions per cycle (superscalar)
- Then dynamic scheduling (out-of-order execution)

- We will build many of these things!

Pinnacle of Single-Core Microprocessors

- Intel Pentium4 (2003)
  - Application: desktop/server
  - Technology: 90nm

  - 55M transistors
  - 101 mm^2
  - 3.4 GHz
  - 1.2 Volts

  - 32/64-bit data (16x)
  - 22-stage pipelined datapath
  - 3 instructions per cycle (superscalar)
  - Two levels of on-chip cache
  - data-parallel vector (SIMD) instructions, hyperthreading

Revolution III: Explicit Parallelism

- Then to support **explicit data & thread level parallelism**
  - Hardware provides parallel resources, software specifies usage
  - Why? diminishing returns on instruction-level-parallelism

- First using (subword) vector instructions..., Intel's SSE
  - One instruction does four parallel multiplies

- ... and general support for multi-threaded programs
  - Coherent caches, hardware synchronization primitives

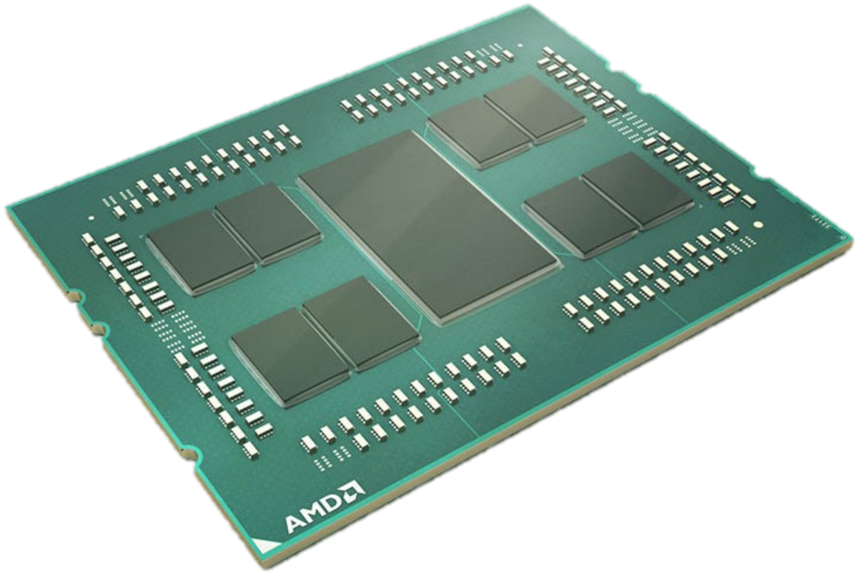
- Then using support for multiple concurrent threads on chip
  - First with single-core multi-threading, now with multi-core

- Graphics processing units (GPUs) are highly parallel

Modern Multicore Processor

- AMD EPYC 7H12
  - Application: server
  - Technology: 7nm

  - 39.5B transistors
  - 1008 mm^2
  - 2.6 to 3.3 Ghz
  - 256-bit data (2x)
  - 19-stage pipelined datapath
  - 4 instructions per cycle
  - 292MB of on-chip cache
  - data-parallel vector (SIMD) instructions, hyperthreading
  - **64-core multicore**



Historical Microprocessor Evolution

Feature	Intel 4004	Intel Pentium 4 Prescott	AMD EPYC Rome
release date	1971	2004	2019
transistor size	10,000 nm	90 nm	7 nm, 14 nm
transistor count	2,300	125M	39.5B
area	13 mm^2	112 mm^2	1008 mm^2
frequency	740 KHz	3.8 GHz	2.6-3.3 GHz
data width	4-bit	64-bit	256-bit
pipeline stages	n/a	31	19
pipeline width	n/a	3	4
core count	1	1	64
on-chip cache	n/a	1MB	292MB

Revolution IV: Accelerators

- Combining **multiple** kinds of compute engines in one die
  - not just homogenous collection of cores
  - System-on-Chip (SoC) is one common example in mobile space

- Lots of stuff on the chip beyond just CPUs
  - Graphics Processing Units (GPUs)
    - throughput-oriented specialized multicore processors
    - good for gaming, machine learning, computer vision, …
  - Special-purpose logic
    - media codecs, radios, encryption, compression, machine learning

- Excellent energy efficiency and performance
  - extremely complicated to program!

SNAPDRAGON 805 PROCESSOR

Faster performance and more multitasking with Krait 450 CPU at up to 2.7 GHz

Console quality gaming with new generation Adreno 420 GPU

More power-efficient apps and system processing with the Hexagon™ QDSP6

Capture and play back Ultra HD video and enjoy 7.1 surround sound on the go or at home with advanced video and audio engines

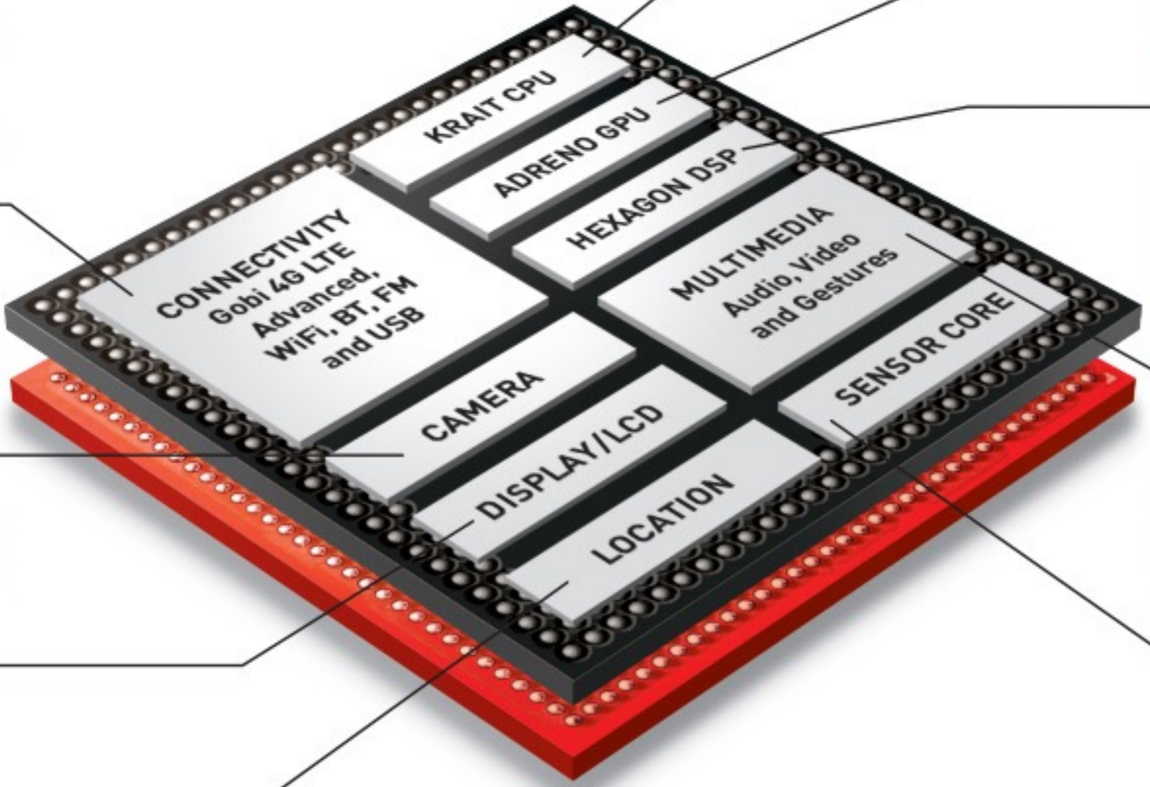
Get more use and greater accuracy from sensor-intensive apps with the dedicated Snapdragon Sensor Engine

Stay connected and stream large files fast with industry leading connectivity, including the world's most advanced 4G LTE and VIVE™ 2-stream 802.11ac Wi-Fi

Capture sharper photos, even in low light, with the mobile industry's first dual ISP

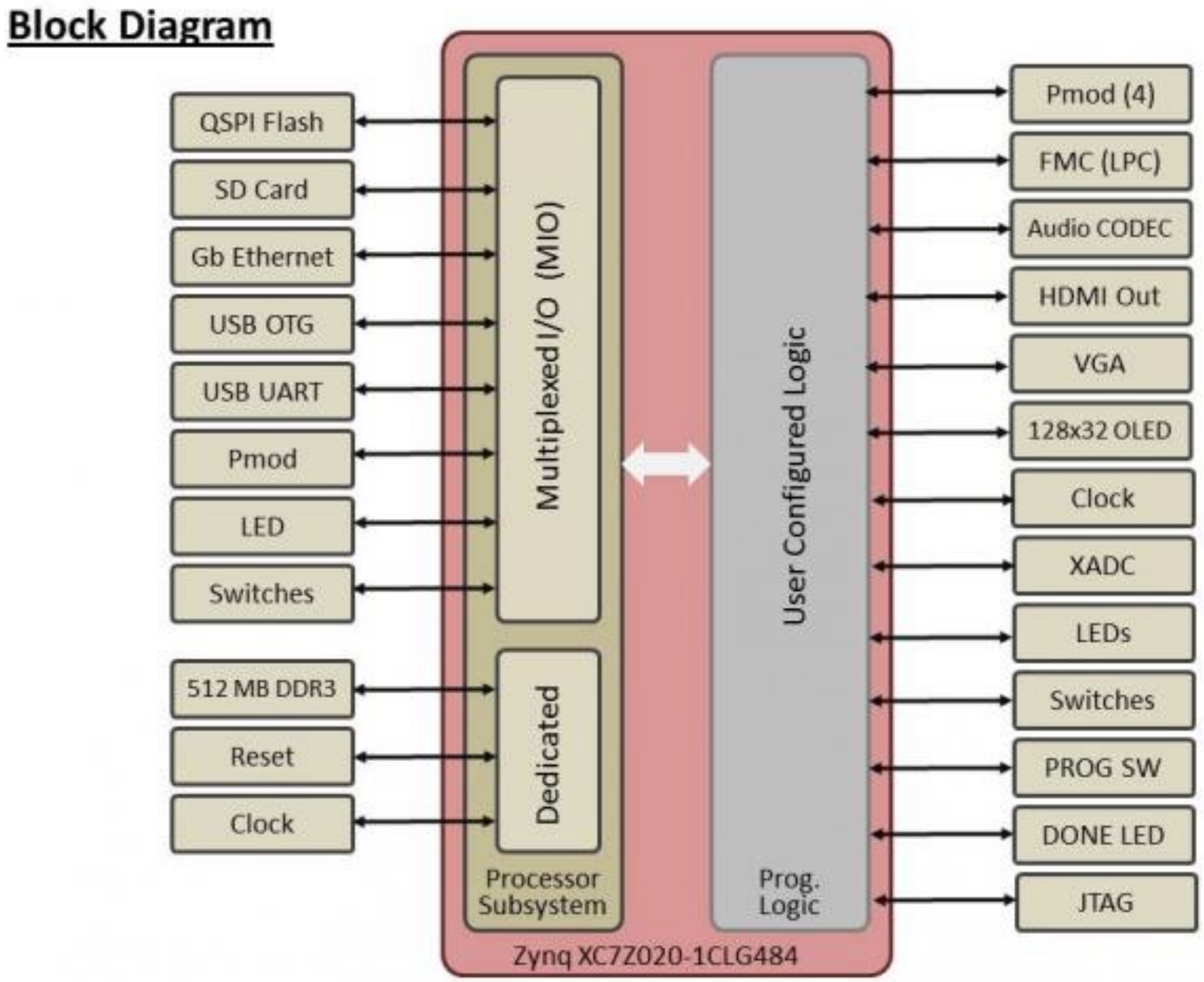
Enjoy Ultra HD resolution content on Ultra HD-capable mobile devices and Ultra HD TVs with the Snapdragon Display Processor

Find your way outdoors and indoors with IZat GNSS with support for GPS, Glonass and BeiDou constellations


KRAIT CPU
ADRENO GPU
HEXAGON DSP
MULTIMEDIA Audio, Video and Gestures
SENSOR CORE
CONNECTIVITY Gobi 4G LTE Advanced, WiFi, BT, FM and USB
CAMERA
DISPLAY/LCD
LOCATION

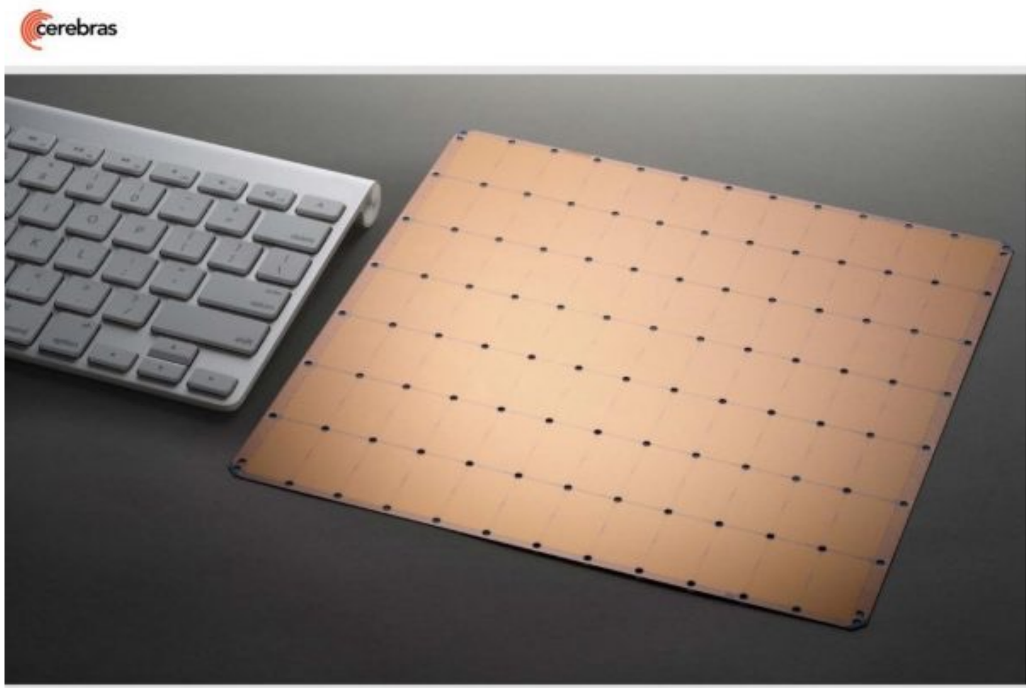
c/o Qualcomm

Our Zedboard SoC



Cerebras Wafer-Scale Engine

- giant 8.5” square chip!
- full of deep learning accelerators
- 18GB on-chip memory
- 9 PB/sec on-chip memory bandwidth
- TSMC 16nm transistors
- ~$2.5M each?



Technology Disruptions

- Classic examples:
  - transistor
  - microprocessor
- More recent examples:
  - flash-based solid-state storage
  - shift to accelerators
- Nascent disruptive technologies:
  - non-volatile memory (“disks” as fast as DRAM)
  - Chip stacking (also called “3D die stacking”)
- The end of Moore’s Law
  - “If something can’t go on forever, it must stop eventually”
  - Transistor speed/energy efficiency not improving like before

“Golden Age of Computer Architecture”

- Hennessy & Patterson, 2018 Turing Laureates
- the end of Dennard Scaling & Moore’s Law means no more free performance
  - “The next decade will see a Cambrian explosion of novel computer architectures”

Themes

- Parallelism
  - enhance system performance by doing multiple things at once
  - instruction-level parallelism, multicore, GPUs, accelerators
- Caching
  - exploiting locality of reference: storage hierarchies
  - try to provide the illusion of a single large, fast memory