ECE 6775
High-Level Digital Design Automation
Fall 2023

Course Overview

Zhiru Zhang

School of Electrical and Computer Engineering

Cornell University

CSL

Agenda

- Important logistics
- Course motivation
- More course organization

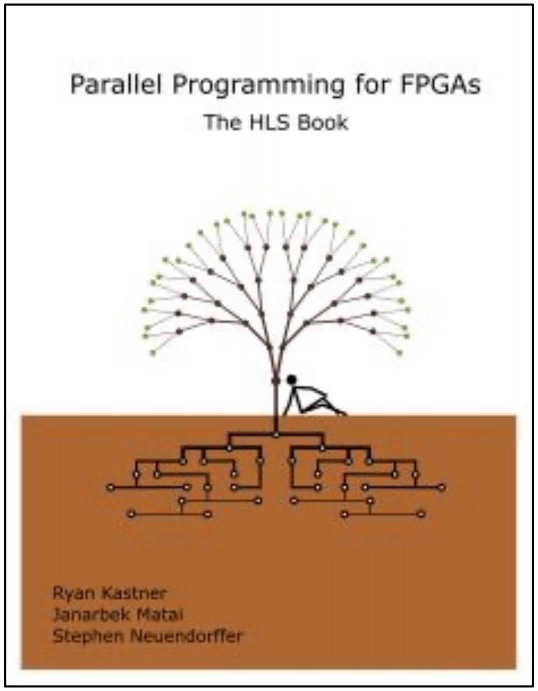
Class Resources

- Course website
  - https://www.csl.cornell.edu/courses/ece6775
  - Lectures slides, handouts, and other readings

- Ed Discussion
  - Announcements and Q&A

- CMS: course management system
  - Assignments and grades
  - Electronic submissions required

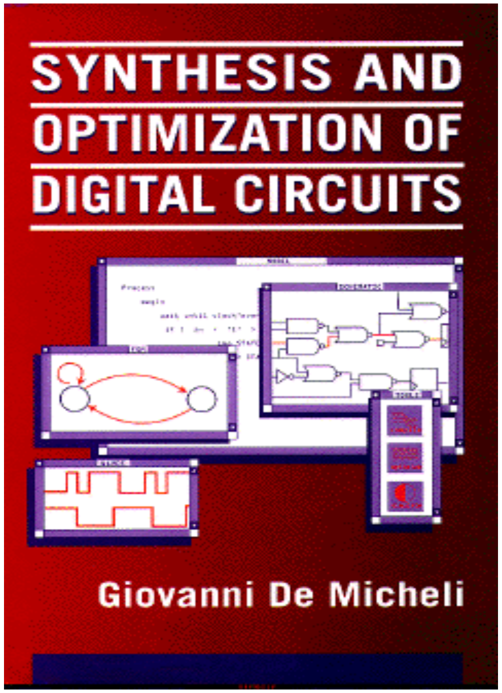
Course Texts



Parallel Programming for FPGAs
The HLS Book

Ryan Kastner
Janarbek Matai
Stephen Neuendorffer

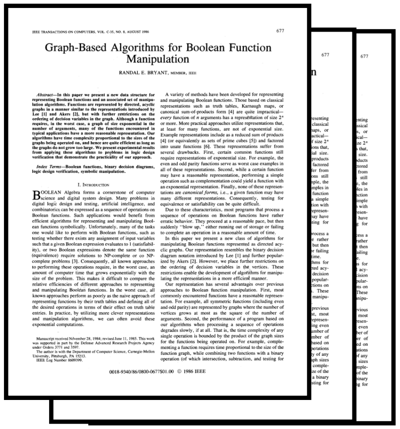
e-book available online



SYNTHESIS AND OPTIMIZATION OF DIGITAL CIRCUITS

Giovanni De Micheli

Get 1st edition
Overhead slides available online



IEEE TRANSACTIONS ON COMPUTERS, VOL. C-35, NO. 8, AUGUST 1986 677

Graph-Based Algorithms for Boolean Function Manipulation

RANDAL E. BRYANT, MEMBER, IEEE

Abstract—In this paper we present a new data structure for representing Boolean functions and an associated set of manipulation algorithms. Functions are represented by directed, acyclic graphs in a manner similar to the representations introduced by Lee [1] and Akers [2], but with further restrictions on the ordering of decision variables in the graph. Although a function requires, in the worst case, a graph of size exponential in the number of arguments, many of the functions encountered in typical applications have a more reasonable representation. Our algorithms have time complexity proportional to the sizes of the graphs being operated on, and hence are quite efficient as long as the graphs do not grow too large. We present experimental results from applying these algorithms to problems in logic design verification that demonstrate the practicality of our approach.

Index Terms—Boolean functions, binary decision diagrams, logic design verification, symbolic manipulation.

Selected papers & software manuals

Seeking Help After Class

- Ed Discussion
  - Questions on lectures, assignments, projects, etc.
  - Monitored by course staff

- Instructor office hours (online)
  - Thursday 4:30-5:30pm, Zoom link posted on Ed

- Email instructor for personal issues/appointment

- PhD TAs:
  - Jordan Dotzel (jad443), Matthew Hofmann (mrh259)

Grading Scheme

- Class participation (4%)
  - Asking & answering questions during lectures
  - Contributing to discussions on Ed
- Paper readings (5%)
- Quizzes (6%)
- Midterm exam (20%)
- Assignments (30%)
- Final project (35%)

This Course is About Hardware/Software Co-Design

- Specify applications/algorithms in software programs

- Synthesize software descriptions into special-purpose hardware architectures, namely, *accelerators*
  - Explore performance-cost trade-offs
  - Exploit automatic compilation & synthesis optimizations

- Realize the synthesized accelerators on FPGAs

This Course Introduces EDA

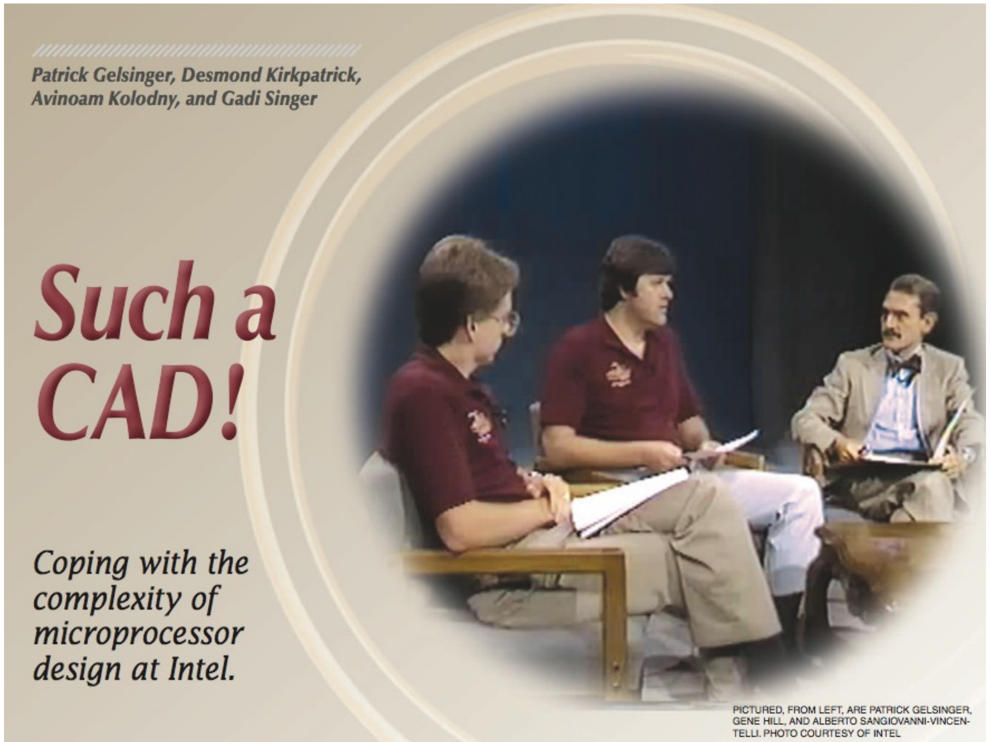
***E**lectronic **D**esign **A**utomation*

- A general methodology for refining **a high-level description down to a detailed physical implementation** for designs ranging from
  - integrated circuits (including system-on-chips),
  - printed circuit boards (PCBs),
  - and electronic systems

- **Modeling**, **synthesis**, and **verification** at every level of **abstraction**

[source: NSF'09 EDA Workshop]

Significance of EDA



Patrick Gelsinger, Desmond Kirkpatrick, Avinoam Kolodny, and Gadi Singer

Such a CAD!

Coping with the complexity of microprocessor design at Intel.

PICTURED, FROM LEFT, ARE PATRICK GELSINGER, GENE HILL, AND ALBERTO SANGIOVANNI-VINCENTELLI. PHOTO COURTESY OF INTEL

Patrick Gelsinger, Desmond Kirkpatrick, Avinoam Kolodny, and Gadi Singer. “Such a CAD!” *IEEE Solid-State Circuits Magazine*, 2010.

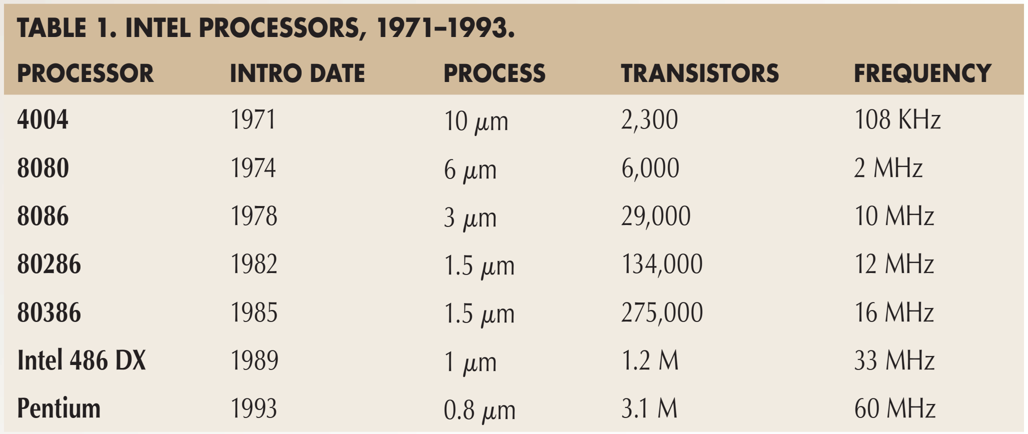


TABLE 1. INTEL PROCESSORS, 1971–1993.

PROCESSOR	INTRO DATE	PROCESS	TRANSISTORS	FREQUENCY
4004	1971	10 μm	2,300	108 KHz
8080	1974	6 μm	6,000	2 MHz
8086	1978	3 μm	29,000	10 MHz
80286	1982	1.5 μm	134,000	12 MHz
80386	1985	1.5 μm	275,000	16 MHz
Intel 486 DX	1989	1 μm	1.2 M	33 MHz
Pentium	1993	0.8 μm	3.1 M	60 MHz

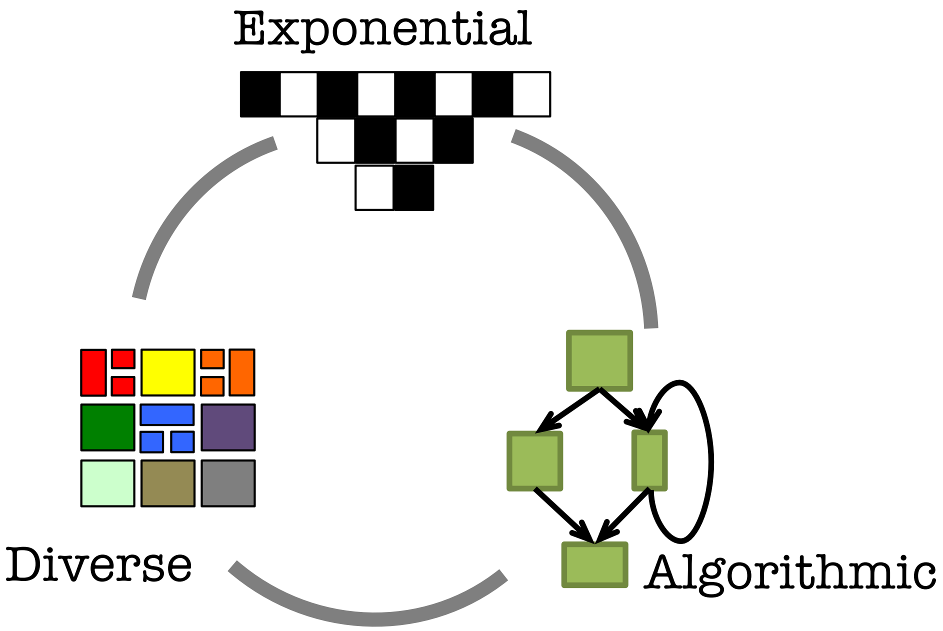
> “This incredible growth rate could not be achieved by hiring an exponentially growing number of design engineers. It was fulfilled by adopting new design methodologies and by introducing innovative design automation software at every processor generation.”

E-D-A: My Other Interpretation

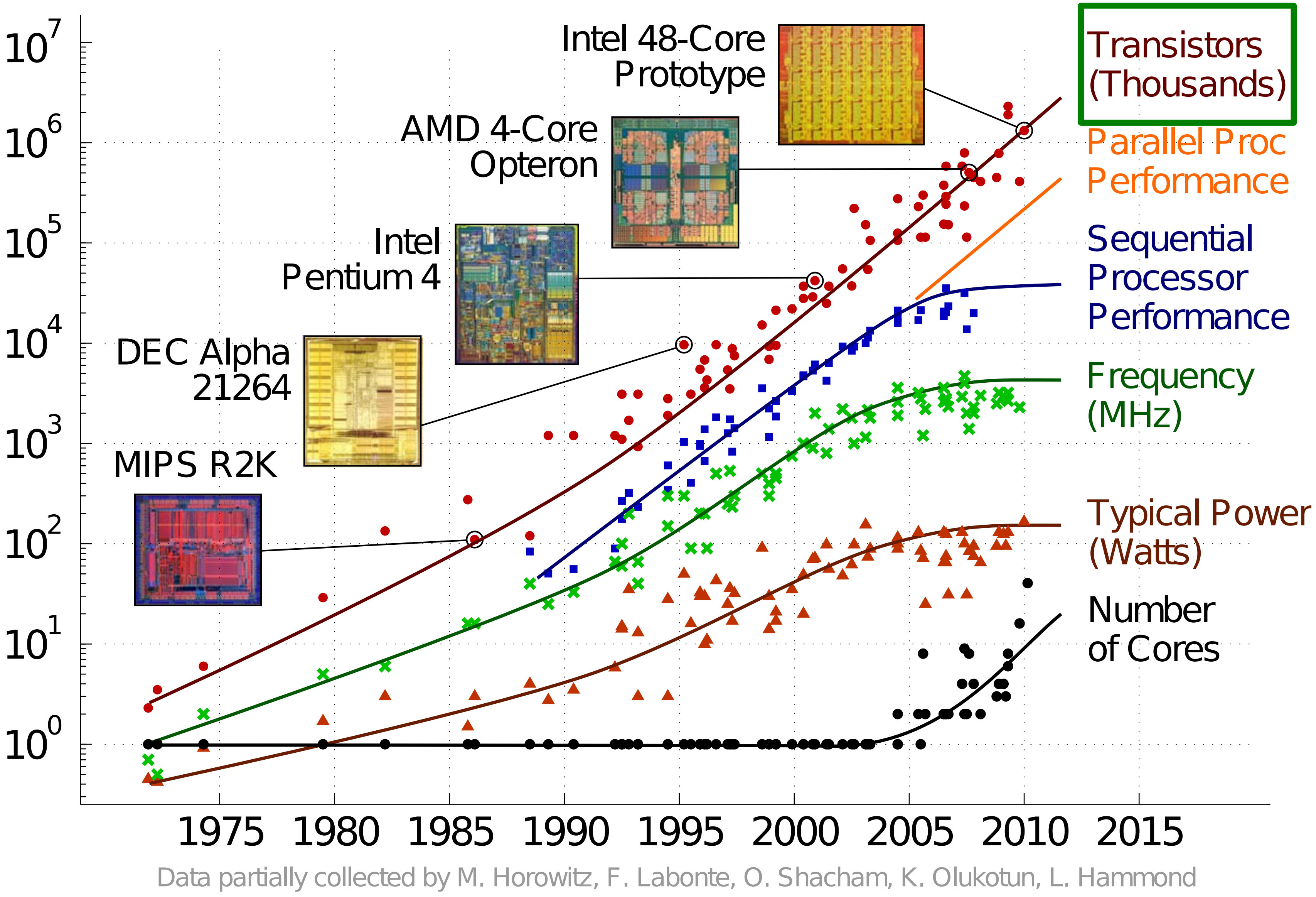
Exponential
in complexity (or **E**xtreme scale)

Diverse
increasing system heterogeneity
multi-disciplinary

Algorithmic
intrinsically computational

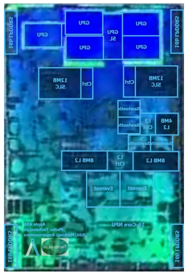


Exponential: Moore's Law

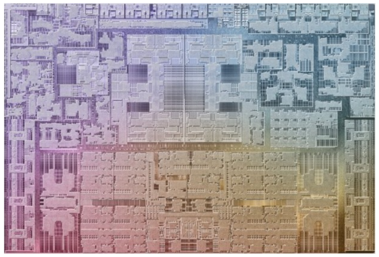


[Figure credit: Christopher Batten, Cornell]

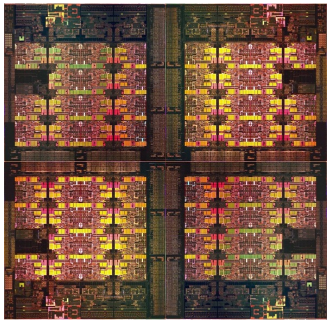
Era of Billion-Transistor Chips



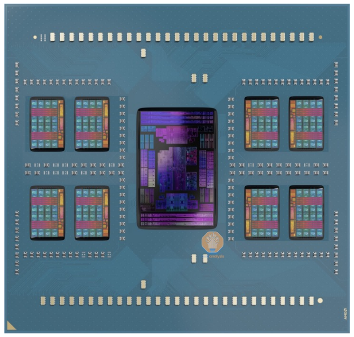
Apple A16
~16B transistors



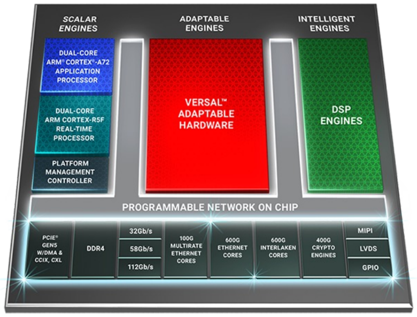
Apple M2 Pro
~40B transistors



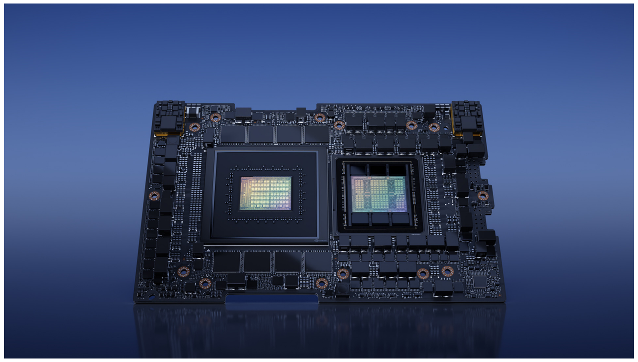
Intel Sapphire Rapids
(quad-chip module)
~48B transistors



AMD EPYC Bergamo
(9-chip module)
~82B transistors

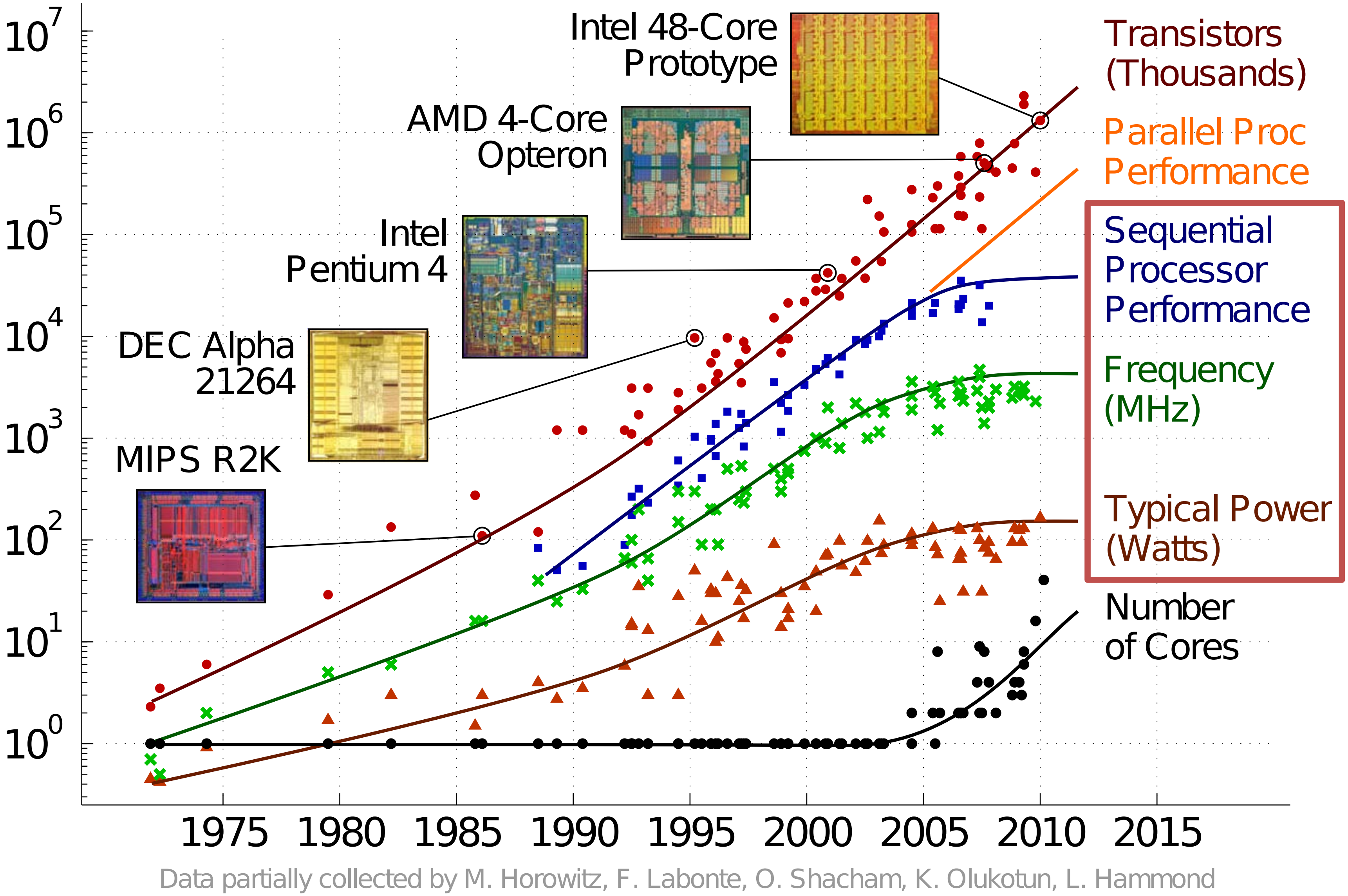

SCALAR ENGINES
ADAPTABLE ENGINES
INTELLIGENT ENGINES
DUAL-CORE ARM® CORTEX®-A72 APPLICATION PROCESSOR
DUAL-CORE ARM CORTEX-R5F REAL-TIME PROCESSOR
PLATFORM MANAGEMENT CONTROLLER
VERSAL™ ADAPTABLE HARDWARE
DSP ENGINES
PROGRAMMABLE NETWORK ON CHIP
PCIE® GEN5 W/DMA & CCIX, CXL
DDR4
32Gb/s
58Gb/s
112Gb/s
100G MULTIRATE ETHERNET CORES
600G ETHERNET CORES
600G INTERLAKEN CORES
400G CRYPTO ENGINES
MIPI
LVDS
GPIO

AMD Xilinx Versal Premium
~92B transistors



NVIDIA GH200
Grace Hopper Superchip
>200B transistors

End of Dennard Scaling:
Power Becomes the Limiting Factor



Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

[Figure credit: Christopher Batten, Cornell]

Power-Constrained Modern Computers

<<1W/chip

~1W/chip

~15W/chip

~50W/chip

~100W/chip

>100W/chip

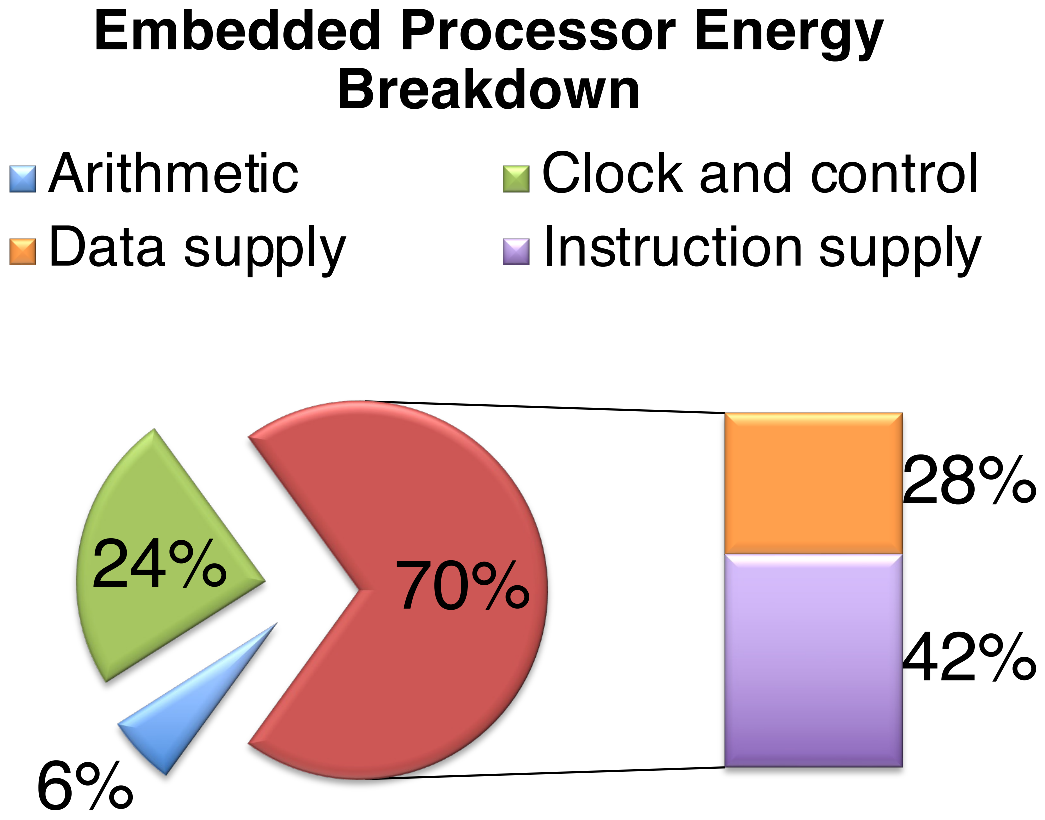
$$Power = \frac{Energy}{Second} = \downarrow \boxed{\frac{Energy}{Op}} \times \boxed{\frac{Ops}{Second}} \uparrow$$

To increase performance (Ops/Sec) in a power-constrained regime, **energy efficiency** (Ops/Joule) **must improve!**

Inefficiency of General-Purpose Computing

- Typical energy overhead (or “tax”) for every 10pJ arithmetic operations
  - 70pJ on instruction supply
  - 47pJ on data supply

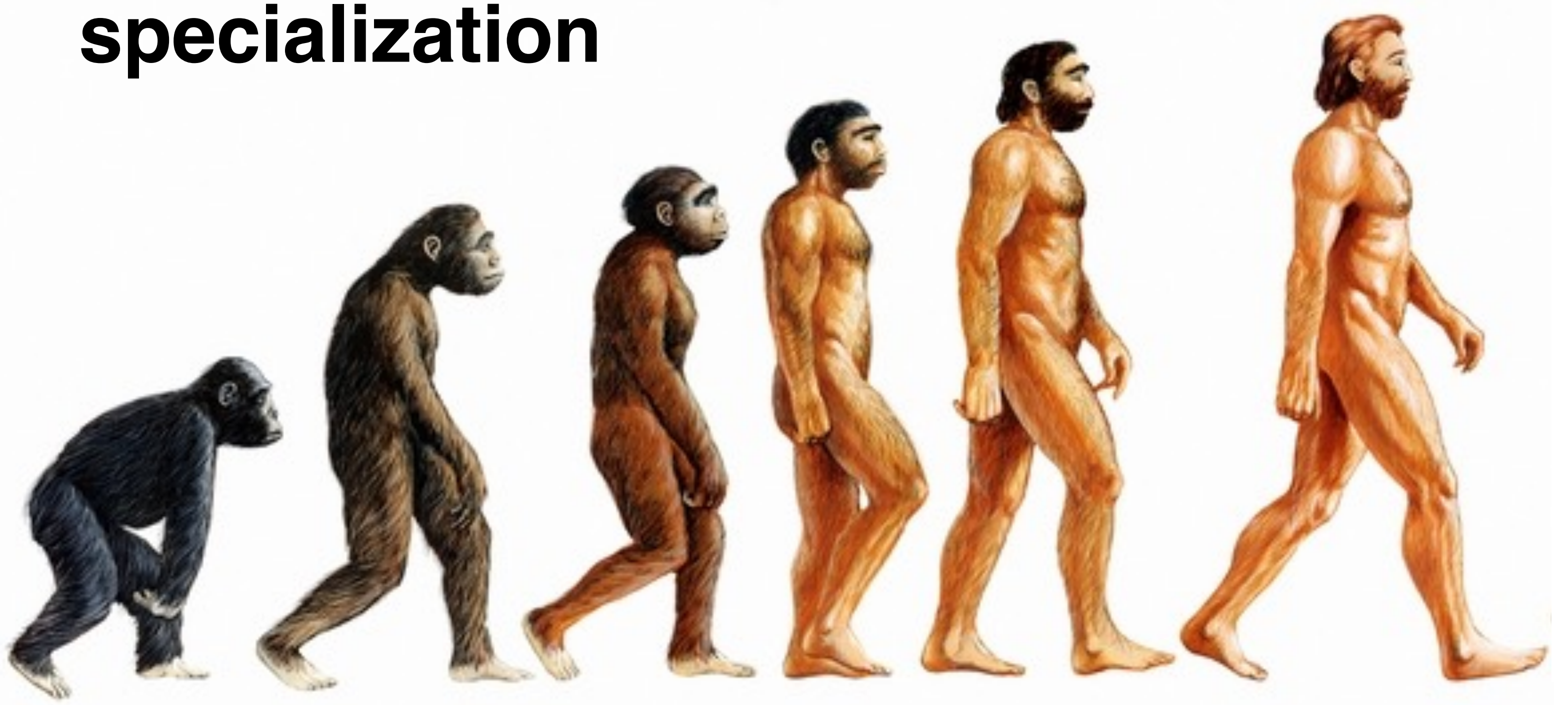
Also, only 59% of the instructions are arithmetic



[source: Dally et al. Efficient Embedded Computing, IEEE’08]

Advance of Civilization

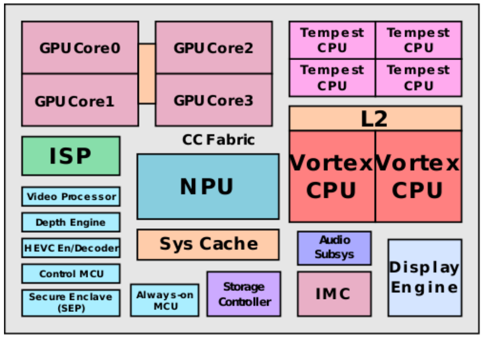
- For humans, Moore’s Law scaling of the brain has ended a long time ago
  - Number of neurons and their firing rate did not change significantly
- Remarkable advancement of civilization via **specialization**



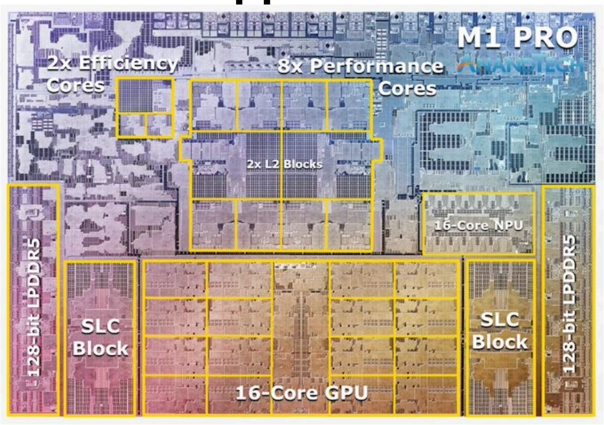
?

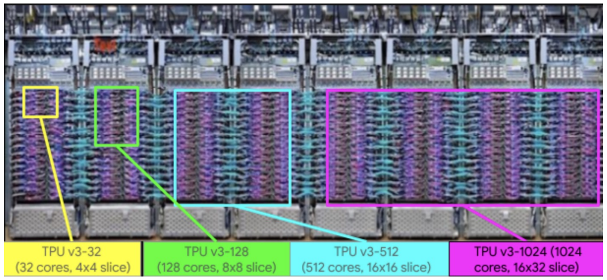
Diverse: Era of Hardware Heterogeneity

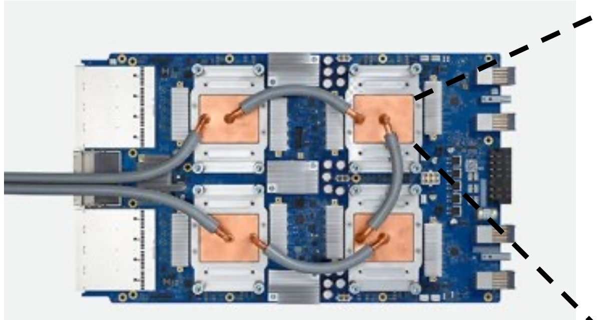
Apple 12 (iPhone X)



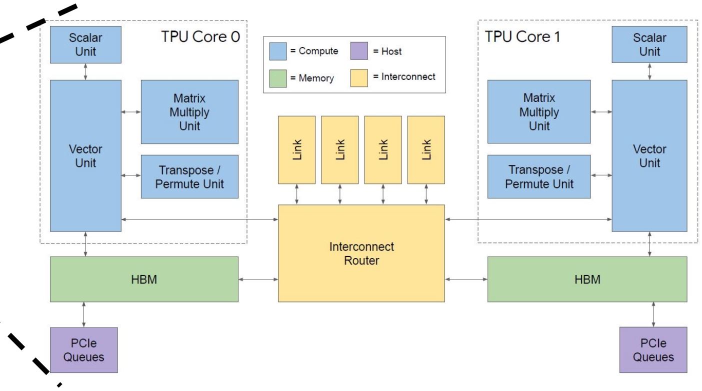
Apple M1 Pro




TPU v3-32 (32 cores, 4x4 slice)
TPU v3-128 (128 cores, 8x8 slice)
TPU v3-512 (512 cores, 16x16 slice)
TPU v3-1024 (1024 cores, 16x32 slice)



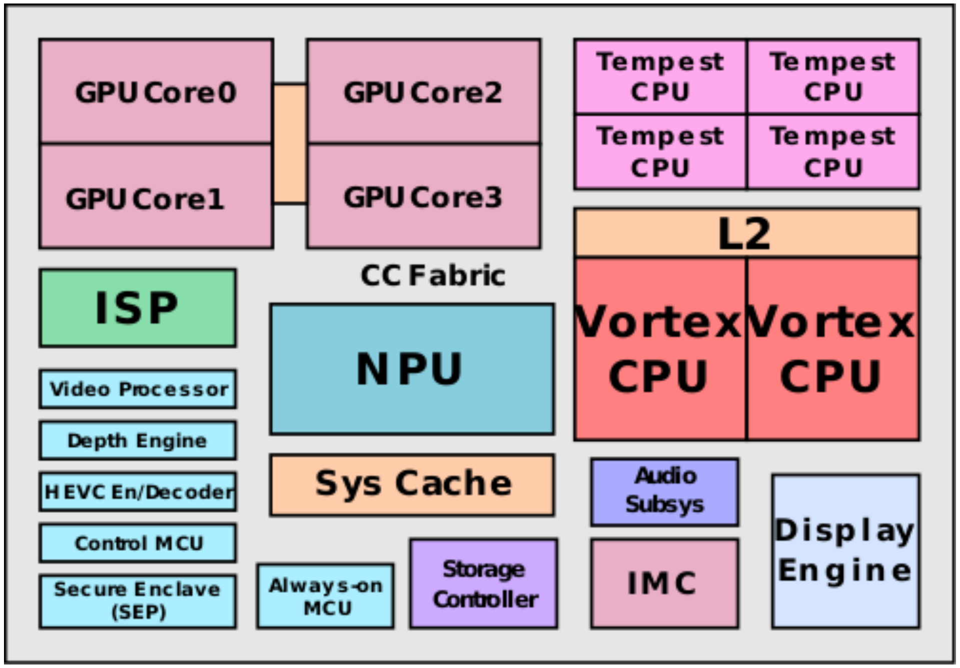
Google TPUv3



Special-purpose accelerators are increasingly deployed to improve performance & energy efficiency both in datacenters and at the edge

Hardware Specialization in Mobile Chips

System on chip (SoC)

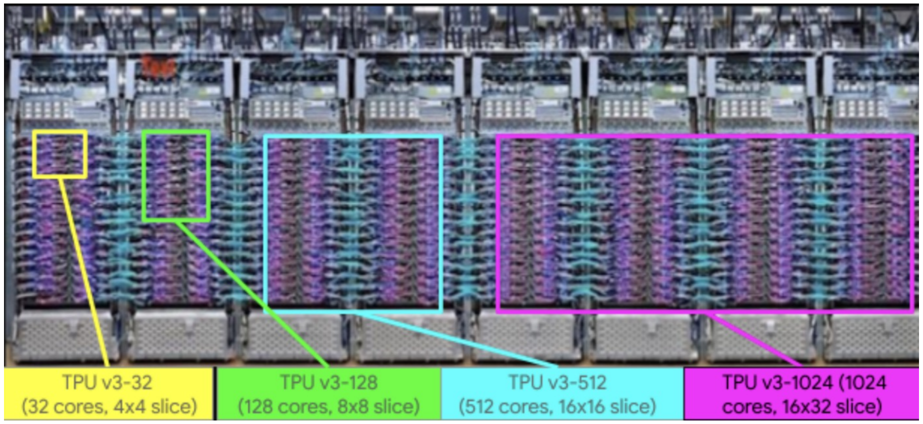


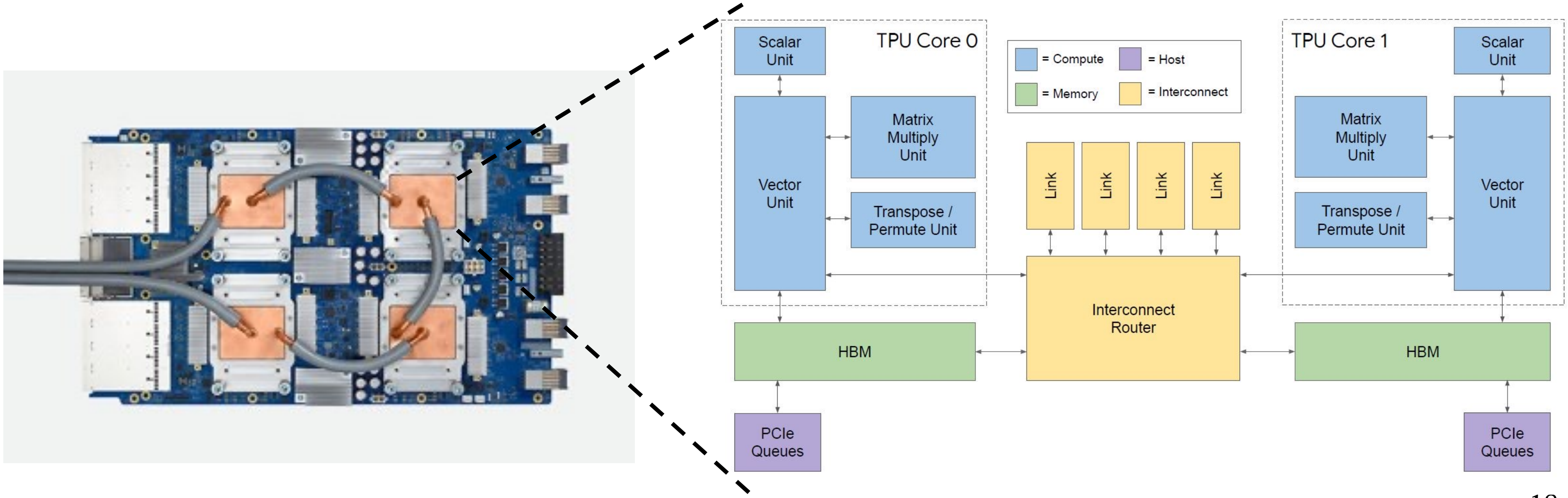
Apple 12 (iPhone X)

- Modern SoCs integrate a rich set of **special-purpose accelerators**
  - Speed up critical tasks
  - Reduce power consumption and cost
  - Increase energy efficiency

Hardware Specialization in Datacenters

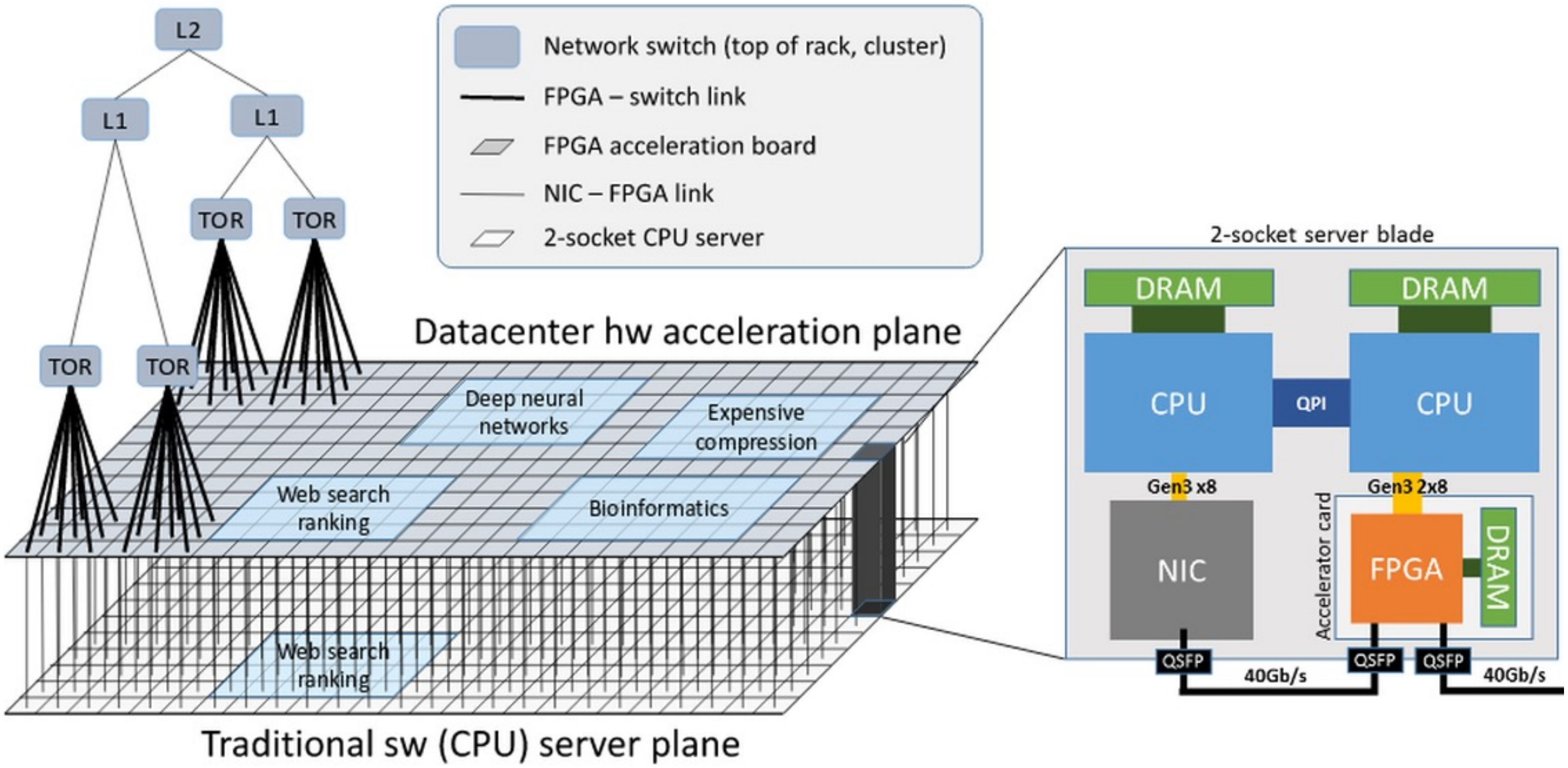
ASIC- and FPGA-based accelerators are being deployed for a rich mix of compute-intensive applications in cloud datacenters





Hardware Specialization in Datacenters

ASIC- and FPGA-based accelerators are being deployed for a rich mix of compute-intensive applications in cloud datacenters



Microsoft Cloud FPGA Platforms

Hardware Specialization for Deep Learning

Blue Chips

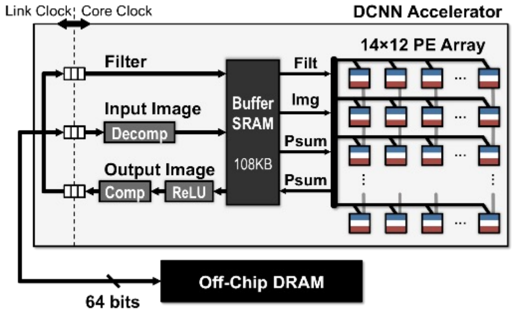
Amazon
Apple
Google
Intel
Microsoft …

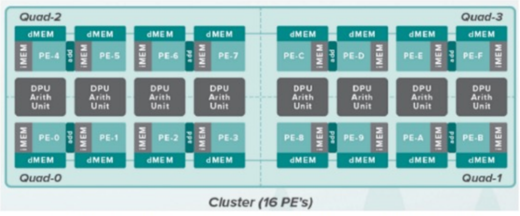
Startups

Cerebras
Graphcore
Groq
Mythic
SambaNova …

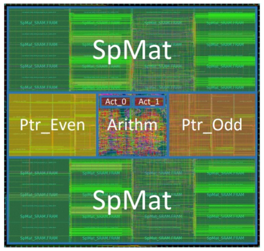
Academia

DianNao [Chen ASPLOS’14]
EIE/ESE [Han ISCA’16, FPGA’17]
Eyeriss [Chen ISCA’16, JSSC’17]
FINN [Umuroglu FPGA’17]
FracBNN [Zhang FPGA’21] …





CGRA Cluster of 16 PEs with 8 Arithmetic Units and 16KB data RAM

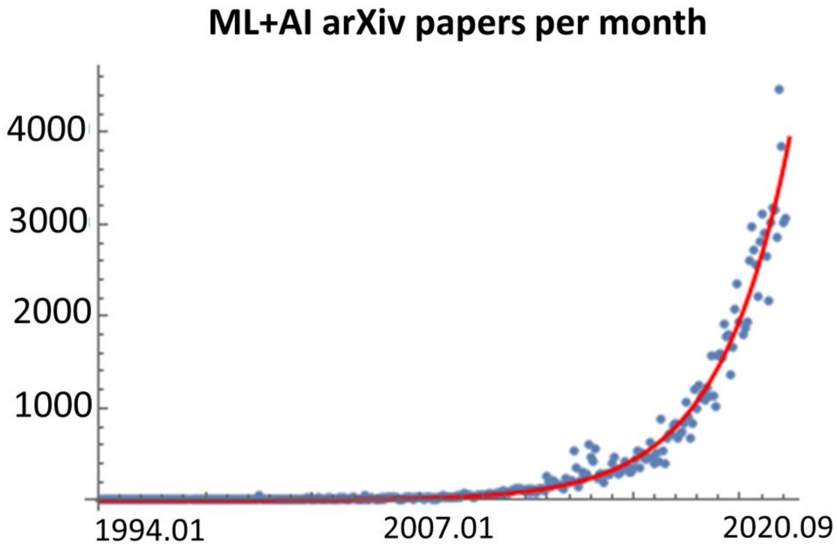


Deep learning has caused a revolution AI and computer hardware industry

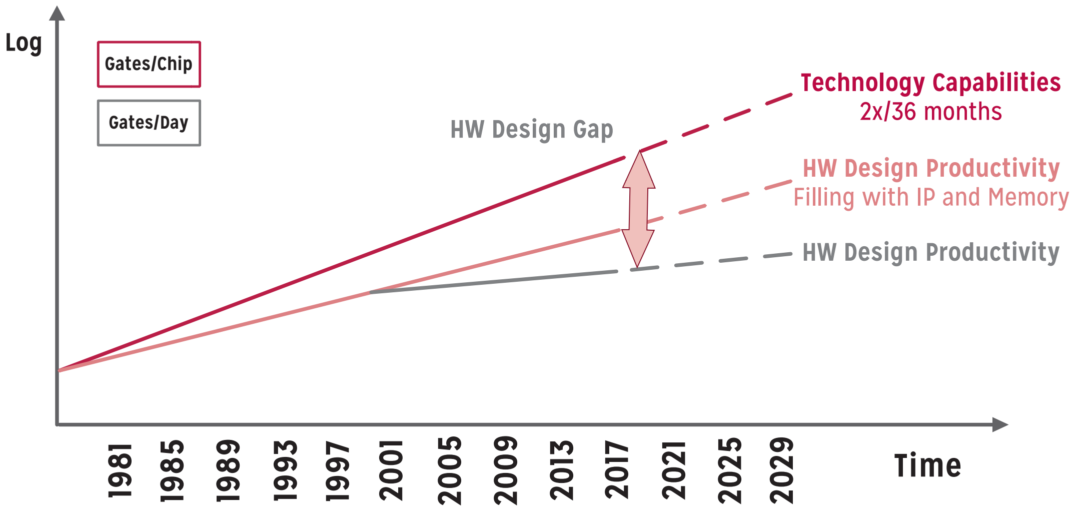
Increasing Specialization Demands (Even) Higher Design Productivity

Can custom hardware evolve fast enough to keep up?

- Target of specialization is moving rapidly

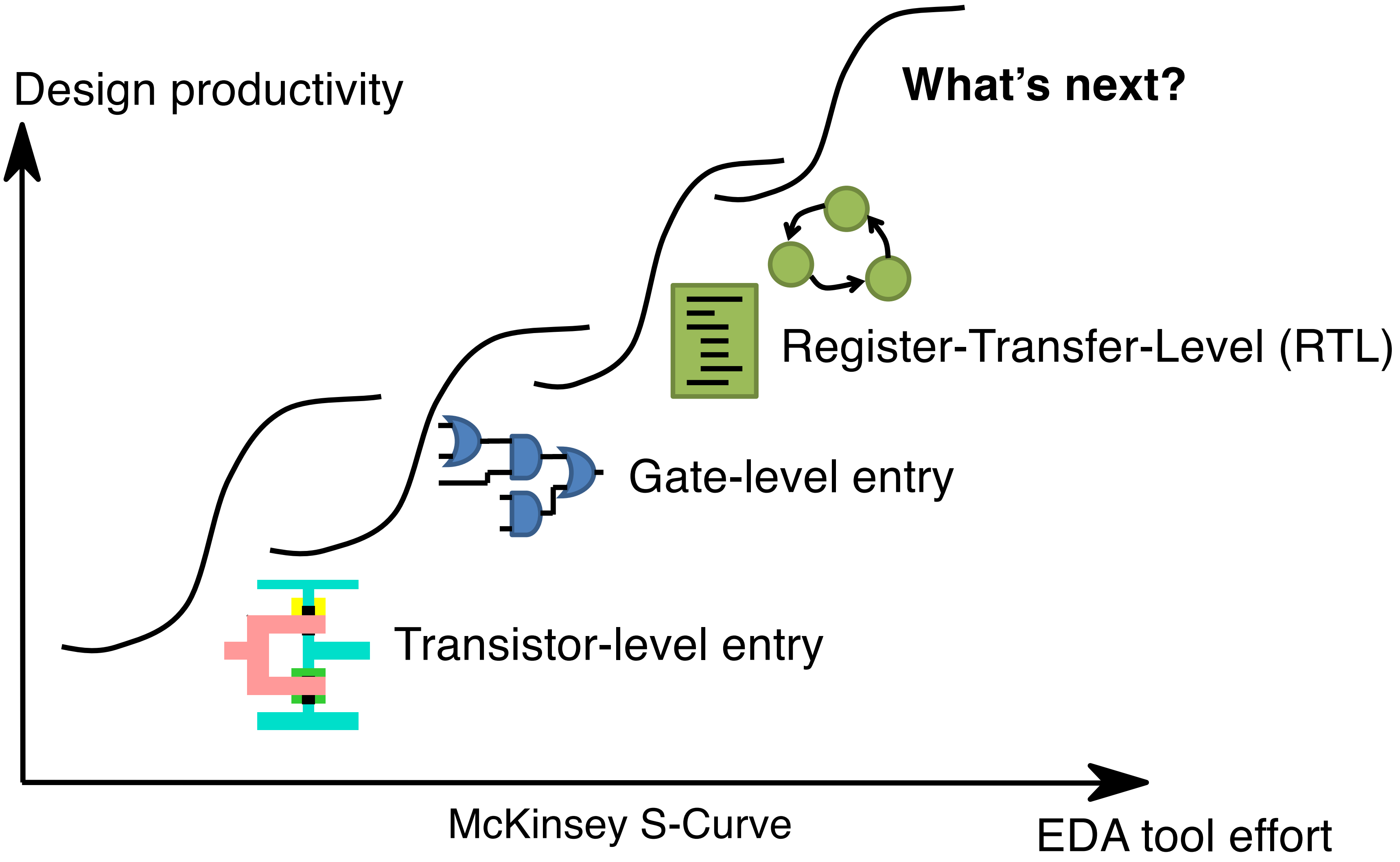


Number of machine learning papers published on arXiv has outpaced Moore's Law
[Dean et al., IEEE Micro 2018]



The design productivity gap
[Source: Workshops on Extreme Scale Design Automation: Challenges and Opportunities for 2025 and Beyond]

Evolution of Design Abstraction



Design productivity

What's next?

Register-Transfer-Level (RTL)

Gate-level entry

Transistor-level entry

McKinsey S-Curve

EDA tool effort

[source: Kurt Keutzer, UCB]

Motivation for High-Level Synthesis (HLS)

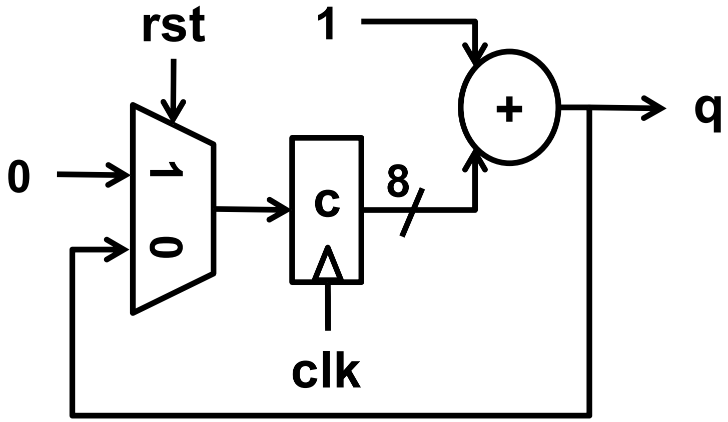
```
module dut(rst, clk, q);
  input rst;
  input clk;
  output q;
  reg [7:0] c;

  always @ (posedge clk)
  begin
    if (rst == 1b’1) begin
      c <= 8'b00000000;
  end
  else begin
    c <= c + 1;
  end

  assign q = c;
endmodule
```

RTL Verilog

vs.

```
uint8 dut() {
  static uint8 c;
  c+=1;
}
```

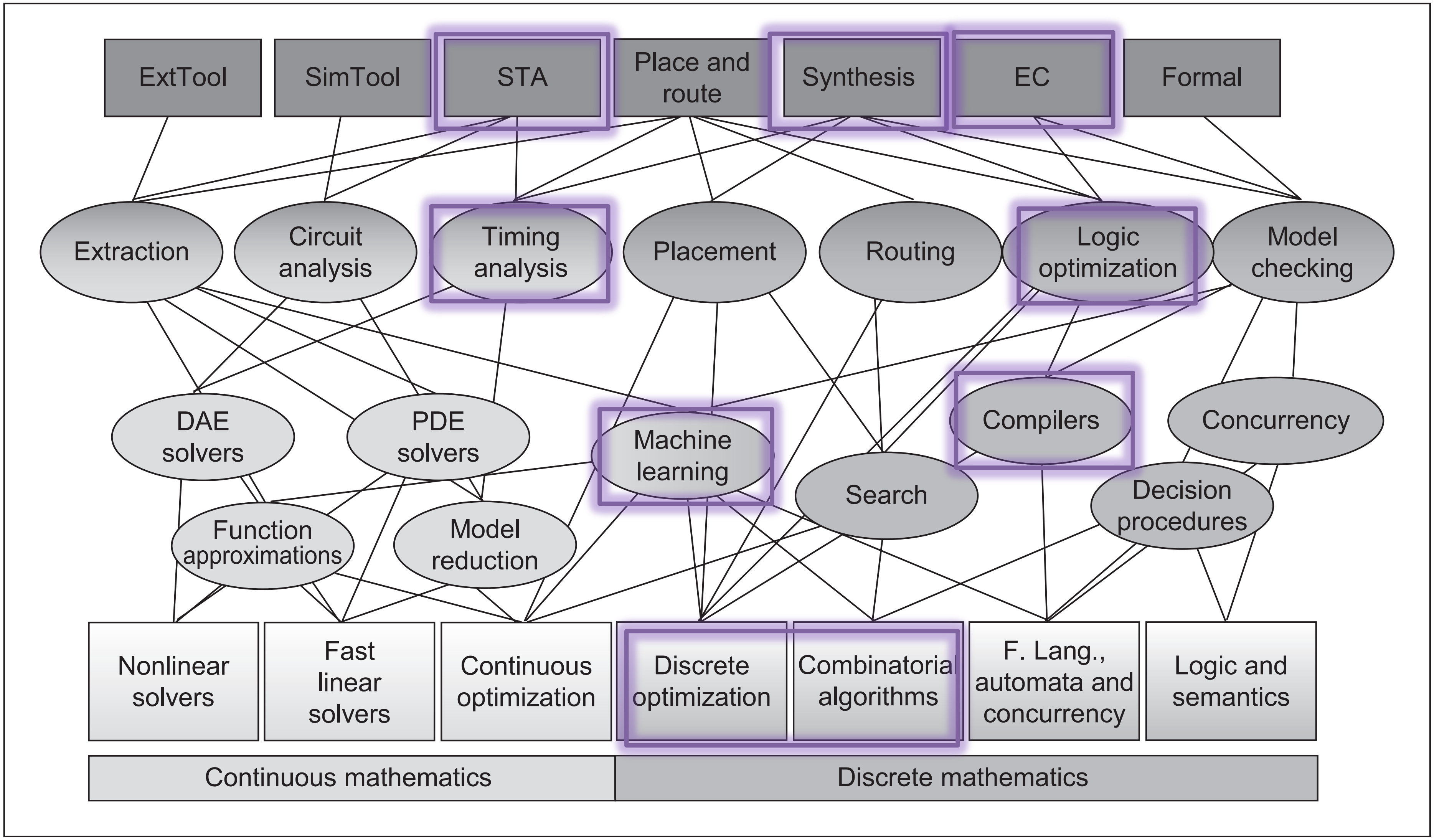
Automated with HLS



An 8-bit counter

Algorithms Drive Automation

Topics touched on in 6775

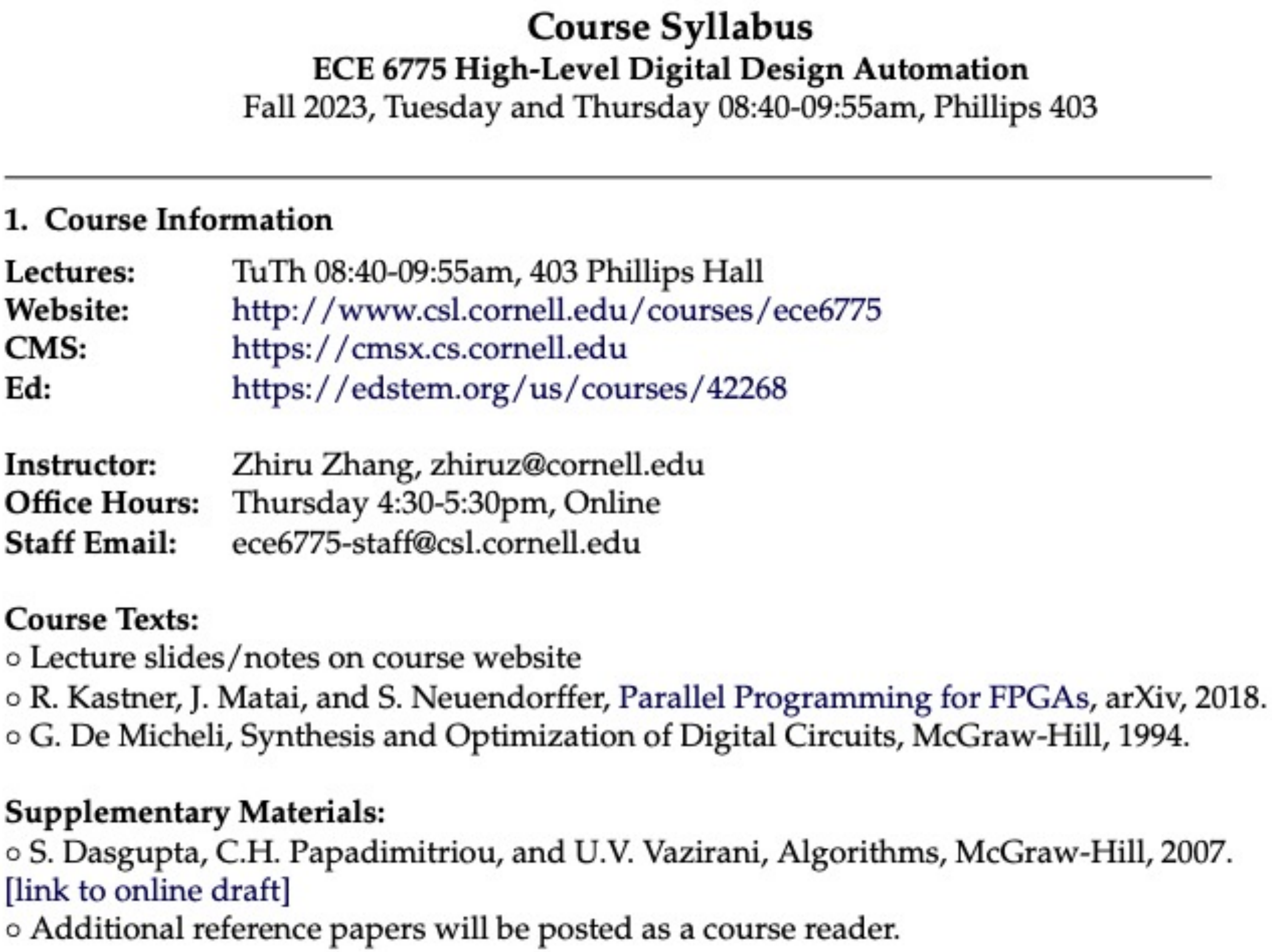


Key Algorithms in EDA

[source: Andreas Kuehlmann, Synopsys Inc.]

Course Organization

- Refer to syllabus for course organization details



Course Syllabus

ECE 6775 High-Level Digital Design Automation

Fall 2023, Tuesday and Thursday 08:40-09:55am, Phillips 403

1. Course Information

Lectures: TuTh 08:40-09:55am, 403 Phillips Hall
Website: http://www.csl.cornell.edu/courses/ece6775
CMS: https://cmsx.cs.cornell.edu
Ed: https://edstem.org/us/courses/42268

Instructor: Zhiru Zhang, zhiruz@cornell.edu
Office Hours: Thursday 4:30-5:30pm, Online
Staff Email: ece6775-staff@csl.cornell.edu

Course Texts:
○ Lecture slides/notes on course website
○ R. Kastner, J. Matai, and S. Neuendorffer, Parallel Programming for FPGAs, arXiv, 2018.
○ G. De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.

Supplementary Materials:
○ S. Dasgupta, C.H. Papadimitriou, and U.V. Vazirani, Algorithms, McGraw-Hill, 2007.
[link to online draft]
○ Additional reference papers will be posted as a course reader.

Course Roadmap

- Lecture and paper discussion sessions
  - **Background**
    - Introduction
    - Hardware specialization
    - Algorithm basics
  - **High-level synthesis**
    - C-based synthesis for FPGAs
    - Front-end compilation
    - Scheduling
    - Resource sharing
    - Pipelining
  - **More advanced topics**
    - Deep learning acceleration
    - Domain-specific programming

Preferred Background

- Working knowledge of the following at undergraduate level
  - C/C++
  - Digital logic and basic computer architecture concepts (e.g., adders, clock, registers, pipelining)

- Experiences with the following would increase appreciation & productivity
  - Algorithms and data structures
  - RTL design for FPGA or ASIC

Learning Outcomes: The Tangibles

- High-level digital **design** methodologies
  - *Design above register transfer level (RTL)*
  - Building realistic accelerators with C-based design flow

- High-level **design automation** algorithms
  - Fundamentals of *high-level synthesis (HLS)*
    - e.g., scheduling, resource sharing, pipelining
  - Useful *combinatorial optimization* techniques
    - e.g., graph algorithms, dynamic programming, greedy algorithms, integer linear programming

Learning Outcomes: The Intangibles

- Develop a principled approach to analyzing accelerator design process and essential design factors (e.g., parallelism, resources, precision)

- Gain comprehensive insights into accelerator design from the perspective of an HLS compiler

Achieve these objectives through a blend of theoretical foundation and practical implementation

NOT Our Goals

- Teach you the design of microprocessors
- Cover the whole breadth of EDA
- Write RTL code
- Make you an expert FPGA programmer

Assignments

- Two problem sets (8%)

- Four lab assignments (22%)
  - Design & programming assignments leveraging high-level synthesis tools and software compilers
  - Experiments to be conducted on **ecelinux** servers
    - % ssh -X <netid>@ecelinux-01.ece.cornell.edu
    - Necessary tools will be installed in common directories

Quizzes and Paper Readings

- Quizzes (6%)
  - You will need to answer pop quiz questions in most lectures (using itempool)
  - TWO lowest scores will be dropped

- Paper Readings (5%)
  - Two reading sessions
  - You are expected to read the paper or book chapter before the lecture, answer quiz questions, and participate in discussions
  - Reading assignment will be announced at least one week in advance

Exam

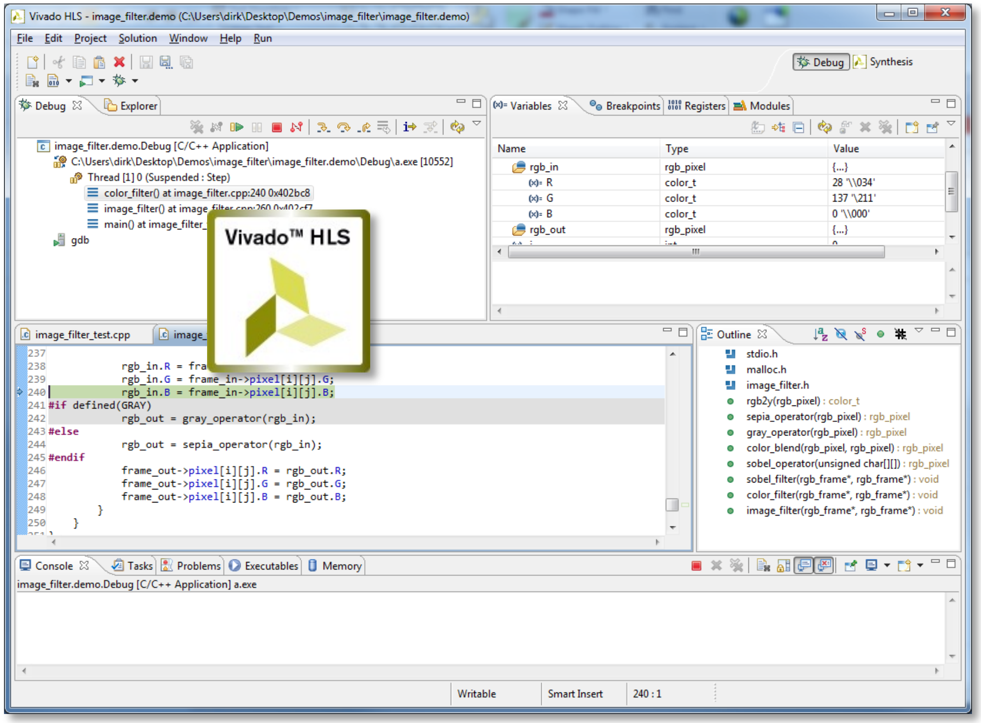
- In-class midterm (20%)
  - Open notes & open book
  - When: Thursday October 19th
  - No sit-down final

Final Project – 35%

- In-depth exploration of a research topic
  - (1) Designing new application-specific accelerators with HLS; OR
    (2) Devising new automation algorithms/tools
  - 3-4 students / team, depending on class size

- Timeline
  - Proposal due after midterm
  - Weekly meeting with the instructor to track progress
  - Demo before the final week
  - Final report due by the final exam date

High-Level Synthesis Tool



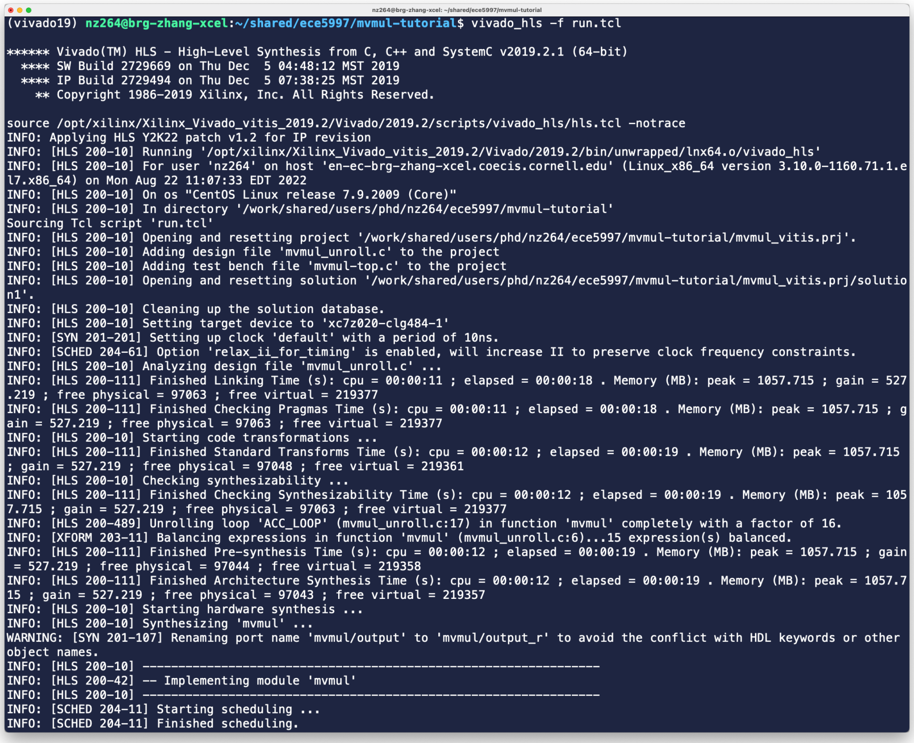
```
image_filter.demo.Debug [C/C++ Application]
  C:\Users\dirk\Desktop\Demos\image_filter\image_filter.demo\Debug\a.exe [10552]
    Thread [1] 0 (Suspended : Step)
      color_filter() at image_filter.cpp:240 0x402bc8
      image_filter() at image_filter.cpp:260 0x402cf7
      main() at image_filter_
  gdb
```

Name	Type	Value
rgb_in	rgb_pixel	{...}
R	color_t	28 '\\034'
G	color_t	137 '\211'
B	color_t	0 '\\000'
rgb_out	rgb_pixel	{...}

```

            rgb_in.R = fra
            rgb_in.G = frame_in->pixel[i][j].G;
            rgb_in.B = frame_in->pixel[i][j].B;
#if defined(GRAY)
            rgb_out = gray_operator(rgb_in);
#else
            rgb_out = sepia_operator(rgb_in);
#endif
            frame_out->pixel[i][j].R = rgb_out.R;
            frame_out->pixel[i][j].G = rgb_out.G;
            frame_out->pixel[i][j].B = rgb_out.B;
        }
    }
```

```
stdio.h
malloc.h
image_filter.h
rgb2y(rgb_pixel) : color_t
sepia_operator(rgb_pixel) : rgb_pixel
gray_operator(rgb_pixel) : rgb_pixel
color_blend(rgb_pixel, rgb_pixel) : rgb_pixel
sobel_operator(unsigned char[][]) : rgb_pixel
sobel_filter(rgb_frame*, rgb_frame*) : void
color_filter(rgb_frame*, rgb_frame*) : void
image_filter(rgb_frame*, rgb_frame*) : void
```

image_filter.demo.Debug [C/C++ Application] a.exe

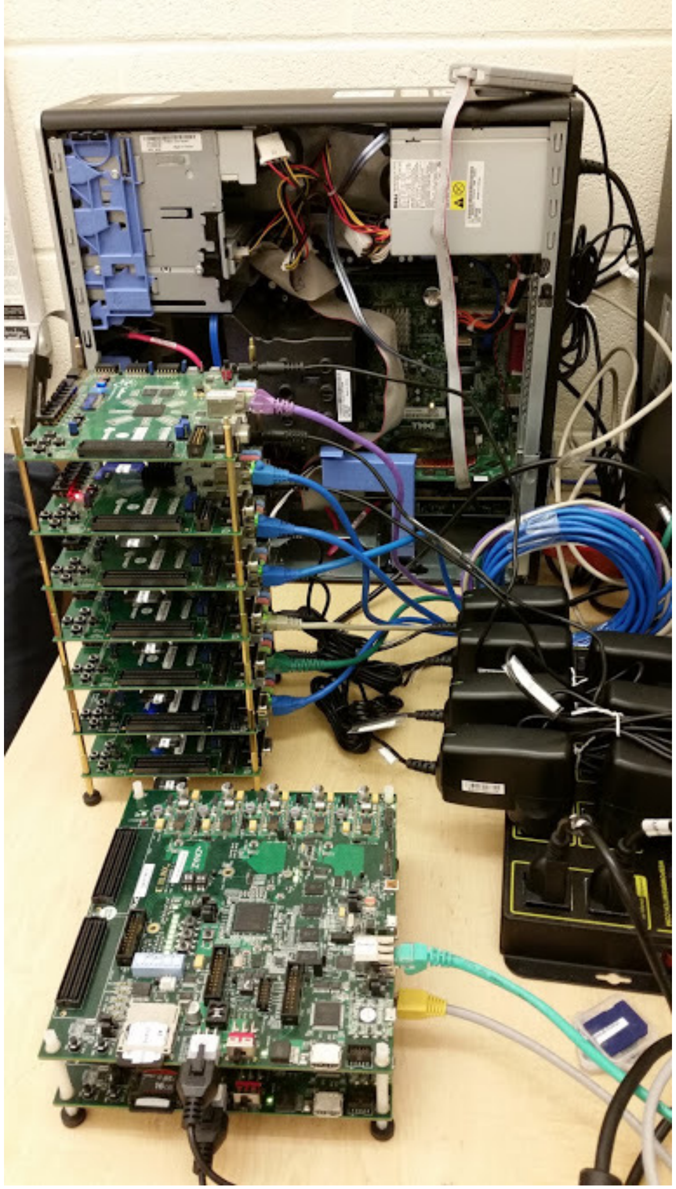


```
(vivado19) nz264@brg-zhang-xcel:~/shared/ece5997/mvmul-tutorial$ vivado_hls -f run.tcl

****** Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC v2019.2.1 (64-bit)
  **** SW Build 2729669 on Thu Dec  5 04:48:12 MST 2019
  **** IP Build 2729494 on Thu Dec  5 07:38:25 MST 2019
    ** Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.

source /opt/xilinx/Xilinx_Vivado_vitis_2019.2/Vivado/2019.2/scripts/vivado_hls/hls.tcl -notrace
INFO: Applying HLS Y2K22 patch v1.2 for IP revision
INFO: [HLS 200-10] Running '/opt/xilinx/Xilinx_Vivado_vitis_2019.2/Vivado/2019.2/bin/unwrapped/lnx64.o/vivado_hls'
INFO: [HLS 200-10] For user 'nz264' on host 'en-ec-brg-zhang-xcel.coecis.cornell.edu' (Linux_x86_64 version 3.10.0-1160.71.1.e
l7.x86_64) on Mon Aug 22 11:07:33 EDT 2022
INFO: [HLS 200-10] On os "CentOS Linux release 7.9.2009 (Core)"
INFO: [HLS 200-10] In directory '/work/shared/users/phd/nz264/ece5997/mvmul-tutorial'
Sourcing Tcl script 'run.tcl'
INFO: [HLS 200-10] Opening and resetting project '/work/shared/users/phd/nz264/ece5997/mvmul-tutorial/mvmul_vitis.prj'.
INFO: [HLS 200-10] Adding design file 'mvmul_unroll.c' to the project
INFO: [HLS 200-10] Adding test bench file 'mvmul-top.c' to the project
INFO: [HLS 200-10] Opening and resetting solution '/work/shared/users/phd/nz264/ece5997/mvmul-tutorial/mvmul_vitis.prj/solutio
n1'.
INFO: [HLS 200-10] Cleaning up the solution database.
INFO: [HLS 200-10] Setting target device to 'xc7z020-clg484-1'
INFO: [SYN 201-201] Setting up clock 'default' with a period of 10ns.
INFO: [SCHED 204-61] Option 'relax_ii_for_timing' is enabled, will increase II to preserve clock frequency constraints.
INFO: [HLS 200-10] Analyzing design file 'mvmul_unroll.c' ...
INFO: [HLS 200-111] Finished Linking Time (s): cpu = 00:00:11 ; elapsed = 00:00:18 . Memory (MB): peak = 1057.715 ; gain = 527
.219 ; free physical = 97063 ; free virtual = 219377
INFO: [HLS 200-111] Finished Checking Pragmas Time (s): cpu = 00:00:11 ; elapsed = 00:00:18 . Memory (MB): peak = 1057.715 ; g
ain = 527.219 ; free physical = 97063 ; free virtual = 219377
INFO: [HLS 200-10] Starting code transformations ...
INFO: [HLS 200-111] Finished Standard Transforms Time (s): cpu = 00:00:12 ; elapsed = 00:00:19 . Memory (MB): peak = 1057.715
; gain = 527.219 ; free physical = 97048 ; free virtual = 219361
INFO: [HLS 200-10] Checking synthesizability ...
INFO: [HLS 200-111] Finished Checking Synthesizability Time (s): cpu = 00:00:12 ; elapsed = 00:00:19 . Memory (MB): peak = 105
7.715 ; gain = 527.219 ; free physical = 97063 ; free virtual = 219377
INFO: [HLS 200-489] Unrolling loop 'ACC_LOOP' (mvmul_unroll.c:17) in function 'mvmul' completely with a factor of 16.
INFO: [XFORM 203-11] Balancing expressions in function 'mvmul' (mvmul_unroll.c:6)...15 expression(s) balanced.
INFO: [HLS 200-111] Finished Pre-synthesis Time (s): cpu = 00:00:12 ; elapsed = 00:00:19 . Memory (MB): peak = 1057.715 ; gain
 = 527.219 ; free physical = 97044 ; free virtual = 219358
INFO: [HLS 200-111] Finished Architecture Synthesis Time (s): cpu = 00:00:12 ; elapsed = 00:00:19 . Memory (MB): peak = 1057.7
15 ; gain = 527.219 ; free physical = 97043 ; free virtual = 219357
INFO: [HLS 200-10] Starting hardware synthesis ...
INFO: [HLS 200-10] Synthesizing 'mvmul' ...
WARNING: [SYN 201-107] Renaming port name 'mvmul/output' to 'mvmul/output_r' to avoid the conflict with HDL keywords or other
object names.
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [HLS 200-42] -- Implementing module 'mvmul'
INFO: [HLS 200-10] ----------------------------------------------------------------
INFO: [SCHED 204-11] Starting scheduling ...
INFO: [SCHED 204-11] Finished scheduling.
```

Tutorial on AMD Xilinx Vivado HLS (v2019.2), Tuesday 9/5

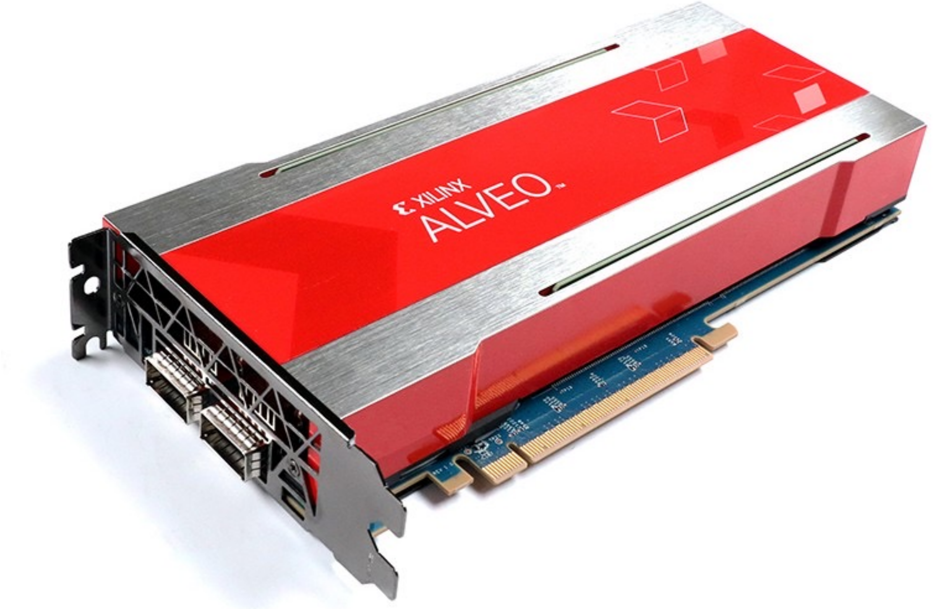
Local Cluster of Embedded FPGAs

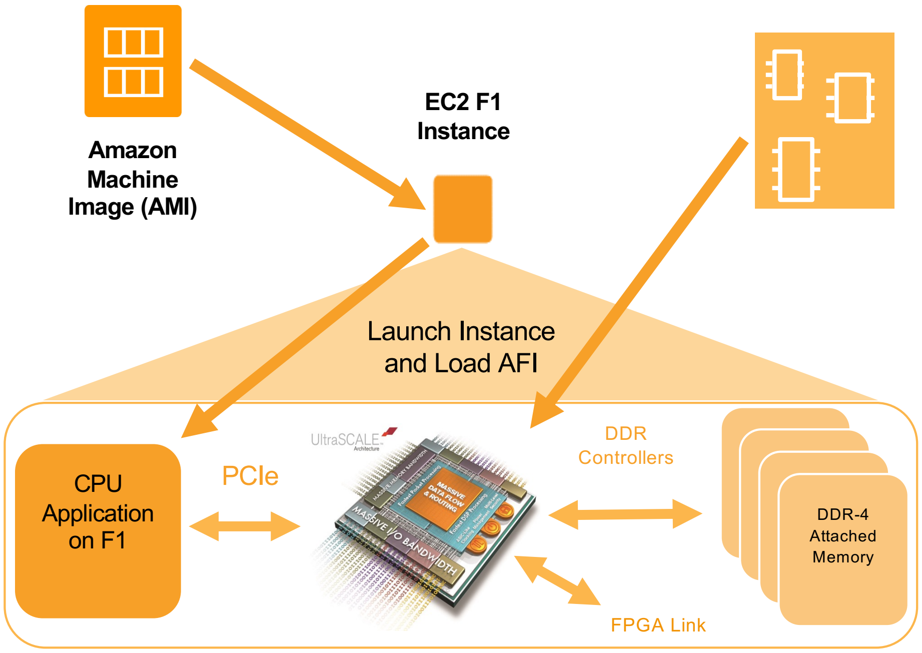
- For labs and project, we will use Zynq-based FPGA development boards (ZedBoard, Ultra96v2)
  - FPGA + Dual-core ARM
  - Boot Linux



Datacenter FPGA Platforms

- For the final project, students can also choose to explore datacenter FPGA platforms such as AMD Xilinx Alveo U280 and AWS F1 cloud instances





Amazon Machine Image (AMI)

EC2 F1 Instance

Launch Instance and Load AFI

CPU Application on F1

PCIe

DDR Controllers

DDR-4 Attached Memory

FPGA Link

Takeaway Points

- End of Dennard scaling leads to increasing **hardware specialization** to sustain improvement in performance and energy efficiency

- Increasing specialization and continued exponential growth in silicon capacity demands higher level of **design abstraction**

- HLS is a promising next step for EDA, which is fueled by sophisticated and yet scalable **algorithms**

Before Next Lecture

- Action items
  - Check out the course website
  - **Read through the course syllabus**
  - **Verify your login on ecelinux**
    - ssh -X <netid>@ecelinux.ece.cornell.edu

Acknowledgements

- These slides contain/adapt materials developed by
  - Prof. Jason Cong (UCLA)
  - Prof. David Z. Pan (UT Austin)