

Digital Design & Computer Arch.

Lecture 3b: Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Ataberk Olgun)
ETH Zurich
Spring 2022
3 March 2022

Lab Sessions

■ Where?

- ❑ On-site
- ❑ Online (Zoom Meetings)

Tuesday	Wednesday	Friday - 1	Friday - 2
HG E19	HG E19	HG D11	HG E19
HG E26.1	HG E26.1	HG D12	HG E26.1
HG E26.3	HG E26.3	HG E26.3	HG E26.3
HG E27	HG E27	HG E27	HG E27

[DDCA Course Catalogue Web Page](#)

■ When?

- ❑ Tuesday 16:15-18:00
- ❑ Wednesday 16:15-18:00
- ❑ Friday 08:15-10:00
- ❑ Friday 10:15-12:00

Grading

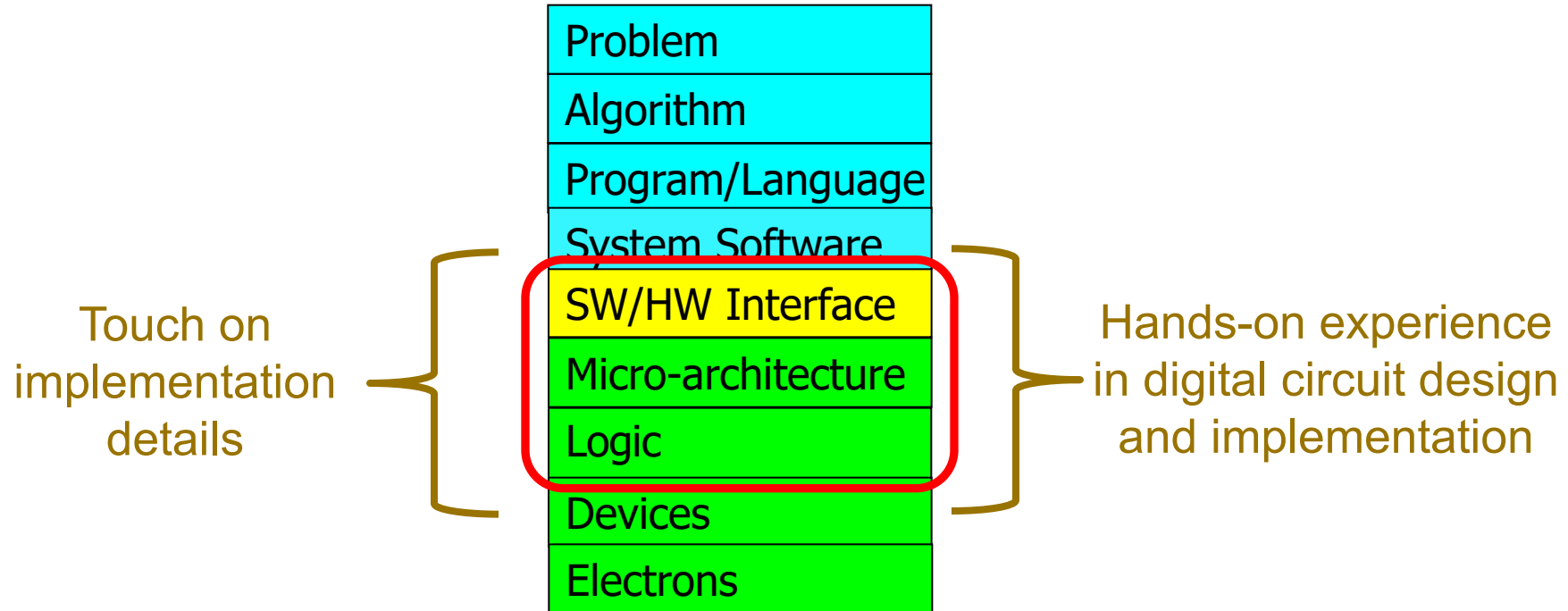
- **10** labs, **30** points in total
- We will put the **lab manuals** online
 - <https://safari.ethz.ch/digitaltechnik/spring2022/doku.php?id=labs>
- **Grading Policy**
 - In-class evaluation (70%) and *mandatory* lab reports (30%)
 - *1-point penalty for late submission of the report*
 - You should finish the labs within 1 week after they are announced
 - You can use your grades for labs from past years
 - You can find your grades in last year's Moodle page: <https://moodle-app2.let.ethz.ch/course/view.php?id=14545>
- **For questions**
 - digitaltechnik@lists.inf.ethz.ch (Emails are sent to all TAs)
 - Moodle forum (per lab/assignment)

Agenda

- Logistics
- **What Will We Learn?**
- What is an FPGA?
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- More about FPGAs
- Programming an FPGA
- Tutorial and Demo

What Will We Learn?

The Transformation Hierarchy



Understanding how a processor works underneath the software layer

What Will We Learn? (2)

- How to make **trade-offs** between **performance** and **area/complexity** in your hardware implementation
- **Hands-on experience** on:
 - ❑ Hardware **Prototyping** on Field Programmable Gate Arrays (FPGAs)
 - ❑ **Debugging** Your Hardware Implementation
 - ❑ Hardware Description Language (**HDL**)
 - ❑ Hardware Design Flow
 - ❑ Computer-Aided Design (**CAD**) Tools

Agenda

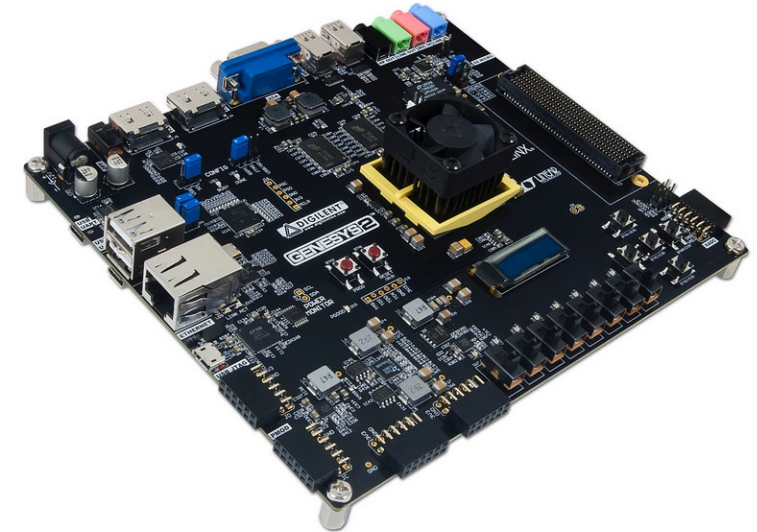
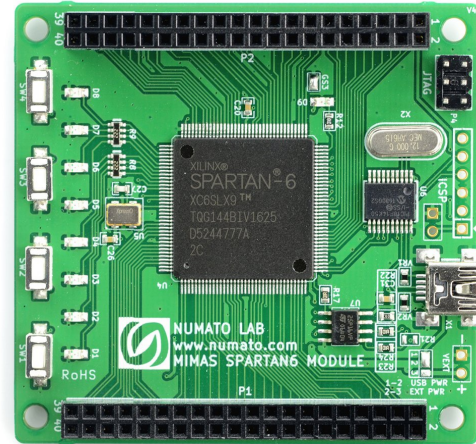
- Logistics
- What Will We Learn?
- **What is an FPGA?**
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- More about FPGAs
- Programming an FPGA
- Tutorial and Demo

What is an FPGA?

- Field Programmable Gate Array: FPGA



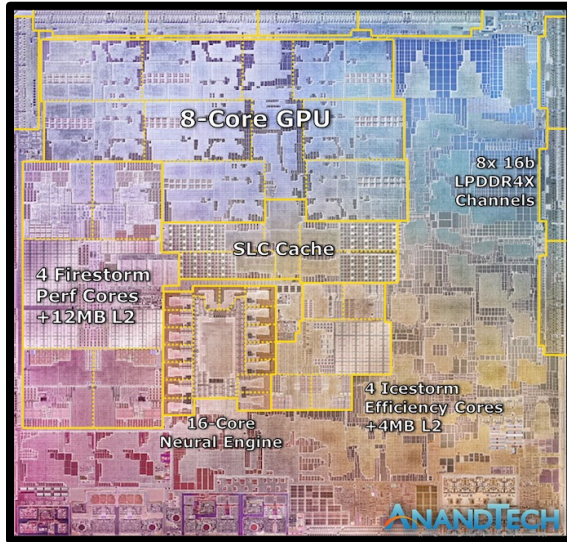
© Raimond Spekking / [CC BY-SA 4.0](#) (via Wikimedia Commons)



- FPGA is a **software-reconfigurable** hardware substrate
 - ❑ Reconfigurable **functions**
 - ❑ Reconfigurable **interconnection** of functions
 - ❑ Reconfigurable **input/output (IO)**

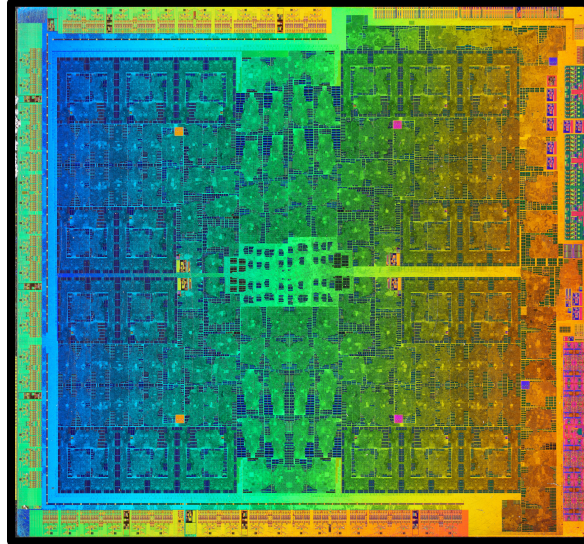
FPGAs & Other Integrated Circuits

CPU_s



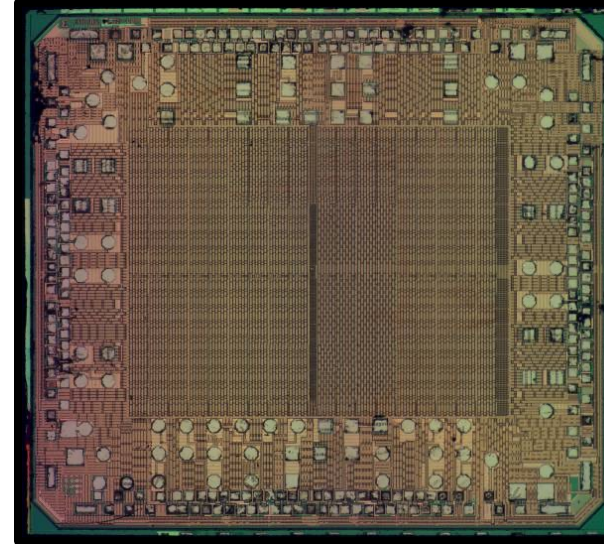
Apple M1

GPU_s



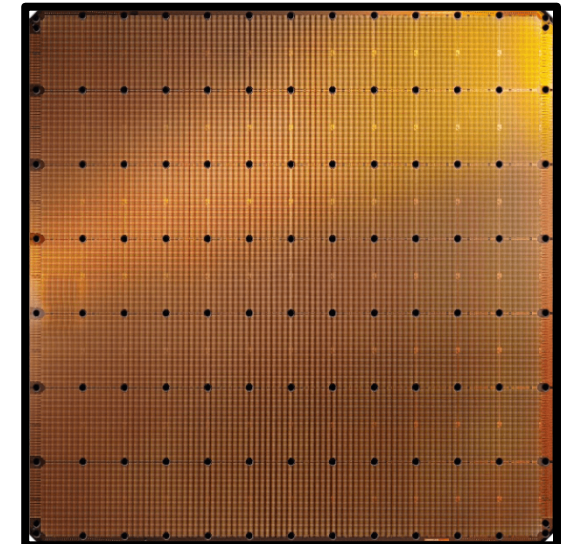
Nvidia GTX 1070

FPGA_s



Xilinx Spartan

ASIC_s



Cerebras WSE-2

**Flexibility
Programming Ease**

Efficiency



Agenda

- Logistics
- What Will We Learn?
- What is an FPGA?
- **FPGAs in Today's Systems**
- Overview of the Lab Exercises
- More about FPGAs
- Programming an FPGA
- Tutorial and Demo

FPGAs in Today's Systems: Project Brainwave

- “Microsoft’s *Project Brainwave* is a **deep learning platform** for real-time AI inference in the cloud and on the edge. A soft Neural Processing Unit (NPU), based on a high-performance **field-programmable gate array (FPGA)**, accelerates deep neural network (DNN) inferencing, with applications in computer vision and natural language processing. Project Brainwave is transforming computing by augmenting CPUs with an interconnected and configurable compute layer composed of programmable silicon.”

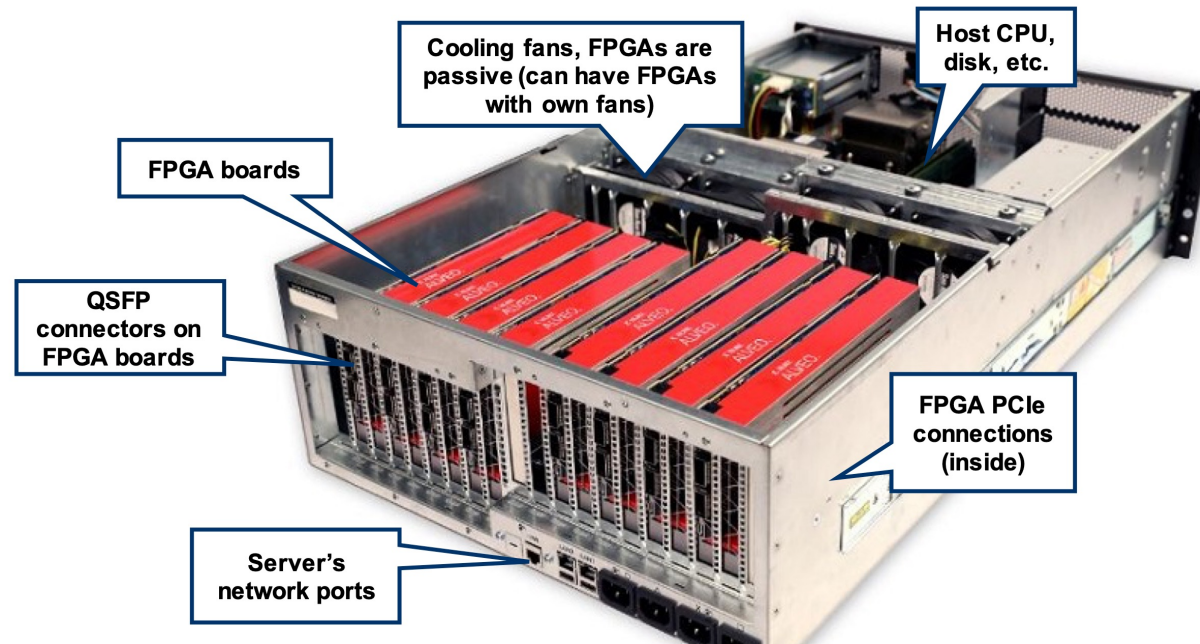


<https://www.microsoft.com/en-us/research/project/project-brainwave/>

<https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/>

FPGAs in Today's Systems: Amazon EC2 F1

- “Amazon EC2 F1 instances use **FPGAs** to enable delivery of **custom hardware accelerations**. F1 instances are *easy to program* and come with everything you need to develop, simulate, debug, and compile your hardware acceleration code, including an FPGA Developer AMI and supporting hardware level development on the cloud. Using F1 instances to deploy hardware accelerations can be useful in many applications **to solve complex science, engineering, and business problems that require high bandwidth, enhanced networking, and very high compute capabilities.**”



<https://aws.amazon.com/ec2/instance-types/f1/>

https://caslab.csl.yale.edu/courses/EENG428/19-20a/slides/eeng428_lecture_001_intro.pdf

FPGAs in Today's Systems: DNA Sequencing

- DRAGEN's suite of analysis pipelines are engineered to run on **FPGAs**, offering hardware-accelerated implementations of genomic analysis algorithms, including BCL conversion, mapping and alignment, sorting, duplicate marking and haplotype variant calling.



Illumina DRAGEN (Dynamic Read Analysis for GENomics) Bio-IT Platform



Illumina NextSeq 2000

<https://www.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html>

FPGAs in Today's Systems: More Bioinformatics

Bioinformatics



Article Navigation

GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping ^{FREE}

Mohammed Alser ✉, Hasan Hassan, Hongyi Xin, Oğuz Ergin, Onur Mutlu ✉, Can Alkan ✉

Bioinformatics, Volume 33, Issue 21, 01 November 2017, Pages 3355–3363,

<https://doi.org/10.1093/bioinformatics/btx342>

Published: 31 May 2017 **Article history** ▼

Bioinformatics



Article Navigation

SneakySnake: a fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs ^{FREE}

Mohammed Alser ✉, Taha Shahroodi, Juan Gómez-Luna, Can Alkan ✉, Onur Mutlu ✉

Bioinformatics, Volume 36, Issue 22-23, 1 December 2020, Pages 5282–5290,

<https://doi.org/10.1093/bioinformatics/btaa1015>

Published: 26 December 2020 **Article history** ▼

Alser+, "[GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping](#)", *Bioinformatics*, 2017.

Alser+, "[SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs](#)", *Bioinformatics*, 2020.



Accelerating Genome Analysis [IEEE MICRO 2020]

- Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu, **"Accelerating Genome Analysis: A Primer on an Ongoing Journey"** *IEEE Micro (IEEE MICRO)*, Vol. 40, No. 5, pages 65-75, September/October 2020.
[[Slides \(pptx\)\(pdf\)](#)]
[[Talk Video \(1 hour 2 minutes\)](#)]

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser

ETH Zürich

Zülal Bingöl

Bilkent University

Damla Senol Cali

Carnegie Mellon University

Jeremie Kim

ETH Zurich and Carnegie Mellon University

Saugata Ghose

University of Illinois at Urbana–Champaign and
Carnegie Mellon University

Can Alkan

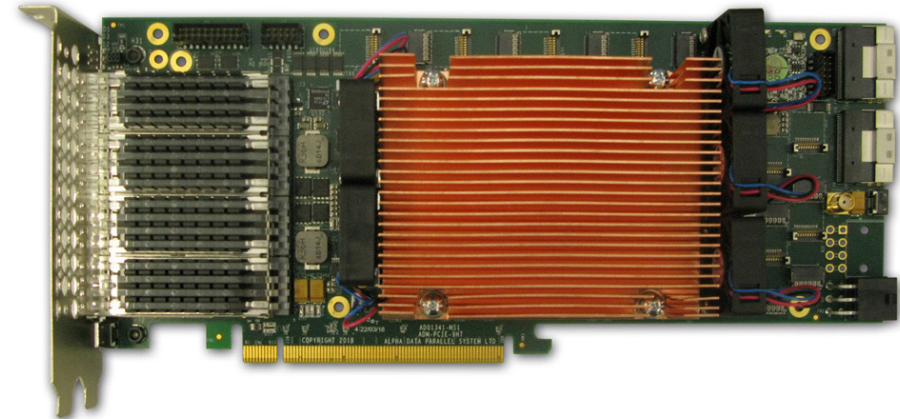
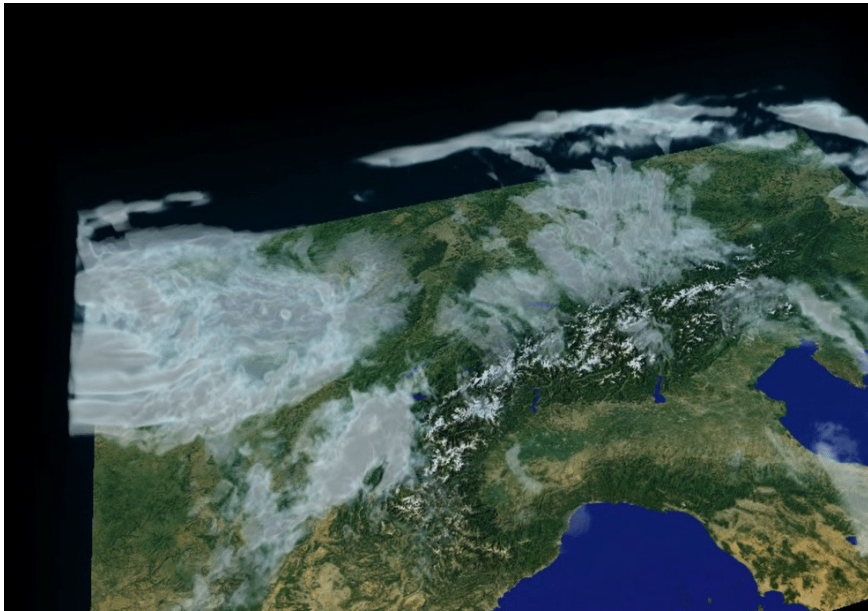
Bilkent University

Onur Mutlu

ETH Zurich, Carnegie Mellon University, and
Bilkent University

Accelerating Climate Modeling Using FPGAs

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, September 2020.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (23 minutes)]
One of the four papers nominated for the Stamatis Vassiliadis Memorial Best Paper Award.



Near-Memory Acceleration using FPGAs

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,
[**"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"**](#)
[*IEEE Micro*](#) (***IEEE MICRO***), 2021.

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] Mohammed Alser[◇] Damla Senol Cali[✕]

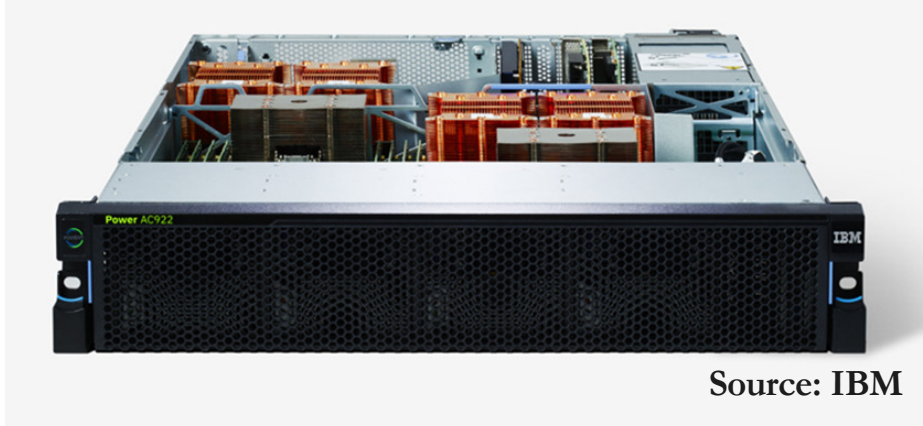
Dionysios Diamantopoulos[▽] Juan Gómez-Luna[◇]

Henk Corporaal^{*} Onur Mutlu^{◇✕}

[◇]*ETH Zürich* [✕]*Carnegie Mellon University*

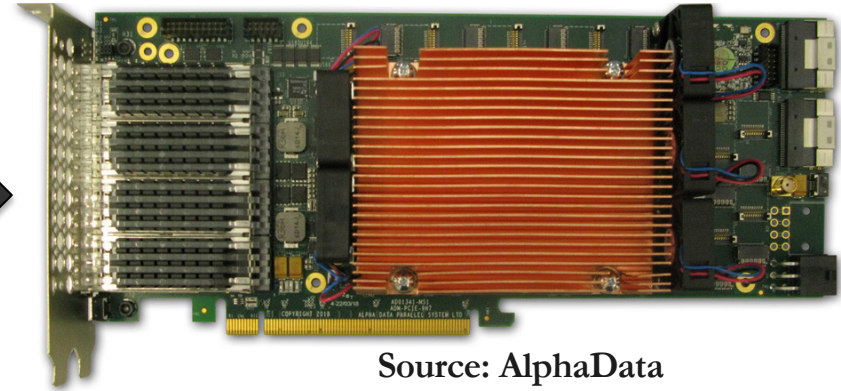
^{*}*Eindhoven University of Technology* [▽]*IBM Research Europe*

Near-Memory Acceleration using FPGAs



Source: IBM

IBM POWER9 CPU



Source: AlphaData

HBM-based FPGA board

FPGA-based Near-Memory Accelerator

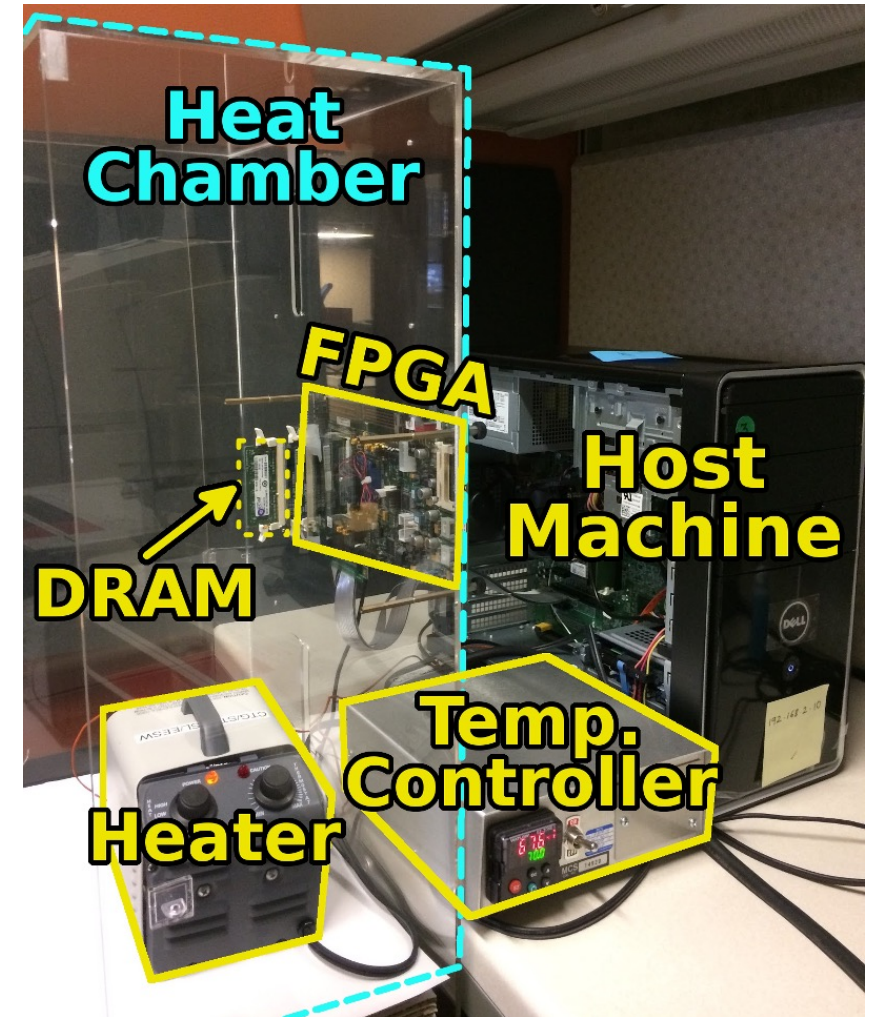
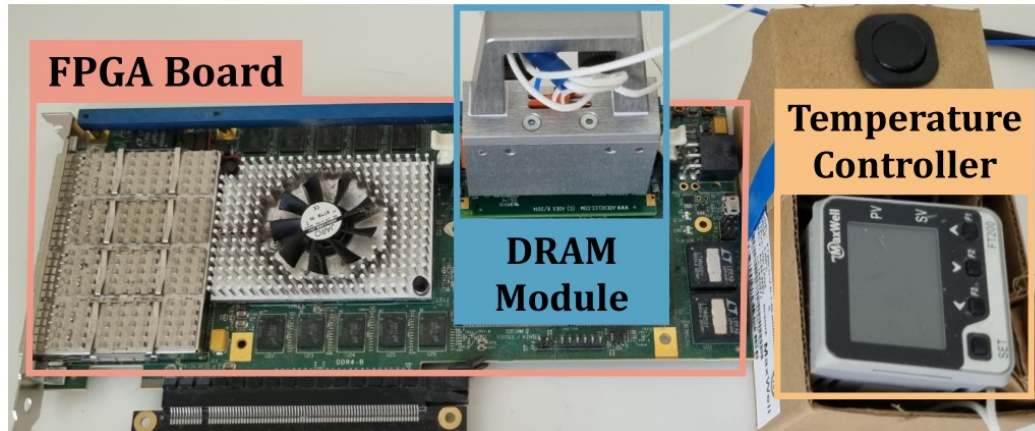
5-27× performance vs. a 16-core (64-thread) IBM POWER9 CPU

12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

FPGAs in Today's Systems: SoftMC

- An open-source FPGA-based infrastructure for experimental studies on DRAM
- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

github.com/CMU-SAFARI/SoftMC



Hassan+, “[SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies](#),” HPCA 2017.

SoftMC on Github

github.com/CMU-SAFARI/SoftMC

CMU-SAFARI / **SoftMC** Public

Watch 17

Fork 28

Star 82

[Code](#) [Issues 1](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#)

master

2 branches 0 tags

Go to file

Add file

Code

Hasan Hassan updating the prebuilt bitfile to be in-sync with the latest source co... 399b703 on Dec 15, 2017 8 commits

hw/boards/ML605	auto refresh control signal fix	5 years ago
prebuilt	updating the prebuilt bitfile to be in-sync with the latest source code;	4 years ago
sw	initial commit;	5 years ago
LICENSE	initial commit;	5 years ago
README.md	Update README.md	5 years ago

README.md

SoftMC v1.0

SoftMC is an experimental FPGA-based memory controller design that could be used to develop tests for DDR3 SODIMMs. SoftMC currently supports only the *Xilinx ML605* board. Soon, we will port SoftMC on other popularly used boards (e.g., *Xilinx VC709*).

About

SoftMC is an experimental FPGA-based memory controller design that can be used to develop tests for DDR3 SODIMMs using a C++ based API. The design, the interface, and its capabilities and limitations are discussed in our HPCA 2017 paper: "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies" <<https://...>

Readme

MIT License

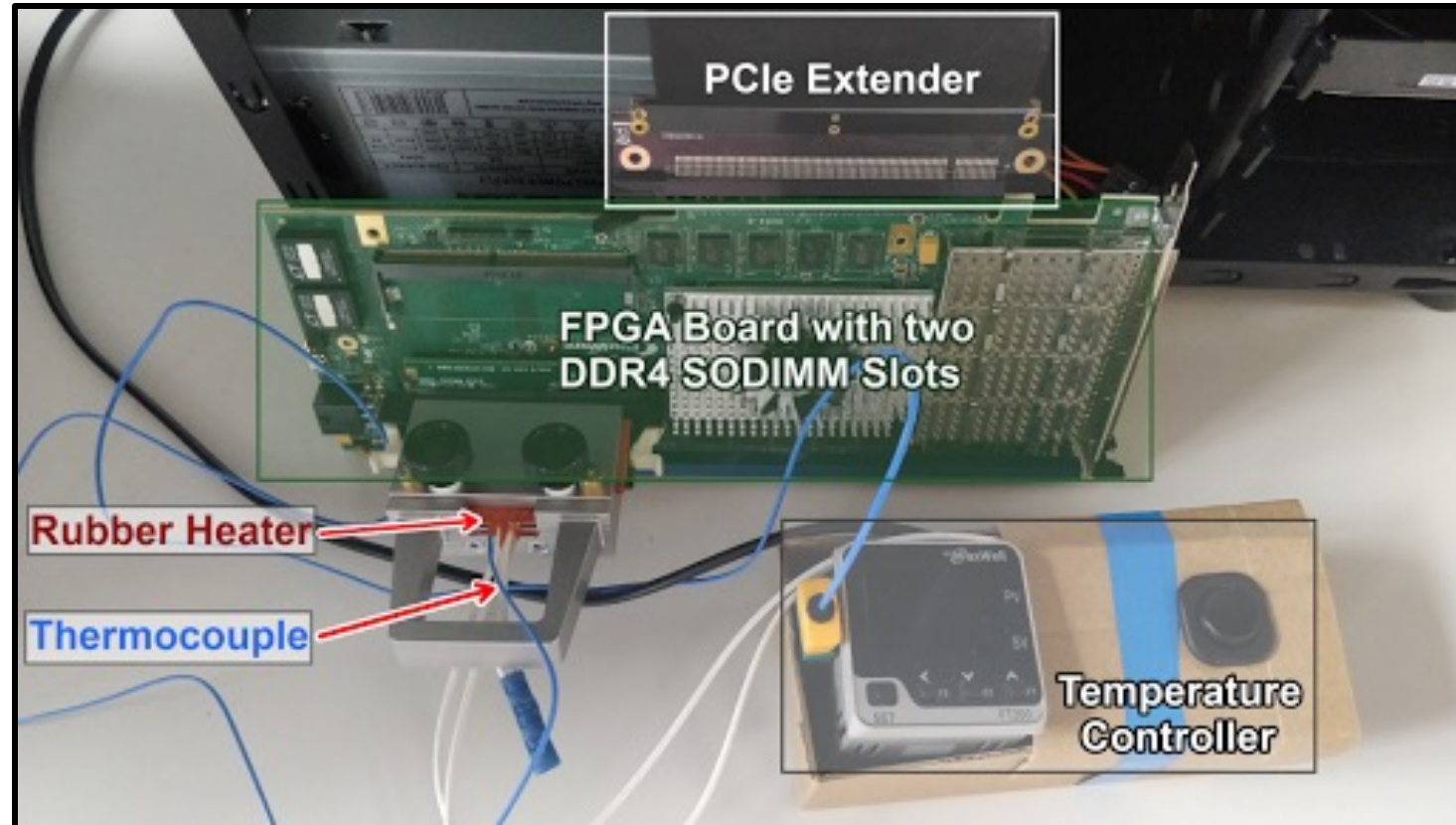
82 stars

17 watching

28 forks

Releases

SoftMC for DDR4



Orosa and Yaglikci+, ["A Deeper Look into RowHammer's Sensitivities..."](#), MICRO 2021
Hassan+, ["Uncovering In-DRAM RowHammer Protection Mechanisms..."](#), MICRO 2021

RowHammer: Eight Years Ago...

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,

**"Flipping Bits in Memory Without Accessing Them:
An Experimental Study of DRAM Disturbance Errors"**

*Proceedings of the 41st International Symposium on Computer Architecture (ISCA),
Minneapolis, MN, June 2014.*

[[Slides \(pptx\)](#)] [[pdf](#)] [[Lightning Session Slides \(pptx\)](#)] [[pdf](#)] [[Source Code and Data](#)]

[[Lecture Video](#)] (1 hr 49 mins), 25 September 2020]

One of the 7 papers of 2012-2017 selected as Top Picks in Hardware and Embedded Security for IEEE TCAD ([link](#)).

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Yoongu Kim¹ Ross Daly* Jeremie Kim¹ Chris Fallin* Ji Hye Lee¹
Donghyuk Lee¹ Chris Wilkerson² Konrad Lai Onur Mutlu¹

¹Carnegie Mellon University

²Intel Labs

RowHammer in 2021 (I)

- Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
**"A Deeper Look into RowHammer's Sensitivities:
Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"**
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Short Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#) (21 minutes)]
[[Lightning Talk Video](#) (1.5 minutes)]
[[arXiv version](#)]

A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa*
ETH Zürich

A. Giray Yağlıkçı*
ETH Zürich

Haocong Luo
ETH Zürich

Ataberk Olgun
ETH Zürich, TOBB ETÜ

Jisung Park
ETH Zürich

Hasan Hassan
ETH Zürich

Minesh Patel
ETH Zürich

Jeremie S. Kim
ETH Zürich

Onur Mutlu
ETH Zürich

RowHammer in 2021 (II)

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu, **"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"**
Proceedings of the 54th International Symposium on Microarchitecture (MICRO), Virtual, October 2021.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Short Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Lightning Talk Slides \(pptx\)](#)] [[pdf](#)]
[[Talk Video](#)] (25 minutes)
[[Lightning Talk Video](#)] (100 seconds)
[[arXiv version](#)]

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan[†]

[†]*ETH Zürich*

Yahya Can Tuğrul^{†‡}

Kaveh Razavi[†]
[‡]*TOBB University of Economics & Technology*

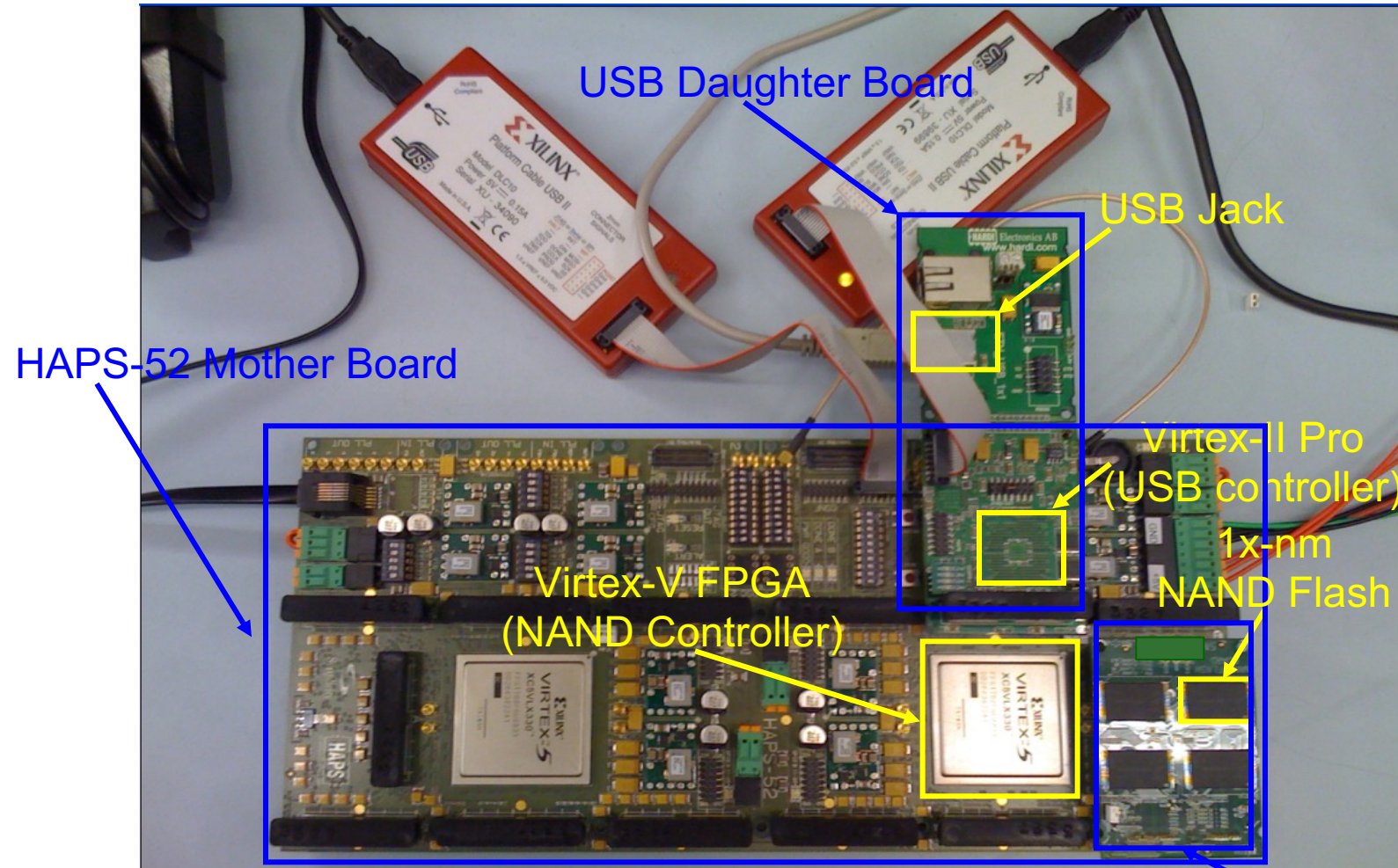
Jeremie S. Kim[†]

Onur Mutlu[†]

Victor van der Veen^σ

^σ*Qualcomm Technologies Inc.*

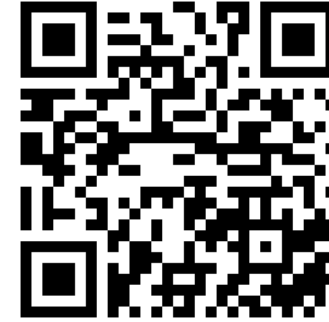
FPGAs in Today's Systems: Characterizing Flash Memories



[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017, PIEEE 2017, HPCA 2018, SIGMETRICS 2018]



Proceedings of the IEEE, Sept. 2017



Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

By YU CAI, SAUGATA GHOSE, ERICH F. HARATSCH, YIXIN LUO, AND ONUR MUTLU

<https://arxiv.org/pdf/1706.08642>

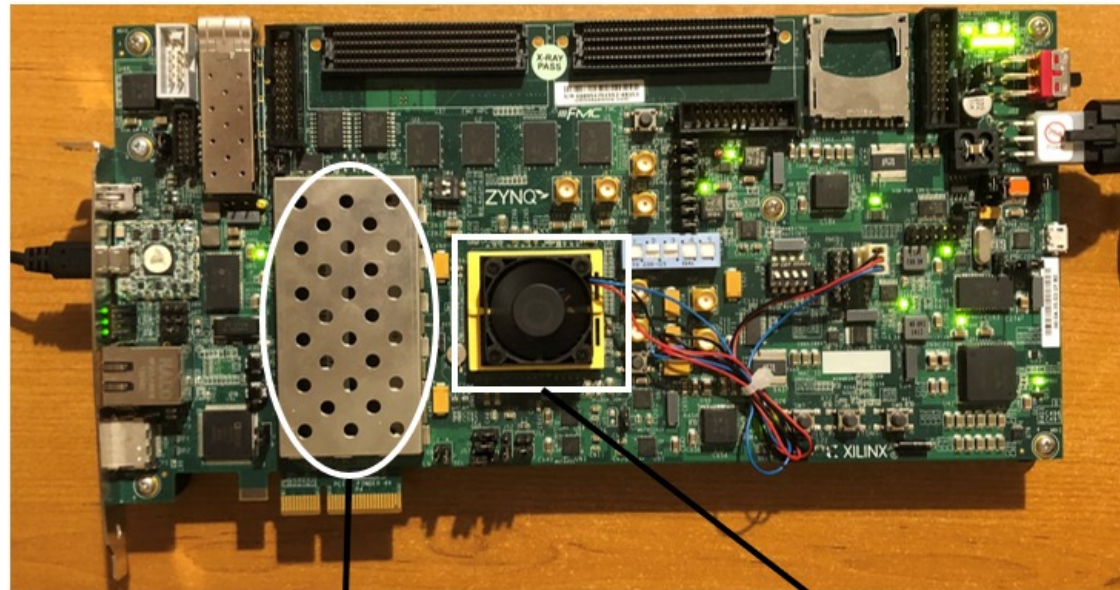
FPGAs for Processing-in-Memory Prototyping: PiDRAM (I)

Ataberk Olgun, Juan Gómez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oğuz Ergin, Onur Mutlu,

"PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM"

arXiv Preprint, November 2021.

[[Github Link](#)]



Compute-Enabled DIMM

RISC-V System


Real System Evaluation

In-DRAM **data copy**




In-DRAM **true random number generation**


FPGAs for Processing-in-Memory Prototyping: PiDRAM (II)

<https://github.com/CMU-SAFARI/PiDRAM>



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)


 **CMU-SAFARI / PiDRAM** Public

[Watch](#) 3 [Fork](#) 1 [Star](#) 16




[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[master](#) [2 branches](#) [0 tags](#)

[Go to file](#) [Add file](#) [Code](#)

 **olgunataberk** Fix small mistake in README

46522cc on Dec 5, 2021 11 commits

 controller-hardware	Add files via upload	4 months ago
 fpga-zynq	Adds instructions to reproduce two key results	3 months ago
 README.md	Fix small mistake in README	3 months ago

About

PiDRAM is the first flexible end-to-end framework that enables system integration studies and evaluation of real Processing-using-Memory techniques. Prototype on a RISC-V rocket chip system implemented on an FPGA. Described in our preprint: <https://arxiv.org/abs/2111.00082>

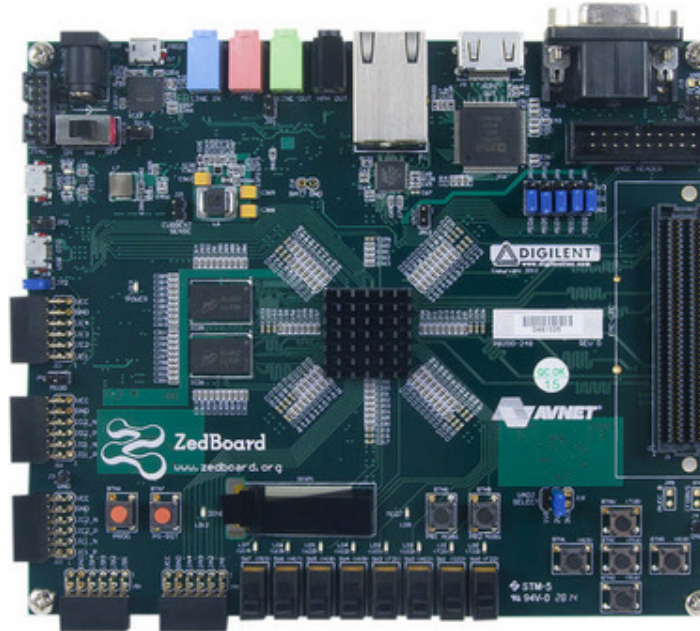
FPGAs for Prototyping New Architectures: MetaSys (I)

Nandita Vijaykumar, Ataberk Olgun, Konstantinos Kanellopoulos, Nisa Bostanci, Hasan Hassan, Mehrshad Lotfi, Phillip B Gibbons, Onur Mutlu,

"MetaSys: A Practical Open-Source Metadata Management System to Implement and Evaluate Cross-Layer Optimizations"

*To appear in Transactions on Architecture and Code Optimization (**TACO**), 2022.*


[[Github Link](#)]



Xilinx Zedboard

FPGAs for Prototyping New Architectures: MetaSys (II)


<https://github.com/CMU-SAFARI/MetaSys>

 Search or jump to... / Pull requests Issues Marketplace Explore

CMU-SAFARI / MetaSys Public Watch 3 Fork 2 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

 **olgunataberk** Update README.md e21ccd2 on Jul 9, 2021 12 commits

common	Initial commit	10 months ago
riscv-tools	Add tools directory	8 months ago
rocket-chip	Initial commit	10 months ago
testchipip	Initial commit	10 months ago
zedboard	Initial commit	10 months ago
LICENSE	Initial commit	10 months ago
README.md	Update README.md	8 months ago
metasys_readme.md	Update metasys_readme.md	8 months ago

About

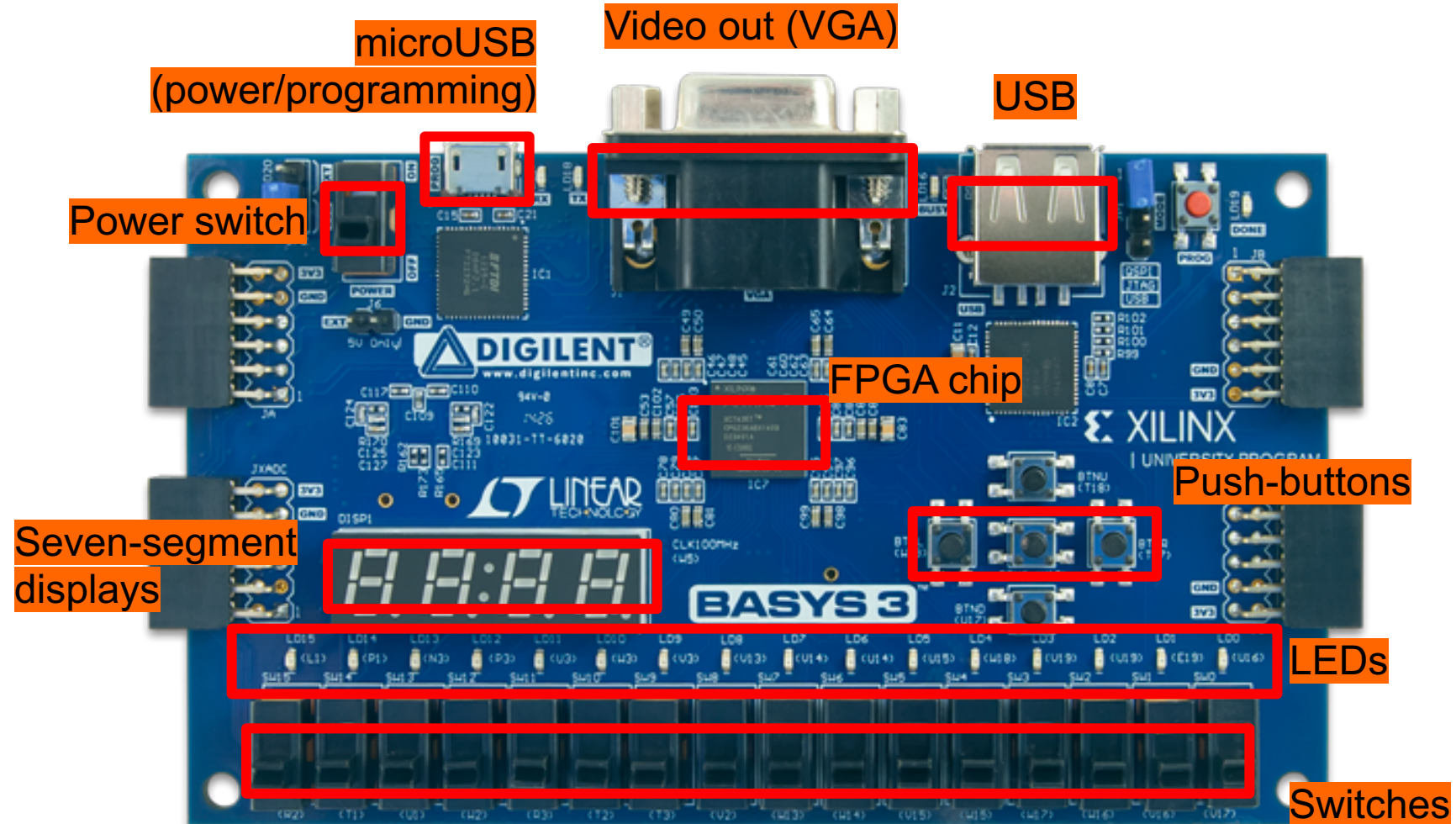
Metasys is the first open-source FPGA-based infrastructure with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer software/hardware cooperative techniques techniques in real hardware. Described in our pre-print: <https://arxiv.org/abs/2105.08123>

- Readme
- View license
- 0 stars
- 3 watching
- 2 forks

Agenda

- Logistics
- What Will We Learn?
- What is an FPGA?
- FPGAs in Today's Systems
- **Overview of the Lab Exercises**
- More about FPGAs
- Programming an FPGA
- Tutorial and Demo

Basys 3: Our FPGA Board



<https://reference.digilentinc.com/reference/programmable-logic/basys-3/start>

High Level Labs Summary

- At the end of the exercises, we will have built a 32-bit microprocessor running on the FPGA board
 - It will be a small processor, but it will be able to execute pretty much any program
- Each week we will have a new exercise
 - Not all exercises will require the FPGA board
- You are encouraged to experiment with the board on your own
 - We may have some extra boards for those who are interested – unlikely, but ask
 - It is not possible
to destroy the board **by programming!**

Lab 1: Drawing a Basic Circuit

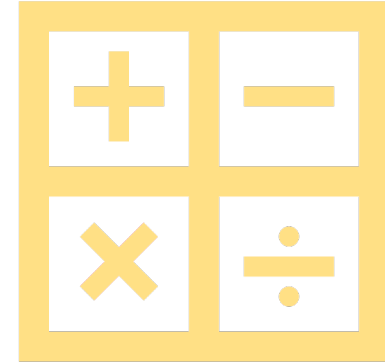
- **Comparison** is a common operation in software programming
 - We usually want to know the **relation** between two variables (e.g., $<$, $>$, $==$, ...)
- We will compare two *electrical signals* (inputs), and find whether they are same
 - The result (output) is also an *electrical signal*
- No FPGA programming involved
 - We encourage you to try later



Lab 2: Mapping Your Circuit to FPGA

- Another common operation in software programming?

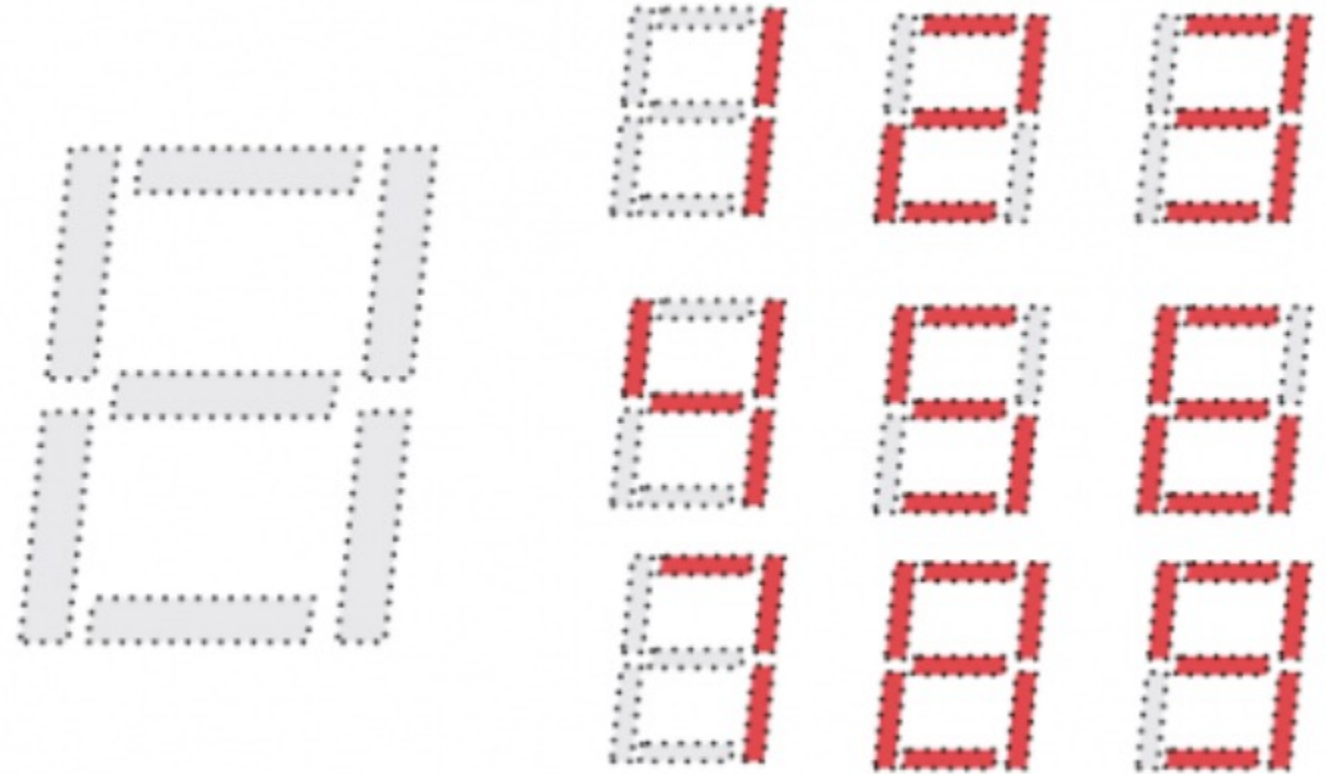
- Addition



- Design a circuit that adds two 1-bit numbers
- Reuse the 1-bit adder multiple times to perform 4-bit addition
- Implement the design on the FPGA board
 - Input: switches
 - Output: LEDs

Lab 3: Verilog for Combinatorial Circuits

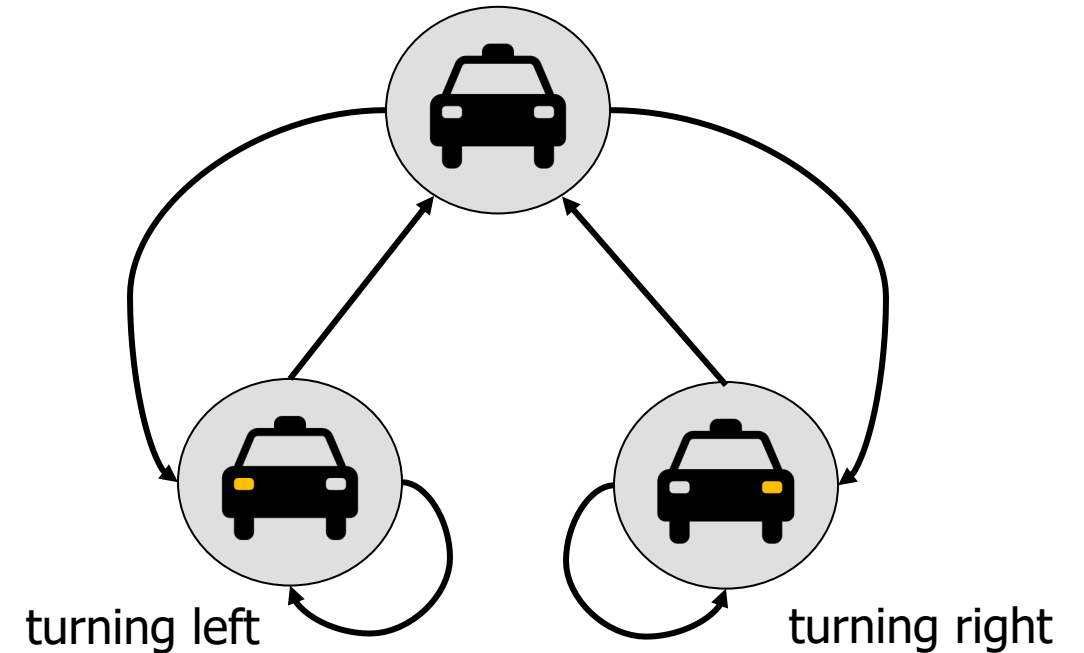
- Show your results from Lab 2 on a **Seven Segment Display**



<https://reference.digilentinc.com/reference/programmable-logic/basys-3/reference-manual>

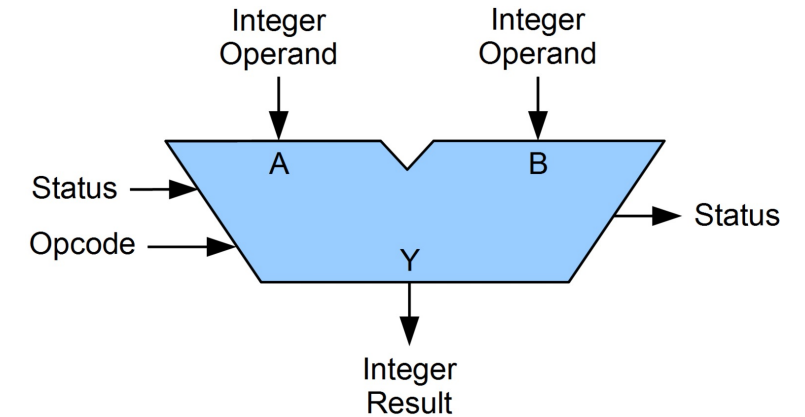
Lab 4: Finite State Machines

- Blinking LEDs for a car's turn signals
 - Implement and use **memories**
 - Change the blinking speed



Lab 5: Implementing an ALU

- Towards implementing your very first **processor**
- Implement your own **Arithmetic and Logic Unit (ALU)**
- An ALU is an important part of the CPU
 - ❑ Arithmetic operations: add, subtract, multiply, compare, ...
 - ❑ Logic operations: AND, OR, ...



Source:

https://en.wikipedia.org/wiki/Arithmetic_logic_unit

Lab 6: Testing the ALU

- **Simulate** your design from Lab 5
- Learn how to **debug** your implementation to resolve problems



Lab 7: Writing Assembly Code

- Programming in **assembly language**
 - MIPS
- Implement a program which you will later use to run on your processor
- Image manipulation

High-level code

```
int sum = 0;

for(int i = 0; i <= 10; i += 2)
{
    sum += i;
}
```

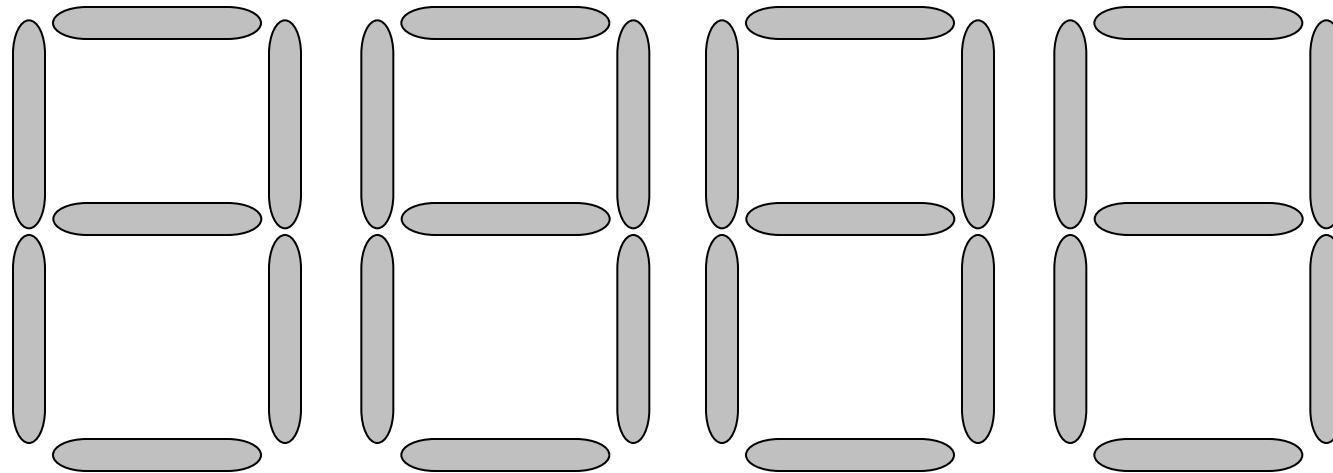
MIPS assembly

```
# i=$s0; sum=$s1

        addi $s0, $0, 0
        addi $s1, $0, 0
        addi $t0, $0, 12
loop:    beq  $s0, $t0, done
        add  $s1, $s1, $s0
        addi $s0, $s0, 2
        j    loop
done:
```

Lab 8: Full System Integration

- Will be covered in two weeks
- Learn how a processor is built
- Complete your first design of a MIPS processor
- Run a "snake" program



Lab 9: The Performance of MIPS

- Improve the **performance** of your processor from Lab 8 by adding new **instructions**
 - ❑ Multiplication
 - ❑ Bit shifting



Agenda

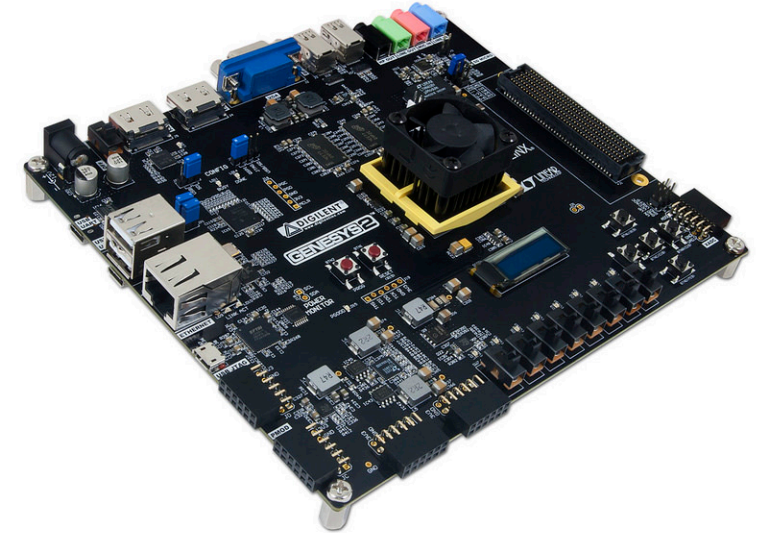
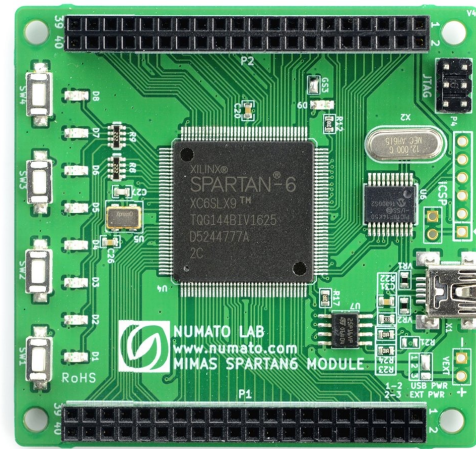
- Logistics
- What Will We Learn?
- What is an FPGA?
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- **More about FPGAs**
- Programming an FPGA
- Tutorial and Demo

What is an FPGA?

- Field Programmable Gate Array: FPGA



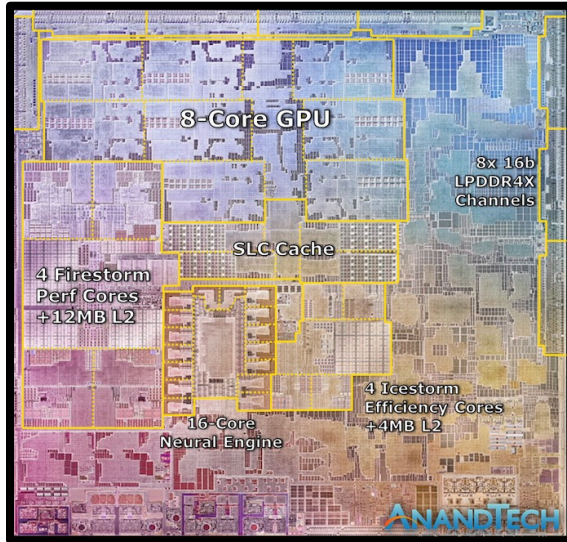
© Raimond Spekking / [CC BY-SA 4.0](#) (via Wikimedia Commons)



- FPGA is a **software-reconfigurable** hardware substrate
 - ❑ Reconfigurable **functions**
 - ❑ Reconfigurable **interconnection** of functions
 - ❑ Reconfigurable **input/output (IO)**

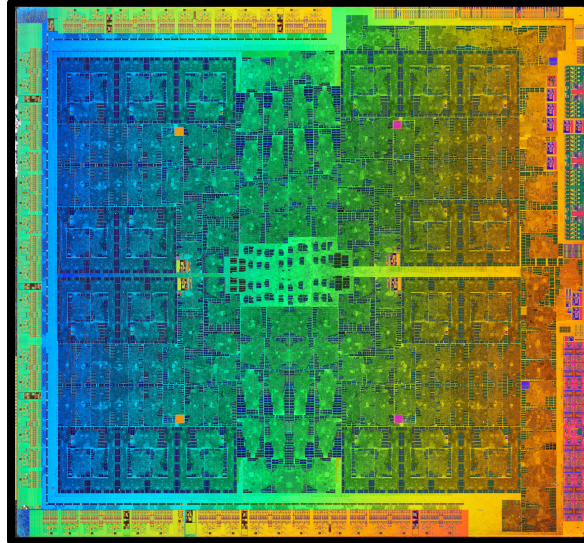
FPGAs & Other Integrated Circuits

CPU_s



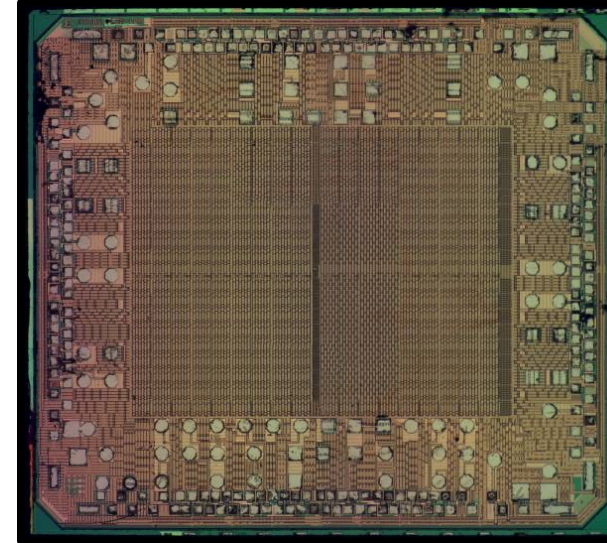
Apple M1

GPU_s



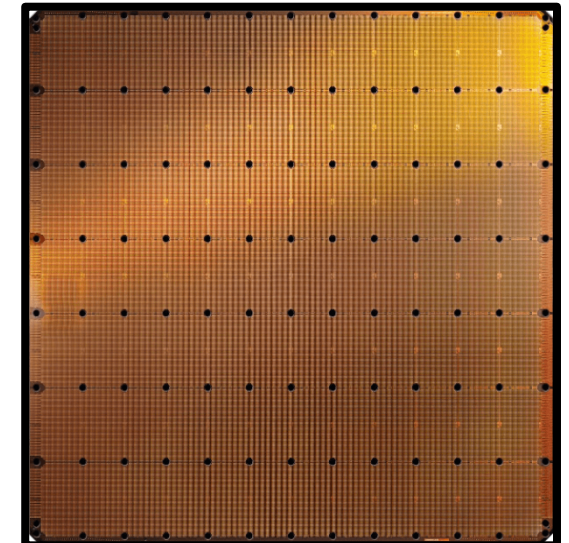
Nvidia GTX 1070

FPGA_s



Xilinx Spartan

ASIC_s



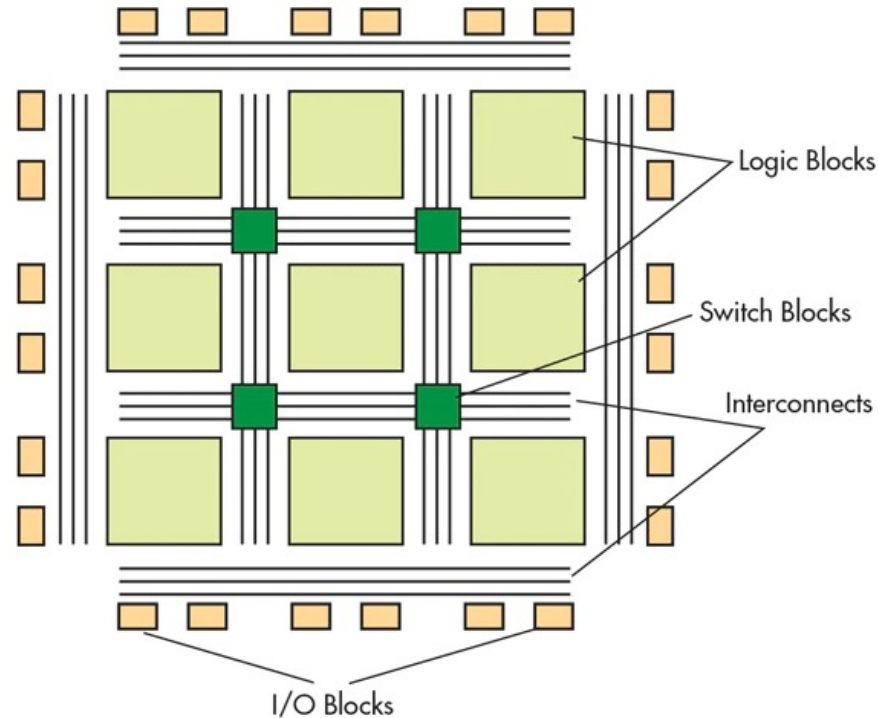
Cerebras WSE-2

**Flexibility
Programming Ease**

Efficiency



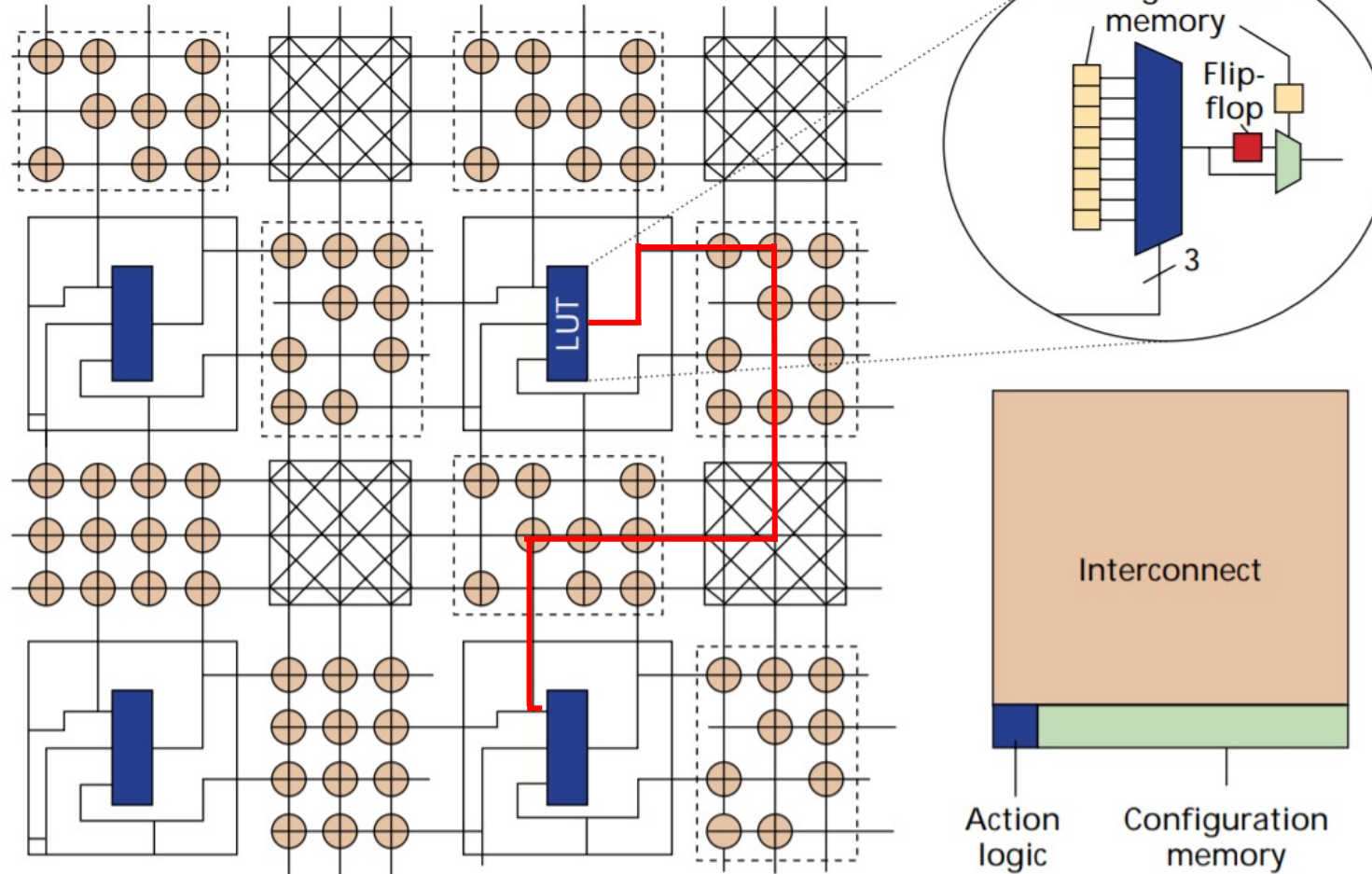
A High-Level Overview of FPGAs



We **configure** logic blocks, their connections, and IO blocks to create hardware circuits and map programs onto those circuits

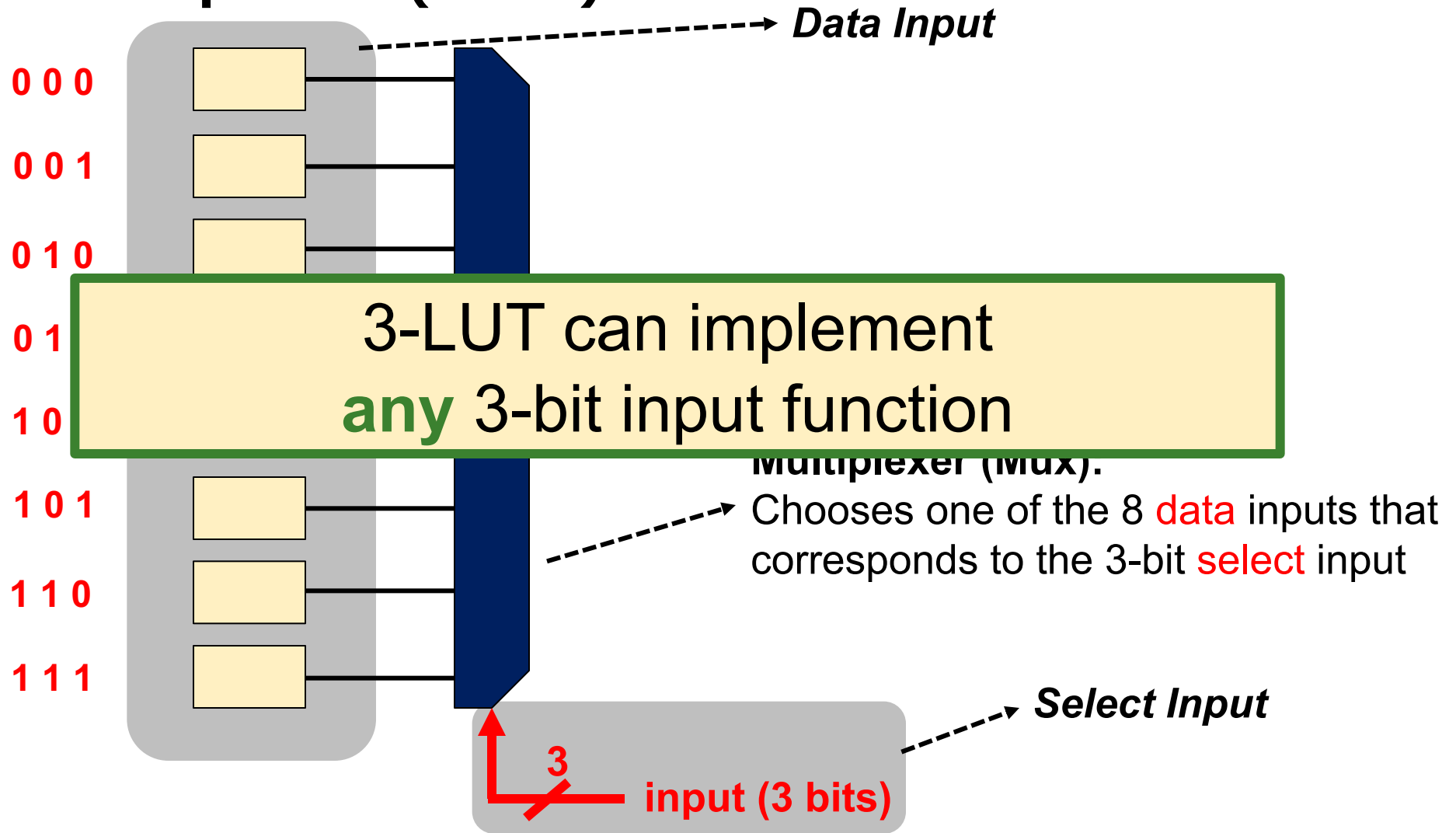
FPGA Architecture - Looking Inside an FPGA

- Two main building blocks:
 - Look-Up Tables (LUT) and Switches



How Do We Program LUTs?

■ 3-bit input LUT (3-LUT)



An Example of Programming a LUT

- Let's implement a function that outputs '1' when there are at least two '1's in a 3-bit input

In C:

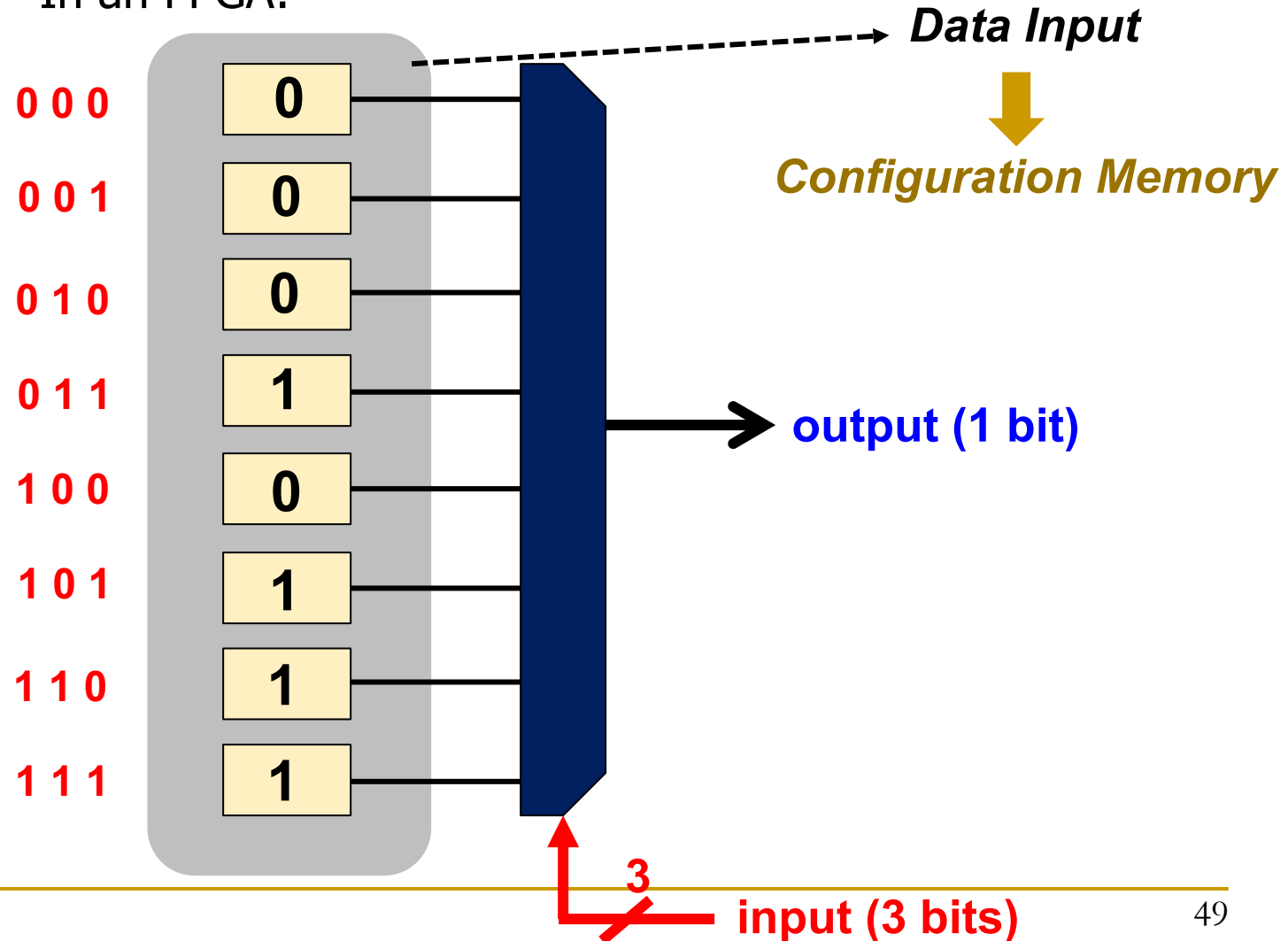
```
int count = 0;
for(int i = 0; i < 3; i++) {
    count += input & 1;
    input = input >> 1;
}

if(count > 1) return 1;

return 0;
```

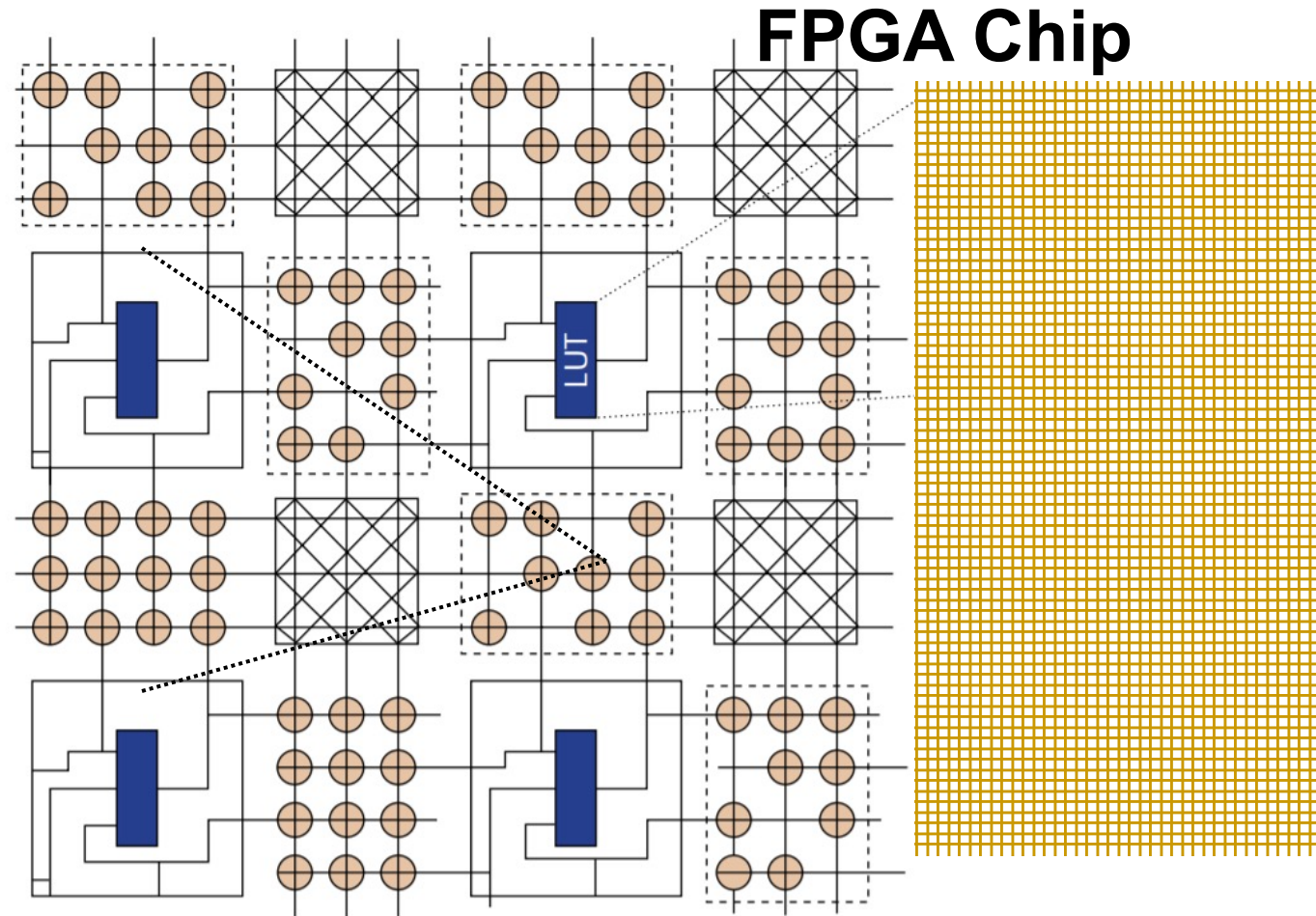
```
switch(input){
    case 0:
    case 1:
    case 2:
    case 4:
        return 0;
    default:
        return 1;}
```

In an FPGA:



How to Implement Complex Functions?

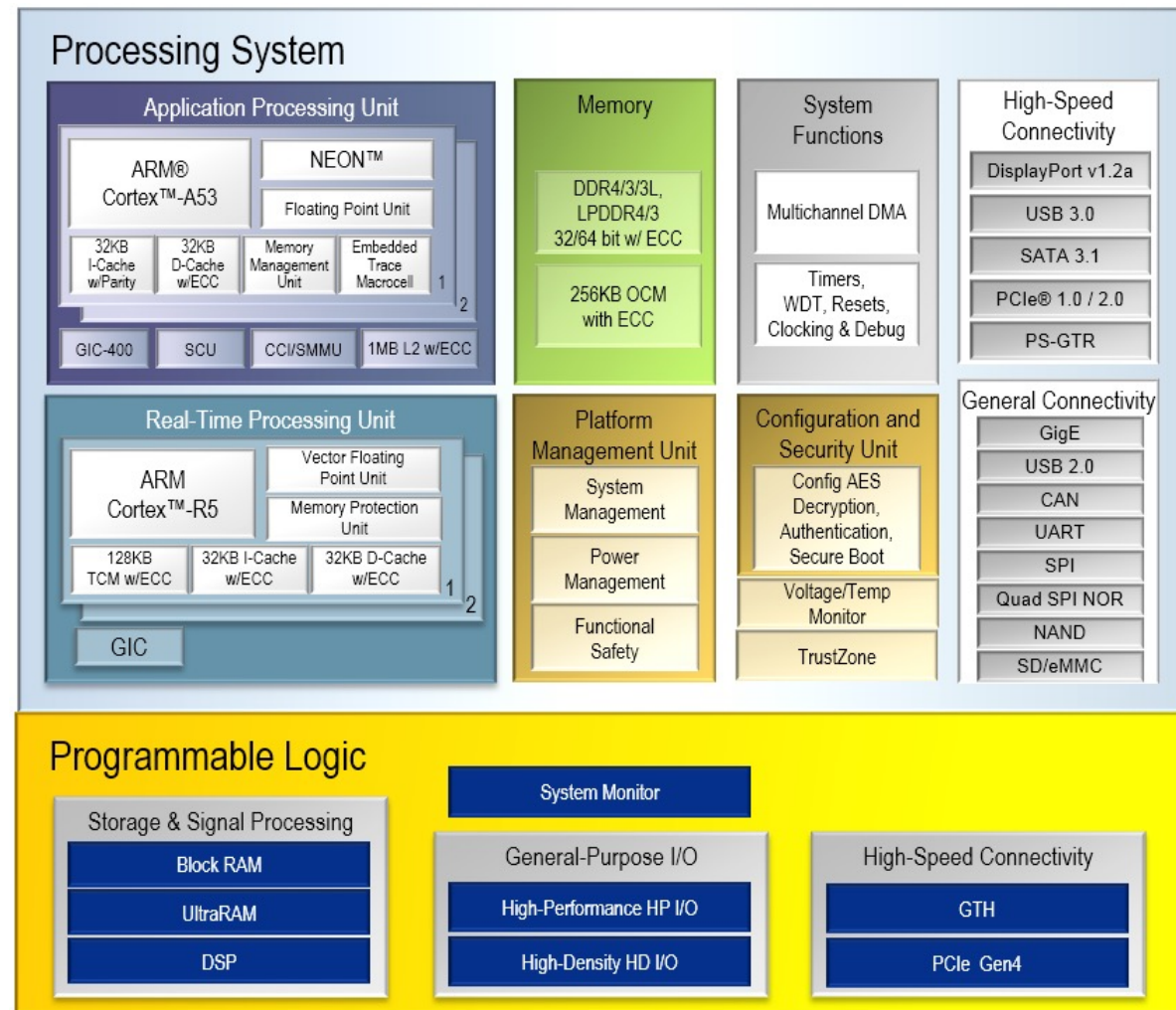
- FPGAs are composed of **many** LUTs and switches



Modern FPGA Architectures

- Typically use LUTs with 6-bit select input (6-LUT)
 - Thousands of them
- MegaBytes of distributed on-chip memory
- Hard-coded special-purpose hardware blocks for high-performance operations
 - Memory interface
 - Low latency and high bandwidth off-chip I/O
 - ...
- Even a general-purpose processor embedded within the FPGA chip

An Example Modern FPGA Platform: Xilinx Zynq Ultrascale+

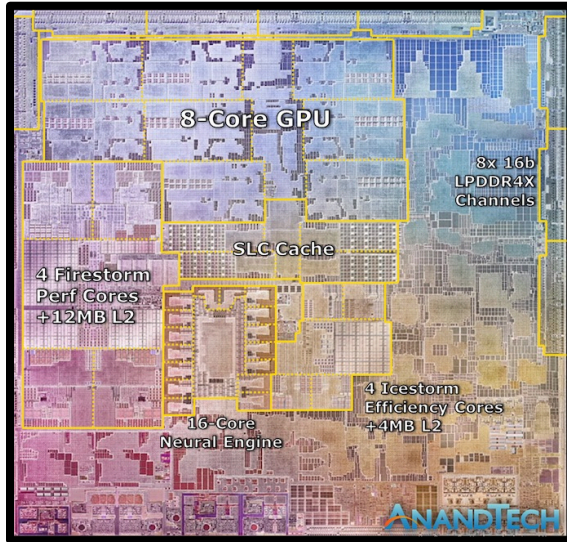


<https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html>

<https://www.pressebox.com/pressrelease/enclustra-gmbh/Xilinx-Zynq-UltraScale-high-bandwidth-MPSoc-module/boxid/934771>

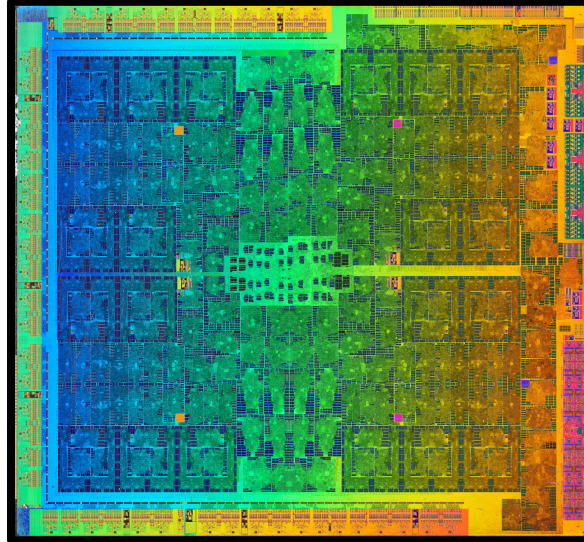
Advantages & Disadvantages of FPGAs (I)

CPU_s



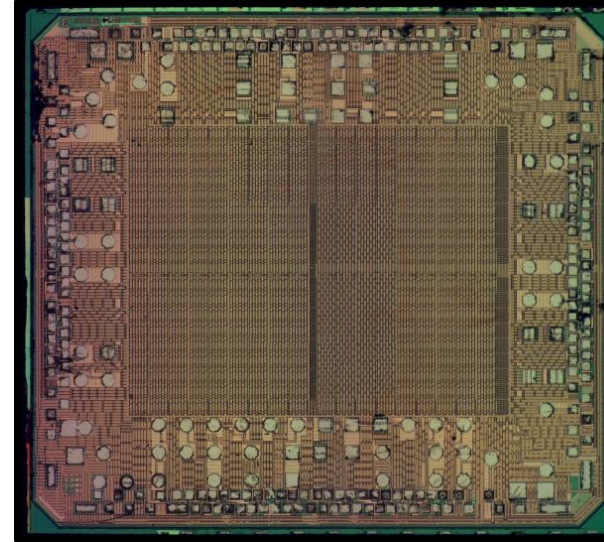
Apple M1

GPU_s



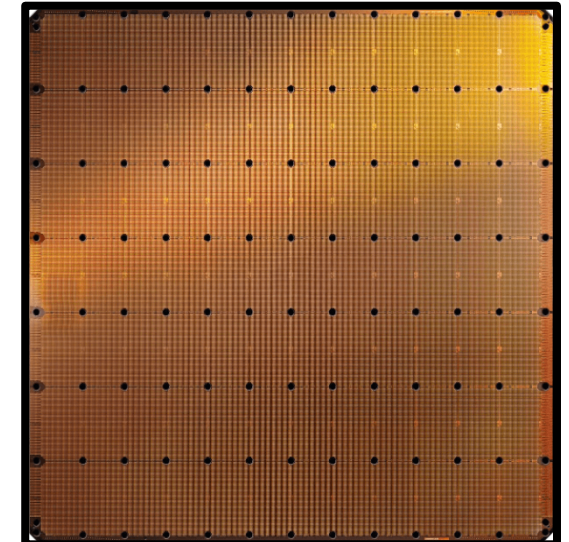
Nvidia GTX 1070

FPGA_s



Xilinx Spartan

ASIC_s



Cerebras WSE-2

**Flexibility
Programming Ease**

Efficiency



Advantages & Disadvantages of FPGAs (II)

■ Advantages

- ❑ An algorithm can be implemented directly in hardware
 - No ISA, high specialization → high performance, high energy efficiency
- ❑ Low development cost (vs. a custom hardware design)
- ❑ Short time to market (vs. a custom hardware design)
 - Reconfigurable in the field
- ❑ Usable and reusable for many purposes
 - Good for both prototyping and application acceleration

■ Disadvantages

- ❑ Not as **fast** and **power efficient** as *dedicated hardware customized for an algorithm*
- ❑ Reconfigurability comes at a cost: significant **area** and **latency overhead**

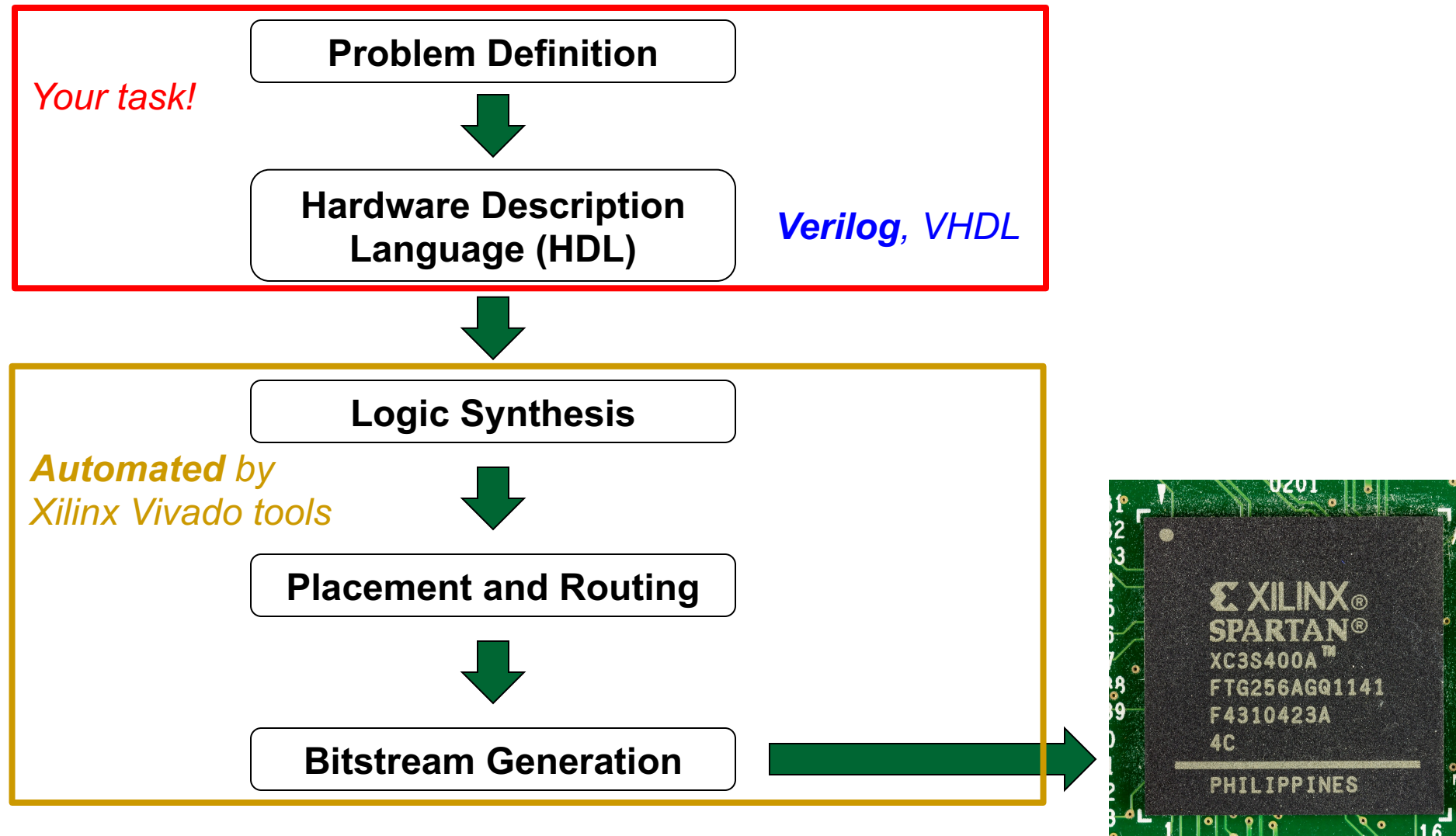
Agenda

- Logistics
- What Will We Learn?
- What is an FPGA?
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- More about FPGAs
- **Programming an FPGA**
- Tutorial and Demo

Computer-Aided Design (CAD) Tools

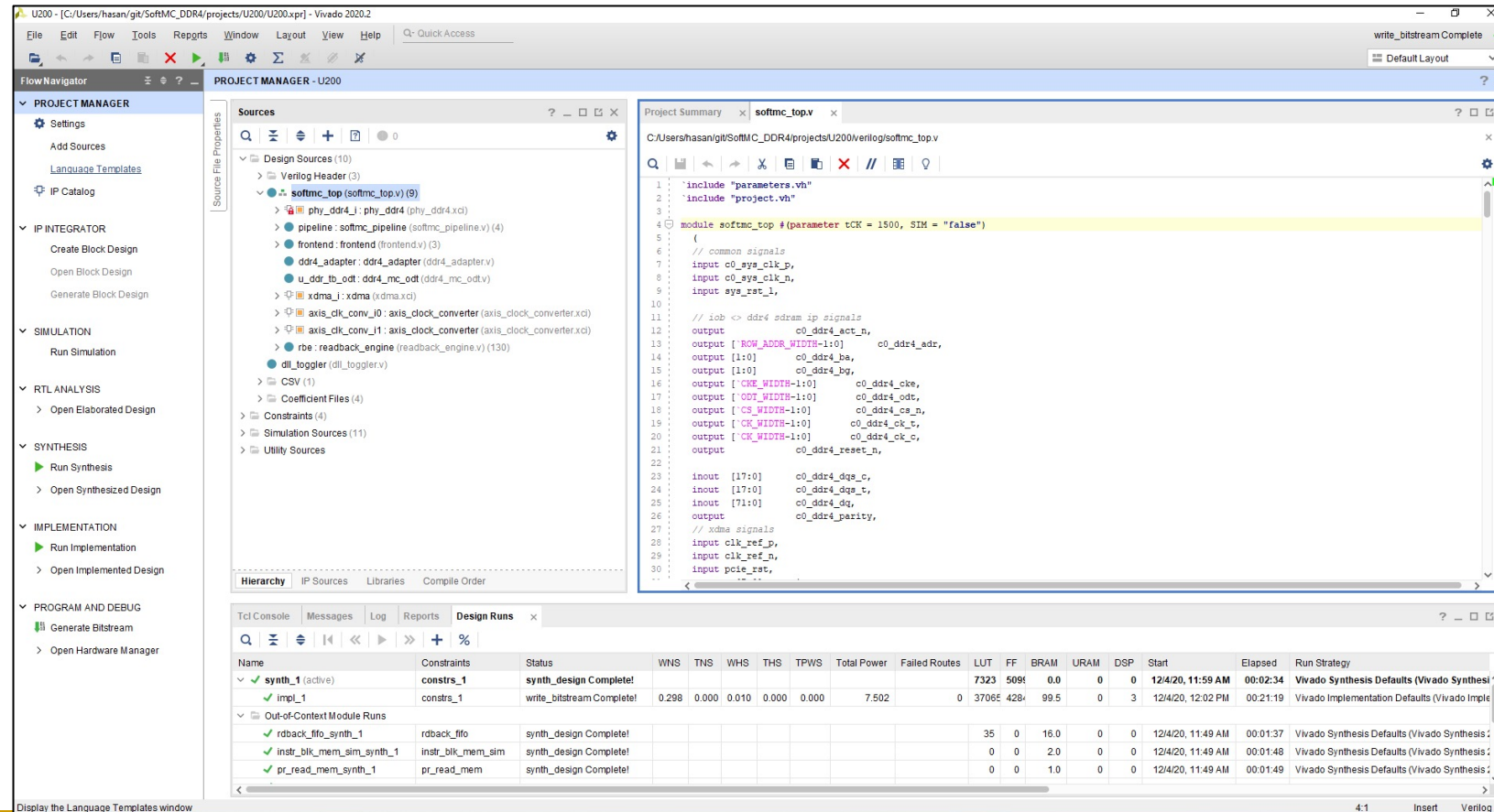
- FPGAs have many **resources** (e.g., LUTs, switches)
- They are **hard** to program manually
- How can we
 - represent a **high-level functional description** of our hardware circuit using the FPGA resources?
 - select the resources to **map** our circuit to?
 - **optimally configure the interconnect** between the selected resources?
 - generate a final **configuration file** to properly configure an FPGA?

FPGA Design Flow



Vivado

- A software tool that helps us throughout the FPGA design flow
- Provides tools to **simulate** our designs
 - ❑ Validate the correctness of the implementation
 - ❑ **Debugging**
- Provides drivers and graphical interface to easily **program** the FPGA using a USB cable
- **Installed** in computer rooms in HG (E 19, E 26.1, E 26.3, E 27)




Tutorial and Demo

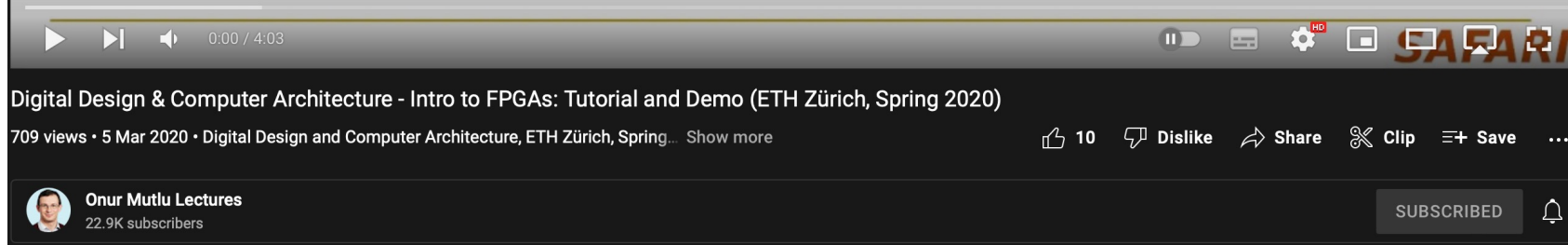
- We will see how to
 - use Vivado to write Verilog code
 - follow the FPGA design flow steps
 - download the bitstream into the FPGA
- Simple Keyboard
 - A simple hardware that you can easily develop by the end of semester
- Link to source files
<https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-keyboard-demo/start>

Simple Keyboard Demo

Introduction to FPGAs: Tutorial and Demo

In this video, we will show how to **program** a **Basys 3 FPGA board** using an example project that **interfaces a keyboard** and **sends the ASCII code of the pressed key**  a serial interface to a computer

<https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-keyboard-demo/start>



Digital Design & Computer Architecture - Intro to FPGAs: Tutorial and Demo (ETH Zürich, Spring 2020)

709 views • 5 Mar 2020 • Digital Design and Computer Architecture, ETH Zürich, Spring... Show more

Onur Mutlu Lectures
22.9K subscribers

SUBSCRIBED

Agenda

- Logistics
- What Will We Learn?
- What is an FPGA?
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- More about FPGAs
- Programming an FPGA
- Tutorial and Demo

Digital Design & Computer Arch.

Lecture 3b: Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Ataberk Olgun)
ETH Zurich
Spring 2022
3 March 2022