Digital Design & Computer Arch.

Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Ataberk Olgun)
ETH Zurich
Spring 2023
5 March 2023

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

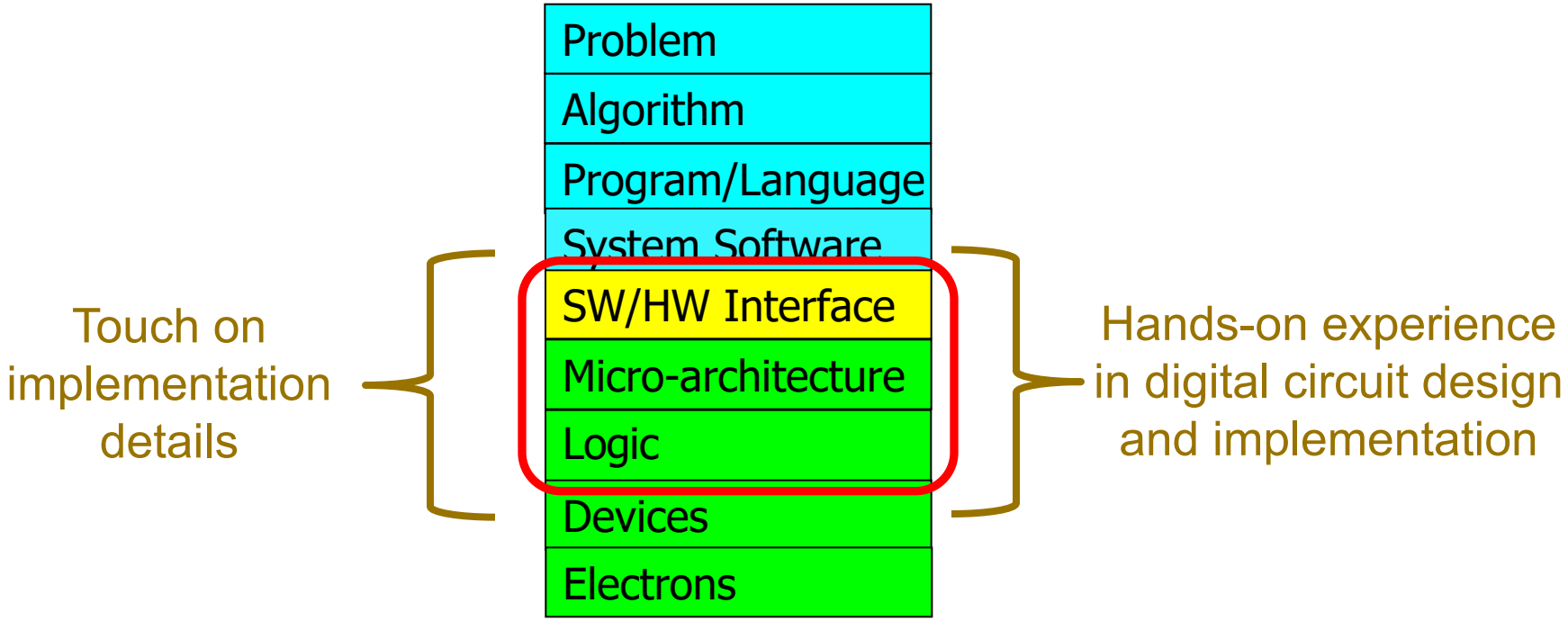
- **10** labs, **30** points in total
- We will put the lab manuals online
  - https://safari.ethz.ch/digitaltechnik/spring2023/doku.php?id=labs
- Grading Policy
  - In-class evaluation (70%) and *mandatory* lab reports (30%)
    - *1-point penalty for late submission of the report*
  - You should finish 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/grade/report/grader/index.php?id=16852
- 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

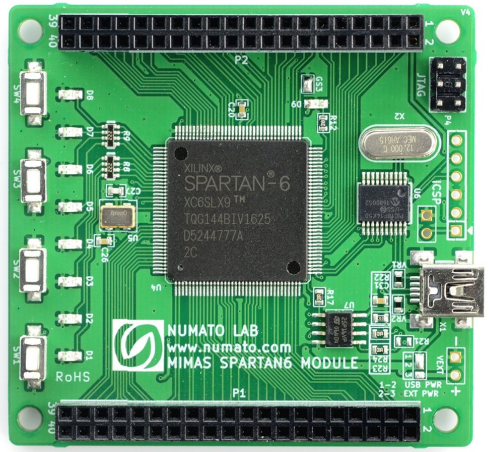
- Hands-on experience on:
  - Hardware Prototyping on Field Programmable Gate Arrays (FPGAs)
  - Debugging Your Hardware Implementation
  - Hardware Description Language (HDL)
  - Hardware Design Flow
  - Computer-Aided Design (CAD) Tools

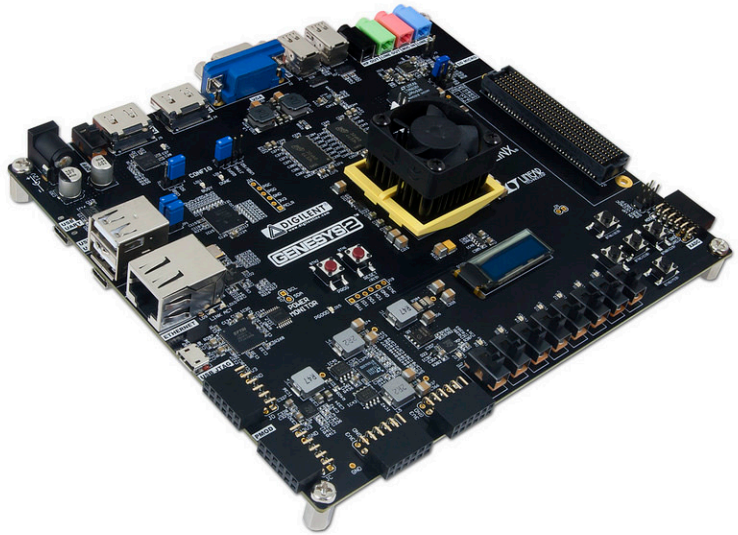
Agenda

- 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

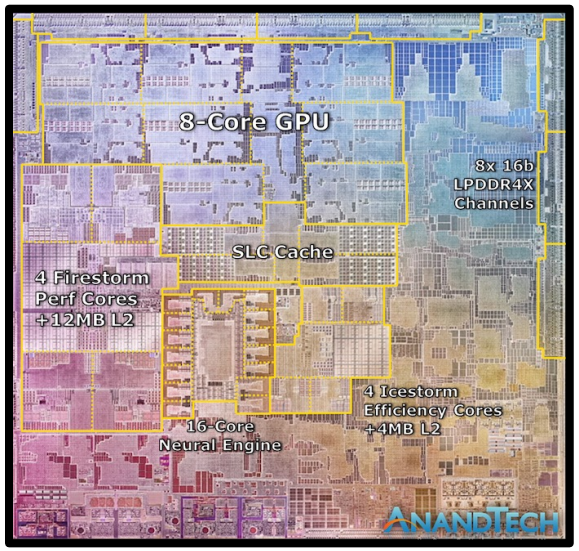




- FPGA is a software-reconfigurable hardware substrate
  - Reconfigurable functions
  - Reconfigurable interconnection of functions
  - Reconfigurable input/output (IO)

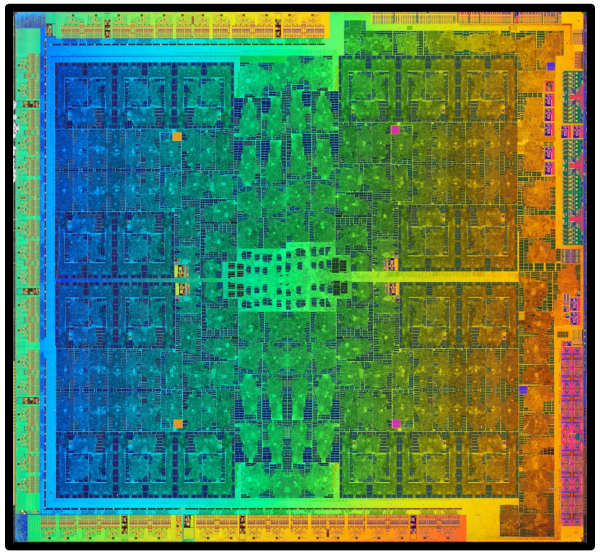
FPGAs & Other Integrated Circuits

CPUs



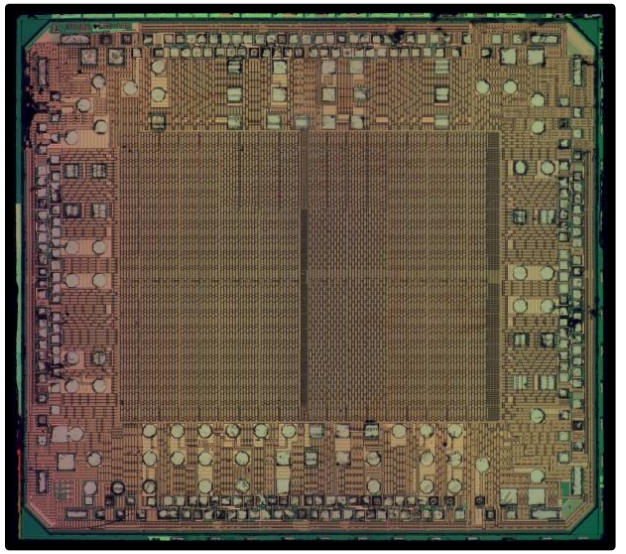
Apple M1

GPUs



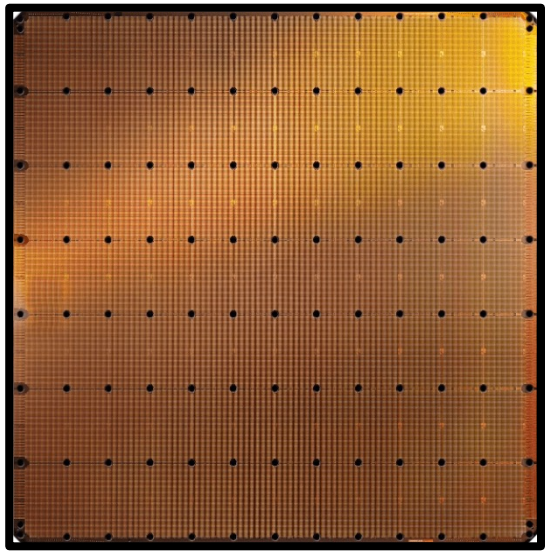
Nvidia GTX 1070

FPGAs



Xilinx Spartan

ASICs



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

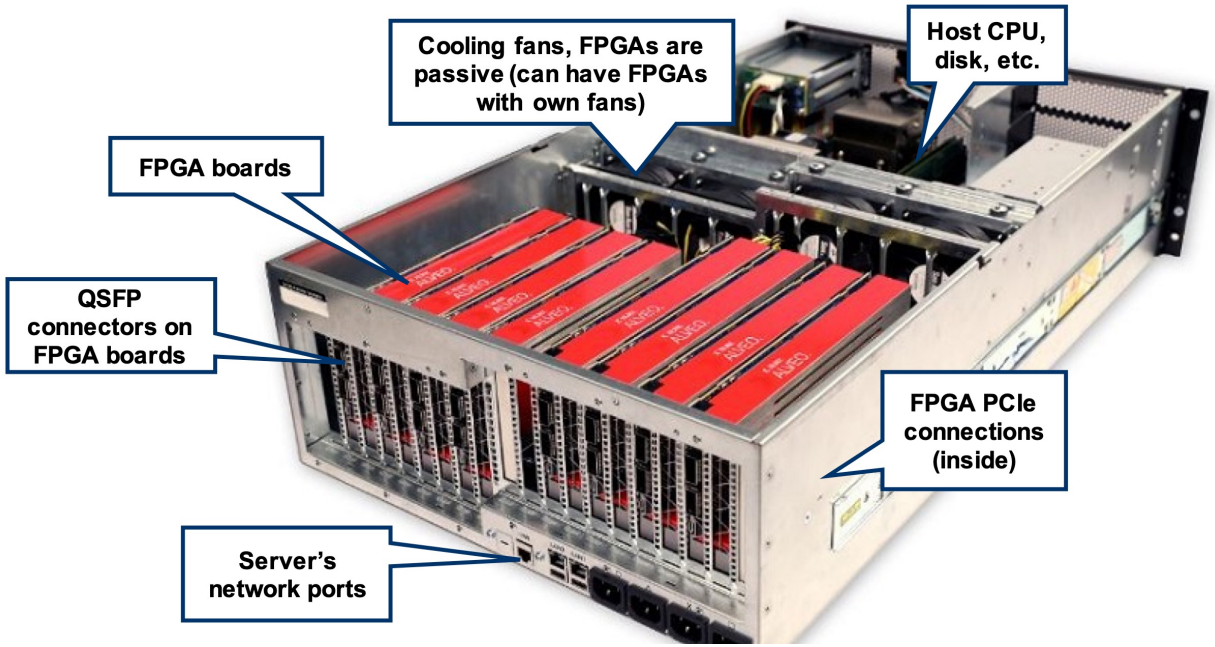
FPGAs in Today's Systems: Project Brainwave

- "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

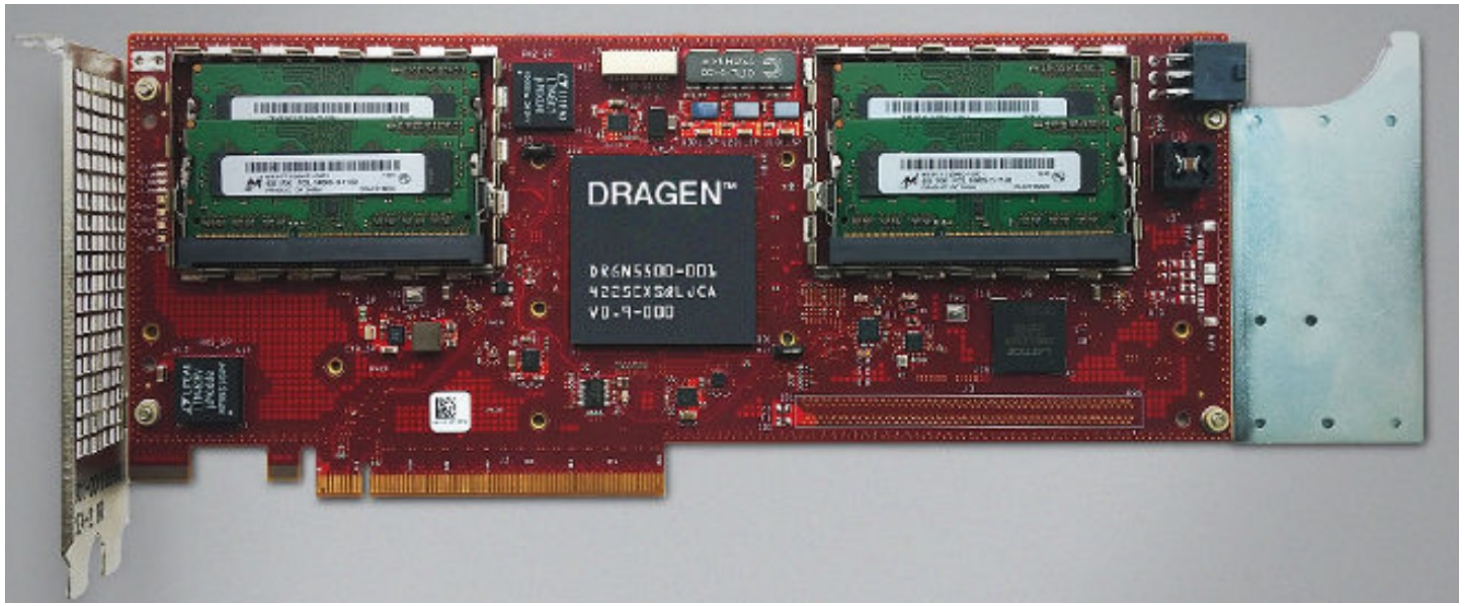
- "Amazon EC2 F1 instances use **FPGAs** to enable delivery of **custom hardware accelerations**. F1 instances are *easy to program* and come with everything you need to develop, simulate, debug, and compile your hardware acceleration code, including an FPGA Developer AMI and supporting hardware level 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.

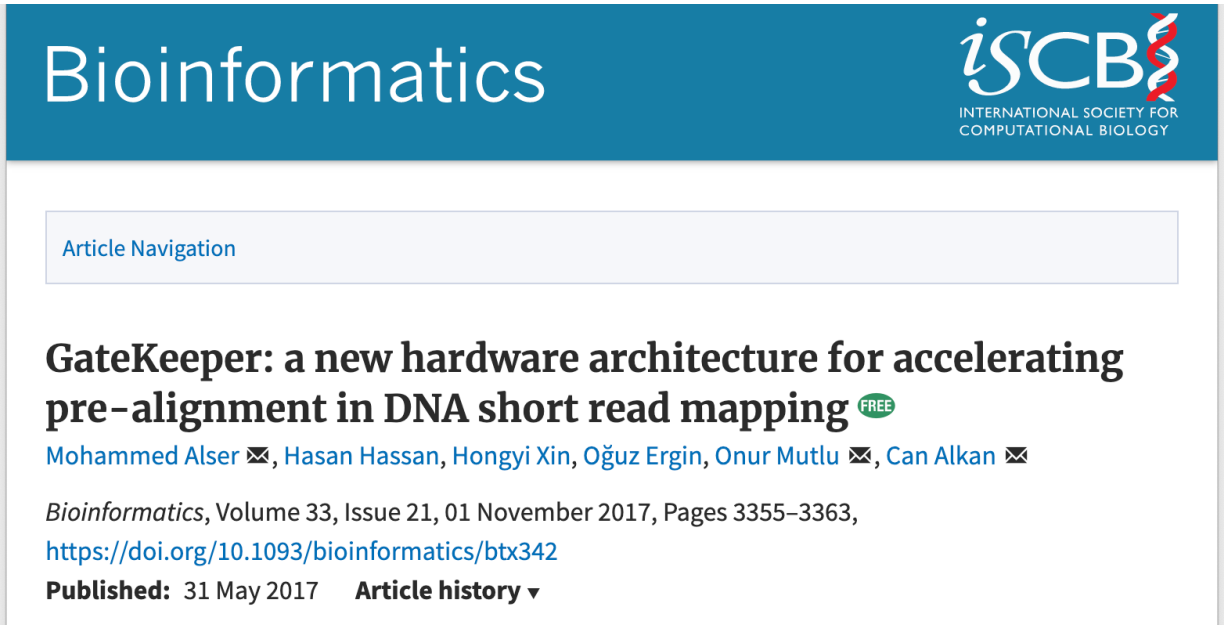


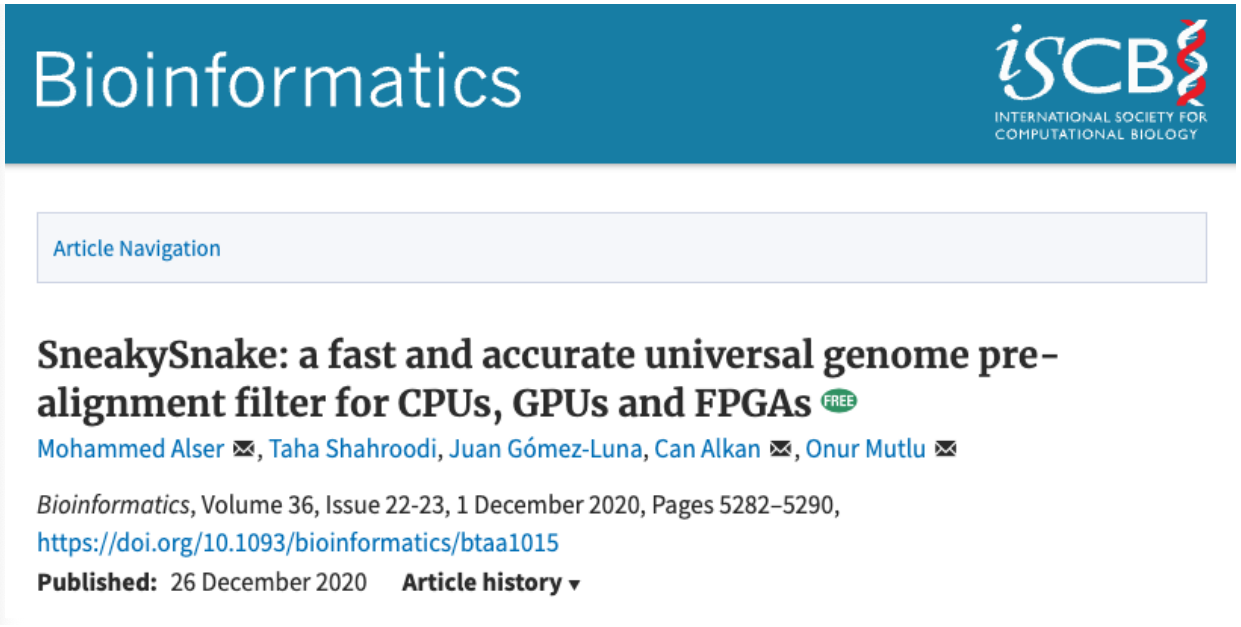
Illumina DRAGEN (Dynamic Read Analysis for GENomics) Bio-IT Platform

Illumina NextSeq 2000

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

FPGAs in Today's Systems: More Bioinformatics



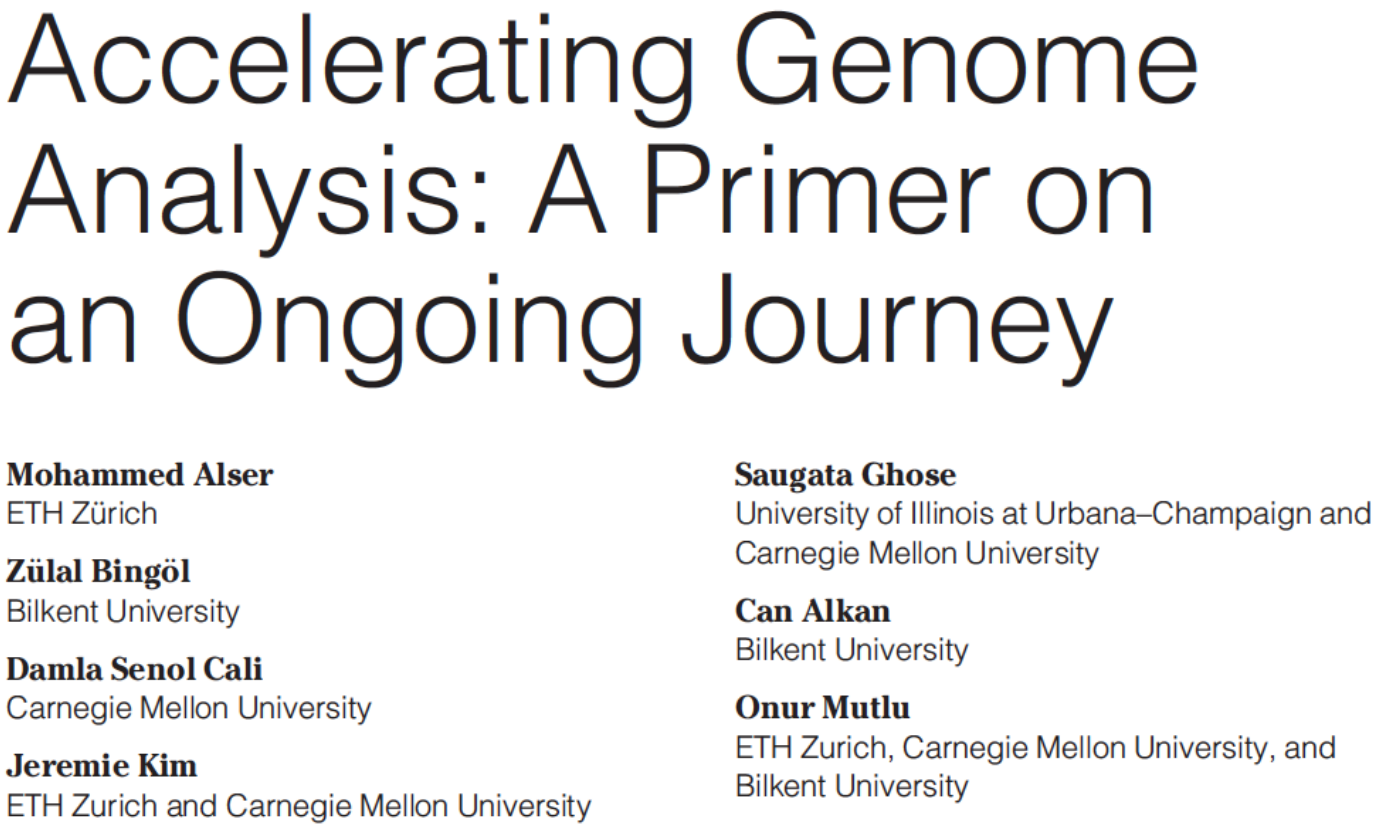


Alser+, **"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"**, Bioinformatics, 2017.

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, Zulal Bingol, 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

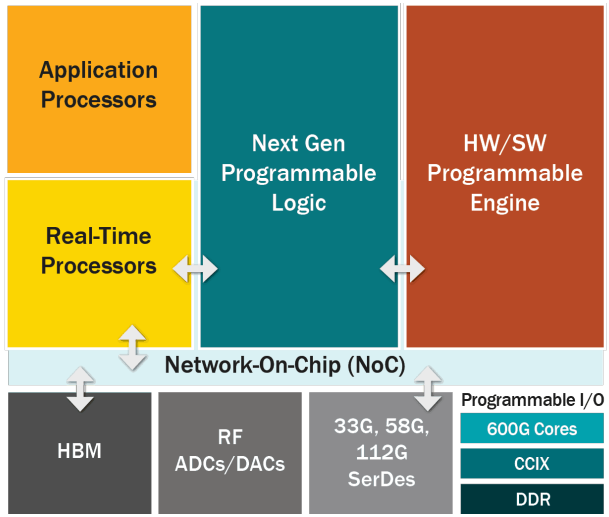
Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University

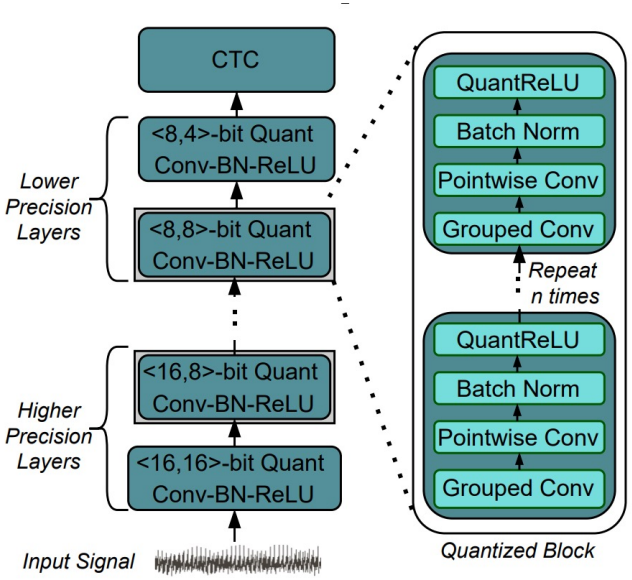
FPGAs in Today's Systems: Genomic Basecaller (I)

A Framework for Designing Efficient Deep Learning-Based Genomic Basecallers

Gagandeep Singh[a] Mohammed Alser*[a] Alireza Khodamoradi*[b]
Kristof Denolf[b] Can Firtina[a] Meryem Banu Cavlak[a]
Henk Corporaal[c] Onur Mutlu[a]
[a]ETH Zürich [b]AMD [c]Eindhoven University of Technology



+



=

~16X basecalling speedup

FPGAs in Today's Systems: Genomic Basecaller (II)

arxiv.org/abs/2211.03079

Computer Science > Hardware Architecture

[Submitted on 6 Nov 2022 (v1), last revised 8 Dec 2022 (this version, v3)]

A Framework for Designing Efficient Deep Learning-Based Genomic Basecallers

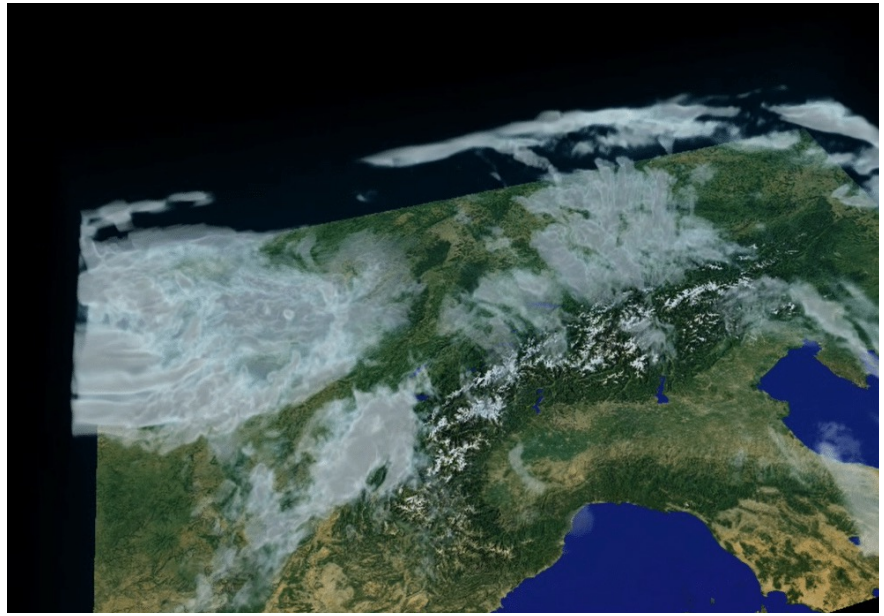
Gagandeep Singh, Mohammed Alser, Alireza Khodamoradi, Kristof Denolf, Can Firtina, Meryem Banu Cavlak, Henk Corporaal, Onur Mutlu

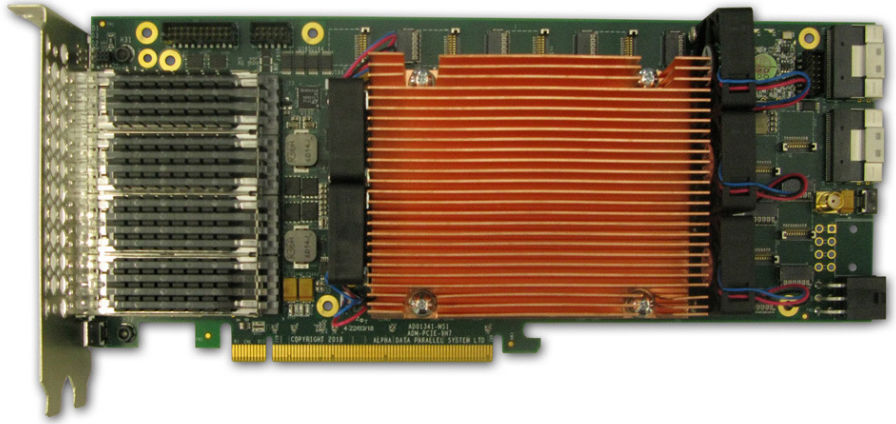
Nanopore sequencing generates noisy electrical signals that need to be converted into a standard string of DNA nucleotide bases using a computational step called basecalling. The accuracy and speed of basecalling have critical implications for all later steps in genome analysis. Many researchers adopt complex deep learning-based models to perform basecalling without considering the compute demands of such models, which leads to slow, inefficient, and memory-hungry basecallers. Therefore, there is a need to reduce the computation and memory cost of basecalling while maintaining accuracy. Our goal is to develop a comprehensive framework for creating deep learning-based basecallers that provide high efficiency and performance. We introduce RUBICON, a framework to develop hardware-optimized basecallers. RUBICON consists of two novel machine-learning techniques that are specifically designed for basecalling. First, we introduce the first quantization-aware basecalling neural architecture search (QABAS) framework to specialize the basecalling neural network architecture for a given hardware acceleration platform while jointly exploring and finding the best bit-width precision for each neural network layer. Second, we develop SkipClip, the first technique to remove the skip connections present in modern basecallers to greatly reduce resource and storage requirements without any loss in basecalling accuracy. We demonstrate the benefits of RUBICON by developing RUBICALL, the first hardware-optimized basecaller that performs fast and accurate basecalling. Compared to the fastest state-of-the-art basecaller, RUBICALL provides a 3.19x speedup with 2.97% higher accuracy. We show that RUBICON helps researchers develop hardware-optimized basecallers that are superior to expert-designed models.

Subjects: **Hardware Architecture (cs.AR)**; Distributed, Parallel, and Cluster Computing (cs.DC); Genomics (q-bio.GN)
Cite as: arXiv:2211.03079 **[cs.AR]**
(or arXiv:2211.03079v3 **[cs.AR]** for this version)
https://doi.org/10.48550/arXiv.2211.03079

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*

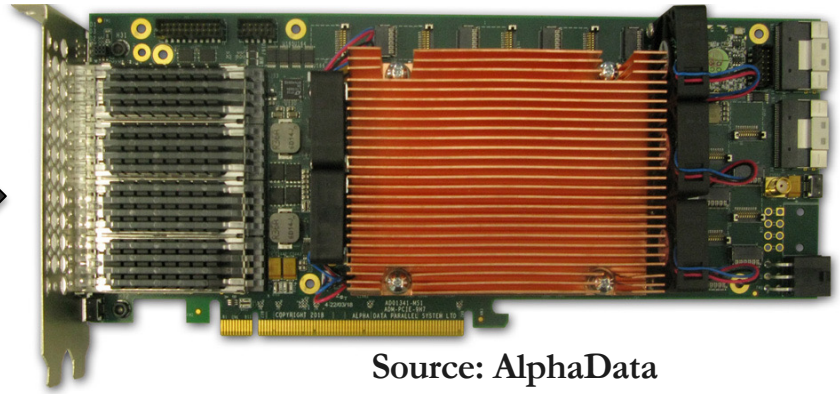
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

Near-Memory Acceleration using FPGAs

Source: IBM

IBM POWER9 CPU

OCAPI



Source: AlphaData

HBM-based FPGA board

FPGA-based Near-Memory Accelerator

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

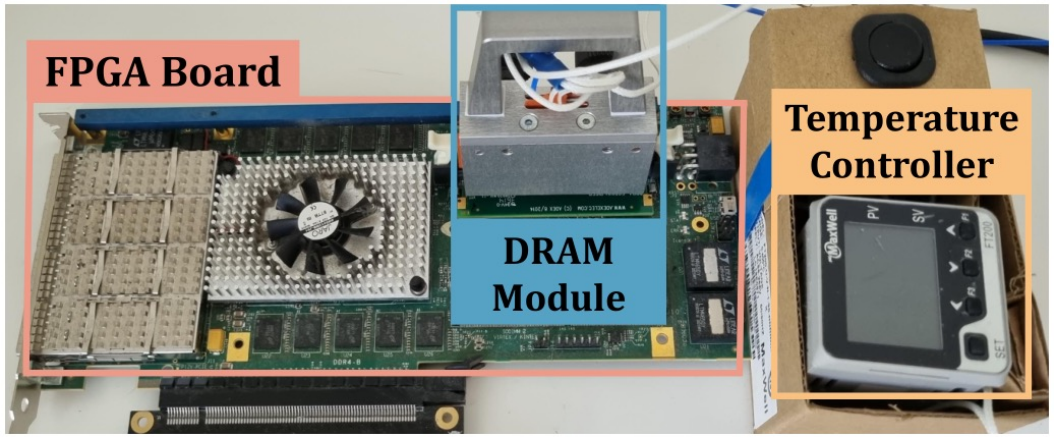
12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

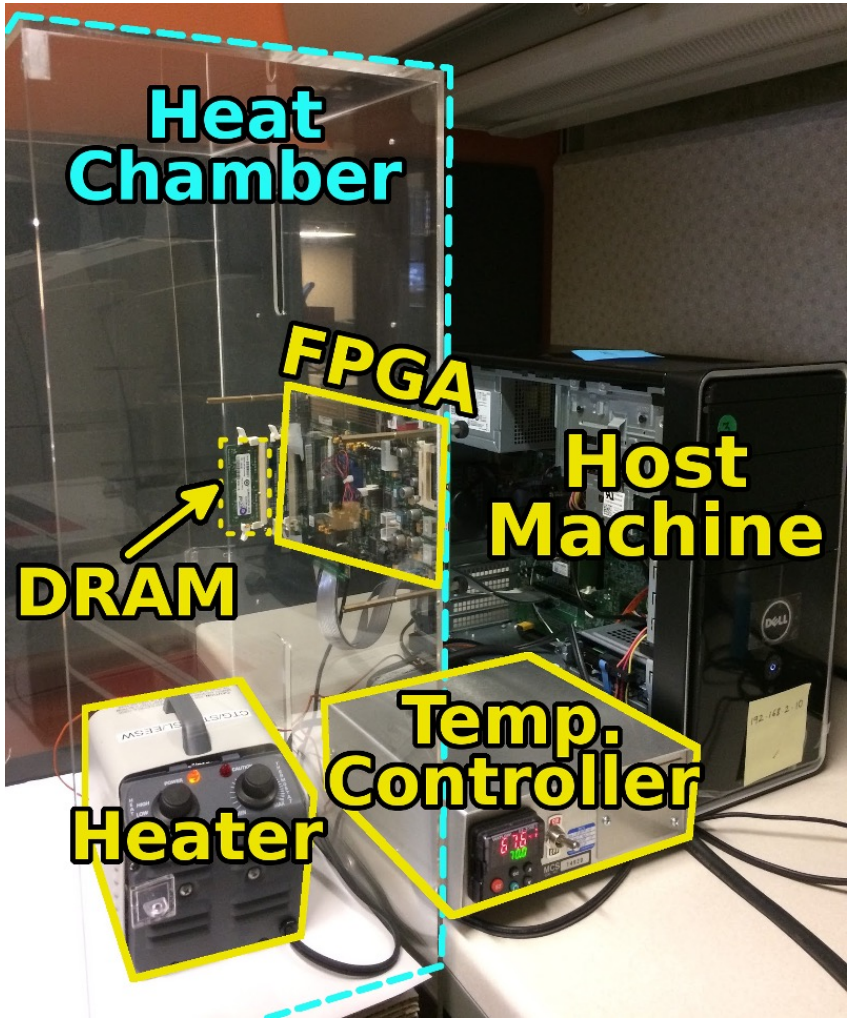
FPGAs in Today's Systems: SoftMC and DRAM Bender

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

  github.com/CMU-SAFARI/SoftMC

  github.com/CMU-SAFARI/DRAM-Bender

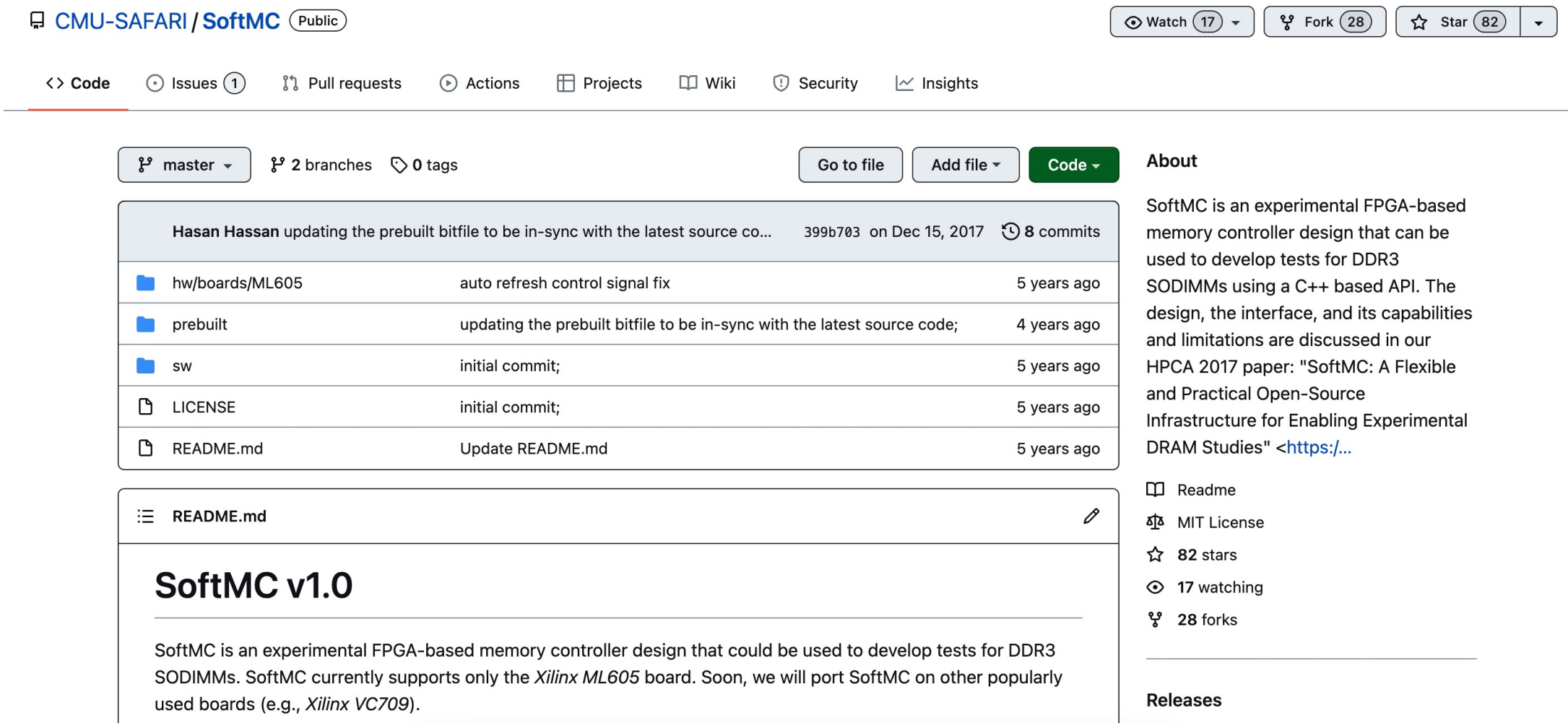




Hassan+, "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies," HPCA 2017.

SoftMC on Github

github.com/CMU-SAFARI/SoftMC



DRAM Bender on Github

github.com/CMU-SAFARI/DRAM-Bender

CMU-SAFARI / **DRAM-Bender** Public

Edit Pins | Unwatch 4 | Fork 0 | Star 8

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 1 branch | 0 tags | Go to file | Add file | Code

olgunataberk Update README.md — 5ddc00b on Dec 2, 2022 — 8 commits

images	Initial commit	6 months ago
prebuilt	Initial commit	6 months ago
projects	Initial commit	6 months ago
sources	Initial commit	6 months ago
.gitignore	Initial commit	6 months ago
.gitmodules	Initial commit	6 months ago
LICENSE	Initial commit	6 months ago
README.md	Update README.md	3 months ago

README.md

DRAM Bender

About

DRAM Bender is the first open source DRAM testing infrastructure that can be used to easily and comprehensively test state-of-the-art DDR4 modules of different form factors. Four prototypes are available on different FPGA boards. Described in our preprint: https://arxiv.org/pdf/2211.05838.pdf

Readme
MIT license
8 stars
4 watching
0 forks

Releases

No releases published
Create a new release

DRAM Bender Preprint on arXiv

arxiv.org/abs/2211.05838

Computer Science > Hardware Architecture

[Submitted on 10 Nov 2022]

DRAM Bender: An Extensible and Versatile FPGA-based Infrastructure to Easily Test State-of-the-art DRAM Chips

Ataberk Olgun, Hasan Hassan, A. Giray Yağlıkçı, Yahya Can Tuğrul, Lois Orosa, Haocong Luo, Minesh Patel, Oğuz Ergin, Onur Mutlu

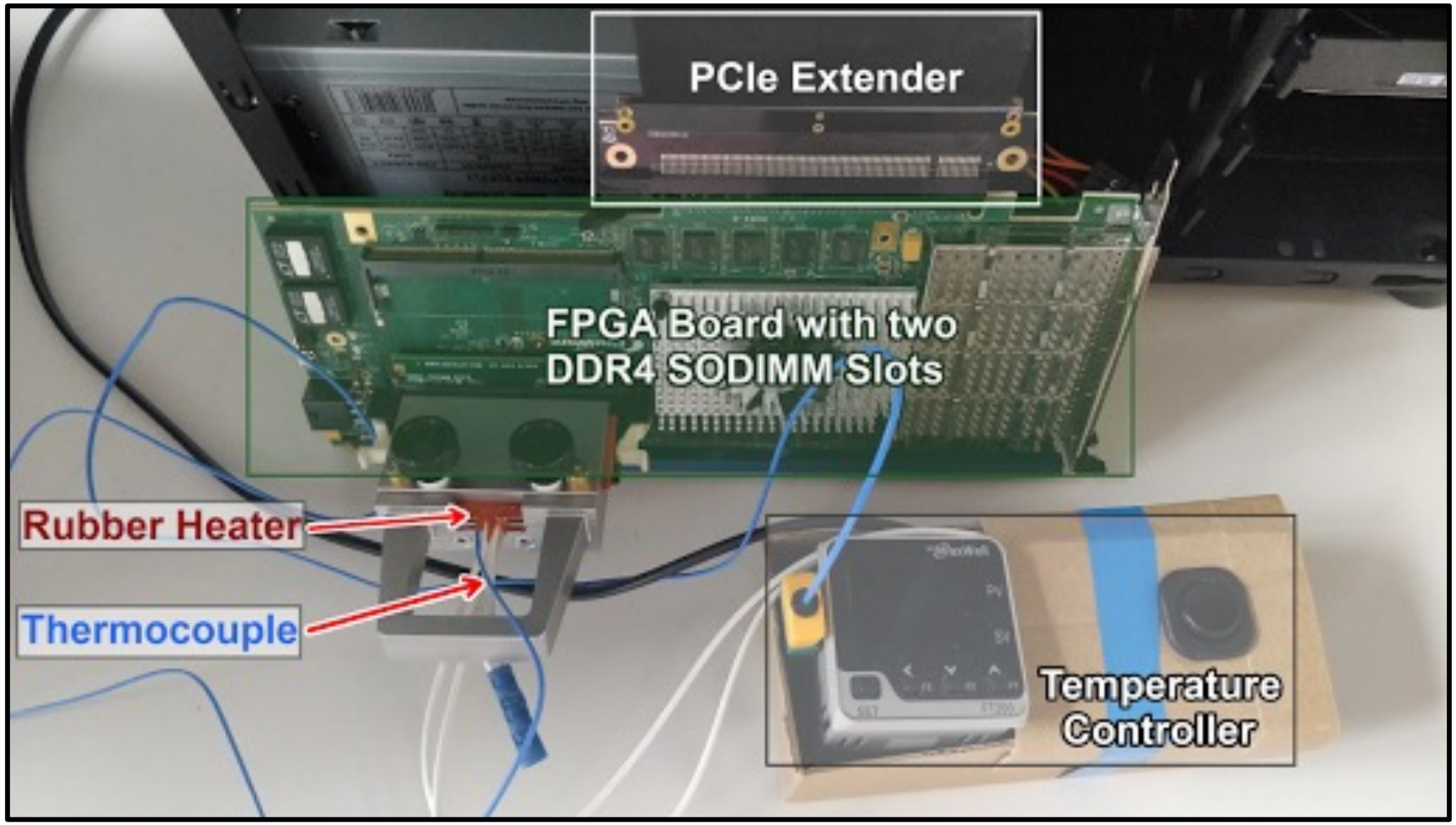
To understand and improve DRAM performance, reliability, security and energy efficiency, prior works study characteristics of commodity DRAM chips. Unfortunately, state-of-the-art open source infrastructures capable of conducting such studies are obsolete, poorly supported, or difficult to use, or their inflexibility limit the types of studies they can conduct.

We propose DRAM Bender, a new FPGA-based infrastructure that enables experimental studies on state-of-the-art DRAM chips. DRAM Bender offers three key features at the same time. First, DRAM Bender enables directly interfacing with a DRAM chip through its low-level interface. This allows users to issue DRAM commands in arbitrary order and with finer-grained time intervals compared to other open source infrastructures. Second, DRAM Bender exposes easy-to-use C++ and Python programming interfaces, allowing users to quickly and easily develop different types of DRAM experiments. Third, DRAM Bender is easily extensible. The modular design of DRAM Bender allows extending it to (i) support existing and emerging DRAM interfaces, and (ii) run on new commercial or custom FPGA boards with little effort.

To demonstrate that DRAM Bender is a versatile infrastructure, we conduct three case studies, two of which lead to new observations about the DRAM RowHammer vulnerability. In particular, we show that data patterns supported by DRAM Bender uncovers a larger set of bit-flips on a victim row compared to the data patterns commonly used by prior work. We demonstrate the extensibility of DRAM Bender by implementing it on five different FPGAs with DDR4 and DDR3 support. DRAM Bender is freely and openly available at this https URL.

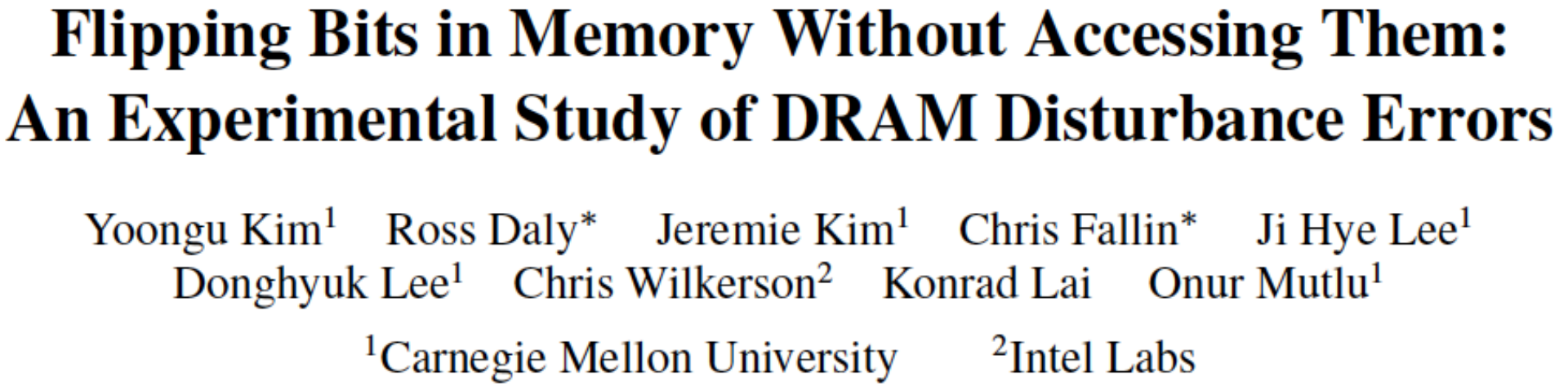
Subjects: **Hardware Architecture (cs.AR)**; Cryptography and Security (cs.CR)
Cite as: arXiv:2211.05838 **[cs.AR]**
(or arXiv:2211.05838v1 **[cs.AR]** for this version)
https://doi.org/10.48550/arXiv.2211.05838

DRAM Bender


PCIe Extender
FPGA Board with two
DDR4 SODIMM Slots
Rubber Heater
Thermocouple
Temperature
Controller

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 Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel 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]

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]

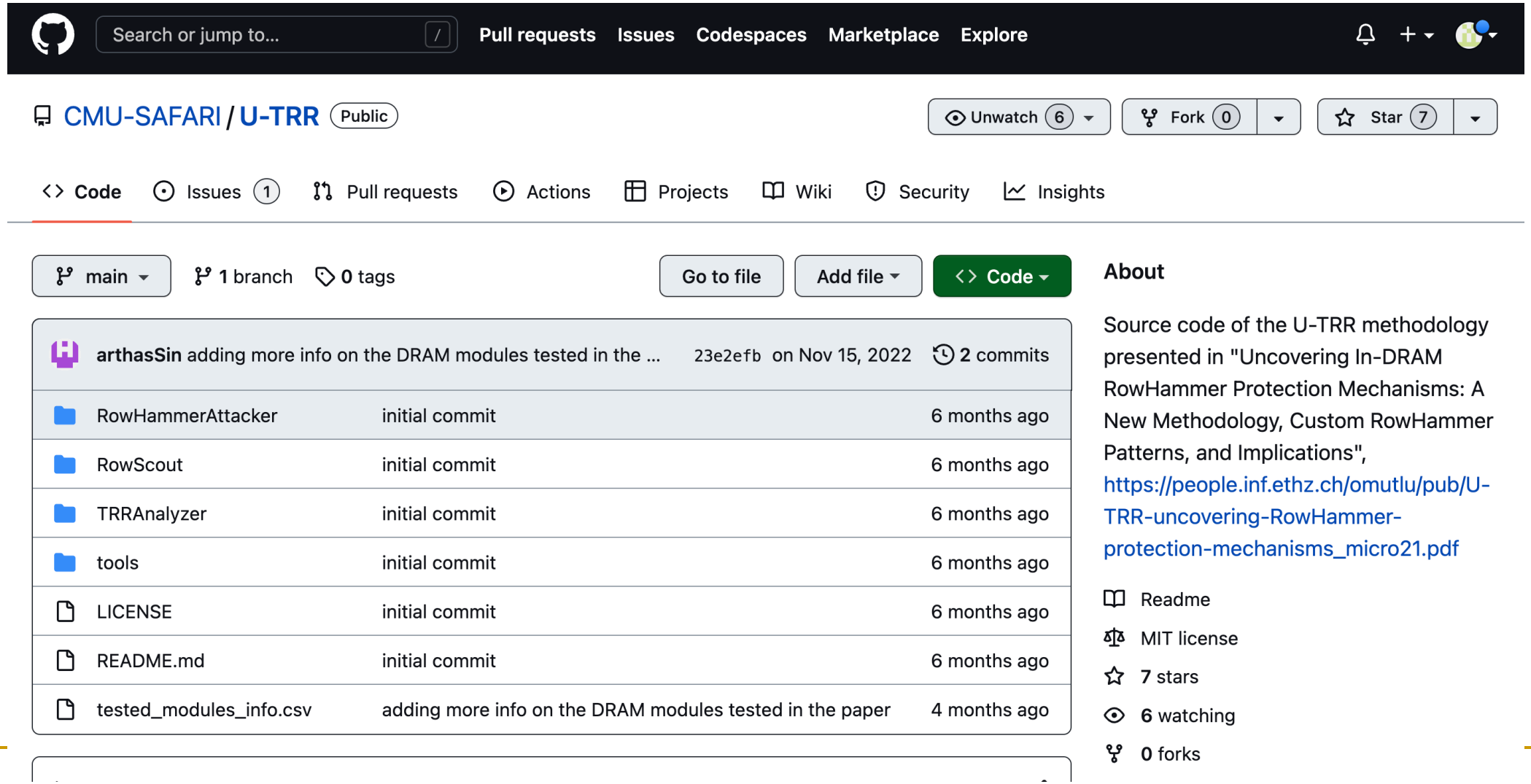
Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan[†] Yahya Can Tuğrul[†‡] Jeremie S. Kim[†] Victor van der Veen[σ]
Kaveh Razavi[†] Onur Mutlu[†]

[†]*ETH Zürich* [‡]*TOBB University of Economics & Technology* [σ]*Qualcomm Technologies Inc.*

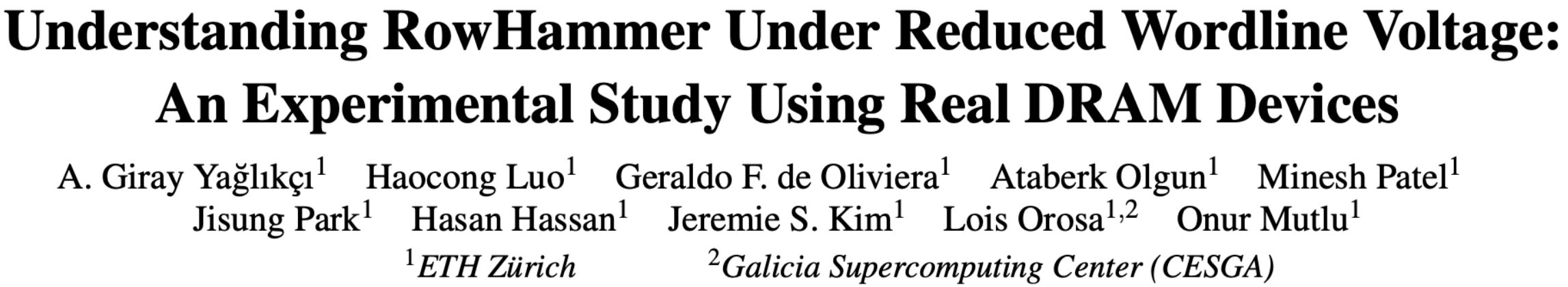
U-TRR on Github

github.com/CMU-SAFARI/U-TRR



RowHammer in 2022

- A. Giray Yağlıkçı, Haocong Luo, Geraldo F. de Oliviera, Ataberk Olgun, Minesh Patel, Jisung Park, Hasan Hassan, Jeremie S. Kim, Lois Orosa, and Onur Mutlu,
  "Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices"
  Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (***DSN***),
  Baltimore, MD, USA, June 2022.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [arXiv version]
  [Talk Video (34 minutes, including Q&A)]
  [Lightning Talk Video (2 minutes)]



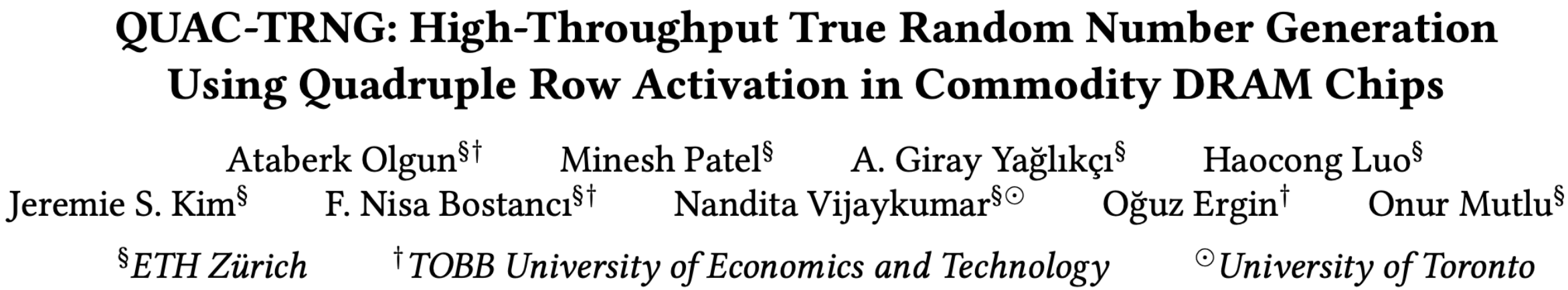
Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices

A. Giray Yağlıkçı[1] Haocong Luo[1] Geraldo F. de Oliviera[1] Ataberk Olgun[1] Minesh Patel[1]
Jisung Park[1] Hasan Hassan[1] Jeremie S. Kim[1] Lois Orosa[1,2] Onur Mutlu[1]

[1]*ETH Zürich* [2]*Galicia Supercomputing Center (CESGA)*

True Random Number Generation in DRAM (I)

- Ataberk Olgun, Minesh Patel, A. Giray Yaglikci, Haocong Luo, Jeremie S. Kim, F. Nisa Bostanci, Nandita Vijaykumar, Oguz Ergin, and Onur Mutlu,
  "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"
  *Proceedings of the 48th International Symposium on Computer Architecture (**ISCA**)*, Virtual, June 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Video (25 minutes)]
  [SAFARI Live Seminar Video (1 hr 26 mins)]



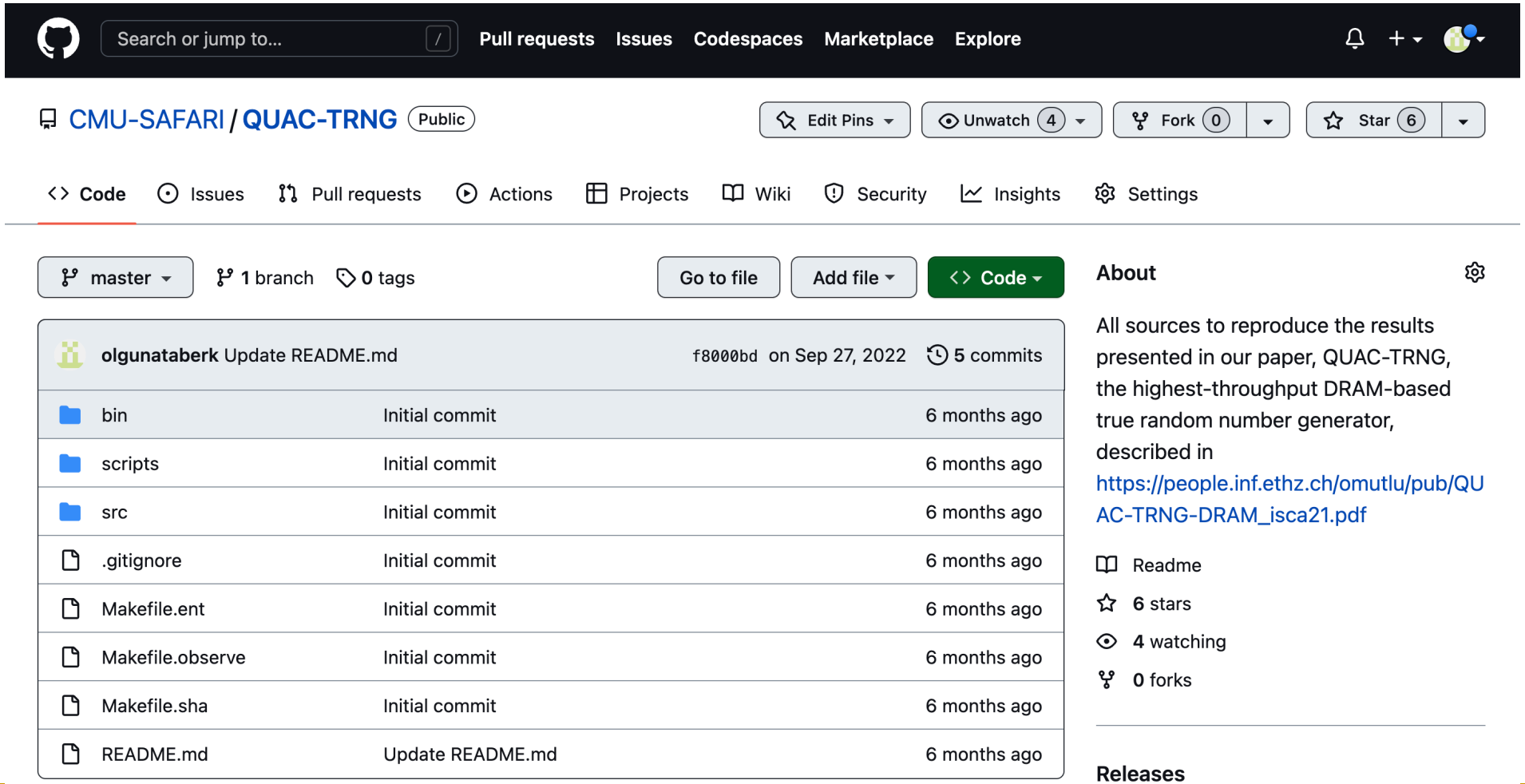
QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips

Ataberk Olgun[§†] Minesh Patel[§] A. Giray Yağlıkçı[§] Haocong Luo[§]
Jeremie S. Kim[§] F. Nisa Bostancı[§†] Nandita Vijaykumar[§⊙] Oğuz Ergin[†] Onur Mutlu[§]

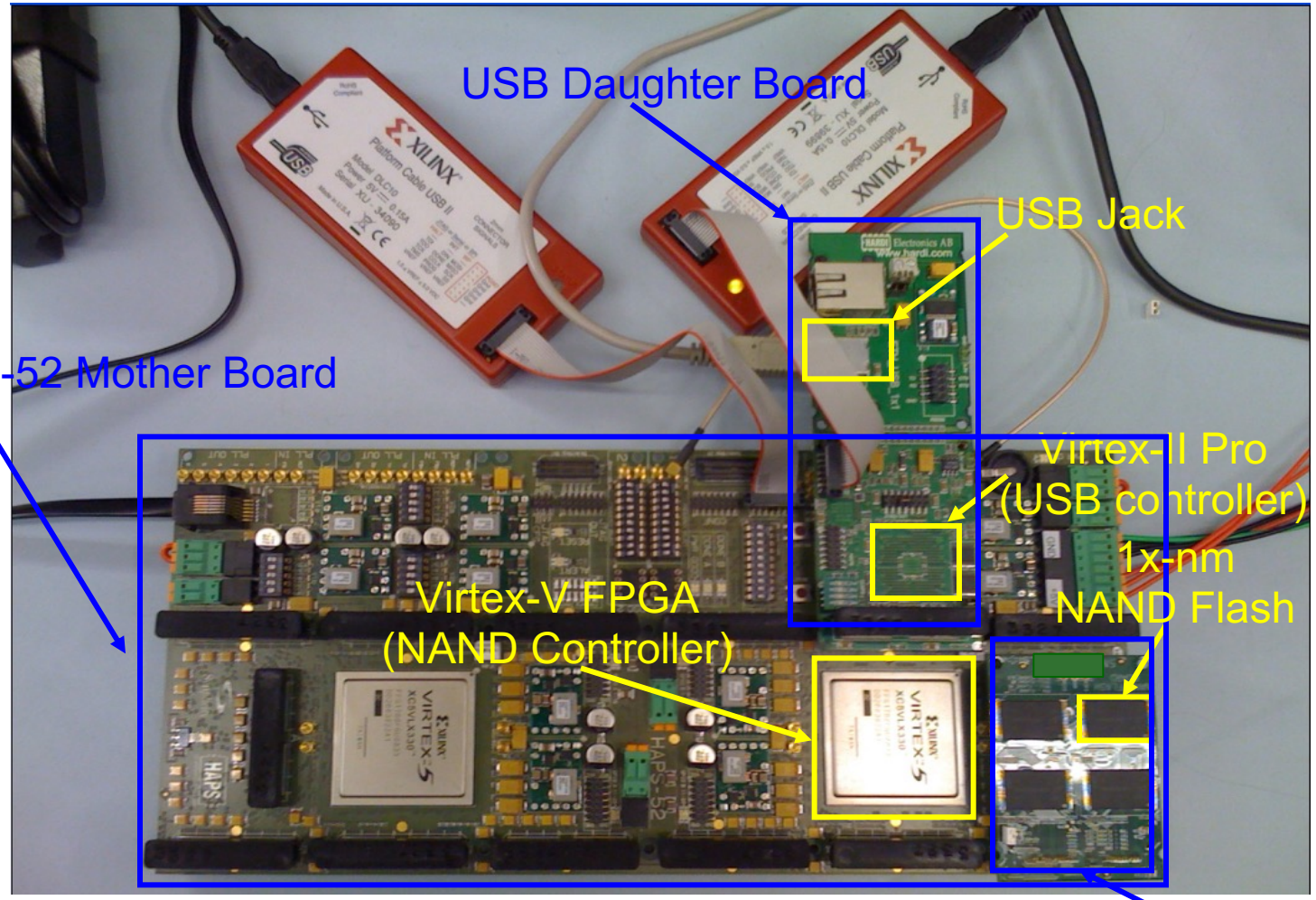
[§]*ETH Zürich* [†]*TOBB University of Economics and Technology* [⊙]*University of Toronto*

True Random Number Generation in DRAM (II)

github.com/CMU-SAFARI/QUAC-TRNG



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]

Cai+, "**Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives**", Proc. IEEE 2017.

Intelligent Flash Controllers [PIEEE'17]

Proceedings of the IEEE, Sept. 2017

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

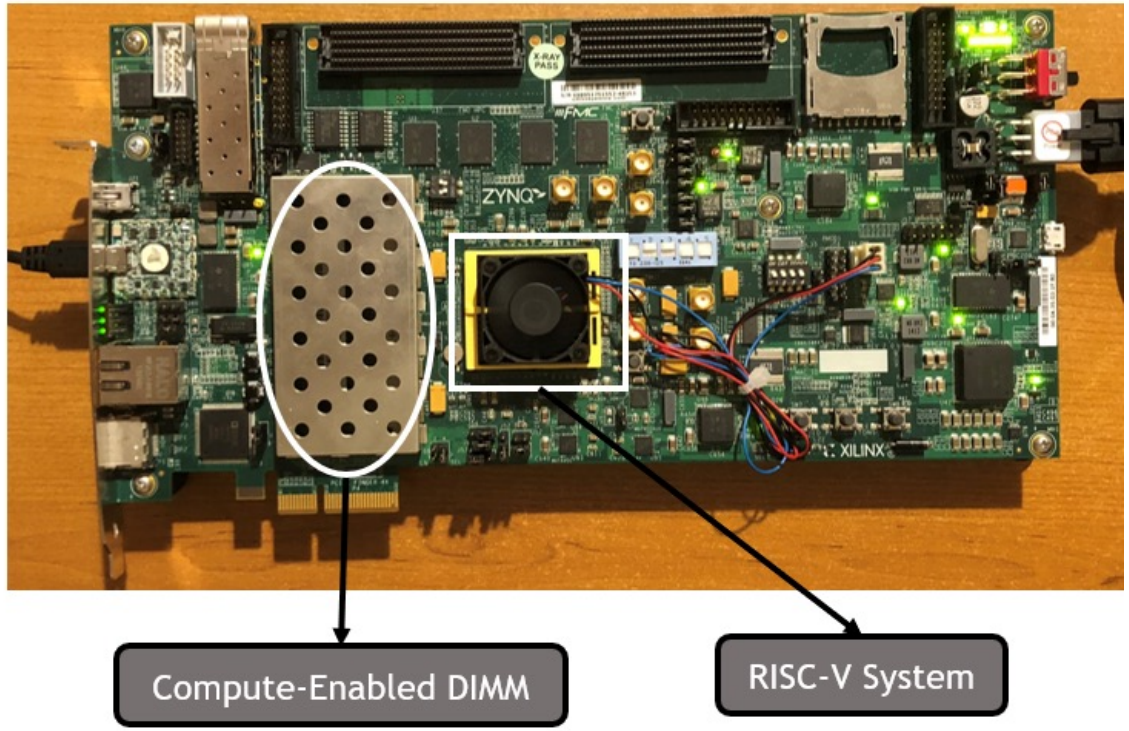
This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

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"
*Transactions on Architecture and Code Optimization (**TACO**)*, November 2022.
[Github Link]



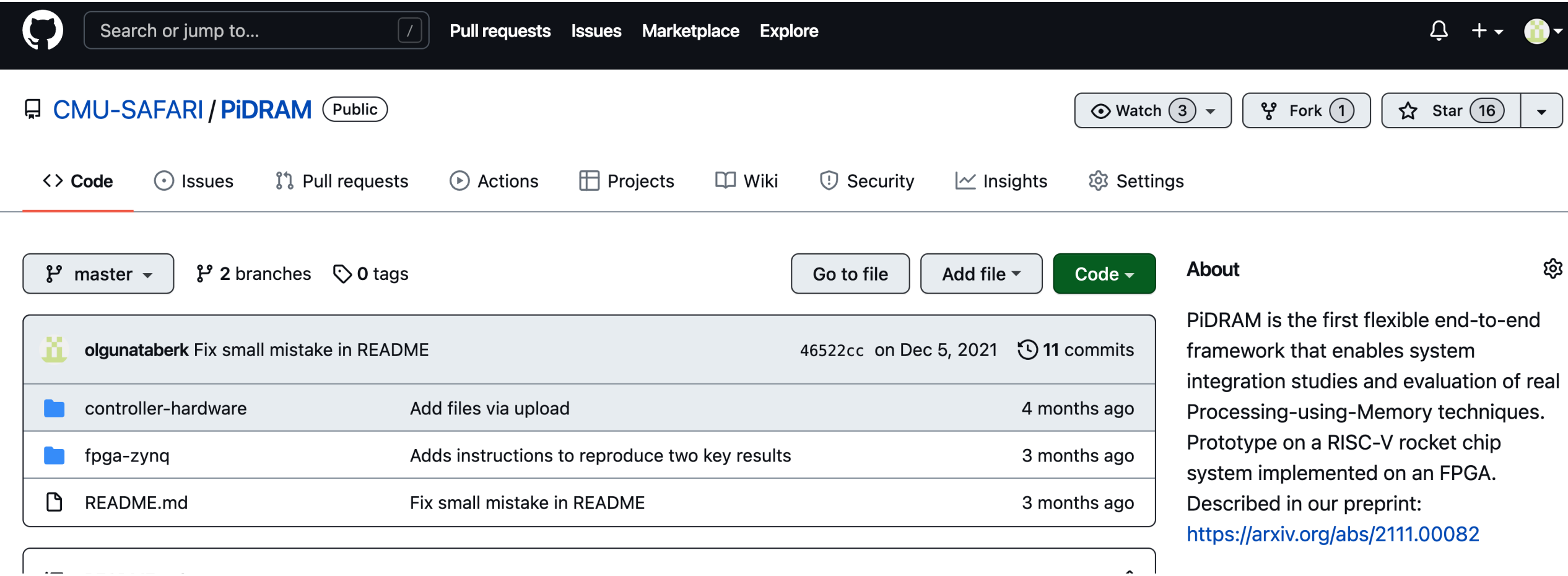
Real System Evaluation

In-DRAM **data copy**

In-DRAM **true random number generation**

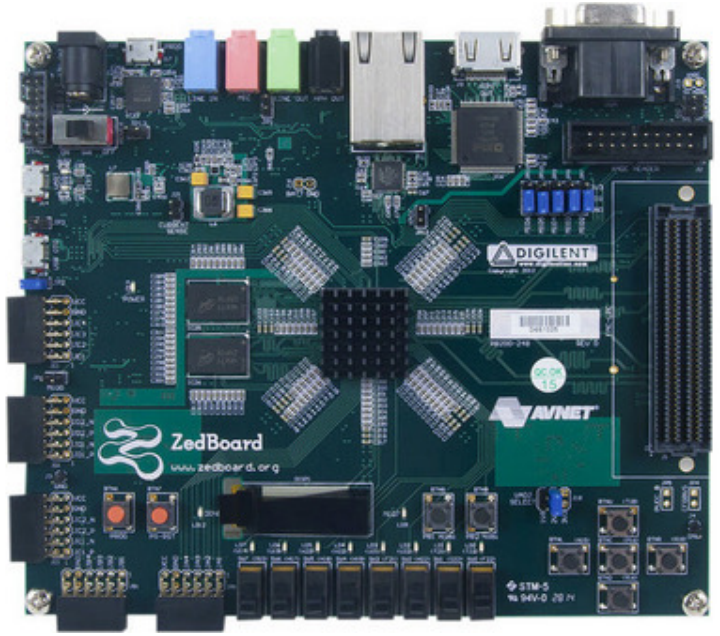
FPGAs for Processing-in-Memory Prototyping: PiDRAM (II)

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



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"
Transactions on Architecture and Code Optimization (***TACO***), 2022.
[Github Link]



Xilinx Zedboard

FPGAs for Prototyping New Architectures: MetaSys (II)

safari.ethz.ch/hipeac-best-paper-award-for-metasys/

Best Paper Award at HiPEAC 2023:

MetaSys: A Practical Open-source Metadata Management System to Implement and Evaluate Cross-layer Optimizations

HiPEAC

ETH zürich

SAFARI

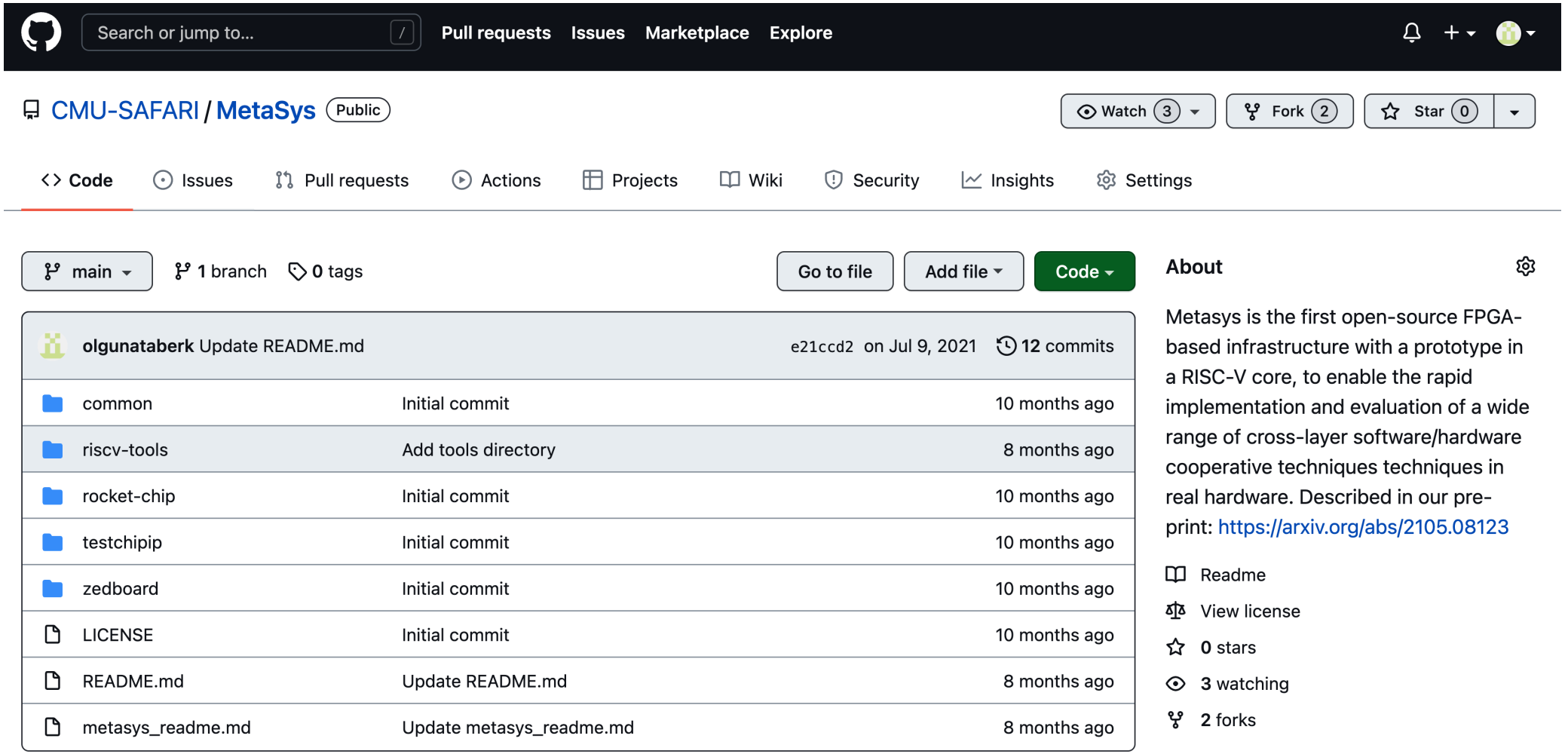
SAFARI Research Group

UNIVERSITY OF TORONTO

embARC Research Group

FPGAs for Prototyping New Architectures: MetaSys (III)

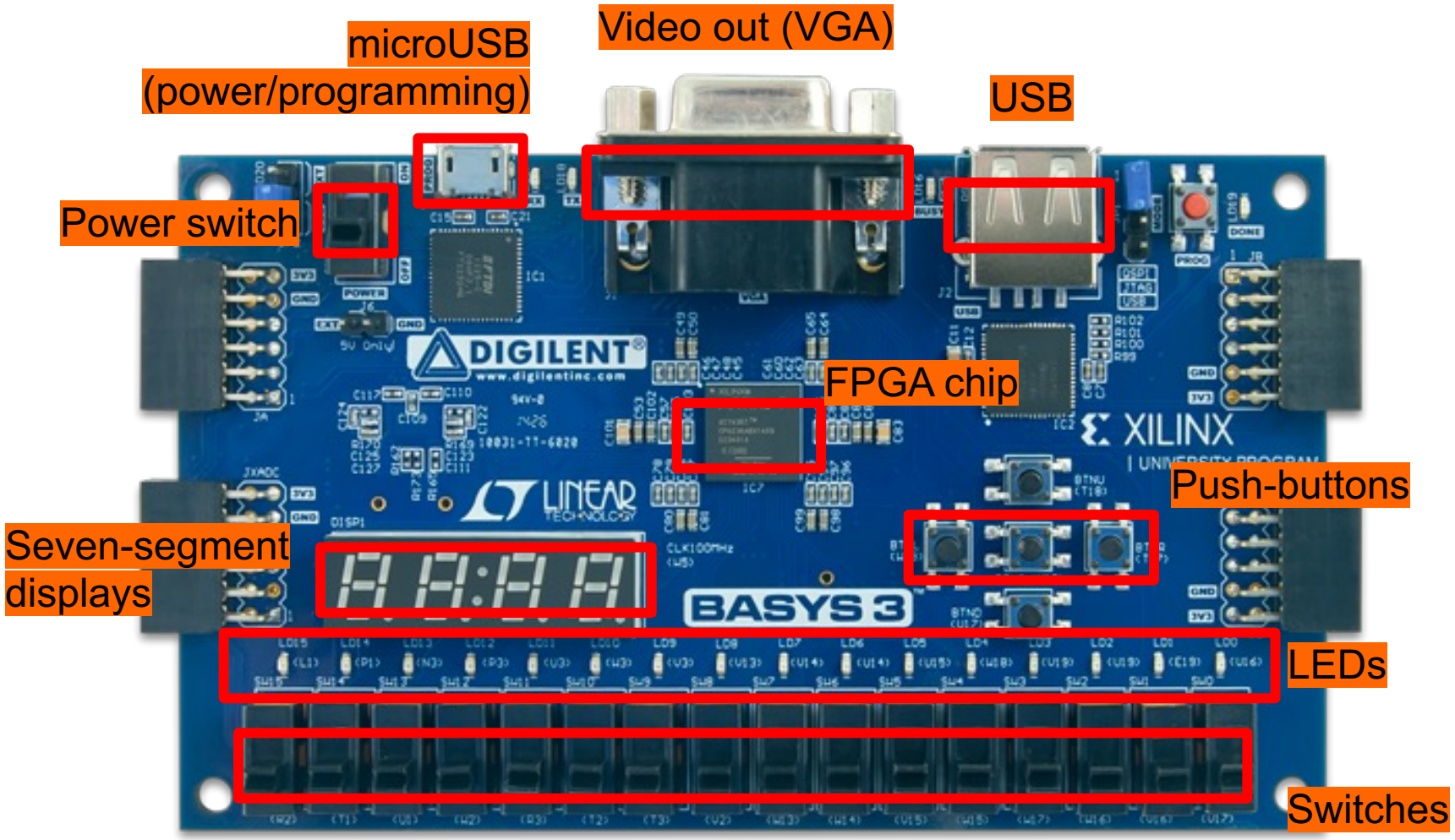
https://github.com/CMU-SAFARI/MetaSys



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

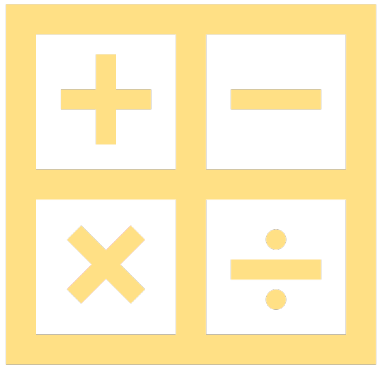
- **Comparison** is a common operation in software programming
  - We usually want to know the relation between two variables (e.g., **<**, **>**, **==**, …)

- 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

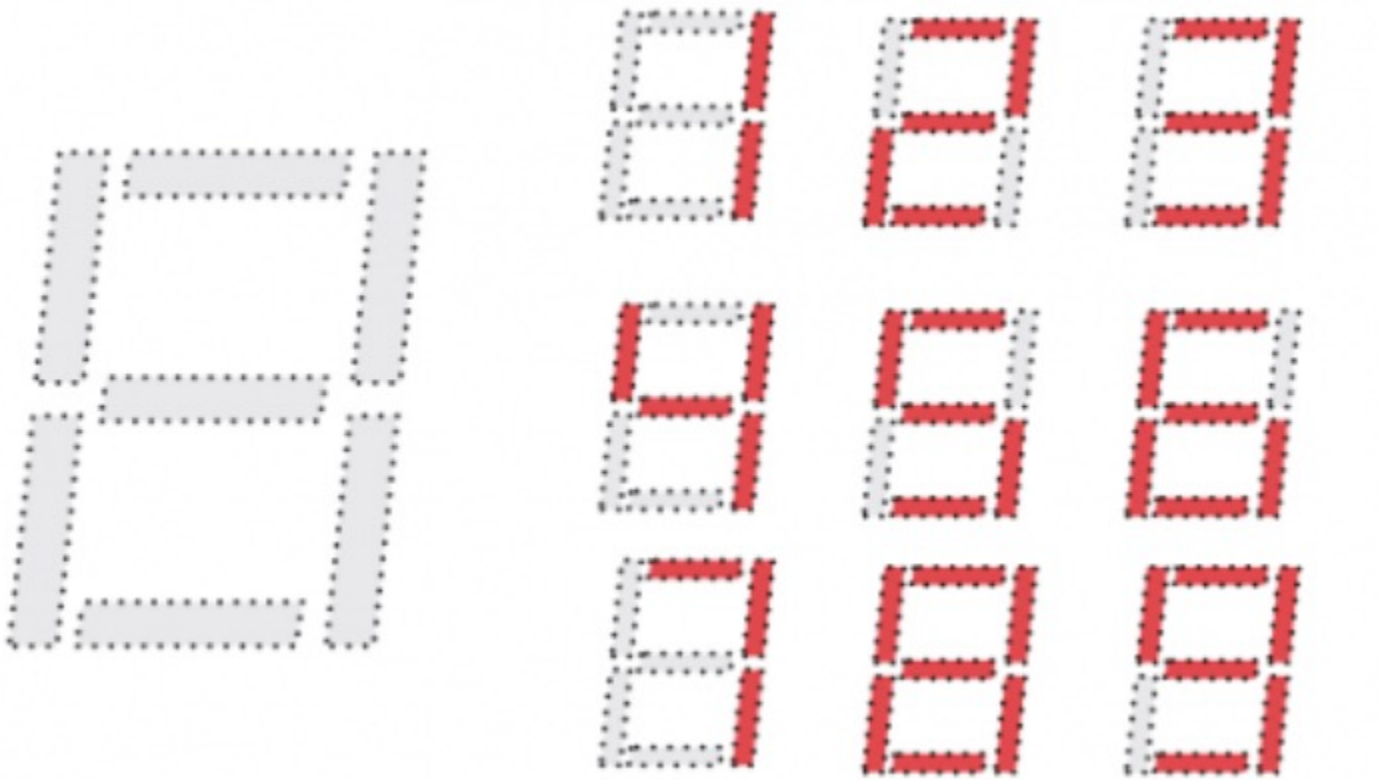
- Another common operation in software programming?
  - Addition



- 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

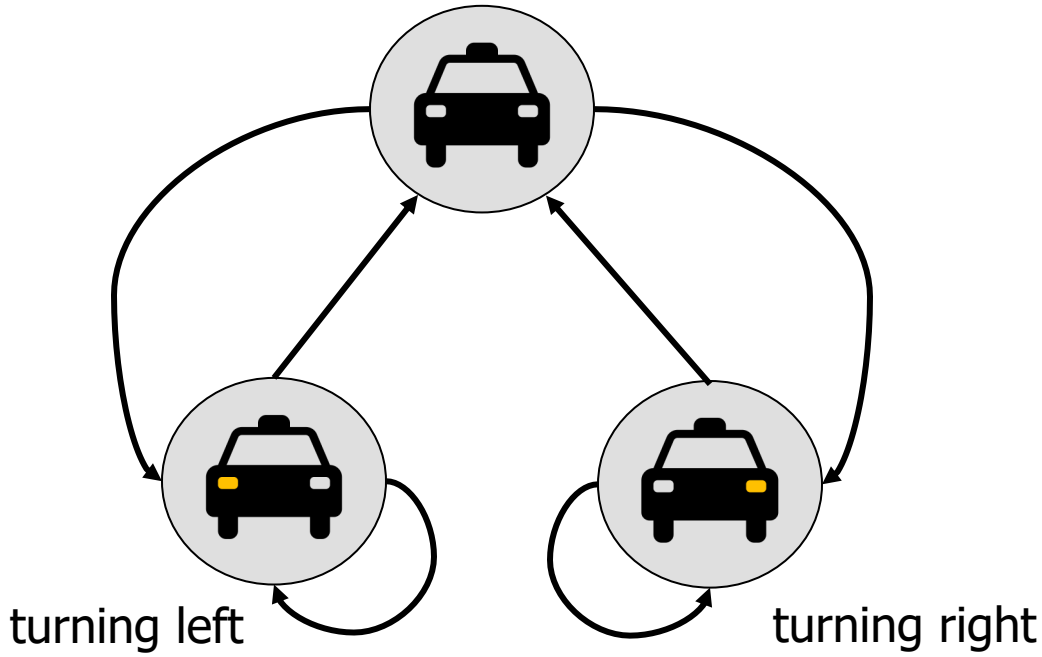
- Show your results from Lab 2 on a Seven Segment Display



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

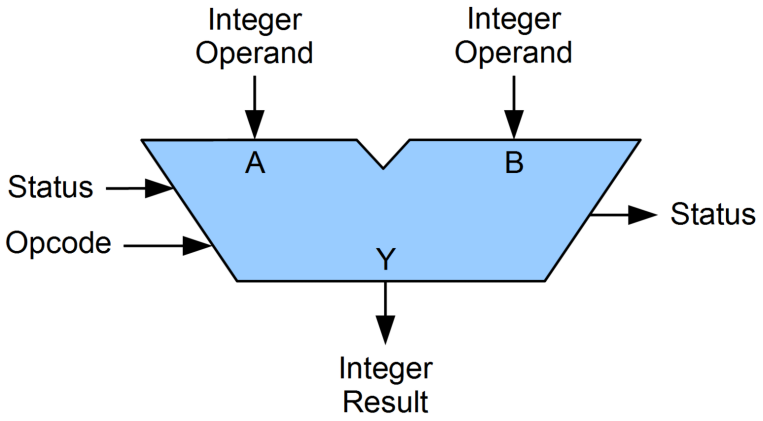
Lab 4: Finite State Machines

- 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;

}
```

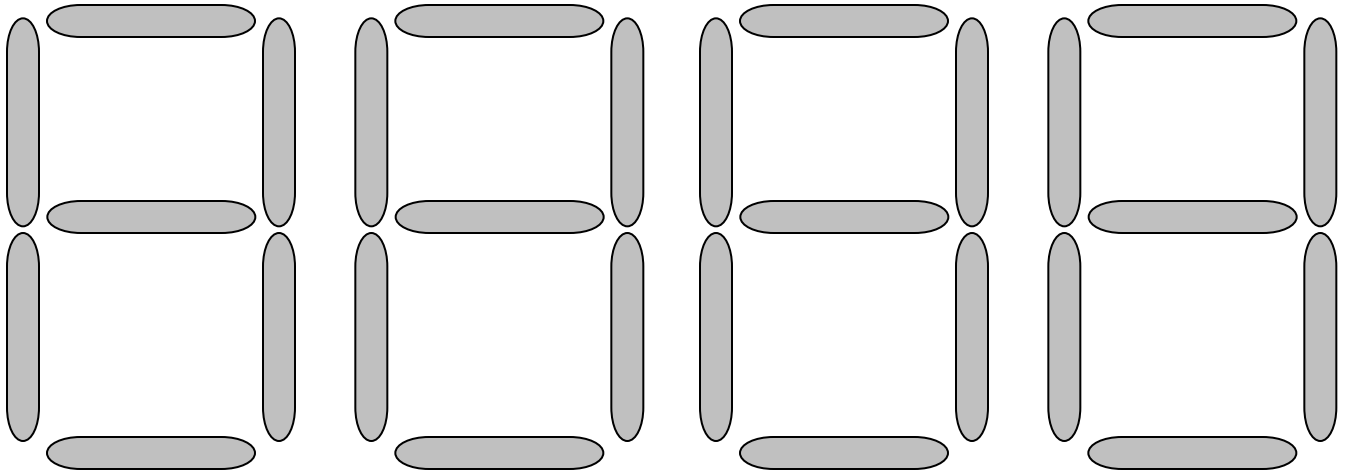
MIPS assembly

```
# 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:
```

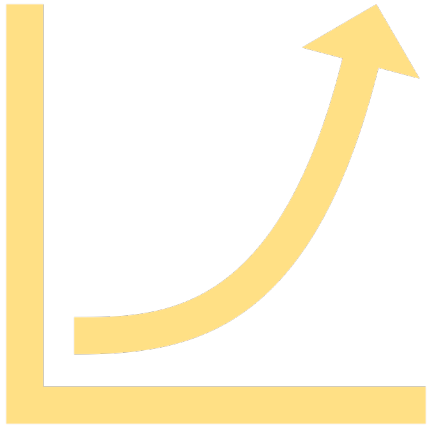
Lab 8: Full System Integration

- 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

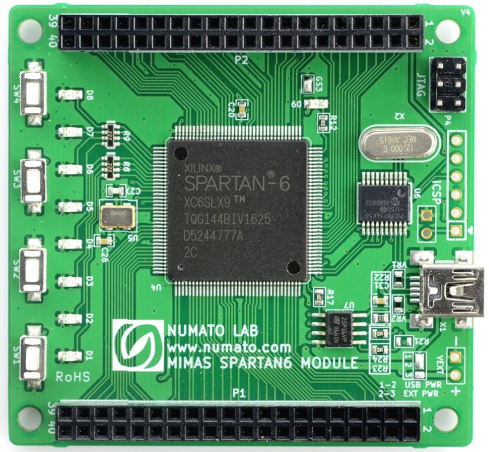


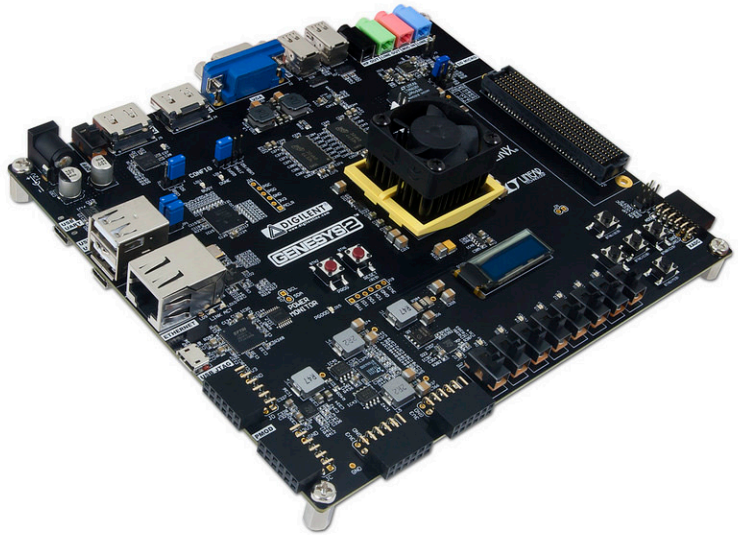
Agenda

- 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

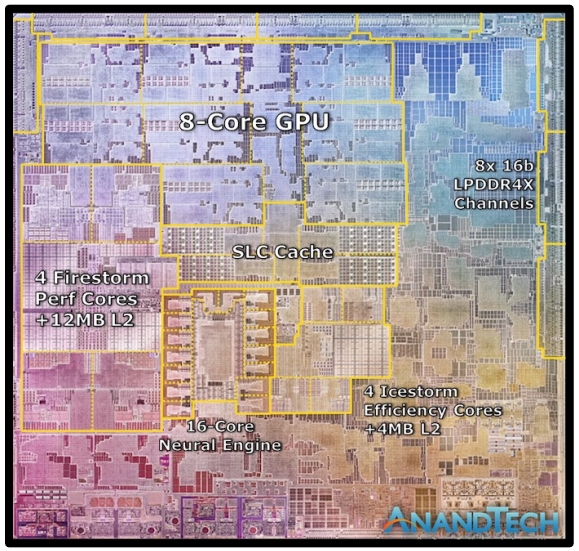




- FPGA is a software-reconfigurable hardware substrate
  - Reconfigurable functions
  - Reconfigurable interconnection of functions
  - Reconfigurable input/output (IO)

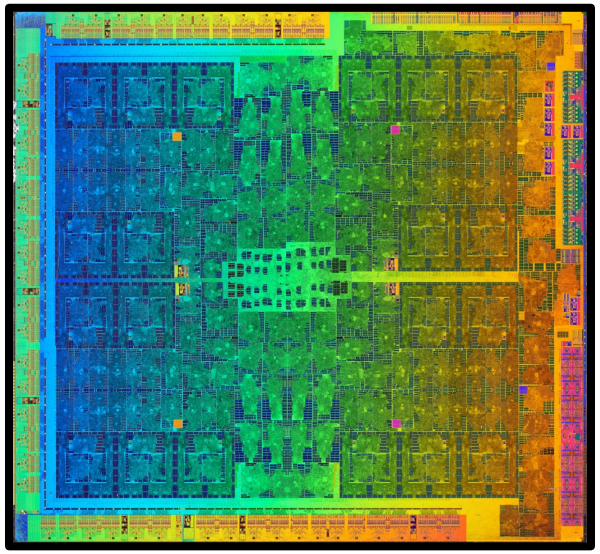
FPGAs & Other Integrated Circuits

CPUs



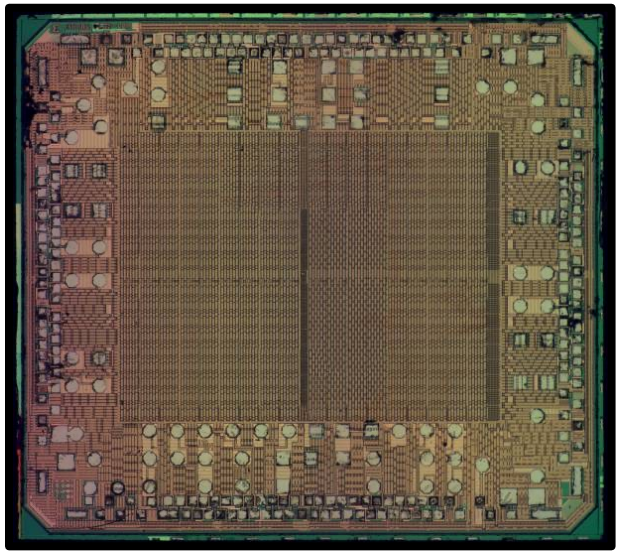
Apple M1

GPUs



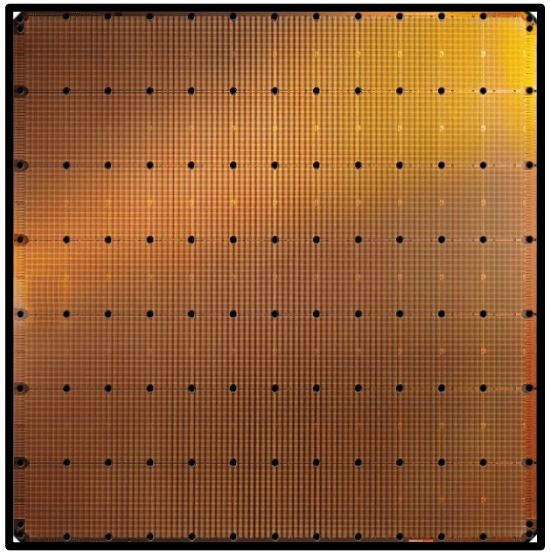
Nvidia GTX 1070

FPGAs



Xilinx Spartan

ASICs

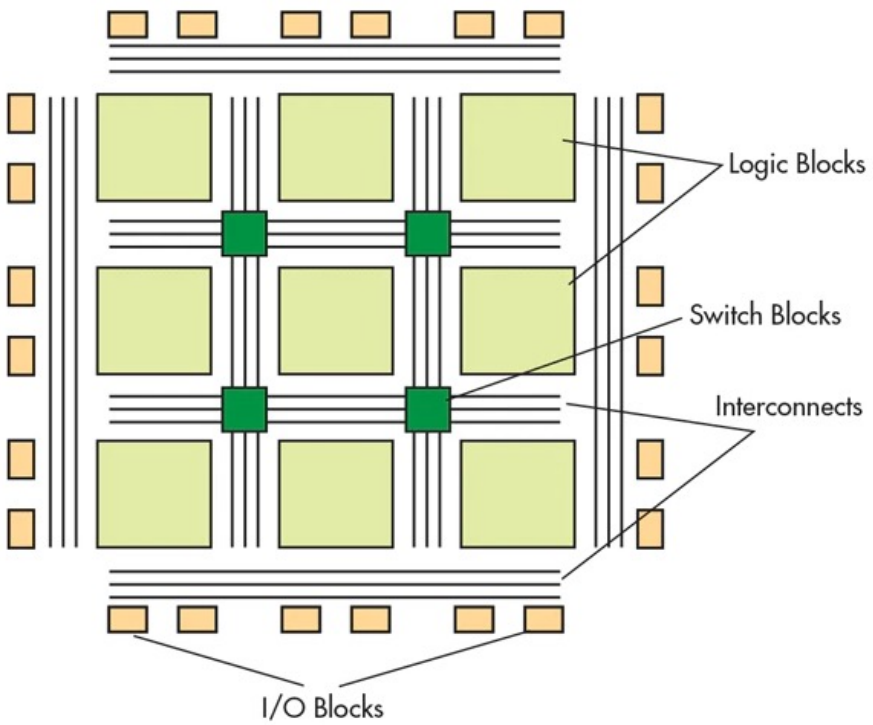


Cerebras WSE-2

Flexibility
Programming Ease

Efficiency

A High-Level Overview of FPGAs

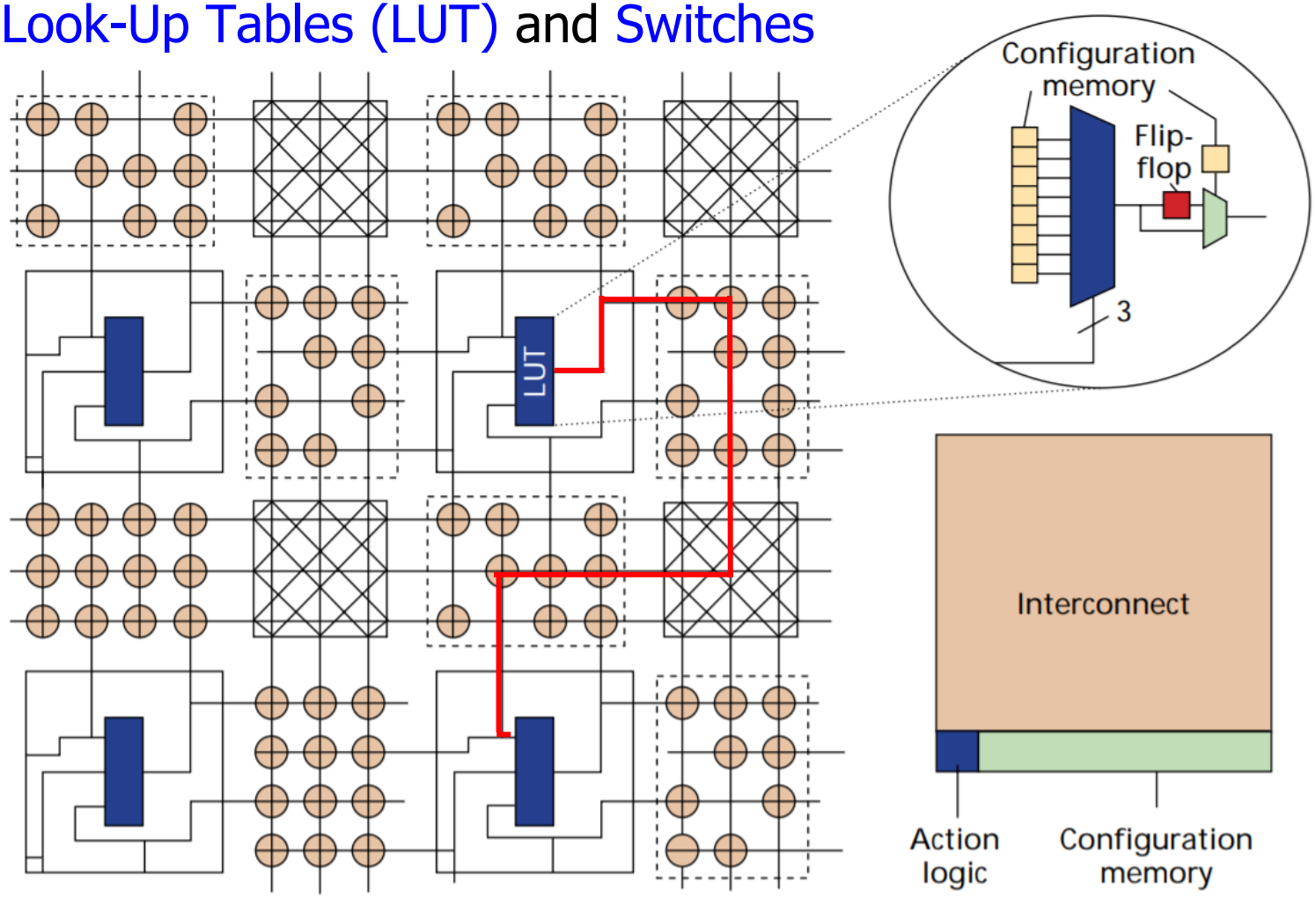


We **configure** logic blocks, their connections, and IO blocks to create hardware circuits and map programs onto those circuits

Image source: https://www.electronicdesign.com/technologies/fpgas/article/21801527/the-principles-of-fpgas

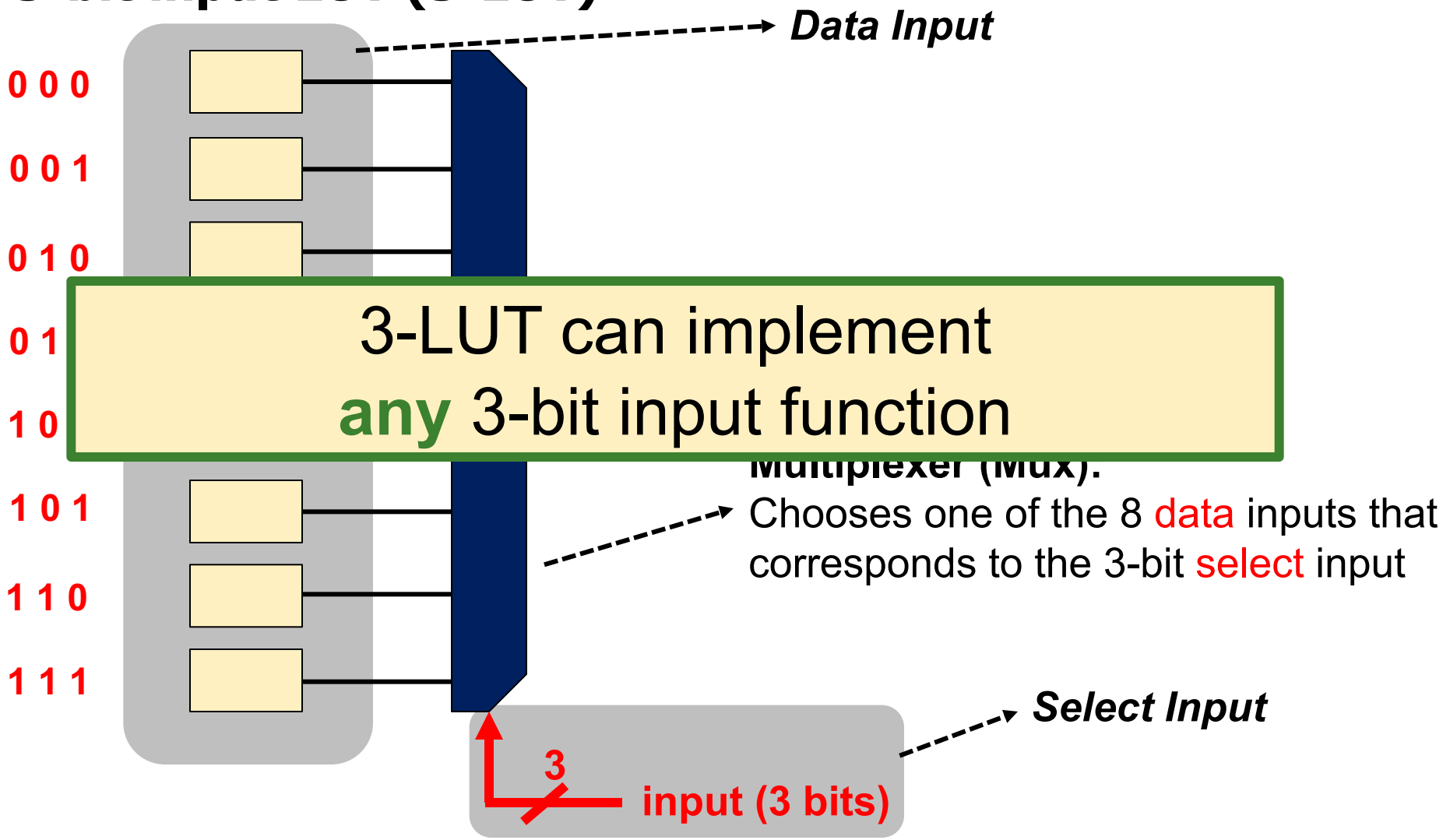
FPGA Architecture - Looking Inside an FPGA

- 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:

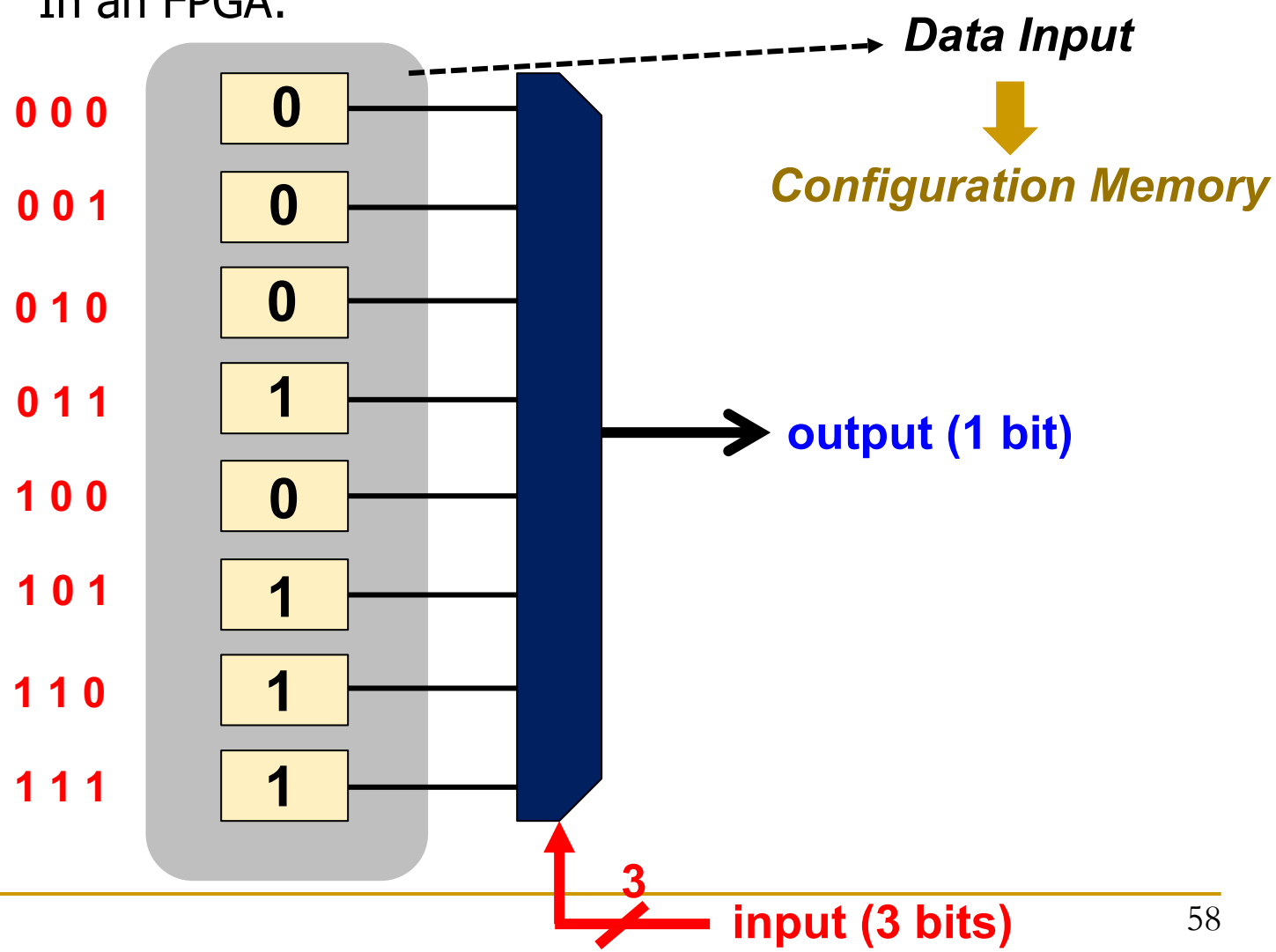
```
int count = 0;
for(int i = 0; i < 3; i++) {
    count += input & 1;
    input = input >> 1;
}

if(count > 1) return 1;

return 0;
```

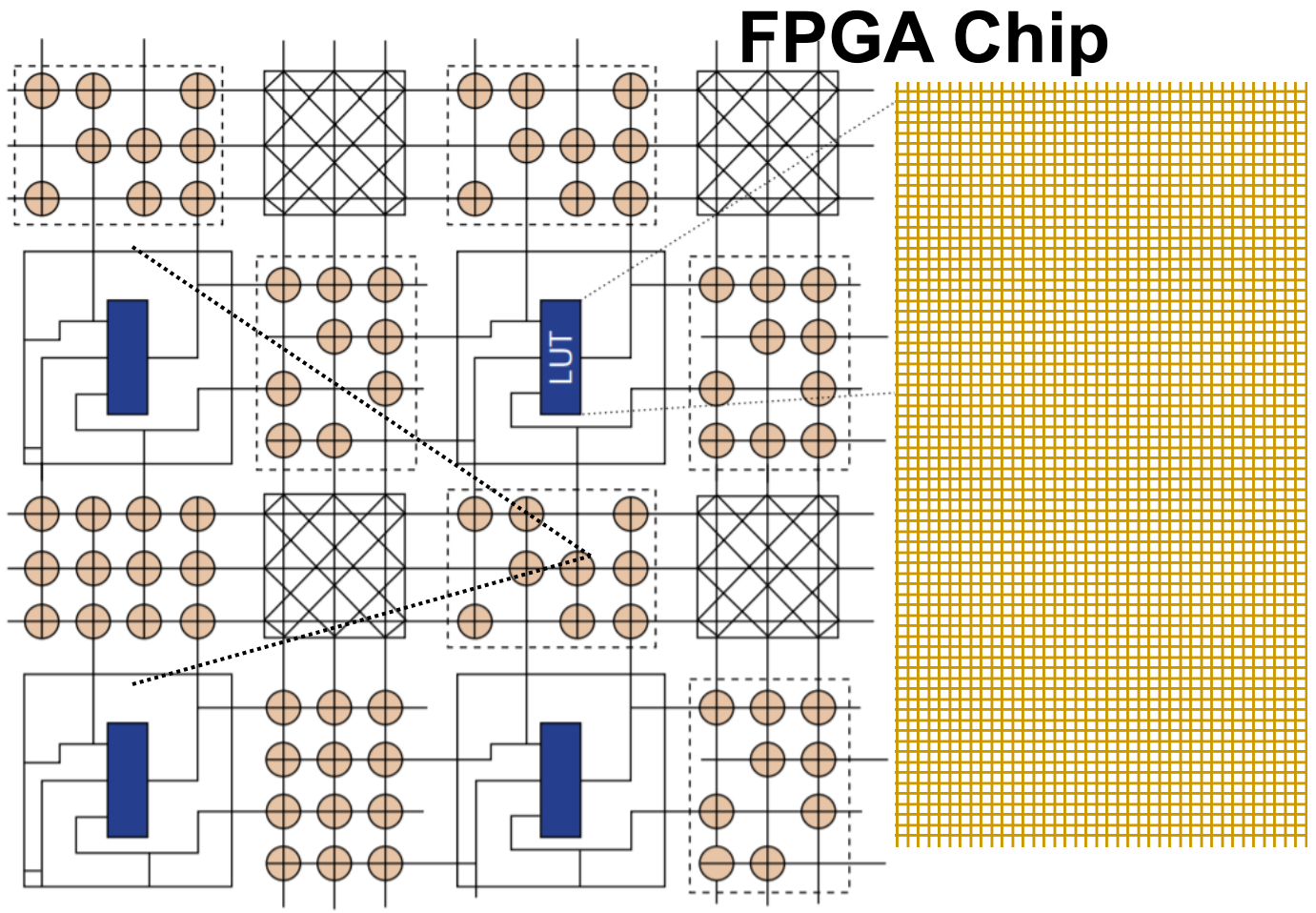
```
switch(input){
   case 0:
   case 1:
   case 2:
   case 4:
      return 0;
   default:
      return 1;}
```

In an FPGA:



How to Implement Complex Functions?

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

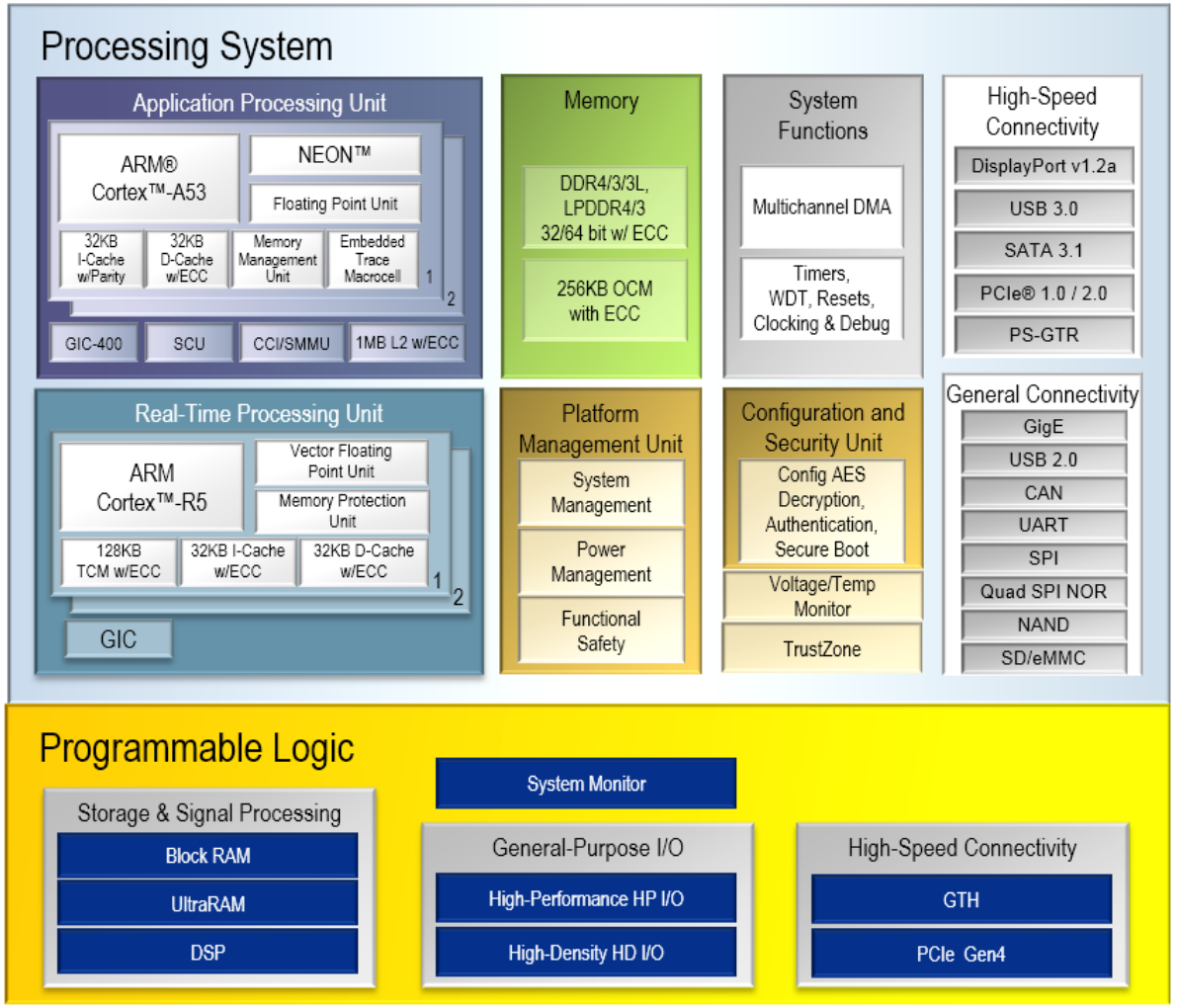


Andre DeHon, “The Density Advantage of Configurable Computing”, *Computer*, 2000

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

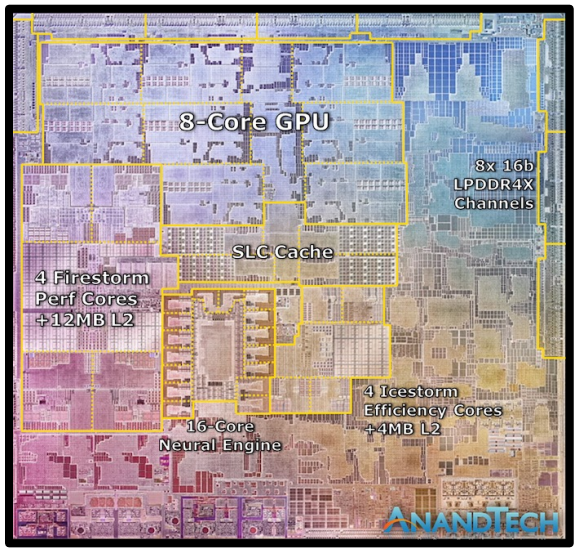
An Example Modern FPGA Platform: Xilinx Zynq Ultrascale+



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

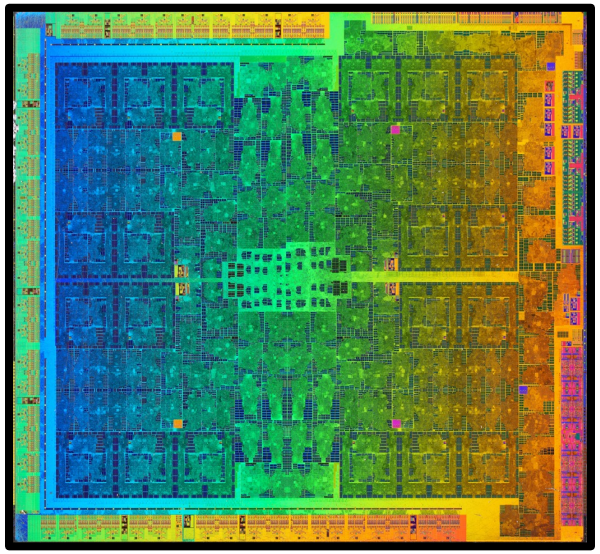
Advantages & Disadvantages of FPGAs (I)

CPUs



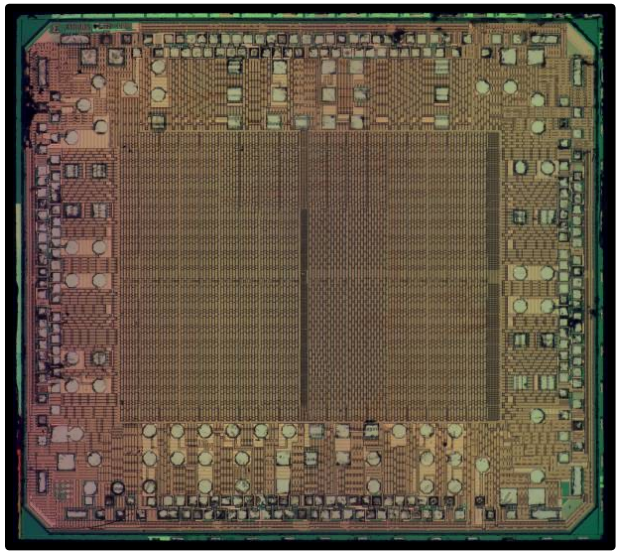
Apple M1

GPUs



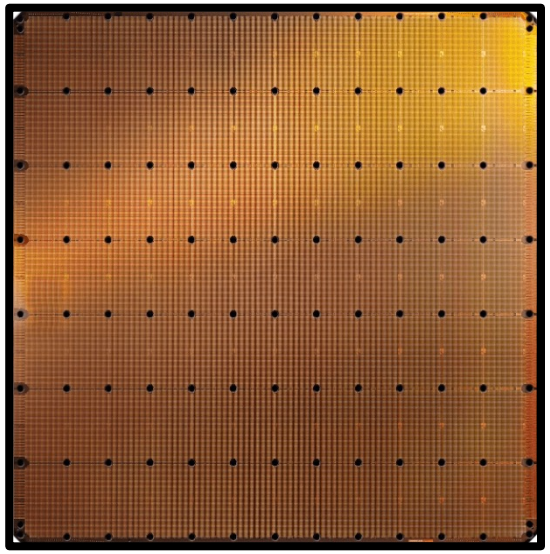
Nvidia GTX 1070

FPGAs



Xilinx Spartan

ASICs



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

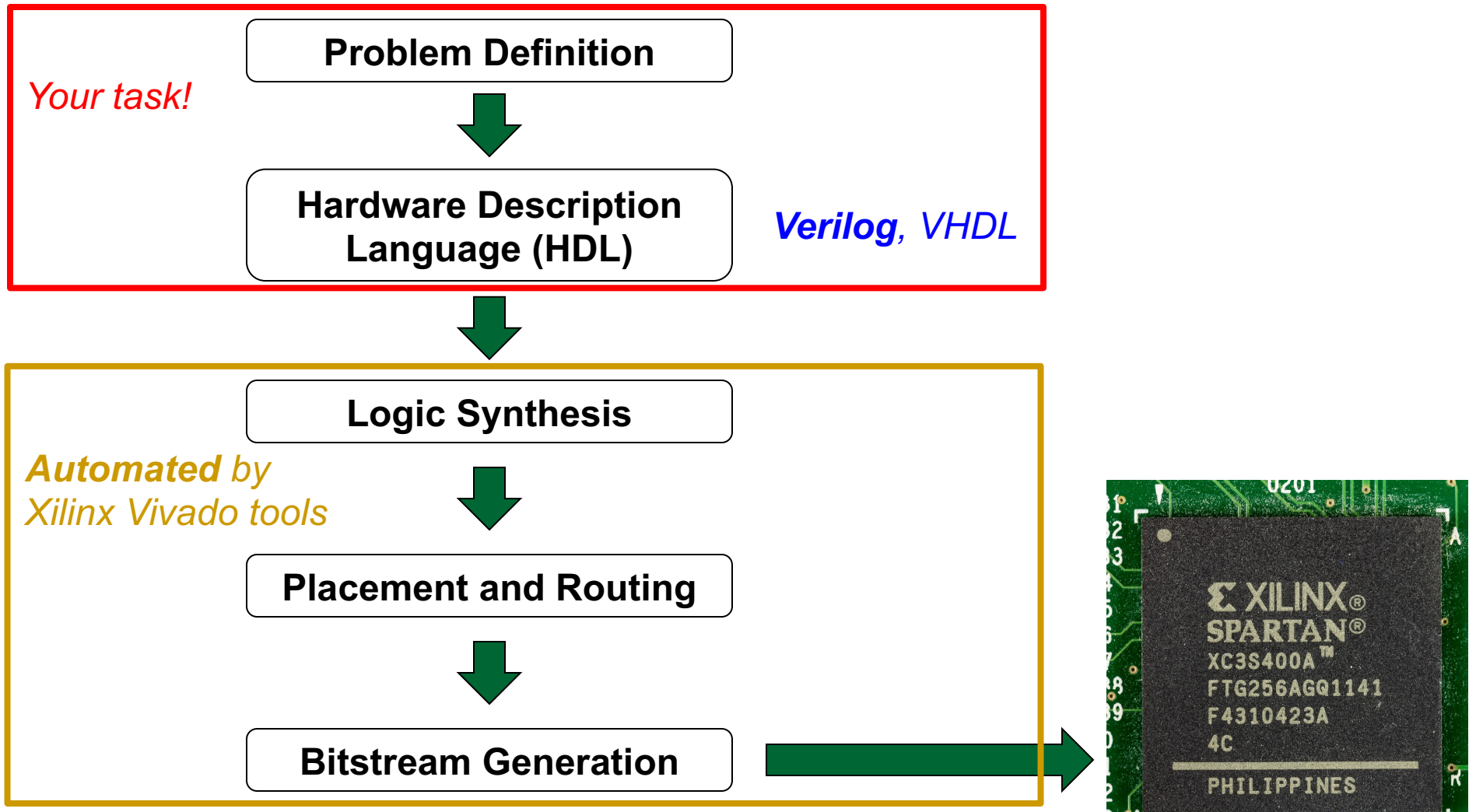
Agenda

- 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

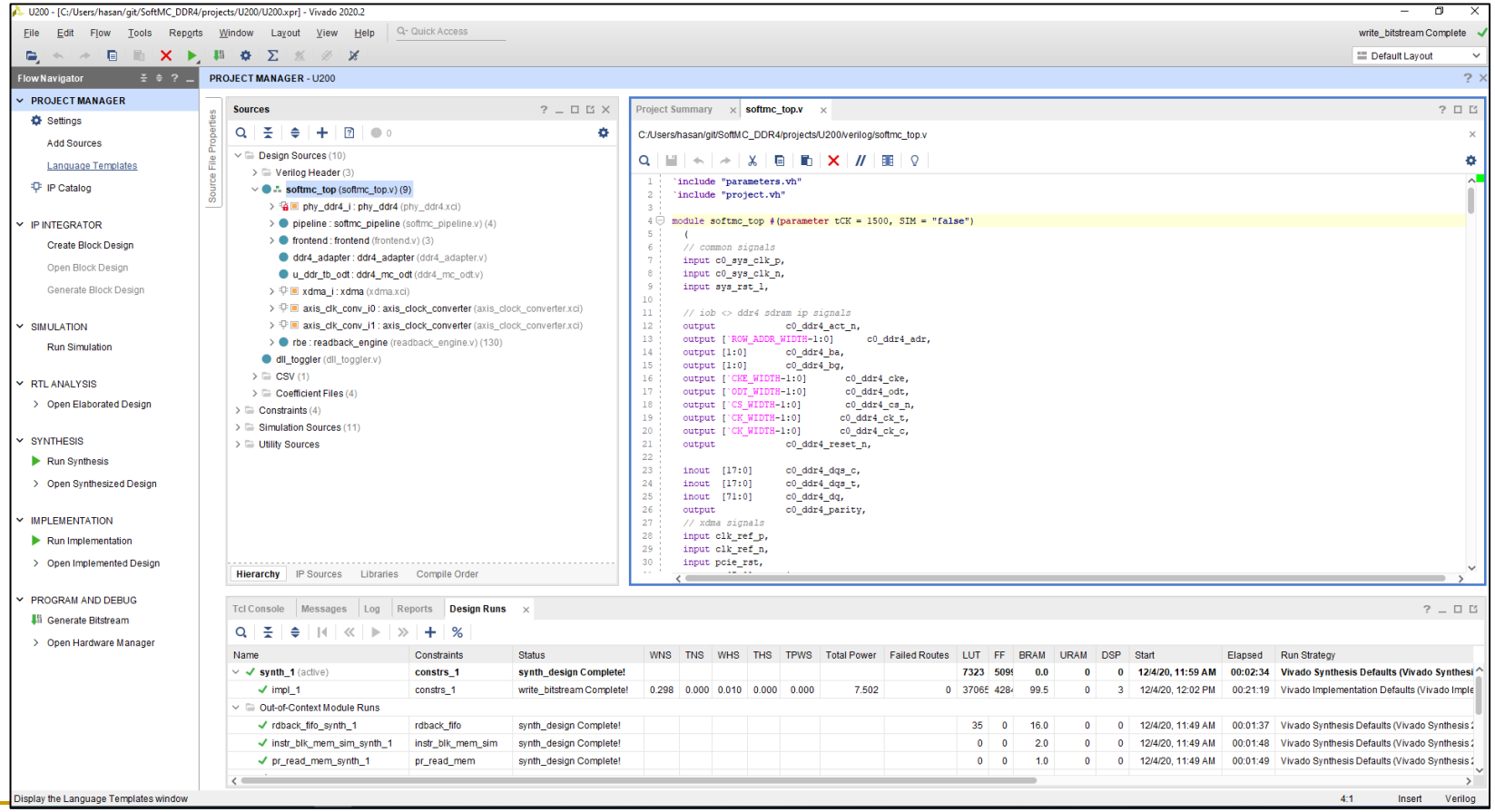
- FPGAs have many resources (e.g., LUTs, switches)
- They are hard to program manually
- How can we
  - represent a high-level functional description of our hardware circuit using the FPGA resources?
  - select the resources to map our circuit to?
  - optimally configure the interconnect between the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



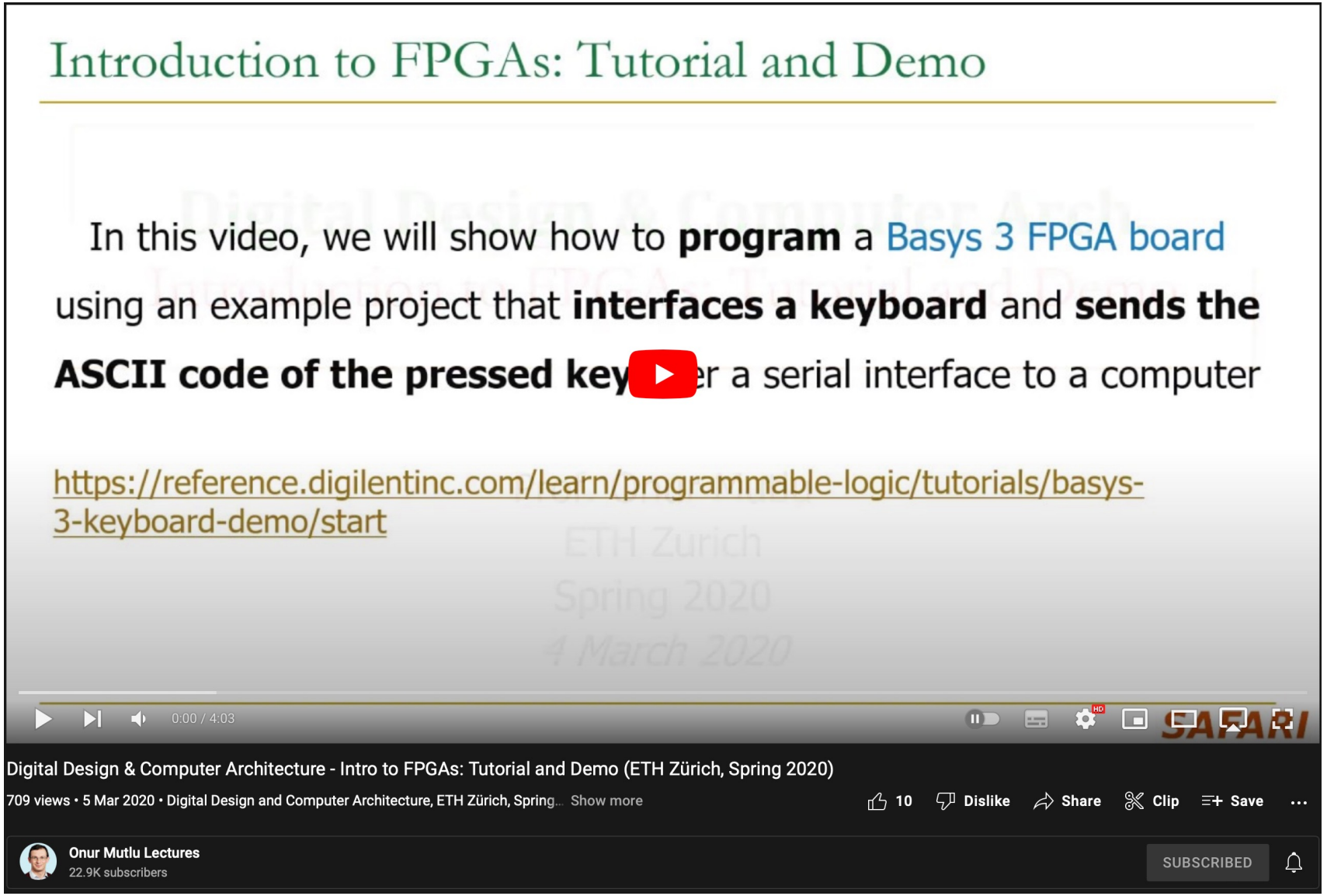
Tutorial and Demo

- 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 will easily be able to develop at the end of semester

- Link to source files
  https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-keyboard-demo/start

Simple Keyboard Demo



http://www.youtube.com/watch?v=33_Z6vrd9qQ

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.

Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Ataberk Olgun)
ETH Zurich
Spring 2023
5 March 2023