

Digital Design & Computer Arch.

Lecture 3b: Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Hasan Hassan)
ETH Zurich
Spring 2020
27 February 2020

Lab Sessions

■ Where?

- **HG E 19, HG E 26.1, HG E 26.3, HG E 27, HG D 11, HG D 12**

■ When?

- Tuesday 15:15-17:00 (E 26.1, E 26.3, E 27)
- Wednesday 15:15-17:00 (E 26.1, E 26.3)
- Friday 08:15-10:00 (D 11, D 12, E 26.3, E 27)
- Friday 10:15-12:00 (E 26.1, E 26.3, E 27)

Grading

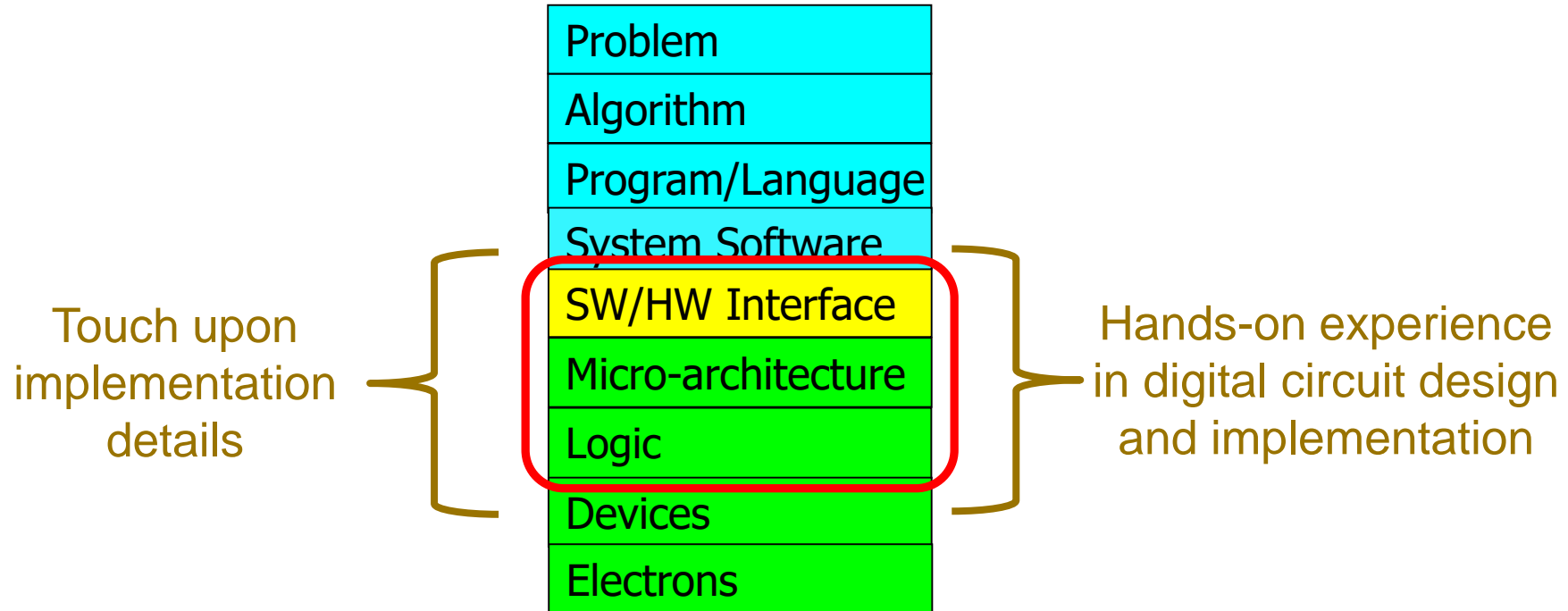
- **10** labs, **30** points in total
- We will put the **lab manuals** online
 - <https://safari.ethz.ch/digitaltechnik/doku.php?id=labs>
- **Grading Policy**
 - In-class evaluation (70%) and *mandatory* lab reports (30%)
 - *1-point penalty for late submission of the report*
 - You can use your grades 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=10483>
You should finish the labs within 1 week after they are announced
- **For questions**
 - Piazza (preferred)
 - digitaltechnik@lists.inf.ethz.ch

Agenda

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

What We Will Learn?

The Transformation Hierarchy



Understanding how a processor works
underneath the software layer

What We Will Learn? (2)

- Considering the **trade-offs** between **performance** and **area/complexity** in your hardware implementation
- **Hands-on experience** on:
 - ❑ Hardware **Prototyping** on FPGA
 - ❑ **Debugging** Your Hardware Implementation
 - ❑ Hardware Description Language (**HDL**)
 - ❑ Hardware Design Flow
 - ❑ Computer-Aided Design (**CAD**) Tools

Agenda

- Logistics
- What We Will learn?
- **FPGAs in Today's Systems**
- Overview of the Lab Exercises
- What is an FPGA?
- 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.”

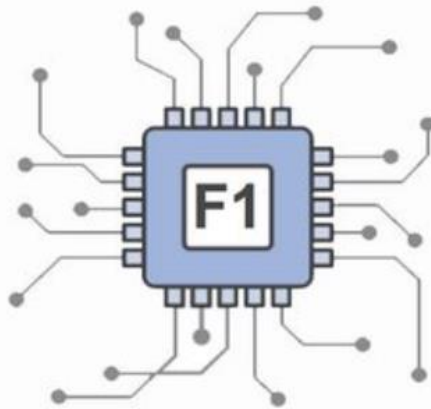


<https://www.microsoft.com/en-us/research/project/project-brainwave/>

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

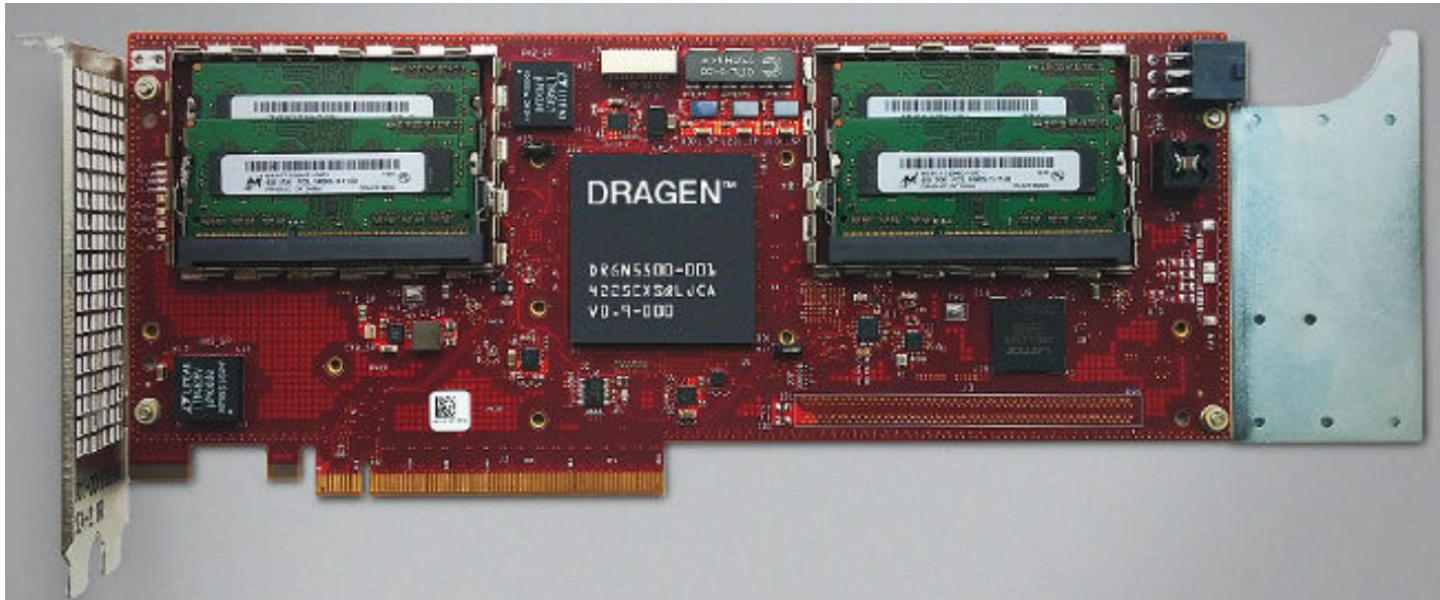
FPGAs in Today's Systems: Amazon EC2 F1

- “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.**”



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: GateKeeper

Bioinformatics



Article Navigation

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

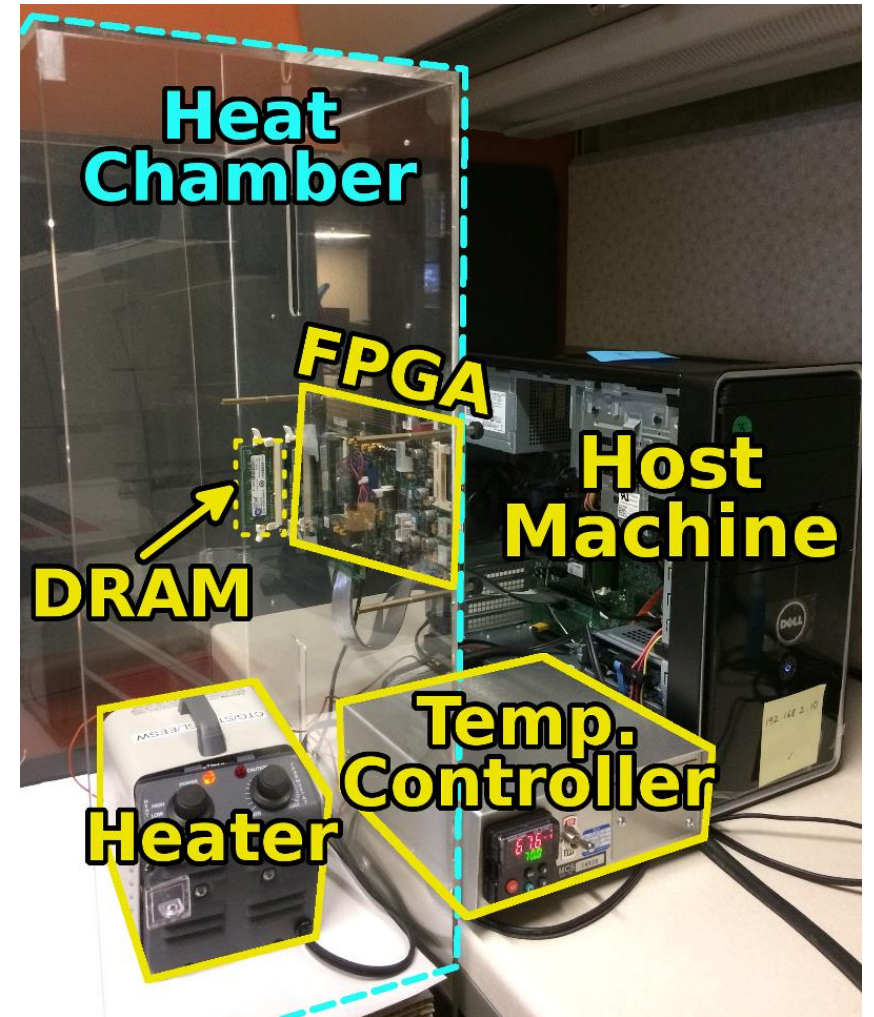
Mohammed Alser ✉, Hasan Hassan, Hongyi Xin, Oğuz Ergin, Onur Mutlu ✉, Can Alkan ✉

Bioinformatics, Volume 33, Issue 21, 01 November 2017, Pages 3355–3363,
<https://doi.org/10.1093/bioinformatics/btx342>
Published: 31 May 2017 **Article history** ▼

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

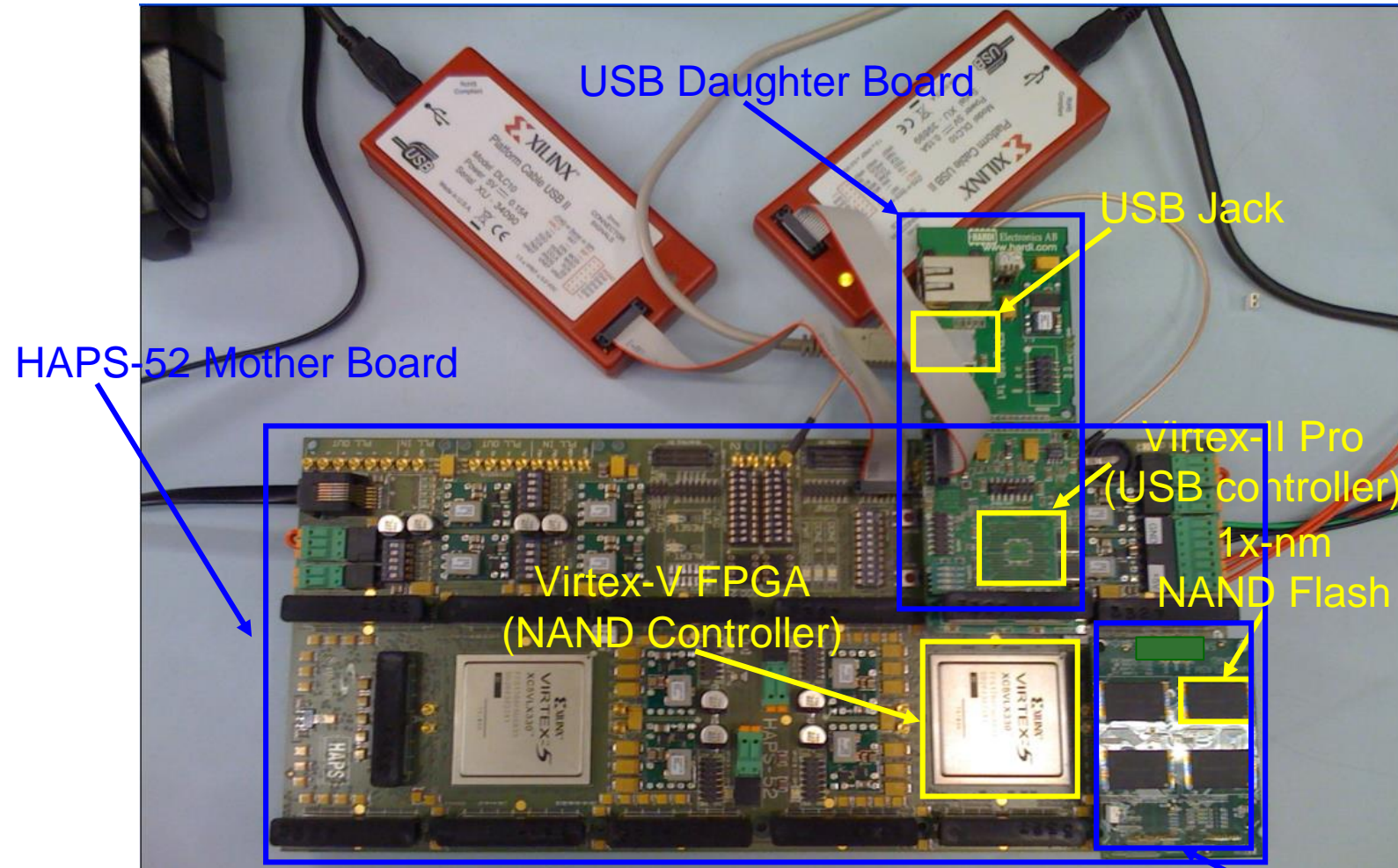
FPGAs in Today's Systems: SoftMC

- 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.

FPGAs in Today's Systems: Characterizing Flash Memories

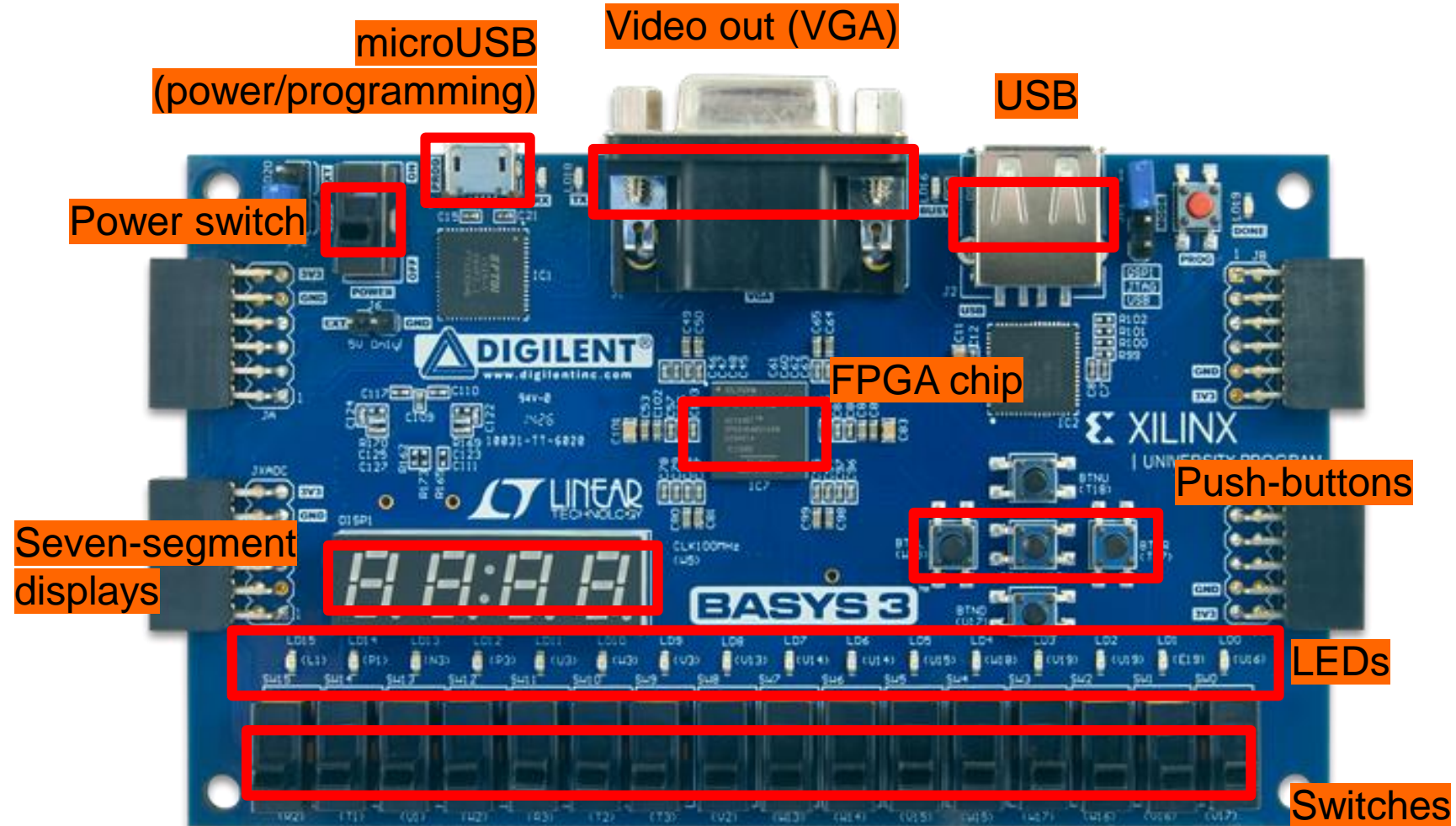


[DATE 2012, ICCD 2012, DATE 2013, ITJ 2013, ICCD 2013, SIGMETRICS 2014, HPCA 2015, DSN 2015, MSST 2015, JSAC 2016, HPCA 2017, DFRWS 2017, PIEEE 2017, HPCA 2018, SIGMETRICS 2018]

Agenda

- Logistics
- What We Will learn?
- FPGAs in Today's Systems
- **Overview of the Lab Exercises**
- What is an FPGA?
- 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
 - It is not possible
to destroy the board **by programming!**

Lab 1: Drawing a Basic Circuit

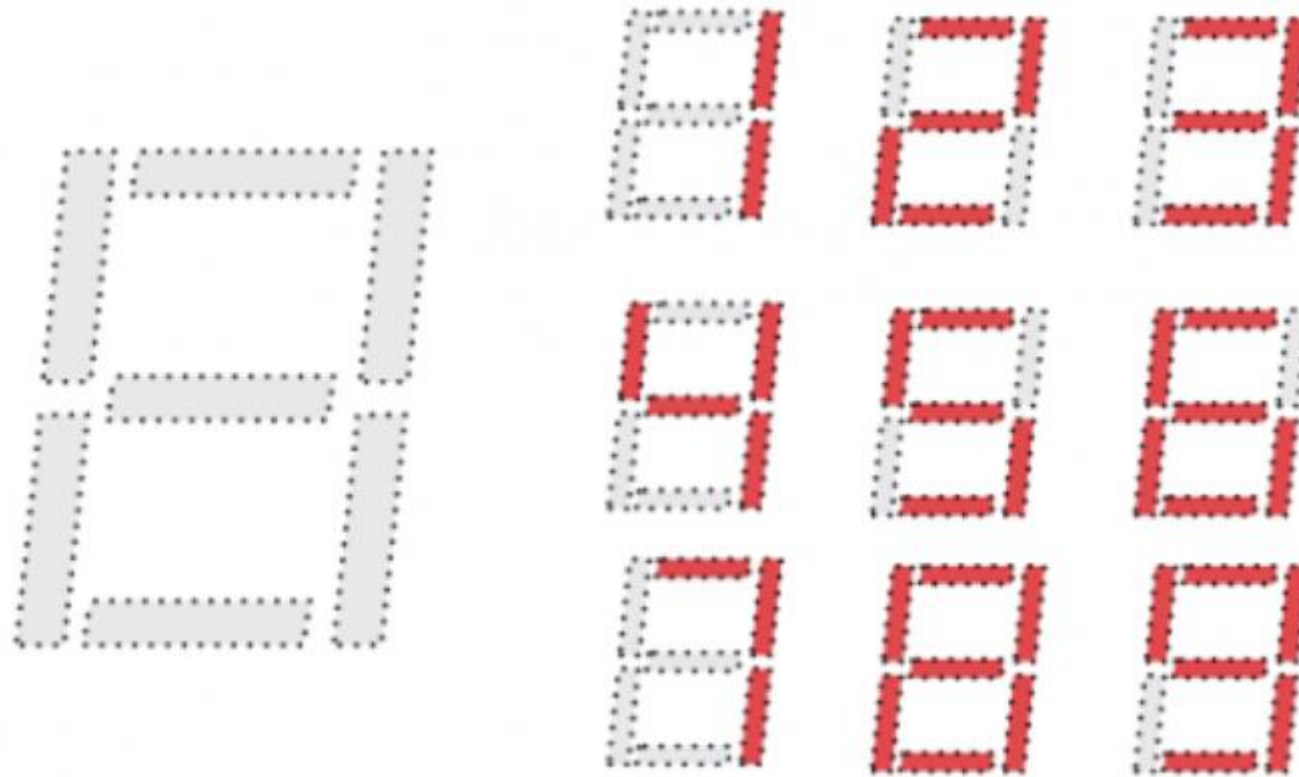
- **Comparison** is a common operation in software programming
 - We usually want to know the **relation** between two variables (e.g., $<$, $>$, $==$, ...)
- We will compare two *electrical signals* (inputs), and find whether they are same
 - The result (output) is also an *electrical signal*
- No FPGA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

Lab 6: Testing the ALU

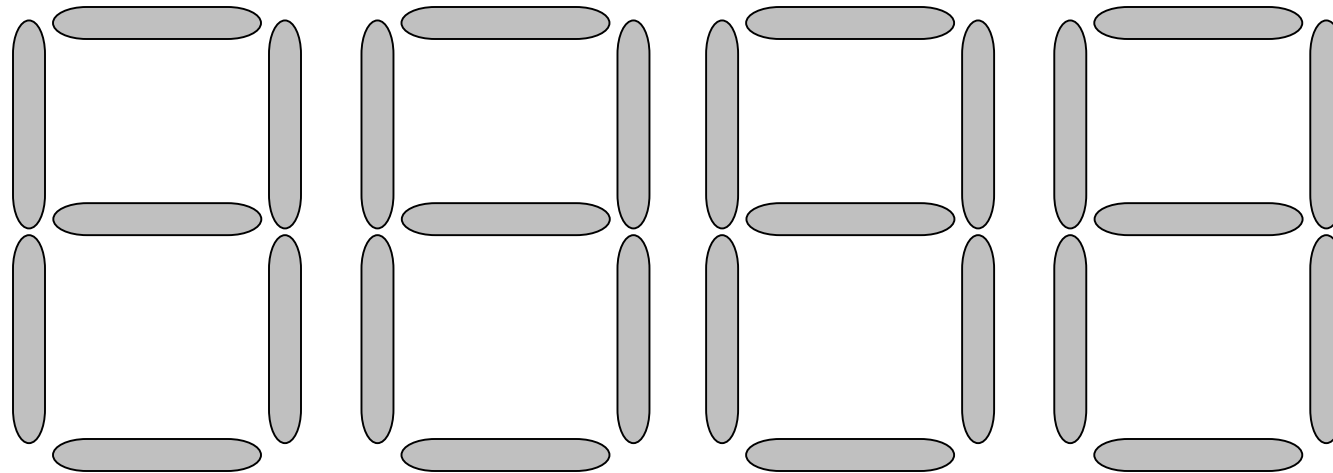
- **Simulate** your design from Lab 5
- Learn how to **debug** your implementation to resolve problems

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

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



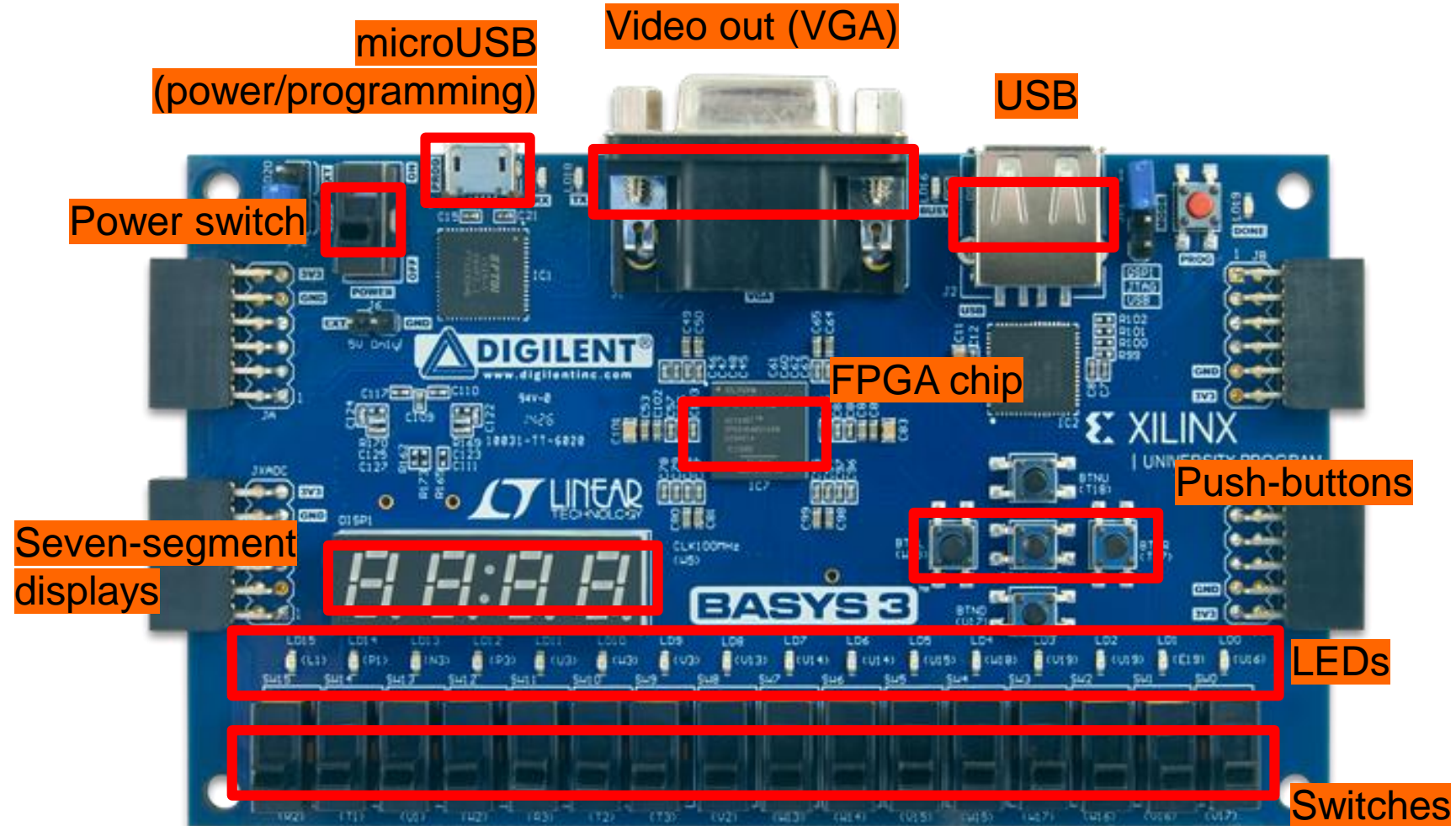
Lab 9: The Performance of MIPS

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

Agenda

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

Basys 3: Our FPGA Board



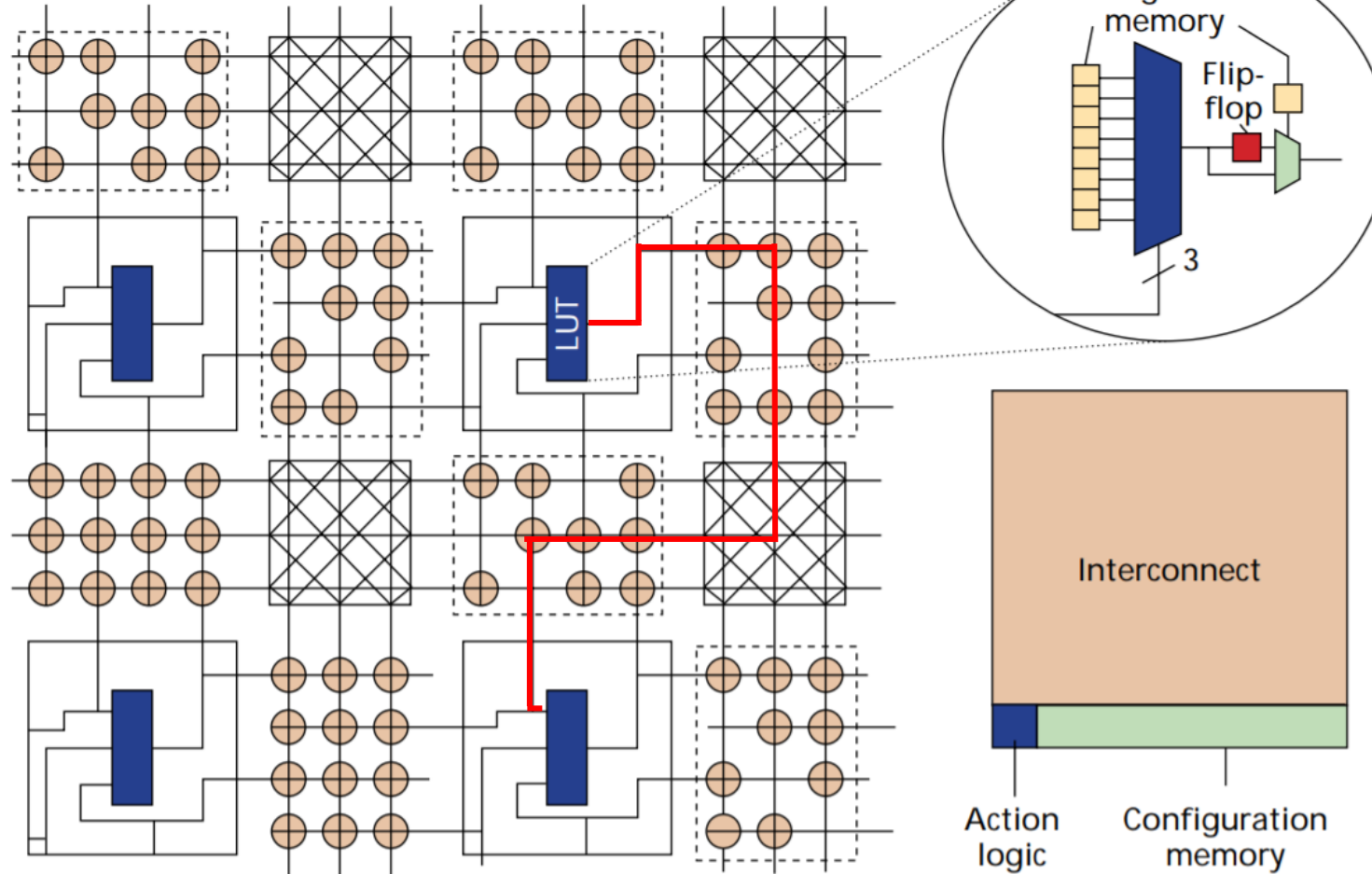
<https://reference.digilentinc.com/reference/programmable-logic/basys-3/start>

What is an FPGA?

- Field Programmable Gate Array
- FPGA is a **reconfigurable** substrate
 - Reconfigurable **functions**
 - Reconfigurable **interconnection** of functions
 - Reconfigurable **input/output (IO)**
 - ...
- FPGAs fill the gap between **software** and **hardware**
 - Achieves **higher performance** than **software**
 - Maintains **more flexibility** than **hardware**

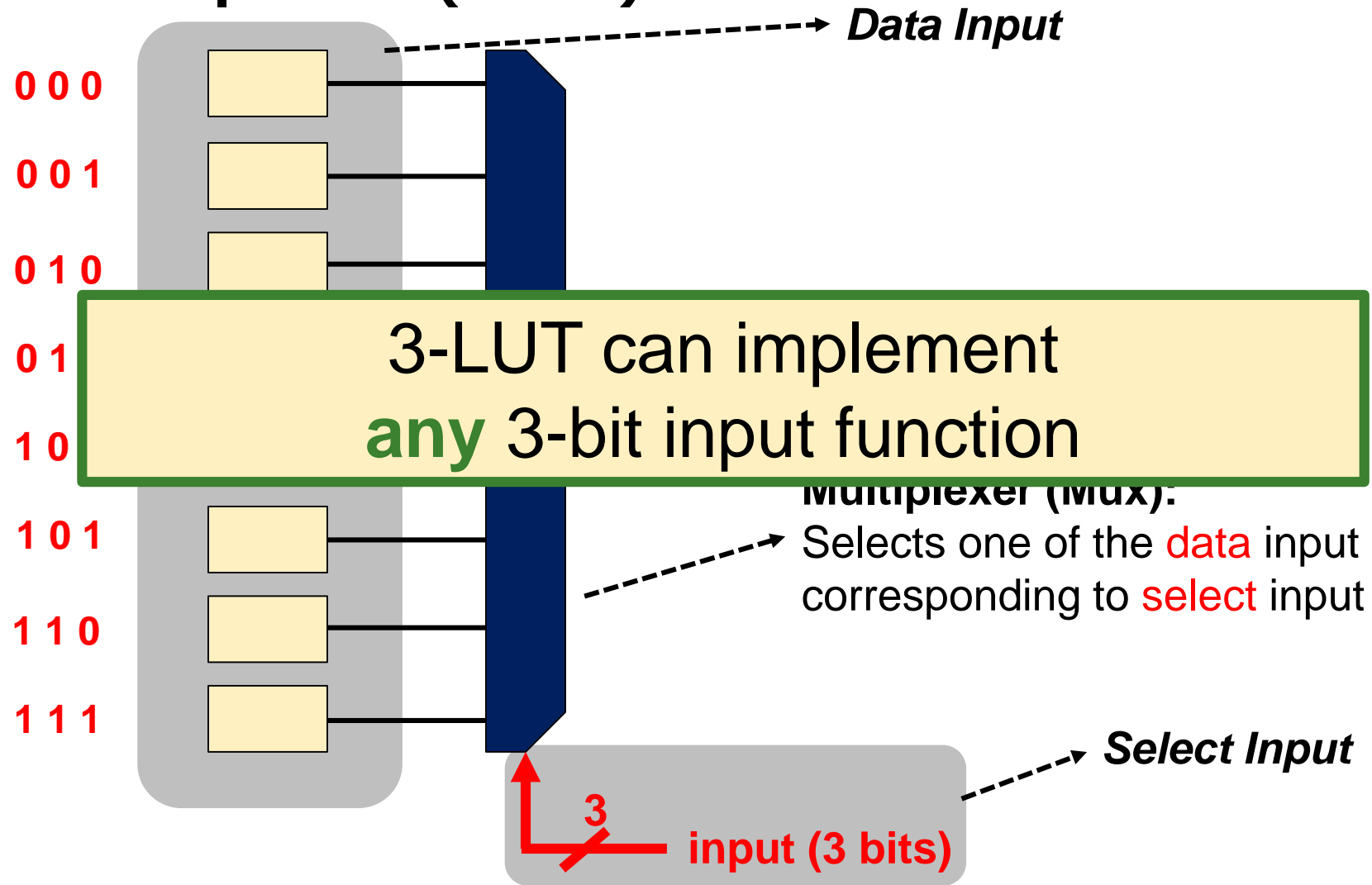
FPGA Architecture - Looking Inside an FPGA

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



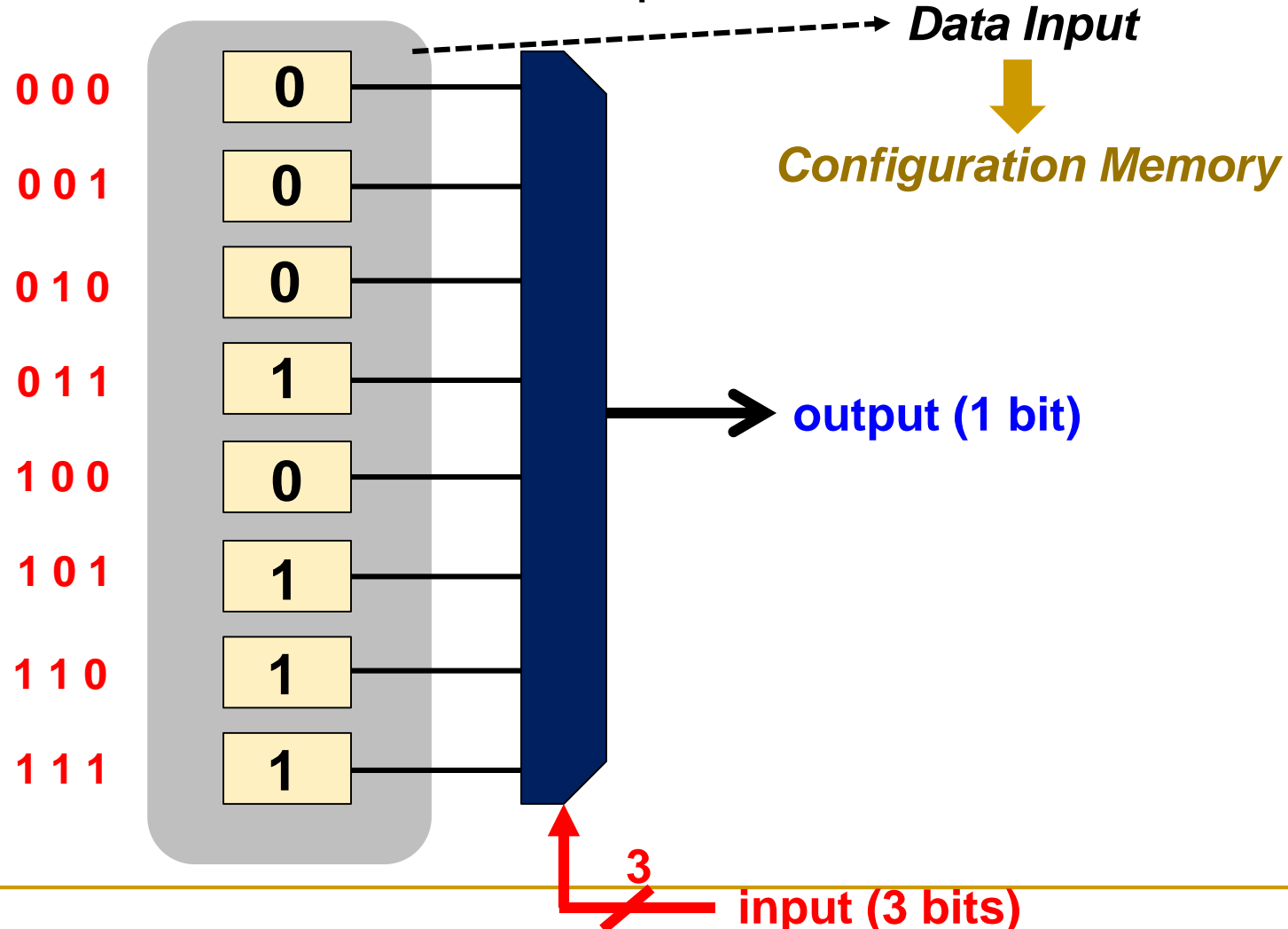
How Do We Program LUTs?

■ 3-bit input LUT (3-LUT)



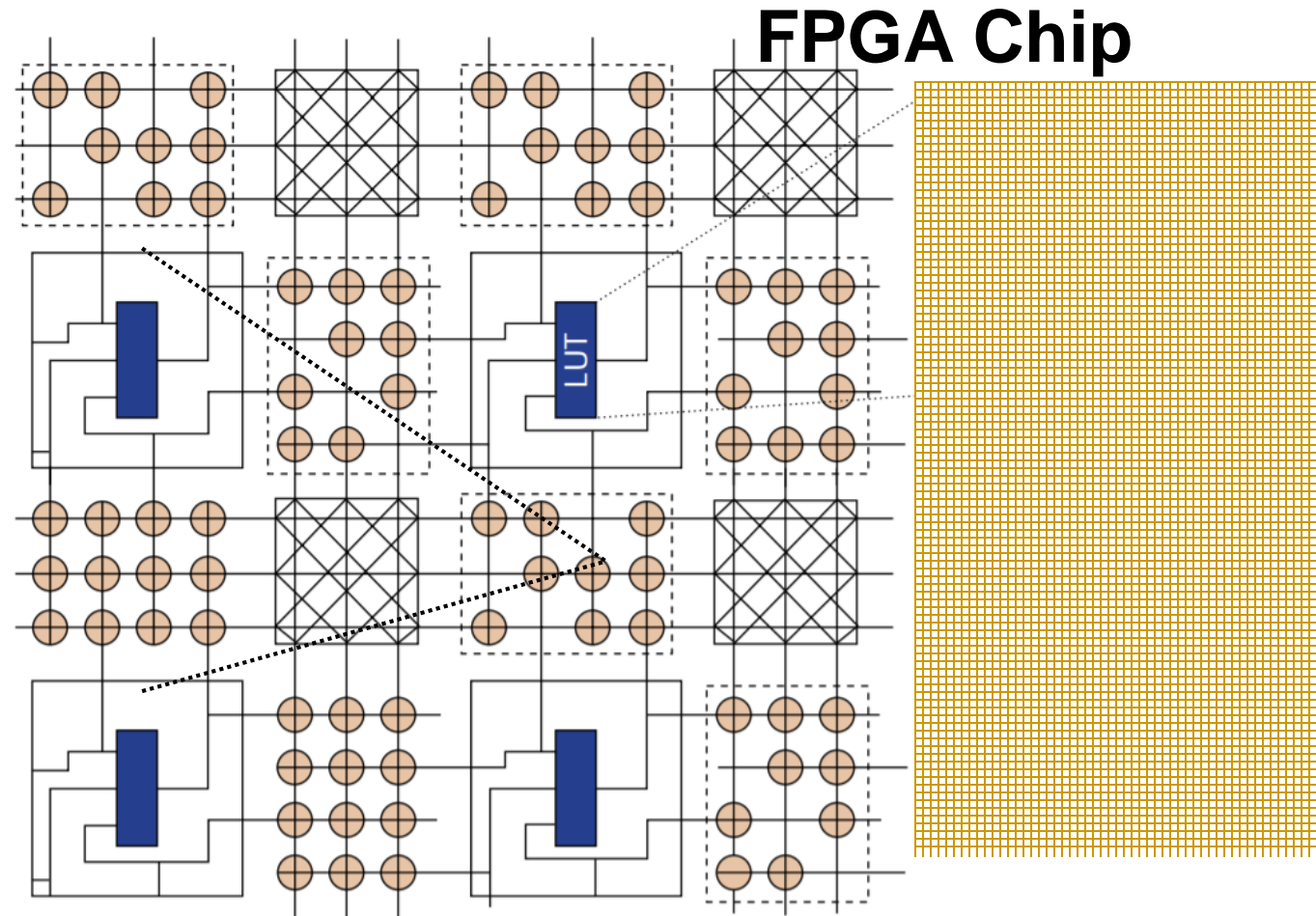
An Example of Programming a LUT

- Let's implement a function that outputs '1' when there are more than one '1' in select inputs



How to Implement Complex Functions?

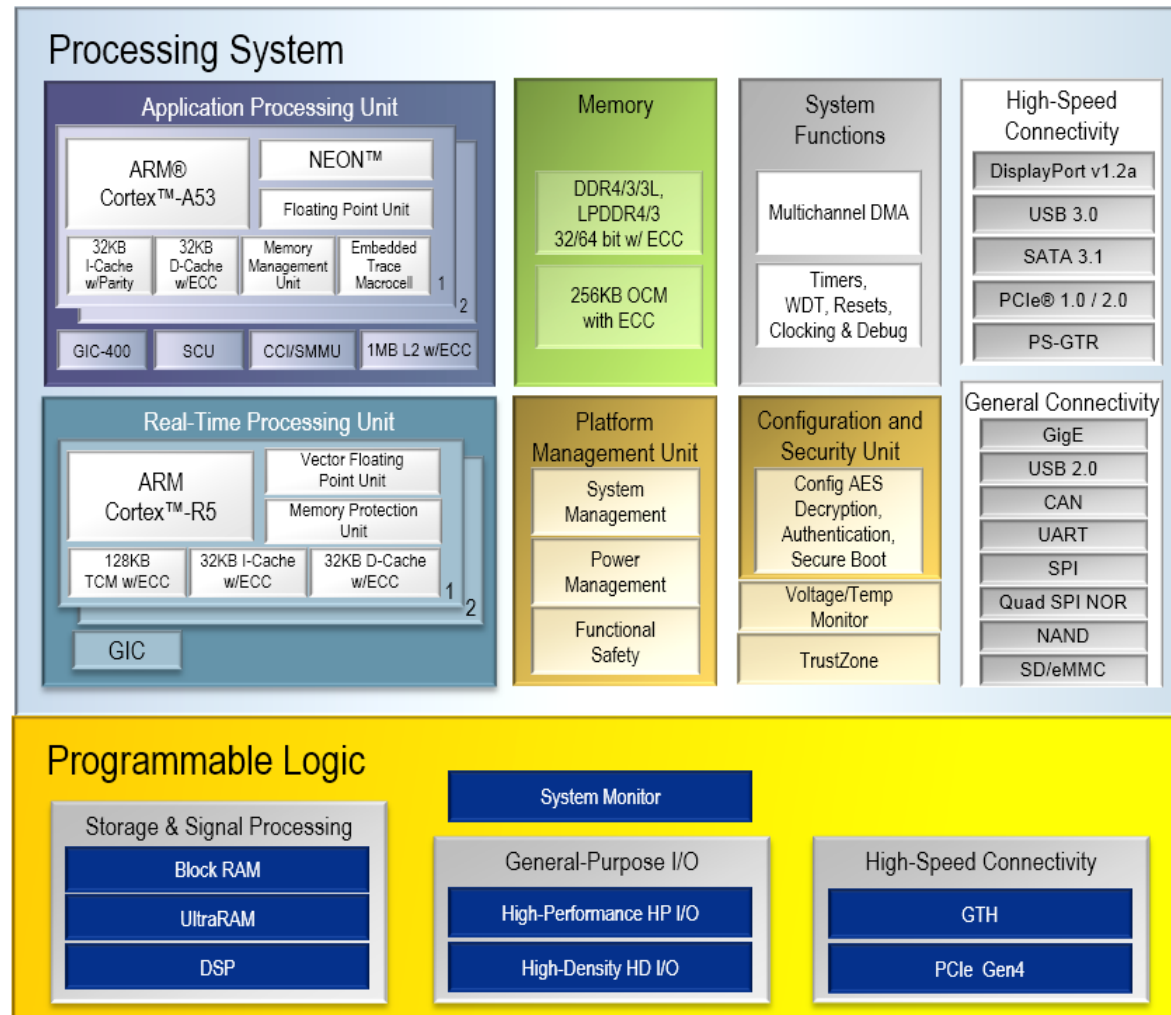
- FPGAs are composed of **many** LUTs and switches



Modern FPGA Architectures

- Typically 6-LUTs
 - Thousands of them
- MBs 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 processor embedded within the FPGA chip

Xilinx Zynq Ultrascale+



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

Advantages & Disadvantages of FPGAs

■ Advantages

- ❑ Low development cost
- ❑ Short time to market
- ❑ Reconfigurable in the field
- ❑ Reusability
- ❑ An algorithm can be implemented directly in hardware
 - No ISA, high specialization

■ Disadvantages

- ❑ Not as **fast** and **power efficient** as *application specific hardware*
- ❑ Reconfigurability adds significant **area overhead**

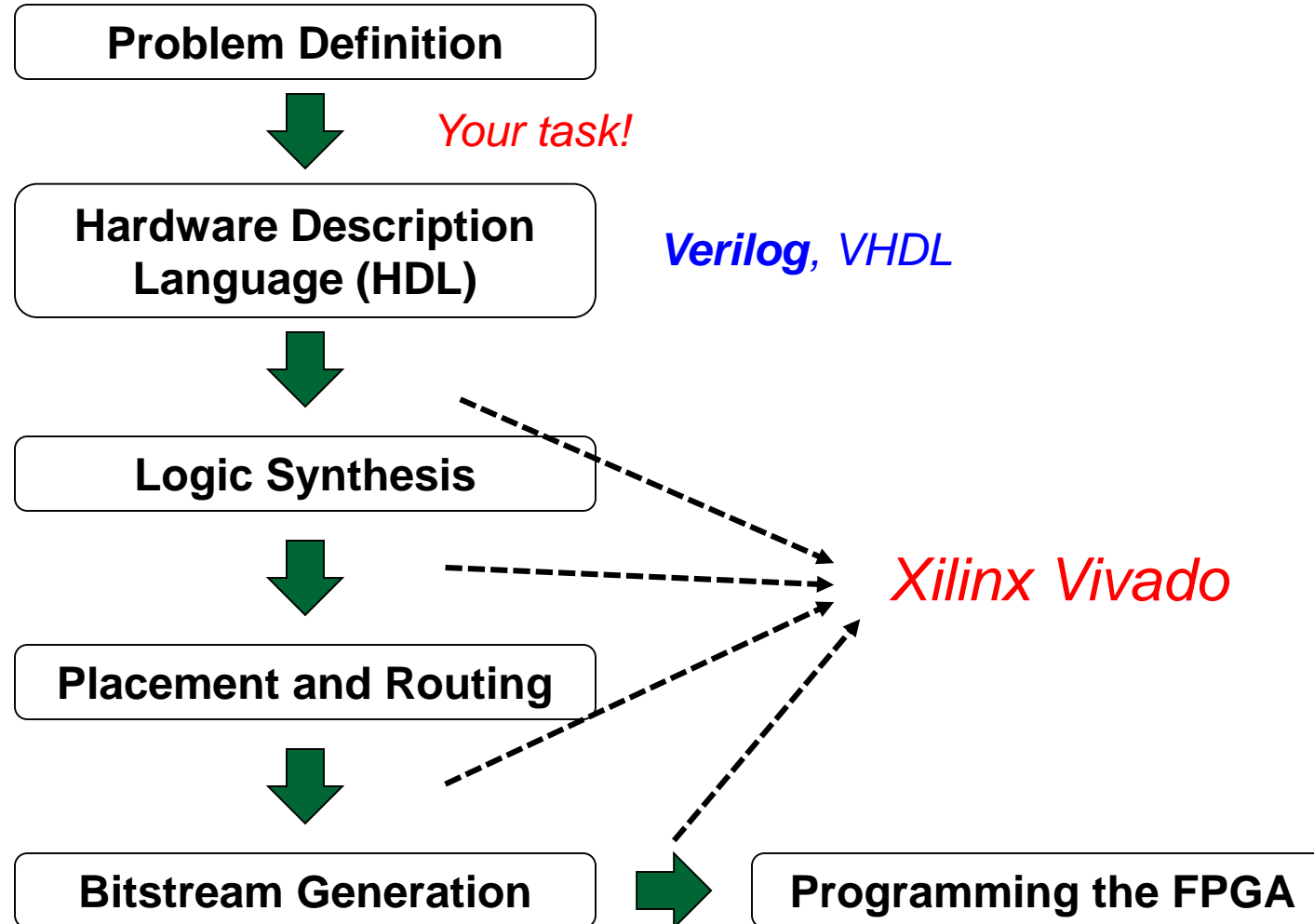
Agenda

- Logistics
- What We Will learn?
- FPGAs in Today's Systems
- Overview of the Lab Exercises
- What is an FPGA?
- **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

- IDE-like software 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 Demo
 - An example for a simple hardware that you can easily develop by the end of semester

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

Today We Covered:

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

Digital Design & Computer Arch.

Lecture 3b: Introduction to the Labs and FPGAs

Prof. Onur Mutlu
(Lecture by Hasan Hassan)
ETH Zurich
Spring 2020
27 February 2020