Introduction

Lecture 1 for Information Processing

Aaron Zhao, Imperial College London, a.zhao@imperial.ac.uk

Introduction - Myself

My name is Aaron, and my research looks at the intersections between algorithm, hardware and security in Deep Learning (DL) Systems. My email is a.zhao@imperial.ac.uk, and my office is 903.

Introduction - Myself

- How do novel hardware architectures improve the efficiency of running ML workloads?
- How to tweak current ML algorithms to make them more efficient on today and future hardware systems?
- ML security related topics with a hardware spin.

Introduction - People

The course is taught by me and Sarim Baig:

- I cover the FPGA parts
- Sarim covers the AWS parts

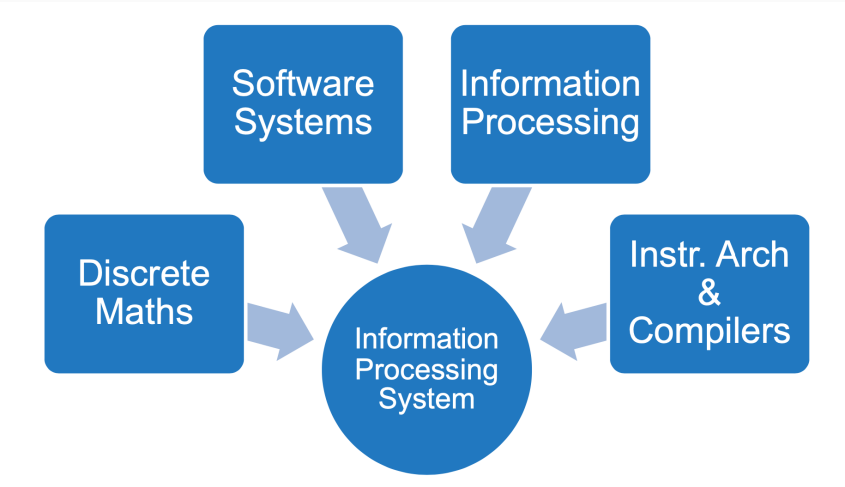
Introduction - Where to find stuff?

Everything is online

- The course webpage (`https://aaron-zhao123.github.io/teaching/info_eng`)
- The Labs (1-4) are in a Github Repository (`https://github.com/Aaron-Zhao123/ELEC50009`)
  - Do not push to this repository.
  - Fork it.
- If you do not understand what is push and fork, check this link: `https://www.youtube.com/watch?v=nT8KGYVurIU&ab_channel=TheCodex`

Objectives and delivery

- Bring together theory and application from other modules
- Create an information processing system
- Project-based learning and integration of knowledge

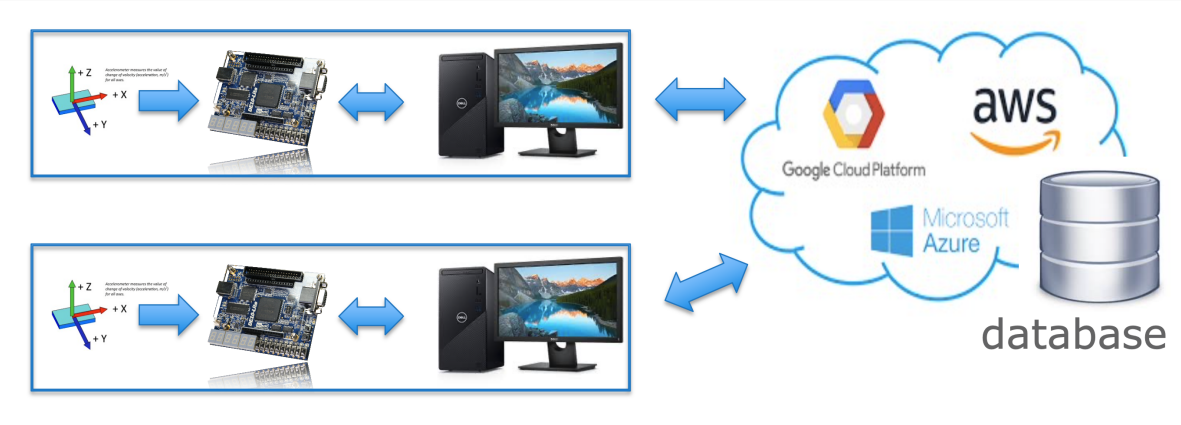


Intended Learning Outcomes

- Design an information processing system that captures, analyses, manages, and outputs signals
- Implement an information processing system using a combination of software, hardware, networks, and databases
- Optimise a system to achieve given performance or quality targets

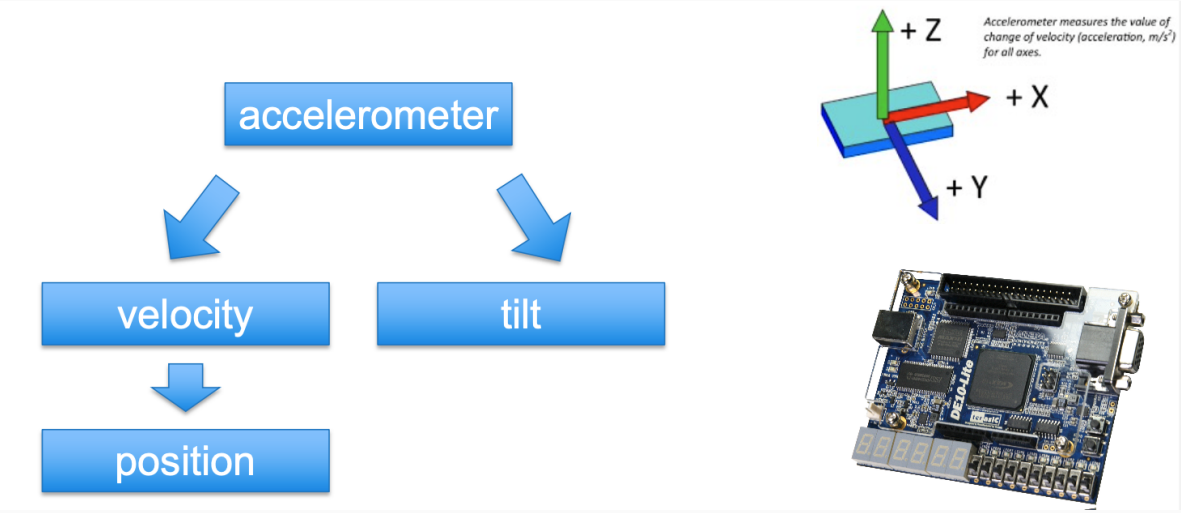
Let's be more specific: Design an IoT system

- Nodes for local (signal) processing of accelerometer data
- Communication to a server
- Integration with a database
- Adapt processing in nodes



Let's be more specific: Accelerometer

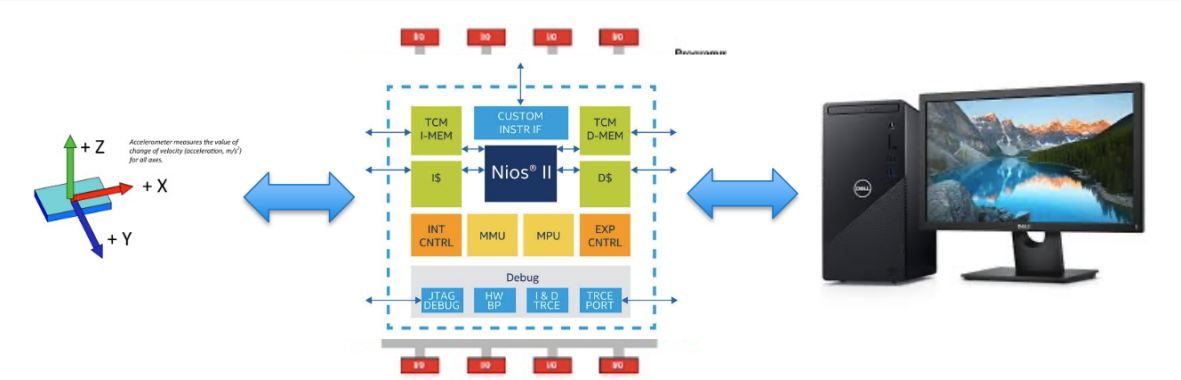
Accelerometer measures the value of change of velocity (acceleration)



Let's be more specific: The Development Board

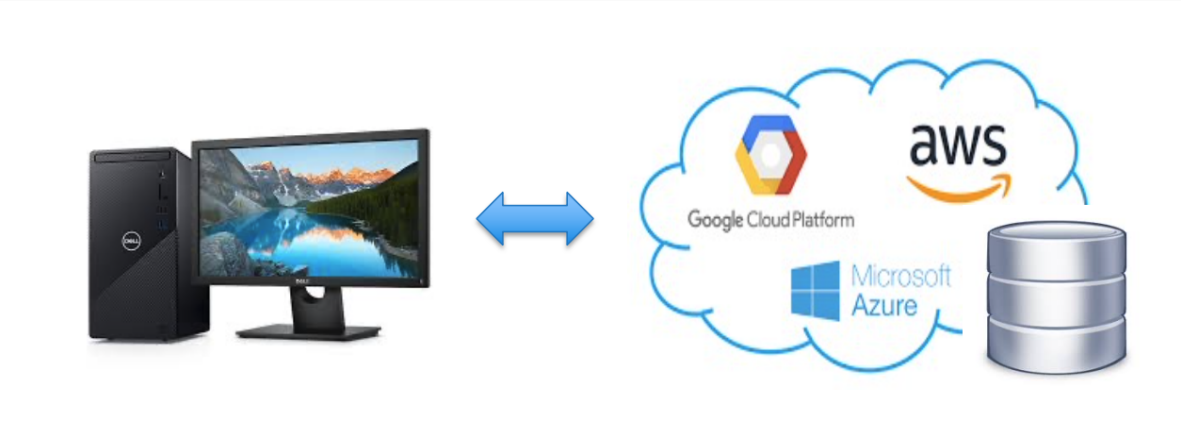
DE10-lite

- FPGA (Intel)
- Instantiation of a soft processor (NIOSII)
- Processing capability (can perform computation) and communication capability (talk to a local PC)



Let's be more specific: AWS DB

- Instantiate and use a database on the cloud
- Communicate from the host PC to the database through network



Structure and dates

- Phase 1 - training (Week 2 - Week 5)
  - Lab based
  - To help you to build an understanding of the system
  - with GTAs for Q&A
- Mid-term Assessment (20%) (Week 6)
  - Lab orals
  - Completion and understanding of the labs
- Phase 2 - group project (Week 7 - Week 10)
  - Functional requirements
  - Non-functional requirements
  - We offer Book an Expert help hours
- Final Assessment (80%) (Week 11)

Phase 1 - Training

- Week 2 (Aaron)
  - Lab 1: Introduction to DE10-Lite. Install tools and learn how to program the device
  - Lab 2: Instantiate a NIOSII system, use the accelerometer
- Week 3 (Aaron)
  - Lab 3: Establish a UART-base communication between the board and the PC
  - Lab 4: Design an IP module for performing a moving average in HW. Connect to NIOSII and process the accelerometer data
- Week 4 (Sarim)
  - Lab 5: Create a remote server in AWS and run a custom service
- Week 5 (Sarim)
  - Lab 6: Create a remote database and perform queries

Phase 2 - Coursework

- In-person and remote working support
- General idea
  - Local node needs to talk to a server (on the cloud).
  - Server needs to talk back, the information needs to propagate back.
- Elaboration
  - Try to see how nodes can affect each other.
  - Detect events, and change the processing in the nodes through a centralised server.
  - Log your events, or perform action on the events.
- Detailed functional and non-functional requirements will be communicated

Logistics

- Lectures (Every Wednesday Weeks 1-5, then in ad-hoc basis as needed)
- Weekly support hours for the Coursework (10min slots).
  - Prepare your questions
  - Book a slot
- Groups and Communications
  - Groups of five or six, have to all come from either group A or B.
  - Private channel on Teams (I will make them)
  - If you cannot find a group, we will make one for you
  - You should start making a group now, deadline for this is 31st Jan
- Course Material
  - Teams
  - Coursework wiki and Github for FPGA related labs

Logistics

- Install tools
  - Virtulbox.
  - Or natively, but make sure you have the right version
  - Or lab machines (Level 1)
- Lab starts tomorrow
- What do I do next?
  - Form a group, declare your group, pick up a board from STORES.
- Any questions?

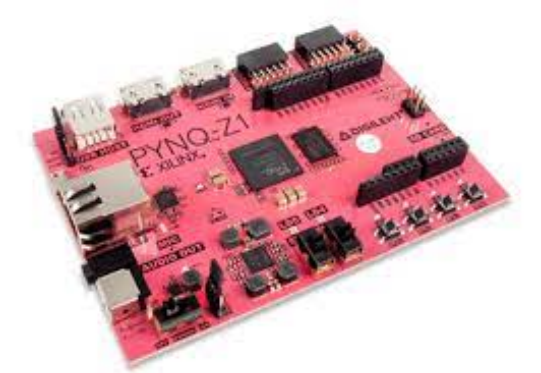
An Introduction to FPGAs

Lecture 2 for Information Processing

Aaron Zhao, Imperial College London, a.zhao@imperial.ac.uk

Why an FPGA is an interesting device to consider

(a) Virtex ultrascale



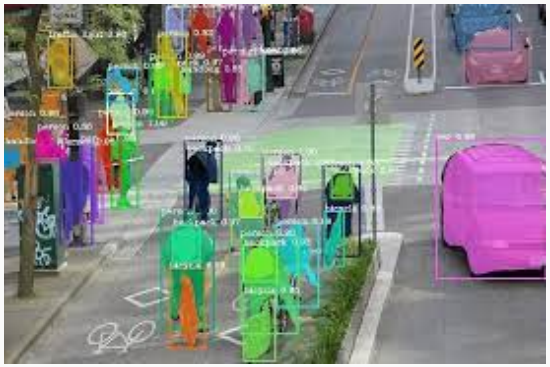
(b) PYNQ Z1

(c) Alveo U250

There is now an incresing need for high-performance systems at low power edge systems

(a) Robotics

(b) UAVs



(c) AI-based Vision systems

High throughput and low latency computation on the cloud

(a) Genetics/genomics

(b) Large-scale simulations

(c) Large AI models

The need for speed and low power

Do we have the 'Tesla"?

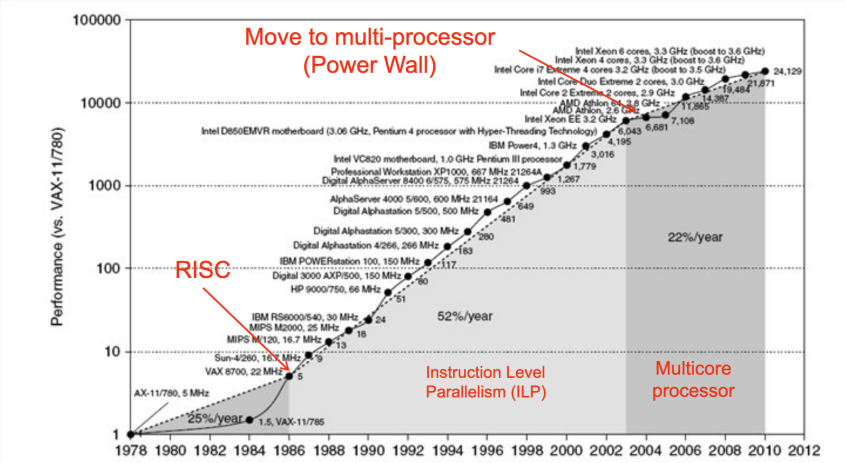
A better metric might be energy efficiency Ops/Second/Watt/Area

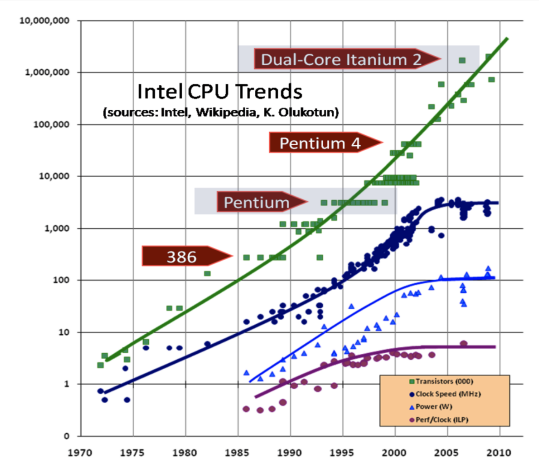
Approach 1: Use existing hardware platforms, but deisng better algorithms or software

- Design or write more “efficient” algorithms
- Use approximations, for instance, subsampling
- Consider the architecture of the system and optimize your program

Approach 2: Buy better hardware

Do nothing. Just wait for the next generation processor.



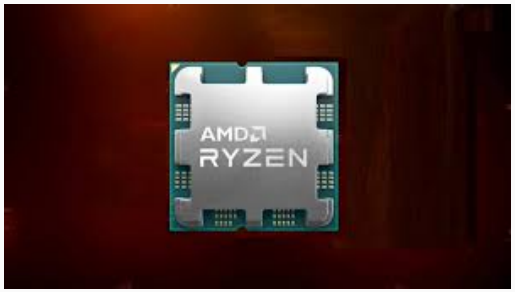


No more free-lunch

Approach 3: Use heterogeneous systems

CPUs combined with other hardware (eg. parallel architectures) on the same SoC.

- Multi- or even Many-core CPUs (AMD Ryzen Threadripper PRO 7995WX, 96 Cores, 192 Threads 2.5GHz and max 5.1GHz)
- Graphic Processing Units (Nvidia H100, 640 Tensor Cores, 128 RT Cores, 80 Streaming Multiprocessors (SMs) and 18,432 CUDA cores)
- Field Programmable Gate Arrays (around 2M Logic Elements/ LUTs, around 5K DSPs)



(a) AMD Ryzen Zen4

(b) Nvidia H100

(c) Intel Stratix 10

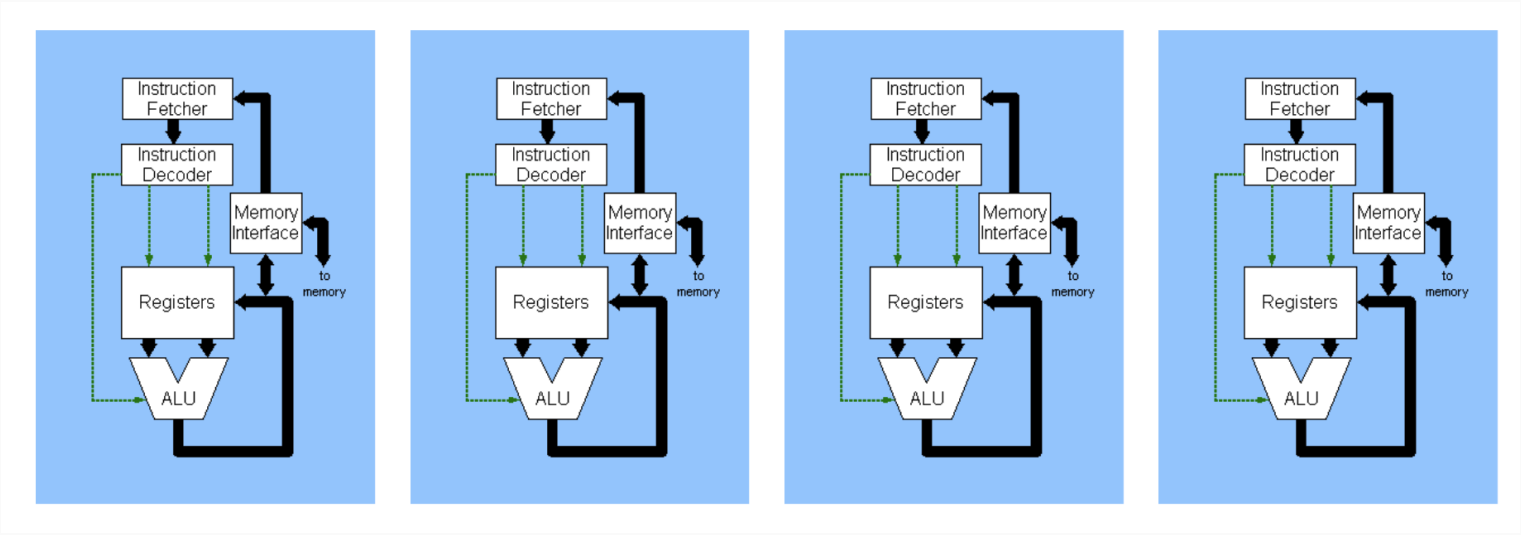
Flexibity and Performance trade-off

- The left-hand side may have better power efficiency on the specific tasks.
- The right-hand side is more flexible and requires shorter dev cycles and efforts.

Benefits come from customising the hardware to the application, and also by tuning your application for the hardware, this is also known as software-hardware co-optimization.

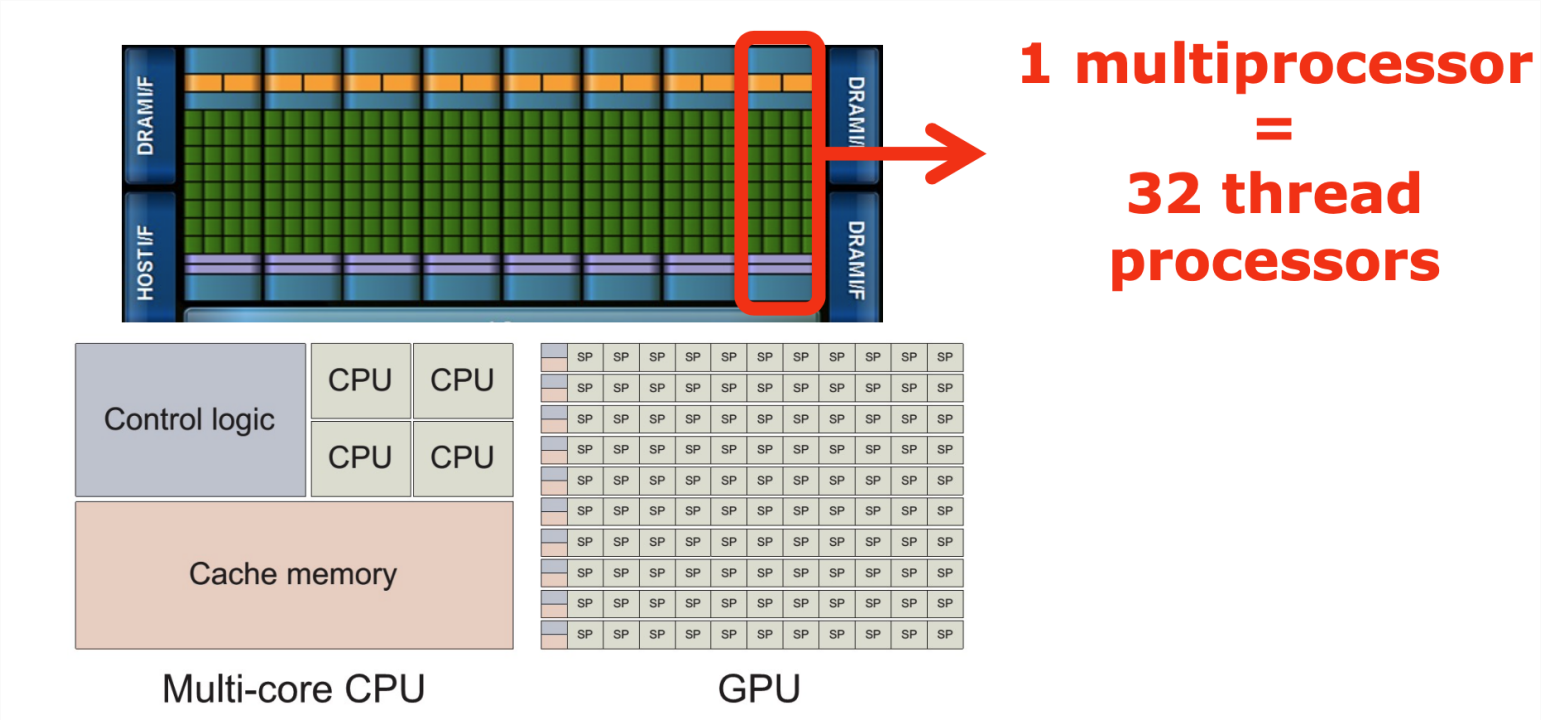
Device Comparison: Multi-core CPUs

- Each core is fairly powerful and runs at a very high frequency.
- Complex memory hierarchy.
- Up to linear speed up (extremely optimistic!).



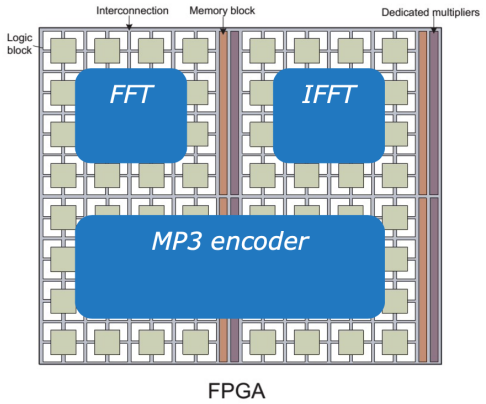
Device Comparison: GPUs

- Many light-weight thread processors (SMs, in Nvidia world) =¿ Hide memory latency.
- All thread processors execute the same sequential code.
- SIMD architecture, massive data parallelism.



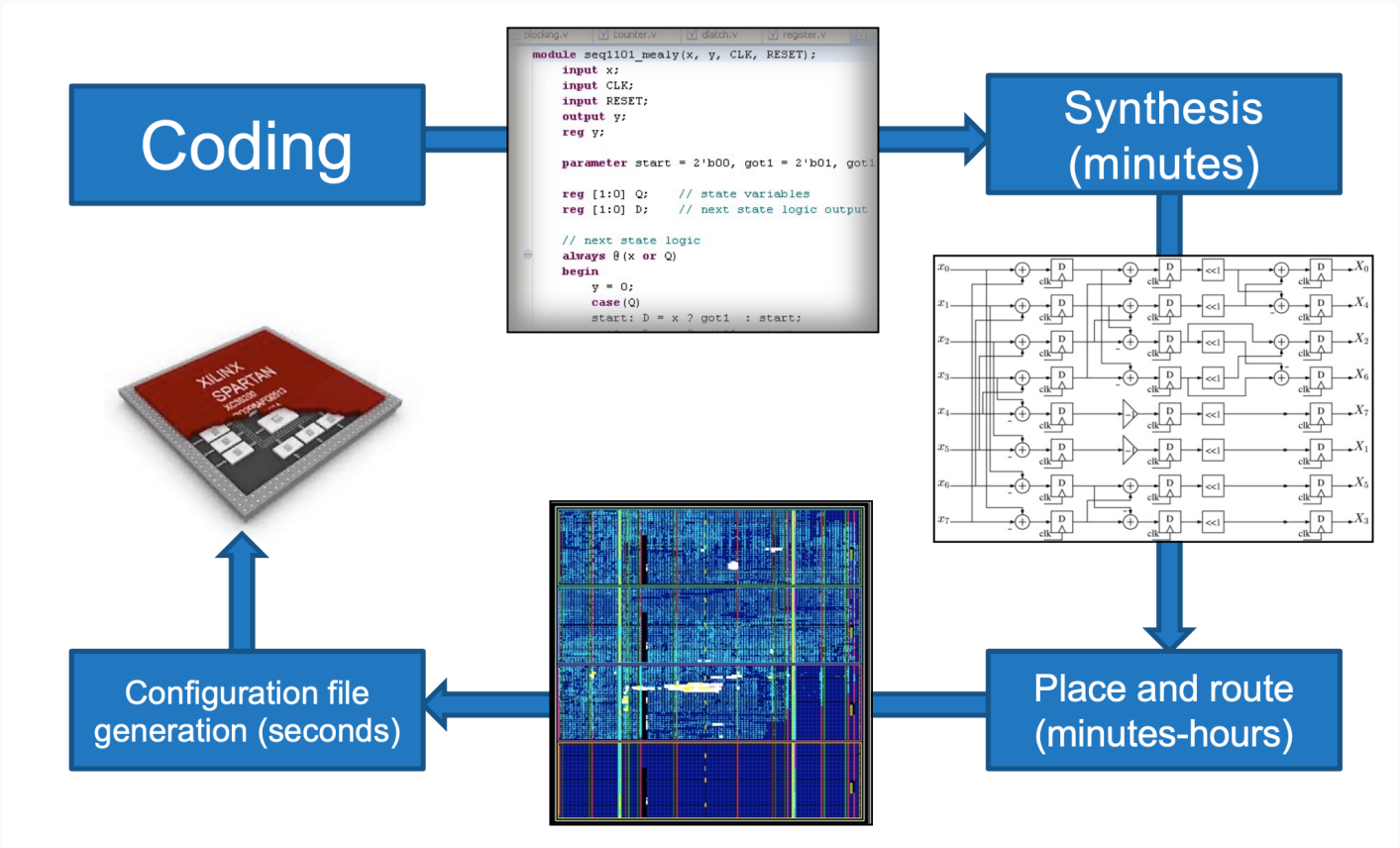
Device Comparison - FPGAs

- (Re-)programmable digital hardware – can implement any digital circuit.
- Can exploit low-level pipeline parallelism
- Logic blocks evaluate simple Boolean functions.
- Interconnection resources connect blocks to implement complex systems.



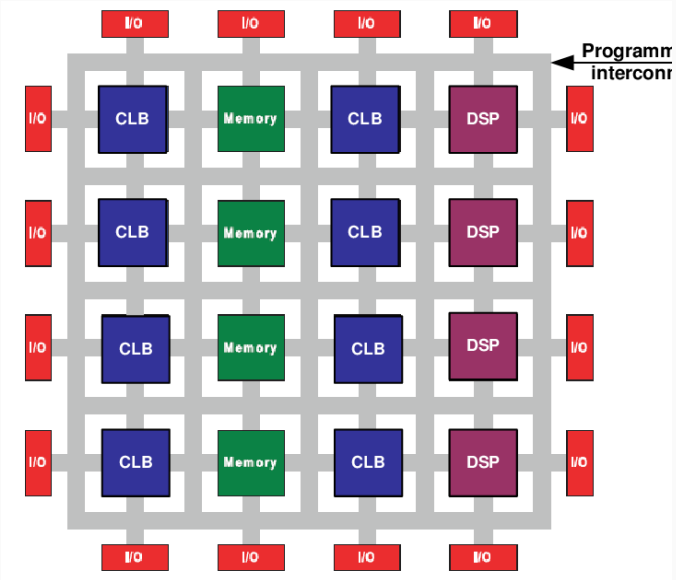
Computation in space

FPGA design flow



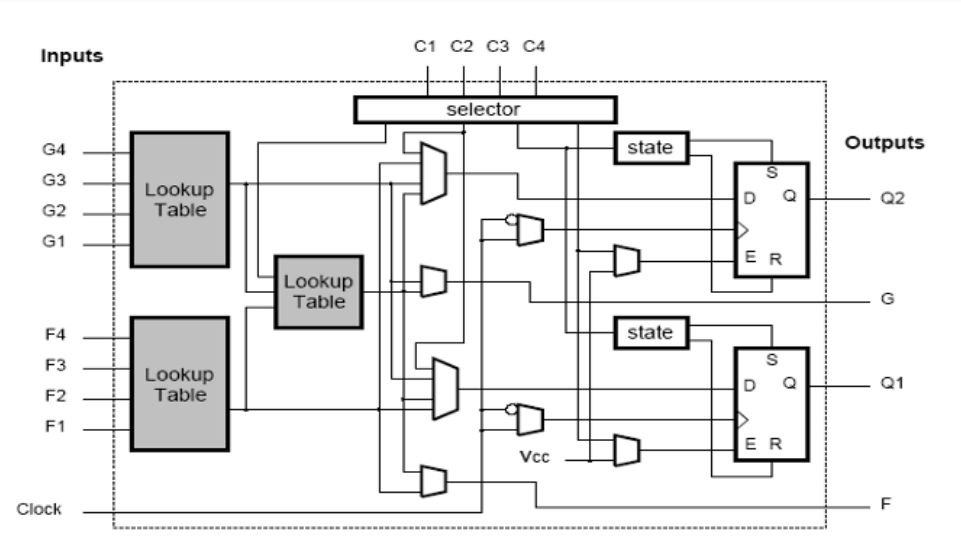
Field Programmable Gate Arrays (FPGAs)

- Xilinx (now part of AMD) is the first to introduce SRAM based FPGA using Lookup Tables (LUTs)
- Components
  - Configurable Logic Block (CLB)
  - IO Block
  - Programmable Interconnects
  - DSPs
  - Memory



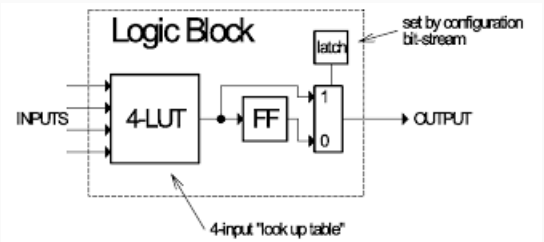
CLB

- Each Configurable Logic Block (CLB) has 2 main Look-up Tables (LUTs) and 2 registers.
- The two LUTs implement two independent logic functions F and G.
- Shown here is the CLB for Xilinx XC4000 devices.

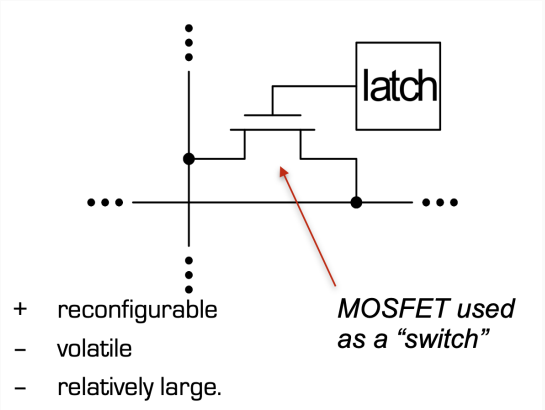


LUT

- LookUp Table (LUT) is implemented using latches:
  - 4-LUT (i.e. 4-input LUT) implements any truth table with 4 inputs, this constructs combinatorial logic functions
  - Requires 24 storage elements, each implemented with a latch (similar to a flip-flop, but half the size roughly, 1-bit memory)
  - Multiplexer select one latch to output
  - "Configuration bit stream" is loaded under user control



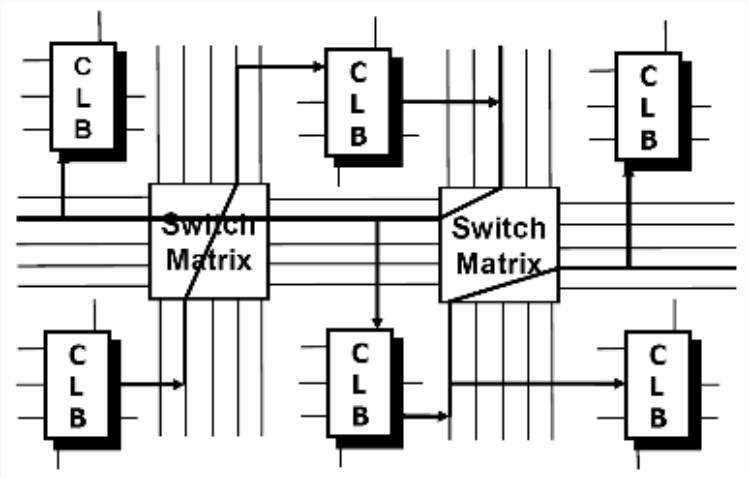
(a) 4-input LUT



(b) Latch

Programmable Interconnect

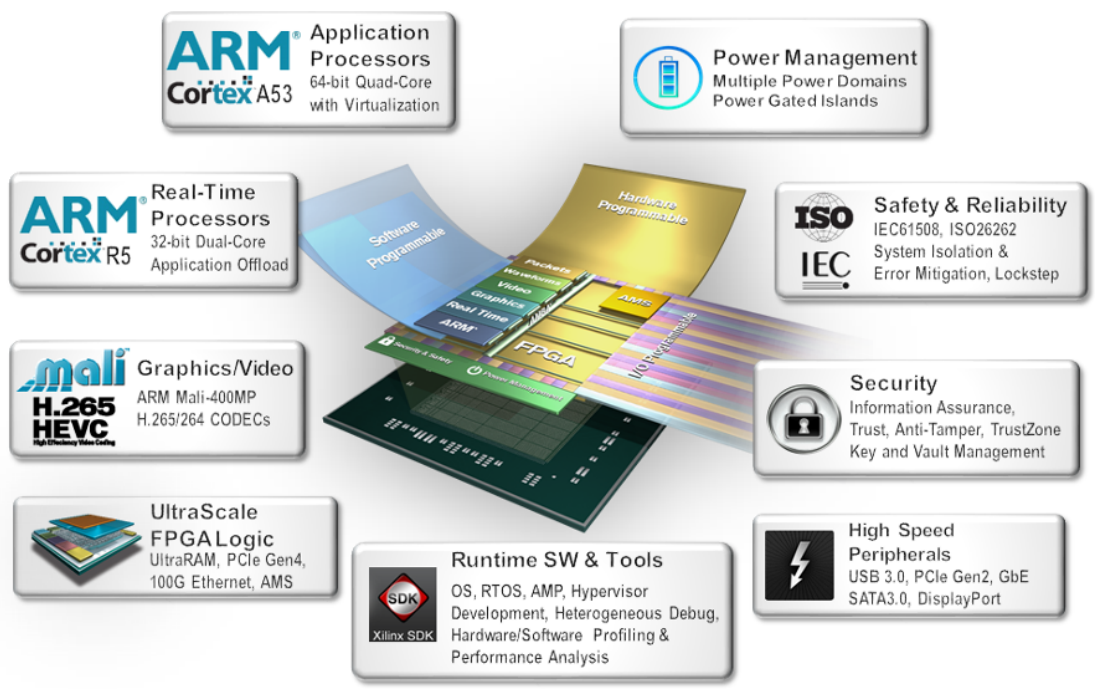
- Switch-box provides programmable interconnect
  - Local interconnects are fast and short
  - Horizontal and vertical interconnects are of various lengths



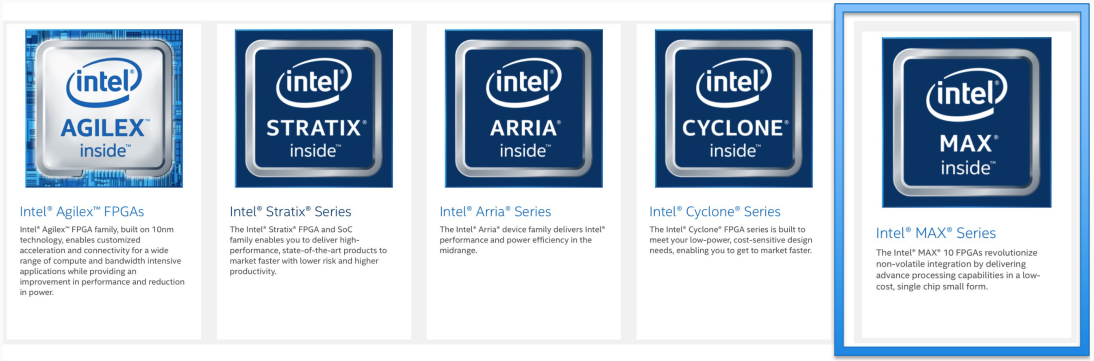
End of the Dinosaurs Age

Modern FPGA devices – Heterogeneous

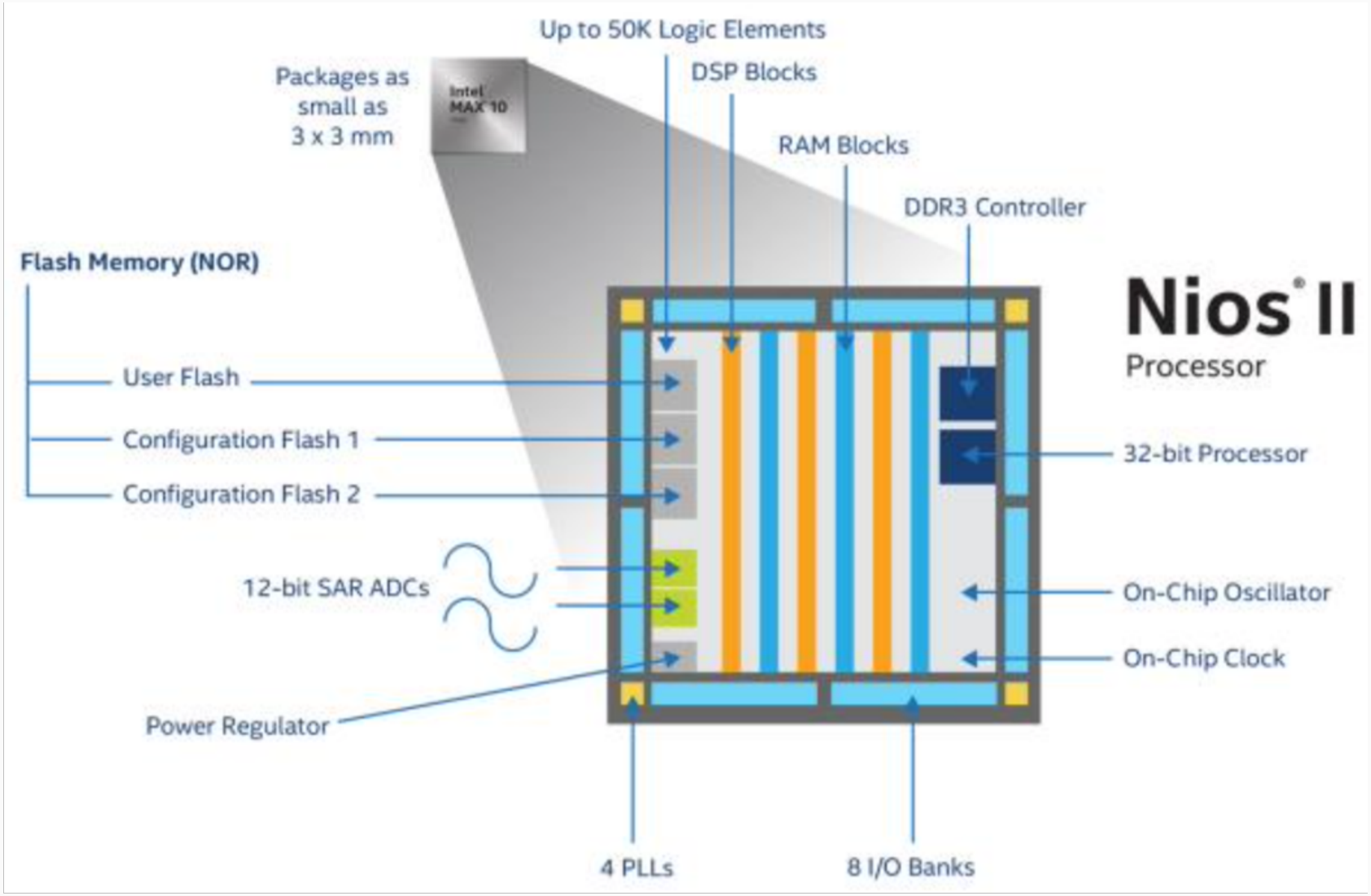
Pre-built ASIC components are already integrated



Current available FPGAs from Intel



NIOSII: the soft core at MAX10



Questions?

Questions?

An Introduction to Lab1

Lecture 3 for Information Processing

Aaron Zhao, Imperial College London, a.zhao@imperial.ac.uk

What is in this lab?

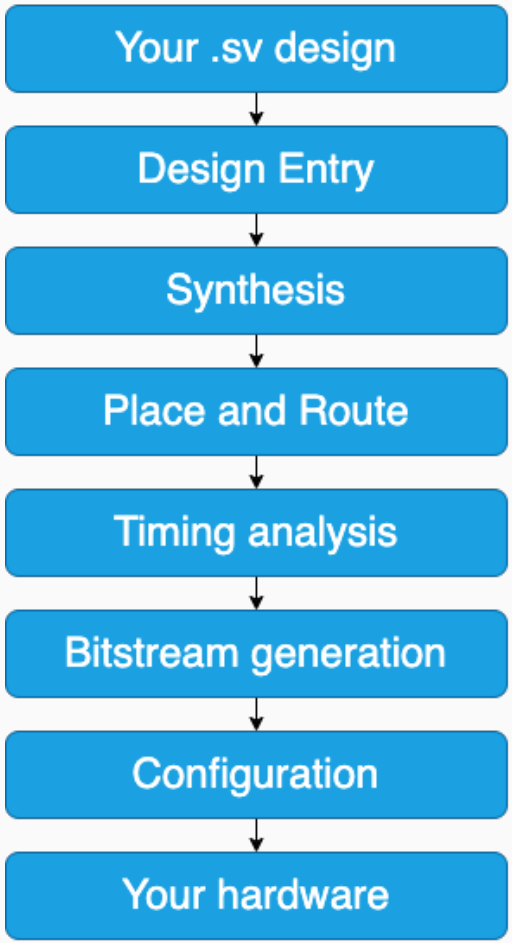
- Setting up Quartus Prime Lite to run.
- Create a directory structure for this and subsequent labs.
- Create a new project in Quartus and complete a basic 7-segment LED display decoder design using Quartus and Verilog.
- Program the MAX10 FPGA chip on the DE10-Lite board with your design.
- Understand the FPGA compilation process.
- Create another project for hex-to-BCD decoding.
- Explore and test your design.

Setting up your environment

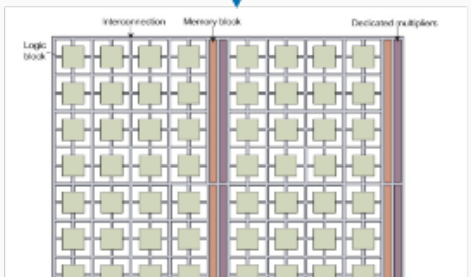
- Quartus Prime (preferred setup is in the VirtualBox or lab machine).
- Explore the programmer: this uses the USB-blaster to program the FPGA from your host machine.
- Explore the Ping Assignment tool.
- Netlist Viewer and Timing Analyzer.
- Be careful with your directory structure!

FPGA Compilation

You will have to go through a number of steps to map your design to the actual FPGA.

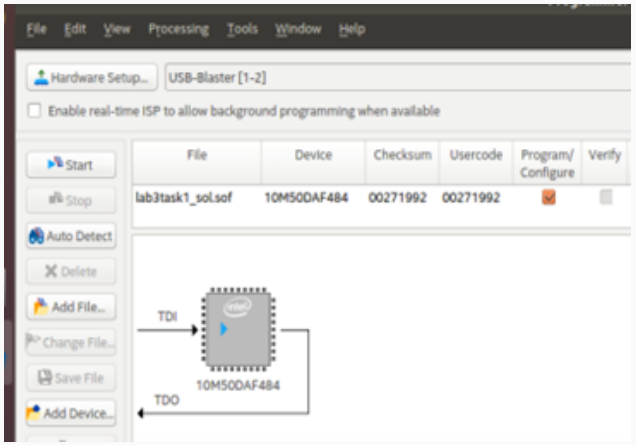



```
//-------------------------------
// Module name: task1_top
// Function: Top level module for Lab 3 Task 1
//           to display 4 switch on a 7-seg display
// Creator:  Peter Cheung
// Version:  1.0
// Date:     31 Oct 2020
//-------------------------------
module  task1_top (
     SW,               // input switches
     HEX0              // Hex output on 7 segment display
   );
  input    [3:0] SW;       // declare input/output ports
  output   [6:0] HEX0;

  hex_to_7seg    SEG0 (HEX0, SW[3:0]);

endmodule
```

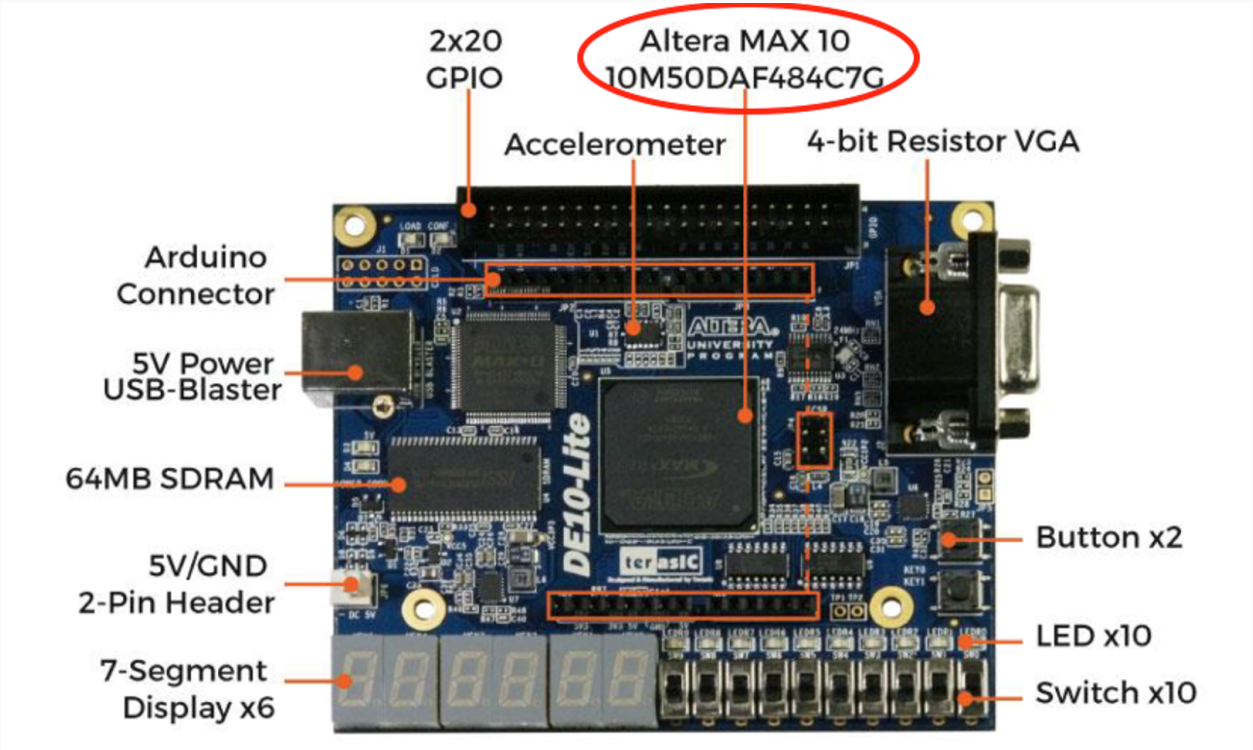


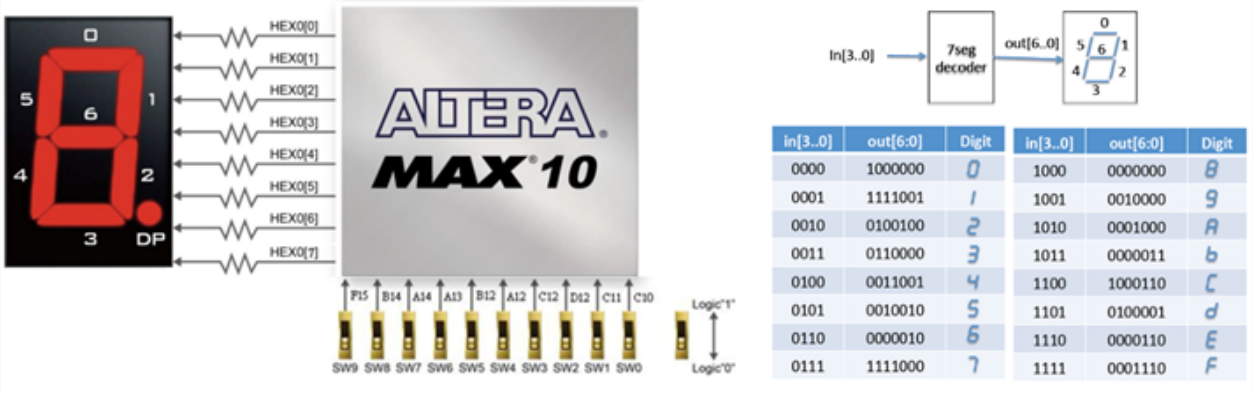
Blasting the FPGA

We have to transfer the bitstream generated on your local PC now to the FPGA device, so we need to use the Programmer in the Quartus toolchain to do this.



7-segment LED display





in[3..0]	out[6:0]	Digit	in[3..0]	out[6:0]	Digit
0000	1000000	0	1000	0000000	8
0001	1111001	1	1001	0010000	9
0010	0100100	2	1010	0001000	A
0011	0110000	3	1011	0000011	b
0100	0011001	4	1100	1000110	C
0101	0010010	5	1101	0100001	d
0110	0000010	6	1110	0000110	E
0111	1111000	7	1111	0001110	F

7-segment LED display - Putting this to Verilog

```
module hex_to_7seg    (out,in);

    output    [6:0] out;     // low-active outp
    input [3:0] in;        // 4-bit binary inp

    reg       [6:0] out;   // make out a varia

    always @ (*)
      case (in)
        4'h0: out = 7'b1000000;
        4'h1: out = 7'b1111001;      //  --0 --
        4'h2: out = 7'b0100100;      //  |     |
        4'h3: out = 7'b0110000;      //  5     1
        4'h4: out = 7'b0011001;      //  |     |
        4'h5: out = 7'b0010010;      //  --6 --
        4'h6: out = 7'b0000010;      //  |     |
        4'h7: out = 7'b1111000;      //  4     2
        4'h8: out = 7'b0000000;      //  |     |
        4'h9: out = 7'b0011000;      //  --3 --
        4'ha: out = 7'b0001000;
        4'hb: out = 7'b0000011;
        4'hc: out = 7'b1000110;
        4'hd: out = 7'b0100001;
        4'he: out = 7'b0000110;
        4'hf: out = 7'b0001110;
      endcase
endmodule
```

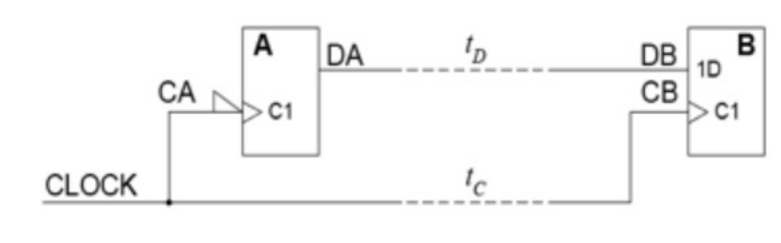
7-segment LED display - Ping assignments

Pin assignment is required, it defines how your block connects to outside world

Signal Name	Pin Location
HEX0[6]	PIN_C17
HEX0[5]	PIN_D17
HEX0[4]	PIN_E16
HEX0[3]	PIN_C16
HEX0[2]	PIN_C15
HEX0[1]	PIN_E15
HEX0[0]	PIN_C14
SW[3]	PIN_C12
SW[2]	PIN_D12
SW[1]	PIN_C11
SW[0]	PIN_C10

Timing Analysis

- Signal Propagation
- Sequential design: How fast can you clock it?
- Timing models in place
- Tools check all possible paths



Questions?

Questions?