Lecture 1

Embedded system design

Introduction

CS476 - ESD
February 18, 2025

Dr. Theo Kluter
EPFL

Motivation
Hw/Sw partitioning
Design example
Hardware
Software

Course build-up and grading

- The first 9 weeks of the semester: theory and practical works (PWs) from which 3 are graded.
- The last 5 weeks of the semester: Realization of a project in groups of 2 students (either proposed by students or chosen from a list) with a final report.
- After the semester: presentations and demo of the project.
- The grading scheme is:
  - 1st graded PW: 15%
  - 2nd graded PW: 17.5%
  - 3rd graded PW: 17.5%
  - Final project: 50%, with following weights:
    - Demo: 10%
    - Presentation: 10%
    - Report: 20%
    - Technical contents (methods used): 40%
    - Quality/organization: 20%
  - Grade = 1 + round(result_in_% * 5 * 4)/4
  - **Important:** For all parts it should be mentioned who of the two students in the group is the author! This to make individual grading possible.
  - **Note:** It is not a requirement that the groups are fixed for all graded parts. It is however important for each graded part to note the name and gaspar number of the group members in the report.

Embedded systems

- There is no formal definition of an *embedded system*, but it is generally accepted that:

  An embedded system is a type of computer designed
  to solve a **specific problem or task**

- This is in contrast to a general-purpose computer such as a laptop or workstation.
- *Embedded systems* typically use a **microprocessor** combined with other hardware and software to solve a specific computing problem.
- The Microprocessor used range from simple (by today's standards) **8-bit versions** to the worlds fastest and most sophisticated **64-bit versions**, or even **multi-core's**.
- *Embedded system* **software** ranges from a small executive to a large **real-time operating system (RTOS)** with a graphical user interface (GUI).
- Typically, the embedded system software must **respond to events** in a **deterministic way** and should be guaranteed **not to crash**.

Embedded systems

The embedded system landscape is as diverse as the world's population:

$\rightarrow$ **no two systems are the same** $\leftarrow$

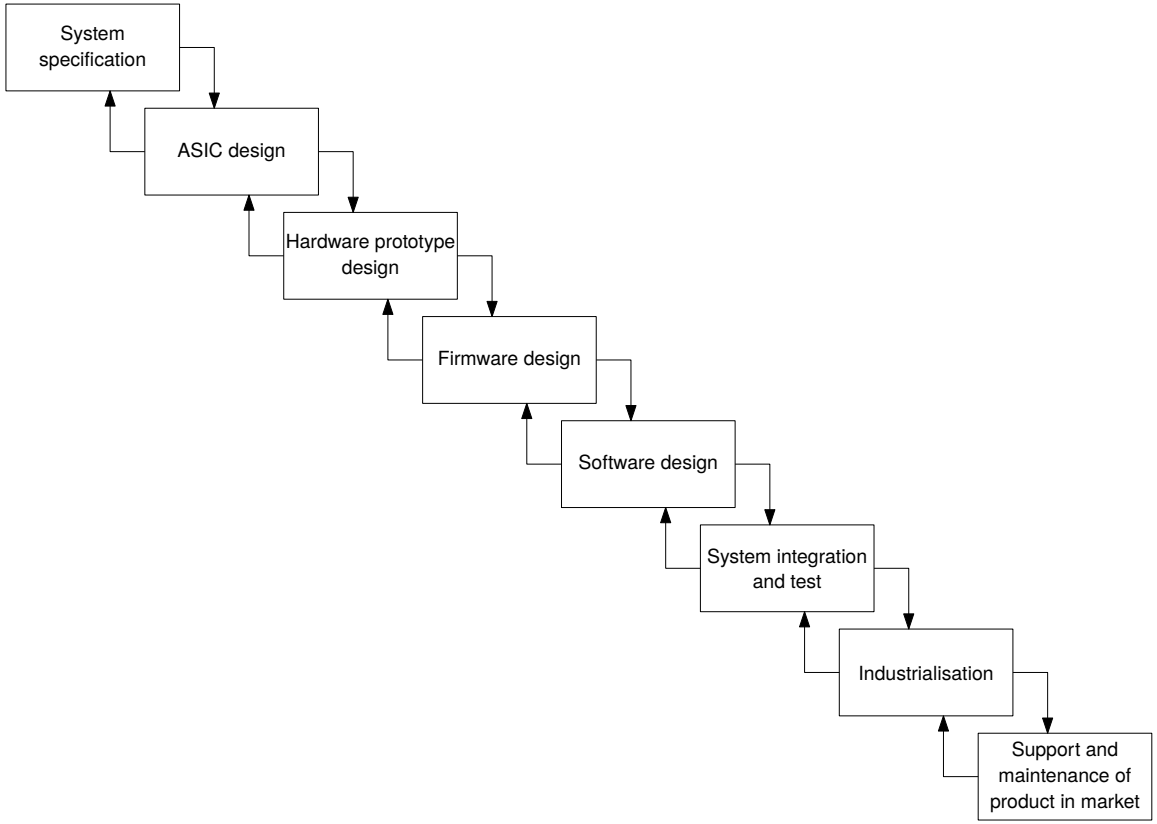
Embedded systems range from large computers such as an air traffic control system to small computers such as a handheld computer that fits into your pocket.

Jason Andrews

Motivation
Hw/Sw partitioning
Design example
Hardware
Software

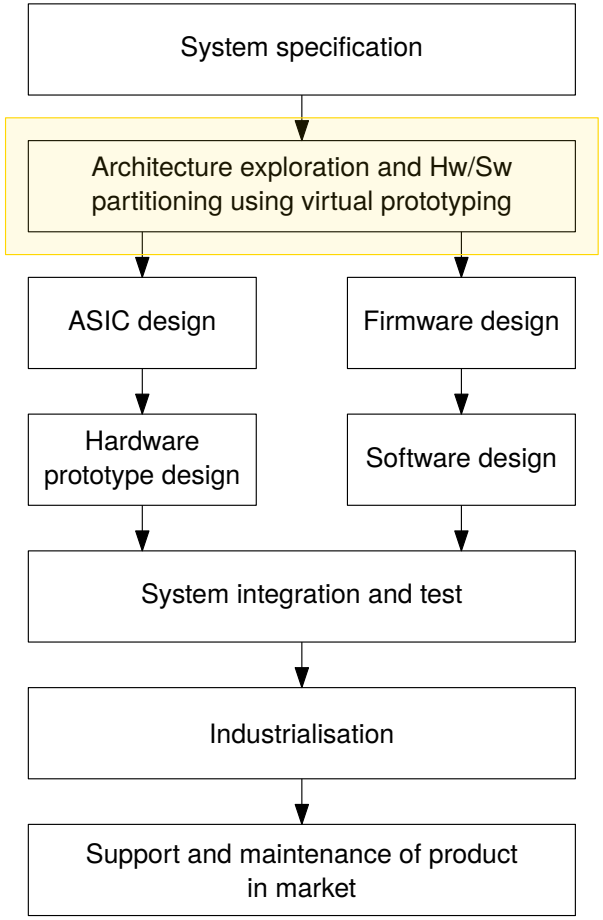
How to design an embedded system

But how to design an embedded system?
The classic method is the so called waterfall model:



How to design an embedded system

With the strict *time-to-market* constraints the classic model does not work, and we need a new method, the *hardware/software co-design* model:



Virtual prototype

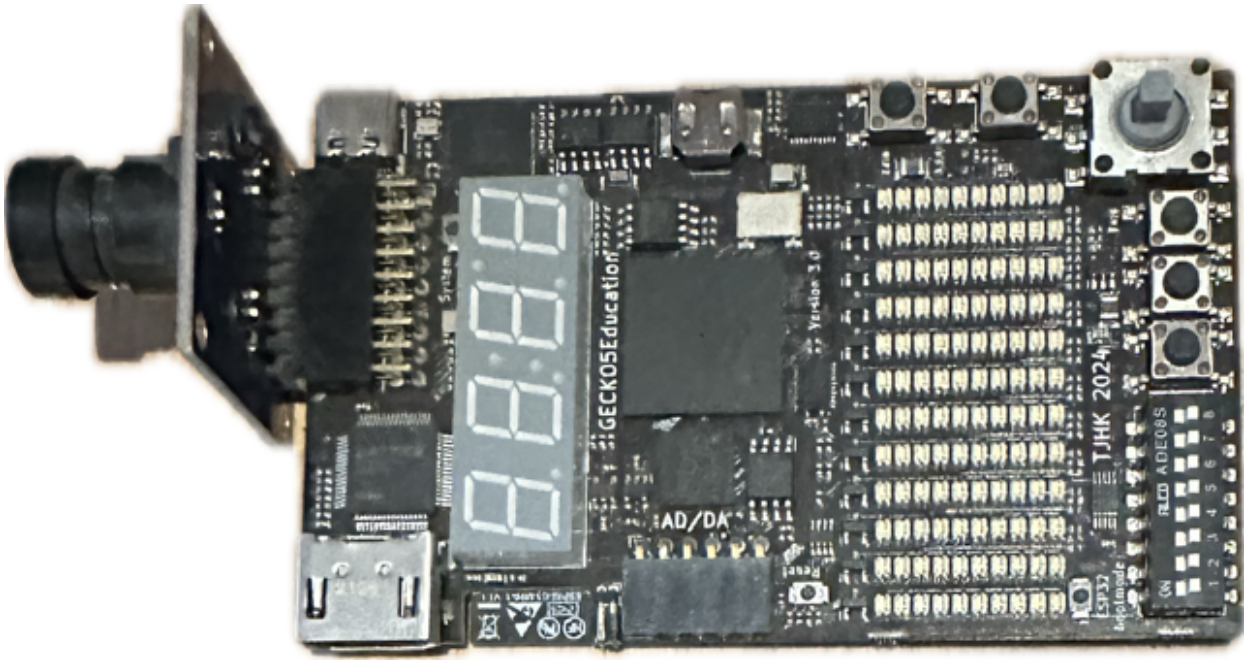
The *hardware/software co-design* model is based on a virtual prototype. A virtual prototype (VP) is a system that emulates the complete system on a functional level (not necessary cycle-true).

Most requirements are met by a “simple” virtual prototype; system requirements dictate which level of accuracy is required (imposing extra time needed to make the virtual prototype and the time gained afterwards).
How to make a Virtual Prototype?

There are many possibilities:

- QEMU
- VHDL/Verilog
- SystemC/SystemVerilog
- C/C++
- MATLAB/Python
- Java
- FPGA
- ...

Virtual prototype

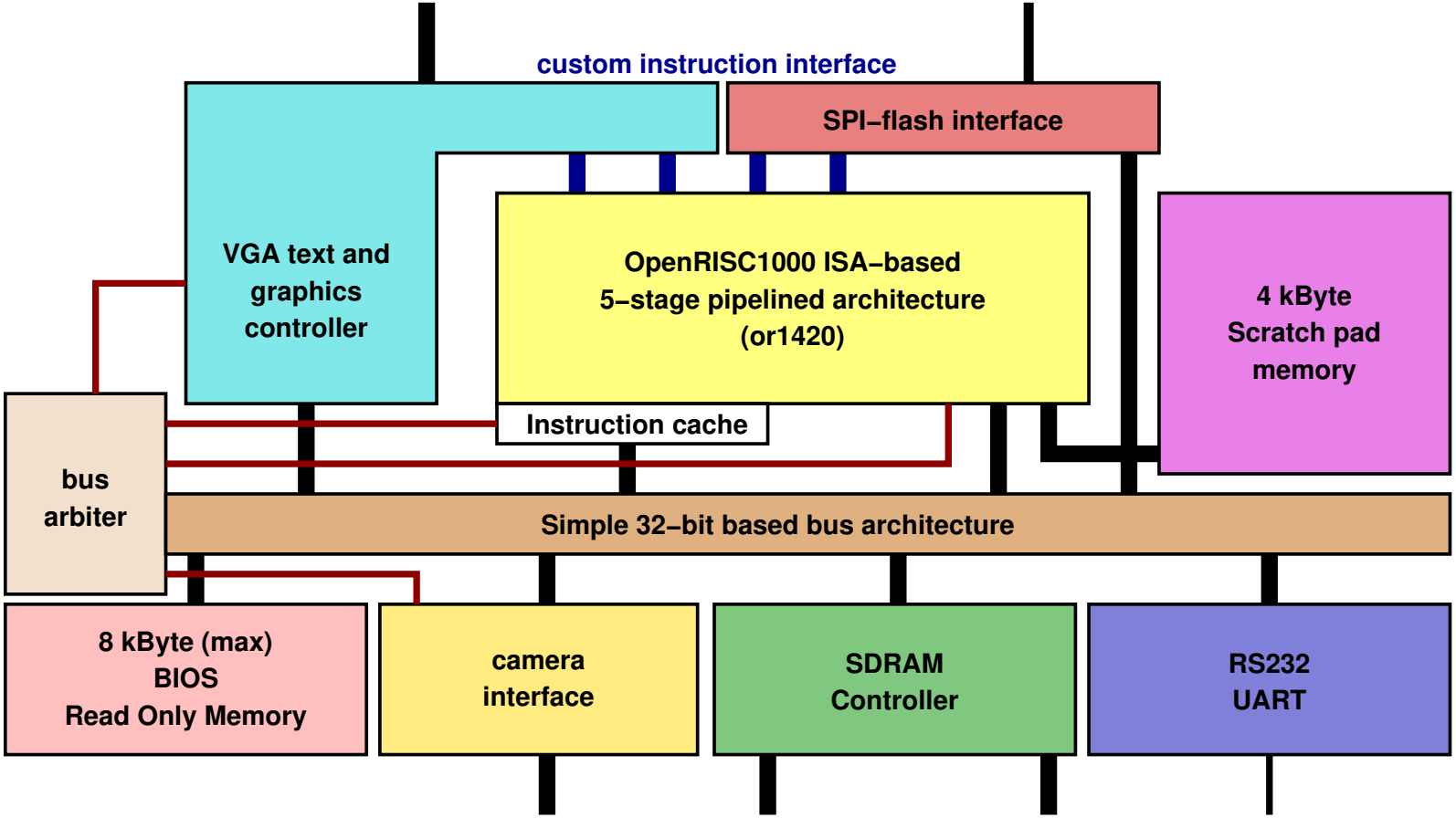


- In this course we are going to work on a cycle-true virtual prototype (VP).
- This VP is completely written in Verilog and is running on the GECKO5Education.
- This VP is completely observable and based on an OpenRISC micro-controller.
- Note: The OpenRISC micro-controller used is a by BFH developed version for industry projects and is not completely compatible with the one found in the internet.

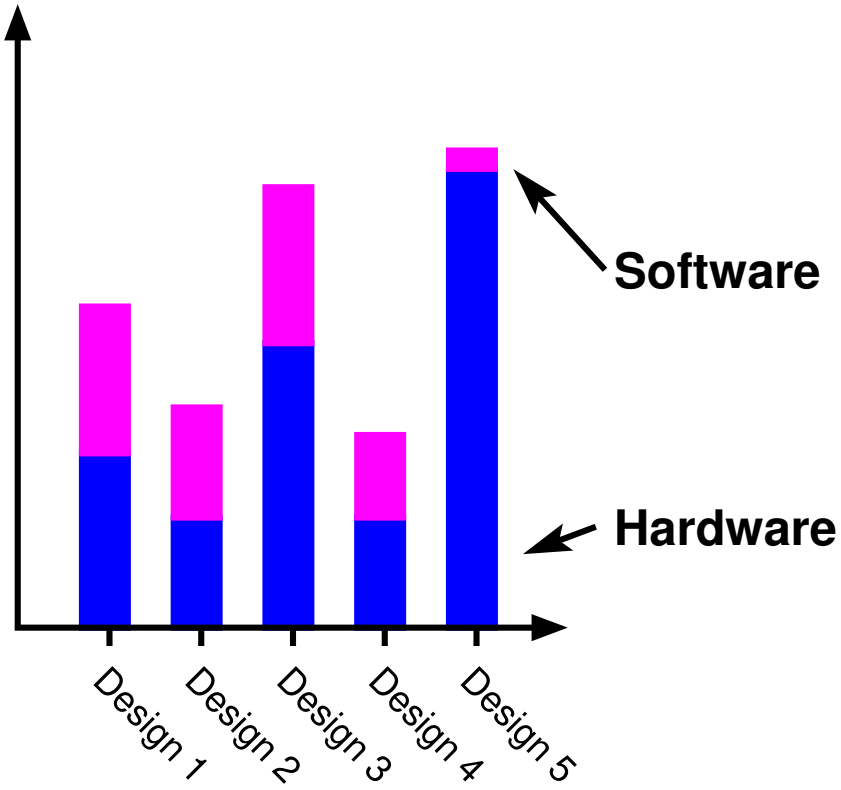
Components of an embedded system

- But what are the components that we find in an embedded system?
- In case of a μC (micro-controller) we find:
  - Processor(s)
  - Memory (memories)
  - Programmable Interface(s)
  - Sometimes external Address, Data, and Control buses.
  - Very often Analog components like analog-digital (AD)-converters and digital-analog (DA)-converters.
- Some examples for μC-based embedded-systems are:
  - The Arduino platform based on an Arm® Cortex®-M0 32-bit SAMD21.
  - The Nucleo family of boards based on the STMicroelectronics STM32-family.
  - The Raspberry Pi platform based on, among others, the RP2040 μC.
  - The HiFive Boards platforms based on the RISC-V μC.
  - ....
- The virtual prototype we are going to use also falls in this class of systems:

Virtual prototype

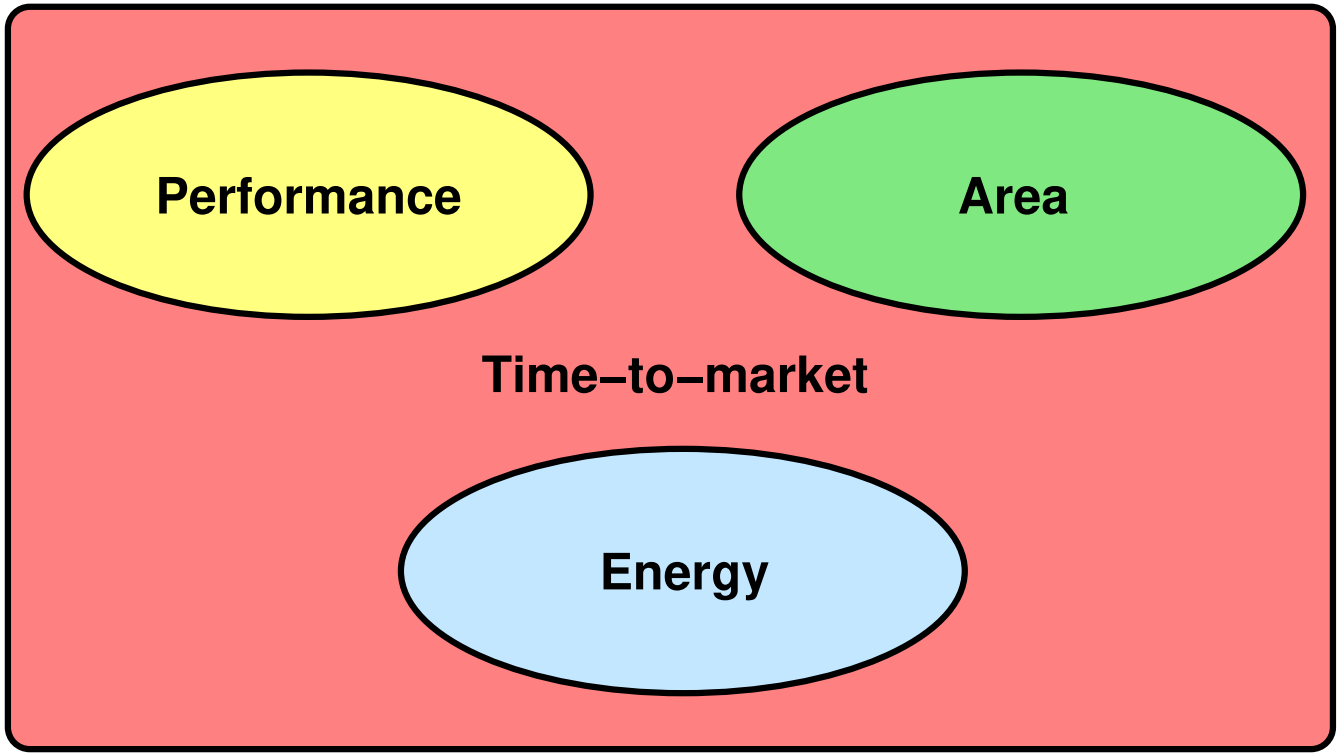


Hardware software partitioning



- In “old-fashioned” designs the hardware part used to be constant whilst the functionality was in software.
- Today, most designs have a varying portion of hardware and software to realize the functionality.
- But how to partition?

Hardware software partitioning

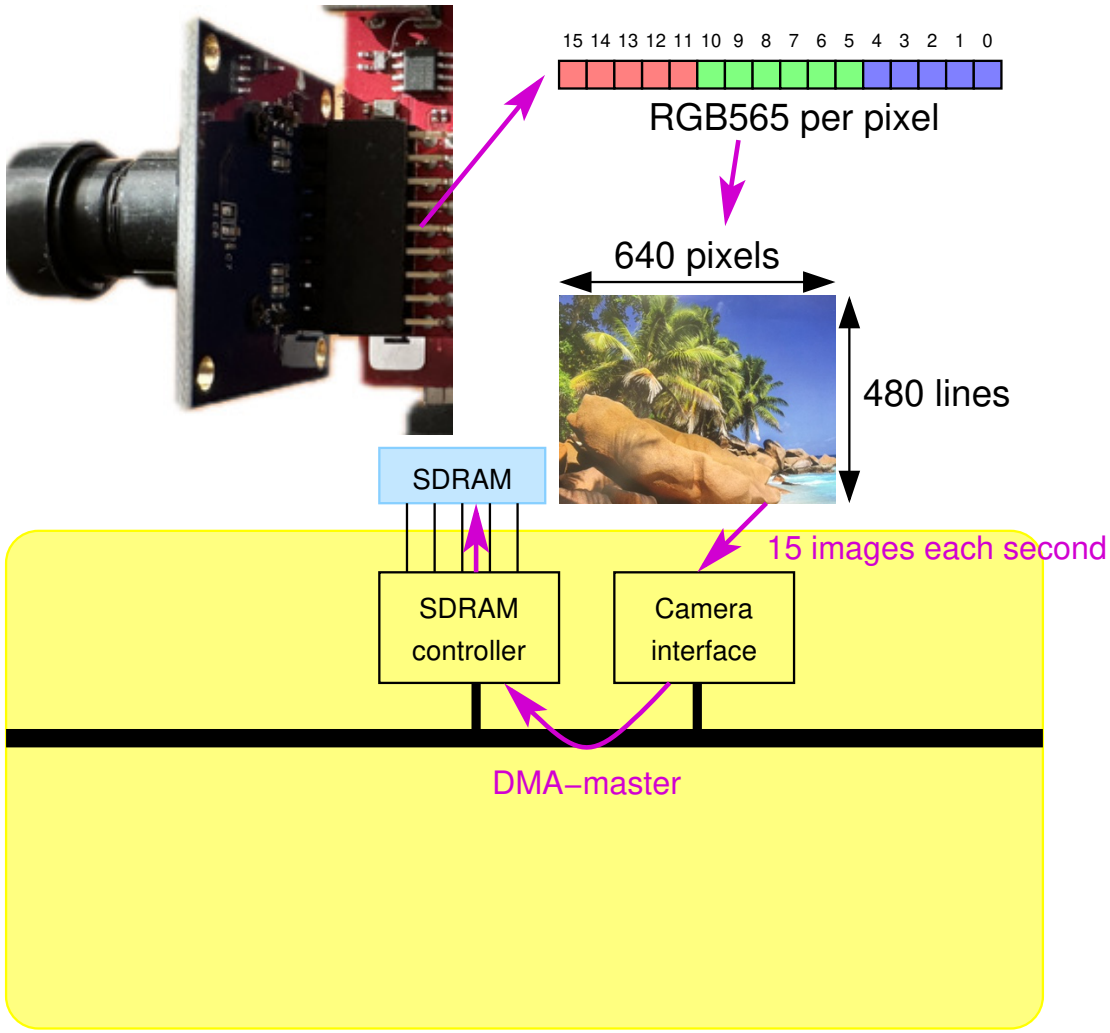


- Which are the aspects that guide our decisions....
- In general it is impossible to meet them all, we have to make trade-offs.
- Worst even, nowadays time-to-market constraints imposes fast design iterations.
- If we do not appear before the competition the effort is in vane....

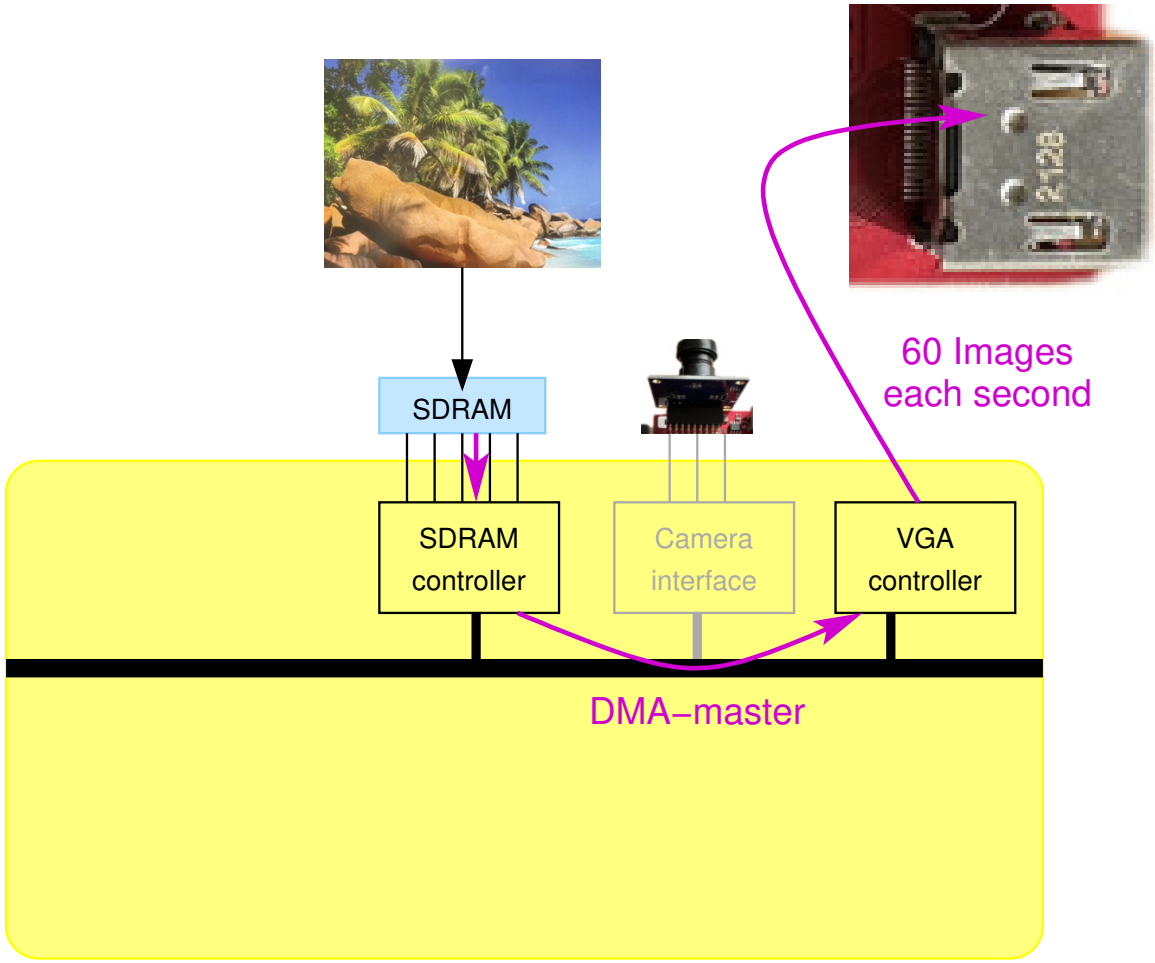
So how do we do it?

Let's take a design example

Camera


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RGB565 per pixel
640 pixels
480 lines
SDRAM
15 images each second
SDRAM controller
Camera interface
DMA−master

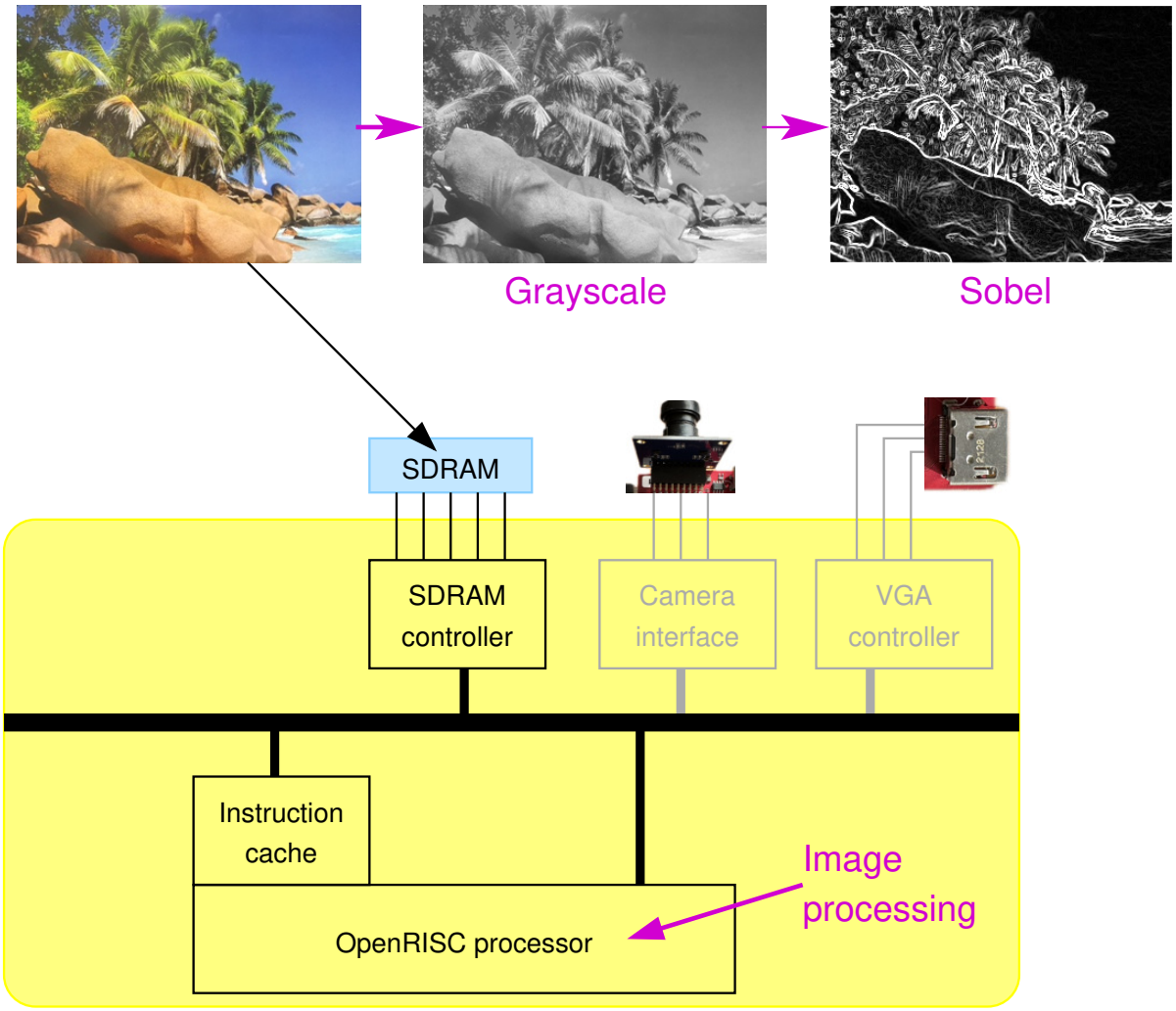
Screen


SDRAM
SDRAM controller
Camera interface
VGA controller
60 Images each second
DMA–master

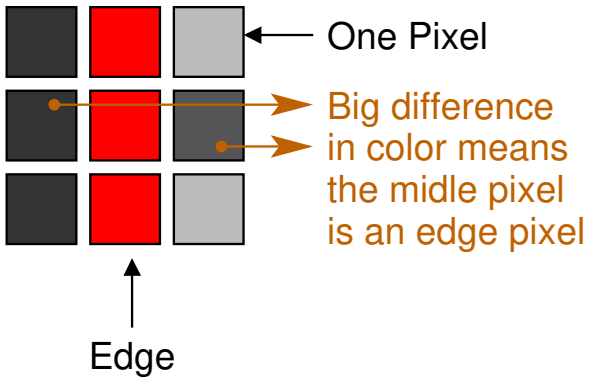
Bare system

DEMO

Image processing

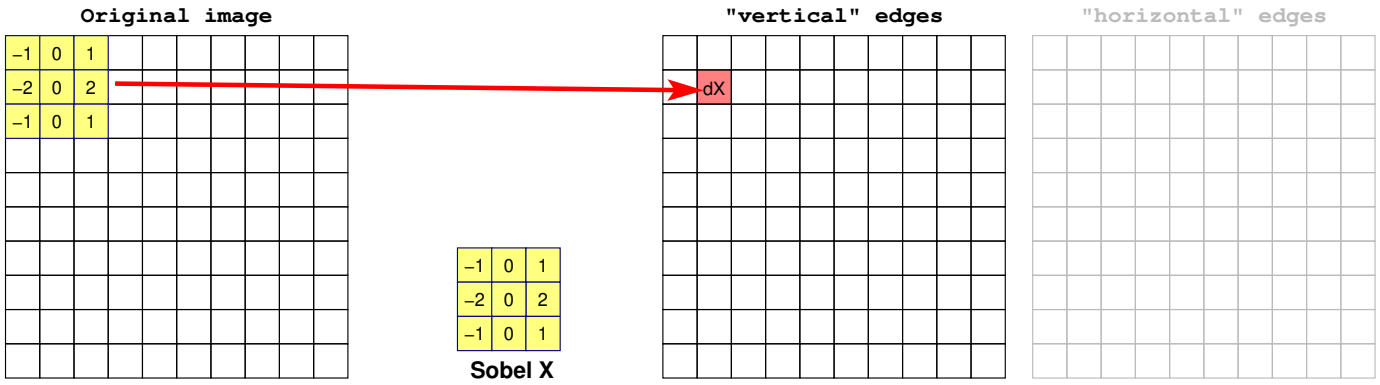


Edge detection (simplified and intuitive)



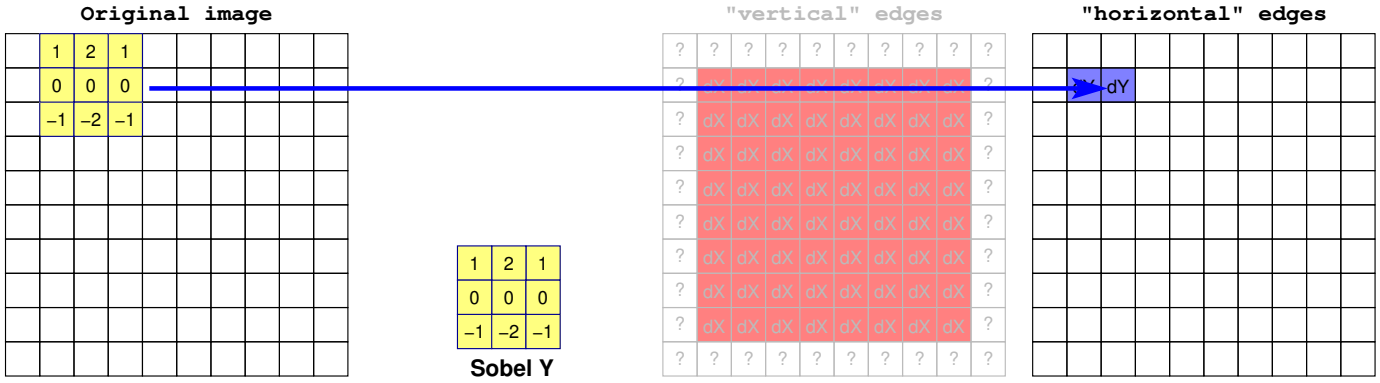
- Let's look at a 3x3 pixel array.
- We can directly see the vertical edge by looking at the colors.
- We can also determine the edge by determining the "difference in color" on both sides (left and right) from the edge-pixel.
- To detect horizontal edges we can do the same thing by taking the "difference in color" on both sides (top and bottom) from the edge-pixel.

Edge detection by Sobel



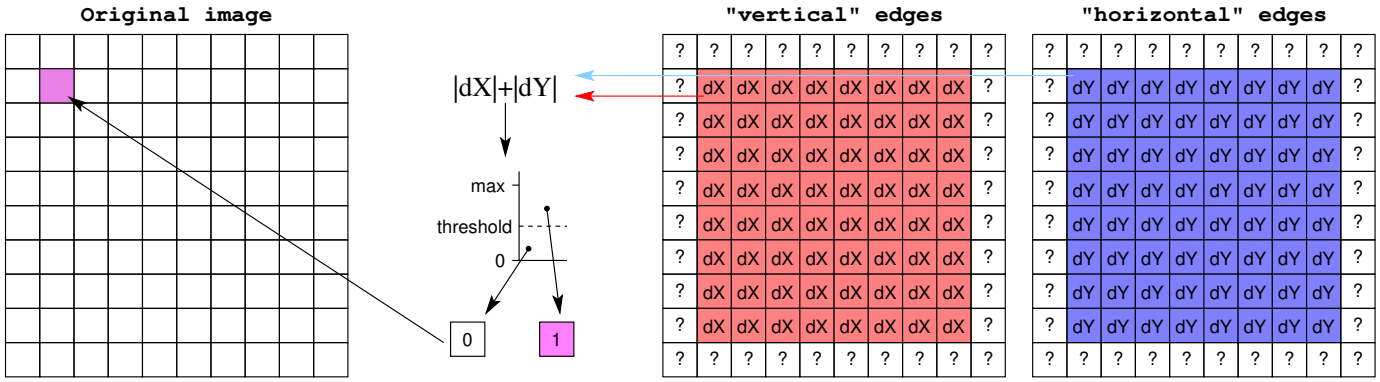
- The algorithm of Sobel does something similar.
- It starts with the “vertical edges" by using a 3x3 X-filter.
- Each “edge value” is calculated for each pixel.
- Except for the outer pixels!

Edge detection by Sobel



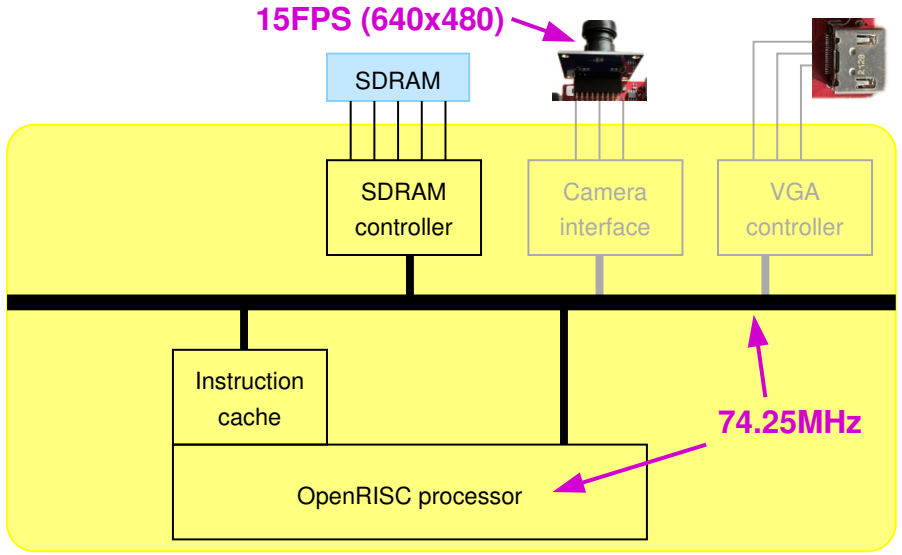
- Next the “horizontal edges" are determined.
- For the “horizontal edges" the 3x3 Y-filter is used.
- Each “edge value” is calculated for each pixel.
- Except for the outer pixels!

Edge detection by Sobel



- Finally the edges are “drawn" by:
- Estimating $\sqrt{dX^2 + dY^2}$ with $|dX| + |dY|$
- Doing thresholding of the above value
- And draw/clear the pixel

Putting some numbers



- Let's put some numbers.
- Hence we have $\frac{74.25 \cdot 10^6}{15} = 4.95 \cdot 10^6 \frac{\text{cpu cycles}}{\text{image}}$ or $\frac{4.95 \cdot 10^6}{640 \cdot 480} \approx 16 \frac{\text{cpu cycles}}{\text{pixel}}$
- Back-of-the-envelope: dX and dY require each 6 multiplications and 6 additions each pixel, the final operation requires 2 `abs()`, 1 addition, 1 subtraction and 1 set; so $\approx 30 \frac{\text{cpu cycles}}{\text{pixel}}$
- Hence we have an impossibility! How to solve this problem, the main topic of this course.

Sobel in reality

DEMO