UiO : University of Oslo

FYS3240- 4240
Data acquisition & control

Embedded systems & Digital electronics

Spring 2019 – Lecture #10

Bekkeng, 30.1.2017

Embedded systems

- An embedded system is a special-purpose system designed to perform one (or a few) dedicated functions

- Some typical characteristics of embedded systems are:
  - Single purpose (with very specific requirements).
  - Not easily adapted.
  - Real-time computing constraints.
  - No operating system or small and simple operating systems.
  - High reliability.
  - Limited computer hardware resource, for instance fixed amount of memory and limited I/O expansion possibilities.
  - Small or non-existent keyboard/mouse or screen.
  - Low power (e.g. 50 mW vs. 50 W or more for a PC).
  - More difficult to program and to interface with compared to a general purpose computer.

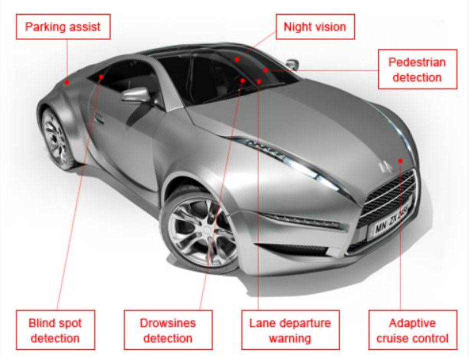
Embedded systems

1968

[DAQ & control in the Apollo program]

Aerospace & Defence

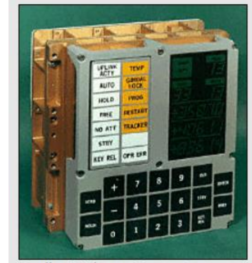
Today – consumer products



ABS	Entertainment system

Embedded systems II

- Once limited to military and space applications, embedded computers now are found in nearly every electronic device.
- One of the first embedded computer systems was the Apollo guidance computer, which was used during the first moon landing in 1969 aboard Apollo 11 and the lunar landing module. The Apollo guidance computer weighed approximately 32 kg, and required 70 watts at 28 volts DC.



Apollo Guidance Computer

- Today we find embedded computers in cars, missiles, spacecrafts, aircrafts, home appliances, medical devices, communication devices, and toys.
- In aircrafts, spacecrafts and missiles **complex mathematical algorithms** is usually implemented. But also in consumer products such as GPS receivers mathematical estimation algorithms such as least squares estimation or Kalman filtering is used. Therefore, embedded systems design often require knowledge of signal processing and mathematical algorithm implementation.

Autonomous Systems

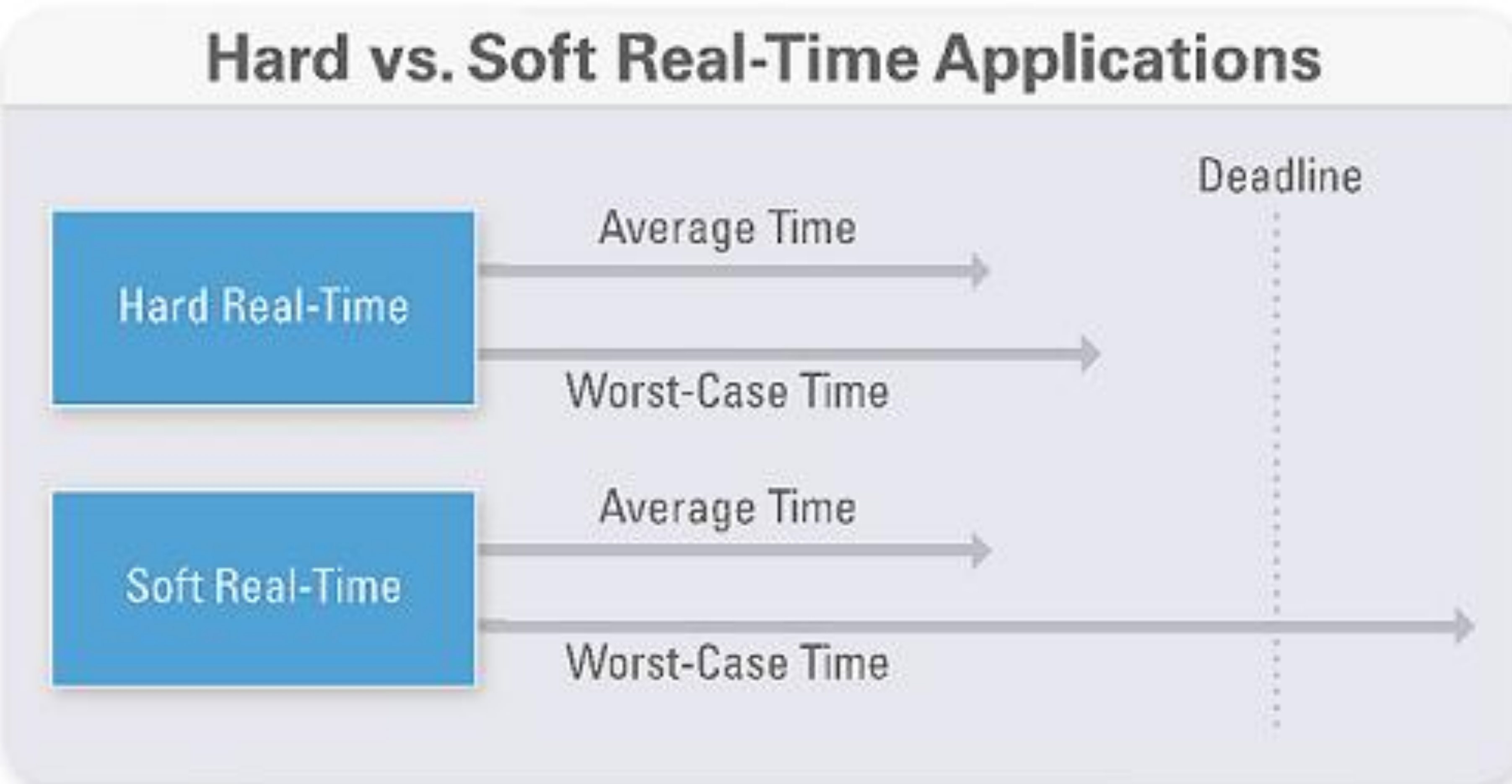
- Embedded systems are everywhere today and will be even more important in the future.
- Most embedded systems perform “simple functions” that are pre-programmed.
- Increasingly, embedded systems are designed to carry out autonomous tasks (smart systems).
- Embedded systems/vehicles will be designed to be able to make <u>decisions</u> based on complex inputs and situation awareness
- (Keywords: Artificial Intelligence (AI), machine learning, deep learning, big data).

What is a real-time (RT) system

- A real-time system gives you **determinism**
  - The correctness of the system depends not only on the logical result but also on the time it was delivered

- Hard real-time
  - systems where it is absolutely imperative that responses occur within the required deadline (Example: Flight control systems)
- Soft real-time
  - allows for some deadlines to be missed with only a slight degradation in performance but not a complete failure (example: DAQ-systems)

- In contrast, on an ordinary desktop PC (with Windows) the OS operates on a fairness basis
  - Each application gets time on the CPU regardless of its priority
  - Even our most time-critical application can be suspended for some routine maintenance


Hard vs. Soft Real-Time Applications
Deadline
Hard Real-Time
Average Time
Worst-Case Time
Soft Real-Time
Average Time
Worst-Case Time

Embedded processors

- Microprocessor
- Microcontroller
- DSP (Digital signal processor)
  - A specialized microprocessor with an optimized architecture for mathematical operations to be performed quickly (e.g. FFT)
- FPGA (Field Programmable Gate Array)
- GPU (Graphics Processing Unit)

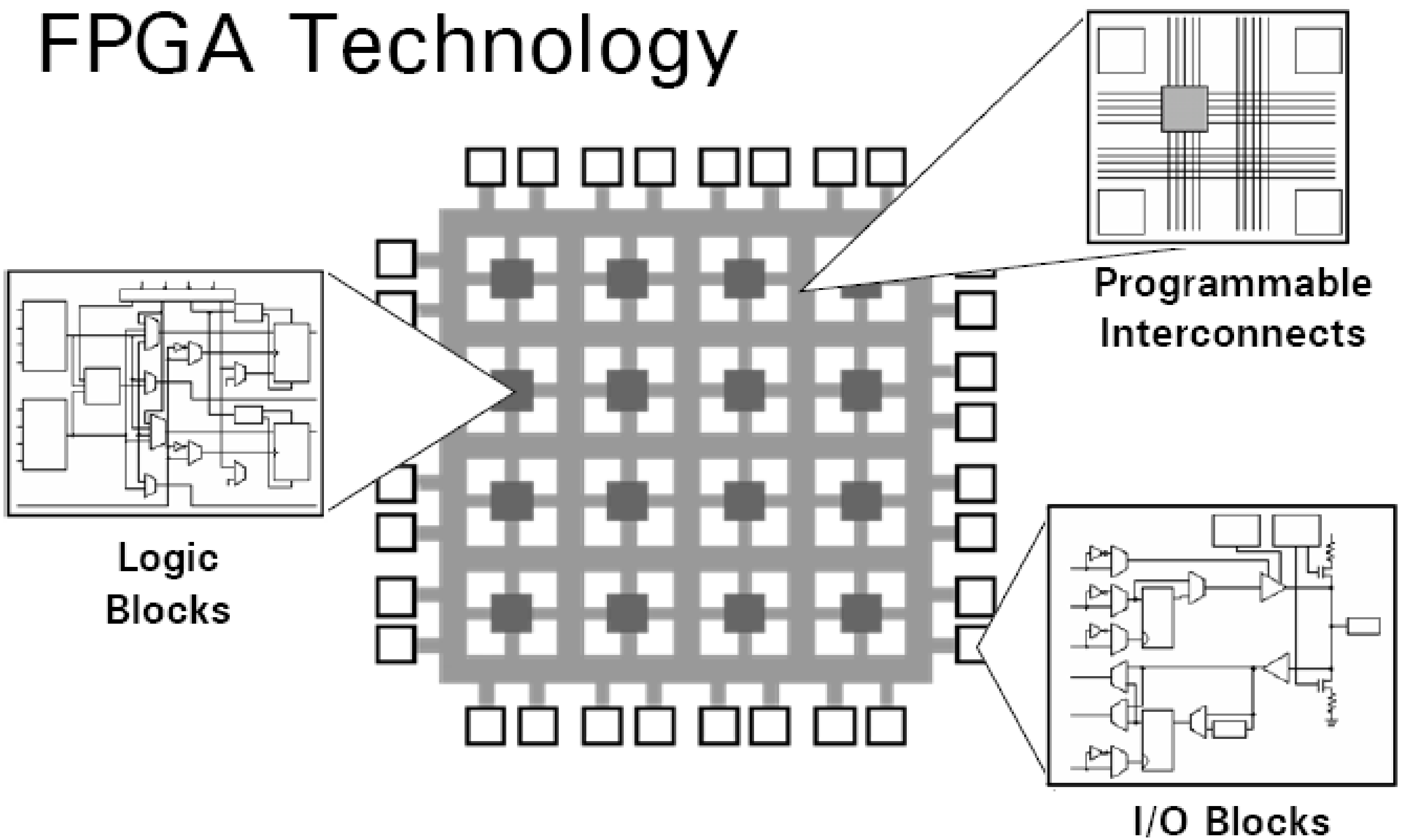
Embedded microprocessors

- Modern x86 CPUs are relatively uncommon in embedded systems and small low power applications, as well as low-cost microprocessor markets (e.g. home appliances and toys).

- Simple 8-bit and 16-bit based architectures are common, although the x86-compatible **AMD's Athlon** and **Intel Atom** are examples of 64-bit designs used in some *relatively* low power and low cost segments

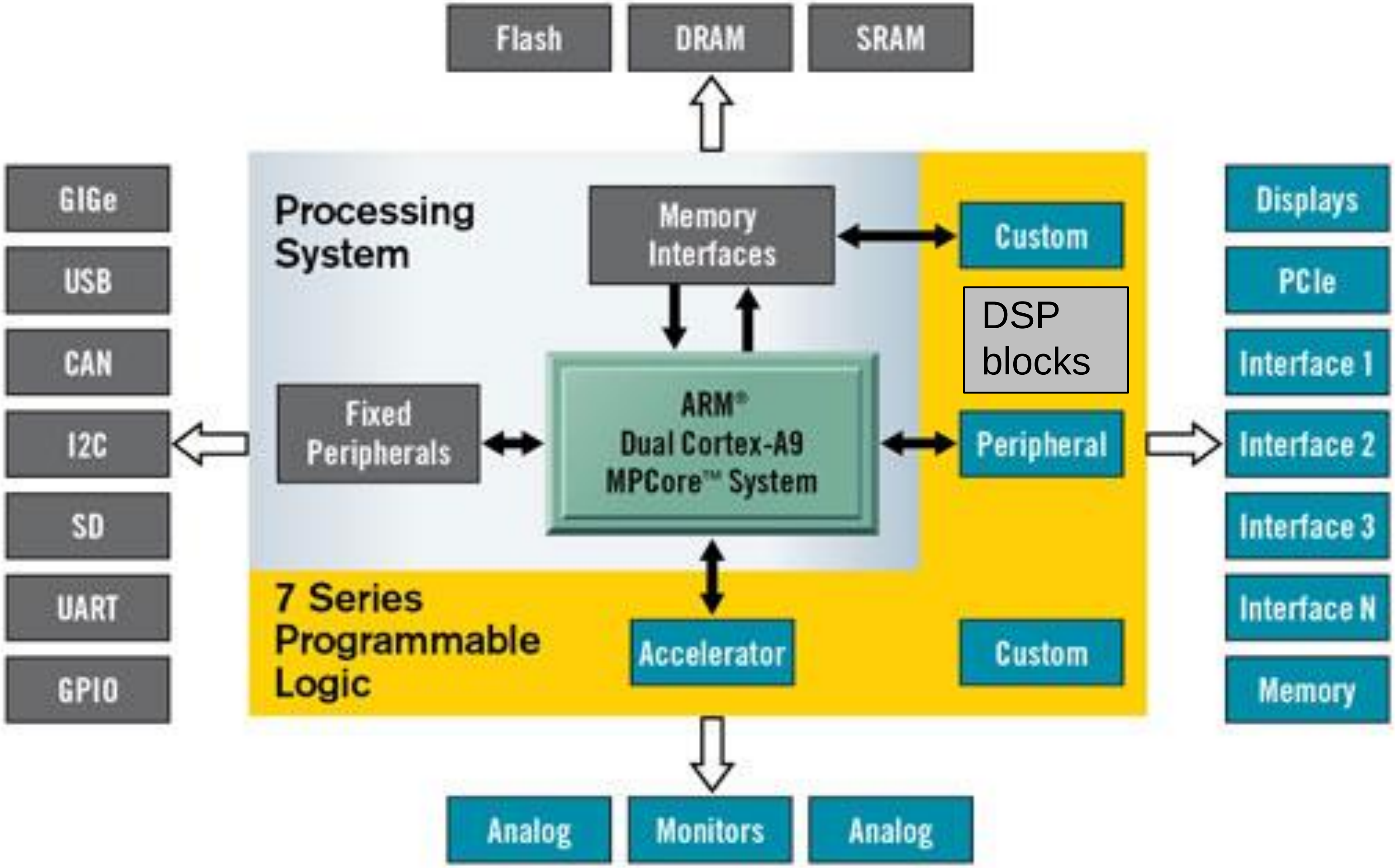
Microcontrollers

- The program instructions written for **microcontrollers** are referred to as **firmware**, and are stored in read-only memory (ROM) or Flash memory chips.
- In contrast, a general-purpose computer loads its programs into random access memory (RAM) each time.



FPGA = Field Programmable Gate Array

Xilinx 7 series FPGA example



SOC : System On a Chip

FPGA advantages

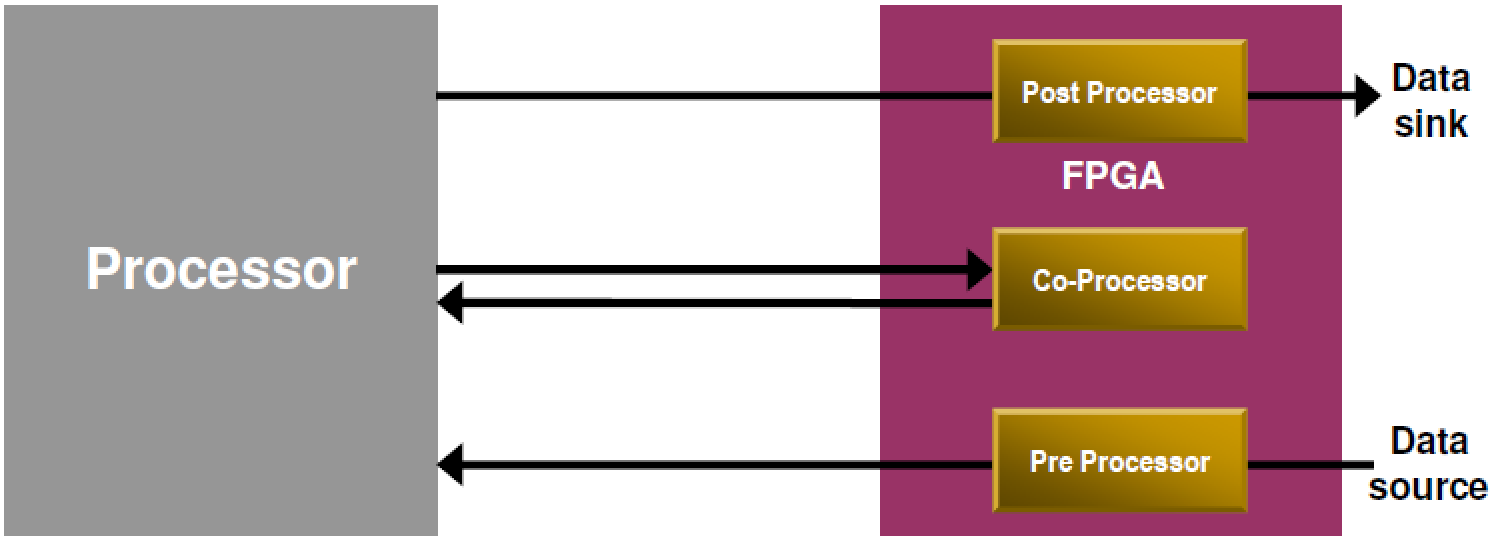
- High reliability
- High determinism
- High performance
- True parallelism
- Reconfigurable

FPGAs **give low-latency** processing, but they have **limitations in terms of floating-point computations**

The highest performance FPGAs (2012) have **600 MHz** clock speed

Processor / FPGA Co-Processor Features

Accelerating Your Success™

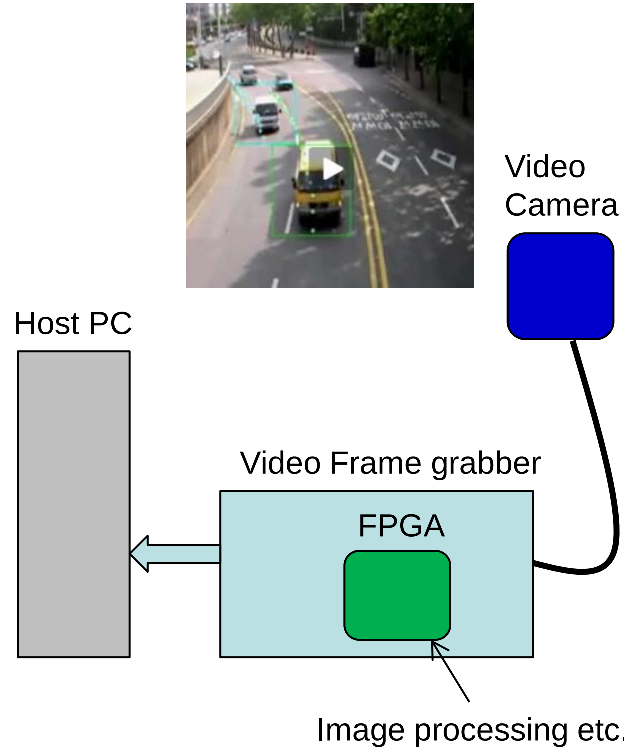


- Operating System
- Legacy code base
- Floating point math
- High-level abstraction (ex. C / C++)
- Complex decision making
- Non time-critical
- Interrupt-driven
- Fixed peripheral set

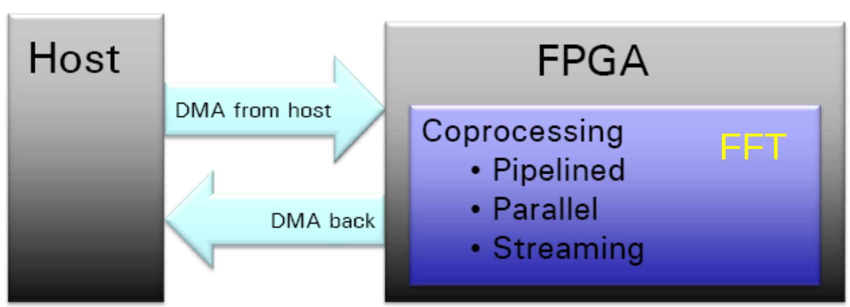
- Parallel execution
- High computation rates in fixed-point math
- Custom peripherals

Common Applications for FPGAs in DAQ and control systems

- High-speed control
- Hardware programmable DAQ-cards
- Onboard processing and data reduction
  - e.g. video processing



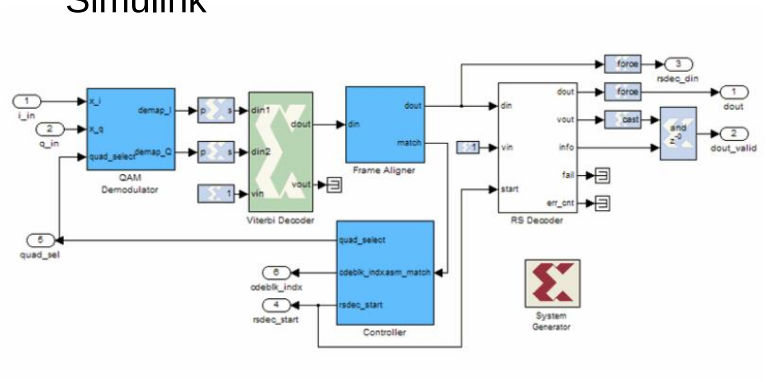
- Co-processing
  - offload the CPU



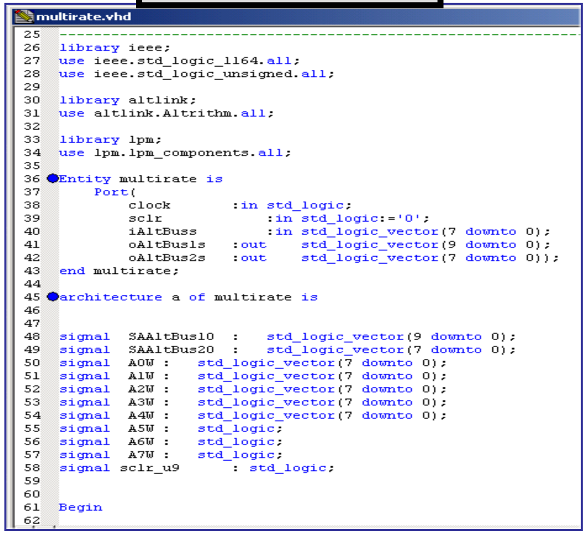
How to program an FPGA ?

- **VHDL** (Hardware Description Language)
- **SystemVerilog**
- **C-code** (need a development tool)
- Automatic Generation of VHDL code (or a bit stream) from a high level development tool, such as
  - **MATLAB** (HDL Coder)
  - **Simulink** (DSP Builder / System Generator for DSP)
  - **LabVIEW** (FPGA Module)

Simulink



VHDL Code


multirate.vhd

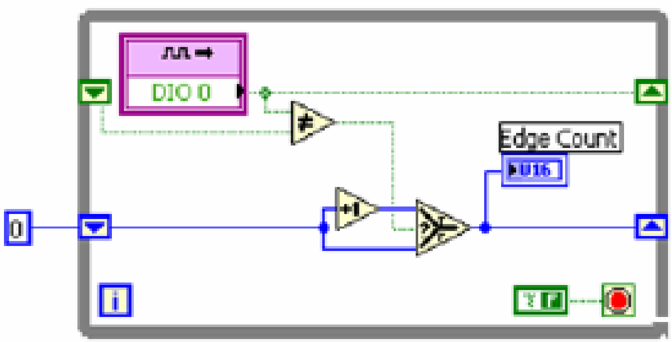
```
-----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

library altlink;
use altlink.Altrithm.all;

library lpm;
use lpm.lpm_components.all;

Entity multirate is
	Port(
		clock		:in std_logic;
		sclr			:in std_logic:='0';
		iAltBuss		:in std_logic_vector(7 downto 0);
		oAltBus1s	:out	std_logic_vector(9 downto 0);
		oAltBus2s	:out	std_logic_vector(7 downto 0));
end multirate;

architecture a of multirate is


signal	SAAltBus10	:	std_logic_vector(9 downto 0);
signal	SAAltBus20	:	std_logic_vector(7 downto 0);
signal	A0W :	std_logic_vector(7 downto 0);
signal	A1W :	std_logic_vector(7 downto 0);
signal	A2W :	std_logic_vector(7 downto 0);
signal	A3W :	std_logic_vector(7 downto 0);
signal	A4W :	std_logic_vector(7 downto 0);
signal	A5W :	std_logic;
signal	A6W :	std_logic;
signal	A7W :	std_logic;
signal sclr_u9		: std_logic;


Begin

```

Extra

From LabVIEW to Hardware

Translation → **Optimization** → **Synthesis** → **Bit Stream**

- Translation: VHDL Generation
- Optimization: Analysis, Logic Reduction
- Synthesis: Place and Route, Timing Verification
- Bit Stream: Generation, Download/Run

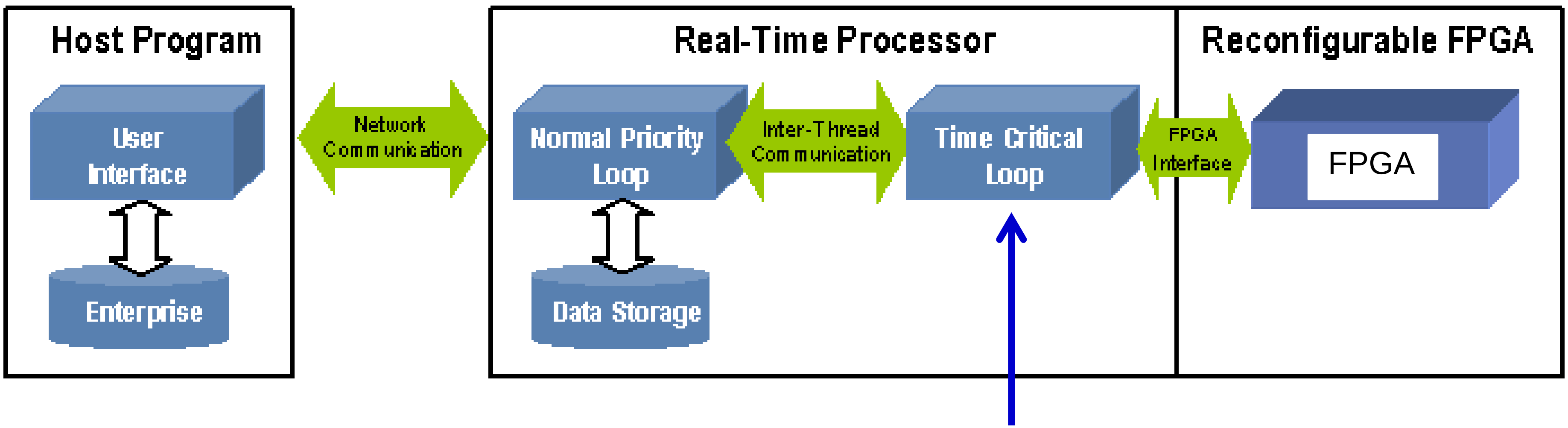


001010010101000101
100101001001010010
001001010101011101
001011010101000101
101001010010100100
010010001001001000

Can also use the LabVIEW IP Integration Node to include **VHDL code**

Architecture for Advanced Embedded DAQ-applications

PC – Windows OS

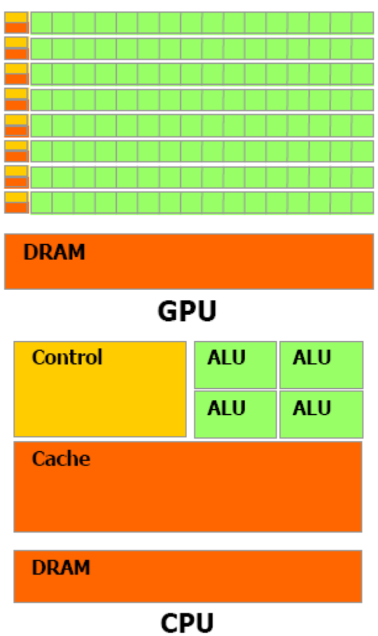


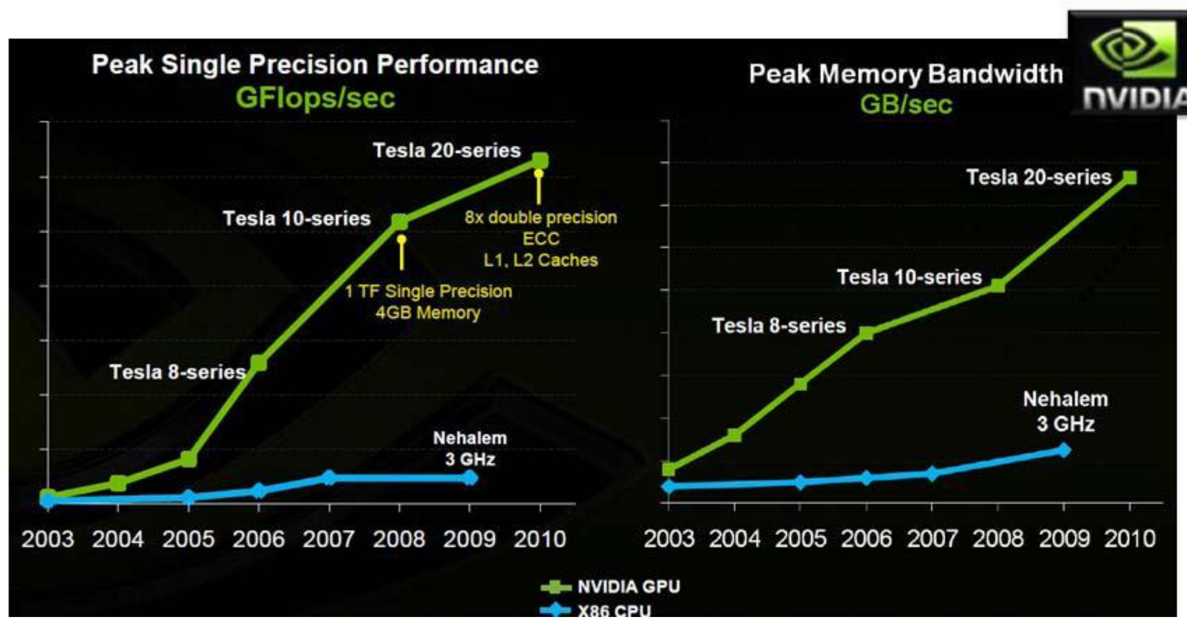
Avoid:

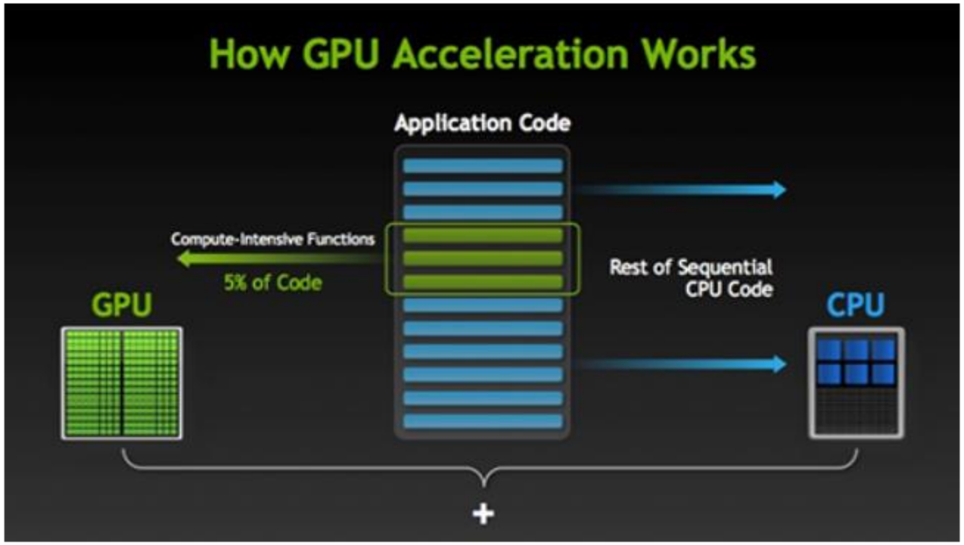
- **File I/O**
- **Networking functions**
- **Memory re-allocation**

GPUs

- GPU = Graphics Processing Unit
- Can be used as **hardware accelerator**
- Can be used in **Real-Time High-Performance Computing** systems
- GPUs have more transistors dedicated for processing than a CPU
  - The performance gain when using GPUs can be significant
- CUDA (Compute Unified Device Architecture) is developed by Nvidia and is a GPU interface for **C**



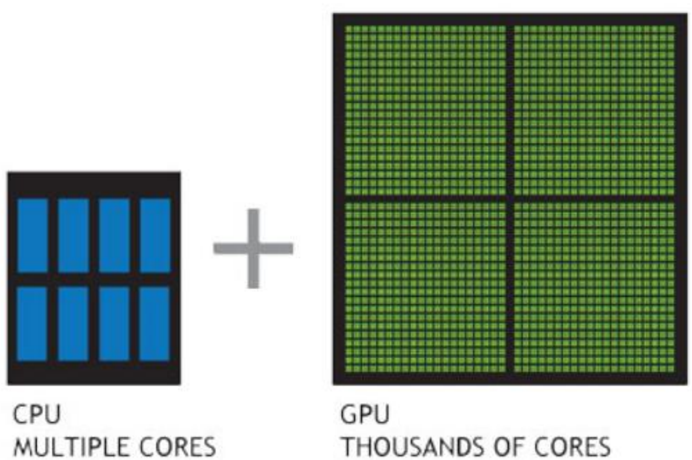




CPU VERSUS GPU

A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously.

GPUs have thousands of cores to process parallel workloads efficiently



From NVIDIA

NVIDIA Tesla GPUs

SELECT THE RIGHT TESLA GPU

Features	Tesla K20X	Tesla K20	Tesla K10	Tesla M2090	Tesla M2075
Number and Type of GPU	1 Kepler GK110		2 Kepler GK104s	1 Fermi GPU	1 Fermi GPU
GPU Computing Applications	Seismic processing, CFD, CAE, Financial computing, Computational chemistry and Physics, Data analytics, Satellite imaging, Weather modeling		Seismic processing, signal and image processing, video analytics	Seismic processing, CFD, CAE, Financial computing, Computational chemistry and Physics, Data analytics, Satellite imaging, Weather modeling	
Peak double precision floating point performance	1.31 Tflops	1.17 Tflops	190 Gigaflops (95 Gflops per GPU)	665 Gigaflops	515 Gigaflops
Peak single precision floating point performance	3.95 Tflops	3.52 Tflops	4577 Gigaflops (2288 Gflops per GPU)	1331 Gigaflops	1030 Gigaflops
Memory bandwidth (ECC off)	250 GB/sec	208 GB/sec	320 GB/sec (160 GB/sec per GPU)	177 GB/sec	150 GB/sec
Memory size (GDDR5)	6 GB	5 GB	8GB (4 GB per GPU)	6 GigaBytes	6 GigaBytes
CUDA cores	2688	2496	3072 (1536 per GPU)	512	448

Build hardware vs. buying COTS

- Buy COTS (Commercial-off-the-shelf) hardware when possible.
  - For instance laboratory data acquisition hardware
- Examples of when a custom build in necessary:
  - High volumes (10,000+)
  - An iteration on an existing custom design
  - Custom size or shape required
  - Very stringent technical requirements
    - Ultralow power consumption
    - Very small form factor
    - MIL- specification
    - Radiation hardening (for space applications)