FYS3240
PC-based instrumentation and microcontrollers

Digital electronics & Embedded systems

Spring 2016 – Lecture #10
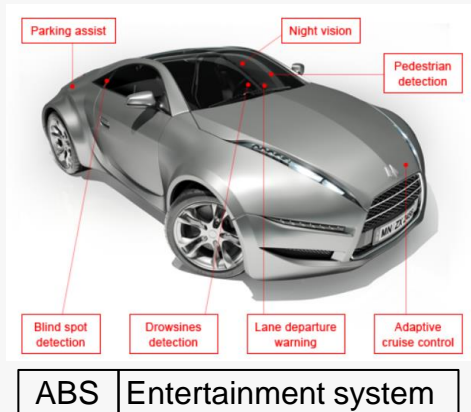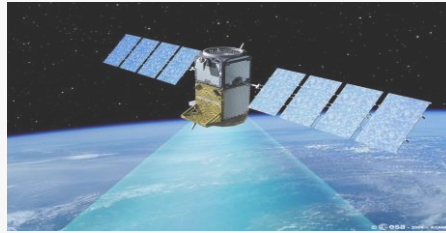
Bekkeng, 18.1.2016

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.

UiO **: University of Oslo**

Embedded systems

Today – consumer products

Aerospace & Defence

1968

Apollo Guidance Computer

DAQ & control in the Apollo program

Parking assist

Night vision

Pedestrian detection

Blind spot detection

Drowsines detection

Lane departure warning

Adaptive cruise control

| 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.
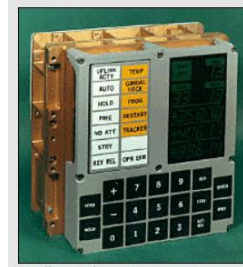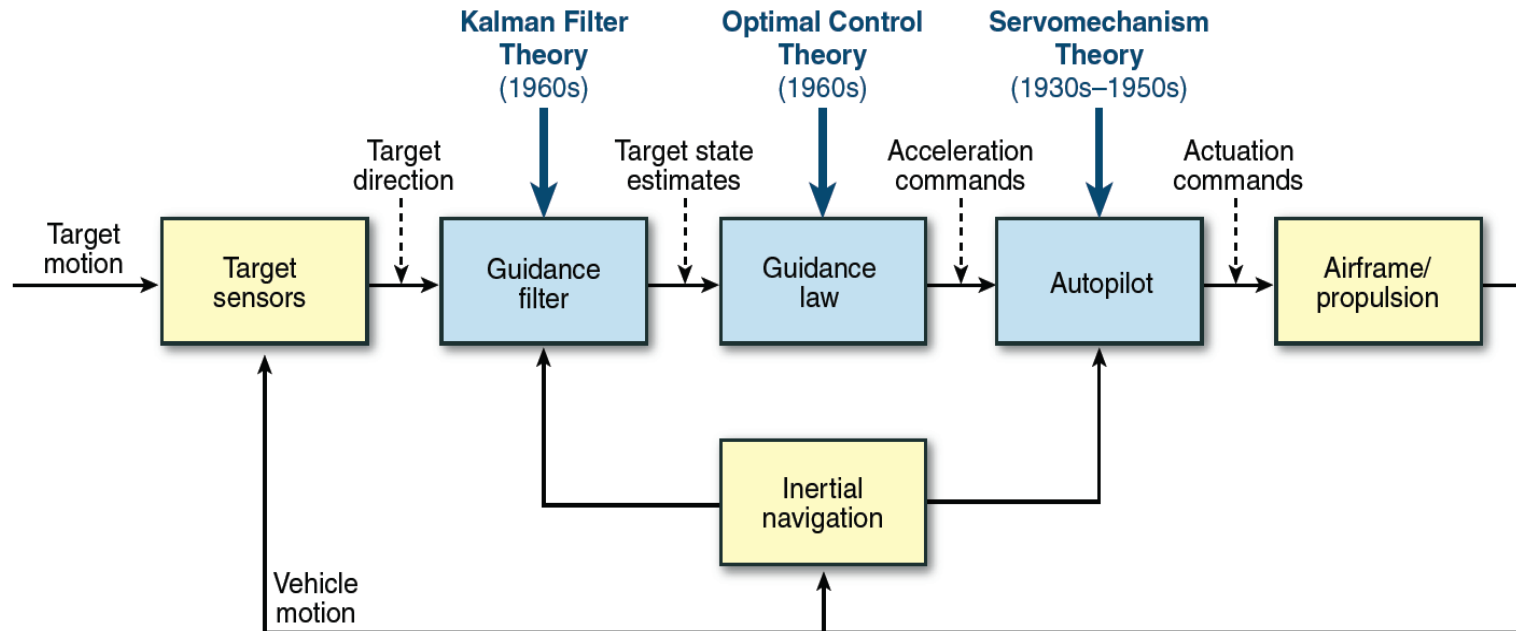
Guidance example

Figure 1. The traditional guidance, navigation, and control topology for a guided missile comprises guidance filter, guidance law, autopilot, and inertial navigation components. Each component may be synthesized by using a variety of techniques, the most popular of which are indicated here in blue text.

Figure from Palumbo, *Johns Hopkins APL Technical Digest*

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 decisions based on complex inputs and situation awareness.

UiO **:** **University of Oslo**

What is a real-time (RT) system

- A real-time system gives you <u>determinism</u>
  - 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)
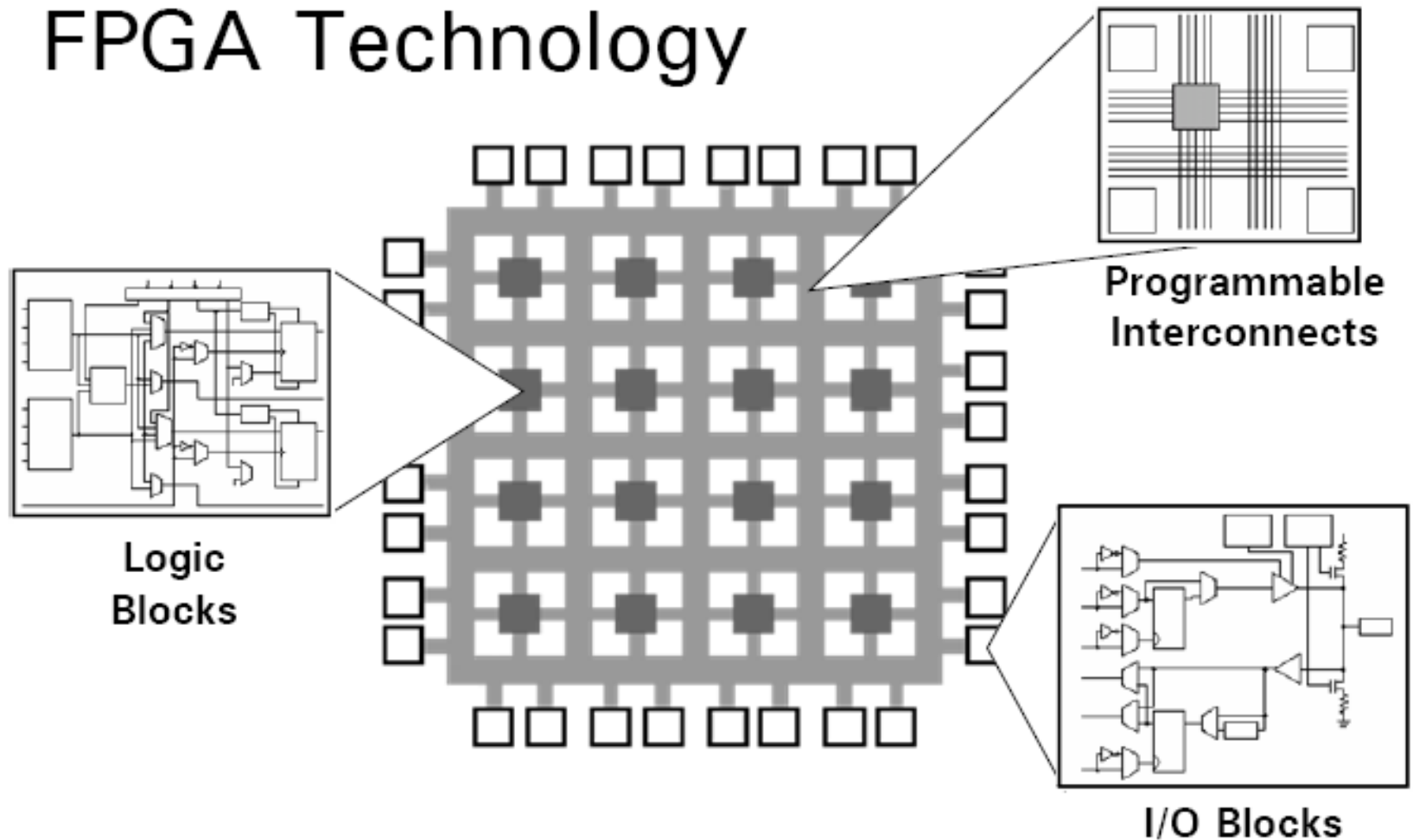- GPU (Graphics Processing Unit)

UiO **:** University of Oslo

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 Technology

Programmable
Interconnects

Logic
Blocks

I/O Blocks

FPGA = Field Programmable Gate Array

UiO : University of Oslo

Xilinx 7 series FPGA example
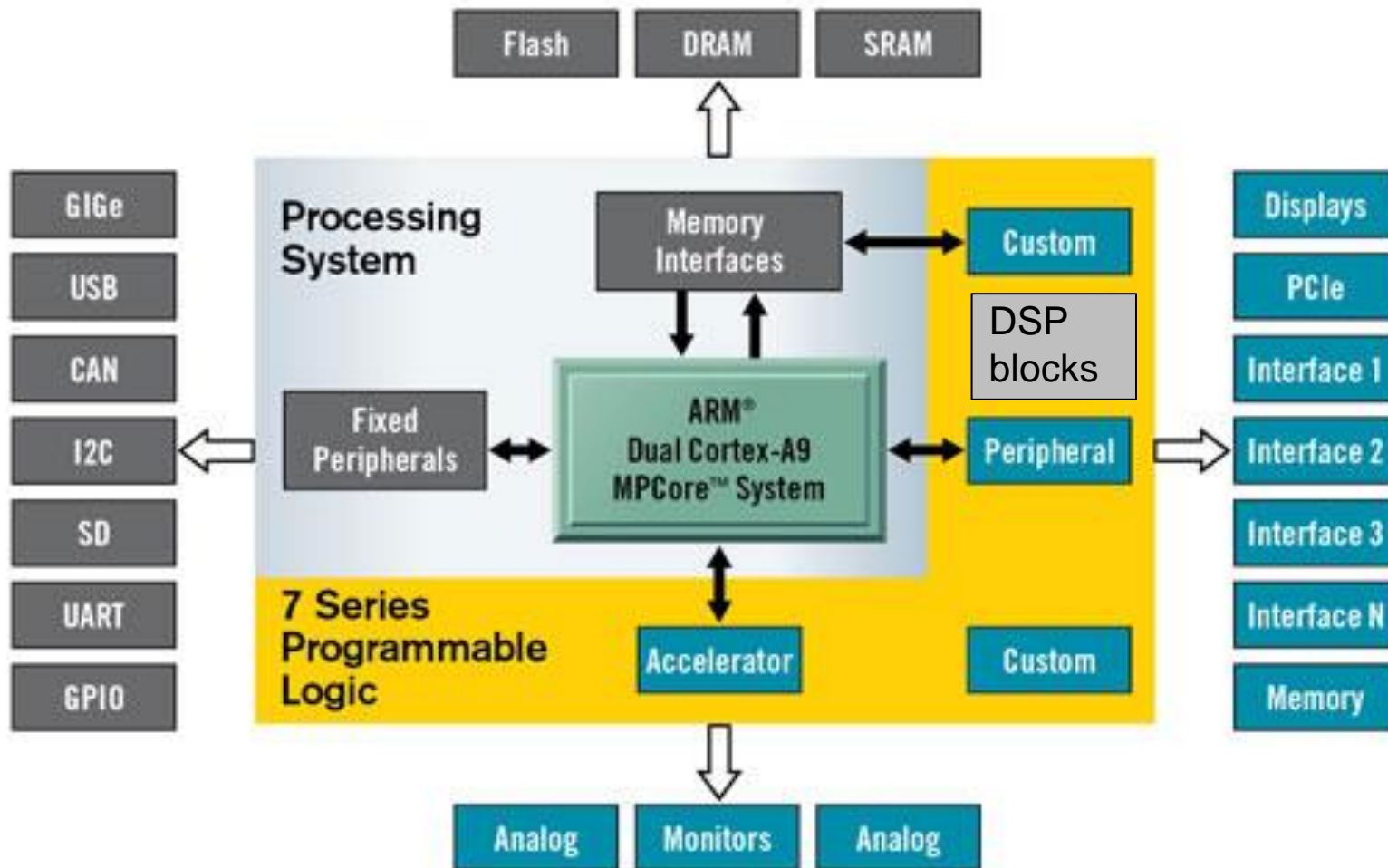
SOC : System On a Chip

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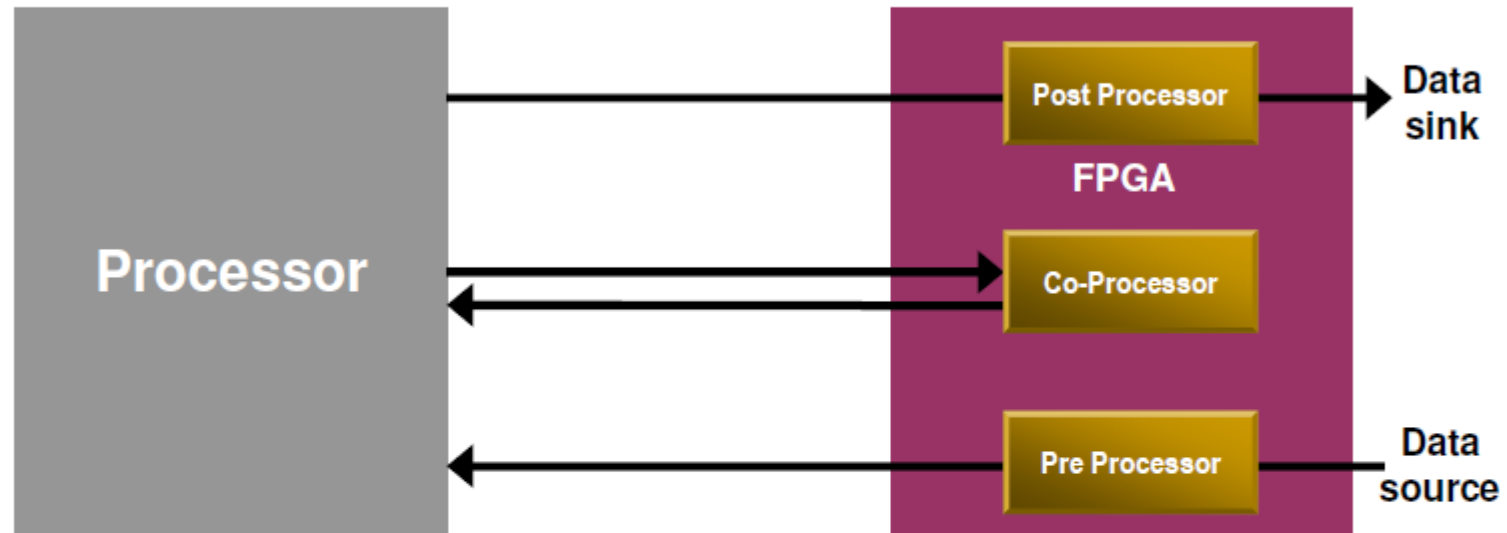
The highest performance FPGAs (2012) have **600 MHz** clock speed

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**
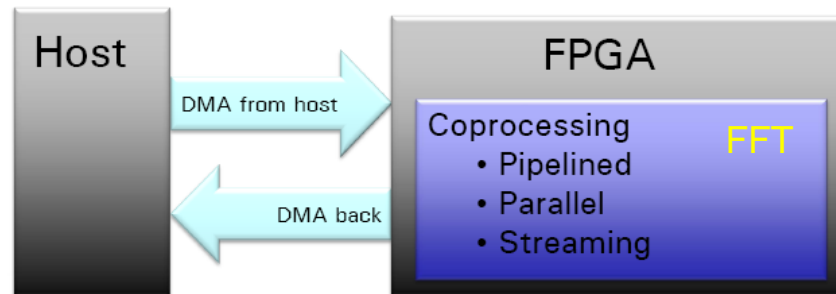- **Fixed peripheral set**

- **Parallel execution**
- **High computation rates in fixed-point math**
- **Repetitive calculations**
- **Nested inner loops**
- **Fast access to deeply pipelined time-skew buffers**
- **Wide data words**
- **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

Video Camera

Host PC

Video Frame grabber

FPGA

Image processing etc.
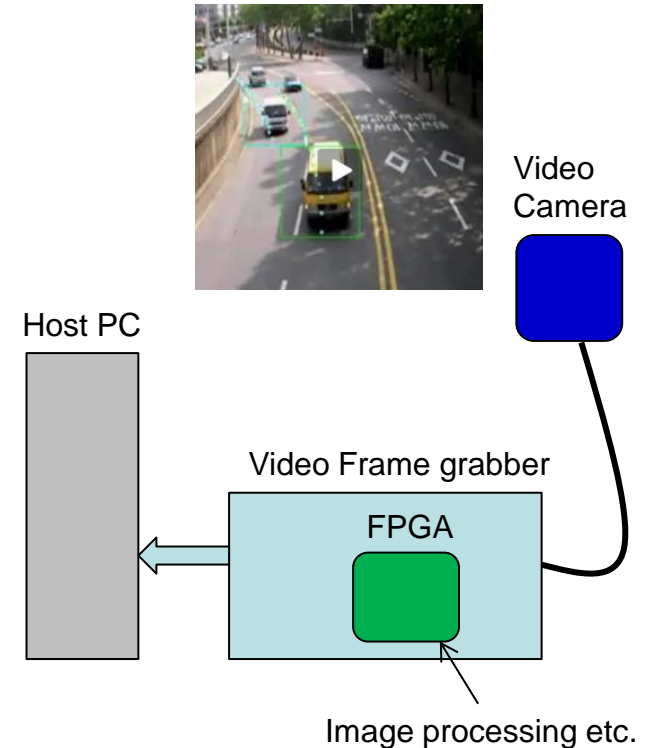
- Co-processing
  - offload the CPU

Host

DMA from host

FPGA

Coprocessing
- Pipelined
- Parallel
- Streaming

FFT

DMA back

How to program an FPGA ?

- **VHDL** (Hardware Description Language)
- **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
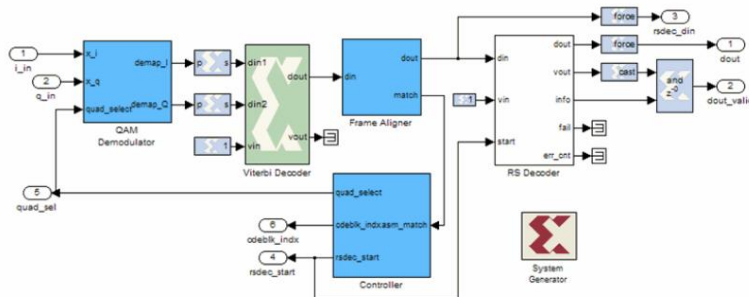
VHDL Code

```
multirate.vhd
25  --------------------------------------------------
26  library ieee;
27  use ieee.std_logic_1164.all;
28  use ieee.std_logic_unsigned.all;
29
30  library altlink;
31  use altlink.Altrithm.all;
32
33  library lpm;
34  use lpm.lpm_components.all;
35
36  Entity multirate is
37      Port(
38          clock       :in std_logic;
39          sclr        :in std_logic:='0';
40          iAltBuss        :in std_logic_vector(7 downto 0);
41          oAltBus1s   :out    std_logic_vector(9 downto 0);
42          oAltBus2s   :out    std_logic_vector(7 downto 0));
43  end multirate;
44
45  architecture a of multirate is
46
47
48  signal   SAAltBus10 :    std_logic_vector(9 downto 0);
49  signal   SAAltBus20 :    std_logic_vector(7 downto 0);
50  signal   A0W :   std_logic_vector(7 downto 0);
51  signal   A1W :   std_logic_vector(7 downto 0);
52  signal   A2W :   std_logic_vector(7 downto 0);
53  signal   A3W :   std_logic_vector(7 downto 0);
54  signal   A4W :   std_logic_vector(7 downto 0);
55  signal   A5W :   std_logic;
56  signal   A6W :   std_logic;
57  signal   A7W :   std_logic;
58  signal sclr_u9      : std_logic;
59
60
61  Begin
62
```
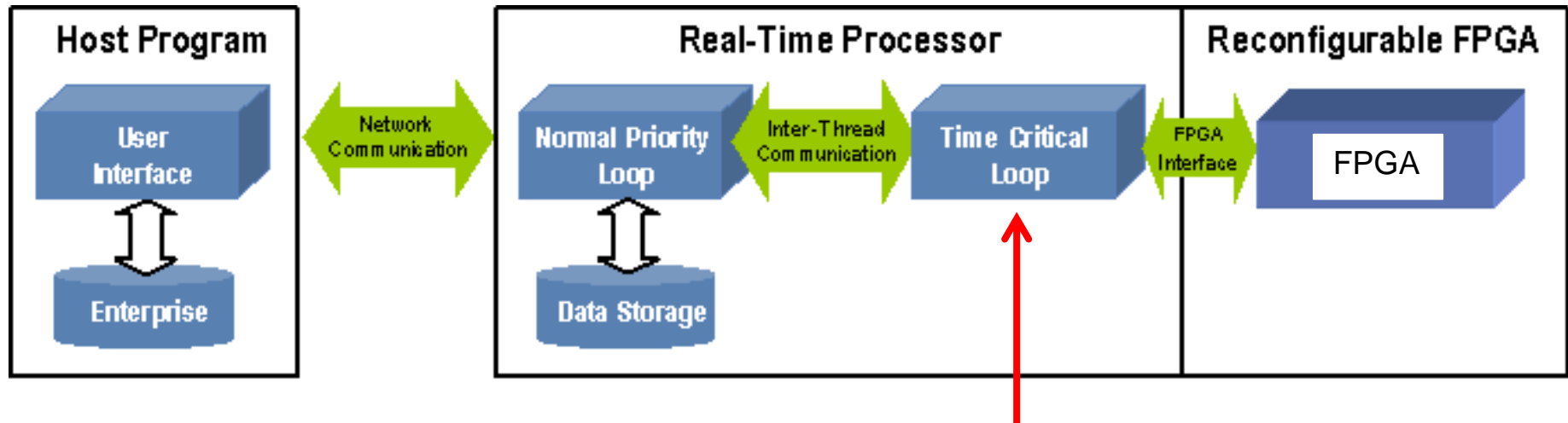
From LabVIEW to Hardware
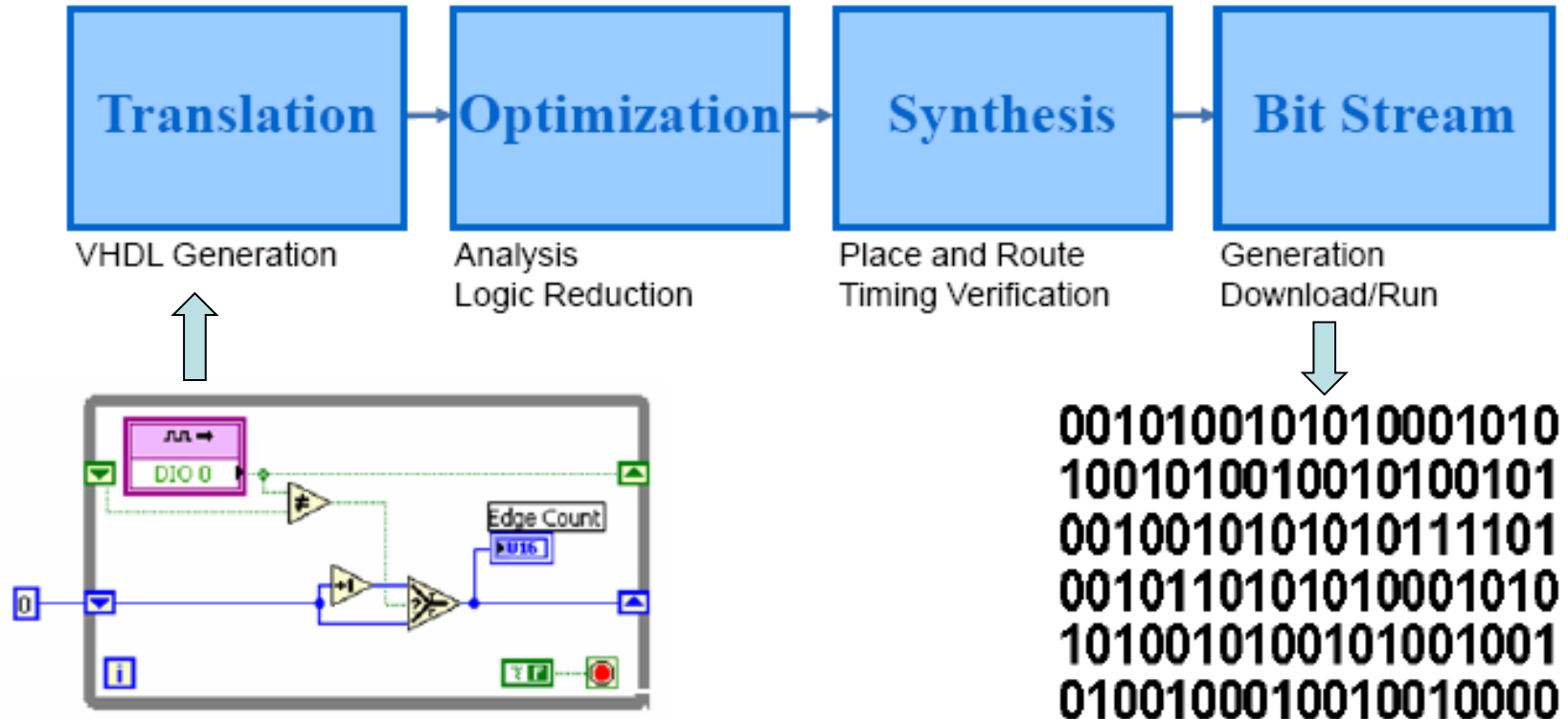
Can also use the LabVIEW IP Integration Node to include **VHDL code**

Architecture for Advanced Embedded DAQ-applications
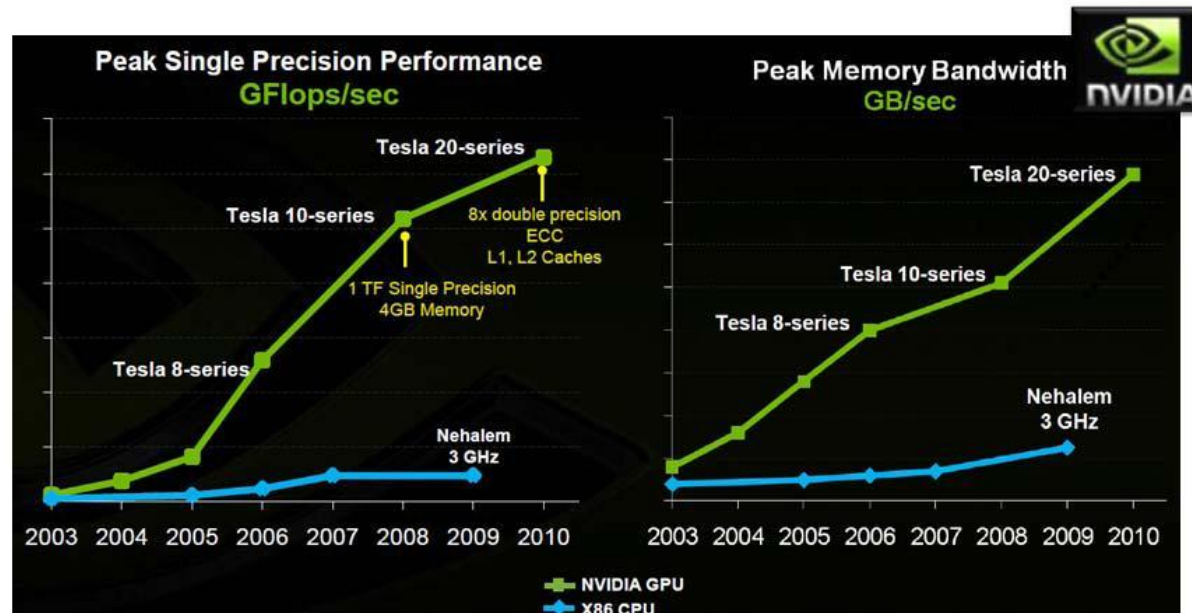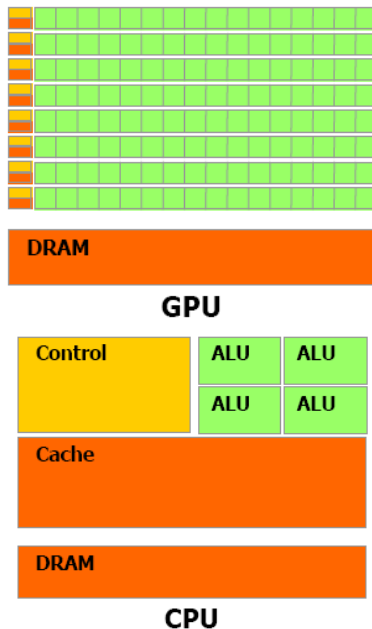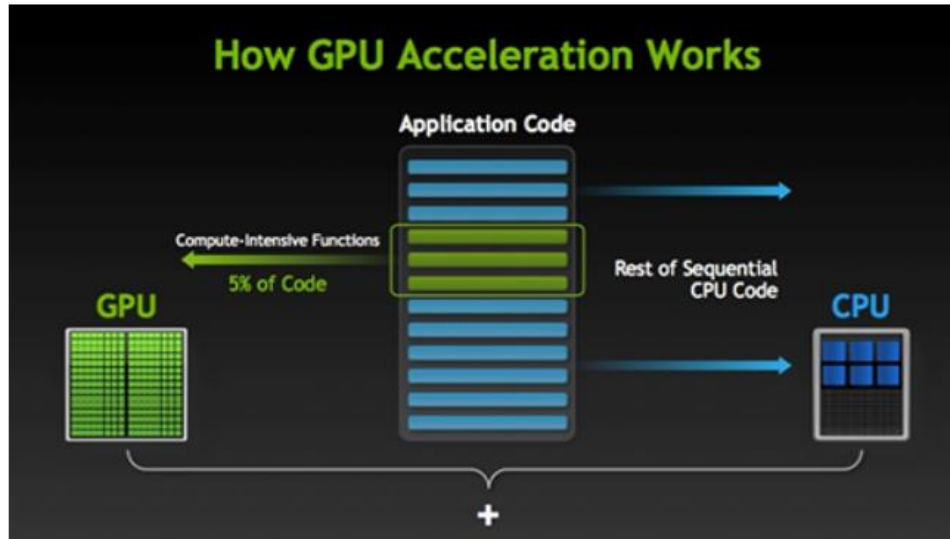
PC – Windows OS

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

GPUs

- GPU = Graphics Processing Unit
- Can be used as <u>hardware accelerator</u>
- 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**

How GPU Acceleration Works

Application Code

Compute-Intensive Functions

5% of Code

GPU

Rest of Sequential CPU Code

CPU

+

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.

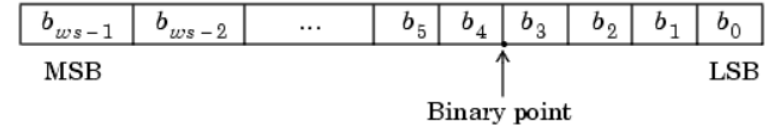**GPUs have thousands of cores to process parallel workloads efficiently**

+

CPU
MULTIPLE CORES

GPU
THOUSANDS OF CORES

From NVIDIA
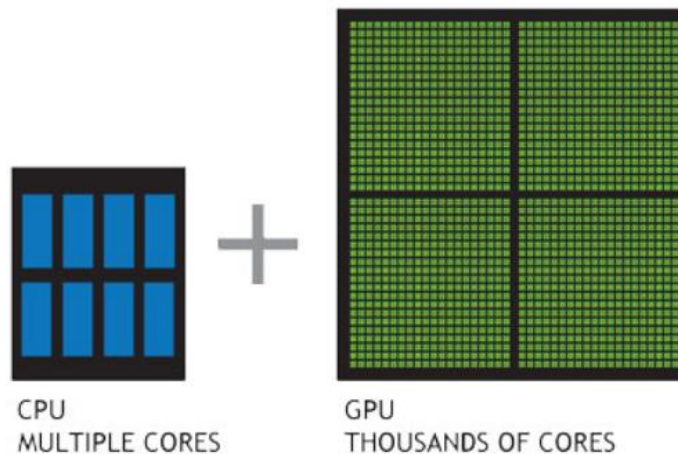
NVIDIA Tesla GPUs

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

Number representation

$$\boxed{b_{ws-1}} \boxed{b_{ws-2}} \boxed{\ldots} \boxed{b_5} \boxed{b_4} \boxed{b_3} \boxed{b_2} \boxed{b_1} \boxed{b_0}$$

MSB   Binary point   LSB

- **Fixed-point number system**
  - For example, the value 1.23 can be represented as 1230 (an integer), with scaling factor of 1/1000.
  - Fixed binary point.
  - Examples: 123.45, 1234.56, 12345.67
  - To represent larger numbers or to achieve an accurate result a larger number of bits are needed.

$$1.2345 = \underbrace{12345}_{\substack{\text{significand} \\ \textbf{mantissa}}} \times \underbrace{10}_{\text{base}}\overbrace{{}^{-4}}^{\text{exponent}}$$

- **Floating-point number system**
  - Floating-point representations are easier to use than fixed-point representations, because they can handle <u>a wider dynamic range.</u>
  - Examples: 1.234567, 123456.7, 0.00001234567, 1234567000000000
  - The logic needed to implement a given arithmetic operation is considerably more complex and area demanding compared to fixed-point numbers.
  - Limited use in FPGA logics, but can be used in DSP blocks and microprocessor core(s) inside the FPGA.

Ethernet for real-time applications

Standard Ethernet

Non-standard Ethernet
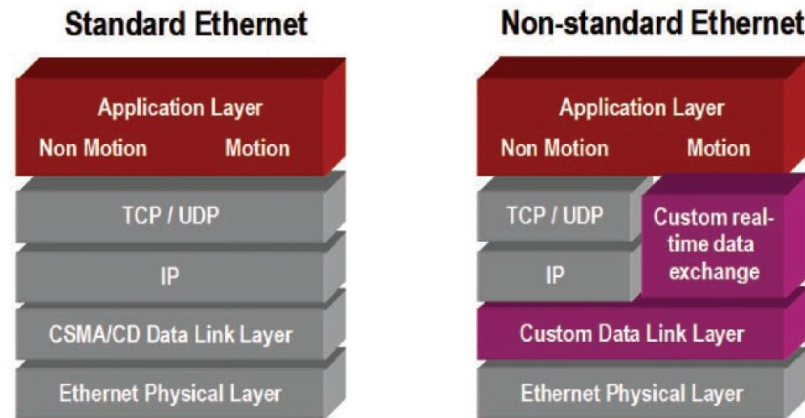
- Remote I/O can demand reaction in the 5-10 ms region. Motion Control demands even higher determinism with cycle times into the microsecond region.

- Standard Ethernet communication utilizes **TCP/IP, which is inherently non-deterministic and has a reaction time in the hundreds of milliseconds**. <u>In an effort to boost determinism some networks utilize custom technologies in the transport and network layers of the Ethernet stack</u>. These networks merely use TCP/IP as a supplemental channel to provide non real-time data transfers. By bypassing the TCP/IP protocols, such proprietary networks limit the end user's ability to use standard, off-the-shelf Ethernet products such as routers, switches, firewalls, etc. This limitation destroys one of the fundamental advantages of standard Ethernet - the availability of low-cost, ubiquitous COTS Ethernet hardware.

- By using UDP instead of TCP the reaction time comes down to about 10 ms at best. **UDP is not suited for "hard" deterministic distributed systems**.

Level shifting of bipolar signals

- Analog input signals can be unipolar or bipolar.
  - Unipolar signals swing between zero and positive full-scale.
  - Bipolar signal swings above and below some reference point, typically ground.

- In battery powered embedded systems **single-supply circuits** are often used to save power. Therefore a level shifting is required to convert a bipolar signal into a unipolar signal.

Level shifter example circuit

- +2.5 V single supply electronics
- +/- 10 V input signal
- NB: Sensor resistance Rs should be low!
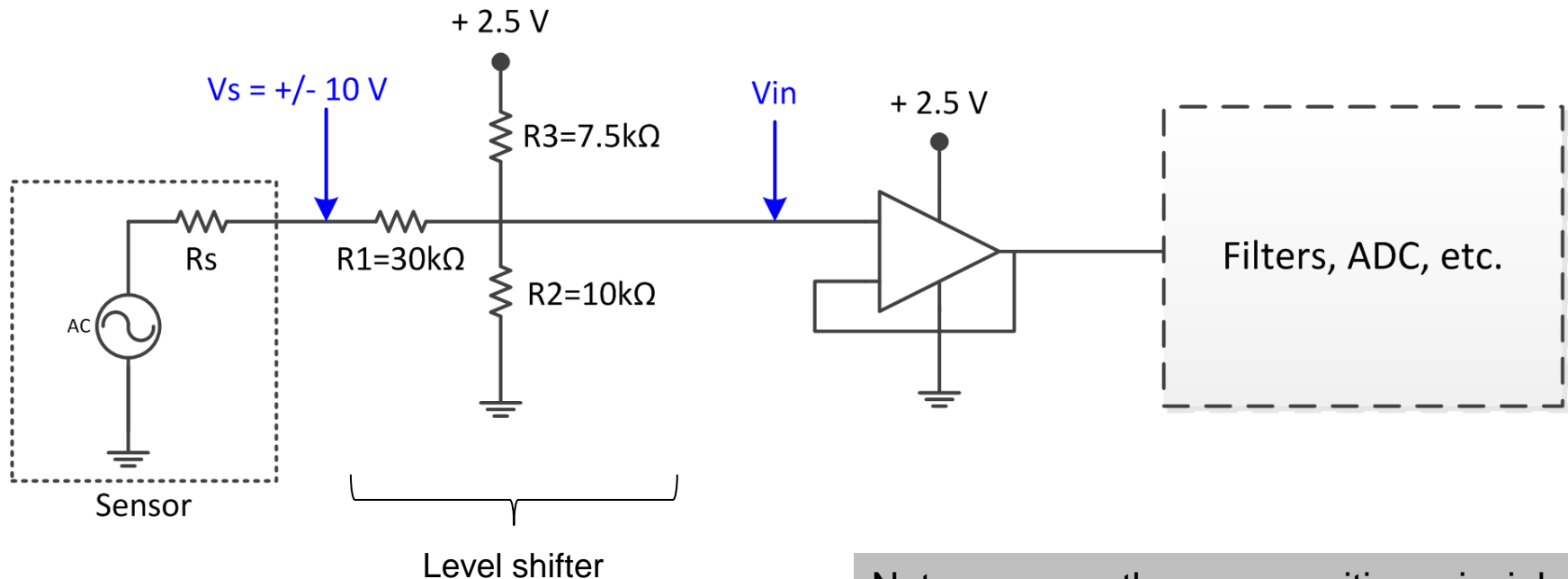
Note: use e.g. the superposition principle to calculate the voltage at point Vin

Telemetry

- A telemetry system is used to transmit data (in real time) from one location to another
  - E.g. send data from a sounding rocket/missile/satellite to a receiver station on the ground.
- A serial digital signal (a sequence of data bits of level '0' or '1' along a single path) is often referred to as a **pulse code modulation (PCM).**

- IRIG telemetry standard 106
  - This is a common telemetry standard worldwide in military and aerospace test and monitoring applications.

- S-band telemetry (2.2 – 2.4 GHz) usually used in military and aerospace test ranges.

UIO CubeStar page

Telemetry II

- **PCM/FM** (FM-modulation of the PCM signal)
  - '1' gives frequency f1 = fc + D
  - '0' gives frequency f1 = fc – D

  Where fc is the carrier frequency and D is the deviation from the carrier frequency

- **Manchester encoding**
  - The data and clock are combined into one signal, so that the receiver can recover the transmitter clock (self-clocking).
    - XOR of data and clock (in principle)
    - Gives at least one transition for each clock period
  - '1' is a low to high transition
 - '0' is a high to low transition
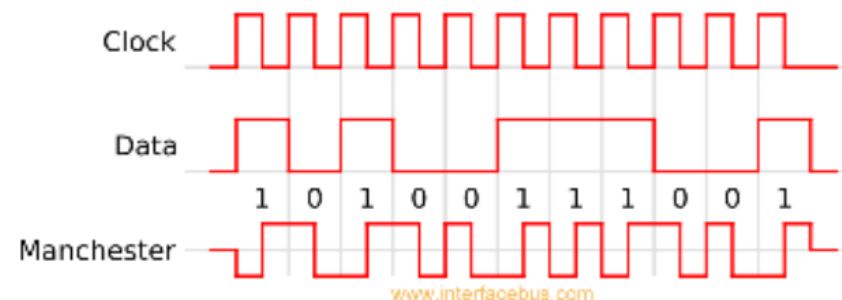
Why use Manchester type encoding?

- Even if the transmitter and receiver are almost perfectly synchronized, the infinitesimal delay of the transmission medium would have to be accounted for.

- Adding a separate clock line when possible doubles the number of wires.

- **For wireless transmission the data and clock has to be combined into one signal.**

- A long string of nulls (zeroes) will look like a dead or disconnected line.

- A long sting of ones look like a stuck level.

- Need transitions between '0' and '1' to recover the clock.

- Voltage averaged over time should tend toward zero (no DC offset).

- Problem:
  - Manchester encoding doubles the bandwidth requirement of the telemetry.

- Solution:
  - Use another "similar" but more effective code, such as RNRZ (Randomized Non Return to Zero) or 8b/10b
  - RNRZ often used by telemetry systems; 8b/10b encoding used in communications systems

UIO Embedded systems examples from aerospace
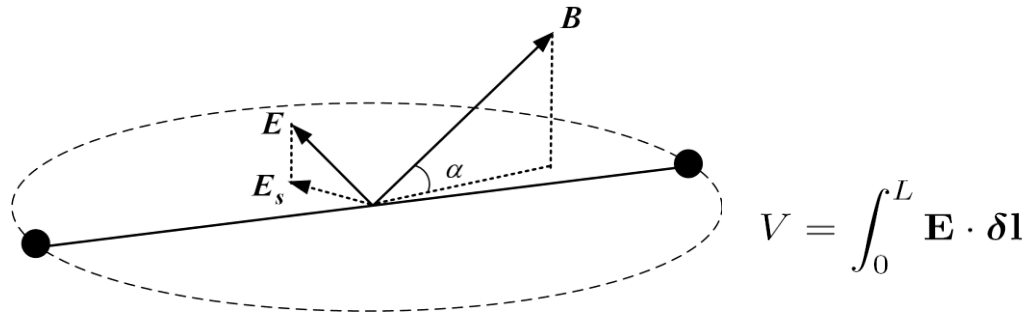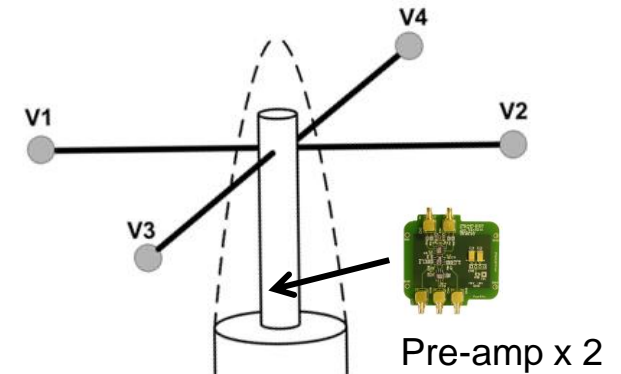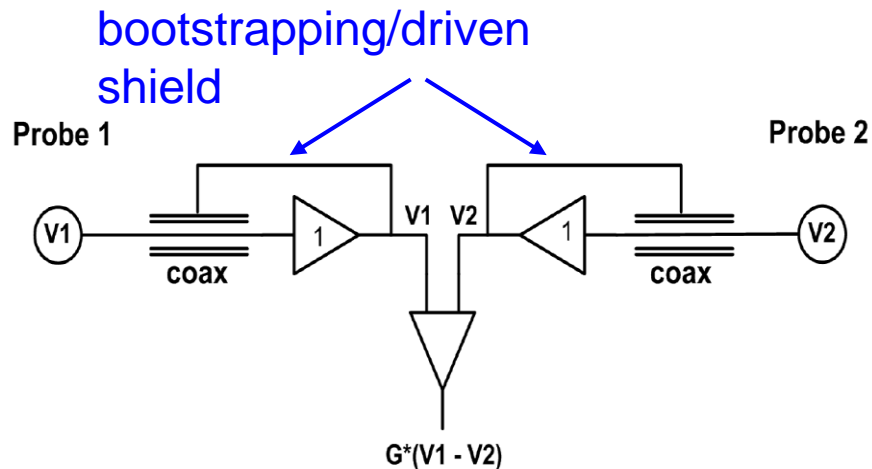
Inertial Reference Unit (IRU)

Roll

Roll (X)

Pitch (Y)

Yaw (Z)
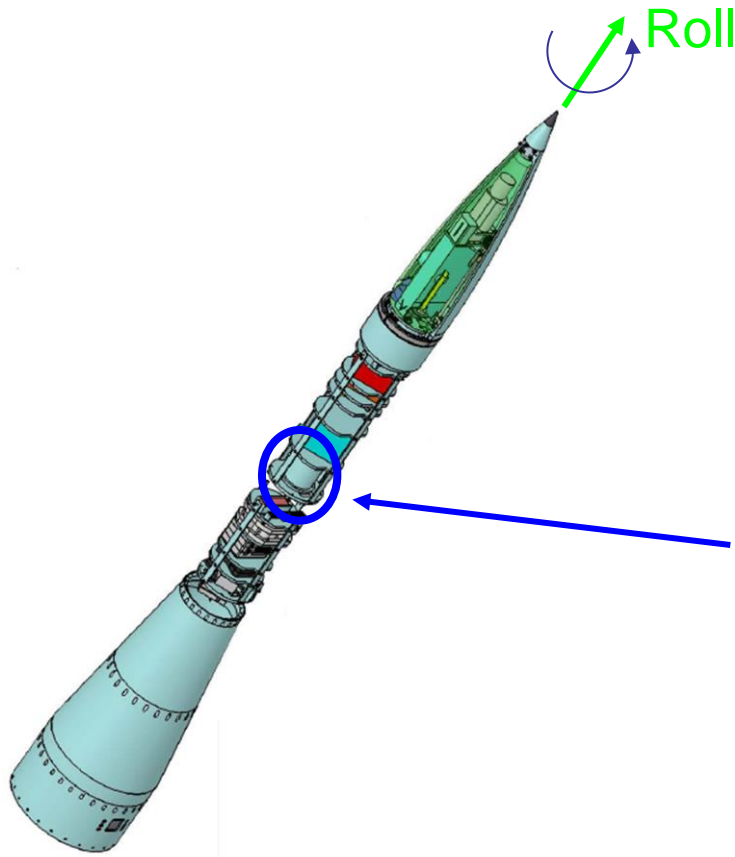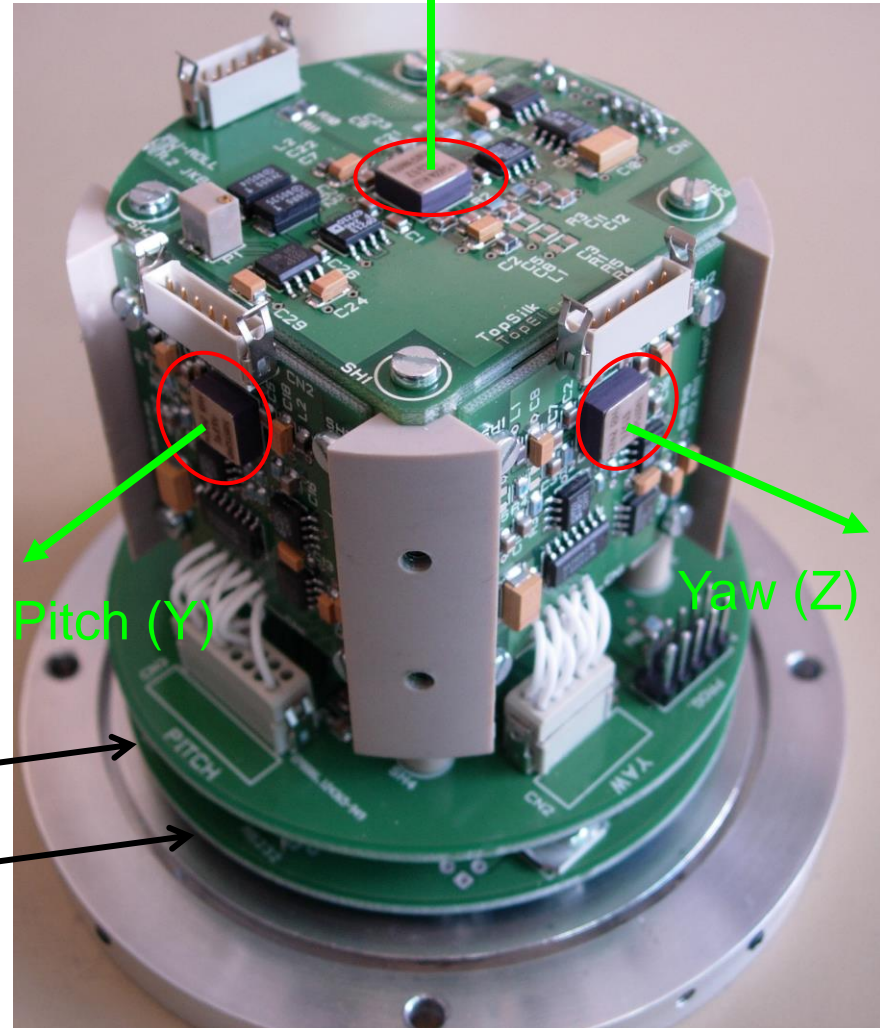
DAQ PCB with FPGA

Power & interface PCB

E-field instrument for sounding rockets

$$V = \int_0^L \mathbf{E} \cdot \delta\mathbf{l}$$

Pre-amp x 2

Pre-amplifier

bootstrapping/driven shield

Probe 1

Probe 2

V1

V2

coax

coax

G*(V1 - V2)

DAQ-unit with FPGA