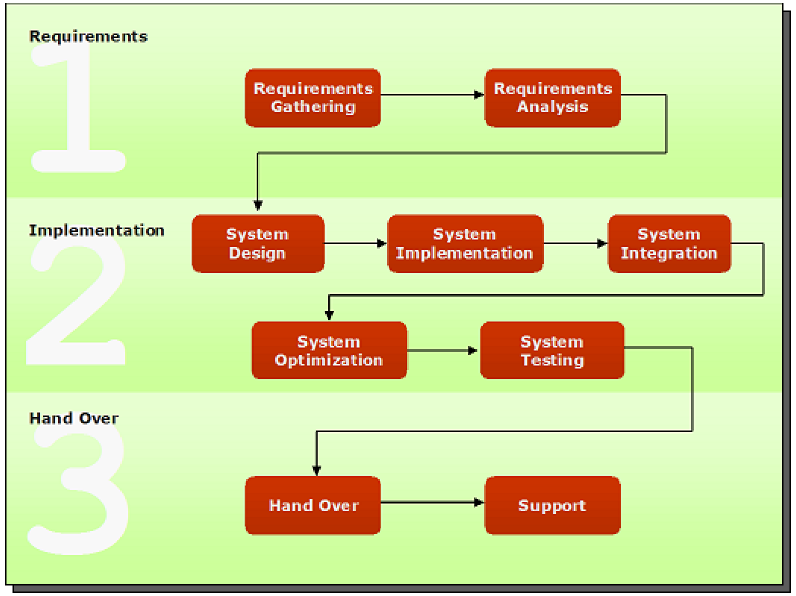
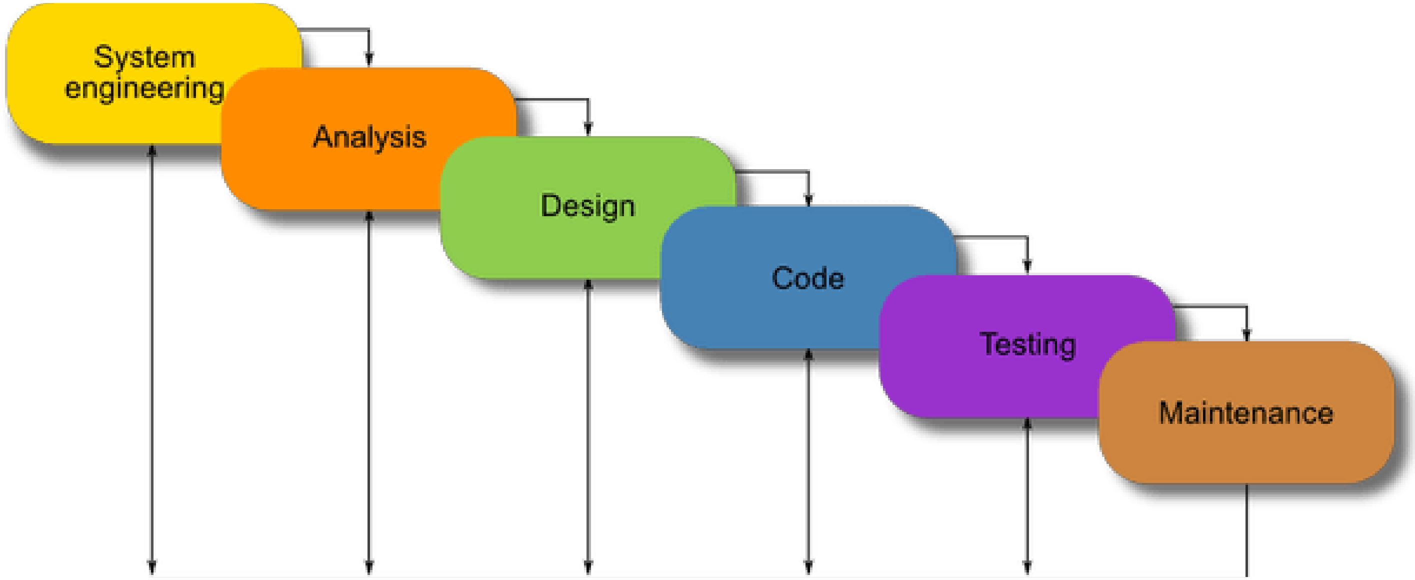
CO3053 – Embedded Systems

3. Embedded System Development Process





Learning Outcome

- Students are expected to be able to …
  - Describe each step in the process
  - Sort the steps in the process in correct order

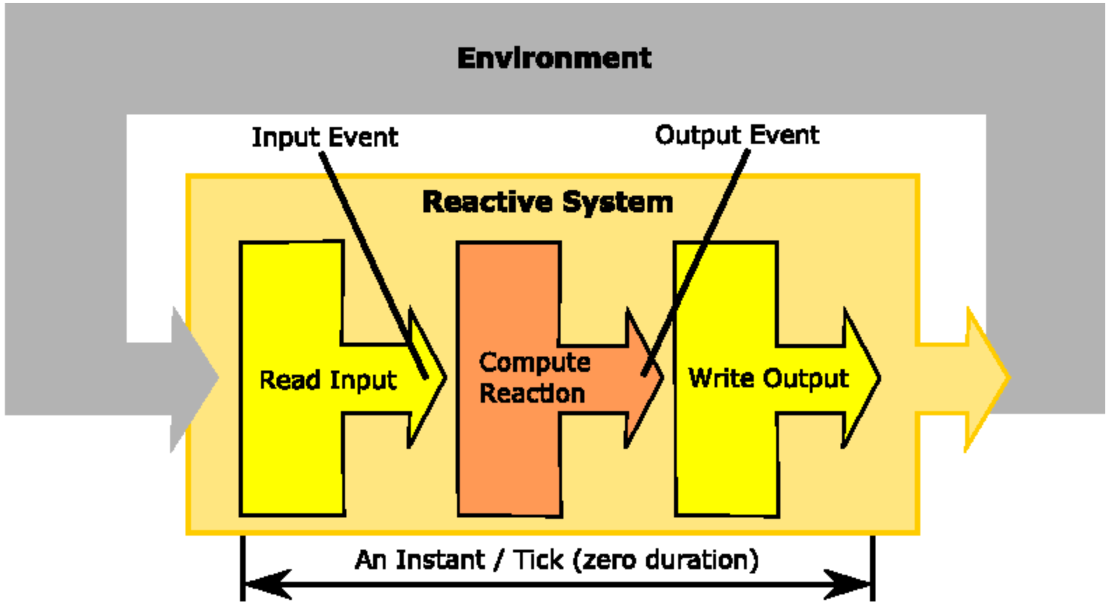
Embedded System Design – Challenges

- **Increasing application complexity**
  - Large systems with legacy functions
  - Flexibility requirements
  - Examples: multimedia, automotive, mobile communication
- **Increasing target system complexity**
  - Mixture of different technologies, processor types, and design styles
  - Large systems-on-a-chip combining components from different sources (IP market)
- **Numerous constraints and design objectives**
  - Examples: cost, power consumption, timing constraints, dependability
- **Reduced and overlapping design cycles**

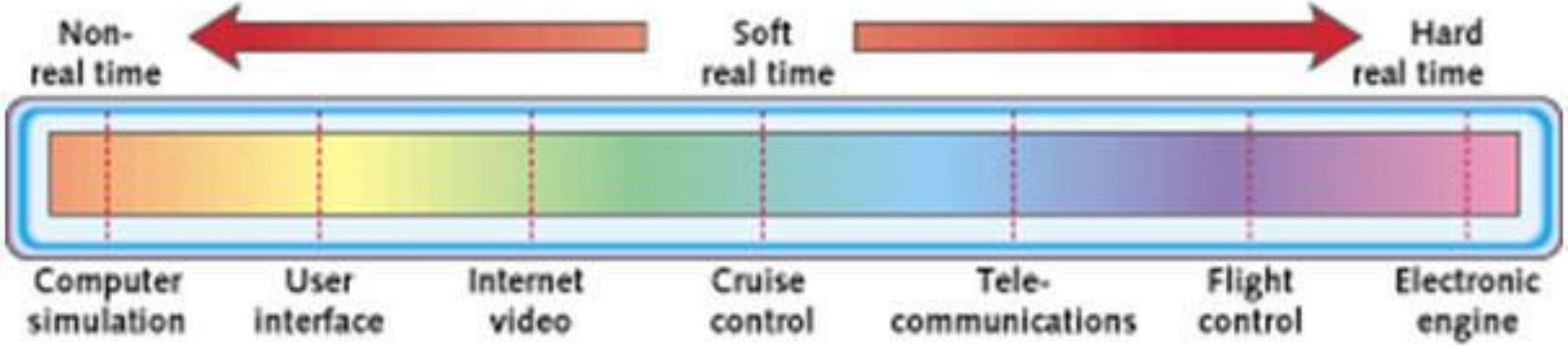
anhpham@hcmut.edu.vn

Embedded System Requirements

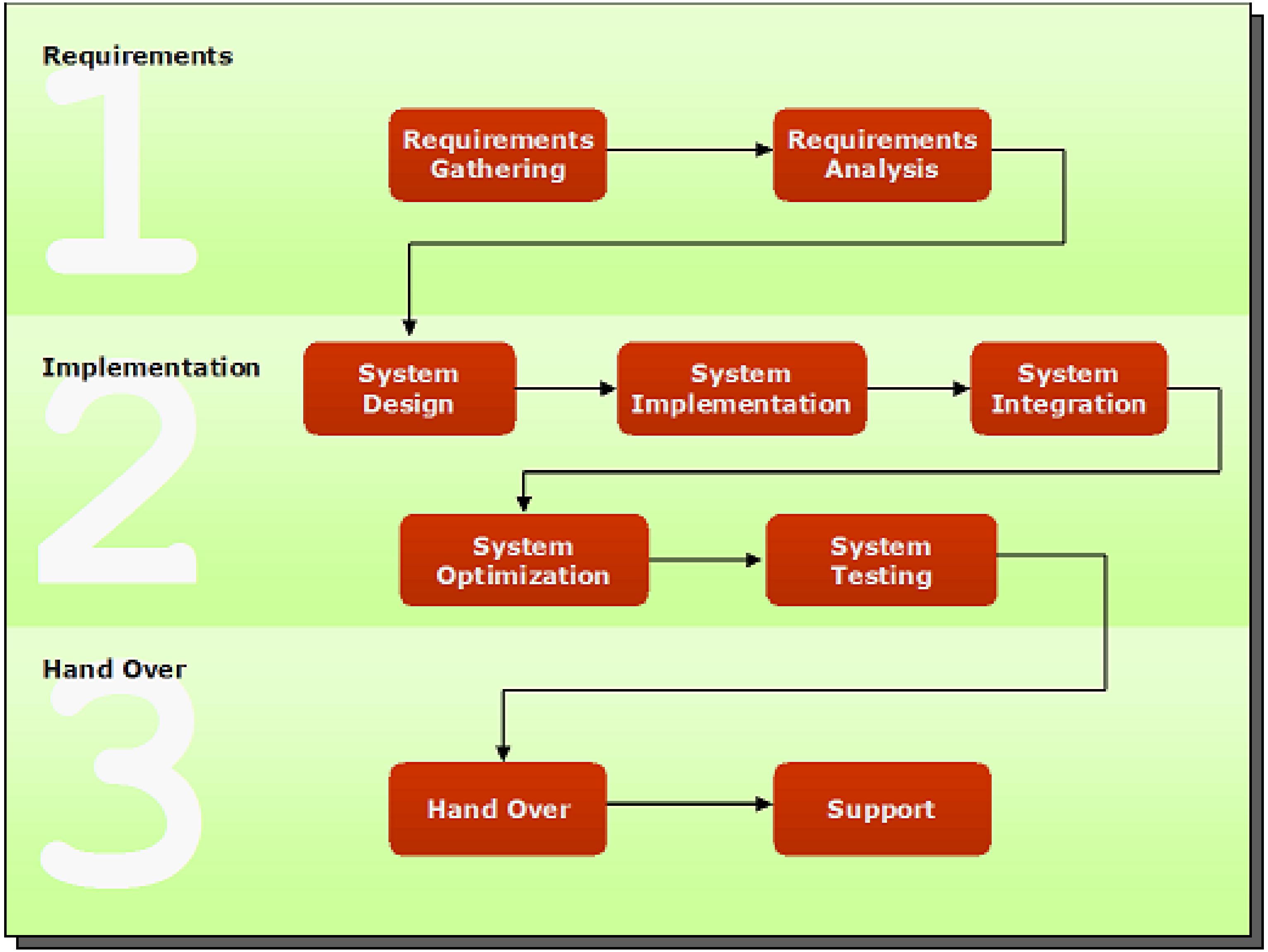
- Reactive systems
  - The system never stops.
  - The system responds to signals produced by the environment.



- Real-time systems
  - Timing constraints on task execution.
  - Hard and soft constraints.



Embedded System Development Process



Requirements Gathering

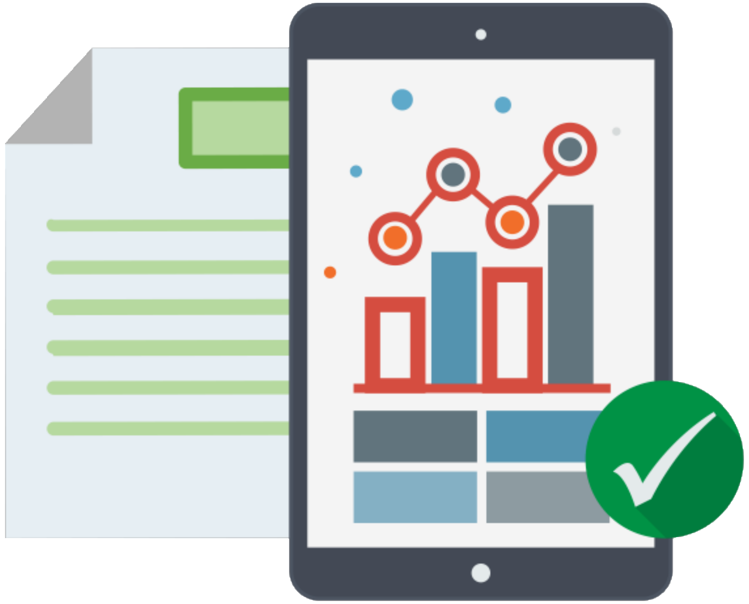
- Understand the problem statement and scope definition.

- Identify Functional and nonfunctional requirements
  - Multimode or multifunctional system
  - Size, cost, weight, etc.

- Determine deployment parameters
  - Application domain and operational environment
  - Legal and regulatory requirements
  - …

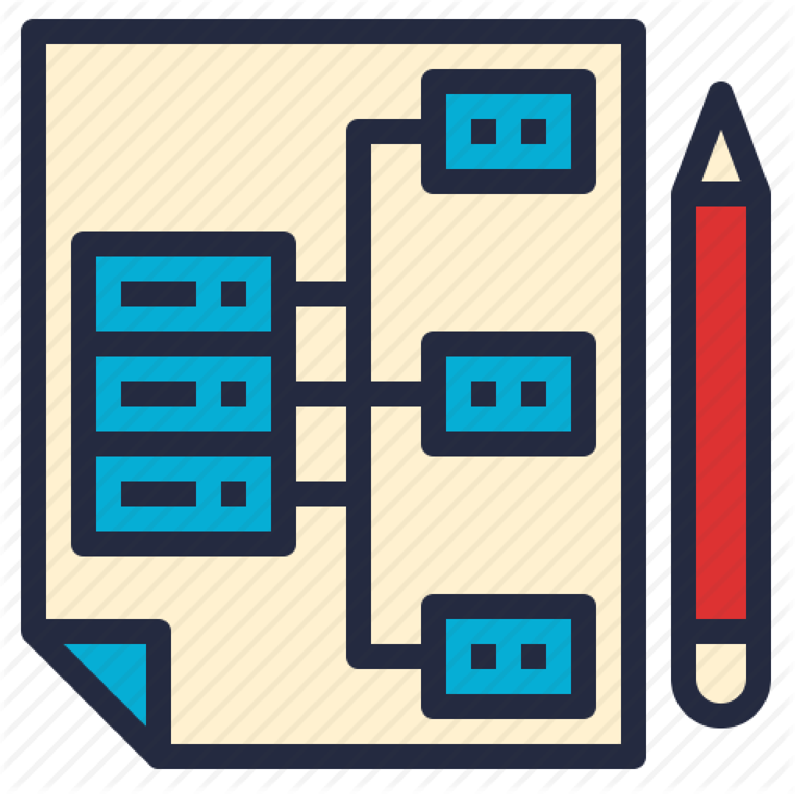
Requirements Analysis

- Identify the variables in performance, hardware, firmware, software, …
- Estimate cost, complexity
- Determine tradeoff



System Design

- System architecture
  - Block diagram
- Hardware-software partitioning
- Hardware and software selection
  - Hardware platform
  - Programming language
  - Operating system
  - Development tools
- Prototyping and testing strategy



System Implementation

- **Hardware Implementation (if needed) & Coding**

- **Cross-platform development**
  - Usually, the ES is not strong enough ➠ need another platform to build application (usually use PC), and the application/OS is executed on ES.
  - Cross: developed on one platform, run on another platform

- **Cross-compiler**: the compiler run on one platform (PC), and it produce executable file to run on another platform (ES)

- **Porting**: reproduce an application/OS which is developed for a platform to run on another platform
  - Ex: We want to reproduce a Linux distribution to run on our ARM platform

Testing

- Unit & Integration testing

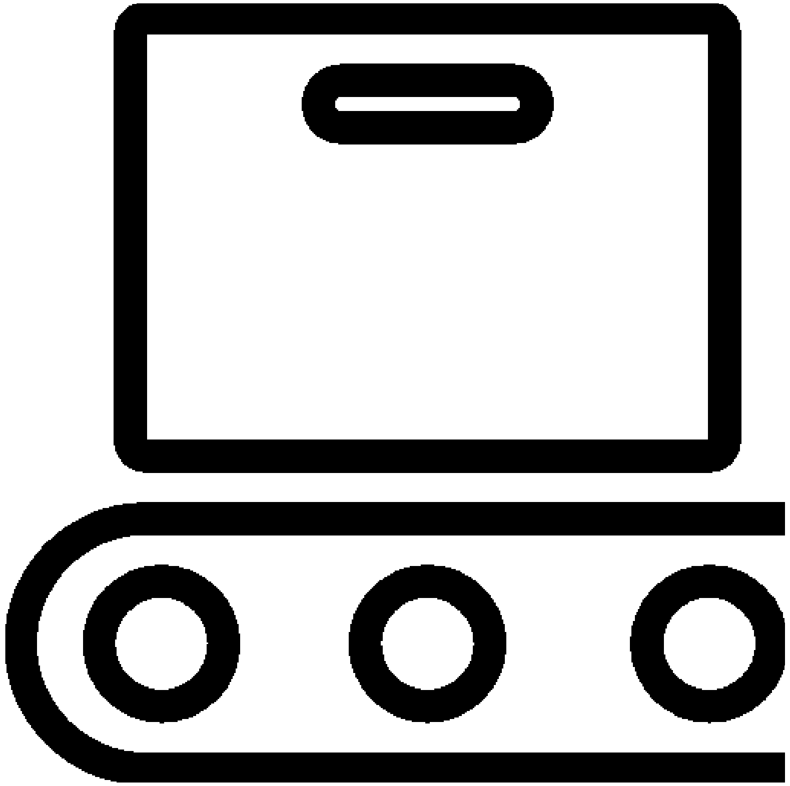
- Verify the Software on the Host System
  - Compile and assemble the source code into object file
  - Use a simulator to simulate the working of the system

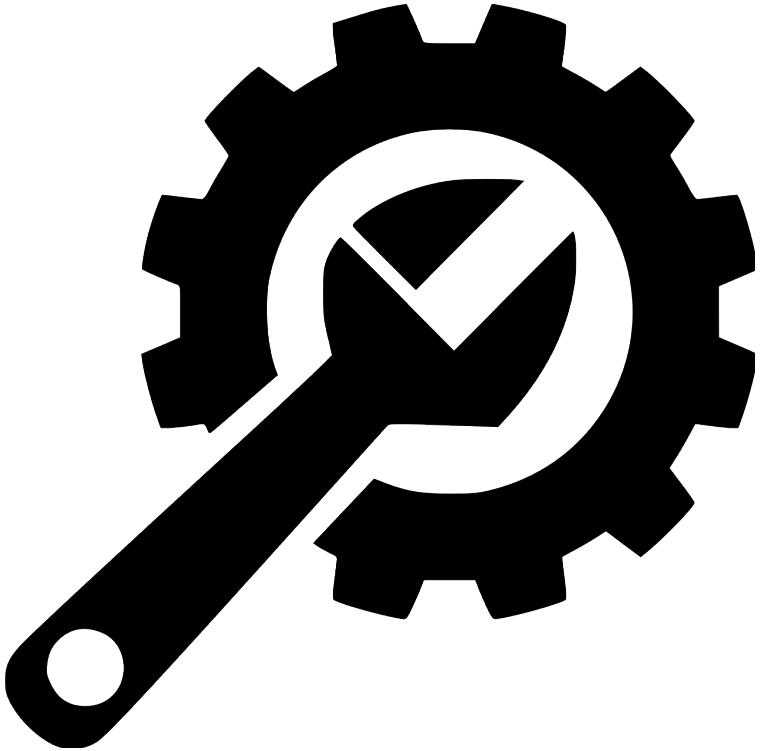
- Verify the Software on the Target System
  - Download the program using a programmer device
  - Use an Emulator or on chip debugging tools to verify the software

Integration & Optimization

- System integration
  - Involve the actual integration of the hardware and software modules to produce the full working system.
- System optimization (if required)
  - Optimize trade-off parameters such as cost or performance.

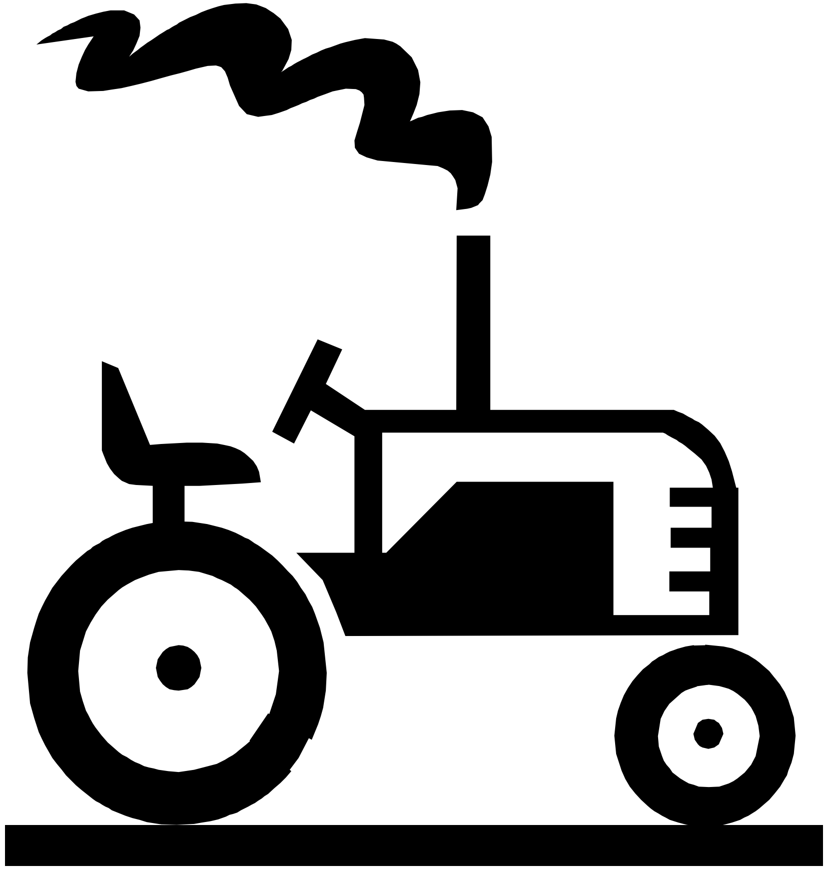
Deployment and Maintenance





Example: Engine Control Unit (ECU)

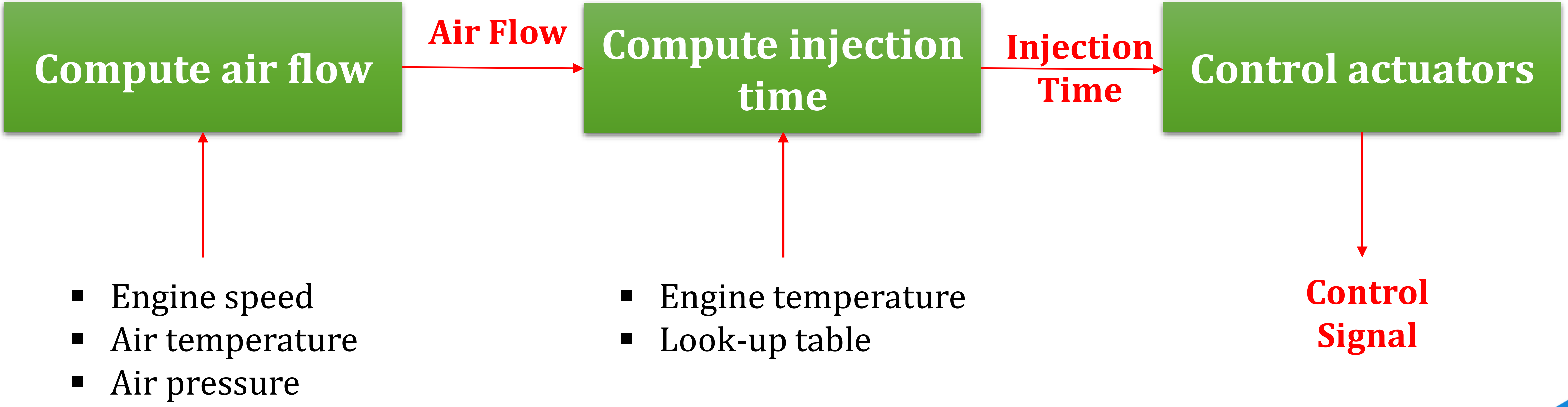
- **Task**: control the torque produced by the engine by the timing fuel injection and spark.
  - **Control Injection Time**



- **Major constraints**
  - Low fuel consumption
  - Low exhaust emission

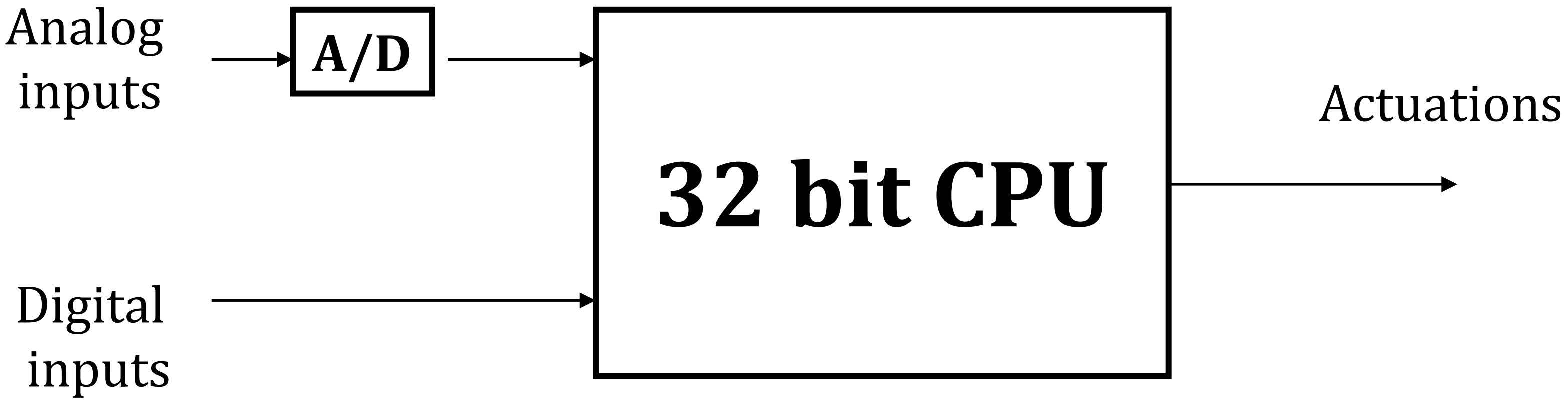
ECU Control Injection Time – Analysis

- **3 Sub Tasks**
  - Compute air flow
  - Compute injection time
  - Control actuators (torque)



ECU – Design Option #1

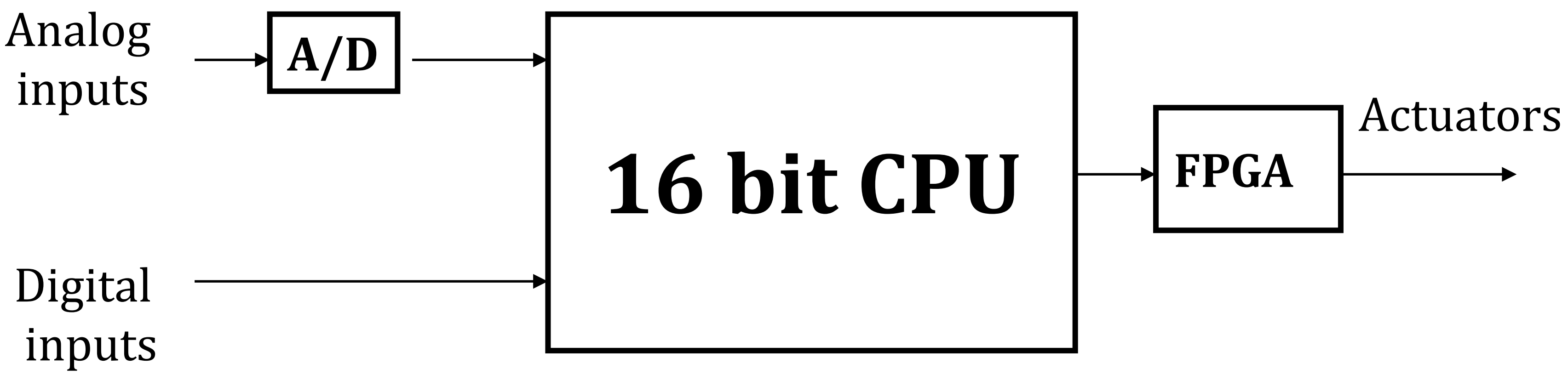
- **Use a single CPU to**
  - Process input data
  - Compute outputs
  - Control actuators



May not meet timing requirements

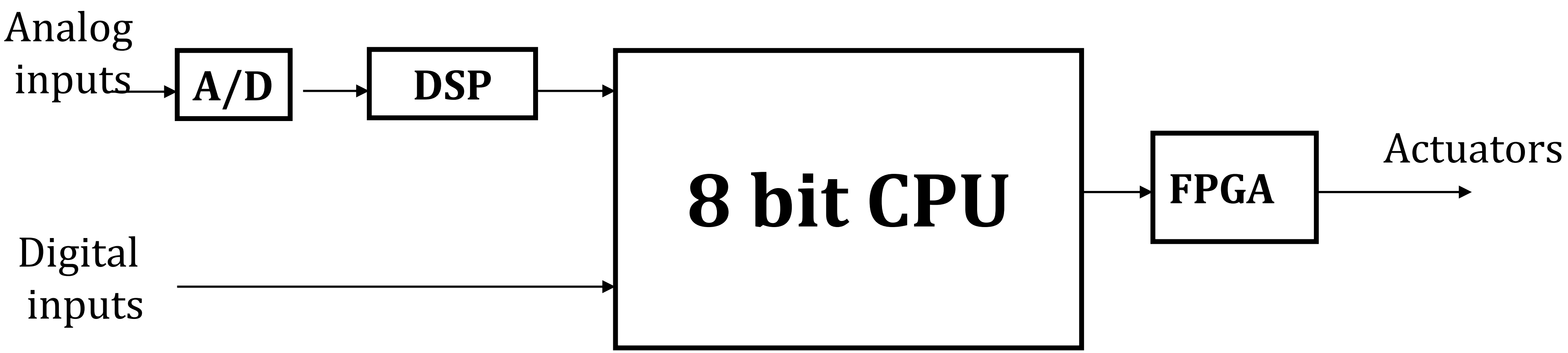
ECU – Design Option #2

- **Combine CPU and FPGA**
  - Use CPU to
    - Process input data
    - Compute outputs
  - Use FPGA to control actuators



ECU – Design Option #3

- **Combine DSP, CPU, FPGA**
  - Use DSP to process input data
  - Use CPU to computes outputs
  - Use FPGA to control actuators



Question and Discussion