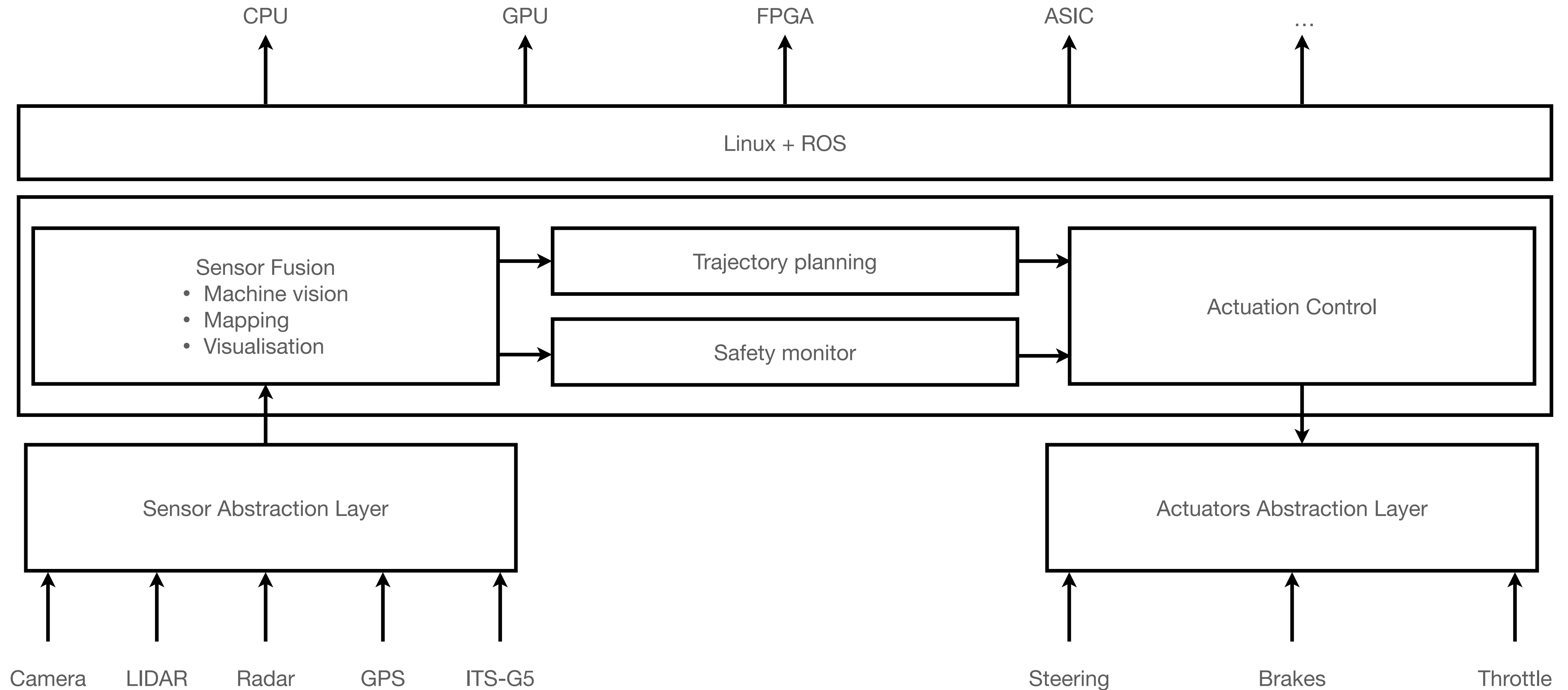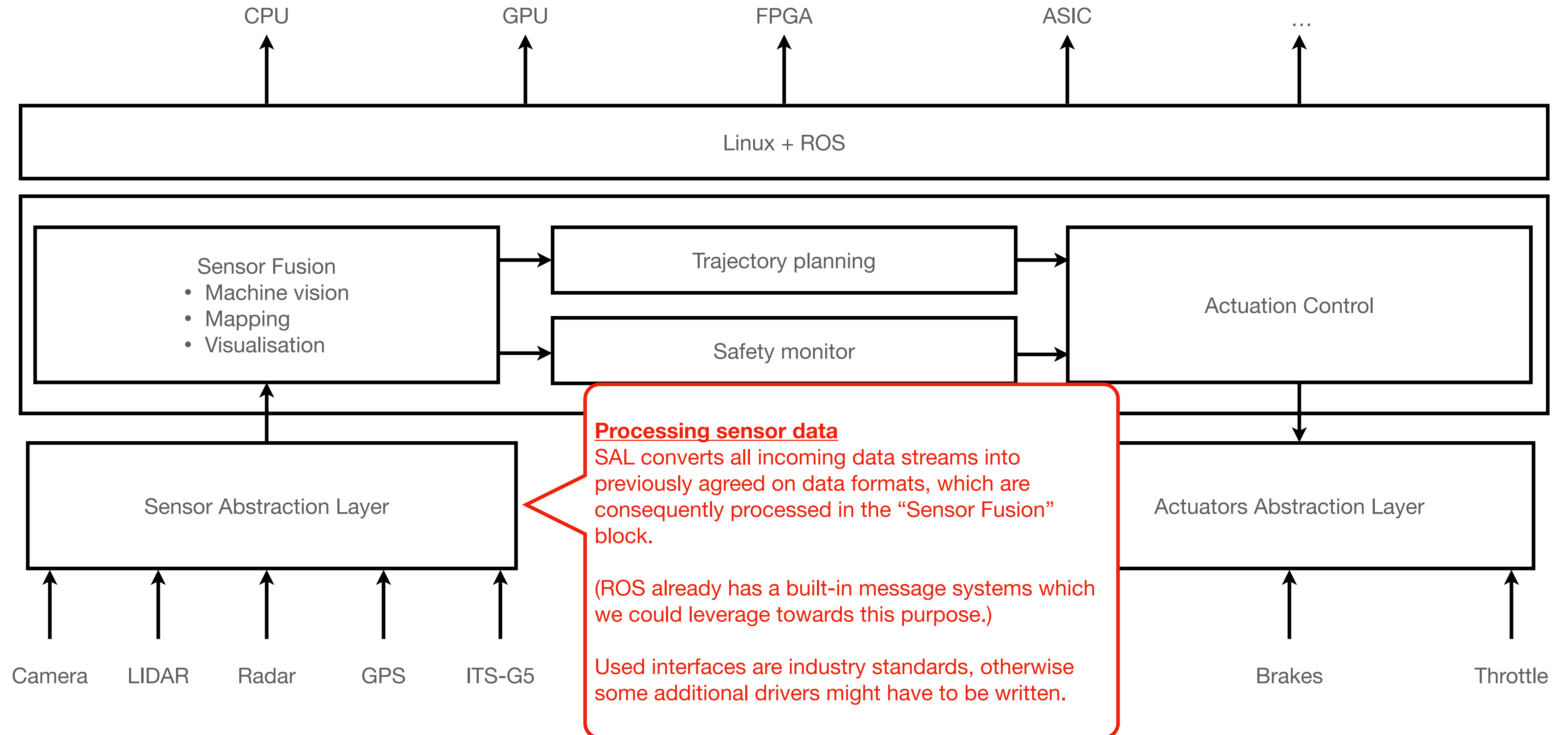# Vehicle Automation

Flanders MAKE

Maksim Nesterenko - 1/02/21

1. What software architecture would you propose
   a. in order to process all sensor data,
   b. with the need for sensor fusion to increase probability detection and
   c. to support modularity and scalability e.g. for future enhancements and use within multiple projects?

Software Architecture

CPU GPU FPGA ASIC ...
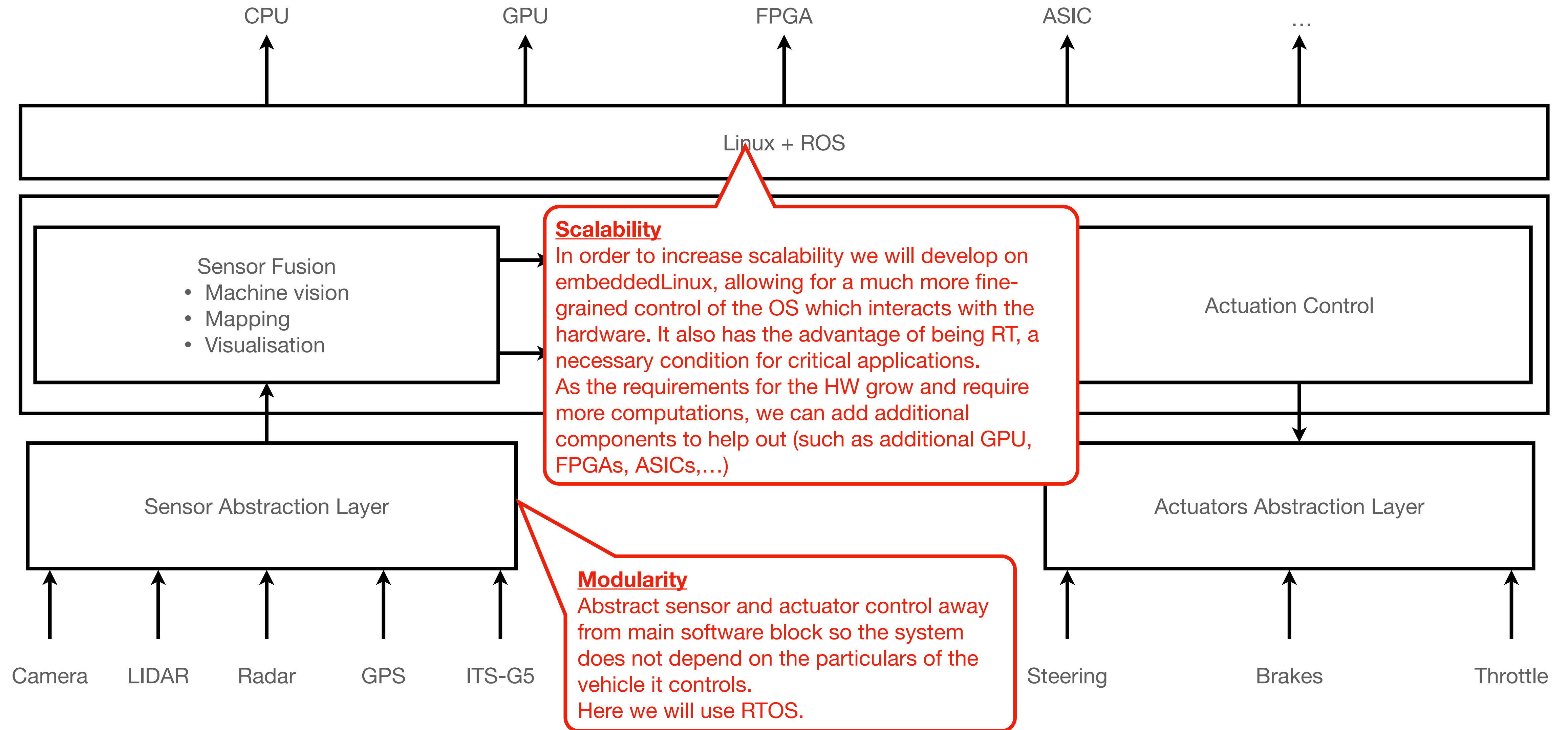
Linux + ROS

Sensor Fusion
- Machine vision
- Mapping
- Visualisation

Trajectory planning

Safety monitor

Actuation Control

Sensor Abstraction Layer

Actuators Abstraction Layer

Camera LIDAR Radar GPS ITS-G5

Steering Brakes Throttle

Software Architecture

CPU   GPU   FPGA   ASIC   ...

Linux + ROS

Sensor Fusion
- Machine vision
- Mapping
- Visualisation

Trajectory planning

Safety monitor

Actuation Control

Sensor Abstraction Layer

Processing sensor data
SAL converts all incoming data streams into previously agreed on data formats, which are consequently processed in the "Sensor Fusion" block.

(ROS already has a built-in message systems which we could leverage towards this purpose.)

Used interfaces are industry standards, otherwise some additional drivers might have to be written.

Actuators Abstraction Layer

Camera   LIDAR   Radar   GPS   ITS-G5

Brakes   Throttle

Software Architecture

Software Architecture

CPU   GPU   FPGA   ASIC   …

Linux + ROS

Sensor Fusion
- Machine vision
- Mapping
- Visualisation

Actuation Control

Scalability
In order to increase scalability we will develop on embeddedLinux, allowing for a much more fine-grained control of the OS which interacts with the hardware. It also has the advantage of being RT, a necessary condition for critical applications.
As the requirements for the HW grow and require more computations, we can add additional components to help out (such as additional GPU, FPGAs, ASICs,…)

Sensor Abstraction Layer

Actuators Abstraction Layer

Modularity
Abstract sensor and actuator control away from main software block so the system does not depend on the particulars of the vehicle it controls.
Here we will use RTOS.

Camera   LIDAR   Radar   GPS   ITS-G5   Steering   Brakes   Throttle

2. Where are the attention points/bottlenecks within the proposed architecture?

Attention points/bottlenecks

- Safety
  - SW
    - Failure to detect object (correctly), faulty planning drive
    - Resistance to outside interference (hack)
    - Condition: HW is operating correctly
  - HW
    - Failure of sensor, computing unit, power,…
    - Redundancy: Overlapping sensor, multiple cores, power supply,…
- Throughput/Latency
  - Optimise software for minimal latency (NN!)
  - Parallelism?
    - Which tasks can take place simultaneously?
    - Dependencies?
  - Batching (increases latency)
  - Task precedence
  - Impact of environment?
    - City environment: lots of variables (pedestrians, road signs,…)
- Testing
  - Determine testing strategy for each unit
  - Usage of feedback loops?
- ROS
  - Mainly meant for r&d
  - "Security and scalability are not main concerns"

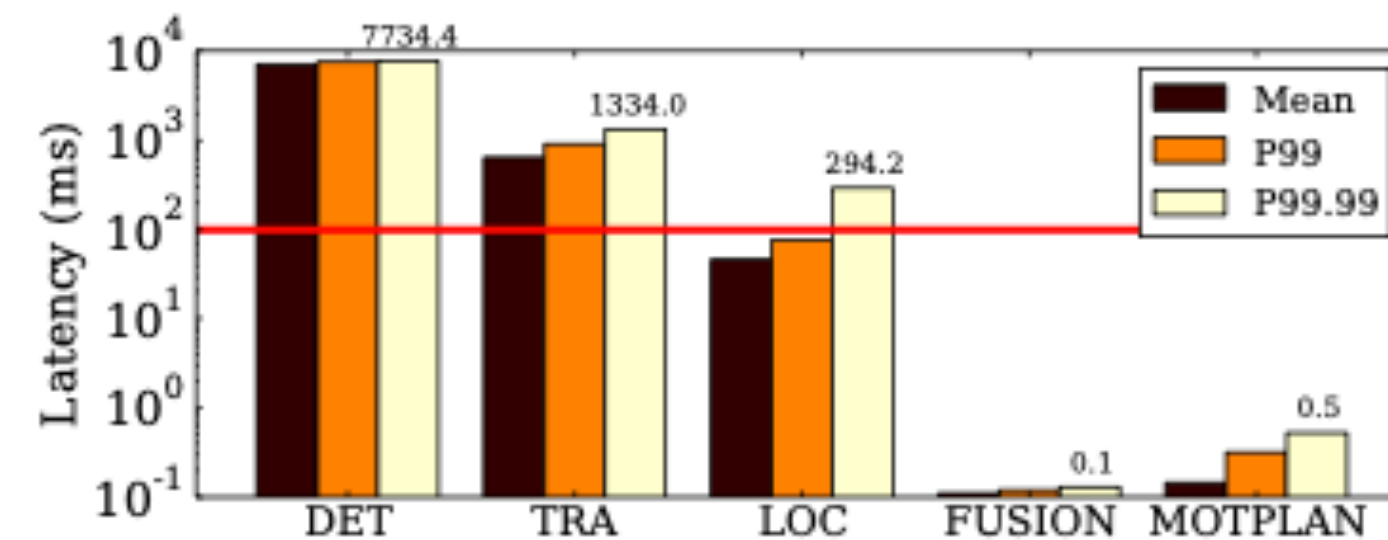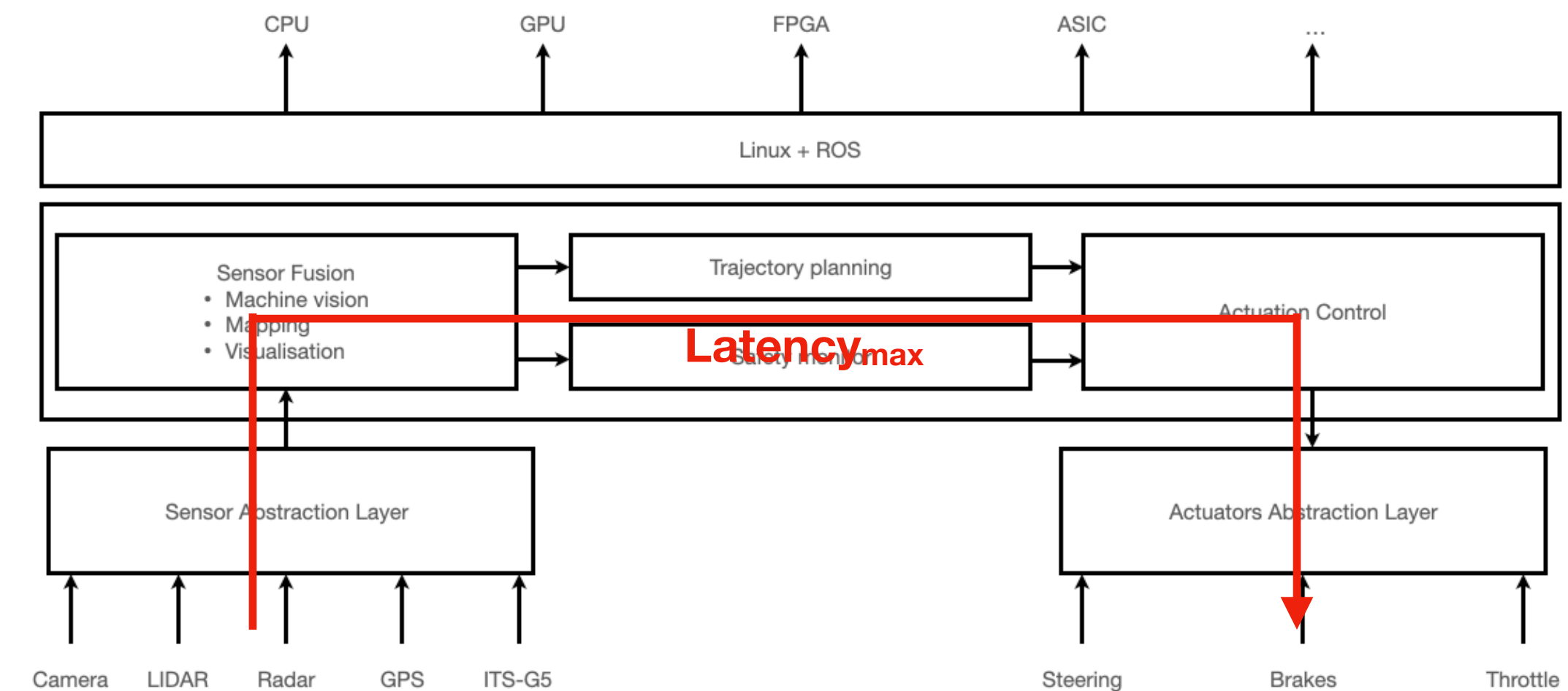| Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

Relative Energy Cost

Source

3. Propose derive hardware requirements, both performance as interface related, for the main blocks of perception, localization, decision making and actuation control?

Hardware requirements

- Power & Energy (<xW)
  - DNN (perception)
    - # of NN layers
    - Bit precision
    - # of off-chip accesses (SRAM <-> DRAM accesses)
  - Cooling
- Latency & Throughput
  - 30 fps -> $Latency_{max} < 33ms$
  - Min BUS speed:
    - 800 + 16 ~= 1 Gbit/s
  - HBM DRAM?
  - Optimize for:
    - Object Detection (YOLO, DNN-based)
    - Object Tracking (GOTURN, DNN-based)
    - Localization (ORB-SLAM, FE)
- Cost < x€
  - Development cost
    - Off-the-shelf hw (GPU) <-> custom (ASIC/FPGA)
  - Unit cost
    - Computing unit
    - Cooling
    - Redundancy
  - Production cost
- Flexibility
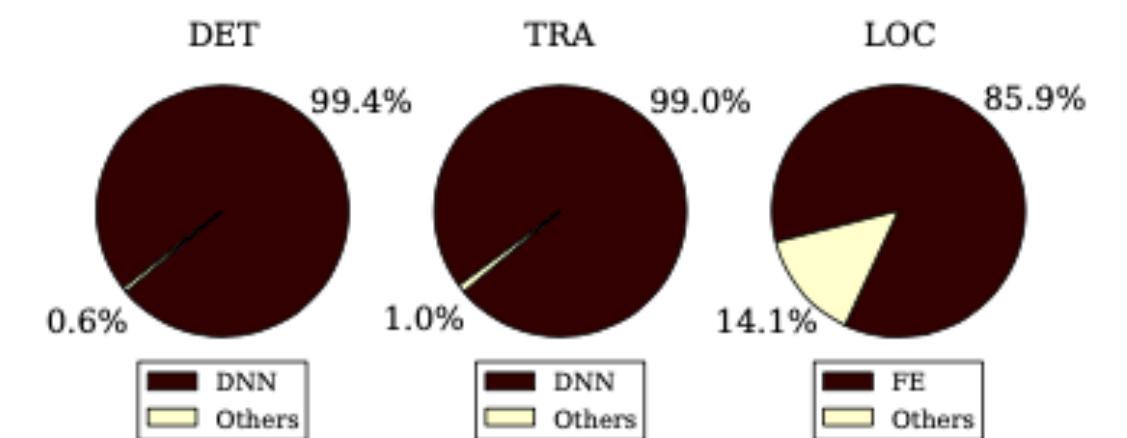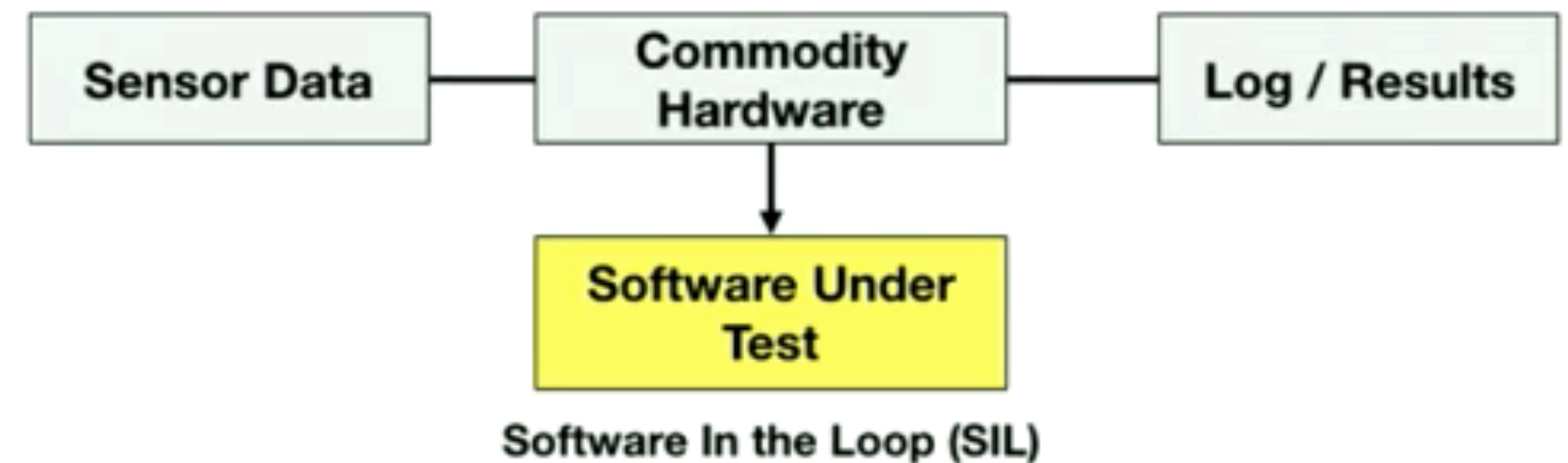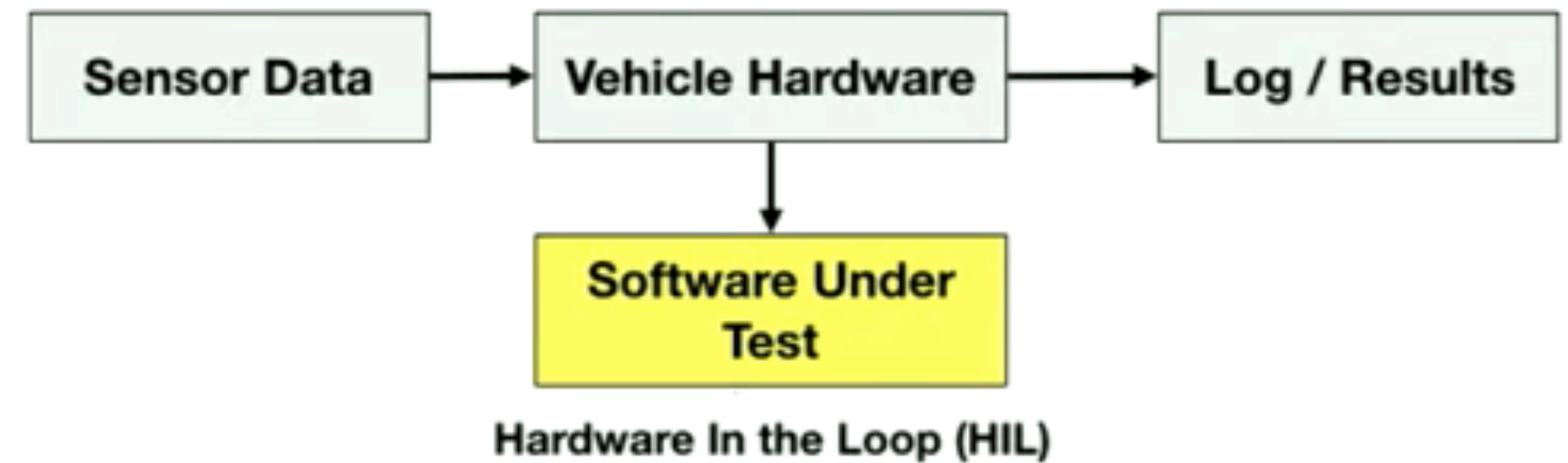- Impulse/vibration resistance
- Thermal constraints

Source

Figure 7. Cycle breakdown of the object detection (DET), object tracking (TRA) and localization (LOC) engines. The Deep Neural Networks (DNNs) portion in DET and TRA, and the Feature Extraction (FE) portion in LOC account for more than 94% of the execution in aggregation, which makes them ideal candidates for acceleration.

4. How can the proposed architecture be validated?

Validation

- Simulation (SW)
  - Unit test for all sensors/actuators
  - Unit test for each software block of self-driving
  - Full testing of self-driving in simulation
    - Allows for intricate test adjustments
    - Account for difference real/simulated data
  - Testing using pre-recorded data (in simulation)
  - Simulation of city environment?
- Run H-I-L to verify hardware requirements
- Real Life Validation
  - Test track
  - City environment

Hardware In the Loop (HIL)

Software In the Loop (SIL)

Source

Sources

- Autonomous driving systems hardware and software architecture exploration: optimising latency and cost under safety constraints

- A Standard Driven Software Architecture for Fully Autonomous Vehicles

- Efficient Processing of Deep Neural Networks: A Tutorial and Survey

- The Architectural Implications of Autonomous Driving: Constraints and Acceleration

- Efficient Processing of DNN

- Self-Driving Cars as Edge Computing Devices