Automotive SPIN

Milano – 17 February 2011

Experience with ISO 26262 ASIL Decomposition

Andrea Piovesan

Safety and Diagnostics Division

John Favaro

Consulting Division

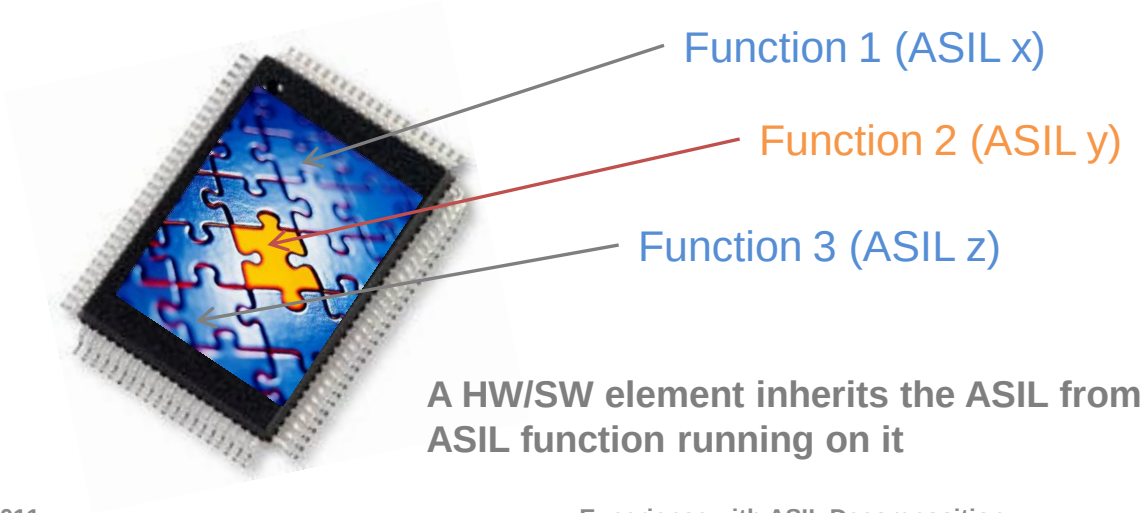
Automotive Safety Integrity Level

- The Automotive Safety Integrity Level (**ASIL**) expresses the criticality associated with a function of the system

- It defines the safety requirements that must be fulfilled by the design and development of the system in such a way that, even in conditions of failure, the system provides a sufficient margin of safety for the users (driver, passengers, road traffic participants, etc.)

ASIL Basics

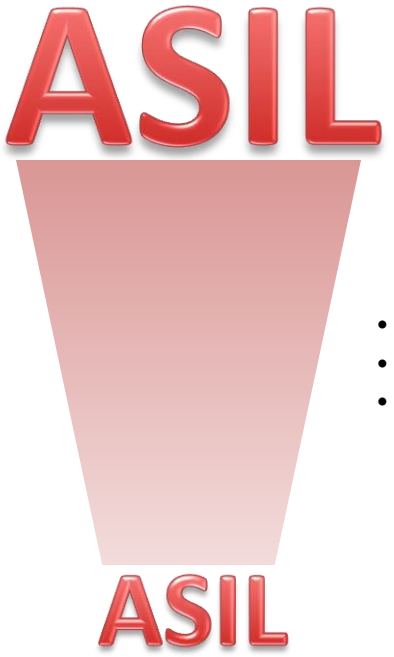
- The ASIL is not calculated for a physical system component - it is calculated for a **function**
- THE ASIL associated with a function is then inherited by the software and hardware elements that realize the function
- It could happen that a hardware component or a software element realizes several functions with different ASILs (e.g. microcontroller)
- In this case, the ASIL associated with the hardware or software component is inherited from the function with the highest ASIL



A HW/SW element inherits the ASIL from the highest ASIL function running on it

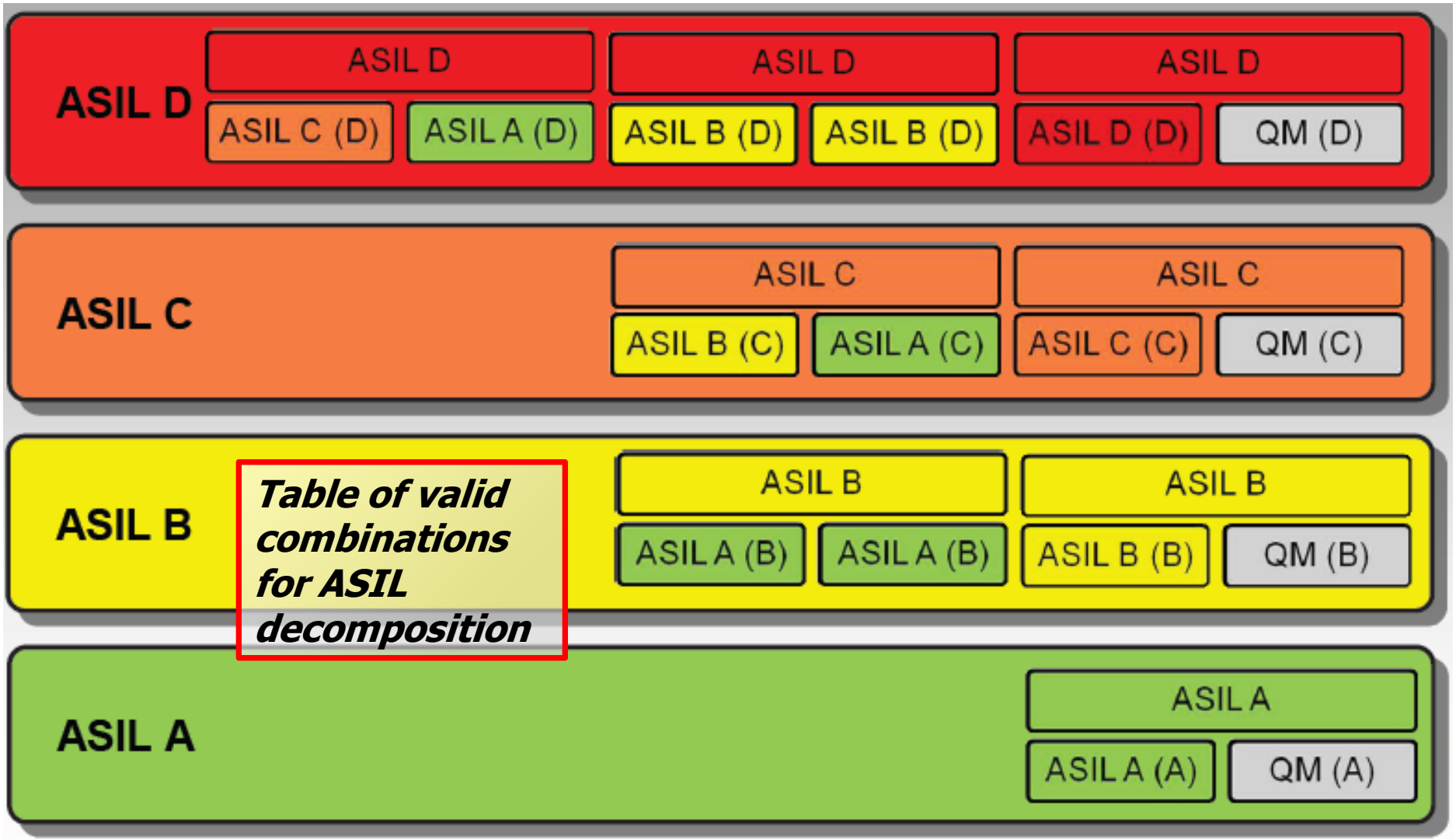
Lowering the ASIL

- Under certain circumstances, the ASIL can be lowered through the technique of **ASIL Decomposition**
- The concept already existed in IEC 61508 – it is not entirely new!
- This can be advantageous – for example, with respect to production costs
  - It usually costs less (labor, time, tools) to develop according to a lower ASIL
- But there are strict underlying concepts and rules that must be respected



- Cost savings
- Time savings
- …

Valid Combinations



ASIL Decomposition Basics

- An element implemented to address a given safety goal, with a given ASIL may be decomposed into **two independent** elements, with possibly lower ASIL
  - Each must address the **same safety goal**
  - And each must take on the **same safe state**
- ASIL Decomposition can be used in the following phases
  - Functional safety concept
  - System design
  - Hardware design
  - Software design
- ASIL decomposition is a **qualitative** concept, more addressing systematic issues (architecture) than random errors (hardware reliability)
  - It can be a way of making architectures more robust
  - Similar to 61508 fault-tolerant architecture concepts

Redundancy?

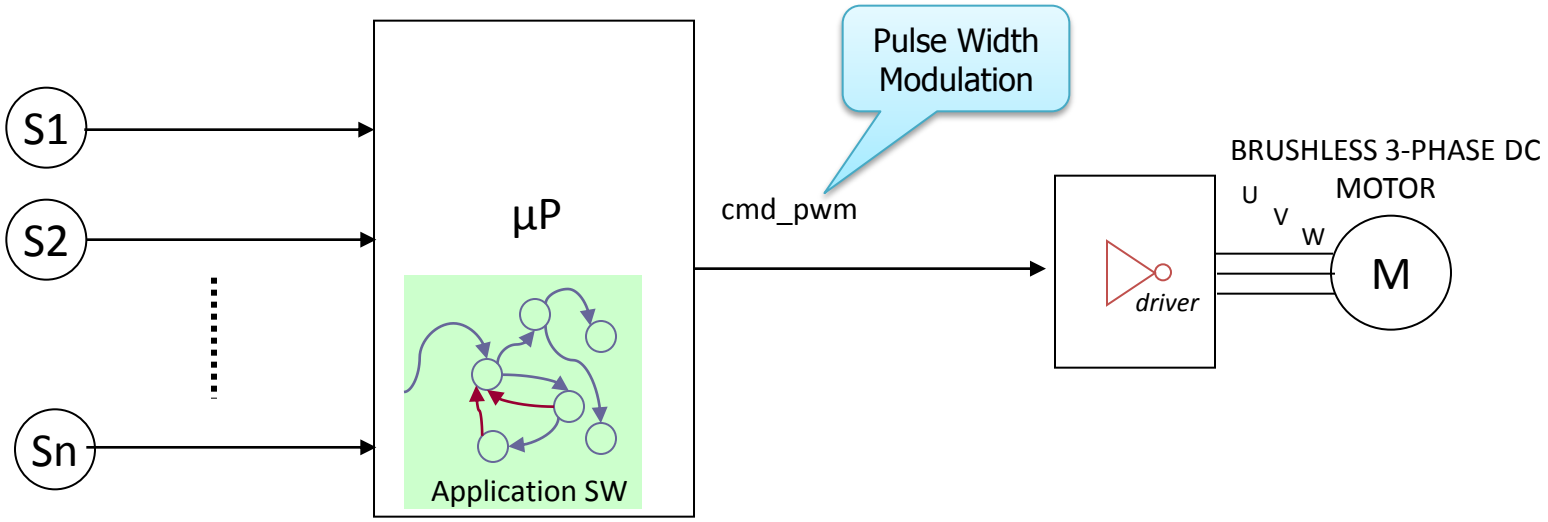
- Is ASIL decomposition a way of introducing redundancy?
  - "yes and no"
- Remember that (in general) there is actually very little redundancy in automotive systems
  - Only one gas pump, only one battery, …
  - Costs! This is well accepted
- ASIL Decomposition introduces **functional redundancy**
  - Two independent architectural elements work toward the same (redundant) **safety goal**
- These independent architectural elements are nearly always diverse
  - **Heterogeneous redundancy through architectural design elements**
  - This is not the homogeneous hardware redundancy we typically think about in 61508

Note that, according to the standard, an element could be either a HW or SW component

Industrial Scenario

Problem Description

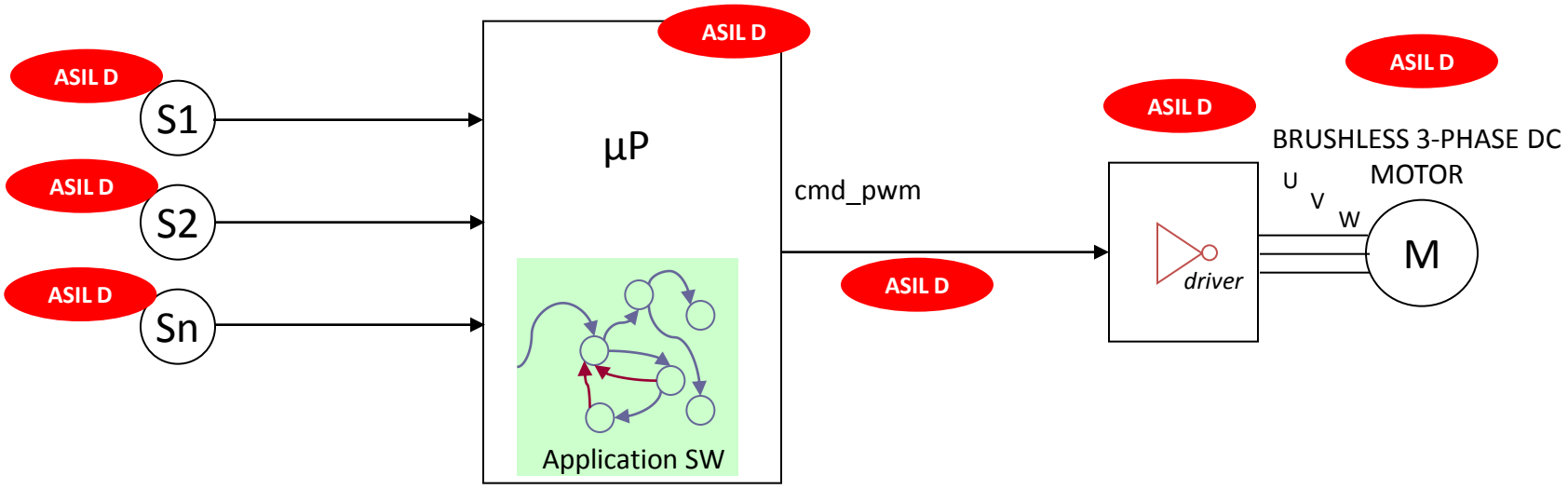
- Consider a function **F** which, upon input from a combination of sensors **S1**, **S2**, … **Sn** issues an activation command to actuator **M** ("Motor")
  - Suppose that the Safe State for F is "**M deactivated**"
  - Suppose that Hazard and Risk Analysis has determined **ASIL D** for the function F



- Suppose that we have identified the following safety goal: "**Avoid the undesired activation of M**"
  - Whereby "undesired" means "as a result of an incorrect combination of sensors S1, S2, … Sn"

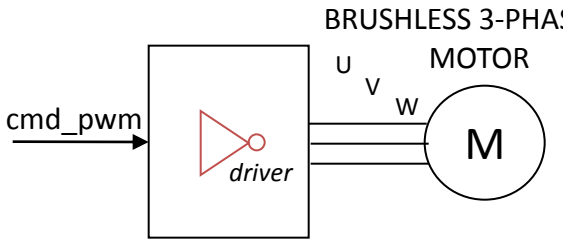
ASIL Allocation

- Suppose further that sensor S1, S2, … Sn measures some different value
  - That is, the sensors are independent of each other and non-redundant
- Furthermore, in this scenario we assume that each of these sensors, if faulty, could **by itself** cause the safety goal to be violated
  - The ASIL theory of the standard says that therefore each of the sensors must also inherit the ASIL D allocated to the function F



First Analyses

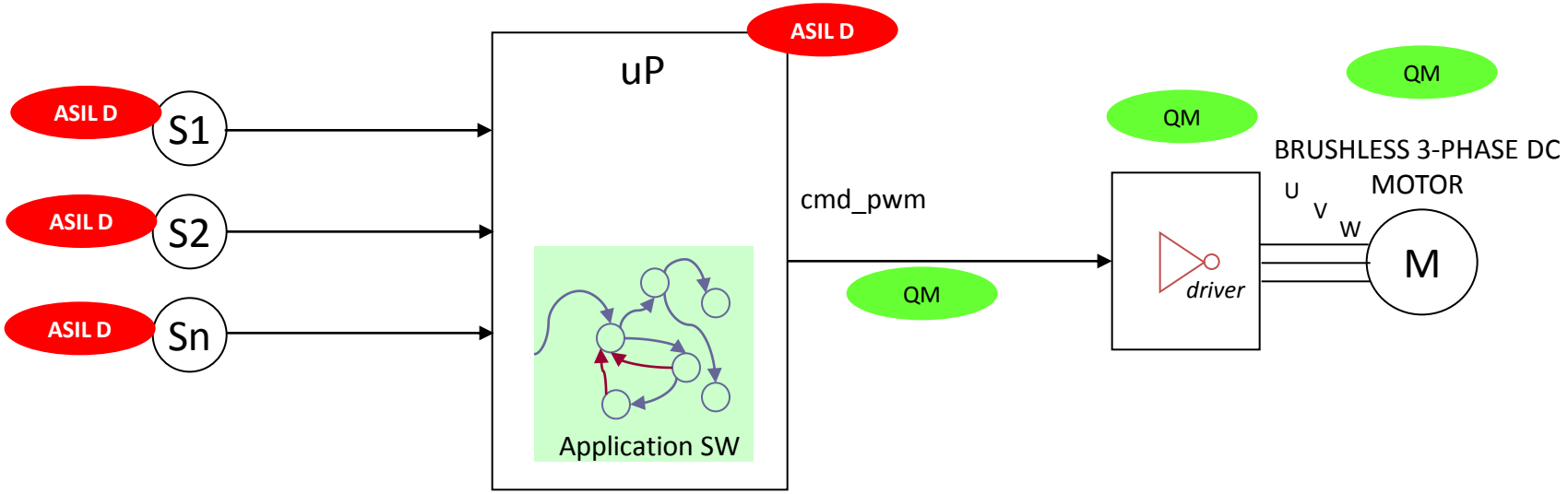
- At this point, we begin to analyze our architecture, reasoning about which elements of the architecture in reality have the capability of violating the safety goal
  - This may exploit specific knowledge of the technology involved
- In this example, we know from the theory of the control of brushless 3-phase DC motors that the three phases need signals that are precisely defined in time
  - Therefore some of the components (e.g. the driver and its associated command channel), in case of failure couldn't possibly produce the precise signals necessary to erroneously activate M
  - And therefore they are incapable *by themselves* of violating the safety goal



Brushless 3-phase DC motor technology needs precise input signals – impossible for a malfunctioning driver to produce

Lowering ASIL

- As a result of this analysis, we are justified in lowering the ASIL of the driver, motor, and command channel to QM
  - Note that this depends entirely on the technology; if the motor were based on continuous technology, it would not have been possible to lower the ASIL to QM



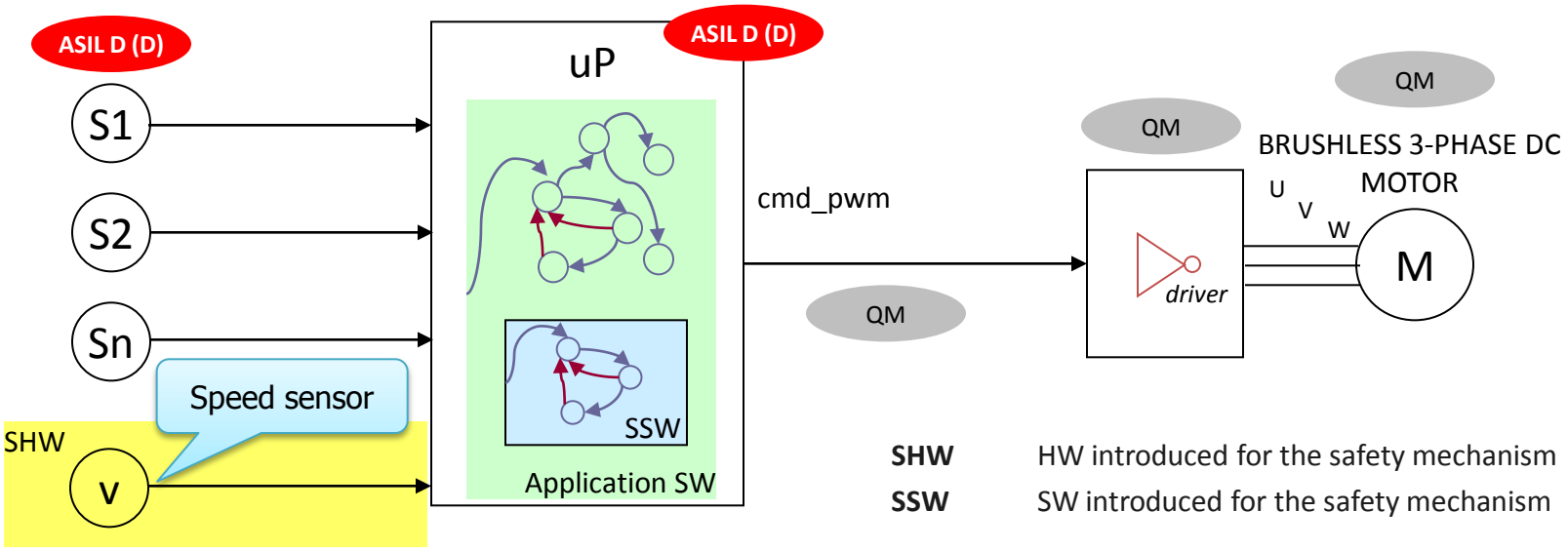
Lesson Learned: Sometimes through examining the technology and its potential for safety goal violation, we can influence ASIL allocation. Sometimes a project might even change its technologies after such analyses.

Exploiting the H&R Analysis

- We now look for ways to improve the safety architecture, by exploiting the results of the **hazard and risk (H&R) analysis**
- In its current form, the architecture considers only "erroneous sensor inputs", regardless of the operational scenario
  - But suppose that the H&R analysis distinguished operational scenarios, such as the speed of the vehicle? (this is typical)
- Suppose that the H&R analysis yielded the result that undesired activation of M was only dangerous at a speed greater than some threshold?
  - (As another example, consider undesired deployment of an airbag – its effect depends on the velocity of the vehicle)
  - Other typical examples of operational scenarios might be "driver-side door open" or "temperature of engine greater than some threshold"
- The results of this H&R analysis yield information that we can exploit to introduce a **safety mechanism** in our architecture

Introducing a Safety Mechanism

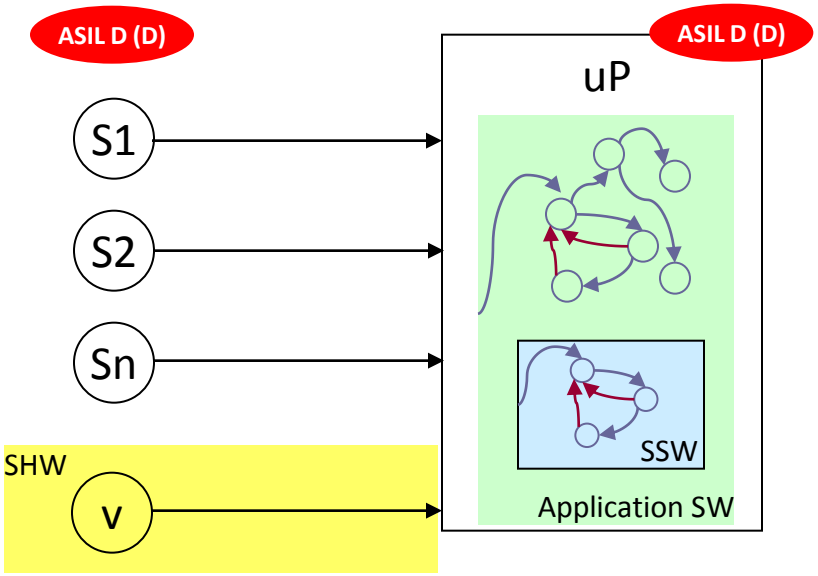
- We now introduce a safety mechanism: "The function M must not be activated when vehicle speed is greater than a specified threshold"
  - This is effectively introducing a kind of "AND" gate to lower the probability of M being erroneously activated
  - The undesired activation of M can only occur now if F fails *and* v>threshold



Lesson Learned: By careful examination of the Hazard and Risk Analysis and sufficiently detailed analysis of operational scenarios, we can discover possibilities for the introduction of safety mechanisms in the architecture.

Safety Mechanism ASIL?

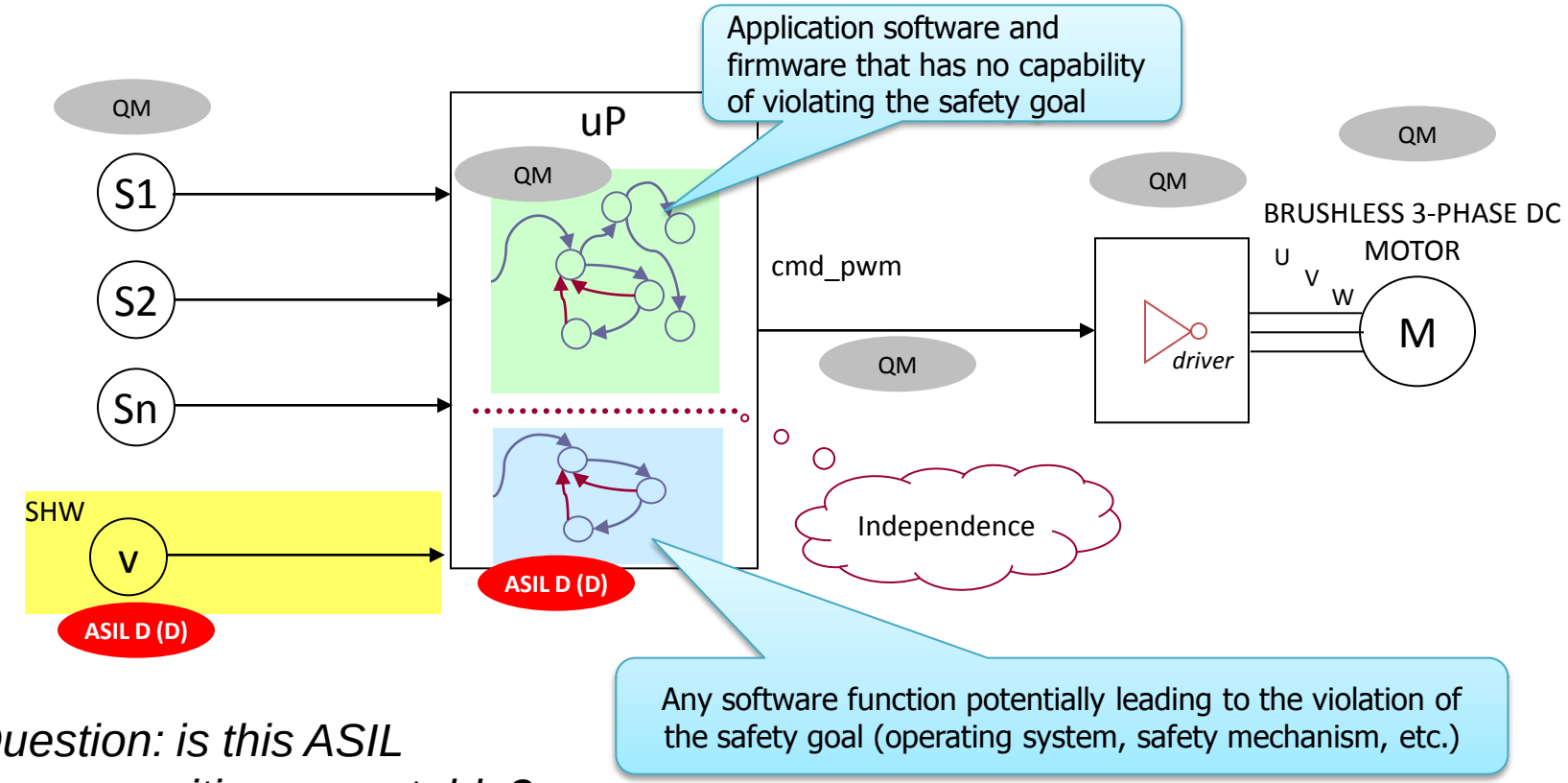
- Note that we have actually changed the architecture now
  - We have introduced a new sensor V
  - We have introduced new software



- But have we changed the ASIL allocation?
  - The answer is "No"
  - The mere addition of a safety mechanism *by itself* does not change the ASIL allocation

SW ASIL Decomposition?

We find ASIL D for our system software to be too high, but we don't want to introduce hardware redundancy into the control logic. So we decide to apply **ASIL Decomposition at the software level**

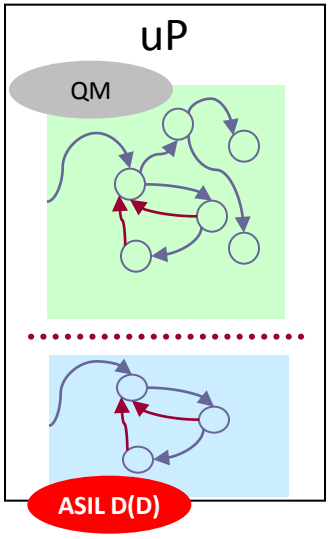


Question: is this ASIL Decomposition acceptable?

Independence?

- Answer: the proposed software-level ASIL decomposition is acceptable **only if the criteria of independence are satisfied**
  - This includes not only examining the software but also the hardware

- Furthermore: What about the hardware metrics? Do they become ASIL QM, or ASIL D? Or some combination based on percentages?
  - Answer: hardware metrics are not affected, so they are still ASIL D!

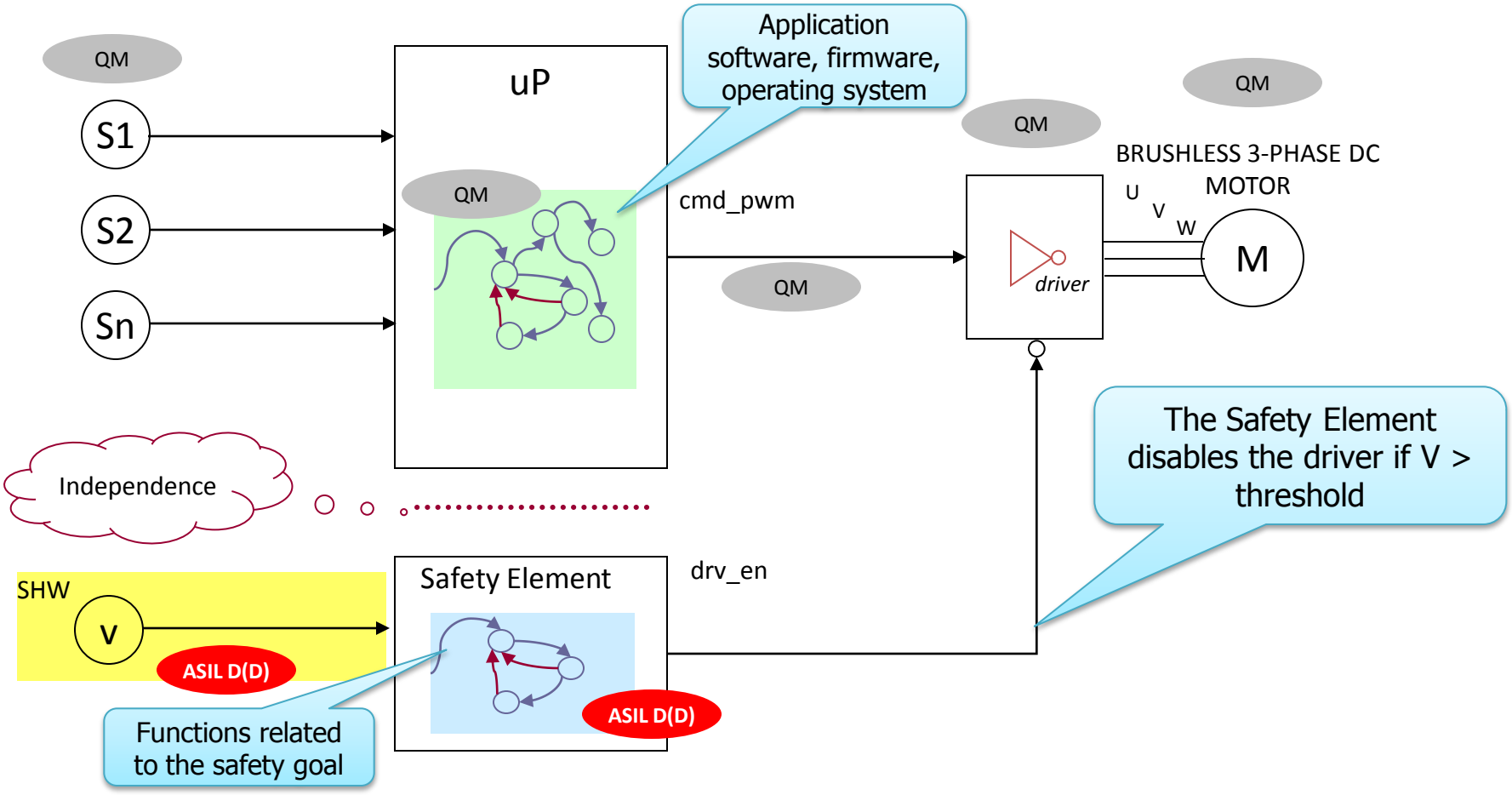


- There are several issues
  - What about sharing of software resources like the underlying operating system?
  - Sharing of firmware?
  - What about sharing of hardware resources like memory, ALU, etc.?

Lessons Learned: Software level ASIL decomposition involves a careful analysis of *both* software and hardware independence. Hardware metrics are not affected by ASIL decomposition at the software level.

HW-Level Decomposition

- Our analysis of software level decomposition reveals that there are too many issues, and we decide to do a HW-level decomposition



The Safety Element

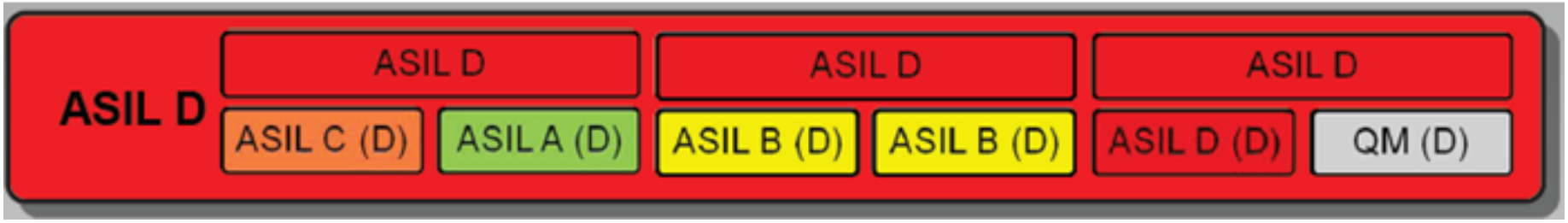
- What exactly is the **Safety Element** in terms of hardware?
  - This doesn't have to be a full microprocessor
  - It might be a **programmable gate array**, essentially just a state machine, programmed only one time, with no operating system
  - They cost only one-tenth of a full micro, and are very reliable, with their own clock and power supply, easy to manage
- There is no embedded logic – so there is no software
  - This has consequences for the 26262 safety process
  - You don't need Part 6 at all any more, only part 5
- That is why it is only called a safety element
  - It depends on the safety function to be carried out

Lesson Learned: Hardware level ASIL decomposition involves deep knowledge of the characteristics of the available hardware, so that independence, functionality, and costs are all correctly balanced.

Alternative Decompositions?

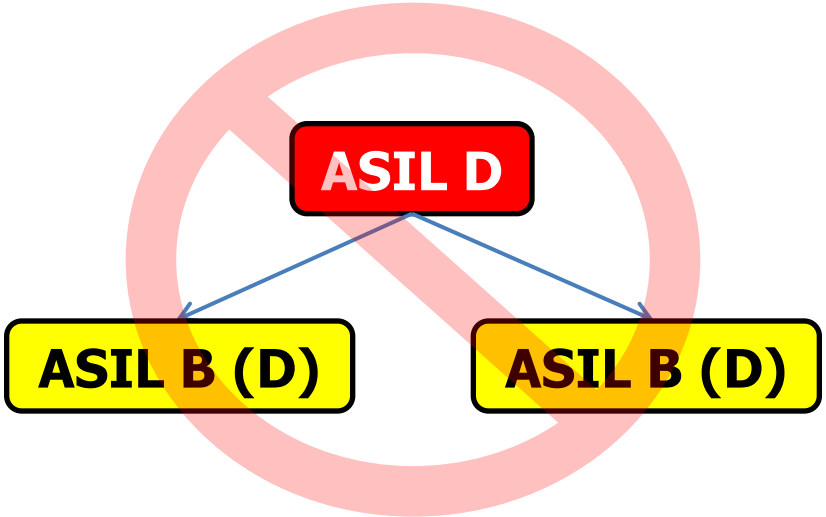
The HW level decomposition we just presented can be favourable when we have many non-critical functions that can be confined to the main micro, and a limited number of safety critical functions sharing the same safe state (driver deactivation).

- What other possibilities exist for decomposing the original ASIL (D) over the two elements?
- Two more possibilities:
  1. ASIL B (assigned to µP) + ASIL B (assigned to safety element)
  2. ASIL C (assigned to µP) + ASIL A (assigned to safety element)



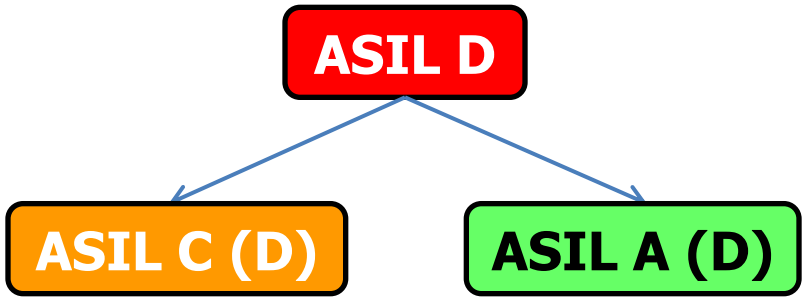
Alternative 1 Decomposition

- The first alternative decomposition represents having two essentially equivalent processors with redundant functionality
  - But it is a prohibitively expensive solution already from the hardware side (a processor is much more costly than e.g. a simple sensor)
  - Furthermore, the software would have to be developed with "diversity" techniques that are also known to be prohibitively expensive



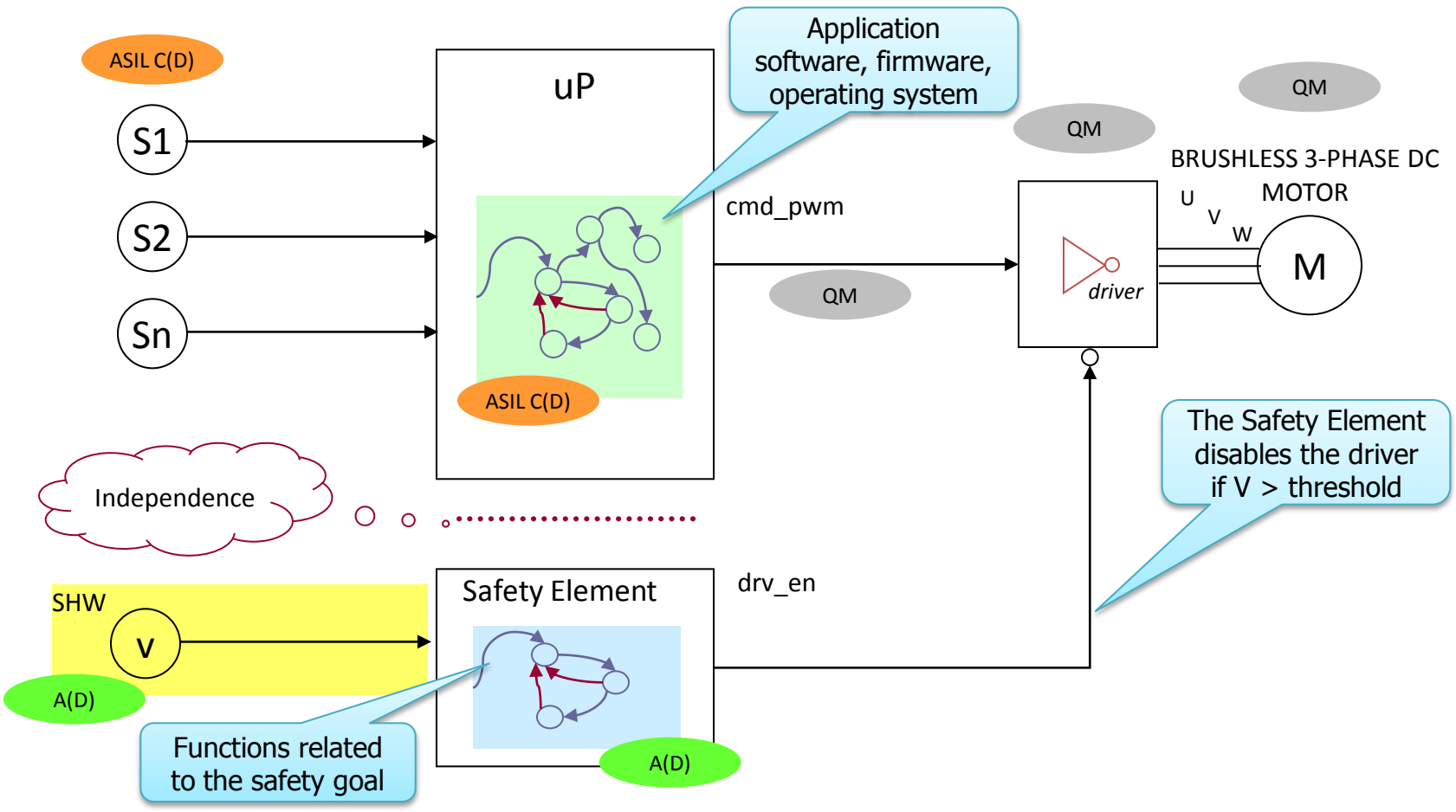
Alternative 2 Decomposition

- The final alternative represents an asymmetric layout once again
  - the software assigned to the main microprocessor implements the overall functions of the Controller
  - The safety element is simple, inexpensive, and reliable
- Why not the reverse? Why not processor ASIL A and safety element ASIL C – which would usually be the intuitive choice?
  - Because in some cases the safety element could be a **legacy element** from previous designs and its use was prescribed in this particular project
  - It may be too simple to be able to handle more complex safety functions



Lesson Learned: The “obvious” decomposition is not always possible due to project-specific constraints such as legacy components. Case-by-case analysis is essential.

Alternative 2 Design



Top Ten ASIL Misconceptions (1)

10. The ASIL deals with hardware development only
    - ASIL has an impact on hardware, software, and supporting processes
9. A hardware element can be designed as ASIL X for any system
    - A hardware element can be designed to satisfy up to ASIL X safety requirements in a given system
8. ASIL decomposition is a form of hardware redundancy
    - Yes and no: ASIL decomposition implies *functional* redundancy but also with diversity, independence and freedom from interference
7. ASIL decomposition is used to reduce the HW metrics targets
    - NO! after the ASIL decomposition the same targets of initial safety goal (before decomposition) apply to the decomposed HW/SW elements
6. ASIL decomposition is primarily about random failures
    - This was true with IEC61508, but doesn't apply with ISO 26262. In reality, it is more about dealing with *systematic* (e.g. architectural) issues

Top Ten ASIL Misconceptions (2)

5. ASIL decomposition is required by the 26262 standard
   - In reality, it is not a required step. It can be seen as an **opportunity** to allocate homogeneously functions with different safety criticality during the SW partitioning onto the HW elements.
4. Software level ASIL decomposition is simpler and cheaper than hardware level decomposition
   - In reality, software level decomposition is often more difficult and more expensive than hardware level decomposition, due to heavy requirements for diversity and independence
3. ASIL decomposition is the only way to lower the ASIL of an element
   - In reality, the ASIL of an element may also sometimes be directly lowered by an informed and careful analysis of the technologies and the architecture involved. ***Many are still unaware of this***.

Top Ten ASIL Misconceptions (3)

2. ASIL decomposition is always possible
   - In reality, the implementation of multiple functions with many different ASILS (as in a modern microcontroller) might make it essentially impossible to effect an ASIL decomposition in certain cases.

And the Number 1 misconception about ASIL decomposition is …

1. ASIL decomposition is always desirable
   - In reality, there is always a cost-benefit trade-off, and often after careful analysis an ASIL decomposition will reveal itself as undesirable.

Conclusions

- ASIL decomposition can be a way to lower required ASIL of system elements, both hardware and software
- It requires architectural decisions to be taken
  - They may affect hardware, software, or both
  - Sometimes these decisions are extremely difficult to evaluate
- Sometimes the problem is demonstrating independence, and too much work to justify the cost
- There are many factors to consider!