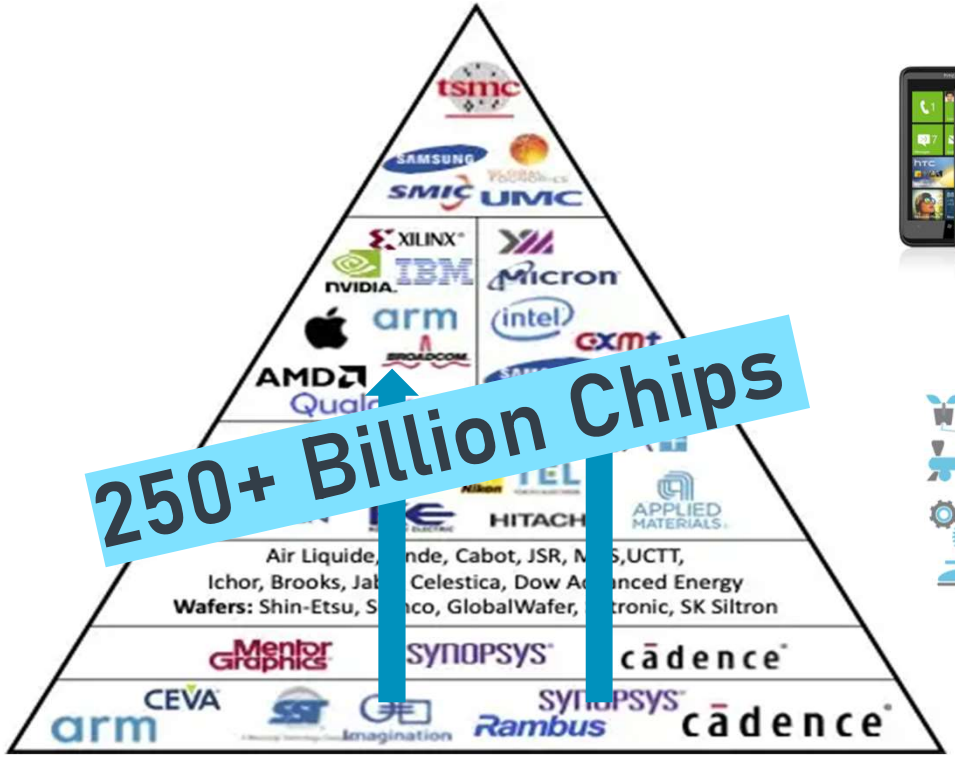
Safety and Security Challenges in Hardware IP Development

Verification Futures Conference 2023 US

Vivek Vedula

14-Sep-2023

Semiconductor Ecosystem: Arm's Role



Source: Semiwiki.org

Key Pillars of Dependable Computing

- Functional correctness
  - "Does the system behave correctly under normal operating conditions?"
  - "For every possible input, does the output satisfy the specification?"
  - Pre-silicon: Simulation, Emulation, Formal verification
  - Post-silicon: DFT, At-speed functional testing, Board-level testing

- Safety and Security
  - "Does the functionally-correct system behave predictably under *adverse operating conditions*?"
  - Adverse conditions due to Murphy's law: Safety
    - *Anything that can go wrong will go wrong, and at the worst possible time.*
  - Adverse conditions due to Devil's law: Security
    - *Anything that can go wrong shall be made to go wrong, and at every possible time.*

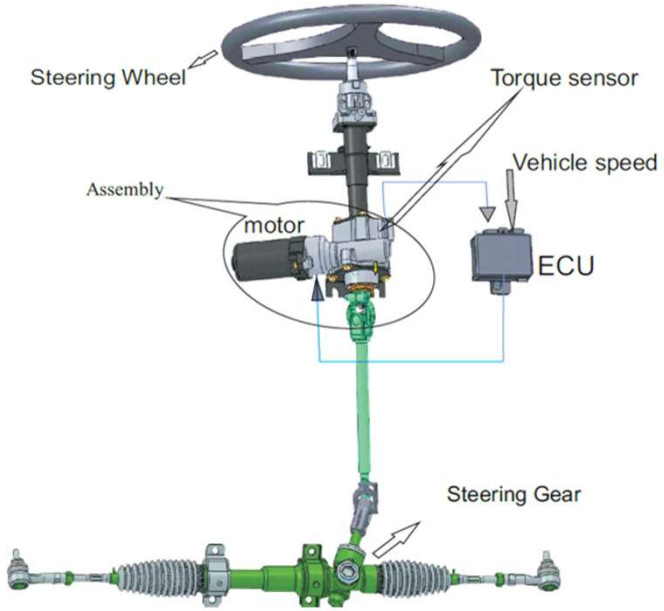
arm

Safety

Example: Electrical power steering

- An example of a control system which must demonstrate functional safety
  - Must continue to function or at least behave predictably in event of a fault
  - By predictable behaviour we mean it must gracefully go into a known “safe” state

- Functionally safe systems aim at preventing hazardous behaviour in event of a fault

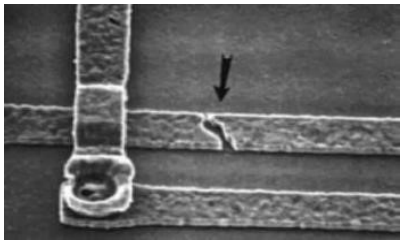


Sources of Failures in E/E Systems

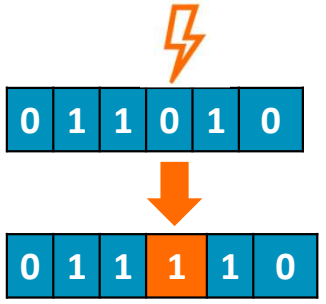
- Systematic Failures: Design Faults, wrong specs, software errors
  - Requires safety analysis on the design to identify causes

- Random Failures: “Acts of God”

  - Permanent faults/Hard Errors
    - Ex: Physical failure of silicon over time



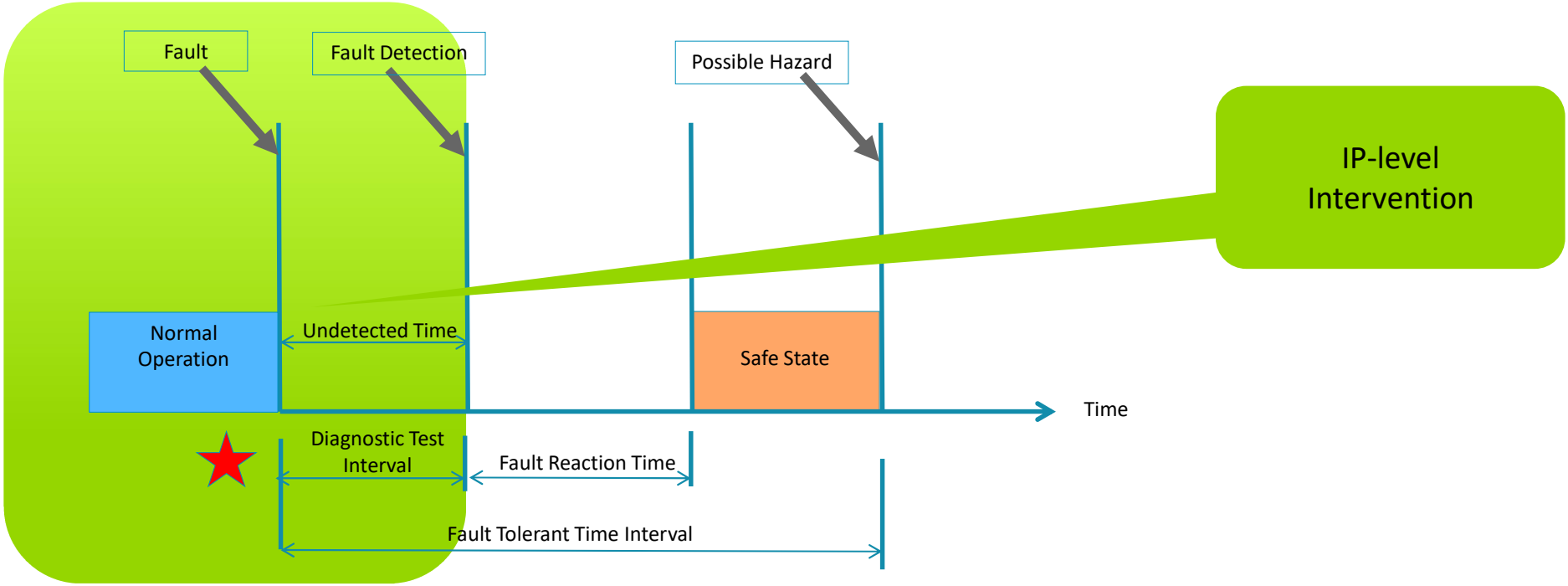
  - Transient/Soft Errors
    - Ex: Atomic particle causing a bit-flip



arm

Fault Tolerant Time Interval -FTTI

FTTI: Time-span in which a fault or faults can be present in a system before the hazardous event
Fault Reaction Time: Time span from detection of fault to reach the safe state



arm

Safety Product Development Lifecycle at System-level

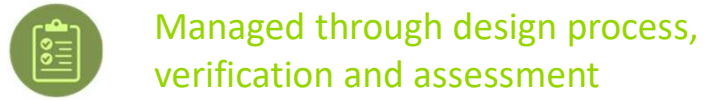
- Identify and classify hazardous events caused by malfunctioning behavior
- Derive safety goals for each hazardous event
- Generate FSRs to meet each safety goals
- Generate TSRs to meet each FSR
- Identify requirements for HW and SW

Hardware Safety Requirements' Mitigation

Systematic Failures

Guidance on:

- Methods for design and testing
- FuSa management and culture



Managed through design process, verification and assessment

Random Failures

Quantitative targets on:

- Probability of failure of a safety function
- Percentage of faults to be detected

Managed by including features for fault detection and control

What makes it challenging for IP-Level design?

- Little or no context of how the IP is going to be used
- Unknown operating conditions
  - False identification of failures (random and systematic)
- HARA is highly subjective and unreliable
  - Resultant safety goals and requirements are questionable
- IPs are expected to be highly configurable for usage in multiple products
  - Risk of over-engineering the safety mitigations (such as dual lock-step)
- Long duration of products in the field (15+ years in Automotive)
  - Higher chance of random failures over the life of the product

10
arm

Challenge #1: Assumptions of Use

- Identification of potential use-cases
- Operating conditions where the system will be deployed
- End-of-life for each use-case
- Process technologies and their failure rates
- Safety functions that will be enabled
- Probability of failure of safety functions

Challenge #2: Implementation of Safety Requirements

- Safety often impacted by interacting entities
- Ensuring reasonable design trade-offs with safety functions
- Gate-level fault analysis requires functionality to be implemented first
- Detection and control need to happen well within FTTI of the product

Challenge #3: Verification of the Safety Measures

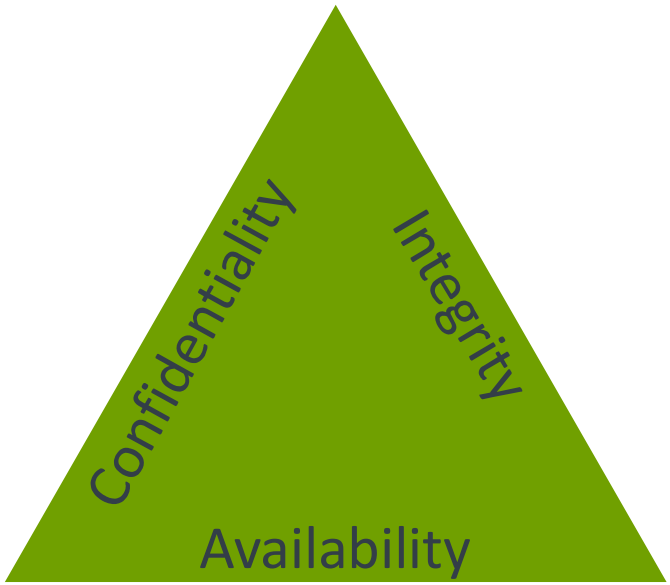
- Traditional fault models are unreliable for modeling **all** safety-related failures
- Safety verification at the IP-level does not directly translate to system-level
- Formal-based tools struggle with gate-level analysis
- Undetected faults require tedious analysis to ensure their impact on Safety

Basic Security Framework

- Security is the state of being “protected” against unauthorized/illegal access to the “assets” through malicious “attacks”
- Asset – Something of value
  - Examples: data, code, encryption keys.
- Attack (*aka* Exploit) – A deliberate action performed to compromise an asset
  - Examples: WannaCry ransomware attack

arm

Classification of Attacks: CIA Traid



- **Confidentiality** is the restriction of access to the asset for approved users only
- **Integrity** is the maintenance of consistency, accuracy and trustworthiness of the asset
- **Availability** is providing access to the asset when required

- Any action that could compromise any of these properties of an asset is referred to as a *Threat*

arm

Vulnerabilities Vs Weaknesses

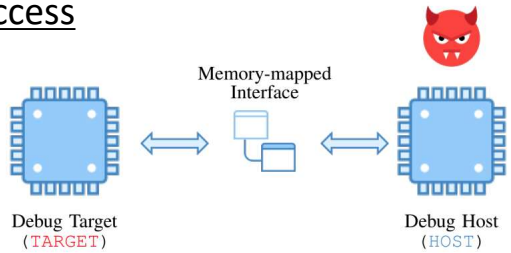
- **Vulnerability**: A flaw or an error in the system that can be used to carry out a threat
  - Ex #1: Bypass of normal privilege checks needed to access data belonging to the OS (Meltdown)
  - Ex #2: Leakage of data through observable properties of speculative execution and branch prediction

- **Weakness**: A design flaw that could potentially contribute to the introduction of a vulnerability
  - Not all weaknesses lead to vulnerabilities
  - Ex #1: Processor optimization that removes/modifies security mechanisms
  - Ex #2: Information Exposure through Microarchitectural State after Transient Execution

Chip-level Vulnerabilities: Examples

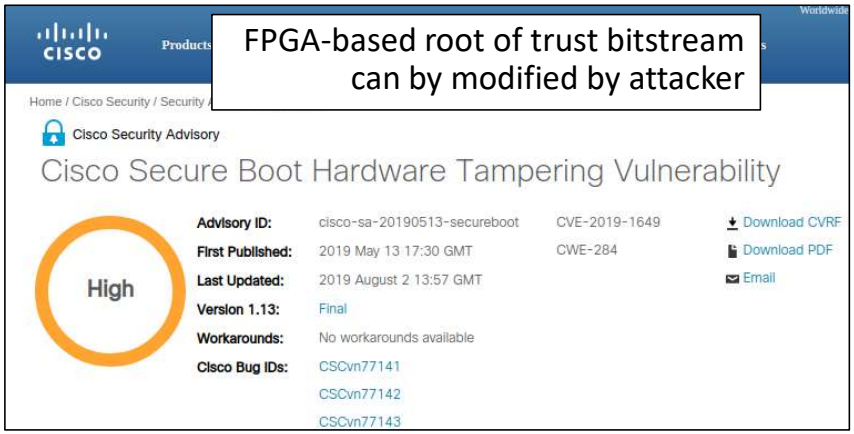
Test and Debug Infrastructure

Nailgun Attack: Privilege escalation via Arm's multi-core debug infrastructure requiring no physical access

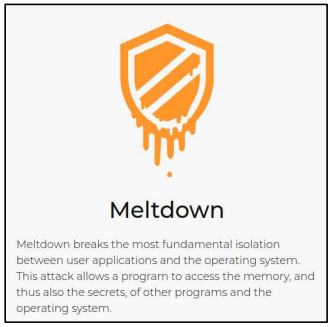


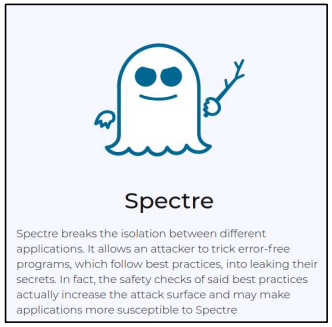
Root-of-Trust

FPGA-based root of trust bitstream can by modified by attacker



Micro-arch Efficiency



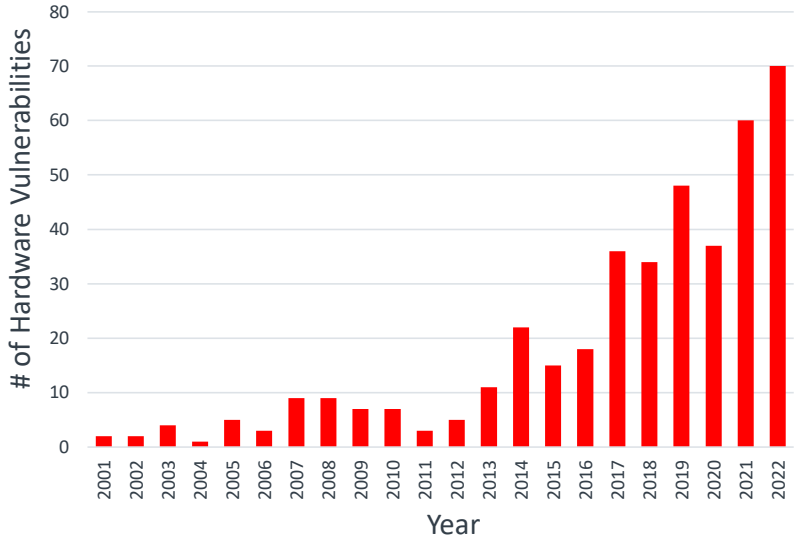


arm

Why Hardware?

MITRE records an increase in hardware vulnerabilities in the last 10 years

- Relied upon as the root-of-trust (Crypto, TEE etc)
  - Hardware attack ROI is high; breaks large investments in cybersecurity
- Aggressive Perf/Power optimizations
- Higher configurability (Ex: chicken bits) for performance, maintenance
  - Integration challenges
- Increased shift towards hardware-based security architectures



Source: NIST/MITRE 12/2020

Challenge #1: IP-level Scope

- IP-level security is foundational
- Out-of-context security requirements
- Specification cannot capture all possible malicious intent
- Risk analysis is highly subjective
- Unknown ROI for mitigations

What is attackable ?

arm

“Top 10” Hardware Security Weaknesses

CWE #	Description	Impact at IP-level
CWE-1189	Improper Isolation of Shared Resources on System-on-a-Chip (SoC)	Disable resource sharing between trusted/untrusted agents
CWE-1191	On-Chip Debug and Test Interface With Improper Access Control	Enable authentication of debug interface
CWE-1231	Improper Prevention of Lock Bit Modification	Ensure appropriate lock bit protection mechanism
CWE-1233	Security-Sensitive Hardware Controls with Missing Lock Bit Protection	Ensure appropriate lock bit protection mechanism
CWE-1240	Use of a Cryptographic Primitive with a Risky Implementation	Disallow usage of non-standard crypto primitives
CWE-1244	Internal Asset Exposed to Unsafe Debug Access Level or State	Allow only trusted agents to access security-sensitive assets over debug interface
CWE-1256	Improper Restriction of Software Interfaces to Hardware Features	Ensure access control for SW-controllable features such as frequency and voltage (Ex: DVFS)
CWE-1260	Improper Handling of Overlap Between Protected Memory Ranges	Ensure priority scheme for programmable memory protection regions
CWE-1272	Sensitive Information Uncleared Before Debug/Power State Transition	During state transitions, ensure clearing of data that is not required in the next state.
CWE-1274	Improper Access Control for Volatile Memory Containing Boot Code	Ensure that the volatile memory region is prevented from being modified by untrusted agents

Source: cwe.mitre.org

Why is security verification so difficult in HW?

- Attacking is fundamentally easier than protection
- Attackers need just **one** vulnerability
- Verification needs to ensure 100% of the vulnerabilities are covered
  - Impossible to know the list of all vulnerabilities
- Proving the non-existence of unknown/0-day vulnerabilities is impossible
- Attack-oriented bug hunting needs “out-of-the-box”, “malicious” mind-set

Challenge #2: Verification Tools and Methodology

Functional Requirements

- FSM must never transition to “Secure” state after reaching “Test” state
- CPU must not be interrupted if running secure code

Data-Protection Requirements

- Data in secure location must not be visible to the CPU, if it is not in secure mode (Confidentiality)
- Secure registers must never be written by a non-secure agent (Integrity)

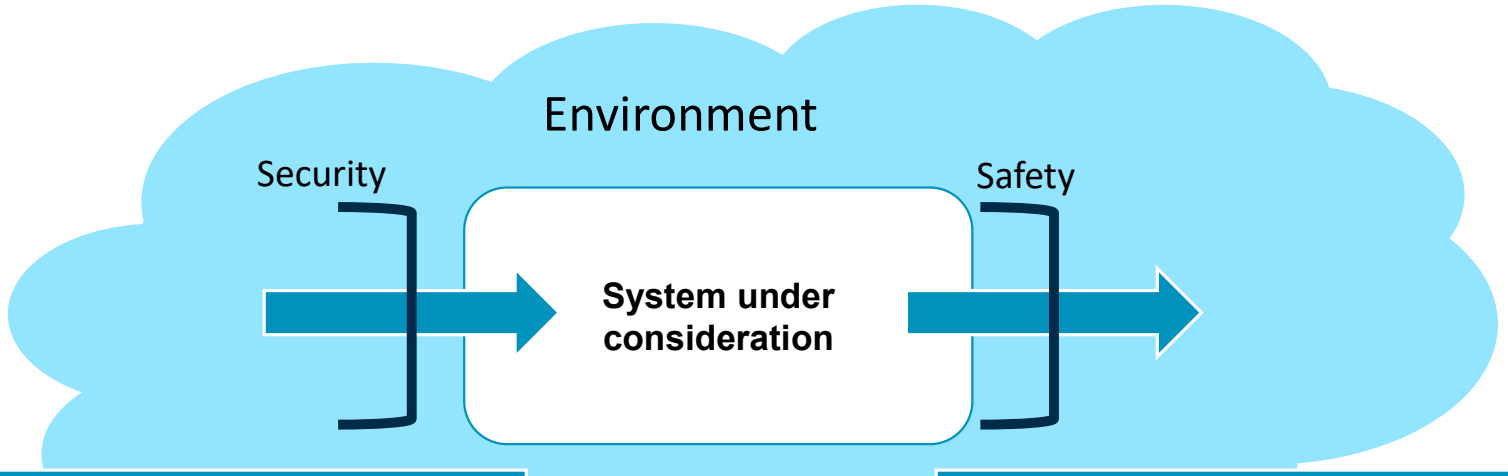
- Within scope of SVA-based specification
- Lends well to traditional solutions (Sim/Formal/Emulation)

- Difficult/impossible to specify using SVA
  - ➢ Do not scale effectively due to difficulty of identifying all prop paths in large designs
- ➢ Legacy EDA solutions are inadequate

How do you measure coverage?

arm

Safety vs. Security



Security
How to prevent the *environment* from affecting *your system*

Safety
How to prevent *your system* from affecting the *environment*

arm

arm

Security meets Safety

Safety and Security: Examples

Safety application

Patient-controlled drug delivery

Safety application

Braking system

Safety application

Avionics

Safety application

Nuclear Power

Security and Safety: Challenges with co-existence of reqs

- Safety Hazards can potentially cause violation of Security goals and vice versa
- A security attack may result in a new safety failure mode previously thought to be benign
  - Ex: a fault made observable through a malicious attack
- A safety failure mode may expose a security vulnerability that was previously considered secure
  - Ex: a fault on register access control may expose the register contents to an external interface
- Long safety-product deployment lifecycle makes threat-modeling difficult

arm

Potential Conflicts in Requirements

- Trade-off between performance overhead to process security functions vs hard real-time safety function to respond to hazards
  - Conflict between “fail-safe” state and “fail-secure” state
- Countermeasures for a security threat could exacerbate a safety hazard
- Security threat of malware trying to read out the private keys from the HSM or forcefully overwrite its firmware
  - When detected, the countermeasure (HSM) would shut down which is a violation to safety

Examples of Safety and Security Failures

		Impact	
Function	Potential Risk	Safety	Security
Authentication or check integrity of flash content	Authentication is ok while flash is corrupted	High	High
Authentication or check integrity of flash content	Authentication is not ok while flash is not corrupted	Low	low
Authorization of transmission to COM buses	Authorization ok to untrusted destination	Low	High
Decryption of secure data	Decrypted data is erroneous	Low	High
Reset	Reset asserted when not expected	High	Low
Detection of side-channel attack	Do not signal failure while attack	High	High
Detection of side-channel attack	Assert signal when there is no attack	High*	High

*Depending on how frequent this happen, it might cause denial of service.

arm

Summing it all up..

- Safety and Security are key pillars of dependable electronic/electronic systems
- IP from multiple providers used in these systems
- Assumptions-of-use could have a significant impact
- Customized solutions needed to address the challenges
- Both requirements within the same IP magnifies these challenges multi-fold

Questions/Comments?

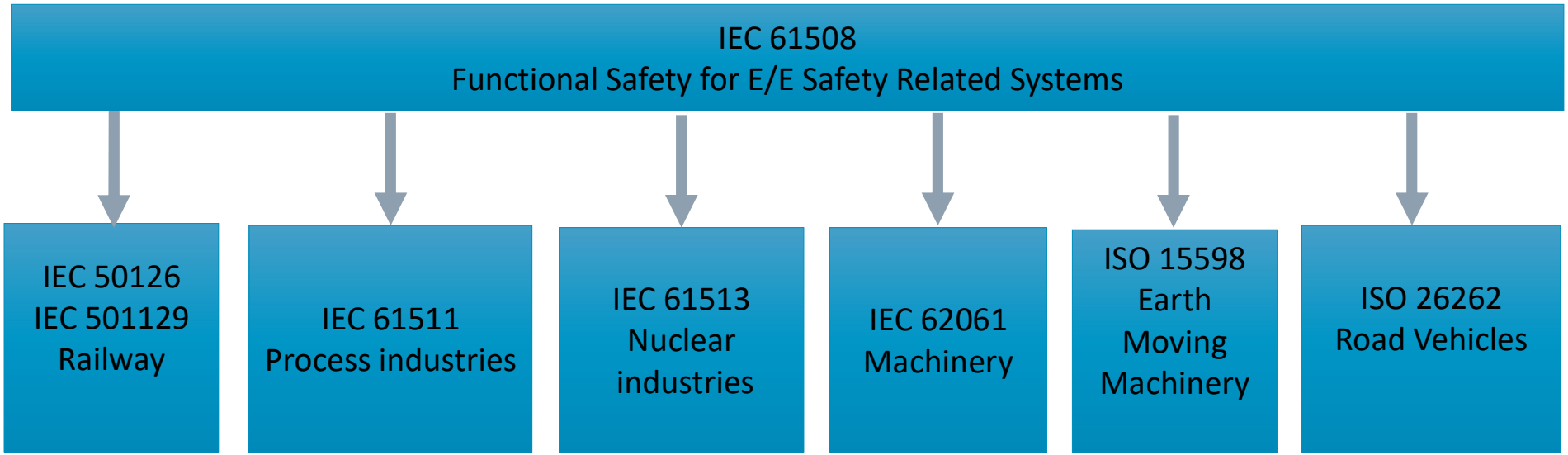
E-MAIL: VIVEK.VEDULA@ARM.COM

arm

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה

Functional Safety Standards



ISO International Organization of Standards

ISO 26262 is for Automotive Developed with OEM

Common Criteria

- International standard/guidelines for computer-security certification
- Enables objective evaluation of a product's compliance to defined set of requirements
- Two key components: Protection profiles and EALs

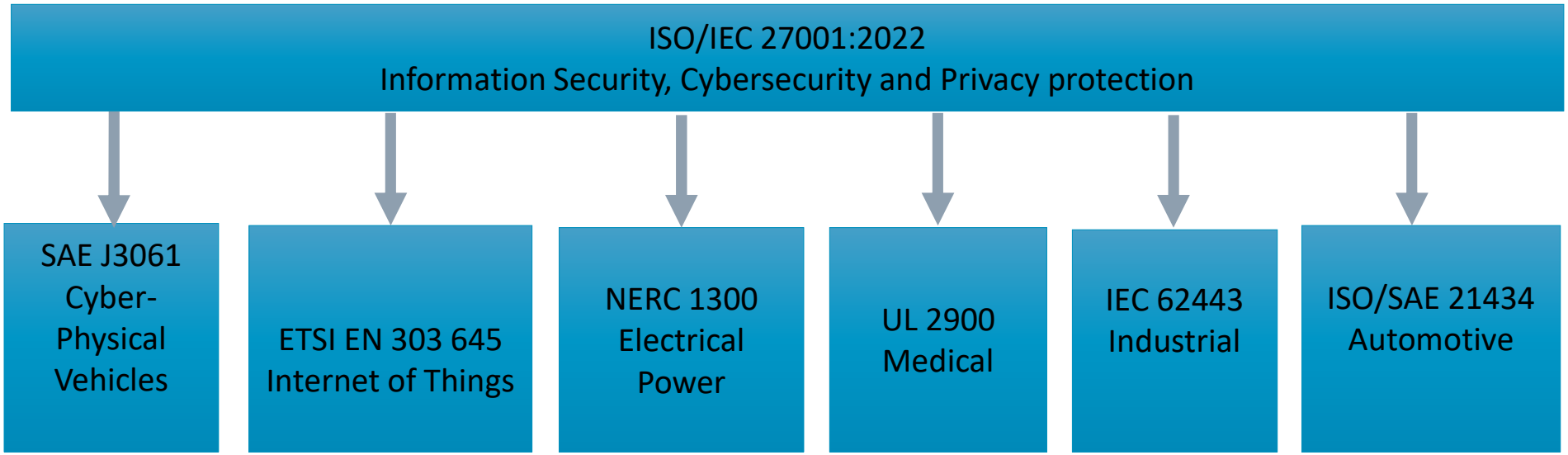
Protection Profiles

- Standard set of security requirements for a specific type of product
  - Derived through threat modeling
  - Identification of vulnerabilities
  - Risk-assessment

Evaluation Assurance Level

- Rating of the rigor/depth of evaluation of the mitigations (SARs)
  - Defines how thoroughly the product is evaluated/tested
  - Higher EAL =/=> Higher security

Security Standards



ISO International Organization of Standards

SAE Society of Automotive Engineers

IEC International Electrotechnical Commission

UL Underwriters Laboratories