2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Is your Hardware Dependable?

Practical Applications for Managing Security and Safety from Software to Silicon

Presented by:

DARPA, AMD, Arm Research, and Synopsys

accellera SYSTEMS INITIATIVE

Tutorial Speakers

- **Practical Strategies for Managing Security in Silicon**
  - Serge Leef: DARPA
- **SoC Functional Safety Overview and Use Cases, Standards, and Dependability Lifecycle**
  - Bala Chavali: AMD
- **Reliability Analysis method for Safety-critical CPU designs : PACE**
  - Reiley Jeyapaul and Balaji Venu: arm Research
- **Automated Solutions for Safety and Security**
  - Meirav Nitzan: Synopsys

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Practical Strategies for Managing Security in Silicon

Serge Leef, DARPA

accellera SYSTEMS INITIATIVE

Goal

Automate inclusion of scalable defense mechanisms into chip designs to enable security vs. economics optimization

Cost and Complexity of Attack Resistance Mechanisms

Source: shutterstock.com

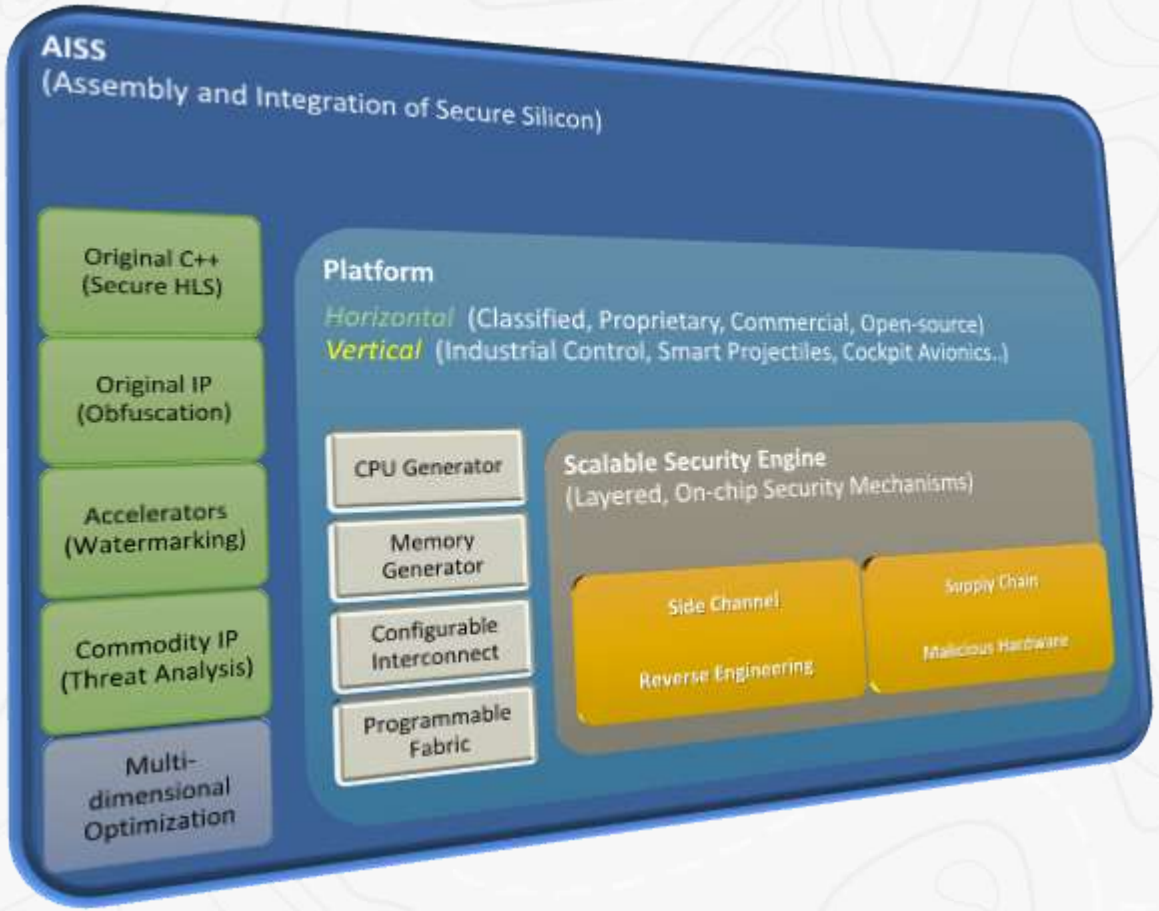
Novel Design Automation Flow with Embedded Security?

Security

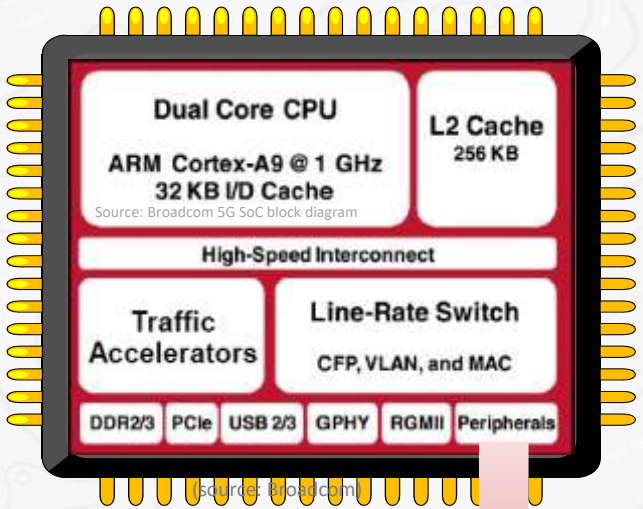
Incorporation of security into next generation of system chips, using platform-based design techniques & advances in high level synthesis

Automation

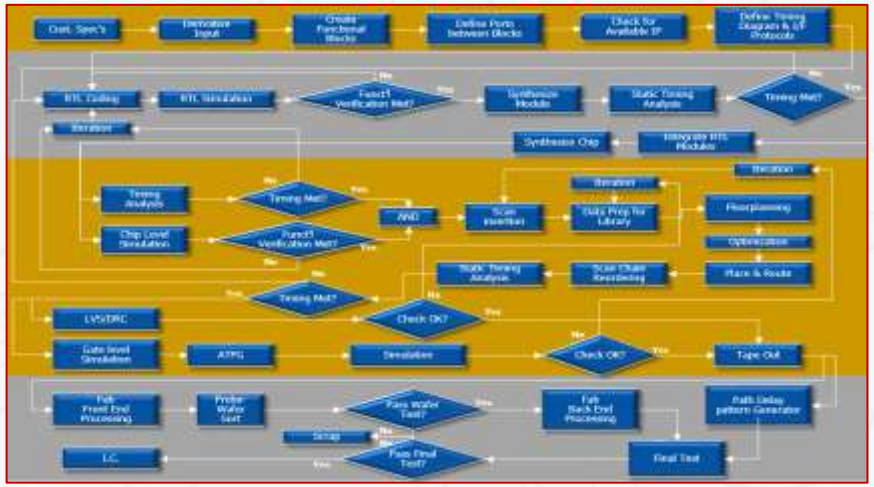
Need for automatic injection of scalable security creates an opportunity for tools & IP that enable semi automated and **automatic** approaches to assembly and integration that can substantially improve design productivity

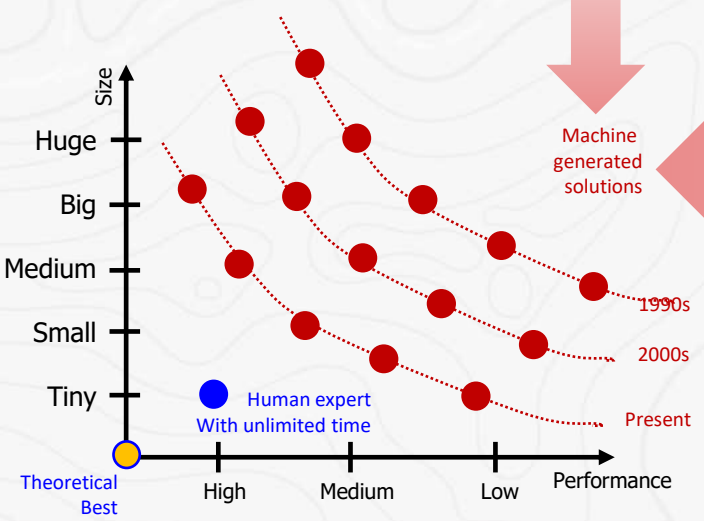


System on Chip (SoC) Design Process



Simplified View of SoC Design Process (source: Mentor)





Current Practice

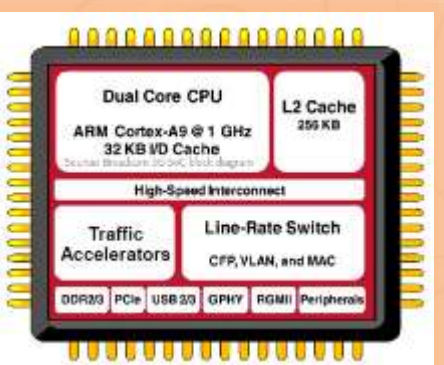
- Manual system integration
- Lengthy and complex simulation runs
- Block level synthesis & optimization

Limitations

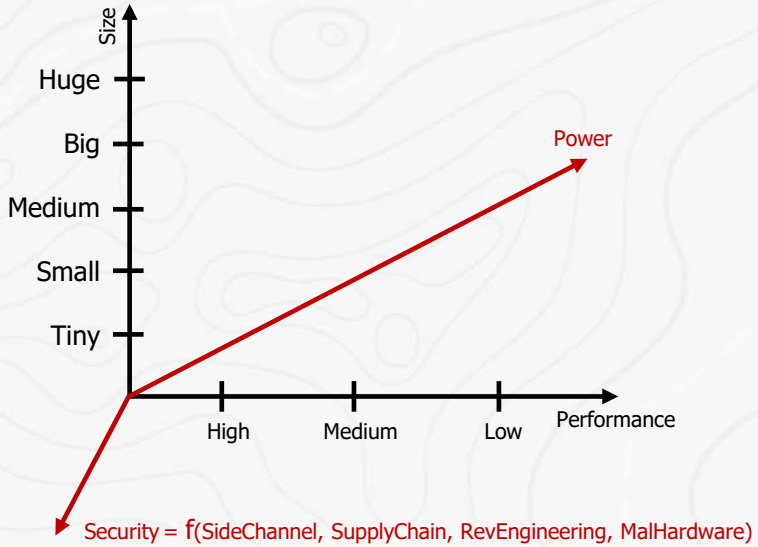
- $30M+ cost for low complexity SoC
- 9-12 month design cycles
- Many human introduced errors
- **Unpredictable power and no security**

Long Term EDA Dream: **System Synthesis**

System synthesis & optimization



1. Σ(a*`Performance`, b*`Size`)
2. Σ(a*`Performance`, b*`Size`, c*`Power`)
3. Σ(a*`Performance`, b*`Size`, c*`Power`, d*`Security`)
4. Σ(a*`Performance`, b*`Size`, c*`Power`, {d*`SideChannel`, e*`SupplyChain`, f*`RevEngineering`, g*`MalHardware`})

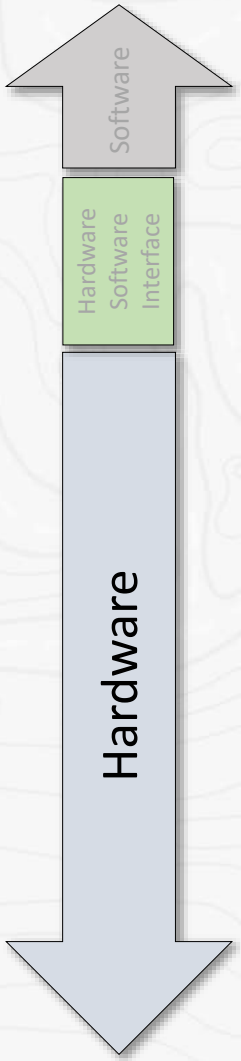


Key challenges:

- *Quantification of security*
- *Rapid estimation of attack resistance*
- *Multi-dimensional optimization*

(source: Broadcom)

Attack Surface Based Reference Model



Moving Target (I20)

- Substantial efforts are on-going in the software community

In Progress (SSITH)

- Alteration of system behavior based on software-accessible points of illicit entry that exist due to hardware design weaknesses or architectural flaws

AISS Focus Areas

- **Side Channel** – extraction of secrets through physical communication channels other than intended
  (assumption: attackers are able to "listen" to emissions)
- **Reverse Engineering** – extraction of algorithms from an illegally obtained design representation
  (assumption: attackers have access to design files)
- **Supply Chain** – Cloning, counterfeit, recycled or re-marked chips represented as genuine
  (assumption: attackers can manufacture perfect clones)
- **Malicious Hardware** – insertion of secretly triggered hidden disruptive functionality
  (assumption: attackers successfully inserted malicious function(s) into the design)

Security Strategies by Company type

Huge merchant semiconductor companies *(Intel, Broadcom, Qualcomm…)*

- See the critical need and have large expert teams to create custom solutions

Mid-size semiconductor and system companies *(NXP, Cisco, Nokia…)*

- Recognize problems but lack expertise and sufficient economic motivation

Defense contractors *(Honeywell, NG, Lockheed…)*

- Possess deep, but limited, expertise (craft) unevenly applied to specific chips

System integrators *(Ring, Fitbit, August…)*

- No interest due to time-to-market focus and lack of in-house competency

AISS Approach to On-Chip Security

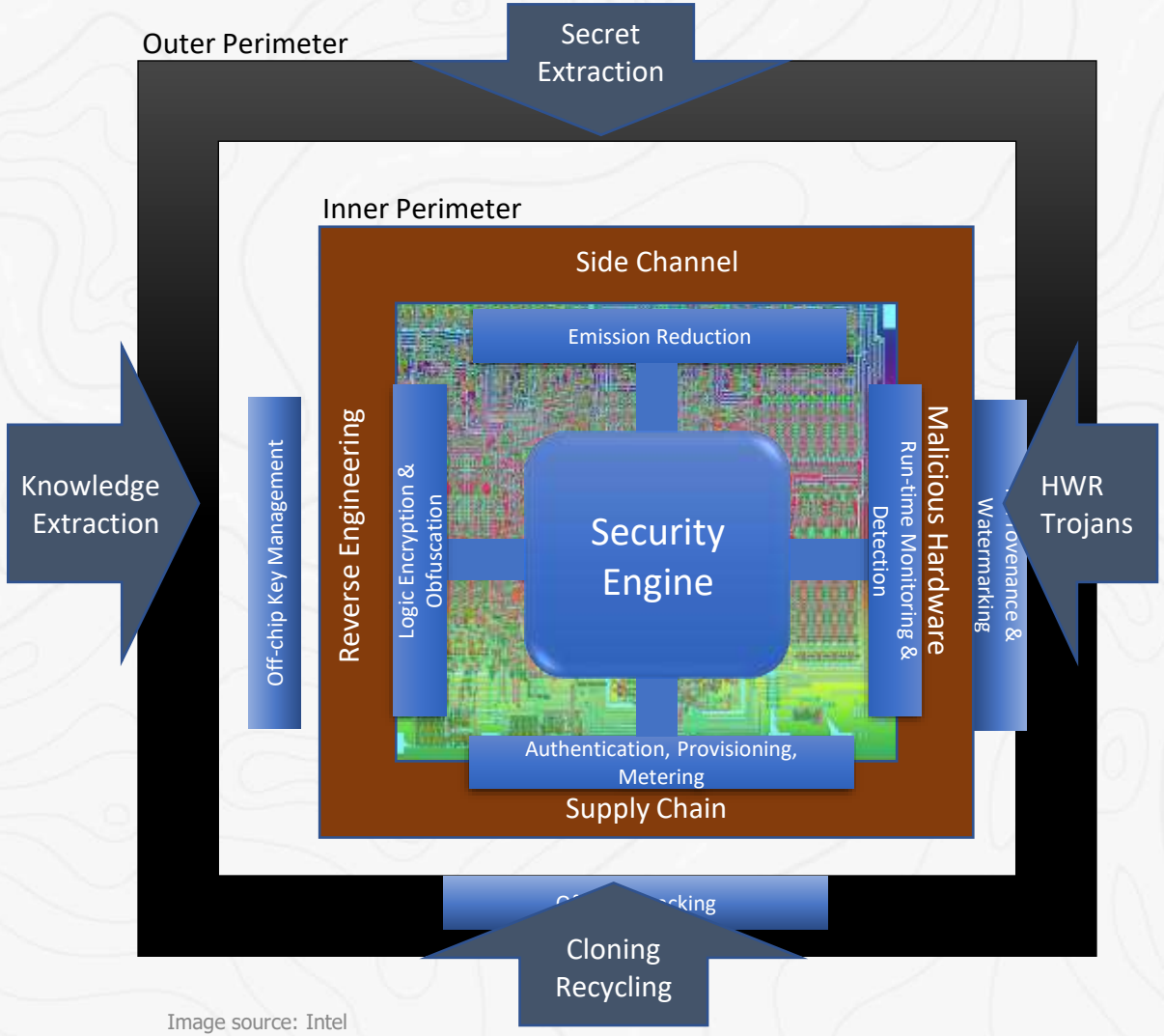
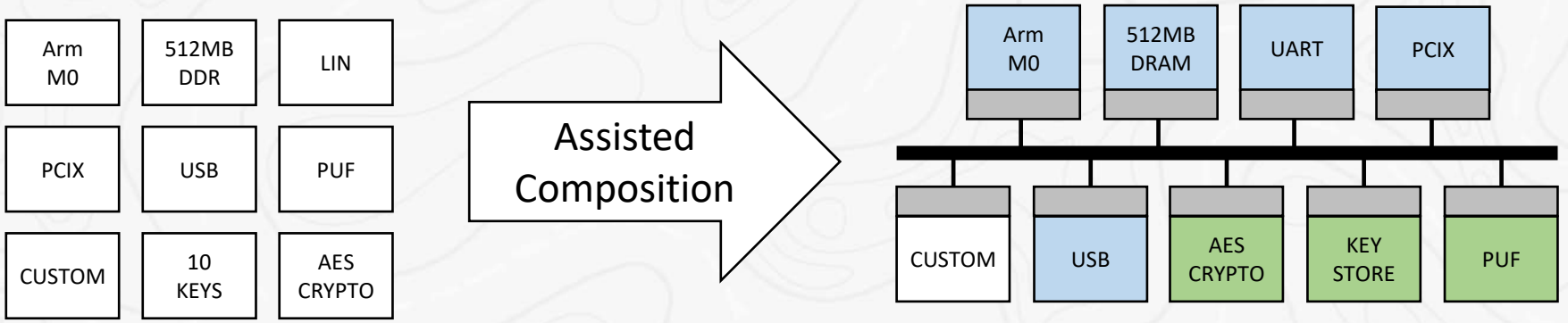


Image source: Intel

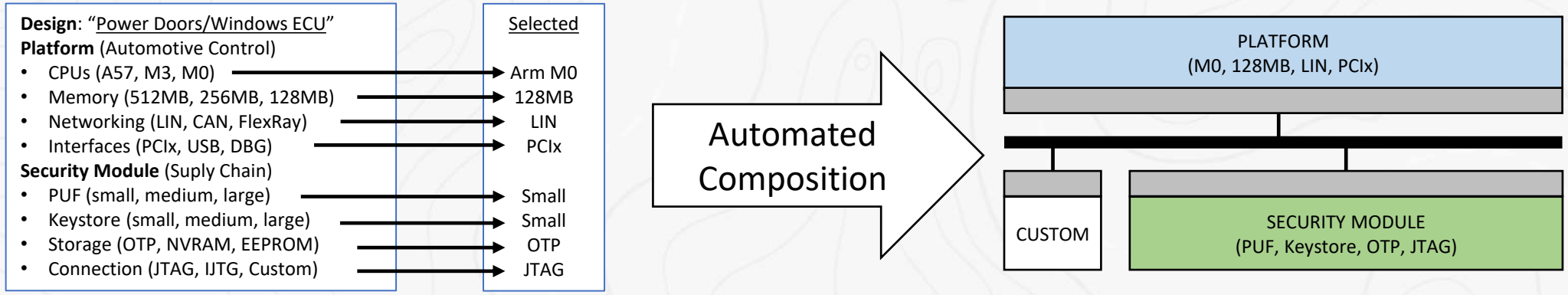
- There are many effective outer perimeter attack strategies
- We are assuming outer perimeter is penetrated or compromised
- AISS focus is only on securing inner perimeter with on-chip structures
- Some level of off-chip support is also needed

AISS: Composition

- Phase I - **Assisted Composition** – Components are specified
  - Processor & security related components are user selected & automatically integrated



- Phase II - **Automated Composition** – Configuration is specified
  - User selects a platform and provides configuration to a tool that automatically generates an integrated system



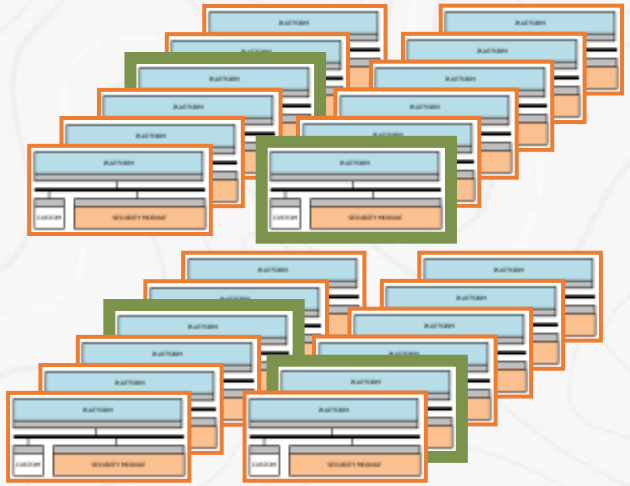
AISS: Optimized Composition

- Phase III - **Optimized Composition** – Objectives are specified
  - User selects a platform and supplies a cost function with size, performance, power and security goals to guide combinatorial optimization to find **best architectures** which are presented to the user for assessment and selection

Design: "Power Doors/Windows ECU"

Platform (Automotive Control)

- Performance = 2
- Size = 9
- Power = 3
- Security = 3
  - Supply Chain = 7
  - Side Channel = 2
  - Reverse Engineering = 5
  - Malicious Hardware = 1



Combinatorial Optimization explores HUGE solution spaces (billions), but requires rapid estimation of "goodness"

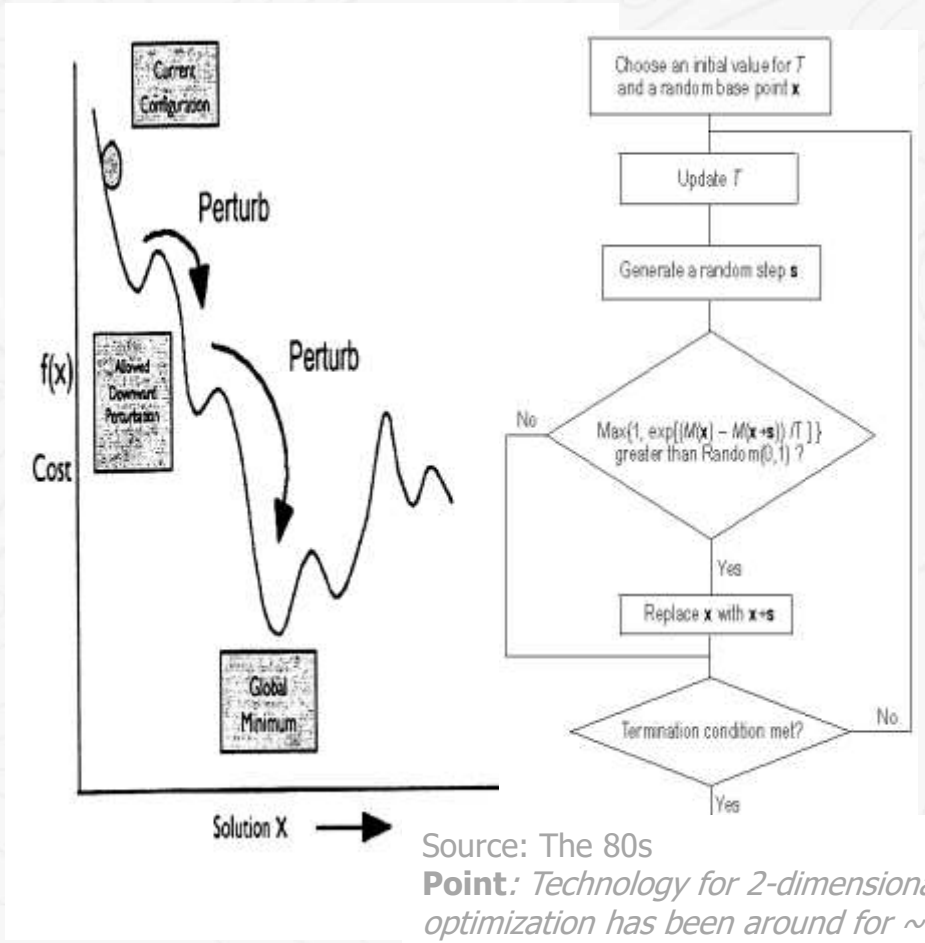
Performance and *Size* estimators are well understood and incorporated in modern tools

AISS will drive discovery of rapid estimation of **power** and **security**

$$f(a,b,c,d) = \sum(a*\textbf{Performance},\ b*\textbf{Size},\ \underbrace{c*\text{Power}}_{\text{estimate}},\ \underbrace{d*\text{Security}}_{\text{estimate}})$$

AISS: Optimization Cost Functions

$$f(a,b) = \sum(a*\textbf{Performance}, b*\textbf{Size})$$



Source: The 80s

Point: *Technology for 2-dimensional optimization has been around for ~40 years*

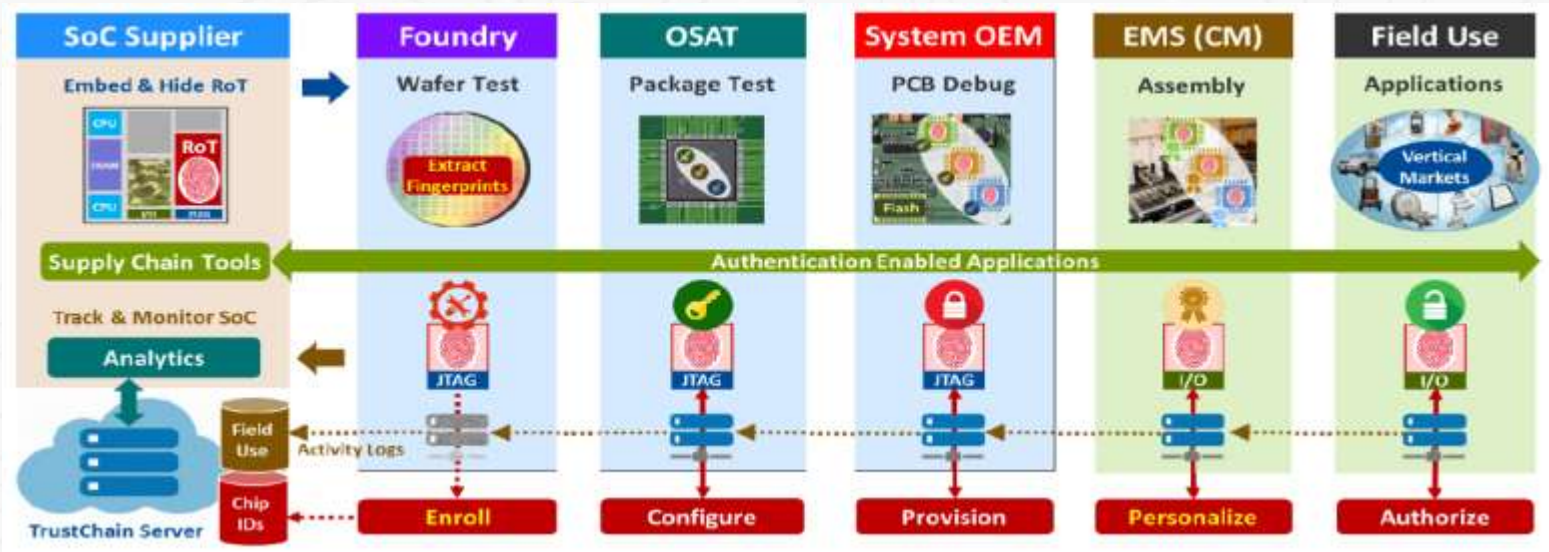
Cost Function Examples

Application	Perf.	Size	Power	Security
Lawn Sprinkler	2	7	9	1
Engine Control	6	5	1	3
Guided Projectile	5	1	9	7
Network Router	9	5	1	8
Mobile Phone	7	9	9	7
Smart Watch	3	6	9	3

Security Cost Function Expansion

Application	Side Channel	Reverse Eng'g	Supply Chain	Malicious Hardware
Lawn Sprinkler	1	1	9	1
Engine Control	1	7	5	2
Guided Projectile	3	9	5	9
Network Router	9	7	8	9
Mobile Phone	8	9	9	6
Smart Watch	6	8	9	1

Summary: Enabling Security in the Supply Chain



- **Design**: Create secure-reconfigurable SoCs with a unique ID based on an inborn Root of Trust
- **Enroll**: Extract chips unique ID into a secure server during first power up at wafer test
- **Configure**: Inject keys to encrypt, sign, or decrypt content for devices or end-applications
- **Provision**: Program SKUs downstream to reduce inventory risk and exploit volume ramp
- **Personalize**: Enables secure device identity during PCB assembly based on the chip's Root of Trust
- **Authorize**: Allow authorized parties to securely sign devices based on the SoC Root of Trust
- **Update**: Securely update firmware and provision SOC hardware features in the field
- **Monitor**: Track field use and evolve Big Data analytics on field failures, intrusions, counterfeits

Source: Mentor Graphics, 2017

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SoC Functional Safety Overview and Use Cases, Standards, and Dependability Lifecycle

Bala Chavali

AMD

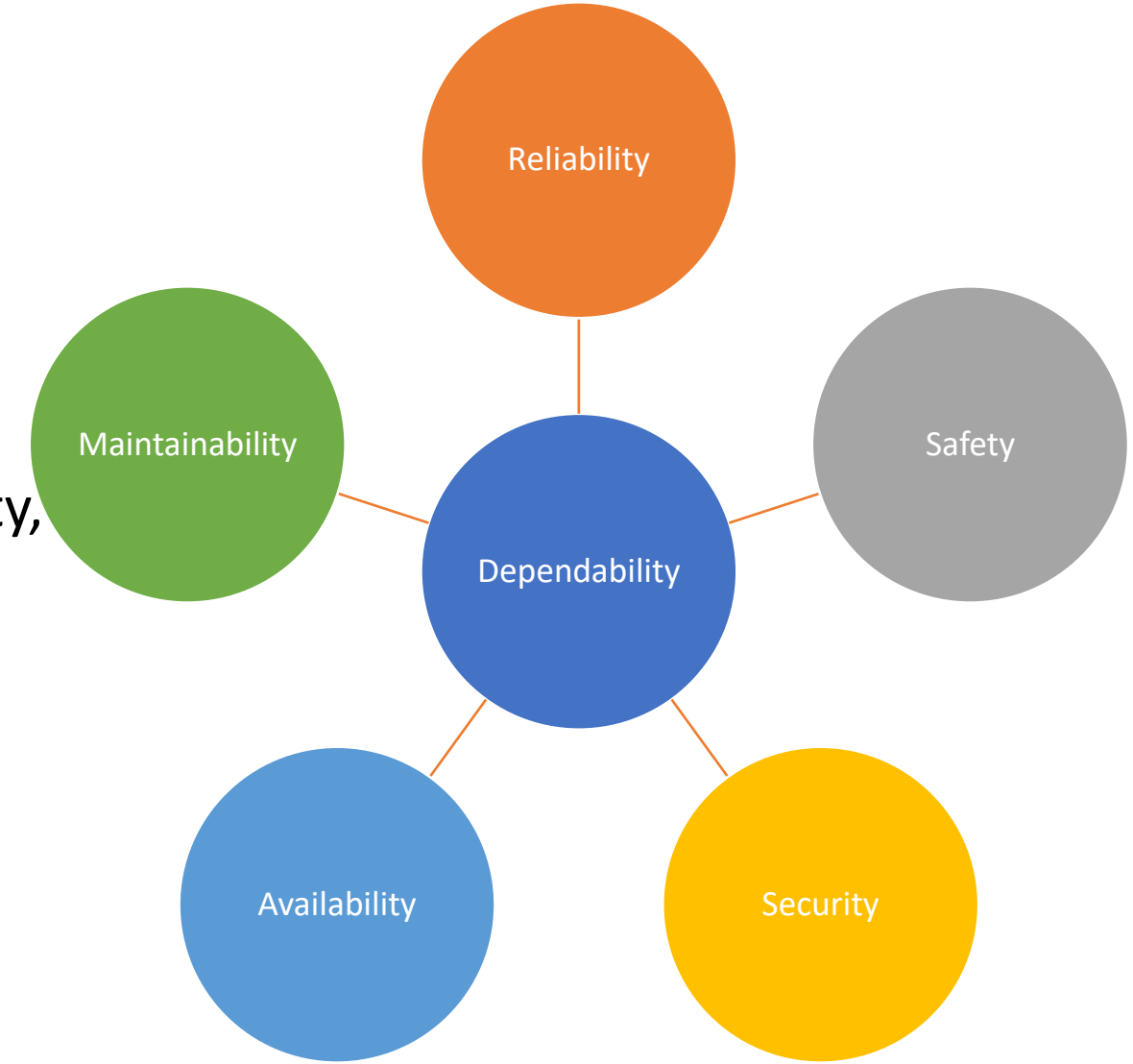
accellera SYSTEMS INITIATIVE

Outline

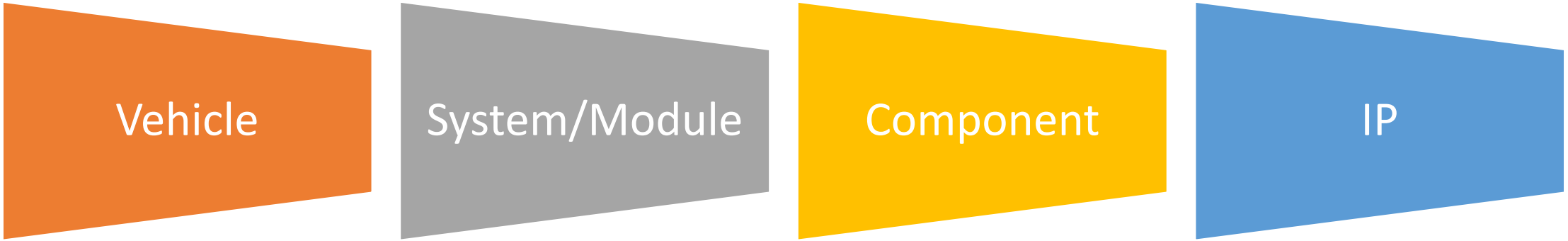
- Define dependable system
- Product hierarchy
- Lifecycle phases
- Functional safety
- SoC Challenges
- Dependability lifecycle challenges
- Addressing the challenges

Dependable System

- Trustworthiness of the system
- Attributes are defined to quantify this behavior
  - Reliability, Safety, Security, Availability, Maintainability
- Challenges are present to achieve this within the product lifecycle
- Standards and multiple industry WGs working towards achieving these goals

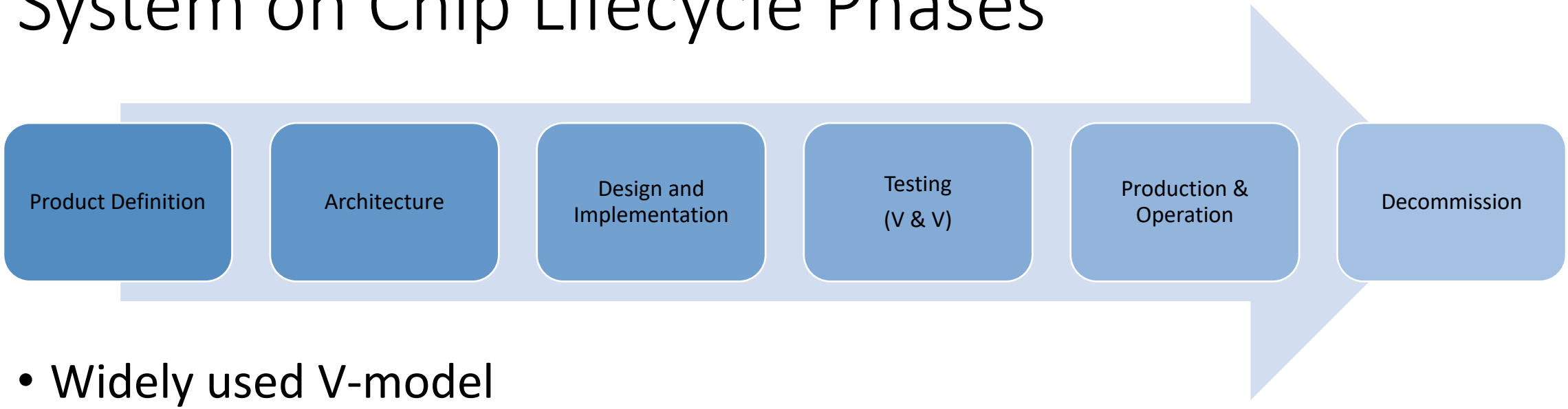


Product Hierarchy - Automotive Application



- Automotive applications require addressing many of these attributes
  - Safety, Security, Reliability, Availability, Maintainability
- Vehicle requirements fan from Item down to IP/Part/Element
  - Functional safety targets to reduce risks in critical applications
  - Reliability addresses error avoidance or detection due to wear and tear, aging, and defects
  - Security to address threat levels and attacks during product operation

System on Chip Lifecycle Phases



- Widely used V-model
- Dependability attributes lifecycle is integrated as part of product lifecycle
- Product need to define the targets for each attribute across the lifecycle
- Product specifies the qualitative and quantitative targets to be MET

Automotive Functional Safety

- Safety standards provide guidelines on management and risk mitigation of a product through the lifecycle
- Addresses this via two methods
  - Random faults
  - Systematics faults

Random Faults

- Quantitative
- Perform safety analysis
- Safety Measures

Systematic Faults

- Qualitative
- Traceability of Requirements
- Design Methods
- Test Methods

SoC Definition and Design Phase

- SoC could implement a combination of internal IPs and third-party IPs
- Internal IP Blocks
  - Product Requirements are captured and published
  - Requirements are converted to architecture and design is implemented
  - All requirements are reviewed and signed off
  - Traceability is established to confirm user features are matched with results
- Third-party IP Blocks
  - Documentation of product, architecture are provided
  - Reports provided to establish coverage

Definition and Integration Phase Challenges

- Data exchange format of requirements from IP vendor to another vendor
- Integration challenges due to non-standard design or custom methodologies
- Lack of common terminology or language between databases and users

- Results in higher effort, product delay, and additional cost to debug or fix bugs

Challenges within Dependability Lifecycle

- Challenges multiply when a product must address multiple lifecycles within one product lifecycle
- Need to comply to multiple standards and produce multiple work products
- Use similar dataset to perform multiple quantitative and qualitative analysis – traceability, risk analysis, threat analysis, safety analysis

- Results in COMPLEXITIY, EFFORT, and COST

Addressing the Challenges

Experts started working groups to address these challenges

- Define a generic dependability development lifecycle
- Identify the wholistic data set to address all applications
- Create a common data exchange language, data models and databases
  - Automotive, Industrial, Avionics, Medical
  - Vehicle, System/Module, Component, IP
- Promote interoperability between automation tools

Industry WG Efforts

- Accellera Functional Safety WG
  - Define a standardize functional safety data exchange
  - Define a language/format to exchange functional safety data across all layers
  - Released a white paper highlighting all these challenges, ongoing work, and data model work
- IEEE P2851 Functional Safety WG
  - Published a white paper
  - Ongoing work on data exchange format and dependability lifecycle

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Reliability Analysis method for Safety-critical CPU designs : PACE

Reiley Jeyapaul and Balaji Venu

arm Research

accellera SYSTEMS INITIATIVE

Agenda

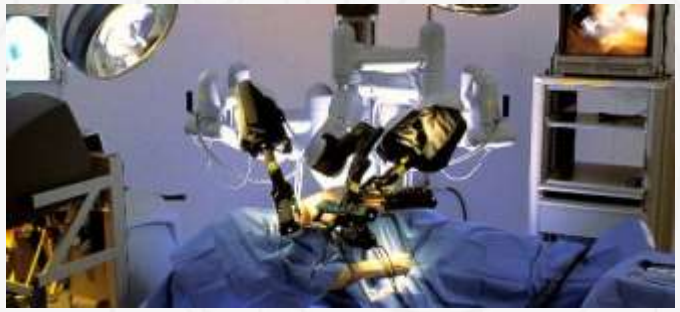
- The need for a scalable reliability analysis method
- PACE methodology
- Discussion of Results
- Use-case in the Arm ecosystem

Markets and Applications

Automotive
Autonomous driving

Industrial
Factory automation

Healthcare
Robotic surgery



Transportation
Train control systems

Avionics
Flight systems

Consumer
Domestic robots

Functional Safety Opportunity

Powertrain

Lower emissions

Vehicle electrification

Autonomous drive

ADAS

Self driving

Information

IVI

Connected car

Designing Error Resilient Systems

An implementation of fault tolerance,

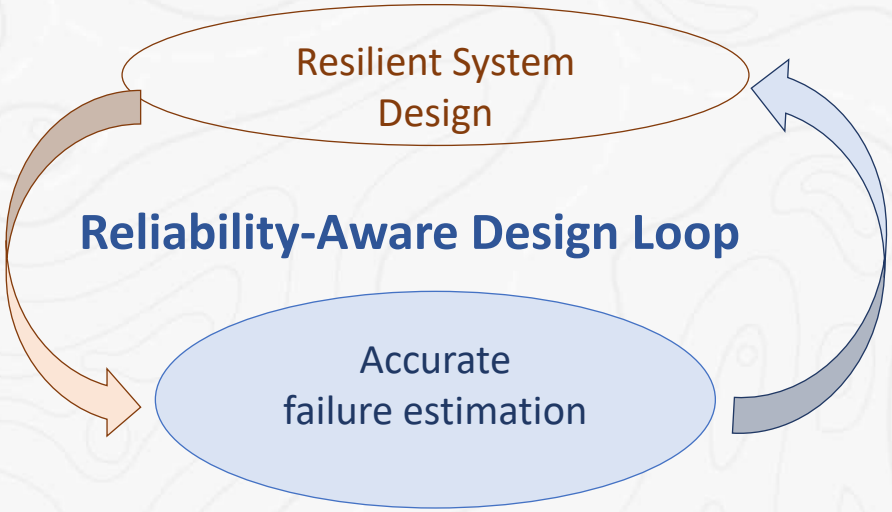
- ➢ involves a mechanism to introduce or enhance the intrinsic masking effects of the design

Accurate vulnerability analysis and failure quantification

- ✓ establishes an understanding of the system's intrinsic resilience
- ✓ enabling efficient and resilient designs

<u>From the Designer</u>

- System specifications
- Design Parameters
- RAS Implementations



<u>Feedback to Designer</u>

- Data-driven study of RAS features/implementations
- Trade-off analysis – reliability vs performance vs power
- Co-design opportunities and analysis

Scalable, fast, and accurate reliability estimation enables robust SoC designs !

	ASIL B	ASIL C	ASIL D
PMHF	100 FIT	100 FIT	10 FIT

Big Picture – What is PACE trying to solve?

Architecture Vulnerability Factor (AVF) plays an important role in resilient system design

1. Eliminating the fraction of safe faults from failure rate calculation

$$FIT_{CPU} = FIT_{CPU\_RAW} * (1 - Diagnostic\_Coverage) * (AVF)$$ [1]

   - Higher diagnostic coverage incurs high power, performance and area (PPA) overhead
   - De-rate FIT rate using AVF.

2. Enable hardware software co-design with reliability awareness
   - Which bits of hardware is highly vulnerable and should be protected?
   - What type of safety mechanisms should I use based on AVF data?

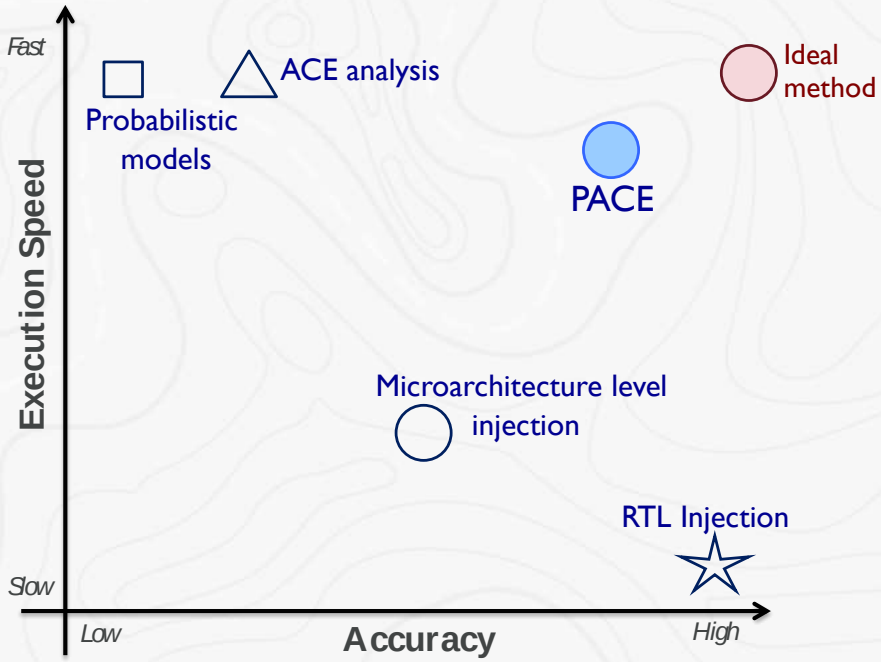
PACE methodology enables fine-grained safeness analysis using formal methods

[1] ISO 26262-5 2018, Part 5, Annex C
[1] FIT rate: Number of failures in 1 billion hours of operation (114,000 years)

Reliability Estimation Methods: landscape

1. Exhaustive Fault Injection (EFI) methodology
   - Number of faults *actually* manifesting as an error in the final output
   - Accuracy is high. Takes long time to estimate

2. PACE (Proof driven ACE analysis)
   - *Probability* of a (soft) fault manifesting as an error visible to the user
   - Pessimistic analysis, trading off accuracy for speed



Goal is to compute AVF (Fraction of the design that is vulnerable to soft errors)

[1] "Merlin: Exploiting dynamic instruction behavior for fast and accurate microarchitecture level reliability assessment", *Proceedings of the 44th Annual International Symposium on Computer Architecture*,

PACE extends ACE analysis to the RTL

- **ACE : A common technique used to estimate AVF**
  It involves manual pre-analysis of array structures in CPU designs and marks a sequential bit (Flip Flop, latch, etc) as
  - ACE (Architecturally Correct Execution) for the portion of time it is vulnerable to transient faults
  - un-ACE for the remaining portion of time.

  Largely carried out on **performance** models of CPU

- **In PACE**
  We have automated carrying out ACE analysis directly on key design blocks of Arm Cortex-R52 CPU **RTL design** using formal methods

- **We compare our results against Exhaustive Fault Injection (EFI)**
  Ground truth numbers
  - Injected faults in every flip flop during every clock cycle of the benchmark and calculated AVF numbers (125 million independent RTL simulations and tera bytes of data ☺).

What is possible with PACE

- AVF estimation using PACE is 276x faster than EFI on key design blocks accounting for 25% of the Arm Cortex-R52 CPU
- AVF estimated using PACE is pessimistic (on average 4.7x higher than EFI)

- **What did formal buy us?**
  Two guarantees, as PACE uses formal methods to prove ACE/un-ACE behavior

1. 100% confidence in estimated AVF numbers. All sequential bits reported as vulnerable by EFI is also captured as vulnerable by PACE.
   - PACE does not impact fault coverage of the design.

2. Fault space pruning – All sequential bits reported by PACE as not vulnerable is also captured as not vulnerable by EFI.
   - PACE can be used as a complementary technique with others known in literature (Statistical fault injection, probabilistic analysis)

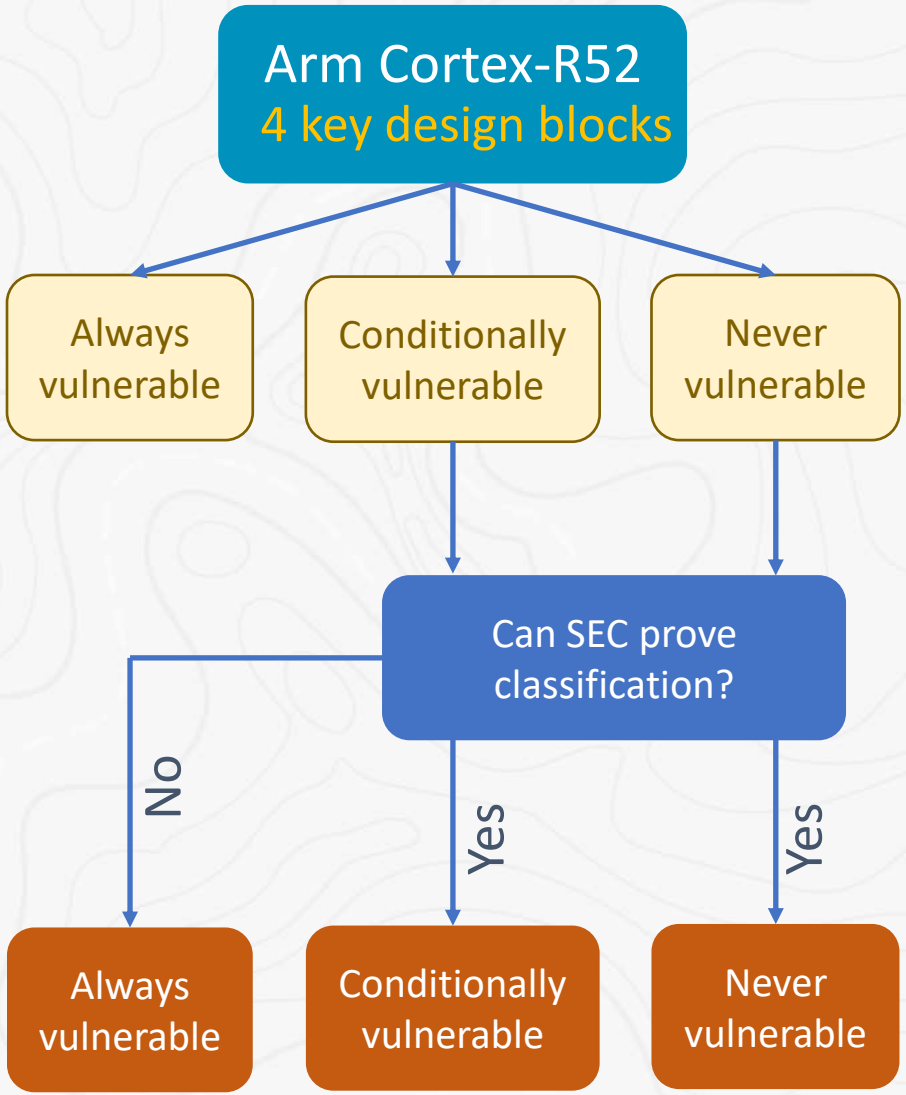
The PACE Methodology

Main idea (Guess and Check)

1) **Classify** Flip Flops using formal properties
   Come up with hypothesis around vulnerability

2) **Prove Classification** using
   Sequential Equivalence Check (SEC) formal tool

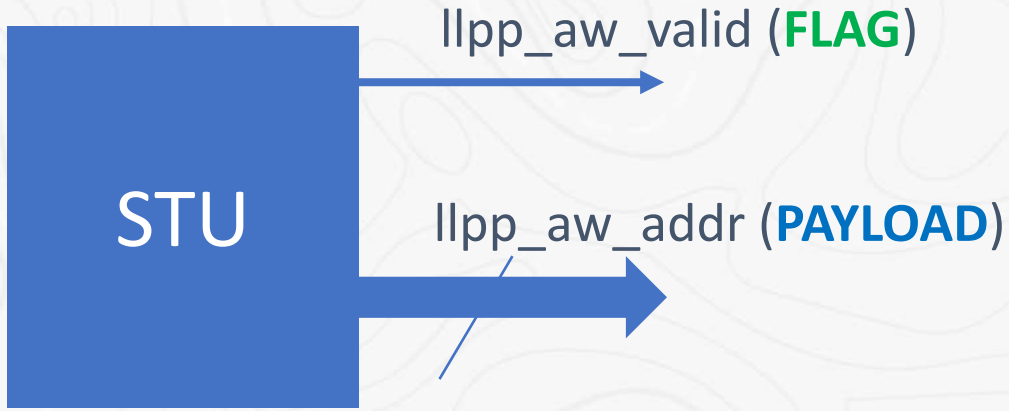
3) **Generate** PACE models using Classification
   AVF calculated for CPU ($F_{safe} = 1 - AVF$)
   Methodology Cross validated with Fault Injection experiments (not a must to be carried out)



Step1: Identify Conditional Vulnerable Flops

Inspired producer-consumer handshake protocol that is implemented in pipeline stages, storage structures and bus protocols on modern CPUs:

1. Purpose of "**FLAG**"
   Notify consumers payload is ready to be consumed
2. Many such "**FLAG** - **PAYLOAD**" pairs
   should exist in the micro-architecture



Vulnerability hypothesis:

If (**FLAG** == 1'b1)
    PAYLOAD is vulnerable to transient faults
else
    PAYLOAD is not vulnerable

Further implementation details available in the publication – *PACE: AVF estimation using formal methods*

Identify “FLAG-PAYLOAD” pairs in the design

System Verilog Assertions (SVA) to the rescue, we did a bit of brute-force approach

$change(PAYLOAD) |-> (FLAG == 1’b1)

Identify List of FLAGs using naming convention and designer inputs

```
dpu_list.dat : u_kite_dpu.u_dp.u_mac.valid_ex1
dpu_list.dat : u_kite_dpu.u_dp.u_mac.valid_ex2
dpu_list.dat : u_kite_dpu.u_dp.u_store0.str_a_valid_ex1
dpu_list.dat : u_kite_dpu.u_dp.u_store0.str_a_valid_ex2
dpu_list.dat : u_kite_dpu.u_dp.u_store1.str_a_valid_ex1
dpu_list.dat : u_kite_dpu.u_dp.u_store1.str_a_valid_ex2
dpu_list.dat : u_kite_dpu.u_etmif.addr_data_valid_ret
dpu_list.dat : u_kite_dpu.u_etmif.dpu_fe_valid_ret_reg
dpu_list.dat : u_kite_dpu.u_etmif.instr_valid_ret
dpu_list.dat : u_kite_dpu.u_etmif.isb_valid_ret
dpu_list.dat : u_kite_dpu.u_etmif.ls_valid_ret
dpu_list.dat : u_kite_dpu.u_etmif.valid_branch_instr_ret
dpu_list.dat : u_kite_dpu.u_ldst.ls_real_valid_iss
dpu_list.dat : u_kite_dpu.u_ldst.ls_valid_ex1
```

Consider all other RTL signals as PAYLOAD

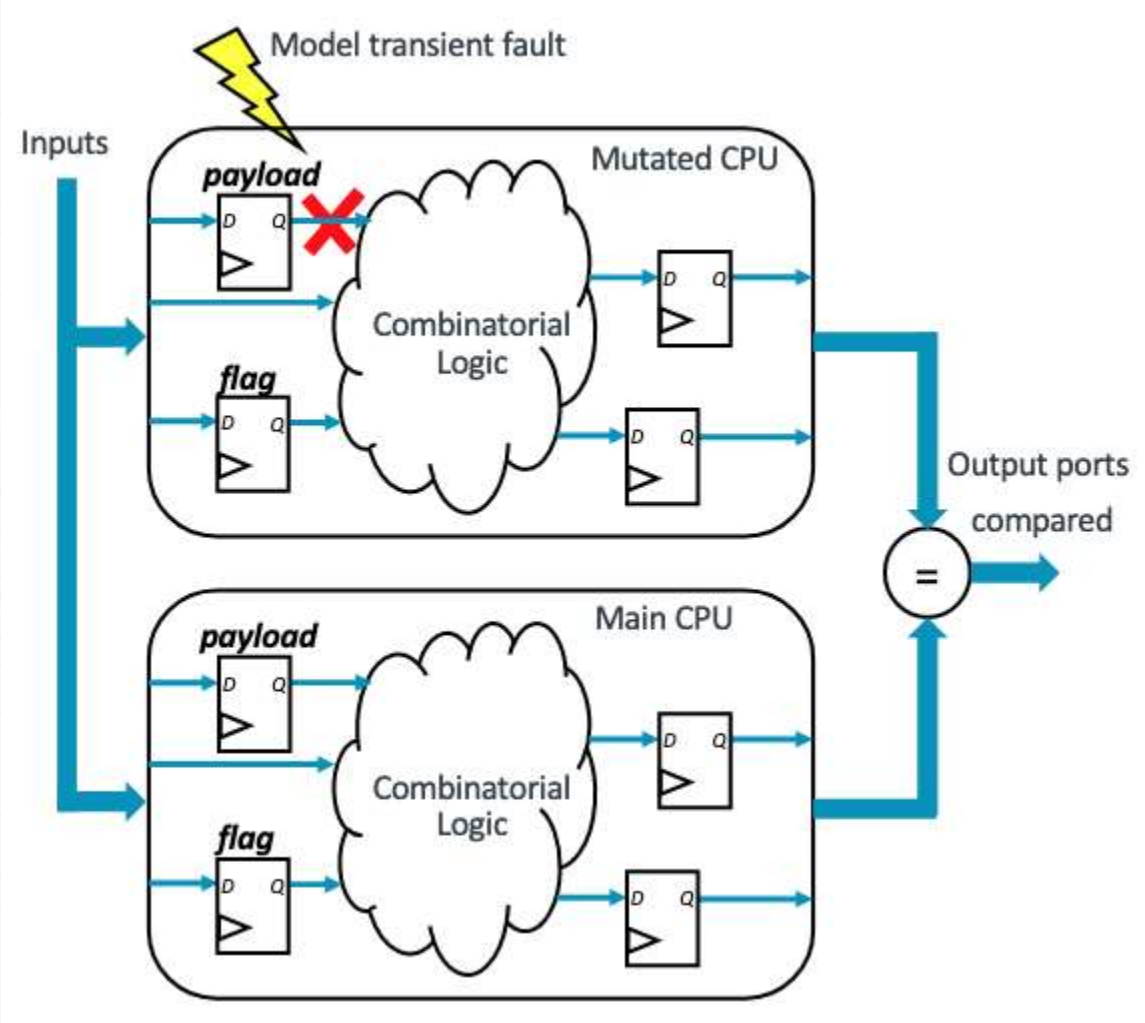
Check validity of property across all combinations of FLAG and PAYLOAD

Combinations for which the property holds are potential pairs

Step2: Proving Classification

Hypothesis (conditionally vulnerable):

If (**FLAG** == 1'b1)
    PAYLOAD is vulnerable to transient faults
else
    PAYLOAD is not vulnerable



Three step process:

1) **Create** a copy of the CPU
2) **Mutate** the "payload" behavior in the copy using SystemVerilog assumptions (more details in the paper) Assumptions model transients to occur for verifying ACE behavior
3) **Prove** equivalence

Results

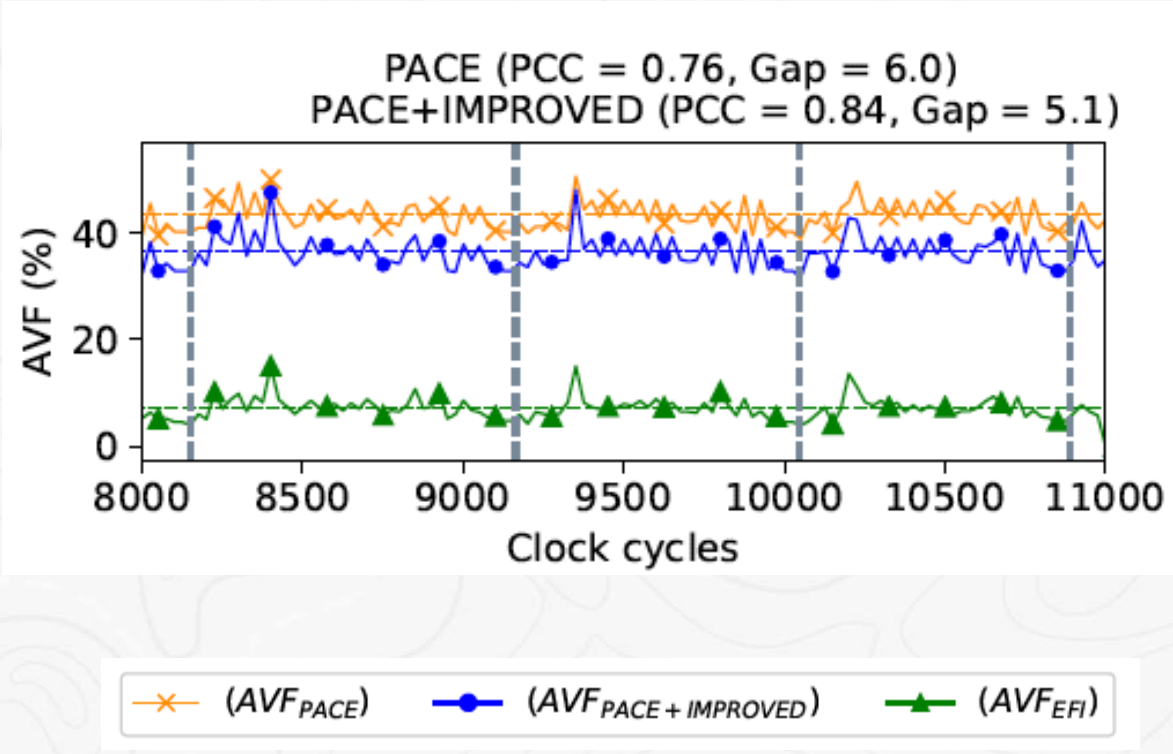
AVF values

1. PACE methodology

$$AVF_{PACE} = \frac{number\ of\ flops\ that\ \textbf{potentially}\ result\ in\ an\ error}{total\ number\ of\ flops}$$

2. Fault Injection methodology

$$AVF_{EFI} = \frac{number\ of\ flops\ that\ actually\ result\ in\ an\ error}{total\ number\ of\ flops}$$

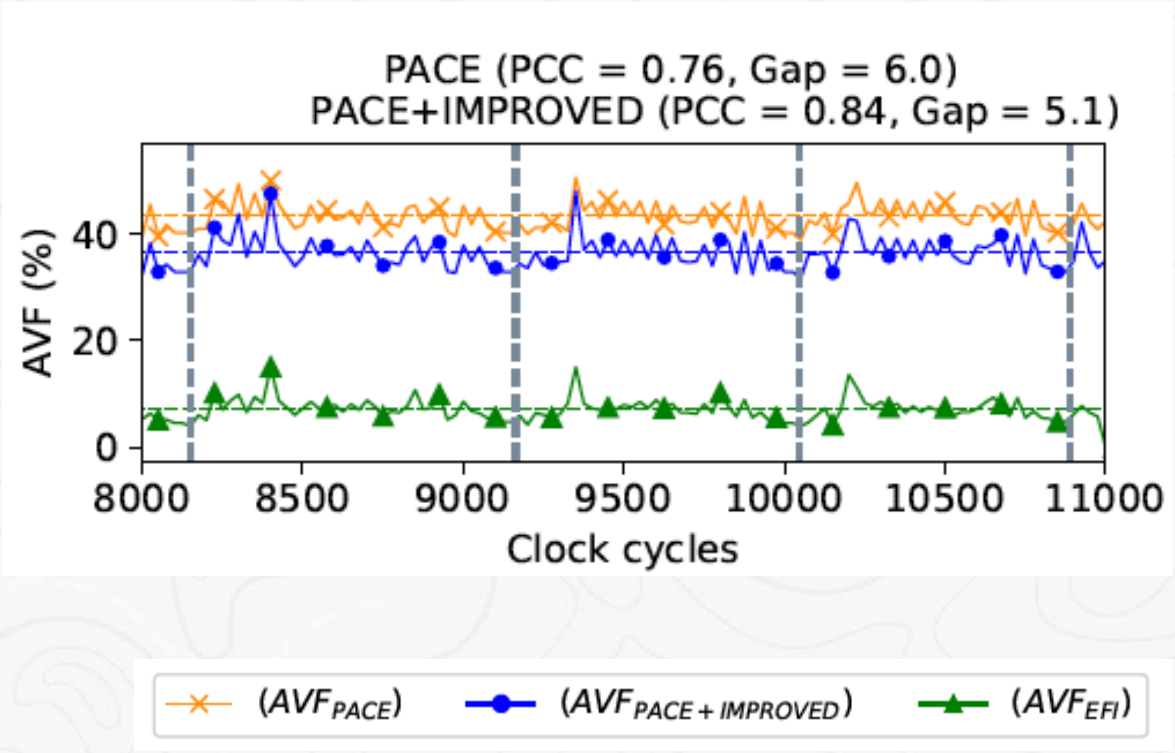


The guarantees:

- All bits captured as vulnerable by the **green** line is captured as vulnerable in the **orange** line
- All bits captured as **not** vulnerable by the **orange** line is also captured as **not** vulnerable by the **green** line.

PACE Improved results

- We investigated the Decode unit that was exhibiting low correlation with EFI results
- It implements a Circular buffer whose vulnerability condition was not captured by our “Flag – payload”
- We came up with these conditions and proved it using SEC and it improved our AVF results.



The guarantees:

- All bits captured as vulnerable by the **green** line is captured as vulnerable in the **orange**/**blue** line
- All bits captured as **not** vulnerable by the **orange**/**blue** line is also captured as **not** vulnerable by the **green** line.

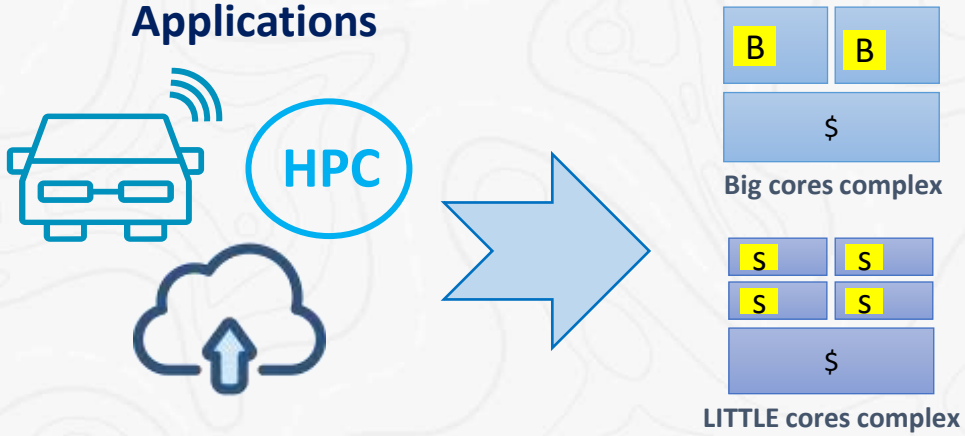
Use-case in the Arm ecosystem

Model Generation Phase (Arm)

PACE Vulnerability Estimation *Models*

- Automatically generate using Formal methods
- Designer inputs
- Model estimates vulnerability and F_{safe}
- Arm licenses model to partners

F_{safe} estimation phase (Partners)



- Partner integrates PACE models in their simulation/emulation platform
- Runs application and computes F_{safe}
- Uses it to de-rate FIT rate based on application behavior
- PACE models can be compiled to Cycle models to enable TIER1s and OEMs

In summary !!!

A **fast** and **automated** methodology to estimate AVF numbers with **100%** confidence !!!

Fine grained vulnerability data (report highly vulnerable portions of the CPU)

2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Automated Solutions for Safety and Security

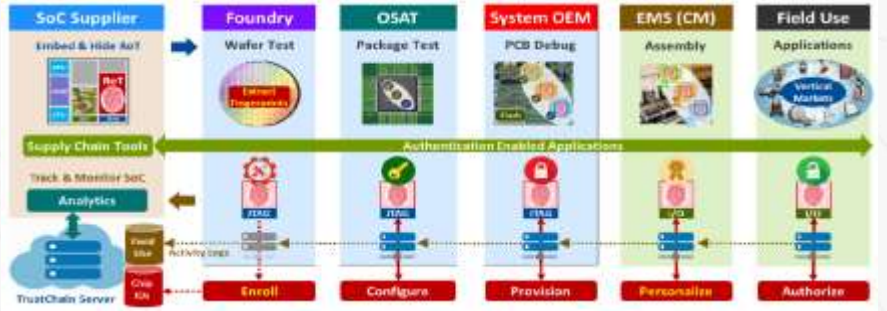
Meirav Nitzan

SYNOPSYS®

accellera SYSTEMS INITIATIVE

Summary of Previous Presentations

- Serge Leef, DARPA:
  - Automate inclusion of scalable defense mechanisms into chip designs to enable security vs. economics optimization
  - *Challenge: Automate Security SoC design development*



- Bala Chavali, AMD:
  - SoC Functional Safety Overview and Use Cases, Standards, and Dependability Lifecycle
  - *Challenge: standardize all aspects of a dependable design data exchange across supply chain*

- Reiley Jeyapaul and Balaji Venu, arm Research
  - Reliability Analysis method for Safety-critical CPU designs : PACE
  - *Challenge: create an innovative solution to AVF and app based FIT de-rate using existing EDA tools*

EDA Understands and Accepts the Challenge!

- Creating new technologies and leveraging existing ones to create a *holistic solution* for semiconductor & system industries
  - Address Safety, Security, Reliability and Time determinism of the product
  - A Software to Silicon solution, and everything in between
- Work with Government and Academy on various initiatives
  - Enable an enhanced solution for design automation
- Develop innovative solutions in collaboration with our customers
- Participate in major standardization bodies to drive, understand and implement new data exchange formats
  - ISO/SAE (21448, 21434, 26262, …)
  - Accellera (IPSA/SA-EDI, FuSa)
  - IEEE (P2851, ...)

Key: a holistic solution for Safety and Security, from Software to Silicon

Key Considerations for Safety and Security

- Functional Safety goal: reduce the risk of hazards due to **Systematic failures** during design or **Random failures** during operation to an acceptable minimum.
  - Considerations:
    - Integration of commercial Safety compliant IPs in the SoC architecture
    - How can my design fail due to insufficient planning or testing (systematic)?
    - How can my design fail due to random faults?

- Security goal: reduce or eliminate the hazard due to **intentional manipulation** of the system functionality with malicious intent
  - Considerations:
    - Integration of commercial Security IPs in the SoC architecture
    - How can my SW fail due to malicious attack?
    - How can my HW fail due to malicious HW attacks?
    - How can someone insert trojans to my design? How can my design secrets get stolen?

Safety design flow: Addressing Systematic Failures

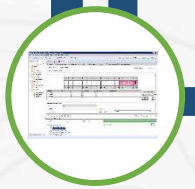
Code analysis, managed testing, & planning

Software and System Testing

Semiconductor IP
(Automotive grade, Security IP)

Software

Semiconductor



Package & MFG

Architectural Exploration

Digital Design

Analog Design

Verification

Verified Hardware /Software

Package Design & Verification

Tape Out and Manufacturing

Function, Safety & Security Requirements

Architectural Exploration

RTL Design

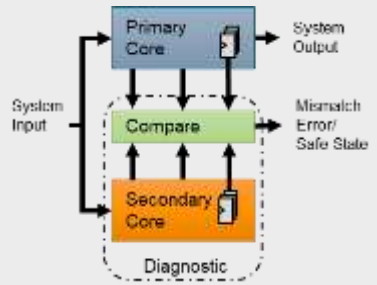
Synthesis

Place and Route

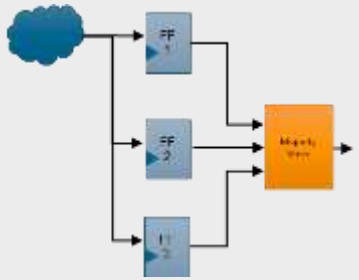
Safety Implementation: Random Faults Detection

EDA Tools Can Automate Insertion of the HW Safety Mechanisms

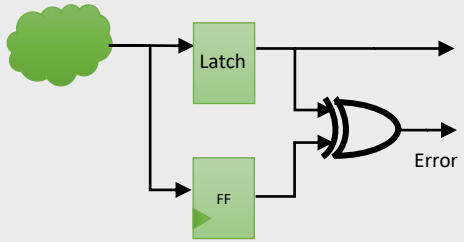
DUAL CORE LOCK STEP (DCLS)



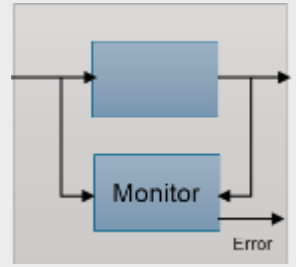
TRIPLE MODE REDUNDANCY (TMR)



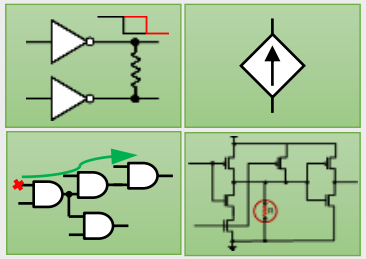
MEMORY ECC/EDC



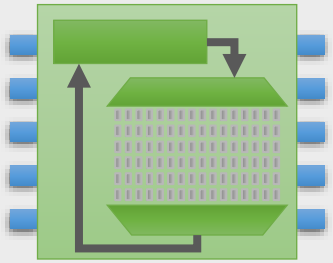
MONITORS



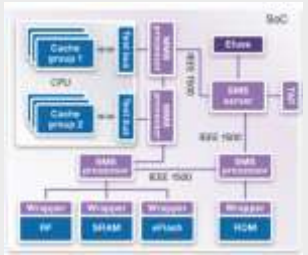
ATPG



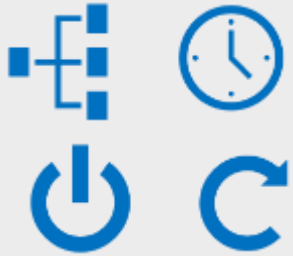
LOGIC BIST



MEMORY BIST

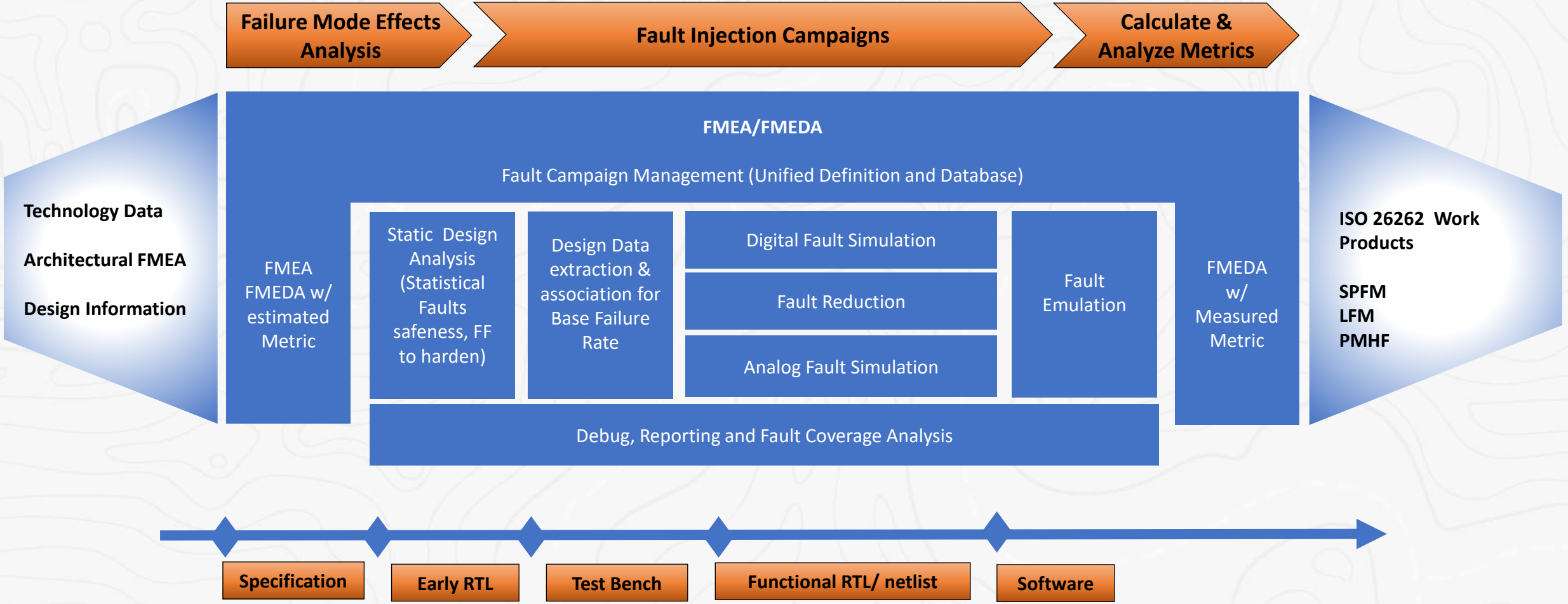


FREEDOM FROM INTERFERENCE



Safety Analysis: FMEDA Flow Automation

End-to-end automated solution for Random Fault Coverage and ISO 26262 Metric computation



Security design flow: from Software to Silicon

Code analysis, managed testing, & planning

Virtual Prototyping | Application Security Testing | SW Composition Analysis | Testing

Sub-system IP
Individual & Processor IP

PPA Analysis for DPA | Security Register Infrastructure | Synthesis for SCA & Beneficial Code Insertion | Programmable Rule Checks

Signoff

Risk Analysis and Threat Models
Architectural Exploration

Testbench for Simulation or Emulation | Track Secure Data | Security Rules and Checks

Information Flow | Digital and Analog Fault Injection | Test Plan Validation with Mutation Analysis

Security Tests Library

Fault Emulation & Power Analysis

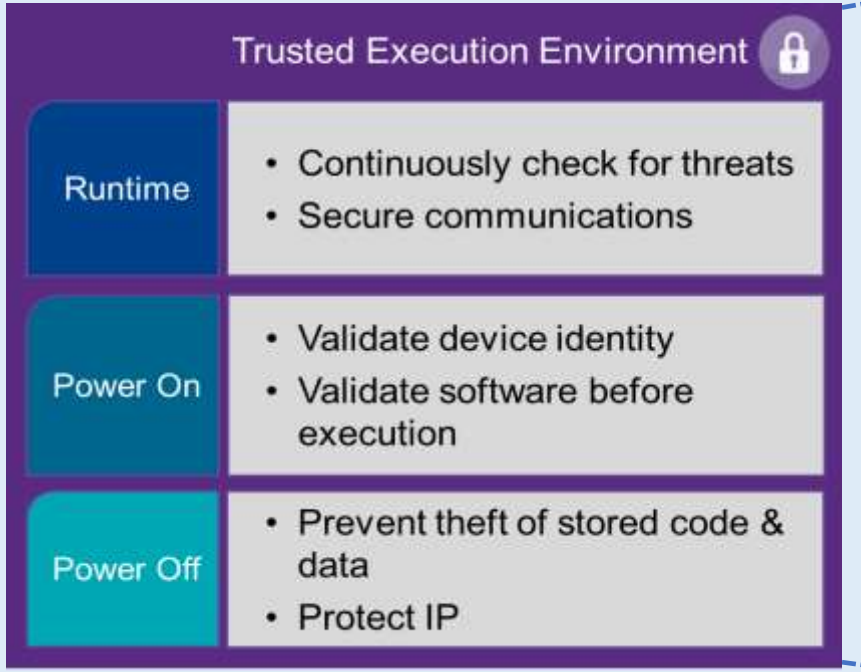
Prototyping

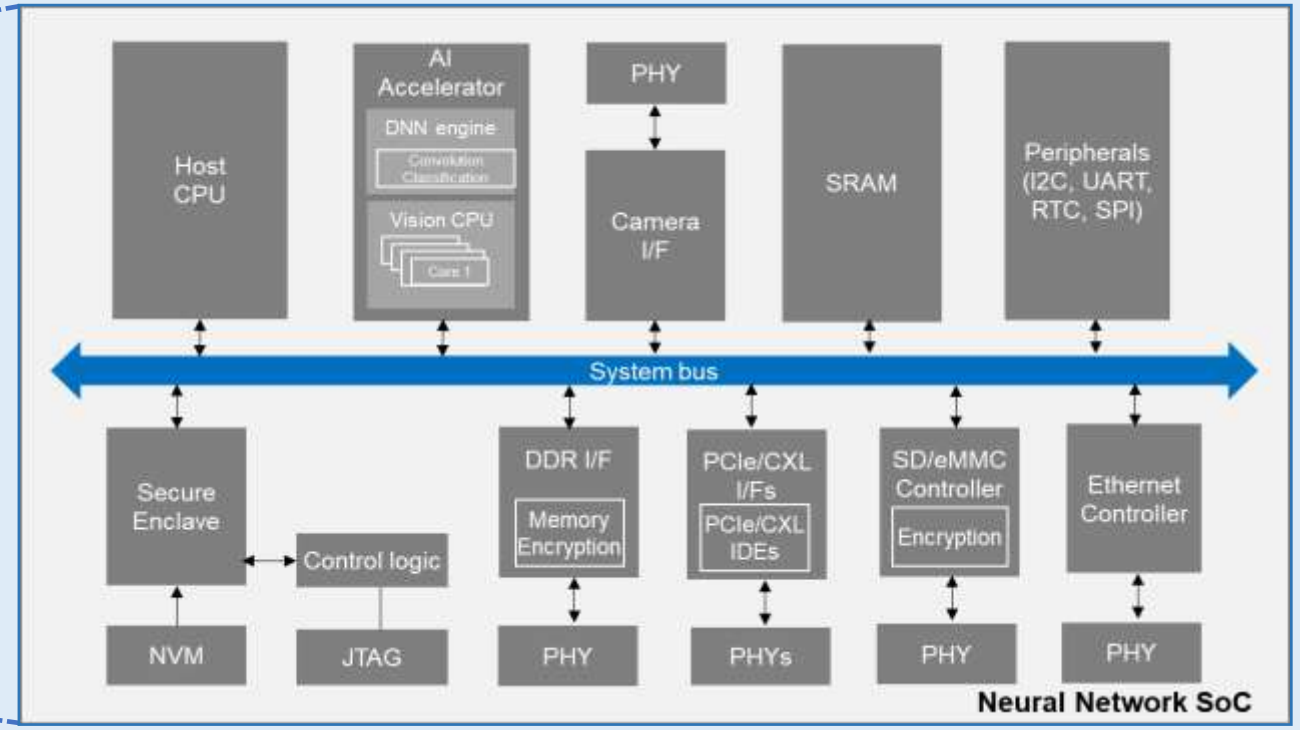
Function & Security Requirements
(e.g., CWE/CVE)

Architectural Exploration | RTL Design | Synthesis | Place and Route

Security-Aware Silicon IPs

Critical Foundation for System Security

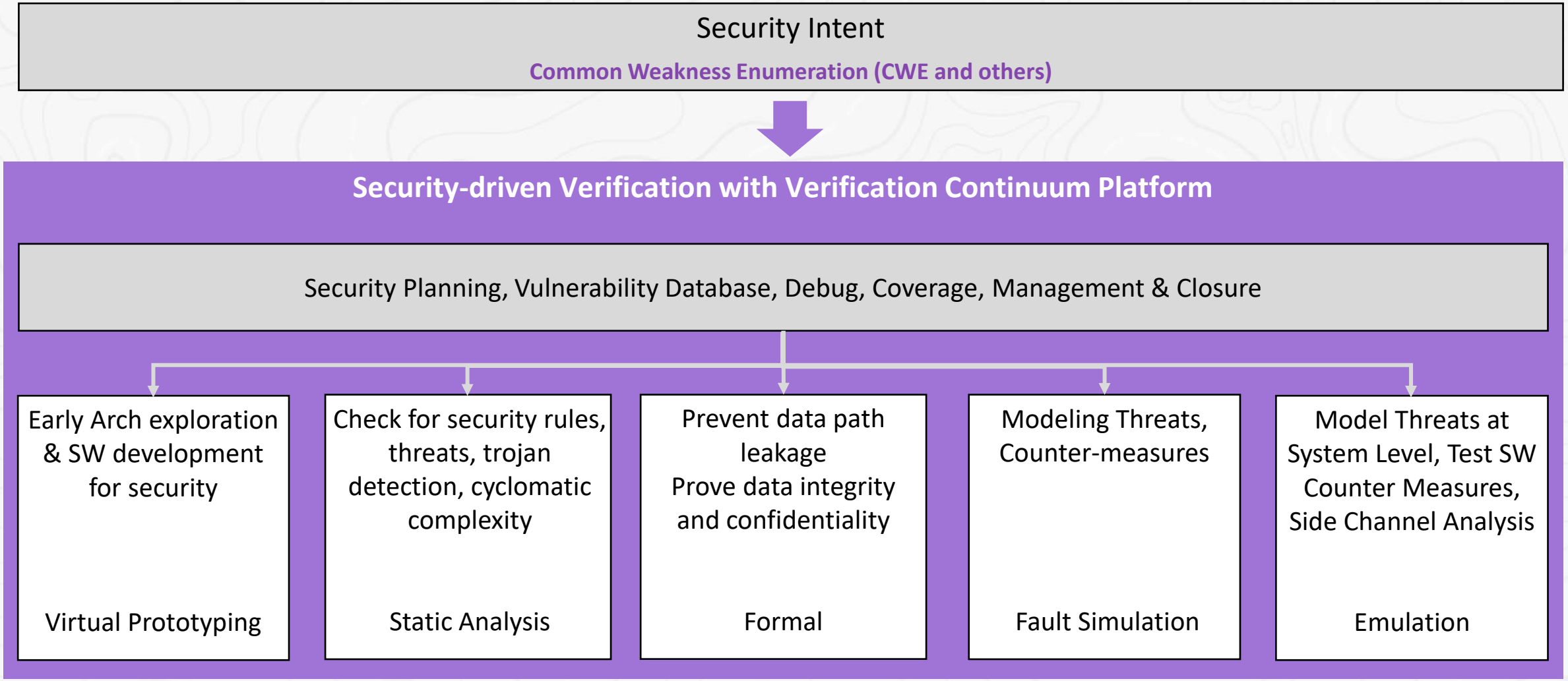




Security Needs

- Overall SoC protection functions (secure bootstrap, key management, secure updates, secure debug/JTAG access…)
- Secure data in motion
- Secure data-at-rest
- Encryption and authentication for model updates, secure communication and inputs from peripherals

Security Verification: Enabling Zero-trust Validation



Automating Secure Digital Design

Robust and secure design

Digital Design: Solutions for Security

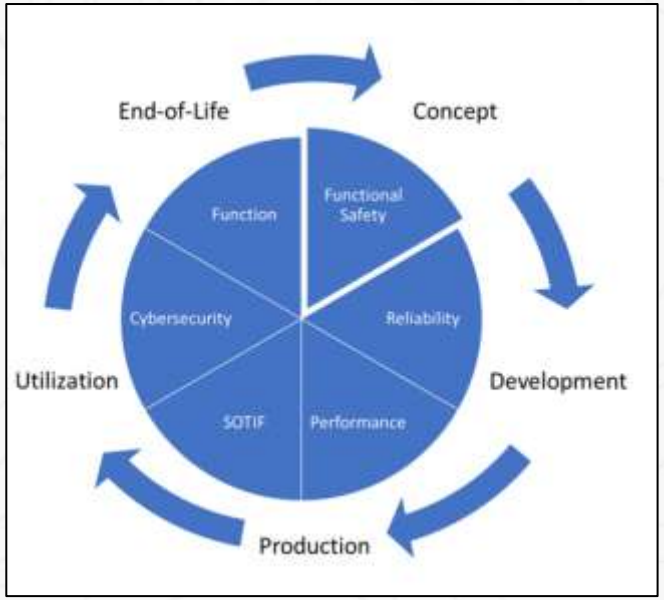
Architectural Exploration	RTL Design	Synthesis	Place and Route
Early PPA analysis to identify vulnerabilities to attacks such as differential power analysis	Integrate registers to communicate with security infrastructure Add register and test mode access controls to support SoC security policy	Security-aware synthesis to avoid optimizing away redundancy Synthesis for watermarking, obfuscation, and logic locking	Security-aware placement avoids malicious implants, protects against fault injection attacks, and optimizes logic redundancy Routing of security-critical signals to avoid power/EM and fault injection attacks	Smart metal fill to resist optical probing and avoid malicious implants Security-aware check-and-report	Timing closure in presence of security features Power analysis to confirm effectiveness of security measures Signoff

Safety-Security Alignment Standardization

- IEEE P2851 Title: "**Standard for functional safety data format for interoperability within the dependability lifecycle**."

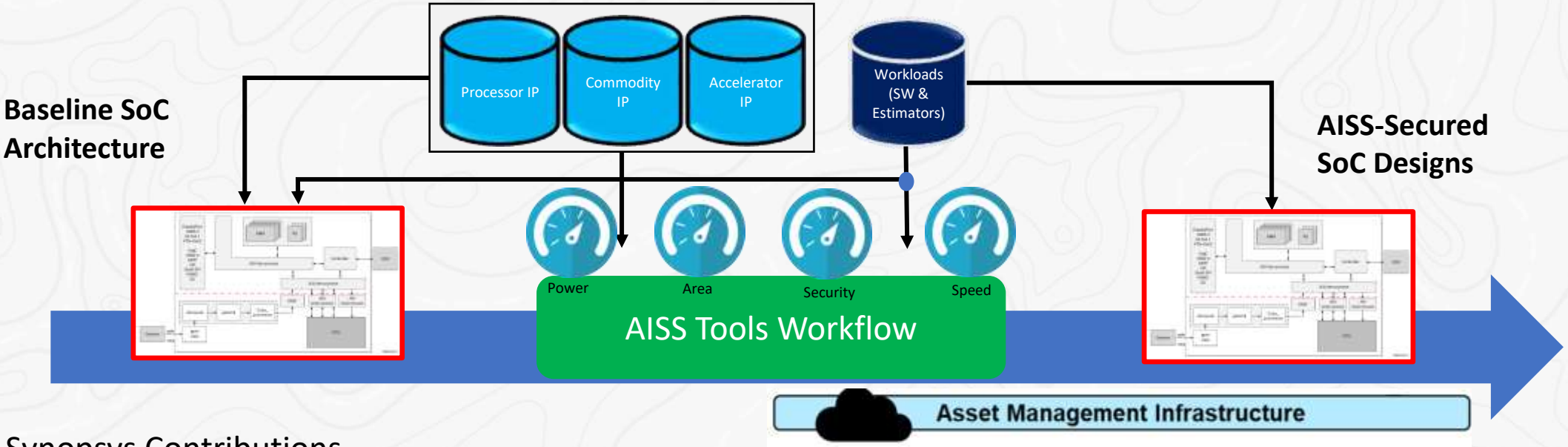
- Safety-Security alignment is one of the topics covered in it

- Approach:
  - Align safety and security requirements
  - Analyze effect of technical requirements from each discipline on the other one



AISS Vision

Automated tool flow creating SoC for meeting PPA and security metrics



Synopsys Contributions

- ✓ Counter reverse-engineering, counterfeiting, and recycling in an untrusted supply chain
- ✓ Discover “extra” circuitry that may be a trojan and report other security vulnerabilities
- ✓ Monitor system bus transactions to detect potential threats
- ✓ Register devices with central authority and prevent unauthorized systems from booting
- ✓ Embed secure root-of-trust and configurable cryptographic functions
- ✓ Add application-specific configurable security defenses balanced against PPA constraints
- ✓ Enable rapid design exploration and final SoC creation

Summary

- EDA has been rising to the challenge and providing solutions for Safety & Security
- Focus is on a holistic solution, spanning from architecture to silicon and SW products
- Existing tools and technologies are leveraged and enhanced to address design, verification and implementation challenges
- Participation in Standardization efforts and cross-industry initiatives helps creating innovative, collaborative solutions which benefit all players in the supply chain

Acknowledgement

- Synopsys would like to thank the Safety & Security experts who contributed to this tutorial:
  - Serge Leef, DARPA
  - Bala Chavali, AMD
  - Reiley Jeyapaul and Balaji Venu, arm Research

Thank You