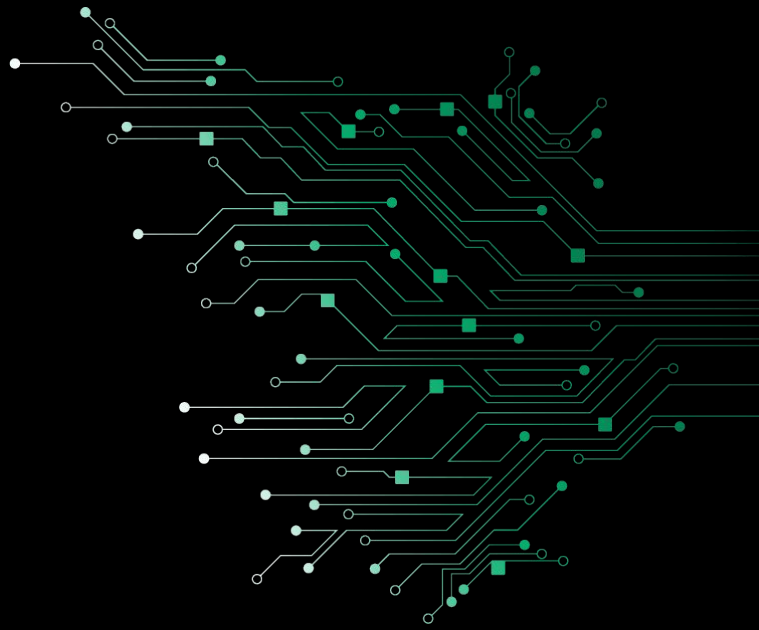
# CMP720 Embedded System Design

## WEEK 1

## 2026 Spring

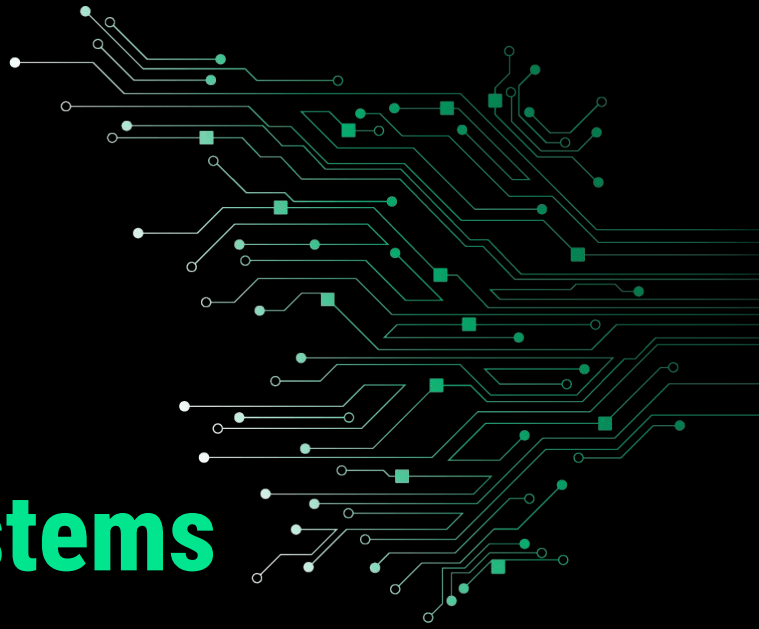
https://piazza.com/hacettepe.edu.tr/spring2026/cmp720

https://github.com/sepidi/hacettepe-cmp720-spring-2026



Challenges and Opportunities for the Future

# Secure, Reliable, Safe, Energy-Efficient, Low-Latency, Area-Aware Embedded Computing Systems



# THE PLAN FOR TODAY

**Week 1 Topics: Introduction to Embedded Systems Design and Research Directions**

Get to know each other and discuss expectations from the course.

Make an introduction to embedded system design and trending research directions.

Discuss the course syllabus and semester plan.

Explain the tracks, goals and purpose of the final projects.

# Short Bio

## Education

2025 - Postdoc in Cybersecurity, *University of Bristol, UK*
Research Project Title: Novel Framework and Methodology for Secure Internet of Things

2022 - Ph.D. in Computer Engineering, *Hacettepe University*
Thesis Title: Optimization Methods In High-Level Synthesis.

2017 - M.Sc. In Computer Engineering, *Gazi University*
Thesis Title: Internet of Things-Based Remote Healthcare Monitoring Application.

2008 - B.Sc. in Computer Science and Electrical-Electronics Engineering, *Sarajevo School of Science and Technology, BIH*; B.Sc. in Computer Science, *University of Buckingham, UK*
Thesis Title: Wavelet Thresholding of ECG Signals.

## Research Interests

Electronic Design Automation | Hardware Design Optimization | Embedded Systems Design | Computer Architecture | High Performance Computing | Internet of Things

# Asst. Prof. Dr. Selma Dilek

No dedicated office hours: please contact over Piazza - meetings can be arranged as needed.

## Selected Publications


S. Dilek, R. Smri, S. Tosun and D. Dal, "A High-Level Synthesis Methodology for Energy and Reliability-Oriented Designs," in IEEE Transactions on Computers, vol. 71, no. 1, pp. 161-174, 1 Jan. 2022, doi: 10.1109/TC.2020.3043885.

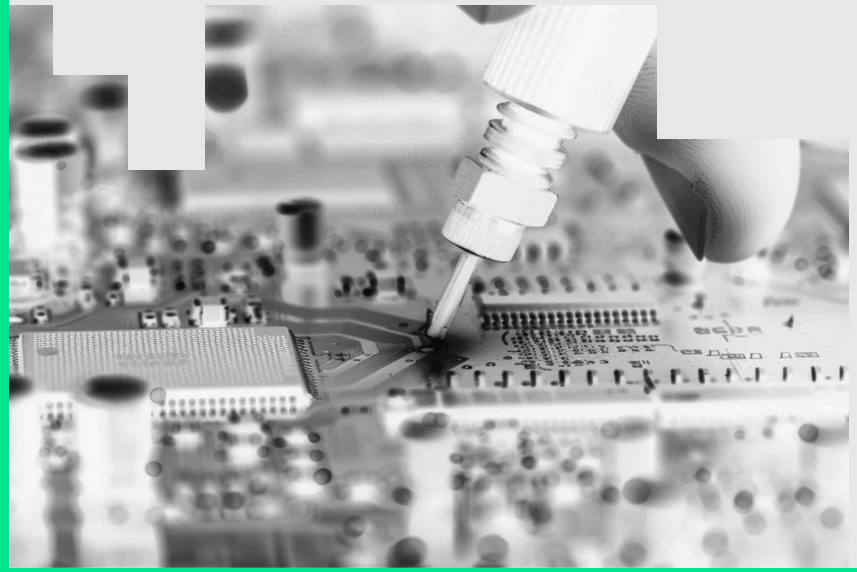
F. Nacar, A. Cakin, S. Dilek, S. Tosun, K. Chakrabarty, "Neuron grouping and mapping methods for 2D-mesh NoC-based DNN accelerators," Journal of Parallel and Distributed Computing, 193, 2024, 104949, doi: 10.1016/j.jpdc.2024.104949.

Z. Colton, A. Oracevic and S. Dilek, "Timestamp Manipulation-Based GPS Spoofing Attacks on the MAVLink 2.0 Protocol for UAV Communication: An Empirical Study," in IEEE Transactions on Communications, vol. 74, pp. 2564-2579, 2026, doi: 10.1109/TCOMM.2025.3644474.

Z. Abushaban, S. E. Yüzbaşıoğlu, S. Dilek and S. Tosun, "Optimizing Deep Learning Models for Ophthalmic Disease Detection on Resource-Constrained Devices," 2025 7th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (ICHORA), Ankara, Turkiye, 2025, pp. 1-9, doi: 10.1109/ICHORA65333.2025.11017237.

Cakin, A., Dilek, S. & Tosun, S. "Energy-aware application mapping methods for mesh-based hybrid wireless network-on-chips." Journal of Supercomputing 80, 15582–15612 (2024). doi: 10.1007/s11227-024-06062-4.

# PLEASE TELL US ABOUT YOURSELF



Name

(Under)graduate Education

Job/Role in Industry or Academia?

Hobbies and Interests

Embedded Systems You Use and Design Considerations?

What are Your Expectations From This Course?

# COMMUNICATION PLATFORM:

piazza

https://piazza.com/hacettepe.edu.tr/spring2026/cmp720

- All communication and course organization - you are responsible for following all correspondence regularly.
- Any questions: please ask via Piazza posts (not via Emails*)
- Ask publicly if relates to all students, privately ONLY if it is specific to your project.

* E-mails may end up in spam, so please, stick to the Piazza for communication.

# Semester Plan and Program (Tentative)

Theoretical / Practical Lecture Topics
Research Project Activities
Student Presentations

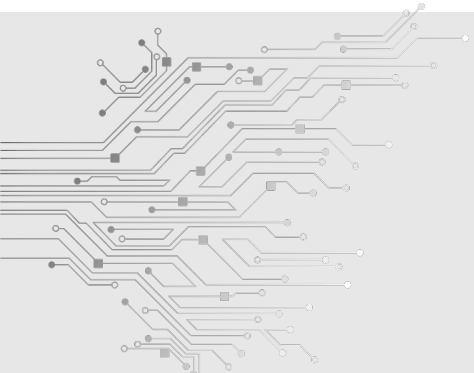
G - graded in-class work

| Week | Lecture Topics \| Research and Project Activities |
|---|---|
| 1 | Introduction - Embedded Systems Design and Research Directions |
| 2 | Specifications, Modeling & Requirements for Embedded Systems G |
| 3 | Embedded Hardware & System Architecture G \| Deadline for Project Group Formation and Final Project Topic Proposal Submission |
| 4 | Sensor & Actuator Interfacing + Timing + Drivers G \| Finalization of Topic Approvals with Instructor Feedback |
| 5 | System Software, RTOS, Scheduling & Real-Time Properties G |
| 6 | Evaluation, Validation & Testing for Embedded Systems G |
| 7 | Optimization G \| Deadline for Submission of Preliminary Literature Review and Updated Final Project Proposal with Methodology Plan |
| 8 | Student Midterm Project Presentations G |
| 9 | Student Midterm Project Presentations G |
| 10 | More Optimization / Hands-On Practice \| Deadline for Presentation Summaries and Analysis Report Submission |
| 11 | More Optimization / Hands-On Practice |
| 12 | More Optimization / Hands-On Practice |
| 13 | Student Final Project Presentations G |
| 14 | Student Final Project Presentations G |
| 15 | Deadline for Submission of Final Project Reports |

# Prerequisites for This Course

**Topics Covered by:**

- BBM231 Digital Design
- BBM234 Computer Architecture
- BBM341 Systems Programming
- BBM432 Embedded Systems
- BBM433 Microprocessors



- This course builds directly on prior coursework in digital design, computer architecture, systems programming, and embedded systems.
- Fundamental topics will not be re-taught; instead, the focus is on advanced design challenges and optimization.
- Students are expected to be well prepared and highly motivated, as this is not an introductory course, but a demanding graduate (PhD level) course intended for highly motivated students with a strong interest in embedded systems design.

# Grading Policy

## Overall Grade (100%)

- **Midterm "Exam": 50%**
- **Final "Exam": 50%**

Passing grade:
- MSc: C1 (>=65)
- PhD: B2 (>=75)

Both are structured as multi-component project-based evaluations (not written exams).

## Midterm "Exam" Grade

- **Participation + In-Class Practicum Projects: 20%**
- **Midterm Project Presentation: 40%**
- **All Presentation Summaries and Analysis Report: 40%**

## Final "Exam" Grade

- **Final Project Topic Proposal: 5%**
- **Literature Review for the Final Project and Updated Proposal with Methodology Plan: 20%**
- **Final Project Presentation: 30%**
- **Final Project Report: 45%**

## Important Rules and Grading Details

- Students are expected to have **pen/paper or alternative note taking props** (starting from Week 2) and **computers** (starting from Week 3) **with installed mandatory tools/platforms** (see next slide). The course will rely on hands-on work in class.
- In-class practicum mini-projects implementation and grading will take place during lectures; attendance is mandatory.
- The project proposals may have to be revised after the instructor's comments.
- Using LaTeX is mandatory for all report writing during this course. Templates and instructions will be provided.
- Usage of `git` version control (GitHub or GitLab accepted) is mandatory for final projects progress tracking and submission. **All submissions will be done via repositories**.

# Tools/Platforms-to-Be-Used

! Mandatory

? Optional

! RENODE™ https://renode.io/

Renode is a development framework which accelerates IoT and embedded systems development by letting you simulate physical HW systems - including CPU, peripherals, sensors, environment and wired or wireless medium between nodes.

? SimulIDE https://simulide.com/

SimulIDE is a simple real time electronic circuit simulator, intended for hobbyist or students to learn and experiment with analog and digital electronic circuits and MCUs.

? arm Virtual Hardware

https://www.arm.com/products/development-tools/simulation/virtual-hardware

Arm Virtual Hardware (AVH) scales and accelerates software development by virtualizing popular development kits, Arm-based processors, and systems in the cloud. It is an evolution of Arm modeling technology that eliminates the wait for hardware and the complexity of building and configuring board farms for testing.

! GUROBI OPTIMIZATION https://www.gurobi.com/

High-Speed Optimization Solver

! git

A free and open source distributed version control system.

Overleaf ! LaTeX https://www.overleaf.com/

Overleaf is software for running a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents.

Prism https://prism.openai.com/

Prism is a free, LaTeX-native workspace for scientists

# Textbooks and Other Useful Resources/Materials

## Textbooks

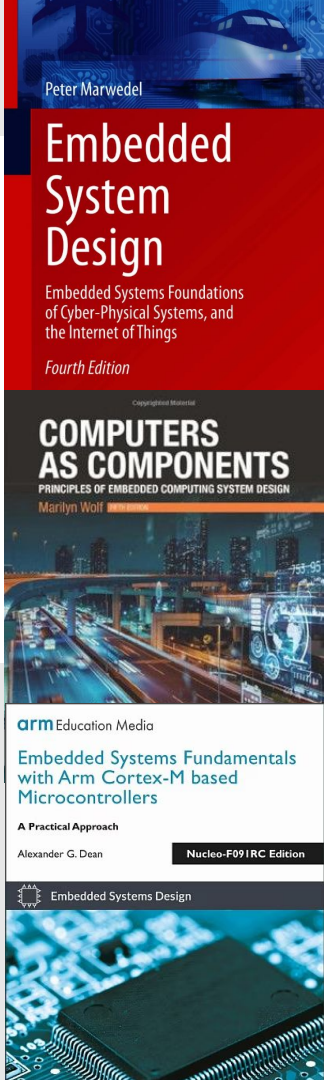
- Peter Marwedel, "*Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*," 2021, doi: https://doi.org/10.1007/978-3-030-60910-8, https://link.springer.com/book/10.1007/978-3-030-60910-8
- M. Wolf, "*Computers as components: principles of embedded computing system design*", Elsevier/Morgan Kaufmann, 2022, *https://dl.acm.org/doi/10.5555/374107*
- Alexander G. Dean, "*Embedded Systems Fundamentals with Arm Cortex-M based Microcontrollers: A Practical Approach*," https://www.arm.com/resources/education/books/efficient-embedded-systems-nucleo

## Course Materials

- GitHub Repository: **https://github.com/sepidi/hacettepe-cmp720-spring-2026**

## Other Resources and Useful Links

- D. D. Gajski, S. Abdi, A. Gerstlauer, and G. Schirner, "Embedded System Design: Modeling, Synthesis, Verification"
- Edward A. Lee and Sanjit A. Seshia, "Introduction to Embedded Systems, A Cyber-Physical Systems Approach", Second Edition, MIT Press, ISBN 978-0-262-53381-2, 2017.
- Renode Documentation: https://renode.readthedocs.io/en/latest/
- ARM Examples for Embedded Developers: https://github.com/Arm-Examples
- ARM Developer: https://developer.arm.com/
- ARM Education: https://github.com/arm-university
- Learn Optimization for Free with Gurobi: https://www.gurobi.com/learn/



# Embedded Systems Design and Research Directions

*“Self-contained, microprocessor-based computer systems typically implemented as components of a larger electrical or mechanical system.”*

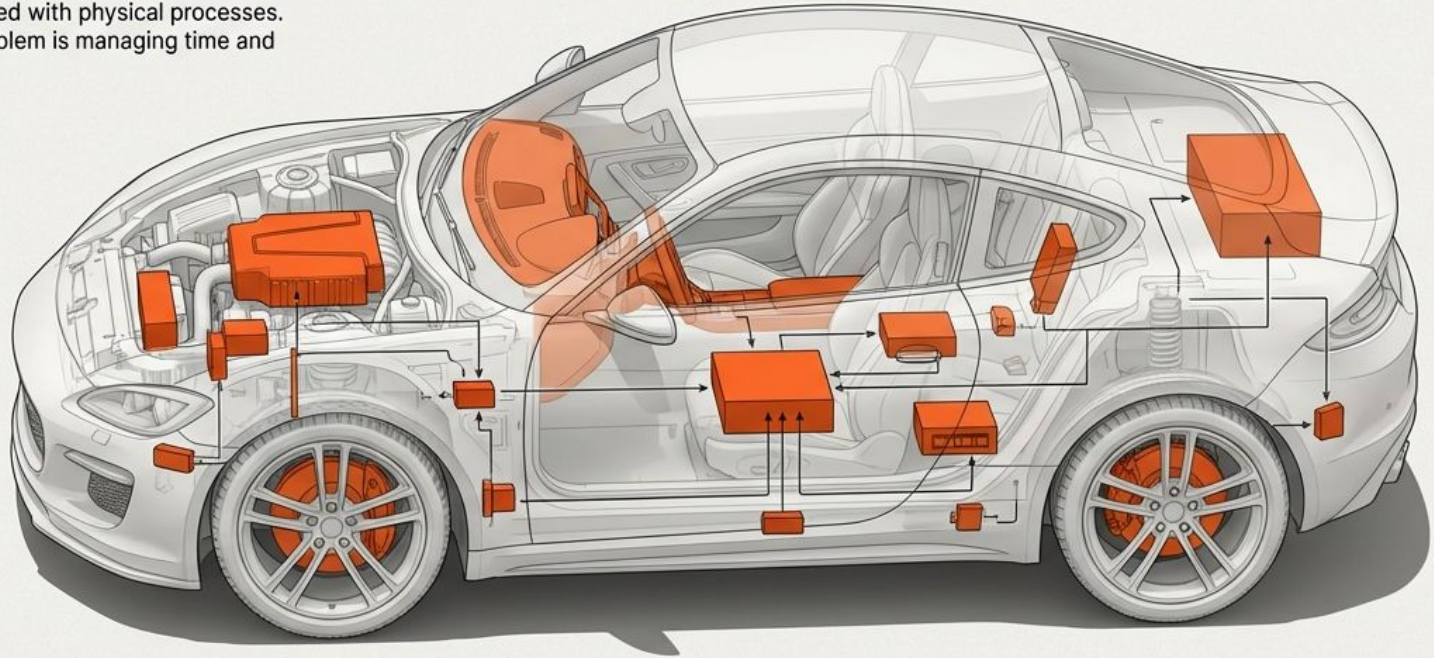
- arm

*“The design of future applications requires knowing fundamental design techniques for embedded systems.”*

# THE BUILDING BLOCK: EMBEDDED SYSTEMS

“Information processing systems embedded into enclosing products.” — Marwedel [Def 1.1]

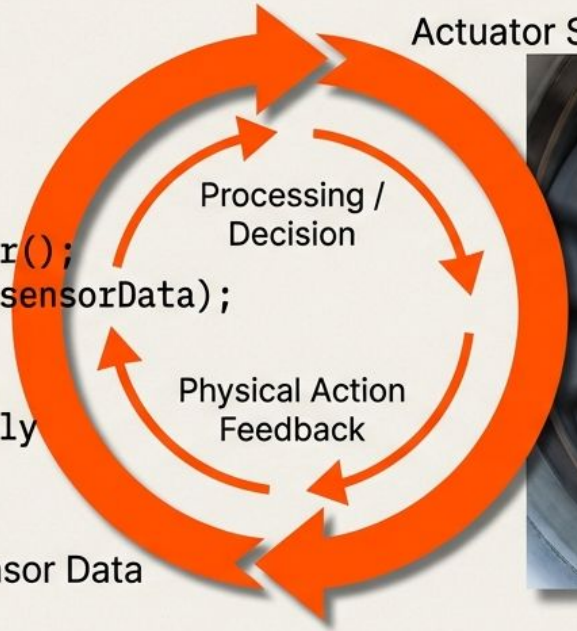
Software integrated with physical processes. The technical problem is managing time and concurrency.



# WHEN CODE MEETS PHYSICS

## CPS = ES + Dynamic Physical Environment

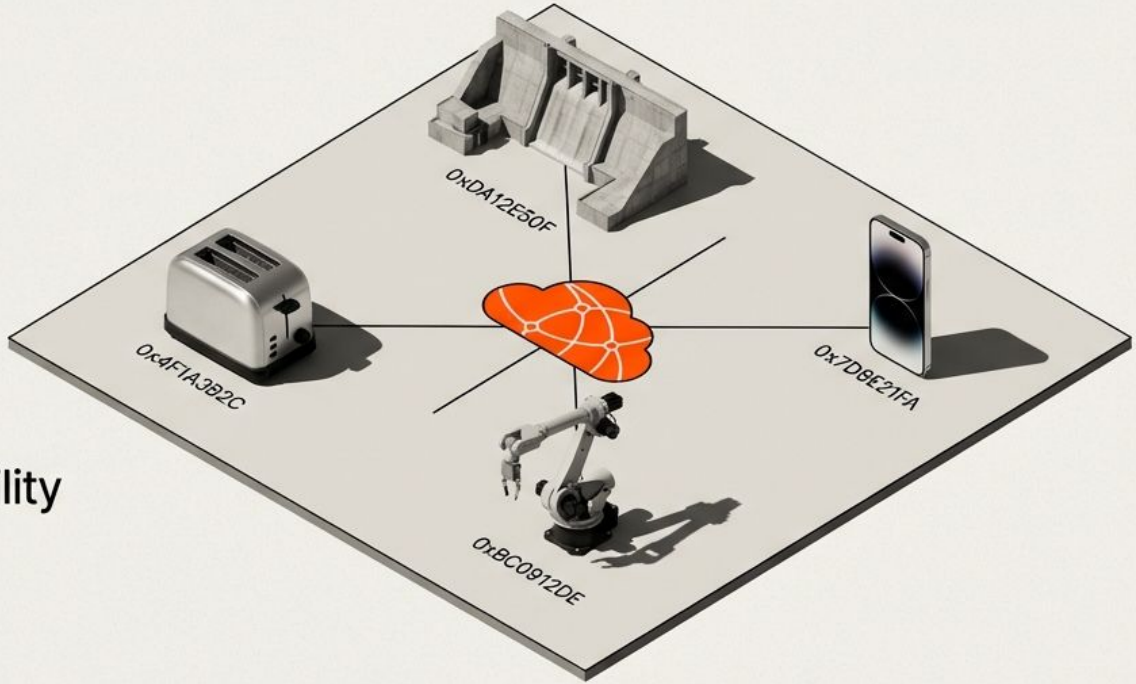
```
#include <sensor.h>
#include <actuator.h>

void controlLoop() {
  float sensorData = readSensor();
  float output = calculatePID(sensorData);
  applyActuator(output);
  logData(sensorData, output);
  // Real-time constraints apply
}
```




"Integrations of computation and physical processes." — Lee [Def 1.2]

# THE COLLECTIVE CONSCIOUSNESS: IOT

**“The pervasive presence of a variety of devices—sensors, actuators, and mobile phones—which, through unique addressing schemes, are able to interact and cooperate.” [Def 1.3]**




- Sensors: Information availability
- Actuators: Remote control
- Scale: Trillions of devices

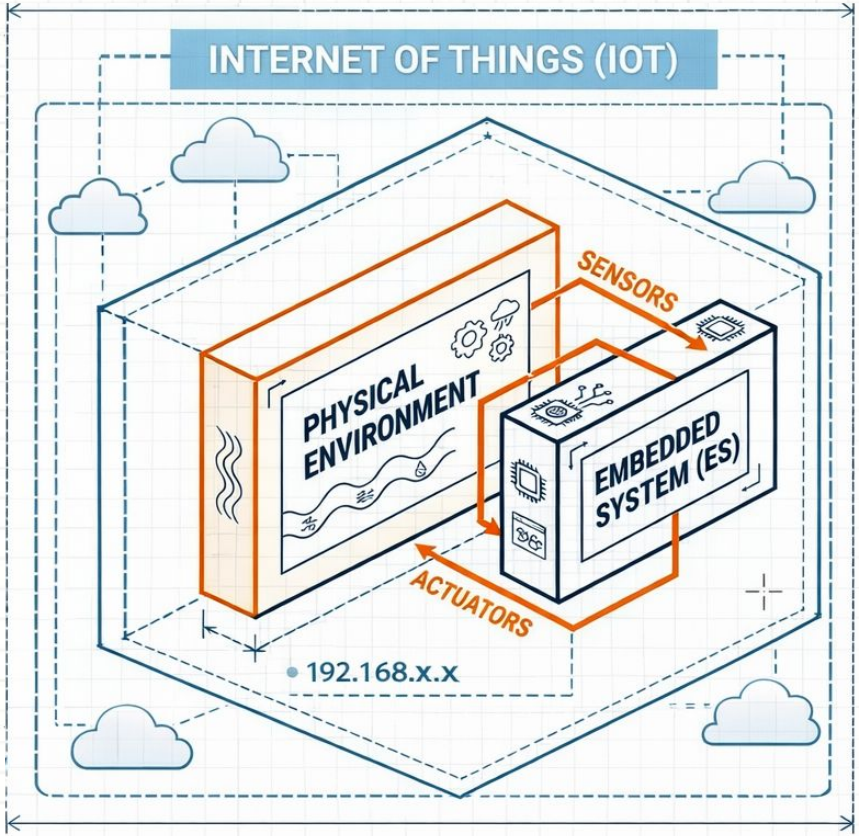
# DECODING THE TERMINOLOGY: A HIERARCHY OF CONNECTIVITY

**Embedded Systems (ES):**
Information processing systems embedded into enclosing products.

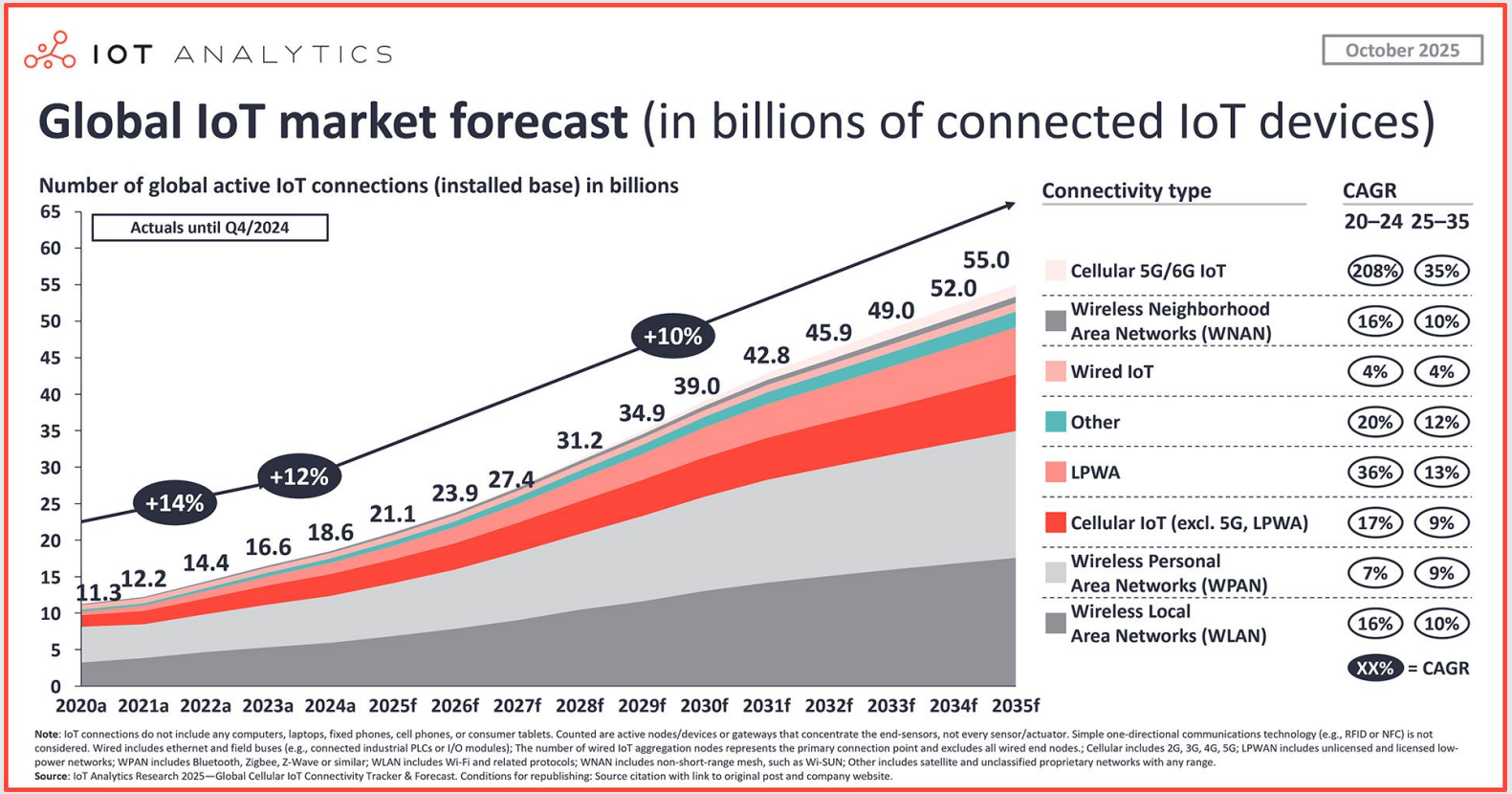
**Cyber-Physical Systems (CPS):**
Integrations of computation and physical processes.

**Internet of Things (IoT):**
Devices (sensors/actuators) that interact and cooperate via unique addressing schemes.

**CPS = Embedded System + (Dynamic) Physical Environment**
**IoT = CPS + Global Connectivity**

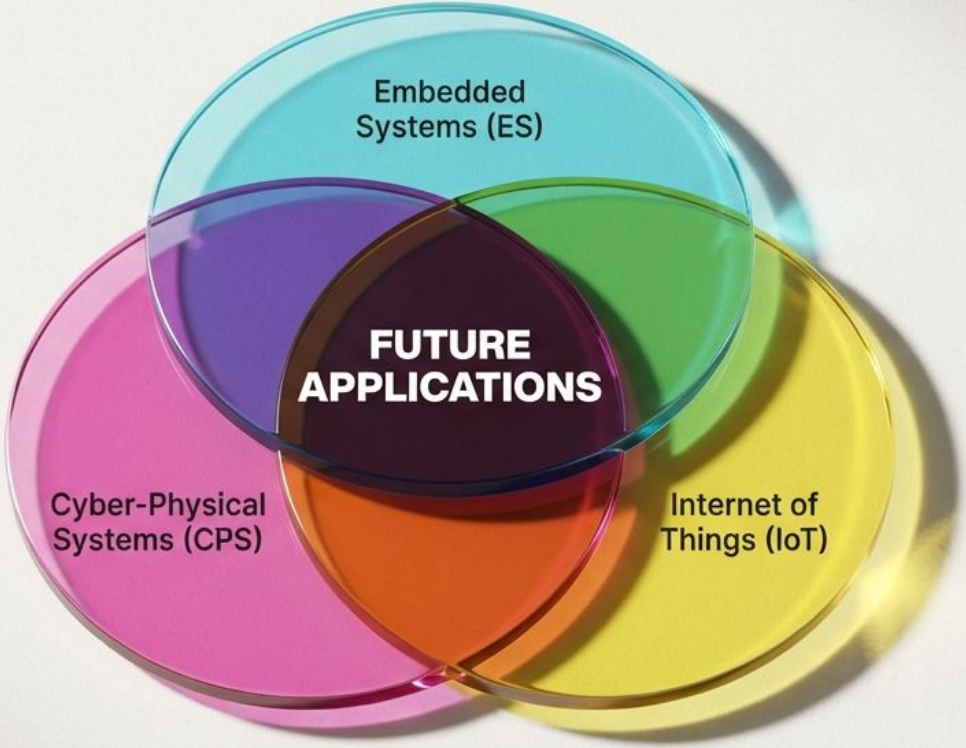


# State of IoT 2025: Number of Connected IoT Devices Growing 14% to 21.1 Billion Globally*



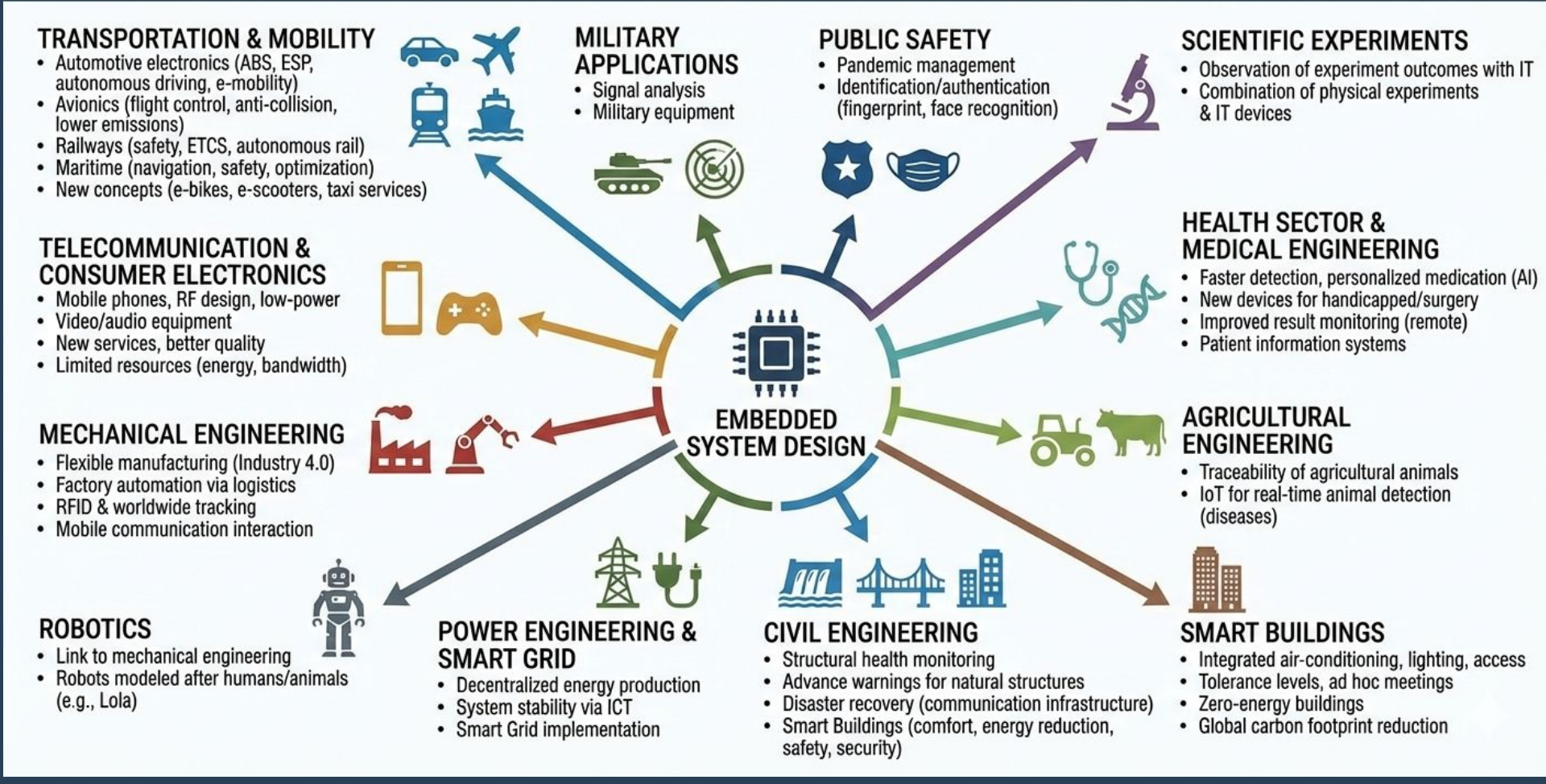

*Sinha, S. (2025, October 28). ***State of IoT 2025: Number of connected IoT devices growing 14% to 21.1 billion globally***. IoT Analytics. Retrieved from https://iot-analytics.com/number-connected-iot-devices/

# THE CONVERGENCE




It is a matter of preference whether linking physical objects to the cyber-world is called CPS or IoT. Together, they comprise the future of IT.

# IMPACT OF EMBEDDED SYSTEM DESIGN ACROSS INDUSTRIES & AREAS OF LIFE




## TRANSPORTATION & MOBILITY
- Automotive electronics (ABS, ESP, autonomous driving, e-mobility)
- Avionics (flight control, anti-collision, lower emissions)
- Railways (safety, ETCS, autonomous rail)
- Maritime (navigation, safety, optimization)
- New concepts (e-bikes, e-scooters, taxi services)

## MILITARY APPLICATIONS
- Signal analysis
- Military equipment

## PUBLIC SAFETY
- Pandemic management
- Identification/authentication (fingerprint, face recognition)

## SCIENTIFIC EXPERIMENTS
- Observation of experiment outcomes with IT
- Combination of physical experiments & IT devices

## TELECOMMUNICATION & CONSUMER ELECTRONICS
- Mobile phones, RF design, low-power
- Video/audio equipment
- New services, better quality
- Limited resources (energy, bandwidth)

## HEALTH SECTOR & MEDICAL ENGINEERING
- Faster detection, personalized medication (AI)
- New devices for handicapped/surgery
- Improved result monitoring (remote)
- Patient information systems

## MECHANICAL ENGINEERING
- Flexible manufacturing (Industry 4.0)
- Factory automation via logistics
- RFID & worldwide tracking
- Mobile communication interaction

## AGRICULTURAL ENGINEERING
- Traceability of agricultural animals
- IoT for real-time animal detection (diseases)

## ROBOTICS
- Link to mechanical engineering
- Robots modeled after humans/animals (e.g., Lola)

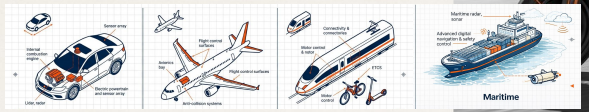
## POWER ENGINEERING & SMART GRID
- Decentralized energy production
- System stability via ICT
- Smart Grid implementation

## CIVIL ENGINEERING
- Structural health monitoring
- Advance warnings for natural structures
- Disaster recovery (communication infrastructure)
- Smart Buildings (comfort, energy reduction, safety, security)

## SMART BUILDINGS
- Integrated air-conditioning, lighting, access
- Tolerance levels, ad hoc meetings
- Zero-energy buildings
- Global carbon footprint reduction

# MOBILITY REIMAGINED

**Automotive**: Autonomous Driving, ESP, ABS
**Avionics**: Flight Control (Dependability)
**Railways**: High-speed signaling
**Maritime**: Autonomous Navigation & Safety-Critical Control
**New Mobility**: E-scooters & E-bikes






## Human-Centric Applications: Health, Comfort, and Safety

**Medical Engineering:**
Remote patient monitoring, robotic surgery, and ML-driven risk detection.

**Smart Buildings:**
Integrating access, lighting, and air-conditioning towards the "Zero-Energy Building".

**Disaster Recovery:**
Flexible communication infrastructures for when main grids fail.

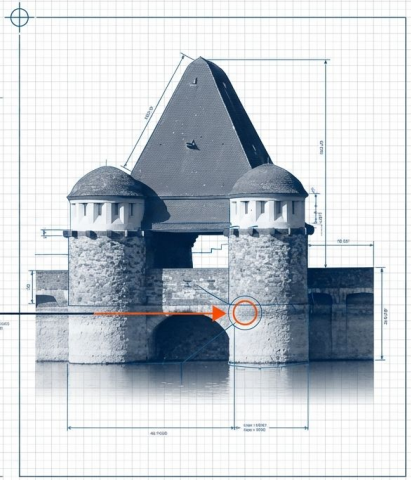
**Robotics:**
The ultimate synthesis of mechanical engineering and embedded control.





## Infrastructure & Industry: The Backbone of the Physical World

1. **Industry 4.0**
   Digitization of manufacturing using IoT for the entire lifecycle.
2. **Logistics**
   Worldwide object identification via RFID and mobile networks.
3. **Smart Grids**
   Managing stability in decentralized energy production.
4. **Civil Engineering**
   Structural health monitoring for bridges and dams to warn of collapse.





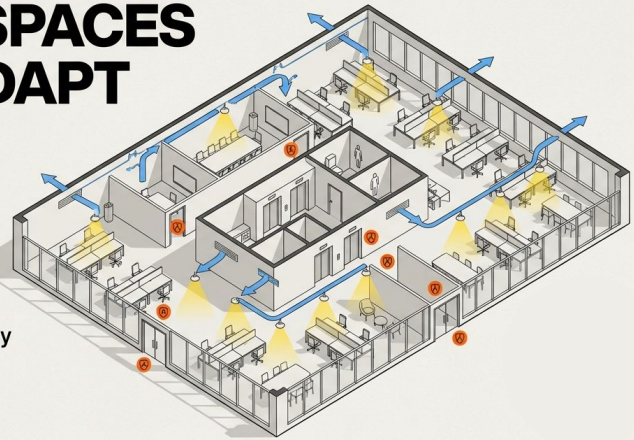


# LIVING SPACES THAT ADAPT

**Goal:**
Zero-Energy Buildings

**Integration:**
Air, Light, Access, Safety

**Response:**
Real-time adaptation to human presence



# PRESERVING LIFE: HEALTH & AGRICULTURE

- **Medical:** Personalized medication, remote monitoring.
- **Agriculture:** Traceability and real-time disease detection (e.g., Corona-19 crisis).


# THE CHALLENGE AHEAD


Great opportunity brings great complexity.
Integration requires untangling conflicting constraints.

*acatech (ed.): Cyber-Physical Systems. Driving Force for Innovation in Mobility, Health, Energy and Production (2011).

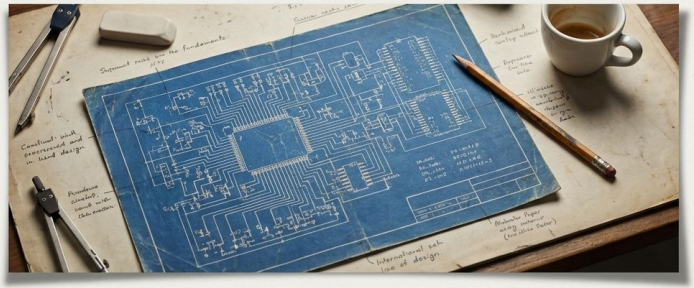
98%

# THE INVISIBLE ECONOMY

98% of all microprocessors are in embedded systems.
The value is not the chip. The value is the enabled product.

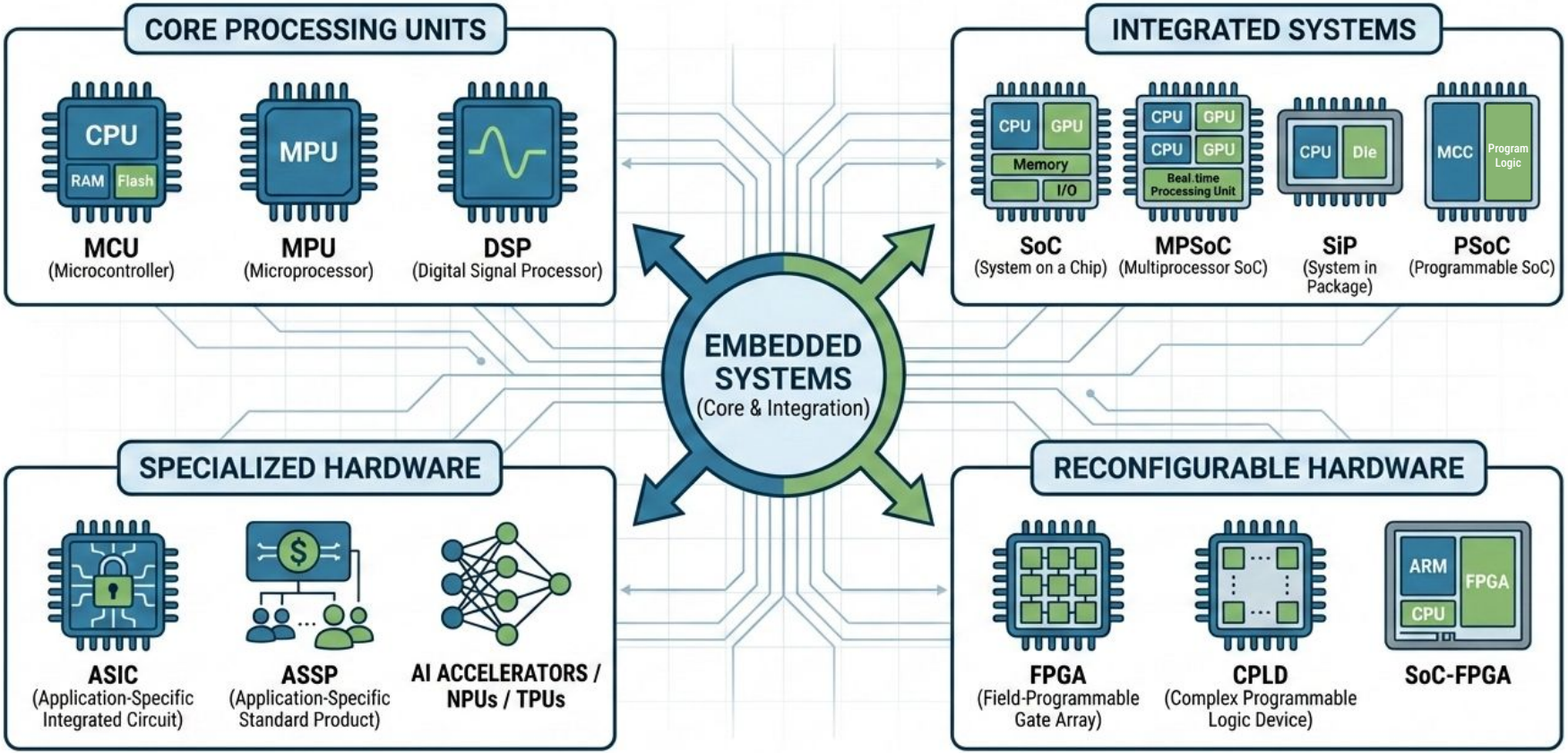
# THE DESIGNER'S MANDATE

We must look beyond the code and master the **fundamentals: Hardware, Software, Time,** and **Concurrency**.

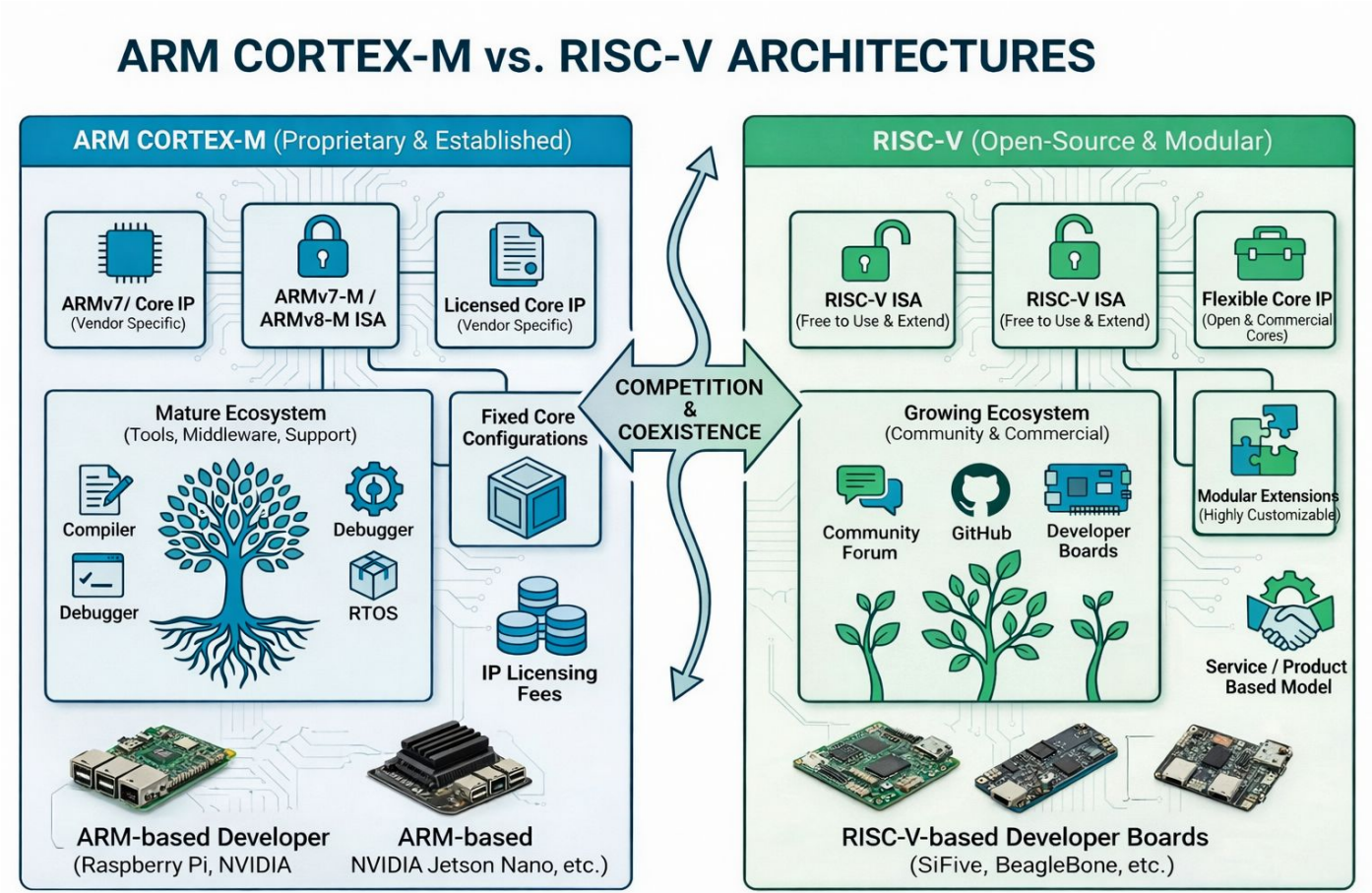


**THE FUTURE IS BEING BUILT NOW. IT IS BUILT ON EMBEDDED SYSTEM DESIGN.**

# Embedded Computing Landscape and System Taxonomy

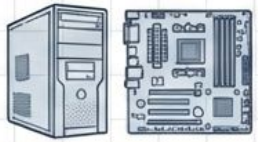
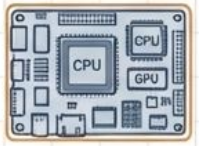
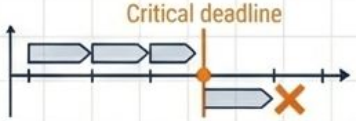
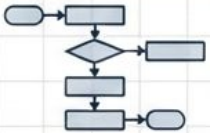
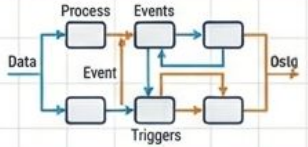


# Popular Embedded System ISAs




More on this in Week 3…

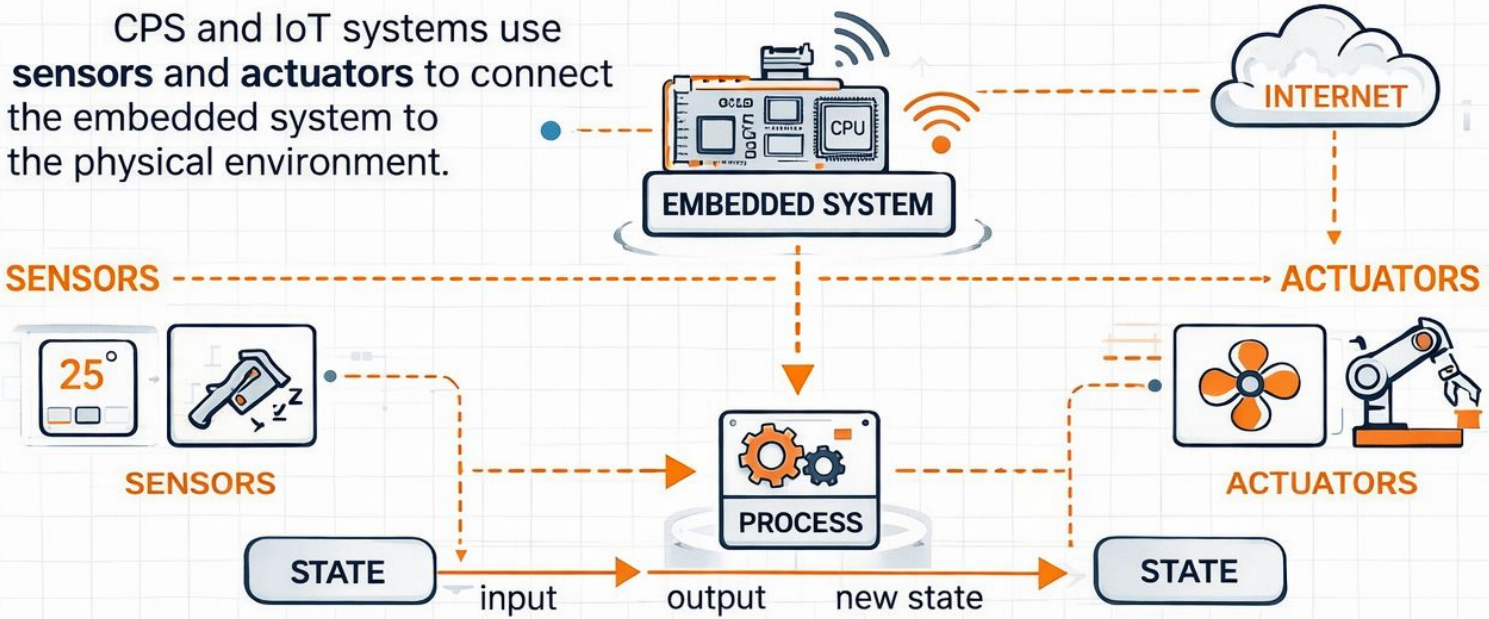
# DISTINCT CHARACTERISTICS: THE DESKTOP VS. THE EMBEDDED REALITY

| | GENERAL PURPOSE PC | EMBEDDED SYSTEM |
|---|---|---|
| **ARCHITECTURE** | Homogeneous (x86), Expandable  | Heterogeneous (SoC), Compact   |
| **REAL-TIME** | Best Effort (Average Performance)  | Strict Constraints ("Late is Wrong")   |
| **USER INTERFACE** | Keyboard, Screen, Mouse  | Dedicated (Pedals, Buttons) or Invisible  |
| **SOFTWARE** | Sequential (C, Java)  | Concurrent, Reactive Models   |

**KEY INSIGHT** **An embedded system is 'Dedicated'—it runs specific software for a specific task, optimizing efficiency by eliminating unused resources.**

# COMMON CHARACTERISTICS OF CPS & IOT SYSTEMS

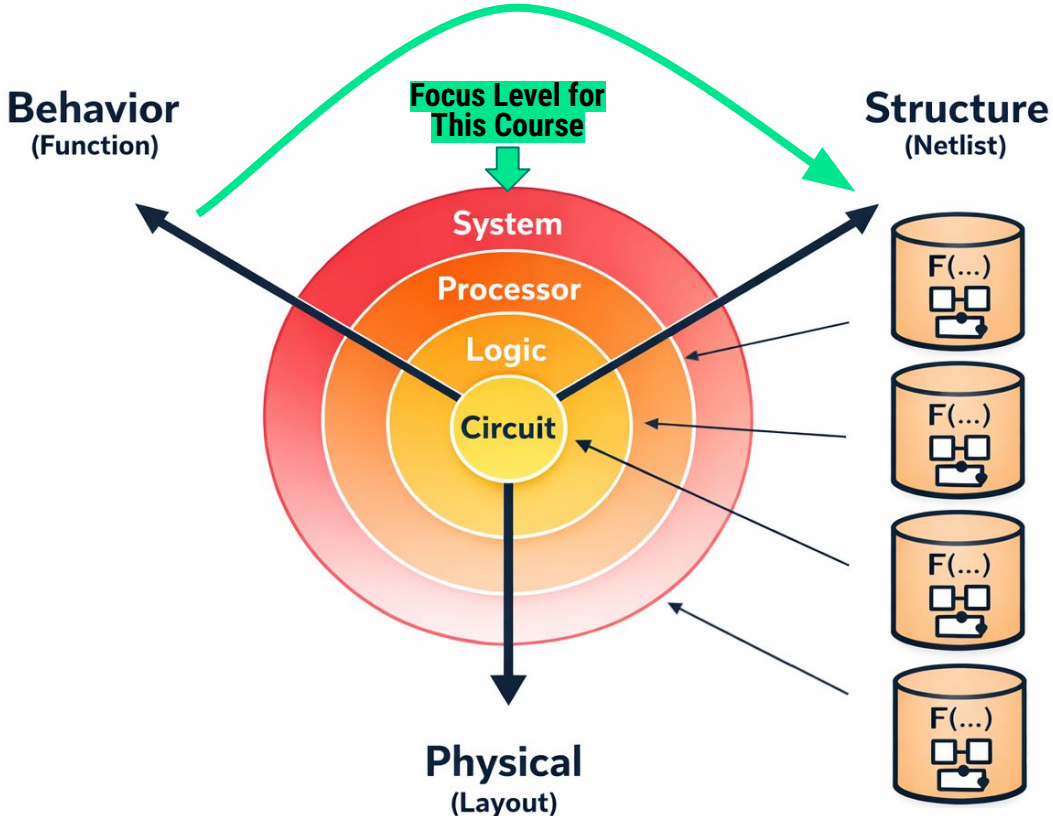
CPS and IoT systems use **sensors** and **actuators** to connect the embedded system to the physical environment.




## REACTIVE SYSTEMS:

Continuously interact with the environment at its **pace,** in a perpetual input-process-output loop.

# Embedded System Design Abstraction Levels (Y Chart, 1980s)

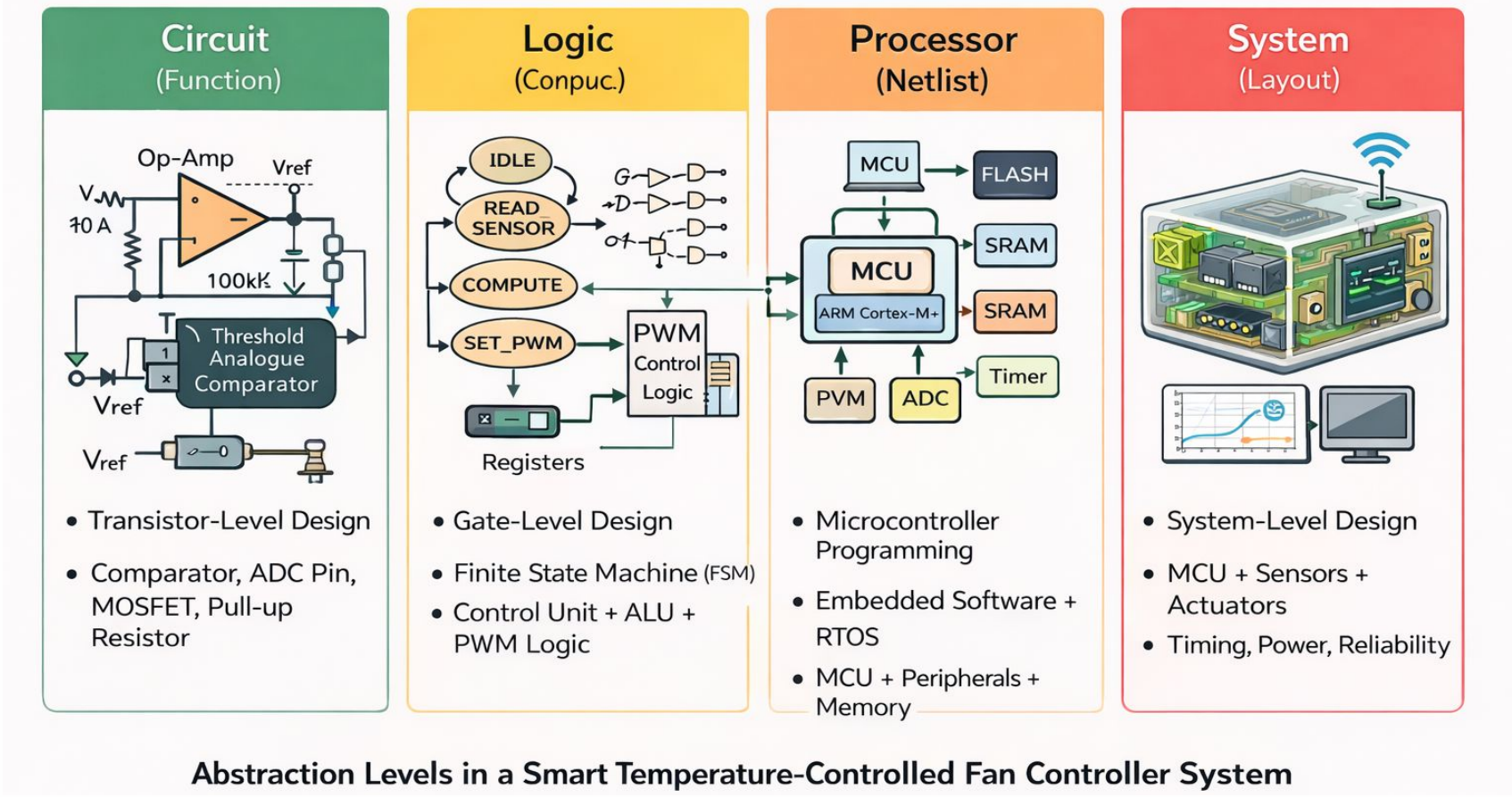



**3 design views:**
- Behavior (functionality)
- Structure (netlist)
- Physical (layout)

**4 abstraction levels:**
- Circuit
- Logic
- Processor
- System

**5 component libraries:**
- Transistor
- Logic (standard cells)
- RTL (ALUs, RF, Memories,…)
- Processor (standard, custom)
- System (multi-cores with NoCs)

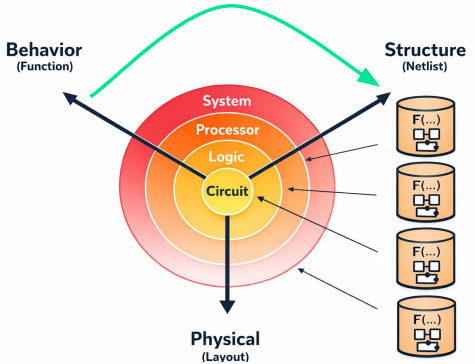
# Embedded System Design Abstraction Levels Illustration



# System Abstraction Level

**Design of multi-core systems:**
- Computation components
- Communication components

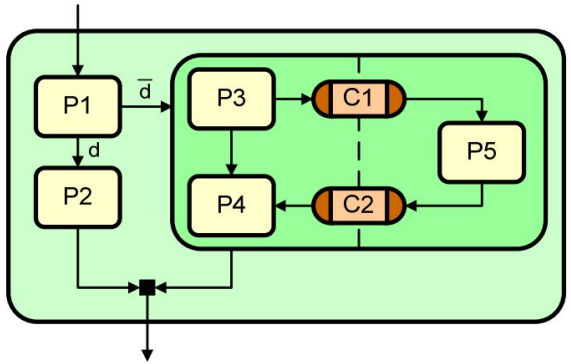
**Synthesis:**
- From behavior specification: Models of Computation (MoCs)
- To system architecture
- With system SW and HW
- Synthesis on system level




## System Behavioral Model

**Many different MoCs:**
- Process-based models
- State-based models

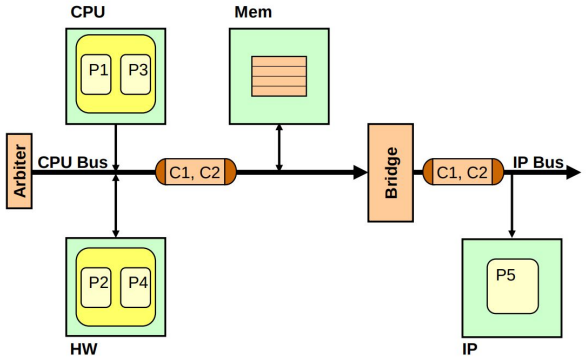
**Universal model:**
- Processes
  - Sequential/parallel, hierarchical
- Channels
  - Message-passing




## System Structural Model

**Set of computational components:**
- Processors, IPs, custom HW components
- Memories

**Set of communication components:**
- Buses, bridges, arbiters, transducers, interfaces

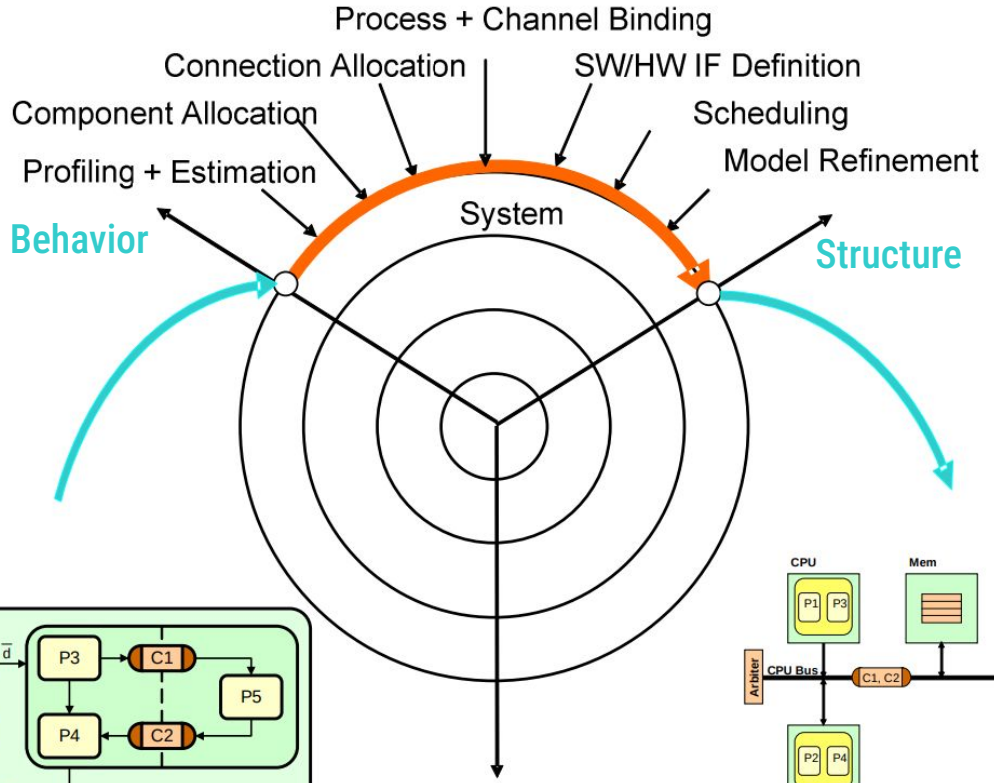


# System Synthesis

**Several tasks: different sequences possible**

- Different MoCs, different libraries, different features, different platforms
- Different tools, different metrics, different quality
- *"Synthesis is the process of generating the description of a system in terms of related lower-level components from some high-level description of the expected behavior."*

  - P. Marwedel

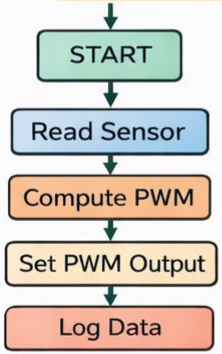


# Embedded System Design Views Illustration

## Smart Temperature-Controlled Fan Controler System

### Behavior
(Function)

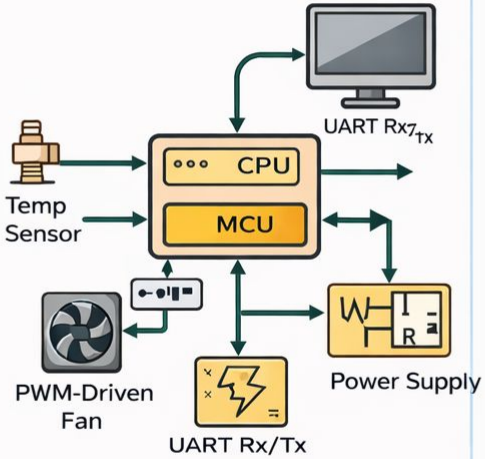
```
every 100ms:
read_sensor(temp)
pwm – compute_contol(temp)
set_pwm_output(pwm)
log_uart(temp)
```




If temp>30℃, increase fan speed.

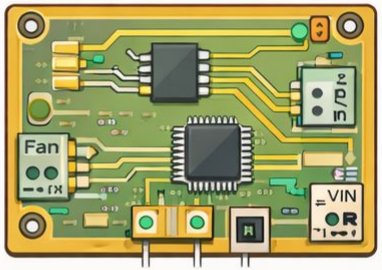
- Temperature control algorithm + timing constraints

### Structure
(Netlist)




- MCU + ADC + PWM + UART + Fan

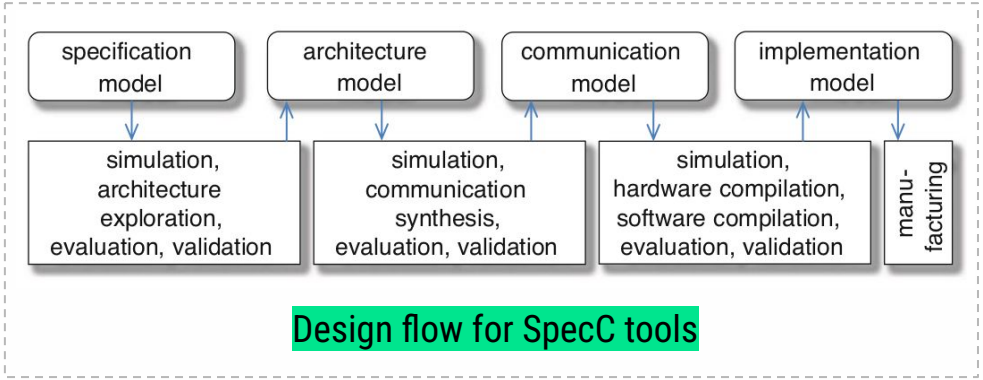
### Physical
(Layout)



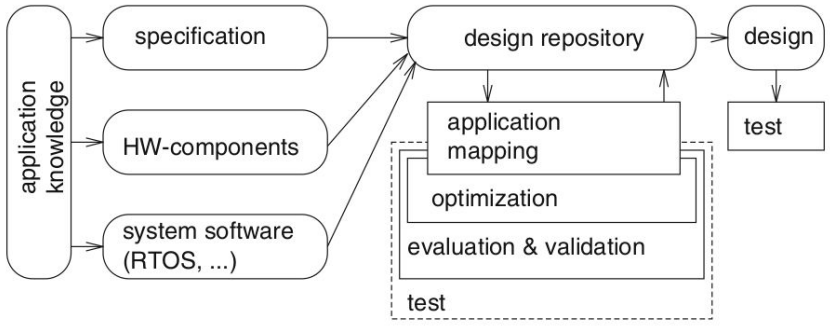

- PCB layout + thermal design power routing

# Design Flows

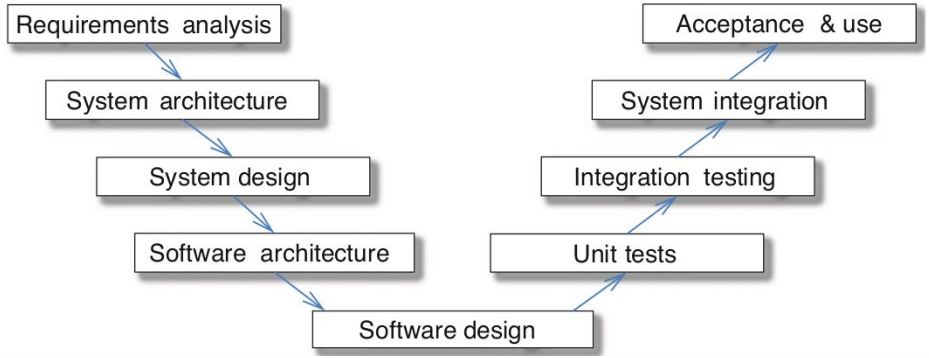
- The design of the considered systems is a rather complex task, which has to be broken down into a number of subtasks to be tractable.
- These subtasks must be performed one after the other and some of them must be repeated.




Design flow for SpecC tools

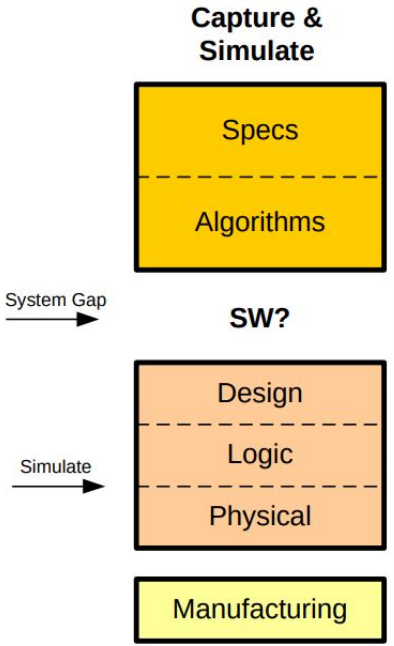



Simplified design information flow



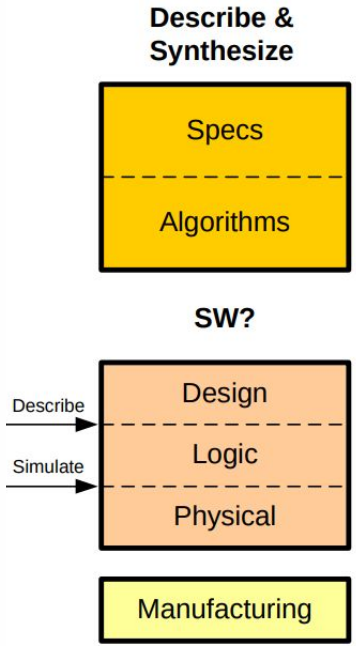

Design flow for the V-model

# Evolution of Embedded System Design Flow



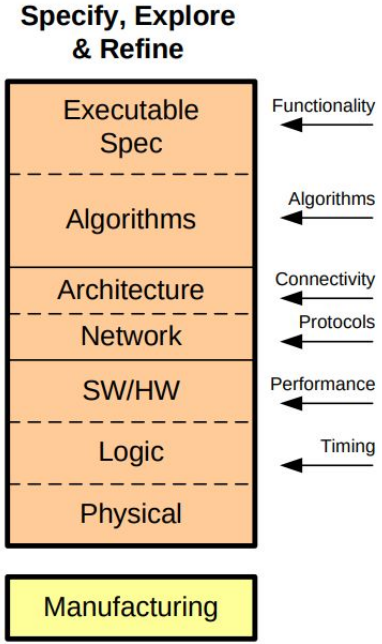

1960's

- System gap between SW and HW
- Simulation at the end of design
- Manual design, no automation




1980's

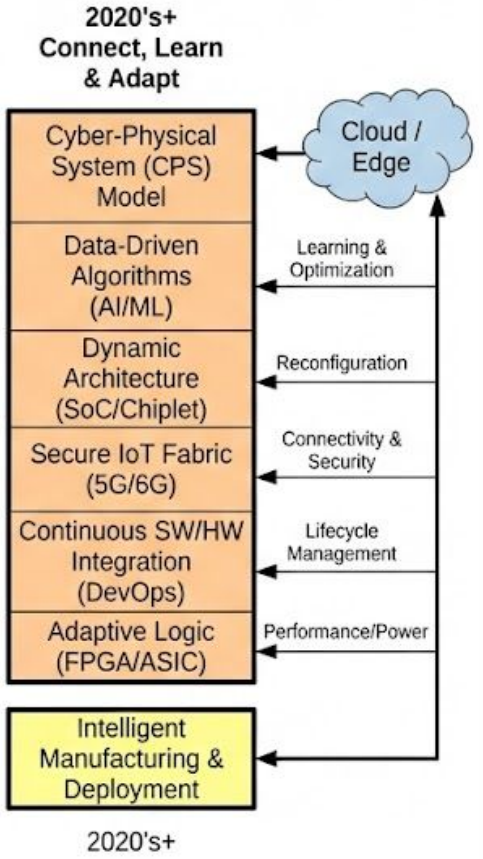
- Simulation before and after synthesis
- Designers describe just functionality, tools synthesize structure
- System gap still persists



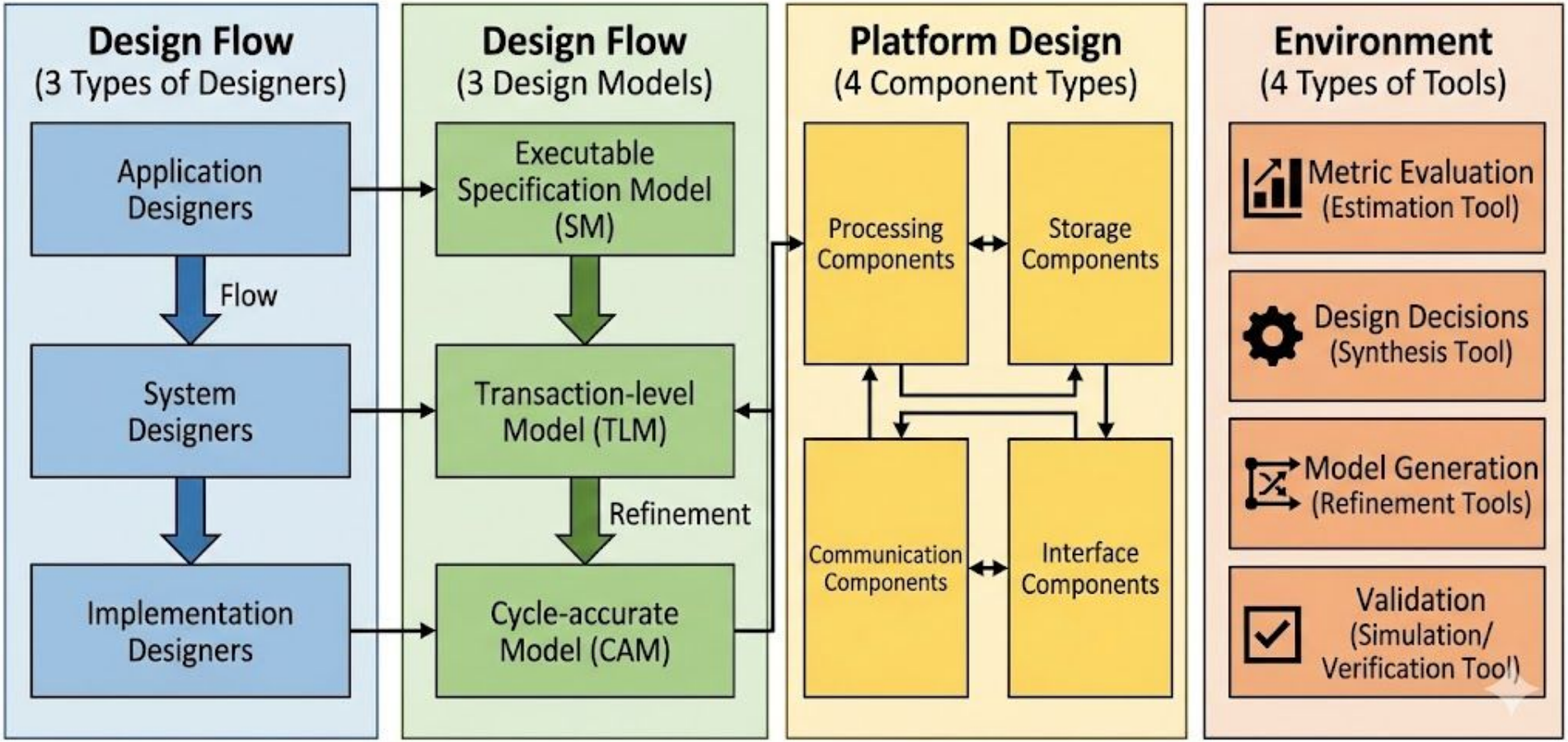

2000's

- Many design levels and models
- Many metrics for validation
- Solution: Step-by-step strategy
- For each model explore design decisions
- Refine model after exploration
- Refined model = specification for the next level

# Evolution of Embedded System Design Flow (Cont.)

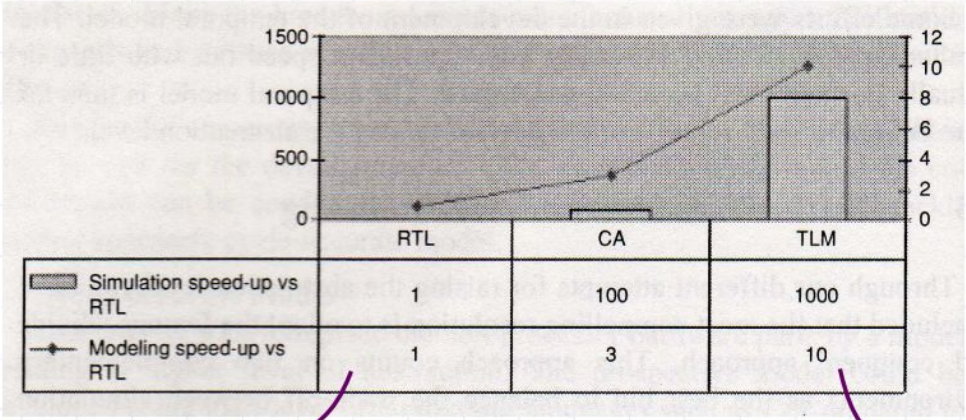



# Necessary Models, Platforms and Tools



# *Efficiency of Modeling Settings*

- Comparisons of RTL, Cycle-Accurate, and TLM models in terms of simulation speed and modeling efforts




Fine-grain simulation at the expense of slower speed and later availability

Early architecture exploration and SW development at lightweight development effort

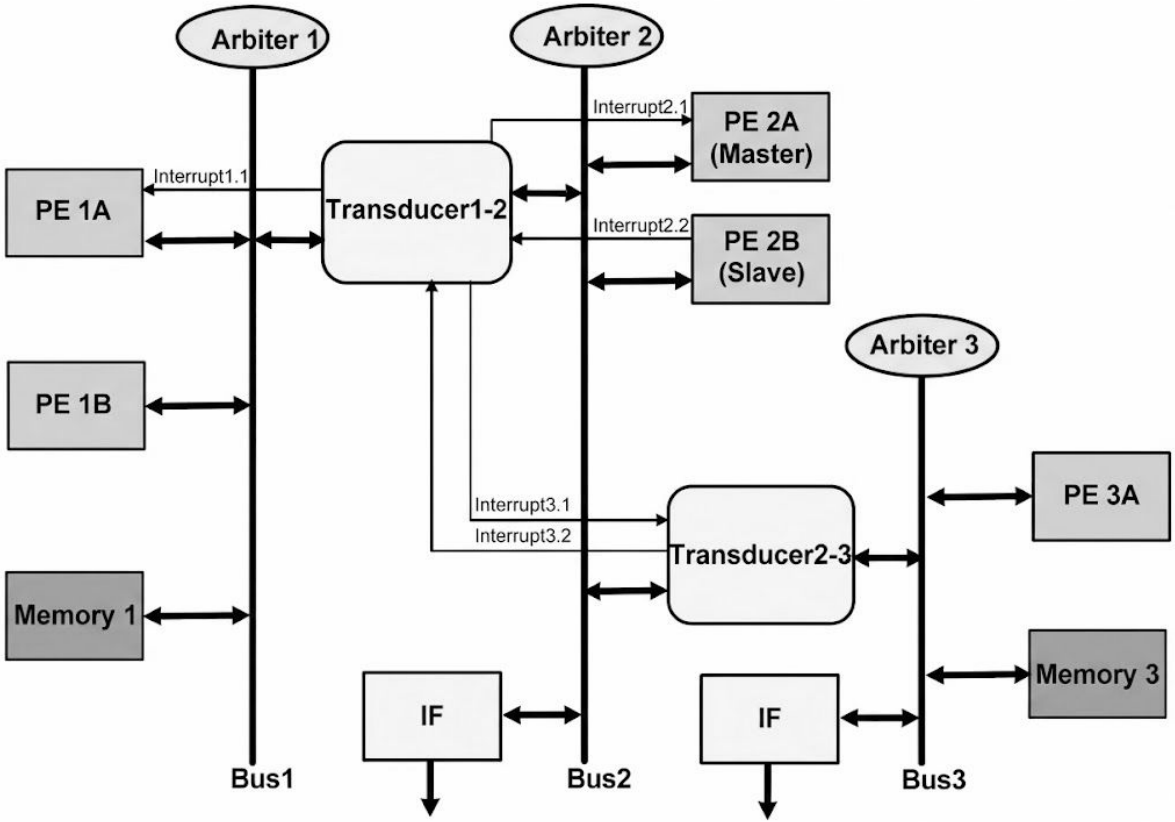
# General Platform Architecture

**Processing components (PEs):**
- Standard and custom processors

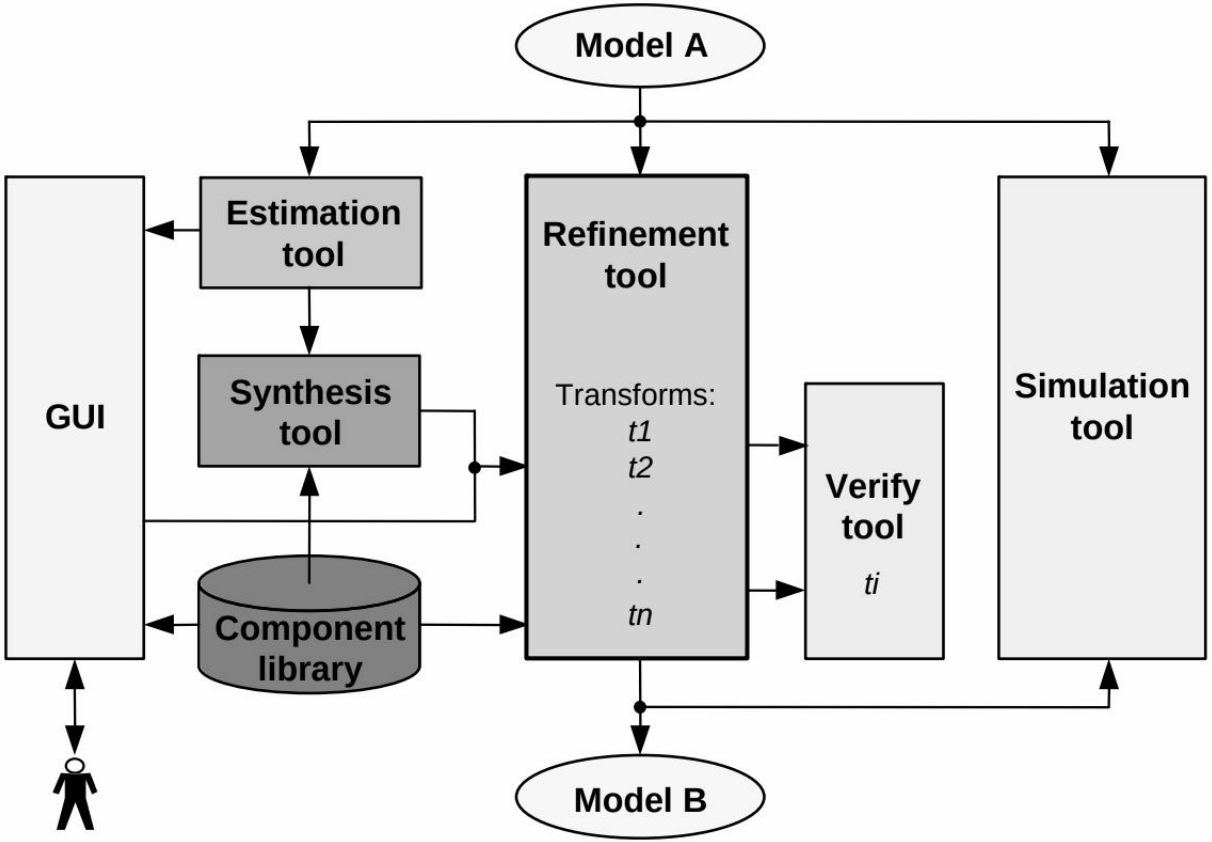
**Storage components:**
- Local and global memories

**Communication components:**
- Bridges, transducers
- NoCs

**Interface components:**
- Arbiters
- Controllers
- DMAs, UARTs



# Model Refinement Tools




**Don’t wait until the end to test — refine + test + verify in loops.**

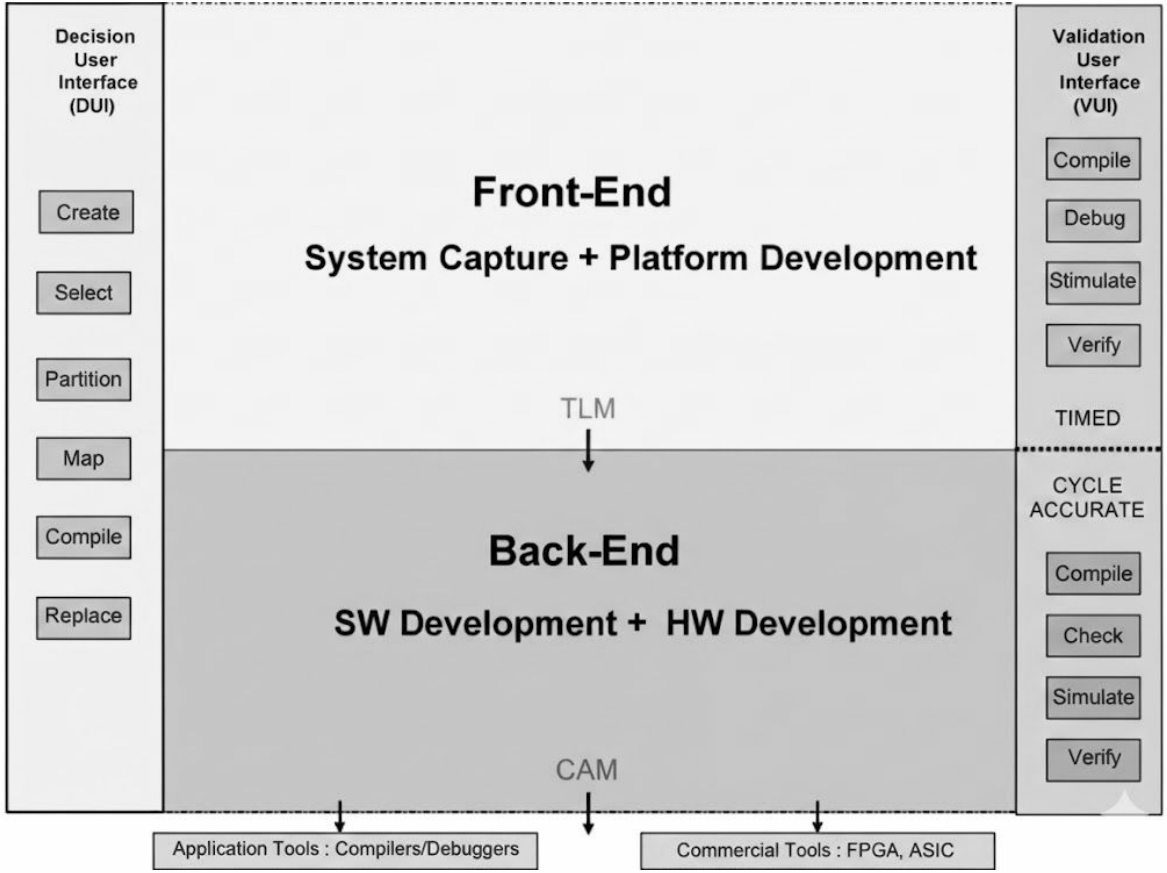
Examples in industry:

- System-level modeling: SystemC/TLM, Simulink/Stateflow, SCADE
- Virtual prototyping: Arm Virtual Hardware (AVH), Synopsys Virtualizer, Cadence VSP
- High-Level Synthesis: Catapult HLS, Vitis HLS
- Verification & Formal: VCS, Questa, JasperGold
- Emulation / Acceleration: Palladium, Veloce

# General Environment for System Design

**Front-End**
- for application developers
- for testing product concepts

**Back-End**
- for SW and HW development
- product prototyping



# Embedded Systems Challenges and Design Requirements/Constraints

- **Dependability is mandatory:** security, confidentiality, safety, reliability, repairability, availability
- **Hard real-time constraints:** guarantees required (average-case is unacceptable)
- **Physical ↔ cyber mismatch:** continuous vs discrete, hybrid dynamics, sampling limitations
- **Concurrency & nondeterminism:** integration introduces unpredictable timing effects
- **Resource constraints:** energy, time, memory, cost drive design decisions
- **Data + ML integration:** learning under resource constraints
- **Heterogeneity & compositional design:** multi-vendor components must cooperate reliably
- **Beyond technical issues:** legal, social, environmental impacts

# The Dependability Requirement: When Failure Is Not an Option

**Definition:** A system is dependable if it provides intended service with high probability and causes no harm.

**Safety:**
Absence of unacceptable risk or harm (Functional Safety).

**Security:**
Protection of confidentiality, integrity, and availability against external attacks.

**Reliability:**
Probability of no failure during operation (internal malfunctions).

**Availability:**
Probability the system is ready when needed (Reliability + Repairability).

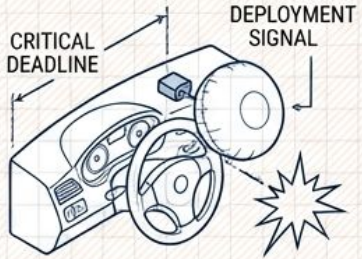
**Making the system dependable must not be an after-thought.**
**It must be designed in from the start.**

# The Tyranny of Real-Time: A Flaw in the Core Abstraction

**Definition:** A guaranteed system response has to be explained without statistical arguments.

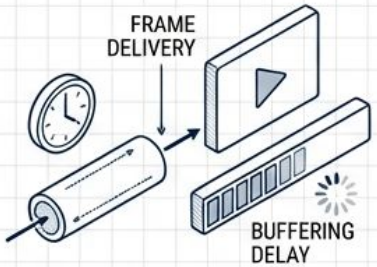
**Hard Real-Time:**

Missing the deadline results in catastrophe. (e.g., Airbag deployment).



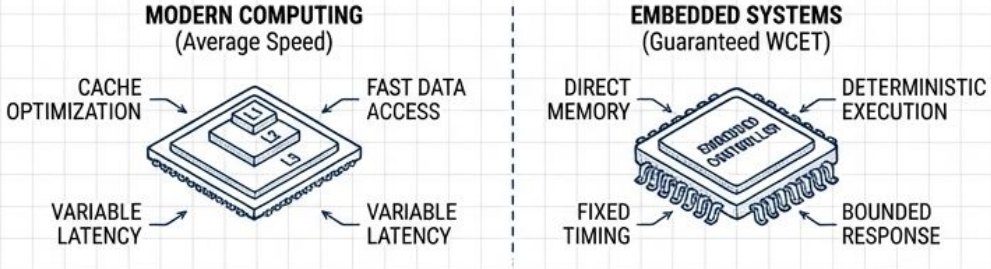

**Soft Real-Time:**

Missing the deadline degrades quality. (e.g., Video buffering).




## The Conflict:

Modern computing uses caches to improve average speed. Embedded systems need guaranteed **Worst Case Execution Time** (WCET).




"The lack of timing in the core abstraction (of computer science) is a flaw." — Edward Lee

# The Cyber-Physical Mismatch: Digital Logic vs. Physical Reality

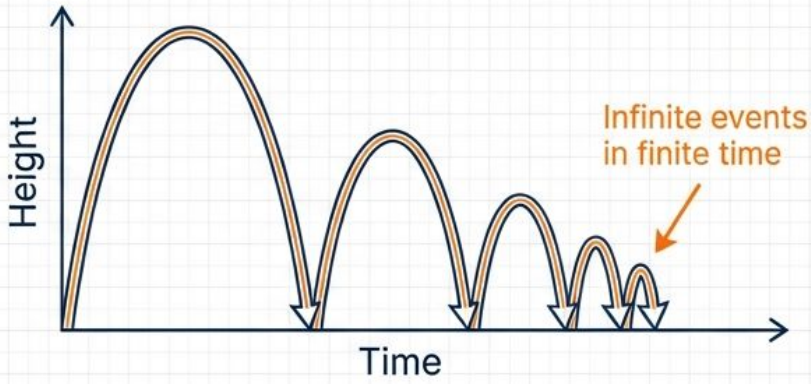
**The Friction:**

Physical World: Continuous time, Concurrent, Inexact, Real numbers.

Cyber World: Discrete time, Sequential, Approximate, Finite representations.

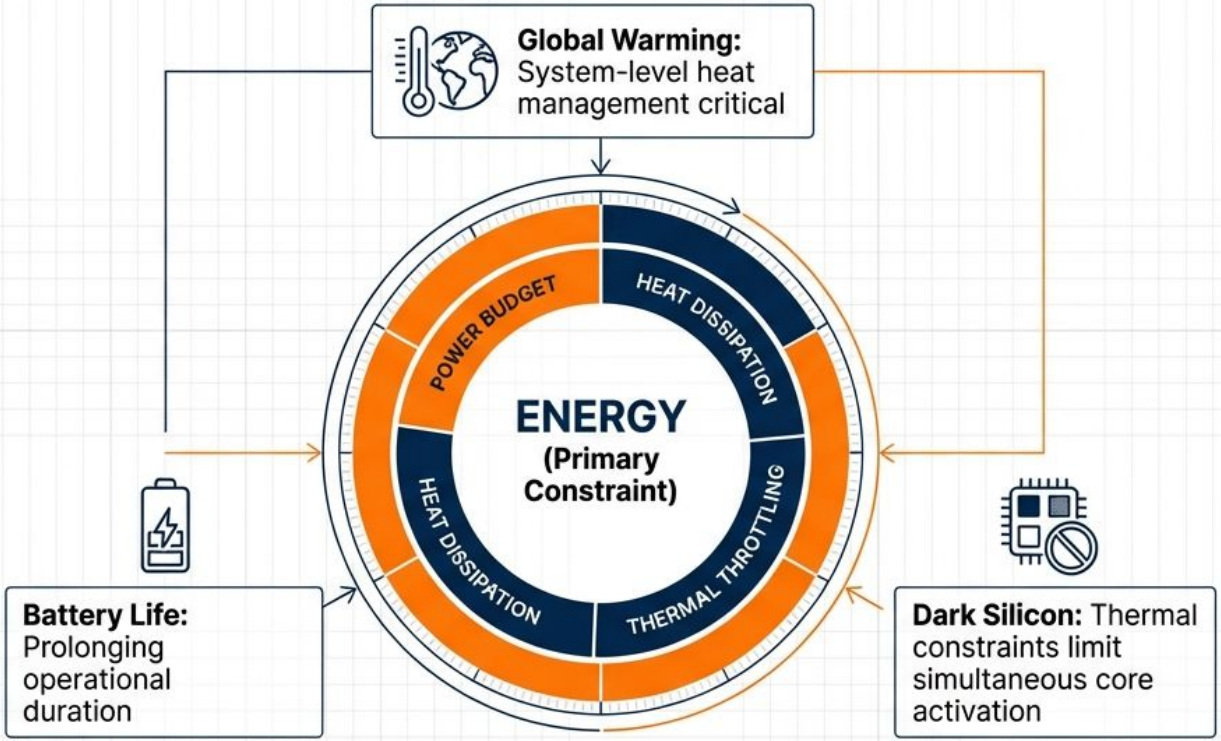
**Consequence:** Physical quantities are merely approximated in a computer.

**The Zeno Effect (Simulation Error):**




Digital computers cannot handle the infinite; they must approximate.

# Resource Efficiency: Optimizing for Scarcity




## OTHER SYSTEM CONSTRAINTS

**Code Size:** Critical for Systems on a Chip (SoC)

**Cost & Weight:** Consumer electronics demand minimum hardware

## THE TRADE-OFF:

Designers must balance software flexibility with custom hardware efficiency.

# When Design Requirements Are an Afterthought: Lessons from Real-World Failures

# Therac-25 Radiation Therapy Accidents (1985–1987)

## Domain: Medical CPS

**Impact:** At least **6 patients received massive radiation overdoses**, several died.
These accidents **highlighted the dangers of unsafe software control, shared-memory misuse, and inadequate system-level safety in medical CPS.**

### What went wrong

- Software race conditions due to **concurrency**
- Shared memory variables updated without proper synchronization
- Inconsistent internal state due to memory/state corruption
- No hardware safety interlocks (removed to save cost)
- Software assumed *impossible states* would never occur
- No effective fault detection, isolation, or redundancy

### Ignored challenges

- ❌ Safety
- ❌ Reliability
- ❌ Concurrency management
- ❌ Dependability must not be an afterthought
- ❌ Human–machine interface

📌 *Classic example of shared-memory concurrency failures replacing hardware safety without equivalent guarantees.*

# Space Shuttle Challenger Explosion (1986)

## Domain: Aerospace CPS

**Impact:** 7 astronauts killed.
This accident **showed that disregarding physical operating conditions can lead to catastrophic safety failures.**

### What went wrong

- O-ring seals failed at low temperatures
- Engineers warned about temperature risks, but model assumptions were violated
- Management ignored uncertainty and risk evidence

### Ignored challenges

- ❌ Safety
- ❌ Mismatch between physical and cyber models
- ❌ Assumptions about operating environment
- ❌ Organizational dependability

📌 *Even perfect designs fail if assumptions are wrong or violated.*

# Space Shuttle Columbia Disaster (2003)

## Domain: Aerospace CPS

**Impact:** 7 astronauts killed.
This accident **highlighted the dangers of ignoring fault detection of safety-critical systems**.

### What went wrong

- Foam strike during launch damaged heat shield
- Sensor data was insufficient to confirm damage
- No recovery or repair strategy was built in

### Ignored challenges

- ❌ Reliability
- ❌ Repairability
- ❌ Fault detection
- ❌ Design for failure and recovery

📌 *Availability and safety collapse without repairability.*

# Ariane 5 Flight 501 Explosion (1996)

## Domain: Embedded real-time control

**Impact:** Rocket self-destructed 37 seconds after launch, ~$370M loss
This accident **highlighted how invalid numeric assumptions and insufficient HW/SW co-design can cause catastrophic failures in safety-critical systems.**

### What went wrong

- 64-bit floating-point value converted to 16-bit integer → overflow
- Code reused from Ariane 4 without revalidating assumptions
- Exception handling disabled for performance reasons

### Ignored challenges

- ❌ Reliability
- ❌ Safety
- ❌ Mismatch between physical and cyber models
- ❌ Real-time exception handling

📌 *Reuse without context validation is dangerous.*

# Mars Climate Orbiter Loss (1999)

## Domain: Space CPS

**Impact:** $125M spacecraft lost
This accident **highlighted how inadequate interface specification and verification can result in catastrophic system failure.**

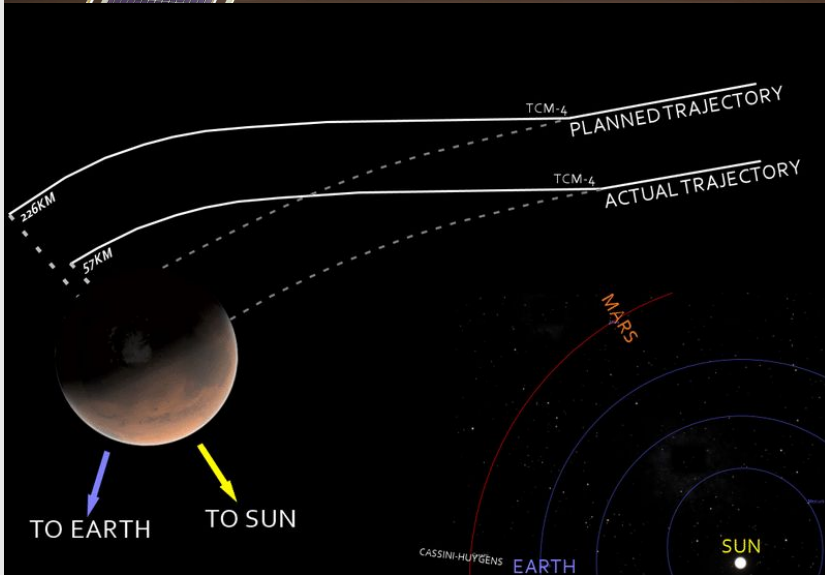
### What went wrong

- One subsystem used imperial units, another metric
- No automated consistency checks

### Ignored challenges

- ❌ Model mismatch
- ❌ Interface specification
- ❌ System integration
- ❌ Verification

📌 *Cyber–physical systems may fail at interfaces.*



# Patriot Missile Failure (Dhahran, 1991)

## Domain: Military real-time embedded system

**Impact:** The missile struck U.S. Army barracks, killing 28 soldiers and injuring 98. This accident **highlighted the consequences of violating real-time and numerical assumptions in critical systems.**

### What went wrong

- Floating-point rounding error accumulated over long uptime
- Missile defence system operating failed to engage an incoming Scud missile (missed it by ~0.34 seconds)
- Assumed short operational durations

### Ignored challenges

- ❌ Hard real-time constraints
- ❌ Numerical modeling
- ❌ Reliability over long runtime
- ❌ Cyber–physical timing mismatch

📌 *"Average-case performance" arguments fail in real-time systems.*

# Tesla Autopilot Fatality (2016–2023, recurring cases)

## Domain: Automotive CPS

**Impact:** Multiple fatal accidents; ongoing investigations.
These incidents **highlighted how overreliance on automation and unclear human–machine interaction can compromise safety in CPSs.**

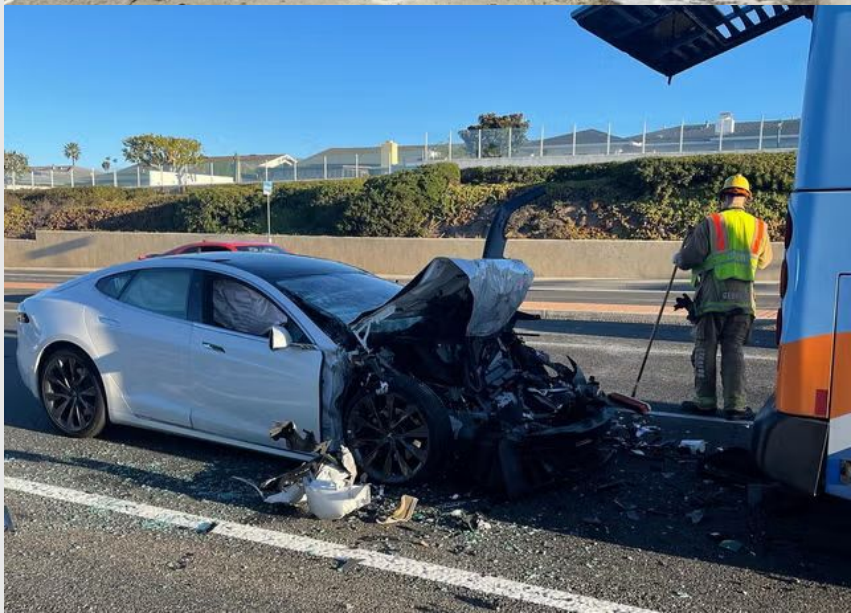
### What went wrong

- Overreliance on automation by human drivers
- Ambiguous system capabilities and limitations
- Vision system misclassified real-world objects
- No robust handover enforcement to the human

### Ignored challenges

- ❌ Human–machine interaction
- ❌ Safety validation
- ❌ Model mismatch (perception vs. reality)
- ❌ Predictability

📌 *Partial/ambiguous automation can be more dangerous than no automation.*



# Boeing 737 MAX MCAS Crashes (2018–2019)

## Domain: Avionics CPS

**Impact:** 346 deaths
These accidents **highlighted how insufficient redundancy, poor human-machine interaction, and unsafe automation can lead to catastrophic failures in safety-critical systems.**

### What went wrong

- Relied on single sensor input, no redundancy or cross-checking
- Pilots not informed of system behavior
- Software repeatedly overrode human control
- Cost-driven design constraints limited system redundancy and pilot retraining

### Ignored challenges

- ❌ Safety, Reliability
- ❌ Human–machine interaction
- ❌ Dependability & redundancy
- ❌ Certification discipline
- ❌ System-level trade-off analysis

📌 *Unsafe automation combined with economic design constraints is an issue.*

# Stuxnet Worm (2010)

## Domain: Industrial CPS (SCADA)

**Impact:** Physical destruction of Iranian nuclear centrifuges.
This accident **highlighted how cyber attacks can silently manipulate physical processes in industrial control systems.**

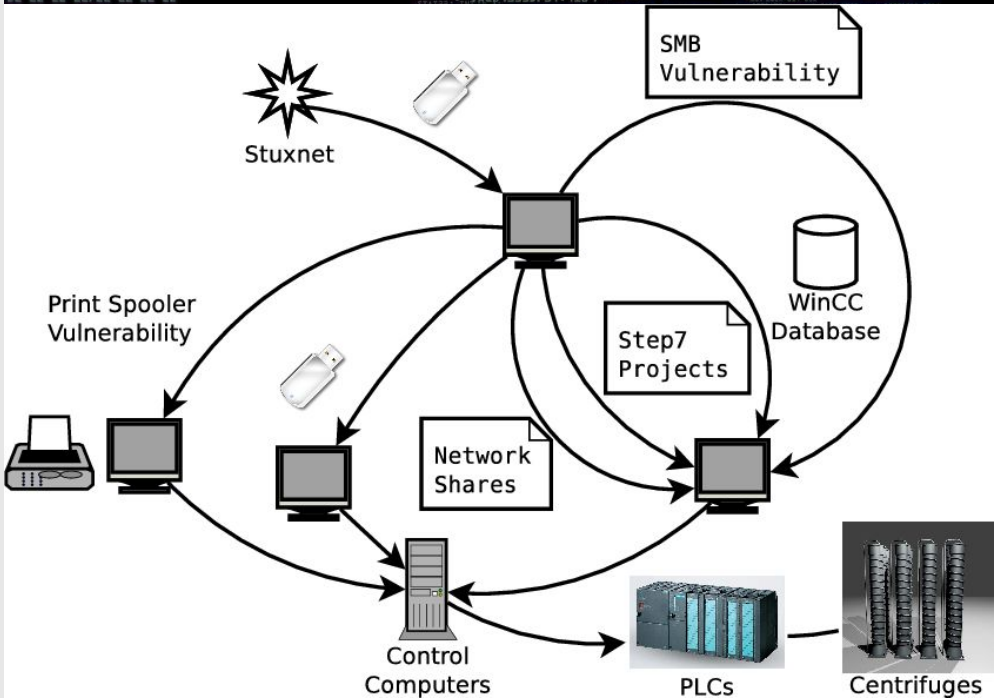
### What went wrong

- Malware manipulated PLC control logic
- Sensors were spoofed to hide damage
- Assumed air-gapped systems were secure

### Ignored challenges

- ❌ Security
- ❌ Integrity
- ❌ Cyber-physical attack awareness
- ❌ Trust assumptions

📌 *Cyber attacks can directly cause physical damage.*

# Samsung Galaxy Note 7 Battery Fires (2016)

## Domain: Consumer embedded systems

**Impact:** Global recall; flights banned; billions in losses, 26 reports of burns and 55 reports of property damage.
This accident **highlighted how insufficient thermal safety margins and inadequate power-system validation can lead to catastrophic device failures.**

### What went wrong

- Battery design left insufficient space for thermal expansion
- Short circuits caused thermal runaway
- Inadequate thermal safety margins
- Accelerated production skipped full validation

### Ignored challenges

- ❌ Power and energy safety
- ❌ Thermal modeling
- ❌ HW/SW co-design
- ❌ Verification under worst-case conditions

📌 *Energy-dense systems demand conservative thermal design.*

# Google Pixel 6 / 7 Overheating Issues (2021–2023)

## Domain: Mobile CPS

**Impact:** Throttling, camera shutdowns, user complaints.
This incident **highlighted how insufficient thermal design compromises performance and user experience in energy-constrained systems.**

### What went wrong

- SoC heat generation exceeded passive cooling capacity
- Thermal throttling triggered aggressively
- Performance–energy trade-offs poorly balanced

### Ignored challenges

- ❌ Power efficiency
- ❌ Thermal modeling
- ❌ User experience under sustained load

📌Thermal throttling is a design compromise, not a bug.

# Xbox 360 “Red Ring of Death” (2005–2009)

## Domain: Consumer embedded systems

**Impact:** Millions of units failed; $1B+ losses
This failure **highlighted how poor thermal design and long-term heat stress can severely reduce hardware reliability.**

### What went wrong

- Excessive heat caused solder joint failures
- Thermal cycling degraded connections
- Inadequate cooling design for GPU/CPU

### Ignored challenges

- ❌ Thermal design
- ❌ Reliability
- ❌ Long-term stress modeling

📌 *Heat kills hardware silently and cumulatively.*

# Mars Pathfinder Resets (1997)

## Domain: Space embedded real-time systems

**Impact:** Repeated system resets on Mars, temporarily interrupting mission operations.
This failure **highlighted how poor resource management can destabilize safety-critical real-time embedded systems.**

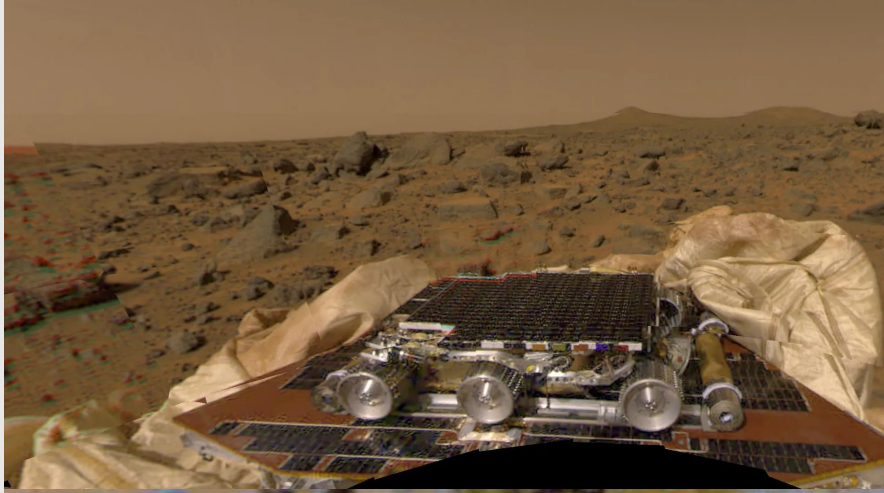
### What went wrong

- Priority inversion between real-time tasks
- Low-priority task held a shared resource
- High-priority task blocked, causing watchdog timeout
- System repeatedly reset due to missed deadlines

### Ignored challenges

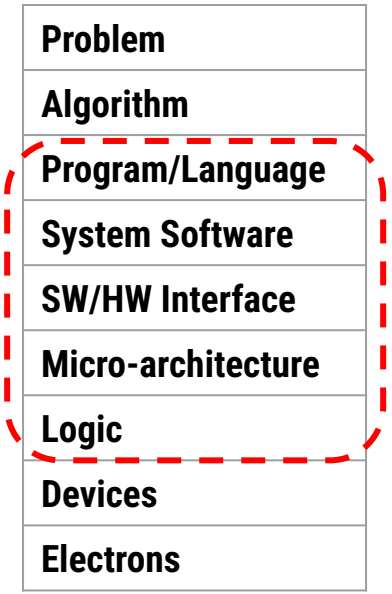
- ❌ Real-time scheduling
- ❌ Synchronization protocols
- ❌ Predictability
- ❌ Worst-case execution analysis

📌 *In real-time embedded systems, poor resource management can trigger failures even when hardware and software are individually correct.*



# Current Key Research Directions & Major Topics: *Building Fundamentally Better Architectures*

- **Fundamentally Energy-Efficient (Memory-/Data-Centric) Architectures**
  - Processing-in-memory / near-memory (DRAM, NVM)
  - Unified memory–storage designs; reducing data movement
- **Fundamentally Low-Latency & Predictable Architectures**
  - Low-latency, low-energy systems with cost-effective memory
  - QoS-aware and predictable memory/system behavior
- **Fundamentally Secure / Reliable / Safe Architectures**
  - Resilience to faults/bit flips; strong reliability guarantees
  - Patchable/upgradeable hardware; secure memory systems
- **Architectures for AI/ML, Genomics, Graph, Medicine & Health**
  - Algorithm–architecture–logic co-design
  - Heterogeneous systems optimized for specific domain workloads
- **Data-Driven & Data-Aware Architectures (cross-cutting)**
  - ML/AI-assisted architectural control, adaptation, and design exploration
  - More “expressive” memory/system abstractions for intelligent management




**Broad research spanning apps, systems, logic with architecture at the center.**

# Final Project Tracks

With good mindset, goals, and focus, you can make a good impact in the world!

## 1. PRACTICAL RESEARCH TRACK (SYSTEM DESIGN & IMPLEMENTATION)

**Goal: Design a novel embedded solution targeting a real-world application domain using either simulation or hardware.**

**This track is recommended for students aiming for industry prototyping roles or embedded product development.**

## 2. SYSTEM OPTIMIZATION TRACK (ALGORITHMS & MODELING)

**Goal: Solve an optimization problem directly relevant to the Embedded Systems Design.**

**This track is recommended for students interested in chip design, NoCs/SoCs, compilers, scheduling, AI accelerators, and other ESD optimization research.**

# TRACK 1. PRACTICAL RESEARCH TRACK (SYSTEM DESIGN & IMPLEMENTATION)

**Forewarnings:**

- Can become expensive if it requires buying external hardware
- Simulation requires **deep validation**, not hand-wavy claims
- Industry-grade convincing is required (“why does this matter?”)

**This track is recommended for:**

**Students aiming for industry prototyping roles or embedded product development.**

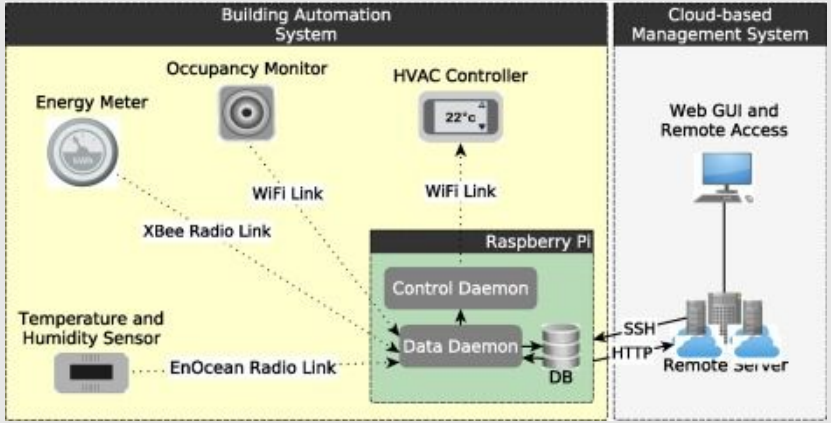
# Suggested Project Topics (You May Suggest Your Own)

## Smart Environmental Monitoring & Alert Station

- **E.g., platform:** STM32 / RISC-V on Zephyr RTOS · Renode-simulated. **Scope:** Periodic sensor reads (temperature, humidity, gas), threshold-based alerts, UART/network logging, low-power sleep modes. **Design Dimensions:** RTOS task scheduling, sensor interfacing, real-time deadlines, energy-aware power modes.

### Some representative publications/prototypes:

M. Aftab et al., "Automatic HVAC Control with Real-time Occupancy Recognition and Simulation-guided Model Predictive Control in Low-cost Embedded System," Energy and Buildings, vol. 154, 2017. — Demonstrates embedded (Raspberry Pi) real-time sensor processing and control scheduling.

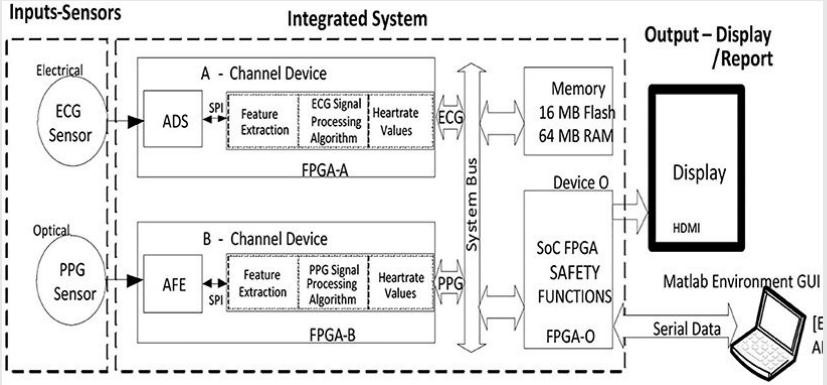



## Fault-Tolerant Patient Vital Signs Monitor

- **E.g., platform:** STM32 on FreeRTOS/Zephyr · Renode-simulated. **Scope**: Wearable-like node: simulated heart rate + SpO2 sensors, redundant readings with majority voting, watchdog-based recovery, alert communication over UART. **Design Dimensions**: Reliability, redundancy, safety-critical design, fault detection.

### Some representative publications/prototypes:

Lakkamraju P, Anumukonda M, Chowdhury SR., "Improvements in Medical System Safety Analytics for Authentic Measure of Vital Signs Using Fault-Tolerant Design Approach," Frontiers in Medical Technology, vol. 3, no. 666671, 2021. — Implements a fault-tolerant architecture for PPG/ECG vital sign monitoring on FPGA; directly addresses redundancy, false alarm reduction, and safety-related design in medical devices.



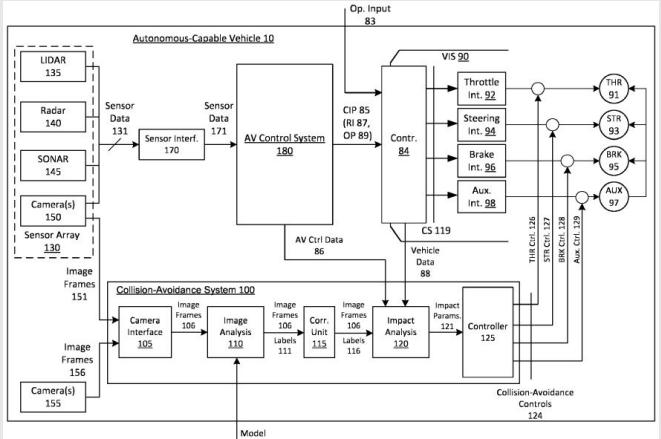
# Suggested Project Topics (You May Suggest Your Own)

## Real-Time Autonomous Vehicle Subsystem

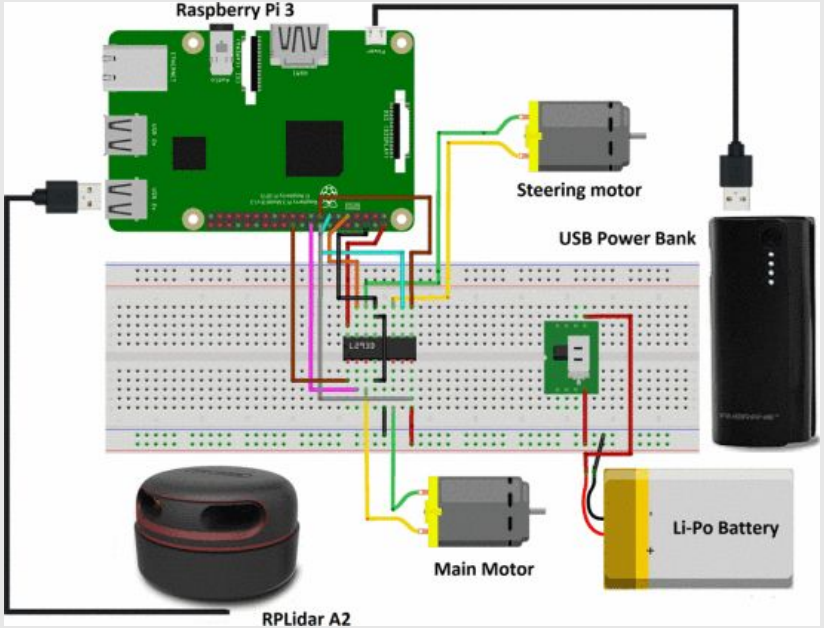
- **E.g., platform:** STM32 on FreeRTOS · Renode-simulated. **Scope**: Simplified collision-avoidance controller: multiple simulated distance sensors feed a decision task with hard deadlines for braking/steering actuator signals. Deliberately includes a priority inversion scenario and demonstrates PIP (Priority Inheritance Protocol). **Design Dimensions**: Hard real-time, concurrency, priority inversion, WCET analysis, preemptive scheduling.

### Some representative publications/prototypes:

Andrew Gray, Collision-avoidance system for autonomous-capable vehicles, https://patents.google.com/patent/EP3418841A1/en — A patent.




N. Baras, G. Nantzios, D. Ziouzios and M. Dasygenis, "Autonomous Obstacle Avoidance Vehicle Using LIDAR and an Embedded System," 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), Thessaloniki, Greece, 2019, pp. 1-4, doi: 10.1109/MOCAST.2019.8742065.



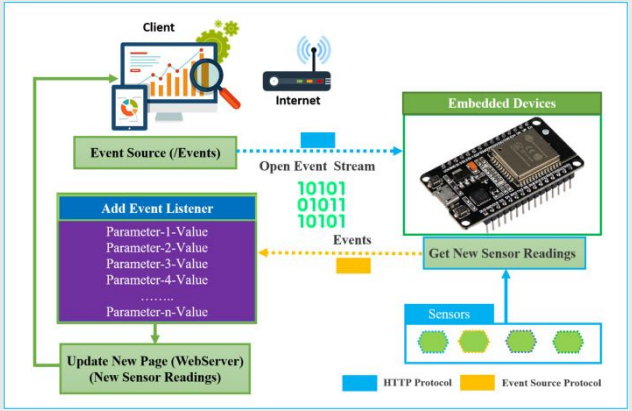
# Suggested Project Topics (You May Suggest Your Own)

## Agricultural Precision Irrigation Controller

- **E.g., platform:** STM32 / RISC-V on Zephyr · Renode-simulated. **Scope**: Soil moisture + weather data drive an irrigation scheduling algorithm with water-budget constraints. Must handle sensor failures gracefully (fallback logic) and log decisions for traceability. **Design Dimensions**: CPS in agriculture, resource constraints, fault tolerance, reactive system design, long-duration reliability.

### Some representative publications/prototypes:

A. Morchid et al., "IoT-enabled Smart Agriculture for Improving Water Management: A Smart Irrigation Control Using Embedded Systems and Server-Sent Events," Scientific African, 2025. architecture, threshold-based scheduling, and fault-aware design.



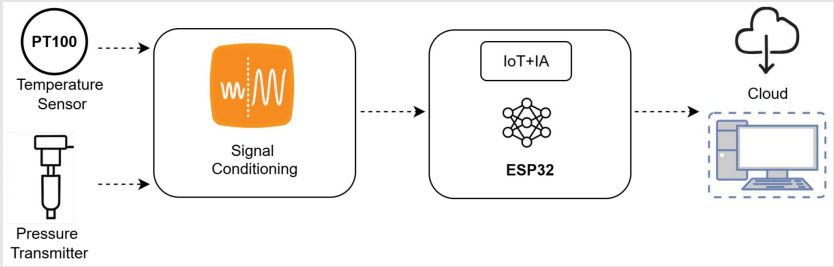

## Industrial Maintenance Node

- **E.g., platform:** STM32 / ESP32 on Zephyr/FreeRTOS · Renode-simulated. **Scope**: Vibration + temperature sensor data processed on-device using lightweight anomaly detection (threshold + moving average or tiny ML inference). Alerts generated when patterns suggest impending failure. **Design Dimensions**: Edge AI/ML on constrained devices, Industry 4.0, data-driven design, power-aware processing.

### Some representative publications/prototypes:

S. Gupta and S.N. Shivhare, "Embedded TinyML for Predictive Maintenance: Vibration Analysis on ESP32 with Real-Time Fault Detection in Industrial Equipment," Int. J. Computational Modelling Applications, vol. 2, no. 2, 2025.

A. Cuenca-Sánchez et al. “AI-Enhanced Embedded IoT System for Real-Time Industrial Sensor Calibration”, Eng. Proc. 2025, 115(1), 13; https://doi.org/10.3390/engproc2025115013 - an AI-enhanced embedded IoT system for real-time industrial sensor calibration.



# If you want to run ML models in your embedded system:

**Apache TVM** https://tvm.apache.org/ https://github.com/apache/tvm/

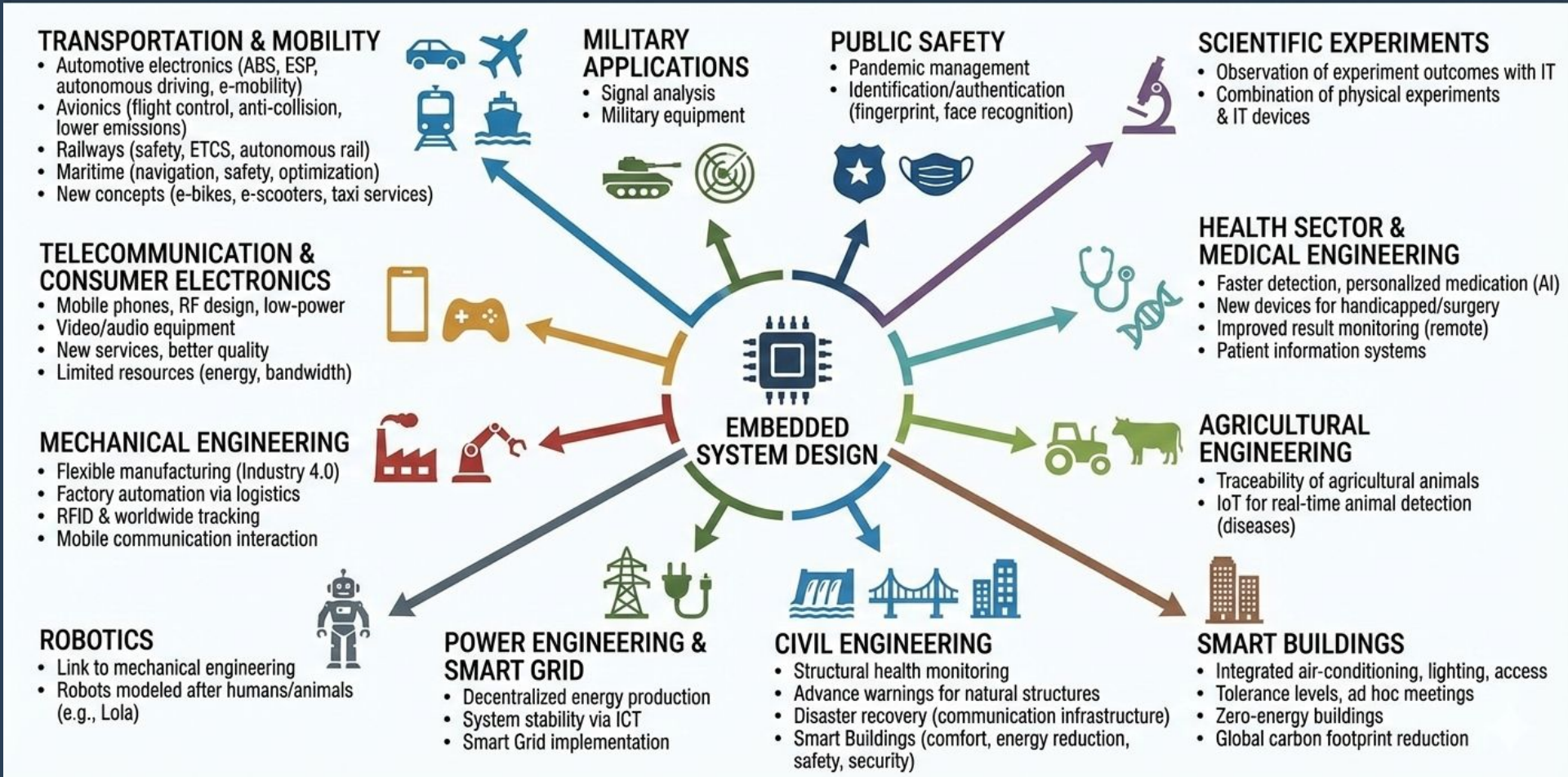
**An Open Machine Learning Compiler Framework**

A machine learning compilation framework, following the principle of Python-first development and universal deployment. It takes in pre-trained machine learning models, compiles and generates deployable modules that can be embedded and run everywhere.

## For Application Ideas:

H. Han and J. Siebert, "TinyML: A Systematic Review and Synthesis of Existing Research," *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Jeju Island, Korea, Republic of, 2022, pp. 269-274, doi: 10.1109/ICAIIC54071.2022.9722636.

# You may choose any application area for your final projects…



# TRACK 2. SYSTEM OPTIMIZATION TRACK (ALGORITHMS & MODELING)

**Forewarnings:**

- Could be a bit challenging for those without optimization or strong math/algorithms background
- However, it is **very** relevant to modern ESD research and industry

**This track is recommended for:**

**Students interested in chip design, NoCs/SoCs, compilers, scheduling, AI accelerators, or optimization research in general.**

# Suggested Project Topics (You May Suggest Your Own)

**Application-Specific Heterogeneous Network-on-Chip Design: E.g.,**

- Communication latency vs. power consumption vs. chip area
- Co-design of major issues: floorplanning, routing topology generation, routing path construction, application mapping, etc.

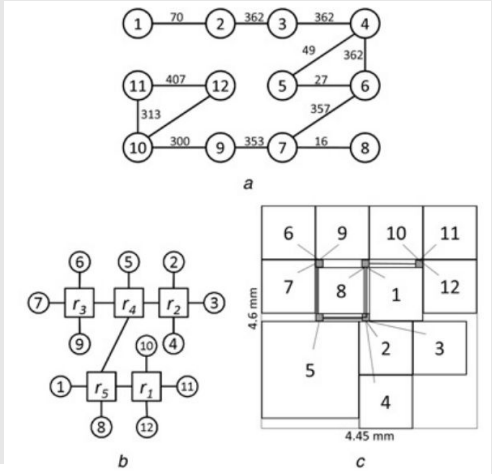
**Some representative publications:**

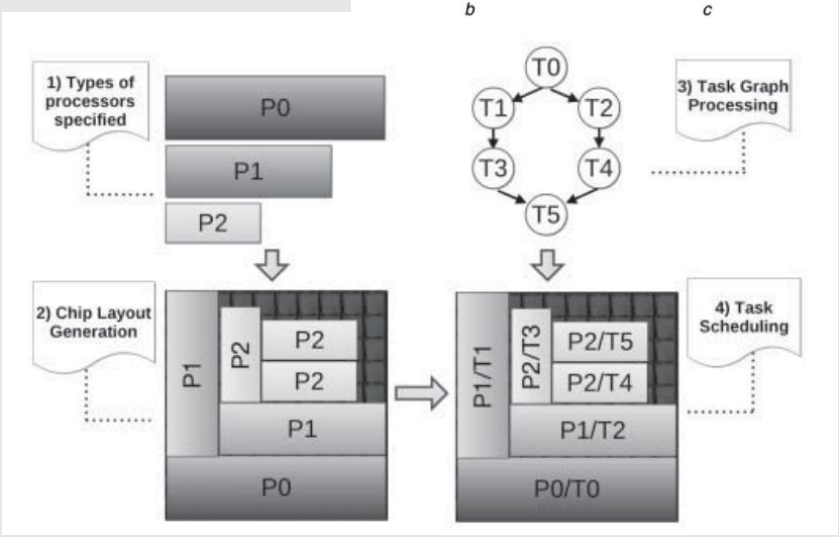

D. Demirbas, I. Akturk, O. Ozturk, U. Güdükbay, “Application-Specific Heterogeneous Network-on-Chip Design”, The Computer Journal, vol. 57, no. 8, August 2014, Pages 1117–1131, https://doi.org/10.1093/comjnl/bxt011

S. Liu and M. Radetzki, "Synergistic Floorplanning and Routing Topology Co-design for Application-Specific NoC Synthesis," 2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), Kuala Lumpur, Malaysia, 2024, pp. 179-186, doi: 10.1109/MCSoC64144.2024.00039.

H. Zheng, K. Wang and A. Louri, "Adapt-NoC: A Flexible Network-on-Chip Design for Heterogeneous Manycore Architectures," 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea (South), 2021, pp. 723-735, doi: 10.1109/HPCA51647.2021.00066.

S. Tosun, Y. Ar, and S. Ozdemir, “Application-specific topology generation algorithms for network-on-chip design”, IET Computers & Digital Techniques, vol. 6, no. 5, https://doi.org/10.1049/iet-cdt.2011.0080

# Suggested Project Topics (You May Suggest Your Own)

**Multi-Objective Trade-offs / Optimization for MPSoCs: E.g.,**

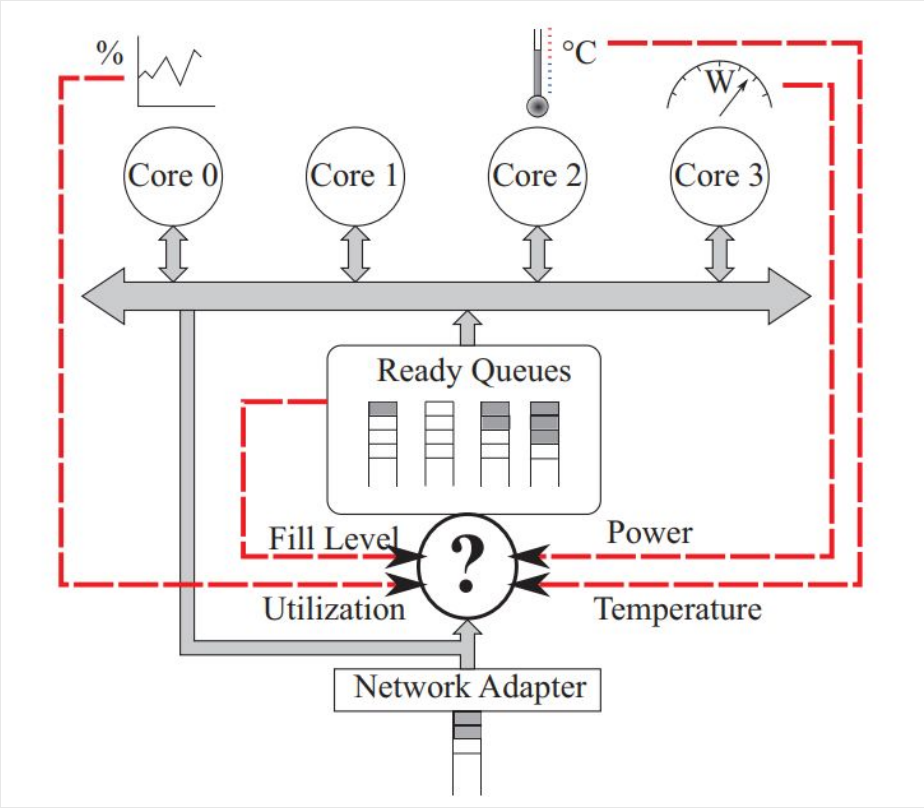
- Performance vs. Power Consumption
- Cost vs. Capability
- Fault-Tolerance vs. Latency
- Speed vs. Code Size, etc.

**Some representative publications:**


H. Youness, A. Omar and M. Moness, "An Optimized Weighted Average Makespan in Fault-Tolerant Heterogeneous MPSoCs," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 8, pp. 1933-1946, 1 Aug. 2021, doi: 10.1109/TPDS.2021.3053150.

S. Hong, S. H. K. Narayanan, M. Kandemir and O. Ozturk, "Process variation aware thread mapping for Chip Multiprocessors," 2009 Design, Automation & Test in Europe Conference & Exhibition, Nice, France, 2009, pp. 821-826, doi: 10.1109/DATE.2009.5090776.

R. K. Pujari, T. Wild and A. Herkersdorf, "A hardware-based multi-objective thread mapper for tiled manycore architectures," 2015 33rd IEEE International Conference on Computer Design (ICCD), New York, NY, USA, 2015, pp. 459-462, do: 10.1109/ICCD.2015.7357148.

# Suggested Project Topics (You May Suggest Your Own)

**High-Level Synthesis Optimization for ASICs: E.g.,**

- Reliability vs. Power Consumption
- Performance vs. Latency
- Fault-Tolerance vs. Latency
- Dynamic Voltage and Frequency Scaling

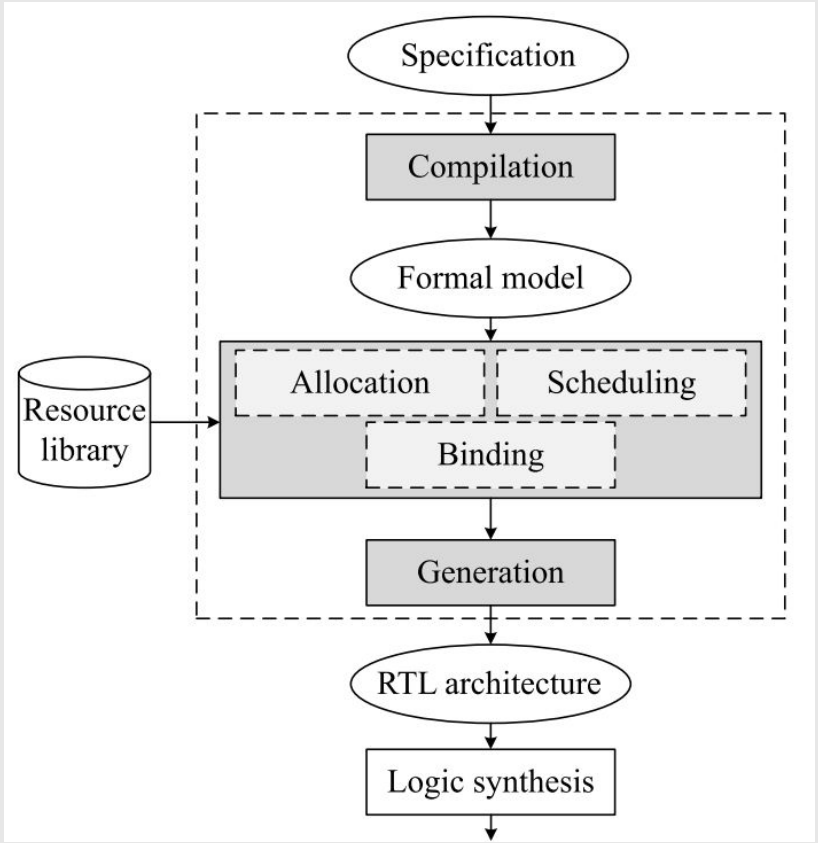
**Some representative publications:**


S. Dilek, R. Smri, S. Tosun and D. Dal, "A High-Level Synthesis Methodology for Energy and Reliability-Oriented Designs," in IEEE Transactions on Computers, vol. 71, no. 1, pp. 161-174, 1 Jan. 2022, doi: 10.1109/TC.2020.3043885.

F. Dabiri, N. Amini, M. Rofouei and M. Sarrafzadeh, "Reliability-Aware Optimization for DVS-Enabled Real-Time Embedded Systems," 9th International Symposium on Quality Electronic Design (isqed 2008), San Jose, CA, USA, 2008, pp. 780-783, doi: 10.1109/ISQED.2008.4479837.

S. Dilek, S. Tosun, A. Cakin, "Simulated annealing-based high-level synthesis methodology for reliable and energy-aware application specific integrated circuit designs with multiple supply voltages". Int J Circ Theor Appl. 2023; 51(10): 4897-4938. Doi: 10.1002/cta.3666.

S. Dilek, S. Tosun, "Integer linear programming-based optimization methodology for reliability and energy-aware high-level synthesis," Microelectronics Reliability, 139, 2022, 114849, doi: 10.1016/j.microrel.2022.114849.

# Suggested Project Topics (You May Suggest Your Own)

**Network-on-Chip Topology Design Methods for Specific Applications like Neural Network Accelerators: E.g.,**

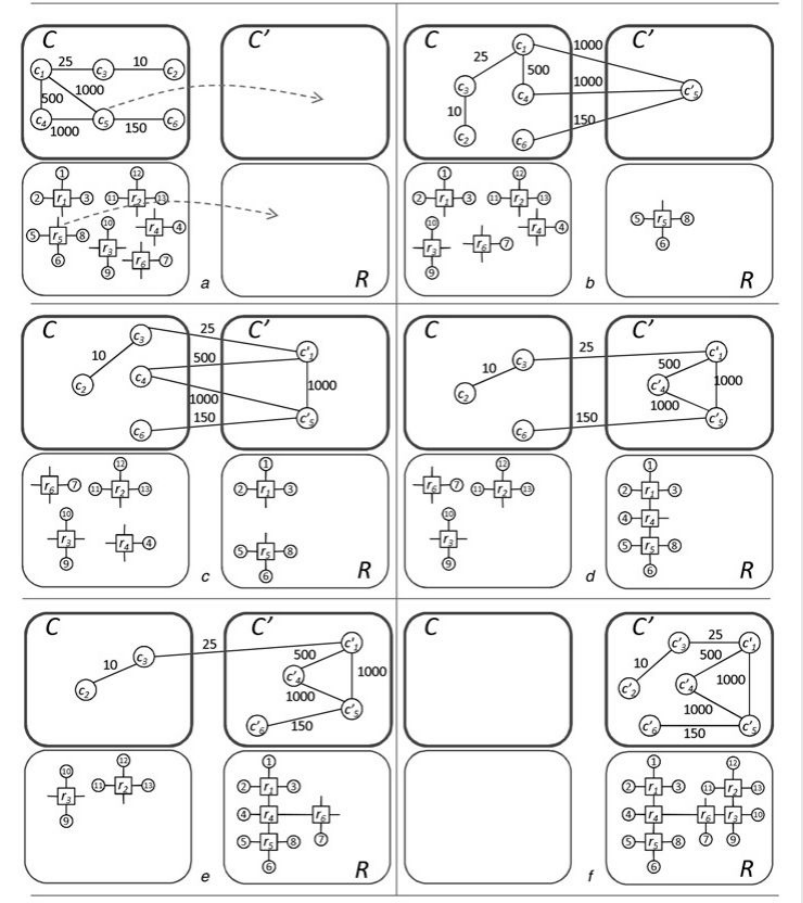
- Communication cost
- Energy consumption
- Latency
- Area

**Some representative publications:**


S. Tosun, Y. Ar, and S. Ozdemir, “Application-specific topology generation algorithms for network-on-chip design,” Computers & Digital Techniques, IET, vol. 6, pp. 318–333, 09 2012. Doi: 10.1049/iet-cdt.2011.0080.

Oliveira, S.d.S.; Carvalho, B.M.d.; Kreutz, M.E. Network-on-Chip Irregular Topology Optimization for Real-Time and Non-Real-Time Applications. Micromachines 2021, 12, 1196. https://doi.org/10.3390/mi12101196.

N.B. Ozgur, A. Cakin, S. Dilek, and S. Tosun, “Irregular Network-on-Chip Topology Design Method for Neural Network Accelerators”, submitted to IEEE Transactions on Computers. Available if topic chosen for final project.

# Suggested Project Topics (You May Suggest Your Own)

**Memory Systems Optimizations: E.g.,**

- Memory Power Management
- Reducing Memory Interference in Multicore Systems
- Efficient Convolutional Dataflows on Low-Power Neural Network Accelerators

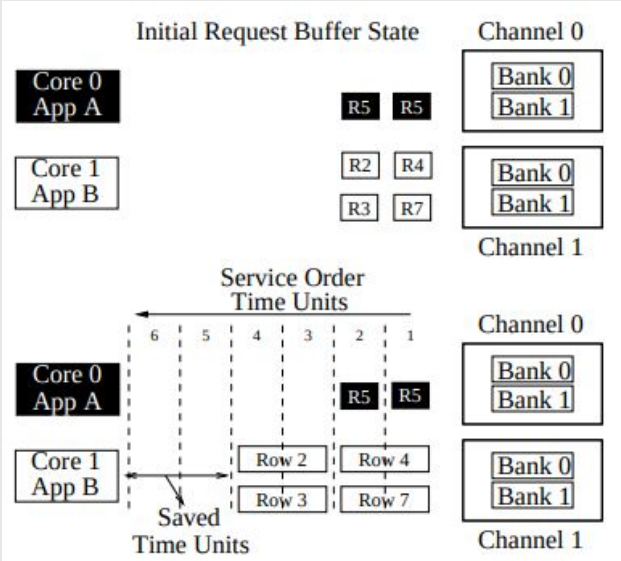
**Some representative publications:**

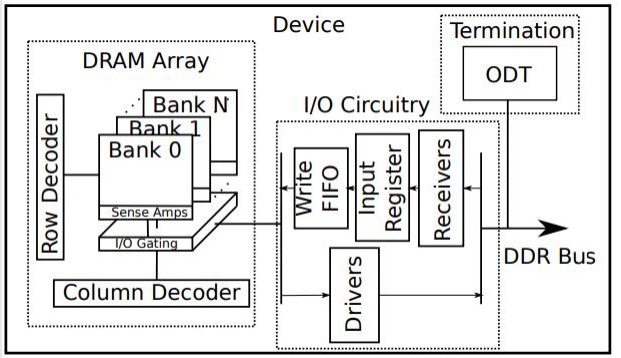

S. P. Muralidhara, L. Subramanian, O. Mutlu, M. Kandemir and T. Moscibroda, "Reducing memory interference in multicore systems via application-aware memory channel partitioning," 2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Porto Alegre, Brazil, 2011, pp. 374-385. https://ieeexplore.ieee.org/document/7851487.

Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. 2011. Memory power management via dynamic voltage/frequency scaling. In Proceedings of the 8th ACM international conference on Autonomic computing (ICAC '11). Association for Computing Machinery, New York, NY, USA, 31–40. https://doi.org/10.1145/1998582.1998590.

L. Orosa et al., "EcoFlow: Efficient Convolutional Dataflows on Low-Power Neural Network Accelerators," in IEEE Transactions on Computers, vol. 73, no. 9, pp. 2275-2289, Sept. 2024, doi: 10.1109/TC.2023.3272282.


See https://safari.ethz.ch/ for more research ideas.



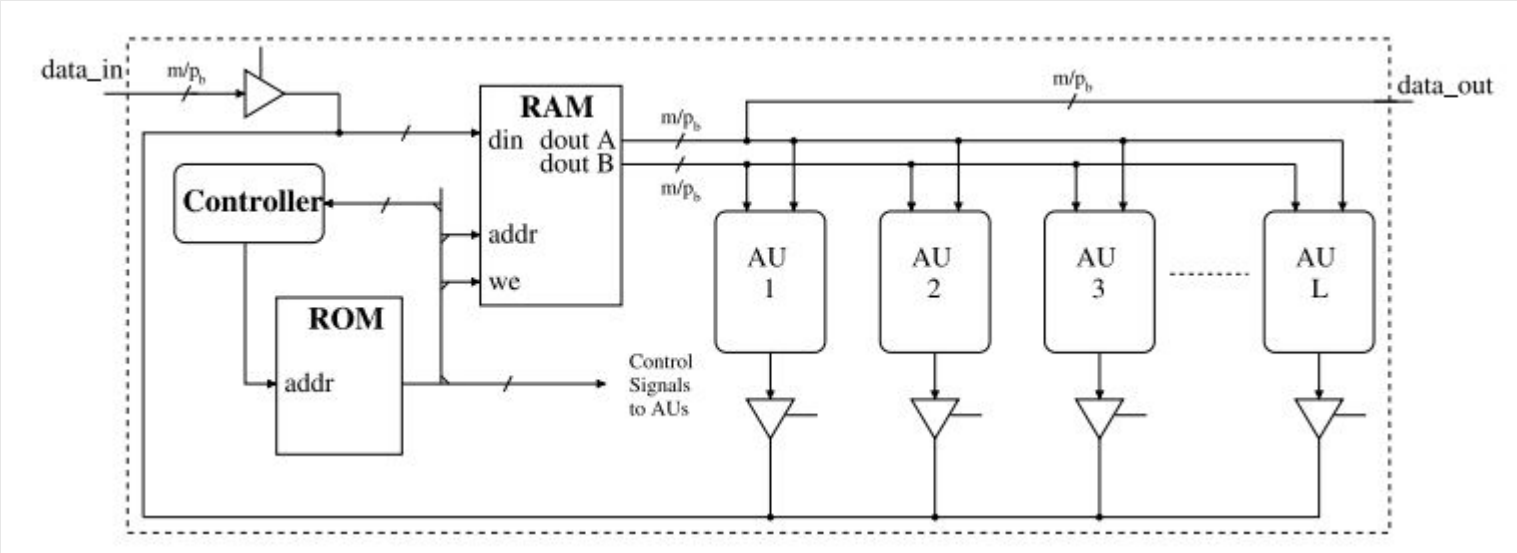
# Suggested Project Topics (You May Suggest Your Own)

**Security Systems Optimizations: E.g.,**

- Elliptic Curve Cryptography on FPGA for Low-Power Applications

**Some representative publications:**

Maurice Keller, Andrew Byrne, and William P. Marnane. 2009. Elliptic Curve Cryptography on FPGA for Low-Power Applications. ACM Trans. Reconfigurable Technol. Syst. 2, 1, Article 2 (March 2009), 20 pages. https://doi.org/10.1145/1502781.1502783



# Where to look for project ideas?

Check the well cited or recent publications in top venues:

| Type | Representative Venues |
| --- | --- |
| Design Automation & CAD | DAC, DATE, ICCAD, ASP-DAC, IEEE TCAD |
| Embedded Systems | EMSOFT, RTAS, TECS, IEEE Embedded Systems Letters |
| Architecture & FPGA | MICRO, ISCA, FPGA Symposium |

Google Patents: https://patents.google.com/

# THANK YOU!

**HW for next week:**

- Brush up on:
  - Finite State Machines and state diagrams
  - Common serial communication interfaces (I2C, UART)
- Inspect documentation on The Arm® Cortex®-M4-based STM32F4 MCU
- Be ready to take notes, write, draw…
- Connect with classmates to make a final project team and research final project topics - to be proposed in Week 3.