

Hardware-Software Co-Design for Cyber-Physical Systems (CPS)

Gaurav Sharma¹, Devansh Kashyap², Manish Patil³

Associate Professor, Assistant Professor

Department of Electronics and Communication

Greenfield Institute of Technology

gauravsharma1ji@yahoo.com, kashyap_devan21@gmail.com,

patilmanish20@rediffmail.com

Abstract

Cyber-Physical Systems (CPS) combine computation, networking, and physical processes. These systems are widely used in automotive control, healthcare devices, industrial automation, smart grids, and robotics. Traditional system design separates hardware and software development, which often leads to inefficiencies, timing issues, and higher costs. Hardware-Software Co-Design is an integrated design approach where both hardware and software components are developed together to meet performance, energy, cost, and timing constraints. This paper reviews the concept of hardware-software co-design for CPS, its methodologies, tools, challenges, and applications. Various partitioning techniques, modeling approaches, communication mechanisms, and real-time considerations are discussed. The paper also highlights how co-design improves system reliability and optimization in safety-critical CPS applications. Finally, future trends such as AI-driven design and reconfigurable architectures are explored.

Keywords: Cyber-Physical Systems, Hardware-Software Co-Design, Embedded Systems, Real-Time Systems, FPGA, System Optimization

1. Introduction

Cyber-Physical Systems (CPS) represent integration of computational algorithms and physical components. Sensors collect real-world data, controllers process the data, and actuators perform actions in the physical environment. Examples include autonomous vehicles, medical monitoring devices, smart manufacturing, and smart energy systems.

In early embedded systems, hardware and software were developed separately. Hardware teams focused on circuits and boards, while software teams wrote code after hardware

completion. This approach is not suitable for CPS where timing, power consumption, and performance requirements are strict. Hence, Hardware-Software Co-Design emerged as a solution where both domains are designed simultaneously.

Co-design helps in deciding which tasks should be implemented in hardware (like FPGA, ASIC) and which in software (running on microcontroller/processor). Proper partitioning reduces delay, power usage, and cost while improving system reliability.

2. Overview of Cyber-Physical Systems (CPS)

Cyber-Physical Systems (CPS) are integrations of physical processes with computation and communication. In these systems, embedded computers and networks continuously monitor and control physical processes through feedback loops. Unlike traditional embedded systems, CPS are tightly coupled with the environment and must react to real-world changes in real time. The correctness of a CPS depends not only on logical results of computation but also on the time at which the results are produced.

A typical CPS is composed of five major elements that work together in a closed control loop.

2.1 Sensors to Capture Physical Parameters

Sensors are the primary interface between the physical world and the digital system. They measure parameters such as temperature, pressure, motion, light, humidity, speed, voltage, biomedical signals, and many others.

Examples:

- Temperature sensors in smart thermostats
- LIDAR, RADAR, and cameras in autonomous vehicles
- ECG and pulse sensors in healthcare monitors
- Vibration sensors in industrial machines

Sensors convert physical signals into electrical signals which are then digitized using Analog-to-Digital Converters (ADC). The quality of sensing directly affects the accuracy and reliability of the CPS. Noisy or delayed sensor data can cause incorrect control actions.

2.2 Embedded Processors for Computation

The embedded processor is the “brain” of the CPS. It processes sensor data, runs control algorithms, makes decisions, and generates commands for actuators.

These processors may include:

- Microcontrollers (ARM Cortex-M, AVR, PIC)
- Microprocessors (ARM Cortex-A, RISC-V)
- Digital Signal Processors (DSP)
- FPGA/ASIC for high-speed parallel processing

The processor executes real-time tasks such as filtering sensor data, running estimation algorithms (like Kalman filters), decision making, and fault detection. In advanced CPS, multiple processors may work together in a distributed architecture.

2.3 Communication Networks

Communication networks allow different components of CPS to exchange data. In many CPS applications, devices are geographically distributed and connected through wired or wireless networks.

Common communication technologies:

- CAN, LIN, FlexRay in automotive systems
- Ethernet and Industrial Ethernet in factories
- Zigbee, Wi-Fi, Bluetooth, LoRa in IoT-based CPS
- 5G for high-speed, low-latency communication

Reliable and low-latency communication is important because delays in data transfer can affect control performance. Network security is also critical to prevent cyber-attacks.

2.4 Actuators to Influence Physical Process

Actuators convert digital commands into physical actions. They are responsible for affecting the environment based on decisions made by the processor.

Examples:

- Motors controlling robotic arms
- Valves controlling fluid flow in industries
- Braking system in vehicles
- Insulin pumps in medical devices

The performance of actuators must be precise and timely. Any delay or inaccuracy can degrade the system behavior.

2.5 Control Algorithms and Software

Control software forms the intelligence of CPS. It includes algorithms that determine how the system reacts to sensor inputs.

These may include:

- PID controllers
- State-space control
- Adaptive and predictive control
- AI/ML-based decision algorithms
- Fault detection and recovery mechanisms

This software usually runs on a Real-Time Operating System (RTOS) to ensure that tasks are completed within deadlines.

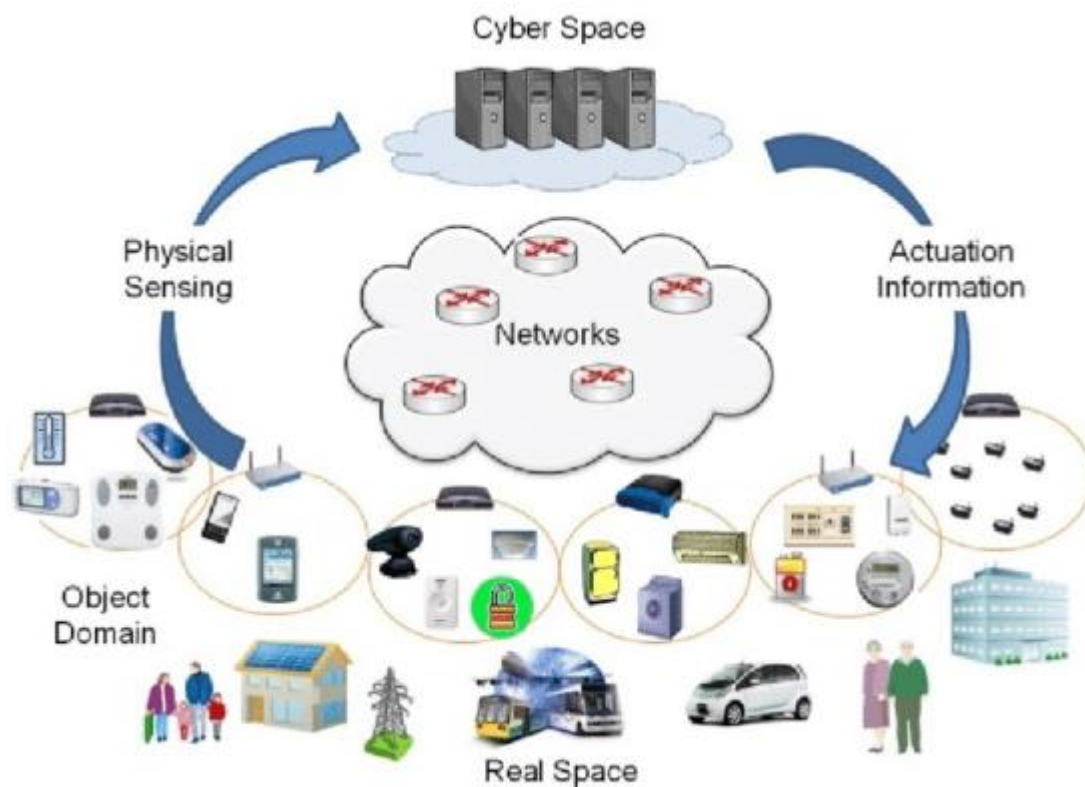


Figure 1: Basic CPS Architecture

3. Concept of Hardware–Software Co-Design

Hardware–Software Co-Design is a design methodology in which hardware and software for a system are developed **simultaneously** and in a coordinated manner. Instead of first finalizing hardware and later writing software for it, both domains evolve together from the early stages of system development. This approach is especially important for Cyber-Physical Systems (CPS), where timing, energy consumption, reliability, and cost are tightly constrained.

In CPS, many functions can be implemented either in hardware (FPGA/ASIC/accelerators) or in software (running on microcontroller/processor). Co-design helps engineers decide the best placement of each function so that the overall system becomes efficient and reliable.

The co-design process typically follows five important stages.

3.1 System Specification

The first step is to clearly define **what the system must do** and under what constraints it must operate.

This specification includes:

- Functional requirements (control tasks, sensing, actuation)
- Timing requirements (deadlines, latency limits)
- Power and energy limits
- Cost constraints
- Safety and reliability requirements
- Communication needs

At this stage, the system is described at a high level without deciding whether functions will be in hardware or software. The focus is on system behavior and performance expectations.

For example, in an automotive braking system, the specification may state that obstacle detection and braking action must occur within a few milliseconds.

3.2 Modeling and Simulation

Before building the actual system, designers create abstract models using tools such as **SystemC, MATLAB/Simulink, UML, or VHDL/Verilog behavioral models**.

Modeling allows:

- Understanding system behavior
- Early performance estimation
- Testing control algorithms
- Evaluating timing constraints
- Detecting design issues before implementation

Simulation at this level is faster and cheaper than testing on real hardware. Different design options can be explored easily.

3.3 Partitioning Between Hardware and Software

This is the **core step** of co-design. Based on simulation results and requirements, system functions are divided into hardware and software components.

General guidelines for partitioning:

- Tasks requiring **high speed and parallelism** → Hardware (FPGA/ASIC)
- Tasks needing **flexibility and frequent updates** → Software
- Time-critical control loops → Hardware assisted
- Complex decision logic → Software

For example:

- Signal filtering and sensor data preprocessing may be done in FPGA.
- Decision making and user interface handled in software.

Proper partitioning improves performance while keeping the system cost-effective.

3.4 Co-Simulation and Verification

After partitioning, both hardware and software parts are simulated **together** to verify system behavior. This is known as co-simulation.

Co-simulation ensures:

- Correct interaction between hardware and software
- Verification of timing and communication
- Detection of integration issues early
- Functional correctness of the complete system

Verification tools check whether deadlines are met and whether data exchange is proper between components.

3.5 Implementation

Once verified, the design moves to physical implementation:

- Hardware is implemented using FPGA/ASIC or microcontroller-based boards.
- Software is developed using C/C++/RTOS and embedded tools.
- Both parts are integrated and tested on the target platform.

Testing at this stage includes real-time performance measurement, power analysis, and reliability checks.

4. Design Methodology in Co-Design

4.1 System Specification

Requirements such as timing, power, cost, and performance are defined.

4.2 Modeling

SystemC, MATLAB/Simulink, UML are used to model system behavior before actual implementation.

4.3 Partitioning

Tasks are divided between hardware and software based on complexity and timing needs.

Criteria	Hardware Implementation	Software Implementation
Speed requirement	High	Moderate
Flexibility	Low	High
Power consumption	Low	Higher
Cost	High initial	Low initial
Upgradability	Difficult	Easy

4.4 Co-Simulation

Both hardware and software models are simulated together to verify functionality.

4.5 Integration and Testing

Final implementation and testing on target platform.

5. Hardware Components Used in CPS Co-Design

In Hardware–Software Co-Design for Cyber-Physical Systems, selection of hardware components is very important because it directly affects timing performance, power usage, flexibility, and cost. Different hardware platforms offer different strengths such as programmability, parallelism, determinism, and scalability. A well-designed CPS often uses a **combination** of these components rather than relying on a single device.

5.1 Microcontrollers and Microprocessors

Microcontrollers (MCUs) and microprocessors (MPUs) form the computational core of most CPS.

Microcontrollers (e.g., ARM Cortex-M, AVR, PIC):

- Integrated CPU, memory, timers, ADC, communication interfaces on a single chip

- Low power consumption
- Suitable for real-time control tasks
- Widely used in automotive ECUs, medical devices, IoT nodes

Microprocessors (e.g., ARM Cortex-A, RISC-V, x86 embedded):

- Higher processing capability
- Support for operating systems like Linux or RTOS
- Used when complex algorithms, image processing, or networking stacks are needed

In co-design, microcontrollers handle control logic and scheduling, while heavy data processing may be offloaded to hardware accelerators.

5.2 FPGA (Field Programmable Gate Array)

FPGA is one of the most important devices in hardware–software co-design.

- Consists of programmable logic blocks and interconnects
- Can be reconfigured after manufacturing
- Supports massive parallel processing
- Deterministic timing behavior
- Ideal for signal processing, sensor data filtering, encryption, and real-time analytics

FPGA allows designers to implement custom hardware tailored for specific tasks without fabricating an ASIC. Because of this **reconfigurability** and **parallel execution**, FPGA is widely used in CPS such as autonomous vehicles, robotics, and industrial control.

For example, multiple sensor streams can be processed simultaneously in FPGA while the processor runs decision algorithms.

5.3 ASICs (Application Specific Integrated Circuits)

ASICs are custom-designed chips built for a specific application.

- Very high performance
- Very low power consumption
- Highly reliable and optimized
- Expensive and time-consuming to design
- Not reprogrammable after fabrication

ASICs are used in mass-produced CPS products where performance and efficiency are critical, such as pacemakers, automotive control chips, and communication processors.

In co-design, once a design is stable and produced in large volume, FPGA prototypes may be converted into ASIC for cost and power benefits.

5.4 Sensors and Actuators

Sensors and actuators are the physical interface of CPS.

Sensors provide environmental data:

- Temperature, pressure, motion, light, sound
- Cameras, RADAR, LIDAR in vehicles
- Biomedical sensors in healthcare

Actuators perform physical actions:

- Motors, relays, valves, pumps
- Steering and braking systems
- Robotic arms

Their precision, latency, and reliability affect the overall CPS performance. Hardware design must ensure proper signal conditioning, ADC/DAC conversion, and protection circuits.

5.5 Communication Modules (CAN, Ethernet, Zigbee)

Communication hardware enables data transfer between CPS components.

- **CAN, LIN, FlexRay** → Automotive networks
- **Ethernet / Industrial Ethernet** → Factory automation
- **Zigbee, Wi-Fi, Bluetooth, LoRa** → IoT-based CPS
- **5G modules** → Low-latency remote CPS

These modules must support reliable and real-time communication. In co-design, communication latency is considered when partitioning tasks between hardware and software.

5.6 Why FPGA is Widely Preferred in CPS Co-Design

FPGA stands out because:

- It combines hardware speed with post-deployment flexibility
- Enables parallel processing of multiple data streams
- Reduces processor load by offloading time-critical tasks
- Allows rapid prototyping before ASIC production
- Provides deterministic execution for real-time systems

Hence, many CPS designs use a **processor + FPGA** architecture for optimal performance.

6. Software Components in Co-Design

- Real-Time Operating Systems (RTOS)
- Device drivers
- Middleware and communication protocols
- Control algorithms
- Application software

RTOS plays a critical role to manage tasks with strict deadlines.

7. Hardware-Software Partitioning Techniques

Partitioning is the core of co-design.

Static Partitioning

Decided at design time based on analysis.

Dynamic Partitioning

Tasks can move between hardware and software during runtime (rare but advanced).

Heuristic Methods

Genetic algorithms, AI-based approaches are used for optimal partitioning.

8. Communication Between Hardware and Software

Efficient communication is required.

- Memory-mapped I/O
- Interrupt mechanisms
- DMA (Direct Memory Access)
- Shared buses

Poor communication design can degrade performance.

9. Real-Time Constraints in CPS

CPS applications must operate within strict deadlines.

Application	Deadline Sensitivity
Automotive brake	Very High
Medical monitor	High

Application	Deadline Sensitivity
Smart home	Moderate
Agriculture IoT	Low

Co-design ensures critical tasks are handled by hardware for faster execution.

10. Tools for Hardware-Software Co-Design

- Xilinx Vivado (FPGA design)
- MATLAB/Simulink
- SystemC
- Keil, IAR Embedded Workbench
- Proteus simulation

These tools allow modeling, simulation, and testing.

11. Applications of Co-Design in CPS

11.1 Automotive Systems

Engine control, ABS, autonomous navigation.

11.2 Healthcare Devices

Pacemakers, wearable monitors.

11.3 Industrial Automation

Robotic arms, PLC systems.

11.4 Smart Grid

Energy monitoring and control.

11.5 Robotics

Real-time sensor processing and motion control.

12. Challenges in Hardware-Software Co-Design

- Complex partitioning decisions
- Tool compatibility issues
- Higher design time
- Verification complexity
- Cost of FPGA/ASIC

13. Advantages of Co-Design

- Performance optimization
- Power efficiency
- Reduced latency
- Better reliability
- Cost effectiveness in long term

14. Future Trends

- AI-based automatic partitioning
- Reconfigurable computing
- Integration with Edge AI
- Digital twins for CPS testing

15. Case Study: Autonomous Vehicle Control Unit

Sensor data processing is implemented in FPGA while decision algorithms run on processor with RTOS. This reduces latency and improves safety.

16. Conclusion

Hardware-Software Co-Design is essential for modern Cyber-Physical Systems. It provides balanced optimization between performance and flexibility. With increasing complexity of CPS, co-design methods and tools will continue to evolve. Future systems will rely on intelligent partitioning and reconfigurable hardware to meet demanding real-time requirements.

References

1. Rajkumar R., Lee I., Sha L., Stankovic J., “Cyber-Physical Systems: The Next Computing Revolution”, IEEE, 2010.
2. Wolf W., “Hardware-Software Co-Design of Embedded Systems”, Proceedings of the IEEE, 2003.
3. Lee E. A., “Cyber Physical Systems: Design Challenges”, 2008.
4. Patterson D., Hennessy J., “Computer Organization and Design”, Morgan Kaufmann.
5. Xilinx Documentation on FPGA Design, 2022.
6. Kopetz H., “Real-Time Systems: Design Principles”, Springer.

7. MATLAB Simulink CPS Modeling Guide.
8. Keil MDK RTOS Documentation.
9. IEEE Transactions on Embedded Systems, Various papers.
10. Recent surveys on CPS Co-Design, ACM Journal, 2021.