Inclusion/Exclusion Documentation Rapid Review RunSoC

Title	Pages	Status	One line Summary	DEC1: Does the study identify the HPC approach (uC + SoC) as an enabler / accelerator to reduce today's system limitations?	DEC2: Which SoC design and properties does the study investigate / elaborate?	DEC3:Which software component properties / requirements are analyzed in the study?	DEC4: Which technologies, methodologies or frameworks are used to map software components to semicon-ductor design partitions?	DEC5: Which are the key challenges in allocating software components to semiconductor design partitions?	DEC6: Which validation methods or case studies are presented to demonstrate the effectiveness of the proposed approach?	DEC7: Are any industry standards or best practices followed in the mapping process?	What are the benefits / targets of the proposed method?	The reason / criteria to use a microprocessor instead of a microcontroller
Deploying and Scheduling Vision Based Advanced Driver Assistance System (ADAS) on Heterogeneous Multicore Embedded Platform	6	Completed	Mapping using single rate data flow graph	Yes	TDA2x System on Chip by Texas Instrument It incorporates four types and a total of 12 processing cores	Focus Area: Applications with set of tasks that have throughput constraints between multiple cores Important Properties: 1. Worst case execution time of each task 2. Slice Time: the time slice assigned to each application 3. Period: the period of the time division multiplexing scheduler * The time division multiplexing schedluar is on the RTOS layer. RTOS layer is responsible for scheduling the different applications and deploys TDM schedular. TDM scheduler gives each application (here ADAS algorithm) a fixed time quanta	Static Mapping 1. System Modelling: Model the system as Single Rate Data Flow (SRDF) graph. 1. Use vertices to represent task. 2. Use edges to represent the data dependency between dependent vertices. 2. Tasks assignment: Identify the type of cores that the tasks need to use. 3. Worst-case execution time calculation: Calculate the worst-case execution time for each task individually 4. Slice time and Period: Choose slice time and the period for the cores. 5. Original MCM: Calculate the MCM for the graph 6. Model the effect of the TDM scheduler: 1. Vertice i's input dges are connected as input edges to latency vertice i_L 2. Output edges of i are connected as output edges to rate vertice i_R 3. An edge from i_L to i_R and self-edge from iR with delay of 1 is added 7. Find static schedule: Find static schedule with the algorithm in Appendix	This algorithm cannot be used to manage dynamics in ADAS applications	The algorithm is tested by a real world lane line departure system. The measured throughput was compatible with the offline calculation	1. Industrial trend: Advanced Driver Assitance Systems are running on heterogeneous multi-core platforms.	1. This algorithm guarantees the throughput constraint 2. This algorithm can synchronize different tasks in each application	Although automotive OEMs currently provide some of the ADAS features in their vehicles, the implementation of these features mostly focuses on one feature per system The accuracy and capability of each system is limited as it either does not have access to the ambient data from the sensors of other systems or could not fully utilize them
A Runtime Manager Integrated Emulation Environment for Heterogeneous SoC Design with RISC-V Cores	8	Completed	This paper presents a runtime manager integrated emulation environment for heterogeneous SoCs with RISC-V cores, using lightweight RISC-V cores to efficiently manage accelerators and improve application execution.	Yes	The study focuses on heterogeneous SoC designs that integrate RISC-V cores and accelerators. A key property investigated is the use of customized lightweight RISC-V cores for managing accelerators. The paper explores the impact of this approach on runtime environment scalability and resource management.	The study analyzes the runtime framework (CEDR) and its ability to manage applications, scheduling heuristics, and accelerators on heterogeneous SoCs. Key requirements include dynamic resource management, efficient task scheduling, and minimizing overhead in managing accelerators.	The paper introduces CEDR, an open-source, unified compilation and runtime framework for heterogeneous architectures. CEDR is used to manage the execution of applications on an emulated SoC with RISC-V cores and accelerators. The methodology involves using customized "little" RISC-V cores for accelerator management to reduce the load on "big" cores that execute application tasks.	The paper addresses the challenge of managing accelerators efficiently in heterogeneous systems. Specifically, it focuses on reducing contention and overhead in the runtime environment when CPU cores handle both application tasks and accelerator management.	The approach is evaluated on a heterogeneous SoC emulated on a Xilinx Virtex-7 FPGA. The evaluation uses applications from domains like radar, signal processing, and autonomous vehicles. The study compares the performance of different accelerator management strategies and scheduling heuristics.	The paper doesn't explicitly detail specific industry standards for mapping software to hardware partitions. However, it leverages established tools and methodologies like RISC-V ISA, Chisel, and FPGA emulation, which are common in the field.	The primary benefit is reducing the overhead associated with accelerator management by offloading it to lightweight RISC-V cores. This leads to improved scalability, reduced contention, and faster application execution. The study targets more efficient utilization of heterogeneous computing resources in SoCs.	The paper doesn't directly compare microprocessors and microcontrollers. Instead, it discusses using different types of RISC-V cores within a SoC: "big" cores for general-purpose computing and "little" cores for dedicated accelerator management. The criteria for using "little" cores are to offload thread management from the "big" cores, reducing overhead and improving overall system performance.
The ZuSE-KI-Mobil AI Accelerator SoC: Overview and a Functional Safety Perspective	6	Completed	The ZuSE-KI-Mobil project develops an AI accelerator SoC and ecosystem for autonomous driving and drone applications, focusing on hardware-software co-design and functional safety.	Yes	The study focuses on the ZuKiMo SoC, a heterogeneous SoC platform designed for AI applications in the mobility domain (autonomous vehicles and drones). Key properties of the ZuKiMo SoC include: Integration of an energy-efficient, scalable-precision AI accelerator (SPA-ML). Heterogeneous architecture with application processors, a DSP, a safety island, and image signal processors. Focus on functional safety for safety-critical applications.	The study analyzes software tooling for AI model deployment and image processing on the SoC. Key requirements and properties include: Efficient compilation and optimization of CNNs. Support for hardware-software co-design. Resilience against faults and adversarial attacks.	The paper presents the ZuKiMo project's approach to hardware-software co-design, which involves: Development of the CEDR framework for SoC simulation. Use of analytical co-design with nested genetic algorithms (AnaCoNGA) to optimize CNNs and hardware accelerators. Hardware-aware partitioning of CNNs for distributed computing. Software tooling based on Apache TVM and a customized Linux operating system.	The paper addresses challenges in: Hardware/software co-design, including task allocation, memory hierarchy design, and hardware-aware quantization. Balancing optimization objectives like computing efficiency, power consumption, and safety requirements. Efficiently deploying AI tasks on edge devices with limited resources.	The paper presents the ZuKiMo project, which involves: Development of a complete SoC platform and ecosystem. Use cases from the mobility domain, specifically autonomous driving (3D object detection using LiDAR) and drone applications. Evaluation of hardware-software co-design methodologies (AnaCoNGA, FLECSim). Analysis of DNN resilience against faults and adversarial attacks.	The ZuKiMo project follows the ISO26262:2018 standard for the development of the Image Signal Processor (ISP). It also leverages established tools and frameworks like Apache TVM, gstreamer, and RISC-V.	The ZuKiMo project aims to develop an SoC platform and ecosystem for efficient AI applications in the mobility domain. Benefits and targets include: Enabling powerful and energy-efficient AI processing for autonomous driving and drones. Facilitating hardware-software co-design for optimal performance. Ensuring functional safety in AI-based systems. Providing software tooling for easy deployment of AI models.	The paper does not explicitly discuss the choice between microprocessors and microcontrollers. Instead, it focuses on the design of a heterogeneous SoC with various processing elements, including ARM Cortex-A65AE application processors, a DSP, and RISC-V cores within the AI accelerator. The criteria for choosing these components are based on the need for high-performance computing, efficient AI acceleration, and functional safety in the target applications.
Real-Time Multi-Task ADAS Implementation on Reconfigurable Heterogeneous MPSoC Architecture	20	Completed	Mapping according to computational intensity + AI multi task learning	Yes	Kria KV260 Multi-processor system-on-chip field programmable gate array (MPSoC-FPGA) platform	-	1. High-speed and computationally intensive parts to programmable logic. 2. Assigning the remaining roles to processing system. 1. Scale and mean value subtraction operations are performed on the ARM, leveraging the capabilities of the Vitis-AI library 2. The remainder of the task is executed within the PL, specificially in the DPU PL: Programmable logic, which is the hardware part of the FPGA that can be reconfigured to implement custom circuts PS: Processing system, which is a processor (like an ARM processor) that can run software	-	-	-	High performance with low energy consumption	-
A Framework for Real-time Automotive Applications to Multicore Platform in Perspective of AUTOSAR	4	Completed	Mapping using Dependency Matrix	Yes	Dual-core: Homogeneous core microcontroller manufactured by nxp semiconductors. Refer to NXP MPC5748G reference manual *The reason why the homogeneous core microcontroller is chosen, is that the communication cost for transferring data from one core to any core will remain the same	Software components properties 1. All the data about application and their runnables (periods, signals, etc) 2. Runnables are having five parameters: worst case execution time, deadline, execution time, period and offset Assumptions 1. The period is equal to its deadline 2. Those runnables mapped to a specific task should have the same period as task and it is assuemed that execution time is less than WCET for each runnable and task 2. Tasks and runnables are strictly periodic and they are following fixed priority scheduling (Strictly periodic means if any runnalbes missed its deadline then there will be serial damage of the system)	1. Data Extraction: Fetch all the data about application and their runnables. 2. Dependency Matrix Development: Extract data of all required and provider port of software components and make a dependency system matrix (DSM). The DSM shows the dependencies between the runnables. The dependencies are determined by the number of shared signals/interfaces. (e. g. R2 is sending 4 signals via provider ports and R1 is receiving signals via required ports, so they share 4 signals.) 3. Allocation step 1. Runnable allocation: 1. Hardware dependent runnables -> main core 2. Runnables shareing more data (Not addressed: How are the "more data" defined) with hardware dependent runnables -> main core 3. Others -> second core 2. Tasks allocation: Equally distributre tasks to both cores 3. Application allocation: Allocate application the cores according to runnables.	The total summed CPU load across both cores will be higher than the original single-core load. -> This is due to context switching and overhead * Side notes 1. A context switch happens when the CPU stops executing on task and switches to another. In a dual-core system, the software needs to coorcinate execution across both cores, leading to more frequent context switches compared to a single-core system 2. Overhead refers to the extra CPU processing required for managing multiple cores. 1. In a single core ECU, there's no need to synchronise between cores. 2. In a dual-core ECU, the system must distribute tasks between the cores, handle inter-core communication, manage synchronization. These operations consume additional cycles, increasing the overall load	1. Timing architect (TA) tool: It manages the timing behavior of AUTOSAR for multi-core ECUs. 2. Indicator of the effectiveness: CPU load distribution 3. Method to evaluate the effectiveness: The TA tool ran for a certain amount of time with provided parameters of the tasks (periodicity, priority and execution time,) to calculate the CPU load	1. Tool: A KPIT AUTOSAR specific tool is used 2. Future outlook: heterogeneous multicore microcontroller and lockstep cores	(Take away info: The paper only elaborates the benefits of the framework, but does not provide enough experimental results to support the statements) 1. Minimize overall communication cost between the runnables (Minimization seems to be an overstatement, there is no calculation or supports) Four types of communication cost are considered: intra-partition, inter-partition, intra-core and inter-core. 2. Efficient processor utilization: The CPU loads get distributed	
How to deploy AI software to self driving cars	1	Completed	No full text	Yes	R-Car V3H system-on-chip from the Renesas automotive platform for ADAS	-	-	-	-	-	-	-
Real-Time Multi-Learning Deep Neural Network on an MPSoC-FPGA for Intelligent Vehicles: Harnessing Hardware Acceleration With Pipeline	12	Completed	This paper introduces a hardware-software co-designed MPSoC-FPGA framework for real-time multi-learning in intelligent vehicles, enhancing ADAS capabilities by efficiently processing multiple tasks with a custom DPU.	Yes	The study focuses on an MPSoC-FPGA (Multi-Processor System-on-Chip - Field Programmable Gate Array) architecture. Key properties include the integration of a hardware accelerator for real-time multi-learning models and the balance of performance and energy efficiency. The paper utilizes the AMD Xilinx Kria KV260 board, which features a Deep Processing Unit (DPU).	The study analyzes a multi-learning framework for ADAS applications. Requirements include real-time processing of multiple ADAS tasks (object detection, segmentation, lane detection, drivable area segmentation, and depth estimation) on a single platform. The software components are designed to work in a hardware-software co-design to optimize performance and resource utilization.	The paper employs a hardware-software co-design approach to map software components onto the MPSoC-FPGA architecture. Here's a more detailed breakdown: Hardware-Software Co-design: The core methodology is the concurrent design and optimization of both software and hardware components to achieve optimal performance and resource utilization for the multi-learning ADAS application. This involves making decisions about which parts of the algorithms are best suited for implementation in software (running on the microprocessor) and which parts will benefit from hardware acceleration in the FPGA. MPSoC Architecture Utilization: The Zynq MPSoC's architecture is leveraged, which consists of a Processing System (PS) (containing microprocessors) and Programmable Logic (PL) (the FPGA). PS (Processing System): The PS is used for tasks like pre-processing of high-resolution video inputs (e.g., resizing, segmentation), queue management, post-processing of results, and display output management. PL (Programmable Logic): The PL is used to implement hardware accelerators for computationally intensive parts of the deep learning models, specifically the Deep Processing Unit (DPU). This hardware acceleration is crucial for achieving real-time	The paper addresses the challenge of meeting the computational and memory demands of real-time video processing for ADAS. It also focuses on optimizing resource utilization and energy efficiency in the hardware-software co-design process. Efficiently allocating tasks between the processing system (PS) and programmable logic (PL) of the Zynq architecture is crucial.	The proposed framework is evaluated on the AMD Xilinx Kria KV260 board. The evaluation includes real-time processing of highway scenes and performance metrics for various ADAS tasks. The study compares the proposed method with state-of-the-art approaches, analyzing factors like GOP performance, power consumption, resource consumption, and frame rate.	The paper utilizes the AMD Xilinx Vitis-AI library, which is a widely used tool for AI acceleration on Xilinx platforms. The research emphasizes hardware-software co-design, a recognized best practice for optimizing performance in embedded systems.	The primary benefit is enhancing ADAS's safety and error prevention capabilities in intelligent vehicles. The method aims to address the computational and memory demands of real-time video processing. It targets improved computational efficiency, reduced power consumption, and efficient execution of multiple ADAS tasks on a single hardware platform.	The paper focuses on using an MPSoC, which integrates a microprocessor (in the processing system or PS) with programmable logic (PL). The choice of MPSoC is driven by the need for high-performance computing capabilities to handle complex DL models and real-time video processing, which typically exceeds the capabilities of microcontrollers.

Inclusion/Exclusion Documentation Rapid Review RunSoC

Predictable data-driven resource management: An implementation using autoware on autonomous platforms	14	Completed	The paper introduces ResCue, a data-driven resource management framework inspired by Heijunka, to address the challenges of memory limitations and ensure predictable execution in autonomous embedded systems by dynamically scheduling and reserving memory for program code.	No	The paper specifically mentions the NVIDIA AGX Xavier SoC as a representative AES platform. It highlights the "integrated architecture" of such SoCs, where computing units like CPUs, GPUs, and accelerators share the same RAM space. Key properties discussed include: Limited memory capacity. The presence of multiple computing units (CPU, GPU, DLAs). The impact of shared memory on data and memory management.	The paper focuses on the challenges of managing program code in memory, especially in the context of autonomous driving software like Autoware. Key properties and requirements analyzed include: Memory footprint of software modules. The need for temporal data availability (code must be in memory when needed). The need for spatial data availability (sufficient memory to hold required code). Task execution timing and deadlines.	This paper did not talk about mapping method The paper does not focus on mapping software components to semiconductor design partitions (i.e., hardware implementation). Instead, it proposes ResCue, a data-driven resource management framework, to manage software components (program code) within the memory space of the SoC. ResCue includes: A dynamic data scheduler to load/unload program code. A flexible memory reservation scheme. The methodology is inspired by Heijunka, a production leveling technique.	Again, the paper's focus is on memory management, not on mapping to semiconductor partitions. However, it identifies challenges related to managing software components (code) in memory: Limited memory space in SoCs. Ensuring temporal and spatial data availability. Preventing unpredictable output due to memory bottlenecks. Managing sporadic task releases caused by code loading delays.	The paper uses the Autoware autonomous driving software as a case study. The ResCue framework is implemented and evaluated on the NVIDIA AGX Xavier SoC. Evaluation metrics include: Meeting output deadlines. Jitter in output generation. Memory consumption. Overhead of the ResCue framework. End-to-end response time.	The paper draws inspiration from Heijunka, a production leveling system developed by Toyota, which is a well-regarded set of manufacturing best practices. It also builds upon the Logical Execution Time (LET) paradigm, which is a known model in real-time systems.	The primary benefit is improved predictability in autonomous embedded systems. Specific targets include: Ensuring temporal and spatial data availability. Meeting output deadlines. Minimizing output jitter. Reducing memory consumption. Improving response time.	The paper doesn't provide an explicit comparison or justification for using microprocessors over microcontrollers. The focus is on high-performance SoCs with multicore CPUs, GPUs, and accelerators, which are necessary to handle the complex computations and data processing demands of autonomous systems. This implicitly suggests that microcontrollers lack the processing power and memory capacity required for such applications.
Configuring ADAS Platforms for Automotive Applications Using Metaheuristics	21	Completed	Optimization strategy for given application and platform models to determine a mapping of tasks to the cores of the platform	Yes	The multi-core platform uses either time-sensitive networking or PCIe as communication backbone In the paper, experiments were conducted on a High Performance Computing cluster, which each node configured with 2xIntel Xeon Processor 2680v3 (10 cores, 2.60 GHz) and 128G memory	In this paper, we model the applications as a set of n periodic tasks A task is defined by the following components: 1. Core: If the task is pre-assigned to a core 2. Worst-case execution time 3. Period 4. Earliasst release time 5. Initial offset / displacement of task arrival times 6. Relative deadline	The algorithm includes two parts, the first part is to generate an initial solution, the second part is to find an optimal solution Find initial solution using greedy algorithm: 1. For each task, we consider the processor affinity constraint 2. We iterate through all cores and identify the core with the lowest utilization 3. We map the task to the respective core There are two different methods for finding optimal solution, for the detailed description, checkout the paper 1. Find Optimal Solution using simulated annealing: SA algorithm is a heuristic method that aims to optimize solutions by randomly selecting a candidate solution in the neighborhood of the current one +/- finding feasible solutions fast and with a lower cost function value 2. Find optimal solution using Genetic Algorithm: -> outperforms for very large TSN-based test cases 1. Encode each solution (chromosome) as an array where each entry (gene) contains information on the mapping, offset and deadline of a task/flow 2. randomly initialize N individuals 3. Evolve some selected candidates by using 4. Recombination 5. Mutation 6. Evolved candidates with better fitness will replace the parent population	The mapping could be just one solution, so the paper proposes method to optimize the mapping problems using various algorithms. The optimization target is defined by the cost function The optimization target is mono, could have a hybrid multi-objective metaheuristic that combines SA and GA and considers several optimization objectives, such as reducing the number of task preemptions.	The paper evaluate the proposed solutions in for both network setups PCIe and TSN using both realistic test cases and synthetic test cases. All experiments were conducted on a High Performance Computing cluster, with each node configured ith 2xIntel Xeon Processor 2660v3	Applications for ADAS system	We have proposed an optimization strategy that, given the application and platform models, determines a mapping of tasks to the cores of the platform and a static schedule of tasks on each core, such that the timing constraints are satisfied"	-
Design and verification of VCU system based on ISO26262	7	Completed	Allocate technical safety requirements to specific hardware elements and software elements to optimize system architecture design	No	A dual-core microcontroller	-	All safety-critical programs should be executed in the lockstep core The lock step cores (main and checker cores) operate independently in lockstep. The main and check core should use the same input data and execute the same program. All functional outputs of the main and check cores shall be compared cycle by cycle through the check comparator	-	-	This paper provides guideline of how to allocate technical safety requirements to specific hardware elements	Safety design provides more guarentee	-
Monitoring Framework to Support Mixed-Criticality Applications on Multicore Platforms	8	Completed	This paper proposes a low-overhead modular monitoring framework with a DSL to support mixed-criticality applications on multicore platforms by addressing resource contention and ensuring real-time deadlines.	Yes	The study focuses on multicore SoC platforms capable of hosting mixed-criticality applications. A key property is the challenge of ensuring isolation and predictability due to contention in shared resources like CPU, bus, and memory. The paper specifically mentions the Xilinx Zynq UltraScale multicore SoC as a platform for their framework instantiation.	The study analyzes the requirements for a monitoring framework that supports mixed-criticality applications. Key requirements include: Ensuring real-time applications meet deadlines. Improving resource utilization for best-effort applications. Low monitoring overhead. Support for heterogeneous resources.	The paper proposes a modular monitoring framework and a Domain-Specific Language (DSL) for configuring it. The DSL helps in configuring platform-specific parameters. The framework uses low-level hardware and software signals (e.g., Performance Monitor Counters (PMCs) and Linux Tracepoints) for monitoring.	The paper primarily focuses on the challenges of monitoring mixed-criticality applications on multicore platforms, rather than the challenges of mapping software to hardware partitions. Key challenges include: Resource contention leading to unpredictable delays. Ensuring isolation and predictability. Balancing the needs of safety-critical and best-effort applications. Monitoring overhead.	The proposed monitoring framework is evaluated on a Xilinx Zynq UltraScale multicore SoC running Linux. The evaluation includes an industry-inspired use case.	The paper doesn't explicitly mention specific industry standards for the "mapping process." However, it addresses the challenges of mixed-criticality systems, which is a relevant concern in industry standards like AUTOSAR.	The primary benefit is a low overhead monitoring framework for mixed-criticality applications on multicore platforms. The framework aims to: Ensure real-time applications meet deadlines. Improve resource utilization. Provide a flexible and configurable monitoring solution.	The paper focuses on multicore platforms and SoCs, which typically include microprocessors. It doesn't explicitly discuss the choice between microprocessors and microcontrollers but emphasizes the need for platforms that can handle the complexity of mixed-criticality applications and resource sharing.
An integrated hardware/software design methodology for signal processing systems	19	Completed	The paper introduces STMCM, a design methodology centered on lightweight dataflow principles, to streamline the development and implementation of signal processing systems on SoCs, addressing the challenges of hardware/software co-design and optimization through a DNN case study.	No	The study uses the Xilinx Zynq Z-7020 SoC as a specific platform for its case study. The SoC is described as a heterogeneous, embedded system-on-chip, featuring multicore CPUs and FPGA acceleration fabric. The properties investigated include resource constraints (processing and storage), computational complexity, and the need for efficient hardware/software co-design.	The software components are designed using LIDE-C, a C-language implementation of the Lightweight Dataflow (LWDF) APIs. Properties and requirements analyzed include: Design objectives and constraints. Memory sizes of buffers. Execution time of actors. Throughput.	The paper explains the mapping from software components to hardware components within the context of their proposed STMCM methodology. Here's a breakdown of the process: LIDE-C for Software The software components of the signal processing system are implemented using LIDE-C. LIDE-C is a C implementation of the Lightweight Dataflow (LWDF) APIs. This allows the developers to define the dataflow graph of the application in a structured manner using C code. LIDE-V for Hardware The hardware components are implemented using LIDE-V. LIDE-V is a Verilog implementation of the LWDF APIs. This enables the design of hardware actors and their interconnections using Verilog. Actor Implementation In LIDE-V, each hardware actor is implemented as two Verilog modules: the Actor Enable Module (AEM) and the Actor Invoke Module (AIM). AEM checks if there is sufficient data and space to fire the actor. AIM executes the actor's function.	Challenges include: Balancing design trade-offs (performance, resource utilization, power consumption). Efficiently mapping complex signal flow structures onto resource-constrained embedded devices. Integrating heterogeneous programming languages and platforms. Managing the diversity of design scales and dataflow techniques.	The paper presents a DNN implementation for vehicle classification on a Xilinx Zynq Z-7020 SoC as a case study. The DNN application is used to demonstrate the application of STMCM and the integration of various dataflow graph optimizations. Validation involves analyzing system throughput, memory footprint, and power efficiency.	The paper leverages the dataflow paradigm, which is a well-established model-based design approach for DSP systems. It also references and builds upon existing dataflow-based design methods and tools.	Benefits and targets of STMCM include: Facilitating experimentation across different levels of abstraction. Enabling rapid iteration on design options. Supporting hardware/software design trade-offs. Improving design agility and retargetability. Optimizing for real-time operation on resource-constrained SoC devices.	The paper does not explicitly provide a comparison or reasons for choosing a microprocessor over a microcontroller. The focus is on SoC platforms, which often integrate microprocessors along with other components, to handle complex signal processing tasks. The choice of SoC with a microprocessor is implied by the need for higher processing capabilities for DNN applications, rather than the more limited capabilities of typical microcontrollers.
Synthesis of a reconfiguration service for mixed-criticality multi-core systems: An experience report	19	Completed	Task to core allocation with failure handling on cores	No	A class of multi-core systems having asymmetric processing cores	Criticality of the tasks. Tasks are considered to be critical and non critical with two different priorities	1. A task is assigned to its primary core in the initial system state, where a core is responsible for executing only its primary tasks 2. We call a non-primary core a backup core of a critical task 3. When a core fails, the services assign the critical tasks that were previously assigned to that failed core to the operational cores 1. The services may kill and suspend temporarily one or more non-critical tasks on the operational cores during a reallocation process to ensure enough processing capacity for the reallocated critical tasks 2. A critical task is allowed to execute further on a backup core only if the primary core is in a failed tate 4. The services kill a critical task on a backup core (if that taksk is initialized or released) and cancel the assignment of that task to that backup core, whenever the primary core recovers from a temporary failure 5. The services reinstate a suspended non-critical task as soon as enough processing capacity for that task is regainned due to the recovery of a core from a temporary failure 6. The services permanently suspend a non-critical task when ti performas some harmful activities, such as illegal memory access	-	Uppaal Tiga-0.17 on a PC with an Intel Core i3 CUP at 2.4 GHz, 4 GB of RAM, and running 64-bit Windows 7	Not presented	The services suspend and reinstate the periodic executions of the non-critical tasks to ensure enough processing capacity for the critical tasks by mainting lookup tables, which keep track of processing capacity	-

Inclusion/Exclusion Documentation Rapid Review RunSoC

Secure separation of shared caches in AMP-based mixed criticality systems	12	Completed		Yes	The paper investigates SoC architectures that combine multiple operating systems (multi-OS) for automotive purposes. The SoC is described as an integrated circuit that combines all components of a computing platform or other electronic systems into a single chip. Key properties elaborated include the use of multi-processor SoCs (MP-SoC) where caches are often shared between multiple processor cores. The design focuses on Asymmetric Multiprocessing (AMP) paradigm, which implies a total split of every resource in the system, with each core having access to and working on a different partition of the main memory and hardware devices. The system configuration is static and set up during the boot phase, avoiding runtime reorganization or expansion of domains, to minimize the attack surface. The architecture considered uses memory maps for hardware accesses (memory mapped I/O). The SoC structure includes CPU subsystems, a common shared bus (CSB), and memory subsystems.	The study analyzes software components in the context of Mixed Criticality Systems (MCS), where different software components with varying functional or non-functional requirements (e.g., for safety or security) are partitioned and combined onto a shared platform. The software components are typically multiple operating systems (multi-OS), which can be of different types, such as real-time OS, general-purpose OS, and mobile OS. Each OS maintains its own memory and physical devices, forming an independent OS-domain. A critical requirement analyzed is the secure separation of functionality, especially to prevent interference and ensure reliability in safety-critical environments. The paper focuses on denial-of-service (DoS) attacks targeting the availability of system assets through exploitation of shared last-level caches (LLC).	The core methodology used for mapping is based on a "suitable mapping scheme in the intermediate address space of an asymmetric multiprocessing environment". This involves using the Memory Management Unit (MMU) and translation tables (TT) to map physical addresses to virtual address spaces. The second stage MMU enforces the assignment of resources to specific OS-domains. The paper proposes a novel "Domain Block Memory Mapping" method to assign way-sets (WS) in the cache to OS-domains. This method introduces "Domain Blocks" (DBs), which are memory regions assigned to specified regions of way-sets in the LLC. These DBs are mapped to OS-domains using the second stage MMU, with the input address space (IPA) being consecutive and the output addresses (PA) generated according to the proposed pattern. The system configuration, including MMU initialization and memory mapping, is statically defined and initialized during the boot phase in privileged hypervisor mode.	A significant challenge is ensuring "strict separation" in the presence of shared cache levels, which can be vulnerable to interference due to pre-qualified and non-reconfigurable hardware components. This interference can lead to denial-of-service (DoS) attacks by adversaries exploiting the caching infrastructure, potentially causing starvation of an OS-partition. Another challenge lies in the implementation of the proposed mapping scheme, which requires careful consideration of system and architectural improvements. Specifically, the initialization of the second stage MMU must be completed before OS-images or configuration files are loaded, and these images might be larger than the defined Domain Blocks, requiring careful loading with respect to the pattern. Direct Memory Access (DMA) of other subsystems connected to the common system interconnect also presents a challenge, as these accesses are performed on consecutive physical memory areas. To use DMA capabilities, the maximum transfer size needs to be limited and aligned to the Domain Blocks, or the DMA device needs to be aware of the mapping scheme and able to translate intermediate physical addresses	The effectiveness of the proposed memory mapping technique is evaluated through an empirical analysis of its separation capabilities. The method aims to mitigate interference on shared caches. The primary validation method involves quantifying the effects of the proposed method by measuring the latency for each memory access of a certain processor. These measurements are executed using specific memory access patterns designed to provoke interference or starvation effects. An experimental setup is used to demonstrate the impact. A Pandaboard incorporating the Texas Instruments OMAP5 SoC, with two ARM Cortex-A15 processors, serves as the platform. Two OSs, an "adversary" and a "victim" OS-domain (both upstream Linux Kernel), run on the platform, and methods to measure the proposed effects are implemented at the OS level. The outcome of the measurements is intended to show that DoS attacks and starvation are possible, and that the proposed solution is effective	The study maps the novel mapping method to security models commonly applied for mixed criticality systems. Specifically, it refers to "Multiple Independent Levels of Security (MILS)" as a well-known security model for high-assurance systems, particularly in aircraft, space, and defense purposes. The MILS model incorporates properties such as data isolation, control of information flow, periods processing, and fault isolation. Security measures to enforce these properties need to implement attributes known as NEAT (Non-bypassable, Evaluable, Always invoked, Tamper proof). The proposed method for separating shared caches is designed to fit with the NEAT attributes to be compliant with the MILS security model.		
Optimal performance prediction of ADAS algorithms on embedded parallel architectures	6	Completed	Performance prediction of mapping	Yes	Nvidia Tegra K1 Architecture Texas Instrument TDA3x SoC	Kernel: blocks of algorithm	This paper did not talk about mapping method It talks about a good strategy of performance prediction 1. Classification of the level of computing instruction of kernel 2. Computing Time Prediction with the computing profile of a kernel 3. Prediction for consecutive Kernels: 1. Interval arithmetic (More straightforward) 2. Probabilistic interval (Provides a degree of confidence) 4. We predic on different processors to test which processor / core provides shorter execution time, means the performance is better	The alogrithm only focuses on parallelism class P_0, no high-level is estimated yet	Use real measurement to compare with the predicted result	Use pedestrian detection application using algorithm uses histogram of oriented gradients	Target: Identify the best mapping method	-
System-on-Chip based highly integrated powertrain control unit for next-generation Electric Vehicles: Harnessing the potential of Hybrid Embedded platforms for Advanced Model-Based Control Algorithms	13	Completed	The paper proposes using SoCs to create a highly integrated powertrain control unit for electric vehicles, combining ECU functions and enhancing control with complex models.	Yes	The study focuses on SoCs that integrate both processors and FPGAs. The key properties emphasized are their ability to provide high processing power, parallelism, and versatile interfacing capabilities. The paper specifically mentions Xilinx Zynq and Altera Cyclone V SoCs, highlighting their suitability for automotive applications due to their performance and development resources.	The study analyzes software components in the context of powertrain control units. Key properties and requirements include the need for real-time processing, handling complex control algorithms, managing communication between different vehicle systems, and ensuring safety and reliability. The paper also discusses the importance of efficient software development workflows, including model-based development and the use of tools like Simulink and AUTOSAR.	This paper did not talk about mapping method To address V-model, the following technology is used: The primary methodology discussed is Model-Based Development (MBD), which uses tools like Simulink to design and simulate software components. The paper also explores the use of High-Level Synthesis (HLS) tools to translate C/C++ code into hardware description languages (VHDL, Verilog) for implementation on FPGAs. For processor implementation, Real-Time Operating Systems (RTOS) and Hardware Abstraction Layers (HAL) are considered.	One key challenge is efficiently utilizing the hardware's capabilities, particularly the parallelism of FPGAs, to meet real-time requirements. Another challenge is ensuring safety and integrity, especially for safety-critical applications, which requires careful consideration of both hardware and software mechanisms. The paper also mentions the complexity of integrating different software components and the need for tools and workflows that support this integration.	The paper presents a concept demonstrator for a highly integrated powertrain control unit. This demonstrator integrates several functions, including traction control and torque distribution, onto a Xilinx Zynq SoC. The effectiveness is demonstrated through the implementation of these functions and the discussion of how advanced algorithms, such as those using neural networks, can be integrated.	Yes, the paper emphasizes the importance of the AUTOSAR standard for software development, which promotes modularity, abstraction, and reusability. It also highlights the significance of the ISO-26262 standard for functional safety in automotive systems, discussing how the proposed solutions and development processes align with these standards.	The proposed method aims to reduce the number of ECUs in modern electric vehicles. This reduction leads to decreased system complexity, lower energy consumption, and reduced development and material costs. The approach integrates the functionality of several ECUs into a single, highly integrated ECU (mECU). It also seeks to harmonize the increasing demands of modern vehicles with the capabilities of emerging hardware and software solutions.	The paper explains that for the Traction-ECU, functionalities involving a large amount of logic, state machines, diagnostics and parameters are best suited for implementation on a processor. This is due to the processing demands and the complexity of these tasks.
Build real-time communication for hybrid dual-OS system	10	Completed	This paper proposes a real-time RPC mechanism (RTRG-RPC) for hybrid multi-OS systems on SoCs, using techniques like SGI message transforming and priority-swapping to achieve time predictability and security.	Yes	The paper doesn't delve into the detailed specifics of SoC design. It emphasizes the property of SoCs enabling the integration of multiple domains and operating systems (hybrid multi-OS systems). The use of ARM TrustZone technology within SoCs for providing isolated environments (normal world and secure world) is a key aspect discussed	The study primarily analyzes the properties and requirements of inter-OS communication, specifically Remote Procedure Calls (RPCs) in hybrid multi-OS systems. Key requirements include: Efficiency and predictability, especially for the real-time OS (RTOS). Safety and security, including preventing interference between OSs and protecting against malicious attacks.	This paper did not talk about mapping method The paper focuses on the software architecture and communication mechanisms within a hybrid multi-OS system on an SoC, rather than the mapping of software components to semiconductor design partitions. It utilizes ARM TrustZone as a hardware-based security extension to provide isolation between the RTOS and general-purpose OS (GPOS). It proposes a real-time RPC scheme (RTRG-RPC) with mechanisms like SGI (Software Generated Interrupt) message transforming, interrupt handler RPC servicing, and priority-swapping to manage communication.	Again, the paper's focus is on inter-OS communication challenges, not the mapping to semiconductor partitions. Key challenges related to communication include: Ensuring time efficiency and predictability in the presence of a non-real-time GPOS. Balancing the need for security with the need for efficient communication. Managing priority inversions and scheduling latency.	The proposed RTRG-RPC scheme is implemented and evaluated on a TrustZone-based hybrid multi-OS system called TZDKS. Performance evaluations are conducted to demonstrate RTRG-RPC's ability to achieve real-time predictability, reduce priority inversion, and provide security without sacrificing efficiency.	The paper mentions AUTOSAR (Automotive Open System Architecture) as an example of an industry specification that promotes ECU consolidation. While not a direct "mapping process" standard, AUTOSAR highlights the trend towards integrating multiple functionalities, which aligns with the paper's SoC-based consolidation approach.	The primary benefit is enabling time-predictable and secure communication in hybrid multi-OS systems. This is crucial for consolidating multiple domains in automotive/avionic/robot systems, leading to reduced cost, space, weight, heat generation, and power consumption.	The paper does not explicitly discuss the choice between microprocessors and microcontrollers. Instead, it focuses on the use of SoCs, which can include microprocessors, to host hybrid multi-OS systems. The discussion is more about the need for a platform capable of supporting the complexity of running multiple OSs and providing hardware-based isolation (like TrustZone).
Adaptive Vehicle Detection for Real-time Autonomous Driving System	6	Completed	Not relevant Designing a partial reconfiguration controller to accelearate the reconfiguration process on Zynq SoC for real-time vehicle and pedestrian detection	No	Zynq SoC	-	-	-	-	-	-	-
E/E Architecture Synthesis: Challenges and Technologies (manually added)	26	Open										
Portable implementations for heterogeneous hardware platforms in autonomous driving systems	-	Open										
CEDR-API: Productive, Performant Programming of Domain-Specific Embedded Systems	10	Completed	This paper introduces an API-based programming model for the CEDR framework to improve productivity and performance in developing applications for Domain-Specific System on Chips (DSSOCs).	Yes	The study focuses on Domain-Specific System on Chips (DSSOCs). These DSSOCs are heterogeneous, integrating a rich set of accelerators to enhance performance. The key property emphasized is the ability of DSSOCs to increase productivity by focusing on a specific application domain.	The study analyzes the properties and requirements of software abstractions and programming models for DSSOCs. Key requirements include: Support for multiple users to coexist and interleave applications. Effective utilization of compute resources in a dynamic way. Hardware-agnostic programming abstractions. The need for intelligent runtime systems to arbitrate and schedule resources.	The paper introduces CEDR, an open-source, unified compilation and runtime framework for DSSOC architectures. CEDR allows for the co-design of applications, scheduling heuristics, and accelerators. The paper proposes an API-based programming methodology to improve productivity and expressiveness.	The paper focuses on the challenges of programming and utilizing heterogeneous computing systems effectively. Key challenges include: The difficulty of programming heterogeneous systems. Inefficiencies caused by runtime resource contention. The need for software abstractions that are agnostic to the underlying hardware. Limitations of DAG-based representations for capturing control flow.	The study evaluates the proposed API-based CEDR using real-world applications from the domains of radar processing, communications systems, and autonomous vehicles. Experiments are conducted on Xilinx ZCU102 MPSoC and Nvidia Jetson AGX Xavier development boards. The results demonstrate that the API-based CEDR achieves a runtime overhead reduction of 19.5% compared to the original DAG-based CEDR	The paper doesn't explicitly mention specific industry standards for the "mapping process" in the way of hardware partitioning. However, it emphasizes the need for hardware-agnostic programming abstractions, which aligns with the general principle of portability and abstraction in software engineering.	The primary benefit is improved productivity in developing applications for DSSOCs. The method targets more rapid integration of new applications without sacrificing performance. It also aims to address the limitations of DAG-based representations and provide a more expressive API-based programming model.	The paper does not explicitly discuss the choice between microprocessors and microcontrollers. Instead, it focuses on the use of DSSOCs, which integrate various processing elements (including microprocessors and accelerators), to handle complex computing tasks.
A flexible software framework for dynamic task allocation on MPSoCs evaluated in automotive context	12	Completed	1. First Come - First Serve 2. Mapping considering priority of tasks	Yes	Heterogeneous Multiprocessor System-on-Chip: it consists an ARM processor and multiple Microblazes	Functional Unit: Tasks. Each task should have its unique id, the respective priority and the data container. *The property of the data container is task specific. Need to be defined according to the focus of application	Dynamic mapping 1. First-come-first-serve algorithm 1. The new tasks are distributed to already-engaged cores 2. Other cores are activated when a defined threshhold load_max is reached 3. If a core undercuts a minimal load load_min, the tasks on this core are distributed to other cores. 2. Priority algorithm 1. Priorization: Define the priorities of tasks in an application. This process could be selective and subjective. (e.g. when car collision algorithm, the region in front of the host car has the highest prority.) 2. Task allocation: Each task container consists of an unique id, the respective priority and a data container. 3. Data transmission: data of each individual task container is directly transmitted over the network in the form of data-packet. 4. Data processing on Microblaze: Each microblaze contains the same software. 1. The maximum number of tasks running on each microblaze is configurable. 2. A function selects unprocessed tasks according to their priority. 3. If the tasks have the same priority, they are exectued according to the order the arrived. 4. If the execution of the task detects critical information in a	This approach (priority algorithm) is really application specific. Need to have really careful designs according to the application requirements	Evaluation method: empiric time measurement for a simulated car-collision situation	The paper uses a car-collision analysis algorithm as an example	Dynamic mapping allows adding, moving and removing tasks at runtime	Due to the rising demands for a large number of various applications, the trend towards integration of Multiprocessor Systems-on-Chip (MPSoCs) in embedded systems increases In recent times, Networks-on-Chip (NoCs) arise as the most promising communication infrastructure for MPSoCs with a high number of heterogeneous processing elements (PEs)
Why Comparing System-Level MPSoC Mapping Approaches is Difficult: A Case Study	8	Completed	Comparing execution time of two frameworks	Yes	Texas Instrument's keystone II	At the application level, programs are described using the KPN model 1. Node: applications are partitioned into different processes, which encapsulate the different parts of the computation 2. Edges: In the KPN model the data exchanging among the processes are abstracted by defining communication channels between processes that act as unbounded FIFO buffers	Several mapping algorithms are employed: Genetic algorithm, Group-based mapping, load balancer, affinity, throughput and simulated annealing	-	The framework Sesame includes performance evaluation part. It uses various benchmarks to indicate the performance of different mapping methods	-	Genetic algorithms found better mappings than heuristic-based ones, with a relative performance ranging from 1× to 5×. At the same time, there are two extreme cases where the heuristic-based algorithm GBM outperforms genetic algorithms. Comparing the computation time, the genetic algorithms take significantly more time to produce the mapping, ranging from 0 to 2 orders of magnitude in the examples considered.	-