ART: Autonomous Radar Transceiver Architecture for Cost-Optimized Satellite Radar Systems

Michael Atzmüller
Infineon Technologies Austria AG
Linz, Austria
michael.atzmueller@infineon.com

Rainer Findenig
Infineon Technologies Austria AG
Linz, Austria
Rainer.Findenig@infineon.com

Bernhard Greslehner-Nimmervoll
Infineon Technologies Austria AG
Linz, Austria
Bernhard.Greslehner-Nimmervoll@infineon.com

Daniel Große
Institute for Complex Systems
Johannes Kepler University Linz
Linz, Austria
Daniel.Grosse@jku.at

Abstract

Distributed ("satellite") automotive radar systems employ multiple front and corner sensors whose raw-data fusion can improve angular resolution and imaging quality. In such central-processing architectures, however, single-chip radar *Monolithic Microwave Integrated Circuit*s (MMICs) underutilize their integrated signal-processing resources, whereas transceiver MMICs still require a dedicated per-module microcontroller for configuration, diagnostics, and control tasks. This paper proposes *Autonomous Radar Transceiver* (ART), a transceiver-centric radar MMIC architecture that enables autonomous module-level operation without an external microcontroller. ART exploits otherwise idle *non-sequencing windows* of an on-chip *RISC-V Sequencer* (RVS) by time-partitioning its RISC-V CPU between (i) deterministic *Frequency-Modulated Continuous-Wave* (FMCW) sequencing and (ii) non-time-critical module control tasks. As a result, ART reduces satellite-radar module *Bill of Materials* (BOM) and integration complexity by consolidating control and sequencing into a single programmable chip while preserving sequencing determinism.

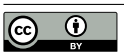
CCS Concepts

• **Hardware** → **System on a chip**; • **Computer systems organization** → *Special purpose systems*; *Real-time system architecture.*

Keywords

Satellite Radar, RISC-V Sequencer, Centralized Radar Processing

ACM Reference Format:
Michael Atzmüller, Rainer Findenig, Bernhard Greslehner-Nimmervoll, and Daniel Große. 2026. ART: Autonomous Radar Transceiver Architecture for Cost-Optimized Satellite Radar Systems. In *Great Lakes Symposium on VLSI 2026 (GLSVLSI '26), June 22–24, 2026, Canandaigua, NY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3787109.3815262



GLSVLSI '26, Canandaigua, NY, USA

ACM ISBN 979-8-4007-2431-2/26/06
https://doi.org/10.1145/3787109.3815262

1 Introduction

Commercial automotive radar chips predominantly follow one of two architectural approaches. Transceiver *Monolithic Microwave Integrated Circuit*s (MMICs) provide the radar frontend but typically omit signal-processing units and therefore require an external control device, commonly a microcontroller [12]. In contrast, single-chip MMICs integrate a radar transceiver, a microcontroller, signal-processing accelerators, and, optionally, hardware security features or high-speed data interfaces (e.g., Ethernet) on a single die [22]. Single-chip MMICs are well suited to lower-performance tasks and are commonly deployed in *edge processing* systems, where signal processing is performed within the radar module and only processed data are forwarded to a central *Electronic Control Unit* (ECU) [7]. Transceiver MMICs, by comparison, are preferred when the required signal-processing performance exceeds what is available on a single-chip MMIC and therefore external processing is required in any case [10]. In both architectures, a microcontroller, either integrated in single-chip MMICs or external in transceiver MMICs, typically handles configuration, control, diagnostics, and status reporting.

As stated in [23], future cars may be equipped with up to twelve radar chips sensing multiple directions. In such configurations, fusing sensor data becomes particularly advantageous because it enables radar sensor networks. At the same time, edge processing becomes less applicable, because fusing raw radar data offers substantial advantages, including improved angular performance, as discussed in [4, 23]. Consequently, the trend shifts toward *central processing*, where raw radar data are streamed to a central ECU that performs sensor fusion and signal processing. In the automotive context, this principle is known as *satellite radar* [7, 21].

Due to the high-volume production of satellite radar modules, a cost-optimized system architecture is required. Considering the two chip types described above, transceiver MMICs are preferable to single-chip MMICs in satellite configurations because the on-chip signal-processing units of the latter are largely underutilized when raw data are forwarded to the ECU. However, conventional transceiver MMICs remain suboptimal because each radar module still requires an additional microcontroller to execute module-level control tasks. Adding a dedicated microcontroller increases *Bill of Materials* (BOM) cost and PCB area and introduces an additional software stack and update path at the module level.

This paper introduces a novel architecture for radar chips, the *Autonomous Radar Transceiver* (ART) MMIC, specifically optimized for satellite radar systems. ART is derived from a transceiver MMIC but enables autonomous operation comparable to a single-chip MMIC, thereby eliminating the need for an external microcontroller. For a representative present-day configuration with one front radar and four corner radars [23], eliminating one microcontroller per module removes five microcontrollers per vehicle; under forecasted twelve-sensor configurations, the corresponding saving increases to twelve microcontrollers per vehicle.

The autonomy of ART is not achieved by integrating a dedicated microcontroller core into the transceiver MMIC. Instead, ART exploits existing on-chip hardware resources that are temporarily underutilized during typical operation. Our approach builds on the *RISC-V Sequencer* (RVS) presented in prior work [3]. Specifically, ART leverages the RISC-V CPU within the RVS during *non-sequencing windows*, identified through architectural analysis of the radar application cycle, to execute low-performance module-level control tasks between measurement bursts.

In addition to removing the external microcontroller, ART provides system integrators with a simpler and more unified programming model. Traditional transceiver MMICs require separate software development for the external microcontroller and for the sequencing unit. With ART, both control and sequencing tasks execute on the same CPU and therefore do not require explicit synchronization. Furthermore, switching between control and sequencing does not require interrupts or traps; instead, the system transitions between these modes via direct invocation of the corresponding functions.

In summary, this paper makes the following contributions:

- We introduce ART, a transceiver-centric radar MMIC architecture that enables autonomous module-level operation in satellite radar systems without an external microcontroller.
- We provide an architectural analysis of the radar application cycle and identify non-sequencing windows of the on-chip RVS compute resources for executing module-level control tasks.
- We evaluate architectural feasibility in a proprietary SystemC-based *Virtual Prototype* (VP) modeling an automotive radar transceiver MMIC that is already deployed in vehicles.
- We describe the resulting programming and integration model and discuss how ART reduces synchronization effort between control execution and deterministic radar sequencing.

This paper is structured as follows. Section 2 discusses related work. Section 3 introduces satellite radar architectures and motivates central processing. Furthermore, it reviews *Frequency-Modulated Continuous-Wave* (FMCW) sequencing and the RVS. Section 4 presents ART at the hardware and software level. Section 5 describes the experimental setup and evaluates feasibility and integration implications. Section 6 concludes.

2 Related Work

This section reviews prior work on satellite radar architectures, radar MMICs integration, and sequencer programmability. While prior work motivates raw-data fusion and satellite radar modules, and recent sequencer architectures improve programmability, cost-optimized satellite radar systems remain insufficiently explored. ART addresses this gap by time-partitioning a programmable RVS between deterministic sequencing and non-time-critical module-level control.

Satellite radar and raw-data fusion. Prior work has motivated radar sensor networks and the benefits of raw-data fusion, including improved angular performance relative to isolated sensors and higher-level fusion of locally processed outputs [4, 23]. Satellite radar architectures operationalize this trend by streaming raw radar data to a central ECU for joint processing [21].

Radar MMICs integration trends. Automotive radar chips span transceiver-only MMICs that require external control [12] and single-chip MMICs integrating control and signal processing [22]. ART targets satellite radar configurations in which signal-processing units of single-chip MMICs are underutilized, yet per-module autonomy remains necessary.

Sequencer architectures and programmability. Conventional radar sequencers employ domain-specific instruction sets and fixed-function control logic (e.g., [8]), which offers high determinism but limited programmability. The RVS [3] replaces a domain-specific sequencer with a RISC-V-based design extended by *Control and Status Register*s (CSRs) [25], enabling higher-level programmability. ART builds on this foundation by time-partitioning the RISC-V CPU between deterministic sequencing and non-time-critical module-level control.

3 Background

This section introduces (i) satellite radar as the target system architecture and (ii) deterministic FMCW sequencing as the timing-critical function implemented in radar MMICs. These concepts motivate the two-mode execution model of ART presented in Section 4.

3.1 Satellite Radar

In the context of automotive radar networks, *satellite radar* denotes a central-processing architecture in which a radar module (the "satellite") streams *raw* measurement data to a central ECU, where computationally intensive signal processing and higher-level interpretation are performed. In contrast, many production radar systems rely on *edge processing*, where signal processing is executed within the radar module and only processed outputs (e.g., detections or object lists) are forwarded to the ECU [7].

3.1.1 From Edge Processing to Central Processing. Automotive radar has evolved from stand-alone sensing units toward multi-sensor setups and, increasingly, networked perception systems. Enabled by substantial cost reductions in millimeter-wave components and MMICs, the research focus has shifted from hardware-centric aspects to system-level concepts such as radar sensor networks [23]. In such networks, spatial diversity across multiple viewpoints can improve robustness and detection performance and can provide significantly enhanced angular performance compared to isolated sensors [4, 23]. While fusion at high abstraction levels (e.g., combining object lists generated locally by each radar module) is well

established, it inherently discards information due to local processing. Consequently, recent work emphasizes *raw data fusion* [23], where measurements are combined before irreversible information reduction. Satellite radar architectures directly support this paradigm by separating the radar sensing frontend from the main compute and fusion stack: instead of outputting a fully processed point cloud or object list, the satellite radar module transmits raw data to a central ECU, enabling joint signal-level processing across sensors.

The understanding of satellite radar used in this work follows the definition and architecture described in [21]. A typical satellite radar module consists of a transceiver MMIC and the associated antennas, a high-speed serializer, a small microcontroller, and a power supply. In [21], an ARM Cortex-M0 controls the radar chip via *Serial Peripheral Interface* (SPI) and the serializer via *General Purpose Input/Output*s (GPIOs). The raw radar output is streamed to the serializer using the MIPI CSI-2 standard [18], which then transmits the data to the central ECU for subsequent signal processing.

3.2 FMCW Radar Sequencing

This section summarizes FMCW radar sequencing. FMCW is a widely established principle for measuring range, velocity, and angle by transmitting electromagnetic waves and analyzing received reflections [15, 19]. The transmit frequency varies over time according to a defined pattern, typically in the form of *ramps*. Several hundred to several thousand such ramps are concatenated to form a *ramp scenario* for measurement [15]. Due to the need for precise timing, FMCW radar chips integrate a dedicated *sequencer* to orchestrate ramp execution and synchronize hardware components. The sequencer is a hardware accelerator inside the radar chip that generates cycle-accurate control signals and distributes them to subsystems such as the transmitter, receiver, monitoring units, and power amplifiers during measurement [3, 8]. This ensures deterministic operation and guarantees that each unit receives the correct configuration values exactly when needed. Traditionally, sequencers have been implemented as domain-specific hardware units with a custom *Instruction Set Architecture* (ISA) optimized for radar operations (e.g., the *Domain-Specific Sequencer* (DSS) [8]). While such designs offer high execution speed, they suffer from a limited software ecosystem and limited flexibility because changes to the instruction set or signal interfaces typically require hardware modifications.

3.2.1 RISC-V Sequencer. To address the limitations of DSSs, prior work [3] proposed the RVS, a sequencer architecture based on the modular and extensible RISC-V ISA [24]. The RVS replaces a fixed-function decoder with a general-purpose RISC-V processor extended by custom CSRs [25] designed for control-signal generation. This approach enables ramp scenarios to be programmed in high-level languages such as C++, abstracting low-level register access and enabling new features, including adaptive ramp scenarios in which ramps are modified at runtime in response to environmental events (e.g., interference or temperature changes).

4 Proposed Architecture

This section presents ART, a transceiver-centric radar MMIC architecture that achieves autonomous module-level operation by reusing the RISC-V CPU in the RVS. We first introduce the core idea of two-mode execution with strict time partitioning (Section 4.1), then describe the required hardware extensions (Section 4.2) and finally, the resulting software model (Section 4.3).

4.1 General Idea

The key observation underlying ART is that the RISC-V CPU in the RVS is active primarily while executing ramp scenarios. Between measurement bursts, the sequencer is idle or can be power-gated, creating an opportunity to reuse the same CPU for module-level control tasks. Whether this reuse is feasible depends on the radar duty cycle and the duration of the *non-sequencing window*. Representative parameters reported in the literature indicate that non-sequencing windows on the order of tens of milliseconds can occur in many configurations (e.g., [2, 9]). For instance, the ETSI EN 303 661 standard which covers the 76 GHz to 77 GHz band [6] specifies a frame time of 50 ms and reports typical duty cycles between 20 % and 50 %, which corresponds to non-sequencing windows, i.e., the remainder of the 50 ms interval outside active measurement, of approximately 25–40 ms. These windows can be exploited for control workloads, such as configuration management, serializer control, and periodic monitoring, provided that control-mode execution is strictly prevented from interfering with sequencing.

We therefore propose to use the RVS not only for deterministic FMCW sequencing, but also for module-level control tasks during the non-sequencing window. The goal is to remove the dedicated external microcontroller from each satellite radar module while maintaining the timing guarantees of the sequencer. In ART, the RVS alternates between two operating modes: (i) a *sequencing mode*, in which the RISC-V core executes the ramp scenario and generates cycle-accurate control signals, and (ii) a *control mode*, in which the same core performs module-level control tasks that would otherwise be handled by a microcontroller.

An alternative would be to delegate module-level control tasks to the on-chip housekeeping CPU. However, as the housekeeping CPU is already used outside sequencing windows, this would cause concurrent execution with the sequencer, introducing non-deterministic bus activity during the ramp scenario that could disturb radar data. Moreover, since the housekeeping CPU is in the MMIC manufacturer's domain, assigning it system-integrator tasks merges separate software domains.

The key requirement is that control-mode execution must never compromise deterministic sequencing. In ART, this is achieved through explicit time partitioning aligned to the radar duty cycle, lightweight mode switching in software, and a well-defined interface between the RVS and the housekeeping CPU.

4.2 ART Hardware

The proposed radar chip architecture is based on a transceiver MMIC that integrates a RVS as programmable sequencer. Compared to a conventional satellite radar module, the external microcontroller is eliminated and its responsibilities are assumed by the RISC-V core inside the RVS. Concretely, ART requires four architectural extensions relative to a transceiver MMIC: (i) integration of the RVS as sequencer CPU, (ii) an interface between the housekeeping CPU and the RVS, (iii) access to *Non-Volatile Memory* (NVM) for

sequencing and control firmware, and (iv) peripheral interfaces for companion components. The following paragraphs describe these extensions.

RVS integration. We assume that the RVS as presented in [3] is integrated as the chip-internal sequencer responsible for generating cycle-accurate control for transmitter/receiver operation and related analog front-end blocks. No changes are required for the timing-critical datapath; the RVS continues to execute the ramp scenario with the same deterministic behavior as in sequencing-only operation.

Interface to the housekeeping CPU. Many automotive radar MMICs include an internal controller (hereafter referred to as *housekeeping CPU*) that manages chip initialization, calibration routines, safety monitoring, and configuration of analog/mixed-signal blocks. In ART, the RVS exchanges configuration and status information with the housekeeping CPU. This connection can be realized through an internal bus or a dedicated mailbox mechanism; the specific implementation is not assumed in this work.

Non-volatile program storage. To replace the external microcontroller, the RVS must execute a control firmware in addition to the ramp-scenario code. ART therefore requires access to NVM, integrated on-chip or provided off-chip, to store both the sequencing program and the control-mode firmware.

External I/O control. To fully replace the external microcontroller at the module level, the RVS must provide peripheral control interfaces, including SPI, *Inter-Integrated Circuit* (I2C), and GPIOs. These enable the configuration and supervision of companion components, in particular the serializer, as well as the execution of low-level module control operations (e.g., reset and enable signaling). Furthermore, they enable the RVS to assume the communication responsibilities between the ECU and the radar module, which are routed via the serializer and were previously managed by the external microcontroller.

Overall, the silicon overhead of ART is confined to RVS-centric storage and connectivity mechanisms rather than additional signal-processing accelerators or a dedicated microcontroller core. Fig. 1 illustrates the proposed architecture at the MMIC and module level. The MMIC integrates the RVS, a housekeeping CPU, and the radar frontend; the housekeeping CPU provides configuration and chip-internal services, while the RVS performs deterministic sequencing and schedules and executes module-level control tasks. The RVS exchanges status and configuration with the housekeeping CPU via an internal channel and controls companion components such as the serializer through peripheral interfaces (e.g., SPI/I2C/GPIOs). The raw radar data path remains unchanged: data coming from the frontend are streamed to the serializer via CSI-2, and the serializer transmits the data to the ECU.

4.3 ART Software

The software concept follows the two-mode operation of the RVS: a control-mode program executes during the non-sequencing window and invokes sequencing-mode execution when a measurement cycle is due. Mode switching does not require a complex interrupt or trap model; instead, sequencing is invoked explicitly from the

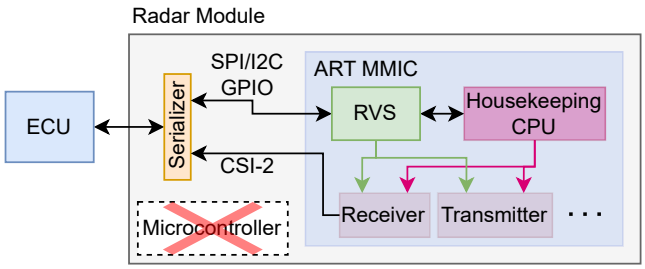


Figure 1: Simplified block diagram of a radar module comprising an ART MMIC, a serializer, and additional components (e.g., power supply; not shown). The ART MMIC integrates the RVS, a housekeeping CPU, and the radar frontend. The housekeeping CPU configures the frontend, while the RVS performs deterministic sequencing and executes module-level control tasks, exchanging status and configuration with the housekeeping CPU via an internal channel and controlling external components via SPI/I2C/GPIOs.

control firmware, which keeps the control flow simple and analyzable.

Control mode. During the non-sequencing window between radar measurement bursts, the RISC-V core executes a lightweight control loop. Typical responsibilities include: (i) receiving configuration commands from the central ECU (e.g., selecting between predefined scenarios such as urban and highway), (ii) configuring and monitoring components such as the serializer, (iii) interacting with the housekeeping CPU (e.g., requesting calibrations, retrieving temperature/status, and applying register configurations), and (iv) preparing parameters for the next ramp scenario.

Sequencing mode. When a measurement cycle starts, the control firmware transfers execution to a dedicated sequencing function that runs the ramp scenario using the RVS control/status registers as in [3]. This sequencing function requests the housekeeping CPU to perform required hardware preparation and then waits for a trigger to start the ramp scenario. During sequencing mode, the software refrains from non-deterministic activities such as communication tasks and focuses solely on deterministic ramp execution. However, it is possible to invoke the housekeeping CPU for configuration tasks, as long as the time determinism of the ramp scenario is guaranteed. After the ramp scenario completes, the RVS returns to control mode.

Fig. 2 shows two radar application cycles. The one shown in Fig. 2a shows the application cycle for a conventional radar module including a microcontroller and a typical sequencer like the DSS or RVS (only used for sequencing). Fig. 2b shows the same application cycle executed on a radar module employing an ART MMIC. In both cases, the execution windows of the housekeeping CPU are equal (in Fig. 2a see second row, in Fig. 2b see first row). However, in the conventional application cycle, the microcontroller takes over configuration and communication tasks whilst in the ART architecture, those are performed by the RVS. One can see, that the previously idle windows (light yellow) and sequencing window

(light purple) of the sequencer in Fig. 2a exactly match the control mode windows and sequencing mode window of the RVS in Fig. 2b.

5 Evaluation

This section evaluates the feasibility and system-level implications of ART. The whole evaluation is performed on a SystemC-based VP. We first summarize the experimental setup and model fidelity. We then evaluate determinism and disturbance avoidance during sequencing, followed by feasibility of microcontroller elimination via idle-time reuse. Next, we discuss the simplified programming model and the resulting module-level cost and integration impact. Finally, we discuss limitations of the ART architecture.

5.1 Experimental Setup

The results are derived from implementing and exercising ART in a proprietary SystemC-based [1] VP modeling an automotive radar transceiver MMIC that is deployed in production vehicles. This VP enables architectural modifications and controlled experiments at the module-control and sequencing level. We use the VP to (i) model the two-mode execution of the RVS, (ii) execute representative control workloads in control mode, and (iii) validate that sequencing-mode execution preserves deterministic ramp timing under the proposed time partitioning. Regarding model fidelity, the RVS is modeled using a cycle-accurate RISC-V CPU model such that instruction execution and timing-critical sequencer interactions can be evaluated at clock-cycle granularity. The surrounding communication interfaces are modeled at the transaction level using SystemC *Transaction Level Modeling* (TLM), following established virtual-prototyping practice for mixed-abstraction architecture exploration and software validation [5, 11, 14, 17]. This modeling approach is sufficient to assess the feasibility of the proposed two-mode execution and to verify that control-mode activity does not perturb deterministic sequencing timing in sequencing mode.

5.2 Determinism and Disturbance Avoidance during Sequencing

A practical advantage of the two-mode model is that it structurally prevents control communication activity during ramp execution. In conventional modules, communication between the external microcontroller and the radar chip may overlap with ongoing measurements, which requires careful integration to avoid unintended timing effects or configuration hazards. In ART, sequencing mode is entered explicitly and excludes non-sequencing tasks; therefore, no control communication is issued during the time-critical ramp scenario by design. This separation simplifies verification of module-level timing determinism because the conditions under which the sequencer runs are fixed and repeatable.

Furthermore, in conventional transceiver-centric modules the external microcontroller must access MMIC-internal registers over SPI/I2C. In ART, the same operations execute on the RVS and therefore translate into on-chip register accesses; as a result, the microcontroller-to-MMIC control link and its associated off-chip transactions are eliminated by construction. This does not affect the raw-data path, which continues to stream from the MMIC to the serializer and onward to the ECU.

5.3 Feasibility of Microcontroller Elimination via Idle-Time Reuse

A core requirement of ART is that the RVS can execute module-level control tasks during the non-sequencing window without compromising deterministic FMCW sequencing. Based on the duty cycle discussion in Section 4.1, many radar configurations provide non-sequencing windows on the order of tens of milliseconds. As discussed in Section 4.1, representative duty cycles yield non-sequencing windows of 25–40 ms. ART leverages this slack by enforcing an explicit two-mode execution model: timing-critical ramp control runs exclusively in sequencing mode, while non-deterministic activities such as communication, configuration, and diagnostics are performed exclusively in control mode.

In the VP, this time partitioning is sufficient to (i) execute representative control routines between measurement bursts and (ii) prevent interference with the sequencer timeline by construction: once sequencing mode is entered, no control-mode communication tasks are scheduled until the ramp scenario completes. Under the considered VP configuration and satellite radar workloads, the RVS can absorb the module-level control tasks of the external microcontroller while preserving deterministic ramp execution.

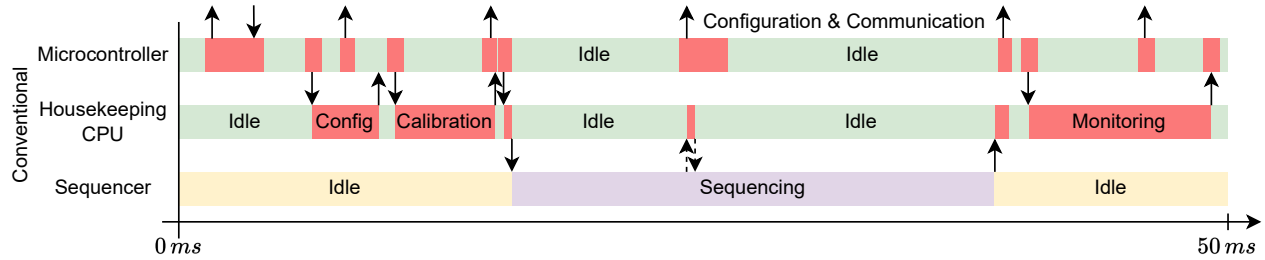
To quantify the overhead introduced by time partitioning, we measured the mode-transition cost in the cycle-accurate VP during the execution of a typical test scenario. Entering sequencing mode from control mode requires 19 clock cycles; returning to control mode after the ramp scenario completes requires 14 clock cycles. As summarized in Table 1, by assuming a representative sequencer clock frequency of $f = 200$ MHz the total round-trip transition overhead is 165 ns, which constitutes less than 0.001 % of a 25 ms non-sequencing window. During this transition time, the sequencing function is called and an internal state machine is entered/exited. This confirms that the mode-switching mechanism introduces negligible overhead relative to the available control-mode execution time. In our tests, the RVS additionally controlled the housekeeping CPU and conducted configuration, calibration and monitoring tasks comparable to those of the baseline transceiver MMIC from which the VP was derived.

Table 1: Total mode-switching overhead.

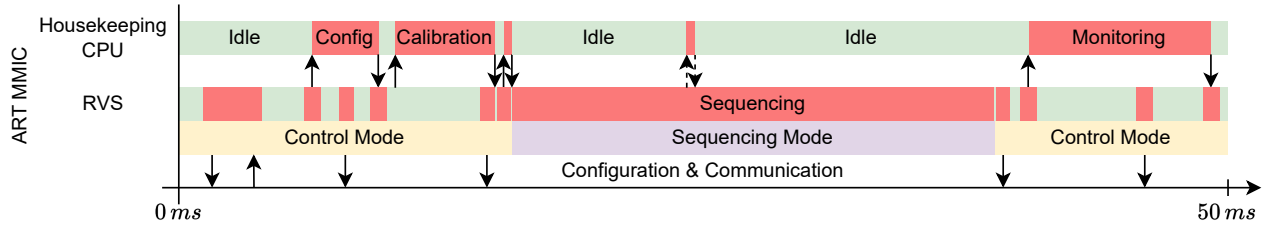
Operation	Cycles	Time @ 200 MHz
Enter sequencing mode	19	95 ns
Exit sequencing mode	14	70 ns
Total switching overhead	33	165 ns

5.4 Simplified Programming Model

ART changes the programming model exposed to the system integrator. In a conventional satellite radar module, two programmable masters must be managed: (1) the external microcontroller firmware and (2) the radar-internal sequencer program. These software domains must be synchronized because the microcontroller configures and triggers the sequencer and because microcontroller activity may overlap with measurement time. With ART, the system integrator programs only the RVS. Both control execution and ramp



(a) Conventional radar module application cycle with an external microcontroller. The microcontroller triggers the housekeeping CPU (e.g., calibration and monitoring) and performs configuration and communication tasks. The sequencer is active only during sequencing (measurement); depending on the configuration, the housekeeping CPU may also execute minor service routines during sequencing.



(b) Radar module application cycle with ART. The RVS executes module-level control tasks in control mode during the non-sequencing window and switches explicitly into sequencing mode to execute the ramp scenario deterministically. Communication tasks are confined to control mode; sequencing mode contains only timing-critical ramp control (and, if required, minor housekeeping CPU service routines).

Figure 2: Typical radar application cycles for two modules: a conventional module with an external microcontroller (top) and a module employing ART (bottom).

sequencing are implemented on the same RISC-V core, with an explicit mode switch defining when deterministic sequencing runs. This eliminates external synchronization between microcontroller firmware and sequencer code and provides a unified control flow for (i) selecting scenarios, (ii) preparing measurement parameters, and (iii) executing the ramp scenario.

We assess this reduction in integration complexity qualitatively based on the number of software artifacts and integration interfaces that a system integrator must maintain. In the conventional design, integrators manage two independent firmware images (microcontroller firmware and sequencer program), two build and test pipelines, and two update and versioning processes, in addition to a synchronization protocol between the two programmable masters. With ART, these responsibilities consolidate into a single RVS firmware image that contains both control-mode code and sequencing functions, and the synchronization protocol reduces to an explicit, intra-firmware mode switch. Consequently, the number of module-internal software interfaces and cross-component timing dependencies that must be validated during integration and regression testing is reduced.

5.5 Module-Level System Cost and Integration Impact

The main module-level outcome of ART is the elimination of the dedicated microcontroller from each satellite radar module. A conventional module typically consists of a transceiver MMIC, antennas, a high-speed serializer, and a small microcontroller that configures the radar chip and the serializer and handles low-rate communication with the ECU. In ART, these responsibilities are migrated to the RVS during non-sequencing windows, reducing the module to ART MMIC + antennas + serializer + power supply. While exact BOM values are design- and supplier-specific and are therefore not disclosed, the cost and integration benefit follows directly from removing one active component and its required supporting infrastructure. In particular, eliminating the per-module microcontroller removes (i) the microcontroller itself, (ii) parts of the module-level programming and update infrastructure, and (iii) a module-internal control plane that system integrators would otherwise need to implement, integrate, and verify alongside deterministic radar sequencing. Removing one microcontroller per module reduces component count and associated PCB area. This effect accumulates across vehicles integrating multiple radar modules (e.g., front plus corner radars).

5.6 Limitations

The main limitation of ART is that the RVS CPU cannot execute arbitrary module-level control workloads while sequencing is active. Tasks that require continuous compute or strict real-time response during the ramp scenario must either be scheduled into the non-sequencing window, delegated to the housekeeping CPU, or handled by the central ECU. Consequently, ART is best suited for satellite radar configurations where module-side computation is intentionally minimal and the control workload fits into the non-sequencing window between measurement bursts. In addition, for extreme duty cycle configurations in which the radar operates close to continuous transmission (i.e., with very short or absent non-sequencing windows), the time-multiplexing approach of ART may not provide sufficient slack to absorb module-level control workloads. In such cases, an external microcontroller or a dedicated on-chip control core remains necessary.

6 Conclusions

This paper presented ART, an autonomous transceiver-centric radar MMIC architecture optimized for satellite radar systems. ART leverages the RISC-V CPU of an on-chip RVS to execute module-level control tasks during non-sequencing windows while preserving deterministic FMCW sequencing during measurement. By consolidating control execution and deterministic sequencing on the same programmable core, ART eliminates the need for a per-module microcontroller and simplifies the programming model by avoiding explicit synchronization between external control firmware and sequencer code.

Future work will (i) characterize worst-case non-sequencing windows and control workload execution time for scenarios requiring higher radar duty cycles, (ii) quantify the silicon and power overhead of the added connectivity and communication mechanisms relative to a state-of-the-art transceiver MMIC, and (iii) evaluate firmware update, safety partitioning, and failure handling when module-level control is consolidated onto the RVS. To support this effort, automated waveform-analysis techniques [16] and VP-based testing methodologies [13, 20] could be employed to systematically check mode-switching behavior and timing properties across broader sets of radar workloads.

Acknowledgments

This work has been partially supported by European Union's Chips Joint Undertaking (ChipsJU) under grant agreement No. 101139892 (EdgeAI-Trust), the Austrian Research Promotion Agency (FFG) under grant No. 909770, and by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria.

References

[1] 2023. IEEE Standard for Standard SystemC® Language Reference Manual. *IEEE Std 1666-2023 (Revision of IEEE Std 1666-2011)* (2023), 1–618. doi:10.1109/IEEESTD.2023.10246125

[2] M. Alhumaidi and M. Wintermantel. 2021. Interference Avoidance and Mitigation in Automotive Radar. In *2020 17th European Radar Conference (EuRAD)*. 172–175. doi:10.1109/EuRAD48048.2021.00053

[3] Michael Atzmüller, Rainer Findenig, Bernhard Greslehner-Nimmervoll, Wolfgang Ecker, and Daniel Große. 2025. Leveraging RISC-V for Flexible and Adaptive Real-Time Radar Sequencing. In *IEEE Design and Verification Conference and Exhibition Europe*. 49–55.

[4] Aitor Correas-Serrano, María Gonzalez-Huici, Renato Simoni, Tobias Bredderman, Ernst Warsitz, Thomas Müller, and Oliver Kirsch. 2022. Performance Analysis and Design of a Distributed Radar Network for Automotive Application. In *2022 23rd International Radar Symposium (IRS)*. 30–35. doi:10.23919/IRS54158.2022.9904987

[5] Tom De Schutter. 2014. *Better Software. Faster!: Best Practices in Virtual Prototyping*. Synopsys Press.

[6] ETSI. 2024. Short Range Devices (SRD); Ground Based Synthetic Aperture Radar (GBSAR) in the frequency range 17,1 GHz to 17,3 GHz and High Definition Ground Based Synthetic Aperture Radar (HD-GBSAR) in the frequency range 76 GHz to 77 GHz; Harmonised Standard for access to radio spectrum. ETSI EN 303 661 V1.1.1. https://www.etsi.org/deliver/etsi_en/303600_303699/303661/01.01.01_60/en_303661v010101p.pdf

[7] Tai Fei. 2025. Automotive Radar Systems: Architecture, Signal Processing, and Future Perspectives. In *Vehicle Technology and Automotive Engineering*, Hasan Koten and Selçuk Sarıkoç (Eds.). IntechOpen, London, Chapter 2. doi:10.5772/intechopen.1008976

[8] Rainer Findenig and Bernhard Greslehner-Nimmervoll. 2024. Modular sequencer for radar applications. US Patent 12,000,952.

[9] Markus Gardill, Johannes Schwendner, and Jonas Fuchs. 2020. An Approach to Over-the-air Synchronization of Commercial Chirp-Sequence Automotive Radar Sensors. In *2020 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNeT)*. 46–49. doi:10.1109/WiSNeT46826.2020.9037593

[10] Michael Gottinger, Marcel Hoffmann, Mark Christmann, Martin Schütz, Fabian Kirsch, Peter Gulden, and Martin Vossiek. 2021. Coherent Automotive Radar Networks: The Next Generation of Radar-Based Imaging and Mapping. *IEEE Journal of Microwaves* 1, 1 (2021), 149–163. doi:10.1109/JMW.2020.3034475

[11] Daniel Große and Rolf Drechsler. 2010. *Quality-Driven SystemC Design*. Springer.

[12] Jürgen Hasch, Eray Topak, Raik Schnabel, Thomas Zwick, Robert Weigel, and Christian Waldschmidt. 2012. Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band. *IEEE Transactions on Microwave Theory and Techniques* 60, 3 (2012), 845–860. doi:10.1109/TMTT.2011.2178427

[13] Christoph Hazott, Florian Stögmüller, and Daniel Große. 2025. Using Virtual Prototypes and Metamorphic Testing to Verify the Hardware/Software-Stack of Embedded Graphics Libraries. *Integr.* 101 (2025).

[14] Vladimir Herdt, Daniel Große, and Rolf Drechsler. 2020. *Enhanced Virtual Prototyping: Featuring RISC-V Case Studies*. Springer.

[15] Mohinder Jankiraman. 2018. *FMCW radar design*. Artech House.

[16] Lucas Klemmer and Daniel Große. 2024. WAVING Goodbye to Manual Waveform Analysis in HDL Design with WAL. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 10 (2024), 3198–3211.

[17] Rainer Leupers, Grant Martin, Roman Plyaskin, Andreas Herkersdorf, Frank Schirrmeister, Tim Kogel, and Martin Vaupel. 2012. Virtual platforms: Breaking new grounds. In *Design, Automation and Test in Europe Conference*. 685–690.

[18] MIPI Alliance. 2025. MIPI Alliance Specification for Camera Serial Interface 2 (MIPI CSI-2). MIPI CSI-2, v4.2. Version 4.2.

[19] Mark A. Richards, James A. Scheer, and William A. Holm. 2010. *Principles of Modern Radar: Basic principles*. SciTech Publishing. arXiv:https://digital-library.theiet.org/doi/pdf/10.1049/SBRA021E doi:10.1049/SBRA021E

[20] Manfred Schlägl and Daniel Große. 2024. Single Instruction Isolation for RISC-V Vector Test Failures. In *IEEE/ACM International Conference on Computer-Aided Design*. 156:1–156:9.

[21] Frank Sickinger, Christian Sturm, Libor Janda, Ondrej Stejskal, and Martin Vossiek. 2018. Automotive Satellite Radar Sensor System based on an LTCC Miniature Frontend. In *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*. 1–4. doi:10.1109/ICMIM.2018.8443535

[22] Karthik Subburaj, Naveen Narayanan, Anil Mani, Karthik Ramasubramanian, Sujaata Ramalingam, Jasbir Nayyar, Krishnanshu Dandu, Karan Bhatia, Manshul Arora, Sai Jayanthi, Kamesh Vengattaramane, Shailesh Joshi, Arun Koityar, Kavithaa Rajagopalan, Dheeraj Shetty, Ben Thomas, Vashishth Dudhia, John Samuel, Rakesh Raavi, Shankar Ram, Abhishek Karkisaval, Pourush Sood, Sriraj Chellappan, Pankaj Gupta, Abhinav Daga, Bhavani Shankar, Indu Prathapan, and Brian Ginsburg. 2022. Single-Chip 77GHz FMCW Automotive Radar with Integrated Front-End and Digital Processing. In *2022 23rd International Radar Symposium (IRS)*. 141–146. doi:10.23919/IRS54158.2022.9905026

[23] Christian Waldschmidt, Juergen Hasch, and Wolfgang Menzel. 2021. Automotive Radar – From First Efforts to Future Systems. *IEEE Journal of Microwaves* 1, 1 (2021), 135–148. doi:10.1109/JMW.2020.3033616

[24] Andrew Waterman and Krste Asanović. 2019. *The RISC-V Instruction Set Manual; Volume I: Unprivileged ISA*. SiFive Inc. and CS Division, EECS Department, University of California, Berkeley.

[25] Andrew Waterman and Krste Asanović. 2019. *The RISC-V Instruction Set Manual; Volume II: Privileged Architecture*. SiFive Inc. and CS Division, EECS Department, University of California, Berkeley.