Report from Dagstuhl Seminar 23341

Functionally Safe Multi-Core Systems

Georg von der Brüggen*[1], Ian Gray*[2], and Catherine Nemitz†[3]

1 **TU Dortmund, DE.** `georg.von-der-brueggen@tu-dortmund.de`
2 **University of York, GB.** `ian.gray@york.ac.uk`
3 **Davidson College, US.** `canemitz@davidson.edu`

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23341 "Functionally Safe Multi-Core Systems". The seminar took place at a time when there is significant debate and disagreement in both academia and industry on how future safety-critical systems can be developed, certified, and deployed. This process is increasingly complex, as on the one hand, modern systems must provide more services while, on the other hand, analysing such systems becomes significantly more challenging in a multi-core scenario than it was in the single-core era. It is therefore a vital question to determine how the same level of certainty can be provided in the future.

The seminar brought together experts from academia and industry for the three major layers involved in safety-critical systems: application, middleware, and platform. They discussed the different perspectives, which problems are deemed specifically important, and potential solutions. One main focus when organizing the seminar was to not only present the different positions but also to provide space for lengthy discussion and disagreement.

Seminar August 20–25, 2023 – https://www.dagstuhl.de/23341
2012 ACM Subject Classification Computer systems organization → Architectures; Computer systems organization → Embedded and cyber-physical systems; Computer systems organization → Real-time systems; Computer systems organization → Dependable and fault-tolerant systems and networks; Theory of computation → Models of computation
Keywords and phrases eda and micro-architectures, middleware, multi-core, safety-critical applications
Digital Object Identifier 10.4230/DagRep.13.8.71

1 Executive Summary

Georg von der Brüggen
Ian Gray

There is a significant problem on the horizon in the field of embedded and real-time systems. The traditional approach for certifying high-integrity systems is no longer possible due to the increased complexity of modern applications, and the hardware platforms on which they execute are becoming heterogeneous multi-core systems on chip.

Considering how functional safety can be provided for multi-core systems, this Dagstuhl Seminar aimed to bring together experts from both academic and industry as well as participants from the three relevant layers, namely, application, middleware, and platform. The goal was to look at these topics from the individual perspective and to inspire interesting and fruitful discussion.

* Editor / Organizer
† Editorial Assistant / Collector

Motivation

High-integrity domains such as automotive and avionics represent an important success story for academia. There is a long and storied history of knowledge transfer from academia into these domains, taking theory, approaches, and tooling and using it to build and certify complex systems in which safety is paramount.

This seminar is organised at a time when the domain is in an unprecedented moment of flux. Application complexity due to increasing consumer demand has skyrocketed over the last decade, and so to meet these demands, hardware manufacturers have created increasingly complex hardware platforms. Where previously software would be used for basic control, engine management, and rudimentary driver assist systems, increasing amounts of automation and high-fidelity entertainment are being deployed. Current estimates suggest a high-end vehicle currently runs over one hundred million lines of code, and that number will only increase. A corresponding increase in complexity has also been observed in the most traditionally conservative domains, such as avionics and space.

The traditional approach to such systems would use simple processing devices upon which small sets of tasks could be allocated. Existing certification of real-time systems could work well in these domains, using CPU models and scheduling theory in order to determine the worst-case response times of all tasks running on each device. The use of timing-aware interconnects, such as the CAN bus, could allow system integrators to ensure that safety requirements could be discharged at the top level.

Unfortunately, the theory has not kept pace with reality. The next generation of safety-critical systems will undoubtedly use multicore processors because there is an increasing need for performance and the availability of single-core processors is reducing. It simply is no longer possible to deliver the kinds of applications that are being demanded by consumers on simple, single-core devices over which we have a high degree of certainty.

The currently available techniques for analysing the timing of such multicore devices are limited. Equally, certification authorities are only just beginning to catch up with this new reality meaning that the certification needs of systems using multi-core are not clear. Recently, the civil aviation industry has produced some guidance in the form of CAST-32A, however, this is raising more questions than it is answering.

Safety-critical systems need strong guarantees of their timing behaviors which includes evidence of when the timing requirements are met, and then evidence for the loss of availability of certain functions in the other cases. It is crucially important that both the timing requirement and loss of service are commensurate with the system safety function that comes from the application. The challenge in providing such evidence comes from the platform's shared resources, e.g., caches and buses. With the introduction of multicore, this has become more complex due to reduced predictability. The unpredictability can be managed through the middleware where the resource management exists, however with appropriate consideration across the three layers during their design and subsequent composition.

This Dagstuhl Seminar aimed to bring together practitioners from three disciplines which represent the three layers (application, middleware, and platform) relevant to safety-critical systems that use multicore to understand how the safety of a system using multicores may be argued; the achievable evidence that can be produced; and how said systems might then be developed.

Program and Structure

This seminar stands at the inflection point for high-integrity systems. There is significant debate and disagreement in both industry and academia about the manner in which future development should progress. It is unknown to what extent it is possible to maintain the same hard real-time guarantees that we have been used to in the past. Modern systems and modern devices simply do not provide the same level of certainty that we have relied upon. Simultaneously, systems are being asked to do even more things for which safety and certification are arguably more important than before.

This seminar was therefore structured to attempt to capture and advance this discussion. The main goal was to put academic leaders and industrial practitioners together in the same room, and to provide space for discussions and disagreements. The overarching goal was to agree as a room on what we believe the most important open questions were, and how research can be best structured over the coming decade to support the growing needs of the industry. Reciprocally, we hoped to capture what industry needs to provide academia, so that its needs can be met.

The first day of the seminar was arranged in advance, and then all subsequent sessions were arranged based on the discussions and the results that were obtained on the previous day. Sessions ran for 75 minutes with two sessions in the morning and two in the afternoon. These sessions usually started with a short talk or a controversial statement followed by extensive discussion. Therefore, we report a brief summary of the discussion in the sections instead of a summary of invited talks. When two sections discussed the same topic they are summarized together. One topic that was of specific interest to the participants from academia was the industry perspective; thus, multiple sessions had an industry focus.

Takeaways

Most takeaways from this seminar came from the open-ended structure and heavy focus on collaboration. A key outcome was the codification of a range of the open research challenges in this area and the way that we might as a community work towards them – these are detailed in Section 14. Other important points that were highlighted through the discussions were:

- Just because a problem is academically interesting to solve, doesn't mean that it is necessarily a key research question. Instead of focussing on specific application domains, it is important to remember the value in more generalised system models and approaches.
- A system designer doesn't specifically care about WCRT, they care about the argumentation chain that will help them discharge their safety requirements. This often involves priorities, deadlines, criticality, and timing analysis, but they are means to an end.
- Industry is very eager to use academic results, but certification requirements mean that they are often unable to do so without mature tool support. Finding a way to solve this problem will massively increase the impact of research.
- The entire certification process is under-served by both academia and industry. Academics have little visibility into the system, and industry is incentivised to remain insular in its approach to the problem. There is a possibility for opening this process up through collaboration and public funding.

- Hardware vendors are unclear on what is actually required by the application layer and the theory. Important future work will attempt to answer just how much predictability we actually need in any given circumstance. This is important because current high-end hardware displays fundamentally unpredictable timing from the perspective of the end user, and so exact timing models cannot exist.
- There is a need to develop more architectural description technologies that can provide more appropriate guarantees that can support timing analysis without being unnecessarily detailed or overly specific. This would involve the standardisation of performance counters which are currently ad hoc.

Feedback from the seminar shows that this intent was captured well. Participant feedback said that "this seminar has had one of the most successful industry-academic collaborations I've experienced in a Dagstuhl Seminar" and others praised the networking opportunities and the "engaging environment for our industrial participants" with "eye-opening panels and discussions". The main weakness of this approach was that the less formalised structure was noted by participants, indicating that a possibility for greater balance exists between discussion and networking opportunities and more formalised talks.

2 Table of Contents

3 Introductory Sessions

The purpose of the introductory sessions was to explain the way in which the seminar was being arranged and focus on the idea that the participants can decide what the outcomes should be, and the sessions that therefore should be scheduled in order to make that happen. The introduction was led by Ian Gray from the University of York. This session explained the background to the original Dagstuhl proposal, which is that critical systems based on multicore processes are an inevitability. Very few OEMs design and build their own processors, and there are very few single-core processors that can be purchased with a supply guarantee for over 20 years – the sort of guarantee that would be required in many industrial systems. Therefore, these new multicourse systems must be used, and must be made safe.

The seminar participants were chosen from three main areas:

- Industry
- Academic
- Junior researchers

Unlike many similar seminars in which the audience is academic with industrial representation, we aimed for a much more even split. Problems and difficulties in this area are encountered from both the theoretical and from the practical side, and so input from both areas was vital. Equally, we ensured that new researchers were also represented to capture a wide range of academic thought. The participants were selected across the Americas, Europe, and Asia, and we encouraged leadership and representatives who identify as genders other than male.

The introductory session contextualised some of the work that has already been done by the certification authorities toward functional safety in multicore architectures. Particularly CAST-32A, which became AMC 20-193, that codifies:

- Measurement-based approaches to timing analysis can only deliver approximate knowledge of a system's timing properties.
- Knowledge has uncertainties (known and unknown).
- There may be extreme behaviour that has not been seen.
- Static analysis approaches make an assumption that valid knowledge is available for all hardware and software, and that there are no anomalies.

This means any design and assurance may need to be stochastic in nature allowing for questionable confidence measures and significant uncertainties. Indeed, the balance between stochastic assurance and proof-based assurance remained an important topic of discussion throughout the entire seminar.

The introductory session used live audience polling technologies to determine the feeling of the room and to start to collect information that would be used to design upcoming sessions. The question was asked "To what extent do we think exact analysis for these systems is still possible?". As can be seen from the results in Figure 1, a wide range of views were held by the participants. This ranged from a complete rejection of the feasibility of exact analysis, through some level of stochastic analysis, to a few believers that exact analysis will always be mandatory.

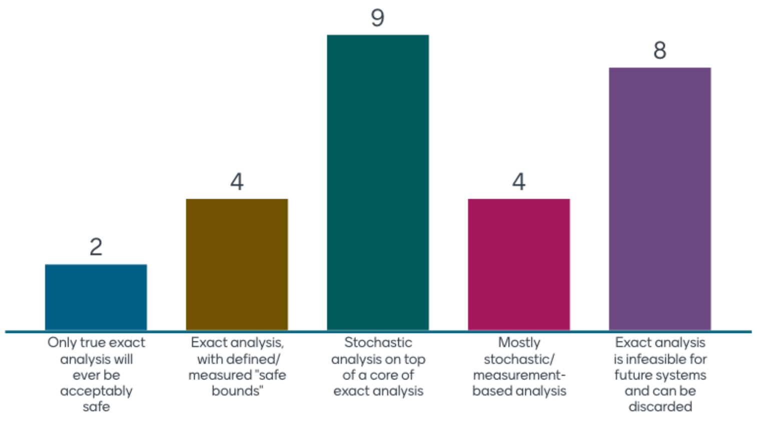

Figure 1 To what extent do we think exact analysis for these systems is still possible?

4 Application, Middleware, and Platform

The second section of the day attempted to break down the singular problem of functionally safe multicore systems into a range of levels at which the issues can be defined and solved. The organisers deemed that these levels were "application", "middleware", and "platform", and so, this section sought to define these layers and solicit beliefs about the biggest challenges at each level.

4.1 Platform

The platform layer encompasses all of the issues that come from the hardware upon which the system software is running. This layer asks a range of questions such as:

- How should the micro-architecture provide properties for timing abstractions and safety assurance?
- How can hardware components be configured to support functional safety?
- How should timing predictability be examined?

The audience was asked to come together online and in-person to determine the issues at this level. The following main points were identified:

- Features enabling performance often impact the timing behaviour. 'Fast' is often hard to analyse.
- Getting the appropriate level of information to perform analysis. What is the resolution that is necessary by the real-time community and how do you know you've identified the right model?
- The composability of analysis to allow larger systems to be analysed from smaller components.
- Machine learning and how it interacts with timing.
- Reliable detection of errors and how they should be accommodated.
- Partitioning of hardware, both spatially and temporally.
- Corner cases, unexpected events, unlikely events, and unknown unknowns – all of which are more likely as systems get more complex.
- How to create usable timing models from low-level architectural models, i.e. VHDL.

4.2 Middleware

The middleware layer encompasses the fact that systems are being developed with off-the-shelf middleware components alongside bespoke in-house development. This can lead to difficult interactions between code that the developer controls and code that they do not. Equally modern middleware has to manage the shared resources of multicore systems, and ensure that access to shared resources can be carefully timed and budgeted. Low-level shared resources and failures can break otherwise well-designed abstractions. This layer is defined by issues of failure propagation, isolation, resource management, robustness, and uncertainty.

The following main issues were identified by the participants:

- Strategies for segregation and isolation so that the limits of failure propagation are understood.
- Runtime monitoring and runtime management.
- How many different middlewares are needed? How can they be combined?
- Bridging the gap between theory and implementation.
- Security, and how it interacts with other issues of system safety.
- Resource management, composition, and run-time monitoring and control.

4.3 Application

The application needs to build on top of the previous layers, collecting all the uncertainties while perhaps adding its own. At this layer, a key problem is that abstractions tend to 'leak'. Lower levels of the system may claim a certain amount of performance or a certain amount of isolation, but it is quite common that at the application level these assumptions can be tested and broken. It is necessary for the application level to understand what it can, and cannot rely upon, and what may cause the system to behave in an unexpected way.

Participants deemed the following issues important at this level:

- To determine an acceptable contribution for timing in the mitigation of hazardous events.
- How to present application and platform co-design throughout certification/stages of involvement.
- Developers should write the least code possible. Can a model-based approach help here?
- How to formally express application requirements?
- Can a metalanguage be developed that captures and expresses the application requirements, and can be used to communicate (and configure) lower layers (middleware/platform)?
- Is it always possible to abstract hardware timing issues (e.g. interference) from the application point of view?
- How do you design systems that are resilient to faults, whether they be logical, temporal, or security-related, while still maintaining predictability?
- What tool support can be provided in order to ensure the developers can avoid common problems, can express application requirements in a consistent manner, and can implement theories and knowledge from lower levels?

4.4 Desired Outcomes

Finally, this section concluded by collecting together the outcomes that participants intended to create from this seminar. Lots of different possibilities were discussed, but chief among them was a desire to establish the key research challenges that could help to drive academic research for the next 10 years. Through this, a route map towards the effective achievement of assurance arguments for modern multi-core systems could be detailed, and therefore, the areas of theory that are not yet sufficiently developed become clear. In particular, the academic participants saw great value in hearing the views of practitioners about the real-world issues that they encounter and how they are beginning to tackle the problem of multicore. Accordingly, the organisers began arranging sessions around this with a view to culminating on the final day with a collection of research challenges.

5 Properties Needed and Functionalities Provided to Ensure Functional Safety and its Verification

After the previous session focused on the perspectives of applications, middleware, and platforms, the two final sessions on Monday started by collecting the thoughts of the participants on the properties needed and the functionalities that must be provided to allow verification.

Observability was determined as a key factor – otherwise, a system may be functionally safe but the safety cannot actually be argued and verified. However, it is not clear how observability can be achieved in practice, especially when abstractions need to cross layers. As a result, model composability is challenging, and if not appropriately scoped, impossible.

It was also mentioned that partitioning on the hardware level may be better than software partitioning due to potential overhead and granularity. However, while this may be an approach for static applications, it may not work well if applications have more dynamic needs. It was also pointed out that partitioning is not always possible – for instance, parallelization is often necessary but avoiding race conditions is non-trivial.

Research on multi-core systems will rely on results in other areas. However, these results may either not be known or assumptions and details may be unclear for people who are not experts in the field. For instance, it was (to the participants) unclear what the status of research into stochastic modeling of network traffic is. In this context, it has been pointed out that incorrect analyses had previously been used in practice, but that the overall over-approximations were still sufficiently pessimistic to prevent errors from occurring in practice. This was seen as an argument for a composable analysis compared to a holistic analysis.

An alternative approach could be to flip the problem – by imposing constraints on software development and preventing worst-case overlaps. For instance, some real-time operating systems do not impose constraints while others require applications to follow a given partition scheme.

This approach could play out in a way that industry is given a list of constraints to validate (without knowing the underlying analysis), which could result in systems that are easier to certify. In addition, tooling to show if meeting these constraints could be provided.

Furthermore, the following issues were selected as key by participants:

- Is there any hope of treating the path as deterministic (e.g., when looking at main memory access due to cache misses)?
- Literature of real-time analysis and WCET analysis is established – but some busy window assumptions do not compose and have been the source of challenges.

For the second session on the topic, the participants were split into 4 groups, each discussing and reporting their thoughts and ideas on what could be necessary to have functionally safe multi-cores and which other major problems still exist.

- Data sheets are sometimes inconsistent and incorrect which makes it hard to determine the "correct" behaviour and to determine where the uncertainties are. Indeed, the lack of accuracy from official documentation would return as an important issue many times over the course of the seminar.
- Related to this point, how do we verify correctness when we utilize functions, algorithms, and libraries? For instance, even seemingly simple things like the implementation of the EDF scheduler in some RTOSes were shown to be incorrect.
- Functional safety may mean different things on different levels. On a general level, a system may be considered safe if there is no harm to humans. Yet, on the system level, safety may mean, for instance, avoiding race conditions or fulfilling timing requirements. It must be made clear which definition or level is considered.
- Is functional safety the same as being certifiable?
- What is the difference between the properties we care about in safety-critical real-time systems and *functionally safe*? When can we say we have a sufficient set to ensure safety and how can we guarantee nothing is missing?
- Safety can be argued either by design or by verification. Is it easier to achieve when coming from one direction – and how can these approaches possibly be combined?
- There is a shift from static to dynamic analysis.
- Will hardware at some point reach sufficient performance with full predictability? Or, will this never be the case meaning that we should therefore focus on resilience and safe degradation?
- Is it always necessary to verify that all functional requirements are satisfied? Or, should we determine how much violation of properties can be tolerated or mitigated (e.g., by looking at fault tolerance approaches)?

6 System Certification

A topic that participants wanted to cover was that of system certification, the process by which high-integrity and safety-critical systems are certified for use by authorities. Academics are rarely involved in the complicated and time-consuming certification process. Yet, if academic results in the embedded system and real-time domain should be utilized in real-world products, they must usually be certified by the relevant authorities. Therefore, knowledge about the certification process itself, how these results can be transferred, and which assumptions, constraints, and techniques are seen as reasonable (or even at all achievable) by industry is highly relevant for academic research. In these two sessions, we discussed the certification process based, on the one hand, on the industry's perspective of Victor Jegu from Airbus and, on the other hand, on the perspective of a collaboration between academia and industry, given by Claire Pagetti from ONERA.

6.1 Industry's Perspective

Airplanes are very complex systems and also have strong certification requirements. As a result, they have a long cycle of development and certification; thus, moving from single-core to multi-core is also a lengthy process, which is just starting. The main driver for this move is not the need for more computation power – to ease certification this need could more comfortably be fulfilled by using multiple single-core CPU boards. However, airplanes must be maintained for decades and the availability of single-core CPUs is becoming an issue.

This lack of availability of other options while multi-core systems are highly available is the main driver for the transition to multi-core systems. One main focus of discussion was how the transfer from (interconnected) single-core systems to (interconnected) multi-core systems can be achieved. From Victor Jegu's perspective, key characteristics of the before-multi-core systems and ideal multi-core systems are as follows:

- Before multi-core:
  - one or more standalone cores with private resources
  - few COTS/IP components
  - ideally in-house hardware development
  - minimal shared resources, static WCET analysis possible
- Ideal multi-core:
  - distributed cores and resources as well as isolated interconnects – to minimize the shared usage to usage strictly needed for communication
  - cores may run in clusters for distributed applications (or when the workload is not real-time critical)
  - domains with large/intensive memory needs should also have dedicated memory controllers

To enable certification, systems should be completely deterministic in the data flow. Thus, tasks should be periodic, and scheduling decisions should be as static as possible. Yet, current multi-core systems are far from this, as there is minimal isolation of interconnects. As a result, there are multiple, often unpredictable, interference channels:

- sharing cache
- sharing buses and interconnects
- sharing targets (like SRAM and DRAM)

The problem of providing a certifiable system gets more complex since not all dependencies visible are in a processor's documentation. Furthermore, software constraints (like semaphores, locks, and rendezvous) have to be considered.

Solutions for these problems may involve:

- more formalism, e.g., by using languages like Rust for development
- lock-free architectures
- inter-core communication only through non-blocking interfaces
- static resource allocation by configuration tables

Restricting critical applications to one core is often not possible, since some resources are shared by the platform by force:

- memory controller
- interconnects
- secondary buses
- interrupt controller

6.2 Collaboration Perspective

Claire Pagetti is a contributor of the PHYLOG [1] project and its follow-up PHYLOG2, with both projects supporting aeronautic industry partners (e.g., Airbus). While PHYLOG focused on defining a certification strategy for multi-core architectures with respect to CAST 32A (now part of the AMC 20-193), PHYLOG2 considers the certification of hybrid architectures – which are multi-core architectures incorporating accelerators (e.g., GPU or FPGA).

The aeronautical certification process considers airworthiness regulations and provides means of compliance summarised in the AC (FAA Advisory Circulars) / AMC (EASA Applicable Means of Compliance) for applicants to define their certification strategy. The A(M)C themselves point to more detailed ARP/DO standards, each detailing objectives and activities to be fulfilled in order to demonstrate the compliance of a final product to regulations. An applicant can alternatively define their own path to compliance, called CRI (Certification Review Item), as long as they have an agreement with the certification authorities. Thus, certification can be seen as an argumentation to convince the certification authorities that compliance with regulations has been reached by validating and/or verifying that all requirements are met.

In particular, for designing an embedded multi-core-based system, which is the scope of PHYLOG, the applicable requirements are those specified by the AMC 20-152A, AMC 20-193, ED 79/ARP 4754, ARP 4761, ED 20x, ED 80/DO 254, and ED-12C/DO 178. Tools used during the design and implementation may have to be qualified as well, particularly if they generate embedded code. PHYLOG outcome is a model-based certification framework based on:

- Assurance cases to organize arguments through structured notations or diagrams. Most of the CAST 32A objectives as generic assurance cases have been translated to be instantiated for each specific system;
- A model-based approach as a replacement or complement to the usual text-based documentation. Therefore, a very generic modeling language, PML (PHYLOG meta-model), has been defined that allows representation of the minimal components that contribute to three types of analyses (interference, capacity, and safety) required by the CAST 32A.
- Formal and automatic analyses to support the certification argumentation. Amongst them, the identification of interference and the characterization of their temporal effect is of great importance. A Monosat + Scala tool that automatically computes the interference from a PML model has been deployed.
- A stressing benchmark strategy is associated with the modeling activity to validate the PML model and characterize the effect of interference.

The purpose of PHYLOG2 is to delve deeper into modeling hybrid architectures.

References

1 PHYLOG project webpage – `https://w3.onera.fr/phylog`

7 Industry Panel

The industry panel was organized as a Q & A session where the audience could ask questions which then were answered by Matteo Andreozzi from ARM, Jan Micha Borrmann from Bosch, Victor Jegu from Airbus, Kevin Quinn from General Dynamics, and Zoë Stephenson from Rapita Systems. The panel was moderated by Mitra Nasri from TU Eindhoven. We paraphrase the questions and summarize the provided answers.

- **Q1:** Do you look at academic papers, e.g., when you have a problem at hand?
  - Industry is more about patents than papers. Unfortunately, already published results can usually not be used in patents.
  - It would be easier if there was a shared wiki with people's research domains, considered models, etc.

- **Q2:** What academic assumptions are (un-)reasonable?
  - Academic papers often focus on meeting timing deadlines exactly, but safety-critical systems have to deal with faults. Hence, they are designed to be deterministic – when a deadline is missed, we know what will happen.
  - Shut down is usually not an option when a failure occurs – systems need to be fail-operative.
  - Mixed-criticality for us is about partitioning the electronics.
  - We don't care specifically about WCET, just that we meet safety requirements and end-to-end response time constraints. WCETs are just the best model we have.
  - Data freshness is important – one should annotate data with timestamps.
  - Focus is more on making correct decisions, not on periodic behavior of polling – it is less helpful to keep polling stale data.
  - For data freshness, the DAG model can be useful. Yet, industry cares about data flow graphs, academics are asking more about control flow graphs.
- **Q3:** What steps are taken to help increase timing predictability? (Question 19 from a recent survey [1] on industry practice in real-time systems. So, the panelists discussed the answers provided in the survey.)
  - Partitioning caches – should be expanded to more, just general "partitioning shared resources".
  - If disabling cache, maybe buy a more predictable machine instead.
  - A partitioning argument must be clearly communicable for certification.
  - A bandwidth servers type thing for highest assurance/integrity parts, can simplify RM analysis, one task every 20ms.
  - Regarding cache replacement policies, LRU is well explored in real-time literature and used by some platforms. Yet, ARM (and other) commercial platforms use a proprietary cache replacement mechanism – details on the mechanism cannot be given but some predictability guarantees can be provided.
  - Tools are used for WCET computation. It would be helpful to have an analysis showing the worst-case cache misses as this could reduce the testing needs.
  - One of the biggest needs is observability.
- **Q4:** Hardware performance counter standardization?
  - PMUs are not designed for dynamic monitoring.
  - Advanced OSes look at splitting up data gathering from monitoring.
- **Q5:** What do you see as a roadmap for the future, what questions should be answered, and what are future trends?
  - One very important thing when moving real-time support is observability.
  - Can we get to a useful workload modeling abstraction that enables sharing needed info without needing to disclose IP?
  - SHIM: software-hardware interface model.
  - Demos are beneficial as they can show the vision of others.
  - What could we do instead of WCET analysis? Is there another way to meet the safety requirements? Especially when considering the out-of-order execution of "modern" architecture. How does out-of-order execution impact jitter?
  - Partitioning can be a means to manage cost. Yet, it is not 100% reliable – could we get some bounds around that unreliability?
  - Software development takes a long time, at a huge cost and drivers can be as large as the primary code base.
  - Hardware reuse is expected, software needs a huge recertification effort.

References

1 Benny Akesson, Mitra Nasri, Geoffrey Nelissen, Sebastian Altmeyer, and Robert I. Davis, *A comprehensive survey of industry practice in real-time systems*, Real-Time Systems 58, 2021.

8 An "Ask" from Static Analysis

After focussing on the industrial perspective in the previous sessions, this session primarily considered potential analysis directions from academia. Rodolfo Pellizzoni from the University of Waterloo provided a general overview from the perspective of static analysis and Sanjoy Baruah from the Washington University in St. Louis moderated the discussion.

A multi-core resource interference analysis is often considering different parts of the system individually. Multiple cores (and other components) thus result in multiple analyses which a performed in a divide-and-conquer approach. For each core, a static analysis and a resource analysis are performed.

The conceptual resource model may be as follows:

- Takes some flow as input (target) and output (initiator) some flows.
- These are the requests (which behave differently for different types).
- Latency depends on the flow under analysis and the interfering ones.
- The process could be (conceptually) iterated to obtain end-to-end latency.

A basic, possibly provable, analysis could be structured as follows:

- Assuming that a model of the resource is available (which is not always the case).
- For each resource and each request, based on the type, an access pattern that causes the worst-case latency is determined.
- Sum the latency over all requests.
- For each program under analysis, sum over resources and add the total latency.

This approach could be safe (with caveats) but would be extremely pessimistic. At least the following sources of pessimism must be considered.

1. Request Distribution
   - There is often no information on request distribution. Yet, these distributions may potentially be obtained by program analysis or by observation. If pathological cases are triggered they may result in a 100x increased latency. Many hardware structures that are designed to not saturate under a "normal" operation flow stall once such a case happens.
   - A key issue is to capture the burstiness over time. Potential modeling approaches:
     a. Determining the number of requests over a given time window – for both the flow under analysis and for interfering ones. Problems with this approach include that it is imprecise, it is unclear how to cover bursts correctly, that large time windows must be considered, and how windows are synchronized.
     b. Request curves as in, for instance, network calculus. These curves separate burst and rate and can be deterministic or stochastic. However, determining arrival curves for network calculus is non-trivial and it must be guaranteed that the backlog is smaller than the buffer size to avoid "backpressure".
2. Latency Sensitivity
   - General idea: sum latency of all requests to the WCET.
   - Yet, not all requests necessarily add latency to the core (e.g., reads can be prefetched).

  - Out of Order cores can reorder operations and tolerate some load latency. This can potentially be accounted for by computing a tolerance term per request if alignment issues can be solved.
3. Missing Pipeline Effects
  - One interfering request may add delay on multiple resources.

Additional problems often arise since a precise resource model is either not available or too complex to be analyzed in practice. One potential solution, often used in academia, is to use a simpler, approximated model where the input-output characterization ensures that the output curve is always "safe". Yet, this property can likely only hold of system configured through partitioning (best in hardware).

Unfortunately, static analysis may not be the solution for the multi-core era, as finding a correct abstract interpretation of a modern superscalar out-of-order core may be (nearly) impossible. Instead, one may try other directions by examining the following questions.

- Is some kind of hybrid analysis possible?
- Could added observability help strengthen the case?
- Can we reach hard guarantees in soft systems?
- What dependencies do we need to look at?
- If we do not need worst-case, what do we need? Can we rely on statistical arguments?
- Could we look at a set of possible WCET values?

9 A Short Tour of Mostly Relevant Parts of DO-178C

As noted earlier in the report, undergoing the lengthy certification process is not something that academics generally take part in. Participants invited Zoö Stevenson from Rapita Systems to give an overview of her experience in supporting certification through DO-178C. It is the main certification document that describes the approval process for commercial aerospace systems. It is used by a range of countries including the US, Canada, and the EU and conformance to DO-178C is a key requirement for many tool vendors. Rapita Systemes provides both software tooling and analysis of user software, with the aim of certifying complex systems to this level.

A key point to highlight is that there is often a disconnection between software-level concerns and certification-level concerns. A feature of key importance to the software developer, may not necessarily be linked to system component of correspondingly high integrity. Software level does not imply failure level, so it is not possible to directly use software reliability as part of the certification argument.

A common misconception about certification documents, such as DO-178C, is that they give a list of system requirements which must be fulfilled and a checklist of tests to be performed. Rather, requirements come from system design, and so certification is concerned with the overall system and the evidence that is necessary to discharge safety requirements. For example, if the developer is using Ada, which contains a range of exception-handling systems, DO-178C will still require such systems are tested, and their efficacy demonstrated, if such systems are linked to the discharging of a safety requirement.

The applicant is responsible for oversight of all of its suppliers. This has wide-ranging implications for the way the software is developed integrated and analysed. The system must be analysed as a whole, with an analysis performed on the integrated software stack. All tool vendors and software vendors in this domain work to support this process.

The resulting certification argument is a combination of reviews, analyses, and tests. Significant training for employees is needed to support all stages of this process. A Software Quality Assurance process (SQA) will often form part of this evidence, through compliance to a relevant standard such as AS9100.

The final observation was on the semantics of certification documents. The previous version of 178C, 178B, was often used by vendors as prescriptive. If the document didn't mention something then it was presumed that you could not use it. This had wide-ranging implications for the range of development techniques, such as object-orientation, and of hardware platforms, such as FPGAs. More recent certification documents do not take this stance, but progress in these areas remains slow.

In the following Q&A, participants noted that it would be useful as a community to have an entire certification packet as an example. This is difficult because as has been noted, certification is a very lengthy and incredibly expensive process and necessarily reveals large amount of commercially sensitive information. Accordingly, it would have to be undertaken as part of a publicly funded project and while there are some burgeoning projects in this area, nothing has as of yet been completed.

10 Programming Languages and (Machine-Checked) Models: What to do About Multi-Core Platforms?

Timothy Bourke from INRIA & ENS in Paris discussed programming languages and (machine-checked) models with a focus on the additional challenges that arise when examining the timing behaviour in a multi-core environment. The process has a wide set of potential requirements and strategies. Hence, there is a tradeoff between opportunities and risks, for instance, for the exploitation of hardware, concurrency and related issues (like race conditions), or when considering inter-core interference sources.

The verification process results in a large argumentation tree, where the correct branch needs to be selected based on the component and the confidence. The interference analysis model can be built using PML. An appropriate documentation must be provided for the purpose of timing analysis. If one wants to do the same for schedulability, it must be determined what guarantees each test provides. In addition, to apply such an approach in multi-core systems, it might be necessary to utilize hardware designed for predictability or hardware that provides increased observability.

11 Combined HW/SW Formal Verification of Timing Behaviour

Mathieu Jan from CEA LIST talked about opportunities for combined hardware/software modeling for the formal verification of timing behaviour, using RISC-V as an example. He highlighted that it is important to identify timing anomalies – these do not necessarily need to be eliminated but they must be accounted for. This can be achieved by considering multiple traces. The 6-stage RISC-V rocket pipeline must be modeled (either manually or with an automatic tool). The TLA+ pipeline specifications are extended with instructions and control path.

12 How Much Predictable Computing Do We Need?

This session was lead by Jian-Jia Chen TU Dortmund University and aimed to answer the question, “how much predictable computing do we actually need?” A key theme that was emerging from the discussions was that it is possible for academics to solve problems that do not actually need to be solved. Fundamentally this comes from the fact that academics want to *solve* problems, and engineers want to *avoid* problems.

Consider the commonly used terminology to determine the real-time constraints of a given system component:

- Hard – the component should never violate timing.
- Firm – the component may violate timing, but this would never be seen at the application level, i.e. violations are handled.
- Soft – this component is allowed timing violations up to some stated level of assurance.

Academics tend to focus on 100% hard real-time systems, but real-world workloads are a mixture of different levels of assurance. This varies heavily from domain to domain; the needs of the automotive industry are very different from that of avionics or manufacturing, for example.

The application domains that should be considered most useful for focusing academic research were then discussed. There is currently a lot of interest in the field of autonomous driving and the interesting problems that it represents, but if pursued too much may result in too narrow a focus.

Instead of putting applications into ‘buckets’ and considering them in isolation, we should instead attempt to develop a range of system models that can be later matched to application classes. The reason for this is that application classes will change over time, and the speed of the industry means that this can sometimes be quite rapid. Cloud computing, big data, and autonomous driving are all examples of application classes that have appeared rapidly and driven a large amount of academic interest.

An example given was that of the Mixed Criticality System (MCS) model. Instead of focusing on a specific domain or application, MCS aims to be a way of building systems that can switch between different levels of assurance. It is a well-studied theory, and its generality means that it is starting to see use in a wide range of areas.

Our system models must recognise that industry is less focused on characterising workloads and more on the system-level requirements that they must fulfill. Timing correctness is an important tool for the verification of these requirements, but it is merely a *derived* requirement.

13 What Properties Do We Need From the Hardware/Middleware?

Chris Gill from Washington University in Saint Louis discussed the properties needed from hardware and middleware based on example projects he participated in. He emphasized that due to the large scope of embedded and cyber-physical systems applications have different needs and constraints; thus, application-specific semantics matter more than ever. For instance, sensor types, energy requirements, memory and cache sizes, the number of cores as well as the core speed, and the overhead to interconnect these cores may differ largely. As a result, general abstractions and generalizations are hard to find and it seems problematic to find a general approach for decomposing systems. Yet, it would be nice to have a list of constraints and construct adaptable scheduling to meet those.

Micha Borrmann from Bosch GmbH then discussed how these issues are addressed in real-world hardware development by enumerating some of the challenges in highly-available automotive computing. Automotive systems are moving from using a large network of low-power devices towards greater centralisation of computing resources in a few high-powered multicore platforms. This results in a large number of safety and security requirements that must be addressed. This necessitates a complete rethink in the way that automotive systems can be built, and is an active area of development.

14 Open Research Challenges

The seminar was concluded with an roundtable session in which the open research challenges as well as questions and thoughts were collected. The goal of this was to collect the thoughts of the participants as they had evolved over the course of the seminar, and to ask the questions would should drive the direction of research for the next decade. We collected top level observations, and the questions that stem from each of these.

- Maybe hard real-time is not the most accurate model.
  - Time is a means to some end – not the end goal.
  - Timing is also still an unsolved problem, and getting more complex
  - In real-time, when we specify tasks, we focus on time parameters. Should we also look at logical correctness? Are the correct values produced? Usually, we assume functional/logical correctness – this assumption may need to be challenged.
  - Is reliability a fundamental thing we must provide?
  - Trust and probability may be interesting metrics to consider.
  - Distinguish functional correctness and fault tolerance.
  - Non-functional properties are important: composability, extensibility, robustness, parametric simplicity.
- Applications are very complex, but platforms are amazingly complex.
  - We consider algorithms (scheduling algorithms, protocols), we need to make sure these are actually correctly implemented – separate concerns: analysis is correct, implementation is correct. For instance, PROSA vs. given implementation – does the implantation match the model?
  - When it comes to partitioning: how do we ensure two components do not interfere? This is a huge engineering effort, with hours of time per line of code.
  - Why should one use more complex architecture, requiring hugely complex analysis? One could consider building a simpler architecture.
  - Is the right way to look at uniprocessor techniques and determine how we can extend them to multicore? OR, should we do an entire redesign of analyses?
  - Safely-critical applications cannot simply just apply a patch, they need recertification.
- There is a value of consensus in the research community.
  - Dependability and fault-tolerance research communities study these and are working with some industries. Do we need to integrate? Can we rely on other work?
  - Do we analyze too complex systems? Engineers often repeatedly simplify problems.
  - It would be nice to know where we are losing. For example, why is a system using X unable to pass certification?
  - Instead of asking if a model is good, try to use it.

- What is the set of requirements?
  - Specifications and documentation will often have flaws.
  - Does the OS meet specs – industry finds this is not always the case. Is a verified OS a solution?
  - There are assumptions even in abstractions, simulations for physical hardware – we cannot fully capture physics in the model.
  - Models need to be as simple as possible for real use vs. expressive, complex models.

Many discussions centred around the system models that we create. It was observed that there is a clear tradeoff between accuracy and model simplicity. Formal models for safety certification ideally should satisfy or include:
- criticality
- environment assumptions
- developer-friendly tooling
- predictions about (worst-case) system behavior
- features acceptable to the certification authority

However, the process of modeling can have violations throughout this process:
- model compliance
- assumption validity
- analysis soundness
- implementation correctness
- analysis accuracy
- explainability

There clearly is therefore an identified need to keep people talking. This seminar showed the unique value in something which is critically lacking in our discipline, the ability for practitioners and academic experts to get together in the same room and to pass ideas to each other. Models need refining which can support certification, be reflected by hardware, be used by developers, and be analysed for correctness by tooling. Only with a cross-cutting integrated approach from all levels of the development stack can this succeed.

Participants

- Sebastian Altmeyer
Universität Augsburg, DE
- Tanya Amert
Carleton College – Northfield, US
- Matteo Andreozzi
Arm – Cambridge, GB
- Sanjoy Baruah
Washington University – St. Louis, US
- Jan Micha Borrmann
Robert Bosch GmbH – Stuttgart, DE
- Timothy Bourke
INRIA & ENS Paris, FR
- Björn B. Brandenburg
MPI-SWS – Kaiserslautern, DE
- Jian-Jia Chen
TU Dortmund, DE
- Christian Ferdinand
AbsInt – Saarbrücken, DE
- Julien Forget
University of Lille, FR
- Anna Friebe
Mälardalen University – Västeras, SE
- Chris Gill
Washington University – St. Louis, US
- Ian Gray
University of York, GB
- Arpan Gujarati
University of British Columbia – Vancouver, CA
- Robin Hapka
TU Braunschweig, DE
- Mathieu Jan
CEA LIST – Gif-sur-Yvette, FR
- Victor Jegu
Airbus S.A.S. – Toulouse, FR
- Eric Jenn
IRT Antoine de Saint Exupéry – Toulouse, FR
- Mitra Nasri
TU Eindhoven, NL
- Geoffrey Nelissen
TU Eindhoven, NL
- Catherine Nemitz
Davidson College, US
- Claire Pagetti
ONERA – Toulouse, FR
- Sri Parameswaran
University of Sydney, AU
- Rodolfo Pellizzoni
University of Waterloo, CA
- Kevin Quinn
General Dynamics – St Leonards on Sea, GB
- Jan Reineke
Universität des Saarlandes – Saarbrücken, DE
- Benjamin Rouxel
University of Modena, IT
- Selma Saidi
TU Dortmund, DE
- Matheus Schuh
Kalray – Montbonnot-Saint-Martin, FR
- Zoë Stephenson
Rapita Systems Ltd. – York, GB
- Jürgen Teich
Universität Erlangen-Nürnberg, DE
- Georg von der Brüggen
TU Dortmund, DE
- Bryan Ward
Vanderbilt University – Nashville, US
- Reinhard Wilhelm
Universität des Saarlandes – Saarbrücken, DE
- Houssam-Eddine Zahaf
University of Nantes, FR