BOOKLET OF PRESENTATIONS

http://www.ada-europe.org/conference2022

In cooperation with

acm In-Cooperation

SIGAda

SIGBED

SIGPLAN

Ada Resource Association ARA

GHENT UNIVERSITY

About this Booklet

This booklet contains short summaries of all the presentations included in the conference core program. The booklet groups presentations by session, prefixing their title with their type: '**IP**' for industrial presentations, '**RP**' for research presentations, '**WiP**' for lightning work-in-progress presentations (which are accompanied by posters exposed during the refreshment breaks).

The proceedings of the research papers will appear in a dedicated Special Issue of Elsevier's Journal on Systems Architecture.

The proceedings of the industrial papers will appear in forthcoming issues of Ada-Europe's Ada User Journal, along with papers drawn from the Work-in-Progress presentations.

We invite you to use this booklet as "navigational tool" throughout the conference program and its blank spaces provided in it, as a pad for your notes and commentaries. Enjoy the conference!

Core Conference Program

	Morning	Before Lunch	After Lunch	Afternoon
Wednesday, June 15th	**Session 1:** *Uses of Ada*	**Session 2:** *Real-Time Systems*	**Session 3:** *Development Challenges*	**Session 4:** *Advanced Systems*
				Spotlight Invited Talk
Thursday, June 16th	**Keynote Talk**	**Session 5:** *Special-Purpose Systems*	**Session 6:** *Verification Challenges*	**Session 7:** *Real-Time Systems*

Conference Sponsors

AdaCore

VECTOR

Join Ada-Europe!

Become a member of Ada-Europe and support Ada-related activities and the future evolution of the Ada programming language.
Membership is open to all, regardless of their residence.

http://www.ada-europe.org/**join**

Table of Contents

SESSION 1: USES OF ADA 4

WiP: Defining a Pattern Matching Language Feature for Ada 4

WiP: A Work Stealing Scheduler for Ada 2022, in Ada 4

IP: Ada for the Interchange of Data 5

IP: Ada on the Raspberry Pi RP2040 6

IP: HAC (the HAC Ada Compiler): from an Abandoned Teaching Project to a Usable Script-Like Ada Tool 7

IP: Renaissance-Ada: Tools for Analysis and Transformation of Ada Code 7

SESSION 2: REAL-TIME SYSTEMS (PART 1) 8

RP: Partition Window Assignment in Hierarchically Scheduled Time-Partitioned Distributed Real-Time Systems with Multipath Flows 8

RP: Response-Time Analysis of Mesh-Based Many-Core Systems 8

RP: Design Patterns Recognition for Applying Multiprocessor Real-Time Scheduling Analysis 8

SESSION 3: DEVELOPMENT CHALLENGES 10

IP: Agile Development for Critical On-Board Software 10

IP: Integration of modelling languages for the development of space domain software applications 11

RP: Model-Driven Development for the seL4 Microkernel Using the HAMR Framework 11

SESSION 4: ADVANCED SYSTEMS 13

WiP: Resilience-aware Mixed-criticality DAG Scheduling on Multi-cores for Autonomous Systems 13

WiP: Artificial Neural Networks for Real-Time Data Quality Assurance 13

IP: Increasing CPS Productivity and Resiliency through Accelerated Software Replication 14

SESSION 5: SPECIAL-PURPOSE SYSTEMS 15

WiP: Deep Learning for Communication Optimization on Autonomous Vehicles 15

WiP: Compiler Support for an AI-Oriented SIMD Extension of a Space Processor 15

WiP: Space Compression Algorithms Acceleration on Embedded Multi-Core and GPU Platforms 16

WiP: Fine-grained runtime monitoring of real-time embedded systems 16

IP: Software Tool for Evaluation of Multi-Sensor Object Tracking in ADAS systems 16

RP: Securing IIoT Communications using OPC UA PubSub and Secure Element 17

SESSION 6: VERIFICATION CHALLENGES 18

WiP: Boosting Simulation and Debugging of Cyber-Physical Systems with Symbolic Exploration 18

WiP: Improving Usability and Trust in Real-Time Verification of a Large-Scale Complex Safety-Critical System 18

WiP: Use of graph databases for static code analysis 19

IP: The Work of Proof in SPARK 19

RP: ATTEST: Automating the Review and Update of Assurance Case Arguments 20

WiP: Tracing and Measuring GPU Execution in Automotive Software Systems 20

SESSION 7: REAL-TIME SYSTEMS (PART 2) 22

RP: Near-Optimal Energy-Efficient Partial Duplication Mapping of Real-Time Parallel Applications 22

RP: Real-Time Fixed Priority Scheduling Synthesis Using Affine Data Flow Graphs: from Theory to Practice 22

RP: EDF Scheduling for Distributed Systems Built upon the IEEE 802.1AS Clock: A Theoretical-Practical Comparison 23

SESSION 1: USES OF ADA

WiP: Defining a Pattern Matching Language Feature for Ada

S. Baird, AdaCore, Mountain View, CA, USA
C. Dross, AdaCore, Paris, France
S.T. Taft, AdaCore, Lexington, MA, USA (*speaker*)

Abstract

Many programming languages, such as OCaml, Python, and Haskell, now include a pattern-matching feature, often introduced with the keyword *match*. These are not primarily focused on string pattern matching, but more on structure pattern matching, where the matching starts from an object of some structured type, and the individual patterns select particular structural patterns for special handling. These pattern matching features can be seen as a generalization of the case or switch statement available in most third-generation programming languages. However, they typically include the ability to associate an identifier with some or all of the pattern, which is then usable inside the handler for the given pattern, knowing that that identifier refers to some part of the original object that satisfies the given part of the pattern.
In general, for a pattern matching language feature three properties are considered important in any legal usage of the feature: complete/exhaustive -- every possibility is covered by some pattern; unambiguous -- there should be no two patterns where a given object could match both, unless the pattern that comes lexically second covers a strict superset of objects of the earlier pattern; nonredundant -- there are no patterns which are redundant, in that no object will reach that pattern, since it is fully covered by earlier patterns.
This presentation will describe the work in progress to define a generalization of the Ada case statement which would provide a general, structural pattern-matching facility, and describe some of the language design issues and some of the interesting implementation challenges.

WiP: A Work Stealing Scheduler for Ada 2022, in Ada

S.T. Taft, AdaCore, Lexington, MA, USA

Abstract

Ada 2022 parallel programming features rely on having multiple (lightweight) logical threads of control for a single Ada (heavyweight) task. This implies the need for a scheduler for such lightweight threads (LWTs). The OpenMP API includes a scheduler which can be used as the basis for this LWT scheduler, as OpenMP has the notion of both lightweight and heavier weight threads. An explicit goal of the design of Ada 2022 was to permit LWT schedulers such as the one provided by OpenMP to handle the scheduling for Ada 2022's "logical threads of control." It is also possible to build an LWT scheduler directly in Ada, using Ada tasks as heavier weight server threads to provide the actual execution resources for the scheduler.
Work stealing is widely recognized as an efficient approach to lightweight thread scheduling and has been adopted by many parallel-programming systems. Work stealing provides a nearly ideal combination of efficient load balancing across a set of cores (or kernel threads), while preserving locality of reference on a single core and separation of reference between distinct cores, which together minimize the unwanted cache effect known as false sharing.
The basic technique of work stealing is to have a separate, doubly-ended queue ("deque") of lightweight threads for each heavyweight server thread. In the context of Ada this means that we use multiple (anonymous) Ada tasks, each with its own deque, to service the various logical threads of control implicit in the use of the Ada 2022 parallel programming constructs of parallel loops and parallel blocks.
When a new light-weight thread (LWT) is spawned, the spawning server adds the LWT to the top of its own doubly-ended queue. When a server completes its current job, it pops the thread from the top of its deque. Hence,

the LWTs are managed as a LIFO stack by a given server. When a server runs out of LWTs on its own deque, it steals from some other server, and when it steals, it steals from the bottom of the stack of that server, so it generally gets an older and bigger job to do, operating in an essentially FIFO manner when stealing.
This talk will report on some of the specific challenges in achieving high performance, and the results achieved in the implementation of various Ada 2022 parallel programming features, in comparison with the use of the OpenMP scheduler.

IP: Ada for the Interchange of Data

M. Schlüter, Schlueter Consultancy, Kiel, Germany

Abstract

This presentation shows that Ada is not only suitable for programming in the narrower sense, but also for off-label-uses, i.e. other purposes where a well-defined formal language is needed. Thus, in many cases, the 'invention' of yet another dirty DSL (Domain Specific Language) can be avoided.
Back in the 90s, the author was responsible for a media company's bulk mailings—sometimes to millions of addressees.
The typical Data Flow was: the customer sent address data on magnetic tapes; these tapes were read and pre-processed on a mainframe; the data were transferred to work stations for further processing.
The occasional bottleneck was the excessive time it took to transfer the files from the mainframe to the work stations. Furthermore, its reliability was limited. Sometimes, without any warning, data got lost in the middle of the transfer.
Sorting was highly efficient on the mainframe. When the data were appropriately sorted—e.g. according to city, postcode, street, house number, and family name—the hamming distance between any two consecutive records was on average much smaller than the record length.
No data compression (like tar, ZIP, etc.) was available on the mainframe. No signature processing was available at that time. Data description and interchange languages like XML or JSON were not yet invented. So the task was to implement a data compression and basic safeguarding facility between the mainframe and the work stations.
The author decided to implement a high-level compression mechanism, making use of the small hamming distances. Only the changed attribute values of each record, compared to the attributes of the previous record, had to be transferred. The data interchange format had to fulfil the following requirements: safe (in contrast to e.g. CSV); simple, but extensible. In many cases, it was simple in the beginning. There was, however, often a subsequent need for distinctions of cases (if–then–else, case), loops, or other language constructs.
The author decided to use a simple DSL (Domain Specific Language) for this purpose. Instead of defining it from scratch, he used a tiny subset of Ada 83. This approach has the following advantages: there are no ambiguities or other nasty surprises; extensibility towards the whole extent of Ada is always guaranteed; for a proof of concept, test data can be compiled and effectively run on a standard APSE (Ada Programming Support Environment).
The Processing Steps then became: On the mainframe, a sort/merge utility sorts the original sequential file, e.g., according to city, postcode, street, etc. A primitive program written in NPL, a COBOL-style language, translates the flat file to be compressed into an Ada program. Only the attributes which have changed with respect to the previous record are transferred. A file transfer utility copies this Ada program to a work station. On the work station, the transferred Ada program is compiled and run; a primitive interpreter for the tiny subset of Ada, itself written in Ada, interprets the Ada program. In both cases, the original sequential file is re-generated.
Ada proved to be a perfect language for such purpose: defining a subset of Ada was much easier and safer than defining a new DSL from scratch. Ada compiler and runtime were already there and ready for compiling and running test data.

IP: Ada on the Raspberry Pi RP2040

J. Grosser, https://synack.me/

Abstract

The RP2040 is an innovative new ARM system-on-a-chip from Raspberry Pi. Throughout 2021, I wrote a HAL library for the RP2040 in Ada. This library is distributed as an Alire crate, taking advantage of that ecosystem's toolchain support and collection of libraries. I wanted to make Ada easier to use on an embedded platform, both for my own projects and for newcomers to the Ada language. The RP2040 is an exciting target for this effort due to its excellent documentation, low cost and innovative new peripherals. The large Raspberry Pi community has embraced this new platform, providing support for nearly any issue you might encounter and plentiful examples. As the RP2040 does not aim to support high reliability or heavily regulated applications, a rapid and iterative development process could be used, rather than a more traditional safety focused approach.

While it is certainly possible to program devices like the RP2040 by manipulating registers directly, this approach does not result in readable, maintainable, or portable code. The HAL aims to provide a high level interface where the developer's intent is clear. The HAL enables a developer to perform common tasks, such as toggling a pin or reading the ADC, without resorting to reading the chip's datasheet.

rp2040_hal achieves these goals, implementing portable abstract interfaces for GPIO, SPI, I2C, UART, and the RTC. A clear and well documented API is defined for the platform specific clock configuration, ADC, PWM, DMA, USB, and Timer peripherals. The RP2040's unique features- PIO, single cycle interpolator, internal frequency counter, and ROM floating point library are also supported. I will discuss how the HAL takes advantage of Ada language constructs to make these features easy to use.

The Alire package manager is used to manage toolchains and provides an extensive collection of libraries that can be used in programs on the RP2040. The HAL takes advantage of several libraries for complex functionality, such as the USB device stack. Alire makes it easy for newcomers to Ada to get up and running with a modern development environment on Windows and Linux.

The HAL uses a restricted subset of the Ada language, known as the Zero Footprint (ZFP) or Light runtime, minimizing the amount of required support code and avoiding many potentially unsafe operations.

A Ravenscar runtime has also been ported to the RP2040, taking advantage of the dual CPU cores to implement concurrent tasks. This implementation provides an excellent demonstration of Ada's protected types and tasking primitives on a low cost device.

Some of the challenges encountered during the development of rp2040_hal include booting from external QSPI flash, issues generating type definitions with svd2ada, the lack of atomic instructions, and integrating the ROM's floating point library with GNAT.

Testing is an important part of building a reusable library. The AUnit framework is used to organize and execute tests on the target hardware. GNATcoverage instruments the unit tests and provides statement and decision coverage information, which has been valuable in determining where to focus testing efforts.

Several demo applications were written during the development of the HAL, used to exercise functionality and inform the design of the HAL's API. I'll talk about interfacing with an I2S audio IC, various LED and LCD displays, and a closed loop PID controller.

rp2040_hal is a useful, open, and well documented library that will continue to be maintained and improved. I expect that Raspberry Pi will release new chips in the RP2XXX line and I hope to extend support to those as well.

IP: HAC (the HAC Ada Compiler): from an Abandoned Teaching Project to a Usable Script-Like Ada Tool

G. de Montmollin, Ada Switzerland

Abstract

In an ideal world you would not need to change programming languages when switching from small scripting jobs (let us call it the “light” side of programming), to large-scale application programming (the “heavy” side) which needs performance and reliability. Such a discrepancy would be fine if we knew in advance that small scripts would always remain small scripts. However, it occurs, more than often, that small experimental scripts become unexpectedly large over time and find their way in the production. Once this situation is reached, only bad options remain: sticking with a script language often needs to ignore performance and maintainability issues; changing languages is time-consuming and often cause a need to hire people just for translating and rewriting scripts, or perhaps interfacing and maintaining partial translations. So, ideally, a language would cover a broad spectrum from small jobs to large applications.

Fortunately, as we know, such a common universal language exists. However, is there an available Ada tool on the "light" side? With HAC (the HAC Ada Compiler), we are starting, finally, to have such a tool, thus contributing to break the language wall between the "light" and the "heavy" sides of programming.

The HAC adventure has begun with the discovery of an abandoned project, SmallAda, which was developed until around 1990 for teaching Ada concurrent programming constructs on low-end machines of that time. Under the surface, SmallAda consisted of a very fast, operational compiler which supported not only tasking constructs of Ada, but also a significant part of the Ada type system (records and arrays), unrestricted subprogram nesting, local declarations of any kind, initialized variables, etc.

After a very long and slow restoration phase (SmallAda's sources were split in huge blocks written in different Pascal dialects and littered with magic numbers and global variables with cryptic names), the HAC project, from now on structured in smaller dedicated Ada packages, has really taken off in 2020. The project is free and open-source, under the MIT license, and available for download, subversion checkout and git checkout.

We present the recent additions to HAC and its future evolution (planned or possible) as a compatible complement of "full Ada" toolset:

- A built-in library has emerged, with variable-size strings, string manipulation, text I/O, execution of external commands, etc.
- Incompatibilities (traces of SmallAda’s past as a Pascal compiler), such as type promotion and Pascal's expressions’ priority levels have been corrected. As a result, the same sources can be compiled on both HAC and a “full Ada” compiler like GNAT.
- Standard exceptions and trace-backs are available.
- Modularity is now available.
- A regression test suite of more than 50 test programs checks the correctness of the HAC compilation on various aspects (complex data structures, recursion, nesting, modularity, numerics ...).

IP: Renaissance-Ada: Tools for Analysis and Transformation of Ada Code

P. van de Laar (*speaker*), A. Mooij, ESI (TNO), Eindhoven, The Netherlands

Abstract

To keep code up to date, developers analyse and transform it. Ada developers can use Libadalang to automate their analysis. The Renaissance approach aims to enhance insight by analysis and to reduce complexity by transformation. The Renaissance approach is supported by programming-language-specific tools. The Renaissance-Ada tools are built on top of Libadalang and provide a more accessible interface to analyse and transform Ada code. Nexperia ITEC has used Renaissance-Ada to their full satisfaction, and recently Renaissance-Ada has become open source at https://github.com/TNO/Renaissance-Ada.

SESSION 2: REAL-TIME SYSTEMS (PART 1)

RP: Partition Window Assignment in Hierarchically Scheduled Time-Partitioned Distributed Real-Time Systems with Multipath Flows

A. Amurri (*speaker*), IKERLAN Research Centre - Basque Research & Technology Alliance
J. J. Gutierrez, M. Aldea, E. Azketa, Software Engineering and Real-Time Group, University of Cantabria

Abstract

Time and space partitioning techniques are implemented in the development of safety-critical applications to ensure isolation among components. A suitable arrangement of the execution of such partitions is a key challenge so that applications meet the timing requirements imposed to software. In this work, the effect of window sizes and context switch overheads in the partition window configuration is studied, with the aim of analysing their impact when the response-time analysis and priority assignment techniques are applied. Then, a heuristic algorithm is proposed, in order to obtain a partition window configuration that enables the schedulability of partition-based safety critical systems. This algorithm is evaluated in synthetic test scenarios, and it is also applied to a safety-critical use-case in the railway domain.

RP: Response-Time Analysis of Mesh-Based Many-Core Systems

D. Garcia Villaescusa (*remote speaker*): IKERLAN Research Centre
M. Aldea Rivas, M. Gonzalez Harbour: Software Engineering and Real-Time Group, University of Cantabria

Abstract

Scheduling models can be used to evaluate whether a particular system is able to meet its timing constrains. In many-core processors, with tens to hundreds of processors in the same chip, the analysis of the timing behaviour needs to include the communications network used to exchange messages between the different processors. This paper presents a schedulability model for many-core systems based on a 2D mesh network-on-chip and store-and-forward switching with a limitation on the maximum link utilization ratio that makes the analysis tractable. The model has been applied to the Epiphany many-core processor which has 16 cores connected by a 4x4 2D mesh. The analysis results have been tested on the real hardware by executing examples with synthetic task workloads. Those tasks are executed in a micro-kernel RTOS that we have developed. We also describe synchronization mechanisms to send messages between the tasks, and we analyse their timing behaviour, so that they can be included in the analysis model.

RP: Design Patterns Recognition for Applying Multiprocessor Real-Time Scheduling Analysis

S. Rubini (*speaker*), V.-A. Nicolas, F. Singhoff, A. Plantec, H.N. Tran, University of Brest, France
P. Dissaux, Ellidiss Technologies, France

Abstract

In order to ease the early verification of uniprocessor real-time systems, the tool Cheddar provides a service that guarantees the applicability of a schedulability analysis method for a given architecture model. This verification service uses a catalogue of design patterns. In this paper, we propose to extend these patterns to multicore architectures.
Designing such extension is a challenge because the knowledge of both the software and the hardware architectures are essential to decide on the schedulability of a task set in that context. Indeed, parallel execution of tasks involves hardware resource sharing, that has in turn an effect on the task execution times. Currently, no

general method is able to assess the schedulability of a multicore system with a limited level of pessimism, except if assumptions or usage restrictions are set to simplify the system analysis.

The research community is developing multiple schedulability tests based on various assumptions which constrain the task models and their execution platforms. In this paper, we propose a framework based on Prolog that allows engineers to control that the conditions for applying a test are met. Prolog facts model the software and hardware architecture, and the inference engine checks whether these facts are conforming to a design pattern associated to a given verification method. This framework is integrated with the Cheddar tool. Three examples of multicore analyses illustrate the proposal. A scalability analysis shows the tool is able to verify the compliance of architectures composed of 600 tasks and 60 cores, in less than 140 seconds on a desktop computer.

SESSION 3: DEVELOPMENT CHALLENGES

IP: Agile Development for Critical On-Board Software

E. Alaña, I. Bachiller, S. Urueña (*speaker*), GMV Aerospace and Defence (GMV), Tres Cantos, Spain
R. Lange, European Space Agency ESTEC, Noordwijk, The Netherlands

Abstract

A failure in a safety-critical system can have catastrophic consequences, like loss of life or human harm, significant property damage or to the environment. Typical examples of these systems are aircraft, medical devices, nuclear power plants, or communication satellites. A software is considered safety-critical when it can produce failures in a safety-critical system, like all the control and management software of the previous examples.

The development of every component used in a mission led by the European Space Agency (ESA) shall follow the ECSS standards (*European Cooperation for Space Standardization*), including the development of the on-board software (OBSW) controlling the orbiting satellites and other Space vehicles. The production of critical software relies on a thorough process that covers different areas: not only requirements, architecture, implementation, verification and validation but also management, product assurance, dependability, and safety. Critical software items, according to the ECSS standards, are those with a Software Criticality Category A, B or C as per the consequence of a failure in the function they perform: A for catastrophic consequences (e.g. loss of human live), B for critical consequences (e.g. loss of mission) and C for major consequences (e.g. loss of some scientific results).

Even if the ECSS standards do not mandate a specific software development life cycle, nowadays a waterfall or V-model is used in most Space projects. This life cycle is mostly sequential with a set of well-defined phases (analysis, design, construction, testing, and deployment): first the software provider elaborates all the software requirements (the so called *Technical Specification*, based on the system requirements allocated to software provided by the system integrator, which are the so called *Requirements Baseline*); then the Architectural and Detailed Software Design, followed by the coding of all the requirements together with the unit / integration testing (based on the software design); finally the software provider performs the validation testing (based on the *Technical Specification*) usually on a simulated environment, and then the system integrator performs the acceptance testing (based on the *Requirements Baseline*) on a hardware environment.

The waterfall model originated in the construction and manufacturing industries, as any change after the construction of these physical elements is prohibitively expensive. In contrast, software is not subject to the same limitations, and in fact changes to software are performed frequently after or during construction (also in the case of safety-critical software). So Agile development models were created to take advantage of this flexibility of the software, producing small working increments of the final product in short time periods (typically from 2 weeks to 2 months), usually called “Sprints”. In each Sprint a set of user requirements is analysed, implemented and validated, refactoring any existing code as needed. This development approach makes it easy to accept changes in the requirements, especially if the requirements have not been implemented in the past Sprints. Changes in requirements also affect safety-critical software. In the case of on-board software, as the system is developed in parallel by the system integrator, the software requirements are usually modified during the coding and testing phases to deal with unexpected changes in the hardware design or its final behaviour. As changes to the system requirements are in practice contractual changes, the system integrator typically must negotiate and procure a contract modification with the software provider to implement any modification to the requirements. And during the mission, the on-board software is also typically updated to solve any problem found or evolution needed, as normally the software is the only component of a Space vehicle that can be modified during flight.

The adoption of Agile development techniques may yield benefits to the production of critical software as well, as for example, providing early feedback from the system integrator or allowing changes in requirements at late development stages. While Agile software development has been adopted in several industries for a long time (even before the publication of the Agile manifesto in 2001) with Scrum being the most widely used Agile

framework (or meta-methodology), its usage for the development of critical software is controversial due to its perceived lack of formality. Even assuming a direct correlation between the quality of the processes used to develop software and the quality of the resulting software products, it does not mean that Agile frameworks cannot be used for a rigorous software development process. This paper focuses on using Agile techniques for developing safety-critical software in the Space domain, which is not new but still not widely adopted by the aerospace community or other industries. The "Agile for on-board software" (AGILE-OBSW) study has evaluated current frameworks and practices applied by different industries, identifying their applicability and impact on the production of critical on-board software for Space systems.

IP: Integration of modelling languages for the development of space domain software applications

Á. Pérez (*speaker*), M.Á. de Miguel, H. Valente, Information Processing and Telecommunications Center (IPTC-UPM), Universidad Politécnica de Madrid, Spain
A. Alonso, J.A. de la Puente, J. Zurera, J. Zamorano, Universidad Politécnica de Madrid, Spain

Abstract

Spacecraft systems comprise multidisciplinary teams which are in charge of the different subsystems such as AOCS (Attitude and Orbit Control System), TCS (Thermal Control System), and OBDH (On-Board Data Handling). These subsystems are mainly developed following an MBSE approach; implemented with different modelling tools and languages at different points of the product's life cycle. For instance, (i) aerospace engineers use MATLAB/Simulink to implement and validate AOCS; (ii) software engineers use CASE tools like TASTE or STOOD based on modelling languages such as AADL, UML, or HRT-HOOD.
Therefore, the integration of modelling tools offers many advantages: the productivity and software quality increase, the effort distribution throughout the system life cycle decreases, interaction between software components is reflected in the system architecture and analytical models for further analysis.
In this work, we improve the current integration of the TASTE tool-chain and QGen. TASTE is a modelling tool for the development of RTES which offers real-time abstractions such as cyclic, sporadic, or protected activities. QGen is a tool-suite that offers automatic code generation (Ada SPARK or MISRA C) and model verification for Simulink and StateFlow models.
This work is developed in the context of the AURORA project that aims to allow interoperability between different tools and improve the TASTE component model for the development of space software applications. UPMSat-2 is a microsatellite developed by aerospace engineers from IDR (Ignacio Da Riva Institute) and the STRAST research group conformed by software and telco engineers. We have used the ACS models and OBDH architecture models developed and used in this satellite (in orbit since September 2020) following a MBE approach.

RP: Model-Driven Development for the seL4 Microkernel Using the HAMR Framework

J. Belt, Robby, J. Hatcliff (*remote speaker*), Kansas State University, USA
J. Shackleton, J. Carciofini, T. Carpenter: Adventium Labs, USA
E. Mercer, Brigham Young University, USA
I. Amundson, J. Babar, D. Cofer, D. Hardin, K. Hoech, K. Slind, Collins Aerospace, USA
I. Kuz, K. Mcleod, Kry10 Limited, USA

Abstract

Verified microkernels such as seL4 provide trustworthy foundations for safety- and security-critical systems. However, their full potential remains unrealized due, in part, to lack of application development environments that help engineers integrate the microkernel's configuration and hosting of application code with modeling, analysis, and verification tools that address broader aspects of the development lifecycle.

This paper presents a model-driven tool chain for the seL4 microkernel based on the open-source High Assurance Modeling and Rapid engineering (HAMR) code generation framework for the Architecture and Analysis Definition Language (AADL). We describe how the semantics of AADL communication and threading can be realized in terms of the access primitives and strong spatial and temporal partitioning mechanisms provided by seL4. For AADL users, seL4 provides a high-assurance platform with formally verified enforcement of component boundaries and communication pathways. For seL4 users, AADL provides high-level abstractions for developing seL4 applications, along with an ecosystem of system engineering and analysis tools. We illustrate the framework by applying a model-based development environment for increasing resiliency against cyber-attacks to an unmanned aircraft flight control system.

SESSION 4: ADVANCED SYSTEMS

WiP: Resilience-aware Mixed-criticality DAG Scheduling on Multi-cores for Autonomous Systems

J. Zou (*speaker*), X. Dai, J. McDermid, University of York, UK

Abstract

Advanced driver-assistance and semi-autonomous systems represent the next major computing demand for road vehicles, which are complex and safety-critical with strict real-time and resource constraints and have a deep processing pipeline with strong dependencies between different functions. Further, tasks with different criticalities share the same hardware, and the scheduling strategy should guarantee high criticality tasks' execution irrespective of any interference from low criticality tasks whilst respecting the precedence constraints among tasks. Most static scheduling work considering task dependencies does not take into account the survivability of low criticality tasks, instead assuming that all low criticality tasks should be suspended or discarded after a mode change. Thus, the schedules for high and low modes are different. More efforts are needed to check the safety of schedules during mode change. There is a potential increase in the migration cost as tasks may be executed on a different core after a mode change.

Moreover, though slack management in high mode can allow the execution of low criticality tasks, it is still challenging for existing work to guarantee the effectiveness of slack utilisation and precedence constraints simultaneously. This work proposed a novel mixed-criticality DAG-based multi-core static scheduling method considering low critical tasks' survivability and precedence constraints between tasks with different criticalities. This produces a consistent schedule for different system modes enabling task-level mode change and improving the resilience of the system. Furthermore, the utilisation of computational resources is also improved by avoiding discarding low tasks.

WiP: Artificial Neural Networks for Real-Time Data Quality Assurance

I. Sousa (*speaker*), A. Casimiro, J. Cecílio, University of Lisbon, Portugal

Abstract

Wireless sensor networks used in aquatic environments for continuous monitoring are typically subject to physical or environmental factors that create anomalies in collected data. A possible approach to identify and correct these anomalies, is to use artificial neural networks, as done by the previously proposed ANNODE (Artificial Neural Network-based Outlier Detection) in order to improve the quality of data.

The ANNODE framework uses neural networks that are trained to detect outliers in a time series dataset. It is split into two parts, one for training neural networks and the other part is a real-time agent that processes data and detects possible anomalies, in this case, outliers.

This work proposes ANNODE+, which extends the ANNODE framework by detecting missing data in addition to outliers and supporting various time series datasets, as long as the target is to detect anomalies in those datasets. A new design and implementation were planned for ANNODE+, as well as an interface in which the framework's behaviour can be seen in real-time, enabling the user to see the detection of anomalies and its correction. In addition, a REST API was also implemented to fetch all stored measurements collected by sensors.

To evaluate the ANNODE+ capabilities, a dataset from a sensor deployment in Seixal's bay, Portugal, is used. This dataset includes measurements of water level, temperature and salinity. ANNODE+ also correlates different variables that might contribute to changes in water quality and is able to identify real events that look like anomalies, thus avoiding false positives. This work was developed in the scope of the AQUAMON project, whose objective is to develop a dependable platform based on wireless sensor networks to monitor aquatic environments.

The new ANNODE+ framework performs as intended, being able to detect anomalies and successfully correcting them. Every received and corrected measurement is stored in a database which is then accessed in order to graphically show the framework's behaviour through a specifically built interface.

IP: Increasing CPS Productivity and Resiliency through Accelerated Software Replication

A. Munera, E. Quiñones, S. Royuela (*speaker*), Barcelona Supercomputing Center, Spain
M. Pressler (*speaker*), A. Hamann, D. Ziegenbein, Robert Bosch, Renningen, Germany

Abstract

Model-driven engineering (MDE) is a software engineering paradigm that typically leans on domain specific modeling languages (DSML) to describe the system from the domain point of view while hiding the complexity of the technical solution. DSML further allow the automation of code generation, which facilitates the verification of the system, as many aspects can be verified at the model level. Consequently, MDE is widely used for the development of complex CPS where dependability is crucial. Advanced functionalities, like steering-by-wire and adaptive cruise control from the automotive domain, also require high-performance capabilities.
In this context, parallel heterogeneous architectures are of particular interest, as they can provide (1) dedicated processors to real-time and safety-critical functionalities, and (2) increased amount of resources to boost throughput. The AMPERE H2020 EU project (https://www.ampere-euproject.eu) addresses the development of correct-by-construction CPS by providing a new generation of software programming environments for low-energy and highly parallel and heterogeneous computing architectures that fulfil the non-functional requirements of the system, including real-time and reliability. Bosch, as industrial partner of the project and use-case provider, contributes with the APP4MC platform (https://www.eclipse.org/app4mc/), tailored for engineering embedded multi- and many-core software systems.
This work takes advantage of the possibilities offered by DSML and parallel processor architectures to boost the productivity of dependable systems that (partially) rely on replication to ensure safety. To that end, the APP4MC framework has been extended at two levels: (1) the AMALTHEA DSML has been augmented with a new redundancy property to increase fault tolerance; and (2) the APP4MC synthetic load generator (SLG) has been augmented with a transformation method that generates parallel replicas and uses a voting-based consensus algorithm to decide the solution and actions to take. The performance of the system is evaluated on the WATERS 2016 challenge (https://waters2016.inria.fr) proposed by Bosch, showing how the overhead of replication is drastically reduced, compared to single threaded execution, by using the idle resources of a parallel system without harming programmability or functional safety.

SESSION 5: SPECIAL-PURPOSE SYSTEMS

WiP: Deep Learning for Communication Optimization on Autonomous Vehicles

J. Loureiro, J. Cecílio (*speaker*), Faculdade de Ciências da Universidade de Lisboa, Lisbon, Portugal

Abstract

Recent breakthroughs in the autonomous vehicle industry have brought this technology closer to consumers. However, the cost of self-driving solutions still constitutes an entry barrier to many potential users due to its reliance on powerful onboard computers. As an alternative, autonomous driving algorithm processing may be offloaded to remote machines, which requires a reliable connection to the cloud servers. Despite significant 5G coverage in many countries, mobile network reliability and latency are still inadequate. This work explores deep learning concepts to forecast signal quality as a vehicle moves, predicting when periods of degraded network signal quality will occur, allowing the car to react and maintain a reliable mobile connection. We develop a Long Short-Term Memory (LSTM)-based neural network, trained on multivariate time series containing historical data on several mobile network parameters. This network allows for predicting multi-step Reference Signal Received Power (RSRP). The data, captured in Cork, Ireland, consists of over 95000 samples, including GPS coordinates, vehicle speed, network connection quality and noise ratio, throughput, and latency. Results show that our model achieves a rapidly increasing Root Mean-Square Error (RMSE), reaching over 8 dBm after 25-time steps, accompanied by vast confidence bands (over 20 dBm after 25-time steps at a 95% confidence level). This error does not allow for the accurate prediction of future signal quality. Therefore, the current model's use in commercial applications is unfeasible. We hypothesize that signal quality for previously unseen locations cannot be predicted from past observations with a satisfactory level of confidence due to the heterogeneity of driving environments. The idea that different approaches may be required to solve this problem is often found as a conclusion in related literature. A promising solution is to forecast signal quality using the distribution and density of buildings surrounding the vehicle's future locations. We will explore this research path in future work, using publicly available collections of building geometries.

WiP: Compiler Support for an AI-Oriented SIMD Extension of a Space Processor

M. Solé Bonet (*speaker*), L. Kosmidis, Barcelona Supercomputing Center (BSC) and Universitat Politècnica de Catalunya (UPC), Spain

Abstract

In this on-going research paper, we present our work on the compiler support for a custom, low-cost AI-oriented SIMD extension for space processors, called SPARROW. The design has been developed during a recently defended Master Thesis which has been awarded the best Master Thesis by the Spanish Chapter of the AESS IEEE and has also received the first prize in the Xilinx Open Hardware 2021 competition.
SPARROW is a SIMD module targeting Cobham Gaisler's space processors LEON3 and NOEL-V. We present the compiler support we have added for SPARROW in two major compiler toolchains, GCC and LLVM. We have extended the base SPARC v8 and RISC-V back-ends with support for the assembly of the new AI instructions. Compiler modifications are kept to the minimum in order to enable the incremental qualification of the toolchains. With the current state we can program SPARROW using C-inline assembly or a lightweight SIMD intrinsics library to further simplify the programming model.
Having worked with both compilers has allowed us to compare the development experience with each one. We also compare the performance of the generated code of various applications on the LEON3 with SPARROW implemented on an FPGA, achieving a speed-up of 15.2x with LLVM and 17.3x with GCC.

WiP: Space Compression Algorithms Acceleration on Embedded Multi-Core and GPU Platforms

A. Jover-Alvarez, Ivan Rodriguez (*speaker*), Leonidas Kosmidis, Barcelona Supercomputing Center (BSC) and Universitat Politecnica de Catalunya (UPC), Spain
D. Steenari, European Space Agency ESTEC, Noordwijk, The Netherlands

Abstract

The on-board processing requirements of future space missions are constantly increasing, requiring new hardware to satisfy this need. Embedded COTS platforms featuring multicore CPUs and GPUs are promising candidates, combining high-performance and low power consumption. Although previous on-board software case studies have demonstrated these properties, compression algorithms have been identified as quite challenging due to the inherent data dependencies which characterize their behaviour. In this on-going work paper, we describe the OpenMP CPU and CUDA and OpenCL GPU parallelization of two space compression algorithms defined by the Consultative Committee for Space Data Systems (CCSDS) standardization committee. In particular, we focus on the parallelization of the CCSDS 121.0-B-3 which covers general purpose data compression in a lossless way and CCSDS 122.0-B-2 which covers both lossless and lossy image compression.
Our results on two candidate embedded platforms for future on-board computers with multi-core CPUs and GPUs, the NVIDIA Xavier and the AMD Embedded Ryzen V1605B show that the parallelization of space compression algorithms for on-board processing is possible and comparable with existing space solutions. Our implementations are available as open source, as part of ESA's OBPMark (On-Board Processing Benchmark) benchmarking suite, focusing on the evaluation of general purpose devices for upcoming space missions, using complex representative applications relevant to the space domain.

WiP: Fine-grained runtime monitoring of real-time embedded systems

Z. Boukili, H.N. Tran (*speaker*), Alain Plantec, Univ. Brest, Lab-STICC, CNRS, UMR 6285, Brest, France

Abstract

Dynamically ensuring the correctness of the functional behaviour of a real-time embedded system is tedious especially in autonomous domain. Even though the current real-time task model provides sufficient information to perform basic schedulability tests, it is inadequate to be used in runtime monitoring to assert and guarantee the correctness of a system under hardware/software malfunctions or malicious cyber-attacks. In this article, we present a runtime monitoring approach based on a fine-grained model of real-time tasks.

IP: Software Tool for Evaluation of Multi-Sensor Object Tracking in ADAS systems

A. Medaglini (*speaker*), S. Bartolini, University of Siena, Italy
V. Di Massa, Thales Italy, Sesto Fiorentino, Italy
F. Dini, Magenta srl, Florence, Italy

Abstract

The automotive market in recent years is shifting towards Advanced Driving Assistance Systems (ADAS). In this domain, simulated tests can be an essential way for the automotive industry to provide the safety requirements of autonomous vehicles. One of the crucial parts in ADAS is the problem of object tracking and obstacle detection (OTOD), which is the focus of this paper.
Developing a multi-sensor ADAS requires evaluating the behaviour of association and tracking algorithms, which are fed with data from sensor fusion. In our work we tackled the problem of testing the behaviour of ADAS systems through a flexible tool for generating synthetic scenarios and evaluating KPIs. We propose a methodology that models the salient aspects of tracking and sensing objects, to effectively abstract the necessary facets to test OTOD behaviour in ADAS systems. In fact, each sensor collects different information from the environment, and we identified that the movement of an object can effectively be described by a limited set of basic motion patterns.

Using our generation language it is possible to specify the initial conditions of each object and modify them during its movement. Motion parameters can also be changed, even randomly, at each time instant within the object lifetime, so that even quite complex motion laws can be easily modelled. Furthermore, such scenarios can be combined with each other to generate more articulated situations, e.g. featuring multiple objects, and to evaluate the consequent interaction effects in the object-detection and tracking algorithms. Then, we allow modelling each sensor accounting for its output data format, transducer characteristics, as well as the noise that can affect it. In this way thousands of randomized variations of each specific scenario can be easily and effectively run, enabling also statistical evaluation approaches.
Therefore, our simulation framework tool generates synthetic scenarios that replicate the sensors' perception of the objects around a vehicle, aiming to be representative and effective without burdening the model with unnecessary details, promoting high modularity and flexibility. The tool aggregates the results from multiple runs and can automatically produce reports summarizing the achieved results through easily specifiable KPIs as well as the setup parameters, for experiment repeatability. Furthermore, thanks to the modularity and flexibility of our tool, it is easy to add and tune new KPIs to manage specific situations. Our methodology brings complementary features compared to existing ones for the evaluation of object detection and tracking systems in ADAS systems, as it allows modular, effective and repeatable, simulation of representative scenarios with limited effort.

RP: Securing IIoT Communications using OPC UA PubSub and Secure Element

O. Gilles (*speaker*), D. Gracia Pérez: Thales Research & Technology, France
P-A. Brameret, V. Lacroix, Systerel, France

Abstract

In the Industry 4.0 context, data are a valuable asset that must be protected. Ensuring the confidentiality and integrity of the data exchanged by the IIoT devices is challenging, especially when those devices are out of the companies' premises or easily accessible (we call them out-premise devices). These devices become primary targets for attackers as a way to compromise data and cause damage to infrastructure or people.
OPC UA PubSub provides the appropriate mechanisms to build scalable (e.g., one-to-many) secure and interoperable solutions with end-to-end encryption. However, authentication of IIoT devices remains a sensitive question, as it requires to securely embed secrets.
We present a novel approach based on open-source soft- ware aiming to secure out-premise devices authentication, enabling the confidentiality and integrity of data exchanges with the rest of the system. Our approach uses a Trusted Platform Module as a secure element to protect the secrets embedded on devices. We further apply the approach on a predictive maintenance use case and evaluate the security level of our solution in such use case.

SESSION 6: VERIFICATION CHALLENGES

WiP: Boosting Simulation and Debugging of Cyber-Physical Systems with Symbolic Exploration

I. Kolesnikov, IRIT, Toulouse, France

Abstract

Cyber-physical systems (CPS) often have a mission critical nature; it is therefore mandatory to ensure their correct functionality at runtime. Simulation and testing are quite common approaches, however, for the most of real-life CPS they fall short to explore all possible execution scenarios, due to the very large or even infinite size of their state space. This well-known state-explosion problem is mainly due to two factors, namely, (i) the number of components and (ii) the number and the domain (i.e., type) of data variables manipulated within the CPS. Several techniques allowing to cope effectively with state-space explosion have been developed in the past; in particular, symbolic exploration methods rely on specific encoding of system executions avoiding explicit enumeration of states, and therefore, provide opportunities to boost the performance of classical simulation and testing techniques.

One of these methods is Bounded Model Checking (BMC) which allows to effectively represent all system executions consisting of a specific number of consecutive steps and to verify related reachability properties.

The use of BMC on real systems has also some known limitations. For example, the number of steps to consider may need to be limited, the types and operations on data variables may need to be restricted in order to obtain formula manageable by existing solvers. Also, the counter-examples extracted when formulas are not satisfied can be counter-intuitive. So it might be difficult to explore a real scale system just using BMC. In this work, we propose the development of the theoretical approach and tools to reach two goals:

1. use BMC as a tool for local search of the state space, helping the user to reach states of interest during explicit state simulation, or to verify that certain conditions are unsatisfiable when starting from the current simulation state.
2. whenever a property is not satisfied, leverage the unsatisfiable core to provide user with some explanation of the counter-example to help him understanding the reason of failure and guiding towards finding a fix, if possible.

We plan to practically implement our approach on the top of TASTE a tool-chain targeting heterogeneous embedded systems, using a model-based development approach. We are currently working on supporting BMC for TASTE models, more precisely for encoding the semantics of TASTE components defined using the SDL graphical programming language as SMT constraints.

WiP: Improving Usability and Trust in Real-Time Verification of a Large-Scale Complex Safety-Critical System

B. Kempa, C. Johannsen, K.Y. Rozier (*speaker*), Iowa State University, Ames, Iowa, USA

Abstract

Large-scale complex safety-critical systems are inherently difficult to both verify in real-time and transparently validate. The iterative specification development process is challenging when the performance and reliability demands of target systems (e.g., flight software) require strict behaviour of verification tools which often trade-off usability for performance and conformance. Providing both strict behavioural guarantees and efficiency, this iterative process allows specification authors and engineers to more quickly deploy their systems and have more confidence in their verification efforts.

Our ongoing work addresses this challenge by providing validation transparency for specification authors during system development while maintaining necessary performance during deployment by extending R2U2, a real-time verification tool specifically designed for resource-constrained systems. We also strengthen the trust in R2U2

by providing a robust suite of tests to show adherence to the strict requirements of safety-critical flight software. These tasks are efforts toward transitioning R2U2 from a research-grade tool to a flight-software-grade tool suitable for real-time safety-critical systems and thereby answering the calls for expanded developmental-to-operational verification by, e.g., the Vehicle System Management (VSM) team of the NASA Lunar Gateway.

WiP: Use of graph databases for static code analysis

Q. Dauprat (*speaker*), Université de Caen Normandie & Adalog, Issy les Moulineaux, France
P. Dorbec, G. Richard, Université de Caen Normandie, Caen, France
J-P. Rosen, Adalog, Issy les Moulineaux, France

Abstract

This paper deals with static code analysis. In some activity sectors like industry (railway, avionics, space), programs become huge and complex. Since their emergence, static code analysis tools kept working on the same structure to analyse the source code, namely an Abstract Syntax Tree (AST). This type of structure shows its weaknesses with today's analysis needs. Furthermore, more and more needs for advanced analysis have emerged, and, consequently, the complexity of analysis induces a long analysis time. The scalability of static analysis tools become one of the current challenges.
Ada programming language is an interesting subject of study, since all its complexity has been delegated to the compiler. As a result, it is very difficult to compile, and the resulting AST is heavy and complex, making it complicated to understand and to query.
In this work-in-progress paper, we try to take advantage of recent technologies (graph databases) to represent the source code, with Code Property Graph (CPG), and pattern matching to find information into a graph. Our goal is to decrease the time of analysis of a source code and improve the effectiveness of the analysis. Lastly, Ada is a good starting point for our study, and we hope that will lead us to provide a general approach for static analysis of other programming languages.
We have created a proof of concept to validate our approach. When trying to answer the same query compared to AdaControl, we manage to find results that were missed by the programmatic approach. The future work will be to provide a benchmark on large code to validate the efficiency of our approach. We suppose that our approach can be beneficial for large volumes of the code, but irrelevant (slower) on small volumes compared to current approaches. We have to select a subset of static analysis rules (based on AdaControl) to perform our benchmark, in order to have enough use cases to have a relevant benchmark.

IP: The Work of Proof in SPARK

C. Dross (*remote speaker*), AdaCore, Paris, France

Abstract

Since Ada targets safety-critical programs, many features of the language introduce safety nets in the form of language-mandated checks. Even if compile-time verification is preferred to runtime verification whenever possible, many of these checks are still done dynamically, an exception being raised in case of violation. The addition of contracts in Ada 2012 follows a similar trend, as a violation causes an exception to be raised when the code is compiled with assertions enabled. The SPARK tool aims at statically verifying that language-mandated checks and the user-written contracts can never fail at runtime.
The SPARK tool verifies the program at the source code level on all possible inputs at once using deductive verification. The user is responsible for annotating their program with contracts that express the properties they want to verify on their code. The tool transforms the annotated program into a set of logical formulas, which can then be verified by automated solvers. If all the formulas are verified, the program is correct.
The verification is modular on a per subprogram basis. Contracts are used to summarize what the guarantees are for each subprogram. SPARK verifies that, for all inputs that fulfil the precondition, the subprogram executes safely (there are no runtime errors) and the postcondition holds on subprogram exit. When encountering a call, SPARK only looks at the contract: it checks the precondition and assumes the postcondition.

This means that, to verify a program using SPARK, it is necessary to annotate all subprograms with contracts precise enough to entail together the correction of the complete program.
SPARK is not a standalone tool. It works by using in the background bleeding edge technology developed by academic researchers in the formal verification domain. The Why3 platform, developed at Inria in France, performs deductive verification on a ML like semi-executable language called WhyMl. It generates the logical formulas, called verification conditions, and translates them into the input language of various automated or manual provers.
Taking a step back, verifying Ada programs using deductive verification is complex, and requires user input, in particular in the form of contracts. To make the tool usable in practice, it is important to choose the right simplifying assumptions. To preserve the soundness of the tool, these simplifying assumptions are generally linked to language restrictions which are enough for the assumption to be correct. In turn, these language restrictions are themselves checked by the SPARK too.
Both the SPARK proof tool and the related language restrictions are evolving continuously. Support for access types was added and extended in the last couple of years, precise support for exceptions is being discussed currently. These discussions occur publicly on ada-spark-rfcs, and the SPARK tool is available online: don't hesitate to join them!

RP: ATTEST: Automating the Review and Update of Assurance Case Arguments

F. Ul Muram (*speaker*): Department of Computer Science and Media Technology, Linnaeus University, Sweden
M. Atif Javed: RISE Research Institutes of Sweden, Sweden

Abstract

The assurance case arguments are created to demonstrate the acceptable system safety and/or security. In this context, a series of propositions expressed by natural language statements (claims) are broken down to sub-claims representing a logical chain of reasoning until the corresponding evidence is obtained. The review and update of assurance arguments for aligning with the process and product counterparts used for their construction are essential tasks. They are although perceived as challenging but can be efficiently supported by using Natural Language Processing (NLP). To date, however, the published studies on assurance cases have not leveraged the NLP. This paper presents our NLP-based assurance framework called ATTEST. At first, the text pre-processing is carried out by using NLP methods. The rules are created, in which both syntactic features captured by using NLP tasks and semantic features captured by internal structure of models and mappings across models are encoded. They are triggered for argument comprehension, well-formedness and sufficiency checks, identifying defeaters and counter-evidence selection. Besides the process, product and assurance case models produced during design and development phase, the operational data is gathered from the configured simulation environments and used for identifying problems as well as the measures for resolving them. Finally, the affected parts of assurance case models are highlighted and underlying reasoning for their adaptation is presented. The applicability of the proposed framework is demonstrated by reviewing and updating assurance cases constructed for vehicular Accelerator Control System (ACS) with Electronic Throttle Control (ETC).

WiP: Tracing and Measuring GPU Execution in Automotive Software Systems

T. Carvalho (*speaker*), L.M. Pinho, Instituto Superior de Engenharia do Porto, Portugal

Abstract

The advance of technology in the automotive industry brought several new functionalities providing more efficiency and safety. This, however, has one important concern: the development has become more complex. AMALTHEA is a framework for automotive system design and development in a model-based development fashion. It includes several features, including testing, software design, simulation, and traceability. AMALTHEA supports the traceability mainly focusing on timing properties. AMALTHEA adopted BTF (Best Tracing Format) as its standard format for providing traceability. Current versions of BTF are essentially focused on tracing CPU and operating system events, with timestamps as the only measurable feature.

With the evolution of computational systems and the increased demand of performance, it is common to include accelerators, and other performing devices, in the system, such as graphic processing units (GPUs) and field-programming gate arrays (FPGAs). These devices are expected to perform better - e.g. in terms of execution time - than a CPU, especially when dealing with data-intensive and concurrent applications.
This paper presents adequate extensions to the BTF format to include traces (and measurements) from devices other than the CPU, mainly accelerators, especially for the communication with AMALTHEA models. As the extraction of such traces is also a missing feature in the AMALTHEA framework, we also present a tool, a work in progress, that takes advantage of CUPTI API to profile CUDA-based applications and generate BTF traces with the new extensions. The final version of the tool will be integrated in the analysis flow of the PANORAMA and AMPERE projects. Despite the extensions being essentially aimed to GPUs, they were designed to be generic enough to comprise other types of accelerators or any other means for offloading computation outside the CPU.

SESSION 7: REAL-TIME SYSTEMS (PART 2)

RP: Near-Optimal Energy-Efficient Partial Duplication Mapping of Real-Time Parallel Applications

M. Cui (*speaker*), A. Kritikakou, L. Mo, E. Casseau, University of Rennes, INRIA, CNRS, IRISA, France

Abstract

Minimizing energy consumption, as well as meeting real-time and reliability constraints, are major goals during system deployment. When com- plex platforms, such as multicore architectures with DVFS, and parallel applications are considered, these goals are significantly impacted by task mapping. To minimize energy consumption, while meeting real-time and reliability constraints, this work proposes a task mapping approach to jointly solve the problem of task allocation, task scheduling, frequency assignment, and task duplication. A novel heuristic algorithm is proposed to cope with this NP-hard problem, consisting of a pruning phase, which maintains only the task configurations that satisfy reliability constraints, and a mapping phase, which minimizes total energy consumption under real-time and precedence constraints. The obtained results show that the proposed heuristic obtains near-optimal results, with low computation time, compared to optimal solvers, while it achieves better energy consumption and finds more solutions compared to other heuristic approaches.

RP: Real-Time Fixed Priority Scheduling Synthesis Using Affine Data Flow Graphs: from Theory to Practice

A. Honorat (*speaker*), T. Gautier, L. Besnard, J-P. Talpin, University of Rennes, INSA, CNRS, France
H.N. Tran: University of Brest, CNRS, France
S.S. Bhattacharyya, University of Rennes, INSA, CNRS & University of Maryland, USA

Abstract

The major drawback of using static schedules to execute dataflow applications is their high inflexibility. This article presents an approach to automatically generate fixed priority schedules from a dataflow specification. To do so, precedence dependencies between actors in the dataflow graphs are abstracted, as well as the task periods, by using affine relations. This model abstraction allows us to synthesize schedules efficiently for different objectives, such as the maximization of throughput and the minimization of buffer sizes. Given a dataflow graph to execute in a real-time environment, we transform it into an Affine DataFlow Graph (ADFG) and compute the task priorities, their mapping, the number of initial delays in the buffers, and the buffer sizes. This article concludes the 12-years work done on ADFG and presents corrections and improvements of the theory, focusing on the fixed priority case. Practical benchmark evaluations demonstrate the robustness and maturity of the approach that our scheduling synthesizer implements. Synthesized schedules are evaluated by using scheduling simulation and real-time implementation. Last but not least, the synthesized periods reach the optimal throughput offered by the number of available processors while the execution time of the synthesis is about one second only for the main proposed algorithms.

RP: EDF Scheduling for Distributed Systems Built upon the IEEE 802.1AS Clock: A Theoretical-Practical Comparison

H. Pérez (*speaker*), J. J. Gutiérrez, Software Engineering and Real-Time Group Universidad de Cantabria, Spain

Abstract

Existing response time analysis and optimization techniques for real-time distributed systems show that EDF schedulers presents better scheduling capabilities when a global clock reference can be used; this scheduling is known as global-clock EDF. In this context, precise clock synchronization is an enabling technology for future distributed real-time systems built upon EDF scheduling. The IEEE 802.1AS protocol can be considered a stable technology for this purpose, as it is part of the Time-Sensitive Networking (TSN) family of standards to provide real-time communication over Ethernet. This paper proposes a system architecture to apply global-clock EDF scheduling in distributed systems with soft real-time requirements. It also develops experiments to (1) evaluate the performance and assess the synchronization capabilities of the clock synchronization mechanism in the proposed architecture, (2) evaluate different scheduling deadline assignment techniques, and (3) contrast the theoretical results obtained by the schedulability analysis against those obtained through the execution of the application.

ORGANIZERS

Conference & Program Chair
Tullio Vardanega
University of Padova, Italy

Work-in-Progress-track Chair
Frank Singhoff
University of Brest, France

Publicity Chair
Dirk Craeynest
Ada-Belgium & KU Leuven, Belgium

Journal-track Chair
Jérôme Hugues
SEI, Carnegie Mellon University, USA

Tutorial & Workshop Chair
Aurora Agar
NATO, The Netherlands

Local Chair
Vicky Wandels
University of Ghent, Belgium

Industrial-track Chair
Alejandro R. Mosteo
Centro Universitario de la Defensa, Zaragoza, Spain

Sponsorship Chair
Ahlan Marriott
White Elephant GmbH, Switzerland

Web Master
Hai Nam Tran
University of Brest, France

JOURNAL-TRACK COMMITTEE

Mario Aldea Rivas (University of Cantabria, Spain), Johann Blieberger (Vienna University of Technology, Austria), Bernd Burgstaller (Yonsei University, South Korea), Daniela Cancila (CEA LIST, France), António Casimiro (University of Lisbon, Portugal), Xiaotian Dai (University of York, England), Juan A. de la Puente (Polytechnic University of Madrid, Spain), Barbara Gallina (Mälardalen University, Sweden), Marisol García Valls (Valencia Polytechnic University, Spain), J. Javier Gutiérrez (University of Cantabria, Spain), Jérôme Hugues (Software Engineering Institute, Carnegie Mellon University, USA), Patricia López Martínez (University of Cantabria, Spain), Kristoffer Nyborg Gregertsen (SINTEF Digital, Norway), Laurent Pautet Telecom (ParisTech, France), Claire Pagetti (ONERA, France), Luís Miguel Pinho (CISTER/ISEP, Portugal), José Ruiz (AdaCore, France), Sergio Sáez (Valencia Polytechnic University, Spain), Selma Saidi (Technical University Dortmund, Germany), Frank Singhoff (University of Brest, France), Tucker Taft (AdaCore, USA), Elena Troubitsyna (KTH Royal Institute of Technology, Sweden), Santiago Urueña (GMV, Spain), Tullio Vardanega (University of Padua, Italy)

INDUSTRIAL-TRACK COMMITTEE

Manuel Béjar (Universidad Pablo de Olavide, Spain), Dirk Craeynest (Ada-Belgium & KU Leuven, Belgium), Claire Dross (AdaCore, France), María-Teresa Lorente (Instituto Tecnológico de Aragón, Spain), Ahlan Marriott (White Elephant, Switzerland), Ana C. Murillo (Universidad de Zaragoza, Spain), Frédéric Praca (Ada-France), Sara Royuela (Barcelona Supercomputing Center, Spain)

WORK-IN-PROGRESS-TRACK COMMITTEE

Kalinka Braco (University of São Paulo, Brazil), Oana Hotescu (ISAE, France), Catherine Dezan (University of Brest, France), Patricia Balbastre Betoret (Valencia Polytechnic University, Spain), José Cecílio (LASIGE/University of Lisboa, Portugal), Héctor Pérez (University of Cantabria, Spain), Bjorn Andersson (Software Engineering Institute, Carnegie Mellon University, USA), Robert Kaiser (RheinMain University of Applied Sciences, Germany), Ali Balador (RISE Research Institute of Sweden, Sweden), Martin Schoeberl (Technical University of Denmark, Denmark), Nicolas Navet (University of Luxembourg, Luxembourg), Faiz Ul Muram (Linnaeus University, Sweden)