ADA USER JOURNAL

Volume 38
Number 2
June 2017

Contents

Editorial Policy for Ada User Journal

Publication

Ada User Journal — The Journal for the international Ada Community — is published by Ada-Europe. It appears four times a year, on the last days of March, June, September and December. Copy date is the last day of the month of publication.

Aims

Ada User Journal aims to inform readers of developments in the Ada programming language and its use, general Ada-related software engineering issues and Ada-related activities. The language of the journal is English.

Although the title of the Journal refers to the Ada language, related topics, such as reliable software technologies, are welcome. More information on the scope of the Journal is available on its website at *www.ada-europe.org/auj*.

The Journal publishes the following types of material:

- Refereed original articles on technical matters concerning Ada and related topics.
- Invited papers on Ada and the Ada standardization process.
- Proceedings of workshops and panels on topics relevant to the Journal.
- Reprints of articles published elsewhere that deserve a wider audience.
- News and miscellany of interest to the Ada community.
- Commentaries on matters relating to Ada and software engineering.
- Announcements and reports of conferences and workshops.
- Announcements regarding standards concerning Ada.
- Reviews of publications in the field of software engineering.

Further details on our approach to these are given below. More complete information is available in the website at *www.ada-europe.org/auj*.

Original Papers

Manuscripts should be submitted in accordance with the submission guidelines (below).

All original technical contributions are submitted to refereeing by at least two people. Names of referees will be kept confidential, but their comments will be relayed to the authors at the discretion of the Editor.

The first named author will receive a complimentary copy of the issue of the Journal in which their paper appears.

By submitting a manuscript, authors grant Ada-Europe an unlimited license to publish (and, if appropriate, republish) it, if and when the article is accepted for publication. We do not require that authors assign copyright to the Journal.

Unless the authors state explicitly otherwise, submission of an article is taken to imply that it represents original, unpublished work, not under consideration for publication elsewhere.

Proceedings and Special Issues

The *Ada User Journal* is open to consider the publication of proceedings of workshops or panels related to the Journal's aims and scope, as well as Special Issues on relevant topics.

Interested proponents are invited to contact the Editor-in-Chief.

News and Product Announcements

Ada User Journal is one of the ways in which people find out what is going on in the Ada community. Our readers need not surf the web or news groups to find out what is going on in the Ada world and in the neighbouring and/or competing communities. We will reprint or report on items that may be of interest to them.

Reprinted Articles

While original material is our first priority, we are willing to reprint (with the permission of the copyright holder) material previously submitted elsewhere if it is appropriate to give it a wider audience. This includes papers published in North America that are not easily available in Europe.

We have a reciprocal approach in granting permission for other publications to reprint papers originally published in *Ada User Journal.*

Commentaries

We publish commentaries on Ada and software engineering topics. These may represent the views either of individuals or of organisations. Such articles can be of any length – inclusion is at the discretion of the Editor.

Opinions expressed within the *Ada User Journal* do not necessarily represent the views of the Editor, Ada-Europe or its directors.

Announcements and Reports

We are happy to publicise and report on events that may be of interest to our readers.

Reviews

Inclusion of any review in the Journal is at the discretion of the Editor. A reviewer will be selected by the Editor to review any book or other publication sent to us. We are also prepared to print reviews submitted from elsewhere at the discretion of the Editor.

Submission Guidelines

All material for publication should be sent electronically. Authors are invited to contact the Editor-in-Chief by electronic mail to determine the best format for submission. The language of the journal is English.

Our refereeing process aims to be rapid. Currently, accepted papers submitted electronically are typically published 3-6 months after submission. Items of topical interest will normally appear in the next edition. There is no limitation on the length of papers, though a paper longer than 10,000 words would be regarded as exceptional.

Editorial

As usual, the June issue of the Journal is finalised shortly after the Ada-Europe conference, which this year took place in Vienna, Austria, in the week of June 12 to 16. The organizers must be congratulated for a very successful conference, with a very rich program, both technical and social. The program included not only a set of high-quality scientific papers and technical presentations, but also a panel on the future of safety-minded languages in general (and Ada in particular). The reader will find in this issue a summary of the panel contributions, by Erhard Ploedereder, panel moderator and Jorge Garrido, rapporteur. Also related to the future of Ada, this issue publishes the call for Community Input for the Maintenance and Revision of the Ada Programming Language, put forward by ISO/IEC JTC 1/SC 22/WG 9, the group of experts responsible for the maintenance and revision of Ada, requesting comments and contributions for the next revision of the language. I encourage you to read, and contribute!

As announced in Vienna, next year the Ada-Europe conference will take place mid-June in Lisbon, Portugal. You will forgive me in recommending (more than usual for obvious reasons) that you plan already to attend: not only for the conference program, which for sure will be worthwhile, but to have the opportunity to visit the “coolest” city in Europe (CNN) or Europe’s best work-and-play capital (BBC). Don’t miss it! I would also like to call your attention to the 19th International Real-Time Ada Workshop, which will return to the beautiful place of Benicàssim, Spain, in April. You can find the preliminary call for papers of these two important events in the Forthcoming Events section of this issue. Together with these you will also find an announcement of the Make with Ada programming competition, which promotes the development of embedded code using Ada and SPARK.

As for the other technical contents of the issue, the first article, by Bo Sandén, from the Colorado Technical University, USA, presents an approach to encapsulate Ada protected objects with a protocol monitor concept. Afterwards, a group of authors from the Ecole Nationale d’Ingénieurs de Sfax, Tunisia, describes an approach to address power management, in order to comply with energy budgets, in multicore embedded devices. Also as usual, the News and Calendar sections, produced by Jacob Sparre Andersen and Dirk Craeynest, the respective editors, complete the issue.

As a post scriptum, I would like to apologise for any delay the issue may have in arriving to your desk. The Ada User Journal is prepared by a team of highly-dedicated volunteers, and occasionally pressing matters delay concluding an issue. When that happens near to vacation periods, as in this issue, printing and distribution may be affected. We are sorry for that and will, as always, persevere in reducing these delays as much as possible.

Luís Miguel Pinho
Porto
June 2017
Email: AUJ_Editor@Ada-Europe.org

Quarterly News Digest

Jacob Sparre Andersen

Jacob Sparre Andersen Research & Innovation. Email: jacob@jacob-sparre.dk

Contents

Ada-related Events

[To give an idea about the many Ada-related events organised by local groups, some information is included here. If you are organising such an event feel free to inform us as soon as possible. If you attended one please consider writing a small report for the Ada User Journal. —sparre]

"Make with Ada" Programming Competition

From: Jamie Ayre <ayre@adacore.com>
Date: Mon, 15 May 2017 14:14:35 +0200
Subject: [AdaCore] Make With Ada competition launched
To: libre-news@lists.adacore.com

Dear GNAT community,

We are pleased to announce the 2nd annual Make with Ada competition launches today! The competition calls on embedded developers across the globe to build cool embedded applications using the Ada and SPARK programming languages and offers over $8000 in total prizes, as well as a 3D printer as the student-only prize.

The competition runs from May 15 to September 15, 2017 and you can register your embedded project idea from 15:00 CEST today at www.makewithada.org. You may participate in the competition as an individual or part of a small team and we invite you to check out our resources to get you started here http://makewithada.org/getting-started.

Don't miss out and start Making with Ada today!

[See also ""Make with Ada" Winners", AUJ 38-1, p. 4. —sparre]

Automotive - Safety & Security

From: Christoph Karl Walter Grein <christ-usch.grein@t-online.de>
Date: Sat, 20 May 2017 11:25:48 -0700
Subject: Call for Participation: Automotive - Safety & Security 2017
Newsgroups: comp.lang.ada

Automotive - Safety & Security 2017

May 30+31, 2017, Stuttgart, Germany

Venue: Robert Bosch Auditorium in Stuttgart/Feuerbach, Germany

http://www.automotive2017.de

Seventh in the series, the conference deals again in aspects of reliability, safety, security, privacy, etc. in automotive systems, many of which are heavily influenced by advances in applied Software Engineering. The presented papers discuss technologies and processes to improve safety and security of these systems.

Keynote speakers are Franco Gasperoni (AdaCore, New York), Christian Wieschebrink (BSI, Bonn), and Stefan Jähnichen (TU Berlin, Einstein Center Digital Future).

The programme of the conference, the associated committee meeting and the social events as well as registration and venue information is found on the website cited above.

The conference language is primarily German or English by choice.

Come join us for two focused days on making our cars and trucks more safe and secure!

Ada-Belgium Spring Event

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Mon, 5 Jun 2017 17:56:01 -0000
Subject: Ada-Belgium Spring 2017 Event, Sun 25 June 2017
Newsgroups: comp.lang.ada, fr.comp.lang.ada, be.comp.programming

----

Ada-Belgium Spring 2017 Event

Sunday, June 25, 2017, 12:00-19:00

Wavre area, south of Brussels, Belgium

including at 15:0

2017 Ada-Belgium General Assembly

and at 16:00

Ada Round-Table Discussion

<http://www.cs.kuleuven.be/~dirk/ada-belgium/events/local.html>

Announcement

The next Ada-Belgium event will take place on Sunday, June 25, 2017 in the Wavre area, south of Brussels.

For the 10th year in a row, Ada-Belgium decided to organize their "Spring Event", though strictly speaking it will already be summer then ;-), which starts at noon, runs until 7pm, and includes an informal lunch, the 24th General Assembly of the organization, and a round-table discussion on Ada-related topics the participants would like to bring up.

Schedule

- 12:00 welcome and getting started (please be there!)

- 12:15 informal lunch

- 15:00 Ada-Belgium General Assembly

- 16:00 Ada round-table + informal discussions

- 19:00 end

Participation

Everyone interested (members and non-members alike) is welcome at any or all parts of this event.

For practical reasons registration is required. If you would like to attend, please send an email before Wednesday, June 21, 21:00, to Dirk Craeynest <Dirk.Craeynest@cs.kuleuven.be> with the subject "Ada-Belgium Spring 2017 Event", so you can get precise directions to the place of the meeting. Even if you already responded to the preliminary announcement, please reconfirm your participation ASAP.

If you are interested to join Ada-Belgium, please register by filling out the 2017 membership application form[1] and by paying the appropriate fee before the General Assembly. After payment you will receive a receipt from our treasurer and you are considered a member of the organization for the year 2017 with all member benefits[2]. Early enrollment ensures you receive the full Ada-Belgium membership benefits (including the Ada-Europe indirect membership benefits package).

As mentioned at earlier occasions, we have a limited stock of documentation sets and Ada related CD-ROMs that were distributed at previous events, as well as back issues of the Ada User Journal[3]. These will be available on a first-come first-serve basis at the General Assembly for current and new members. (Please indicate in the above-mentioned registration e-mail that you're interested, so we can bring enough copies.)

[1] http://www.cs.kuleuven.be/~dirk/ada-belgium/forms/member-form17.html

[2] http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html

[3] http://www.ada-europe.org/auj/home/

Informal lunch

The organization will provide food and beverage to all Ada-Belgium members. Non-members who want to participate at the lunch are also welcome: they can choose to join the organization or pay the sum of 15 Euros per person to the Treasurer of the organization.

General Assembly

All Ada-Belgium members have a vote at the General Assembly, can add items to the agenda, and can be a candidate for a position on the Board[4]. See the separate official convocation[5] for all details.

[4] http://www.cs.kuleuven.be/~dirk/ada-belgium/board/

[5] http://www.cs.kuleuven.be/~dirk/ada-belgium/events/17/170625-abga-conv.html

Ada Round-Table Discussion

As in recent years, we plan to keep the technical part of the Spring event informal as well. We will have a round-table discussion on Ada-related topics the participants would like to bring up. We invite everyone to briefly mention how they are using Ada in their work or non-work environment, and/or what kind of Ada-related activities they would like to embark on. We hope this might spark some concrete ideas for new activities and collaborations.

Directions

To permit this more interactive and social format, the event takes place at private premises in the Wavre area, south of Brussels. As instructed above, please inform us by e-mail if you would like to attend, and we'll provide you precise directions to the place of the meeting. Obviously, the number of participants we can accommodate is not unlimited, so don't delay...

Looking forward to meet many of you!

Dirk Craeynest, President Ada-Belgium

Dirk.Craeynest@cs.kuleuven.be

Acknowledgements

We would like to thank our sponsors for their continued support of our activities: AdaCore, Atos, Esterline, and KU Leuven

(University of Leuven)

If you would also like to support Ada-Belgium, find out about the extra Ada-Belgium sponsorship benefits:

http://www.cs.kuleuven.be/~dirk/ada-belgium/member-benefit.html#sponsor

----

Ada-Europe 2017

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Tue, 6 Jun 2017 04:29:01 -0000
Subject: Press Release - Reliable Software Technologies, Ada-Europe 2017
Newsgroups: comp.lang.ada, fr.comp.lang.ada, comp.lang.misc

FINAL Call for Participation

*** UPDATED Program Summary ***

22nd International Conference on Reliable Software Technologies - Ada-Europe 2017

12-16 June 2017, Vienna, Austria

http://www.ada-europe.org/conference2017

** Check out tutorials and workshop! **

*** Full Program available on conference web site ***

*** Online proceedings available at event ***

*** Register now! ***

Press release:

22nd Ada-Europe Conference on Reliable Software Technologies

International experts meet in Vienna

Vienna, Austria (6 June 2017) - TU Vienna and Ada-Europe organize from 12 to 16 June 2017 the "22nd International Conference on Reliable Software Technologies - Ada-Europe 2017" in Vienna, Austria. The event is organized in cooperation with the Ada Resource Association (ARA), and with ACM's Special Interest Groups on Ada (SIGAda) and on Programming Languages (SIGPLAN).

The Ada-Europe series of conferences has over the years become a leading international forum for providers, practitioners and researchers in reliable software technologies. These events highlight the increased relevance of Ada in general, and in safety- and security-critical systems in particular, and provide a unique opportunity for interaction and collaboration between academics and industrial practitioners.

This year's conference offers two days of parallel tutorials, a workshop, three keynotes, a full technical program of refereed papers and industrial presentations, an industrial exhibition and vendor presentations, and a social program.

Eight excellent tutorials on Monday and Friday cover a broad range of topics: Introduction to SPARK 2014; Ada on ARM Cortex-M, a Zero-Run-Time Approach; Software Measurement for Dependable Software Systems; Real-Time Parallel Programming with the UpScale SDK; Using Gnoga for Desktop/Mobile GUI and Web development in Ada; Frama-C, a Collaborative Framework for C Code Verification; On beyond ASCII: Characters, Strings, and Ada 2012; Modular Open System Architecture for Critical Systems.

In addition, on Friday the conference hosts for the 4th consecutive year the International Workshop on "Challenges and new Approaches for Dependable and Cyber-Physical Systems Engineering" (De-CPS 2016), now with a focus on "Transportation of the Future".

Three eminent keynote speakers have been invited to open each day of the core conference program.

Giovanni Battista Gallus (Array, Italy), in "The laws of robotics and autonomous vehicles may be much more than three, but don't panic... yet", will talk about the future European legal framework, which is relevant for the development of autonomous vehicles, and especially programming issues.

Thomas Henzinger (IST, Austria), in "Behavioral Software Metrics", will show how the classical satisfaction relation between programs and requirements can be replaced by quantitative preference metrics that measure the "fit" between programs and requirements.

Kay Römer (TU Graz, Austria), in "Dependable Internet of Things", will introduce the Dependable Things research center at TU Graz and present recent results on improving the dependability of wireless communication and localization, embedded computing, and networked control for the Internet of Things.

The technical program presents 14 refereed and carefully selected papers on the latest research, new tools, applications and industrial practice and experience, a collection of 9 industrial presentations reflecting current practice and challenges, 4 presentations and a discussion in a special panel session on "The Future of Safety-Minded Languages", and vendor presentations. Springer Verlag publishes all peer-reviewed papers in the proceedings of the conference, as LNCS

Vol. 10300. The remainder of the proceedings will be published in the Ada User Journal, the quarterly magazine of Ada-Europe.

The industrial exhibition opens Tuesday morning and runs until the end of Thursday afternoon. Exhibitors include AdaCore, PTC Developer Tools, Rapita Systems, VectorCAST, and Ada-Europe.

The social program includes a Welcome Reception plus robotics presentations on Tuesday evening at "TU the Sky"; its terraces at the top of TU Vienna's buildings offer a terrific view on the city. On Wednesday evening there will be a Vienna bus tour, followed by the traditional Ada-Europe Conference Banquet, held at a very famous "Heuriger". Each day, coffee breaks in the exhibition area and sit-down lunches offer ample time for interaction and networking.

The Best Paper Award will be presented during the Conference Banquet, the Best Presentation Award during the Closing session.

The conference is hosted by TU Vienna at Palais Eschenbach, which is located near the center of Vienna and can easily be accessed by metro.

The full program is available on the conference web site. Online registration is still possible.

Latest updates:

The 16-page "Final Program" is available at http://www.ada-europe.org/conference2017/AE2017_final_program.pdf

Check out the 8 tutorials in the PDF program, or in the schedule at http://www.ada-europe.org/conference2017/tutorials.html.

Registration fees are very reasonable and the registration process is done on-line. Don't delay! For all details, select "Registration" at http://www.ada-europe.org/conference2017 or go directly to https://adaeurope.upv.es/index.html.

For those who can't attend the full conference, note that Wednesday 14 June is "Meet Ada-Europe day!", and single day registration is discounted.

The proceedings, published by Springer Verlag as Lecture Notes in Computer Science Vol. 10300, are already available online. See https://link.springer.com/book/10.1007/978-3-319-60588-3. A printed copy is included in every full conference registration.

Help promote the conference by advertising for it!
http://www.ada-europe.org/conference2017/promotion.html.
Put up the poster at
http://www.ada-europe.org/conference2017/picts/AE2017_poster.png

Recommended Twitter hashtags: #AdaEurope and/or #AdaEurope2017.

For the latest information consult the conference web site http://www.ada-europe.org/conference2017.

Ada-related Resources

RosettaCode

From: Alejandro R. Mosteo <alejandro@mosteo.com>
Date: Fri, 10 Feb 2017 17:51:56 +0100
Subject: Bernoulli numbers at RosettaCode
Newsgroups: comp.lang.ada

While reading on the first algorithm attributed to our favorite lady, I saw that there's no Ada implementation at RosettaCode. Just in case someone has some time in their hands to fix this omission.

https://rosettacode.org/wiki/Bernoulli_numbers

Funnily enough, the Pascal impl is taken from an Ada one:

https://marquisdegeek.com/code_ada99

Social Network Site

From: Tomek Walkuski <tomek.walkuski@gmail.com>
Date: Thu, 6 Apr 2017 01:01:35 -0700
Subject: [ANN] Ada community on Gitter
Newsgroups: comp.lang.ada

Hi, I've recently started Ada developers community on Gitter, feel free to join if you want!

https://gitter.im/ada-lang

Repositories of Open Source Software

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Mon Jun 26 2017
Subject: Repositories of Open Source software

GitHub:	907 repositories	[1]
	431 developers	[1]
	850 issues	[1]
Rosetta Code:	635 examples	[2]
	32 developers	[3]
	0 issues	[4]
Sourceforge:	258 repositories	[5]
BlackDuck OpenHUB:	210 projects	[6]
Bitbucket:	78 repositories	[7]
OpenDO Forge:	24 projects	[8]
	521 developers	[8]
Codelabs:	21 repositories	[9]
AdaForge:	8 repositories	[10]

[1] https://github.com/search?q=language%3AAda&type=Repositories

[2] http://rosettacode.org/wiki/Category:Ada

[3] http://rosettacode.org/wiki/Category:Ada_User

[4] http://rosettacode.org/wiki/Category:Ada_examples_needing_attention

[5] http://sourceforge.net/directory/language%3Aada/

[6] https://www.openhub.net/tags?names=ada

[7] https://bitbucket.org/repo/all?name=ada&language=ada

[8] https://forge.open-do.org/

[9] http://git.codelabs.ch/

[10] http://forge.ada-ru.org/adaforge

[See also “Repositories of Open Source Software”, AUJ 37-4, p. 184. —sparre]

Ada on Social Media

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue Jun 27 2017
Subject: Ada on Social Media

Ada groups on various social media:

- LinkedIn:	2_643 members	[1]
- Reddit:	998 readers	[2]
- StackOverflow:	811 followers	[3]
- Google+:	766 members	[4]
- Freenode	85 participants	[5]
- Gitter:	48 people	[6]
- Twitter:	10 tweeters	[7]

[1] https://www.linkedin.com/groups?gid=114211

[2] http://www.reddit.com/r/ada/

[3] http://stackoverflow.com/questions/tagged/ada

[4] https://plus.google.com/communities/102688015980369378804

[5] #Ada on irc.freenode.net

[6] https://gitter.im/ada-lang

[7] https://twitter.com/search?f=realtime&q=%23AdaProgramming

[See also “Ada on Social Media”, AUJ 37-4, p. 184. —sparre]

Ada-related Tools

Ada for Automation

From: Stéphane Los <new.stephane.los@gmail.com>
Date: Thu, 23 Mar 2017 08:10:52 -0700
Subject: "Ada for Automation" Demo Portal
Newsgroups: comp.lang.ada

Thanks to Gnoga (D. Botton) and Simple Components (D. Kazakov), Ada for Automation can offer a Demo Portal so that you can play with it without having to install / build anything.

It features:

- applications running in the cloud and using Modbus TCP protocol to communicate thanks to libmodbus binding
- and applications using PROFINET IO protocol thanks to Hilscher cifX 50-RE board and netRAPID module.

Here is the link to it: http://ada4automation.slo-ist.fr/

[Project homepage: http://slo-ist.fr/ada4autom]

Pixmap to Screen

From: LaeMing Ai <laeming@exemail.com.au>
Date: Sat, 25 Mar 2017 18:01:59 -0700
Subject: Best way to put an array-based pixmap on a screen?
Newsgroups: comp.lang.ada

I am hoping to learn some Ada with an explicit interest in simple software 3D rendering (i.e.: writing the render code in Ada, NOT calling external libraries such as OpenGL). I want to set up a shim between my Ada environment and my Linux Desktop to facilitate monitoring my output, without having to mess about with the daunting (I quickly found) task of interfacing to SDL or GTK+!

All I really want to see from my code's perspective is a 2D array of 32-bit pixel values that I can manipulate from my developing Ada code, and not have to worry about the intricacies of window managers. Is it possible to get help constructing a project template providing the following:

- Function to create a non-resizable X11 window of dimensions N high by 2N wide (N is nominally valued at 512, but could be any value of 2^x above 256).
- Callback for if the above window's close box is activated, to terminate the program.
- Function to copy an array of 2N by N of 32 bit values (8:Red 8:Green 8:Blue 8:ignored) to the above window.

At this stage I am not concerned with input to the window (other than the close box terminating the application, would be nice).

Packing it all into a single source file that I can put to the side and largely ignore would be nice too!

I particularly don't want event loops, or anything else dragged in from outside the Ada environment if at all possible - the point of using Ada is to learn to do all that /in/ Ada! :-)

I have played around with some AdaGTK examples but keep getting bogged down.

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sun, 26 Mar 2017 09:35:11 -0000
Subject: Re: Best way to put an array-based pixmap on a screen?
Newsgroups: comp.lang.ada

> [...]

You might want to take a look at the code Ludovic Brenta provided for his series of presentations in several Ada Developer Rooms at past FOSDEM events. He used a minimal Ada interface to the xcb-library to do exactly that, i.e. to display a 2D array of pixel values on the screen.

For more info, see:

Ada DevRoom @ FOSDEM 2013

Ada Tasking: Multithreading Made Easy

https://www.cs.kuleuven.be/~dirk/ada-belgium/events/13/130203-fosdem.html#multithreading

Ada DevRoom @ FOSDEM 2014

Ada Task Pools: Multithreading Made Easy

https://www.cs.kuleuven.be/~dirk/ada-belgium/events/14/140201-fosdem.html#multithreading

video registration:

http://ftp.belnet.be/FOSDEM/2014/K4601/Saturday/Ada_Task_Pools_Multithreading_Made_Easy.webm

Ada DevRoom @ FOSDEM 2015

Multithreading Made Easy, part 3 - Bounded Work Queues

https://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem.html#multithreading

video registration (very low sound level):

https://ftp.heanet.ie/mirrors/fosdem-video/2015/devroom-ada/multithreading__CAM_ONLY.mp4>

source distribution (latest version):

https://www.cs.kuleuven.be/~dirk/ada-belgium/events/15/150131-fosdem/06-ada-multithreading.tgz

From: reinert <reinkor@gmail.com>
Date: Sun, 26 Mar 2017 06:27:58 -0700
Subject: Re: Best way to put an array-based pixmap on a screen?
Newsgroups: comp.lang.ada

In case it could be useful, here is a simple test program using GLOBE_3D (texture). I just started to use GLOBE_3D for fast (interactive) video-like rendering (programming in Ada):

```
with Text_IO; use Text_IO;
with GL,  GLUT.Devices;
use  GL,  GLUT.Devices;
procedure a1 is

  package Flt_Io is new
          Text_IO.Float_Io(Float);
  package Int_Io is new
          Text_IO.Integer_Io(Integer);
  use Flt_Io,Int_Io;

  GLUT_Problem: exception;
  use GLUT;

  texDat : array (1..64) of aliased gl.ushort;
  texDat_Ptr : constant pointer :=
          to_Pointer(texDat(texDat'First)
          'unchecked_access);

  tex    : aliased GL.uint;
  tex_Ptr : constant GL.uintPtr :=
          tex'unchecked_access;

  n,k : Integer := 1;

begin

  glut.Init;
  glut.SetOption(GLUT.
          ACTION_ON_WINDOW_CLOSE,
          GLUT.ACTION_GLUTMAINLOOP
          _RETURNS);
  glut.InitDisplayMode(GLUT.DOUBLE or
          GLUT.RGB or GLUT.DEPTH );
  glut.InitWindowSize(800,600);
  glut.InitWindowPosition(1, 1);
  if glut.CreateWindow( "Tittel A" ) = 0
  then
    raise GLUT_Problem;
  end if;
  glut.Devices.Initialize;

  n := texDat'First;
  for i in 1..8 loop
     for j in 1..8 loop
        k := (if i = j then 1 else 0);
        k := (if (i + j) mod 2 = 0
         then 1 else 0);
        texDat(n) := gl.ushort(k*100 + 100);
        n := n + 1;
     end loop;
  end loop;

  gl.GenTextures(1, tex_Ptr);
  gl.BindTexture(GL.TEXTURE_2D, tex);

  gl.TexParameter(GL.TEXTURE_2D,
          GL.TEXTURE_MIN_FILTER,
          GL.NEAREST);
  gl.TexParameter(GL.TEXTURE_2D,
          GL.TEXTURE_MAG_FILTER,
          GL.NEAREST);

  gl.TexImage2D(GL.TEXTURE_2D, 0,
          GL.LUMINANCE, 8, 8, 0,
          GL.LUMINANCE_ALPHA,
          GL_UNSIGNED_BYTE,
          texDat_Ptr);

  gl.BindTexture(GL.TEXTURE_2D, 0);
  gl.MatrixMode(GL.PROJECTION);
  gl.Ortho(0.0, 800.0, 0.0, 600.0, -1.0, 1.0);
  gl.MatrixMode(GL.MODELVIEW);
  gl.ClearColor(1.0, 1.0, 1.0, 0.0);
  gl.Clear(GL.COLOR_BUFFER_BIT);
  gl.BindTexture(GL.TEXTURE_2D, tex);
  gl.Enable(GL.TEXTURE_2D);
  gl.Color(1.0, 1.0, 1.0, 1.0);
  gl_Begin(QUADS);
```

```
      gl.TexCoord(0.0, 0.0);
      gl.Vertex(100, 100);
      gl.TexCoord(0.0, 1.0);
      gl.Vertex(100, 500);
      gl.TexCoord(1.0, 1.0);
      gl.Vertex(500, 500);
      gl.TexCoord(1.0, 0.0);
      gl.Vertex(500, 100);
      gl_End;

      gl.Disable(GL.TEXTURE_2D);
      gl.BindTexture(GL.TEXTURE_2D, 0);
      gl.Flush;
      glut.SwapBuffers;

      Delay 10.0;

   end a1;
```

[See also “GLOBE_3D”, AUJ 37-4, p. 186. —sparre]

GNATCOLL

From: Emmanuel Briot <briot@adacore.com>
Date: Tue Apr 4 2017
Subject: New strings package in GNATCOLL
URL: http://blog.adacore.com/new-strings-package-in-gnatcoll

GNATCOLL has recently acquired two new packages, namely GNATCOLL.Strings and GNATCOLL.Strings_Impl. The latter is a generic package, one instance of which is provided as GNATCOLL.Strings.

But why a new strings package? Ada already has quite a lot of ways to represent and manipulate strings, is a new one needed?

This new package is an attempt at finding a middle ground between the standard String type (which is efficient but inflexible) and unbounded strings (which are flexible, but could be more efficient).

GNATCOLL.Strings therefore provides strings (named XString, as in extended-strings) that can grow as needed (up to Natural'Last, like standard strings), yet are faster than unbounded strings. They also come with an extended API, which includes all primitive operations from unbounded strings, in addition to some subprograms inspired from GNATCOLL.Utils and the Python and C++ programming languages.

Small string optimization

GNATCOLL.Strings uses a number of tricks to improve on the efficiency. The most important one is to limit the number of memory allocations. For this, we use a trick similar to what all C++ implementations do nowadays, namely the small string optimization.

[...]

[See also “GNATColl”, AUJ 37-3, p. 128. —sparre]

GNATColl.JSON Support Packages

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Thu, 6 Apr 2017 06:37:12 +0200
Subject: [ANN] gnatcoll-JSON-v1.1.0
Newsgroups: comp.lang.ada

JSON-Support for most types in the Ada.Containers hierarchy.

- Simplify mapping of one-dimensional containers
- Added very crude code generator.

https://github.com/persan/gnatcoll-json/releases/tag/gnatcoll-JSON-v1.1.0

[See also “GNATColl.JSON Support Packages”, AUJ 37-2, p. 75. —sparre]

Simple Components

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Mon, 17 Apr 2017 21:54:13 +0200
Subject: ANN: Simple Components for Ada v4.21
Newsgroups: comp.lang.ada

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, multiple connections server/client designing tools.

http://www.dmitry-kazakov.de/ada/components.htm

The new version extends WebSocket support. It introduces a HTTP server implementation or base type capable to handle connections running over WebSockets instead of normal sockets.

No changes in the implementation of existing connection objects required. For example the same MQTT_Connection object implementing the MQTT protocol can be used this way, and so providing MQTT over WebSocket.

[See also “Simple Components”, AUJ 38-1, p. 4. —sparre]

Ada 2005 Math Extensions

From: Simon Wright <simon@pushface.org>
Date: Thu, 27 Apr 2017 11:52:40 +0100
Subject: ANN:Ada 2005 Math Extensions 20170427
Newsgroups: comp.lang.ada

The latest release of this project[1] is available at Sourceforge[2].

Changes:

- The procedure declaration for the generalized real eigensystem solution was wrong: the eigenvectors should have been complex. This has been corrected.
- The tests now succeed with LAPACK 3.2 to 3.4, and 3.5 to 3.7 (the Fortran results for Real Generalized Eigensystems changed at 3.5). This may be a symptom of some deeper problem, to be investigated.
- The package can be installed with the compiler by 'make install'.
- The code is compatible with GNAT GPL 2016 and FSF GCC 7 (a minor change was required to avoid a compilation warning).

Windows users: there's now a version of LAPACK for Windows[3]; I haven't tried it, but it has to be an improvement on the situation in 2013!

[1] http://gnat-math-extn.sourceforge.net/index.html/

[2] https://sourceforge.net/projects/gnat-math-extn/files/20170427/

[3] http://icl.cs.utk.edu/lapack-for-windows/lapack/

[See also “Ada 2005 Math Extensions”, AUJ 34-3, p. 138. —sparre]

From: Simon Wright <simon@pushface.org>
Date: Tue, 09 May 2017 13:49:01 +0100
Subject: Re: ANN:Ada 2005 Math Extensions 20170427
Newsgroups: comp.lang.ada

> So if understand well, this is an extension of the Ada Arrays in row-major order, right? How is it using Lapack since Fortran has column-major order for its matrices?

By transposing input matrices before handing over to LAPACK, and transposing the results before handing back to the caller.

> I have also seen different projects that use Lapack:

> - Ada Lapack: https://sourceforge.net/projects/ada-lapack/

This is a translation of the Fortran library to Ada, and uses native Ada ordering (row-major).

> - AdaLAPACK: https://sourceforge.net/projects/adalapack/

This is a binding, and its interface is explicitly in terms of Fortran-convention objects; for example,

```
type Fortran_Real_Matrix is
  array (Fortran_Integer range <>,
         Fortran_Integer range <>) of Real;
pragma Convention (Fortran,
         Fortran_Real_Matrix);
```

```
...

procedure SGEEV
  (JOBVL : Character;
   JOBVR : Character;
   N     : Fortran_Integer;
   A     : Fortran_Real_Matrix;
   LDA   : Fortran_Integer;
   WR    : out Fortran_Real_Vector;
   WI    : out Fortran_Real_Vector;
   VL    : out Fortran_Real_Matrix;
   LDVL  : Fortran_Integer;
   VR    : out Fortran_Real_Matrix;
   LDVR  : Fortran_Integer;
   WORK  : in out Fortran_Real_Vector;
   LWORK : Fortran_Integer;
   INFO  : out Fortran_Integer);

pragma Import (Fortran, SGEEV,
"sgeev_");
```

GNAT and UTF-8

From: Simon Wright <simon@pushface.org>
Date: Sun, 30 Apr 2017 18:10:42 +0100
Subject: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

ACATS 4.1 test C250002 involves unit names with UTF-8 characters (the source has the correct UTF-8 BOM, the relevant unit is named C250002_Z where Z is actually UTF-8 C381, latin capital letter a with acute; gnatchop correctly generates a source file with the BOM and name c250002_z where z is actually UTF-8 C3A1, latin small letter a with acute).

On compiling, the compiler (GNAT GPL 2016, FSF GCC 7.0.1) fails to find the file; it says e.g.

```
GNATMAKE GPL 2016 (20160515-49)
Copyright (C) 1992-2016, Free Software
        Foundation, Inc.
gcc -c -I../../../support -gnatW8
        c250002.adb
gcc -c -I../../../support -gnatW8
        c250002_0.ads
End of compilation
gnatmake: "c250002_?.adb" not found
```

I suspect that the problem is down to the .ali file. macOS says

```
$ file -I *
c250002.adb:   text/plain; charset=utf-8
c250002.ali:   text/plain; charset=
        unknown-8bit
c250002.lst:   text/plain; charset=us-ascii
c250002.o:     application/x-mach-binary;
        charset=binary
c250002_0.ads: text/plain; charset=utf-8
c250002_á.adb: text/plain; charset=utf-8
c250002_á.ads: text/plain; charset=utf-8
```

(the last 2 were actually a-acute on the terminal) but the .ali file is confused about whether the representation of the a-acute is C3A1 (good, assuming it gets interpreted as UTF-8 without a BOM) or E3A1 (bad), particularly about the corresponding .ali file name.

Any thoughts? is this a known issue?

(C250001, which has BOMs and UTF-8 identifiers but not file names, works fine with no -gnatW8 messing)

From: Simon Wright <simon@pushface.org>
Date: Sat, 17 Jun 2017 18:20:28 +0100
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> [...]

PR ada/81114 refers[1].

It turns out that this failure occurs on Windows and macOS. The problem is that GNAT smashes the file name to lower case if it knows that the file system is case-insensitive (using an ASCII to-lower, so of course 'smash' is the right word if there are UTF-8 characters in there).

There is an undocumented environment variable that affects this:

```
$
GNAT_FILE_NAME_CASE_SENSITIVE=1
  gnatmake c250002
  gcc -c c250002.adb
  gcc -c c250002_á.adb
  gnatbind -x c250002.ali
  gnatlink c250002.ali
 $ ./c250002

 ,.,. C250002 ACATS 4.1 17-06-17 18:05:55
 ---- C250002 Check that characters above
ASCII.Del can be used in identifiers,
character literals and strings.
 - C250002 C250002_0.TAGGED_Ã _ID.
 ==== C250002 PASSED ============.
```

I wonder why, if the FS is case-insensitive, GNAT bothers at all? (there was, I think, some remark about detecting whether two filenames represented different files).

What do people who actually need to use international character sets do about this? Do you just avoid using international characters in Ada unit names? Or have I just missed the relevant part of the manual?

[1] https://gcc.gnu.org/bugzilla/show_bug.cgi?id=81114

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue, 27 Jun 2017 15:22:11 +0200
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> What do people who actually need to use international character sets do about this? [...]

One of my customers simply has a policy saying that all identifiers have to be in English (the policy doesn't say if it should be American English or proper English), and thus neatly works around the problem.

This reminds me that Jean-Pierre Rosen had a very entertaining tutorial on glyphs, graphemes, alphabets, characters, character sets, encodings, etc. at Ada-Europe 2017 in Vienna. We learnt all kinds of stuff we really don't want to know and worry about. ;-)

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Wed, 28 Jun 2017 00:45:55 +0300
Subject: Re: GNAT vs UTF-8 source file names
Newsgroups: comp.lang.ada

> What do people who actually need to use international character sets do about this? [...]

I use ISO-Latin-1 identifiers in some Ada programs written in a Finnish context, using the Finnish alphabet letters ä, ö, and sometimes the Swedish å. Worked OK for me until *some* of the file systems I use changed from file names with 8-bit characters to UTF-8 file names, after which CVS was quite messed up. I have since limited myself to ASCII in all identifiers that become file name parts in GNAT's file-naming convention, but I still use ISO Latin 1 for other identifiers.

> [...] all identifiers have to be in English [...] works around the problem.

Only if you stick to "modern" English spelling. Otherwise you could have, for example,

```
package Coördinates is ...
```

DB_Maker

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Tue, 2 May 2017 22:19:41 +0200
Subject: DB_Maker
Newsgroups: gmane.comp.lang.ada.gnoga

A generic for creating simple DBs (one table in an RDBMS) with PragmARC.Persistent_Skip_List_Unbounded and a Gnoga UI.

https://github.com/jrcarter/DB_Maker

Includes Movies, a demo program, that could be used to catalog your extensive collection of BetaMax videotape cassettes.

Question: Z-Wave Support?

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 3 May 2017 18:36:12 +0200
Subject: Ada Z-wave support
Newsgroups: comp.lang.ada

Is there serious interest in having Z-wave stack on top of GNAT.Serial_Communications?

(I studied Z-ware protocol a bit. It is overly complicated and has multiple design issues that prevent having a straightforward protocol stack implementation. Existing implementations like openzwave are correspondingly messy. The protocol itself [serial API] is not openly available and must be reverse engineered. All that will require considerable efforts and nothing will work at first.)

[See also “Open Z Wave”, AUJ 37-1, p. 15. —sparre]

Qt5Ada

From: Leonid Dulman <leonid.dulman@gmail.com>
Date: Wed, 31 May 2017 22:39:06 -0700
Subject: Announce : Qt5Ada version 5.9.0 (541 packages) release 01/06/2017 free edition
Newsgroups: comp.lang.ada

Qt5Ada is Ada-2012 port to Qt5 framework (based on Qt 5.9.0 final) Qt5ada version 5.9.0 open source and qt5c.dll,libqt5c.so(x64) built with Microsoft Visual Studio 2015 in Windows, gcc x86-64 in Linux. Package tested with gnat gpl 2012 ada compiler in Windows 32bit and 64bit , Linux x86-64 Debian 8.5

It supports GUI, SQL, Multimedia, Web, Network, Touch devices, Sensors,Bluetooth, Navigation and many others thinks.

Changes for new Qt5Ada release:

Added packages for modules:

QJson,QBson,QMongo (Mongo DB) modules support.

My configuration script to build Qt 5.9.0 is: configure -opensource -release -nomake tests -opengl dynamic -qt-zlib -qt-libpng -qt-libjpeg -openssl-linked OPENSSL_LIBS="-lssleay32 -llibeay32" -plugin-sql-mysql -plugin-sql-odbc -plugin-sql-oci -icu -prefix "e:/Qt/5.9"

As a role ADA is used in embedded systems, but with QTADA(+VTKADA) you can build any desktop applications with powerful 2D/3D rendering and imaging (games, animations, emulations) GUI, Database connection, server/client, Internet browsing , Modbus control and many others thinks.

Qt5Ada and VTKAda for Windows, Linux (Unix) is available from https://drive.google.com/folderview?id=0B2QuZLoe-yiPbmNQRl83M1dTRVE&usp=sharing (google drive. It can be mounted as virtual drive or directory or viewed with Web Browser)

The full list of released classes is in "Qt5 classes to Qt5Ada packages relation table.docx" VTKAda version 7.1.0 is based on VTK 7.1.0 (OpenGL2) is fully compatible with Qt5Ada 5.9.0

I hope Qt5Ada and VTKAda will be useful for students, engineers, scientists and enthusiasts With Qt5Ada you can build any applications and solve any problems easy and quickly.

[See also “Qt5Ada”, AUJ 38-1, p. 6. —sparre]

ASIS2XML

From: Simon Wright <simon@pushface.org>
Date: Sat, 03 Jun 2017 15:25:23 +0100
Subject: ASIS2XML 20170603
Newsgroups: comp.lang.ada

Minor release, at

https://sourceforge.net/projects/asis2xml/files/20170603/

- Compatible with GNAT GPL 2016, FSF GCC 6, 7.
- Fixes bug #3, Package head components missing.
- Updates Makefile, add install target.

[See also “ASIS2XML”, AUJ 35-2, p. 80. —sparre]

ArchiCheck

From: Lionel Draghi <Lionel.Draghi@Ada-France.org>
Date: Thu, 8 Jun 2017 00:06:01 +0200
Subject: Ann: ArchiCheck v0.1
Newsgroups: comp.lang.ada

Archicheck is a simple tool to describe and enforce architecture/design decision that are not easily checked by languages and compilers.

This code was written twelve years ago, and even announced on comp.lang.ada (http://groups.google.com/group/comp.lang.ada/browse_thread/thread/4a195a443fce793e/41bb2cb527464bab?q=comp.lang.ada+example+of+layered+software#41bb2cb527464bab) in November 2005, but never released.

Here it is, essentially as it was in 2005, build in the same environment (Darcs, Gnat, NaturalDocs, Dia, OpenToken, etc.).

This tool was written because of my frustration that simple design decision where not complied with.

A classical and recurring case was that for a bug fix, someone adds a “with” in the code, that created a visibility to a package supposed to be in an upper layer.

Few month of such a code spaghettization resulted in subtle elaboration order problems, and obviously in code degradation.

Another classical design decision not so easy to verify is to forbid any dependencies on OS specific code, except in an appointed “portability” package.

Ada couldn’t prevent this (not to mention other languages), and I found no external tools able to verify it.

This is why I started the Archicheck project, as a tool that read a text file containing in simple statements the architecture description, and check the code compliance.

Consider the my_rules.txt file:

```
Presentation_Layer contains A B C packages
Application_Layer contains D E F packages
Domain_Layer contains G H I packages
Presentation_Layer is a layer over Application_Layer
Application_Layer is a layer over Domain_Layer
Only Hardware_Abstraction_Layer can use Interfaces.C
```

Just put:

```
archicheck my_rules.txt –I ./my_src
```

in your make file, and that’s it.

This is a POC, the code is not at all robust, or even well tested, and it check only Layer rules at this stage (cf. Tests section here: http://lionel.draghi.free.fr/Archicheck/index.html).

Any feedback or comment is welcome (even on the tool’s name!).

More info on http://lionel.draghi.free.fr/Archicheck/index.html

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 7 Jun 2017 20:04:54 -0500
Subject: Re: ArchiCheck v0.1
Newsgroups: comp.lang.ada

> [...]

pragma Profile (No_Implementation_Extensions);

forbids any dependence on implementation-defined stuff. That eliminates most (not all) sources of non-portable code, it clearly includes target-dependent packages like Ada.Directories.Information. See 13.12.1(9-13/3). By far the best way to avoid OS-specific code is to stick to the language-defined packages!

Of course, you could use an OS-Dependent package of your own design (or one picked up off the Internet), and the language can't possibly help with that. So your tool certainly has value beyond what the language (any language for that matter) can do.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 8 Jun 2017 22:27:34 -0500
Subject: Re: ArchiCheck v0.1
Newsgroups: comp.lang.ada

[...]

Another item that would help for this sort of thing is

```
pragma Restrictions (No_Dependence,
        <some package name>);
```

which prevents any use of the named package (which doesn't actually have to exist anywhere).

For instance,

```
pragma Restrictions (No_Dependence,
        GNAT);
```

prevents any use of package GNAT or any of its children in your program. (Which will get rid of a lot of GNAT-defined facilities.)

In your case, one can use it to prevent using various locally defined packages that might not be portable.

Ada-related Products

State of the Compiler Market

From: John McCabe <john@mccabe.org.uk>
Date: Wed, 22 Feb 2017 16:26:17 -0800
Subject: State of the compiler market
Newsgroups: comp.lang.ada

I have honestly been searching on Google for this, but I can't easily find out who produces a commercial (or free) compiler for Ada 2012, other than AdaCore. Can anyone give me some pointers please?

I've looked at some compilers I've used in the past but they seem to have stagnated around Ada 95.

Also, are there any modern (i.e. Later than Ada 95) compilers for a MIPS target, or for any DSPs?

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Thu, 23 Feb 2017 06:16:11 +0100
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

[...]

And concerning MIPS there is a open cross built on debian: https://packages.debian.org/sid/devel/gnat-5-mips-linux-gnu

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Wed, 22 Feb 2017 22:01:56 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

For Ada 2012 it's probably AdaCore's GNAT only so far.

For Ada 2005 there are some other options - whether they are complete is another question. At least PTC ObjectAda can build correctly a program using maps, containers and the dot notation for objects (e.g. GLOBE_3D demos).

Here is a compiler list obtained by googling, then filtered over time: http://unzip-ada.sourceforge.net/#adacomp

From: Joakim Strandberg <joakimds@kth.se>
Date: Thu, 23 Feb 2017 01:01:44 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

The ICC Ada compiler supports Ada 2005: http://www.irvine.com/

From: Randy Brukardt <randy@rrsoftware.com>
Date: Thu, 23 Feb 2017 15:22:28 -0600
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

So far as I know, only AdaCore has a full Ada 2012 compiler.

The next release of Janus/Ada will support a smattering of Ada 2012 features, to go with a smattering of Ada 2007 features, and most (but not quite all) of Ada 95. (Customers should be able to try a preview of this version next month.)

Both PTC compilers (ObjectAda and Apex) are Ada 2005, as is Irvine.

Everything else seems to be stuck on Ada 95.

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue, 27 Jun 2017 10:00:25 +0200
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

PTC has announced that they are working on adding Ada 2012 features to PTC ObjectAda.

If I remember the presentation at Ada-Europe 2017 correctly, the first step is aspects, contracts, conditional expressions and expression functions.

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 26 Feb 2017 00:20:45 +0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

This is an issue for us open source people, the idea of a new compiler has been discussed here before. Randy has stated people shouldn't bother, but I disagree as there are reasons to have another one. One reason being that if there isn't an alternative there will be one company monopolising the future of the language and there will be no further work to expand the language into new areas and will continue to stay in aerospace.

If this happens, a new language derived from Ada would need to happen and fast. Again, this has been discussed, see the thread started by David Botton about the getadanow logo competition.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Sat, 25 Feb 2017 20:26:49 -0600
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

To be fair, I've said people shouldn't bother simply because the odds of it reaching a usable state are near zero. As several people have noted, the parser is the easy part. Figuring out Ada resolution rules, Ada tasking, the Ada optimization rules, and many other things will sap anyone of energy long before they complete it. (Note that we never had an intent to build a full Ada compiler when we started out. We got pushed that way when others got interested. And 35+ years have elapsed...)

I think a better approach would be to convince an existing Ada 95 implementation to go open source and then enhance that to do the things desired (Ada 2020 support, etc.). I'd consider it if (a) there were people truly interested and (b) it was reasonably obvious how to monitize the result (that is, provide enough revenue for living).

From: Paul Rubin
Date: Fri, 24 Feb 2017 01:32:21 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> Could agree on that but a fair question is: How many real compiler-vendors are there for C and C++?

Gcc, Clang, icc, Microsoft, I guess that's all the current C++ that I can think of though there are probably a few more. "Current" for now means the compiler supports C++14 or comes pretty close.

There's plenty of C compilers around, especially in the embedded world. CompCert (compcert.inria.fr) has a formally verified code generator, which has never been done for Ada.

From: Dirk Craeynest <dirk@cs.kuleuven.be>
Date: Sun, 26 Feb 2017 10:14:07 -0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> I think a better approach would be to convince an existing Ada 95 implementation to go open source and then enhance that to do the things desired (Ada 2020 support, etc.).

Hear, hear!!!

Action for all: Talk to your favorite Ada95/2005 vendor, and ask them about their plans to upgrade to newer Ada features!

From: reinkor <reinkor@gmail.com>
Date: Wed, 1 Mar 2017 00:43:40 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

I got a bit scared from this discussion since I am investing much time in developing an application in Ada.

Can Adacore's (gnat) public licence safeguard the language is available the coming 10-20 years? I.e. can I stay relaxed? :-)

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue, 27 Jun 2017 10:29:57 +0200
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...] can I stay relaxed? :-)

I think you can.

My bet is that Ada will be around at least another 20 years.

At some point we will have accept to break backwards compatibility, but I can't see how and when it will happen.

RapiTestFramework

From: Rapita Systems
Date: Tue May 9 2017
Subject: An early look at RapiTestFramework
URL: https://www.rapitasystems.com/blog/early-look-rapitestframework

We have been hard at work recently developing RapiTestFramework, a new RVS tool that generates and runs unit, integration and system tests on embedded targets or host systems. RapiTestFramework injects test framework code, builds a test harness, runs this on the target and reports results.

RapiTestFramework has come along very well during its early development, and has been used by both the embedded software industry and ourselves (as part of our new software verification services). What's more, we have been working on kits to help you qualify the tool for use in your DO-178B/C or ISO 26262 projects.

In this post, we thought we'd share with you some of the design decisions we made during the development of RapiTestFramework, which led to the strong position the tool is in today.

Thinking of the future

All of our tools are built to meet the evolving needs of embedded software verification, and RapiTestFramework is no exception. Because these needs change over time, we built RapiTestFramework to be flexible from the outset.

We did this by spending the early design period concentrating on the features we need from the tools that parse source code and inject tests, based on real needs identified from our customers.

By using a bottom-up approach and leaving questions about higher-level features such as the user-interface for later, we could spend the time needed to engineer a flexible tool that will be relevant for years to come.

There are some really exciting features we can implement using RapiTestFramework, such as data and control coupling and identifying infinite loops in your source code before running it, but you'll have to wait until a future blog post for more on that.

[...]

[See also “Highlights from Ada Europe 2016”, AUJ 37-3, p. 124. —sparre]

ObjectAda

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue Jun 13 2017
Subject: Jacob Sparre on Twitter
URL: https://twitter.com/Jacob_Sparre/status/874657941814730752

PTC announces work on #Ada2012 support in Object Ada. #AdaProgramming #AdaEurope2017

[Slide transcription follows. --sparre]

Ada Products in Development

- Corrections / enhancements to existing product releases
- ObjectAda V10.0 for Windows
  o Visual Studio 2017 & windows 10 SDK
  o Phased introduction of Ada 2012 support
    + V10.0: new expression forms, Aspects & Contracts

[End slide transcription. --sparre]

[See also “Apex Ada, Object Ada and Ada World”, AUJ 37-1, p. 17. —sparre]

Ada and Operating Systems

Debian: gprinstall?

From: Simon Wright <simon@pushface.org>
Date: Tue, 14 Feb 2017 17:01:17 +0000
Subject: Debian & gprinstall
Newsgroups: comp.lang.ada

Has anyone devised a setup to use gprinstall to install a library to match the Debian Ada Policy?

I don't care so much (perhaps I should) about getting the so names right, but I'd like at least to get the library GPRs installed where Debian looks for them.

At the moment, on jessie, gprinstall installs:

Sources in /usr/include/<library>/

ALI files in /usr/lib/<library>/

Objects in /usr/lib/<library>/
(for a relocatable library, a symlink to the so in /usr/lib/<library>/ is put in /usr/lib)

GPRs in /usr/share/gpr/

and at the very least GPRs should be in /usr/share/ada/adainclude/. That's easy enough using
--project-dir=share/ada/adainclude,
but what about the other directories? (they need to be reasonably out of the way of anywhere that Debian would install software).

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Tue, 14 Feb 2017 18:57:10 +0100
Subject: Re: Debian & gprinstall
Newsgroups: comp.lang.ada

> [...]

But gprinstall cannot follow policies of all Linux flavors. Fedora's policy is very different, e.g. it has a system gpr project to include into the library project and get directories' paths from there.

If indeed gprinstall were to care of the Linux target I would suggest to generate a proper project installation file, *.deb, *.rpm and use it in the standard system installer.

I wrote an extremely ugly bash script to generate Debian and Fedora project files. It would be a great help if gprinstall did this (and *msi setup for Windows). Not that I believe that to happen...

From: Simon Wright <simon@pushface.org>
Date: Tue, 14 Feb 2017 21:30:32 +0000
Subject: Re: Debian & gprinstall
Newsgroups: comp.lang.ada

> [...]

I only want a set of switches that will make gprinstall do something that works on Debian with the system compiler (no problem if using GNAT GPL, of course, since the default setup is what GNAT GPL expects) and doesn't trample over other software.

Is the Fedora equivalent of the Debian Ada policy written up anywhere?

> [...]

But I'm not going to distribute binary library packages for any Linux flavours. So I don't need to understand .deb, .rpm, etc etc. I just want the _source_ packages to be easily installable.

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 15 Feb 2017 09:44:17 +0100
Subject: Re: Debian & gprinstall
Newsgroups: comp.lang.ada

> [...]

I think a natural way would be if AdaCore added package Installer to the project file which could contain variables and commands to gprinstall.

> Is the Fedora equivalent of the Debian Ada policy written up anywhere?

https://fedoraproject.org/wiki/Packaging:Ada

> [...] I just want the _source_ packages to be easily installable.

Sure, but there is no easy way, not any more. I mean, with all these multi-arch systems and cross compilers it becomes more and more complicated. So maybe in the end it would be easier to do it in a hard way...

From: Simon Wright
<simon@pushface.org>
Date: Wed, 15 Feb 2017 09:29:48 +0000
Subject: Re: Debian & gprinstall
Newsgroups: comp.lang.ada

[...] See package Install in

http://docs.adacore.com/gprbuild-docs/html/gprbuild_ug/gnat_project_manager.html#packages

Not sure when it was introduced.

Mac OS X: AdaControl

From: Jean-Pierre Rosen
<rosen@adalog.fr>
Date: Thu, 30 Mar 2017 07:21:36 +0200
Subject: AdaControl available for MacOS
Newsgroups: comp.lang.ada

Thanks to Pascal Pignard, an executable distribution of the current version of AdaControl is available from AdaControl's page:

http://www.adacontrol.fr

Nitpick:

To install it, type:

gprinstall inst.gpr

(will be in the doc with next version)

[See also “AdaControl”, AUJ 37-4, p. 189. —sparre]

CP/M: Towers of Hanoi

From: Alexey Voronin
<dreamy16101976@gmail.com>
Date: Wed, 19 Apr 2017 14:44:38 -0700
Subject: Ada compiler on Arduino Nano 3.0
Newsgroups: comp.lang.ada

I ran the Ada language compiler on my Arduino Nano-based emulator of computer with Intel 8080 and OS CP/M!

https://acdc.foxylab.com/sites/default/files/cpm_mk1.jpg

http://acdc.foxylab.com/node/76 (in russian)

https://github.com/Dreamy16101976/cpm4nano

Compilation of the program TOWERS.ADA for solving the problem of the "Hanoi Tower":

https://acdc.foxylab.com/sites/default/files/towers_ada_comp.png

Running the compiled program TOWERS.COM:

https://acdc.foxylab.com/sites/default/files/towers_ada_3.png

From: Weston Pan
<pan.weston@gmail.com>
Date: Wed, 19 Apr 2017 16:59:09 -0700
Subject: Re: Ada compiler on Arduino Nano 3.0
Newsgroups: comp.lang.ada

Congratz!

There is an old Janus/Ada 1.5 CP/M compiler as well (http://www.retroarchive.org/cpm/lang/JANADA15.ZIP). Maybe you can try that out too.

Mac OS X: GCC

From: Simon Wright
<simon@pushface.org>
Date: Mon, 15 May 2017 20:14:20 +0100
Subject: ANN: GCC 7.1.0 for macOS
Newsgroups: comp.lang.ada

GCC 7.1.0 for macOS El Capitan and Sierra, built for C, C++, Ada, Fortran, Objective-C, and Objective-C++, and including Ada ASIS, AUnit, GDB, Gprbuild, GNATColl, and XML/Ada, is available at https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/7.1.0/native-2016

[See also “Mac OS X: GCC”, AUJ 37-2, p. 77. —sparre]

Mac OS X: GCC for ARM-EABI

From: Simon Wright
<simon@pushface.org>
Date: Sat, 20 May 2017 16:49:27 +0100
Subject: ANN: GCC 7.1.0 for arm-eabi
Newsgroups: comp.lang.ada

GCC 7.1.0, rebuilt as a cross-compiler from macOS to arm-eabi, is available at https://sourceforge.net/projects/gnuada/files/GNAT_GCC%20Mac%20OS%20X/7.1.0/arm-eabi/

The compiler is known to run on El Capitan and Sierra; it may not run on earlier OS X releases.

The compiler was tested with the Cortex-M3 as found on the Arduino Due[1] and the Cortex-M4 as found on the STMicroelectronics[2] STM32F4 Discovery and STM32F429I Discovery boards. Note, however, that GCC 7 has implemented multilib support for other ARM chips.

GNAT GDB 2016 (rebuilt for arm-eabi) is included.

The compiler comes with no Ada Runtime System (RTS). See the Cortex GNAT Run Time Systems project[3] for candidates.

NOTE 1: the compiler/RTS interface has changed; for the time being, you will need to check out the latest repository update.

NOTE 2: for the same reason, this compiler can't presently be used with AdaCore's embedded-runtimes repository at Github[4].

[1] http://www.arduino.com

[2] http://www.st.com

[3] https://sourceforge.net/projects/cortex-gnat-rts/

[4] https://github.com/AdaCore/embedded-runtimes

[See also “Mac OS X: GCC for ARM-EABI”, AUJ 37-2, p. 78. —sparre]

References to Publications

Up-to-date Version of the RM

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Mon, 20 Feb 2017 16:03:22 -0600
Subject: Re: relax double parens in expression_function_declaration?
Newsgroups: comp.lang.ada

> [...] However, I can't find a rule in the ARM that allows it.

You're looking at the wrong version of the RM, then: that was fixed in the Corrigendum. Since "Corrigendum" means "bug-fixes" in ISO-speak, it's silly to use the original version. Find the Corrigendum version of the RM on Ada-Auth.org: http://www.ada-auth.org/standards/ada12_w_tc1.html. (Or from the main ARM page: http://www.ada-auth.org/arm.html). Since there are fixed scattered throughout the manual, one can never guess whether changes were made to a particular rule.

You can also find the working version of the RM on Ada-auth.org, which includes everything that's been approved for the next version of Ada (whenever that will be); that includes more bug fixes as well as some new features: http://www.ada-auth.org/standards/ada2x.html (Whether or not any compiler supports the new stuff is of course up to them; I know GNAT does support some of the new features.)

SPARK: Proving GCD

From: Yannick Moy <moy@adacore.com>
Date: Wed, 5 Apr 2017 16:52:44 -0700
Subject: Re: [Spark] Proving GCD
Newsgroups: comp.lang.ada

I had proved various versions of GCD some time ago, and needed some motivation to write a blog post about it. Your question gave me the impulse! Here is the blog post, that should answer your question:

http://www.spark-2014.org/entries/detail/gnatprove-tips-and-tricks-proving-the-ghost-common-denominator-gcd

Ada Inside

SparForte

From: Ken O. Burtch
<koburtch@gmail.com>
Date: Sun, 26 Mar 2017 13:47:00 -0700
Subject: [ANN] SparForte 2.0.3
Newsgroups: comp.lang.ada

A new version of SparForte, my Ada-based shell, scripting language and web template engine, has been released. It is maintained by myself and other volunteers.

The focus of version 2.0.3 is additional bug fixes following the release of SparForte 2.0. This release includes 25 changes.

A major change in this version: the "new" functions (e.g. doubly_linked_lists.new_list) have been replaced with a new parameterized type system (e.g. list : doubly_linked_list.list(strings);).

The release notes can be viewed from

http://www.sparforte.com/news/2017/news_mar2017.html

SparForte may be downloaded from

http://www.sparforte.com/download.html

If you encounter any problems, please note I don't read comp.lang.ada regularly.

[See also “SparForte”, AUJ 38-1, p. 11. —sparre]

KDF9 Emulator

From: Bill Findlay <findlaybill@blueyonder.co.uk >
Date: 11 Apr 2017 00:50:30 GMT
Subject: KDF9 emulator
Newsgroups: alt.folklore.computers, comp.lang.ada

A new version of ee9, my emulator for the English Electric KDF9, is now available, with much other info about the '9, at:

http://www.findlayw.plus.com/KDF9/

Binaries for Mac OS and Linux are provided. The latter also runs on Windows 10 (!)

I hope to offer binaries for Windows <10, and perhaps for the Raspberry Pi in due course.

Older binaries for the latter two systems are already available.

ee9 is written in Ada 2005, using GNAT, the GNU Ada compiler.

[See also “KDF9 Emulator”, AUJ 36-4, p. 208. —sparre]

MAX! Home Automation

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 19 Apr 2017 18:30:25 +0200
Subject: ANN: MAX! home automation v1.9
Newsgroups: comp.lang.ada

The software is an Ada GUI application to control wireless radiator thermostats produced under brands MAX! and ELV.

http://www.dmitry-kazakov.de/ada/max_home_automation.htm

Changes to the previous version:

- MQTT on WebSocket support added;
- The MQTT server's topic names have the leading '/' removed because some MQTT clients reported having troubles with names starting with /';
- When controlled over HTTP or MQTT the target temperature can be specified to be airing, comfort, and eco according to the thermostat settings;
- The thermostat address can be omitted when in control commands over HTTP or MQTT. In this case the command applies to all thermostats;
- HTTP access control (CORS) support added;
- HTTP server JSONP support added.

[See also “MAX! Home Automation”, AUJ 38-1, p. 11. —sparre]

Pasta!

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Sat, 20 May 2017 12:48:53 +0000
Subject: Re: Experience with Amazon Web Services ?
Newsgroups: gmane.comp.lang.ada.gnoga

There is now a permanent URL: http://pasta.phyrama.com/

The program is work in progress of course - first priority, availability of Gnoga for touchscreen devices...

From: Gautier de Montmollin <gautier.de.montmollin@gmail.com>
Date: Thu, 1 Jun 2017 23:44:16 +0000
Subject: Pasta! - levels 11 to 15 available
Newsgroups: gmane.comp.lang.ada.gnoga

...same address: http://pasta.phyrama.com/

Eventually you'll have to redo level 10 - sorry for that.

There was a bug occurring right after last available level was completed, before the increment of the counter of completed levels; this has been corrected.

Now there is some basil leaves appearing among the pasta...

Have fun!

[...]

PS: level 15 is though, but can be done with some luck and experience. I

test levels right after designing them.

The level design is done (as for the Leaves demo) in Ada too, e.g. :

```
when 14 =>
 return
  (columns       =>  7,
   rows          =>  5,
   thresholds    =>  1,
   needed        =>
           (1 => (kind => 'A', min => 10)),
   startup_board  => (
     1 | 7  => (3 => blocking_2,others => ' '),
     3 | 5  => (3 => blocking_3,others => ' '),
     others => (others => ' ')
   ),
   moves          => 10,
   level_probs    => ('A' .. 'E' => 1.0 / 5.0,
           others => 0.0),
   logo_x         => logo_x,
   logo_y         => logo_y
  );
```

Ada in Context

Dot Notation and Untagged Types

From: Luke A. Guest <laguest@archeia.com>
Date: Sun, 5 Feb 2017 06:17:04 -0800
Subject: Why can't Ada use dot notation on private types?
Newsgroups: comp.lang.ada

A lot of people dislike the fact that you cannot use dot notation on a tagged type if it's private. Is there a reason for this? Surely the compiler knows it's tagged when it looks it up?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 6 Feb 2017 13:44:37 -0600
Subject: Re: Why can't Ada use dot notation on private types?
Newsgroups: comp.lang.ada

> [...]

That was the original idea. But we ran into semantic problems with elementary types (and private types completed with elementary types), so we ended up restricting it to tagged only. (The usual reason applied here: it's better to get a limited version of a feature right -- it can be expanded later if necessary -- that to get a general version of a feature wrong -- because then we're stuck with the mistakes forever.)

The problems mainly come from the possibility of implicit .all and 'Access. I personally think we could have done without the later, but the former is traditional Ada semantics which would be weird to not support in this prefix form.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 8 Feb 2017 17:26:38 -0600
Subject: Re: Why can't Ada use dot notation on private types?
Newsgroups: comp.lang.ada

Dmitry A. Kazakov wrote:

> [...] I want to override "all", so let me do this.

When elementary types are included in prefix notation, there are potentially an infinite number of implicit .all or 'Access combinations. A compiler would have trouble figuring them out (and it could get *very* expensive), and a reader would be even more confused.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Mon, 6 Feb 2017 13:40:26 -0600
Subject: Re: Why can't Ada use dot notation on private types?
Newsgroups: comp.lang.ada

> [privately tagged type]

That would be privacy breaking, of course: the legality of the source would depend upon the private details.

But of course it is legal if you have a tagged private type. (Once done, of course, the fact that the type is tagged is visible.) Completing an untagged private type with a tagged type causes various semantic issues, so I'd recommend against that anyway.

Compile-Time Check for Constant

From: Niklas Holsti <niklas.holsti@tidorum.fi>
Date: Thu, 9 Feb 2017 20:46:00 +0200
Subject: Re: Test for constant (or "in parameter" in procedure)
Newsgroups: comp.lang.ada

[Check if a entity is a constant/in parameter at compile-time.]

> [...]

I think I see a valid point in the OP's question. One can imagine a program in which a package A supplies a value ("i" in the example), and in which another, loosely related package B, perhaps in quite another part of the program, implements an algorithm that logically depends on this value never changing (after elaboration). It would be nice to be able to express this inter-package constancy constraint somehow, at compile time, as a guard against future mistakes in program evolution that might make the value non-constant and so destroy the logical basis for the algorithm in package B.

At present, it seems to me that Ada offers only two ways to do this.

The first way is to make the constant a named number in A, or a constant with a static value, and make a copy as a named number in B. Like this:

```
package A is
   I : constant Integer := 5;
   -- A named number or (as here) a
  -- constant with a static value.
end A;

with A;
package B is
   My_I : constant := A.I;
   -- A named number. This does not
   -- compile if A.I is non-constant,
   -- or has a non-static value (even if
  -- constant).
end B;
```

However, this only works if the type of the constant is numeric.

The second way is just to document the reasons for the object being declared constant, at the point of that declaration:

```
package A is
   I : constant Integer := 5;
   -- This must be a constant, because
   -- package B etc. etc.
end A;
```

I see less reason for a run-time check of constancy. However, there is a similar thing in Ada: the attribute 'Constrained. Perhaps there would be some interest in an attribute 'Constant, which could be applied to any object or named number and would be True for named numbers and constant objects, and only for those. On first look, it seems very simple to implement (but perhaps there is some complication in shared-code generics...)

Preferred GNAT Options?

From: reinkor <reinkor@gmail.com>
Date: Wed, 8 Feb 2017 20:17:44 -0800
Subject: Hint to Ada newbies
Newsgroups: comp.lang.ada

Just sharing experience:

I tried gnat compiler option "-gnatwa" and revealed a lot of weakness in my Ada project. It also helped to improve the structure of the code which I have worked on and modified over time.

Just a hint to newbies :-)

https://gcc.gnu.org/onlinedocs/gnat_ugn/Warning-Message-Control.html

From: Per Sandberg <per.s.sandberg@bahnhof.se>
Date: Thu, 9 Feb 2017 07:06:26 +0100
Subject: Re: Hint to Ada newbies
Newsgroups: comp.lang.ada

Some more hints on switches:

This is a snippet of my usual project file during development. It will force me to keep a consistent layout and catch a lot of errors during compile and tests.

my.gpr:

```
project My is

   for Source_Dirs use ("src");
   for Object_Dir use ".obj";
   for Main use ("main.adb");

  package Builder is
   for Default_Switches("Ada")
          use ("-k", "-j0");
   for Global_Configuration_Pragmas
          use project'Project_Dir & "my.gpp";
  --  Initialize data to bad values.
  end Builder;

  package Compiler is
     for Switches ("ada") use
       ("-gnatyybcfhiklnprtu", "-gnatyN256",
          "-gnaty3", -- Check layout.
          "-gnatVa",
          --  Turn on validity checking options
          "-gnatwa", --  Turn most info/Warning
          "-gnatwe",
          --  Treat all warnings as errors
          "-gnata",  --  Enable Assertions.
          "-gnateE",
          --  Generate extra information in
          --  exception messages
          "-gnatQ");
          --  Don't quit, write ali/tree file even if
          --  compile errors
    end Compiler;

    package Binder is
      for Switches ("ada") use
        ("-E");  -- Store callstack in exceptions.
    end Binder;
  end Test_My;
```

my.gpp

```
  pragma Initialize_Scalars;
```

From: Simon Wright <simon@pushface.org>
Date: Thu, 09 Feb 2017 08:02:21 +0000
Subject: Re: Hint to Ada newbies
Newsgroups: comp.lang.ada

> [...]

(a) personally I use -gnaty on its own (equivalent of »-gnaty3abcefhiklmnprst«, run 'gnatmake -h' for the switches). If I really didn't want e.g. the standard -gnatye (check end/exit labels present), I think I'd say "-gnaty -gnaty-e".

(b) -gnatyN256? M256 I could understand, if you don't mind unreadably long lines, N means "turn off all checks", N256 is undefined.

From: reinkor <reinkor@gmail.com>
Date: Thu, 9 Feb 2017 01:26:54 -0800
Subject: Re: Hint to Ada newbies
Newsgroups: comp.lang.ada

> [...]

I often use Pretty_Printer defaults like this:

```
  package Pretty_Printer is
    for Default_Switches ("ada")
            use ("-c2", "-l2", "-c3", "-c4");
  end Pretty_Printer;
```

This seems to be conflicting with your layout check. Any proposal for Pretty_Printer Default_Switches ?

From: stevenselectronicmail@gmail.com
Date: Sat, 11 Feb 2017 09:40:38 -0800
Subject: Re: Hint to Ada newbies
Newsgroups: comp.lang.ada

> [...]

I prefer using pragmas for these.

```
  pragma Initialize_Scalars;
  pragma Assertion_Policy (CHECK);
  pragma Debug_Policy (CHECK);
  pragma Check_Float_Overflow;
  pragma Overflow_Mode (Strict);
  pragma Unsuppress (Overflow_Check);
  pragma Validity_Checks (ALL_CHECKS);
```

```
   pragma Prefix_Exception_Messages;

   pragma Style_Checks (On);
   pragma Style_Checks
("3aAbCdefiIklnprsStu");

   pragma Warnings (On);
   pragma Warnings ("ah.i.ktFK");
```

Formally Verified Compiler?

From: Paul Rubin
Date: Sun, 26 Feb 2017 01:14:30 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

1. I found a few google hits for Ada 2020 but nothing like a concrete list of changes or proposals. Just "ARG is working on it". Is there anything public?

2. Is there really serious difficulty extending an Ada 95 compiler to handle Ada 2012?

I'm a FOSS supporter myself, but in the case of an alternate Ada implementation, I'd think the most interesting possibility would be for someone (maybe Adacore) to team up with the CompCert guys and make a verified Ada compiler.

> Figuring out Ada resolution rules, Ada tasking, the Ada optimization rules, and many other things will sap anyone of energy long before they complete it.

How bad is this really? Is the ARM that hard to read (I've never tried)? On the surface Ada doesn't seem particularly harder than Java, and certainly easier than C++. But I haven't gone into the deeper corners or used it much.

From: Waldek Hebisch
Date: Sun, 26 Feb 2017 17:35:29 +0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...] make a verified Ada compiler.

Maybe a blasphemy here, but if you go through full formal verification does Ada offer significant advantages over C or C++? L4.sec guys deliver code via C and apparently this did not hamper their efforts. Potentially Ada may be more productive, however in formally verified project actual C coding is likely to be small part of total effort, so replacing C with Ada should not make much difference. I would think that Ada strength is in intermediate area where better checking in compiler and safer programming practices lead to faster delivery of reasonably good program.

From: Paul Rubin
Date: Sun, 26 Feb 2017 14:32:56 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...] if you go through full formal verification does Ada offer significant advantages over C or C++?

You mean if the application is verified? Depends on what has been verified, I suppose ;). The idea of CompCert is just that the compiler is verified so there's less worry about compiler bugs. That's independent of the verification processes you might use for your application.

> L4.sec guys deliver code via C and apparently this did not hamper their efforts.

SEL4 is apparently around 10 KLOC of C and 480 KLOC of Isabelle/HOL proofs, and the C was apparently produced (not sure whether manually or automatically) by some kind of derivation from an executable specification written in Haskell. I haven't actually read the SEL4 papers but that comes from a quick glance. They are accessible from here: http://ts.data61.csiro.au/projects/seL4/

> Ada strength is in intermediate area where better checking in compiler and safer programming practices lead to faster delivery of reasonably good program.

This is reasonable to say. Also Ada's verification tools like SPARK supply more automation than what's available for C as far as I can tell.

From: Waldek Hebisch
Date: Mon, 27 Feb 2017 02:38:48 +0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

Once you have verified compilation process it is natural to go for full formal verification, from specification to machine code. Actually, given that formal verification of chips is quite advanced once can go for verification up to logic gates. This is it should be possible to exclude logic bugs in chips.

> [...]

I read one of the [SEL4] papers and they write that translation from Haskell to C was done by hand. They state that they used subset of Haskell avoiding features hard to translate to lever level language, so presumably one could create appropriate translator. OTOH IIRC according to their time report writing C code took less than 15% of total time, so it is possible that doing translation manually was faster than developing translator program.

> [...]

Few years ago I talked with guy from Microsoft Research doing formal verification. He claimed that their tools checked more things than SPARK. And also he claimed that they needed less annotations (he said that methodology of SEL4 required about 8 times more annotations than required by Microsoft tools). Main point was availability of quite strong proof engine and automatic generation of intermediate conditions.

From: Paul Rubin
Date: Sun, 26 Feb 2017 18:54:16 -0800
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

[...]

> according to their time report writing C code took less than 15% of total time

Given that it was 2% of the code per the above, 15% of the time doesn't make it sound easy.

> [...] Microsoft Research [...] tools checked more things than SPARK.

I can believe that, especially with older versions of SPARK. I'd be interested to know which verification system the guy was describing.

[...]

From: Waldek Hebisch
Date: Mon, 27 Feb 2017 03:54:02 +0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

That 15% included related proof annotations -- going from verified Haskell to verified C.

> [...]

He was from M. Leino group. IIRC system in question were boogie and VCC.

From: Randy Brukardt
<randy@rrsoftware.com>
Date: Tue, 28 Feb 2017 14:51:12 -0600
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [Ada 2020 proposals?]

The date isn't even firm yet, so it's rather early to read too much into that. But all of the work is publically available on Ada-auth.org. Summaries and the like will likely wait until we get closer to a deadline (who can tell today if the parallel block and loop stuff will get mature enough to include??).

> 2. Is there really serious difficulty extending an Ada 95 compiler to > handle Ada 2012?

Everything is possible with software, but how much effort are you willing to put in? There aren't any major breakages, but there's a lot of new stuff to integrate. It took me a month to get the grammar (alone) to be usable and not break anything already implemented.

> How bad is this really? [...]

Ada resolution is unique in that the type of anything can determine the type of an expression. This allows overloading on function results (only) which is generally not allowed in other languages. But it also can lead to expressions that are very complex to figure out the types:

```
   F1.all := F2 + F3;
```

All of these can be overloaded (as well as "+", which acts like a function for this purpose), and if there is a unique solution, the Ada compiler has to find it.

I often find bizarre (to me) special cases in the code of Janus/Ada, and I often try to get rid of them. And then most of the time, I have to put them back because they're needed for some wacky ACATS test (or old user bug report).

We had 14 man-years of development into our Ada 83 compiler; while I don't have an exact number for our Ada 95 compiler, it has to be double that. Some of it clearly came from the much smaller hosts of the era, but most of it is just Ada being big.

From: Luke A. Guest <laguest@archeia.com>
Date: Tue, 28 Feb 2017 21:29:33 +0000
Subject: Re: State of the compiler market
Newsgroups: comp.lang.ada

> [...]

If parallel blocks / loops aren't included, Ada will miss the boat *again* wrt massively parallel architectures. I can see Ada being used for GPGPU and shaders, e.g. SPIR-V generation using Ada.

If this happens, sorry to say but we may as well let it die and define a new language inspired by Ada for future work as Ada will never seem to move on and expand into different and well deserving areas.

[...]

Checking for Local Declarations of Variables of a Specific Type

From: Jacob Sparre Andersen <jacob@jacob-sparre.dk>
Date: Tue, 14 Mar 2017 14:24:04 +0100
Subject: [AdaControl] Limiting where objects of a specific type can be declared
Newsgroups: comp.lang.ada

On a project I'm working on, we have a type with some ugly C backing, which means that declaring local variables of the type results in memory leaks.

The solution to this is to remember to declare variables of this type as global variables in a package.

The fast way to check for this with this AdaControl rule:

```
check entities (local
      SQL.Statement_Type);
```

This looks good until AdaControl finds a clever programmer writing:

```
procedure Something (...) is
   Stmt : SQL.Statement_Type renames
      Global_Stmt;
begin
```

Since this declaration doesn't leak memory, I don't want to stop the programmer from using it.

The next interesting declaration AdaControl finds is a main program, which declares a global variable for later use:

```
procedure Main is
   Stmt : SQL.Statement_Type;
begin
   [...]
end Main;
```

In this case the solution may simply be to ignore this group of utility programs, when running AdaControl, but a modification to the rule would be nice, as that would reduce the amount of thinking involved in deciding which units to check and which not to check.

What do we do? (Besides phasing out the problematic library.)

From: Jean-Pierre Rosen <rosen@adalog.fr>
Date: Tue, 14 Mar 2017 14:47:49 +0100
Subject: Re: [AdaControl] Limiting where objects of a specific type can be declared
Newsgroups: comp.lang.ada

> check entities (local SQL.Statement_Type);

> [...]

> Stmt : SQL.Statement_Type renames Global_Stmt;

There is a subtility here: you are asking to check all local usages of type Statement_Type, which certainly applies here. What you /want/ is to check all local declarations of variables whose type is Statement_Type.

Alternatively, you can use:

```
check object_declarations (type, variable
      SQL.Statement_Type);
```

but this will give you all variables of type Statement_Type, including the global ones.

I take it as an enhancement suggestion to be able to specify a location for the above rule.

Of course, it is always possible to disable the line.

> [...]

Since there is nothing special to a main program in Ada, this is clearly a local usage... Disabling is probably the best thing to do.

UEFI Programming

From: George J <ivanov_george@list.ru>
Date: Fri, 24 Mar 2017 02:53:37 -0700
Subject: Ada UEFI programming
Newsgroups: comp.lang.ada

[...] has anyone built a UEFI (.efi) file using Ada, or does anyone have experience with connecting EDK II or UDK with Ada? [...]

From: Zhu Qun-Ying <zhu.qunying@gmail.com>
Date: Mon, 27 Mar 2017 10:49:21 -0700
Subject: Re: Ada UEFI programming
Newsgroups: comp.lang.ada

According to http://www.rodsbooks.com/refind/getting.html, in Linux, you may need the TianoCore EFI Development Kit 2 (EDK2) (https://sourceforge.net/projects/tianocore/) or GNU-EFI development tools (https://sourceforge.net/projects/gnu-efi/).

From: Luke A. Guest <laguest@archeia.com>
Date: Tue, 28 Mar 2017 09:09:08 -0700
Subject: Re: Ada UEFI programming
Newsgroups: comp.lang.ada

[...]

You have read these right?

http://wiki.osdev.org/PE

http://wiki.osdev.org/UEFI#GNU-EFI_and_GCC

You need an x86_64-pe targeted gcc cross compiler, you can then follow what I've done http://wiki.osdev.org/Ada_Bare_bones and https://github.com/Lucretia/bare_bones (this is more up to date than the osdev link as I now have a secondary stack) for a bare metal runtime to target this kind of application area.

You would use gcc to generate a binding from a uefi.h file (if there is one), or just bind as you normally would.

[See also “Ada in Coreboot”, AUJ 37-4, p. 191. —sparre]

Why Isn't System.Storage_Pools Pure?

From: Edward R. Fish <onewingedshark@gmail.com>
Date: Mon, 17 Apr 2017 23:31:18 -0700
Subject: Is there a reason System.Storage_Pools isn't Pure?
Newsgroups: comp.lang.ada

Looking at the specification for System.Storage_Pools [RM 13.11(5)] there doesn't seem to be anything that requires the Preelaborate pragma... is there any real reason that it wasn't made a Pure unit?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 18 Apr 2017 13:32:32 -0500
Subject: Re: Is there a reason System.Storage_Pools isn't Pure?
Newsgroups: comp.lang.ada

Originally, it was not Pure because it was a child of System, which was not Pure. So I can't find any discussion of the merits.

However, Pure packages are automatically Remote_Types packages (that is, values of the type can be transmitted between partitions). We'd never want that to be the case with a storage pool, so there doesn't seem to be any point in it being Pure.

Additionally, the Pure package rules assume that no storage pools can be specified for access types (because there aren't rules banning that at library-level, and there need to be such rules to prevent hidden state). That could be changed, I suppose, but given that a Pure storage pool could only be used for a local access type in a Pure package(something that mainly exists in ACATS tests), it would be close to useless (or get used for back-door state). Note that state has to be strictly prohibited as Pure packages are replicated when used in a distributed system (thus each partition would have different state, which wouldn't make sense).

IMHO, Pure packages are too restricted to be useful (and not restricted enough to be useful when synchronization is involved); it makes sense for individual subprograms but not for an entire package. So I recommend only trying to make packages Preelaborated. (That's especially true in Ada 2012, where limited I/O is possible.) [Distribution might change this thinking; I'm only considering stand-alone programs that don't use Annex E.]

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 19 Apr 2017 15:42:33 -0500
Subject: Re: Is there a reason System.Storage_Pools isn't Pure?
Newsgroups: comp.lang.ada

> A lot of use cases conflated into single pragma Pure: [...]

More like abandoned. Our current thinking is to essentially replace it by specifications of the Global aspect (which is stricter, so the result can be usefully used in parallelism applications) and finer grained (so individual subprograms can each have appropriate settings, no more "everything in the package has to be the same").

That would leave the only real purposes of Pure to be distribution and elaboration; the former uses don't want access types at all, and the latter can be handled just using Preelaborate does the job. (Ergo: most of what was done to Pure in Ada 2005 was a mistake, an attempt to fix the muddled mess resulting in a bigger muddled mess.)

Portable Memory Barrier?

From: Jere <jhb.chat@gmail.com>
Date: Fri, 5 May 2017 19:23:43 -0700
Subject: Portable memory barrier?
Newsgroups: comp.lang.ada

So I am implementing a circular buffer, which means I don't need to use a protected object between the Consumer and Producer (assuming 1 each ... Consumer adjusts one index, Producer adjusts another), but I have run into one area where CPU architecture could get me.

In one procedure (Producer's Put procedure), I run into the situation where I need to update the buffer, then the index (in that order) to ensure that the consumer doesn't see the index change before the data is actually there.

However, I know some CPUs can further reorder instructions. I know those specific processors usually have something like a memory barrier to force one write before the other, but I wanted to avoid doing inline assembly.

Are there any Ada language constructs that can help? I don't mind using protected objects, but I don't know if that guarantees me a memory barrier for CPU's that support that (or whatever means an architecture supplies). I don't, however, want to use a protected object between the Consumer and the Producer so that one doesn't block out the other.

If a language portable construct does not exist, does a GNAT one exist? I've been searching around tonight but haven't run into anything yet in the Ada front. For GNAT I think Pragma Enable_Atomic_Synchronization [1] might work (assuming I use the Atomic pragma on a correctly sized variable)?

[0] (C.6 16/3) http://www.ada-auth.org/standards/12rm/html/RM-C-6.html

[1] https://gcc.gnu.org/onlinedocs/gnat_rm/Pragma-Enable_005fAtomic_005fSynchronization.html

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Sat, 6 May 2017 10:47:57 +0200
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

Volatile means that something other than the program may change the value. It also means that all tasks of the program see the same sequence of updates. It does not mean that accesses are non-overlapping. If task A does

```
I := I + 1;
```

and task B does

```
if I > 0 then
```

with I marked Volatile, then A may update half of I, B read the partly updated value, and then A complete its update.

Atomic means Volatile plus all access are sequential; in the example, if A begins to update I, B won't read it until A's update is complete.

For your purposes, marking the (type of the) indices as Atomic should be all you need.

From: Jere <jhb.chat@gmail.com>
Date: Sat, 6 May 2017 07:17:10 -0700
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

[...] I'm more worried about situations like these:

Task A:

```
-- Do some checks to make sure buffer
--  not full
-- Not full, so put data
My_Array(Put_Index) := Some_Value;
if Put_Index = Buffer_Size then
   Put_Index := 0;
else
   Put_Index := Put_Index + 1;
end if;
```

Task B:

```
if Put_Index = Get_Index then -- empty
   return False;
end if;

Result_Value := My_Array(Get_Index);
-- Do other operations that update
Get_Index
```

Assuming the indexes are atomic and volatile, then I am not worried about partial writes to the indexes. I'm more worried about some architectures that allow for instruction reordering. What I don't want to happen is in Task A that the store to My_Array happens after the update to the Put_Index. If that happens, then there are cases where Task B verifies the buffer isn't empty and tries to access the array but Task A hasn't gotten around to updating the array because the CPU decided to rearrange the stores.

In C++, I would handle this with a fence, which is part of the standard:

```
My_Array[Put_Index] = Some_Value;

// force CPU to do the previous store before
// doing ones after the fence
std::atomic_thread_fence
          (std::memory_order_release);
if (Put_Index == Buffer_Size) {
   Put_Index = 0;
else
   Put_Index++;
}
```

With that I know that architectures that do reordering and supply a means to prevent that, the appropriate mechanism will be used to ensure Put_Index is updated after My_Array;

I'm not so much worried about atomic vs non atomic updates as I am sequential updates of different variables.

I'm looking for a language defined way in Ada (or in GNAT if not) that will implement a fence like mechanism on architectures that require it.

Side note: If I read it right, you seemed to indicate that volatile has nothing to do with sequence. I'm no Ada expert, so I wanted to ask your interpretation of section 16/3 of C.6. I'm used to C and C++ where if you have two volatile variables being updated like this:

```
Vol_Var_1 = 0;
Vol_Var_2 = 23;
```

That the C and C++ standard guarantees the "compiler" will not reorder those two statements. Note this says nothing about atomicness (which is only one reason among others why Volatile is not sufficient for threading). It also doesn't prevent the processor from reordering them. When I read that section in C.6 of the RM, the verbage made me think Ada also prevented the compiler from reordering the statements (again no relation to atomicness or CPU reordering).

Is this not the case?

From: Randy Brukardt <randy@rrsoftware.com>
Date: Tue, 9 May 2017 14:49:28 -0500
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

Atomic implies sequential access for atomic AND volatile objects (note that volatile does NOT imply that). See 9.10 if you dare. (And if you understand 9.10, please join the ARG, we need you. ;-)

In particular, Atomic implies that any needed memory fences are used. (Again, volatile alone does not make such a requirement.) The idea is that one uses atomic objects to protect larger volatile data structures.

Caveat: that's an untestable requirement, so you're totally at the mercy of your compiler vendor to get this right.

P.S. Yes, we've discussed this in the ARG. We believe the RM description has the right effect, but the issues and wording are both hairy, so we could be wrong.

From: Jere <jhb.chat@gmail.com>
Date: Tue, 9 May 2017 15:07:12 -0700
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

Thanks for all the replies. That is interesting. I had assumed Ada's definition was similar to C or C++, but indeed it is different. The C and C++ definitions of volatile guarantee that the compiler will not reorder load/store operations of volatile variables. From this statement it sounds like Ada makes no such guarantee with Volatile. NOTE: The C and C++ definitions hold no guarantee versus CPU instruction reordering like Ada does with pragma Atomic.

I got a test going today and tried it on two different GNAT implementations on two different processors. I found that the GPL version of GNAT (gcc 4.9.3) works as you state. Atomic prevents changes in instruction sequence. However, my older version (FSF GCC 4.1.2) does not. Atomic had no affect on sequence (from a CPU perspective at least). I am guessing that Atomic's definition was weaker back then.

From: Randy Brukardt <randy@rrsoftware.com>
Date: Wed, 10 May 2017 20:14:53 -0500
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...] I am guessing that Atomic's definition was weaker back then.

It was never "weaker", but the implications on compilers were not as well understood (especially the ones about fences). AI05-0117-1 was the original discussion on this topic, it was mostly worked on in 2010. (It originated earlier, but all work was in 2010 and 2011.) AI05-0275-1 clarified this further in late 2011 and early 2012.

So it's likely that compilers implemented prior to 2010 don't implement any fences for atomic objects, mostly likely because the need for them wasn't understood. (I personally had no idea...) More recent compilers are much more likely to do it right.

Also note that the 2010 AI was classified as an Amendment, so it only applies to Ada 2012. I'm not sure why that was (the definition in Ada 2012 is *weaker* than the one in earlier Ada versions), but it does mean that any Ada 95 or Ada 2005 implementations may not deal with this properly.

From: Robert I. Eachus <rieachus@comcast.net>
Date: Mon, 8 May 2017 21:02:44 -0700
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

[...]

In general do not use fences in Ada code, use protected operations. At a minimum the lock needed for a cleverly written protected object can be replaced by a read-modify-write (RMW) instruction, and the CPU should execute it in under a nanosecond. Code cannot be moved past RMW operations, and they are much faster than fences.

Why isn't a fence needed here? When it comes to shared variables, one thread only writes. But the example is flawed, in that we do have an access that needs to be protected. To make the buffer a circular buffer you need to add a test and adjustment. For simplicity? change:

```
FIFO (Write_Index + 1) := Element;
Write_Index := Write_Index + 1;
```

to:

```
Temp := Write_Index;
Temp := (if Temp = Max then 1
         else Temp + 1);
FIFO (Temp) := Element;
Write_Index := Temp;
```

Making Temp explicit is a belt and suspenders operation. The CPU will generate the same set of micro-ops, or at least the same set as once we make the buffer circular. But now it should be clear that the two writes cannot be reordered. Note that the writes to Temp can be replaced by writes to R1 and are reads for the x86 rules.

[...]

I could probably teach a graduate course on this, and not include anything not touched on here. And, yes I did teach a graduate course in compiler construction several times back before this revolution in how CPUs are put together. If I was teaching that course now, I would either limit it to the front end and a bit of optimization, or stretch it into two semesters. I won't say that this stuff makes quantum mechanics look easy, but they are at the same level of complexity. [...]

From: Dmitry A. Kazakov <mailbox@dmitry-kazakov.de>
Date: Wed, 10 May 2017 09:13:23 +0200
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> Is there a method besides Atomic? If I am implementing a generic FIFO (lock free) and the FIFO elements are complex data types that may not be able to be atomic, do I have any other options or are protected objects my only way out?

But you don't need elements atomic only index has to. If I understood Randy's explanation correctly atomic access to the buffer index will give you the barrier between index and element accesses which is all that is required.

From: Robert I. Eachus <rieachus@comcast.net>
Date: Wed, 10 May 2017 09:45:16 -0700
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

That is my reading as well, and should result in efficient implementations. Also, for portable code, there are times when Volatile is required. But you really need ancient (in Internet years) hardware for Volatile to have any effect at all. (At least in non-erroneous programs.) In general, Atomic should be implemented as a RMW (read-modify-write) instruction on x86, and will add fences on any hardware that requires them.

The other part, which occurs in this code, is that (user) Atomic variables should only be written by one task/thread. Hmm. Compiler back-ends should probably insert a RMW of a non-program variable to cause synchronization where an Atomic variable is only being read. Of course, this would only eliminate most erroneous executions. Best is for programmers to use protected objects which can correctly (in an interrupt protected way) use RMWs for synchronization.

From: Simon Wright <simon@pushface.org>
Date: Wed, 10 May 2017 18:28:09 +0100
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

I think Randy recently pointed out that Volatile should ideally apply only to the memory-mapped i/o case; for the external effects.

From: Jeffrey R. Carter <jrcarter@acm.org>
Date: Wed, 10 May 2017 18:38:48 +0200
Subject: Re: Portable memory barrier?
Newsgroups: comp.lang.ada

> [...]

If your indices are Atomic and your array has Volatile_Components, then I think things will work the way you want them to. The important reference here is ARM 1.1.3, which says

- Any read or update of an atomic or volatile object is part of the program's external interactions.
- An Ada compiler must produce a program with external interactions in the order and timing specified by Ada's semantics.

Ada's semantics for a sequence of statements say the statements are executed in order.

So, if you have in task A

```
A (I + 1) := Item;
I := I + 1;
```

Then from the outside an observer must see in order

1. Read I

2. Write A (I + 1)

3. Read I

4. Write I

```
If task B does
  if I > 0 then
```

then B's "Read I" will also be something the external observer can see. It might happen before 1., between 1. and 3., between 3. and 4., or after 4., but won't happen during any of 1, 3, or 4. That guarantees that B will always get a valid view of I.

Thanks to Holsti for pointing this out [through ARM C.6(20)]. My previous understanding of this was incorrect.

SPARK and Tasks

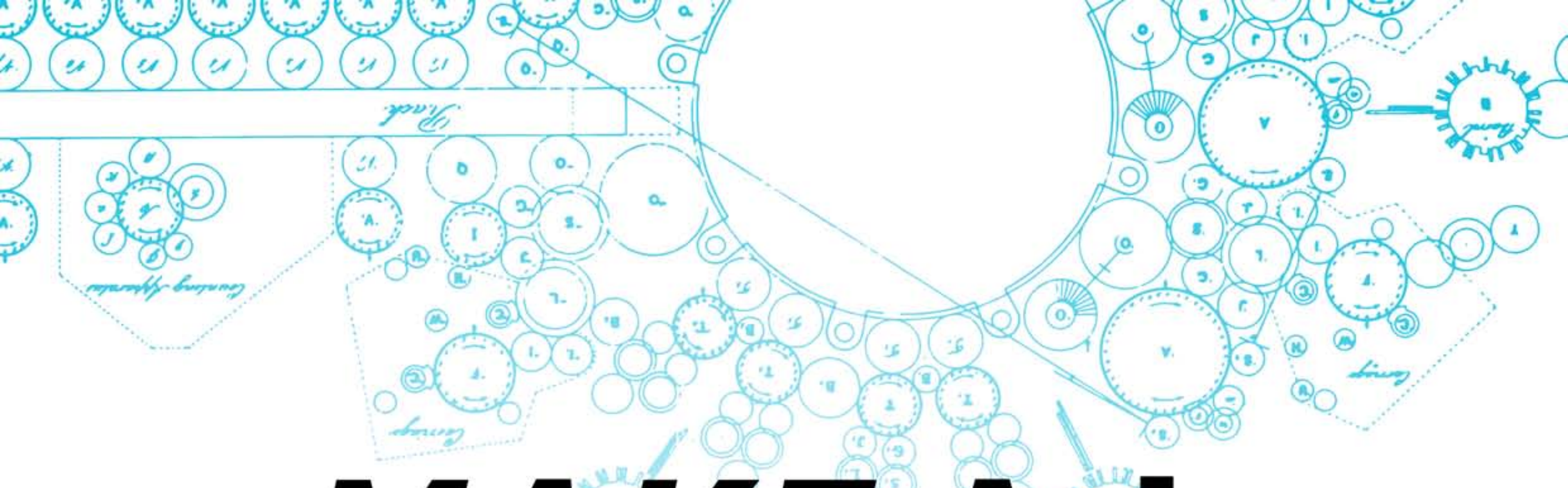
From: John Smith <yoursurrogategod@gmail.com>
Date: Tue, 16 May 2017 05:46:39 -0700
Subject: Does SPARK support tasks?
Newsgroups: comp.lang.ada

I remember reading that SPARK 2014 does not support tasks for multiprocessing. However, this was supposed to change in the future. My question, does SPARK support it now? Has anything changed?

From: Mark Lorenzen <mark.lorenzen@gmail.com>
Date: Tue, 16 May 2017 06:00:57 -0700
Subject: Re: Does SPARK support tasks?
Newsgroups: comp.lang.ada

> [...]

Yes: http://docs.adacore.com/spark2014-docs/html/lrm/tasks-and-synchronization.html



MAKE with Ada

PROGRAMMING COMPETITION

Make with Ada is an embedded software project competition sponsored by AdaCore. It is open to individuals and small teams using the Ada or SPARK languages to develop dependable, open, inventive and collaborative projects.

1st place	2nd place	3rd place
5000€	2000€	1000€
$5500	$2200	$1100

There will be a special student-only prize available to entrants who supply a student ID: an Assembled Printrbot Simple 3D printer!

May 15 - September 15, 2017

www.makewithada.org

Tools to get you there. Safely.
Ada and The GNAT Pro High-Integrity Family
www.adacore.com
AdaCore
The GNAT Pro Company

Conference Calendar

Dirk Craeynest

KU Leuven. Email: Dirk.Craeynest@cs.kuleuven.be

This is a list of European and large, worldwide events that may be of interest to the Ada community. Further information on items marked ♦ is available in the Forthcoming Events section of the Journal. Items in larger font denote events with specific Ada focus. Items marked with ☺ denote events with close relation to Ada.

The information in this section is extracted from the on-line *Conferences and events for the international Ada community* at: http://www.cs.kuleuven.be/~dirk/ada-belgium/events/list.html on the Ada-Belgium Web site. These pages contain full announcements, calls for papers, calls for participation, programs, URLs, etc. and are updated regularly.

2017

July 04-08 — 41st **Annual IEEE Conference on Computers, Software and Applications** (COMPSAC'2017), Turin, Italy. Event includes symposiums on Computer Education & Learning Technologies (CELT), Emerging Advances in Technology & Applications (EATA), IT in Practice (ITiP), Security, Privacy, & Trust (SEPT), Software Engineering Technology & Applications (SETA), etc.

July 05-07 — **International Conference on Software and Systems Process** (ICSSP'2017), Paris, France. Topics include: mining software/business process repositories (including code, bug trackers, etc.) to improve processes; empirical evidence of the effectiveness of agile/lean practices and approaches in software systems development and evolution; process issues in developing evolving software systems; processes for cutting-edge software technologies, including (but not limited to) multi-core technologies; empirical studies and experience reports, encompassing complete or parts of software and systems development lifecycle; etc.

July 17-21 — **Software Technologies: Applications and Foundations** (STAF'2017), Marburg, Germany. Successor of the TOOLS federated event. Topics include: practical and foundational advances in software technology.

> July 19-20 — 11th **International Conference on Tests And Proofs** (TAP'2017). Topics include: many aspects of verification technology, including foundational work, tool development, and empirical research; the connection between proofs (and other static techniques) and testing (and other dynamic techniques); verification and analysis techniques combining proofs and tests; program proving with the aid of testing techniques; deductive techniques to support testing: generating testing inputs and oracles, supporting coverage criteria, and so on; program analysis techniques combining static and dynamic analysis; testing and runtime analysis of formal specifications; model-based testing and verification; using model checking to generate test cases; testing of verification tools and environments; applications of testing and proving to new domains, such as security, configuration management, and language-based techniques; case studies, tool and framework descriptions, and experience reports about combining tests and proofs; etc.

July 22-28 — 29th **International Conference on Computer-Aided Verification** (CAV'2017), Heidelberg, Germany. Topics include: theory and practice of computer-aided formal analysis and synthesis methods for hardware and software systems, algorithms and tools for verifying models and implementations, specifications and correctness criteria for programs and systems, deductive verification using proof assistants, program analysis and software verification, verification methods for parallel and concurrent systems, testing and run-time analysis based on verification technology, applications and case studies in verification and synthesis, verification in industrial practice, formal models and methods for security, etc.

July 26-28 — IEEE **International Conference on Software Quality, Reliability and Security** (QRS'2017), Prague, Czech Republic. Since 2015, merger of SERE (International Conference on Software Security and Reliability) and QSIC (International Conference on Quality Software). Topics include: reliability,

	security, availability, and safety of software systems; software testing, verification and validation; program debugging and comprehension; fault tolerance for software reliability improvement; modeling, prediction, simulation, and evaluation; metrics, measurements, and analysis; software vulnerabilities; formal methods; benchmark, tools, and empirical studies; etc. Includes half-day tutorial on "The Correctness-by-Construction Approach to Programming" by Ina Schaefer, Technische Universität Braunschweig, Germany, and Bruce W. Watson, Stellenbosch University, South Africa.
☺ August 16-18	23th **IEEE International Conference on Embedded and Real-Time Computing Systems and Applications** (RTCSA'2017), Hsinchu, Taiwan. Topics include: multi-core embedded systems; operating systems and scheduling; embedded software and compilers; fault tolerance and security; embedded systems and design methods for cyber-physical systems; real-time operating systems; real-time scheduling; timing analysis; programming languages and run-time systems; middleware systems; design and analysis tools; applications and case studies of IoT and CPS; cyber-physical co-design; etc.
Aug 30 - Sep 01	43rd **Euromicro Conference on Software Engineering and Advanced Applications** (SEAA'2017), Vienna, Austria. Topics include: information technology for software-intensive systems; main tracks on Embedded Software Engineering (ESE), Model-based Development, Components and Services (MOCS), Software Process and Product Improvement (SPPI), Software Product Lines and Software Ecosystems (SPLSeco), etc.; special sessions on Teaching, Education and Training for Dependable Embedded and Cyber-Physical Systems (TET-DEC), Cyber-Physical Systems (CPS), Software Engineering and Technical Debt (SEaTeD), etc.
September 03-06	**Federated Conference on Computer Science and Information Systems** (FedCSIS'2017), Prague, Czech Republic. Event includes: 2nd International Workshop on Language Technologies and Applications (LTA), 6th Workshop on Advances in Programming Languages (WAPL), 4th International Workshop on Cyber-Physical Systems (IWCPS), 37th IEEE Software Engineering Workshop (SEW), etc.
September 04-08	10th **Joint European Meeting of the Software Engineering Conference** and the ACM SIGSOFT **Symposium on the Foundations of Software Engineering** (ESEC/FSE'2017), Paderborn, Germany. Topics include: API usage and design; debugging, fault localization, and repair; dependability, safety, and reliability; development environments and tools; empirical studies; formal methods and verification; model-driven software engineering; parallel, distributed, and concurrent systems; performance and scalability; program analysis; refactoring, reengineering, and migration; security and privacy; software architecture; software economics; software evolution and maintenance; software processes and project organization; software testing; variability management and software product lines; etc. Deadline for early registration: July 19, 2017.
September 04-08	15th **International Conference on Software Engineering and Formal Methods** (SEFM'2017), Trento, Italy. Topics include: real-time, hybrid and embedded systems; verification and validation; light-weight and scalable formal methods; software evolution, maintenance and reuse; application and technology transfer; case studies, best practices and experience reports; tool integration; education; safety-critical, fault-tolerant and secure systems; software certification; programming languages; abstraction and refinement; etc.
☺ September 12-15	**International Conference on Parallel Computing** 2017 (ParCo'2017), Bologna, Italy. Topics include: all aspects of parallel computing, including applications, hardware and software technologies as well as languages and development environments; new concepts for parallel computing architectures for all levels of parallelism (multicore and manycore systems, accelerators, including GPUs, FPGAs, ...); software engineering methodologies, methods and tools for developing and maintaining parallel software; parallel programming languages, compilers, libraries and environments; testing and debugging techniques and tools; best practices of parallel computing on multicore, manycore and stream processors; the application of parallel computing to solve all types of business, industrial, scientific and engineering problems using high-performance computing technologies; etc. Deadline for submissions: July 31, 2017 (full papers).
September 13-15	11th **International Symposium on Theoretical Aspects of Software Engineering** (TASE'2017), Nice, France. Topics include: theoretical aspects of software engineering, such as abstract interpretation, component-based systems, cyber-physical systems, distributed and concurrent systems, embedded and real-time systems, formal verification and program semantics, integration of formal methods, language design, model checking and theorem proving, object-oriented systems, run-time verification and

monitoring, software architecture, software testing and quality assurance, software security and reliability, static analysis of programs, type systems and behavioural typing, tools exploiting theoretical results, etc.

September 13-16 17th **International Conference on Runtime Verification** (RV'2017), Seattle, Washington, USA. Topics include: monitoring and analysis of the runtime behaviour of software and hardware systems. Application areas include cyber-physical systems, safety/mission-critical systems, enterprise and systems software, autonomous and reactive control systems, health management and diagnosis systems, and system security and privacy.

September 20-22 13th **International Conference on integrated Formal Methods** (iFM'2017), Turin, Italy. Topics include: hybrid approaches to formal modeling and analysis; i.e., the combination of (formal and semi-formal) methods for system development, regarding both modeling and analysis, and covering all aspects from language design through verification and analysis techniques to tools and their integration into software engineering practice.

October 02-06 17th **International Conference on Formal Methods in Computer-Aided Design** (FMCAD'2017), Vienna, Austria. Topics include: theory and applications of formal methods in hardware and system verification; synthesis and compilation for computer system descriptions, modeling, specification, and implementation languages, model-based design, correct-by-construction methods, ...; experience with the application of formal and semi-formal methods to industrial-scale designs; tools that represent formal verification enablement, new features, or a substantial improvement in the automation of formal methods; etc.

October 15-20 **Embedded Systems Week** 2017 (ESWEEK'2017), Seoul, South Korea. Includes CASES'2017 (International Conference on Compilers, Architecture, and Synthesis for Embedded Systems), CODES+ISSS'2017 (International Conference on Hardware/Software Codesign and System Synthesis), EMSOFT'2017 (International Conference on Embedded Software).

October 15-20 ACM SIGBED International Conference on Embedded Software (EMSOFT'2017). Part of ESWEEK, EMSOFT brings together researchers and developers from academia, industry, and government to advance the science, engineering, and technology of embedded software development. EMSOFT is a venue for cutting-edge research in the design and analysis of software that interacts with physical processes, with a long-standing tradition for results on cyber-physical systems, which compose computation, networking, and physical dynamics.

October 15-20 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'2017). Part of ESWEEK, CASES is a forum where researchers, developers and practitioners exchange information on the latest advances in compiler and architectures for high-performance, low-power embedded systems. The conference has a long tradition of showcasing leading edge research in embedded processor, memory, interconnect, storage architectures and related compiler techniques targeting performance, power, predictability, security, reliability issues for both traditional and emerging application domains. In addition, we invite innovative papers that address design, synthesis, and optimization challenges in heterogeneous and accelerator-rich architectures.

☺ October 22-27 ACM **Conference on Systems, Programming, Languages, and Applications: Software for Humanity** (SPLASH'2017), Vancouver, Canada. Topics include: all aspects of software construction, at the intersection of programming, languages, systems, and software engineering. Deadline for submissions: July 2, 2017 (GPCE - Generative Programming: Concepts & Experiences, Scala Symposium abstracts), July 9, 2017 (Scala Symposium papers), July 15, 2017 (posters), July 17, 2017 (Student Research Competition), August 1 - September 7, 2017 (workshop contributions). Deadline for early registration: September 30, 2017.

October 23-27 14th **International Colloquium on Theoretical Aspects of Computing** (ICTAC'2017), Hanoi, Vietnam. Topics include: principles and semantics of programming languages; models of concurrency, security, and mobility; real-time, embedded, hybrid and cyber-physical systems; program static and dynamic analysis and verification; software specification, refinement, verification and testing; model checking; case studies, theories, tools and experiments of verified systems; etc.

November 07-09	30th IEEE **Conference on Software Engineering Education and Training** (CSEET'2017), Savannah, USA. Topics include: curriculum development; empirical studies; personal or institutional experience; software assurance, quality, and reliability; methodological aspects of software engineering education; open source in education; cooperation between industry and academia; etc.
☺ December 05-08	38th **IEEE Real-Time Systems Symposium** (RTSS'2017), Paris, France. Topics include: all aspects of real-time systems theory, design, analysis, implementation, evaluation, and experiences. Deadline for submissions: September 15, 2017 (workshop papers), September 22, 2017 (demo abstracts).
December 10	Birthday of Lady Ada Lovelace, born in 1815. Happy Programmers' Day!

2018

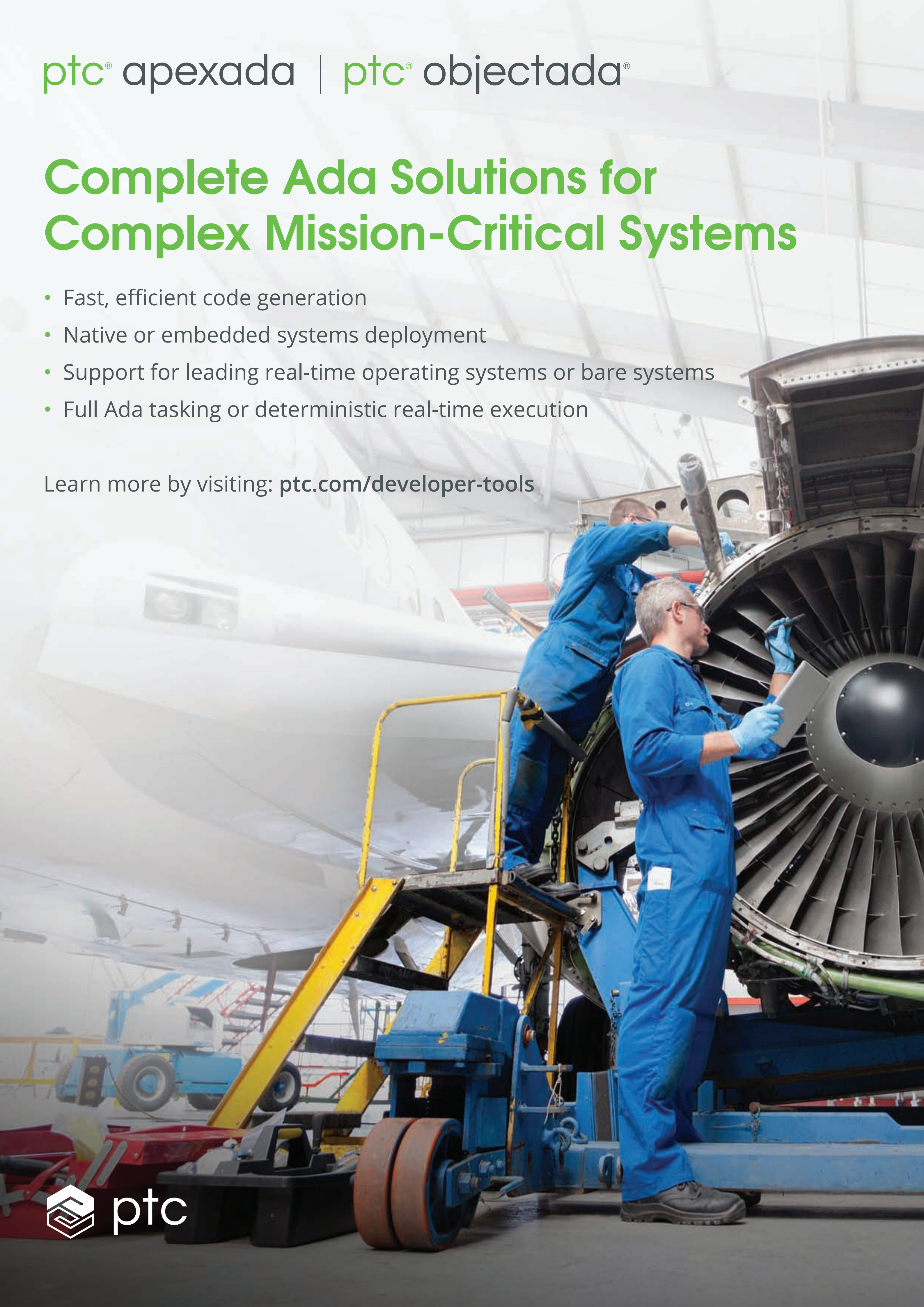
♦ April 18-20	19th **International Real-Time Ada Workshop** (IRTAW'2018), Benicàssim, Spain.
♦ June 18-22	23rd **International Conference on Reliable Software Technologies - Ada-Europe'2018**. Lisbon, Portugal. Sponsored by Ada-Europe. Deadline for submissions: January 22, 2018 (regular papers, industrial presentations, tutorials, workshops).

ptc® apexada | ptc® objectada®

Complete Ada Solutions for Complex Mission-Critical Systems

- Fast, efficient code generation
- Native or embedded systems deployment
- Support for leading real-time operating systems or bare systems
- Full Ada tasking or deterministic real-time execution

Learn more by visiting: **ptc.com/developer-tools**



ptc

19th International Real-Time Ada Workshop – IRTAW 2018

Hotel Voramar, Benicàssim, Spain
18-20 April 2018
http://www.ada-europe.org/irtaw2018

Preliminary Call for Papers

The International Real-Time Ada Workshop series has provided a forum for identifying issues with real-time system support in Ada and for exploring possible approaches and solutions, and has attracted participation from key members of the research, user, and implementer communities worldwide. Recent International Real-Time Ada Workshop meetings contributed to the Ada 2005/Ada 2012 standards, especially with respect to the tasking features, the real-time and high-integrity systems annexes, and the standardization of the Ravenscar Tasking Profile.

In keeping with this tradition, the goals of the 19th edition of IRTAW will be to:

- Review Ada 2012 Issues vis-a-vis real-time systems;
- Examine experiences in using Ada 2012 for real-time systems and applications;
- Implementation approaches for Ada 2012 real-time features;
- Consider developing other real-time Ada profiles in addition to the Ravenscar profile;
- Implications to Ada with multiprocessors in development of real-time systems;
- Paradigms for using Ada for real-time distributed systems, with special emphasis on robustness as well as hard, flexible and application-defined scheduling;
- Analysis of specific patterns and libraries for real-time systems development in Ada;
- Ada in context of the certification of safety-critical and/or security-critical real-time systems;
- Examine the Real-Time Specification for Java and other languages for real-time systems development, their current implementations and their interoperability with Ada in embedded real-time systems;
- Industrial experience with Ada and the Ravenscar Profile in real-time projects;
- Consider the language vulnerabilities of the Ravenscar and full language definitions;
- Consider testing for compliance with the Real-Time Annex.

Participation at the 19th IRTAW is by invitation following the submission of a position paper addressing one or more of the above topics or related real-time Ada issues. Alternatively, anyone wishing to receive an invitation, but for one reason or another is unable to produce a position paper, may send in a one-page position statement indicating their interests. Priority will be given to submitted papers.

Submission Requirements

Position papers should not exceed ten pages in typical IEEE conference layout, excluding code inserts. All accepted papers will appear, in their final form, in the Workshop Proceedings, which will be published as a special issue of Ada Letters (ACM Press). Selected papers will also appear in the Ada User Journal.

Authors with a relevant paper submitted to the 23rd International Conference on Reliable Software Technologies – Ada-Europe 2018 (deadline 24 January, 2018) may offer an extended abstract of the same material to IRTAW. Please submit position papers, in PDF format, to the Program Chair by e-mail: *stephen.michell@maurya.on.ca*

Important Dates

Paper Submission: **4 February, 2018**
Notification of Acceptance: 23 February, 2018
Confirmation of Attendance: 9 March, 2018
Final Paper Due: 30 March, 2018
Workshop: April 18-20, 2018

Program Chair
Brad Moore, *General Dynamics Mission Systems, Canada*

Workshop Chair
Jorge Real, *Universitat Politècnica de València, Spain*

Ada-Europe 2018

23rd International Conference on Reliable Software Technologies
18-22 June 2018, Lisbon, Portugal

Conference Chair
Nuno Neves
LASIGE/U. Lisboa

Program Chair
António Casimiro
LASIGE/U. Lisboa

Special Session Chair
Marcus Völp
University of Luxembourg

Tutorial and Workshop Chair
David Pereira
CISTER/ISEP

Industrial Co-Chairs
Marco Panunzio
Thales Alenia Space
José Rufino
LASIGE/U. Lisboa

Publication Chair
Pedro Ferreira
LASIGE/U. Lisboa

Exhibition Co-Chairs
José Neves
GMV
Ahlan Marriott
White Elephant GmbH

Publicity Chair
Dirk Craeynest
Ada-Belgium & KU Leuven

Local Secretariat
Madalena Almeida
Viagens Abreu S.A.

General Information

The **23rd International Conference on Reliable Software Technologies – Ada-Europe 2018** will take place in Lisbon, Portugal. Following its traditional style, the conference will span a full week, including a three-day technical program and vendor exhibition from Tuesday to Thursday, along with parallel tutorials and workshops on Monday and Friday.

Schedule

22 January 2018	Submission of papers, industrial presentation, tutorial and workshop proposals
9 March 2018	Notification of acceptance to all authors
24 March 2018	Camera-ready version of papers required
8 May 2018	Industrial presentations, tutorial and workshop material required

Topics

The conference is a leading international forum for providers, practitioners and researchers in reliable software technologies. The conference presentations will illustrate current work in the theory and practice of the design, development and maintenance of long-lived, high-quality software systems for a challenging variety of application domains. The program will allow ample time for keynotes, Q&A sessions and discussions, and social events. Participants include practitioners and researchers representing industry, academia and government organizations active in the promotion and development of reliable software technologies.

This edition of Ada-Europe features a focused **Special Session on Security in Safety-Critical Systems**. Safety-critical systems, on which we daily bet our lives, have become increasingly more complex, networked and distributed. In combination with the growing professionalism of adversarial teams, this demands for not only safe systems, but systems that also remain safe while under attacks. This session seeks (but is not limited to) contributions aiming at bridging the safety and security gap in cyber-physical and other safety-critical systems. The topics of interest include: Software and System Aspects of Secure and Dependable CPS, Vulnerabilities and Protective Measures for Safety-Critical System Infrastructures, and Fault and Intrusion Tolerance and Long-Term Unattended Operation for Safety-Critical Systems. For further information, please contact the Special Session Chair directly.

The topics of interest for the **general track of the conference** include, but are not limited to (full list on the website): Real-Time and Embedded Systems, Mixed-Criticality Systems, Theory and Practice of High-Integrity Systems, Software Architectures, Methods and Techniques for Software Development and Maintenance, Formal Methods, Ada Language and Technologies, Software Quality, Mainstream and Emerging Applications, Experience Reports in Reliable System Development, Experiences with Ada.

http://www.ada-europe.org/conference2018

Call for Regular and Special Session Papers

Authors of papers that are to undergo peer review for acceptance are invited to submit original contributions by 22 January 2018. Paper submissions shall be 14 LNCS-style pages in length. Authors for both the general track and the special session shall submit their work via EasyChair at *https://easychair.org/conferences/?conf=adaeurope2018*. The format for submission is solely PDF.

The International Conference on Reliable Software Technologies is listed in DBLP, SCOPUS and Web of Science Conference Proceedings Citation index, Google Scholar, and Microsoft Academic Search, among others.

Proceedings

The conference proceedings will be published in the Lecture Notes in Computer Science (LNCS) series by Springer, and will be available at the conference. The authors of accepted regular and special session papers shall prepare camera-ready submissions in full conformance with the LNCS style, strictly by 24 March 2018. For format and style guidelines, authors should refer to *http://www.springer.de/comp/lncs/authors.html*. Failure to comply and to register for the conference by that date will prevent the paper from appearing in the proceedings.

Call for Industrial Presentations

The conference seeks industrial presentations that deliver value and insight but may not fit the selection process for regular papers. Authors are invited to submit a presentation outline of at least 1 page in length by 22 January 2018. Submissions shall be made via EasyChair following the link *https://easychair.org/conferences/?conf=adaeurope2018*. The format for submission is solely PDF.

The Industrial Committee will review the submissions and make the selection. The authors of selected presentations shall prepare a final short abstract and submit it by 8 May 2018, aiming at a 20-minute talk. The authors of accepted presentations will be invited to submit corresponding articles for publication in the *Ada User* Journal (*http://www.ada-europe.org/auj/*), which will host the proceedings of the Industrial Program of the Conference. For any further information, please contact the Industrial Co-chairs directly.

Awards

Ada-Europe will offer honorary awards for the best regular paper and the best presentation.

Call for Tutorials

Tutorials should address subjects that fall within the scope of the conference and may be proposed as either half- or full-day events. Proposals should include a title, an abstract, a description of the topic, a detailed outline of the presentation, a description of the presenter's lecturing expertise in general and with the proposed topic in particular, the proposed duration (half day or full day), the intended level of the tutorial (introductory, intermediate, or advanced), the recommended audience experience and background, and a statement of the reasons for attending. Proposals should be submitted by e-mail to the Tutorial Chair. The authors of accepted full-day tutorials will receive a complimentary conference registration as well as a fee for every paying participant in excess of 5; for half-day tutorials, these benefits will be accordingly halved. The *Ada User Journal* will offer space for the publication of summaries of the accepted tutorials.

Call for Workshops

Workshops on themes that fall within the conference scope may be proposed. Proposals may be submitted for half- or full-day events, to be scheduled at either end of the conference week. Workshop proposals should be submitted to the Tutorial and Workshop Chair. The workshop organizer shall also commit to preparing proceedings for timely publication in the *Ada User Journal*.

Call for Exhibitors

The commercial exhibition will span the three days of the main conference. Vendors and providers of software products and services should contact the Exhibition Co-chairs for information and for allowing suitable planning of the exhibition space and time.

Grants for Reduced Student Fees

A limited number of sponsored grants for reduced fees is expected to be available for students who would like to attend the conference or tutorials. Contact the Conference Chair for details.

Venue

The conference venue is the VIP Executive Art's Hotel (left image), in the Parque das Nações area (central images), in Lisbon, Portugal. June is full of events in Lisbon, including the festivities in honour of St. António, with music, grilled sardines and popular parties taking place in the old neighbourhoods of Alfama and Bairro Alto, downtown (image on the right). Plan in advance! It is absolutely worth it!

Proven Test Solutions for Reliable Embedded Software

"VectorCAST is unique in that it provides us with the ability to increase the reliability and quality of our flight software."

-- Honeywell

VECTOR software

vectorcast.com

VectorCAST is a TÜV SÜD Certified Software Tool for Safety Related Development

Vector Software, Inc.
Golden Cross House | 8 Duncannon Street | London WC2N 4JF UK | +44 203 603 0120 | sales@vectorcast.com

Community Input for the Maintenance and Revision of the Ada Programming Language

ISO/IEC JTC 1/SC 22/WG 9 (WG 9) [1] is responsible for the maintenance and revision of the Ada Programming Language and associated standards and technical reports. As part of the language maintenance activity, WG 9 has established a group of Ada experts as the Ada Rapporteur Group (ARG). The ARG receives input from the Ada community at large to consider for inclusion in revision to the Ada programming language standard. WG 9 has produced a number of revisions to the language in accordance with ISO policy and to address the evolution of technology (Ada 87, Ada 95, Ada 2005, Ada 2012).

Presently, the ARG is beginning work on a revision to Ada 2012 so that ISO standardization of the new version can be completed by 2020. This is a relatively short horizon, but it ensures that the language continues to evolve, and at the same time requires that the changes to the language are evolutionary and do not present an undue implementation burden on existing compilers and users.

WG 9 welcomes suggestions for language enhancements at any time (should be sent to ada-comment@ada-auth.org) [2]. For enhancement requests (as opposed to identification of errors in the Ada 2012 Standard), it is very important to describe the programming problem and why the Ada 2012 solution is complex, expensive, or impossible. A detailed description of a specific enhancement is welcome but not necessarily required.

The goal of the ARG is to solve as many programming problems as possible with new/enhanced Ada features that fit into the existing Ada framework. Thus the ARG will be looking at the language as a whole, which may suggest alternative solutions to the problem. For a more detailed discussion, the guidelines [3] presented for the Ada 2005 revision can be used as the ARG requirements are little changed.

In order for suggestions to be considered for inclusion in the next revision of Ada, they must be received by 15 January 2018. Suggestions received after that date will only be considered if they relate to topics already under development; others will be considered only for future versions of Ada.

WG 9 has directed the ARG to focus its work on three areas of particular interest to the Ada community: additional facilities for multi-core and multithreaded programming, improved facilities for program correctness, and enhanced container libraries. There are numerous proposed enhancements in these and other areas [4]. Some of these proposals originated with members of the ARG, and others from members of the community at large.

The WG 9 encourages members of the Ada community at large to use the guidelines outlined above to provide input to WG 9 and the ARG for needed revisions and upgrades to the Ada programming language.

[1] http://www.open-std.org/jtc1/sc22/wg9/

[2] As described in the RM Introduction (http://www.ada-auth.org/standards/rm12_w_tc1/html/RM-0-3.html#p58)

[3] http://archive.adaic.com/news/pressrelease/call4apis.html

[4] The interested reader can find the current state of these at http://www.ada-auth.org/AI12-SUMMARY.HTML.

Panel Session Summary: The Future of Safety-Minded Languages

Erhard Ploedereder, Session Chair
Jorge Garrido, Rapporteur

The session combined the presentation of two papers accepted for the conference, an invited position statement and an invited talk. Throughout, the speakers formed a panel that discussed the presented ideas in more detail.

1 Paper Presentation: A New Ravenscar-Based Profile

Presenter and panelist: Pat Rogers

Pat Rogers presented a Ravenscar-inspired, yet less constrained profile supported by a run-time implementation developed by AdaCore, aimed at real-time and embedded applications not requiring rigorous certification and safety analysis but still in need of schedulability analysis. While Ravenscar strives to maximize simplicity, analysability, and efficiency for the production of predictable systems, its heavy restrictions imply a loss in expressiveness bothersome in many projects.

The new profile relaxes some of the rules related with tasks synchronization by allowing multiple protected entries, multiple queued callers per entry, and more expressive entry barriers (pure_barriers instead of simple_barriers). The use of Ada.Calendar is also allowed, e.g., for logging purposes, although not recommended for real-time purposes.

The developed run-time system shows increased overhead between 13% and 58% for the ESA Ravenscar Benchmark. This was said to be acceptable since the base compared to was highly efficient. A small blocking time is also added. Both drawbacks were identified as acceptable and it was pointed out that applications may achieve higher efficiencies due to the increased expressiveness and less complex constructions required to overcome the limitations on the number of entries and entry queues.

The discussion made it clear that the measured times were exclusively the distributed overhead in the absence of usage of newly allowed features, which are not exercised by the ESA benchmark. The potential need for a new guide for the use of this Ravenscar-extended profile was identified.

2 Paper Presentation: OpenMP Tasking Model for Ada: Safety and Correctness

Presenter and panelist: Eduardo Quinones

The paper proposes the adoption of OpenMP in Ada systems, aiming to increase the current performance, programmability and portability of parallel systems by providing means to express potential parallel sequences of code and data dependencies.

The proposers argue that, while defining a new parallel model might provide higher adaptability to Ada application requirements and better integration in the language, the benefits of OpenMP outweigh the potential of even better but specialized support. These benefits are the maturity of the OpenMP model, its widespread use, easily achieved portability, and the reduced effort required to implement the paradigm on top of a parallel run-time system. On the downside, OpenMP does not provide safety properties. This is being studied and planned to be improved in the Upscale initiative (`www.upscale.com`).

The proposal includes 3 main tasks:

- Extend OpenMP to support Ada: this is viewed by the proposers as an opportunity for the Ada community to influence the evolution of OpenMP.
- Extend Ada to support OpenMP: mainly add pragmas to indicate the parallel sections.
- Provide compiler support of OpenMP.

As the main challenges for the evolution of OpenMP, the discussion identified the implied non-determinisms in the execution, race conditions, potential deadlocks and the lack of fault tolerance mechanisms.

The panel discussion dealt with the different domains or levels of concurrency targeted by OpenMP, Ada tasking, Ada tasklets and other approaches. The panel concluded that parallelism is a highly desirable property, but that the ways to realize it needed further studies and experiments.

3 Position Statement: Enhanceable Tooling for Every-Day Programmers

Presenter and panelist: Oliver Schneider, KIT, Germany

The presentation focused on different approaches taken by relatively new scripting languages and by older, security-minded languages, such as Ada, in providing a development infrastructure, such as tool development and distribution. While new approaches tend to allow a fast integration of tools and libraries by means of packaging services, usually integrated into the language, mature languages keep approaches in which it is the responsibility of the developer to locate, acquire and integrate compilers, libraries, IDEs, analysis tools, etc.

The Ada approach is recognized to pose difficulties to the developers along some of the mentioned points, particularly the availability of a rich toolset and the easy acquiry and integration of tools into a local infrastructure. On the other hand, regulated means of tool integration have benefits for building the target Ada systems in terms of safety, reliability and control over the development process.

Following this line, the differences in the environments of scripting languages and of Ada were pointed out. While scripting languages and their environments are mainly community developed in an open-source manner, Ada is a standardized language with different target systems and different vendors with business models adapted to the current distribution approach.

4 Invited Presentation: Safe, Contract-Based Parallel Programming

Presenter and panelist: Tucker Taft, AdaCore, USA

The presentation brought back the differences between concurrency and parallelism already present in the second presentation. Concurrency is defined as a means to allow the developer to reflect the problem domain dynamics in writing the program such that the potential for parallel execution is not unintentionally hampered, while parallelism is defined as a means to allow the developer to achieve net speedups by using divide and conquer approaches on parallel hardware.

Tucker Taft presented existing approaches in different languages to support parallelism or concurrency either by means of libraries (Rust, Java fork/join, TBB), or by pragmas (OpenMP), or as part of the language syntax itself (Click+, Go, Ada 2012/202X, ParaSail).

Different language approaches to provide safety to parallel constructs were suggested. Support for expressing the needs for safety checks in the form of checkable claims (contracts) was considered the best approach, leaving it open to which extent the contracts were checked at compile- or run-time.

During the presentation and subsequent discussion the existence of a so-called "sequential mindset" was addressed, by which developers tend to address problems with sequential solutions as a result of their experience using languages with limited or non-existent parallel expressiveness. While the panel agreed with the existence of this tendency to provide primarily sequential solutions, it did not come to agreement on whether the concurrency support should be an integral part of the language or provided through standardized libraries.

Finally, the panel discussion addressed the way concurrency or parallelism should be supported, with divided points of view on the need of fully structured concurrency as in Ada 202X (bottom-up approach) in contrast to other solutions with top-down approaches where dependencies are claimed to be easier to specify and address, e.g., the issues surrounding the use of shared data.

Protocol Monitors: a Control-System Structuring Concept

Bo I. Sandén

Colorado Technical University, 4435 N. Chestnut, Colorado Springs, CO 80907, USA; Tel: +1 719-531-9045; email: bsanden@acm.org

1 Introduction

A protocol monitor (PM) as presented here is an Ada package that forms a shell around one or more protected objects (POs). In a PM, a programmer can encapsulate operations that are unsuited to be inside a PO together with the POs on which they rely. A basic example is a package that encapsulates a PO working as a semaphore and some lengthy computation that a calling task executes while holding a lock. While that particular PM enforces a simple acquire-release protocol, a PM can also enforce protocols with more states as well as simultaneous protocols. The paper aims to establish the term "protocol monitor" (PM) for this kind of programming device and make it a reliable tool for a programmer to reach for.

2 Hoare monitors

Hoare [3] presented the concept of a *monitor* as a software object that defines all the possible operations on a shared data structure [1]. It is designed so that only one calling task at a time can be executing any of those operations: A calling task stalls if another task is already executing an operation on the same object. This paper's title pays homage to Hoare [3]. "Control system" refers to software that manages a physical plant such as a flexible manufacturing system, and must, for example, give automated vehicles exclusive access to a stand. The access in such a case can be of considerably longer duration than an operation on a shared data structure.

A Hoare monitor uses a hardware mutex semaphore or the equivalent to enforce the mutual exclusion, but the programmer does not deal with this directly. Instead, the compiler inserts the necessary mutex operations. Both as a matter of defensive programming and for programmer convenience, this is a great advantage over manipulating "naked" semaphores directly. The time taken by each call must be similar for all operations and predictably short. We may want each operation to be nearly instantaneous so that we can ignore the time a task has to wait for exclusive access. A Hoare monitor can also block tasks calling an operation until a certain condition holds, which an Ada programmer accomplishes with a protected entry.

Like similar devices in various languages, Ada's protected objects (POs) such as the one below are "monitor like". They go beyond Hoare's semaphore by letting multiple tasks call *functions* on the PO simultaneously as long as no other task is executing a procedure or entry at the time.

```
protected Monitor is
    procedure … ;
    function … ;
    entry … ;
end Monitor;
```

Java and C# provide somewhat similar support. The Monitor Object pattern [8] lets programmers in other languages create their own monitors based on available synchronization primitives.

Hoare [3] also introduced a monitor that itself functions as a semaphore for some resource as with this *semaphore PO*:

```
protected Semaphore is
    entry Acquire;
    procedure Release;
  private
    Free : Boolean := True;
end Semaphore;
```

We can use a semaphore PO when, for any reason, the logic where the resource is used cannot or should not be encapsulated in a protected operation. Hoare [3] also introduced monitor *types* to be instantiated for each of a number of like resources, and also a single monitor that guards a set of equivalent resources and gives each calling task exclusive access to one of them. The stands in the manufacturing system may be an example.

We may want to use a semaphore PO *type* in problems such as the classic dining philosophers [4], who are seated around a table, each behind a bowl of spaghetti. There is a fork to the right of each bowl, but because they all use a two-fork eating technique, each philosopher must also borrow the left-hand neighbor's fork. Thus each fork is a shared resource. Here is the specification of such a protected type:

```
protected type Semaphore_Type is
    entry Acquire;
    procedure Release;
  private
    Free : Boolean := True;
end Semaphore_Type;
```

The task calling Acquire returns to the task body (or some operation called from the body) in a state where it has exclusive access to the resource. It is up to the programmer to ensure that the task always ultimately meets its obligation to call Release.

3 Basic protocol monitors

As a matter of information hiding and defensive programming, a software engineer may well encapsulate a semaphore PO in a package together with the logic for operating on a shared resource under exclusive access. We shall here refer to such a package as a *protocol monitor* (PM). The name distinguishes it from a Hoare monitor while hinting at the similarity. A semaphore PO plays the same role within the protocol monitor as does the mutex in a Hoare monitor on a lower level of abstraction. A PM enforces adherence to a protocol. At its simplest, the protocol is acquire-release or some variation thereof. We shall call that kind a *basic* PM.

3.1 Protocol monitor for dining philosophers

The dining philosophers' problem is interesting because the situation where they all sit down to eat at once is rotationally symmetric and prone to deadlock. For example, all philosophers could simultaneously grab the forks on their right and stubbornly hold on to them. They can prevent such deadlock in various ways. For example, they can number the forks 1 .. *n* starting at any fork and continuing around the table, and agree to abide by the convention that none of them may hold on to one fork while waiting for another one with a lower number. That way, the philosopher flanked by forks *n* and 1 is barred from picking fork *n* first and must break the symmetry.

We focus here on the part where the philosophers obtain and later release their two forks. We use a Philosopher task type and the Semaphore_Type introduced earlier. The protocol monitor Eating_PM has the public procedure Eat with the integer parameter S, which is simultaneously a seat number and the number of the fork to the right of that seat. Below is an outline of Eating_PM in a simplified notation (and assuming *n* > 1).

```
package body Eating_PM is

  Sema : array (1 .. n) of Semaphore_Type;
  -- PM operation:
  procedure Eat (S : Integer; …) is
     Fork1: Integer := S;
     Fork2: Integer := S+1;
  begin
     if S = n then
        -- Correct order for nth philosopher
        Fork1 := 1;  Fork2 := n;
     end if;
     Sema (Fork1).Acquire;
     Sema (Fork2).Acquire;
     -- Eating, possibly a lengthy operation,
     -- occurs here

     Sema (Fork2).Release;
     Sema (Fork1).Release;
  end Eat;
end Eating_PM;
```

The procedure Eat with its sequential logic reads from top to bottom. It can involve anything such as a computation, accesses, or an animation of the philosophers in some form.

3.2 Another example

A more practical example is a PO encapsulating a large table whose individual records are updated frequently by multiple tasks in quick calls to protected procedures. Occasionally, a transmission task reads and transmits all the data. A protected operation that reads all the data could disrupt the updating of the table prohibitively. Instead, we can encapsulate the PO in a PM with an operation that iterates through the data by calling a quick protected operation repeatedly. The PM would also expose procedures that simply forward each call to the protected procedures that make the quick updates.

3.3 Limitations

Even though a PM is a higher-abstraction-level analog to a Hoare monitor, the programmer must insert the necessary calls to PO operations into the PM operations. Another difference is that multiple tasks can be executing operations on a PM at the same time except for statement sequences inside a PO.

While it might be desirable to place all critical sections bracketed by acquire and release inside protocol monitors, this may not always work. The primary example is the case where the critical sections are not nested. For example, a switch engine pulling cars in a switchyard may be able to acquire access to one track segment at a time in this fashion: acquire segment A, acquire segment B, release segment A, acquire segment C, release segment B, etc. [7].

Hoare [3] gave a different example where one task acquires exclusive access to a buffer from a pool and uses it to send data to another task, which then releases the buffer. Similarly, the stands in the manufacturing system may serve as a staging area between two kinds of automated vehicles where one vehicle places an item to be picked up by another, which then releases the stand [7].

We may also question whether a PM is worthwhile in the philosophers' problem, where effectively all the program logic ends up inside the PM. It may be different with an animation that includes the philosophers' movements to and from the table or the like, which would happen outside the PM.

4 General protocol monitors

While PMs encapsulating semaphore POs together with the logic using one or more shared resources may be quite common, the protocol-monitor concept is more general and can involve larger state spaces than "free" and "occupied." For example, such a PM may encapsulate a communication protocol for a distributed system. By calling an operation on a PM, which in turn calls operations on the PO, a calling task can do some computation outside a PO but within the PM. One example is a multicast algorithm; another implements an algorithm for distributed mutual exclusion. In such cases, the protocol-state machine has more states than a semaphore. We shall call such a more general device a *state-machine PO* [6, 7]. Each action such as sending a message involves a number of states that are traversed sequentially. A semaphore PO is a specialized state-machine PO.

For this discussion, we define a general PM in Ada terms as a package that meets the following requirements:

1. It hides one state-machine PO, or, if necessary, more.
   - Having two or more state-machine POs in a PM may be justified if tasks calling the PM can be executing more than one protocol at the same time. In the dining-philosophers example, this happens both because each philosopher needs exclusive access to two forks simultaneously and because two or more philosophers may be eating at the same time.
   - In the simplest and commonest case the POs are local to the PM-package body, but they can be hidden in other ways, as we shall see.
2. It exposes unprotected procedures and functions that in turn call the PO operations. At least one of the unprotected operations should be unsuited to be inside a PO.

4.1 Example: Two-phase total-order multicast

In total-order multicast, *originator* nodes broadcast messages to a group of *receiver* nodes called group *members*. All receiver nodes in the group must agree on the order they deliver the messages to the application layer. For this, the communication layer at each group member buffers each message until its proper place in the order has been established [2]. One use of such total-order multicast is for distributed mutual exclusion, as we shall see later.

The total-order multicast protocol has two phases [2]:

1. The originator sends a message and collects an acknowledgment from each receiver containing a *proposed sequence number* acceptable to that receiver.
2. The originator determines an *agreed* sequence number and sends it to the group in a *commit* message.

After receiving an agreed sequence number, each group member may be in a position to deliver one or more messages to its application layer.

We are interested in the sender side of the protocol. For simplicity we assume that each node can be multicasting only one message at a time. The sender keeps the state of the ongoing multicast in a state-machine PO – here called *Sending_PO*. Below is an outline in simplified Ada of the Total_Order_Sender_PM package. The simplified style intends to highlight the interaction between the protocol-monitor operations and the encapsulated state-machine PO. Also, the algorithms for multicast and distributed mutual exclusion are simplified and intended only to illustrate protocol monitors.

Two types of tasks interact through this PM: *Application tasks* that need to multicast a message, and *listener tasks* that deliver acknowledgments from other nodes.

```
package body Total_Order_Sender_PM is

 protected Sending_PO
 -- State-machine PO for total-order protocol
 -- from sender's point of view.
   entry Start_Send when Free
   -- Get exclusive access to sender machinery
   procedure Ack_Received
   entry Commit
   -- Block until all acks received
   procedure Done
   -- Release exclusive access to machinery
  end Sending_PO;

  -- PM operations:
  procedure Send
  -- Application task wants to send
      Sending_PO.Start_Send  -- Wait for exclusive
                             -- access to multicast
      -- Here: Send a message to each recipient
      Sending_PO.Commit  -- Wait for all acks
      -- Here: Send agreed sequence no. to each
      -- recipient
      Sending_PO.Done   -- Release exclusive access
  end Send;

  procedure Ack_Received
      Sending_PO.Ack_Received -- Patch call through
                              -- to Sending_PO
  end Ack_Received
end Total_Order_Sender_PM;
```

The interaction is as follows:

An application task calls the procedure Total_Order_Sender_PM.Send, whose logic ensures that the multicast protocol is followed: From within Send, the task first calls Sending_PO.Start_Send to obtain exclusive access to the multicast machinery. Upon return from Start_Send, it then sends the messages, and calls the entry Commit to block until all acknowledgments are in. Back in Send from that entry call, it sends the commit message and finally calls Done to release the exclusive access before returning to the point after its call to Send. Thus the application task does the actual sending outside the PO but within the PM.

While an application task is still busy sending, acknowledgments can start arriving from other nodes. Each acknowledgment represents an event occurring to the protocol state machine. For that reason, the sending application task cannot do the sending from within the state-machine PO.

Each time a listener task receives an acknowledgment from a group member it calls Total_Order_Sender_PM.

Ack_Received and from there Sending_PO. Ack_Received. The latter operation counts the acknowledgments. The entry Commit's barrier opens when all have been received, which allows the application task to proceed. (Total_Order_Sender_PM.Ack_Received is an example of a PM operation that only forwards each call to the corresponding PO operation.)

4.2 Example: Distributed mutual exclusion

The PM we are going to discuss next uses Total_Order_Sender_PM just described. Mutual exclusion is a textbook example of a distributed algorithm [2]. It is used in situations where the members of a group of distributed nodes must agree on which one of them – if any – holds a certain *lock* at each point in time. We shall refer to it as the *distributed lock*. What a node is entitled to do when holding the lock does not concern us. That is by convention.

Each node can have multiple tasks, which may compete among themselves as well as with other nodes for the distributed lock. At most one application task at a time gets to compete with requests from other nodes, however: In the example below, a task that enters Acquire changes the protocol state thereby forcing any other calling tasks to wait. It is the local winner and contends for the distributed lock with other nodes.

In Ricart and Agrawala's distributed mutual exclusion algorithm [5], a node requests a distributed lock by multicasting a *request* to all other nodes. Each node maintains a queue of pending requests ordered by their timestamps. Figure 1 shows the protocol states and state transitions for each node. Different nodes can be in different protocol states at any given time. Only one can be in state *Held*, but more than one can be in *Wanted* or *Released*.

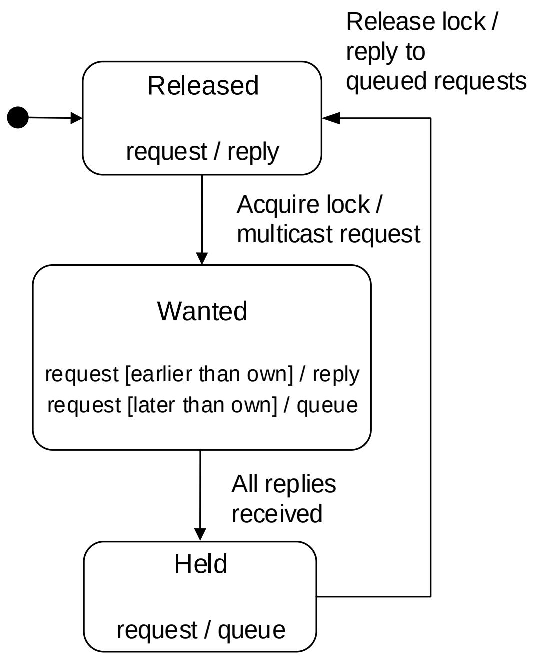

Release lock / reply to queued requests
Released
request / reply
Acquire lock / multicast request
Wanted
request [earlier than own] / reply
request [later than own] / queue
All replies received
Held
request / queue

Figure 1 Protocol-state diagram of Ricart and Agrawala's distributed mutual exclusion algorithm

Each node *replies* to each new request according to its own protocol state when it receives the request:

- If the node is in state *Held*, it is holding the distributed lock, and replies once it has released it.
- If the node is in state *Wanted*, it has submitted its own request. If the new request is older than the own request, the node replies immediately. If the new request is younger than the own request, the node replies only after releasing the lock.
- If the node is in *Released*, it replies immediately upon receiving a request.

A node receiving a *reply* from a node X removes X's request (if any) from the top of the queue. When a node has received all its *reply* messages it enters state *Held* and holds the distributed lock.

As with the multicast, it is prudent to encapsulate the protocol state in a PO at each node. The multicasting of the requests should not be inside the PO because reply messages from the other nodes may start arriving before the multicast is complete: Listener tasks receiving those replies must be able to call the PO to deliver them. For this, each node has a PM with the following outline:

```
package Ricart_PM
    protected Lock_PO
    -- State-machine PO for distributed mutex
        entry Acquire when Protocol_State = Released
            -- Set state to Wanted
        entry Confirmed when True
            -- Establish the commit time. Requeue
            -- application task for entry Replies_Collected
        procedure Reply_Received
            -- Count replies;
            -- set state to Held when all received
        procedure Release;
            -- Set state to Released
        procedure Request_Received
                    (… Reply : out Boolean)
            -- Tell caller to reply or to queue request
    private
        entry Replies_Collected
                when Protocol_state = Held;
        Protocol_State :  …  := Released;
        Replies : Integer := 0;
        -- A queue of pending requests goes here
    end Lock_PO;

    -- PM operations:
    procedure Acquire_Lock
     -- Application task wants lock
        Lock_PO.Acquire;   -- Upon return, state is
                           -- Wanted
        Total_Order_Sender_PM.Send …
        Lock_PO.Confirmed
        -- Block on Replies_Collected till lock acquired
    end Acquire_Lock
    procedure Release_Lock
     -- Patch call through to Lock_PO.Release
    procedure Reply_Received
    -- Patch call through to PO.Reply_Received.
    -- Called by some listener task
    procedure Request_Received
    -- Called by listener task when another node
    -- wants lock
      Lock_PO.Request_Received (Reply : out Boolean)
      -- Here: If Reply then send reply to node
    end Request_Received
  end Ricart_PM;
```

As with Total_Order_Sender_PM, two types of tasks interact through Ricart_PM: *Application tasks* that need the distributed lock and *listener tasks* that deliver messages from other nodes. Here is how Ricart_PM works:

An application task that needs the distributed lock calls Ricart_PM.Acquire_Lock. From within that procedure it calls the entry Lock_PO.Acquire and blocks until Protocol_State = *Released* in case another local application task is already pursuing or holding the distributed lock. Upon return from its call to Lock_PO.Acquire, the application task holds the exclusive right to pursue the distributed lock on behalf of this node. The protocol state is *Wanted*. Back in Ricart_PM.Acquire_Lock, the application task calls Total_Order_Sender_PM.Send to multicast its request. After completing the multicast, the task calls Lock_PO.Confirmed and then blocks on entry Replies_Collected until the protocol state is *Held*.

Meanwhile, for each reply received, a listener task calls Ricart_PM.Reply_Received, which is effectively a call to Lock_PO.Reply_Received. The latter operation counts the replies and changes the state to *Held* when the last reply has been received. At that time, the application task returns from Ricart_PM holding the distributed lock. It ultimately calls Ricart_PM.Release_Lock and changes the protocol state to *Released*.

The application programmer interface

Application and listener tasks all call Ricart_PM. The application-program interface (API) should be separate from the listener-task interface. This presents us with two additional issues as follows.

1. We need an API that includes only those operations that application tasks use. The implementation in Figure 2 shows a child package Ricart.api that only exposes Acquire_Lock and Release_Lock. Those operations in turn patch any calls through to the corresponding operations of the parent package Ricart_PM. Those operations should be in the private part of Ricart's package specification and not exposed to the listener tasks. This is not reflected in the above outline of Ricart_PM.

2. We are now also presented with the kind of problem that a basic protocol monitor can solve: Ricart.api exposes the operations Acquire_Lock and Release_Lock to the application program. In order to enforce the acquire-

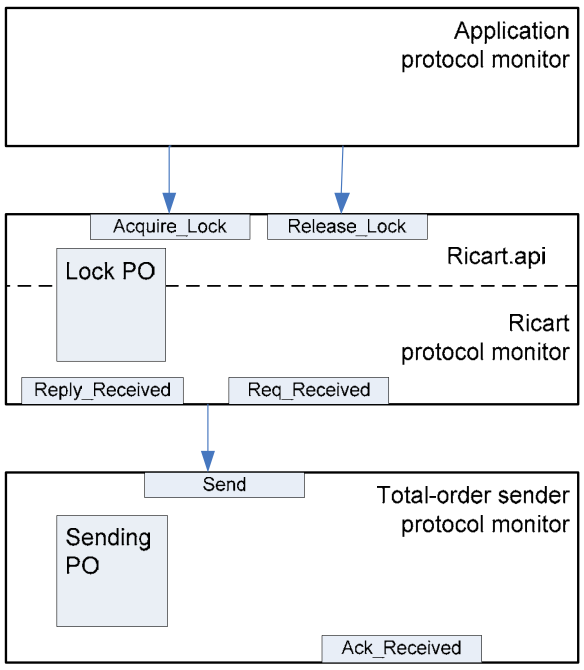


Figure 2 PM stack consisting of Application protocol monitor, Ricart, and Total_Order_Sender_PM. Ricart.api is a child package. Reply_Received and Req_Received are called by listener tasks.

release protocol, the application programmer should encapsulate in a PM the calls to Acquire_Lock and Release_Lock together with those operations carried out by the application task under exclusive access. This PM is called *Application protocol monitor* in Figure 2. Notably, *Application protocol monitor* has no embedded PO but instead shares Ricart_PM's Lock_PO.

5 Conclusion

Ada's protected-object syntax is helpful for the construction of PMs in that a PO encapsulated in a PM, shows up in the program text as a box within a box. True, this does not stop a programmer from declaring variables in the PM outside the PO. While such variables may be useful, they should be viewed with suspicion because variables in a PM must usually be declared inside a PO.

In other languages (such as Java), we might implement a state-machine PO as an inner class with synchronized operations. But since the syntax doesn't stop us, it might be tempting to simplify things and mix in synchronized operations, which should be private, with the PM class's public operations. If not done carefully, this may lead to errors where variables that should only be accessed by a single thread holding a lock are also accessed by threads executing the PM's public operations.

It is often easy to use basic protocol monitors, which encapsulate operations together with semaphore POs. The situation is different if we are designing a PO and find the need to add logic that is not nearly instantaneous such as sending messages and waiting for replies. Unless we are already familiar with the protocol monitor as an available programming tool, the idea of wrapping the PO in a PM may not spring to mind. The added layer of indirection also incurs some overhead especially if many PM operations simply "patch" calls through to corresponding PO operations.

Acknowledgments

Jeffrey R. Carter pointed out the importance of protocol monitors.

References

[1] P. Brinch Hansen (1977), *The architecture of concurrent programs*, Prentice-Hall.

[2] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair (2012), *Distributed systems: Concepts and Design*, 5th Ed., Addison-Wesley.

[3] C. A. R. Hoare (1974), *Monitors: An operating system structuring concept*, *CACM*, vol. 17, no.10, pp. 549-557.

[4] C. A. R. Hoare (1985), *Communicating Sequential Processes*. Prentice-Hall International.

[5] G. Ricart, G. and A. K. Agrawala (1981), *An optimal algorithm for mutual exclusion in computer networks*, *CACM* vol. 24, no. 1, pp. 9-17.

[6] B. I. Sandén (2001), *A Design Pattern for State Machines and Concurrent Activities*, in *Ada-Europe 2001*, D. Craeynest and, A. Strohmeier, A. (eds.), LNCS, vol. 2043, pp. 203-214.

[7] B. I. Sandén (2011), *Design of Multithreaded Software: The entity-life modeling approach*, IEEE Computer Society/Wiley. ISBN 978-0470-87659-6.

[8] D. C. Schmidt (2000), *Monitor Object − an Object Behavior Pattern for Concurrent Programming*, *C++ Report*, SIGS, Vol. 12, No. 4, May.

Towards a Power Adaptation Strategy in Multi-core Embedded Devices. A Case Study: a HMI for Wheelchair Command Technique

Agnès Ghorbel, Nader Ben Amor, Mohamed Jallouli
Computer and Embedded Systems Laboratory, Ecole Nationale d'Ingénieurs de Sfax, Tunisia; email: ghorbel.agnes@gmail.com

Abstract

Power management is one of the most important challenges in the design of today's nomad embedded systems. Power management goal is to maximize performance under a given power budget. Power management techniques should include the opportunity to balance between the demanding requirements for great computing performance/throughput and the effect of offensive power consumption and negative thermal impacts. In this paper, we propose a power management technique that aims at improving energy efficiency of multimedia embedded systems with multiple processors. This technique relies on balancing the use of both CPU cores and online deciding at what core speed to process each greedy application function. As a case study, we use a Human Machine Interface for an intelligent wheelchair control technique based on face and eyes movements. We study the challenges in power management of this application on a dual-core ARM based system. The obtained results demonstrate the effectiveness of our methodology in achieving a best performance/power saving ratio. We can decrease power consumption by 6.5% while respecting computing performance limit in term of execution time.

Keywords: Multi-core, power management, battery life, performance, wheelchair control.

1 Introduction

The demand for mobile multimedia applications have been rapidly increased which led designers to rival in areas such as hardware requirements, coding efficiency and cost. Nevertheless, such mobile and battery operated systems impose a common challenge: How to achieve on a computing limited resources system different user requirements such a minimal acceptable service and a satisfying battery lifetime while facing different computing and memory resource limitations.

Battery-powered mobile devices are becoming, more and more, an important element of our lives. Performance guarantee and energy efficiency are important factors for the economic success of an embedded system.

Modern embedded systems are becoming more and more complex and they often require both high performance and low power consumption. To achieve both requirements, exploiting application parallelism (thread, instruction and data) opportunities are already well known solutions. Also, the need of power management techniques can aim at improving energy efficiency of embedded systems.

Time to market deadlines combined with strict timing/power constraints have created the necessity for computing platforms able to support a range of multimedia applications running on such systems. Multiprocessor system-on-chip occurs as a viable platform for these applications, as they usually contain divers processing elements cores, memories, I/O components, etc. Many commercial MPSoC community oriented platforms are now provided including the TI OMAP, Samsung Exynos, Mediateck, Qualcom and Broadcomm.

In order to efficiently and productively use these platforms, designers should have to explore alternative mapping of multimedia application tasks resulting in implementations with different energy and time configurations.

In this paper, we discuss the problem of how to implement a complex media oriented application on a multi-processor embedded system so that performance is guaranteed while reducing power consumption. For the purpose of our investigation, we propose a low power technique that considers adaptation under different constraints. The system is always adjusted to provide the minimum required computing power. The timing constraint is always met. But when it is possible, the computing power can be reduced for energy saving purpose. We perform these adaptations by dynamically reconfiguring the number of active processor cores and their frequencies while maintaining a minimal elapsed time.

As a case study, we consider a Human Machine Interface (HMI) based on human face for wheelchair command technique as our major test to apply our methodology. We highlight also the need of power management in embedded systems and survey several research works which are aimed at improving energy efficiency of embedded systems.

The rest of the paper is organized into five sections. Section 2 gives an overview of literature on power management techniques in modern processors. In section 3, we present our approach for power management in where we describe how the controller can map complex functions to the adequate core according to time analyse of the code needs. The application and the embedded platform in where we apply our mechanism, and the experimental results are depicted in section 4. And finally, section 5 concludes our study and discusses future research directions.

2 Background

Embedded systems are becoming an increasingly popular computing platform. They have to run real-time multimedia applications. But unlike conventional desktop and server, they are frequently subject to resource restrictions such as limited battery life. Thus, they are required to self adapt internally to their restricted/limited computational and power resources. The recent transition to multi-core and multi-threaded processors has created new series of challenges for dynamic power management. The decision criteria for adapting shall firstly consider the performance cost of the adaption and the probability of meeting a particular demand for performance.

Researchers have then focused on the integration of power adaptation strategies in embedded systems. Among the adaptation techniques, there are the dynamic voltage scaling (DVS) and the dynamic power management (DPM) techniques. The DVS [1] have to adjust the frequency and the power supply to the CPU. It allows managing the used energy according to the workload of the system and is based on the application workload prediction using either heuristic methods [2] or the CPU time estimation [3, 4].The DPM [1] is a design methodology that dynamically reconfigures an electronic system to provide the demanded services and performance levels with a minimum number of active components or a minimum load on such components. There are numerous works that propose the application of both DVS and DPM techniques to maximize energy savings. In most studies, DVS is used to decrease processors energy consumption and DPM is used to shut down other peripheral devices [5, 6, 7]. In [8], authors are studied a real-time system with a DVS-enabled processor and a fixed number of offchip devices applied by the tasks during execution. In this strategy, DPM is used at the offchip devices and not at the processors level. The authors in [9] have proposed the application of a DVS mechanism in a real-time heterogeneous 2-levelgrid. The results were encouraging as they achieved an important energy reduction without important performance degradation. Also, the same authors in [10] study the performance and the energy efficiency of a real-time distributed system with four heterogeneous clusters with DVS processors that can adjust their operating frequency depending on the required workload. With the application of power saving mechanisms, they reached energy consumption saving, depending on the system workload.

Our study differs in that we study power adaptation in the field of a multi-core processor. A multi-core framework which promotes several low-power cores running at low processors frequencies is better rather than a single high-speed power hungry core. Different possibilities to manage power consumption in Linux and most specifically in Embedded Linux exist, but most of them were designed for systems with only one processor. Low power techniques in multi-core systems based in ARM architecture can be divided in two groups, those techniques derived from general Linux power management (and working in current embedded devices) and those developed and implemented for recent multi-core integrated devices.

In this paper, we are interested to present an energy-efficient technique for a mobile multiprocessor SoC. This technique seeks to minimize the total energy consumed by the SoC without violating the time constraints. To achieve this goal, we exploit both instruction-level parallelism and data-level parallelism for our application in a previous publication [11] and we integrate a script that gives the possibility of switching of one of the cores of the processor and deciding at what CPU speed to run applications tasks. The most interesting benefits of this technique are the lower consumption ensured using only one of the cores, and a simpler condition to enter in processors low power modes.

3 Overview of the approach

The essence of our proposed approach is that it considers targeting application functions to an adequate hardware configuration. The main of this technique is a tradeoff between power saving (battery life) and real time constraint (limit for the CPU time allocated for each application). This implies the existence of various combinations according to the number of cores and the component configuration set by the designer: clock frequency.

In image/video processing algorithms, functions can range from the simplest one (necessitating 1 core) to the complex one (necessitating 2 or more processing units). Also, all tasks can be processed on software with various operating frequencies allowed by the platform. So, for a low complexity function, the controller activates only one core at a low frequency and all the others components are in a standby mode. On demand, the second core can be dynamically allocated with a corresponding clock. Our work-flow is composed of two main steps:

- Off-line characterization step: this step is based on a study of the target application functions in order to determine the elapsed processing time of each function for different configurations (according to the number of cores and the frequency).
- The result of this step is the input of the composite step of the software controller. This later consists, according to timing characterization table, in balancing the use of both cores at different operating frequencies to attain the best performance / power saving ratio (running the application under a minimum consumed energy while remaining in a time interval not to be exceeded). Figure 1 presents the strategy for our software controller.

The application is running, as default, under the configuration that provides the minimum energy (one core at the low frequency). Then, the software controller script performs the timing comparison/exploration at every task entry to choose the closet configuration mode that best meets the system objective, according to the type and the need of the function.

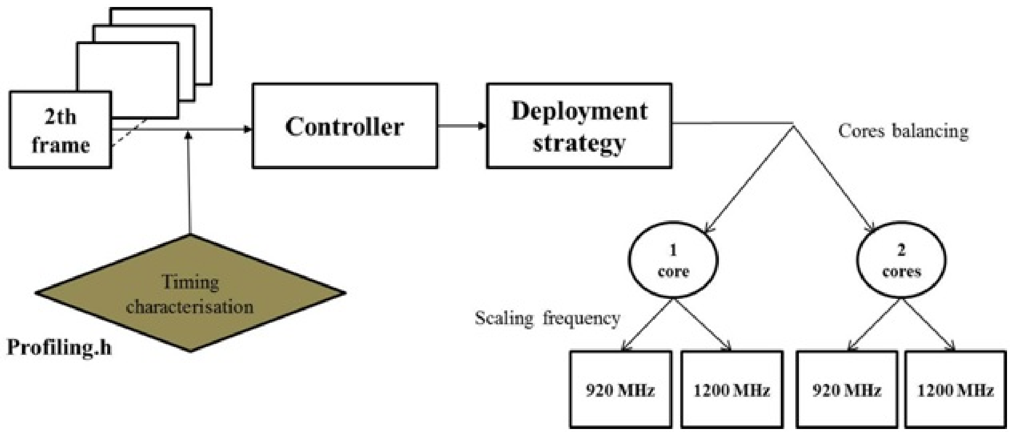


Figure 1: The software controller principle

The software controller script does the following:

```
Data: Timing characterization table, $Time_{limit}$
Result: The best configuration
for each task $T_i$ do
    if $(Time_i C_{g_H} - Time_i C_{g_L})/Time_{limit} * 100 <= 3\%$ $of\, Time_{limit}$ then
        $C_g <- C_{g_L}$ ;
    else if
     $(Time_i C_{g_H} - Time_i C_{g_L})/Time_{limit} * 100 >= 10\%$ $of\, Time_{limit}$ then
        $C_g <- C_{g_H}$ ;
    else
        $C_g <- C_{g_b}$;
    end
end
```

Algorithm 1: The software controller

Where:

$Time_{limit}$: The time limit not to be exceeded by the application;
T_i: Application functions;
C_{g_H}: The configuration in which 2 cores are active at the high frequency (which supplies the highest performance);
C_{g_L}: The configuration in which 1 core is active at the low frequency (which consumes the least energy);
C_{g_b}: The configuration in which 2 cores are active at low frequency;
$(Time_i$: Time elapsed by each function in each configuration C_g.
In the next section, testing environment and several tests done with the Pandaboard-ES based on ARM architecture using our methodology will be presented.

4 Validation

4.1 Software application

Our target systems are embedded systems running multimedia applications. As a case study to validate our mechanism, we consider a complex multimedia application which enables to control an intelligent electrical wheelchair (IEW). An IEW is a system developed to deliver a significant improvement to the life conditions of handicapped with limited capacity.

This IEW must also provide users foolproof driving conditions with a rapid reaction (particularly in suspect places). A major aspect for this purpose is the development of a control system capable to react to the user's intention in the shortest time and with the extreme accuracy.

Until now, divers HMIs for wheelchair motion control have been cautiously studied like head gesture [12], voice recognition [13], tongue piercing [14], bio-potential signals (EMG, EEG, EOG, etc.) [15, 16, 17, 18, 19] and etc.. Our wheelchair application relies on human face features where face directions and eyes blinks are used to implement an intelligent algorithm for wheelchair motion. More details about the interface are presented in [20]. The process applied to recognize user's intention is conducted according to these following steps: detection, recognition and conversion as described in Figure 2.

The proposed interface allowed the intended person to immediately control the chair by altering face orientation and eyes blinks. If the user wished to go forward, he just only blinked his eyes twice. If he desired stopping the wheelchair, he just blinked his eyes twice again. For turning control commands, two signals types are applied: turning right whenever the inclination of face is positive and turning left whenever left whenever the inclination of face is negative.

4.2 Embedded platform

We implemented this algorithm on the pandaboard-ES, an embedded multi-core architecture based on SoC design with moderate consumption of electrical energy. Pandaboard-ES ships with an OMAP4 4460 chip, that integrates a dual-core ARM Cortex-A9 MPCore at 1.2GHz, 1 GB low power RAM and Bluetooth 2.1, Ethernet, Wireless, HDMI The ARM Cortex - A9 MPCore CPUs are the heart of the processing system which also includes On-Chip Memory (OCM), external memory interfaces and a set of I/O peripherals. Clock speed is dynamic and can be adjusted at a very granular level depending on performance/power ratio.

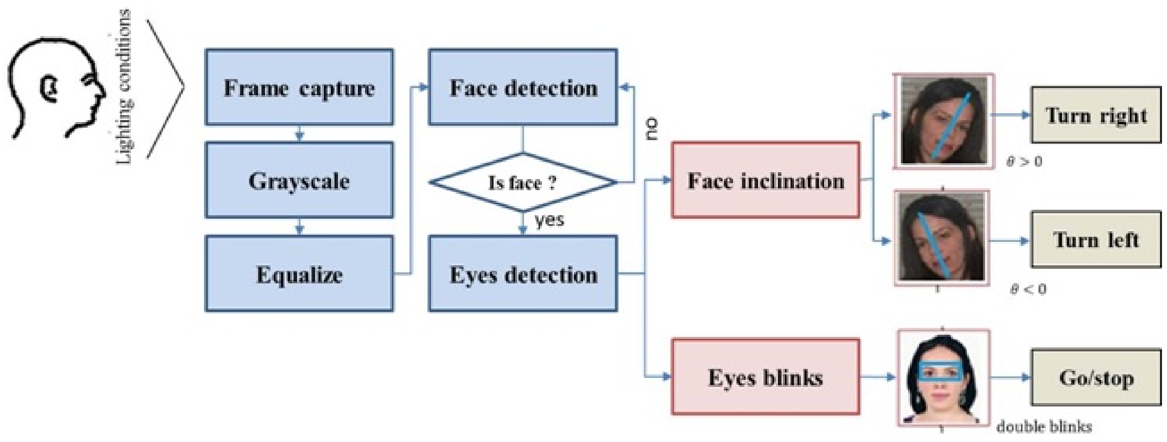


Figure 2: The process to recognize user's intentions

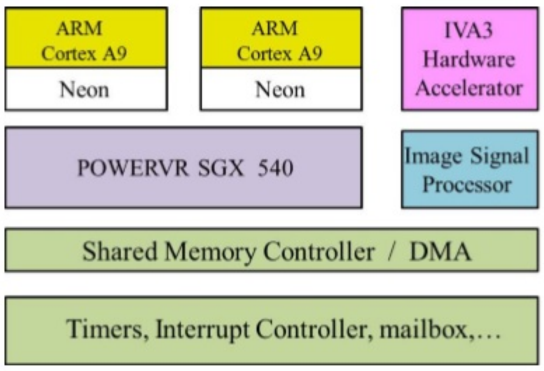

ARM Cortex A9
Neon
ARM Cortex A9
Neon
IVA3 Hardware Accelerator
POWERVR SGX 540
Image Signal Processor
Shared Memory Controller / DMA
Timers, Interrupt Controller, mailbox,...

Figure 3: The pandaboard-ES platform

4.3 Experimental Results

4.3.1 Power measurements

We performed a set of power measurements on OMAP4460 mobile SoC. The OMAP chip has several power pins on its package, designed to achieve detailed power control adjusted to application or use-case needs. In the Pandaboard-ES reference manual [21], the power of these pins is supplied by a dedicated TWL6030 power regulator IC. To analyze the power consumption of the two cores CortexA9, we measure the voltage on and the current through the inductors L15 using it ohmic internal resistance. So measuring the current is done by measuring the DC voltage drop, as well as the DC resistance. According to the datasheet of the inductor, the L15 should have a resistance of <0.05 Ohm. The SMPS input voltage (Vbat) was measured to be 3.68 Volt. When executing application, the DC voltage drop across the inductor (Vdrop) is measured. The resulting power consumption is now expressed as 1:

$$P = Vbat * Vdrop(L15)/R(L15) \qquad (1)$$

All measurements are done using just a standard multi-meter.

4.3.2 Results

The purpose of this experiment was to prove that we can calculate such power consumption measurements with sufficient precision for software source-code optimization. Moreover, it is our objective to evaluate the speed/power tradeoffs of our proposed approach.

Figure 4 illustrates the maximum power consumption during wheelchair navigation using one and two cores.

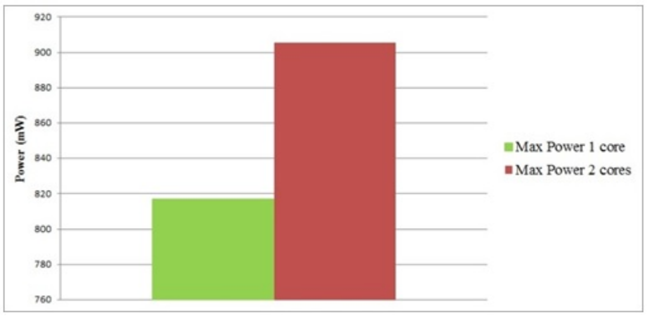


Figure 4: The maximum power consumption during application

The clock is operated to run on the platform at different available frequencies ranging from 350 MHz to 1200 MHz. Figure 5 presents timing results for a $640 * 480$ image resolution when all tasks are processed on software with two different operating frequencies 920 MHz and 1200 MHz and are implemented on one core and both two cores of the processor.

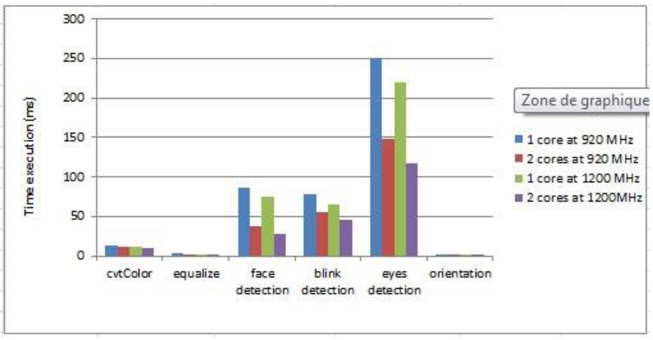


Figure 5: Profiling results on $640 * 480$ image resolution

Table 1: Time execution (ms) of functions implemented on different frequencies and number of cores

Frequency (MHz)	Number of cores	Functions								
		capture	cvt color	equalize	face detection	blink detection	eyes detection	orientation	display	total
920	1	80	13.331	4.272	85.62	78.739	250.231	1.289	80	593
	2	65	11.291	2.689	38.269	55.799	148.745	0.732	65	397
1200	1	70	12.142	2.441	75.269	64.788	220	0.747	70	525
	2	50	9.265	1.958	27.456	46.19	116.791	0.615	50	312

Table 2: Time execution (ms) of functions implemented with the software controller

Functions	capture	cvt color	equalize	face detection	blink detection	eyes detection	orientation	display	total
Dynamic core management	50	13.331	4.272	27.456	46.19	116.791	1.289	50	319.329

Figure 6 presents the relationship between frequencies and power consumption of the HMI application on various cores of the processor. The various measures of power consumption for HMI application are distinguished with application on one core or two cores and with no application.

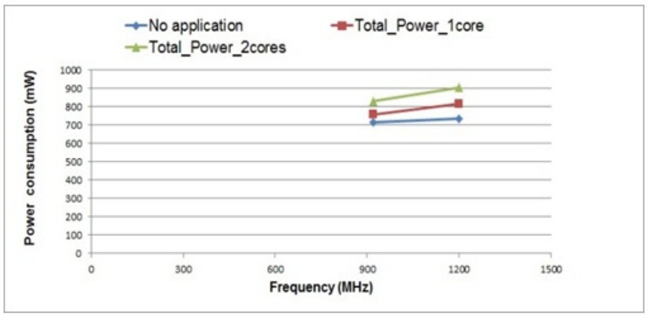


Figure 6: Power consumption with a 640 * 480 input image

A comparison between frequencies and number of activated cores in power consumption and execution time is made. Figure 7 illustrates the power consumed during a video frames for different configurations and Figure 8 (a zoom on the dotted part of Figure 7) illustrates the execution time for a frame also for the different configurations.

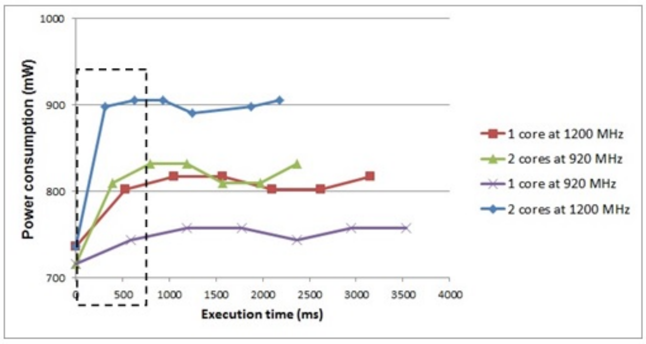


Figure 7: Power consumption comparison during 6 frames

As we can see in the graphics, at a defined frequency, using only one core power consumption on Pandaboard-ES was near half the power consumption when using two cores. For frame execution time and at a defined frequency, the application is faster when using two cores than one and the difference is about 200 ms. As our goal is to guarantee high speed with low power consumption, we decided to exploit CPU in a dynamic

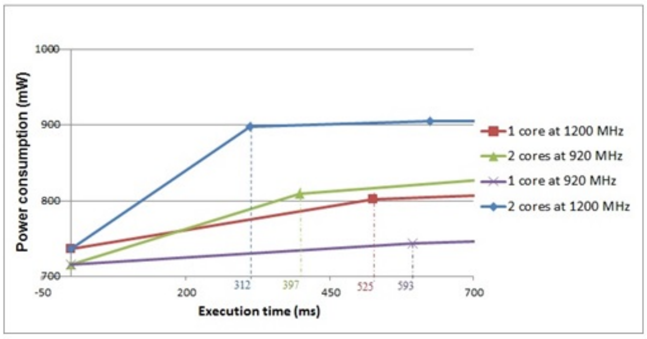


Figure 8: Execution time comparison during 6 frames

behavior. Instead of executing all tasks with one or two cores, we activated or deactivated the second core of the SoC and adjust it on two different frequencies scaling according to timing characterization.

Table 1 depicts the timing characterization of each task running on one or two cores and at two core frequencies.

The limit in execution time for the HMI that not be exceeded is 400 ms per frame.

When applying our script to our application, we obtained the results illustrated in Table 2.

With core management script, we enabled the second core at a higher frequency when task timing measure is too high for only one core at a lower frequency. Figure 9 exposes this specific behavior and demonstrates how the dynamic core management performs compared to the other configurations.

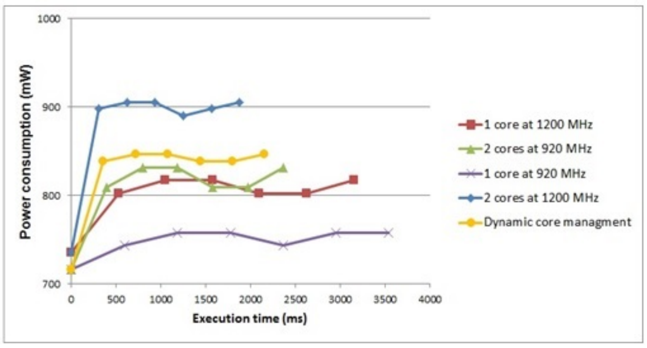


Figure 9: The dynamic core management

Table 3: Power/Energy of Wheelchair control system

Image resolution	frequency (MHz)	Number of cores	Power (mW)	Time (ms)	Energy (mJ)
640 * 480	920	1	758.03	593	449.511
	1200	2	905.28	312	282.44
	dynamic core management		846.4	319.329	270.001

The power consumption achieved using the script for dynamic core management produces power consumption as better than two cores at high frequency (Table 3) and an execution time as a little less rapid than two cores at high frequency. The consumed power is lower than using two cores, but with greater performance than using only one core.

In addition, the energy consumed by a system is the number of power dissipated during a certain period of time. For instance, if a task T is running on a MPSoC during an execution interval of Time: [a, b] then the energy consumed by the MPSoC during this time interval is given by equation 2:

$$E_T = \int_a^b P(t)\,\mathrm{d}t \tag{2}$$

So, as illustrated in Table 3, the higher frequency is scaled, the higher power is taken. In contrast, the higher frequency is scaled, the lower energy is taken as time frame is low for a higher frequency. For our dynamic script management, we have succeed in saving the energy consumed by the application as we have reduced the power while maintaining the frame time as close as provided by the highest frequency.

To conclude, we have achieved a reduction on average power consumption by 6.5% while reducing performance by an average of less than 3%. The performance loss doesn't matter as long as the control is able to process everything in an interval of time to not exceed.

5 Conclusions

Power management of multi-core processors is highly important as it permits power/ energy savings. In this paper, we propose an approach that is able to reduce power consumption without great degradation of performance but in an acceptable time interval. The achieved average power consumption reduction is about 6.5% with 3% performance loss.

In ongoing work, we can propose acceleration technique like integrating a GPU to ameliorate the execution time while maintaining a minimum of consumed energy as the GPU is characterized by its low energy consumption.

References

[1] S. Srinath and A. Pillai (2016), *Adaptive interplay of dvs and dpm for power consumption reduction in real-time embedded processors*, Indian Journal of Science and Technology, vol. 9, no. 30.

[2] S. Y. Bang, K. Bang, S. Yoon, and E. Y. Chung (2009), *Run-time adaptive workload estimation for dynamic voltage scaling*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 28, no. 9, pp. 1334–1347.

[3] M. K. Bhatti, C. Belleudy, and M. Auguin (2011), *Hybrid power management in real time embedded systems: an interplay of dvfs and dpm techniques*, Real-Time Systems, vol. 47, no. 2, pp. 143–162.

[4] Y. Hayamizu, K. Goda, M. Nakano, and M. Kitsuregawa (2011), *Application-aware power saving for online transaction processing using dynamic voltage and frequency scaling in a multicore environment*, in Proceedings of the 24th international conference on Architecture of computing systems, pp. 50–61, Springer-Verlag.

[5] V. Devadas and H. Aydin (2010), *Dfr-edf: A unified energy management framework for real-time systems*, Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE, pp. 121–130, IEEE.

[6] B. Zhao and H. Aydin (2009), *Minimizing expected energy consumption through optimal integration of dvs and dpm*, Proceedings of the 2009 International Conference on Computer-Aided Design, pp. 449–456, ACM.

[7] J. Zhuo, C. Chakrabarti, and N. Chang (2007), *Energy management of dvs-dpm enabled embedded systems powered by fuel cell-battery hybrid source*, Proceedings of the 2007 international symposium on Low power electronics and design, pp. 322–327, ACM.

[8] F. Kong, Y. Wang, Q. Deng, and W. Yi (2010), *Minimizing multi-resource energy for real-time systems with discrete operation modes*, in Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on, pp. 113–122, IEEE.

[9] G. Terzopoulos and H. Karatza (2012), *Maximizing performance and energy efficiency of a real-time heterogeneous 2-level grid system using dvs*, in Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, pp. 185–191, IEEE Computer Society.

[10] G. Terzopoulos and H. Karatza (2013), *Performance evaluation and energy consumption of a real-time heterogeneous grid system using dvs and dpm*, Simulation Modelling Practice and Theory, vol. 36, pp. 33–43.

[11] A. Ghorbel, M. Jallouli, and N. B. Amor (2015), *Speeding up execution time of a smart wheelchair command technique using parallel computing*, in Signal Processing Conference (EUSIPCO), 2015 23rd European, pp. 1586–1590, IEEE.

[12] P. Jia, H. Hu, T. Lu, and K. Yuan (2007), *Head gesture recognition for hands-free control of an intelligent wheelchair*, Industrial Robot: An International Journal, vol. 34, no. 1, pp. 60–68.

[13] F. Wallam and M. Asif (2011), *Dynamic finger movement tracking and voicecommands based smart wheelchair*, International Journal of Computer and Electrical Engineering, vol. 3, no. 4, p. 497.

[14] J. Kim, H. Park, J. Bruce, E. Sutton, D. Rowles, D. Pucci, J. Holbrook, J. Minocha, B. Nardone, D. West, A. Laumann, E. Roth, M. Jones, E. Veledar, and M. Ghovanloo (2013), *The tongue enables computer and wheelchair control for people with spinal cord injury*, Science translational medicine, vol. 5, no. 213, pp. 213ra166–213ra166.

[15] L. Wei and H. Hu (2011), *A hybrid human-machine interface for hands-free control of an intelligent wheelchair*, International journal of mechatronics and automation, vol. 1, no. 2, pp. 97–111.

[16] R. Maskeliunas and R. Simutis (2011), *Multimodal wheelchair control for the paralyzed people*, Elektronika ir Elektrotechnika (Electronics and Electrical Engineering), vol. 5, pp. 81–84.

[17] F. B. Taher, N. B. Amor, and M. Jallouli (2013), *Eeg control of an electric wheelchair for disabled persons*, in Individual and Collective Behaviors in Robotics (ICBR), 2013 International Conference on, pp. 27–32, IEEE.

[18] A. Santana and C. Yang (2012), *Robotic control using physiological emg and eeg signals*, in Conference Towards Autonomous Robotic Systems, pp. 449–450, Springer.

[19] A. S. Brando, L. B. Felix, T. F. Bastos-Filho, and M. Sarcinelli-Filho (2011), *Controlling devices using biological signals*, International Journal of Advanced Robotic Systems, vol. 8, no. 3, p. 30.

[20] A. Ghorbel, N. B. Amor, and M. Jallouli (2014), *An embedded real-time hands free control of an electrical wheelchair*, in Visual Communications and Image Processing Conference, 2014 IEEE, pp. 221–224, IEEE.

[21] T. Instruments (2011), *Omap4460 multimedia device technical reference manual*, version Y (Public Version), Section, vol. 8, no. 3.7, p. 13.

National Ada Organizations

Ada-Belgium

attn. Dirk Craeynest
c/o KU Leuven
Dept. of Computer Science
Celestijnenlaan 200-A
B-3001 Leuven (Heverlee)
Belgium
Email: Dirk.Craeynest@cs.kuleuven.be
URL: www.cs.kuleuven.be/~dirk/ada-belgium

Ada in Denmark

attn. Jørgen Bundgaard
Email: Info@Ada-DK.org
URL: Ada-DK.org

Ada-Deutschland

Dr. Hubert B. Keller
Karlsruher Institut für Technologie (KIT)
Institut für Angewandte Informatik (IAI)
Campus Nord, Gebäude 445, Raum 243
Postfach 3640
76021 Karlsruhe
Germany
Email: Hubert.Keller@kit.edu
URL: ada-deutschland.de

Ada-France

attn: J-P Rosen
115, avenue du Maine
75014 Paris
France
URL: www.ada-france.org

Ada-Spain

attn. Sergio Sáez
DISCA-ETSINF-Edificio 1G
Universitat Politècnica de València
Camino de Vera s/n
E46022 Valencia
Spain
Phone: +34-963-877-007, Ext. 75741
Email: ssaez@disca.upv.es
URL: www.adaspain.org

Ada-Switzerland

c/o Ahlan Marriott
Altweg 5
8450 Andelfingen
Switzerland
Phone: +41 52 624 2939
e-mail: president@ada-switzerland.ch
URL: www.ada-switzerland.ch