Considerations on Domain-Specific Architectures Applicability in Future Avionics Systems

10th Aerospace Technology Congress, Stockholm

Denis S. Loubach
dloubach@ita.br

Johnny C. Marques
johnny@ita.br

Adilson M. Cunha
cunha@ita.br

Department of Computer Systems
Computer Science Division – IEC
Aeronautics Institute of Technology – ITA

October 8-9, 2019

Agenda

About Me

Denis **Loubach**
Assistant professor at ITA, 2018
Department of Computer Systems
Computer Science Division – IEC

http://www.comp.ita.br/~dloubach
dloubach@ita.br

Background

- PhD in Electronics and Computer Engineering, ITA (2012)

Previous experience

- Unicamp (2014 $\sim$ 2018)
  - Assistant professor, embedded systems and digital systems
- Embraer, Mectron ($\approx$ 5 years)
  - Software and avionics systems engineer

Introduction

Avionics industry community tends to be conservative: **safety-critical** requirements

- *e.g.* flight control

Development follows the DO-178 for the **software**, and the DO-254 for the **hardware**

Introduction (cont.)

Nowadays, computer architecture improvement to deliver performance enhancements is getting harder to achieve:

- end of **Dennard** scaling
- slowdown of **Moore's** Law

[Chi18] points out the **disruption from bellow**

- end of Dennard's scaling
- shift to multicore
- slowing in the clock rate growth
- considerable architectures customization in **embedded devices**

Those facts have impact on the avionics systems domain as the system **complexity increases** exponentially from one generation to the next

Introduction (cont.)

A current trend in high-performance computer (HPC) architecture

- use of **domain-specific architectures** (DSA) instead of the general-purpose (GP) ones [HP17]

DSA is able to provide **performance** and **power** benefits to face GP architectures that are not having significant performance improvements in the last years (flat curve)

Potential research areas:

1. **open source architectures**
2. **reconfigurable computing**
   - supported by system-on-chip (SoC) composed by a hard processor and FPGA with runtime full and **partial reconfiguration** [Lou16]

Introduction (cont.)

Paper goal

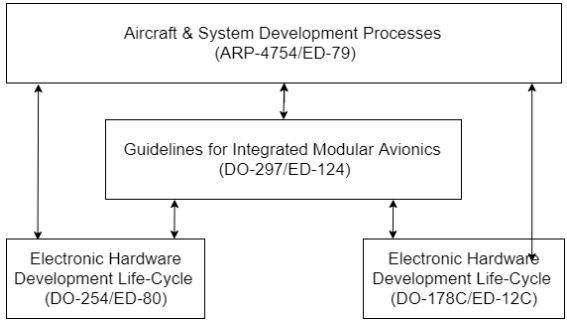
To discuss next challenges of future avionics systems considering the safety-critical aspects and the possibility of

- **formal models of computation** application
- together with **DSA**

Software & Hardware Working Together

Avionics Standards

In avionics systems development, there are four **standards** typically required to be used as part of the aircraft certification process



Relationship among certification standards

Software & Hardware Working Together

Domain-Specific Architectures

Semiconductor boom (supported by the Moore's Law) GP code was **accelerated** in GP cores:

- cache hierarchy
- 512-bit single instruction
- multiple data (SIMD) floating-point units
- various pipeline stages
- dynamic branch prediction
- out-of-order execution
- speculative execution
- multithreading and multiprocessing

Software & Hardware Working Together

Domain-Specific Architectures (cont.)

Nowadays, it is needed a **paradigm shift** in computer architecture to achieve a new level of efficiency

This shift concerns moving from GP hardware to **domain-specific architectures** [HP17]

Software & Hardware Working Together

Domain-Specific Architectures (cont.)

Architecture paradigm and domain [Gup19]

Architecture	**Domain**
Scalar (*e.g.* CPU)	Complex algorithms with diverse decision tree. Limited in performance scaling
Vector (*e.g.* GPU)	Efficient in reduced set of parallelizable functions. Suffers latency and penalties related to memory hierarchy
Programmable logic (*e.g.* FPGA)	Customized to computer a particular function. May take hours to compile and synthetize

Central Processing Unit (CPU); Graphics Processing Unit (GPU); Field Programmable Gate Array (FPGA).

Software & Hardware Working Together

Domain-Specific Architectures (cont.)

Different from GP architectures, the hardware can be **specific** and **optimized** for a particular domain in DSA

This can be done in **runtime**

In this sense, **computers** will be more and more **heterogeneous**

Software & Hardware Working Together

Reconfigurable Computing

Reconfigurable computing: a hardware with **fixed** and **variable** parts

For a specific application requirement, *i.e.* **domain-specific**, and at given time frame

- the reconfigurable devices' spatial **structure** is **changed** to comply with a given objective [Bob07; Koc13]

Software & Hardware Working Together

Reconfigurable Computing (cont.)

SoC with **scalar** and **programmable logic** elements integrated in a single package → DSA

These architectures can be applied to the avionics systems domain, as far as they are able to provide the **safety** inherent from this domain

Possible solution → **models of computation** (MoC) + **runtime reconfiguration** = flexibility and performance with safety to future avionics systems

Software & Hardware Working Together

Formal Models of Computation

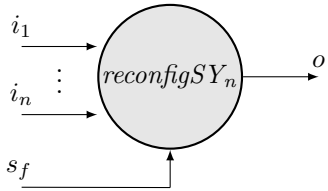
MoCs are based on [Jan03]:

- Event – **elementary information** unit exchanged between processes
- Signal – processes communicate to each other by writing to and reading from signals, which are a **sequence of events**. Signals preserve the order that events are entered. Each event has a *tag* and a *value*. Tags can be used to model physical time, events order, and other key properties of a MoC
- Process – receives and send events. The process activity is comprised of **evaluation cycles**, *i.e.* an application of a **function which maps inputs to outputs**. Then, in each evaluation cycle the process receives inputs, computes, and sends outputs

Software & Hardware Working Together

Formal Models of Computation (cont.)

Example of a synchronous (SY)[1] MoC reconfigurable process



Process constructor (higher-order function) for *reconfigSY* [SJ08]

$$ap_f = reconfigSY_n \tag{1}$$

where:

$$o = ap_f(s_f, (i_1, \ldots, i_n))$$
$$o[k] = s_f[k](i_1[k], \ldots, i_n[k])$$

[1] The synchronous (SY) MoC splits the time domain into slots

Software & Hardware Working Together

Examples

1. N-modular redundancy
   - safety-critical system → dependability
   - different modules implemented differently, w/ distinct hardware architectures
   - final output is the same
2. Reduced instruction set computers (RISC)
   - multiple designs for different specific computation
   - different RISC bitstreams in the memory
   - load/unload them in runtime using partial reconfiguration
3. Software-defined radio (SDR)
   - different implementations
   - using them depending on the present context to address challenges such as SWaP [2]

[2] size, weight, and power

Software & Hardware Working Together

Possible Benefits

- Hardware reuse resulting in a reduced circuit area
- Contributes to minimizing single event upset (SEU)
- Full control of the trade-off involving power consumption and performance

Challenges & Trends

1. **How to port software** (*e.g.* C/C++ and Ada) to different architectures
   - problem similar to the one faced back in the 80's
   - how to port code written in Assembly to different μC w/o the need to rewrite almost all of it
   - high level languages and hardware room?

2. Integrated modular avionics + **runtime hardware reconfiguration**
   - issue: how to ensure the reconfiguration is safely performed in a deterministic way?

Confidently, **DSA** together with formal **MoC** figure as a promising trend for the next years

Summary

- We presented some considerations on **domain-specific architectures** applicability to future avionics systems
- A research path is open concerning
  - more and more application of formal **models** in real-time embedded safety-critical systems, *i.e.* avionics
- Reconfigurable computing figure as one promising **underlying element** to achieve power-consumption and performance efficiency in domain-specific architectures

References

Christophe Bobda. Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications. Springer, 2007.

Andrew A. Chien. "Computer Architecture: Disruption from Above". In: Commun. ACM 61.9 (Aug. 2018), pp. 5–5. ISSN: 0001-0782. DOI: 10.1145/3243136. URL: http://doi.acm.org/10.1145/3243136.

Amit Gupta. "Advances in Adaptable Computing". In: Proceedings of the 2019 International Symposium on Physical Design. ISPD '19. San Francisco, CA, USA: ACM, 2019, pp. 37–38. ISBN: 978-1-4503-6253-5. DOI: 10.1145/3299902.3311064. URL: http://doi.acm.org/10.1145/3299902.3311064.

J. L. Hennessy and D. A. Patterson. Computer Architecture: A Quantitative Approach. 6th. Morgan Kaufmann, 2017.

Axel Jantsch. Modeling Embedded Systems and SoC's: Concurrency and Time in Models of Computation. 1st. San Francisco, USA: Morgan Kaufmann, 2003.

Dirk Koch. Partial Reconfiguration on FPGAs: Architectures, Tools and Applications. 1st ed. Springer-Verlag New York, 2013.

Denis S. Loubach. "A Runtime Reconfiguration Design Targeting Avionics Systems". In: 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). Sacramento, USA: IEEE, Sept. 2016, pp. 1–8.

Ingo Sander and Axel Jantsch. "Modelling Adaptive Systems in ForSyDe". In: Electronic Notes in Theoretical Computer Science 200.2 (2008). Proceedings of the First Workshop on Verification of Adaptive Systems (VerAS 2007), pp. 39 –54. ISSN: 1571-0661. DOI: http://dx.doi.org/10.1016/j.entcs.2008.02.011. URL: http://www.sciencedirect.com/science/article/pii/S1571066108000935.

Acknowledgments

- Research Development Foundation (*Fundacao de Desenvolvimento da Pesquisa)* - FUNDEP/MCTIC/MDIC
- *Ecossistema Negocios Digitais Ltda*
- *Casimiro Montenegro Filho Foundation* (FCMF)
- Brazilian Aeronautics Institute of Technology - ITA

Questions & Answers

Corresponding author:

Denis **Loubach**
Assistant professor at ITA
Department of Computer Systems
Computer Science Division – IEC

http://www.comp.ita.br/~dloubach
dloubach@ita.br