IEEE
Advancing Technology for Humanity

INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS™

2023 UPDATE

AUTONOMOUS MACHINE COMPUTING

THE IRDS™ IS DEVISED AND INTENDED FOR TECHNOLOGY ASSESSMENT ONLY AND IS WITHOUT REGARD TO ANY COMMERCIAL CONSIDERATIONS PERTAINING TO INDIVIDUAL PRODUCTS OR EQUIPMENT.

IEEE COMPUTER SOCIETY | IEEE | IEEE rebooting COMPUTING | IEEE COMPUTER SOCIETY TFRC Task Force on Rebooting Computing | IRDS

THE INTERNATIONAL ROADMAP FOR DEVICES AND SYSTEMS: *2023*
COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

IEEE International Roadmap for Devices and Systems (IRDS)

Autonomous Machine Computing

White Paper 2023

Shaoshan Liu[1] *Senior Member IEEE*
Jean-Luc Gaudiot[2] *Life Fellow IEEE*

[1]PerceptIn Inc
[2]University of California, Irvine, U.S.A.

1. Introduction

In the past few decades, the semiconductor industry has been a key contributor to our economy. The latest statistics from the Semiconductor Industry Association (SIA) indicate that global semiconductor industry sales totaled $556 billion in 2021. In addition to the direct impact on the economy, the semiconductor industry is the key engine that powers all information technology fields, including personal computing, mobile computing, cloud computing, *etc*. and therefore has a much deeper and broader impact than expected at first sight on our modern economy.

Our central argument is that **advancing semiconductor technologies will enable more interesting applications, hence fostering an ecosystem with a market size that is tens of times bigger than the enabling semiconductor sector** [1]. One concrete example (Figure 1) is the mobile computing industry: back in the early 2000s, mobile phones had already become commodities and were called feature phones. Although feature phones were almost ubiquitous, they possessed limited functions or applications and 90% of the compute power was dedicated to basic communication functions such as encoding and decoding. Since less than 10% of the available compute power was dedicated for application, this did not leave much room for many interesting applications. The whole market size of the mobile computing ecosystem was around $10 billion. Then smart phones came to market and generated ever increasing demands for compute power in order to sustain an ever-increasing number of interesting mobile applications. As demand increased, a single embedded chip evolved into complex mobile systems-on-chip consisting of multi-core CPUs, mobile GPUs, and mobile DSPs, along with extremely good power management systems. Today, the mobile computing ecosystem enjoys a market size of $800 billion.

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

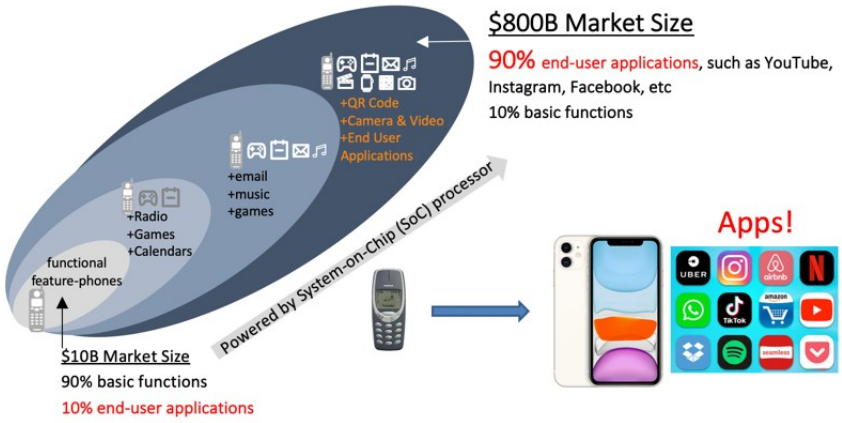

Figure 1: evolution of mobile computing

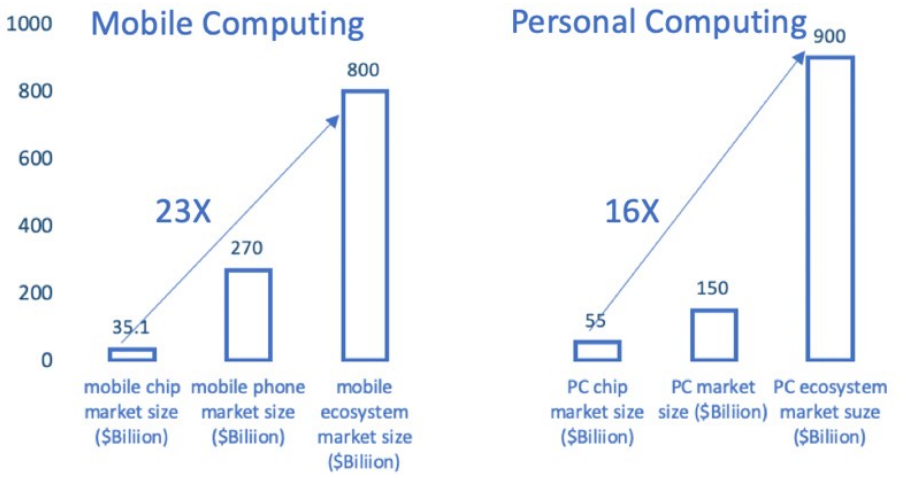

Figure 2: ecosystem vs semiconductor market size

As the personal computing market and the mobile computing market have matured, let us examine **the relation between the ecosystem market size and the semiconductor market size** (Figure 2) for both personal computing and mobile computing. Today, the mobile processor market size is $35 billion, the mobile phone market size is $270 billion, while the mobile computing ecosystem market size is $800 billion, 23 times the size of the mobile processor market. The personal computing processor market size is $55 billion, the personal computer market size is $150 billion, and the personal computing ecosystem market size is $900 billion, 16 times the size of the personal computing processor market. **Thus, as a computing era matures, the semiconductor sector should be capable of supporting an ecosystem 15 to 25 times its market size.**

Next, let us examine the current status of autonomous machine computing (AMC) and consider the steps required to initiate the virtuous circle of growing an AMC ecosystem: the current AMC compute system designs spend 50% of autonomous machine compute power for perception, 20% of compute power for localization, 25% of compute power for planning, leaving merely 5% for applications. This is very similar to the situation in feature phones in the early stage of the mobile computing era. With this distribution of compute power, it is impossible to have the autonomous machines perform intelligent tasks, **which means that, currently, the AMC ecosystem is virtually non-existent.**

What we aim for is that the basic operations including perception, localization, and planning should be done in hardware and consume less than 20% of compute power, thus leaving 80% of compute power for intelligent applications. **It is the only way to unlocking the imagination of software developers and enabling an ecosystem for AMC applications**. For instance, one interesting application of AMC is the autonomous mobile clinics [2], which will be capable of solving the healthcare access and equity problem by delivering healthcare services to the patient's

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

doorstep by order of the patient. Similarly, many other interesting autonomous machine applications can be defined to expand the AMC ecosystem.

In this white paper, we examine three key research directions for autonomous machine computing, including Defining Architectures for Autonomous Machine Computing (Section 2), Programming Autonomous Machines with Only A Few Lines of Code (Section 3), and Linking Systems and Technologies (Section 4).

2. Defining Architectures for Autonomous Machine Computing

To fully realize the economic potential of the AMC ecosystem detailed in section 1, the key technical challenge is to enable efficient programming of AMC applications, thus an abstraction between the application programming layer and the underlying compute substrates is imperative. **This part aims to define a common computer architecture for AMC and to popularize it to become a global standard, similar to the ARM architecture in the mobile computing era**. In the past five years, we have explored and identified three promising architecture candidates, including dataflow accelerator architecture (DAA), factor graph architecture (FGA), and learning-based architecture (LBA). We plan to leverage on the advantages of each architecture proposal to derive a common abstraction for AMC [3].

2.1. Dataflow Accelerator Architecture

The central idea of dataflow architecture was to do away with the classic von Neumann architecture, where instructions are executed in the explicit order specified by the control flow. Control flows limit the window in which the instruction-level parallelism (ILP) can be exploited, presenting an artificial performance roadblock. In a dataflow architecture, execution is data-driven in that an instruction, in principle, executes as soon as all its inputs are available rather than when the control flow gets to it.

We see an analogy between the bottlenecks in conventional programs and those in autonomous machines software, both of which can be addressed by the dataflow principle. **The key for this abstraction is to view the software stack of an autonomous machine, such as the autonomous vehicle computation graph (Figure 3), as a macro dataflow graph (M-DFG),** where each node represents a high-level task such as localization and motion planning.

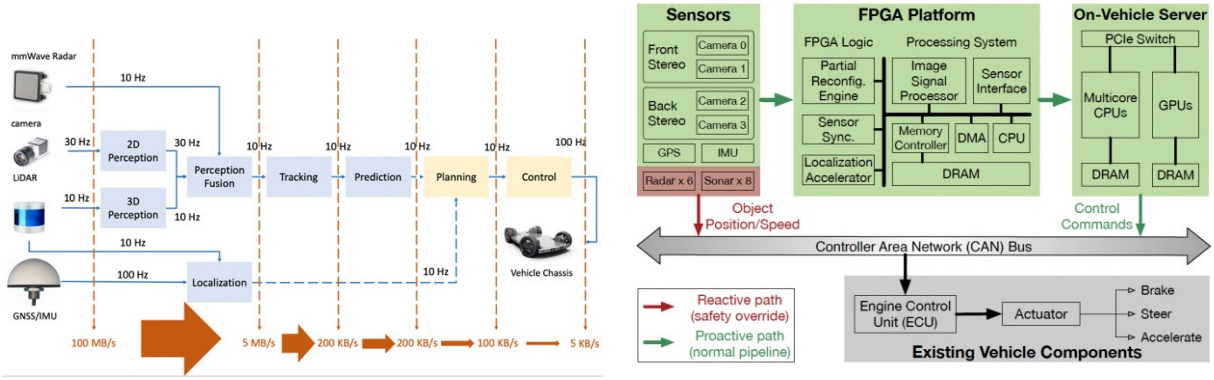

Figure 3: M-DFG of an Autonomous Vehicle and a commercial instantiation ([5])

This realization gives rise to the notion of **dataflow accelerator architecture (DAA), where accelerators directly communicate with each other through dedicated on-chip buffers and coordinate autonomously.** This architectural model has two benefits. First, it exposes higher levels of parallelisms with each accelerator (M-DFG node) firing whenever its input data are ready. Second, it accelerates the firing rates of M-DFG nodes by making operands more readily available to consumers; this is achieved by allowing producers and consumers to directly communicate using a per-accelerator on-chip buffer rather than through the main memory. (**Initial Results**: [4][5][6][7])

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

2.2. Factor Graph Architecture

A factor graph is a graph representing the factorization of a probability distribution function and has been utilized in many AMC functions, such as localization, tracking, planning and control. Currently, researchers around the world are developing algorithms to use factor graph as a common abstraction for AMC functions. The factor graph abstraction brings several benefits: first, it provides a concise representation of robotic optimization problems to allow autonomous machine programmers to construct their programs. Second, its graph structure facilitates sparse data storage. Third, the robotic optimization problems can be solved incrementally based on historical information, thus guaranteeing high accuracy and low computation latency. **We identify factor graph as a common abstraction for many of autonomous machine basic functions, and thus a common factor graph accelerator can be utilized to accelerate many autonomous machine tasks and greatly simplifies the programming interface.** (**Initial Results**: [8][9])

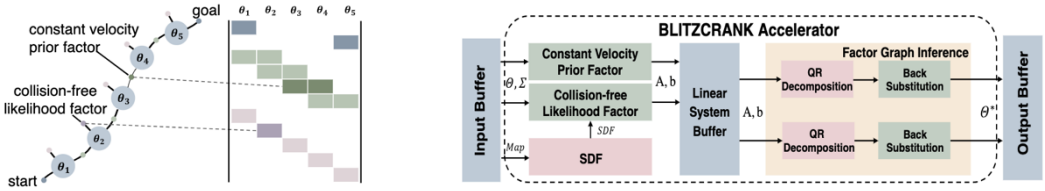

Figure 4: a factor graph for motion planning and an instantiation on FPGA ([8])

2.3. Learning-Based Architecture

As many autonomous machine computing tasks are accomplished using learning-based techniques (e.g. transformer models), AMC is evolving from Autonomy 1.0, or a modular approach, to Autonomy 2.0, an end-to-end learning-based approach. In Autonomy 2.0, any autonomous machine performs two main tasks: perception and action, reflecting the natural dichotomy of the past and the future. The perception module is trained using supervised learning and self-supervised learning to infer one unique ground truth of world states. In contrast, the action module needs to search and choose from many acceptable action sequences, while anticipating the behaviors of other agents. Therefore, the action module makes use of methods from reinforcement learning, imitation learning, and model predictive control. Interestingly, while the fundamental distinctions of the two modules have not changed in Autonomy 2.0, there is a growing convergence of the implementation of the two modules: recent successes of large language models (LLM) to comprehend a large amount of information to perform multiple sub-tasks suggest that both modules can be implemented using a similar architecture based on Transformer. Transformer is a great algorithmic substrate for both the perception and action modules because of its ability to generalize. For perception, a transformer can effectively fuse perceptual data from multiple sensors and multiple moments into a unified representation, avoiding information loss from sparsification and module serialization. For action, the sequential nature of transformer makes it a perfect fit for processing and generating temporal data, especially for sampling multiple possible future paths. **Hence, a common architecture abstraction is to focus on accelerating the transformer workloads and simplifies the perception and action tasks to different forms of transformation computation, which greatly simplifies the programming interface.** (**Initial Results**: [10][11][12])

3. Programming Autonomous Machines with Only A Few Lines of Code

Programming autonomous machines today is extremely challenging due to the lack of good system abstractions and lack of a good runtime system to manage real-time constraints as well as system resource allocation. As a result, programmers have to possess knowledge in autonomous machine applications, autonomous machine algorithms, and computing systems. To liberate programmers from the systems details, **this part aims to develop a programming**

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

and runtime system on top of the defined architecture to allow programmers to develop various autonomous machines with only a few lines of code.

Particularly, autonomous machines currently rely on a wealth of specialized components according to the various tasks they are required to perform, which includes (hard) real-time tasks related to the outside environment, *i.e.*, Localization and Navigation, Object Detection and Avoidance, *etc.* Each task communicates its data to other tasks within a strict performance envelop, and may also rely on very different hardware targets, *e.g.*, scheduling with CPUs, GPUs for neural network processing, FPGAs or DSPs for image processing, *etc.* Thus, there should be **expressive "languages" to describe, at a high level, what each task should consist of, the appropriate (domain-specific) semantics for it, and at the same time, the interfacing between languages**. As a whole, real-time and run-time contexts can freely dictate which part of the underlying hardware will be used to run them at a specific moment during the motion of the target autonomous machine. To do so, simply designing a new set of DSLs is probably necessary, but insufficient. As these autonomous machines tend to heavily rely on Machine Learning techniques for at least some of their tasks, there should also be a way to lower the high-level description, semantic and performance expectation, provided for each task down to an intermediate representation which would allow the compiler to produce code for a family of heterogeneous devices.

We have developed a prototype, which is a concise and precise high-level language to represent the computation graphs of autonomous machines. As autonomous machine computing can be represented as dataflow graphs, naturally a functional programming paradigm provides an efficient way to describe the behavior of autonomous machines. With functional programming, autonomous machine application programmers can describe their applications with a few lines of specifications and code as illustrated in the following examples. (**Initial Results**:[13][14][15])

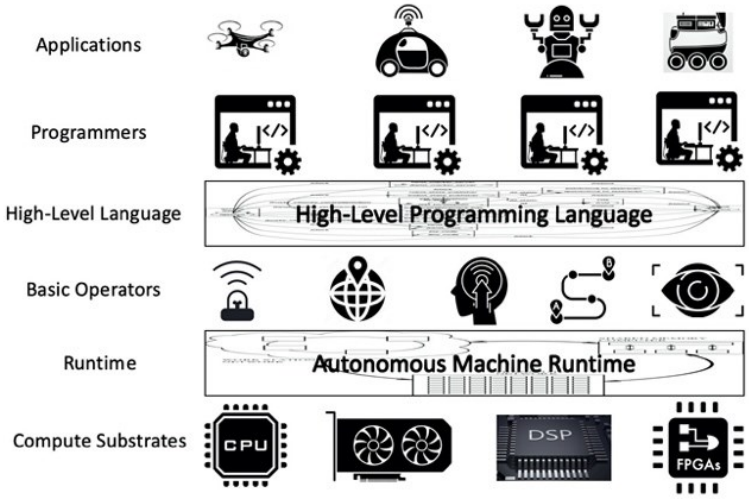

Figure 5: Programming Autonomous Machines ([13])

4. Linking Systems and Technologies

Advancements in the semiconductor technologies will extend the capabilities of autonomous machines in terms of sensing, computing, and communication, and thus empowering more emerging AMC applications. **this part aims to link emerging technologies with AMC system architectures, particularly in the following areas**:

- Sensing: sensing in 2D and 3D is essential to any autonomous machine. Recent advancements in die-stacked image sensors allows high-level computations (e.g., DNNs) to take place directly inside an image sensor, drastically reduce the data transfer cost. Single-Photon Avalanche Diode (SPAD), meanwhile, has the potential to capture real-time 3D scene information while mitigating many of the downsides of mechanical LiDARs.
- Computation: The key technical challenge of designing an autonomous machine computing system is to execute the complex computation graphs from different types of autonomous machines while meeting the real-time performance, cost, and energy constraints. We argue that the defined AMC architecture, when coupled

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

with emerging technologies such as multi-die chiplet designs, heterogeneous integration, processing-in-memory, and optical-analog-digital co-design will enable truly real-time AMC.

- Communication: while traditional autonomous machines utilize only on-machine intelligence, future autonomous machines will heavily rely on the cooperation between autonomous machines, edge servers, and the cloud infrastructure [16]. The key technical challenge for the cooperative autonomous machine paradigm is communication. To address this problem, InP/InGaAs has the potential to significantly improve the communication bandwidth and overcome the latency limitation. (**Initial Results**: [17][18][19])

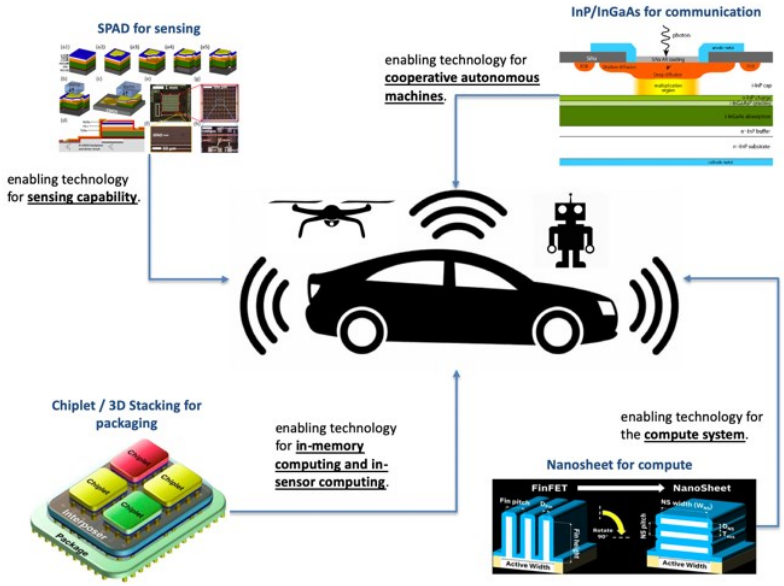

Figure 6: linking systems and technologies

References

1. https://www.forbes.com/sites/forbestechcouncil/2022/08/25/growth-of-the-autonomous-machine-computing-ecosystem-driven-by-semiconductor-innovations/, Forbes
2. Liu, S., Kong, A., Huang, Y. and Liu, X., 2022. Autonomous mobile clinics. Bulletin of the World Health Organization, 100(9)
3. Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y. and Shi, W., 2019. Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE, 107(8).
4. Liu, S., Tang, J., Zhang, Z. and Gaudiot, J.L., 2017. Computer architectures for autonomous driving. Computer, 50(8)
5. Yu, B., Hu, W., Xu, L., Tang, J., Liu, S. and Zhu, Y., 2020, October. Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations. In 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)
6. Gan, Y., Bo, Y., Tian, B., Xu, L., Hu, W., Liu, S., Liu, Q., Zhang, Y., Tang, J. and Zhu, Y., 2021, February. Eudoxus: Characterizing and accelerating localization in autonomous machines industry track paper. In 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)
7. Liu, W., Yu, B., Gan, Y., Liu, Q., Tang, J., Liu, S. and Zhu, Y., 2021, October. Archytas: A framework for synthesizing and dynamically optimizing accelerators for robotic localization. In MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)
8. Hao, Y., Gan, Y., Yu, B., Liu, Q., Liu, S.S. and Zhu, Y., 2023, July. BLITZCRANK: Factor Graph Accelerator for Motion Planning. In 2023 60th ACM/IEEE Design Automation Conference (DAC).
9. Hao, Y., Yu, B., Liu, Q., Liu, S. and Zhu, Y., 2022, October. Factor Graph Accelerator for LiDAR-Inertial Odometry. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD)
10. Wu, S., Yu, B., Liu, S. and Zhu, Y., 2023. Autonomy 2.0: The Quest for Economies of Scale. arXiv preprint arXiv:2307.03973.
11. Guan, H., Liu, S., Ma, X., Niu, W., Ren, B., Shen, X., Wang, Y. and Zhao, P., 2021. CoCoPIE: Enabling real-time AI on off-the-shelf mobile devices via compression-compilation co-design. Communications of the ACM, 64(6).

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.

12. Yu, B., Tang, J. and Liu, S.S., 2023, July. Autonomous Driving Digital Twin Empowered Design Automation: An Industry Perspective. In 2023 60th ACM/IEEE Design Automation Conference (DAC) IEEE.
13. Liu, S., Li, X., Geng, T., Zuckerman, S. and Gaudiot, J.L., 2022, October. Programming Autonomous Machines: Special Session Paper. In 2022 International Conference on Embedded Software (EMSOFT).
14. Jiang, X., Luo, X., Guan, N., Dong, Z., Liu, S. and Yi, W., 2023, April. Analysis and Optimization of Worst-Case Time Disparity in Cause-Effect Chains. In 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)
15. Liu, L., Tang, J., Liu, S., Yu, B., Xie, Y. and Gaudiot, J.L., 2021. π-rt: A runtime framework to enable energy-efficient real-time robotic vision applications on heterogeneous architectures. Computer, 54(4)
16. Liu, S., Yu, B., Tang, J. and Zhu, Q., 2021, December. Towards fully intelligent transportation through infrastructure-vehicle cooperative autonomous driving: Challenges and opportunities. In 2021 58th ACM/IEEE Design Automation Conference (DAC)
17. Liu, S., Yu, B., Tang, J., Zhu, Y. and Liu, X., 2022. Communication challenges in infrastructure-vehicle cooperative autonomous driving: A field deployment perspective. IEEE Wireless Communications, 29(4).
18. Feng, Y., Liu, S. and Zhu, Y., 2020, October. Real-time spatio-temporal lidar point cloud compression. In 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)
19. Zhang, Z., Liu, S., Tsai, G., Hu, H., Chu, C.C. and Zheng, F., 2018, May. PIRVS: An advanced visual-inertial slam system with flexible sensor fusion and hardware co-design. In 2018 IEEE International Conference on Robotics and Automation (ICRA)

Author Information:

SHAOSHAN LIU is the founder and CEO of PerceptIn Inc, an intelligent transportation company. Shaoshan Liu's technical research focuses on Autonomous Driving technologies and Robotics. He has published 90 research papers, 40 U.S. patents, and over 150 international patents. Dr. Shaoshan Liu have published three books on robotics technologies "Creating Autonomous Vehicle Systems (Morgan & Claypool)", "Engineering Autonomous Vehicles and Robots: The DragonFly Modular-based Approach (Wiley - IEEE)", and "Robotic Computing on FPGAs (Morgan & Claypool)" Dr. Shaoshan Liu was a founding member of Baidu USA, where he built the Autonomous Driving Systems team, and performed R&D work at LinkedIn, Microsoft, Microsoft Research, INRIA, Intel Research, and Broadcom. He is a senior member of IEEE, a Distinguished Speaker of the IEEE Computer Society, a Distinguished Speaker of the ACM, and a founder of the IEEE Special Technical Community on Autonomous Driving Technologies. Dr. Shaoshan Liu received Master of Public Administration (MPA) from Harvard Kennedy School, Ph.D. in Computer Engineering from University of California, Irvine. Contact him at shaoshan.liu@perceptin.io

JEAN-LUC GAUDIOT is a professor in the Department of Electrical Engineering and Computer Science at the University of California, Irvine, California, 92697, USA. His research interests include multithreaded architectures, fault-tolerant multiprocessors, and the implementation of reconfigurable architectures. Gaudiot received a Ph.D. degree in computer science from the University of California, Los Angeles. Gaudiot served as the 2017 IEEE Computer Society President. He is a Fellow of IEEE and the American Association for the Advancement of Science. Contact him at gaudiot@uci.edu

COPYRIGHT © 2023 IEEE. ALL RIGHTS RESERVED.