Giuseppe Di Guglielmo

Associate Research Scientist
Columbia University
✉ giuseppe@cs.columbia.edu
http://www.cs.columbia.edu/ giuseppe

Research Statement

In the last decade, we have observed the rise of two significant paradigms that have rapidly proliferated throughout the scientific domains: heterogeneous computing and machine learning. Heterogeneous computing offers a solution to the decline of Moore's Law and Dennard Scaling by replacing homogeneous cores with custom, heterogeneous cores, known as accelerators. Heterogeneous computing enables each portion of the computation to be executed with the specialized hardware best suited to optimize its performance and cost. Machine learning (ML) is a semiautomatic process that creates programs capable of solving broad classes of problems, such as regression and classification. As with traditional programming, machine learning can benefit significantly from heterogeneous computing, especially with specialized hardware tailored for the ML problem space. In my research, I aim to architect and promote an automated design process for ML accelerators to optimize the performance and cost of groundbreaking scientific applications.

Previous Research Experience

In my Ph.D. thesis, I proposed a set of automatic-test pattern generation (ATPG) methodologies for the functional validation of embedded systems [C6, TH2]. The importance of validation and verification (VV) for shipping successful hardware systems is undeniable. While the purpose of verification is to ensure that the design was manufactured correctly, validation aims to ensure that the design meets its functional and extra-functional intent before manufacturing. My research was particularly focused on the functional validation of digital systems in particular. Functional validation is the process that detects and addresses the designer's errors to ensure that the hardware's logical design satisfies the specification. As digital system complexity increases with each generation, the VV problem has become more challenging. The trend is even more pronounced for embedded systems that combine a heterogeneous mix of hardware and software modules. Hardware design teams dedicate between 60% and 80% of their efforts to VV.

The ATPG methodologies proposed in my Ph.D. are "concolic" in nature in that they combine concrete and symbolic simulation of the design specifications made in languages such as VHDL/Verilog and C/SystemC [C18]. The symbolic simulation solves for inputs that lead along specific hard-to-traverse paths. When the traversal problem becomes infeasible for the solver to handle, concrete values are fed to run the design simulation. This solution allows the traversal procedure to cover parts of the design specifications that could not be reached through either pure symbolic methodologies or randomly-generated stimuli. In addition, during my Ph.D. I investigated (1) design models to represent the design under test [J1, J3]; (2) fault models to guide the stimuli generation at a higher level of abstraction together with metrics to evaluate the generated stimuli [J5]; and finally (3) efficient simulation engines [C8].

As a Postdoctoral Researcher, I extended my research interests from hardware to designing and validating software for embedded systems. The importance of specification definition in the embedded-software design flow has been proven over the years. The entire process relies on the accuracy of the design specification, which is often ill-defined and prone to human error. I investigated the automatic mining of functional properties to simplify specification definition for designers. At the time, the existing approaches were limited to detecting either arithmetic, non-temporal invariants of programs or temporal properties limited to the Boolean domain. I defined the first dynamic mining approach that can infer linear temporal logic (LTL) properties for embedded software. The resulting mined properties extend temporal relationships to arithmetic expressions beyond the Boolean domain [C23]. Since my approach considers the execution traces only, it does not require static analysis and is entirely independent from the code implementation. I have also extended this software approach to the behavioral specification of hardware components [C27].

Later on, I focused on (1) model-driven design (MDD) to elevate software design to a higher abstraction level than traditional programming languages and (2) assertion-based verification (ABV) to define temporal properties for the design functional verification. Both MDD and ABV had been adopted as effective methodologies for embedded hardware design and verification. However, at the time, both MDD and ABV had limitations that were preventing their integration into the software design and verification flow. In particular, MDD requires integrating a practical methodology for monitoring specification conformance, and dynamic ABV relies on simulation assumptions that were satisfied in the hardware domain but were not easily guaranteed during the software execution. I proposed a suitable combination of MDD and dynamic ABV as an effective software design and verification solution. In collaboration with both academic and industrial partners, I worked on a suite of tools to support this integrated approach [J6].

Current Research Activity

As an Associate Research Scientist at Columbia University, I have contributed to the development and dissemination of Embedded Scalable Platforms (ESP), a new system-on-chip platform that combines a heterogeneous computing architecture with a companion system-level-design methodology [C38, C45, T4-T8].

The heterogeneous system-on-chip (SoC) has emerged as the most critical computing platform across many application domains, from embedded systems to data centers. The natural progression towards heterogeneous SoC results from the end of Dennard's ideal CMOS scaling. Unable to continue scaling down supply voltages, the integrated-circuit designers made two consecutive moves. First, they moved towards parallelism by building homogeneous multicore architectures that integrated multiple, relatively simple processor cores instead of a single, more complex, hyper-pipelined superscalar processor. Next, they moved towards heterogeneity by combining the processor cores with an increasing number of specialized-hardware accelerators, each capable of executing a dedicated function in a way that is orders of magnitude more efficient than its corresponding software execution.

It is within this current paradigm of heterogeneous computing that ESP was conceived. The ESP architecture addresses the complexity of component integration by balancing hardware specialization and design regularity with a tile-based approach. The ESP methodology seeks to increase productivity by moving the bulk of the engineering effort to the system level and reducing the gap between hardware design and domain expertise. Building on top of the ESP methodology, my research activity develops around four topical areas: the design of hardware accelerators, the accelerator design-space exploration with high-level synthesis, the security and verification of heterogeneous SoCs, and the acceleration of ML applications.

Researchers in academia and in industry have developed many different accelerators and accelerator-rich architectures to obtain energy efficiency and high performance in embedded applications. As part of my research on ESP, I have contributed to the definition of loosely-coupled accelerators for heterogeneous SoCs [C30, C31, C34]. A loosely-coupled accelerator can achieve better performance than processor cores thanks to specialized micro-architectures for both the accelerator logic and the local memories. The accelerator logic can exploit spatial parallelism to execute multiple operations in parallel. Additionally, while processor memories are designed for sequential access – even in the case of memory sharing with the accelerator – local memories can be customized to allow the accelerator logic to perform multiple memory operations within the same clock cycle and increase the hardware parallelism [C28, C29, J7].

In ESP, we also leverage high-level synthesis (HLS) to expedite the design of accelerators, improve the process of design-space exploration (DSE), and promote the reuse of accelerators across different target SoCs. HLS enables the automatic generation of hardware designs from high-level specifications given in languages such as C/C++ or SystemC. With HLS, designers can specify complex functionalities by working at a higher abstraction level than the register-transfer level. HLS tools offer a powerful set of parameters, known as knobs, to optimize an accelerator's architecture and evaluate different trade-offs in terms of performance and cost. However, exploring a large region of the design space and identifying a rich set of Pareto-optimal implementations are still complex tasks. I have co-authored papers that define strategies to discover the most effective implementations from a cost and performance perspective, while minimizing the number of HLS runs [C35, J8, J12].

The ESP methodology also extends to the field of security. Software-based attacks can exploit security vulnerabilities or bugs in software applications, obtain unauthorized control of applications, and inject malicious code. Dynamic information flow tracking (DIFT) has been proposed as a promising security technique to protect systems against software attack. DIFT monitors the hardware operations to detect suspicious data flows during the application execution to ensure that vulnerabilities are not exploited and do not cause a security violation. Most of the approaches on hardware-based DIFT focus only on securing processor cores and the associated logic, memories, and communication channels. However, loosely coupled accelerators are vulnerable to such kinds of attacks as well. To address this vulnerability, I collaborated on a methodology to extend the support of DIFT to loosely-coupled accelerators in heterogeneous SoCs [C36, C43, J9].

In recent years, the design of specialized accelerators for machine learning has become the primary trend across all computing systems. While the initial focus was mostly on systems in the cloud, the demand for applying machine learning approaches to edge devices continues to grow. To date, most research efforts have focused on the accelerator design in isolation, and rarely on integrating the accelerator design into a complete SoC. I have worked on a complete system-level design flow which leverages many heterogeneous accelerators in order to implement SoCs for embedded applications [C41]. I have also collaborated on the realization of a proof-of-concept system architecture, showing the functionality of photonic switched optically connected memory for large networks in deep learning [J10]. Finally, I have investigated the design of ML accelerators in FPGA systems equipped with High-Performance Memory (HPM) [C44].

Future Research Goals

Machine learning has led to discoveries in various scientific disciplines, including biology, climate research, and physics. Although the initial catalysts were the improved computational capabilities and the release of large benchmark datasets, the recent adoption of machine learning in various scientific applications has been fueled by well-documented open-source ML frameworks, such as TensorFlow, Caffe, and PyTorch, as well as by educational materials, such as Fast.ai and Coursera. Armed with these tools, domain scientists with minimal computational training can create highly performant models.

Significantly less attention has been focused on deploying machine learning in scientific practice, where figures of merit are the latency per inference, computational cost, reliability, security, and ability to operate in extreme environments. Some examples of application domains are: the trigger acquisition systems for rare events such as the Large Hadron Collider where the latency of the sensing system is less than 25ns; the ambulatory-health monitors at kHz frequencies where wireless-transfer of data is not possible due to power limitations or security requirements; and the processing of data streams from materials spectroscopy and quantum-computing, which are on the order of Tb/s. Computing-service companies provide scalable and accessible cloud and on-premises computation solutions, yet, there are unresolved problems for scientific applications. Integrating machine learning with sensing systems is difficult and results in inefficient and costly data transfer. Additionally, some ML applications are constrained to operate with a limited energy supply. Finally, existing hardware cannot be customized for ultra-low latency applications. Tools to design custom ML devices require in-depth knowledge of custom programming languages and hardware, and thus are only used by engineers. These tools currently do not support the rapid prototyping required by scientific-domain experts. The core problem is the decoupling of ML tools and hardware design automation tools.

My research aims at developing and promoting an automated design process for ML applications with a multi-objective optimization goal targeting performance, latency, energy, reliability, and security. As an alternative to the distributed computing paradigm, I will focus my attention on edge computing. Many key advantages come with this choice for scientific applications. Edge computing brings efficient computing and memory as close as possible to the information source in order to reduce latency. It also improves efficiency and scalability thanks to customized ML hardware. Finally, it guarantees reliability and security via dedicated hardware that does not rely on networks and data centers.

To tackle these challenges, I have joined two novel research communities to broaden my expertise and understand domain-scientist problems: FastML and tinyMLPerf. FastML is a group of physicists, engineers, and computer scientists interested in deploying ML algorithms for unique and challenging scientific applications. tinyMLPerf is a working group for the fast-growing field of ML technologies and applications capable of performing on-device computation at extremely low power. Both of these communities are part of the recent Open Source Hardware movement that fosters technological knowledge and encourages research that is accessible, collaborative, and respectful of user freedom.

In this context of multi-disciplinary collaboration, I am actively contributing to the hls4ml project, a framework for co-design and optimization of ML models on customizable devices. We have already seen significant results in supporting with hls4ml a variety of models such as boosted decision trees, distance-weighted graph neural networks, large convolutional neural networks and binary/ternary compressed neural networks [U4,U6,J11,J13]. I intend to make hls4ml the platform for my research activity to bridge the gap between software tools for scientific model training and design automation for ML hardware. hls4ml will provide utilities for ML device co-design under practical performance, resource, latency, and power constraints. It will also automate the construction of intellectual property blocks for deployment on FPGAs and ASICs. The power of hls4ml lies in its ability to improve the performance-cost trade-off of ML models and compile such models for hardware.

To summarize, my primary research goal is to develop an open-source scientific framework which combines ML tools with hardware design automation. I have pursued this research to date by investigating the design and verification of the hardware acceleration for heterogeneous computing. In the future, I would like to offer my expertise to the scientific communities deploying ML models for their research needs.

Publications

Complete references to the publications cited in this statement are available in the attached CV.