Towards On-Chip Learning for Low Latency Reasoning with End-to-End Synthesis

Vito Giovanni Castellana
Nicolas Bohm Agostini[*]
Ankur Limaye, Vinay Amatya, Marco Minutoli,
Joseph Manzano, Antonino Tumeo
Pacific Northwest National Laboratory
Richland, WA, USA

Serena Curzel, Michele Fiorito
Fabrizio Ferrandi
Politecnico di Milano
Milano, Italy

ABSTRACT

The Software Defined Architectures (SODA) Synthesizer is an open-source compiler-based tool able to automatically generate domain-specialized systems targeting Application-Specific Integrated Circuits (ASICs) or Field Programmable Gate Arrays (FPGAs) starting from high-level programming. SODA is composed of a frontend, SODA-OPT, which leverages the multilevel intermediate representation (MLIR) framework to interface with productive programming tools (e.g., machine learning frameworks), identify kernels suitable for acceleration, and perform high-level optimizations, and of a state-of-the-art high-level synthesis backend, Bambu from the PandA framework, to generate custom accelerators. One specific application of the SODA Synthesizer is the generation of accelerators to enable ultra-low latency inference and control on autonomous systems for scientific discovery (e.g., electron microscopes, sensors in particle accelerators, etc.). This paper provides an overview of the flow in the context of the generation of accelerators for edge processing to be integrated in transmission electron microscopy (TEM) devices, focusing on use cases from precision material synthesis. We show the tool in action with an example of design space exploration for inference on reconfigurable devices with a conventional deep neural network model (LeNet). Finally, we discuss the research directions and opportunities enabled by SODA in the area of autonomous control for scientific experimental workflows.

CCS CONCEPTS

• **Hardware → High-level and register-transfer level synthesis**.

KEYWORDS

High Level Synthesis, Design Automation, Machine Learning, Neural Networks, Edge Computing

ACM Reference Format:
Vito Giovanni Castellana, Nicolas Bohm Agostini, Ankur Limaye, Vinay Amatya, Marco Minutoli, Joseph Manzano, Antonino Tumeo, Serena Curzel, Michele Fiorito, and Fabrizio Ferrandi. 2023. Towards On-Chip Learning for Low Latency Reasoning with End-to-End Synthesis. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3566097.3568360

[*]Also with Northeastern University.

1 INTRODUCTION

Modern experimental instruments, such as electron microscopes, mass spectrometers, particle accelerators, and more, acquire unprecedented volumes of multimodal data produced at high velocity [1]. These instruments cannot store all the data locally and need to process them in real time to enable autonomous control of the experimental workflow. Domain scientist may need to set up a large variety of very specific experiments to collect heterogeneous data, often at the same time, and with limited opportunities to repeat the experiments. Machine learning and, more in general, artificial intelligence methods promise the ability to analyze and efficiently classify the input data, also in presence of noise. However, as data rates and features to capture increase, so does the complexity of the models, making it difficult to reach the low processing latency required to perform decisions on-the-fly without highly specialized accelerators. Additionally, beside performing in situ analysis, there might be the need to reduce and ship the acquired data to large-scale high-performance computing services to perform further analyses, execute scientific simulation, and re-train models in a closed loop. While able to reach high peak computational rates, heterogeneous solutions composed only of general-purpose processing elements (central processing units and graphic processing units) typically are better suited to process large batches of data, optimizing the overall throughput, rather than optimizing the latency in continuously streaming inputs.

Domain scientists typically develop their data analysis algorithms in high-level productive programming frameworks, which trade-off productivity for performance. With the conventional, and extremely time expensive, approach, expert hardware designers implement specialized accelerators that match the most common computational patterns with hardware design languages (HDLs) at the register transfer-level (RTL). In machine learning and artificial intelligence, new methods and models are developed all the time. If algorithms change, the main computational patterns may also change, rendering all previously designed accelerators inefficient. The semiconductor manufacturing costs also require significant volumes to justify the production of application specific integrated circuits (ASICs), making reconfigurable devices such as field programmable gate arrays (FPGAs) for these highly critical sectors more appealing today, at the price a lower overall performance. New design automation tools, able to generate specialized hardware accelerators automatically, or with very limited human intervention, starting from the high-level specification of the algorithms are required to bridge this hardware productivity gap.

Conventional High-level Synthesis (HLS) tools, which are today provided by many companies that design FPGAs (AMD Xilinx and Intel Altera), or provide design automation tools (Synopsys, Cadence, Mentor) typically start from languages such as C or C++, with tool-specific libraries and annotations that mix the imperative description of the algorithms with hardware specific information. While they remove the need to write HDL code, they still need expert developers to finely tune and hand optimize the code to obtain high quality of results. Approaches that leverage these tools to generate accelerators from higher-level programming frameworks (e.g., in Python) typically resort in specializing HLS code templates, providing a higher abstraction than HDL code templates, but still significantly limiting the number of mappable computational patterns in name of performance. However, high-level programming frameworks usually are functional in nature, thus provide many opportunities to perform automated and transparent optimizations.

To address some of these gaps, we introduced the Software Defined Accelerators (SODA) Synthesizer [3], an open-source, modular, and extensible end-to-end compiler-based framework for generating highly specialized hardware accelerators from algorithms designed in high-level programming frameworks. SODA is composed of SODA-OPT [2], a frontend based on the MultiLevel Intermediate Representation (MLIR) [11] framework, that interfaces with high-level programming frameworks and applies high-level optimizations, and PandA-Bambu [8], a state-of-the-art HLS tool that generates hardware designs in HDL (Verilog). SODA interfaces with external logic synthesis tools for FPGAs and ASICs.

In this paper, we discuss how the SODA Synthesizer can provide a fundamental research infrastructure to generate custom accelerators and systems for low latency reasoning and autonomous control, focusing on the unique case of experimental scientific instruments. We provide a brief overview of the current SODA framework, present the use case of precision material synthesis in terms of requirements and opportunities for autonomy, show preliminary results with the framework to explore design tradeoffs with a common autoencoder model, and discuss future research directions to enable in situ on-chip learning.

2 THE SODA SYNTHESIZER

The SODA Synthesizer (Figure 1a) consists of two major components: i) SODA-OPT[2], the frontend compiler for system-level partitioning and high-level optimizations, and ii) PandA-Bambu [8], a state-of-the-art HLS tool.

Inputs of the SODA toolchain are algorithm descriptions written in high-level programming languages and frameworks, such as Python. The framework generates the related hardware accelerators described in Verilog RTL, synthesizable to FPGA or ASIC targets.

The high-level specification provided to the compilation pipeline is translated into a high-level intermediate representation (IR) in the early stages of the high-level optimizer. This IR actually consists of several *dialects* (i.e., specialized IRs derived from the same meta-IR) of the Multi-Level Intermediate Representation (MLIR) [11].

SODA-OPT performs hardware/software partitioning of the input program and architecture-independent optimizations by leveraging MLIR features. SODA-OPT generates two different types of LLVM IR outputs. The first one is an optimized LLVM IR file without external dependencies that represents the kernels identified for acceleration. This file is in turn passed as an input to PandA-Bambu [8]. The other output is an LLVM IR file representing the host program that orchestrates calls to the accelerators. All the optimizations performed by the SODA framework are implemented as different compiler transformations.

SODA provides a design space exploration engine that selects a suitable combination of compiler passes and parameters, optimizing the design for a chosen target metric (performance, area, power, etc.).

2.1 SODA Frontend

SODA-OPT [2] is the framework's compiler frontend (Figure 1b). It is developed by extending LLVM's MLIR, which allows to build a modular and reusable compiler infrastructure through the definition of *dialects*, i.e., specialized, and self-contained IRs compliant with MLIR's meta-IR syntax.

Conventional HLS methodologies often require significant code modifications and/or take advantage of compiler directives provided via pragma annotations. Such directives influence HLS optimizations, for example exposing more or less task-level parallelism in parallel loops, or, controlling their unrolling factor. SODA-OPT takes an orthogonal approach by leveraging the semantic information carried by context-specific MLIR dialects and automatically applies the high-level transformations while preparing the input program for hardware synthesis.

SODA-OPT provides compilation passes to *S*earch, *O*utline, *Optimize*, *D*ispatch, and *A*ccelerate parts of the initial specification coming from high-level frameworks. SODA-OPT defines the soda MLIR dialect, used for the automatic partitioning of the input application into a host program responsible for orchestrating the runtime execution, and the custom hardware accelerators [2].

In the search phase, SODA-OPT analyzes the MLIR representation of the specification to identify code regions suitable for acceleration. Such regions are then extracted into separate MLIR modules (outline), which undergo further optimization passes. SODA-OPT can leverage MLIR dialects and the associated optimizations directly included within the MLIR distribution in the LLVM compiler framework or provided externally. For example, SODA-OPT leverages MLIR's linalg and affine dialects to identify operators and perform loop optimizations.

Machine Learning frameworks (e.g., TensorFlow, ONNX-MLIR, and TORCH-MLIR), software for scientific computing (e.g., NPCOMP), and general-purpose programming languages (e.g., the FLANG Fortran compiler) are designing MLIR dialects, optimizations, and lowering passes to optimize their input programs. SODA-OPT can directly interface with all the frameworks that lower to dialects provided in the MLIR distribution.

2.2 SODA Synthesizer Backend

Bambu (Figure 1c) is an open-source state-of-the-art HLS tool from the PandA framework, and it acts as the main synthesis backend of

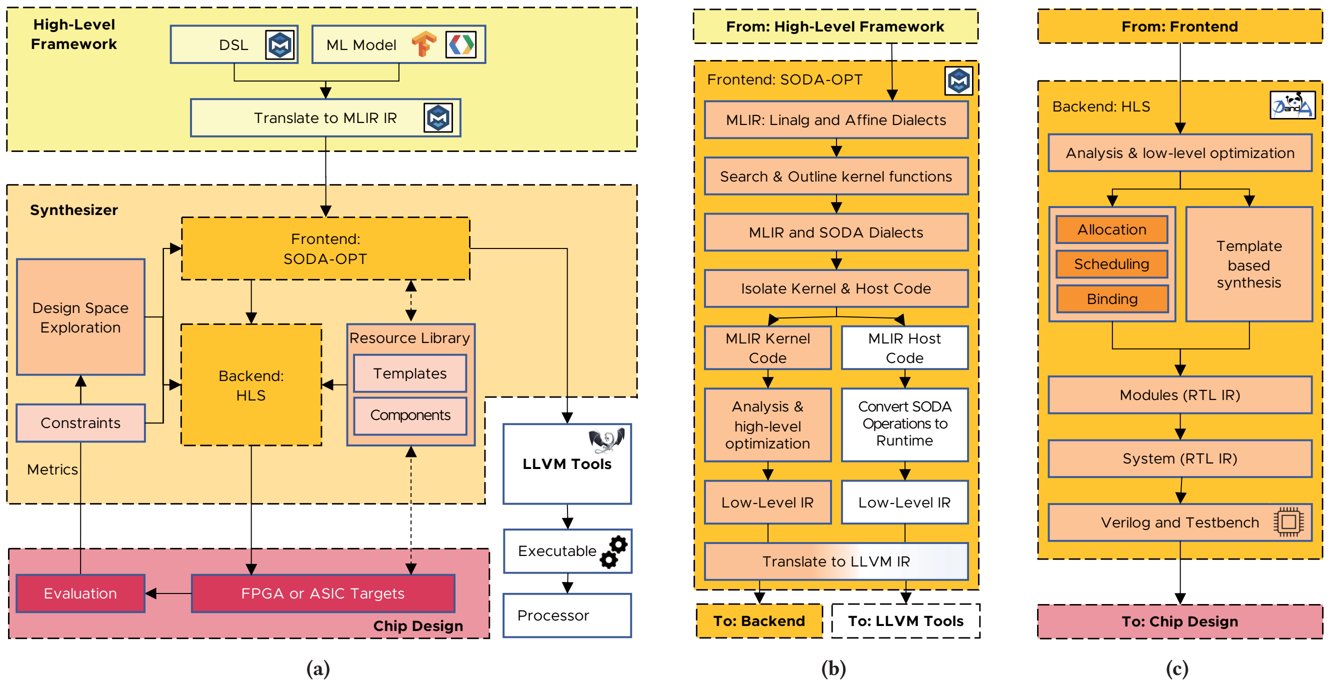

Figure 1: The SODA framework is an open-source, multi-level, modular, extensible, hardware generator composed of a high-level compiler and an HLS backend

the SODA framework. Bambu synthesizes the accelerators designs starting from the LLVM IR produced by SODA-OPT, accepted by the tool as input, in addition to C and C++ programs, through a dedicated Clang plug-in. This feature allows SODA-OPT to directly interface with the tool, acting as an additional specialized frontend. Starting from LLVM IR generated after SODA-OPT high-level optimizations for HLS results in superior quality of results compared to accelerators synthesized starting from C/C++. Internally Bambu builds several IRs to perform HLS steps (resource allocation, scheduling, and binding) and optimizations (e.g., operations chaining, bitwidth analysis, loop optimizations). It finally generates as output the RTL description of the generated hardware, in Verilog or VHDL. In addition to the synthesizable RTL code, Bambu can also automatically generate testbenches for verification, together with simulation script for different tools, such as ModelSim and Verilator. Bambu enables the SODA framework to target FPGAs (from Xilinx, Altera, Lattice, NanoXplore) and ASICs. For ASICs, SODA supports Verilog-to-GDSII generation using both commercial (Synopsis Design Compiler) and open-source (OpenROAD [9]) logic synthesis tools. The final RTL designs are generated in behavioral HDL and can synthesized targeting any type of device and process technology. However, several steps in the HLS process can benefit from a detailed characterization (area, delay, power) of functional units and components in the resource library, with respect to specific target devices and technology. In addition to individual modules, integrating technology-specific interconnect models can improve the quality of the results of the generated designs. Module binding and operations scheduling can use this information to optimize for various metrics (e.g., overall latency and area of the accelerator) while meeting synthesis constraints like a target frequency. For example, if sufficient slack exists, two functional units can be *chained* to execute in the same control step.

For this purpose, Bambu integrates *Eucalyptus*, a resource characterization tool. Eucalyptus runs micro-benchmarks with the backend logic synthesis tools and annotates the relevant information for each component in the resource library, for each target technology or FPGA device. Currently, Bambu already includes characterization for a variety of FPGA devices and various ASIC technology libraries. However, new targets can be added, further extending opportunities for design space exploration, with semi-automatic tradeoff evaluations across different target technologies.

Bambu, by default, generates RTL designs following the finite state machine with datapath (FSMD) model, but also integrates methodologies to support alternative parallel accelerator designs. It can, in fact, compose FSMD components as processing elements in coarse-grained dataflow designs [5] or generate completely dynamic dataflow accelerators at the single instruction level [4]. To achieve high-throughput in parallel irregular workloads, Bambu can exploit dynamically scheduled, multithreaded parallel templates [16]. In addition, Bambu integrates modular synthesis methodologies [17] to enable inter-procedural resource sharing across the design hierarchy.

MLIR descriptions are naturally parallel and hierarchical, making it possible to trigger Bambu's advanced synthesis methodologies from SODA-OPT. Rather than requiring manual annotations on the input code, we can define the design hierarchy at a higher level of abstraction by exploiting MLIR. This approach demonstrates how clean interfaces and integration between the two tools facilitate

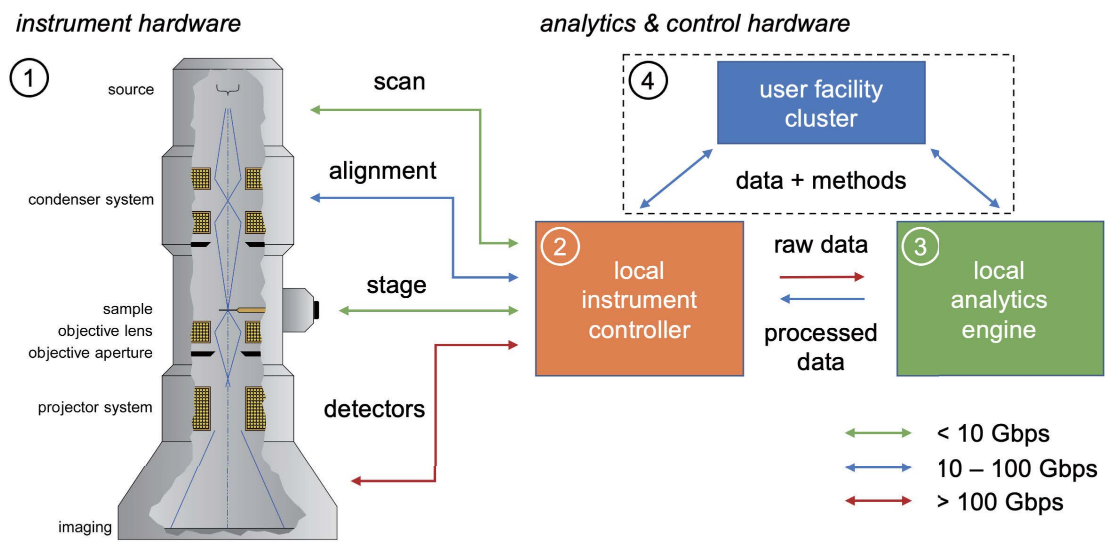

Figure 2: Opportunities for co-design in Transmission Electron Microscopy (TEM)

hardware design, removing the need to provide input code with hardware information in the form of annotations.

3 PRECISION MATERIAL SYNTHESIS USE CASE

Artificial intelligence platforms provide a unique opportunity to integrate new functionalities in materials evolution processes. Applications ranges from sentient and adaptive biological systems to 3D-printing to synthesis of atomically precise quantum devices.

A key part of precision material synthesis is Transmission Electron Microscopy (TEM). TEM provides insightful information about the structure and dynamics of physical phenomena.

Figure 2 provides an overview of signals and data incoming from a TEM instrument (1) and data that local instrument controller (2) and local analytics engine (3) process. In the experimental workflow, data might then be sent to a user facility (4) where additional analytics and control software are located. The user facility may use acquired data to perform additional scientific simulations and retrain models for the on-instrument local analytics engine.

Modern TEM instruments have high resolution and acquisition rates, and provide detector designs that enable manufacturing processes to be analyzed *in operando* at acquisition speeds. However, reaching the latency required to perform real-time decision-making needs effective edge processing solution. Some of the latest TEMs provide opportunities to connect to FPGAs, which are commonly used to process streaming data, or other on-edge processing designs.

With effective local processing capabilities, there are several opportunities for applying AI to improve the experimental workflows in TEM. Three opportunities are:

(1) Electron Energy Loss Spectrometry (EELS). Even if EELS data acquisition speeds with electron counting regularly reaches 400 fps with near-zero read noise, signal to noise ratio (SNR) remains a challenge, and low SNR can render datasets too noisy for analysis [18]. Machine learning can provide solutions for denoising. For example, RapidEELS [18, 20] is a framework composed of an autoencoder and a binary classifier to denoise low SNR EELS spectra and classifying their oxidation state. The autoencoder is trained on high framerate, low SNR spectra and their corresponding denoised, background subtracted data. The classifier is trained on the latent space representation from the autoencoder, and it establishes if the oxidation state is "initial" or "annealed". The classifier consists of a single, two-neuron, dense layer with Softmax activation.
(2) Reflection high-energy electron diffraction (RHEED). RHEED is a powerful in-situ surface characterization technique used during molecular beam epitaxy (MBE) and pulsed laser deposition (PLD) growth to provide insights into surface structure and quality [14]. The real-time qualitative and post-growth quantitative analysis of RHEED diffraction patterns provides crucial information on crystalline phase, surface roughness, in-plane strain, growth rate, and the presence of undesirable secondary phases and/or polycrystalline components.
Recent work has demonstrated the promise of applying data analytics to RHEED pattern images [19, 22]. RHEED data generated during MBE growth has been successfully analyzed after the deposition with both multivariate methods (e.g., principal component analysis) using k-means clustering algorithms and a convolutional neural network architecture to provide phase identification and qualitative insight into film quality [13].
(3) Neural Networks Potentials (NNPs). NNPs are a way to represent the multidimensional potential energy surface (PES) leveraging machine learning [10]. The PES is of central importance for reaching an atomic-level understanding of any type of system, and contains all the information about the

stable and metastable structures, the atomic forces driving the dynamics at finite temperatures, the transition states and barriers governing reactions and structural transitions, and also the atomic vibrations. NNPs learn the shape of the PES from reference data obtained from high-level electronic structure calculations. This allows to represent the atomic interactions in large scale simulations (such as molecular dynamics) significantly faster than the electronic structure calculations without a significant loss in accuracy.
Providing the ability to accelerate simulations on the edge with surrogate models allows generating additional data that can then be used to perform automated decision-making.

4 EXPERIMENTAL EVALUATION

Table 1: Execution latency (Clock Cycles) of LeNet with different outlining strategies. Merged cells correspond to a single accelerator.

Layer type	Individual Ops	Fused as TF kernels	Fused Coarser	Full Network
Conv2D - 6x5x5 filters	2,353,598	2,388,122	2,443,073	5,965,844
ReLU	34,526			
AvgPooling2D - 2x2	84,338	84,338		
Conv2D - 16x5x5 filters	4,835,522	4,835,522	4,853,938	
ReLu	11,312			
AvgPooling2D - 2x2	28,842	28,842		
Dense - 120 units	926,402	927,362	927,362	
ReLU	842			
Dense - 84 units	195,722	196,394	196,394	
ReLU	590			
Dense - 10 units	16,372	16,731	16,731	
Softmax	410			
Total	8,488,476	8,477,311	8,427,498	5,965,844

We demonstrate the end-to-end capabilities of the SODA framework by automatically generating specialized hardware accelerators from high-level programming frameworks. We automatically translate a LeNet model [12] trained in TensorFlow to the linalg dialect and employ SODA-OPT to search, outline, and optimize different regions of the network. We generate different specialized accelerators with the SODA framework targeting a Xilinx xc7z045-2ffg900 FPGA device, and frequency of 200 MHz.

Table 1 presents, for each layer of the model, the execution latency in number of clock cycles obtained with different outlining strategies. We first generate one accelerator for each individual operator of the network, and progressively move to coarser granularity by fusing operators as usually performed in TensorFlow. Finally, we outline the entire network to generate a single accelerator. SODA-OPT allows exploring custom accelerators for fused operators at any level of granularity. As the results show, outlining the entire network provides the greatest opportunity for performance improvement, with 1.42x speedup compared to the outlining of individual operations. Combining layers also significantly affects the area and frequency, as summarized in Table 2. While fusing layers could provide more opportunities for optimization and reduce the execution latency of the network, it could also result in bigger accelerators or lower maximum achievable frequency.

The ability to perform outlining at arbitrary granularity at the IR level enables to explore a much wider design space than just combining parametrized accelerators for each operator. In fact, depending on the desired optimization objectives (latency or area) and system constraints (e.g., the target frequency), it is also possible to describe a system composed of accelerators at different granularity.

5 RESEARCH DIRECTIONS AND OPPORTUNITIES

Achieving autonomy (self-driving vehicles, experimental workflows, and more) requires the ability to process multimodal data, often unstructured, at very low latency to perform real-time decision. In the case of scientific experimental workflows, for example, the latencies are highly dependent on the evolution of the physical processes. These situations justify the need of highly specialized processing, as more generalized accelerators design may not completely fit the complex tradeoffs required by these applications. Even if reconfigurable devices, able to be specialized after their production, seems to provide an opportunity to support the diverse application areas requiring autonomy, developing and exploring highly specialized accelerators still is highly inconvenient. Research in the data analytics approaches needed to identify decisions to take are performed in high-level frameworks, which typically do not consider the strict latency requirements of autonomous control. Research in the area of design automation is today looking at bridging the gap from high-level framework. SODA is part of such a context, providing a modular and extensible infrastructure to move from high-level data science algorithms formulation to their hardware implementation. However, there are still significant challenges to overcome and many opportunities arising from the investments on advanced semiconductor manufacturing.

First, the hardware design ecosystem needs to provide both open-source tools and the ability to seamlessly integrate with proprietary solutions. Open-source tools have been mainly research-focused, and as such have the freedom to try to quickly address new emerging requirements. However, they sometimes lack the production-ready quality of proprietary tools. Conversely, commercial tools are typically difficult to directly integrate into automated flows and require significant manual efforts, since some of their algorithms and interfaces are proprietary, significantly limiting opportunities for design space exploration. A prominent example in the area of low latency control for experimental workflows is HLS4ML [7]. Born with only the support for Xilinx HLS tools, HLS4ML is now moving towards supporting other commercial tools for FPGAs and ASICs (e.g., Mentor Catapult C). However, the tool employs a library-based approach, developing specific HLS templates for each network under consideration. While this leads to high performance for the considered cases, it might not allow to quickly explore new algorithms (even if at lower quality of results), and makes necessary to write solutions for each and every HLS backend with custom and proprietary directives. We also are investigating the integration of the SODA framework with several different tools of the open-source and proprietary ecosystems. For SODA-OPT we have implemented initial support for commercial FPGA synthesis tools (Vitis HLS) by also generating optimized LLVM IR inputs, extending the work in [23]. In terms of target technologies, instead, our open-source HLS backend can already support FPGAs from

Table 2: Area requirements (Slice LUT/FlipFlop Pairs)) and maximum frequency (MHz) of LeNet with different outlining strategies. Merged cells correspond to a single accelerator.

Layer type	Individual Ops		Fused as TF kernels		Fused Coarser		Full Network	
	Pairs	Fmax	Pairs	Fmax	Pairs	Fmax	Pairs	Fmax
Conv2D - 6x5x5 filters	4,497	222.76	5,288	221.33	33,135	202.265	134,573	134.74
ReLU	923	234.57						
AvgPooling2D - 2x2	13,325	210.03	13,325	210.03				
Conv2D - 16x5x5 filters	7,315	221.92	7,315	221.92	39,607	201.32		
ReLu	2,175	236.01						
AvgPooling2D - 2x2	12,315	214.59	12,315	214.59				
Dense - 120 units	2,052	219.34	2,347	218.24	2,347	218.24		
ReLU	234	315.65						
Dense - 84 units	2,148	221.09	2,444	218.57	2,444	218.57		
ReLU	234	310.17						
Dense - 10 units	2,934	207.90	6,765	173.19	6,765	173.19		
Softmax	2,792	216.68						
		min(Fmax)		**min(Fmax)**		**min(Fmax)**		
Total	50,944	207.90	49,799	173.19	84,298	173.19	134,573	134.74

different vendors and ASIC, leveraging technology characterization to improve quality of results.

While FPGAs appear a promising target due to their ability to be configured post fabrication, with the new investments in semiconductor manufacturing capabilities (e.g., the CHIPS & Science Act) that could lower the access to production lines for small volume highly specialized designs there is an opportunity to assemble custom ASIC solutions through chiplets. From this point of view, interfacing with both commercial and open-source ASIC tools, is trying to foster opportunities to perform end-to-end synthesis. On this line, advanced manufacturing capability for small scale production and prototyping can allow composition and generation of entire domain specialized systems. SODA-OPT can already reason about system-level design. It performs code partitioning, optimizations specific for custom hardware generation, and composition of a system architecture, generating glue code for control processors or assembling accelerators in dynamically scheduled architectures. A similar approach could be further extended by integrating with rapid prototyping platforms from the open-source ecosystem, such as the Embedded Scalable Platforms (ESP) [15]. We are currently working to integrate both SODA-OPT and Bambu with ESP. SODA-OPT can drive the system-level design, leveraging the services offered by ESP to invoke accelerators. Bambu can provide ESP with an open-source HLS backend for custom accelerators, which will be generated from code partitioned, optimized, and mapped on the ESP SoC by SODA-OPT.

Considering the availability of open-source IPs and architectural templates, several of these modules can either become targets for SoC design (e.g., platforms provided with RISC-V cores) or part of the HLS tool resource library. For example, Bambu can integrate templates of systolic arrays in its resource library, allowing it to generate specialized processing elements, similar to the approach presented in [16].

An additional opportunity, particularly relevant for specialized designs with quickly changing algorithms, is supporting generators for domain-specific FPGAs. In the case of experimental workflows for material synthesis, the reconfigurable substrate could be specialized to deal with the input data types and the instrument interfaces. SODA could integrate with solutions such as OpenFPGA [21], performing high-level analysis to identify patterns that might require additional hard macros in the hardware substrate while still leveraging fine-grained reconfigurability. The HLS tool could perform design space exploration, leveraging the hard macros through the resource library, or even synthesizing them on-the-fly and reusing them as necessary.

On a longer term, providing a modular hardware design toolchain can also allow to explore new computing paradigms leveraging existing components of the infrastructure. For example, we previously have shown the SODA framework modularity by proposing a new MLIR dialect for spiking neural networks (SNN) [6]. The dialect allows mapping SNNs on digital neurons that could be synthesized on FPGAs using Bambu (as often done with other SNNs frameworks). Domain specialized FPGAs might even provide components that facilitate design of SNNs (for example, memristive devices). In general, supporting design and generation new computing paradigms could provide significant advantages for edge devices in areas such as experimental instruments: remaining near the sensors, and in the same domain of the acquired data (e.g., analog) beside being power efficient might drastically reduce or remove conversion latency and thus facilitate real-time control.

6 CONCLUSIONS

In this paper we presented an overview of the SODA framework, an end-to-end, multi-level, open-source hardware compiler. SODA incorporates a frontend based on the MLIR infrastructure and a backend leveraging a state-of-the-art HLS engine. SODA interfaces with a variety of high-level data science frameworks such as TensorFlow, and generates custom hardware accelerators targeting both FPGAs and ASICs. We discussed the need for hardware acceleration in research areas such as precision material synthesis, focusing on the low latency required to enable autonomous control for monitoring the evolution of the physical processes. We demonstrated how end-to-end solutions like SODA could provide the agility needed to go from algorithmic formulation to hardware implementation of specialized accelerators for machine learning models, and perform design space exploration to meet the edge processing requirements. Finally, we discussed research challenges and opportunities to enable on-chip learning at the edge leveraging end-to-end hardware design tools for critical areas such as autonomous scientific systems.

REFERENCES

[1] E. Bethel and eds. 2016. *Report of the DOE Workshop on Management, Analysis, and Visualization of Experimental and Observational data – The Convergence of Data and Computing*. Technical Report.

[2] Nicolas Bohm Agostini, Serena Curzel, Vinay Amatya, Cheng Tan, Marco Minutoli, Vito Giovanni Castellana, Joseph Manzano, David Kaeli, and Antonino Tumeo. 2022. An MLIR-based Compiler Flow for System-Level Design and Hardware Acceleration. In *41st IEEE/ACM International Conference on Computer-Aided Design (ICCAD'22)*. To appear.

[3] Nicolas Bohm Agostini, Serena Curzel, Jeff Zhang, Ankur Limaye, Cheng Tan, Vinay Amatya, Marco Minutoli, Vito Giovanni Castellana, Joseph Manzano, David Brooks, Gu-Yeon Wei, and Antonino Tumeo. 2022. Bridging Python to Silicon: The SODA Toolchain. *IEEE Micro* (2022). https://doi.org/10.1109/MM.2022.3178580

[4] Vito Giovanni Castellana and Fabrizio Ferrandi. 2013. An automated flow for the High Level Synthesis of coarse grained parallel applications. In *2013 International Conference on Field-Programmable Technology (FPT)*. 294–301. https://doi.org/10.1109/FPT.2013.6718370

[5] Vito Giovanni Castellana, Antonino Tumeo, and Fabrizio Ferrandi. 2021. High-Level Synthesis of Parallel Specifications Coupling Static and Dynamic Controllers. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS'21)*. 192–202. https://doi.org/10.1109/IPDPS49936.2021.00028

[6] S. Curzel, N. Bohm Agostini, S. Song, I. Dagli, A. Limaye, M. Minutoli, V. G. Castellana, V. Amatya, J. Manzano, A. Das, F. Ferrandi, and A. Tumeo. 2021. Automated Generation of Integrated Digital and Spiking Neuromorphic Machine Learning Accelerators. In *ICCAD: International Conference On Computer Aided Design*. 1–7.

[7] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu. 2018. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation* 13, 07 (jul 2018), P07027–P07027. https://doi.org/10.1088/1748-0221/13/07/p07027

[8] Fabrizio Ferrandi, Vito Giovanni Castellana, Serena Curzel, Pietro Fezzardi, Michele Fiorito, Marco Lattuada, Marco Minutoli, Christian Pilato, and Antonino Tumeo. 2021. Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications. In *58th ACM/IEEE Design Automation Conference (DAC'21)*. 1327–1330. https://doi.org/10.1109/DAC18074.2021.9586110

[9] Andrew B. Kahng and Tom Spyrou. 2021. The OpenROAD Project: Unleashing Hardware Innovation. In *Government Microcircuit Applications and Critical Technology Conference*. 1–6.

[10] Emir Kocer, Tsz Wai Ko, and Jörg Behler. 2021. Neural Network Potentials: A Concise Overview of Methods. https://doi.org/10.48550/ARXIV.2107.03727

[11] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. 2021. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. In *IEEE/ACM International Symposium on Code Generation and Optimization (CGO'21)*. 2–14. https://doi.org/10.1109/CGO51591.2021.9370308

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[13] Haotong Liang, Valentin Stanev, Aaron Gilad Kusne, Yuto Tsukahara, Kaito Ito, Ryota Takahashi, Mikk Lippmaa, and Ichiro Takeuchi. 2022. Application of machine learning to reflection high-energy electron diffraction images for automated structural phase mapping. *Phys. Rev. Materials* 6 (Jun 2022), 063805. Issue 6.

[14] John E. Mahan, Kent M. Geib, G. Y. Robinson, and Robert G. Long. 1990. A review of the geometrical fundamentals of reflection high-energy electron diffraction with application to silicon surfaces. *Journal of Vacuum Science & Technology A* 8, 5 (1990), 3692–3700.

[15] Paolo Mantovani, Davide Giri, Giuseppe Di Guglielmo, Luca Piccolboni, Joseph Zuckerman, Emilio G. Cota, Michele Petracca, Christian Pilato, and Luca P. Carloni. 2020. Agile SoC Development with Open ESP. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD'20)*. 1–9. https://doi.org/10.1145/3400302.3415753

[16] Marco Minutoli, Vito Giovanni Castellana, Nicola Saporetti, Stefano Devecchi, Marco Lattuada, Pietro Fezzardi, Antonino Tumeo, and Fabrizio Ferrandi. 2022. Svelto: High-Level Synthesis of Multi-Threaded Accelerators for Graph Analytics. *IEEE Trans. Comput.* 71, 3 (March 2022), 520–533. https://doi.org/10.1109/TC.2021.3057860

[17] Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo, and Fabrizio Ferrandi. 2015. Inter-procedural resource sharing in High Level Synthesis through function proxies. In *25th International Conference on Field Programmable Logic and Applications (FPL'15)*. 1–8. https://doi.org/10.1109/FPL.2015.7293958

[18] Cassandra Pate, James Hart, and Mitra Taheri. 2021. RapidEELS: machine learning for denoising and classification in rapid acquisition electron energy loss spectroscopy. *Scientific Reports* 11 (09 2021). https://doi.org/10.1038/s41598-021-97668-8

[19] Sydney R. Provence, Suresh Thapa, Rajendra Paudel, Tristan K. Truttmann, Abhinav Prakash, Bharat Jalan, and Ryan B. Comes. 2020. Machine learning analysis of perovskite oxides grown by molecular beam epitaxy. *Physical Review Materials* 4, 8 (aug 2020). https://doi.org/10.1103/physrevmaterials.4.083807

[20] W. Snyder. 2022. RapidEELS. https://github.com/patecm/rapidEELS Online, accessed 11-2022.

[21] Xifan Tang, Edouard Giacomin, Aurélien Alacchi, Baudouin Chauviere, and Pierre-Emmanuel Gaillardon. 2019. OpenFPGA: An Opensource Framework Enabling Rapid Prototyping of Customizable FPGAs. In *29th International Conference on Field Programmable Logic and Applications (FPL'19)*. 367–374. https://doi.org/10.1109/FPL.2019.00065

[22] Rama K. Vasudevan, Alexander Tselev, Arthur P. Baddorf, and Sergei V. Kalinin. 2014. Big-Data Reflection High Energy Electron Diffraction Analysis for Understanding Epitaxial Film Growth Processes. *ACS Nano* 8, 10 (2014), 10899–10908.

[23] Ruizhe Zhao, Jianyi Cheng, Wayne Luk, and George A. Constantinides. 2022. POLSCA: Polyhedral High-Level Synthesis with Compiler Transformations. In *32nd International Conference on Field Programmable Logic and Applications (FPL'22)*. To appear.