ONNC-based Software Development Platform for Configurable NVDLA Designs

Wei-Fen Lin, Cheng-Tao Hsieh, Cheng-Yi Chou
Skymizer Taiwan Inc.

***Abstract*—With the proliferation of deep learning and the increasing pressure to deploy inference applications at the edge, many AI chip makers integrate the open source NVIDIA Deep Learning Accelerator (NVDLA) design in their AI solutions. Lack of open source compiler support and having only limited configurability support in the software stacks erect a barrier for developers to freely explore the NVDLA design space at system level. This paper presents an ONNC-based software development platform that includes the first open source compiler for NVDLA-based designs, a virtual platform with various CPU models as well as configurable NVDLA models, and auxiliary tools for debugging. The platform is tightly coupled with the hardware design tradeoffs and provides extendibility for compiler optimization, more CPU types, and more NVDLA hardware configurations. It lifts many restrictions of software development for those who like to leverage the NVDLA design in inference applications.**

***Index Terms*—Deep learning accelerators, Compilers, NVDLA, ONNC**

I. Introduction

Accelerating AI at the edge is critical in enabling new applications in many industries. The NVIDIA Deep Learning Accelerator [1] (NVDLA) is a free and open architecture that provides a scalable, configurable and modular design to address the computational demands of neural network inference. In addition to the open source hardware, NVDLA also has an open source software stack with a virtual platform, a pre-built Linux kernel, a user-mode driver (UMD), and a kernel-mode driver (KMD) for developers to explore the full-system design. However, NVIDIA only released its compiler (nvdla_compiler) in the binary form and the compiler only supported limited AI models in a specific NVDLA configuration (nv_full). Lack of an open source compiler and support for various hardware configurations in the software stack has become an inevitable barrier for developers and researchers to improve and optimize an NVDLA-based design.

ONNC (Open Neural Network Compiler) is a compilation framework designed specifically for proprietary deep learning accelerators [2]. Its software architecture expedites porting ONNC to any DLA design that supports ONNX (Open Neural Network Exchange) operators. ONNC is the first open source compiler available for NVDLA-based hardware designs. Its NVDLA backend can compile a model into an executable NVDLA Loadable file. Integrating ONNC with the NVDLA software stack opens up opportunities for developers and researchers to explore the NVDLA-based inference design at system level. In this paper, we present an ONNC-based software development platform for configurable NVDLA designs. It facilitates the software/hardware co-design process and early-stage software development for NVDLA-based designs. We will discuss the hardware design tradeoff in Section II, describes the software development platform and how it facilitates research and development in Section III, and conclude in Section IV.

II. Hardware Design Tradeoff

Innovation in hardware is slowing due to rising costs of chip design and diminishing benefits from Moore's law and Dennard scaling.

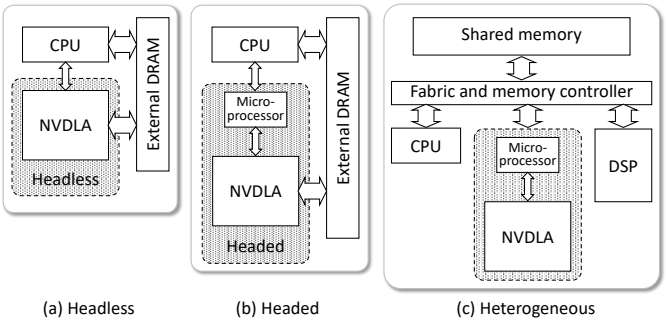

Fig. 1: NVDLA System Architecture

Leveraging the NVDLA open source project to shrink design cycles is appealing to many inference chip makers. Making decisions on the hardware design tradeoffs determines the levels of support needed in the software stacks.

A. NVDLA Configuration

NVDLA has a wide array of hardware parameters that can be configured to balance area, power, and performance [3]. It is a good practice to choose a configuration that best fits the target applications. For instance, by analyzing the operator types and the maximum tensor size, a developer may enable the necessary hardware modules and set an appropriate CBUF size for the convolution modules. Take the Tiny-YOLO v1 model as an example. The released nv_small configuration is probably good enough to process 720x480 images at a frame rate of 15 FPS for a 400MHz NVDLA-based inference chip. Once the NVDLA configuration is determined, it requires support in the compiler, kernel mode driver, and the NVDLA model of the virtual platform in the released software stack.

B. Embedded CPU

NVDLA needs an embedded CPU running Linux for developers to leverage the NVDLA software stack. Although NVIDIA and Arm team up to integrate the open-source NVDLA architecture into Arm's Project Trillium platform for machine learning, many developers favor to choose a free open source RISC-V core over a licensed ARM core. This requires RISC-V support in the virtual platform as well as Linux kernel.

C. System Architecture

The system architecture determines how NVDLA interacts with memory and the embedded CPU. NVDLA provides headless and headed implementations as shown in Fig 1(a) and 1(b). The headed implementation has an additional micro-controller aside the embedded CPU to offload the NVDLA control task. Some high-end design may even integrate another DSP to form a heterogeneous system architecture as shown in Fig 1(c) for high-performance applications. In that case, it requires heterogeneous system support in the compiler.

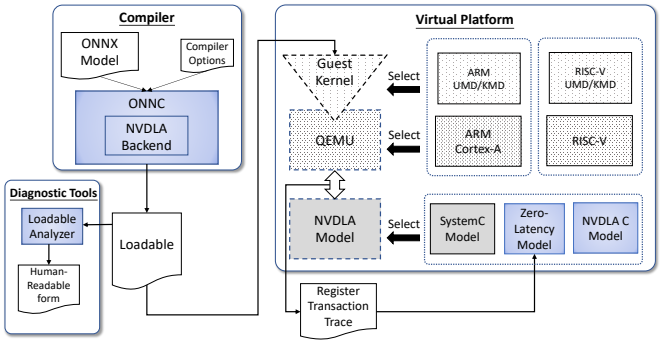

Fig. 2: Software Development Platform

III. Software Development Platform

An ONNC-based software development platform as shown in Fig 2 is built to support various hardware design tradeoffs mentioned in section II. This platform enables the developers to customize their software stacks at the early development phase, explore the hardware/software co-design space, and optimize their designs at the system level.

A. ONNC Compiler Porting

The official NVDLA compiler is released in the binary form and it only supports limited operators and models. On the contrary, its Linux drivers, UMD and KMD, are released with source code and exist as defined APIs. By tracing the code, we successfully demystify the Loadable file format and the NVDLA register specification. Based on the reverse engineering results, we implemented the first open source compiler for NVDLA-based designs. The NVDLA backend in the ONNC compiler compiles a model into an NVDLA Loadable file and the Loadable file is valid if it can correctly run on the officially released virtual platform. As listed in Table 1, ONNC supports more ONNX models than the official NVDLA compiler. ONNC can compile 6 models and run on the NVDLA virtual platform successfully. VGG-19 and ZFNet-512 are not supported by either hardware configurations because they have layers that exceed hardware limitation. The other 4 unsupported models need the support of more operators. For the nv_small configuration, there is no official compiler available. Only one Alexnet Loadable file is available in the released testbench. ONNC has the same support in the nv_small configuration as in the nv_full configuration. The results in Table 1 are applicable regardless which CPU model is used. The basic ONNC support for nv_full is released in the ONNC GitHub repository V1.0 [3] to help the research community.

B. NVDLA Virtual Platform and UMD/KMD Porting

The officially released virtual platform only supports the ARM Cortex-A57 core and the QEMU version is out of date. We upgraded QEMU to a new version to support a wider range of CPU models including the RISC-V CPU model. In addition, we ported Linux OS, UMD, and KMD for both nv_full and nv_small configurations. The SystemC model of NVDLA is generated from the official repository as specified by the official document for both nv_full and nv_small configurations. With these efforts, the compiled Loadable files can run on a software platform that adapts to a wide range of hardware configurations.

C. Diagnostic Tools

It is common that a Loadable is invalid due to mismatched hardware configuration or compiler errors. Two diagnostic tools and a utility are built to ease the debugging process. A Loadable reader translates the Loadable file in the binary form into a human-readable text format. A Loadable analyzer shows the layer information in hardware executable layers for users to understand how ONNC maps the model layers into hardware hybrid layers for execution. In addition, it also checks the validity of the Loadable parameter setting and shows the layer dependencies. Another useful utility is built to collect the register transaction trace between the embedded CPU and the NVDLA model for debugging purpose and drives the zero-latency model in the next section.

TABLE I: Compiler support status for ONNX model zoo.

model	nv_full		nv_small	
	NVDLA	ONNC	NVDLA	ONNC
AlexNet	O	O	O	O
GoogleNet	x	O	x	O
CaffeNet	O	O	x	O
R-CNN ILSVRD13	O	O	x	O
DenseNet-121	x	x	x	x
Inception v1	x	O	x	O
Inception v2	x	x	x	x
ResNet-50	O	O	x	O
ShuffleNet	x	x	x	x
SqueezeNet	x	x	x	x
VGG-19	N/A	N/A	N/A	N/A
ZFNet-512	N/A	N/A	N/A	N/A

D. NVDLA Models

Although the NVDLA open source release includes an NVDLA SystemC model, providing a register-accurate system on which software can be quickly developed and debugged, it runs slowly and developers might not need as many execution details as it provides. To speed up simulation for trivial code changes, we built a zero-latency model that returns NVDLA responses immediately after receiving the requests using information from the register transaction trace. For the Alexnet model, the zero-latency model runs an inference in 45 seconds compared to 5000 seconds running on a SystemC model. Furthermore, a fast NVDLA C-model is also built to explore customized NVDLA designs. For those who like to go wild about the hardware design while still leveraging the NVDLA software stack, implementing their ideas in a fast NVDLA C-model is probably a great start point.

IV. Conclusion and Future Work

This paper presents a software development platform for NVDLA-based designs. As developers make decisions on hardware design tradeoffs, they need great support in the software stack to expedite their development process. The ONNC-based software development platform supports hardware design tradeoffs in compiler, virtual platform, drivers and NVDLA models. One future work is to enhance the NVDLA C model for architectural exploration. Furthermore, running the compiled Loadable in a real hardware platform will help performance calibration and verification. Lastly, supporting more operators in ONNC will make this platform more powerful in practice.

References

[1] (2017) NVDLA deep learning accelerator. [Online]. Available: http://nvdla.org/

[2] W. F. Lin, D. Y. Tsai, L. Tang, C. T. Hsieh, C. Y. Chou, P. H. Chang, and L. Hsu, "ONNC: A compilation framework connecting ONNX to proprietary deep learning accelerators," in *IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS 2019)*. IEEE, 2019.

[3] (March 2019) ONNC open source project. release v1.0. [Online]. Available: https://github.com/ONNC/onnc