Xilinx Open Hardware 2021 design contest

Project name:
FGPU: An Configurable Soft-Core SIMT Accelerator

Team participants:
Hector Gerardo Muñoz Hernandez
Liliia Kudelina
Mitko Veleski

Supervisor:
Marcelo Brandalero

University:
Brandenburg University of Technology Cottbus - Senftenberg

1. Introduction:

Machine Learning (ML) is a subtopic of Artificial Intelligence (AI) that significantly grew in importance in recent years \cite{Dean2018}. Especially in the domain of embedded computing, ML is of great interest due to low latency, data privacy, and restricted bandwidth concerns. CNNs have proven to be an efficient way of handling ML tasks because they can achieve high accuracy and frequently outperform traditional AI approaches \cite{venieris2018toolflows}. Due to their flexibility, CNNs have been widely used in object detection and classification, autonomous driving, and drone navigation \cite{venieris2018toolflows}.

CNNs require high computational density and are thus often executed in GPUs and FPGAs, both of them having different benefits like parallel execution and programmable hardware \cite{GPUvsFPGAs}. The GPUs are known for offering high performance for parallel computation at the expense of high energy consumption. On the other hand, FPGAs are known for their increased energy efficiency,

but programming them requires hardware knowledge, and it is more time-consuming than its counterpart. Even with tools such as HLS, where higher level programming languages like C or C++ can be used, the designer still has to bare in mind the hardware structure of the target in order to write an efficient code.

Other than FPGAs and GPUs, a third option for CNN implementations is a soft processor or an overlay, which can be implemented into an FPGA and programmed via software. The FGPU \cite{fgpu} is such an overlay architecture that provides the architecture of a GPU. The FGPU allows saving space compared to the FPGAs because it can be reused for more than one application. It is also less time consuming to program than an FPGA, as the hardware doesn't need to be re-programmed for every new application.

1.1 FGPU:

A more detailed description of the FGPU architecture, presented in Figure 1, can be found in [1]. The FGPU is an open-source 32-bit multi-core GPU-like processor based on the Single Instruction Multiple Thread (SIMT) execution model. It does not replicate, even partly, any other existing GPU architecture. Even more, the FGPU has its own instruction set architecture, which is composed of 49 MIPS-like instructions inspired by the OpenCL execution model. The translation from a high-level OpenCL code is performed by deploying the dedicated LLVM-based compiler. Another significant feature is that FGPU supports floating-point operations. Up to 8 Compute Units (CUs) can be accommodated, each consisting of 8 processing elements (PEs). A single computing unit can run up to 512 work items (threads). Each work item owns a private memory of 32 registers that can be extended using scratchpad memories. Additionally, an off-chip memory limited to 4 GB can be accessed by any working item. The FGPU includes a direct-mapped, multi-ported, and write-back cache

system that can simultaneously serve multiple read/write requests. Finally, the FGPU is interfaced and controlled over the AXI4-lite bus.

2. Methodology

We showcase a static configuration of the FGPU running in a Zedboard: a standard Xilinx board featuring a ZYNQ-7000 System on Chip (SoC). The FGPU configuration uses 1CU, is running at 100MHz, and has floating point support directly on hardware.

The kernels used by the FGPU where compiled using the LLVM custom compiler that the FGPU project provides. To this end, we wrote kernels corresponding to each layer of the CNN in OpenCL, and got the corresponding compiled kernels in return.

On top of this, we created a Vitis project that uses the bitstream containing the FGPU, and ran a simple CNN.

The chosen CNN is able to recognize handwritten digits, and using the MNIST dataset, it consists of one convolutional layer, one maxpooling layer, one Fully Connected layer (FC), and an output layer.

The input required by this application is a grayscale image of 28 by 28 pixels where a handwritten digit is shown. Furthermore, our application needs this array of pixels to be flatten into a one dimensional array. We provide a header file (image.h) with the arrays of some digits, (where the user has to comment/uncomment for its use) for testing purposes, but the user can use his/her own images, provided that the pixel values get flatten to one dimensional array.