Diffusion Model Acceleration on FPGA

Hongwu Peng[1], Xi Xie[1], Amit Hasan[1], Jiahui Zhao[1], Wei Zhang[1], and Caiwen Ding[1]
[1]University of Connecticut, CT, USA
{hongwu.peng, xi.xie, amit.hasan, jiahui.zhao, wei.13.zhang, caiwen.ding}@uconn.edu

***Abstract*—Stable diffusion has become popular nowadays. However, most stable diffusion works put their model on GPU to gain acceleration, yet to be done on FPGA. GPU features its' bulk parallelism and is extremely efficient for DNN training and inference applications with large batch size. But GPU is inefficient for DNN inference applications with low batch size and irregular computation patterns. FPGA gate level customization enables its high energy efficiency under dedicated acceleration tasks such as single batch low latency DNN model inference and irregular graph processing. In this work, we utilize a simple dataset, MNIST for handwritten digit image generation, to present a naive stable diffusion network functionality. Then we build the model on FPGA and make an inference calculation on it, to demonstrate the effectiveness of single batch stable diffusion acceleration speedup on FPGAs.**

***Index Terms*—Stable Diffusion, FPGA acceleration, coarse-grained pipeline**

I. Plan of the project

We will use A) Conditional Diffusion model on MNIST following a public available Github Repo [1]. B) Then we do the FPGA implementation. The FPGA design follows [2]. We will utilize coarse-grained pipelining architecture to create the proper FPGA overlay for stable diffusion acceleration.

A. Conditional Diffusion model for MNIST

The main part of the model is a Context-UNet architecture. It consists of a Residual Convolutional block layer, two UNet Down block layers, a Sequential layer (which consists of 2-Dimensional Average Pooling block and Gaussian Error Linear Unit), four Embedded Fully Connected layers, a Sequential layer (which consists of 2-Dimensional Transposed Convolutional block, Group Normalization layer, Rectified Linear Unit), two UNet Up block layers, and a Sequential layer (which consists of 2-Dimensional Convolutional block, Group Normalization layer, Rectified Linear Unit, 2-Dimensional Convolutional block).
The model training step is: first we fetch the ground truth image data and the corresponding classes using the data loader and then we mix the image data with noise to produce diffused image data. We dropout context with some probability using the Bernoulli method that draws binary random numbers (0 or 1) from a Bernoulli distribution. We send the diffused image data and the context mask to the model to generate the output. Then we calculate the MSE (Mean Squared Error) loss between the output and the noise. Finally, we backpropagate the loss to update the weights of the model.

B. FPGA acceleration for this model

We will build the layers by High-level synthesis (HLS), in Vitis HLS. Then export it in RTL and download it to the FPGA board such as ZYNQ or Alveo U280. We will take advantage of some publicly available open-source FPGA projects such as [3] to implement the layers by HLS design. We will implement coarse-grained pipelining architecture with PE groups for the parallel calculation inside the layers. Then we will compare the computation time between FPGA boards and CPU.

References

[1] Tim Pearce. Conditional Diffusion MNIST. Retrived from https://github.com/TeaPearce/Conditional_Diffusion_MNIST. Accessed: 2022, Oct. 30th.
[2] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 161–170, 2015.
[3] YOLOv2 Accelerator in Xilinx's Zynq-7000 Soc(PYNQ-z2, Zedboard and ZCU102). Retrived from https://github.com/dhm2013724/yolov2_xilinx_fpga.