Term Project Summary Report – Group 18

Key Concept and Motivation

The key concept of this paper is Cambricon-D, a novel hardware accelerator architecture designed specifically for diffusion models. Cambricon-D aims to address these challenges through two key innovations:

- Full-network differential computing: Instead of computing on raw input values, Cambricon-D operates on the small differences (deltas) between timesteps. This allows for more efficient computation using lower precision arithmetic.
- Sign-mask dataflow: A novel technique that enables differential computing to work across nonlinear operations like activation functions, which traditionally blocked full-network differential approaches.

The motivation stems from the computational challenges posed by diffusion models:

- Diffusion models require many iterations of running the same neural network on slightly altered input data across timesteps.
- This iterative process leads to significant computational redundancy, as most of the input data remains similar between iterations.
- Existing hardware struggles to efficiently handle this redundancy, resulting in high computational costs and energy usage.

Brief Summary of Results

- 1.46x - 2.38x speedup over an NVIDIA A100 GPU across various diffusion models and resolutions
- 66-82% reduction in off-chip memory access compared to previous differential computing approaches
- Minimal model accuracy loss (0-4%) despite aggressive quantization
- 25-42% reduction in energy consumption

Preliminary Implementation Ideas

To implement a project based on Cambricon-D, the following approach is proposed:

Cambricon-D Implementation

- Design and develop the differential computing architecture in SystemVerilog
- Implement RTL logic for PE array design, sign-mask dataflow mechanism, on-chip SRAM buffer (using CACTI) and cycle-accurate timing control. Performance monitoring to calculate FLOPS can also be implemented for evaluating system throughput.
- Simulate basic functionality using a custom-made testbench.

Simulation Environment

- Develop a cycle-accurate simulator in C++ to simulate the Cambricon-D architecture

- Integrate with an existing DRAM simulator like Ramulator for memory modeling
- Create a performance model to estimate execution time, power usage, and area

Differential Computing Engine

- Implement the core differential computing logic
- Design the PE array structure with support for both int3 and fp16 computations
- Develop the outlier handling mechanism

Sign-Mask Dataflow in Details

- Implement the sign-bit management system
- Design the compressed delta format
- Create the near-data processing engine for sign-bit updates

Quantization Scheme

- Develop the proposed quantization algorithm in PyTorch
- Integrate with existing diffusion model implementations

Evaluation Framework

- Set up benchmarks using Guided-Diffusion and Stable Diffusion models
- Implement evaluation metrics for performance, energy, and model accuracy

Tools and Resources

- SystemVerilog for RTL development
- C++ for simulator development
- PyTorch for quantization and model modifications
- Ramulator for DRAM simulation
- Synopsys tools for RTL synthesis (if doing ASIC implementation)
- CACTI for on-chip memory modeling
- Guided-Diffusion and Stable Diffusion codebases
- GPU cluster for training and baseline comparisons

Key Milestones

- Cambricon-D hardware accelerator RTL implementation
- Basic C++ simulator framework and PE array implementation
- Differential computing engine with simple dataflow
- Sign-mask dataflow integration
- Quantization scheme and model integration
- Full system integration and initial benchmarking
- Optimization and final evaluation

We prefer to work on Cambricon-D's core hardware accelerator SystemVerilog implementation to develop the actual differential computing architecture.

Group Members (Group 18)

Athul John Kurian, Avinash Singh, Shubham Kumar, Shubham Santosh Kumar