ECE046853 - Advanced Computer Architectures
Winter 2022/23 - Project Proposals

Nimrod Admoni, Samer Kurzum, Uri Weiser

December 2022

1 Guidelines

The course project is an opportunity to conduct an initial research work in a computer architecture related topic, which could even be a stepping stone to a Master thesis. You may choose to work on a project instead of attending the final exam. You can choose working in a team of two or by yourselves. You will be required to do the following for the final project:

1. Choose a project and submit a one page proposal. This must detail *what* you plan to do, *why* you plan to do it, and *how* you plan to do it. **Deadline is 16/01/2023**.
    - Projects may require Python, specifically PyTorch. Some projects may require C/C++, CUDA, and SystemVerilog.
    - For most project we have some written code you can use.
    - Feel free to contact the TAs for questions.
2. Submit a detailed final report. The report should be about 5 page long (not including the reference section), written in English, in a two-columns journal/conference template that we'll provide[1]. The report must include these sections: abstract, introduction, (related work), implementation, evaluation, and insights/conclusions. **Deadline is 06/03/2023**.
3. Prepare an oral presentation of 20 minutes. Presentation will take place on the first week of 2023 spring semester (19/03/2023 – 23/03/2023). **Attendance is mandatory**.

[1]We recommend writing the report in Overleaf, online LaTeX editor.

2 Projects

2.1 Non-Blocking SMT Buffered Systolic Array

The following project is related to non-blocking simultaenous multithreading (NB-SMT). NB-SMT is a new approach to tackle sparsity and increase hardware efficiency [2]. In the same manner that SMT keeps several hardware threads to increase utilization of hardware resources, we propose maintaining a number of "DNN threads" that run in parallel so as to increase utilization of DNN hardware resources. In practice, NB-SMT "squeezes" two, or more, threads together to shared resources by temporarily reducing their numerical precision.

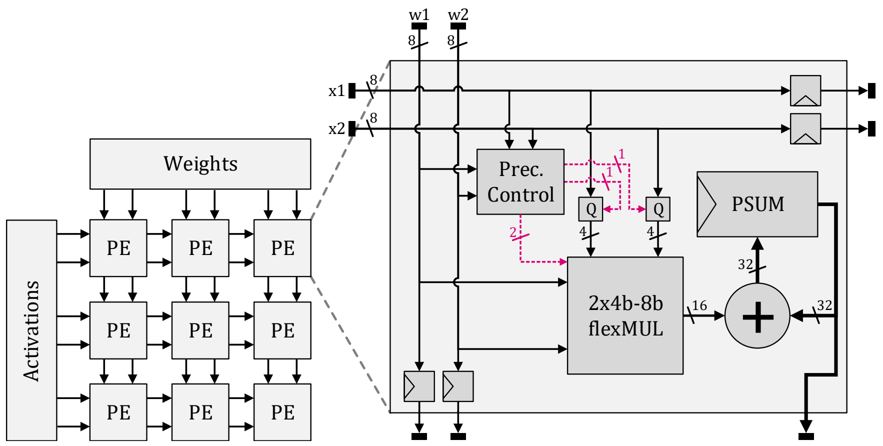

Figure 1: NB-SMT systolic array structure.

Abstract. In this project, you will explore a hybrid of a buffered SMT systolic array [1] and a non-blocking systolic array. That is, buffers will hide threads collisions, thereby eliminating some precision reductions.

Project Goals.

- Implement a buffered systolic array simulator with multiple thread number (we may already have a C/C++ implementation of a blocking multithreaded systolic array [1]).
- Explore the buffered systolic array implementation with a number of quantized CNN models.
- Show the models' accuracy as a function of the buffer sizes and thread number and point out insights on performance. Analyze buffers occupancy.
- Is there a preferable scheme to manage the buffers? do not forget, the hardware is also a consideration.

2.2 Model Resiliency in Training with Floating-Point Representation

Abstract. Training deep neural networks consist of larger amount of Multiply-and-accumulate operations in comparison to inference. Efficiency in training is critical for the overall power and energy consumed. In this project, you will examine models resiliency on floating-point representation. You will explore the different training passes and evaluate the effects of decreasing precision in floating point values, that is, decreasing the bits allocated for the mantissa representation.

Project Goals.

- Simulate 32-bit and 16-bit floating point system on multiple models. Your baseline will be training in BF16.

- Test your model with a reduced representation of floating-point precision. Examine different number of bits allocated for the mantissa, within the convolutional layers multiplication in all three training passes.
- bound the minimal number of bits assigned for the exponent, point out impacts on model performance.
- Explore the effects of applying the reduced representation within different parts of the computation. How accuracy is affected if it is applied over the accumulators in addition to the multiplications?
  - Bonus - Evaluate energy and power savings as a result of the reduced precision.

2.3 Asymmetric Dynamic Quantization in Training DNN

Abstract. DNN training is an energy-consuming operation, yet it is fault-tolerant. Quantization is a technique used to simplify computation by using a new representation which is less energy consuming. Asymmetric Dynamic Quantization (ADQ) examines one matrix for each layer dynamically. For example, let's W and A be the matrices of the wights and activations respectively. Then ADQ will be applied on one of the matrices in the forward pass $W * A$. For a matrix, multiplications are divided to pairs, and for each pair, the large value remains unchanged, while the second value is quantized (e.g. reduced number of precision bits, assign zero, etc.).

In training, you will examine multiple flows, as opposed to inference.

Project Goals.

- Simulate 32-bit and 16-bit floating point system on multiple models. Your baseline will be training in BF16.
- Apply ADQ to the forward pass, backward pass, and weight updates and examine your results.
- Apply ADQ within different epochs of computations. For example, for half of the epochs train with ADQ while the other half train with normal FP32.
- Propose an architecture that can take advantage of your best ADQ. Analyze area and power efforts in comparison to a traditional FP32 training architecture.
  - Bonus - Examine how different ADQ criteria affects accuracy performance.

2.4 Analysis of a Massively Simultaneous Multi-Threading (MSMT) Engine

Abstract. Super-scalar machines consist of a wide machine that can execute multiple instructions concurrently. Due to under-utilization of these systems the SMT was introduced to execute multiple threads simultaneously on the same hardware. We would like to check the variability of MSMT machine to improve the overall utilization and throughput while compromising the single thread performance. In this project, you will analyze the basic bottle-necks of an advanced In-Order super-scalar system running MSMT. The project you will build a simulator to run on benchmark traces that will be provided (e.g. grpah500, Coral, Spec17 FP). The simulator will be based on dependency of instructions which will be analyzed at a given instruction window and availability of resources to evaluate the potential of MSMT performance and Throughput.

Project Goals.

- The simulator will consist of a machine with pre-defined cache miss and branch miss prediction rates to further understand the resources utilization in thread-saturated environment as well as the single thread performance.
- The simulator will use a two-level thread scheduler. The first level (L1) is a pool of all threads that are ready to execute, while the second level (L2) consists of threads pending for execution due to cache miss or branch misprediction.

- Run benchmark on an in-order super-scalar simulator with multiple threads.
- Analyze utilization of single thread and the impact of additional threads on machine's throughput and single thread's performance. Provide the graph of Throughput (and Single thread performance) vs. number of threads.
- The simulator will have the following configuration:
  - Scheduling policies
  - Instruction window size
  - Number of execution units and their corresponding latencies
  - Number of threads

Abstract. Motivated by these observations, we would like to check model resiliency of a decreased with of partial sum representation, which is usually set to 32 bits.

2.5 Extremely low-bit Representation for DNN Inference

Abstract. Approximate computing "fits like a glove" to the new environment introduced by deep neural networks (DNNs). First, DNNs are usually trained and used for error-tolerant applications; and second, DNNs exhibit inherent algorithmic resiliency to some inaccuracies in their intermediate results, for example, their activations and weights can be pruned and quantized with minor degradation in accuracy [2]. Motivated by these observations, we would like to check model resiliency of a decreased width of partial sum representation, which is usually set to 32 bits, but with diffrenent approch – instead of using integer representation we would like to explore very low floating-point representations.

Project Goals.

- Explore the number of actual partial sum bits required for a number of precision-reduced CNN models in different granularities: model, layer, and channel. further analyze mantisa and exponent divisions and bias values.
- Cherry pick a number (4) of precision formats for predefined acceptable accuracy degradation levels, and bound the number bits required for the partial sum register.
- Evaluate impact on power & area compared to similar performance integer-quantisized networks.

2.6 Your Idea!

Have a cool idea related to the course material? Write it down, send it over, and schedule a meeting. We are open for ideas.

References

[1] Gil Shomron, Tal Horowitz, and Uri Weiser. SMT-SA: Simultaneous multithreading in systolic arrays. *IEEE Computer Architecture Letters*, 18(2):99–102, 2019.

[2] Gil Shomron and Uri Weiser. Non-blocking simultaneous multithreading: Embracing the resiliency of deep neural networks. In *International Symposium on Microarchitecture (MICRO)*, 2020.