### Preliminary Project Planning Form

Due day: 2:00pm 10/27/2021

One per team. Submit to the course website on Moodle.

(Grades of this form is part of final project. **Please answer with cautions!**)

TEAM Name: 凱斯雷伯但沒有柏維專家

Team Leader Name: 魏晉成

Members Name: 楊芸甄

李秉軒

劉彥麟

陳奕瑋

戴源

|  |  |
| --- | --- |
| **Target Application for ASPU or Duo-Core** | TPU Accelerator Evaluated by MobileNet v2 |
| Please describe your target application with short motivation and key components that will be related to your application processor | Overview:  Applications of deep learning have been thriving these days, such as image classification, object detection, and image segmentation. With these applications, we can enable developments in robotics, self-driving cars and computer vision.  As a proof-of-concept application, we choose MobileNet v2[1], a widely used model for edge devices, as our target application to run on our system. Based on MobileNet v2, we can perform rather low multiplication-accumulation operations with accuracy that top-1 is 70% and top-5 is 90% in 1000-category classification.  Key Components:  In DNN inference, there are multiplication-accumulation (MAC) operations, activation operations and pooling operations. Among all the above operations, MAC operations take most of the computational power, because one of these operations involves one multiplication and one addition. On the other hand, activation or pooling take less complexity and occur less frequently than MAC operation. Taking our target application, MobileNet v2, for example, the number of MAC operations is 300 million. As a result, our system architecture would be mainly optimized for MAC operations. |
| *Please describe your application with targeting specification and how the application processor will work with CPU & memory on both hardware and software sides.* | Overview:  With neural network models getting deeper, computational power required to process MAC operations is higher nowadays to run a DNN model. Thus, we want to build a generic DNN-accelerating EPU as a co-processor in our system. This EPU would offload MAC operations from RV64IM CPU and reduce latency it takes to perform DNN model inference.  Specification:   * Data flow: Systolic array * Size of MAC array: 288 MAC * Data type: 8-bit weight * Inference latency: 480 ms * Live demo: 2 FPS   To build this EPU, we would take the successful experience from TPU [2] to build our dataflow as systolic array in MAC array. By applying pretrained 8-bit quantized model [3], we would perform inference with top-1 accuracy at around 70%.  As for inference latency, we take two different factors into account. One is data processing latency and the other is data access latency. For data processing latency, because we would apply an 18x16 systolic array to increase utilization of 3x3 convolution, we would make 288 MAC/cycle. Preliminary, we are targeting 100MHz frequency, so that would be 28.8 GMAC/s. For our application MobileNet v2, which in total has 300 million MACs, it would take 10.4 ms to execute one inference optimistically.  On the other hand, DRAM accesses would also cause latency in execution, we take an optimistic estimation that DRAM would be all row hits and no bottleneck occur on the path from DRAM to TPU. On a conventional 2400MT/s DRAM device with 64-bit width, we can presume that DRAM can be accessed at 1.92GB/s. As for estimation in number of bytes needed to be transferred, due to 300 million MACs in one inference, 3 DRAM accesses, input, weight and output, would occur in worst case. In total, there would be 900 million accesses per inference. Under these assumptions, it would take 469 ms to transfer all data in one inference.  In total, it would take 480 ms to perform an inference. Thus, we are targeting 2 FPS in our live demo, trying to beat Raspberry Pi 3, quadcore 1.2 GHz ARM processor [4] by implementing TPU in 100MHz processor.  Execution Flow:  (Fig. 2) Simulation Architecture    (Fig. 3) Simulation Flow  In RTL level simulation, for both pre- and post-synthesis simulation, testbench would setup DRAM with an input picture and compiled program for CPU to perform inference with TPU (1). Later, after CPU has gone into inference block (2), CPU would first move weight into weight SRAM in TPU (3), then move input into input SRAM in TPU (4), and later command systolic array to perform MAC with current data in input and weight SRAMs (5). After the systolic array has finished processing data in SRAMs, generating output data in output SRAM (6), TPU would interrupt CPU and CPU would move data in output SRAM back to DRAM (7). Later, CPU would continue steps 3~7 until whole model has been processed.    (Fig. 4) Demo Architecture    (Fig. 5) Demo Flow  There’s not much difference between simulation flow and demo flow, except for an additional CPU side in PS side of ZCU-104 FPGA. Because it’s hard to add a driver to use web camera on our RISC-V CPU, so we want to utilize embedded CPU on FPGA. As a result, from step 1~4, CPU on PS side in FPGA would read image from web camera and store the image into DRAM segment shared by CPU in PS side and RISC-V CPU in PL side. Later, CPU in PL side would perform inference like what it has performed in simulation. After CPU in PL side has done inference, it would move result back to shared segment in DRAM. Finally, CPU in PS side would display inference result. | |
| *Please provide task assignment for every member. There shall be at least one person dedicate to verification of IPs.* | 魏晉成、楊芸甄: Implementation group  李秉軒、劉彥麟: Verification group  陳奕瑋、戴源: Software group  Implementation group:  This division is to implement whole system, including TPU, CPU, bus, DMA, wrappers, and interconnections. After pre-synthesis verification, this group would implement this system on ZCU104 FPGA for further evaluation in real world detection.  Verification group:  This division would be dedicated to writing testbench and would be co-operating with implementation group to make the system closer to what we expected it to be.  Software group:  This division would need to generate golden data processed by stable neural network framework, so that verification group can utilize these data to do corresponding verification. In the second half of the implementation period, they would implement the driver for our TPU on RISC-V CPU so that inference can take place on our systolic array. | |
| *Please provide project time schedule by providing all members milestones for their own tasks using a chart. Note that please plan by week and DO check the dates for demo and final presentation.*  **Be aware of the schedule posted on the course website.** | (Fig. 6) Workload in Phase 1  In phase 1, i.e., until the deadline of homework 2, we would focus on TPU implementation and verification.    (Fig. 7) Workload in Phase 2  In phase 2, i.e., until the deadline of homework 3, we would focus on CPU-TPU communication and its corresponding testbench. At the same time, the software group would also be dedicated to modification of NN framework, so that we can port our TPU on it.  (Fig. 8) Workload in Phase 3  In phase 3, i.e., until the deadline of homework 4, the software group would keep on working TPU-compatible framework. Implementation and verification groups would focus on tiling and data movement.    (Fig. 9) Workload in Phase 4  In phase 4, which is the boosting period for our final project. Verification group would work on layer-wise and whole-model inference verification in the first two weeks. At the same time, implementation and software groups would be working on porting systems onto ZCU-104 FPGA and the communication between already existing CPU at the PS side and RISC-V CPU at PL side.  In last two weeks, we would be dedicated to synthesizing our system and evaluating it with RTL simulation and FPGA implementation so that we can demo our system timingly. | |

Reference:

[1] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[2] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & Yoon, D. H. (2017, June). In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture (pp. 1-12).

[3] Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342.

[4] Jetson Nano: Deep Learning Inference Benchmarks. (2021, January 5). NVIDIA Developer. https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks