ECE 479 ICC: IoT and Cognitive Computing

Spring 2023, Homework 3

Please submit your answers by using LaTeX or Word compiled pdf.

Question 1: GPU and GPU programming (25 pts)

1. List at least three significant differences between GPU and CPU. [**3 pts**]

2. Parts of a GPU: write down the letter that describes the part [**8 pts**]

   A. A group of threads (typically 32 for NVIDIA) executed concurrently

   B. The smallest unit of execution

   C. The largest memory pool available in a GPU, accessible by all threads

   D. Specialized hardware components designed to accelerate mixed-precision matrix arithmetic

   E. General-purpose, high-speed serial bus standard widely used in computers to connect various peripherals

   F. A smaller, faster memory pool located within each SM, accessible by all threads within a single thread block executing on that SM

   G. The highest level of abstraction for organizing threads in a GPU kernel launch

   H. A larger group of threads that enables flexible synchronization and scheduling

   I. Point-to-point interconnect technology that enables faster communication between multiple GPUs or between GPUs and CPUs

   J. The fundamental building block of a GPU, consisting of multiple processing cores

( ) Grid

( ) Block

( ) Warp

( ) Thread

( ) Tensor Core

( ) NVLink

( ) Shared Memory

( ) Global Memory

3. Consider the image blurring algorithm in lecture 15, slide 44. Answer the following questions.

   (a) To blur a monochrome image with the size of $(600, 400)$, calculate the Grid dimension for the kernel given the block size of $(16, 16, 1)$. **[3 pts]**

   (b) Does the `BLUR_SIZE` have effects on your calculated Grid dimension? **Justify your answer. [3 pts]**

   (c) If `BLUR_SIZE` is 2, and height and width of the image are much larger than `BLUR_SIZE`. How many different possible values could variable `pixels` take right before line 10 (exit the for loop) of the algorithm? (**Hint:** consider different boundary conditions) **[3 pts]**

4. Please review the concepts regarding to GPU programming and embedded GPU. For each of the following statements, write down True or False according to your best knowledge. To receive full credit for each False statement you identify, you need to **justify your answer. [5 pts]**

   (a) GPU is winning CPU in all scenarios thanks to its massive number of execution units with zero-latency context switch.

   (b) Threads are grouped into blocks for easy-managing and better cooperation, and thus blocks are scheduling and execution units in GPU architecture.

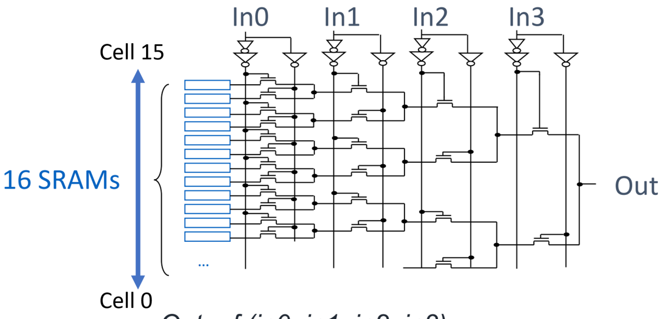
   (c) By mapping an algorithm onto the GPU, we could decrease the computational complexity big-$\mathcal{O}$ of the algorithm by exploiting the parallelism in the algorithm and the hardware.

   (d) Tensor Cores accelerate training and inference of DNN by massive concurrent Multiply-Add operations.

   (e) Similar to desktop-level (full-powered) GPUs, mobile (embedded) GPUs like Jetson Xavier NX also suffer latency concerns while copying data between host and device.

Question 2: FPGA Basics and High-Level Synthesis (20 pts)

1. Suppose that we have a simple 4-bit look-up table as shown below (only the upper 12 SRAM cells are shown):



Out = f (in0, in1, in2, in3)

(a) Briefly explain why this LUT can work as any 4-bit binary logic function. (4-bit input and 1-bit output) **[3 pts]**

(b) Suppose that we want to access cell 9 of this LUT. Write down the input to the four input bits (MSB-LSB [in3 in2 in1 in0]) and highlight the activated pass transistors on the path to the output in the diagram. **[3 pts]**

2. In the following toy HLS C code snippet, there are multiple opportunities to apply the optimization techniques. Apply at least 2 of them and write down the line number where you can insert the pragma. Describe in one or two sentences if you need to change the code structure. [**6 pts**]

```
1    void top(){
2      int array[100];
3      foo(array);
4      bar(array, 20);
5      return;
6    }
7
8    void foo(int array[100]){
9      for (int i=0;i < 100;i ++){
10       array[i] = 0;
11     }
12     return;
13   }
14
15   void bar(int array[100], int boundary){
16     for (int i=0;i < 100;i ++){
17       array[i] = i;
18     }
19     for (int i=50;i < 100;i ++){
20       array[i] *= 2;
21     }
22
23     for (int i=0;i < boundary;i ++){
24       array[i] += 10;
25     }
26     return;
27   }
28
```

3. In the following HLS C code for a convolution, there is a pipelining pragma at line 10. Read the code and answer the following questions.

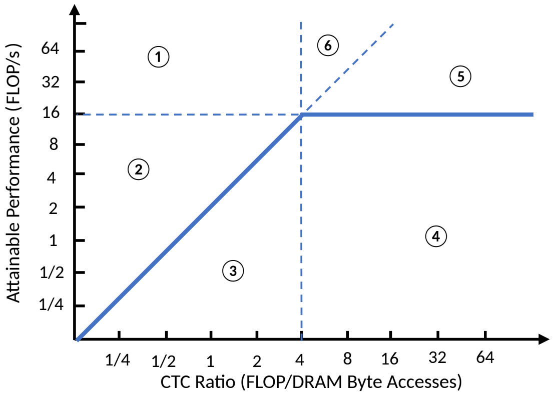
```
1  void conv3x3(
2      int32 feature[3][112][112],
3      int32 kernel[32][3][3][3],
4      int32 output[32][110][110])
5  {
6      for(int x_out = 0; x_out < 110; x_out ++) {
7          for(int c_out = 0; c_out < 32; c_out ++) {
8              for(int y_out = 0; y_out < 110; y_out ++) {
9                  int32 acc = 0;
10 #pragma HLS PIPELINE
11                 for(int c_in = 0; c_in < 3; c_in ++) {
12                     for (int y_k = 0; y_k < 3; y_k ++) {
13                         for (int x_k = 0; x_k < 3; x_k ++) {
14                             acc += kernel[c_out][c_in][y_k][x_k]
       * feature[c_in][y_k + y_out][x_k + x_out];
15                         }
16                     }
17                 }
18                 output[c_out][y_out][x_out] = acc;
19             }
20         }
21     }
22 }
23
```

(a) Explain what is "II" when we talk about pipelining. Can we achieve II = 1 in this code if no other optimization is done? Why or why not? Assuming the memory has two ports for read and write. (Hint: How many concurrent memory access do we need to achieve II = 1?) [**5 pts**]

(b) If we want to achieve II = 1, what other optimization technique do we must perform? [**3 pts**]

Question 3: Efficient DNN Accelerator (20 pts)

1. **Roofline Model**
   Suppose that a naive accelerator design falls in region 2 of the roofline in the following diagram.



   (a) Is this a feasible design? Why or why not? Which regions indicate a feasible design? [**3 pts**]

   (b) What does it mean when the graph "flats out" after the CTC ratio is greater than 4 FLOP/DRAM? [**2 pts**]

   (c) What is the key challenge that this design is facing in terms of improving performance? Propose a possible technique to solve this challenge. [**2 pts**]

2. **Depthwise-pointwise Convolution**

   (a) Standard convolution takes an $h_i \times w_i \times d_i$ (representing height, width, and depth) input tensor L_i, and applies convolutional kernel $K \in R^{k \times k \times d_i \times d_j}$ to produce an $h_i \times w_i \times d_j$ output tensor L_j with *stride* $= 1$. Please compute the number of operations (regarding both multiplications and additions). [**2 pts**]

(b) Depthwise convolution takes an $h_i \times w_i \times d_i$ input tensor L_i, and applies convolutional kernel $K \in R^{k \times k \times d_i}$ to produce an $h_i \times w_i \times d_i$ output tensor L_j with $stride = 1$. Please compute the number of operations (regarding both multiplications and additions). **[2 pts]**

(c) Pointwise convolution takes an $h_i \times w_i \times d_i$ input tensor L_i, and applies convolutional kernel $K \in R^{1 \times 1 \times d_i \times d_j}$ to produce an $h_i \times w_i \times d_j$ output tensor L_j with $stride = 1$. Please compute the number of operations (regarding both multiplications and additions). **[2 pts]**

(d) By combining the depthwise and pointwise convolutions, we can replace the standard one as described in Lecture 19 page 44. Compute the number of operations in a depthwise convolution and a pointwise convolution in (b) and (c), and compare it to the operation number of a standard convolution (a). What is the ratio of operation reduction if using a depthwise separable convolution instead of a standard one? Please explain why it can save operations. **[3 pts]**

(e) If other hyperparameters are kept the same, replacing normal convolution with the combination of depthwise and pointwise convolutions may cause other issues. (1). When we design accelerators on resource-constrained embedded devices such as a small FPGA, what can be a problem using depthwise and pointwise convolutions regarding the hardware design? (2). Compared to the normal convolution, do depthwise and pointwise convolutions have better or worse accuracy? Why? **[4 pts]**

Question 4: Attention is all you need (15 pts)

1. Describe one similarity and one difference between Computer Vision tasks and Natural Language Processing tasks in machine learning. [**4 pts**]

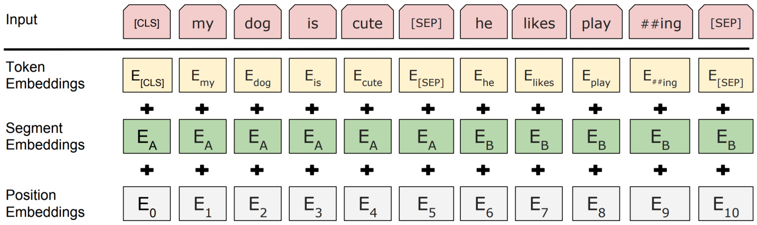
2. When constructing the Q-K matrix, we find that the Query has a dimension of $(64, 1024, 25)$ (batch size, channel depth, sequence length).

   (a) We have 32 heads in the multi-head attention structure. What are the dimensions of Q' fed into each attention module after the head-split? [**1 pts**]

   (b) What is the dimension of the Q-K matrix in each attention module? [**1 pts**]

   (c) Think about the matrix multiplication operation and the number of the Q-K matrices produced by the head-split. Explain why multi-head is necessary. [**3 pts**]

3. Referring to the following picture about BERT, answer the following questions.



   (a) Explain in two sentences how BERT training differs from GPT at the time. [**2 pts**]

(b) How does the “Segment Embeddings” help BERT train? [**2 pts**]

(c) Why do Transformer models need “Position Embeddings”? [**2 pts**]

4. **(Bonus)** Follow the MidJourney Quick Start instructions, create a picture with your favorite animal doing some human activities. Post your picture here and on Piazza. The bonus score is given based on your creativity. Note that sometimes the server is very crowded so you may need to wait for the next day to generate your artwork. [**5 pts**]

Question 5: IoT Security and Cybersecurity (20 pts)

1. In the lecture, we have introduced three security solutions proposed for Intel, ARM, and RISC-V architectures. The common strategy, on which these approaches all focus, is *Isolation.* Pick one scenario from Intel SGX, ARM TrustZone, and RISC-V Keystone, and on a high level, explain in roughly three sentences how they implement this strategy. [**9 pts**]

2. Many edge devices operate on battery power. They are usually in sleep mode when no data processing tasks are needed. Under such an energy constraint, the device is prone to Denial-of-Sleep attacks from malicious nodes. Consider a simple IoT device that operates on battery power. To avoid the Man-in-the-Middle attack where a malicious node intercepts the packet halfway in the transmission, the designers of this IoT device have decided to encrypt both the data and the header. Upon receiving the packets, the device first decrypts the header to identify whether the packet comes from a trusted node. If not, the packet is discarded.

   (a) Explain in one sentence how Denial-of-Sleep attack affects an IoT edge device. What specifically does it attack? [**3 pts**]

   (b) As the attacker, you know how to send an encrypted packet to the device, but you cannot authenticate yourself as a trusted node after the device has decrypted your packet. Nevertheless, you can still launch a Denial-of-Sleep attack from your malicious node. Briefly explain how you can conduct this attack. Hint: encryption and decryption data can consume a considerable amount of energy. [**2 pts**]

3. In the lectures, we have discussed one specific Cyber-attack malware called Mirai. It is used in some massive DDoS attacks.

   (a) What is a DDoS attack? How does it differ from regular DoS attacks? [**2 pts**]

   (b) On a high level, what devices does Mirai take advantage of? [**2 pts**]

   (c) We know that Mirai is badly written. How does it succeed in infecting many devices? [**2 pts**]