

Hardware System Design, Lab 7

2013-11392 김지현

개요

이번 과제의 목표는 convolution lowering을 구현하는것으로, Lab 2의 결과를 일부 가져다 써야한다. Convolution lowering 이외의 모든 소스코드는 뼈대코드로 모두 주어져있어, 구현해야하는 부분이 많지 않다.

구현체 설명

먼저 LargeMV() 함수 구현은 Lab 2에서 구현한 것을 그대로 사용하였다. 이미 Lab 2에서 풀었던 문제이므로 LargeMV에 대한 설명은 따로 덧붙이지 않겠다.

```
for (int row = 0; row < conv_channel; ++row) {
    for (int z = 0; z < input_channel; ++z) {
        for (int y = 0; y < conv_height; ++y) {
            for (int x = 0; x < conv_width; ++x) {
                new_weights[row][z*conv_height*conv_width + y*conv_width + x] =
                    cnn_weights[row][z][y][x];
            }
        }
    }
}

const int row_count = input_height - conv_height + 1;
const int col_count = input_width - conv_width + 1;
for (int z = 0; z < input_channel; ++z) {
    for (int y = 0; y < conv_height; ++y) {
        for (int x = 0; x < conv_width; ++x) {
            for (int offset_y = 0; offset_y < row_count; ++offset_y) {
                for (int offset_x = 0; offset_x < col_count; ++offset_x) {
                    new_inputs[z*conv_height*conv_width + y*conv_width + x][offset_y*col_count + offset_x] =
                        inputs[z][offset_y + y][offset_x + x];
                }
            }
        }
    }
}
```

Figure 1. FPGA::convLowering() 함수 구현

convLowering() 함수는 수업시간에 배운것과 동일하게 구현하였다. 다만, 수업 PPT에 있는 convolution lowering 예시 그림에서 inputs를 new_inputs로 바꾸는 부분에 순서 오류가 있어, 해당 부분은 고쳐서 구현하였다. Convolution filter를 바로 적용시키는것과, convolution lowering을 수행한 뒤 단순 행렬곱을 수행하는것의 결과가 같아지도록 만드는데에 초점을 맞춰 구현했다.

실험 결과 분석

```
root@01c64332252:~/snucse.hsd/lab-07# ./benchmark
[*] Arguments: Namespace(m_size=64, network='mlp', num_test_images=100, run_type='cpu', v_size=64)
[*] Read MNIST ...
[*] The shape of image: (100, 28, 28)
[*] Load the network ...
[*] Run tests ...
[*] Statistics ...
{'accuracy': 0.97,
 'avg_num_call': 627,
 'm_size': 64,
 'total_image': 100,
 'total_time': 0.33359718322753906,
 'v_size': 64}

⇒ Accuracy should be 0.97

[*] Arguments: Namespace(m_size=64, network='cnn', num_test_images=100, run_type='cpu', v_size=64)
[*] Read MNIST ...
[*] The shape of image: (100, 28, 28)
[*] Load the network ...
[*] Run tests ...
[*] Statistics ...
{'accuracy': 1.0,
 'avg_num_call': 741,
 'm_size': 64,
 'total_image': 100,
 'total_time': 0.4174790382385254,
 'v_size': 64}

⇒ Accuracy should be 1.0
```

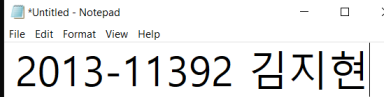


Figure 2. benchmark 실행 결과

과제에서 요구하는 것과 동일한 정확도가 나온 것을 확인할 수 있다. Convolution Lowering만 새롭게 구현한 것이고, 행렬곱 부분은 구현이 바뀌지 않았기 때문에 block size에 따른 성능 변화는 Lab 2와 크게 차이 나지 않았다.

결론

수업시간에서 복잡한 연산인 Convolution filtering을 어떻게 행렬곱으로 바꾸어 생각할 수 있는지 배웠는데, 이를 직접 실습해볼 수 있었던 간단하고 좋은 과제였다. 과제에서 주어진 CNN 모델의 성능이 MLP 모델보다 훨씬 뛰어났던 부분도 인상적이었다. 이번 과제에선 CNN 연산을 CPU로만 수행했는데, 이 연산을 FPGA로 대체하는 실습을 빨리 해보고 싶었다.