single_layer_test

January 7, 2022

0.1 # DSD-2021 Project (CIFAR-10)

0.2 Layerwise Inference Check

- This is a python script to help you check if your RTL impelementation for each layer is correct

0.3 Usage

- Run `Board connection`, `Setting the VDMA`, `Save parameters`
- Run the layers that you want to experiment in `Inference`
- Example : To test Convolution 1 + ReLU, run the below two cells
  - Cell 1

```
######################################################################
#        Convolution 1 + ReLU
######################################################################
# Convolution
# - in:       (n, 3, 32, 32)
# - out:     (n, 32, 28, 28)
# - weight:    (32, 3, 3, 3)
# - bias:               (32)
# ReLU
# - in:      (n. 32. 32. 32)
# - out:     (n. 32. 32. 32)
######################################################################
I = {'IN_CH': 3, 'OUT_CH': 32, 'FLEN': 32}
F = {'BASE_ADDR': 0x0000_0000, 'STRIDE_SIZE': 3*32*32, 'HSIZE': 3*32*32, 'VSIZE': 1}
W = {'BASE_ADDR': 0x0200_0000, 'STRIDE_SIZE': 3*32*9, 'HSIZE': 3*32*9, 'VSIZE': 1}
B = {'BASE_ADDR': 0x0210_0000, 'STRIDE_SIZE': 32, 'HSIZE': 32, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0600_0000, 'STRIDE_SIZE': 32*32*32, 'HSIZE': 32*32*32, 'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

  - Cell 2

```
SU.check_result("./data/new_cifar10_random_data/conv1_relu_out_2s.txt", 0x0600_0000)
```

[1]:
```
from utils.scale_uart import *
from utils.board import *
import time
import time
```

0.4 Board connection

[2]:
```
port_list = get_port_list()
SU = get_scale_uart(port_list)
```

```
Current OS: Windows
['COM1', 'COM4']
COM1 port cannot be connected.
COM4 port connected!
```

0.5 Setting the VDMA

[3]:
```
## DO NOT CHANGE
## IT IS VDMA AND EACH MODULE'S BASE ADDRESS FOR CONTROL APB + AXI
##### PARAMETER INFORMATION
VDMA0_BASE_ADDR= 0x0c00_0000
VDMA1_BASE_ADDR= 0x0c10_0000
VDMA2_BASE_ADDR= 0x0c20_0000

FC_BASE_ADDR   = 0x0d00_0000
CONV_BASE_ADDR = 0x0d10_0000
POOL_BASE_ADDR = 0x0d20_0000

### FIXED FOR OUR NETWORK
OP_SIZE                         = 4
ADDR_SIZE                       = 28
DATA_SIZE                       = 32
```

0.6 Save parameters

[4]:
```
print("conv1 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x0200_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv1_weight_quan.npy")
SU.su_set_conv_b({'BASE_ADDR': 0x0210_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv1_bias_quan.npy")
print("conv1 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv1 parameter load
conv1 set done
        Total time: 0.11 sec
```

[5]:
```
print("conv2 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x0220_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv2_weight_quan.npy")
```

```
SU.su_set_conv_b({'BASE_ADDR': 0x0270_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv2_bias_quan.npy")
print("conv2 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv2 parameter load
conv2 set done
        Total time: 2.13 sec
```

[6]:
```
print("conv3 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x0280_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv3_weight_quan.npy")
SU.su_set_conv_b({'BASE_ADDR': 0x02C0_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv3_bias_quan.npy")
print("conv3 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv3 parameter load
conv3 set done
        Total time: 8.37 sec
```

[7]:
```
print("conv4 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x0300_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv4_weight_quan.npy")
SU.su_set_conv_b({'BASE_ADDR': 0x0390_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv4_bias_quan.npy")
print("conv4 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv4 parameter load
conv4 set done
        Total time: 16.89 sec
```

[8]:
```
print("conv5 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x03A0_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv5_weight_quan.npy")
SU.su_set_conv_b({'BASE_ADDR': 0x03F0_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv5_bias_quan.npy")
print("conv5 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv5 parameter load
conv5 set done
        Total time: 33.86 sec
```

[9]:
```
print("conv6 parameter load")
start = time.time()
SU.su_set_conv_w({'BASE_ADDR': 0x0400_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv6_weight_quan.npy")
SU.su_set_conv_b({'BASE_ADDR': 0x0490_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_conv6_bias_quan.npy")
print("conv6 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
conv6 parameter load
conv6 set done
        Total time: 67.67 sec
```

[10]:
```
print("fc1 parameter load")
start = time.time()
SU.su_set_fc_w({'BASE_ADDR': 0x0500_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc1_weight_quan.npy")
SU.su_set_fc_b({'BASE_ADDR': 0x0530_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc1_bias_quan.npy")
print("fc1 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
fc1 parameter load
fc1 set done
        Total time: 29.81 sec
```

[11]:
```
print("fc2 parameter load")
start = time.time()
SU.su_set_fc_w({'BASE_ADDR': 0x0540_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc2_weight_quan.npy")
SU.su_set_fc_b({'BASE_ADDR': 0x0550_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc2_bias_quan.npy")
print("fc2 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
fc2 parameter load
fc2 set done
        Total time: 1.88 sec
```

[12]:
```
print("fc3 parameter load")
start = time.time()
SU.su_set_fc_w({'BASE_ADDR': 0x0560_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc3_weight_quan.npy")
SU.su_set_fc_b({'BASE_ADDR': 0x0570_0000}, "./data/cifar10_network_quan_param/
 ↪cifar10_fc3_bias_quan.npy")
print("fc3 set done")
print("\tTotal time: {:.2f} sec".format(time.time() - start))
```

```
fc3 parameter load
```

```
fc3 set done
        Total time: 0.08 sec
```

0.7 Inference

[13]:
```
SU.save_file("./data/new_cifar10_random_data/input_2s.txt", 0x0000_0000)
```

[14]:
```
######################################################################
#        Convolution 1 + ReLU
######################################################################
# Convolution
# - in:        (n, 3, 32, 32)
# - out:      (n, 32, 28, 28)
# - weight:     (32, 3, 3, 3)
# - bias:                (32)
# ReLU
# - in:       (n. 32. 32. 32)
# - out:      (n. 32. 32. 32)
######################################################################
I = {'IN_CH': 3, 'OUT_CH': 32, 'FLEN': 32}
F = {'BASE_ADDR': 0x0000_0000, 'STRIDE_SIZE': 3*32*32, 'HSIZE': 3*32*32,␣
 ↪'VSIZE': 1}
W = {'BASE_ADDR': 0x0200_0000, 'STRIDE_SIZE': 3*32*9, 'HSIZE': 3*32*9, 'VSIZE':␣
 ↪1}
B = {'BASE_ADDR': 0x0210_0000, 'STRIDE_SIZE': 32, 'HSIZE': 32, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0600_0000, 'STRIDE_SIZE': 32*32*32, 'HSIZE': 32*32*32,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[14]:
```
1
```

[15]:
```
SU.check_result("./data/new_cifar10_random_data/conv1_relu_out_2s.txt",␣
 ↪0x0600_0000)
```

```
Count is 1000
Count is 2000
Count is 3000
Count is 4000
Count is 5000
Count is 6000
Count is 7000
Count is 8000
All Results are Correct
```

[16]:
```
#SU.save_file("./data/new_cifar10_random_data/conv1_relu_out_2s.txt",␣
 ↪0x0600_0000)
```

[17]:
```
######################################################################
#        Max Pool 1
######################################################################
# Max Pooling
# - in:       (n. 32. 32. 32)
# - out:      (n, 32, 16, 16)
######################################################################
I = {'IN_CH': 32, 'FLEN': 32}
F = {'BASE_ADDR': 0x0600_0000, 'STRIDE_SIZE': 32*32*32, 'HSIZE': 32*32*32,␣
 ↪'VSIZE': 1}
R = {'BASE_ADDR': 0x0610_0000, 'STRIDE_SIZE': 32*16*16, 'HSIZE': 32*16*16,␣
 ↪'VSIZE': 1}
SU.su_pool_control(I, F, R, VDMA2_BASE_ADDR, POOL_BASE_ADDR)
```

[17]: 1

[18]:
```
SU.check_result("./data/new_cifar10_random_data/pool1_out_2s.txt", 0x0610_0000)
```

```
Count is 1000
Count is 2000
All Results are Correct
```

[19]:
```
#SU.save_file("./data/new_cifar10_random_data/pool1_out_2s.txt", 0x0610_0000)
```

[20]:
```
######################################################################
#        Convolution 2 + ReLU
######################################################################
# Convolution
# - in:       (n, 32, 16, 16)
# - out:      (n, 64, 16, 16)
# - weight:    (64, 32, 3, 3)
# - bias:                (64)
# ReLU
# - in:       (n. 64. 16. 16)
# - out:      (n. 64. 16. 16)
######################################################################
I = {'IN_CH': 32, 'OUT_CH': 64, 'FLEN': 16}
F = {'BASE_ADDR': 0x0610_0000, 'STRIDE_SIZE': 32*16*16, 'HSIZE': 32*16*16,␣
 ↪'VSIZE': 1}
W = {'BASE_ADDR': 0x0220_0000, 'STRIDE_SIZE': 32*64*9, 'HSIZE': 32*64*9,␣
 ↪'VSIZE': 1}
B = {'BASE_ADDR': 0x0270_0000, 'STRIDE_SIZE': 64, 'HSIZE': 64, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0620_0000, 'STRIDE_SIZE': 64*16*16, 'HSIZE': 64*16*16,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[20]: 1

[21]:
```
SU.check_result("./data/new_cifar10_random_data/conv2_relu_out_2s.txt",␣
↪0x0620_0000)
```

```
Count is 1000
Count is 2000
Count is 3000
Count is 4000
All Results are Correct
```

[22]:
```
#SU.save_file("./data/new_cifar10_random_data/conv2_relu_out_2s.txt",␣
↪0x0620_0000)
```

[23]:
```
#######################################################################
#        Max Pool 2
#######################################################################
# Max Pooling
# - in:       (n. 64. 16. 16)
# - out:        (n, 64, 8, 8)
#######################################################################
I = {'IN_CH': 64, 'FLEN': 16}
F = {'BASE_ADDR': 0x0620_0000, 'STRIDE_SIZE': 64*16*16, 'HSIZE': 64*16*16,␣
↪'VSIZE': 1}
R = {'BASE_ADDR': 0x0630_0000, 'STRIDE_SIZE': 64*8*8, 'HSIZE': 64*8*8, 'VSIZE':␣
↪1}
SU.su_pool_control(I, F, R, VDMA2_BASE_ADDR, POOL_BASE_ADDR)
```

[23]:
```
1
```

[24]:
```
SU.check_result("./data/new_cifar10_random_data/pool2_out_2s.txt", 0x0630_0000)
```

```
Count is 1000
All Results are Correct
```

[25]:
```
#SU.save_file("./data/new_cifar10_random_data/pool2_out_2s.txt", 0x0630_0000)
```

[26]:
```
#######################################################################
#        Convolution 3 + ReLU
#######################################################################
# Convolution
# - in:         (n, 64, 8, 8)
# - out:      (n, 128, 8, 8)
# - weight:  (128, 64, 3, 3)
# - bias:              (128)
# ReLU
# - in:       (n. 128. 8. 8)
# - out:      (n. 128. 8. 8)
#######################################################################
I = {'IN_CH': 64, 'OUT_CH': 128, 'FLEN': 8}
```

```
F = {'BASE_ADDR': 0x0630_0000, 'STRIDE_SIZE': 64*8*8, 'HSIZE': 64*8*8, 'VSIZE':␣
 ↪1}
W = {'BASE_ADDR': 0x0280_0000, 'STRIDE_SIZE': int(64*128*9/2), 'HSIZE':␣
 ↪int(64*128*9/2), 'VSIZE': 2}
B = {'BASE_ADDR': 0x02C0_0000, 'STRIDE_SIZE': 128, 'HSIZE': 128, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0640_0000, 'STRIDE_SIZE': 128*8*8, 'HSIZE': 128*8*8,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[26]: 1

[27]:
```
SU.check_result("./data/new_cifar10_random_data/conv3_relu_out_2s.txt",␣
 ↪0x0640_0000)
```

```
Count is 1000
Count is 2000
All Results are Correct
```

[28]:
```
#SU.save_file("./data/new_cifar10_random_data/conv3_relu_out_2s.txt",␣
 ↪0x0640_0000)
```

[29]:
```
######################################################################
#        Convolution 4 + ReLU
######################################################################
# Convolution
# - in:       (n, 128, 8, 8)
# - out:      (n, 128, 8, 8)
# - weight: (128, 128, 3, 3)
# - bias:              (128)
# ReLU
# - in:       (n. 128. 8. 8)
# - out:      (n. 128. 8. 8)
######################################################################
I = {'IN_CH': 128, 'OUT_CH': 128, 'FLEN': 8}
F = {'BASE_ADDR': 0x0640_0000, 'STRIDE_SIZE': 128*8*8, 'HSIZE': 128*8*8,␣
 ↪'VSIZE': 1}
W = {'BASE_ADDR': 0x0300_0000, 'STRIDE_SIZE': int(128*128*9/4), 'HSIZE':␣
 ↪int(128*128*9/4), 'VSIZE': 4}
B = {'BASE_ADDR': 0x0390_0000, 'STRIDE_SIZE': 128, 'HSIZE': 128, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0650_0000, 'STRIDE_SIZE': 128*8*8, 'HSIZE': 128*8*8,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[29]: 1

[30]:
```
SU.check_result("./data/new_cifar10_random_data/conv4_relu_out_2s.txt",␣
 ↪0x0650_0000)
```

```
Count is 1000
Count is 2000
All Results are Correct
```

[31]:
```
#SU.save_file("./data/new_cifar10_random_data/conv4_relu_out_2s.txt",␣
↪0x0650_0000)
```

[32]:
```
#######################################################################
#        Max Pool 3
#######################################################################
# Max Pooling
# - in:       (n. 128. 8. 8)
# - out:      (n, 128, 4, 4)
#######################################################################
I = {'IN_CH': 128, 'FLEN': 8}
F = {'BASE_ADDR': 0x0650_0000, 'STRIDE_SIZE': 128*8*8, 'HSIZE': 128*8*8,␣
↪'VSIZE': 1}
R = {'BASE_ADDR': 0x0660_0000, 'STRIDE_SIZE': 128*4*4, 'HSIZE': 128*4*4,␣
↪'VSIZE': 1}
SU.su_pool_control(I, F, R, VDMA2_BASE_ADDR, POOL_BASE_ADDR)
```

[32]:
```
1
```

[33]:
```
SU.check_result("./data/new_cifar10_random_data/pool3_out_2s.txt", 0x0660_0000)
```

```
All Results are Correct
```

[34]:
```
#SU.save_file("./data/new_cifar10_random_data/pool3_out_2s.txt", 0x0660_0000)
```

[35]:
```
#######################################################################
#        Convolution 5+ ReLU
#######################################################################
# Convolution
# - in:       (n, 128, 4, 4)
# - out:      (n, 256, 4, 4)
# - weight: (256, 128, 3, 3)
# - bias:              (256)
# ReLU
# - in:       (n. 256. 4. 4)
# - out:      (n. 256. 4. 4)
#######################################################################
I = {'IN_CH': 128, 'OUT_CH': 256, 'FLEN': 4}
F = {'BASE_ADDR': 0x0660_0000, 'STRIDE_SIZE': 128*4*4, 'HSIZE': 128*4*4,␣
↪'VSIZE': 1}
W = {'BASE_ADDR': 0x03A0_0000, 'STRIDE_SIZE': int(128*256*9/8), 'HSIZE':␣
↪int(128*256*9/8), 'VSIZE': 8}
B = {'BASE_ADDR': 0x03F0_0000, 'STRIDE_SIZE': 256, 'HSIZE': 256, 'VSIZE': 1}
```

```
R = {'BASE_ADDR': 0x0670_0000, 'STRIDE_SIZE': 256*4*4, 'HSIZE': 256*4*4,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[35]: 1

[36]:
```
SU.check_result("./data/new_cifar10_random_data/conv5_relu_out_2s.txt",␣
 ↪0x0670_0000)
```

```
Count is 1000
All Results are Correct
```

[37]:
```
#SU.save_file("./data/new_cifar10_random_data/conv5_relu_out_2s.txt",␣
 ↪0x0670_0000)
```

[38]:
```
######################################################################
#        Convolution 6 + ReLU
######################################################################
# Convolution
# - in:         (n, 256, 4, 4)
# - out:        (n, 256, 4, 4)
# - weight:  (256, 256, 3, 3)
# - bias:               (256)
# ReLU
# - in:         (n. 256. 4. 4)
# - out:        (n. 256. 4. 4)
######################################################################
I = {'IN_CH': 256, 'OUT_CH': 256, 'FLEN': 4}
F = {'BASE_ADDR': 0x0670_0000, 'STRIDE_SIZE': 256*4*4, 'HSIZE': 256*4*4,␣
 ↪'VSIZE': 1}
W = {'BASE_ADDR': 0x0400_0000, 'STRIDE_SIZE': int(256*256*9/16), 'HSIZE':␣
 ↪int(256*256*9/16), 'VSIZE': 16}
B = {'BASE_ADDR': 0x0490_0000, 'STRIDE_SIZE': 256, 'HSIZE': 256, 'VSIZE': 1}
R = {'BASE_ADDR': 0x0680_0000, 'STRIDE_SIZE': 256*4*4, 'HSIZE': 256*4*4,␣
 ↪'VSIZE': 1}
SU.su_conv_control(I, F, W, B, R, VDMA1_BASE_ADDR, CONV_BASE_ADDR)
```

[38]: 1

[39]:
```
SU.check_result("./data/new_cifar10_random_data/conv6_relu_out_2s.txt",␣
 ↪0x0680_0000)
```

```
Count is 1000
All Results are Correct
```

[40]:
```
#SU.save_file("./data/new_cifar10_random_data/conv6_relu_out_2s.txt",␣
 ↪0x0680_0000)
```

[41]:
```
#####################################################################
#        Max Pool 4
#####################################################################
# Max Pooling
# - in:       (n. 256. 4. 4)
# - out:      (n, 256, 2, 2)
#####################################################################
I = {'IN_CH': 256, 'FLEN': 4}
F = {'BASE_ADDR': 0x0680_0000, 'STRIDE_SIZE': 256*4*4, 'HSIZE': 256*4*4,␣
 ↪'VSIZE': 1}
R = {'BASE_ADDR': 0x0690_0000, 'STRIDE_SIZE': 256*2*2, 'HSIZE': 256*2*2,␣
 ↪'VSIZE': 1}
SU.su_pool_control(I, F, R, VDMA2_BASE_ADDR, POOL_BASE_ADDR)
```

[41]: 1

[42]:
```
SU.check_result("./data/new_cifar10_random_data/pool4_out_2s.txt", 0x0690_0000)
```

```
All Results are Correct
```

[43]:
```
#SU.save_file("./data/new_cifar10_random_data/pool4_out_2s.txt", 0x0690_0000)
```

[44]:
```
#####################################################################
#        Fully-Connected 1 + ReLU
#####################################################################
# Fully-Connected
# - in:              (1024,)
# - out:              (256,)
# - weight:      (256, 1024)
# - bias:             (256,)
# ReLU
# - in:               (256,)
# - out:              (256,)
#####################################################################
F = {'BASE_ADDR': 0x0690_0000, 'STRIDE_SIZE': 1024, 'HSIZE': 1024, 'VSIZE': 1}
W = {'BASE_ADDR': 0x0500_0000, 'STRIDE_SIZE': int(1024*256/8), 'HSIZE':␣
 ↪int(1024*256/8), 'VSIZE': 8}
B = {'BASE_ADDR': 0x0530_0000, 'STRIDE_SIZE': 256, 'HSIZE': 256, 'VSIZE': 1}
R = {'BASE_ADDR': 0x06A0_0000, 'STRIDE_SIZE': 256, 'HSIZE': 256, 'VSIZE': 1}
SU.su_fc_control(F, W, B, R, VDMA0_BASE_ADDR, FC_BASE_ADDR)
```

```
[0, 0, 0, 227]
```

[44]: 1

[45]:
```
SU.check_result("./data/new_cifar10_random_data/fc1_relu_out_2s.txt",␣
 ↪0x06A0_0000)
```

```
All Results are Correct
```

[46]:
```
#SU.save_file("./data/new_cifar10_random_data/fc1_relu_out_2s.txt", 0x06A0_0000)
```

[47]:
```
#########################################################################
#        Fully-Connected 2 + ReLU
#########################################################################
# Fully-Connected
# - in:              (256,)
# - out:              (64,)
# - weight:       (64, 256)
# - bias:             (64,)
# ReLU
# - in:               (64,)
# - out:              (64,)
#########################################################################
F = {'BASE_ADDR': 0x06A0_0000, 'STRIDE_SIZE': 256, 'HSIZE': 256, 'VSIZE': 1}
W = {'BASE_ADDR': 0x0540_0000, 'STRIDE_SIZE': 256*64, 'HSIZE': 256*64, 'VSIZE':␣
 ↪1}
B = {'BASE_ADDR': 0x0550_0000, 'STRIDE_SIZE': 64, 'HSIZE': 64, 'VSIZE': 1}
R = {'BASE_ADDR': 0x06B0_0000, 'STRIDE_SIZE': 64, 'HSIZE': 64, 'VSIZE': 1}
SU.su_fc_control(F, W, B, R, VDMA0_BASE_ADDR, FC_BASE_ADDR)
```

```
[0, 0, 0, 24]
```

[47]:
```
1
```

[48]:
```
SU.check_result("./data/new_cifar10_random_data/fc2_relu_out_2s.txt",␣
 ↪0x06B0_0000)
```

```
All Results are Correct
```

[49]:
```
#SU.save_file("./data/new_cifar10_random_data/fc2_relu_out_2s.txt", 0x06B0_0000)
```

[50]:
```
#########################################################################
#        Fully-Connected 3 + ReLU
#########################################################################
# Fully-Connected
# - in:               (64,)
# - out:              (10,)
# - weight:        (10, 64)
# - bias:             (10,)
# ReLU
# - in:               (10,)
# - out:              (10,)
#########################################################################
F = {'BASE_ADDR': 0x06B0_0000, 'STRIDE_SIZE': 64, 'HSIZE': 64, 'VSIZE': 1}
W = {'BASE_ADDR': 0x0560_0000, 'STRIDE_SIZE': 640, 'HSIZE': 640, 'VSIZE': 1}
B = {'BASE_ADDR': 0x0570_0000, 'STRIDE_SIZE': 10, 'HSIZE': 10, 'VSIZE': 1}
R = {'BASE_ADDR': 0x06C0_0000, 'STRIDE_SIZE': 10, 'HSIZE': 10, 'VSIZE': 1}
```

```
SU.su_fc_control(F, W, B, R, VDMA0_BASE_ADDR, FC_BASE_ADDR)
```

```
[0, 0, 0, 9]
```

[50]:
```
1
```

[51]:
```
SU.check_result("./data/new_cifar10_random_data/fc3_out_2s.txt", 0x06C0_0000,␣
↪mode="fc3_out_2s")
```

```
All Results are Correct
```

[52]:
```
##########################################################################################
# Below code can be revised according to your apb register setting
##########################################################################################
# Read label index from apb register (our design output the index to that␣
↪address)

# We assign FC_BASE_ADDR + 0x20 apb register to return max-value index
label = SU.su_read_data(FC_BASE_ADDR + 0x20)
label = int.from_bytes(label, 'big', signed=True)
# Predicted (computated) label
print(label-1)
if(label-1 == 8):
    print("Max index is Correct")
else:
    print("Max index is Wrong")
```

```
8
Max index is Correct
```

[ ]:
```

```