In [3]:
```
from pynq import Overlay
ol = Overlay("final.bit")
```

In [4]:
```
ol.ip_dict
```

```
   'NUM_READ_THREADS': '1',
   'NUM_WRITE_OUTSTANDING': '1',
   'NUM_WRITE_THREADS': '1',
   'PHASE': '0.0',
   'PROTOCOL': 'AXI4LITE',
   'READ_WRITE_MODE': 'READ_WRITE',
   'RUSER_BITS_PER_BYTE': '0',
   'RUSER_WIDTH': '0',
   'SUPPORTS_NARROW_BURST': '0',
   'WUSER_BITS_PER_BYTE': '0',
   'WUSER_WIDTH': '0',
   'MAX_READ_BURST_LENGTH': '16',
   'MAX_WRITE_BURST_LENGTH': '16'},
  'registers': {'CTRL': {'address_offset': 0,
    'size': 32,
    'access': 'read-write',
    'description': 'Control signals',
    'fields': {'AP_START': {'bit_offset': 0,
      'bit_width': 1,
      'access': 'read-write',
```

In [5]:
```
my_ip = ol.CNN_0
```

In [6]:

```
my_ip.register_map
```

Out[6]:

```
RegisterMap {
  CTRL = Register(AP_START=0, AP_DONE=0, AP_IDLE=1, AP_READY=0, RESERVED_1=0, AUTO_RESTART=0, RESERVED_2=0, 
INTERRUPT=0, RESERVED_3=0),
  GIER = Register(Enable=0, RESERVED=0),
  IP_IER = Register(CHAN0_INT_EN=0, CHAN1_INT_EN=0, RESERVED_0=0),
  IP_ISR = Register(CHAN0_INT_ST=0, CHAN1_INT_ST=0, RESERVED_0=0),
  kernel_1 = Register(kernel=write-only),
  kernel_2 = Register(kernel=write-only),
  kernel_1_1 = Register(kernel_1=write-only),
  kernel_1_2 = Register(kernel_1=write-only),
  kernel_2_1 = Register(kernel_2=write-only),
  kernel_2_2 = Register(kernel_2=write-only),
  kernel_3_1 = Register(kernel_3=write-only),
  kernel_3_2 = Register(kernel_3=write-only),
  kernel_4_1 = Register(kernel_4=write-only),
  kernel_4_2 = Register(kernel_4=write-only),
  kernel_5_1 = Register(kernel_5=write-only),
  kernel_5_2 = Register(kernel_5=write-only),
  bias_1 = Register(bias=write-only),
  bias_2 = Register(bias=write-only),
```

In [7]:

```python
import numpy as np

# Hàm load dữ liệu từ file và chuyển đổi
def load_data(file_path, dtype=np.int16, scale=2**13):
    return (np.loadtxt(file_path) * scale).astype(dtype)

# Nạp input từ file
# input_data = load_data('txt_output/healthy_Not_Cancer__1026_batch_927.txt')

# Nạp trọng số và bias cho các lớp convolution
layer1_weight = load_data('conv2d_4_weights.txt')
layer1_bias = load_data('conv2d_4_bias.txt')

layer2_weight = load_data('conv2d_5_weights.txt')
layer2_bias = load_data('conv2d_5_bias.txt')

layer3_weight = load_data('conv2d_6_weights.txt')
layer3_bias = load_data('conv2d_6_bias.txt')
layer4_weight = load_data('conv2d_7_weights.txt')
layer4_bias = load_data('conv2d_7_bias.txt')
layer5_weight = load_data('dense_2_weights.txt')
layer5_bias = load_data('dense_2_bias.txt')
layer6_weight = load_data('dense_3_weights.txt')
layer6_bias = load_data('dense_3_bias.txt')

# Kiểm tra kích thước dữ liệu đã nạp
# print("Input shape:", input_data.shape)
print("Layer 1 weights shape:", layer1_weight.shape, "Bias shape:", layer1_bias.shape)
print("Layer 2 weights shape:", layer2_weight.shape, "Bias shape:", layer2_bias.shape)
print("Layer 3 weights shape:", layer3_weight.shape, "Bias shape:", layer3_bias.shape)
print("Layer 4 weights shape:", layer4_weight.shape, "Bias shape:", layer4_bias.shape)
print("Layer 5 weights shape:", layer5_weight.shape, "Bias shape:", layer5_bias.shape)
print("Layer 6 weights shape:", layer6_weight.shape, "Bias shape:", layer6_bias.shape)
```

```
Layer 1 weights shape: (72,) Bias shape: (8,)
Layer 2 weights shape: (1152,) Bias shape: (16,)
Layer 3 weights shape: (4608,) Bias shape: (32,)
Layer 4 weights shape: (18432,) Bias shape: (64,)
Layer 5 weights shape: (230400,) Bias shape: (100,)
Layer 6 weights shape: (100,) Bias shape: ()
```

In [8]:

```
import numpy as np
from pynq import Overlay, allocate

# 🟢 Tạo buffer đầu vào
input_buffer = allocate(shape=(128*128,), dtype=np.int16)
# 🟢 Tạo buffer cho các lớp convolution
layer1_weight_buffer = allocate(shape=(8*3*3,), dtype=np.int16)
layer1_bias_buffer = allocate(shape=(8), dtype=np.int16)


layer2_weight_buffer = allocate(shape=(16*8*3*3,), dtype=np.int16)
layer2_bias_buffer = allocate(shape=(16,), dtype=np.int16)

layer3_weight_buffer = allocate(shape=(32*16*3*3,), dtype=np.int16)
layer3_bias_buffer = allocate(shape=(32,), dtype=np.int16)

layer4_weight_buffer = allocate(shape=(64*32*3*3), dtype=np.int16)
layer4_bias_buffer = allocate(shape=(64,), dtype=np.int16)
layer5_weight_buffer = allocate(shape=(230400), dtype=np.int16)
layer5_bias_buffer = allocate(shape=(100,), dtype=np.int16)
layer6_weight_buffer = allocate(shape=(100), dtype=np.int16)
layer6_bias_buffer = allocate(shape=(1,), dtype=np.int16)
output_buffer = allocate(shape=(1,), dtype=np.int16)


# 📌 In ra kích thước của buffer
print(f"Input buffer size: {input_buffer.shape}")
print(f"Output buffer 1 size: {output_buffer.shape}")


print(f"Layer 1 weight buffer size: {layer1_weight_buffer.shape}")
print(f"Layer 1 bias buffer size: {layer1_bias_buffer.shape}")

print(f"Layer 2 weight buffer size: {layer2_weight_buffer.shape}")
print(f"Layer 2 bias buffer size: {layer2_bias_buffer.shape}")

print(f"Layer 3 weight buffer size: {layer3_weight_buffer.shape}")
print(f"Layer 3 bias buffer size: {layer3_bias_buffer.shape}")

print(f"Layer 4 weight buffer size: {layer4_weight_buffer.shape}")
print(f"Layer 4 bias buffer size: {layer4_bias_buffer.shape}")
print(f"Layer 5 weight buffer size: {layer5_weight_buffer.shape}")
print(f"Layer 5 bias buffer size: {layer5_bias_buffer.shape}")
print(f"Layer 6 weight buffer size: {layer6_weight_buffer.shape}")
print(f"Layer 6 bias buffer size: {layer6_bias_buffer.shape}")
```

```
Input buffer size: (16384,)
Output buffer 1 size: (1,)
Layer 1 weight buffer size: (72,)
Layer 1 bias buffer size: (8,)
Layer 2 weight buffer size: (1152,)
Layer 2 bias buffer size: (16,)
Layer 3 weight buffer size: (4608,)
Layer 3 bias buffer size: (32,)
Layer 4 weight buffer size: (18432,)
Layer 4 bias buffer size: (64,)
Layer 5 weight buffer size: (230400,)
Layer 5 bias buffer size: (100,)
Layer 6 weight buffer size: (100,)
Layer 6 bias buffer size: (1,)
```

In [9]:

```
# Sao chép input vào input_buffer
# np.copyto(input_buffer, input_data.flatten())

# Sao chép kernel vào các buffer tương ứng
np.copyto(layer1_weight_buffer, layer1_weight.flatten())  # Lớp 1
np.copyto(layer1_bias_buffer, layer1_bias.flatten())  # Bias lớp 1

np.copyto(layer2_weight_buffer, layer2_weight.flatten())  # Lớp 2
np.copyto(layer2_bias_buffer, layer2_bias.flatten())  # Bias lớp 2

np.copyto(layer3_weight_buffer, layer3_weight.flatten())  # Lớp 3
np.copyto(layer3_bias_buffer, layer3_bias.flatten())  # Bias lớp 3

np.copyto(layer4_weight_buffer, layer4_weight.flatten())  # Lớp 4
np.copyto(layer4_bias_buffer, layer4_bias.flatten())  # Bias lớp 4

np.copyto(layer5_weight_buffer, layer5_weight.flatten())  # Lớp 4
np.copyto(layer5_bias_buffer, layer5_bias.flatten())  # Bias lớp 4

np.copyto(layer6_weight_buffer, layer6_weight.flatten())  # Lớp 4
np.copyto(layer6_bias_buffer, layer6_bias.flatten())  # Bias lớp 4
```

In [10]:

```
# Ghi địa chỉ vật lý của các buffer vào các thanh ghi tương ứng của IP
my_ip.write(0xa0, input_buffer.physical_address)  # Địa chỉ vật lý của input_buffer
my_ip.write(0xac, output_buffer.physical_address)  # Địa chỉ vật lý của output_buffer

# Ghi địa chỉ vật lý của các buffer trọng số và bias cho các lớp CNN
my_ip.write(0x10, layer1_weight_buffer.physical_address)  # layer1_weight_buffer
my_ip.write(0x58, layer1_bias_buffer.physical_address)  # layer1_bias_buffer
my_ip.write(0x1c, layer2_weight_buffer.physical_address)  # layer2_weight_buffer
my_ip.write(0x64, layer2_bias_buffer.physical_address)  # layer2_bias_buffer
my_ip.write(0x28, layer3_weight_buffer.physical_address)  # layer3_weight_buffer
my_ip.write(0x70, layer3_bias_buffer.physical_address)  # layer3_bias_buffer
my_ip.write(0x34, layer4_weight_buffer.physical_address)  # layer4_weight_buffer#my_ip.write(0x64, layer4_bias_b
my_ip.write(0x7c, layer4_bias_buffer.physical_address)  # layer1_bias_buffer
my_ip.write(0x40, layer5_weight_buffer.physical_address)  # layer4_weight_buffer#my_ip.write(0x64, layer4_bias_b
my_ip.write(0x88, layer5_bias_buffer.physical_address)  # layer1_bias_buffer
my_ip.write(0x4c, layer6_weight_buffer.physical_address)  # layer4_weight_buffer#my_ip.write(0x64, layer4_bias_b
my_ip.write(0x94, layer6_bias_buffer.physical_address)  # layer1_bias_buffer
```

In [11]:

```
import numpy as np
out= (output_buffer / 8192.0)
print(out.astype(np.float16))
```

```
[0.]
```

In [ ]: