AOC Design Proposal

Group 2 軟硬兼施

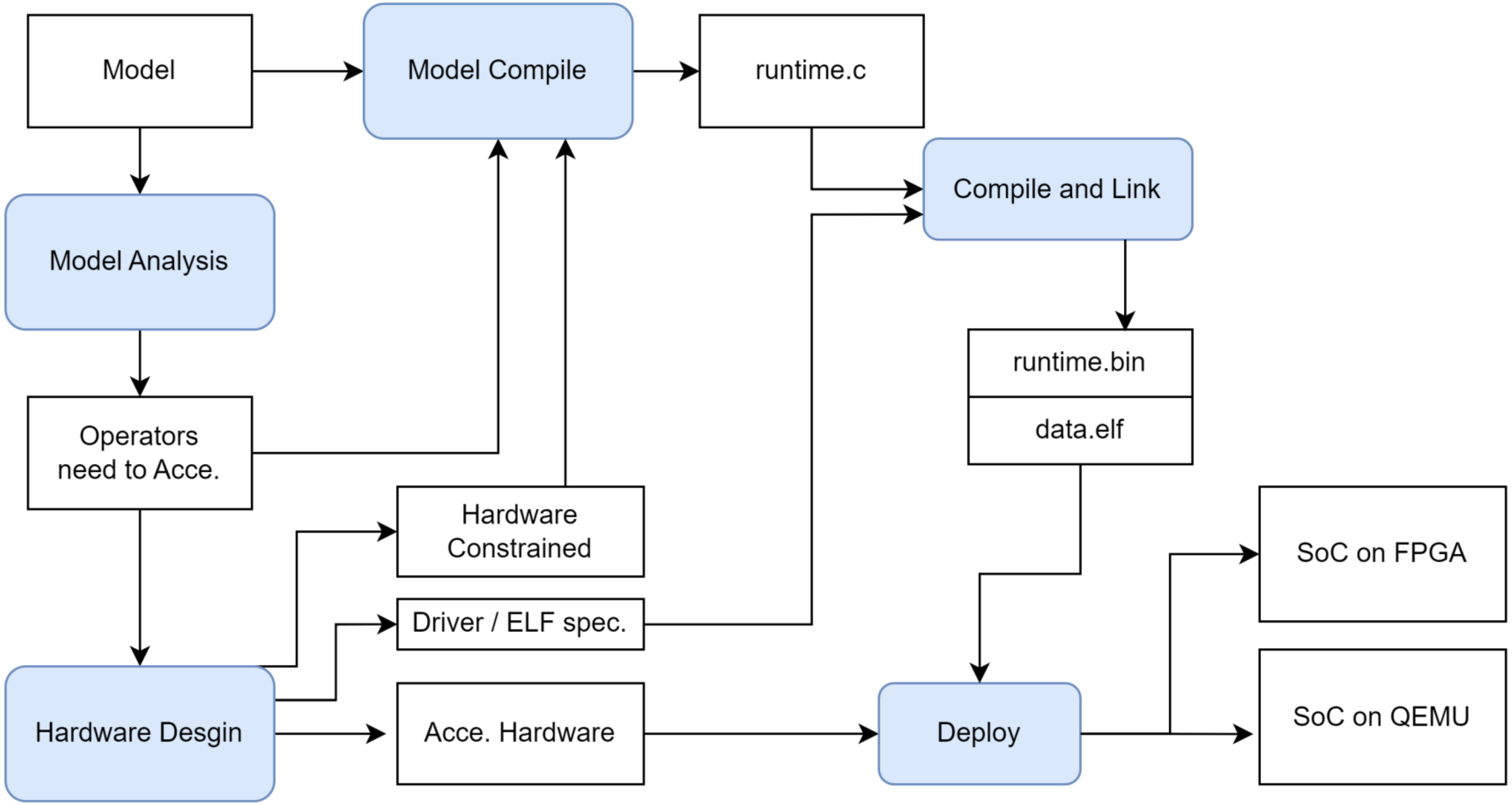
施宇庭 NN6124030 陳均嫚 NP8124014
葉宣佑 E24096695 王力恩 E24091108 施尚甫 E64096148

Targeting Tasks

- Image Classification on Embedded System
  - GoogLeNet
    - Operator - Conv2D, MaxPool
    - Parameters - 6.6M
    - model size - 27MB / 7MB
  - SqueezeNet
    - Operator - Conv2D, MaxPool2D, BatchNorm, Concat, GlobalAvgPool2D, Softmax
    - Parameters - 1.2M
    - model size - 5MB / 2MB

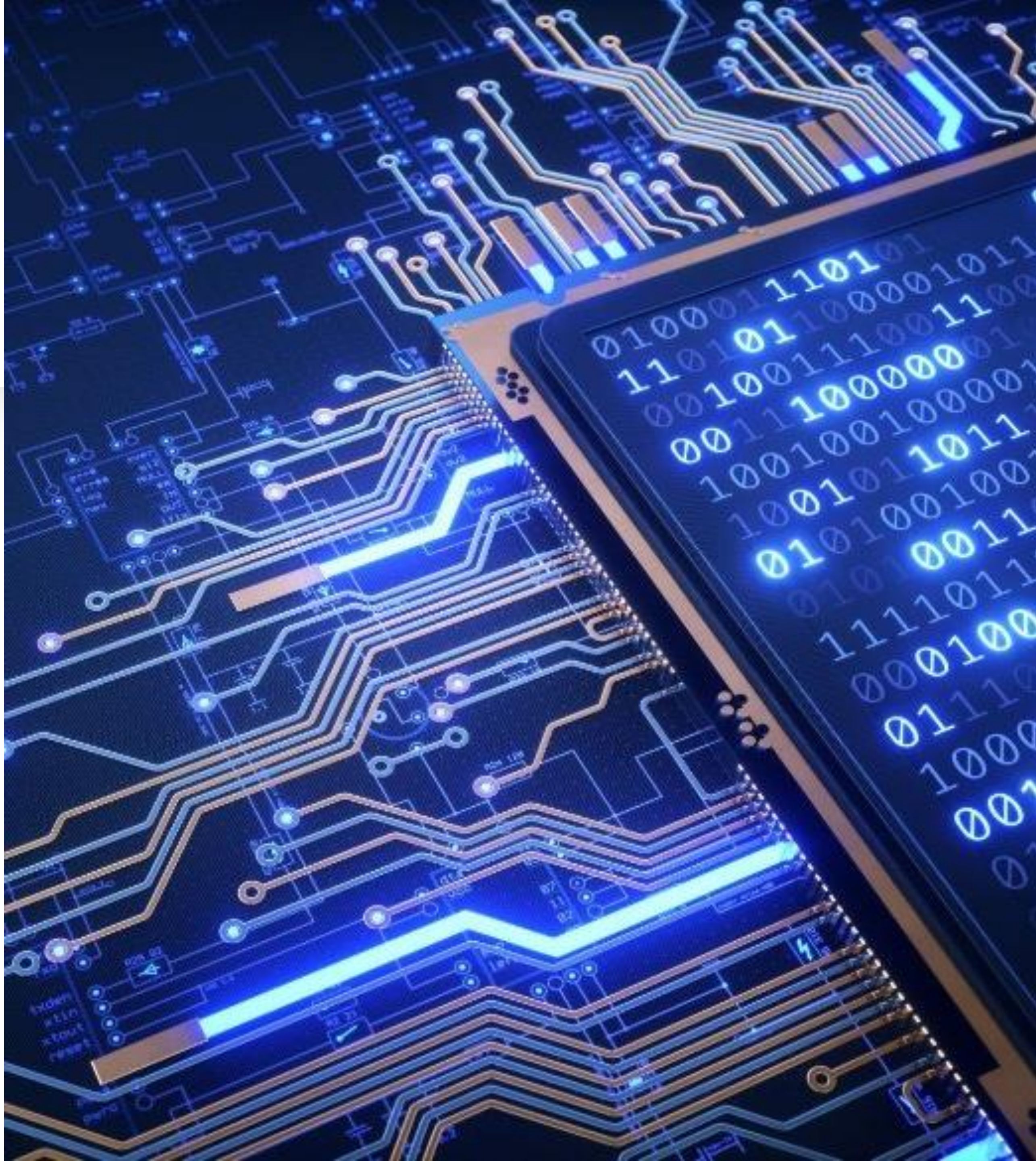
- **Machine Comprehension**
  - BERT-Squad [illegible]
    - BERT-Squad (416 MB)
    - BERT-Squad-int8 (119 MB)
  - ~~BiDAF~~
    - ~~BiDAF (41.5 MB)~~
    - ~~BiDAF-int8 (12 MB)~~
- **Image Classification**
  - MobileNet Yoyo Yeh
    - MobileNet v2-1.0-fp32 (13.3 MB)
    - MobileNet v2-1.0-int8 (3.5 MB)
  - ~~ResNet~~
    - ~~ResNet50-fp32 (97.8 MB)~~
    - ~~ResNet50-int8 (24.6 MB)~~
  - SqueezeNet yuting
    - SqueezeNet 1.0 (5 MB)
    - SqueezeNet 1.0-int8 (2 MB)
  - ~~VGG~~
    - ~~VGG 16-fp32 (527.8 MB)~~
    - ~~VGG 16-int8 (132.0 MB)~~
  - ~~AlexNet~~
    - ~~AlexNet (233 MB)~~
    - ~~AlexNet-int8 (58 MB)~~
  - GoogleNet yuting
    - GoogleNet (27 MB)
    - GoogleNet-int8 (7 MB)
  - ~~CaffeNet~~
    - ~~CaffeNet (233 MB)~~
    - ~~CaffeNet-int8 (58 MB)~~
  - DenseNet-121 yuting
    - DenseNet-121-12 (32 MB)
    - DenseNet-121-12-int8 (9 MB)
  - Inception
    - Inception-1 (27 MB)
    - Inception-1-int8 (10 MB)
  - [x] ShuffleNet Yoyo Yeh
    - ShuffleNet-v2-fp32 (8.79MB)
    - ShuffleNet-v2-int8 (2.28MB)
  - ZFNet-512 [illegible]
    - ZFNet-512 (333 MB)
    - ZFNet-512-int8 (83 MB)
  - EfficientNet-Lite4 [illegible]
    - EfficientNet-Lite4 (51.9 MB)
    - EfficientNet-Lite4-int8 (13.0 MB)
  - YOLOv7 [illegible]

Design Flow


Model
Model Compile
runtime.c
Model Analysis
Compile and Link
runtime.bin
data.elf
Operators need to Acce.
Hardware Constrained
Driver / ELF spec.
SoC on FPGA
Hardware Desgin
Acce. Hardware
Deploy
SoC on QEMU

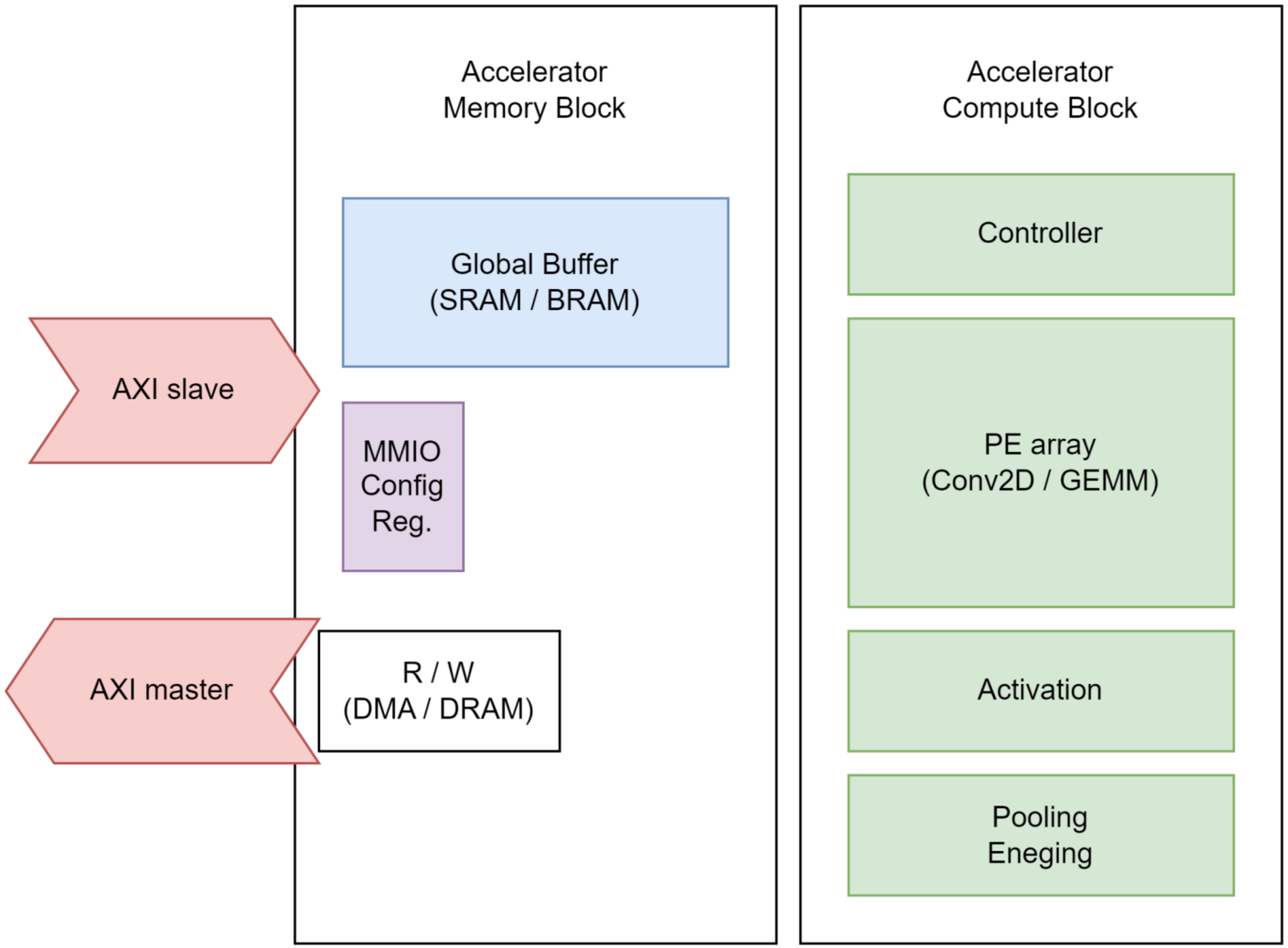
Hardware Design

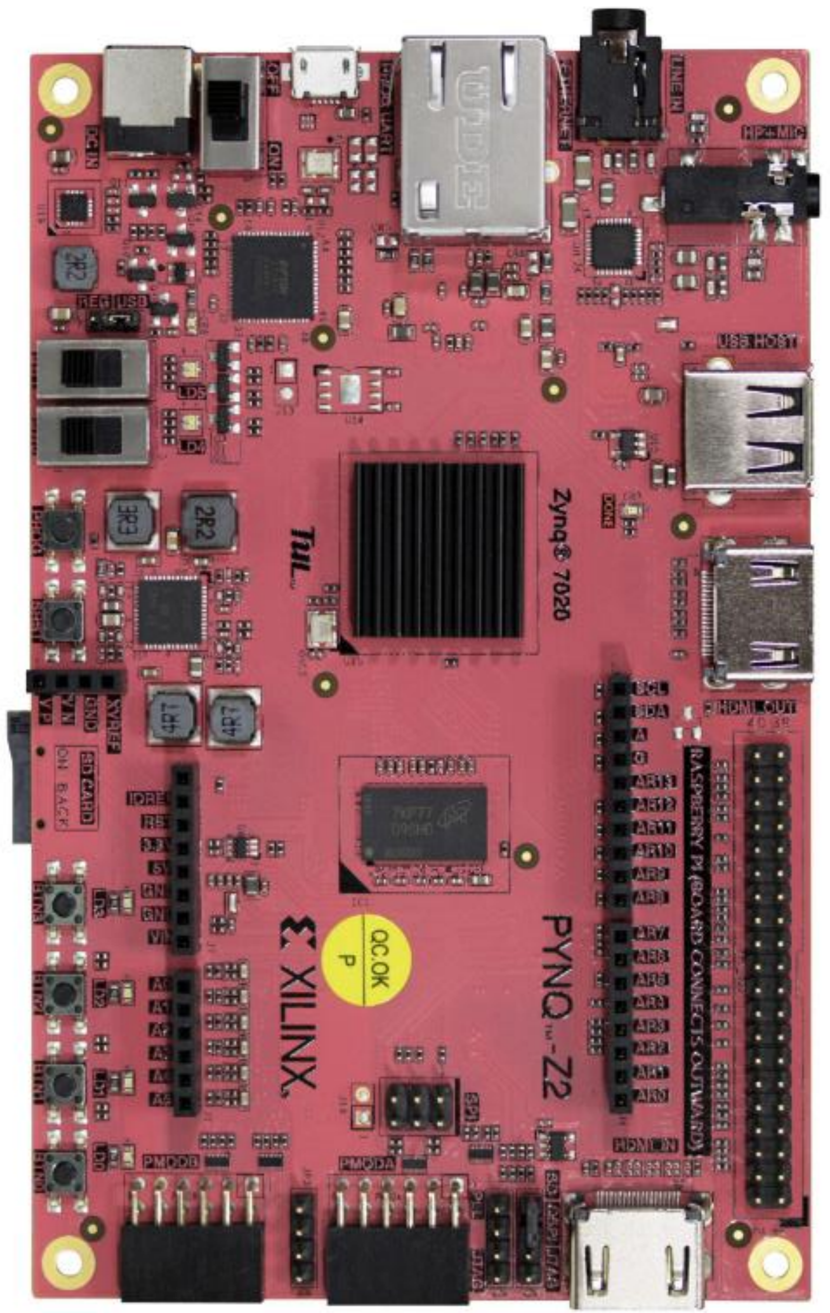
- Architecture
- Verification



Architecture

- Design based on
  - NVDLA
  - Eyeriss
- Support Operation
  - Conv2D
  - GEMM
  - MaxPool2D
  - AvgPool2D
  - ReLU
  - Softmax



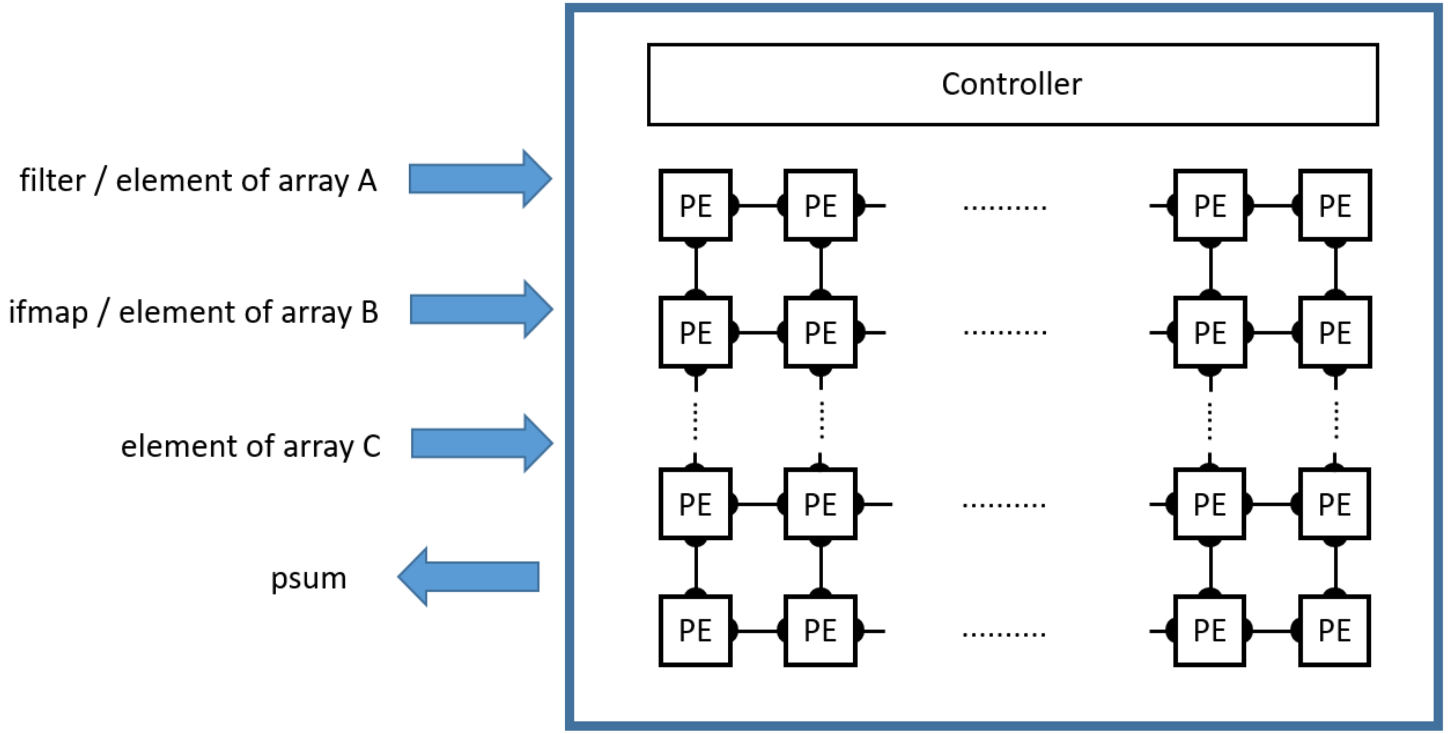


Verification Platform

- Plan A – Porting on FPGA
  - PYNQ-Z2
  - RAM512MB
  - Pros: cycle accurate
  - Cons: synthesis technical problem (AXI / BRAM)
- Plan B – QEMU with C simulation
  - QEMU with custom virtual device
  - Pros: fast deploy
  - Cons: only behavior simulations

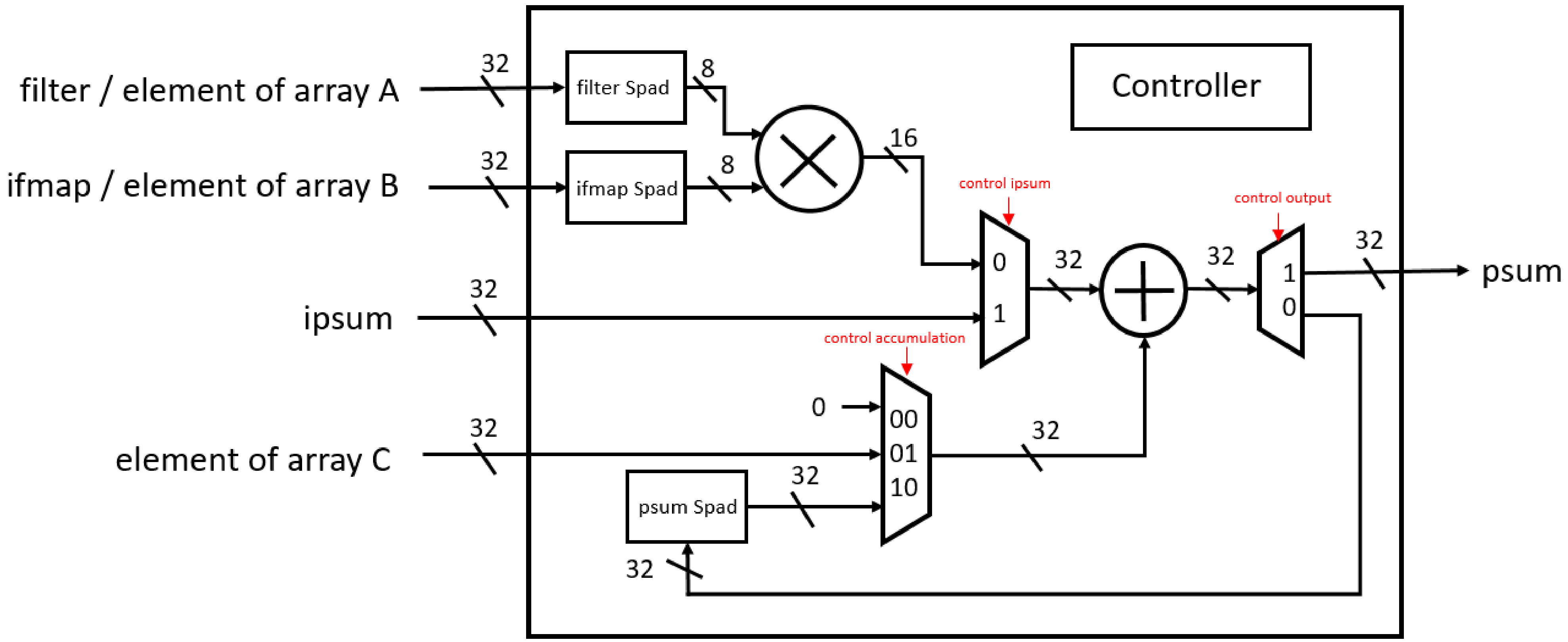
PE array

- Conv2D (1x1, 3x3, 5x5, 7x7), GEMM
- Size of PE array will be determined by the supported Conv2D size



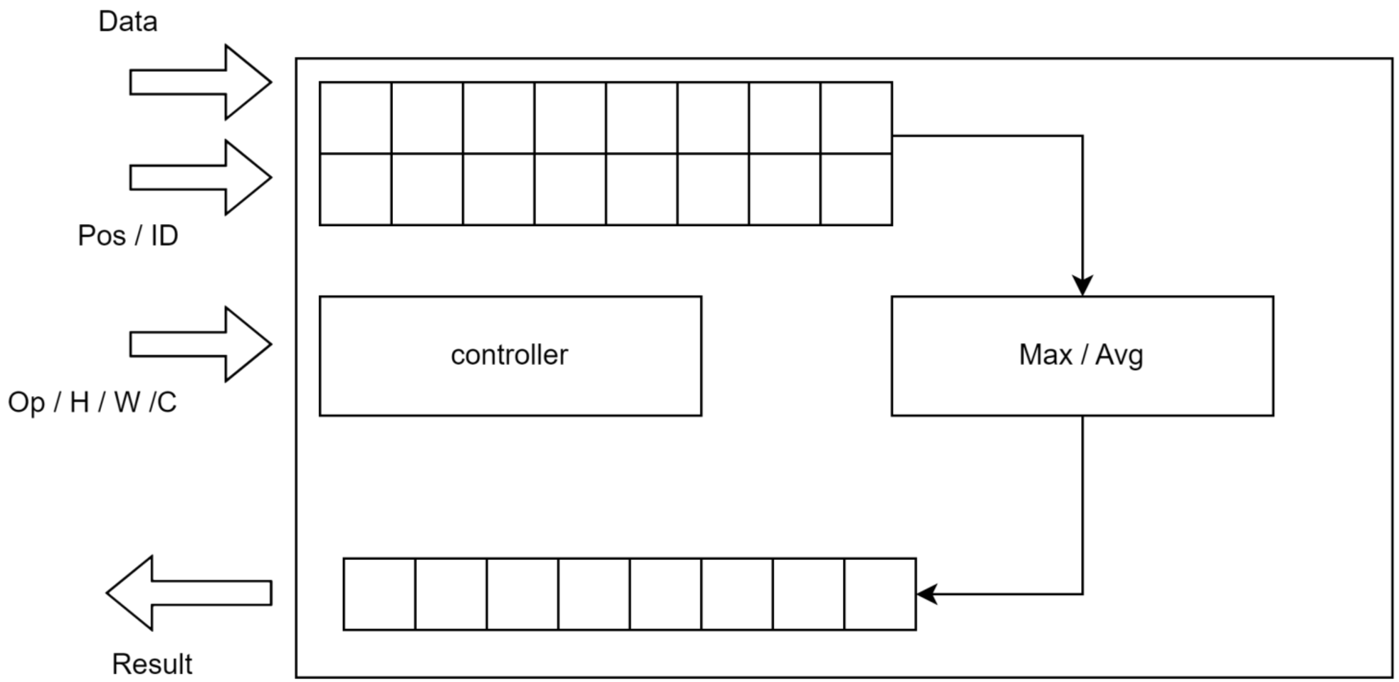
PE Architecture

- Scratchpad size will be determined by the result of analytical model



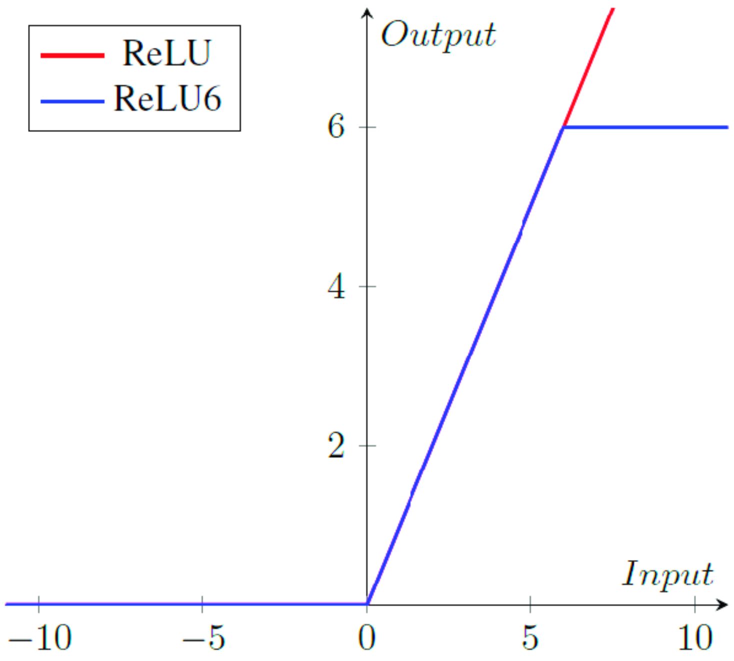
Pooling Engine

- MaxPooling
    - (2x2, stride=(2,2))
    - (3x3, stride=(1,1))
    - (3x3, stride=(2,2))
- AvgPooling
    - GlobalAveragePooling2D



Activation Engine

- ReLU
- ReLU6



- Softmax
  - 3-pass to 2-pass

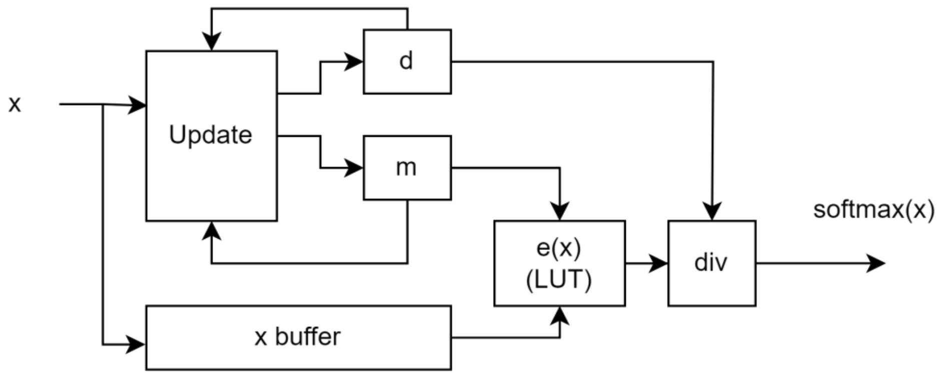
$$m_i \leftarrow \max(m_{i-1}, x_i)$$

$$d_i \leftarrow d_{i-1} + e^{x_i - m_N}$$

$$a_i \leftarrow \frac{e^{x_i - m_N}}{d_N}$$

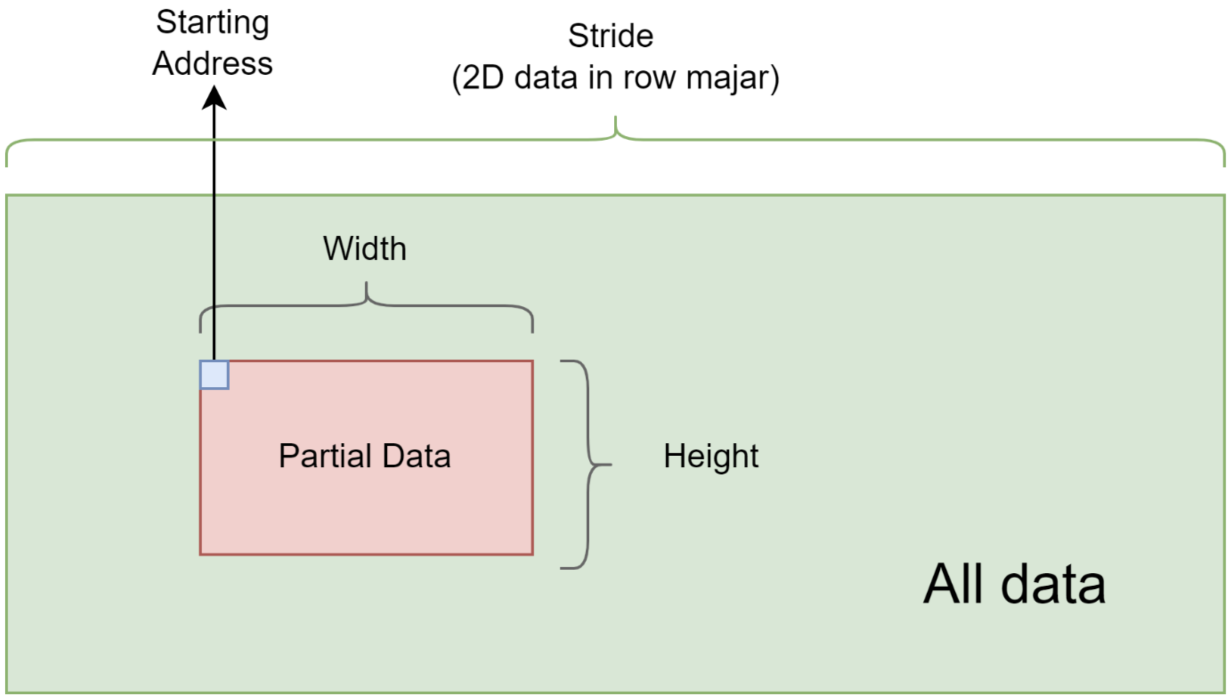
$$m_i \leftarrow \max(m_{i-1}, x_i)$$
$$d'_i \leftarrow d'_{i-1} e^{m_{i-1} - m_i} + e^{x_i - m_i}$$

$$a_i \leftarrow \frac{e^{x_i - m_N}}{d'_N}$$



Loadable – Data Representation

```c
typedef struct Data {
    uint8_t *addr;          // starting address of the first element
    uint32_t height;        // number of rows in a channel
    uint32_t width;         // number of columns in a row
    uint32_t stride;        // distance between two adjacent rows
} Data;
```



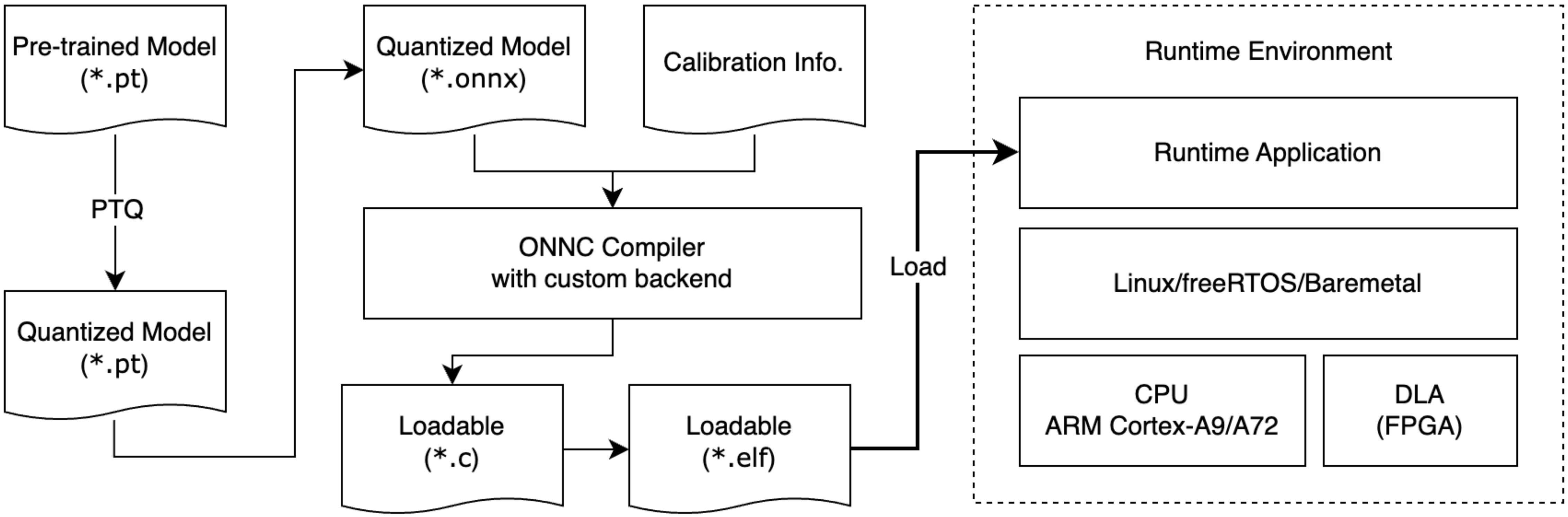
Loadable – Supported Operators

- Conv2D
- GEMM
- MaxPool
- AvgPool
- Softmax
- ReLU
- ReLU6

```c
typedef struct ConvArgs {
    Data *x;            // input
    Data *W;            // weight
    Data *B;            // bias
    Data *y;            // output
    uint32_t padding;
    uint32_t stride;
} ConvArgs;
```

```c
typedef struct Maxpool2DArgs {
    Data *x;                // input
    Data *y;                // output
    uint32_t kern_shape;    // assuming square kernel (only supports 2 or 3)
    uint32_t padding;
    uint32_t stride;
} MaxpoolArgs;
```

Compilation Flow



https://github.com/ONNC/onnc

Quantization

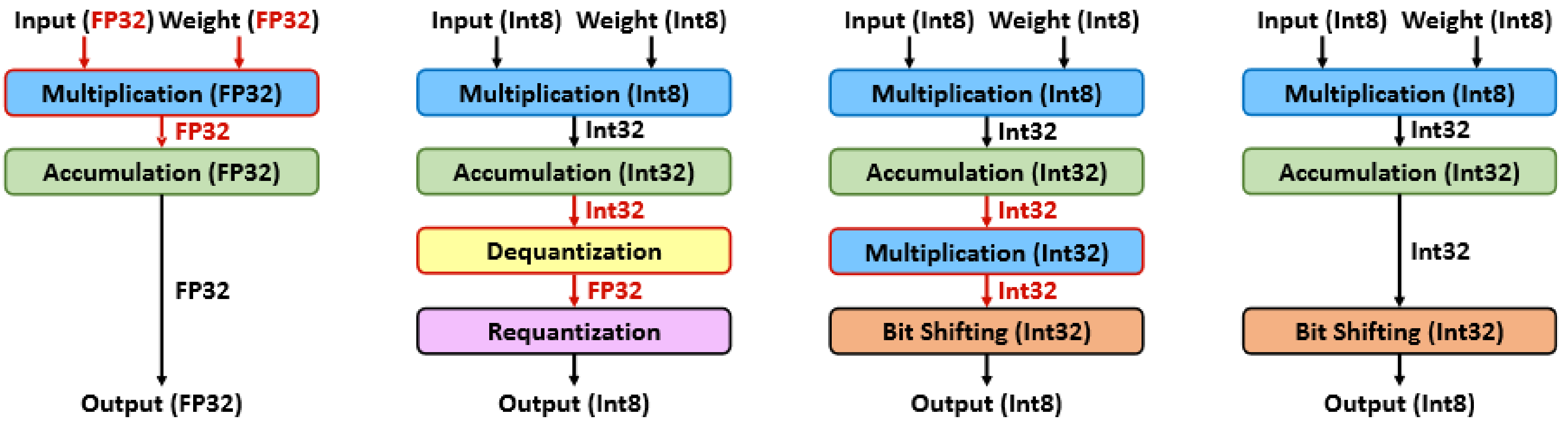
- Quantization method
  - Post training quantization
  - Uniform quantization
  - Symmetric quantization

- Granularity
  - Per-channel quantization

- Integer-only arithmetic

[1] Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference
[2] HAWQV3: Dyadic Neural Network Quantization
[3] Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks

Quantization



(a) Full-precision. (b) Simulated quant. (c) Integer-only quant. (d) Fixed-point quant.

- **Integer-only quant**

  Paper [1][2] : fixed-point multiplication & bit shifting

$$2^{-n}M_0 := \frac{S_1 S_2}{S_3}$$

  - n is a non-negative integer
  - M_0 is a fixed-point multiplier in the interval [0.5, 1)

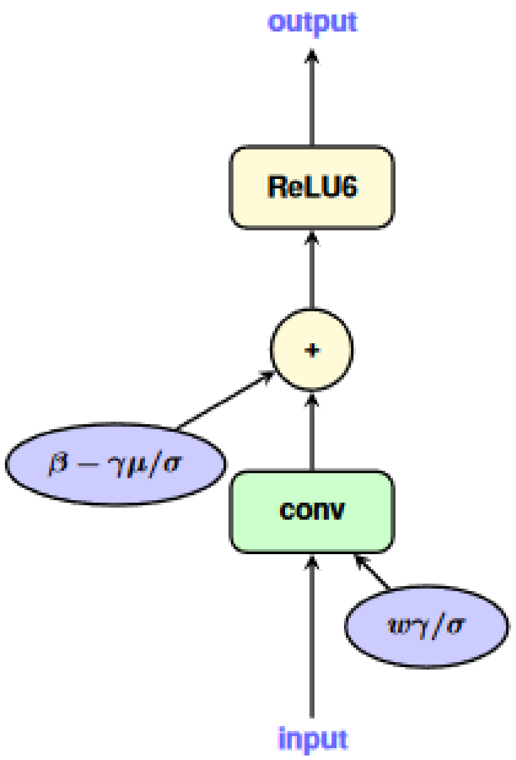
- **Fixed point quant**

  Paper [3] : bit-shift
  by constraining scale-factors s_1, s_2, s_3 to strict power-of-2, the scaling operation reduces to a rather simple bit-shift (with round-to-nearest):

$$2^{-f} = \frac{S_1 S_2}{S_3}$$

Operator Fusion

Fusing Convolution and Batch Normalization



Batch Normalization

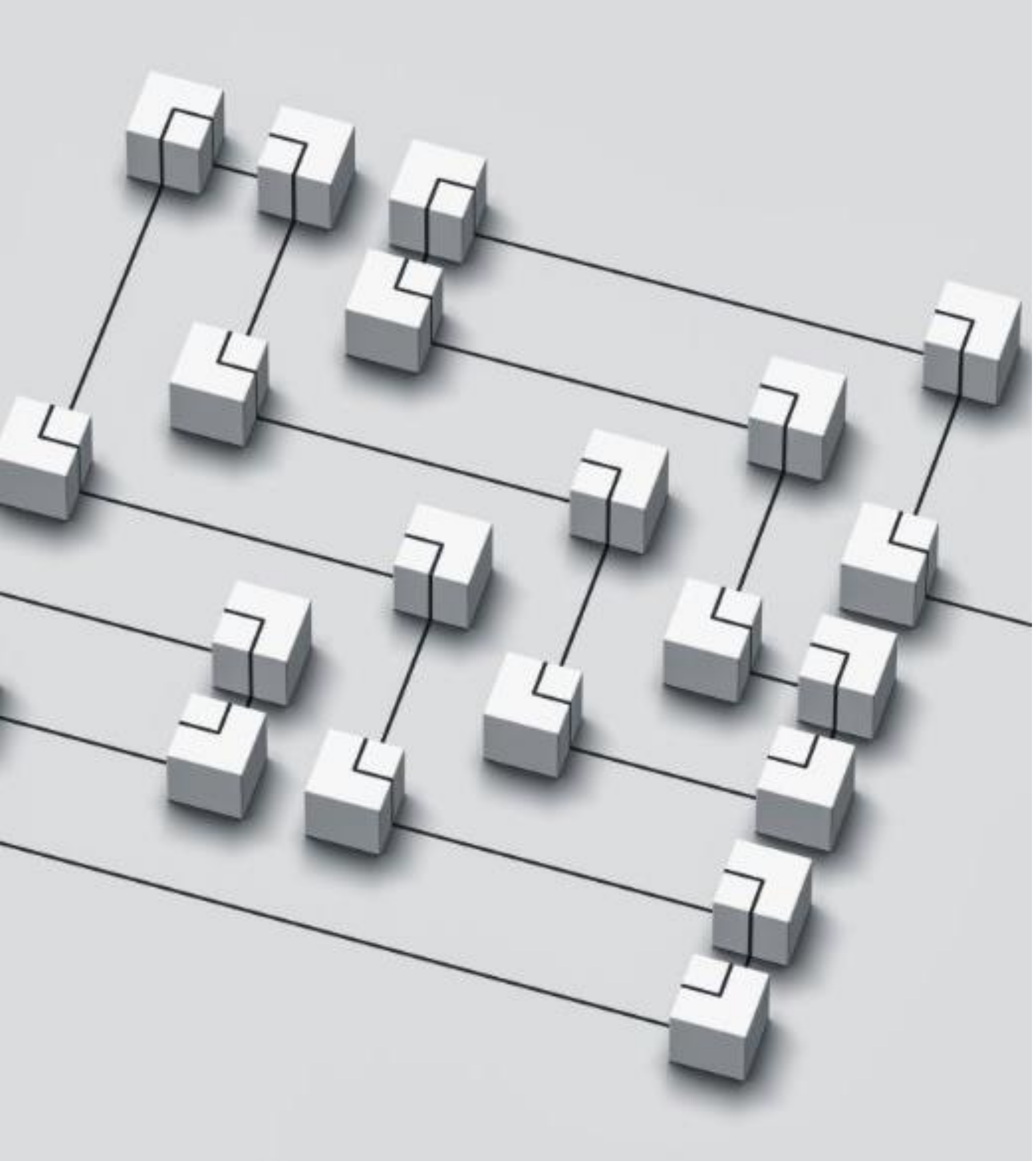
$$y_i = \gamma \left(\frac{x_i - \text{mean}}{\sqrt{\text{variance} + \epsilon}} \right) + \beta$$

Compute effective weights (W_eff) by scaling original weights W with the gamma parameter, normalized by the variance plus a small epsilon for stability:

$$W_{\text{eff}} = W \cdot \left(\frac{\gamma}{\sqrt{\text{variance} + \epsilon}} \right)$$

Compute effective bias (b_eff) by scaling original bias b with gamma, adjusting for the mean-weight product, and adding beta:

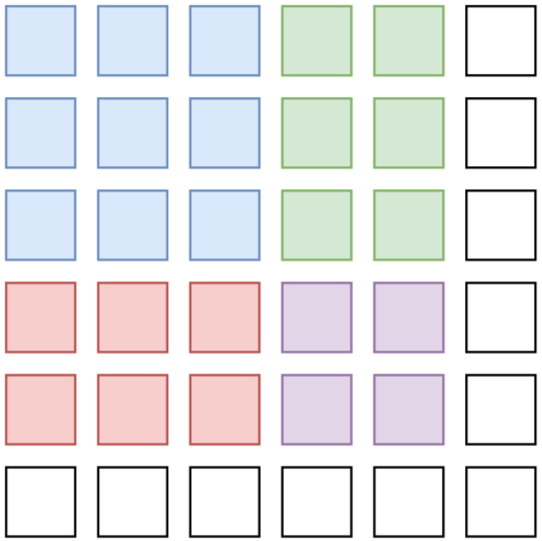
$$b_{\text{eff}} = (b \cdot \gamma) - (\text{mean} \cdot W_{\text{eff}}) + \beta$$

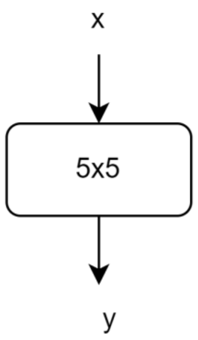


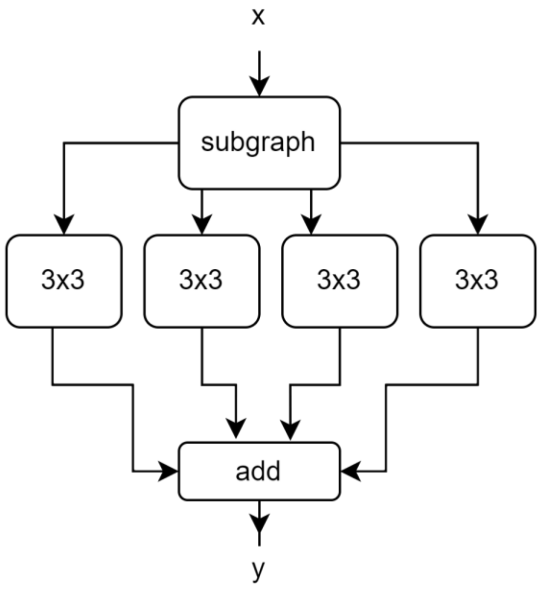
Evaluation

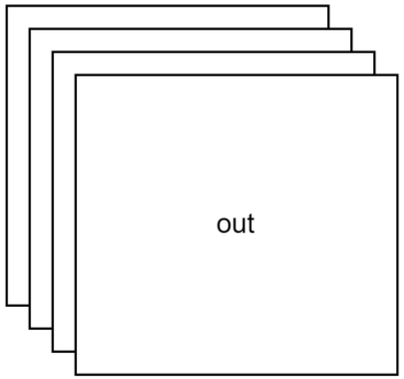
- Inference time (cycle time)
  - Single operator
  - Full model
- Accuracy (ImageNet)
  - Top-1 Accuracy
  - Top-5 Accuracy
- Model size

Question – Software tiling









Question – Memory Requirement

```
# N: number of ifmaps/ofmaps
# M: number of filters
# H/W: ifmap height/width
# R/S: filter height/width
# E/F: ofmap height/width
# U: stride

# m: ofmap channels in global buffer
# n: number of ifmaps in a pass
# e: ofmap width (PE array width)
# p: number of filters in a pass
# q: (ifmap or filter) channels in a pass
# r: number of PE sets for different (ifmap/filter) channels
# t: number of PE sets for different filters
```

```
param = dict(
    m = 64,
    n = 1,
    e = 14, # 224
    p = 16, # psum Spad (filter in PE)
    q = 4, # ifmap/filter channels/pass
    r = 1, # different
    t = 4, # (different filter in PE sets)
    W = 56,
    H = 56,
    R = 3,
    S = 3,
    E = 56,
    F = 56,
    C = 64,
    M = 192,
)
m = Mapping(**param)

print(conv2)
print(conv3)
print(m)
```

```
[Memory Requirement]
[Name]  Alexnet.conv3
================================
[glb_ifmap]              7.0 KiB
[glb_filter]             4.5 KiB
[glb_ofmap]             84.5 KiB
[glb_total]             96.0 KiB
```

```
[Memory Requirement]
[Name]  Alexnet.conv2
================================
[glb_ifmap]              3.8 KiB
[glb_filter]             3.1 KiB
[glb_ofmap]             91.1 KiB
[glb_total]             98.0 KiB
```

```
[Memory Requirement]
[Name]  mapping_2
================================
[glb_ifmap]             24.5 KiB
[glb_filter]             1.1 KiB
[glb_ofmap]             98.0 KiB
[glb_total]            123.6 KiB
```