HLS Final Report
Object Detection Inference On PYNQ-Z2

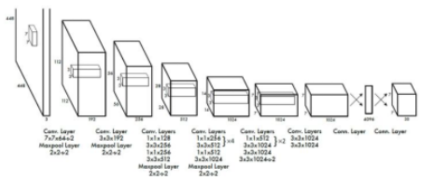
電子碩一 范詠為
電子碩一 林子軒
資工碩一 吳泓毅

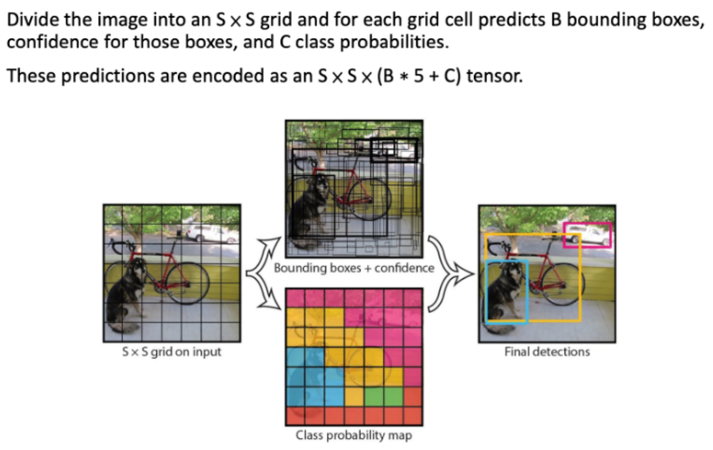
https://github.com/Anderson-Wu/Object-Detection-Inference-On-PYNQ-Z2

1. Introduction

Object Detection is widely applied in many areas of computer vision, including security, automatic vehicle systems, traffic monitoring, inventory management, and so on.

2. YOLOv1 Introduction

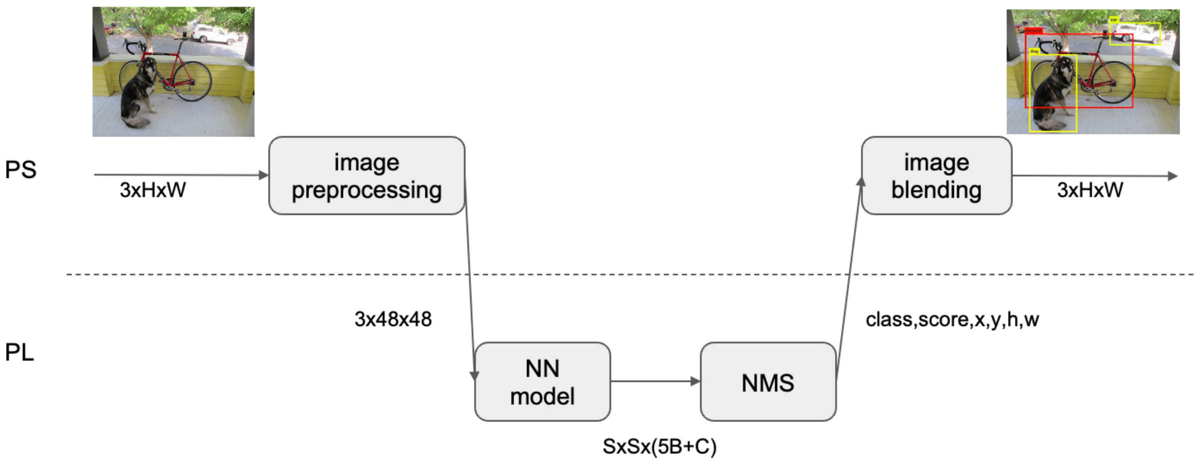




Divide the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.

These predictions are encoded as an S × S × (B ∗ 5 + C) tensor.

3. System Flow



4. Dataset

- PASCAL Visual Object Classes (VOC):
  - year 2007
  - 20 classes.
  - 2501 images
- Using training set to train and validation:
  - YOLOv1 needs pretrained weights to prevent overfitting, but due to time restriction, we didn't pretrained on other dataset like COCO or ImageNet on our model to get pretrained weights and do transfer learning on VOC dataset.

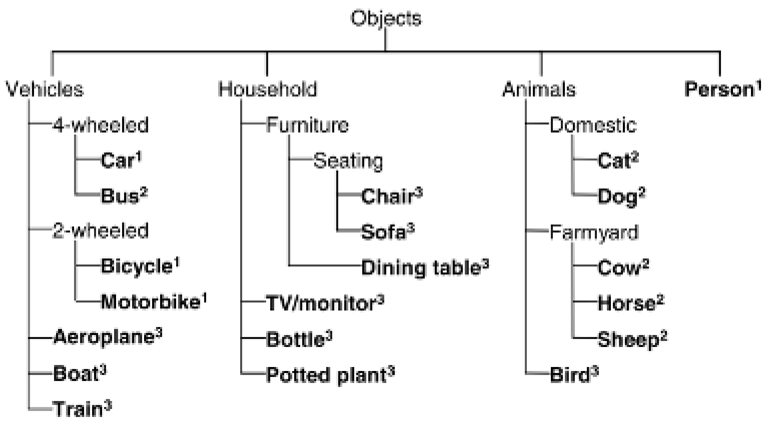
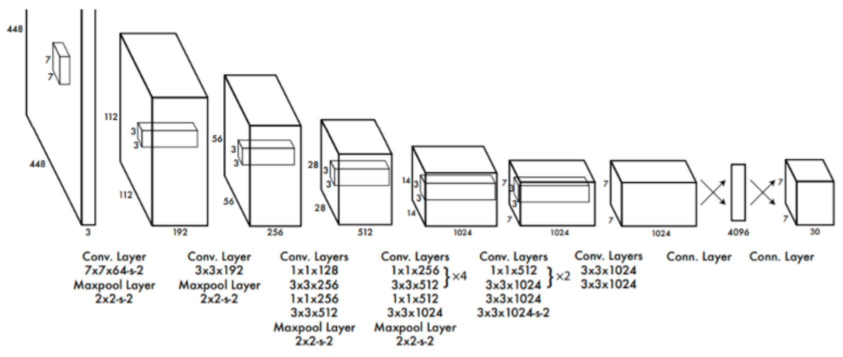
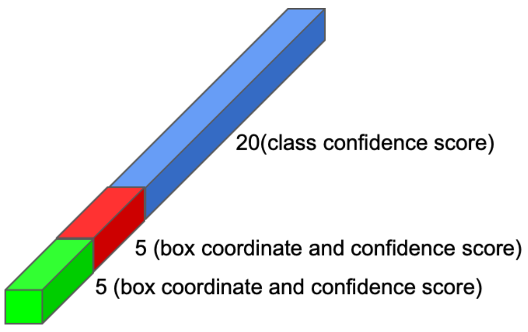


Fig. 2 VOC2007 Classes. Leaf nodes correspond to the 20 classes. The year of inclusion of each class in the challenge is indicated by superscripts: 2005^1, 2006^2, 2007^3. The classes can be considered in a notional taxonomy, with successive challenges adding new branches (increasing the domain) and leaves (increasing detail)

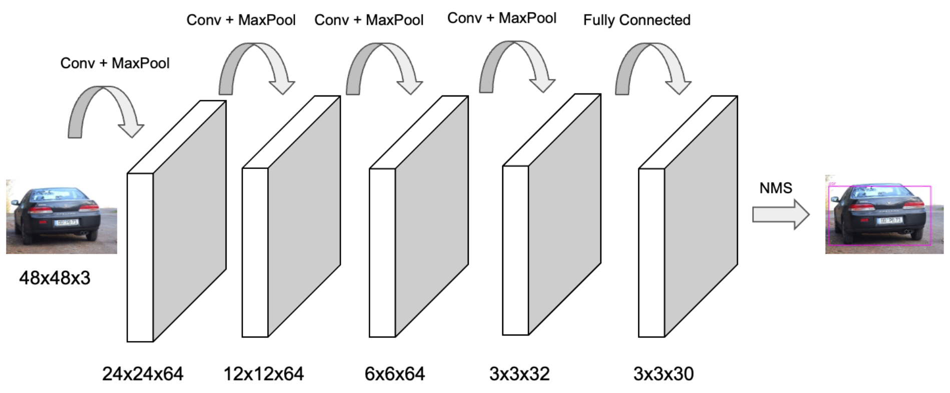
5. Implementation & Encountered Problem
    - model architecture & experiment
        - YOLOv1's Model Architecture
            - Image size is 448x448
            - 24 Conv Layer + 2 fully connected Layer
            - The last layer need 4096x(7x7x30) = 6021120 parameters



        - Model Compression
            - Change image size from 448x448 to 48x48 for lower inference time
            - Change 24 Conv Layer + 2 fully connected Layer to 4 Conv Layer and 1 fully connected Layer
            - Shrink dimension of last layer from 7x7x30 to 3x3x30, now it needs (3x3x32)x(3x3x30) = 77760 parameters, which is much smaller than 6021120 parameters
            - If we remove one box, number of parameters change from 7x7x30 to 7x7x(30-5)=1225, If we shrink grid size to 3x3, number of parameters change from 7x7x30 to 3x3x30=270


20(class confidence score)
5 (box coordinate and confidence score)
5 (box coordinate and confidence score)

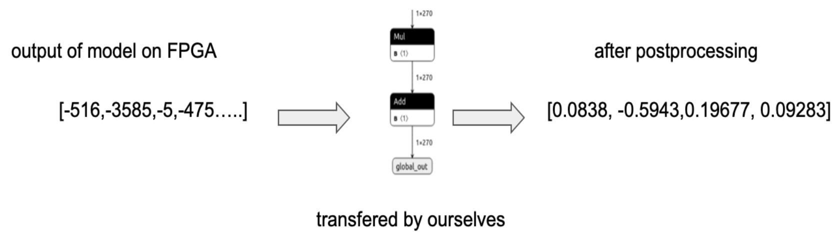
- Our Model Architecture



- Experiment
    - It can be observed that the same model has different accuracy on PC side and PYNQ-Z2 side, the reason we thought is that on PC side, we inferenced with floating point type, and PYNQ-Z2 side with 8 bit int. Since object detection does not only classification but also localization, the error may come from the precision of data type.

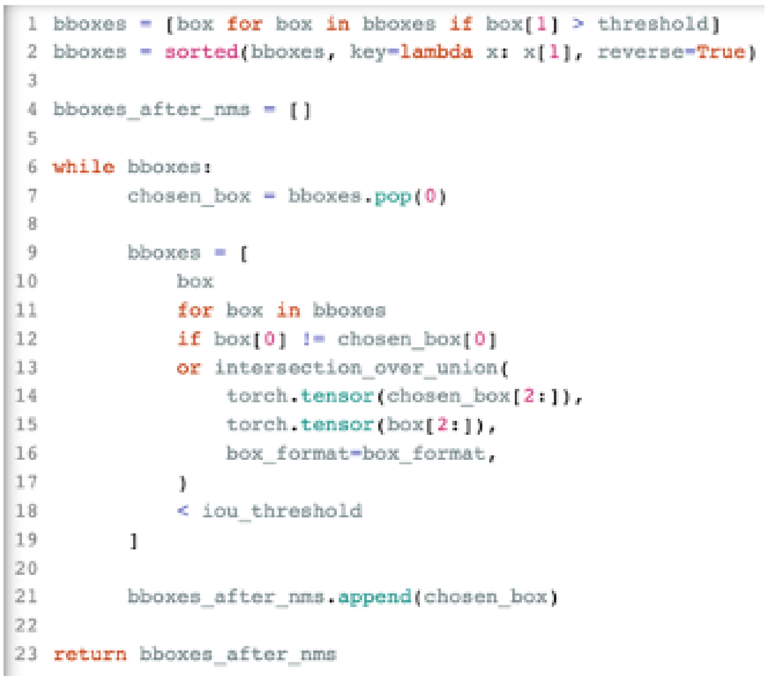
Model	mAP(PC side)	mAP(PYNQ-Z2 side)
3W3A	50.9%	37.7%
4W4A	70.2%	52.1%
5W5A(lower PE and SIMD)	91.9%	68.2%

- Other observation:
    - Since we use Binarized Neural Network, The output should be integer type, but the ideal outputs of YOLOv1 are small floating numbers, so we need to go through the intermediate transferred model to get the operation that is eliminated by FINN and do postprocessing by ourselves to get real output.



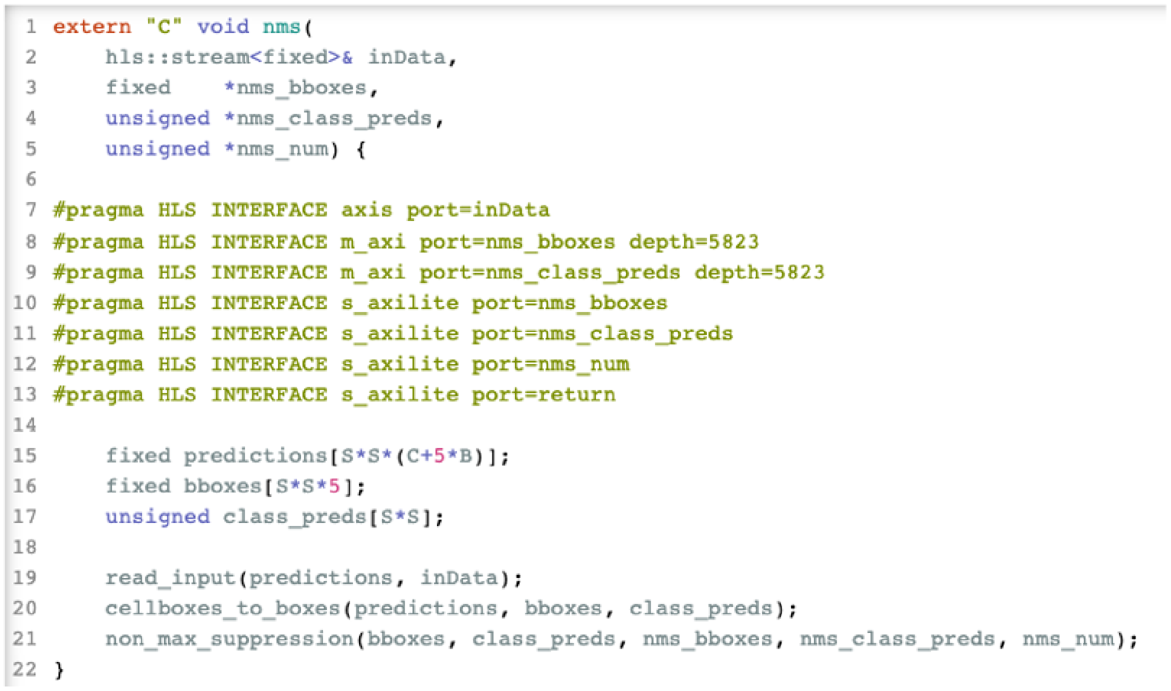
- non maximum suppression
    - select best bounding boxes out of many overlapping bounding boxes
    - algorithm
        - remove bbox with score < score_th
        - sort by score from highest to lowest
        - while boxes list is not empty:
            - pop one bbox (B1), for other bboxes (B2), keep B2 in list if:
                a. B1 and B2's predicted class are different
                b. B1 and B2's predicted class are same and iou(B1, B2) < iou_th
            - append B1 to final return list

- python code



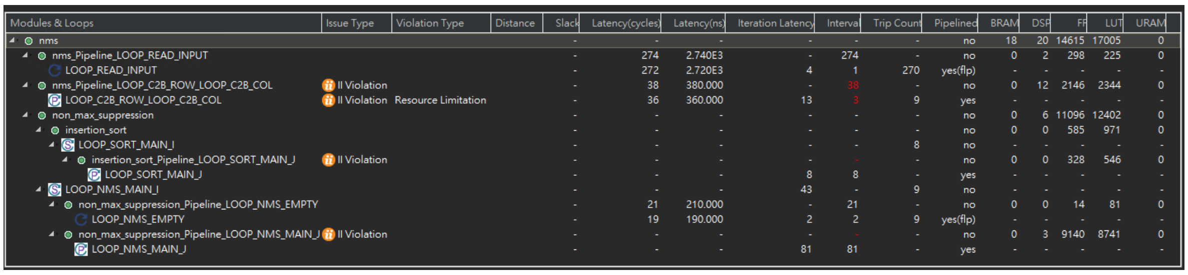
```python
bboxes = [box for box in bboxes if box[1] > threshold]
bboxes = sorted(bboxes, key=lambda x: x[1], reverse=True)

bboxes_after_nms = []

while bboxes:
      chosen_box = bboxes.pop(0)

      bboxes = [
          box
          for box in bboxes
          if box[0] != chosen_box[0]
          or intersection_over_union(
              torch.tensor(chosen_box[2:]),
              torch.tensor(box[2:]),
              box_format=box_format,
          )
          < iou_threshold
      ]

      bboxes_after_nms.append(chosen_box)

return bboxes_after_nms
```

- c++ top interface



```cpp
extern "C" void nms(
    hls::stream<fixed>& inData,
    fixed    *nms_bboxes,
    unsigned *nms_class_preds,
    unsigned *nms_num) {

#pragma HLS INTERFACE axis port=inData
#pragma HLS INTERFACE m_axi port=nms_bboxes depth=5823
#pragma HLS INTERFACE m_axi port=nms_class_preds depth=5823
#pragma HLS INTERFACE s_axilite port=nms_bboxes
#pragma HLS INTERFACE s_axilite port=nms_class_preds
#pragma HLS INTERFACE s_axilite port=nms_num
#pragma HLS INTERFACE s_axilite port=return

    fixed predictions[S*S*(C+5*B)];
    fixed bboxes[S*S*5];
    unsigned class_preds[S*S];

    read_input(predictions, inData);
    cellboxes_to_boxes(predictions, bboxes, class_preds);
    non_max_suppression(bboxes, class_preds, nms_bboxes, nms_class_preds, nms_num);
}
```

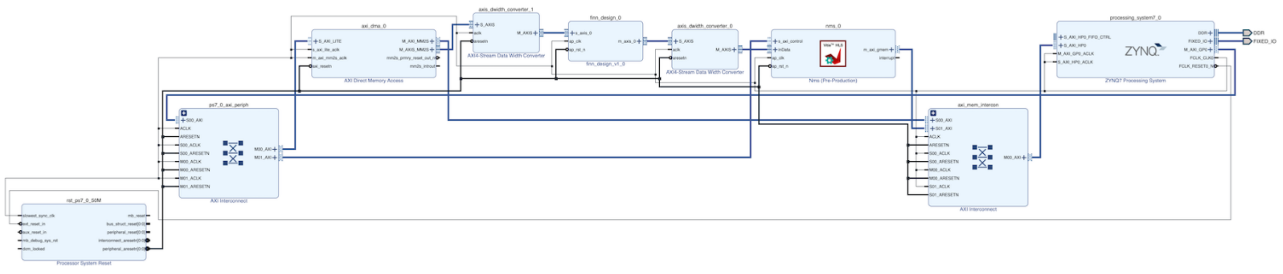
- synthesis result



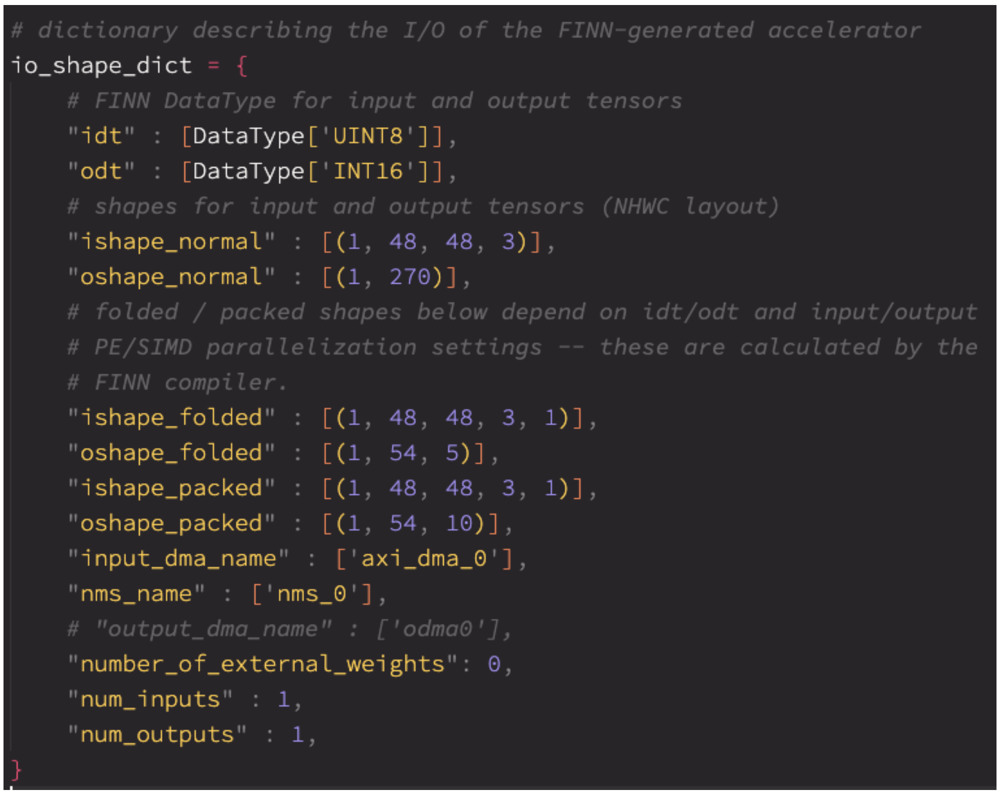
Modules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
nms				-	-	-	-	-	-	no	18	20	14615	17005	0
nms_Pipeline_LOOP_READ_INPUT				-	274	2.740E3	-	274	-	no	0	2	298	225	0
LOOP_READ_INPUT				-	272	2.720E3	4	1	270	yes(flp)	-	-	-	-	-
nms_Pipeline_LOOP_C2B_ROW_LOOP_C2B_COL	II Violation			-	38	380.000	-	38	-	no	0	12	2146	2344	0
LOOP_C2B_ROW_LOOP_C2B_COL	II Violation	Resource Limitation		-	36	360.000	13	3	9	yes	-	-	-	-	-
non_max_suppression				-	-	-	-	-	-	no	0	6	11096	12402	0
insertion_sort				-	-	-	-	-	-	no	0	0	585	971	0
LOOP_SORT_MAIN_I				-	-	-	-	-	8	no	-	-	-	-	-
insertion_sort_Pipeline_LOOP_SORT_MAIN_J	II Violation			-	-	-	-	-	-	no	0	0	328	546	0
LOOP_SORT_MAIN_J				-	-	-	8	8	-	yes	-	-	-	-	-
LOOP_NMS_MAIN_I				-	-	-	43	-	9	no	-	-	-	-	-
non_max_suppression_Pipeline_LOOP_NMS_EMPTY				-	21	210.000	-	21	-	no	0	0	14	81	0
LOOP_NMS_EMPTY				-	19	190.000	2	2	9	yes(flp)	-	-	-	-	-
non_max_suppression_Pipeline_LOOP_NMS_MAIN_J	II Violation			-	-	-	-	-	-	no	0	3	9140	8741	0
LOOP_NMS_MAIN_J				-	-	-	81	81	-	yes	-	-	-	-	-

- encountered problem
    - cannot use dataflow due to algorithm characteristic
        - use stream input then buffer values to perform nms algorithm
    - cannot synthesize to 10ns if using “float” datatype
        - due to long latency of floating point divider (iou = intersection / union)
        - change datatype to ap_fixed<32, 4>

- vivado ip flow (stitich model IP & nms IP)

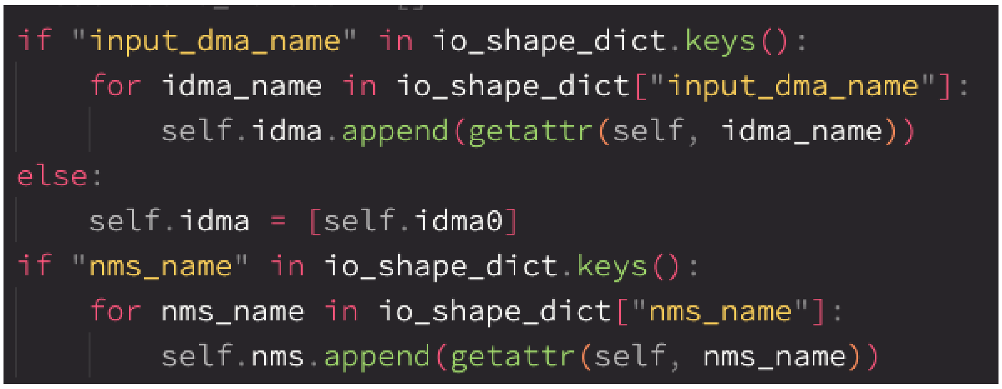


- host program
    - driver.py

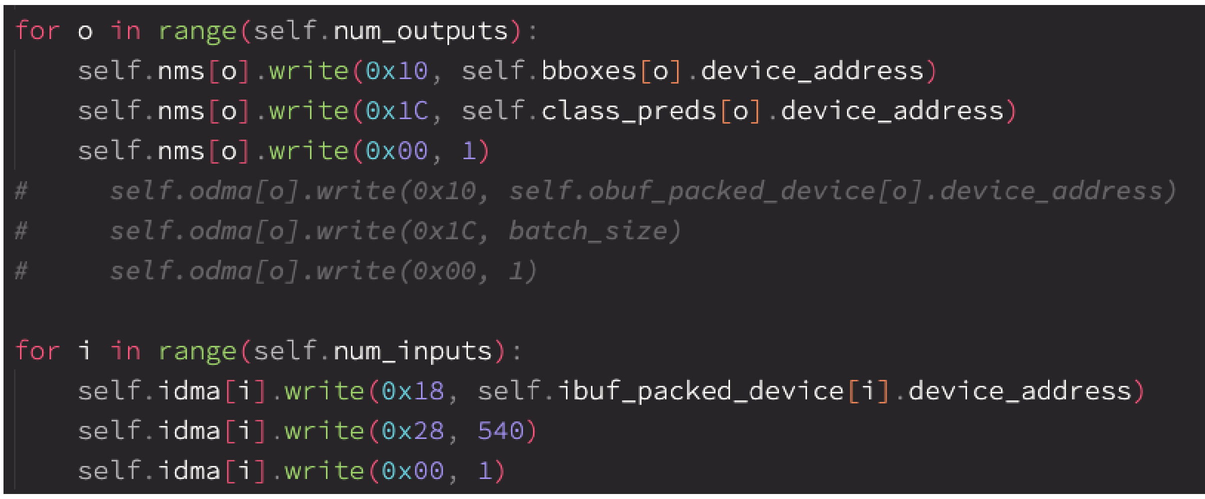


```python
# dictionary describing the I/O of the FINN-generated accelerator
io_shape_dict = {
    # FINN DataType for input and output tensors
    "idt" : [DataType['UINT8']],
    "odt" : [DataType['INT16']],
    # shapes for input and output tensors (NHWC layout)
    "ishape_normal" : [(1, 48, 48, 3)],
    "oshape_normal" : [(1, 270)],
    # folded / packed shapes below depend on idt/odt and input/output
    # PE/SIMD parallelization settings -- these are calculated by the
    # FINN compiler.
    "ishape_folded" : [(1, 48, 48, 3, 1)],
    "oshape_folded" : [(1, 54, 5)],
    "ishape_packed" : [(1, 48, 48, 3, 1)],
    "oshape_packed" : [(1, 54, 10)],
    "input_dma_name" : ['axi_dma_0'],
    "nms_name" : ['nms_0'],
    # "output_dma_name" : ['odma0'],
    "number_of_external_weights": 0,
    "num_inputs" : 1,
    "num_outputs" : 1,
}
```

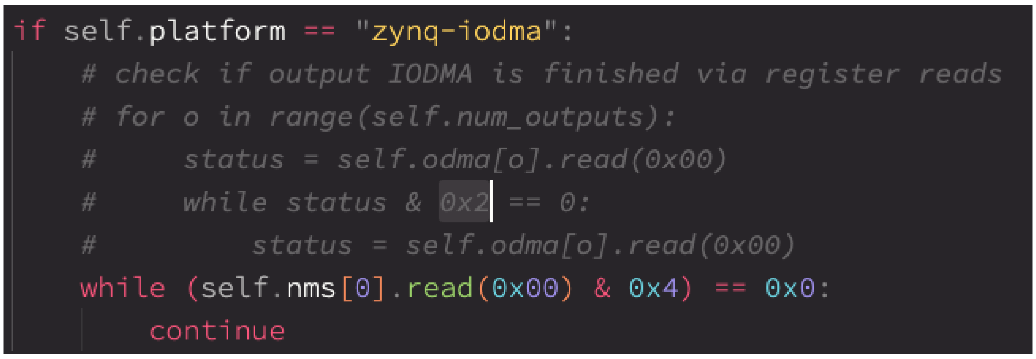
    - driver_base.py



```python
if "input_dma_name" in io_shape_dict.keys():
    for idma_name in io_shape_dict["input_dma_name"]:
        self.idma.append(getattr(self, idma_name))
else:
    self.idma = [self.idma0]
if "nms_name" in io_shape_dict.keys():
    for nms_name in io_shape_dict["nms_name"]:
        self.nms.append(getattr(self, nms_name))
```



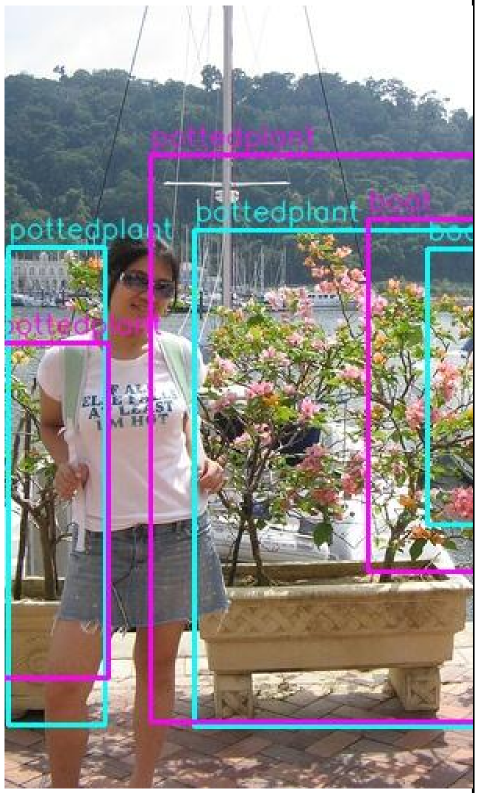
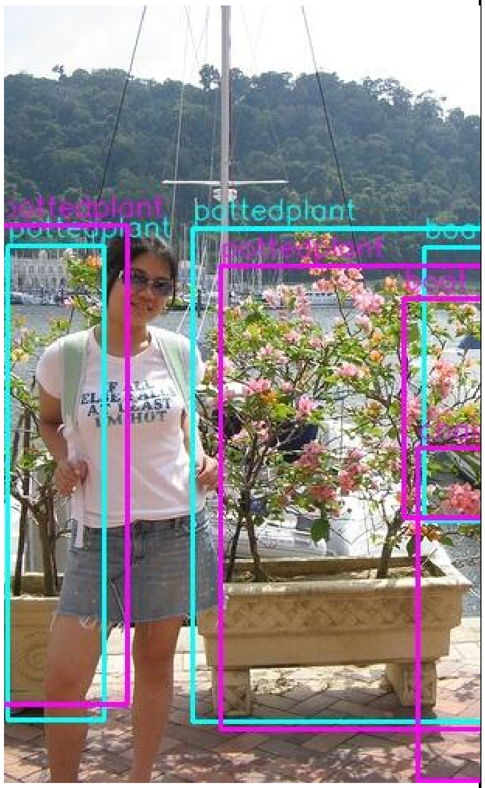
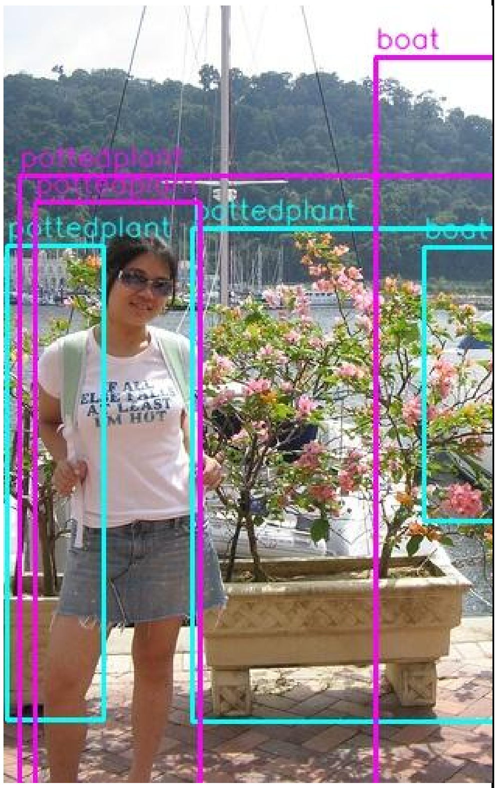
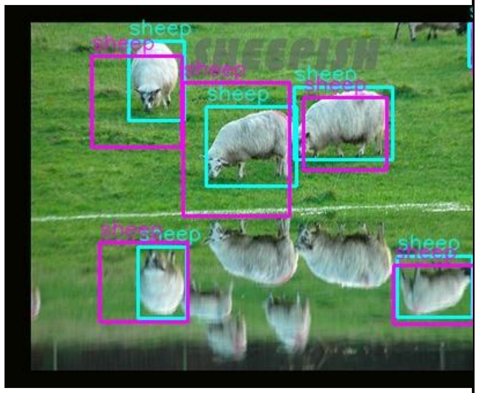
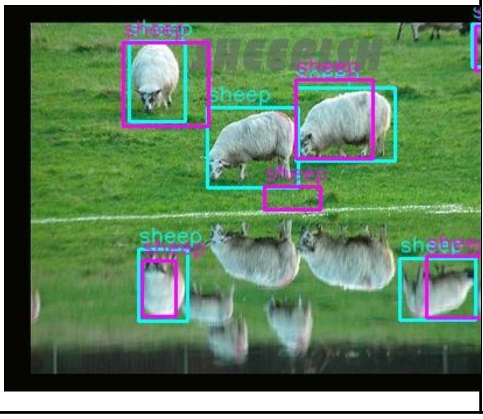
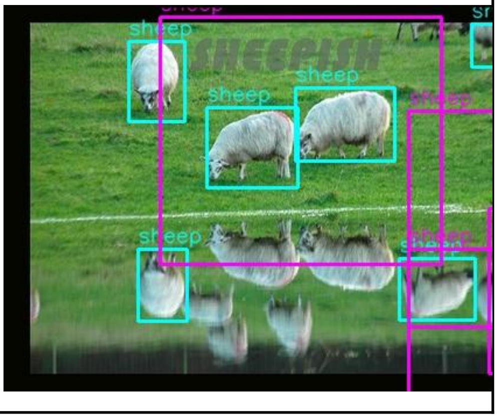
```python
for o in range(self.num_outputs):
    self.nms[o].write(0x10, self.bboxes[o].device_address)
    self.nms[o].write(0x1C, self.class_preds[o].device_address)
    self.nms[o].write(0x00, 1)
#     self.odma[o].write(0x10, self.obuf_packed_device[o].device_address)
#     self.odma[o].write(0x1C, batch_size)
#     self.odma[o].write(0x00, 1)

for i in range(self.num_inputs):
    self.idma[i].write(0x18, self.ibuf_packed_device[i].device_address)
    self.idma[i].write(0x28, 540)
    self.idma[i].write(0x00, 1)
```



```python
if self.platform == "zynq-iodma":
    # check if output IODMA is finished via register reads
    # for o in range(self.num_outputs):
    #     status = self.odma[o].read(0x00)
    #     while status & 0x2 == 0:
    #         status = self.odma[o].read(0x00)
    while (self.nms[0].read(0x00) & 0x4) == 0x0:
        continue
```

- encountered problem
    - the host program cannot finish running, maybe there are some problem in our host code

6. Blender Result

3w3a	4w4a	5w5a
 pottedplant pottedplant boat pottedplant boat pottedplant	 pottedplant pottedplant pottedplant pottedplant boat boat	 boat pottedplant pottedplant pottedplant pottedplant boat
 sheep sheep sheep sheep sheep sheep sheep sheep sheep		

7. Conclusion

- YOLOv1 needs pretrained weights to prevent overfitting, since we modify the model structure, we cannot directly load original pretrained weights, and we don't have time to pretrain on large dataset like ImageNet, either.
- We don't familiar with how to stitch FINN model and other IPs together, so it is still a major problem to be solved.

8. Reference

- You Only Look Once: Unified, Real-Time Object Detection
- LPYOLO: Low Precision YOLO for Face Detection on FPGA
- Lightweight CNN Image Classifier with On-chip Preprocessing

- Xilinx_QNN-MO-PYNQ_tiny-yolo-image
- Fixed Point Implementation of Tiny-Yolo-v2 using OpenCL on FPGA