XJTU-Tripler:
DNN Accelerator for Autonomous Agents

Boran Zhao, Wenzhe Zhao, Tian Xia, Fei Chen, Long Fan, Pengchen Zong,

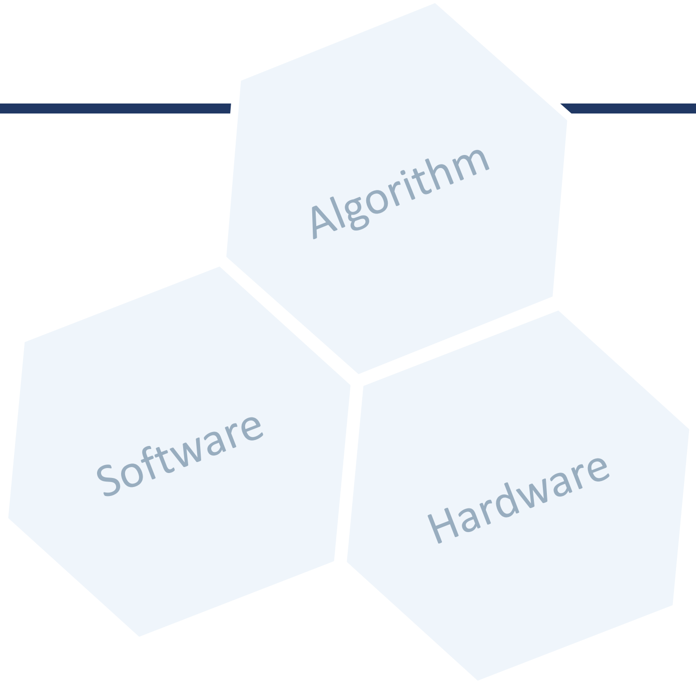
Yadong Wei, Zhijun Tu, Zhixu Zhao, Zhiwei Dong, and Pengju Ren*

XJTU-Tripler Group, IAIR, Xi’an Jiaotong University

(pengjuren@xjtu.edu.cn)*

2019-06-05

Outline



- Background & Motivation
- Network selection and quantization for single object detection
- HiPU: a General-purpose Fixed-point DNN accelerator on Xilinx FPGA
- Data flow of HiPU
- Optimization of HiPU and Zynq system
- Performance analysis
- Roadmap
- Demo 1: Dual-channel real-time object detection on Ultra96(Xilinx Zu3)
- Demo 2: Autonomous agents cooperation on TX2 now, on Ultra96 in near future

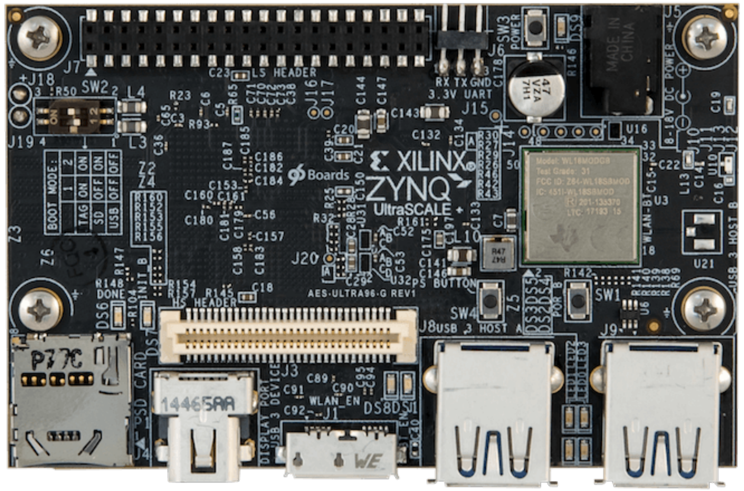
Background & Motivation

Background

The DAC system design contest aims to implement single-object detection algorithm on various class of pictures, including car, person, bicycle, and other indicated objects.

The best technology supporting such detection task is Deep Neural Network, which requires tremendous amount of computation.

Ultra96(Xilinx ZU3) is an outstanding platform to implement DNN accelerator with low power.



ZU3 resource	
LUT	70K
FF	141K
BRAM	216
DSP	360

Motivation

- Algorithm

Customized YOLO is exploited to trade off the accuracy and energy.

1 Replace the feature extraction with ShuffleNet V2

2 Exploit 8-bit quantization and fine tune technique

- FPGA implementation

A general-purpose DNN accelerator, HiPU, is implemented to support all of the special network operators.

1 HiPU supports CONV, depth-wise CONV, padding, pooling, RELU, channel shuffle/concat/divide

2 HiPU supports inter-layer optimization to reduce DDR bandwidth by the embedded RISC-V core

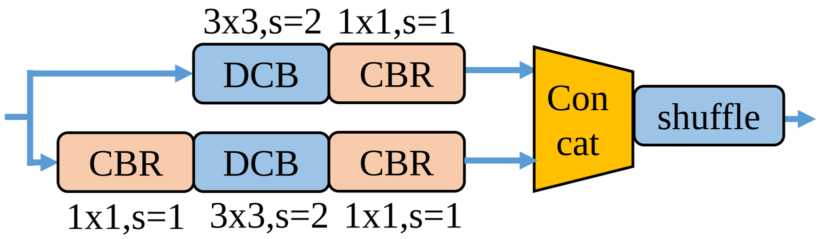
3 The system concurrently processing LD/CONV, PS/PL

Network selection for single object detection: YOLO with ShuffleNet V2

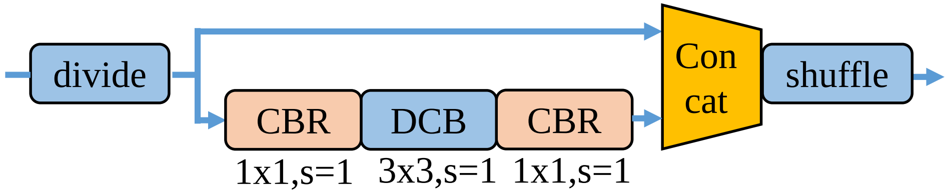
At first, we define CBR as the combination of Conv BN ReLU , and DCB is the combination of Depthwise conv BN .

Then, 2 typical bottlenecks of ShuffleNet V2 can be described as follows:

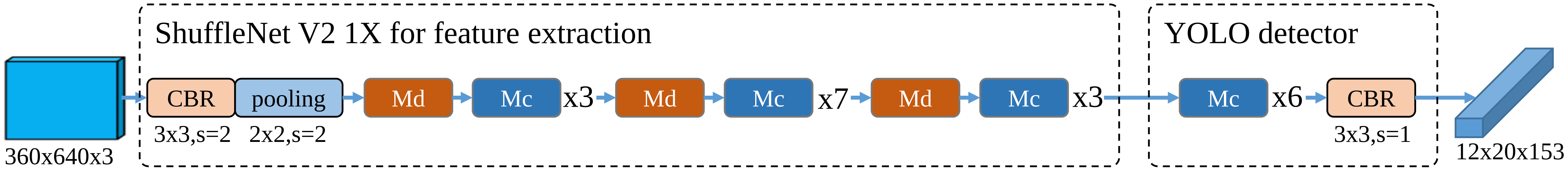
Module_d Md is:



Module_c Mc is:



Our final detection network is:



In order to reduce the parameter quantity of conventional YOLO, ShuffleNet V2 is exploited to replace the feature extraction layers.

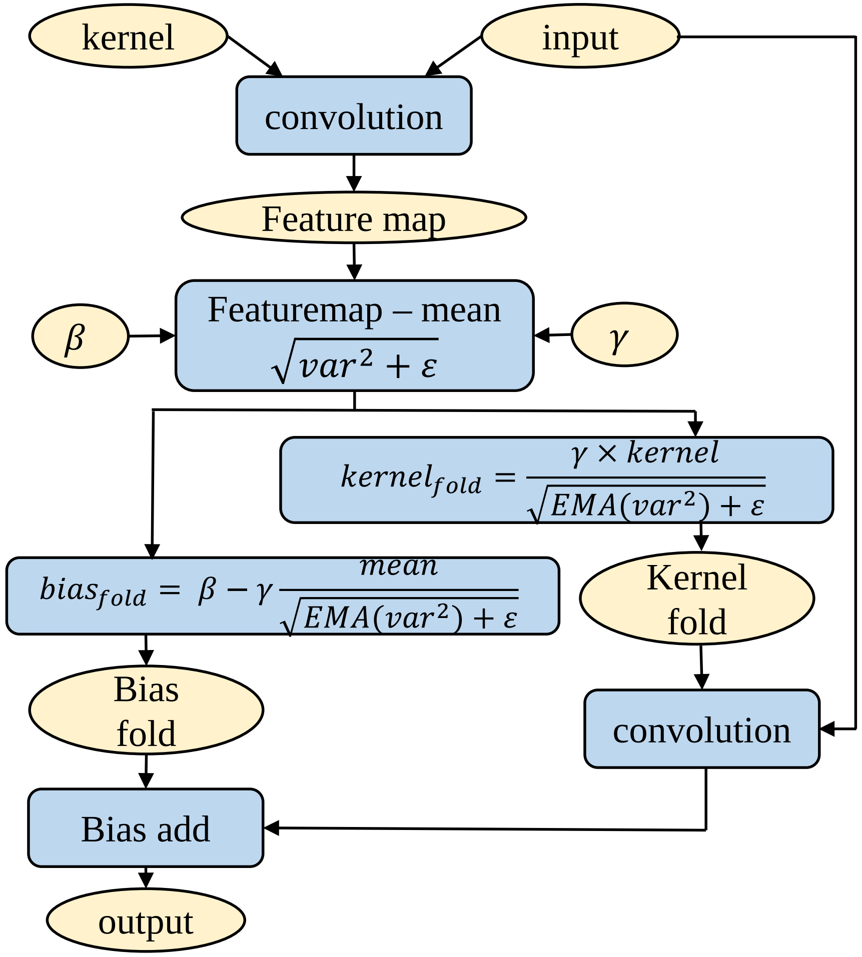
At first, the parameters of feature extraction layers are pre-trained on ImageNet classification dataset. Parameters before the first Module_c was reloaded and trained when it was fine-tuned on DAC dataset.

At last, YOLO detector on DAC19 dataset was trained.

To improve the detection accuracy, the input image remains original size.

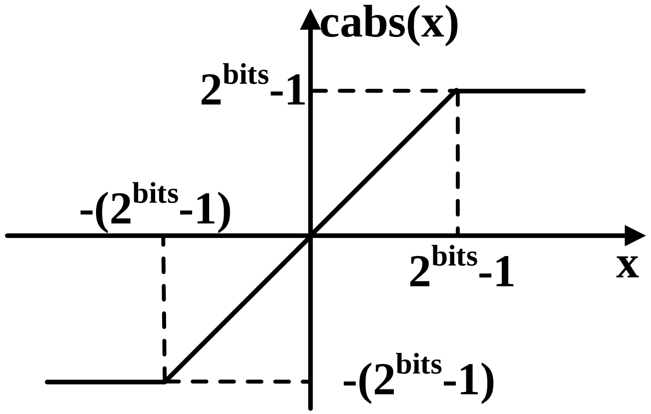
Quantization of Neural Network

Before quantization, make fusion of BN layer:


kernel
input
convolution
Feature map
β
$\frac{\text{Featuremap} - \text{mean}}{\sqrt{var^2 + \varepsilon}}$
γ
$kernel_{fold} = \frac{\gamma \times kernel}{\sqrt{EMA(var^2) + \varepsilon}}$
$bias_{fold} = \beta - \gamma \frac{mean}{\sqrt{EMA(var^2) + \varepsilon}}$
Kernel fold
Bias fold
convolution
Bias add
output

Then, $output = input \times kernel_{fold} + bias_{fold}$

8bit symmetry quantization:



$$ratio_{index} = \left[\frac{2^{bits} - 1}{max(tensor)}\right] \quad (1)$$

$$ratio = 2^{ratio\_index} \quad (2)$$

$$quan(x) = [x] \quad (3)$$

$$cabs(x) = \begin{cases} 2^{bits} - 1 & x > 2^{bits} - 1 \\ -(2^{bits} - 1) & x < -(2^{bits} - 1) \\ x & else \end{cases} \quad (4)$$

$$output = \frac{quan(cabs(ratio_a \times input)) \times quan(cabs(ratio\_k \times kernel))}{ratio\_a \times ratio\_k} + \frac{quan(cabs(ratio\_b \times bias))}{ratio\_b}$$

After quantization, the network should be fine-tuned to restore accuracy.

Once the fixpoint parameter is determined, transform the parameters to FPGA format.

The size of our proposed network is: Weight: 1.94MB, Bias: 78KB

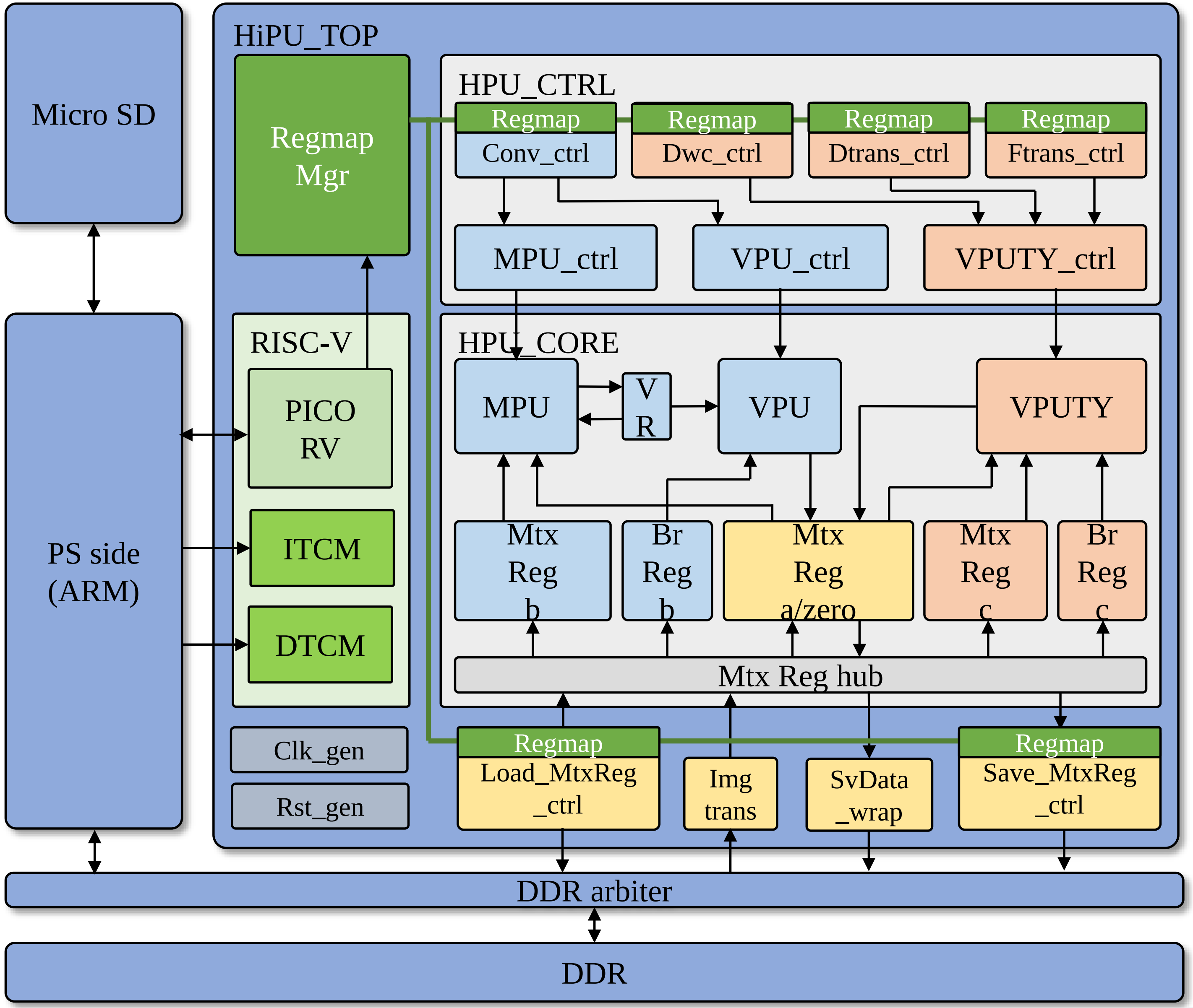
HiPU: a General-purpose Fixed-point DNN accelerator on Xilinx FPGA

HiPU is a general purpose processing unit, which can

1. calculate Matrix mul-add by MPU;
2. calculate Vector mul/add/cmp/shift operation by VPU or VPUTY;
3. calculate Scalar operations and controlling logic by embedded RISC-V core.

DNN operator can be processed by abovementioned modules:

DNN operator	Module of HiPU
Convolution	MPU, VPU
Depth-wise CONV	VPUTY
Padding	VPUTY, RISC-V schedule
Pooling	VPUTY
RELU	VPU, VPUTY
Channel shuffle	VPU, VPUTY
Concat/divide	RISC-V schedule



Data flow of HiPU: Load MtxReg(LDMR) and Save MtxReg(SVMR)

The weight, bias, and input image/feature map are stored in DDR.

Since HiPU only processes the data on MtxReg, we should transform data from DDR to MtxReg before calculation, and transform data from MtxReg to DDR after calculation is finished.

LDMR:

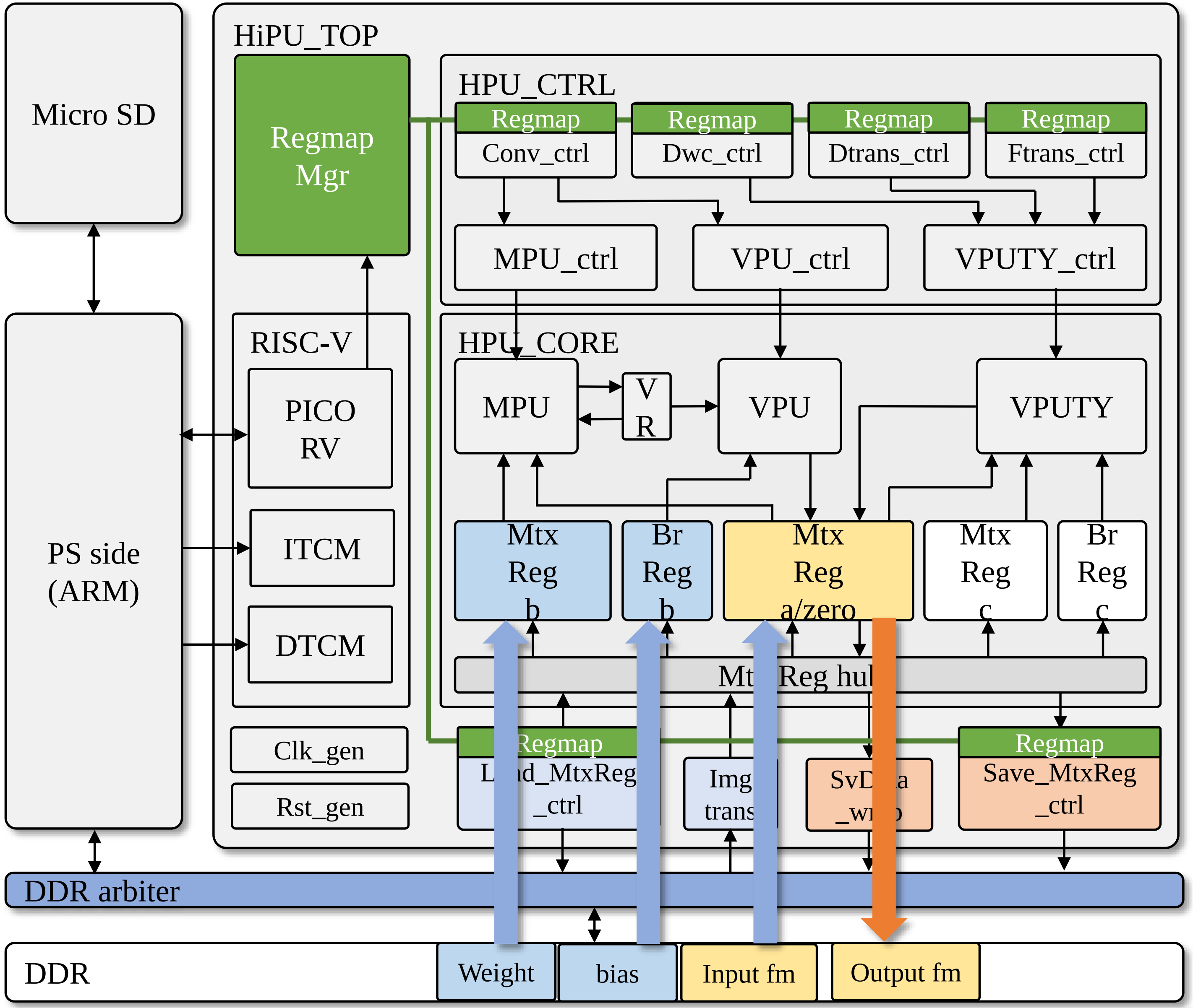
The conventional convolution data preparation is shown as the BLUE arrow of right figure.

According to the regmap configuration, LDMR_ctrl module transforms weight, bias, and input fm from DDR to MtxReg.

SVMR:

The conventional convolution data output is shown as the ORANGE arrow of right figure.

According to the regmap configuration, SVMR_ctrl module transforms output fm from MtxReg to DDR.



Data flow of HiPU: convolution(CONV) and depth-wise convolution(DWConv)

HiPU supports variety of operators, which can be classified in 4 group: Conv, Depth-Wise Calc(DWC), Data-Trans(Dtrans), and Format-Trans(Ftrans).

Take CONV and DWConv as example:

CONV:

Phase1: Conv_ctrl module sends a series of instructions to MPU and VPU;

Phase2: MPU reads weight and input fm from MtxReg;

Phase3: MPU makes Matix-mul-acc operations and output result to VPU;

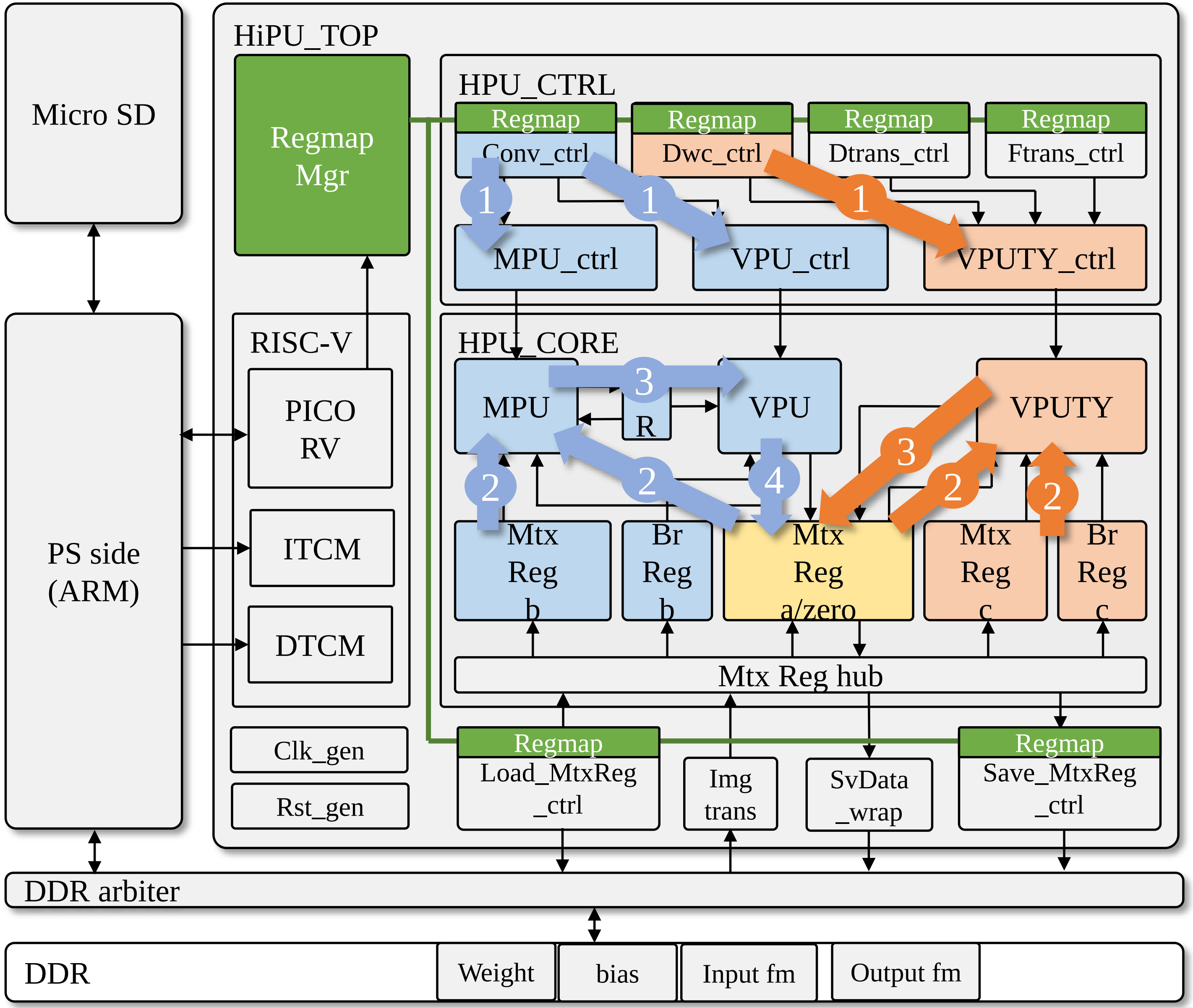
Phase4: VPU processes adding bias, ReLU, shuffle operations for every point of output fm, and writes back to MtxReg.

DWConv:

Phase1: Dwc_ctrl module sends a series of instructions to VPUTY;

Phase2: VPUTY reads weight and input fm from MtxReg;

Phase3: VPUTY makes Vector-mul-acc, adding bias, ReLU, shuffle operations, and write back to MtxReg.



Optimizations of HiPU

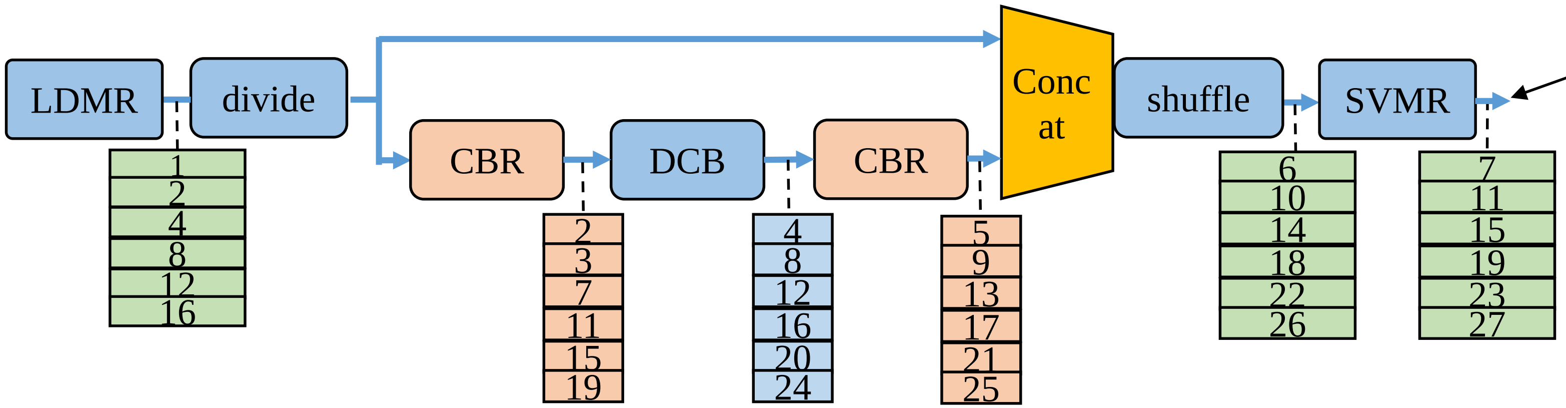
Tip 1: Processing all layers of a bottleneck together to reduce DDR bandwidth

Since the fm size at DNN bottleneck is limited, it is reasonable to load fm from a bottleneck starts, and write back fm from the bottleneck ends. A carefully schedule on RISC-V is needed.

Give module_c as example.

The number of block means the height of fm.

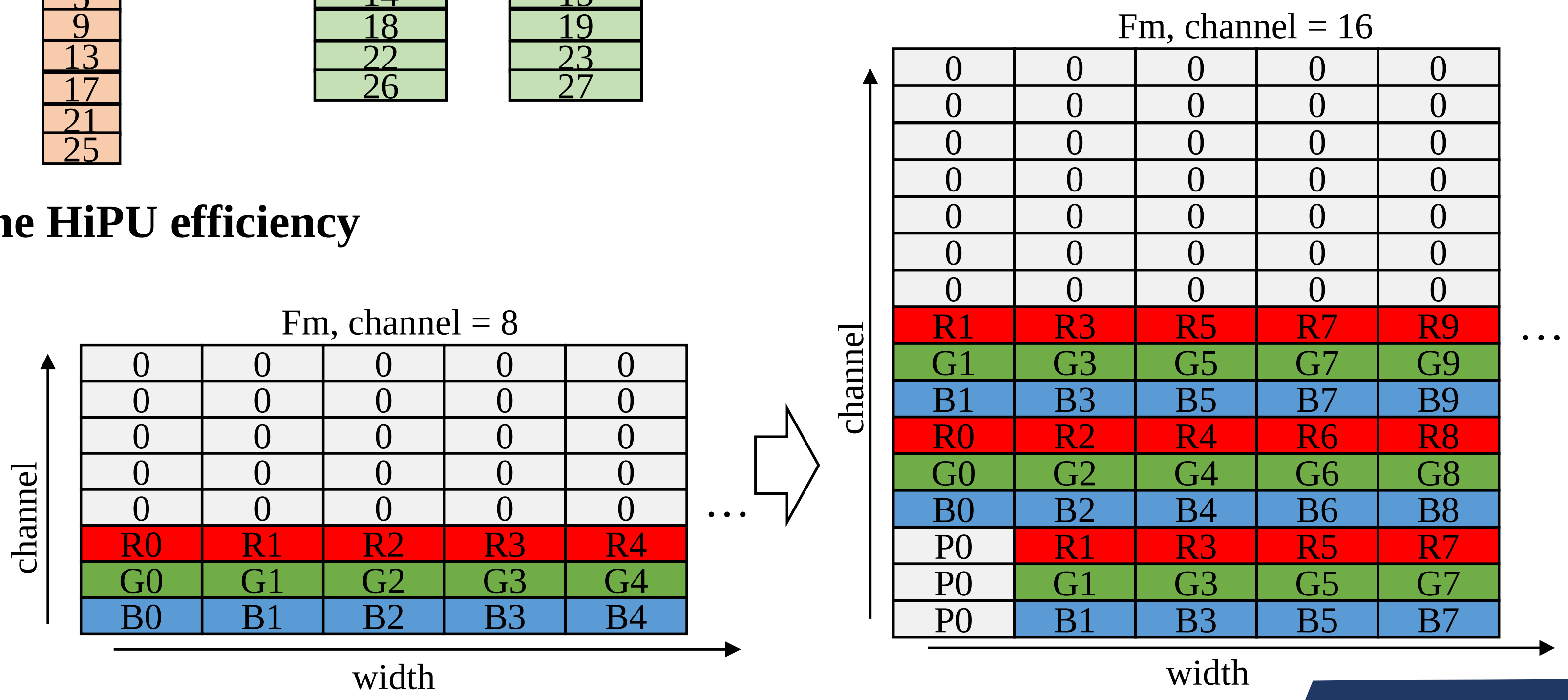
When processing Module_c, DDR read/write operation needs only 1 time.



Tip 2: Transform input image to increase the HiPU efficiency

Since the channel of input image is only 3, leading to low efficiency of HiPU calculating. HiPU has a special transform module to extend the channel of input image from 3 to 9 if the kernel of Conv1 is 3x3.

Increase the HiPU efficiency from 0.38 to 0.56 for Conv1



Fm, channel = 8

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
R0	R1	R2	R3	R4
G0	G1	G2	G3	G4
B0	B1	B2	B3	B4

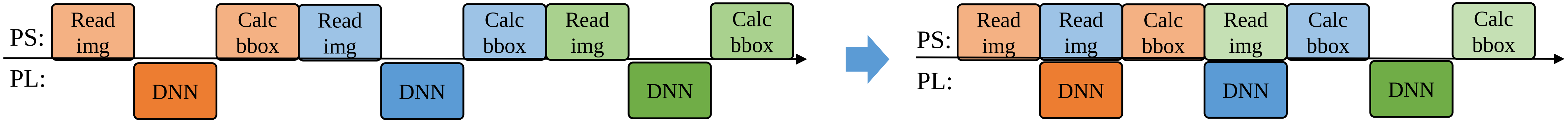
Fm, channel = 16

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
R1	R3	R5	R7	R9
G1	G3	G5	G7	G9
B1	B3	B5	B7	B9
R0	R2	R4	R6	R8
G0	G2	G4	G6	G8
B0	B2	B4	B6	B8
P0	R1	R3	R5	R7
P0	G1	G3	G5	G7
P0	B1	B3	B5	B7

Optimizations of Zynq system

Tip 1: Parallel in read image and HiPU processing

Since HiPU does not have dependence with ARM core on PS, read image can work concurrently with HiPU calculation.



The FPS of processing increases from 30.3Hz to 49.9Hz

Tip 2: PS computation is accelerated by C functions

Some time-consuming software routines are accelerated using C code and imported to Python with ctypes:

- Bounding box generation acceleration

  Customized C functions. Precompute the addresses of confidence data and bbox data in the output data of PL. Find the max confidence and the corresponding bbox, then compute the absolute bbox coordinates quickly at run time.

  The time of calc bbox module decreases from 2ms to 0.6ms

- Image reading acceleration

  Customized image reading C function using libjpeg.

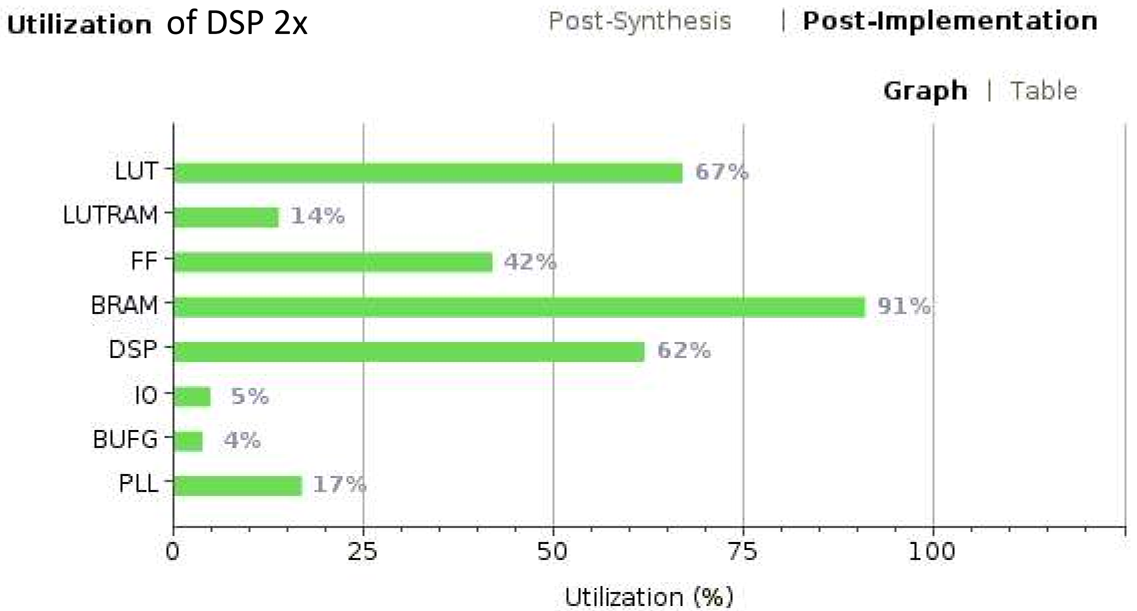
  This code is just finished, not include in our final version. This should improve the FPS from 49.9Hz to 52Hz.

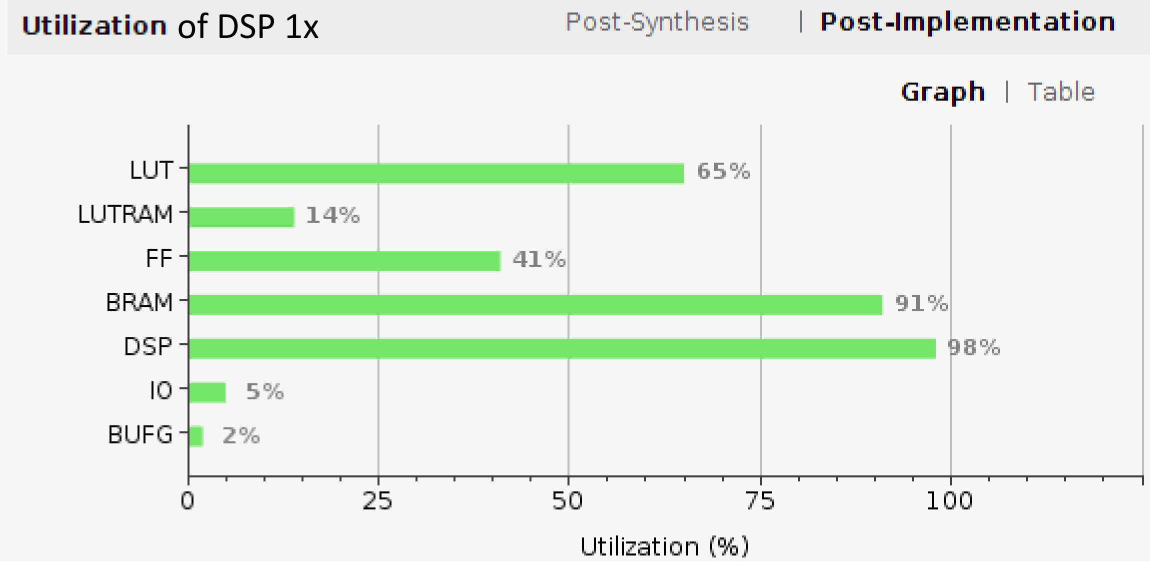
Tip 3: Exploit clock gating to reduce power

In order to reduce the energy consumption of HiPU, close the clock of PL side when HiPU finishes calculation of 1 picture.

Performance of HiPU

Characteristics of HiPU		
Frequency		233MHz
Peak Performance		268Gops
Efficiency		>80%
Developing API		C/RISC-V ASM
Operator support	CONV	Kernel size (1 – 7)
	Stride	Stride (1 – 256)
	Dilation	Dilation size(0 – 255)
	Padding	Padding size(0 – 255)
	Pooling	Range: (1-7), Global; Type: Min/Max/Ave
	Depth-wise Conv	Kernel size (1-7)
	concat/divide	Any
	Channel shuffle	Just support Grp=2
	DeConv	Will support in future
	Ele-wise Add/Mul	Support
	Activation	Only support ReLU
Reliance	Could be deployed on any kind of Xilinx FPGA (ZU, KU, VU, 7series) , do not rely on Zynq.	





Performance analysis

Test Environment： SD-Card type is SanDiskA1-U1-Class10-64GB, 40260 test images in size of 640*360

System Freq, DSP2x	Latency(ms)	FPS	Power(W)	Energy(KJ)	Efficiency
75MHz	62.89	15.90	5.88	19.43	0.37J/pic
166MHz	21.30	46.95	8.96	10.02	0.19J/pic
200MHz	21.20	47.17	9.08	10.11	0.19J/pic

System Freq, DSP1x	Latency(ms)	FPS	Power(W)	Energy(KJ)	Efficiency
133MHz	29.80	33.56	6.59	10.21	0.2J/pic
150MHz	26.90	37.17	6.89	9.7	0.18J/pic
166MHz	21.30	46.95	7.76	8.68	0.17J/pic
200MHz	21.30	46.95	7.79	8.71	0.17J/pic
233MHz	21.00	47.62	7.83	8.64	0.16J/pic
200MHz(Clk Gating)	20.84	47.98	7.21	7.89	0.15J/pic
200MHz(CG, opti conv1)	19.93	50.15	7.40	**7.75**	**0.15J/pic**

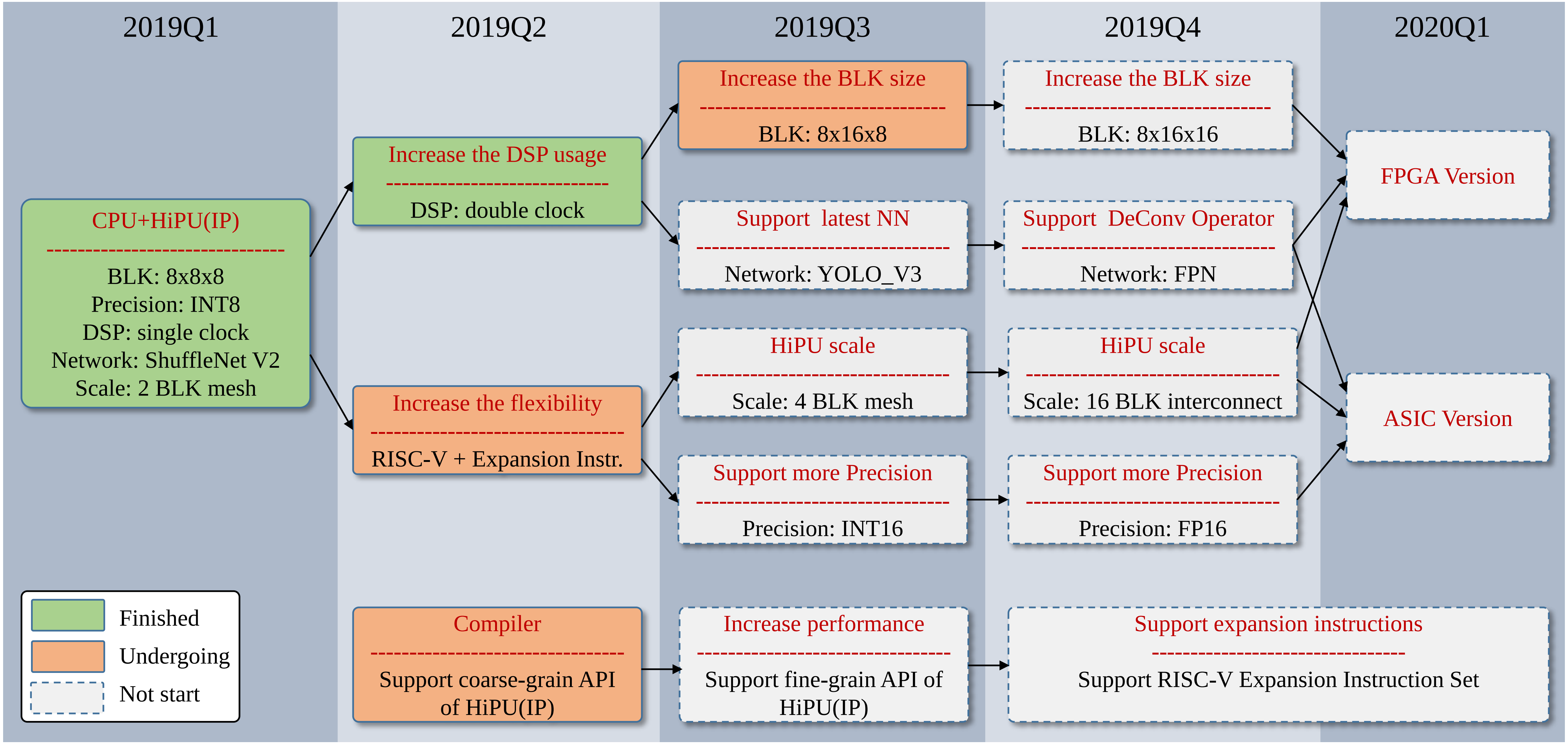
Notice: The energy data in the table are equivalent to the scale of the competition's test set (52,500 images).

Conclusion 1:
When system clock is higher than 166MHz, the FPS is difficult to increase. HiPU is not the bottleneck of the system.

Conclusion 2:
We tried to make DSP work on double clock to reduce the FPGA resource. However, the energy is higher than the equivalent single clock DSP version.

According to the performance analysis, we choose to submit the version of DSP1x 200MHz, with clock gating, optimized Conv1 schedule.

Roadmap



2019Q1

CPU+HiPU(IP)

BLK: 8x8x8
Precision: INT8
DSP: single clock
Network: ShuffleNet V2
Scale: 2 BLK mesh

2019Q2

Increase the DSP usage

DSP: double clock

Increase the flexibility

RISC-V + Expansion Instr.

Compiler

Support coarse-grain API of HiPU(IP)

2019Q3

Increase the BLK size

BLK: 8x16x8

Support latest NN

Network: YOLO_V3

HiPU scale

Scale: 4 BLK mesh

Support more Precision

Precision: INT16

Increase performance

Support fine-grain API of HiPU(IP)

2019Q4

Increase the BLK size

BLK: 8x16x16

Support DeConv Operator

Network: FPN

HiPU scale

Scale: 16 BLK interconnect

Support more Precision

Precision: FP16

Support expansion instructions

Support RISC-V Expansion Instruction Set

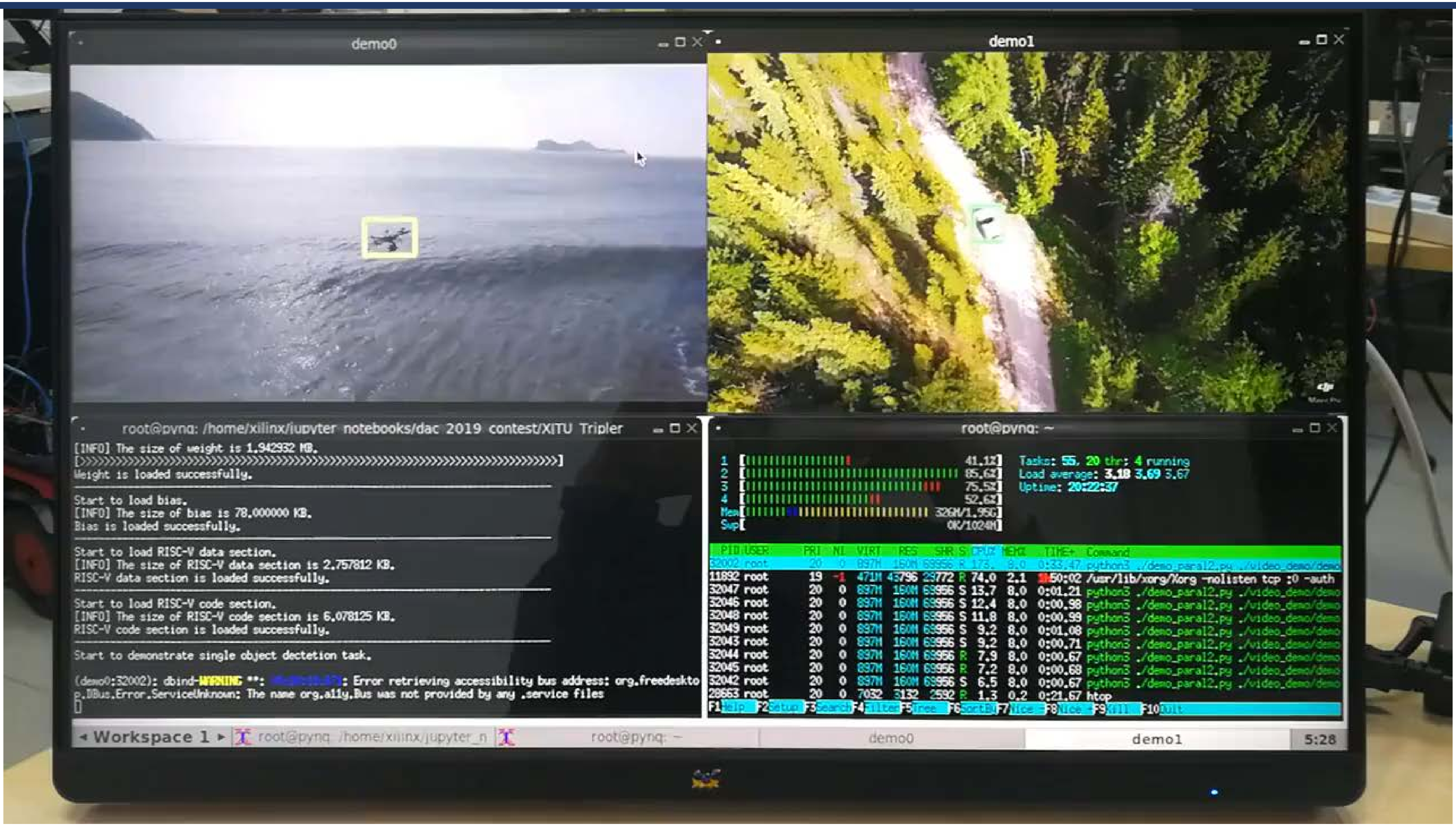
2020Q1

FPGA Version

ASIC Version

- 🟩 Finished
- 🟧 Undergoing
- ⬜ Not start

Demo 1: Dual-channel real-time object detection on Ultra96(Xilinx Zu3)


demo0
demo1
root@pynq: /home/xilinx/jupyter notebooks/dac 2019 contest/XITU Tripler
root@pynq: ~
Workspace 1
root@pynq /home/xilinx/jupyter_n
root@pynq: ~
demo0
demo1
5:28

```
[INFO] The size of weight is 1.942932 MB.
[>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>]
Weight is loaded successfully.
------------------------------------------------------------------------------
Start to load bias.
[INFO] The size of bias is 78.000000 KB.
Bias is loaded successfully.
------------------------------------------------------------------------------
Start to load RISC-V data section.
[INFO] The size of RISC-V data section is 2.757812 KB.
RISC-V data section is loaded successfully.
------------------------------------------------------------------------------
Start to load RISC-V code section.
[INFO] The size of RISC-V code section is 6.078125 KB.
RISC-V code section is loaded successfully.
------------------------------------------------------------------------------
Start to demonstrate single object dectetion task.

(demo0:32002): dbind-WARNING **: [illegible]: Error retrieving accessibility bus address: org.freedeskto
p.DBus.Error.ServiceUnknown: The name org.a11y.Bus was not provided by any .service files
```

```
  1  [||||||||||||||||||                       41.1%]   Tasks: 55, 20 thr; 4 running
  2  [||||||||||||||||||||||||||||||||||||||||  85.6%]   Load average: 3.18 3.69 3.67
  3  [||||||||||||||||||||||||||||||||||||||    75.5%]   Uptime: 20:22:37
  4  [||||||||||||||||||||||||||               52.6%]
  Mem[|||||||||||||||||||||||||||||||||   326M/1.95G]
  Swp[                                      0K/1024M]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
32002 root       20   0  897M  160M 69956 R 173.  8.0  0:33.47 python3 ./demo_paral2.py ./video_demo/demo
11892 root       19  -1  471M 43796 29772 R 74.0  2.1  1h50:02 /usr/lib/xorg/Xorg -nolisten tcp :0 -auth
32047 root       20   0  897M  160M 69956 S 13.7  8.0  0:01.21 python3 ./demo_paral2.py ./video_demo/demo
32046 root       20   0  897M  160M 69956 S 12.4  8.0  0:00.98 python3 ./demo_paral2.py ./video_demo/demo
32048 root       20   0  897M  160M 69956 S 11.8  8.0  0:00.99 python3 ./demo_paral2.py ./video_demo/demo
32049 root       20   0  897M  160M 69956 S  9.2  8.0  0:01.08 python3 ./demo_paral2.py ./video_demo/demo
32043 root       20   0  897M  160M 69956 S  9.2  8.0  0:00.71 python3 ./demo_paral2.py ./video_demo/demo
32044 root       20   0  897M  160M 69956 R  7.9  8.0  0:00.67 python3 ./demo_paral2.py ./video_demo/demo
32045 root       20   0  897M  160M 69956 R  7.2  8.0  0:00.68 python3 ./demo_paral2.py ./video_demo/demo
32042 root       20   0  897M  160M 69956 S  6.5  8.0  0:00.67 python3 ./demo_paral2.py ./video_demo/demo
28663 root       20   0  7032  3132  2592 R  1.3  0.2  0:21.67 htop
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Demo 2: Autonomous agents cooperation on TX2 now, on Ultra96 in near future

Cooperative Navigation System of UAV and UGV

空地协同导航系统

Cognitive Architecture Group

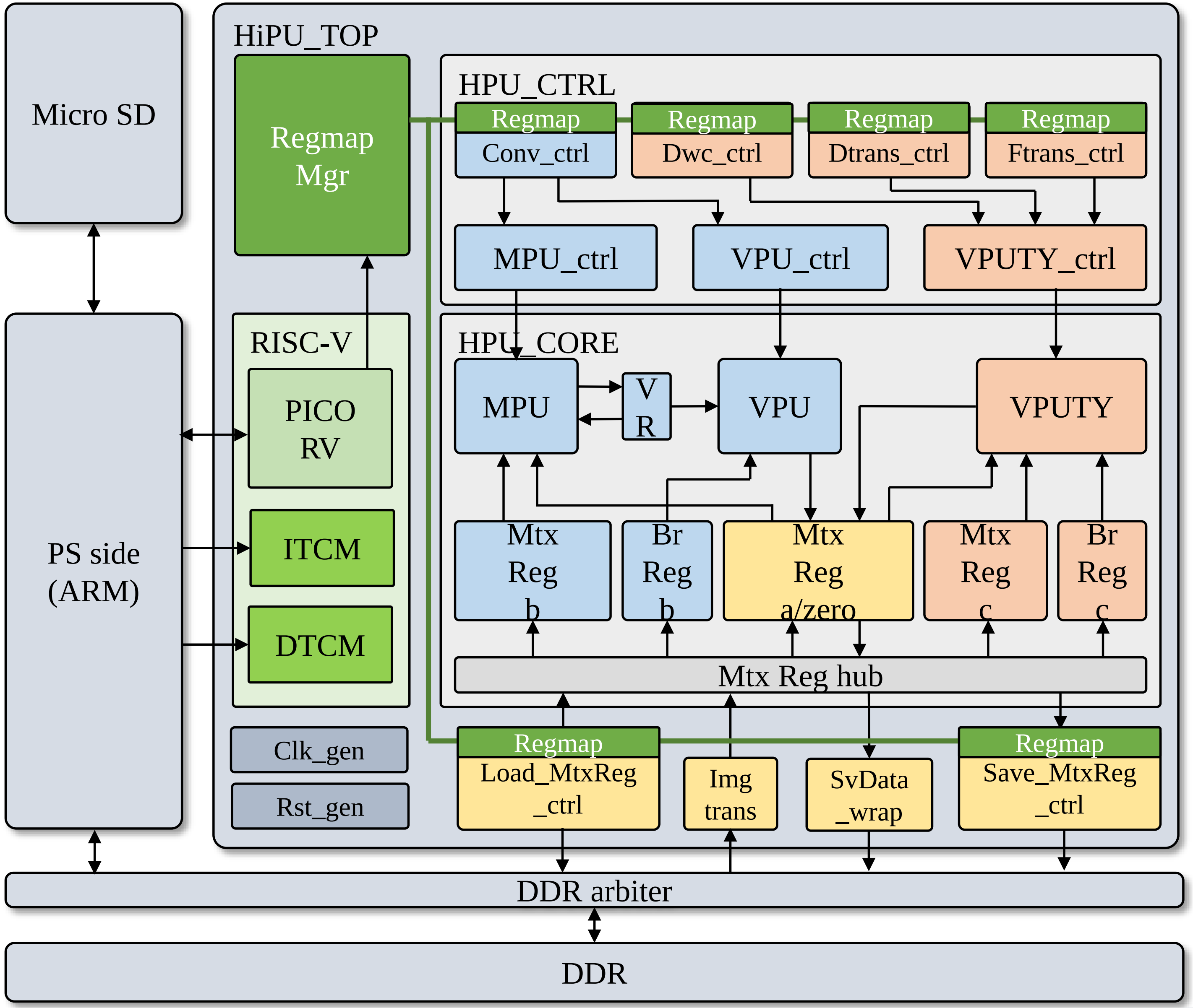
Xian Jiaotong University

Q & A

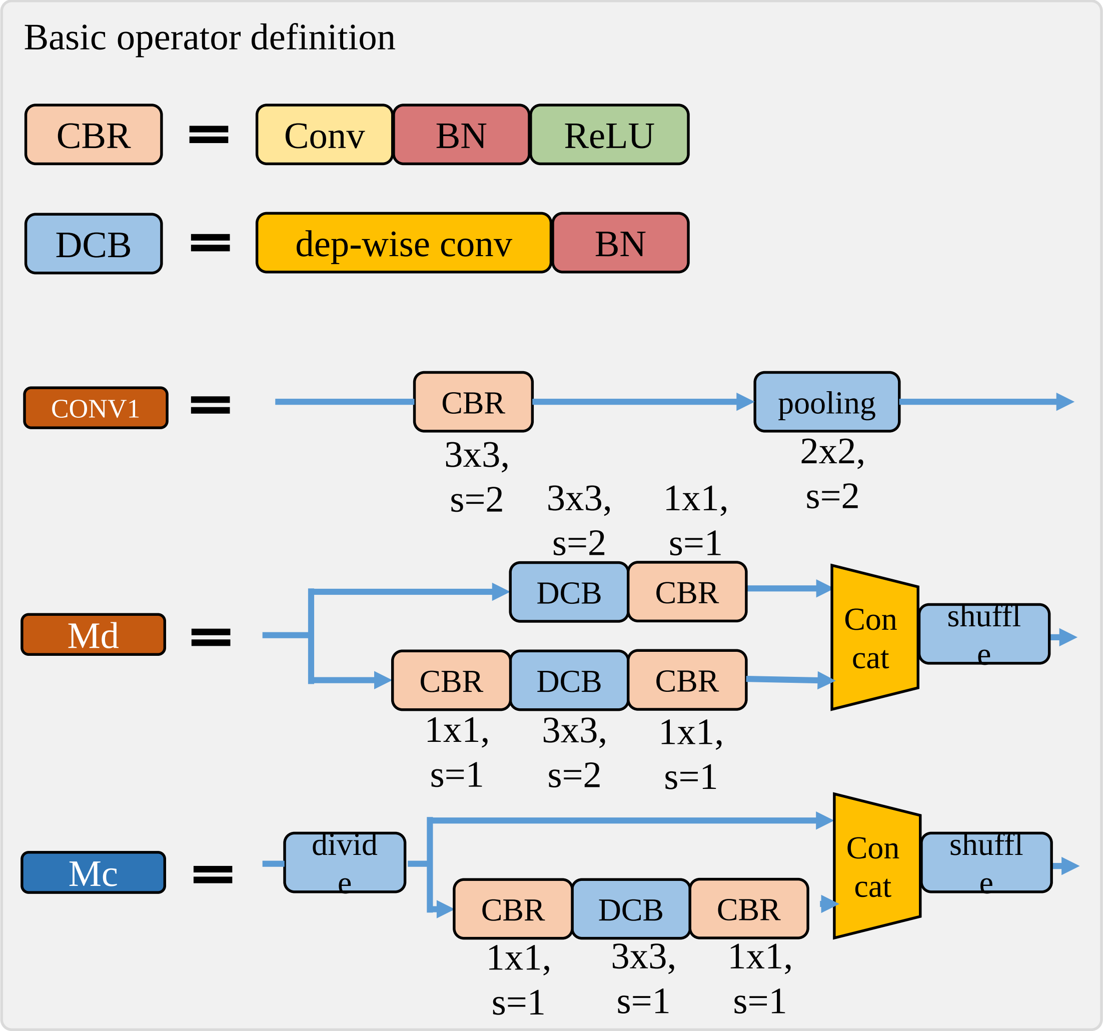
Thank you for your attention

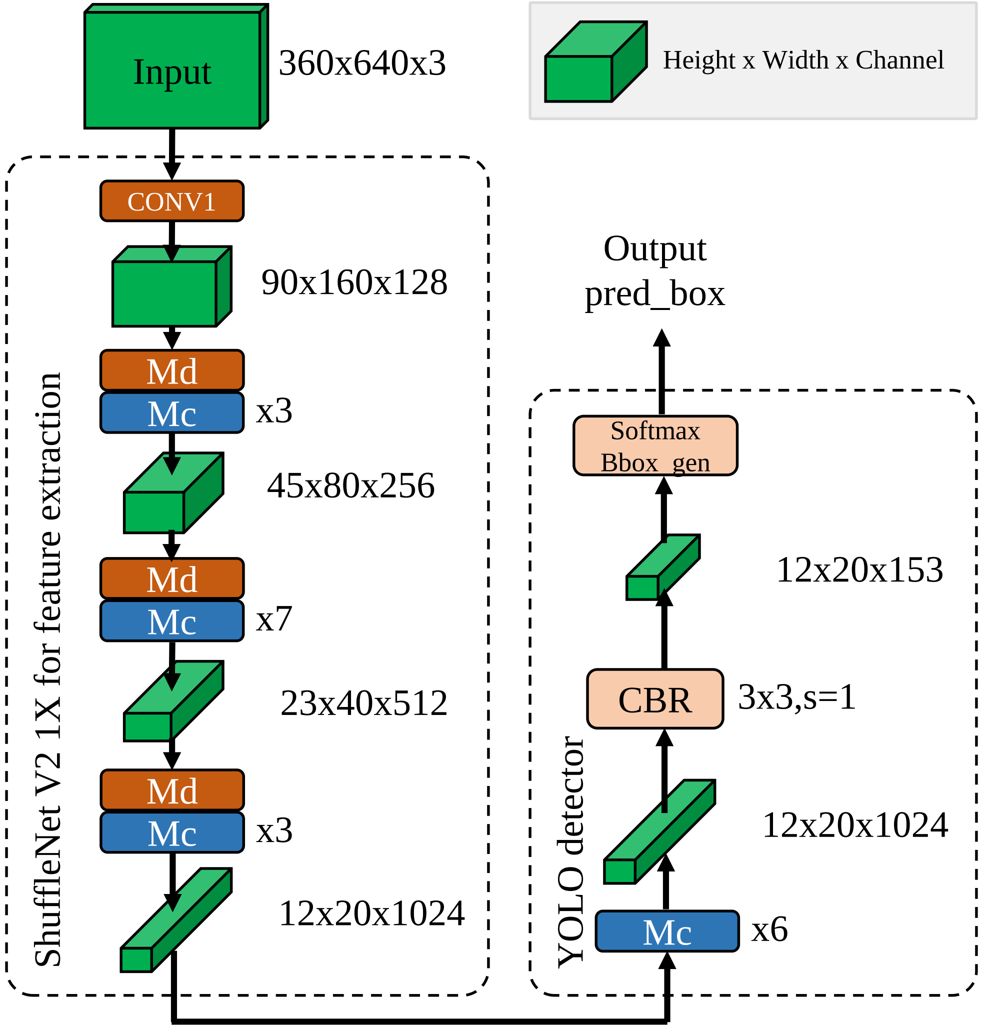
HiPU: a General-purpose Fixed-point DNN accelerator on Xilinx FPGA

Characteristics of HiPU		
Frequency		233MHz
Peak Performance		268Gops
Efficiency		>80%
Developing API		C/RISC-V ASM
Operator support	CONV	Kernel size (1 – 7)
	Stride	Stride (1 – 256)
	Dilation	Dilation size(0 – 255)
	Padding	Padding size(0 – 255)
	Pooling	Range: (1-7), Global; Type: Min/Max/Ave
	Depth-wise Conv	Kernel size (1-7)
	concat/divide	Any
	Channel shuffle	Just support Grp=2
	DeConv	Will support in future
	Ele-wise Add/Mul	Support
	Activation	Only support ReLU
Reliance	Could be deployed on any kind of Xilinx FPGA (ZU, KU, VU, 7series) , do not rely on Zynq.	



Network selection for single object detection: YOLO with ShuffleNet V2


Basic operator definition
CBR = Conv BN ReLU
DCB = dep-wise conv BN
CONV1 = CBR 3x3, s=2 pooling 2x2, s=2
Md = DCB CBR 3x3, s=2 1x1, s=1
CBR DCB CBR 1x1, s=1 3x3, s=2 1x1, s=1
Con cat
shuffl e
Mc = divid e
CBR DCB CBR 1x1, s=1 3x3, s=1 1x1, s=1
Con cat
shuffl e


Input
360x640x3
Height x Width x Channel
ShuffleNet V2 1X for feature extraction
CONV1
90x160x128
Md
Mc
x3
45x80x256
Md
Mc
x7
23x40x512
Md
Mc
x3
12x20x1024
Output
pred_box
YOLO detector
Softmax
Bbox_gen
12x20x153
CBR
3x3,s=1
12x20x1024
Mc
x6