Mipsology

Zebra®

VMAccel™ FPGA Cloud

AMD/Xilinx® Versal VCK5000

Getting Started Guide

Zebra Version: V2022.versal.alpha

Mipsology

Table of Contents

Mipsology

1 Introduction & Scope

This document is a getting started guide for *running* Mipsology® Zebra CNN inference acceleration software *on* AMD/Xilinx® VCK5000 PCIe Acceleration Card *hosted at* VMAccel® FPGA Cloud.

This is an Alpha release with following goals:

- Demonstrates Zebra functionality on VCK5000 board.
- Demonstrates Zebra software Ease-of-Use.
    - Accelerate post-training Convolution Neural Network (CNN) model *without* any structural modification. NO pruning or re-training of the model.
    - Automatic and in-line quantization/calibration. No offline or separate compilation tool.
- Performance for many CNNs is high; but not best possible. Details in later section.
- Accuracy for most models is high; but not best possible. Details in later section.

2 License

VMAccel Zebra virtual machines (VM) are pre-configured with software license. This license is **not** designed for production deployment. Any CNN inference running continuously for more than 15minutes will experience significant slowdown in execution.

3 Contact & Support

Please email support@mipsology.com for questions, concerns, technical help or discuss your project's unique requirements.

Please email licenses@mipsology.com for questions related to Zebra License.

4 Requirements

VMAccel Zebra instances are pre-configured with all required software and hardware. Only requirement on client side is a computer with internet connection and web browser.

5 VMAccel Cloud Access

To gain access to VMAccel cloud, please fill the form https://www.vmaccel.com/zebrademo

For any questions or concerns, please contact support@vmaccel.com

6 Launching Zebra VM Instances

Once you have the access credentials from VMAccel, please follow "Getting Started" section from https://vmaccel.atlassian.net/wiki/spaces/docs/pages/59212022/Getting+Started

Summary:

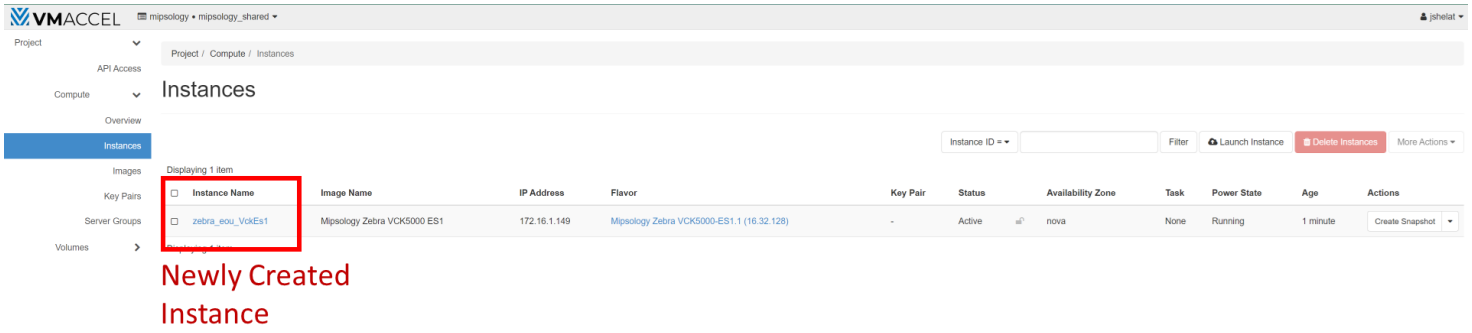
- In a web browser navigate to: https://xilinx2.vmaccel.com/dashboard/project/
- On left hand side: Click on "Instances"
- On right hand side: Click on "Launch Instance"
- Follow GUI instructions/options for configuration

IMP: Use following details when creating Zebra VM instance:

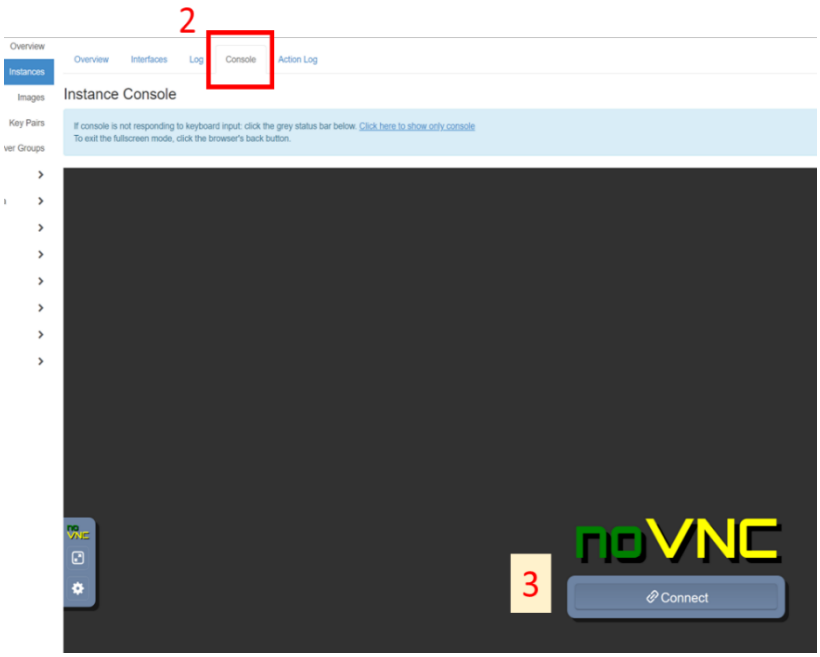
- Source : Secure Boot Image = "Mipsology Zebra VCK5000 ES1"
- Flavor = "Mipsology Zebra VCK5000-ES1.1 (16.32.128)"
- Network = "mipsology_local"

For advanced features like enabling external `ssh` access, follow instructions here : https://vmaccel.atlassian.net/wiki/spaces/docs/pages/38830174/Connect+to+Instance+via+SSH

Once created, you should see the instance listed as a row on "Instances" web page/view. Example screenshot:



7 Starting Zebra on VM Instance



After the VM instance is created successfully:

1. Start the instance by clicking on the "Instance Name"
2. Click on "Console"
3. Connect to "noVNC"

Inside VNC session:

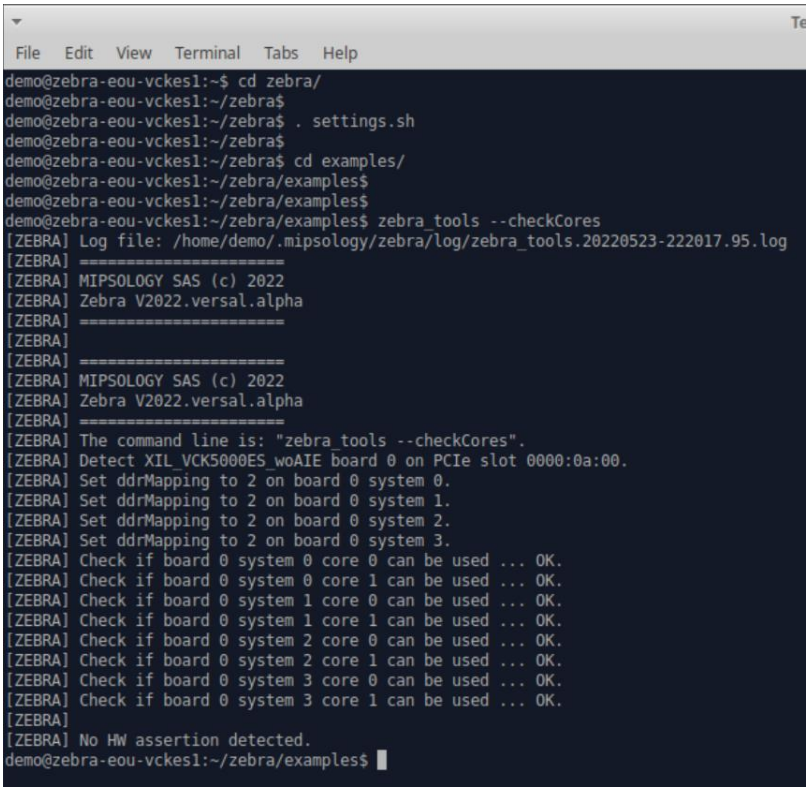
Right-click on desktop
Click on "Open Terminal Here"

```
cd
```

```
cd zebra/V2022.versal.alpha/examples
```

```
./docker/run.bash
```
(start Zebra Docker)

```
cd zebra
. settings.sh
cd examples
zebra_tools --checkCores
```



Zebra is ready IF above command shows status without any errors.

At this point, User is inside Zebra docker

running on user-configured Zebra VM instance

targeting VCK5000 board.

Mipsology

8 Mipsology Examples

Zebra provides example scripts and software to execute inference on various CNN post-trained models. All the models are open-source – i.e. downloaded from internet (graph and weights/parameters).

8.1 Quick start

Inside Zebra docker

```
cd
cd zebra
. settings.sh
cd examples

./run_classification.sh -n resnet50 -f tensorflow
```

(Run TensorFlow1 ResNet-50v1 Inference on VCK5000)

- NOTE: when a network is executed for 1st time, Zebra will automatically perform Quantization/Calibration for INT8 inference. Result of calibration is saved and reused in future.

```
./run_classification.sh -n resnet50 -f pytorch
```

(Run PyTorch ResNet-50v1.5 inference on VCK5000. NOTE: when executed 1st time, the model gets automatically downloaded from Internet. Downloaded models are saved and reused in future.)

```
./run_classification.sh -n resnet50 -f tensorflow2
```

(Run TensorFlow2 ResNet-50 Inference on VCK5000. NOTE: when executed 1st time, the model gets automatically downloaded from Internet. Downloaded models are saved and reused in future.)

Results of inferences can be found in `predict.log` file.

8.2 Details

Zebra provided example application software is intended to demonstrate CNN inference with the input being the post-training model. All the models are either pre-downloaded from open-source links or, in most cases, get automatically downloaded when executing the script.

Intent of Examples is to demonstrate seamless flow for FPGA acceleration of the CNN model.
Intent of Examples is NOT to demonstrate full application execution.

User interface is "`run_classification.sh`" script located inside `examples` directory. The associated python software is developed by Mipsology. This can be seen as an application that computes inference for chosen post-trained ***network*** (a.k.a. model) inside user's ***framework***.

Table below shows list of networks supported across frameworks in current release. Please use ***network*** and ***framework*** names in this table to run inference on associated model. For e.g.:

PyTorch ResNet-18 : `./run_classification.sh -f pytorch -n resnet18`
TensorFlow2 ResNet101 : `./run_classification.sh -f tensorflow2 -n resnet101`
TensorFlow1 Inceptionv4 : `./run_classification.sh -f tensorflow -n inceptionv4`

pytorch	tensorflow	tensorflow2
Model Source: https://pytorch.org/vision/0.9/models.html	Model Source: Open-source models gathered from internet.	Model Source: https://tfhub.dev/ https://storage.googleapis.com/
resnet50	inceptionv4	resnet50
resnet18	inceptionv3	resnet50v2
resnet34	vgg16	resnet101
resnet101	vgg19	resnet101v2
resnet152	resnet50	resnet152
densenet121	resnet152	resnet152v2
densenet161	mobilenet_v1	inceptionv1
densenet169	mobilenet_v2	inceptionv2
densenet201	yolov1	inceptionv3
inceptionv3	yolov2	inception_resnet_v2
mobilenet_v2	yolov3	mobilenet_v1
wide_resnet50_2*		mobilenet_v2
wide_resnet101_2*		vgg16
squeezenet		vgg19
squeezenet1_1		
vgg11		
vgg11_bn		
vgg13		
vgg13_bn		
vgg16		
vgg16_bn		
vgg19		
vgg19_bn		

* some reports of VCK5000 card instability. Under investigation. May be due to card thermal/power.

`run_classification.sh` has many options that you can discover by adding the option "--help". Note that some options will force the mapping, optimization and/or quantization of the network to be redone as they impact the way the computation of the network is done on Zebra; then requiring more time than a simple inference execution.

Note that in the log file, all lines not preceded by "[ZEBRA]" are from libraries (like Python). "[ZEBRA]" is the header for lines printed by Zebra or by the application.

Expectation for Future Release

In upcoming releases, Zebra is expected to support and demonstrate increasing number of CNN models (and associated layers) across all three popular ML Frameworks.

9 Mipsology Demos

Zebra software ships with many demos inside the provided Docker image. All docker demos, including application software and post-trained CNN models, are sourced from open-source GitHub repositories.

9.1 Quick Start

Inside Zebra docker

```
cd
source zebra/settings.sh
cd tensorflow-yolov4-tflite
```

(taking YOLO-v4 as an example)

```
zebra_config --system --add runSession.enableTimeStatistics=true
```

(enable printing performance table/statistics at end of Zebra run. By adding "`system`" flag, this option is enabled globally for all subsequent Zebra runs.)

```
./run ~/zebra/examples/VIDEO/paris_cut1.mkv
```

(run TF1 YOLO-v4 inference on VCK5000)

NOTES:

- By default, the application will show output video/pictures in a new GUI window.
- By default, inference is executed using batch=1.
  - Which means only 1 Zebra core is being utilized.
  - For VCK5000 this means performance is 1/8th of full Zebra performance.
- All this information is available in the summary table printed (in terminal window) by Zebra at end of inference execution.
- This demo supports other pretrained CNN models – YOLO-v3, TinyYOLO-v4 and TinyYOLO-v3.
- Please study the '`run`' script and `detect.py` application for more options.

9.2 Details

Zebra executes inside an ML framework. Zebra executes "in-line" with user application, including Quantization and Calibration. When accelerating CNN inference with Zebra, there is no additional tool for offline processing. Mipsology does not provide a model-zoo because Zebra software is designed to accelerate neural networks trained on GPU without modification.

Intent of Demos is to demonstrate effortless FPGA acceleration of applications.
Intent of Demos is not to demonstrate a fully optimized and ready-to-deploy application.

Zebra aims to run inference with no change to application software and the model/graph. However, many GitHub repositories are not designed for easy execution on CPU or any accelerator (including GPU). Hence, Zebra makes following modifications for smooth User Experience (UX):

a) For repositories that 'only' give post-trained weights; Zebra generates a frozen graph before running the demo when required.

b) Provide ability to use videos OR image OR directory_of_images as input.
c) Provide a 'run' script.

This is a wrapper script that enables all demos to run with similar command line. E.g.:

```
./run <input_source> [--batch B] [--out_file <file>] [--inputSize WxH]
```

- input_source = video file / image file / directory with images / usb cameras
- B = size of batch to use. Default = 1
- out_file = video file to save the output. Default = display output in new window.
- WxH = input image size to Neural Network for inference.
- Unless the post-trained model has strict restrictions, the input image size can be user defined.

User is encouraged to study the 'run' script and related application `*.py` code to understand various options.

Table below shows the various demos and associated CNN model along with how to enable Zebra.

Demo Name	Framework	Supported CNN Models	Command to enable Zebra
darkflow	TF1	YOLO-v2, TinyYOLO-v2	`source settings.sh`
tensorflow-yolo3	TF1	YOLO-v3	`source settings.sh`
tensorflow-yolov4-tflite	TF1	YOLO-v4, TinyYOLO-v4, YOLO-v3, TinyYOLO-v3	`source settings.sh`
yolov5	PT	YOLO-v5 N/S/M/L/N6/S6/M6 NOTE: This demo supports 7 different models. Models X, L6 and X6 are not supported in this release.	`source settings.sh`
Ildoonet-tf-pose-estimation	TF1	Pose-Estimation	`source settings.sh legacy`
EDSR	PT	Super Resolution	`source settings.sh`
VDSR	TF1	Super Resolution	`source settings.sh`

Table below gives list of GitHub source link:

Demo Name	GitHub Source Link
darkflow	https://github.com/thtrieu/darkflow
tensorflow-yolo3	https://github.com/aloyschen/tensorflow-yolo3
tensorflow-yolov4-tflite	https://github.com/hunglc007/tensorflow-yolov4-tflite
yolov5	https://github.com/ultralytics/yolov5
Ildoonet-tf-pose-estimation	https://github.com/jiajunhua/ildoonet-tf-pose-estimation
EDSR	https://github.com/thstkdgus35/EDSR-PyTorch
VDSR	https://github.com/DevKiHyun/VDSR-Tensorflow

Mipsology

9.3 Input for Docker Demos

Docker demos can accept input in either of the following formats:

- Video file
- Image file
- Directory of Images

Users are encouraged to use the input of their choice. Off course some demos may need appropriate input – for example Pose-Estimation demo needs input where it can detect human pose. Unless specified by the demo, the input can be of variable dimensions.

To make it easier for Users to run the demo on VMAccel Cloud instances, Mipsology provides sample inputs videos. These video files are located inside `~/zebra/examples/VIDEO` directory (let's call it `<VID_DIR>` in table below) which is automatically mounted when starting the docker container.

Table below gives example of command to start the demos.

Demo Name	Basic Command	Optional Switches
darkflow	`./run <VID_DIR>/paris_cut1.mkv`	`--batch 8 --out_file YLv2_dk_out.mp4`
tensorflow-yolo3	`./run <VID_DIR>/paris_cut2.mkv`	`--batch 8 --out_file YLv3_tf_out.mp4`
tensorflow-yolov4-tflite	`./run <VID_DIR>/paris_cut3.mkv`	`--batch 8 --out_file YLv4_tf_out.mp4`
yolov5	`./run <VID_DIR>/paris_cut2.mkv`	`--batch 8 --out_file YLv5s_tf_out.mp4`
Ildoonet-tf-pose-estimation	`./run <VID_DIR>/kulam_dance_27sec.mp4`	`--batch 8 --out_file pose_tf_out.mp4`

NOTE: the output videos will be lost when docker is closed and when saved inside docker the video cannot be viewed. User can choose to save the video outside docker by mounting a directory when starting the docker container. For e.g.: `./docker/run.bash -v <dir_of choice>:/VIDS`

Expectation for Future Release

In upcoming releases, Zebra is expected to enable more demos from open-source repositories covering wide variety of CNN models and end applications.

10 Performance

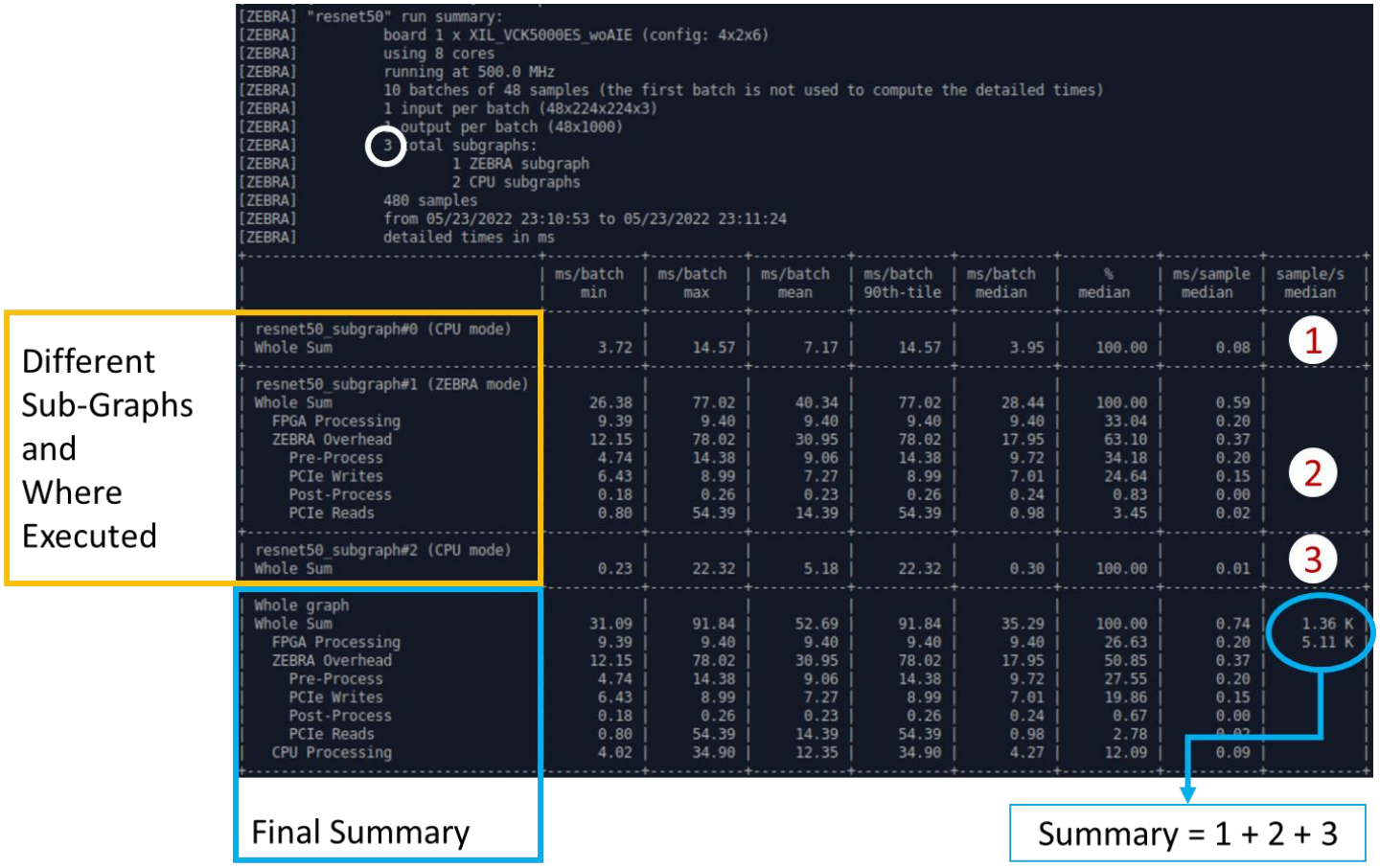
In this Alpha release, current Zebra performance is medium-to-high depending on CNN. In near future we expect rolling releases/updates with significant performance improvements to all supported CNNs. For example, when targeting ResNet50 with a dedicated configuration, Zebra today can achieve 1.5x-3x better FPS (depending on framework and model). And we expect to boost this number further. Similar increase in performance is expected for all supported CNNs.

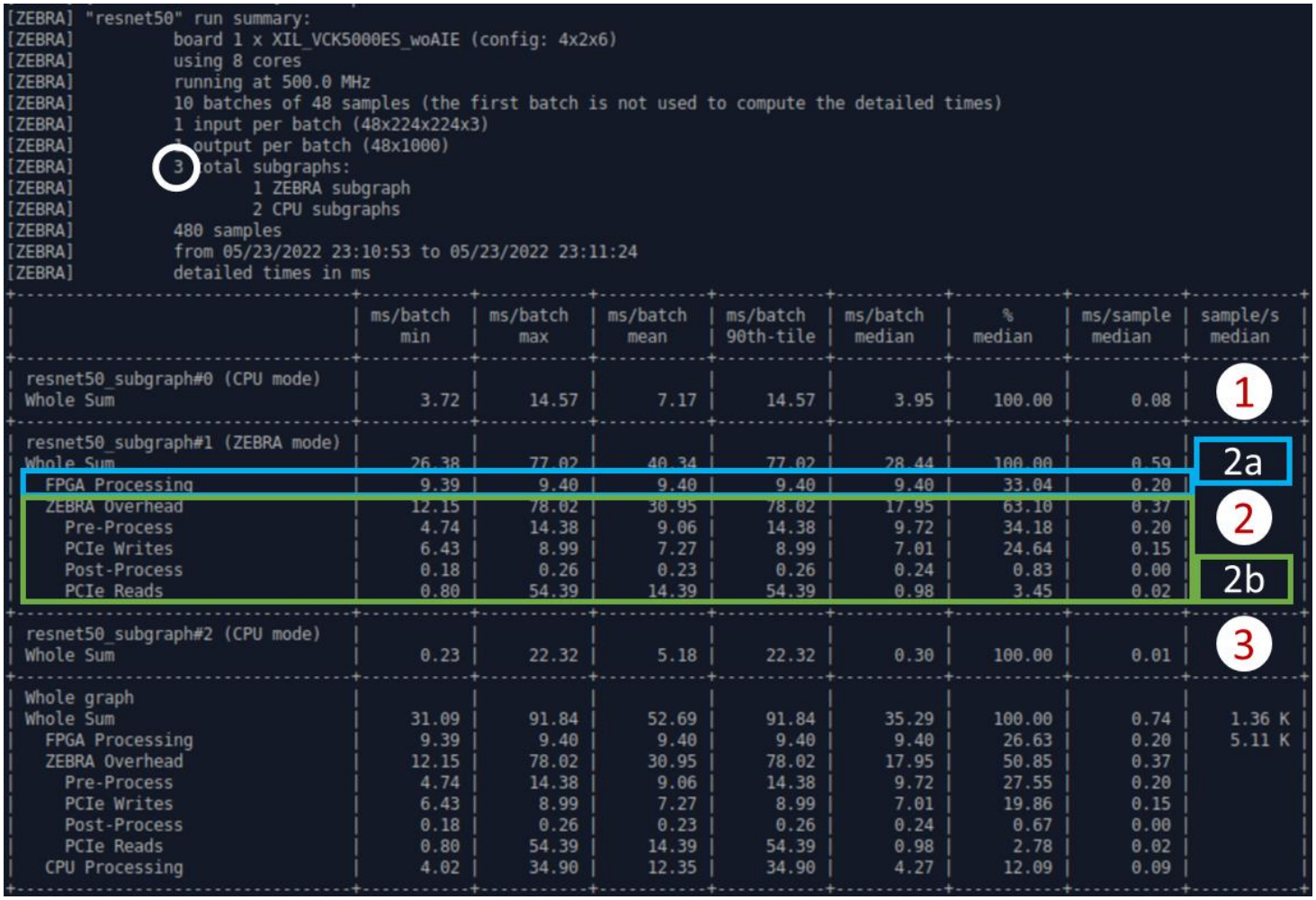
When analyzing performance from the summary table printed by Zebra, there are couple important points to remember:

- The main line of interest is the “FPGA” line – this is the CNN inference on FPGA
- Non-FPGA portion (i.e. Software running on CPU) is expected to be pipelined in future releases
    - Which will reduce the gap between ‘FPGA’ and ‘Whole Graph’ reported FPS

Each customer’s requirement will be unique. For questions or concerns, please reach out to Mipsology.

The two figures below show examples of performance summary table printed by Zebra on terminal at the end of the inference execution when statistics are enabled. Taking TF1 ResNet-50 as an example:





```
[ZEBRA] "resnet50" run summary:
[ZEBRA]          board 1 x XIL_VCK5000ES_woAIE (config: 4x2x6)
[ZEBRA]          using 8 cores
[ZEBRA]          running at 500.0 MHz
[ZEBRA]          10 batches of 48 samples (the first batch is not used to compute the detailed times)
[ZEBRA]          1 input per batch (48x224x224x3)
[ZEBRA]          1 output per batch (48x1000)
[ZEBRA]          3 total subgraphs:
[ZEBRA]                1 ZEBRA subgraph
[ZEBRA]                2 CPU subgraphs
[ZEBRA]          480 samples
[ZEBRA]          from 05/23/2022 23:10:53 to 05/23/2022 23:11:24
[ZEBRA]          detailed times in ms
```

	ms/batch min	ms/batch max	ms/batch mean	ms/batch 90th-tile	ms/batch median	% median	ms/sample median	sample/s median
resnet50_subgraph#0 (CPU mode)								
Whole Sum	3.72	14.57	7.17	14.57	3.95	100.00	0.08	
resnet50_subgraph#1 (ZEBRA mode)								
Whole Sum	26.38	77.02	40.34	77.02	28.44	100.00	0.59	
FPGA Processing	9.39	9.40	9.40	9.40	9.40	33.04	0.20	
ZEBRA Overhead	12.15	78.02	30.95	78.02	17.95	63.10	0.37	
Pre-Process	4.74	14.38	9.06	14.38	9.72	34.18	0.20	
PCIe Writes	6.43	8.99	7.27	8.99	7.01	24.64	0.15	
Post-Process	0.18	0.26	0.23	0.26	0.24	0.83	0.00	
PCIe Reads	0.80	54.39	14.39	54.39	0.98	3.45	0.02	
resnet50_subgraph#2 (CPU mode)								
Whole Sum	0.23	22.32	5.18	22.32	0.30	100.00	0.01	
Whole graph								
Whole Sum	31.09	91.84	52.69	91.84	35.29	100.00	0.74	1.36 K
FPGA Processing	9.39	9.40	9.40	9.40	9.40	26.63	0.20	5.11 K
ZEBRA Overhead	12.15	78.02	30.95	78.02	17.95	50.85	0.37	
Pre-Process	4.74	14.38	9.06	14.38	9.72	27.55	0.20	
PCIe Writes	6.43	8.99	7.27	8.99	7.01	19.86	0.15	
Post-Process	0.18	0.26	0.23	0.26	0.24	0.67	0.00	
PCIe Reads	0.80	54.39	14.39	54.39	0.98	2.78	0.02	
CPU Processing	4.02	34.90	12.35	34.90	4.27	12.09	0.09	

Total time = 1 + 2a + 2b + 3 { 52.69 ms/batch (mean) == 7.17+9.40+30.95+5.18 }

- 2a == ALL / Majority CNN layers accelerated on FPGA
- 1, 2b, 3 == processes executing on CPU
- Future Zebra release(s) will pipeline/parallelize 1, 2b and 3

The whole-Graph performance (e.g.: 1.36k) will be very close to FPGA (e.g. 5.11k) assuming application can provide a large enough batch size.

11 Accuracy

In this Alpha release, current Zebra delivers good accuracy. However, some CNNs may show larger than expected accuracy drop. It is recommended to compare Zebra accuracy with an inference execution done on CPU/GPU using the same trained mode, dataset, image pre-processing, etc. Zebra makes switching between FPGA and CPU/GPU execution very easy : 1-line Linux command to enable and disable Zebra.

In case the inference accuracy is observed to be lower than expected in a default run, Zebra provides SW switches to improve the accuracy (the same FPGA bitstream is used).

Examples of Zebra SW switches/APIs to improve accuracy:

- `quantization.mode=dynamic` (default = constrainedCalibrationV1.5)
- `quantization.forceSatCheckOnLastLayer=false` (default = true)
- `quantization.algorithmVersion=1.0` (default = 3.1)
- `quantization.ignoreNegativeValuesOnLastLayer=false` (default = true)
- `runOptimization.addOptimizers=PrecisionRecovery:RUN`

Example : `zebra_config --add quantization.mode=dynamic`

In our experience, accuracy is a topic that usually attracts intellectual discussions. Please check this FAQ question for more information on understanding accuracy and Zebra Quantization/Calibration.

Note that achieving desired accuracy for CNN Inference is a 1-time R&D effort. Once achieved, Zebra saves the results of quantization/calibration process in a file and always re-use for future execution. The quantization results file can be deployed on target inference servers.

Quantization/Calibration is executed again in event of a change in inputs that can influence the quality of results – for e.g. change in model weights.

Please email Mipsology for any questions or concerns or unique requirements for Your CNN.

12 Release Limitations

12.1 Layers

In this alpha release:

- Zebra does not support the "`element-wise mul`" layer. Trying to execute a model with this layer (e.g. `efficientdet`, `efficientnet`, `mobilenet_v3`) may freeze the FPGA which will require cold reboot.
- Zebra expects the CNN graph/model should include only layers supported by ONNX. Expect Zebra to fail when the model includes layers not supported by ONNX.
    - For e.g. FB Detectron2 does not work because the graph includes custom layers.
    - Zebra may be able to support these graphs 'easily' on case-by-case basis. Please reach out to Mipsology for this type of requirement.

12.2 All Frameworks

In this alpha release

- Some open-source models may experience an error during conversion to ONNX.

- In such cases, the error message will mention which `opset_version` to use.
- Please use Zebra's SW API to force this setting and re-run the application. For e.g.:

```
zebra_config --add debug.opset_version=<num_in_error_message>

<run_your_application_again>
```

12.3 PyTorch

- This alpha release does not support the explicit "`forward`" inference API. For example:
  - The following code will FAIL with Zebra error
    `output = model.forward(input)`
  - The following code will PASS
    `output = model(input)`

12.4 TensorFlow 1 & 2

- Some demos may experience Zebra error related to automatic graph splitting.
  - We are still working on covering all ways in which TF developers train models and generate graphs.
- The solution is to use Zebra Software API to 'manually' split the graph (a.k.a. 'Legacy' mode)
- Details about graph splitting provided in a dedicated section of this document.

Expectations for Future Release

In upcoming releases, Zebra is expected to improve automatic graph splitting for all frameworks and support models/graphs with custom layers (i.e. layers not supported by ONNX).

13 Other Details

13.1 Zebra Legacy Mode

Some demos in this release uses 'manual' graph splitting achieved with the Zebra SW API. For this to work, user needs to enable Zebra in 'Legacy' mode. Below is one example.
Inside Mipsology Docker container:

```
cd
source zebra/unset.sh
source zebra/settings.sh legacy
```

(Enables 'legacy' mode. This is different than default instructions)

```
cd ildoonet-tf-post-estimation
zebra_config --add runSession.enableTimeStatistics
./run /VIDEO/kulam_dance_27sec.mp4
```

13.2 Graph Splitting

In this alpha release

- PyTorch based models/graph are automatically managed by Zebra
    - Decision to run some layers on CPU vs. accelerating on FPGA as well as the associated data managemnet is transparent to user.
- TensorFlow based models/graphs
    - Some applications work automatically out-of-box. For e.g. YOLO-v4 docker demo.
    - For some models/graphs, Zebra needs user's help. For e.g. Pose-Estimation docker demo.

This is the first release with Automatic graph splitting. In some cases (depending on the CNN), the splitting creates many sub-graphs which may not be optimal. In future release(s), Zebra is expected to improve the splitting feature for all supported CNNs.

When performing 'manual' graph splitting, Zebra needs to be enabled in Legacy mode.

Manual graph splitting is an advanced concept that assumes user is well aware of CNNs and their analysis using framework tools. To use this explicit API, user needs to identify appropriate layers for FPGA (or CPU) execution and instruct Zebra using software API/command. Zebra will then split the graph as per user directive. Note that the execution and data management is performed automatically by Zebra without user intervention.

Example of the SW API/command:

```
zebra_config --add runSession.subGraphs=<MODE>:<startLayer>:<endLayer>
```

where:

- <MODE> == FPGA or CPU
- <startLayer> == Name of 1st layer in the model to be executed on MODE
- <endLayer> == Name of last layer in the model to be executed on MODE

If there are multiple subgraphs to be declared, all the subgraphs must be declared in the same command. The description of subgraphs must be separated by a "**|**" (pipe). For example, declaring two subgraphs would look like:

```
subGraphs=<MODE1>:<startLayer1a,startLayer1b>:<endLayer1a,endLayer1b>|<MODE2>
:<startLayer2>:<endLayer2>
```

Identification of layer names can be done using Framework APIs or using graphical tools like Netron.

Please email Mipsology for any questions around 'manual' graph splitting and use of Legacy mode.

14 FAQ

Can I control FPGA operating frequency?

On this alpha release for VCK5000, user cannot control FPGA operating frequency. We expect to enable this feature in next release.

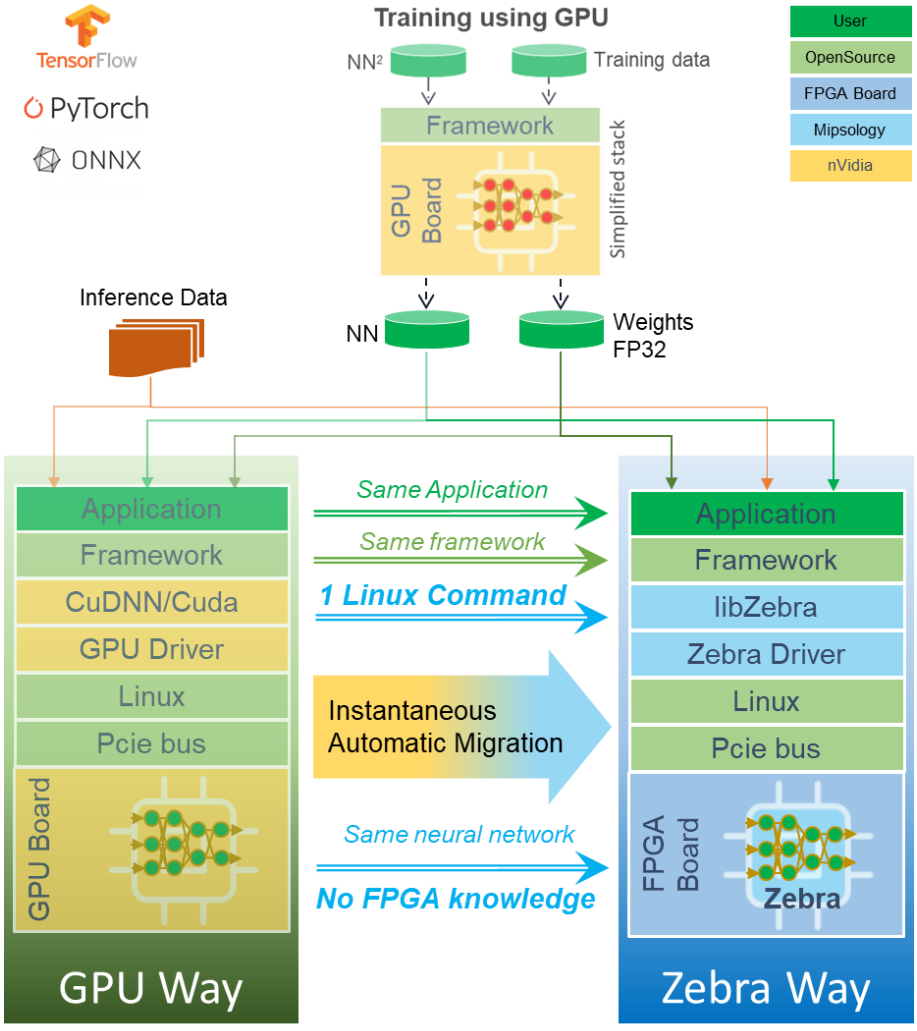
Why do I get CUDA related messages when running some demos?

Depending on the demo, some CUDA related messages may be printed on terminal by the ML Framework. This should not result in any error during execution. This is not related to Zebra. If absolutely needed, these messages can be suppressed by compiling the framework from source.

Does Mipsology provide a Model-Zoo?

Mipsology does not provide a model-zoo. This is because Zebra software is designed to accelerate neural networks (CNNs) trained on GPU without modification. In other words, Zebra accelerates post-training CNN graph as-is without any structural change. User does not need to prune the model. There are no offline tools to use before running Zebra.

Zebra works inside User's ML Framework and in-line with User's Application. Figure here shows simplified Zebra software stack. Please reach out to Mipsology for further questions or to discuss project's unique requirements.



Mipsology

Can You give more information about Accuracy and Zebra Quantization?

High Accuracy for CNN Inference is important for production deployment. Inference accuracy depends on many factors like model training, dataset used, image pre-processing, etc. Based on our experience, it is not a correct practice to compare Zebra result with theoretical accuracy found in an article or on internet. Best practice is to compare results of 2 executions – one with CPU/GPU and one with Zebra on FPGA – using exact same weights/parameters, input data, pre-processing and application software.

Zebra does not need any offline tool for quantization. The process of FP32 to INT8 conversion happens in-line with user's application. From User's point of view, they run the inference application just as they would normally run on CPU/GPU.

Zebra makes switching between FPGA and CPU/GPU very easy – 1-line Linux command : `source settings.sh".`

Zebra aims to provide optimal accuracy by default. In case the accuracy is still observed to be lower than expected, Zebra provides SW switches to improve the accuracy (NOTE: the same FPGA bitstream is used). Some examples of this are shown in Accuracy section.

Most of the software options are influencing the calibration and quantization algorithms, and don't impact performance. The reason different algorithms may be required is that Zebra does not use the training data or expected results to map a model, which sometimes can influence the quality of the results. Typically, the options found for a given model will be reusable if the model goes through various training.

It is also a good practice that the first batch of images, which is used by Zebra for quantization/calibration, are diversified and of good quality. For example:

- Images should cover a good spread of target classes (classification) and objects (detection)
- Not all images should be very dark or very bright
- Not all images expected to give wrong result
- Not all images should be known outliers
- Not all images with extreme size (e.g. largely enlarged)

Users well versed with CNN model and intent of the application typically understand these requirements.

Note that achieving desired accuracy for CNN Inference is a 1-time R&D effort. Once achieved, Zebra saves the results of quantization/calibration process in a file and always re-use for future execution. The quantization results file can be deployed on target inference servers.

Quantization/Calibration is executed again in event of a change in inputs that can influence the quality of results – for e.g. change in model weights.

Please email Mipsology for any questions or concerns or unique requirements for Your CNN.

How do I contact Mipsology for support or questions?

Please email support@mipsology.com with any questions or concerns or to discuss your unique requirements.

Mipsology

Legal Notice

The information disclosed to you (the “User”) hereunder is provided solely for the selection and use of Mipsology products. The information, the applications, and the tools (together referred as the “Materials”) are made available “AS IS” and with all mistakes, errors, inconsistencies, or defects, without warranty of any kind. To the maximum extent permitted by applicable law:

(1) The Materials are provided "as is" and "as available", without warranty of any kind. Mipsology, its affiliates, its officers, its employees, or its suppliers and representatives, do not warrant in any way that the Materials is error free or satisfy licensee's specific requirements and disclaim any and all warranties of any kind or nature, whether express, implied, or statutory, relating to or arising with respect to the Materials, including but not limited to implied warranties of merchantability, warranty of fitness for a particular purpose, title, and noninfringement. Mipsology makes no warranty concerning the data, results or information resulting in using the Materials.

(2) To the maximum extent permitted by applicable law, in no event shall Mipsology, its affiliates, its officers, its employees, or its suppliers and representatives be liable for any special, exemplary, consequential, incidental, punitive, direct or indirect damages whatsoever including, but not limited, to loss of business profit, loss of use, loss of data, business interruption, loss of revenue, loss of orders, loss of business or profits, anticipated savings, loss of information and data, damage to brand image, or any other financial loss arising out of or in connection with the use of the Materials or the operation of the application or any other product or services, even if advised beforehand of the possibility of such damages. In no event will Mipsology total liability under or arising out of this agreement exceed the actual received payment from User, directly or through the cloud, in the last billing period or the duration of the incident, whichever is the lowest amount, reduced by any other amount Mipsology would have paid back to User. To the extent that the applicable jurisdiction limits licensee's ability to disclaim any implied warranties, this disclaimer shall be effective to the maximum extent permitted. Without limiting the foregoing, the User is responsible for determining and verifying that the Materials, its environment, and the hardware used to run the application are compatible. Mipsology further decline any warranties of any kind or nature on the hardware used in conjunction with the Materials. Mipsology shall not be liable to User nor any third parties (whether arising in contract, tort (including negligence), breach of statutory duty or otherwise) for failure of fitness or any of its or a third party's systems that results in the inability to process or use the Material, User's failure to meet any of its payment obligations, negligence, fraud or fraudulent misrepresentation of User or any other actions which result from misuse or inappropriate use of the Materials.

Without prior written agreement, User will not knowingly, or allow others, including internally, to copy, reproduce, modify, obliterate, distribute, or publicly display the Materials in any form, partially or fully, whatsoever except for the normal usage of the Materials.

Mipsology assumes no obligation to correct any errors contained in the Materials or to notify User of updates to the Materials. This document is subject to change without notice.

Please refer to Mipsology’s End User License Agreement (EULA.txt) and other legal notices available in the ‘doc’ directory of the provided release.

Copyright