Using AI hardware accelerators for real-time DSP on embedded devices

NPU, TPU, … make them run SDR instead of AI !

Sylvain Azarian – F4GKR

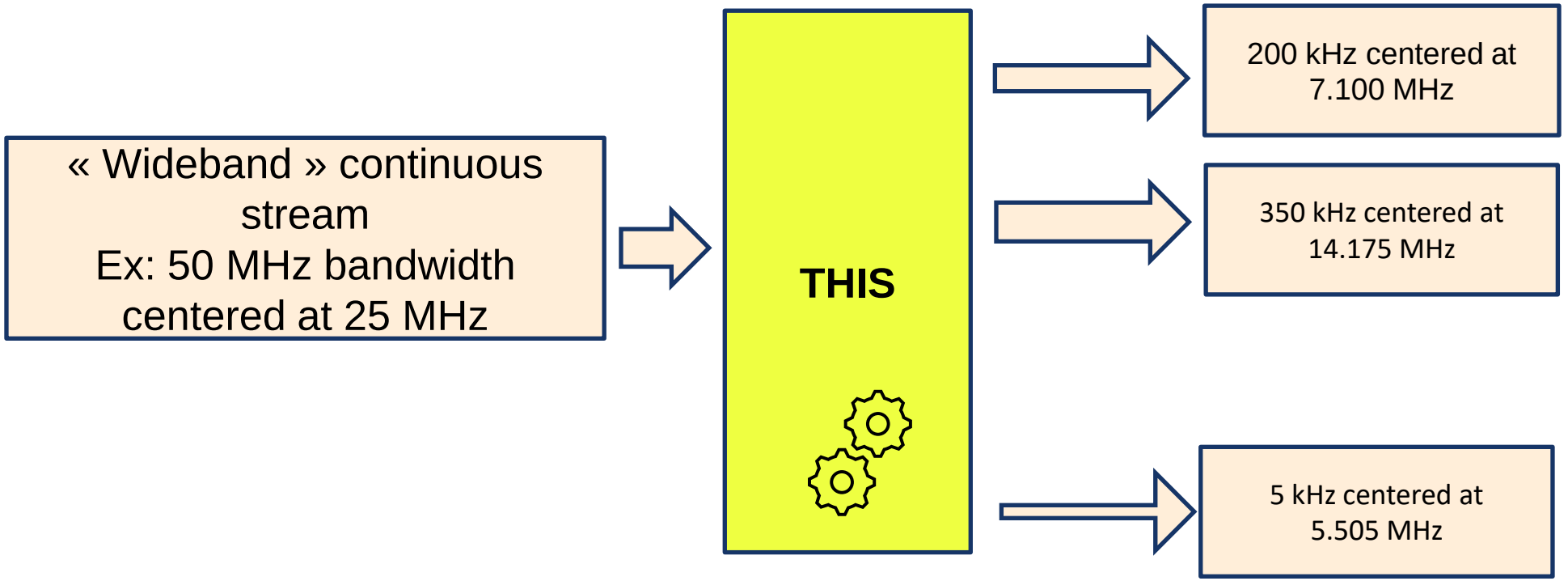
Intro & Outline

- **Author** : Sylvain Azarian – F4GKR
  - Founder of « SDR-Technologies » , small French company around Paris
  - Involved in Amateur Radio (President of IARU R1)

- **Outline of the talk**
  - Motivation and starting point
  - The paths to paradise are strewn with pitfalls
  - Status and what’s next
  - Q&A

In the last episode (FOSDEM24…)

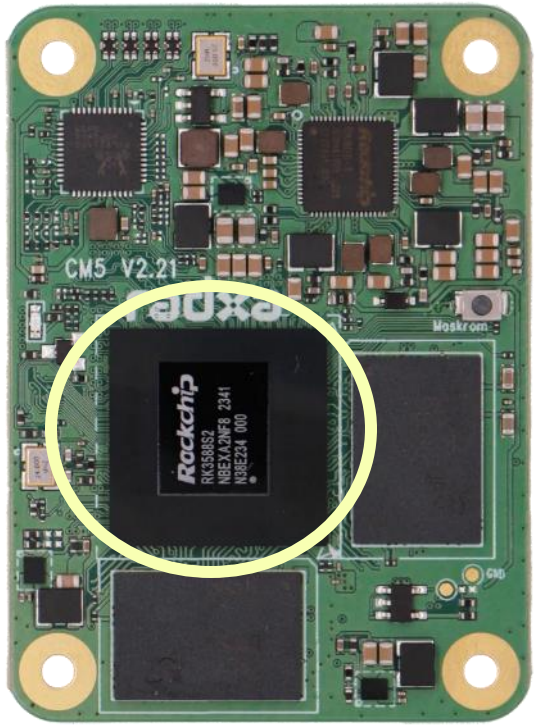
- I presented the project « libGKR4GPU », a multi “Digital Downconverter” C++ library implemented in CUDA, working (only) on NVIDIA GPU
- It provides multiple sub bands from one single input, with different specifications (bandwidth, oversampling, …)



Motivation

- Replace the « NVIDIA thing » by something much cheaper… either embedded in the CPU or as an optional module

RV1109
$35

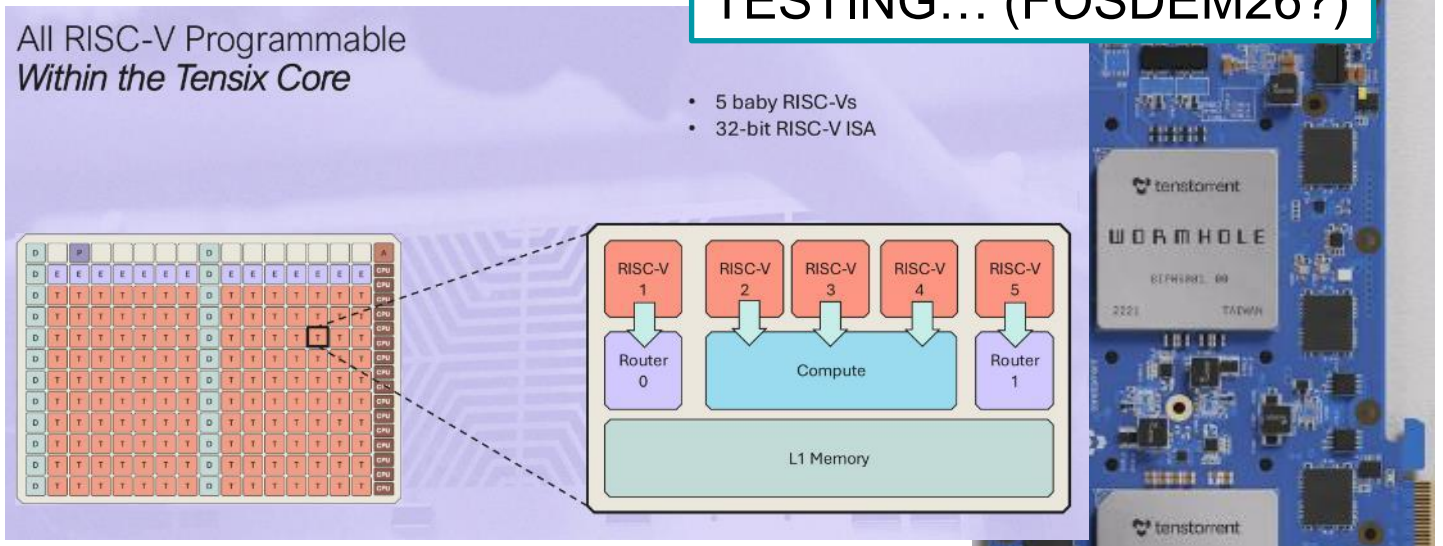


RK3588
54€

Google CORAL
~ 30€

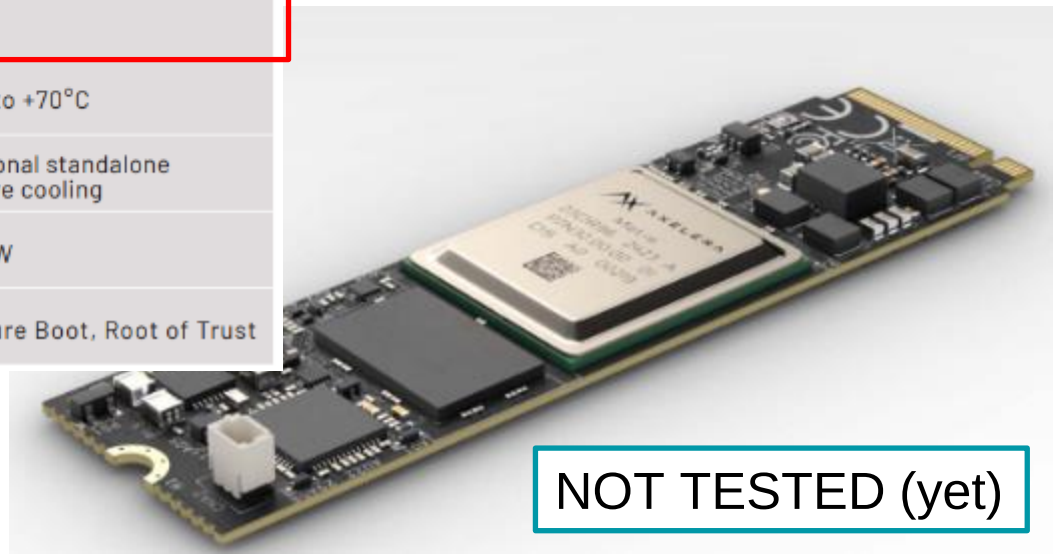
Also in mind…

TESTING… (FOSDEM26?)



KEY TECHNICAL SPECIFICATIONS

Form Factor	M.2 2280 M-key
Host Interface	PCIe Gen3 x4 – 4 GB/s bidirectional
AIPU (AI Processing Unit)	1x Metis AIPU
AIPU Memory	1 GB DRAM
Peak INT8 TOPS	214
Operating temperature	-20 to +70°C
Thermal solution	Optional standalone active cooling
Typical Application Power	4-8 W
Security Features	Secure Boot, Root of Trust

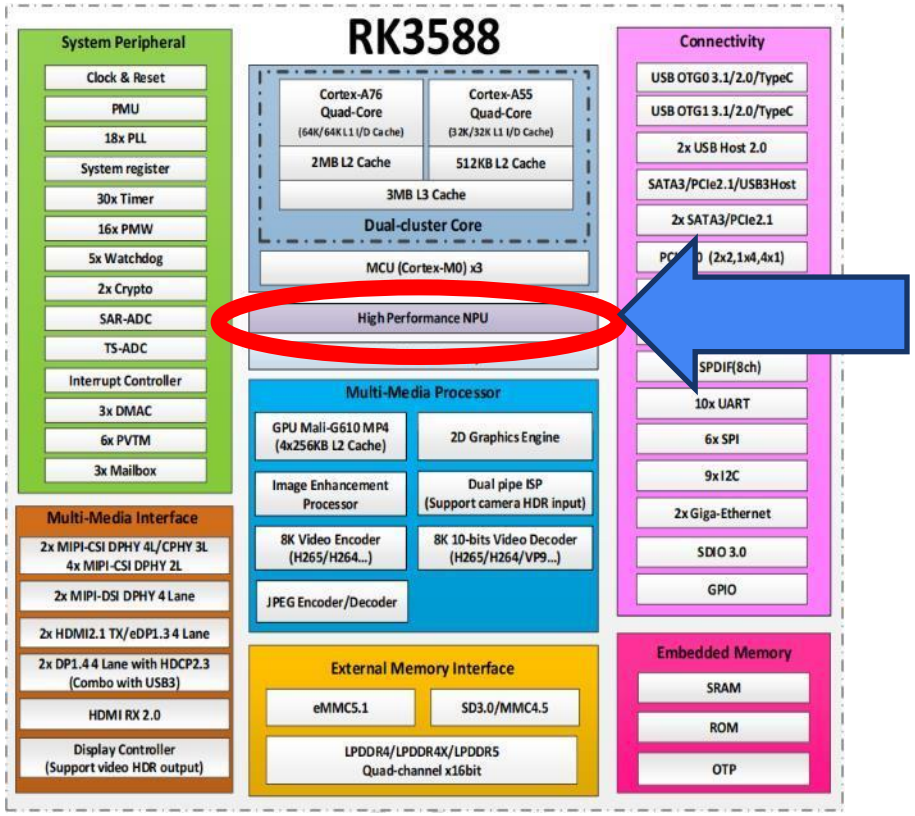


Axelera

NOT TESTED (yet)

Tenstorrent

The NPU promise



NPU (Neural Process Unit) :

- Neural network acceleration engine with processing performance up to **6 TOPS**
- Include triple NPU core, and support **triple core co-work, dual core co-Work, and work independently.**
- Embedded **384KBx3** internal buffer. Multi-task, multi-scenario in parallel.
- Support deep learning frameworks: TensorFlow, Caffe, Tflite, Pytorch, Onnx NN, Android NN, etc.

The NPU promise

Model support

In addition to exporting the model from the corresponding respository, the models file are available on https://console.zbox.filez.com/l/8ufwtG (key: rknn).

Category	Name	Dtype	Model Download Link	Support platform
Classification	mobilenet	FP16/INT8	mobilenetv2-12.onnx	RK3566|RK3568|RK3588|RK3 RV1103|RV1106 RK1808|RK3399PRO RV1109|RV1126
Classification	resnet	FP16/INT8	resnet50-v2-7.onnx	RK3566|RI RK1808|RI RV1109|R'
Object Detection	yolov5	FP16/INT8	./yolov5s_relu.onnx ./yolov5n.onnx ./yolov5s.onnx ./yolov5m.onnx	RK3566|RI RV1103|R' RK1808|RI RV1109|R'

WARNING !

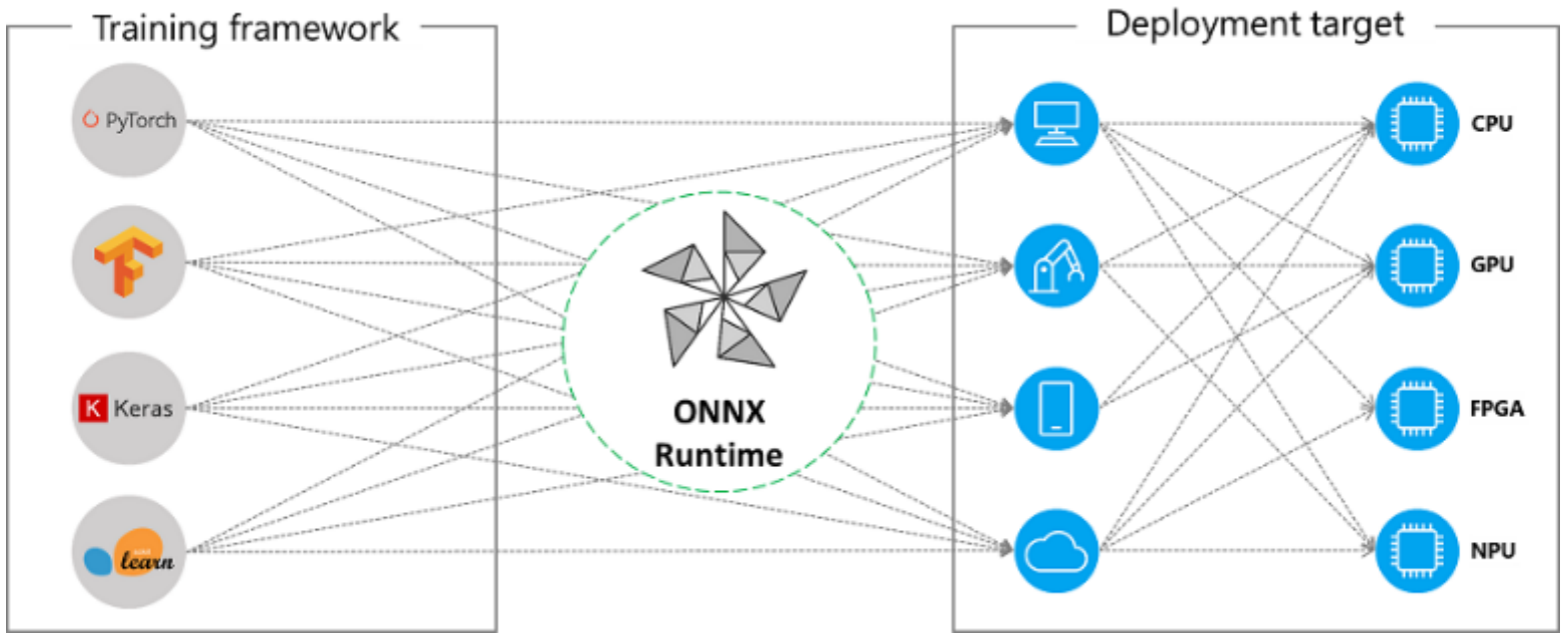
FP16 … INT8…

Model performance benchmark(FPS)

demo	model_name	inputs_shape	dtype	RK3566 RK3568	RK3562	RK3588 @single_core
mobilenet	mobilenetv2-12	[1, 3, 224, 224]	INT8	180.7	281.3	450.7
resnet	resnet50-v2-7	[1, 3, 224, 224]	INT8	37.9	54.9	110.1
yolov5	yolov5s_relu	[1, 3, 640, 640]	INT8	25.5	33.2	66.1

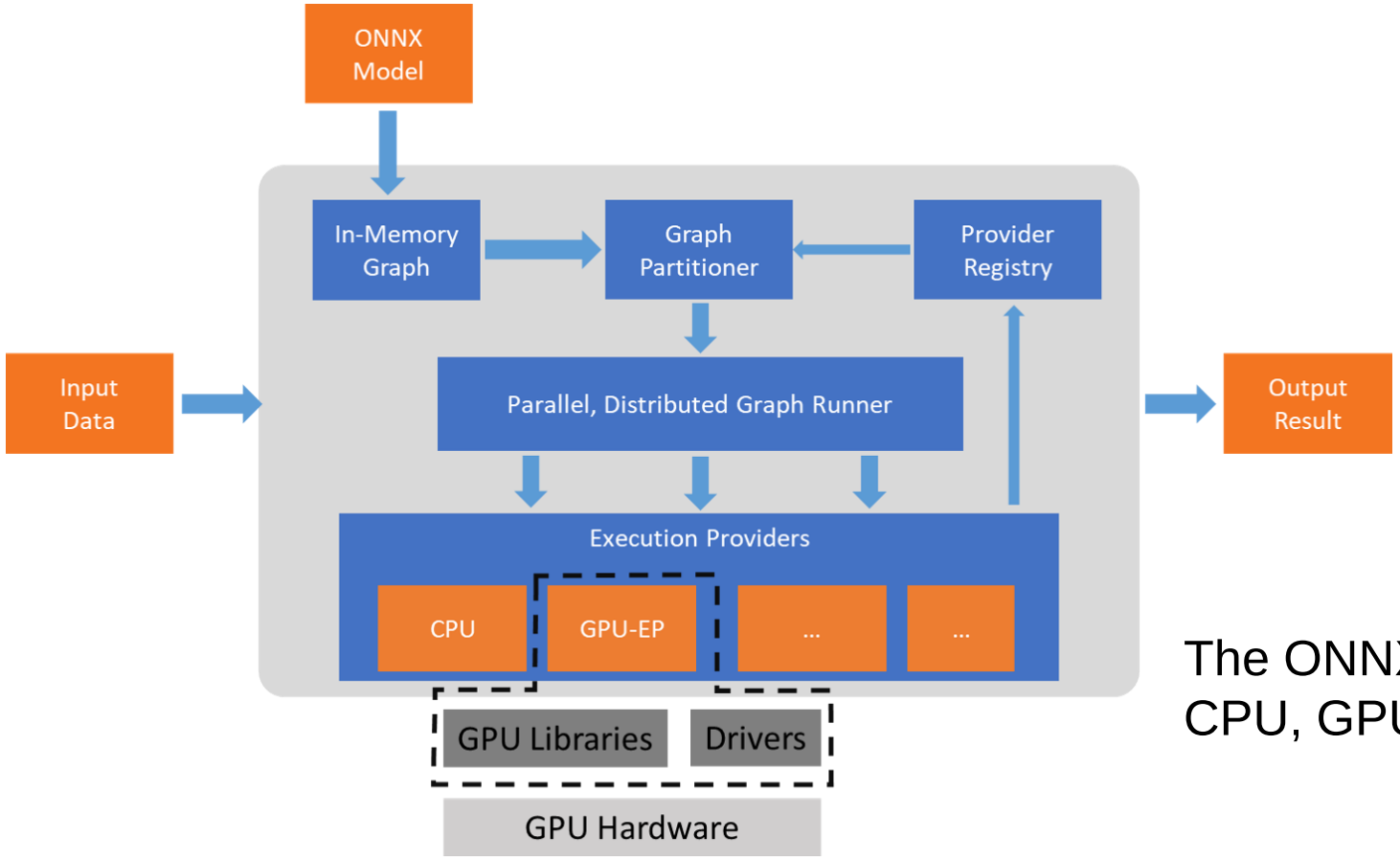
What is ONNX ???

Open Neural Network Exchange (ONNX)



ONNX is an intermediary machine learning framework used to convert between different machine learning frameworks.

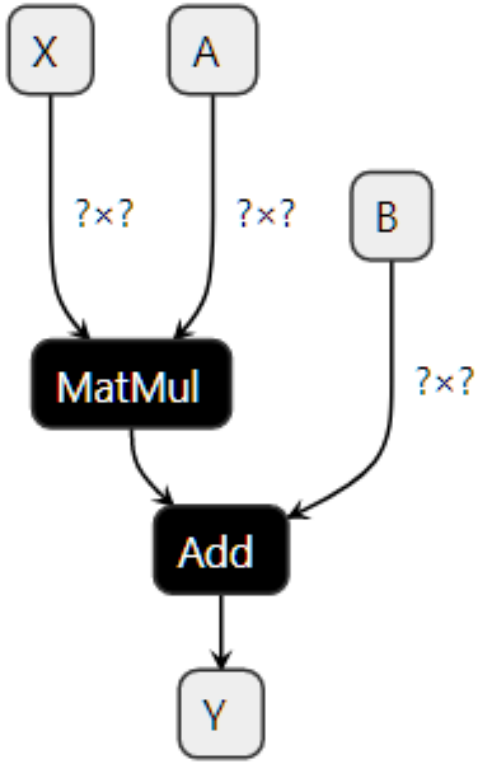
ONNX Runtime (“Execution Provider”)



The ONNX model can be run in CPU, GPU or dedicated chip

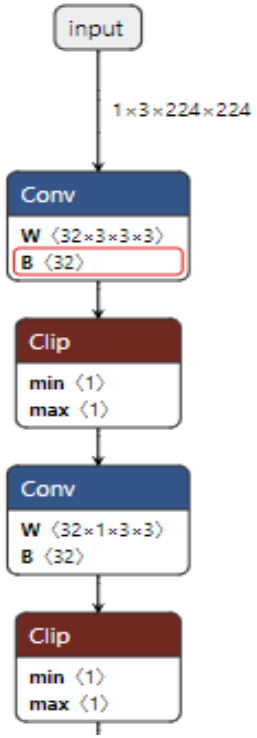
Introduction to ONNX

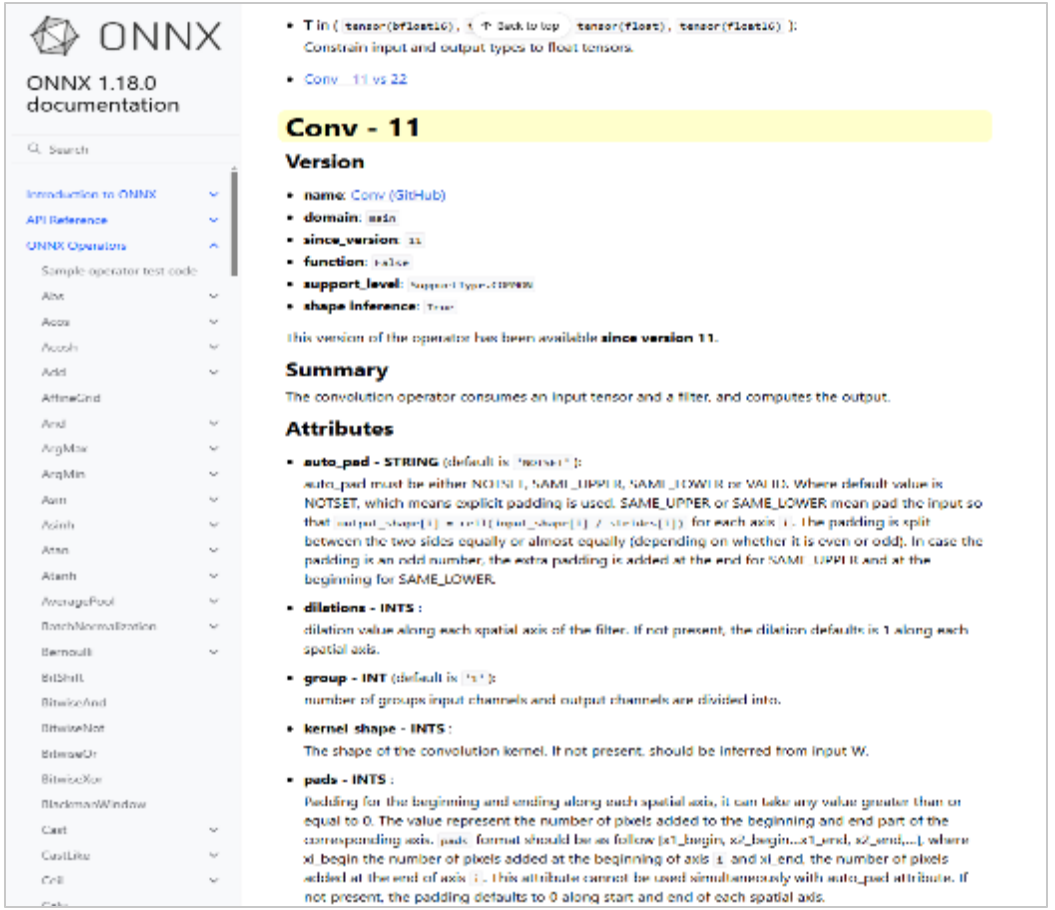
- Processing is described by a graph: the execution flow
- Multiple inputs and outputs possible
- Inputs and outputs are “tensors”
- Each node of the graph preforms one operation



$$[Y] = [X] * [A] + [B]$$

Introduction to ONNX





Introduction to ONNX

Designed for AI:

- most of the operators are the common functions used in Deep Learning
- Embedded chips are mainly targeting real-time image processing and are optimized for inputs=images, outputs=detections of objects

What could we have ?

- Some « SDR » blocks able to run either on dedicated hardware or by main CPU (emulation) like recently proposed for Audacity
- Possibility to use low-cost Linux platforms with decent performance
- Have a flowgraph tool « GnuRadio Companion like » to create DSP chains ?

Hold my beer ?

- We discuss ONNX models, but in ROCKCHIP the proposed API is **RKNN**, supporting only a **subset** of instructions
- Google CORAL uses “Tensor Flow Light” models
- My understanding : RKNN is ~ ONNX version 11…

RKNNToolkit2 OPs Support

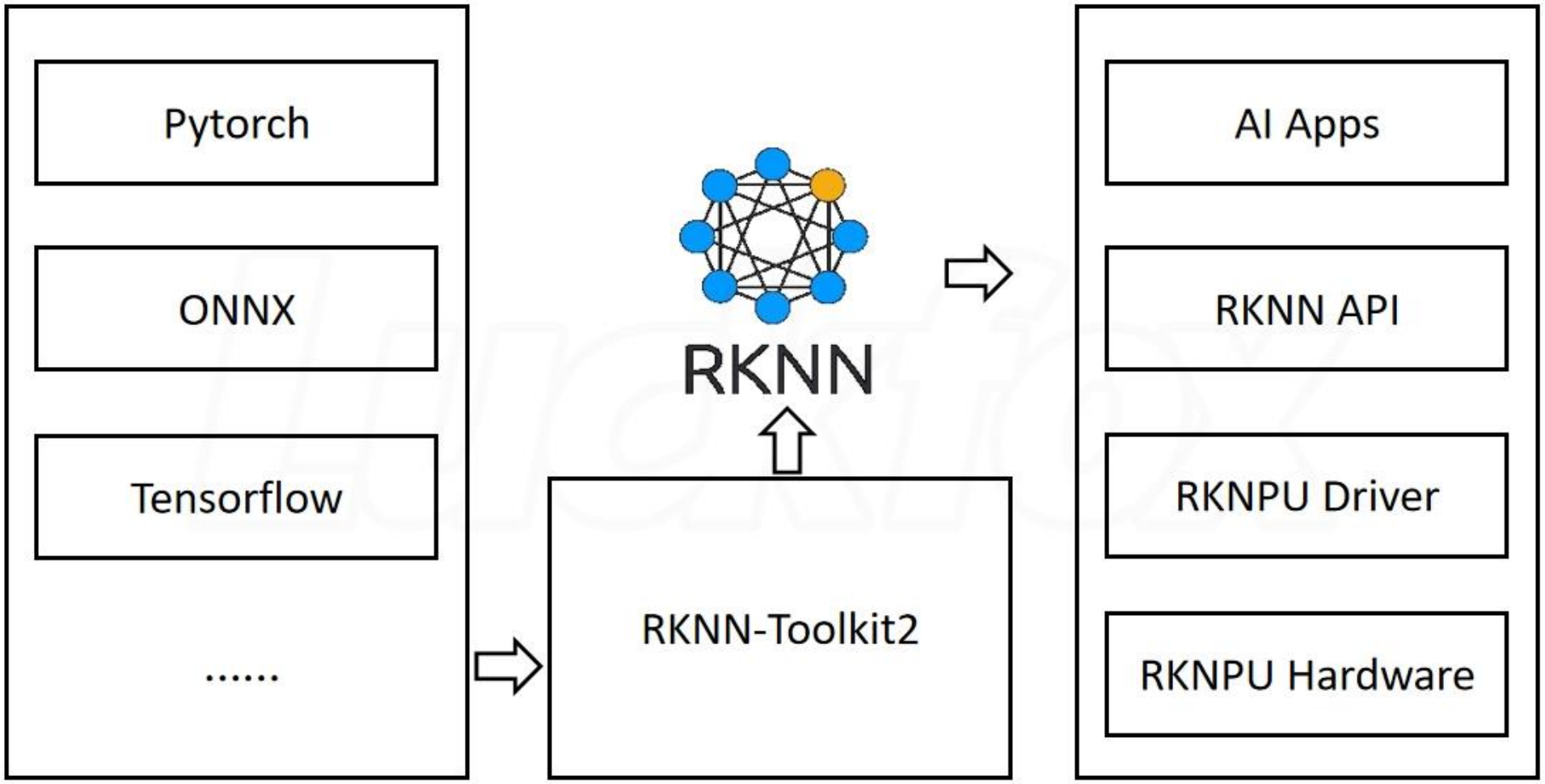
ONNX OPs supported by RKNN Toolkit2

According to ONNX official instructions, the corresponding ONNX opset version is 19. follows:
(For more restrictions, please refer to <RKNN_Compiler_Support_Operator_List>)

Operators	Remarks
Abs	Not Supported
Acos	Not Supported
Acosh	Not Supported
Add	
And	
ArgMax	
ArgMin	
Asin	Not Supported
Asinh	Not Supported
Atan	Not Supported
Atanh	Not Supported

https://github.com/airockchip/rknn-toolkit2/blob/master/doc/RKNNToolKit2_OP_Support-2.3.0.md

RKNN


Pytorch
ONNX
Tensorflow
.....
RKNN
RKNN-Toolkit2
AI Apps
RKNN API
RKNPU Driver
RKNPU Hardware

What do we need ?

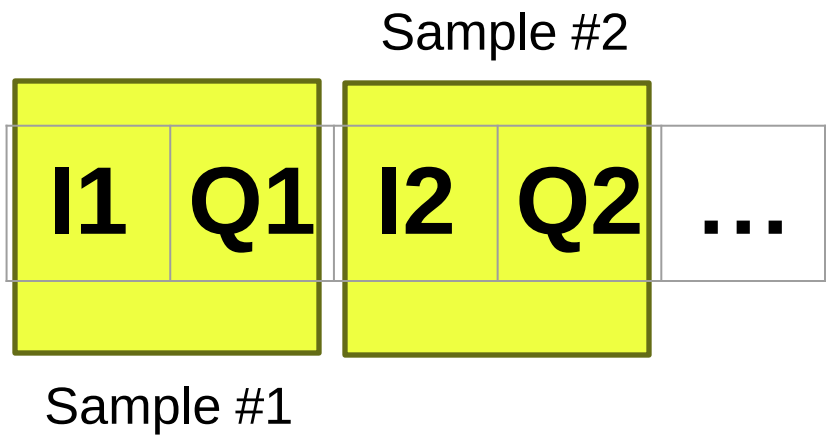
- **Complex** number arithmetic (multiplication…)
- **Convolution** (for filters)
- **Trigonometric** functions (cos() and sin() to generate our local oscillators)

Do we have this ? No but it is doable*!

* (done in ONNX so far…)

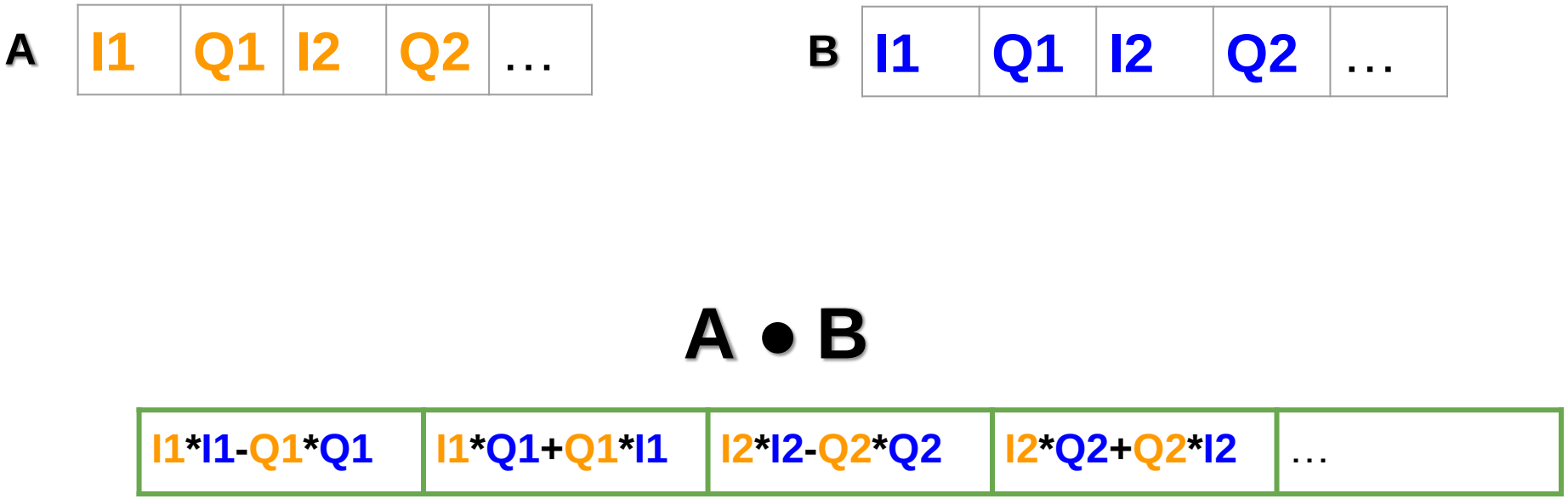
Complex numbers

- We need two numbers: the *real* part and the *imaginary* part
- The optimal approach here is to keep the interleaved approach

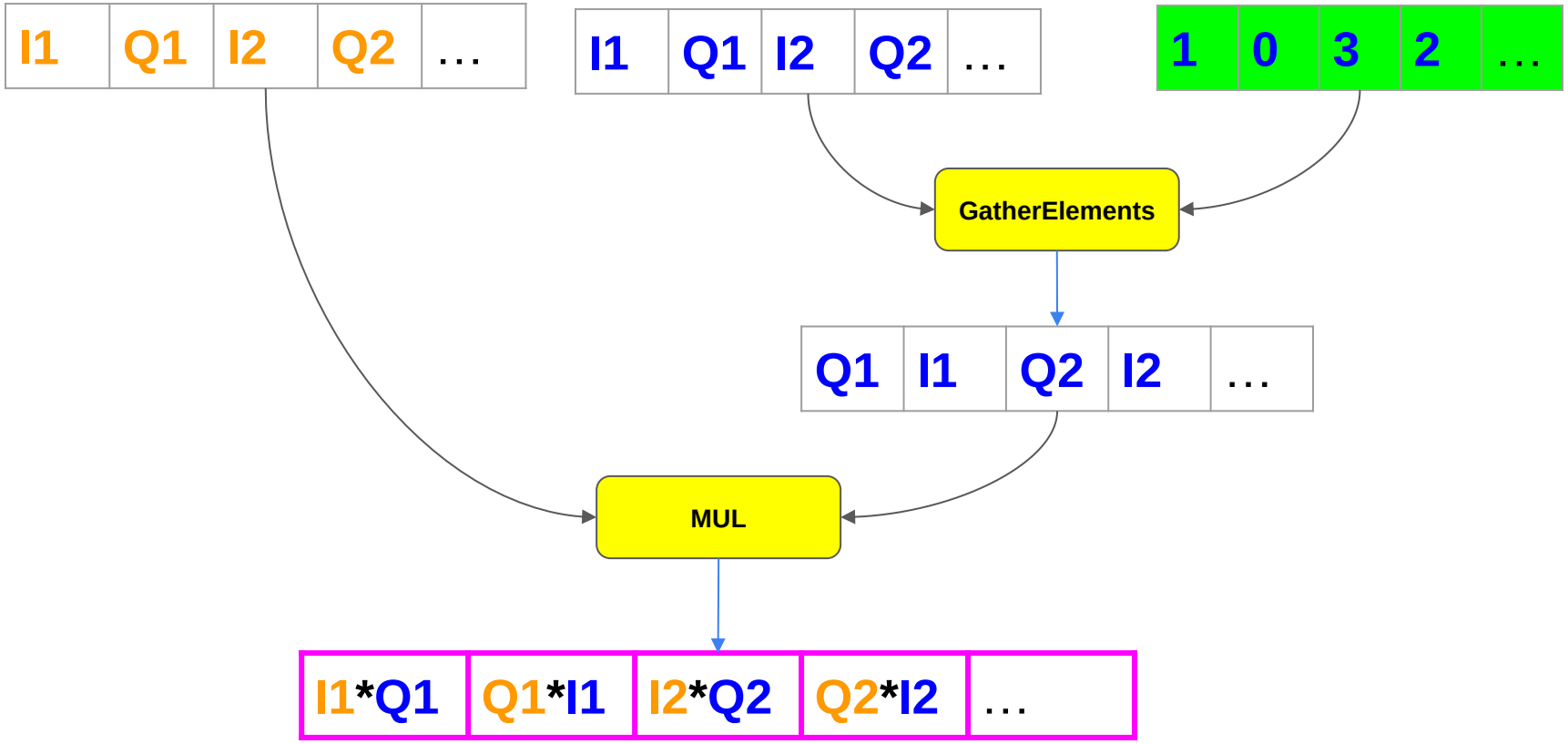


Complex (numbers) multiplications

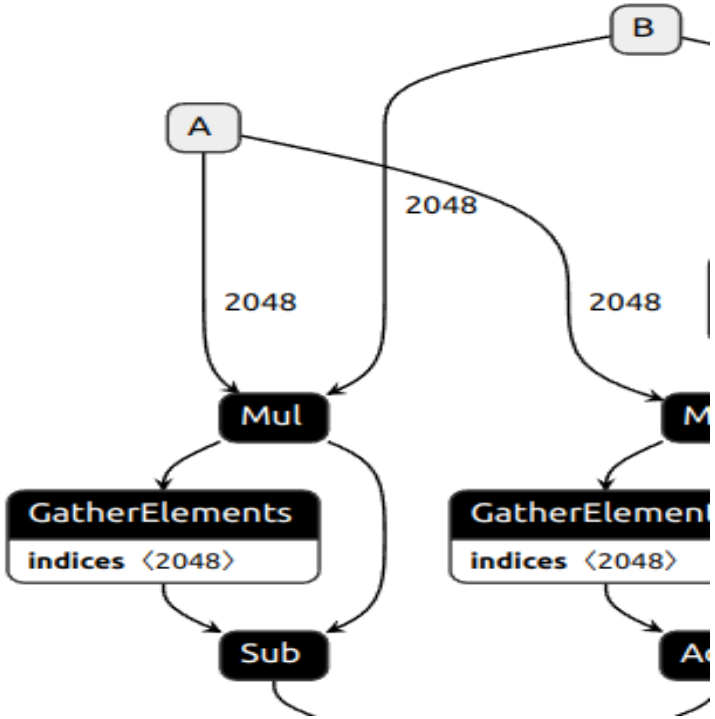
- We have two input vectors of numbers; we want the pointwise multiplication of the two



The ONNX Fun of complex multiplication



Implementation

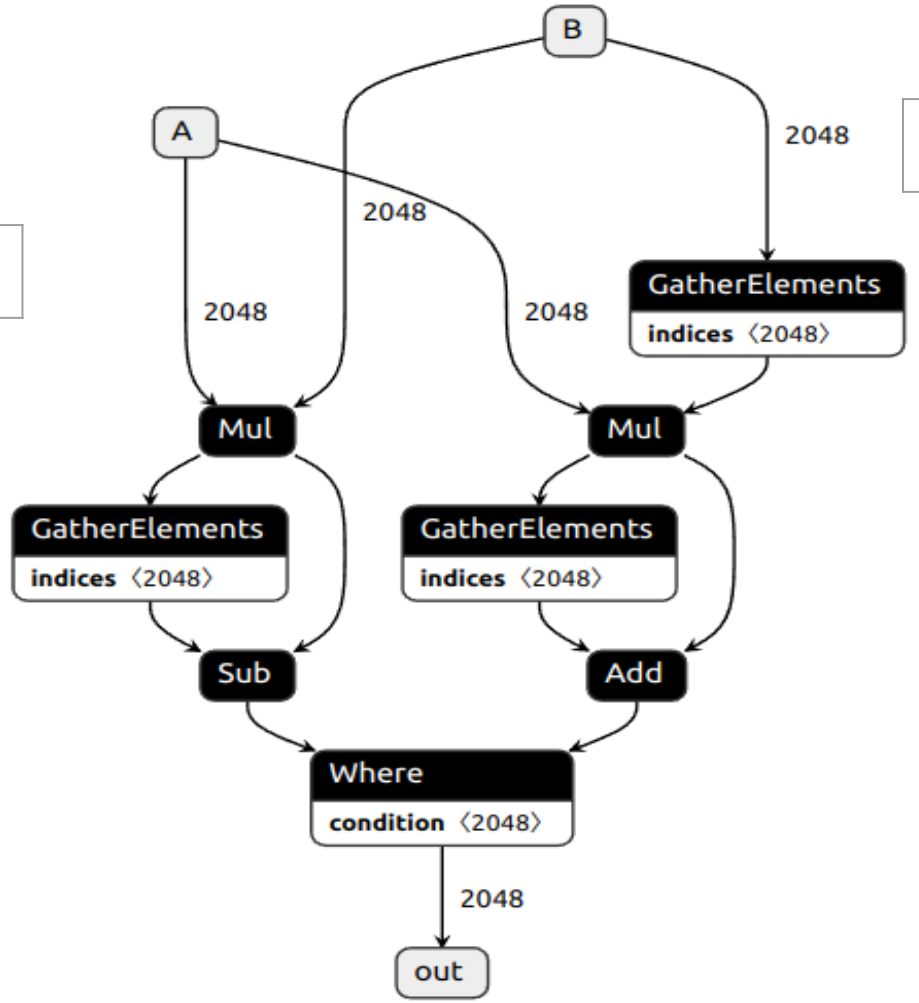


A Python function generates the ONNX graph

```
#-------------------------------------------------------
m1 = make_node('Mul',
               inputs=['A','B'],
               outputs= ['AB'])

s1 = make_node('GatherElements',
                 inputs=['AB', 'selector'],
                 outputs= ['sAB'],
                 axis=0 )

realparts = make_node('Sub',
                 inputs=['AB','sAB'],
                 outputs= ['reals'])
```

The complex multiplication

I1	Q1	I2	Q2	…



I1	Q1	I2	Q2	…

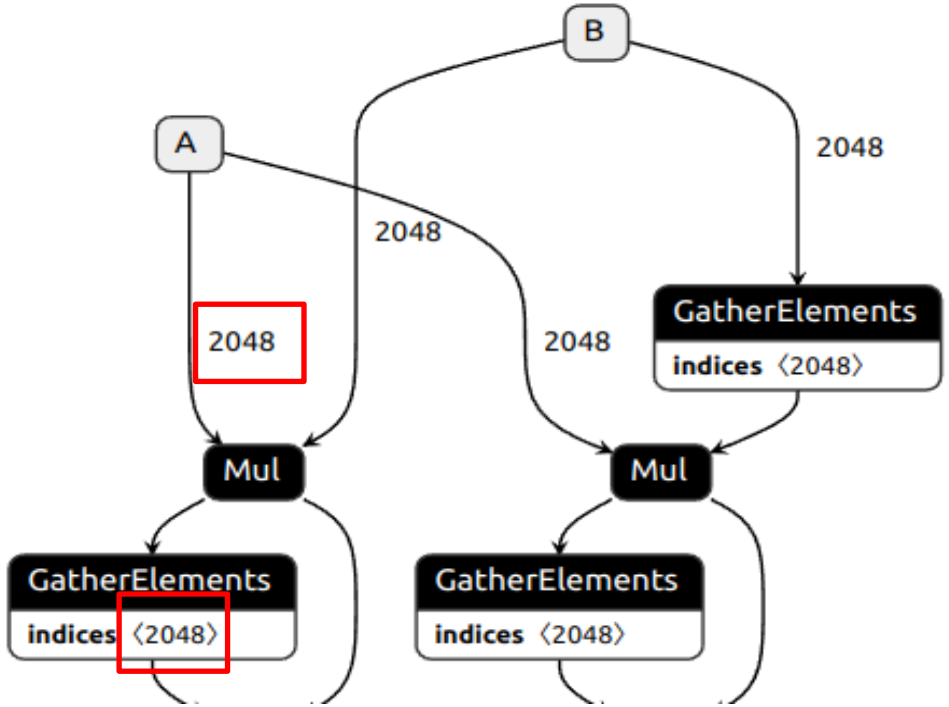
Size matters…

- The input “tensor” (the data) has a fixed size, and this size is IN the ONNX file…
- My Python code generates the ONNX files for a given size…

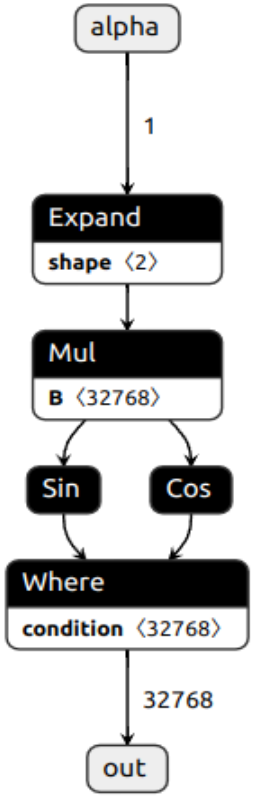
f4gkr For FOSDEM25

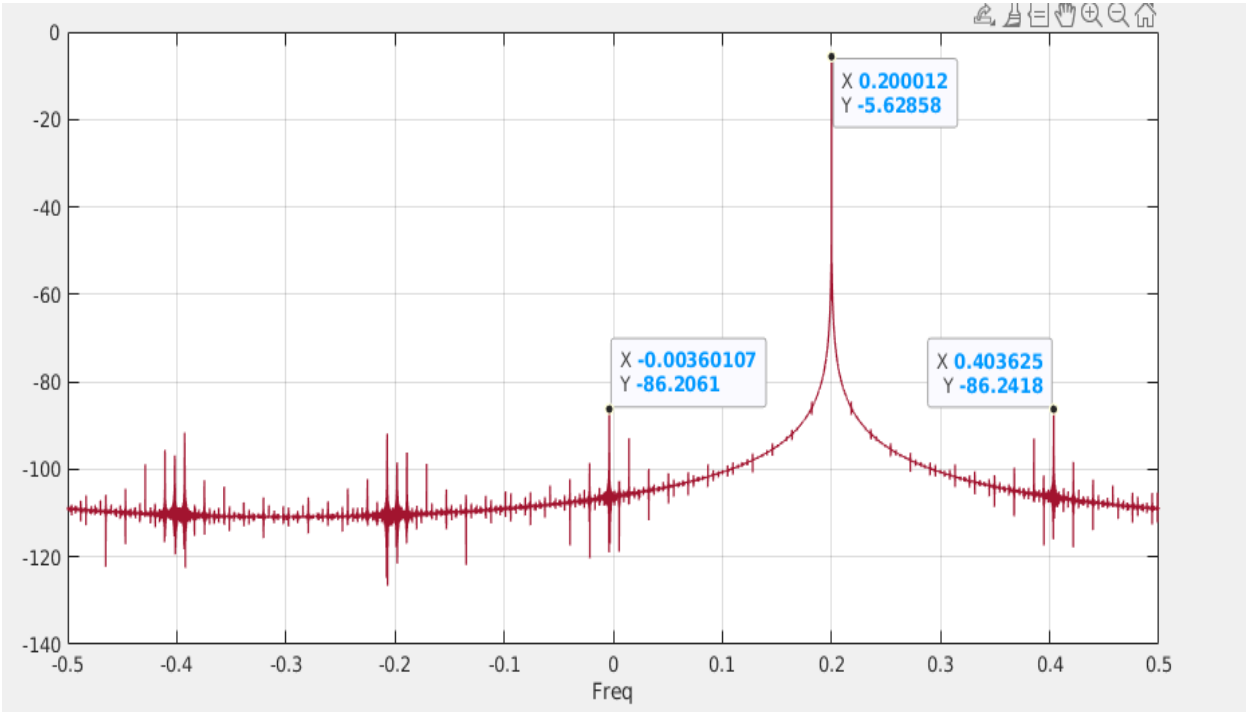
Code Blame 94 lines (73 loc) · 2.82 KB

```python
from onnx import numpy_helper,TensorProto
from onnx.helper import (
    make_model, make_node, make_graph,
    make_tensor_value_info)
from onnx.checker import check_model
from onnx.reference import ReferenceEvaluator
from onnx import version_converter
import numpy

cpxcount  = 1024
floatcount = cpxcount * 2

# generate sequence (0,1,3,2,5,4,....)
vSelector = numpy.zeros( floatcount, dtype=numpy.int32)
vSelector[0] = 1
vSelector[1] = 0
for x in range(2,floatcount):
    vSelector[x] = vSelector[x-2]+2
```



IQ Oscillator



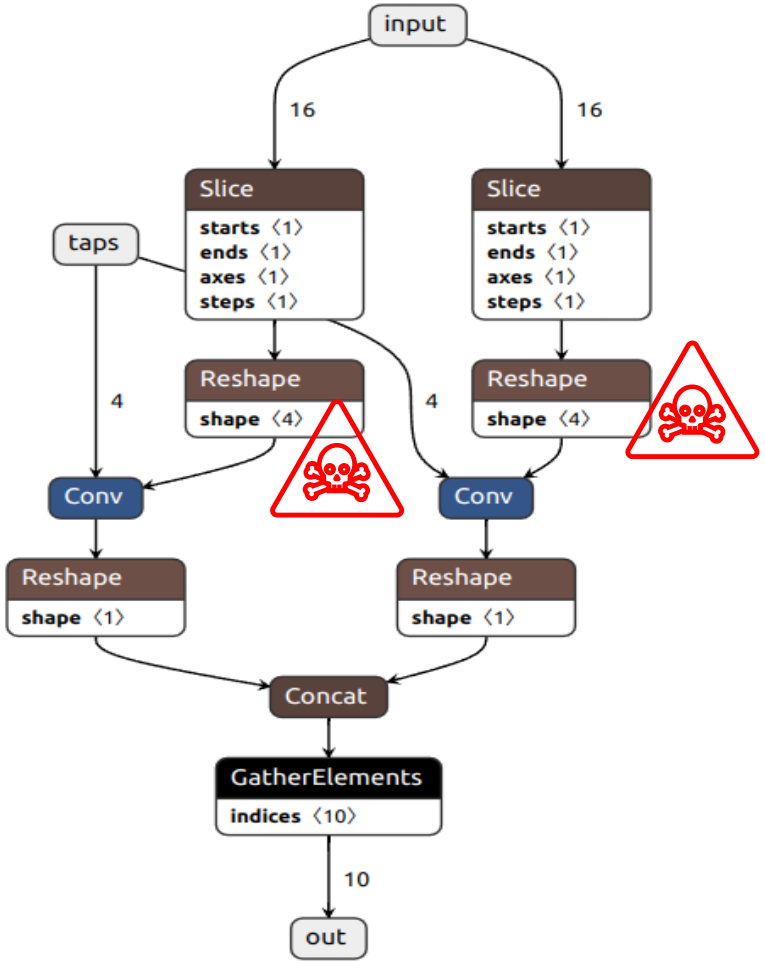


« alpha » parameter is the phase increment

Example :

- SR : 1 MHz
- Oscillator at 200 kHz
- 16384 complex samples
- Output from Python + CPU Runtime

Filter

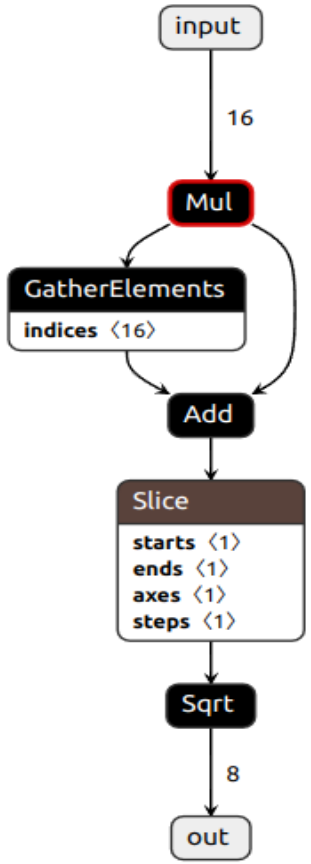


Notes :

- Taps are real
- Real part and imaginary part split and computed separately
- Output formed by re-interleaving real & imaginary parts

- Fun: the “Conv” operator as some “nice” requirements and wants “tensors” with specific shapes

AM demodulator



Notes :

- Taps are real
- Real part and imaginary part split and computed separately
- Output formed by re-interleaving real & imaginary parts

FM demodulator

- RKNN does not implement *atan()*
- More time needed…

Is this mature ?

This is an invalid model. Type Error: Type 'tensor(double)' of input parameter (selector) of operator (GatherElements) in node () is invalid

Load model from ../../../cpxmult.onnx failed:/onnxruntime_src/onnxruntime/core/graph/model_load_utils.h:46 void onnxruntime::model_load_utils::ValidateOpsetForDomain(const std::unordered_map<std::__cxx11::basic_string<char>, int>&, const onnxruntime::logging::Logger&, bool, const std::string&, int) ONNX Runtime only *guarantees* support for models stamped with official released onnx opset versions. Opset 22 is under development and support for this is limited. The operator schemas and or other functionality may change before next ONNX release and in this case ONNX Runtime will not guarantee backward compatibility. Current official support for domain ai.onnx is till opset 21.

Type Error: Type 'tensor(int32)' of input parameter (selreal) of operator (Where) in node () is invalid.

References

- RKNN : https://github.com/airockchip/rknn-toolkit2/
- Supported : https://github.com/airockchip/rknn-toolkit2/blob/master/doc/RKNNToolKit2_OP_Support-2.3.0.md

- ONNX operators : https://onnx.ai/onnx/operators/index.html
- Python API: https://github.com/scailable/sclblonnx

- Netron to view the graphs : https://github.com/lutzroeder/netron
- Python intro: https://towardsdatascience.com/creating-editing-and-merging-onnx-pipelines-897e55e98bb0

My conclusion

- Promising but clearly needs time
- Software architecture:
  - Have a “signal flow” description (very much like GnuRadio companion)
  - Generate the ONNX file
  - Run it…

- Benchmarks **are** required

Where is that ?

- https://github.com/f4gkr/onixradio
- Code :
  - Python generators for some DSP blocks provided
  - C code to run the complex multiplication as an example

  - Still a lot of work to be done…