Skymizer | ONNC Software Architecture Overview

Po-Yen Chen (poyenc@skymizer.com)

a

Open Neural Network Compiler (ONNC)

ONNC as a framework…

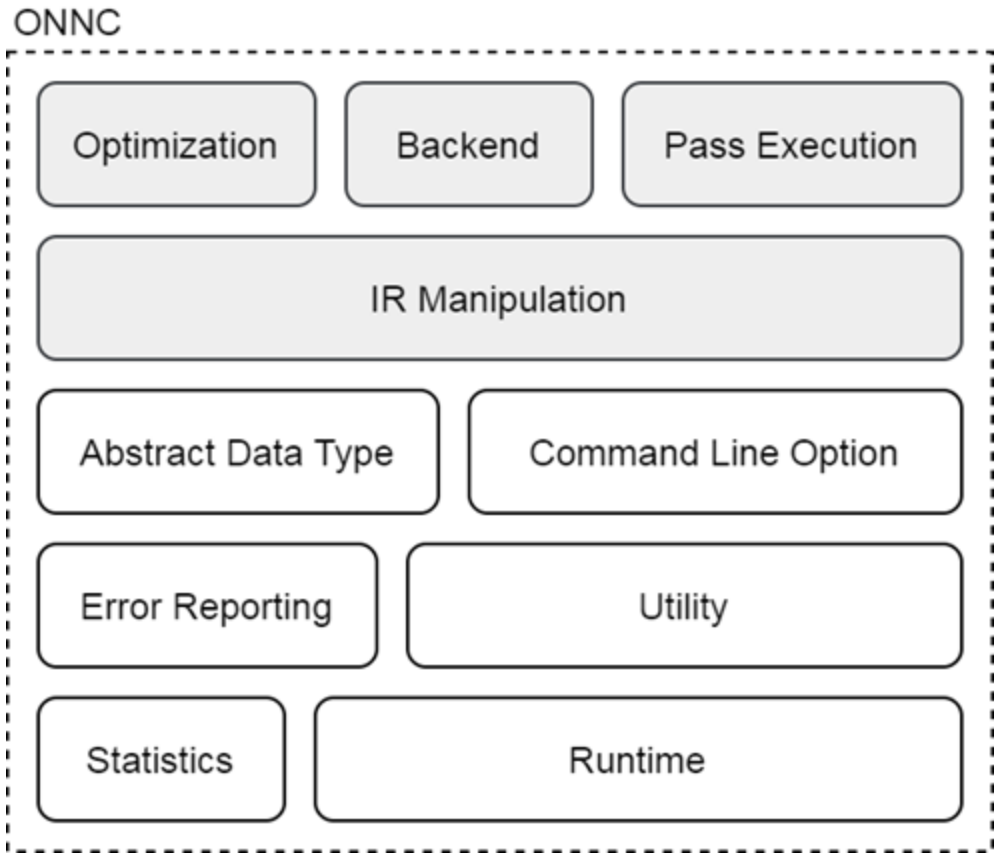
- "*Retargetable compilation framework designed specifically for proprietary deep learning accelerators.*"

ONNC as a compiler…

- "*ONNC is the first open source compiler available for NVDLA-based hardware designs.*"

skymizer

ONNC As A Framework

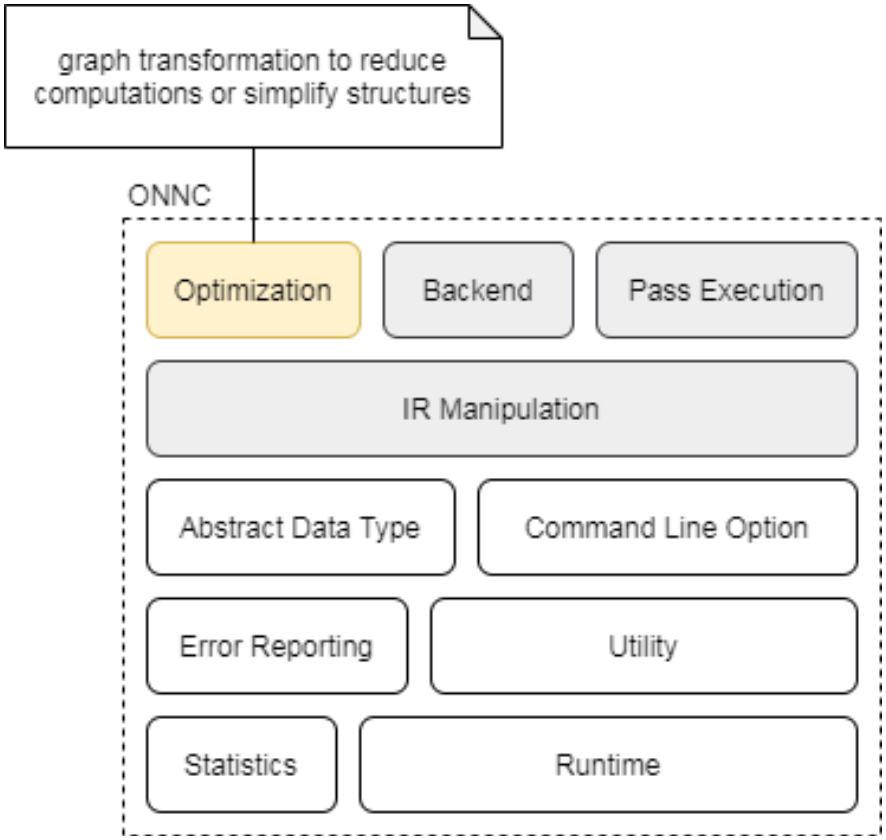


skymizer

ONNC As A Framework – Optimization

Built-in optimizations

- Partition *GlobalAveragePool* into multiple *AveragePool*
- Expand *BatchNormalization* into *Add* & *Mul*
- Convert *Gemm* to *Conv*
- Split *Conv* channels
- Propagate constant tensor
- Eliminate *Identity*
- Eliminate *Cast*

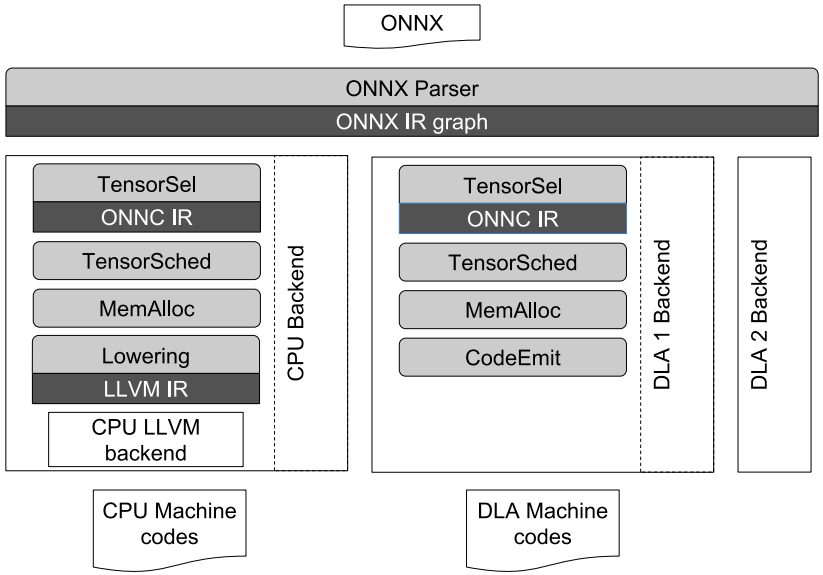


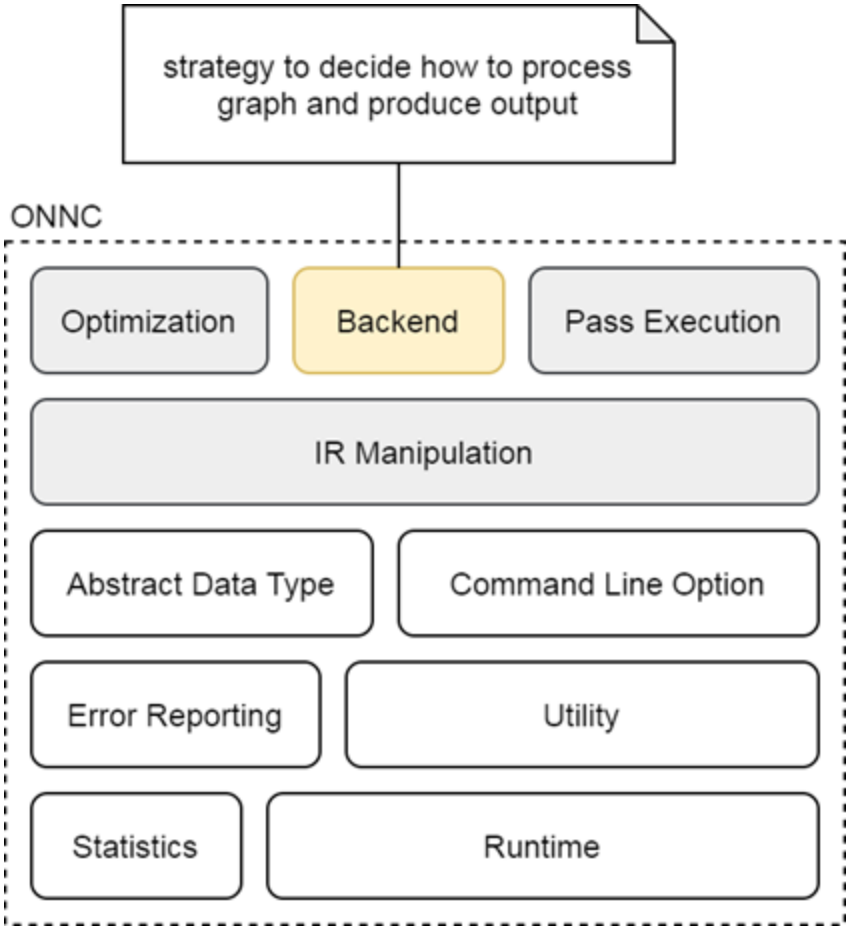
skymizer

ONNC As A Framework – Backend

Built-in optimizations

- *NVDLA* Backend: generate *NVDLA* loadable
- x86 Backend: default backend (no output)
- C Backend: create inference API in C



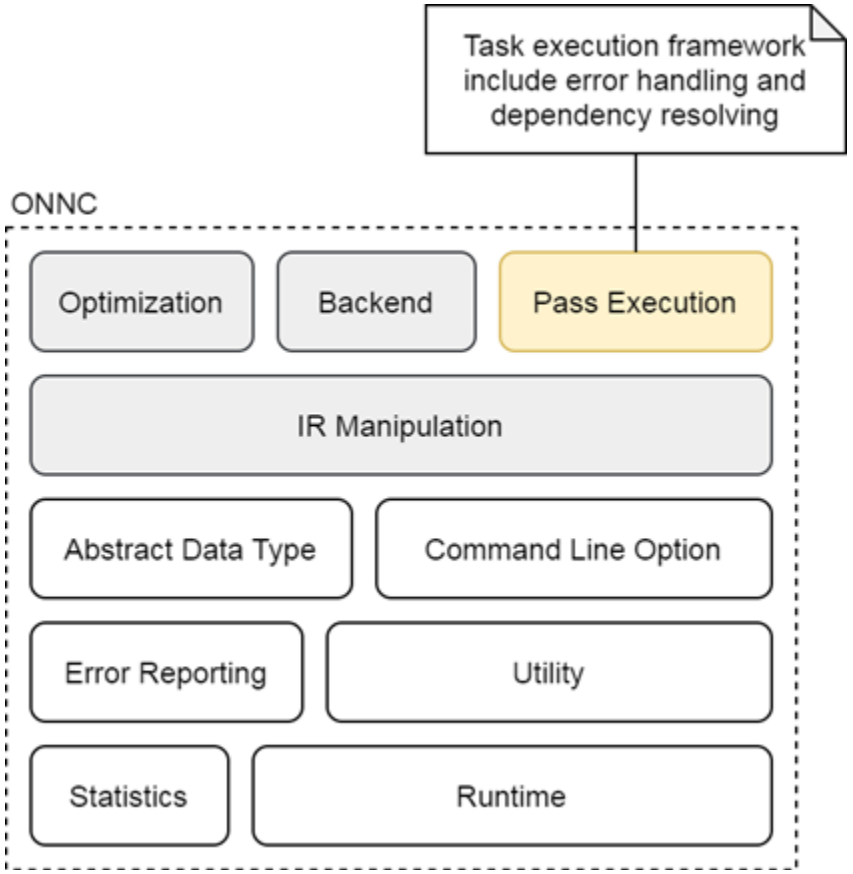


skymizer

ONNC As A Framework – Pass Execution

Types involved in task execution

- *Pass*: task itself
- *PassManager*: manage pass instances and run/stop passes
- *AnalysisUsage*: define dependency between passes

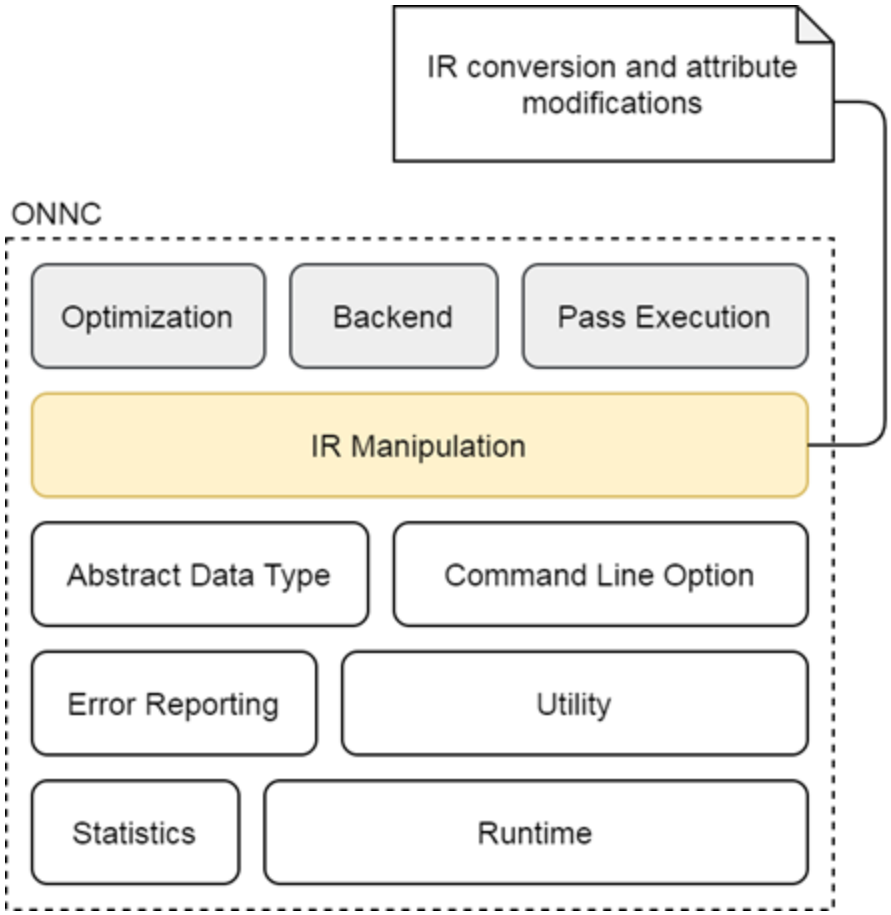


skymizer

ONNC As A Framework – IR Manipulation

IR manipulation infrastructures

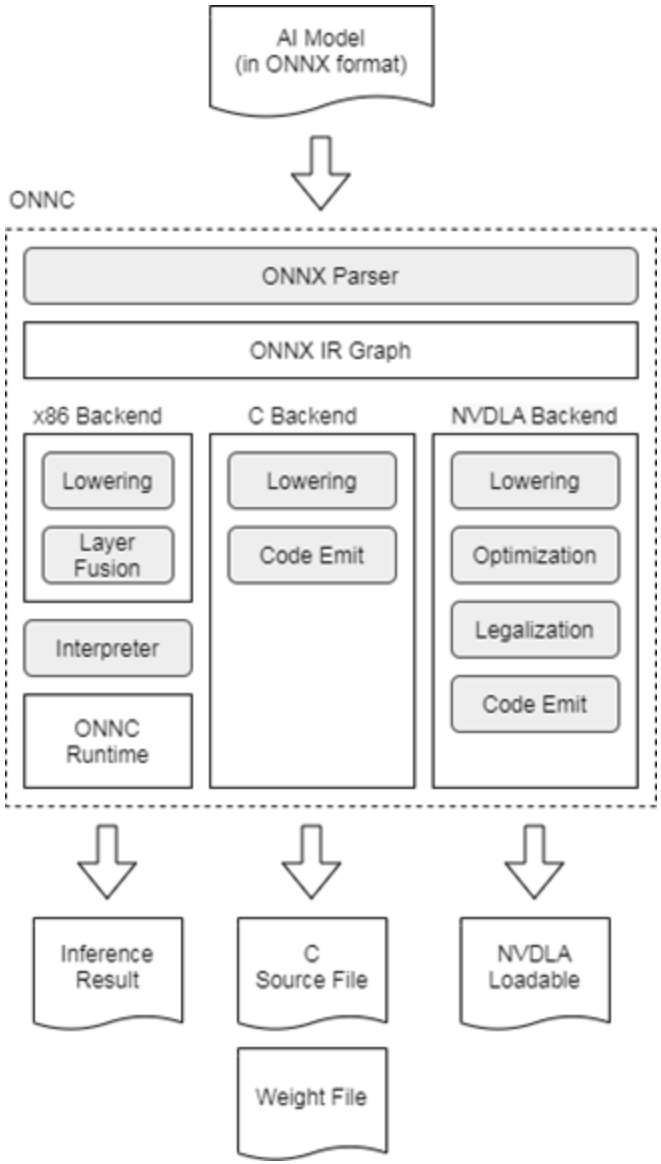
- *Lower*: convert ONNX IR to ONNC IR
- *Module*: the abstraction of model
- *ComputeGraph*: IR graph in model
- *ComputeOperator*: base type for every IR
- *Tensor*: base type for IR input/output storage
- *IRBuilder*: helper type for module creation



skymizer

ONNC As A Compiler

- ONNC accepts ONNX model as input
- All built-in backends convert ONNX IR to ONNC IR before taking any other actions
- Backend may perform *target-dependent* transformations
- Backend can work alone or be used as frontend in any scenarios



skymizer

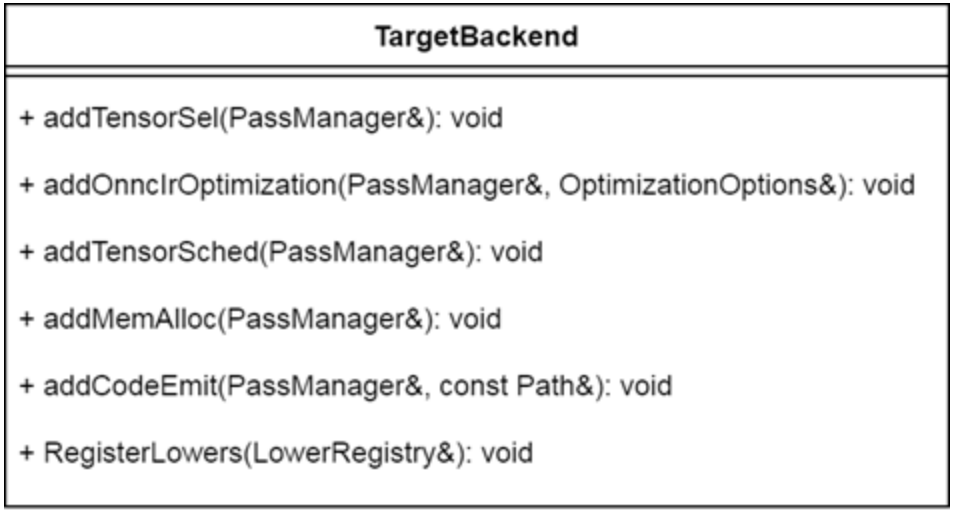
a

ONNC As A Compiler – Implement A Backend (I)

TargetBackend is the base type for every backend

Backend lifetime events:

1. *addTensorSel*: converts ONNX IR to ONNC IR
2. *addOnncIrOptimization*: perform any graph transformation
3. *addTensorSched*: schedule IR executions
4. *addMemAlloc*: analyze and allocate memory to store results
5. *addCodeEmit*: generate output files



skymizer

a

ONNC As A Compiler – Implement A Backend (II)

Steps to create your own backends…

1. Create a backend class inherited from *TargetBackend*
2. Override *RegsterLowers()* method - define valid ONNX operators
3. Override lifetime events if necessary
4. Register backend class into ONNC
5. Rebuild ONNC

Details covered in *Backend Developer Guide*,
or reference the example backend: *VanillaBackend*
or use prepared script *create-new-backend.sh*

```
# under onnc repository root, run script to create a new backend called Test (TestBackend)
$ ./script/create-new-backend.sh Test
# after build ONNC with new backend, use TestBackend to compile a model
$ onnc -mquadruple test <onnx model file>
```


skymizer

ONNC As A Compiler – Implement A Backend (III)

Override *RegsterLowers()* method: define valid ONNX operators

lib/Target/Vanilla/VanillaBackend.cpp:115

```
115  void VanillaBackend::RegisterLowers(LowerRegistry& pRegistry) const
116  {
117    pRegistry.emplace<AddLower>();
118    pRegistry.emplace<AveragePoolLower>();
119    pRegistry.emplace<BatchNormalizationLower>();
120    pRegistry.emplace<ConcatLower>();
121    pRegistry.emplace<ConvLower>();
122    pRegistry.emplace<FlattenLower>();
123    pRegistry.emplace<GemmLower>();
```


skymizer

skymizer

Boost deep learning accelerator
with compiler technology

Skymizer Taiwan Inc.

CONTACT US

E-mail sales@skymizer.com **Tel** +886 2 8797 8337

HQ 12F-2, No.408, Ruiguang Rd., Neihu Dist., Taipei City 11492, Taiwan

BR Center of Innovative Incubator, National Tsing Hua University, Hsinchu Taiwan

https://skymizer.com