Tensil.ai

Technology Overview

NIME Workshop - 2022/06/28

"One-click" accelerators

What if you could just run this to get a custom ML accelerator specialized to your needs?

```
$ tensil rtl --arch <my_architecture>
```

What if compiling your ML model for that accelerator target was as easy as running this?

```
$ tensil compile --arch <my_architecture> --model <my_model>
```

What is Tensil?

Tensil is a set of **free** and **open source** tools for running machine learning models on custom accelerator architectures. It includes:

- RTL generator
- ML model compiler
- Drivers

It enables you to create a custom accelerator, compile an ML model targeted at it, and then deploy and run that compiled model.

From prototype to production

The primary goal of Tensil is to allow anyone to run their applications on domain-specific accelerators.

- Tensil makes it easy to get started with an accelerator for prototyping and early products
- To scale up to mature production, we crank up the optimizations and the user gets a drop-in replacement (paid service)

Why should you use Tensil (as of June 2022)?

- you have a convolutional neural network based ML workload
- you need to run it at the edge (i.e. not in a data-center)
- you want to avoid changing your model

Current limitations

- doesn't support recurrent neural networks
- driver support for FPGAs only

Help us plan our roadmap!

How to install

To install from Docker, run:

```
$ docker pull tensilai/tensil:latest
$ docker run -v $(pwd):/work -w /work -it tensilai/tensil:latest bash
```

You will be dropped into a shell inside the Tensil container. Run

```
$ tensil compile --help
```

to verify that it is working correctly.

Generate an accelerator (RTL blob)

```
$ tensil rtl -a <tarch_file> -d <axi_port_width>
```

You should see some output like this:

```
$ tensil rtl -a /demo/arch/ultra96v2.tarch -d 128
Elaborating design...
Done elaborating.
--------------------------------------------------------------------
ARTIFACTS
--------------------------------------------------------------------
Verilog bram_dp_256x4096:   /work/bram_dp_256x4096.v
Verilog bram_dp_256x20480:  /work/bram_dp_256x20480.v
Verilog top_ultra96v2:      /work/top_ultra96v2.v
Driver parameters C header: /work/architecture_params.h
--------------------------------------------------------------------
```

Integrate RTL block

1. Instantiate Tensil IP
2. Connect AXI interfaces
    a. One AXI-S slave receives instructions from host
    b. Two AXI masters for host memory access
3. Generate bitstream

Compile an ML model

```
$ tensil compile -a <tarch_file> -m <onnx_file> -o output_node -s true
```

You should see some output like this:

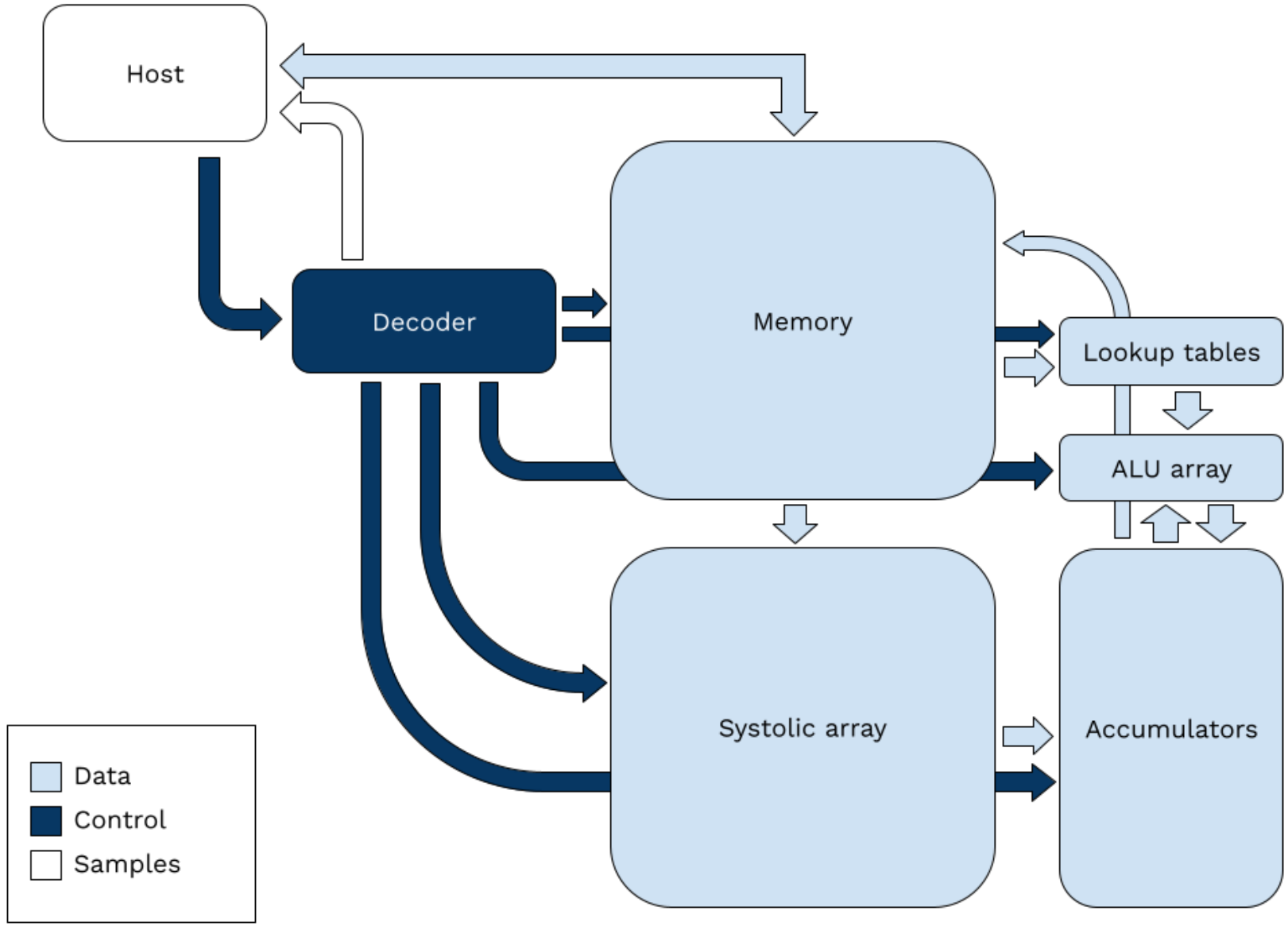
```
$ tensil compile -a /demo/arch/ultra96v2.tarch -m /demo/models/resnet20v2_cifar.onnx -o "Identity:0" -s true
--------------------------------------------------------------------------------
COMPILER SUMMARY
--------------------------------------------------------------------------------
Model:                                          resnet20v2_cifar_onnx_ultra96v2
Data type:                                      FP16BP8
Array size:                                     16
Consts memory aggregate usage (vectors/scalars): 35,743                  571,888
Vars memory aggregate usage (vectors/scalars):   46,097                  737,552
Number of layers:                               23
Total number of instructions:                   102,741
Consts utilization (%):                         97.210
True MACs (M):                                  61.476
------------------------------------------
ARTIFACTS
------------------------------------------
Manifest:  /work/resnet20v2_cifar_onnx.tmodel
Constants: /work/resnet20v2_cifar_onnx.tdata
Program:   /work/resnet20v2_cifar_onnx.tprog
```

Run the compiled model (PYNQ example)

Get the Tensil PYNQ driver, then import it and run!

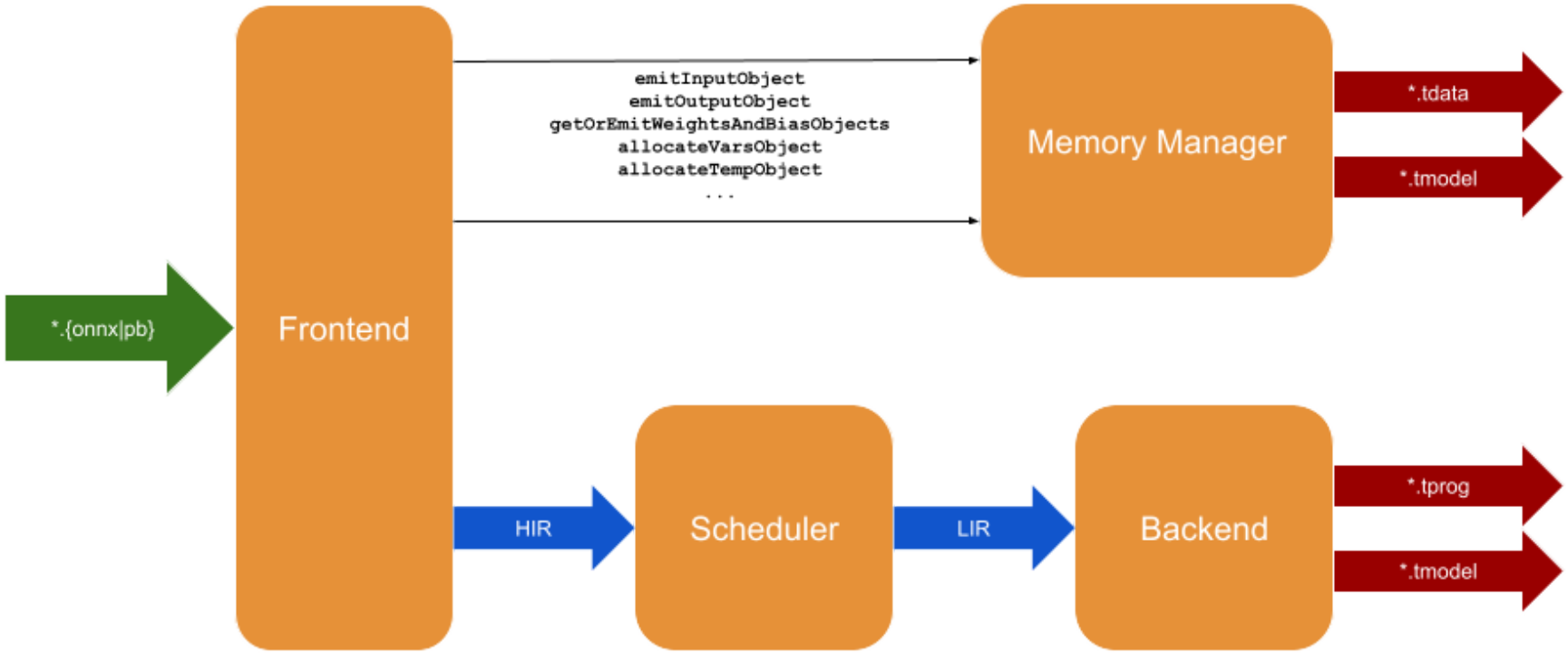
```
from pynq import Overlay
from tcu_pynq.driver import Driver
from tcu_pynq.architecture import ultra96

bitstream = '/home/xilinx/ultra96-tcu.bit'
overlay = Overlay(bitstream)
tcu = Driver(ultra96, overlay.axi_dma_0)

resnet = '/home/xilinx/resnet20v2_cifar_onnx_ultra96v2.tmodel'
tcu.load_model(resnet)

img = ...
output = tcu.run({'x:0': img})
```

Accelerator architecture



Instruction set

Name	Description
MatMul	Load input at memory address into systolic array and store result at accumulator address
DataMove	Move data between the main memory and either the accumulators or one of two off-chip DRAMs
LoadWeight	Load weight from memory address into systolic array
SIMD	Perform computations on data in the accumulator
Configure	Set configuration registers



Compiler structure

Benchmarks

ResNet-20v2 trained for CIFAR

FPGA Board	Tensil Array Size	Clock (MHz)	Latency (ms)	Frames per second
Arty A7-35	8x8	150	21	48
Pynq Z1	12x12	150	14	71
Ultra96-V2	16x16	300	4	250

Learn more

Website: www.tensil.ai

Email: contact@tensil.ai

Github: github.com/tensil-ai

Discord: discord.gg/TSw34H3PXr

Thank you!