EE310 - Hardware Description Languages
Spring 2025
Lab Assignment 4:
Chisel-based Convolution Module Design and Simulation

1 Introduction

In this lab assignment, students will implement and simulate various convolution modules using the Chisel hardware description language. The focus is on modeling and testing depthwise, pointwise, and regular convolution operations used in machine learning workloads. All work is to be done in simulation only; no synthesis or FPGA implementation is required. The goal is to understand architectural modularization using traits, inheritance, and parameterized hardware design in Chisel.

2 Tools and Directory Structure

This project uses Chisel (Scala-based HDL), sbt (Scala Build Tool), and Verilator (for simulation). The directory structure of the project is as follows:

```
lab4_template/
  pw.py
  dw.py
  rc.py
  build.sbt
  project/ (Automatically generated upon running)
  src/
    main/
      scala/
        conv/
          Conv.scala
          Main.scala
    test/
      scala/
        conv/
          ConvSpec.scala
  target/(Automatically generated upon running)
```

Your convolution modules will be located in **Conv.scala** file. **Main.scala** has an example module to demonstrate how you can emit a SystemVerilog file from your Chisel code. Update its contents according to the module you would like to emit.
Inside **ConvSpec.scala**, you may find example test suites inside comments. You may refer to their structure when building your own.
A tutorial on how to install and configure the required dependencies (Scala, sbt, Verilator) is provided separately on the course website. Please refer to that tutorial before starting this assignment.

3 Background

Convolution is a fundamental operation in image processing and deep learning, often used for extracting features from input data. This assignment involves three types of convolutions:

Depthwise Convolution (DW)

In depthwise convolution, each input channel is convolved with its own separate filter. This operation only captures spatial relations between cells.
Visualization

Pointwise Convolution (PW)

Pointwise convolution is a 1x1 convolution applied across the depth dimension, allowing for channel mixing and often used to increase or decrease channel dimensions. This operation only captures temporal relations between cells.
On the below animation, first Depthwise Convolution takes place, after that Pointwise Convolution operation starts. For the operation of Pointwise Convolution, the latter part of the animation will be implemented.
Visualization

Regular Convolution (RC)

Regular convolution uses multi-channel filters across both spatial and depth dimensions. Each filter generates one output channel by aggregating data from all input channels.
Visualization

Operation Comparison

The following table summarizes the input, filter, and output shapes for each convolution operation. All of the operations will be performed with a stride of 1.

$$\begin{aligned}\texttt{input} &= h_{in} \times w_{in} \times d_{in} \\ \texttt{filter} &= (h_{filt} \times w_{filt} \times d_{filt}) \times n_{filt} \\ \texttt{output} &= h_{out} \times w_{out} \times d_{out}\end{aligned}$$

The specific configurations for each convolution type are summarized in the table:

Op	**Input**	**Filter**	**Output**
DW	$h_{in} \times w_{in} \times d_{in}$	$(h_{filt} \times w_{filt} \times 1) \times d_{in}$	$(h_{in} - h_{filt} + 1) \times (w_{in} - w_{filt} + 1) \times d_{in}$
PW	$h_{in} \times w_{in} \times d_{in}$	$(1 \times 1 \times d_{in}) \times d_{out}$	$h_{in} \times w_{in} \times d_{out}$
RC	$h_{in} \times w_{in} \times d_{in}$	$(h_{filt} \times w_{filt} \times d_{in}) \times d_{out}$	$(h_{in} - h_{filt} + 1) \times (w_{in} - w_{filt} + 1) \times d_{out}$

Examples

In this subsection, visual examples of depthwise, pointwise, and regular convolution operations are provided below. Each operation is shown using small input tensors and filter sizes for clarity. The helper functions used to generate those examples are included in the provided template.

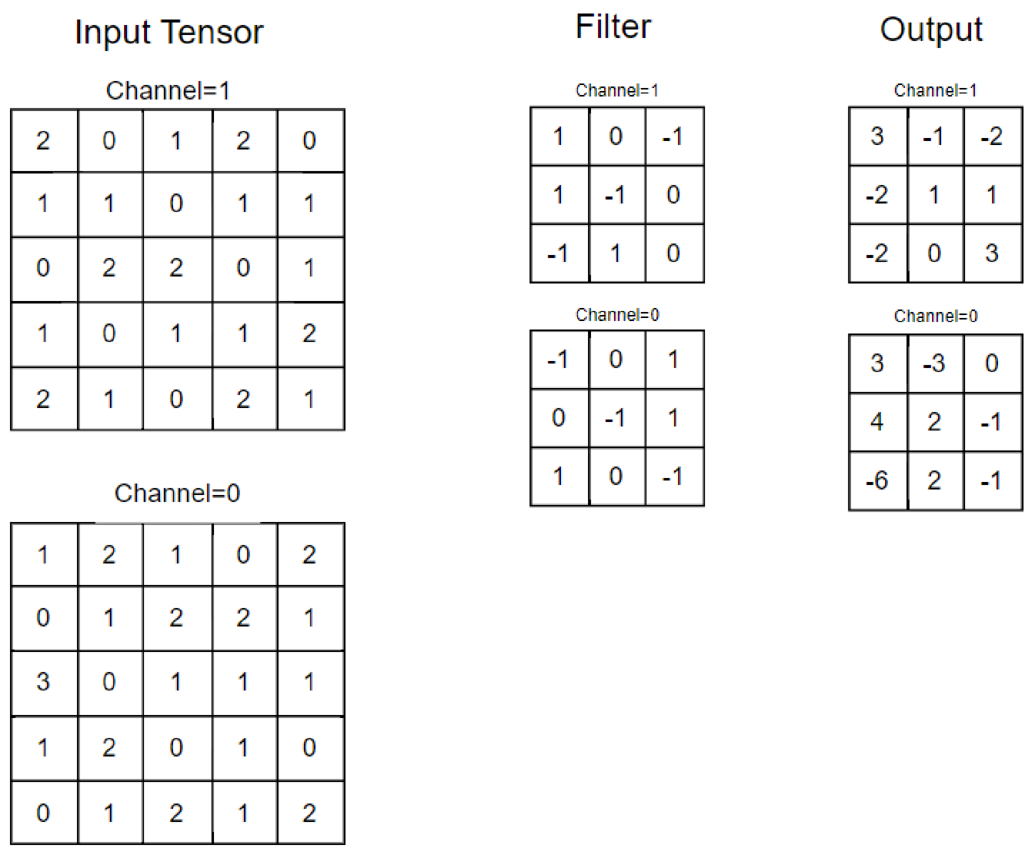

Figure 1: Depthwise Convolution

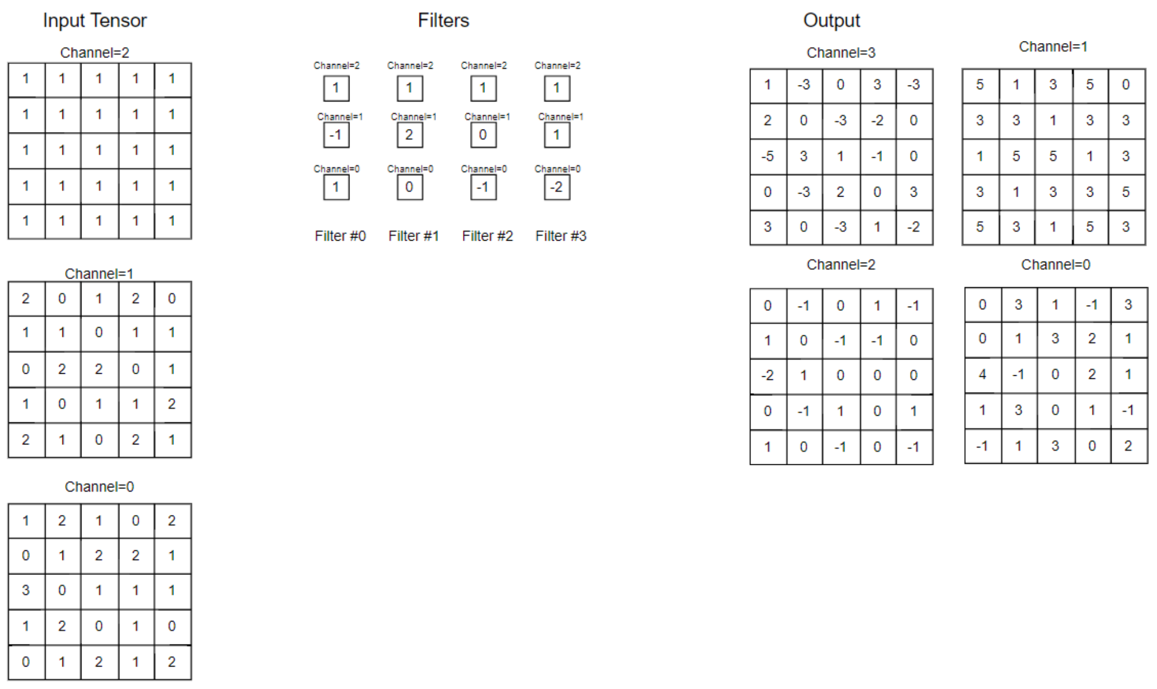

Figure 2: Pointwise Convolution

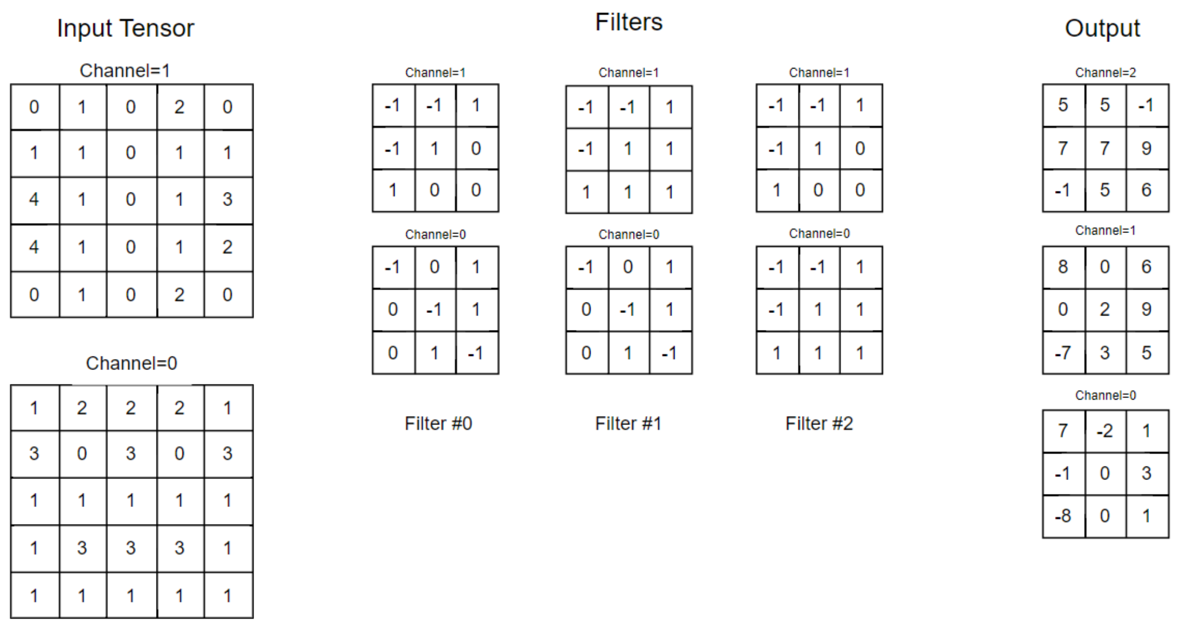

Figure 3: Regular Convolution

4 Task

Students should begin by downloading the provided project template, which contains partial implementations and testing infrastructure. The project is built around modular hardware design with object-oriented concepts such as abstract classes and traits.

The provided code includes:

- `DotProduct` module
- `BaseConvolution` abstract class
- `SpatialComputation` trait
- `DepthwiseConv` class

Students are required to implement:

- `TemporalComputation` trait
- `PointwiseConvolution` class (must extend `TemporalComputation`)
- `RegularConv` class (must extend either `TemporalComputation`, or `SpatialComputation`, or both)

Each implemented module must include a dedicated testbench in the style of the provided test suites. Proper verification of functionality using Chisel's simulation utilities is expected.

5 Submission Requirements

Your zipped submission must include:

- All source code and test suites
- A short report (PDF)

The report should describe your design methodology and highlight any interesting or unique decisions made during the implementation. There is no need to include waveform screenshots or simulation results.

Deadline: 14.05.2025