Auto-scheduling Deep Neural Networks for Hardware

Xuan Yang, Mingyu Gao, Qiaoyi Liu, Ankita Nayak, Jing Pu, Jeff Setter, Steven Bell, Heonjae Ha, Priyanka Raina, Christos Kozyrakis, Mark Horowitz

Introduction

Motivation

- DNNs are widely used
- Various DNN accelerators were proposed
- Large parallelism and locality in DNNs

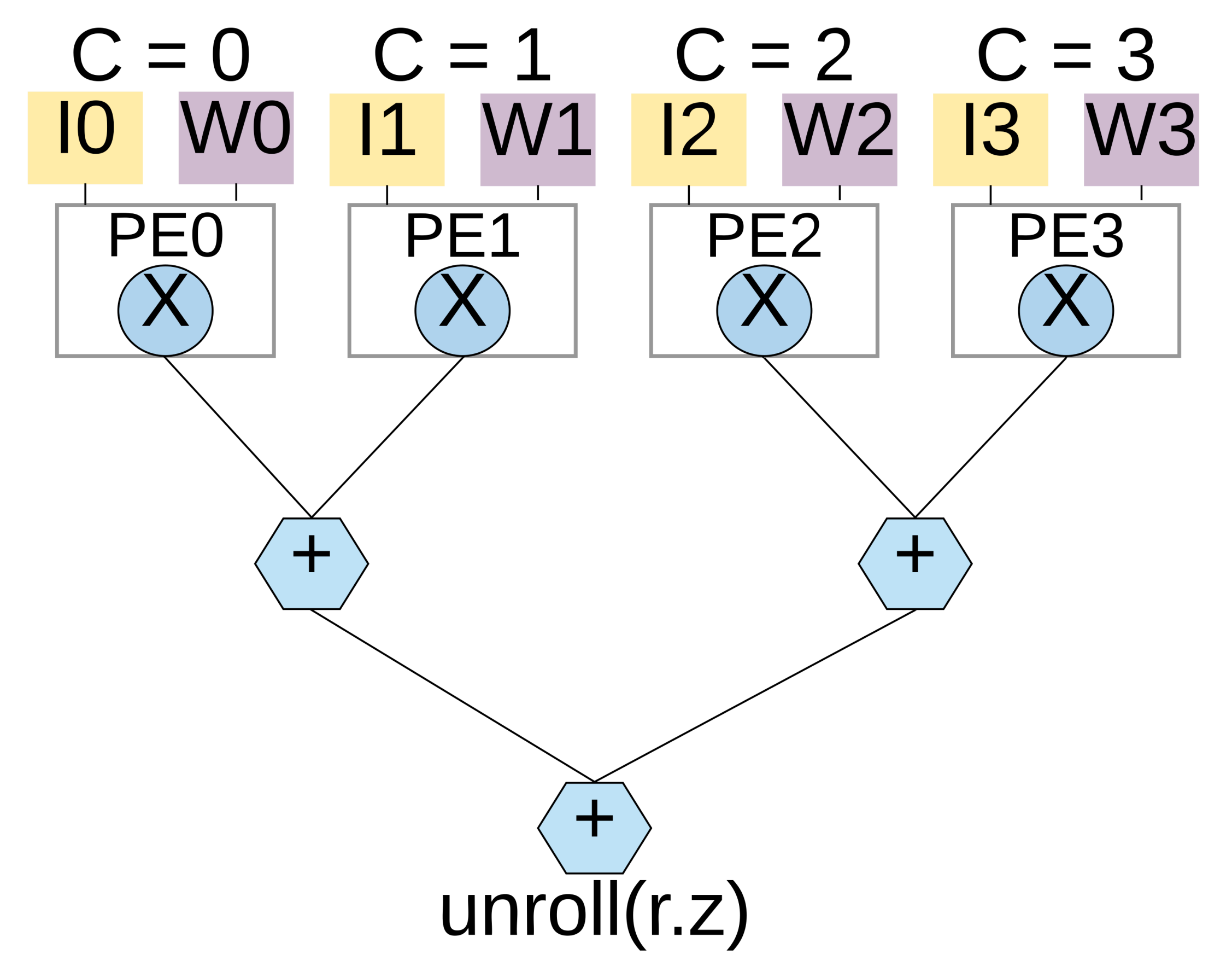
Architectural Optimization

- Balance different types of data reuse
- Loop blocking, parallelization

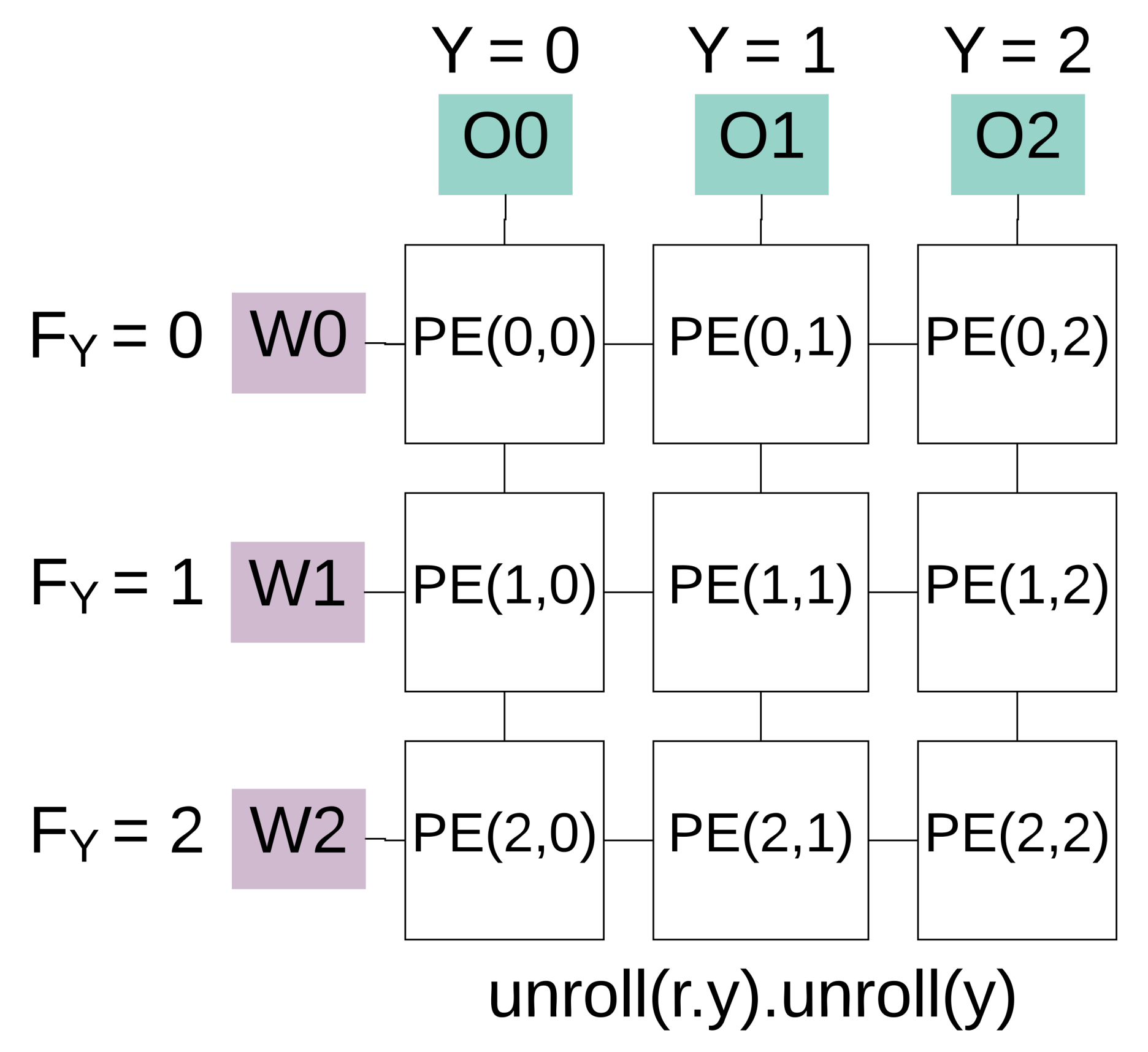
Language: Halide

- Halide splits *algorithm* from *schedule*
- Compact schedule to express hardware micro-architecture and mappings

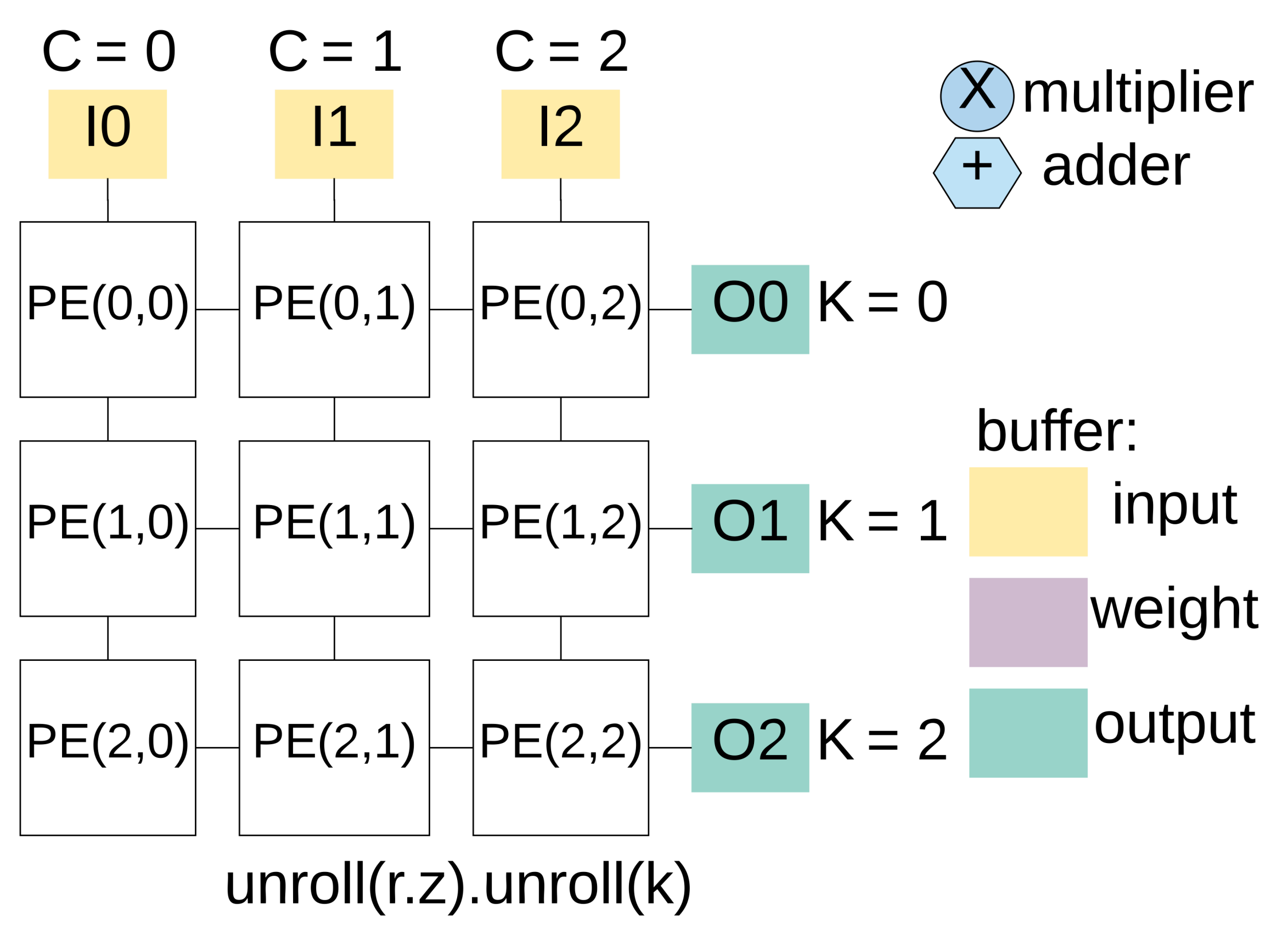
Micro-architectures Generation



Unrolling input channel dimension generates a reduction tree architecture.

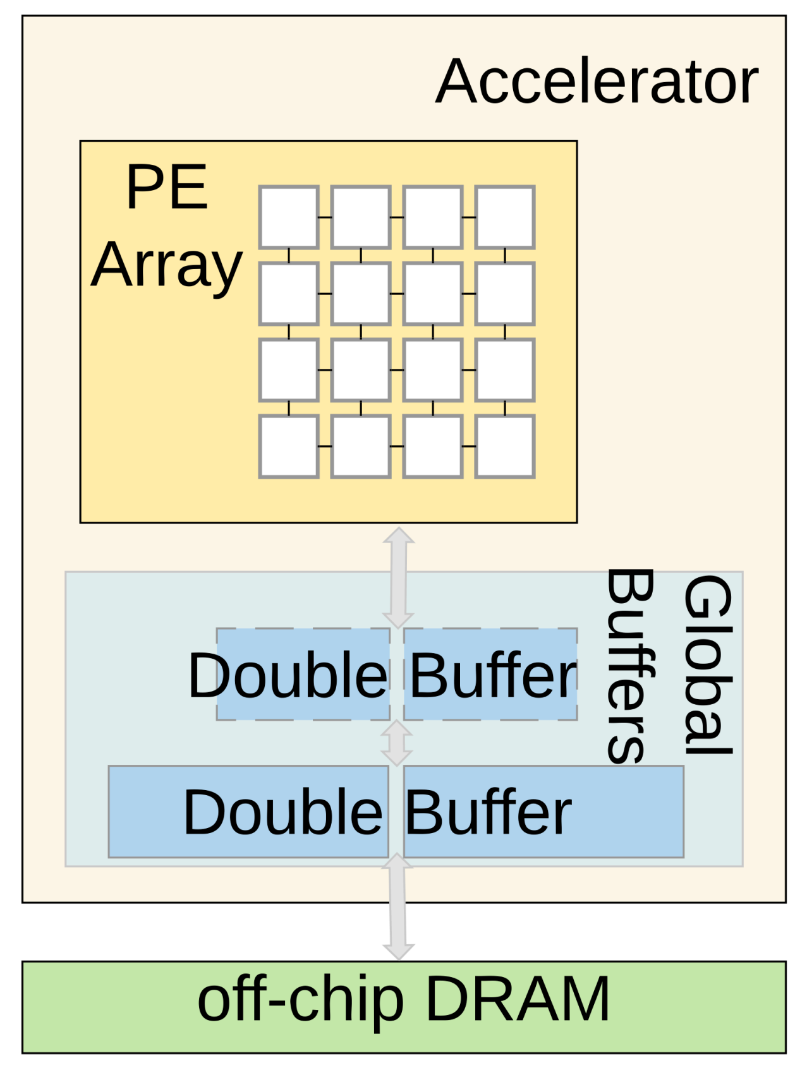


Unrolling image height and filter height dimensions, generates a systolic architecture, like Eyeriss.

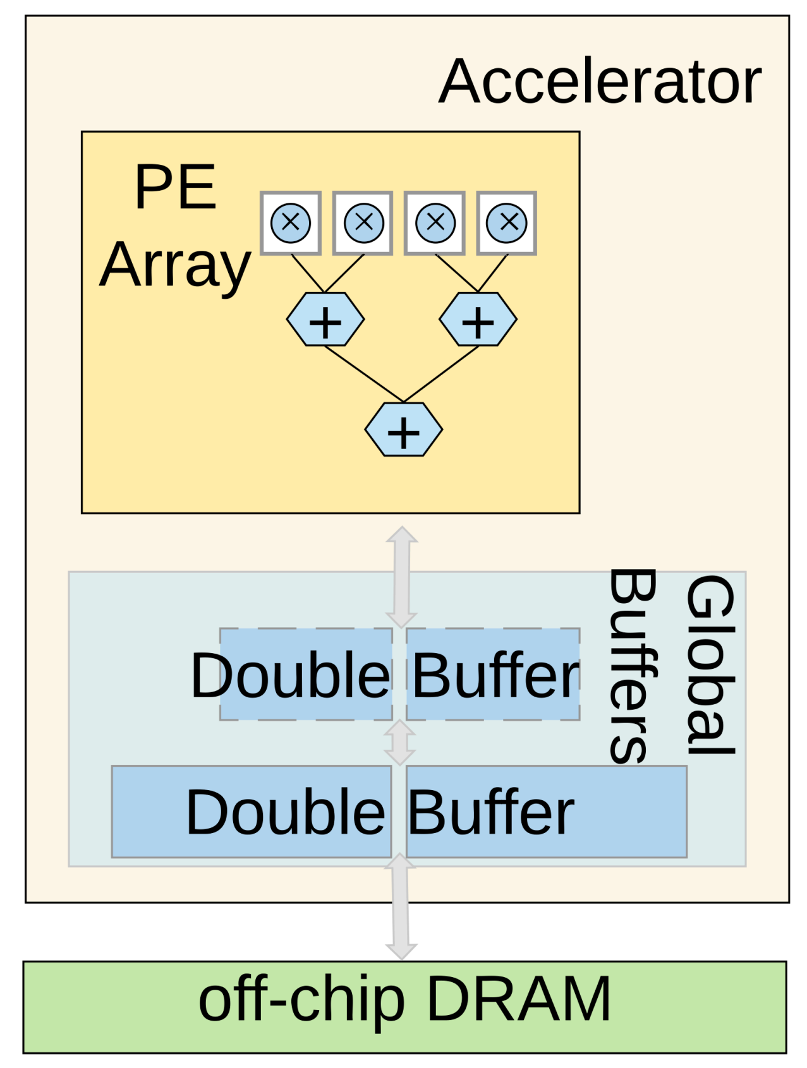

C = 0
C = 1
C = 2
I0 I1 I2
PE(0,0) PE(0,1) PE(0,2) O0 K = 0
PE(1,0) PE(1,1) PE(1,2) O1 K = 1
PE(2,0) PE(2,1) PE(2,2) O2 K = 2
unroll(r.z).unroll(k)
multiplier
adder
buffer:
input
weight
output

Unrolling input channel and output channel dimensions, generates a systolic architecture, like TPU.

Architecture Template



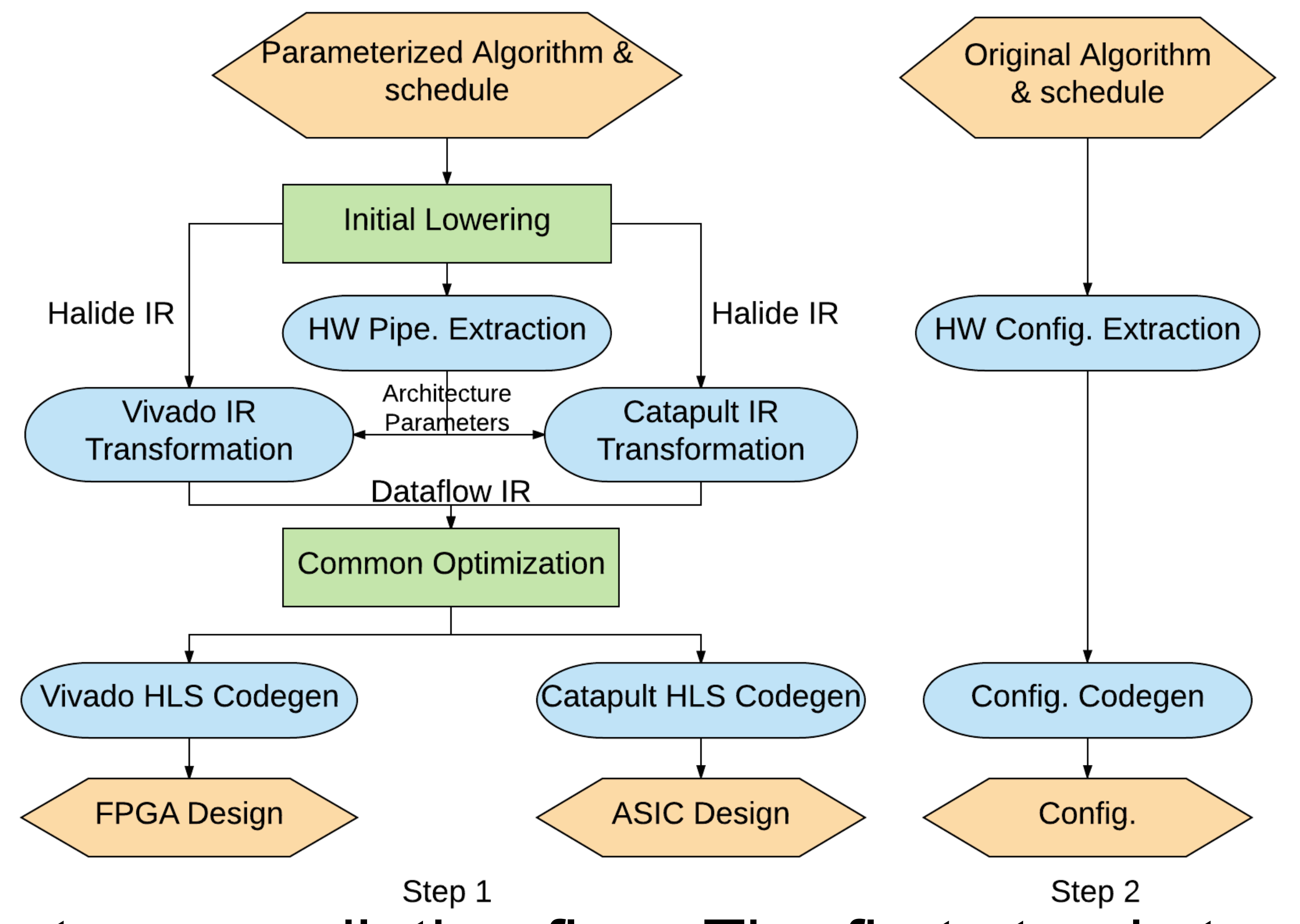
(a) Systolic Array



(b) Reduction Tree

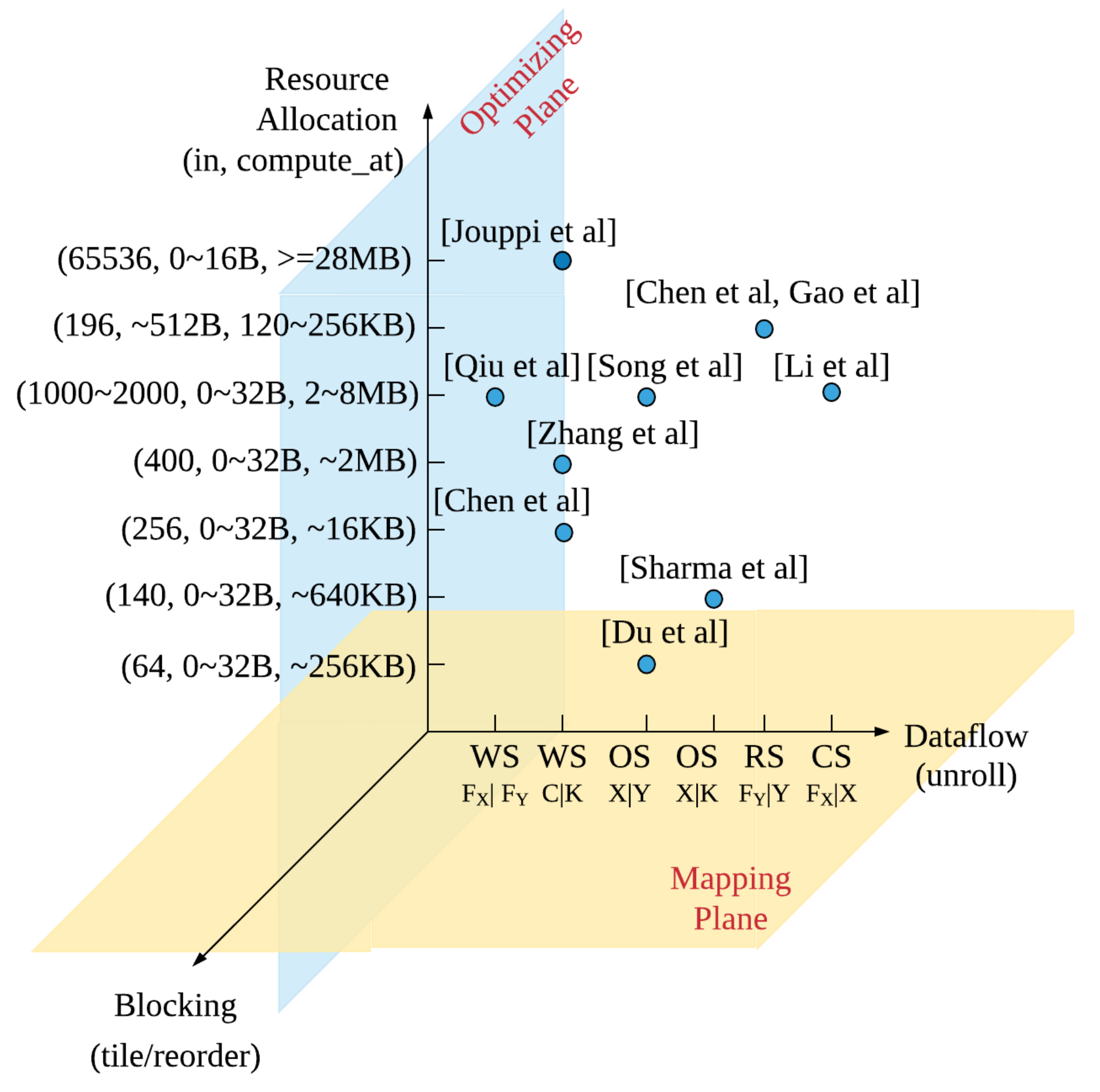
Architecture template for DNNs, the accelerator is composed of a PE array, a memory hierarchy with double buffers.

Virtualized Hardware Generation

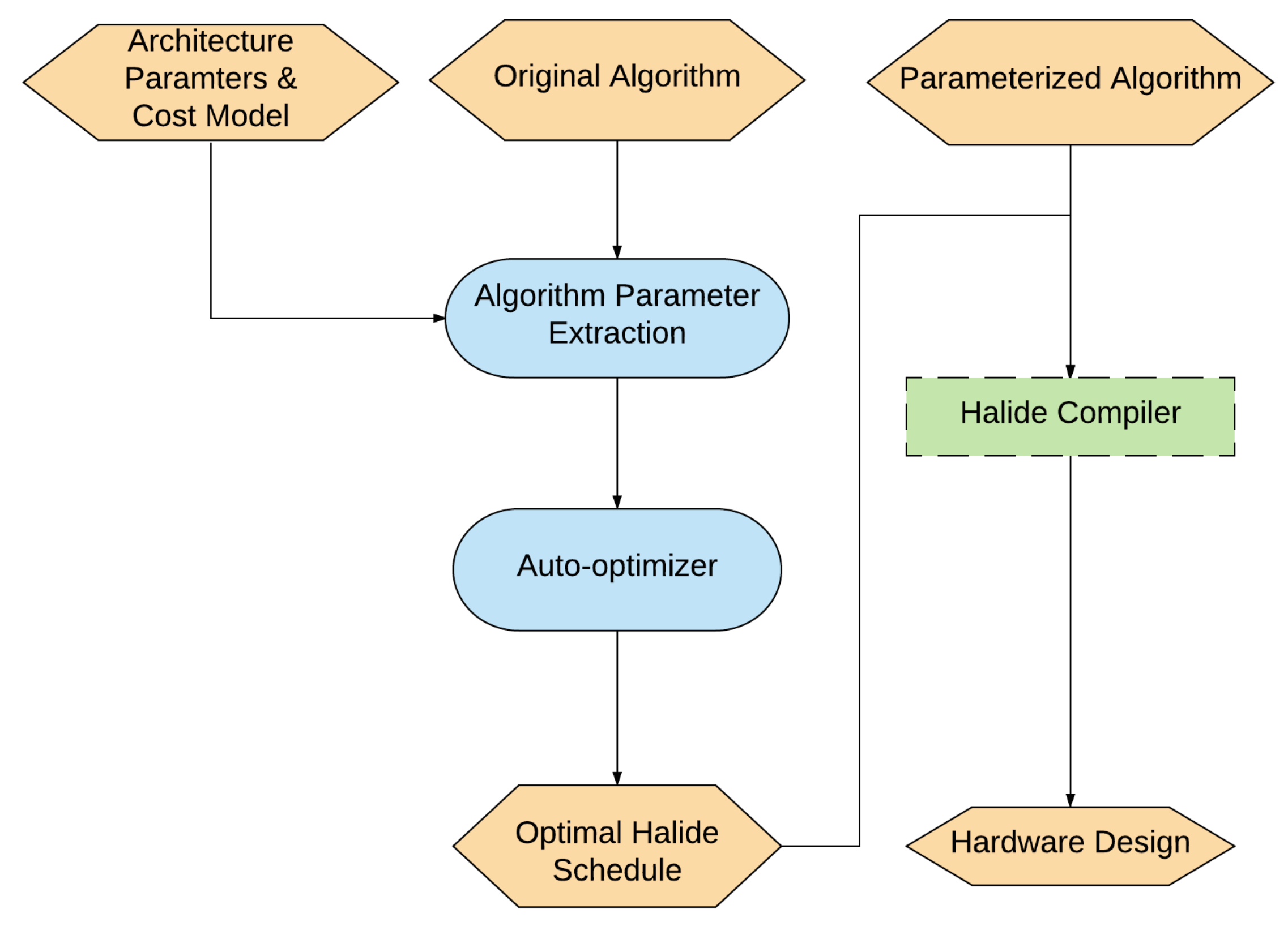


Two–step compilation flow. The first step is to generate a configurable hardware from a parameterized algorithm, the second step is to generate the configuration.

Auto-scheduler



The design space, Each hardware optimization dimension (*Blocking*, *Dataflow*, *HW Resource Allocation*) can be mapped to Halide schedule primitives (tile/reorder, unroll, in).



The auto-scheduler flow takes the original algorithm, architecture parameters and cost model to find the optimal schedule .

intel

ISTC Agile