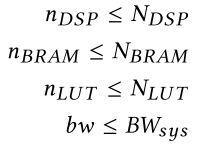
Reuse Kernels or Activations? A Flexible Dataflow for Low-latency Spectral CNN Acceleration

——FPGA‘20 University of Southern California

1. Contributions
   1. Provided a complexity analysis of spectral convolutional layers, focusing on the on-chip storage versus bandwidth tradeoff.
   2. Propose a flexible dataflow for choosing the optimal data reuse strategy and design a unified architecture that can adjust dataflow on the fly.
   3. Design an approximate exact-cover based scheduling algorithm to optimally schedule on-chip memory access with minimum read conflicts.
   4. Implement our design on a state-of-the-art FPGA platform.
2. Background
   1. Spectral CNNs
      1. Convert convolution in spatial domain into element-wise product in spectral domain with FFT.
      2. It reduces computation complexity but enlarges kernel size in spectral domain.
   2. FPGA acceleration
      1. FPGA designs are aimed to optimize certain objective function(maximize throughput, minimize latency or minimize power consumption) under resource constraints below.

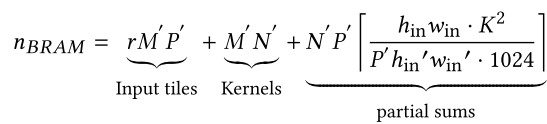


is the number of DSP, is the number of BRAM block, is the number of LUT and is off-chip bandwidth.

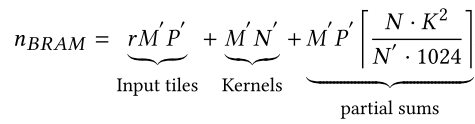
1. Complexity analysis

The complexity analysis is based on a general architecture model, in which data are originally stored in off-chip memory, then streamed into on-chip memory.

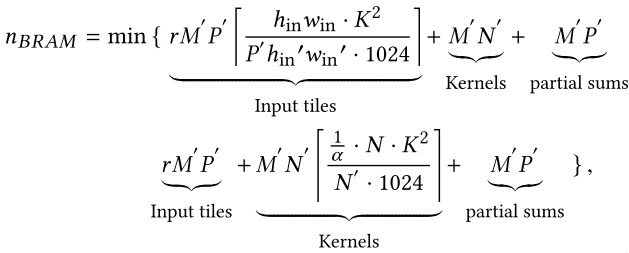
* 1. On-chip storage
     1. Suppose we parallelize kernels, input channels, and input tiles, given the fact that each BRAM can only support one concurrent access, each parallel line needs at least one BRAM to support accesses.
     2. In the case of reusing kernels and output partial sums（FLOW #1）, we keep kernels and partial sums on chip. The required BRAMs will be at least:



* + 1. In the case of reusing input tiles and output partial sums, streaming kernels(FLOW #2). The required BRAMs will be at least:



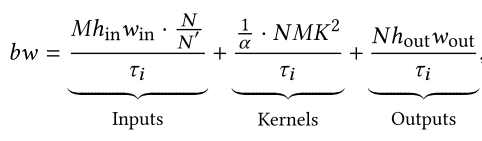
* + 1. In the case of reusing kernels and input tiles, streaming partial sums(FLOW #3), will be at least:



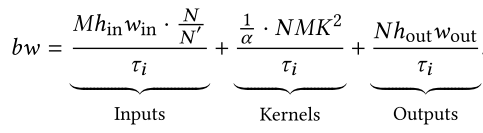
* + 1. Through this analysis, we can accurately describe if certain flow is bounded by BRAMs. And find out which flow needs minimum number of BRAMs.
  1. Communication bandwidth

Off-chip communication consists of reading inputs and kernels, writing outputs, writing and re-loading partial sums.

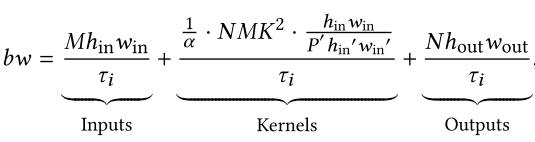
* + 1. In FLOW #1, given latency in layer , the required bandwidth is:



* + 1. In FLOW #2, the required bandwidth is:



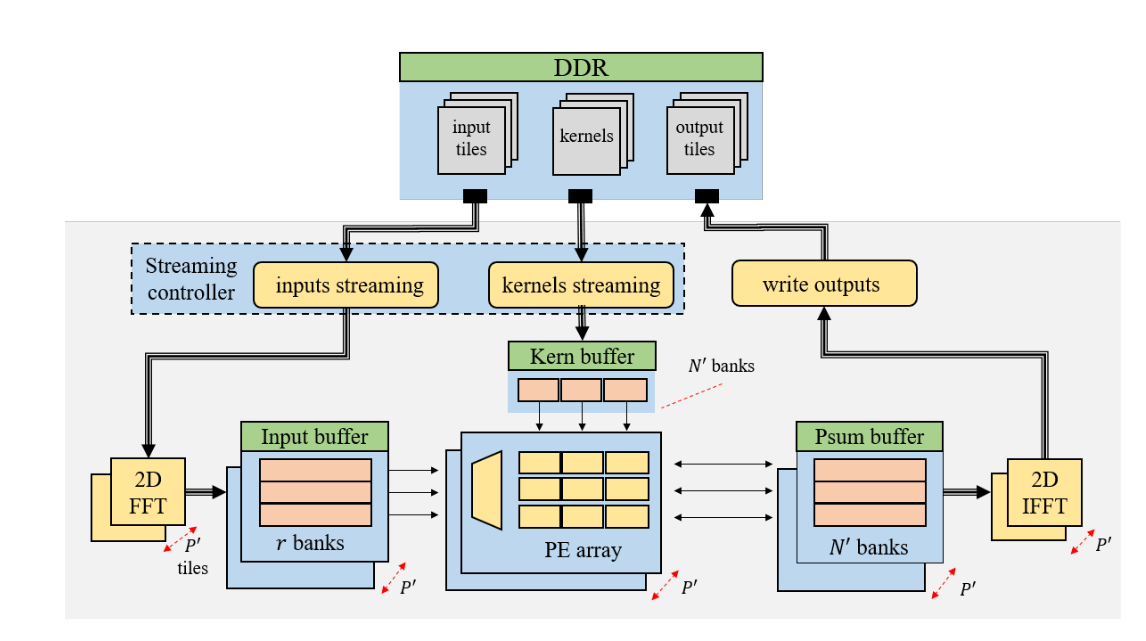
* + 1. In FLOW #3, the required bandwidth is:



* + 1. …

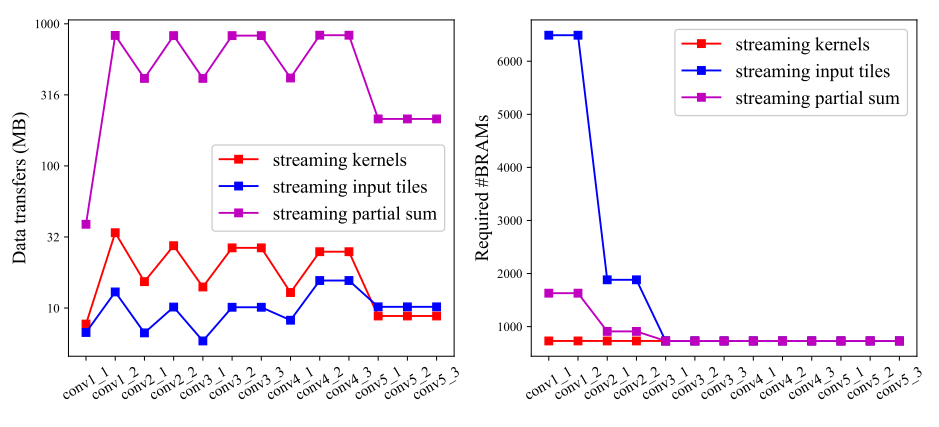
1. Architecture design
   1. Overview

A unified architecture that can adjust dataflow on the fly. Using a streaming controller to decide what data should be reused or streamed based on each layer’s configuration.



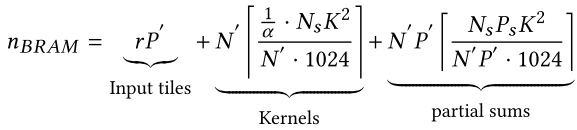
Overview Architecture

* 1. Flexible dataflow
     1. The required on-chip memory and communication bandwidth in three dataflows for compressed VGG16:

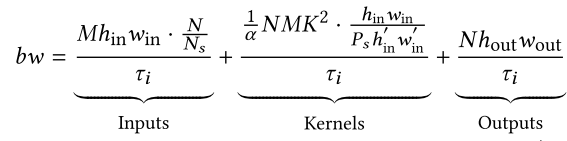


Complexity in VGG16

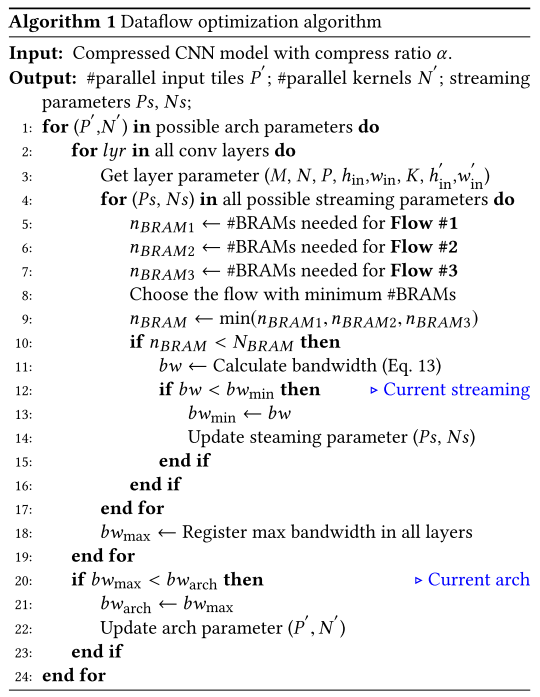
* + 1. Introduce two parameters (kernels) and (input tiles) as streaming parameters, then the required BRAMs is at least:



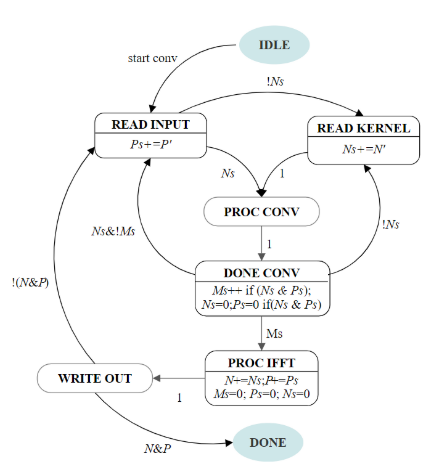
The required bandwidth is at least:



* + 1. Given a compressed model, the algorithm below does a heuristic search in architecture parameter space and streaming parameter space.
    2. Streaming options are managed by a internal state machine.

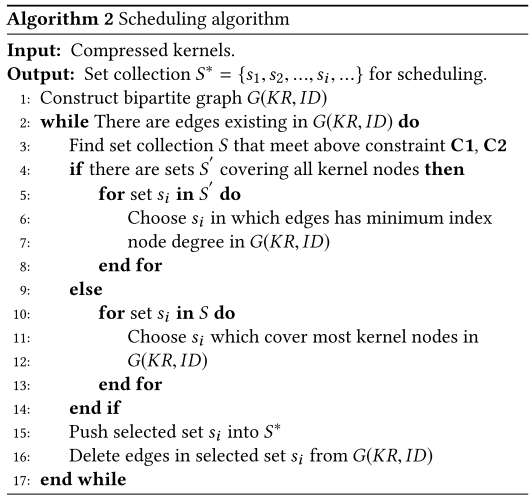


Algorithm



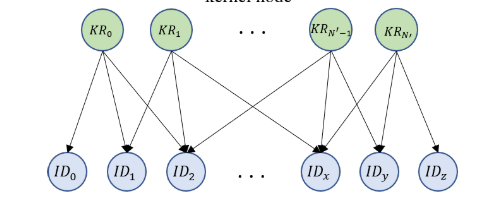
State machine in streaming controller

* + 1. …
  1. Memory access scheduling
     1. To prevent multiple kernels from starving for no available inputs, we use replicas to increase throughput of input BRAMs.
     2. Use the scheduling algorithm below to group access addresses so that the number of distinct addresses in each read cycle is less than .



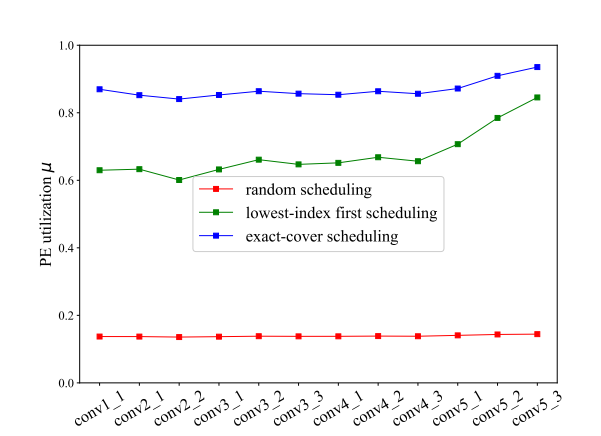
Scheduling algorithm

* + 1. Use bipartite graph to represent sparse kernels.
    2. …



Bipartite graph

1. Design evaluation
   1. PE utilization



* 1. Comparison with other works

