Update on Ara

04/08/2021

Matteo Perotti
Matheus Cavalcante
Nils Wistoff

Professor Luca Benini
Integrated Systems Laboratory
ETH Zürich

Benchmarks

- Add benchmark-CSR to indicate if Ara is busy
- Preliminary benchmark performance measure: iconv2d, dropout, jacobi2d
- New Benchmark: RoI Align

Improve the Cycle Count

```
start = start_timer();

vectorial_kernel();

end = end_timer();
cycle_count = end - start;
```

Improve the Cycle Count

```
start = start_timer();

vectorial_kernel();

end = end_timer();
cycle_count = end - start;
```

```
vectorial_kernel() {
  ...
  vload
  vmul
  vstore (in processing)
}
```


Problem! Maybe Ara is still computing vector instructions when end_timer() is called!

Improve the Cycle Count

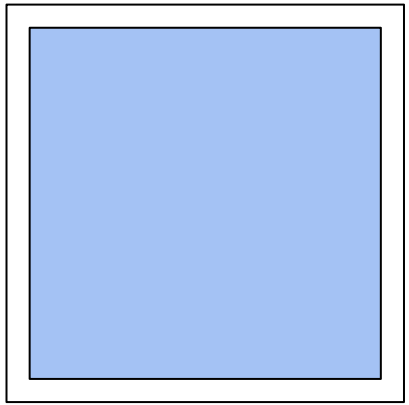
Cycle count problem:

- The timer used to measure cycles is inaccurate
- Ara, when the kernel is over, can be still busy!
- Critical issue for small kernels (e.g., dropout)

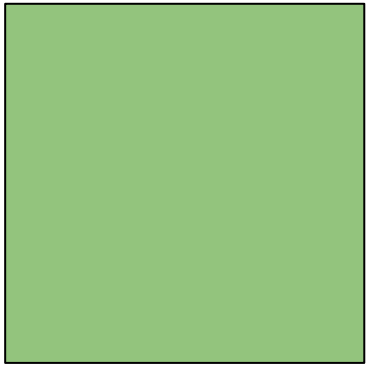
Solution:

- Add an AXI-accessible control register: `benchmark_reg`
- `benchmark_reg[0] = (ara_busy) ? 1'b1 : 1'b0;`
- Poll it in SW at the end of the kernel
- Stop the timer when `benchmark_reg[0] == 0`

iconv2d



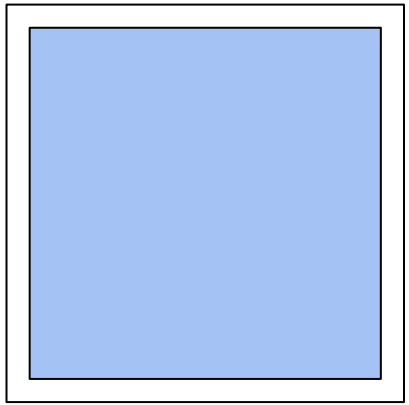
Filter (3x3, 5x5, 7x7)



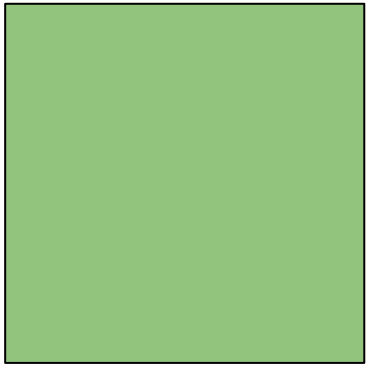
Padded input image (up to 128x128)

Unpadded output image

iconv2d



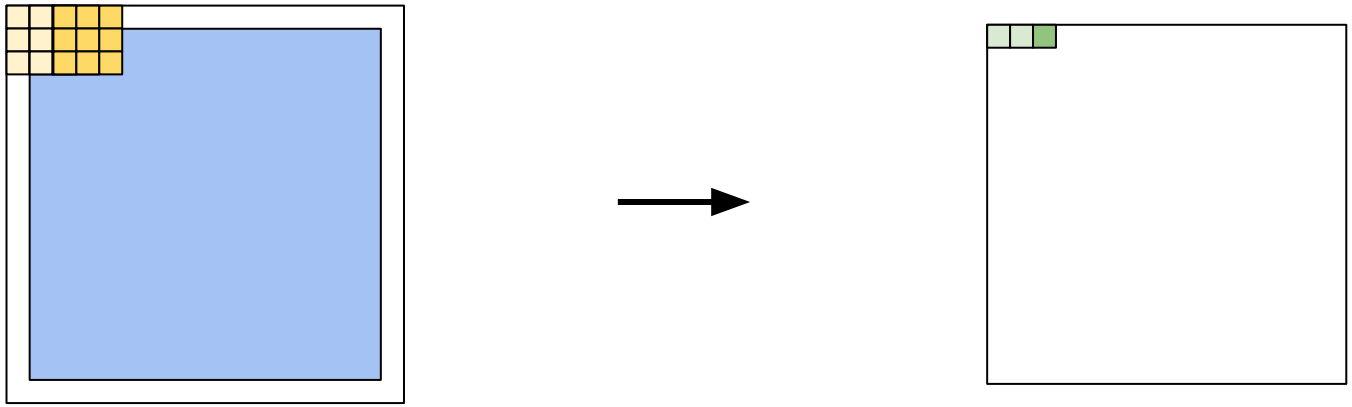
Filter (3x3, 5x5, 7x7)



Padded input image (up to 128x128)

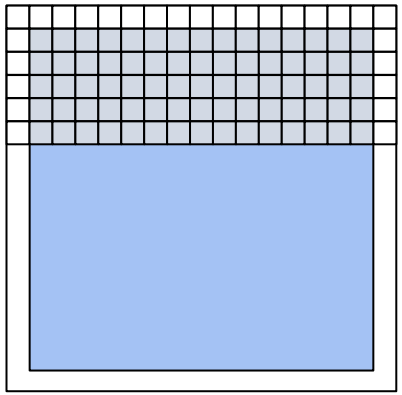
Unpadded output image

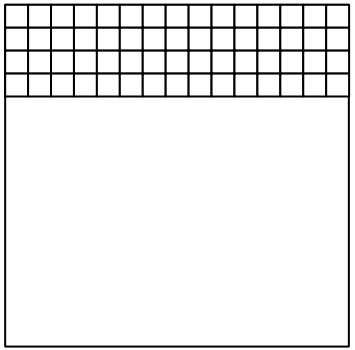
iconv2d - concept



iconv2d - vector (3x3 filter)

Every iteration:





Work on 6 input rows

Work on 4 output rows

iconv2d - vector (3x3 filter)

Every iteration:

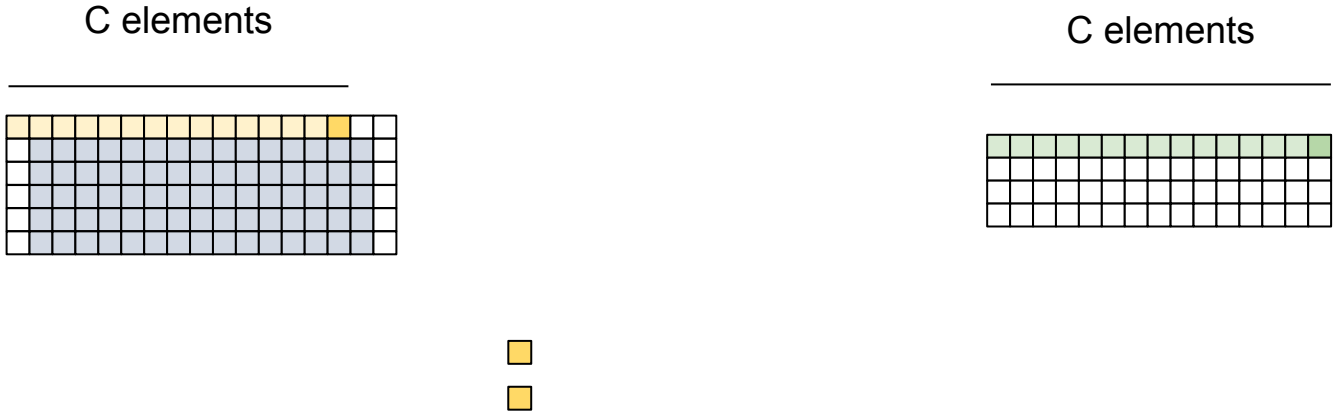
Fetch the first column of the Filter

iconv2d - vector (3x3 filter)

Every iteration:

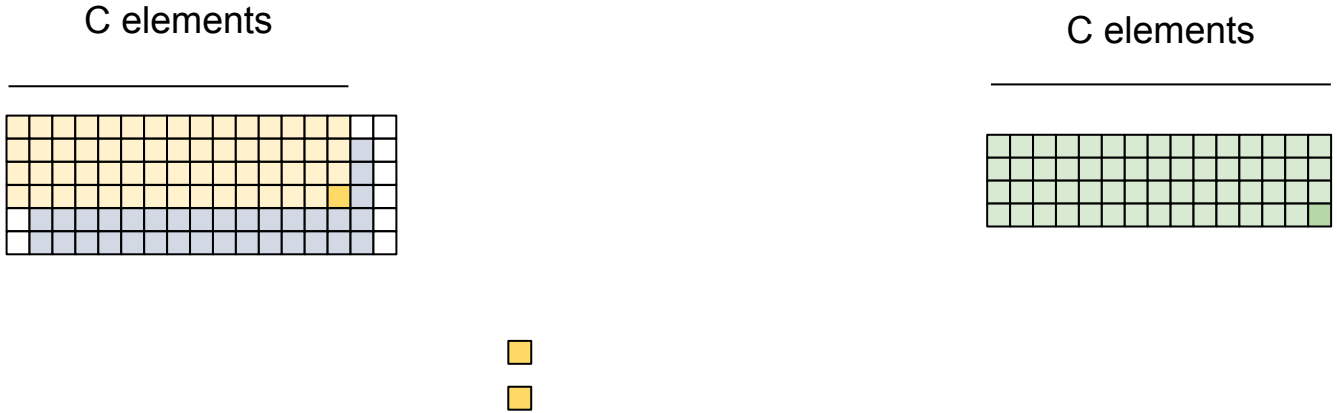
Each filter element is multiplied with four input vectors, to generate the output vectors

iconv2d - vector (3x3 filter)



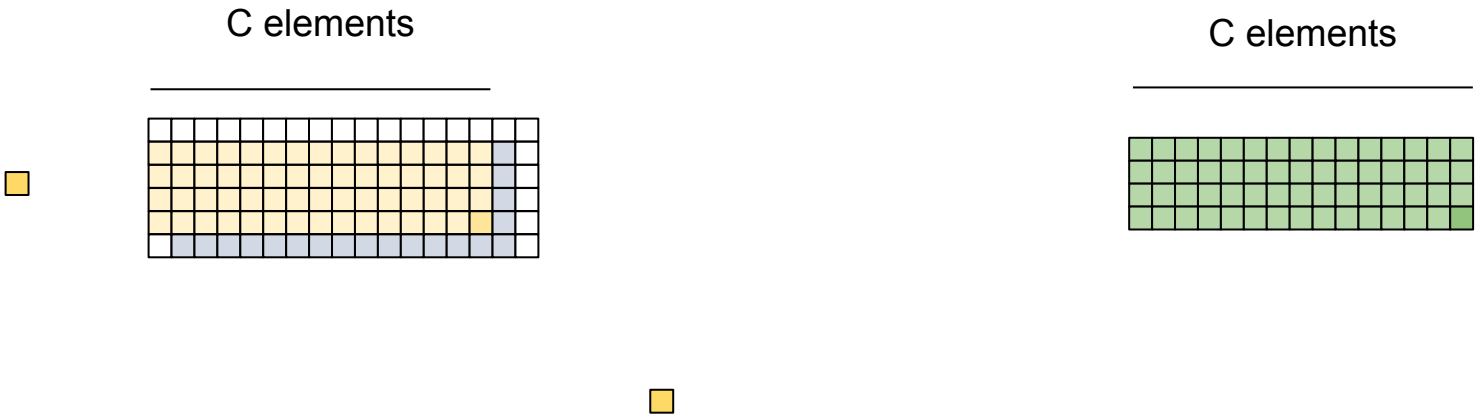
Each filter element is multiplied with four input vectors, to generate the output vectors

iconv2d - vector (3x3 filter)



Each filter element is multiplied with four input vectors, to generate the output vectors

iconv2d - vector (3x3 filter)

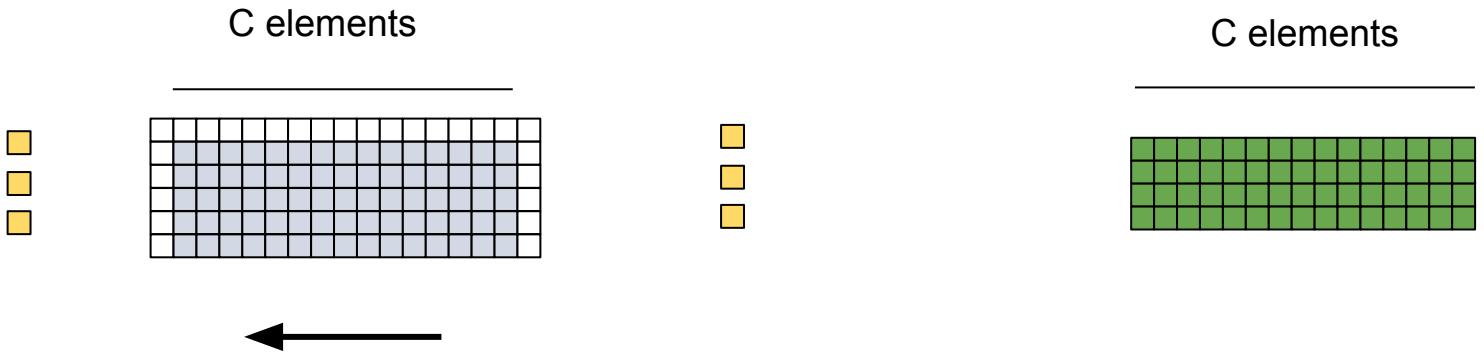


Each filter element is multiplied with four input vectors, to generate the output vectors

iconv2d - vector (3x3 filter)

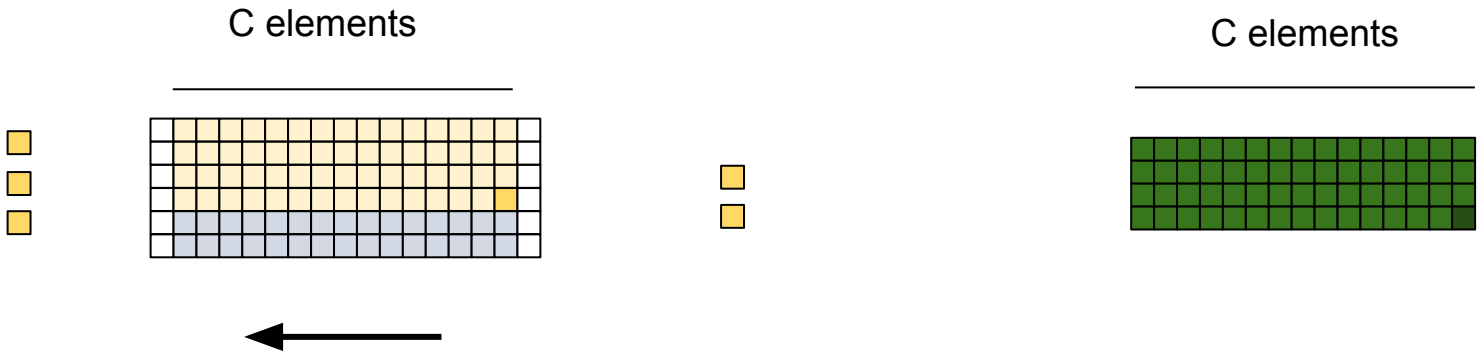
Each filter element is multiplied with four input vectors, to generate the output vectors

iconv2d - vector (3x3 filter)



Then, the input vectors are slided down by one, and the second column of the filter is fetched

iconv2d - vector (3x3 filter)



The FMAs are repeated for all the columns of the filter to generate the four output rows.
Every time a new column is fetched, the input vectors are slided down again.

iconv2d - vector (3x3 filter)

When one full iteration is over, two input rows are re-used and four input rows are fetched from memory

iconv2d - optimization in progress

Max performance: `8 DPOP/cycle, #(4_lanes)`

Arithmetic intensity: `(F * F / 8) > 1`

Reached: `5.35 DPOP/cycle`, `67% utilization`
We can do better!

- Hide the latency of the memory/slide/move instructions
- Hide unnecessary dependencies

Dropout (32-bit float)

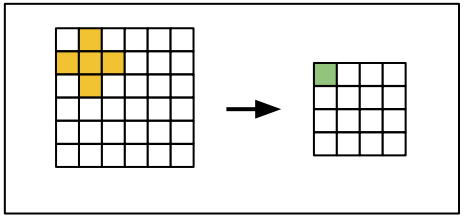
- Mixed FP/integer computation
- Count only useful computation for performance (single-precision FP)

- #Operations: `N SPFLOP`
- #B transfers: `3 * N * 4 B = 12*N B`
- Ara Memory BW: `16 B/cycle`
- Arithmetic intensity: `1/12 SPFLOP/B`
- Max Performance: `1.33 SPFLOP/cycle`
- Obtained: `1.1 SPFLOP/cycle (82% on max)`
- No FMAs (max performance is `8 SPFLOP/cycle`)

- Intrinsics: lower performance for now
- Benchmark-CSR is crucial to correctly measure the cycle count

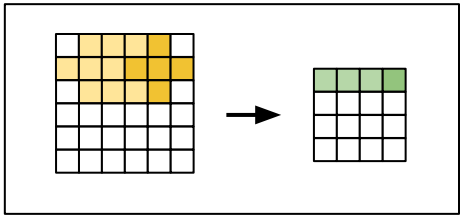
Jacobi2d

- Stencil with double precision FP



- Tot DPFLOP: `5 * N * N`
- Tot Memory transferred B: `2 * N * N * 8`

- No FMAs
- Arith intensity: 5/16
- Breakout arithmetic intensity: ¼
- Compute bound?
- Misaligned memory accesses kill memory BW
- Performance: 0.99 DPFLOP/cycle (25% on total)



RoI Align

- Different Region of Interests (RoIs) in a batch of images
- Crop each RoI to a fixed-size small matrix
- Divide each RoI in a small number of boxes
- Sample points in each box, and apply bilinear interpolation on them
- The found value is one element of one of the output matrices
- https://github.com/pulp-platform/ara/blob/benchmark.iconv2d%2Cdropout%2Cjacobi2d%2Croialign/apps/roi_align/main.c

RoI Align

- Ported the first scalar version
- Provided the first naive vectorized version
- Exploration on how to parallelize it
- Parallelize on different RoI/different images?
- Naive implementation: parallelize on channels (number of expected channels? 3?)
- Cannot parallelize on every dimension

Further - Benchmarks

- Implement new benchmarks from list
- Optimize RoI align
- Include performance plot for each benchmark
- Maximize performance for available benchmarks to explore architectural limits