MAESTRO Directives & Examples

Prasanth Chatarasi, PhD student, Habanero Research Group (Prof. Vivek Sarkar), Georgia Institute of Technology
Revisiting Spatial Dataflow Accelerator Model

- L1/L2 are scratchpad buffers but not caches!
- Assumption: Entire data required for a layer computation fits into L2 buffer
What factors influence energy and runtime?

Shared Buffer (L2 Buffer)

Network-on-Chip (NoC)

PE 0
Private L1
ALU

PE 1
Private L1
ALU

PE N-1
Private L1
ALU

Energy

Runtime
What factors influence energy and runtime?

DNN Layer Sizes

Mapping (Dataflow)

HW Resources

Energy

Runtime
MAESTRO: Analytical Cost/Benefit Model

DNN Layer Sizes

HW Resources

Mapping (Dataflow)

Buffer Analysis
- Size Requirement
- Access Count (Energy)

NoC Analysis
- BW Requirement
- NoC Activity Count

Runtime Analysis
- Roofline Throughput
- Expected Runtime

Abstract HW Model

Data Reuse Analysis

Communication Analysis

Computation Analysis
Input specification to MAESTRO

DNN Layer Sizes

HW Resources

Mapping (Dataflow)

//Layer Description
1 | //Layer Description
2 | Layer CONV VGG16_C1
3 | K=64;C=3;R=3;S=3;Y=224;X=224
4 | endLayer

//Hardware Resource Description
1 | //Hardware Resource Description
2 | L1Size 64
3 | L2 Size 1024
4 | NoCBW 64
5 | Multicast True
6 | NumPEs 256

//Mapping (Dataflow) Description
1 | //Mapping (Dataflow) Description  ???
1D Convolution -- Weight stationary

```
for(int x = 0; x < X'; x++)
for(int s = 0; s < S; s++)
    Output[x] += Weight[s] * Input[x+s]
```

Loop-nest representation for weight-stationary

```
for(int x2 = 0; x2 < X'2; x2++)
    for(int s2 = 0; s2 < S2; s2++)
        parallel_for(int x1 = 0; x1 < 1; x1++)
            parallel_for(int s1 = 0; s1 < 3; s1++)
                for(int x0 = 0; x0 < X'0; x0++)
                    for(int s0 = 0; s0 < S0; s0++)
                        x = f(x2, X'2, x1, 1, x0, X'0);
                        s = f(s2, S2, s1, 3, s0, S0);
                        Output[x] += Weight[s] * Input[x+s]
```

Accessed Weights

Accessed data in PE0

Loop nest representation as per in Eyeriss V2 (https://arxiv.org/pdf/1807.07928.pdf)
Weight stationary – Data movement

Spatial Dimension (PEs)

Temporal Dimension

PE0
PE1
PE2

O[0..2]
W[0]
I[0..2]
O[3..5]
W[0]
I[3..5]

O[0..2]
W[1]
I[1..3]
O[3..5]
W[1]
I[4..6]

O[0..2]
W[2]
I[2..4]
O[3..5]
W[2]
I[5..7]

I[3..5]

I[4..6]

I[5..7]
Weight stationary – Data movement

- Weight is spatially distributed across PE’s for parallelization
- Output is replicated to all PE’s in a given time step, but temporally distributed across time
1) Spatial_Map (size, offset) \( d \)

- Spatially distributing indices of the dimension \( d \) of an array across PE’s
  - Size – Number of indices mapped on a PE
  - Offset – Indices offset between neighbor PE’s

\[ \text{Spatial_Map}(\text{size} = 1, \text{offset} = 1) \quad d \]

where \( d \) refers to first dimension of the array \( W \)

- If PE’s are not sufficient to cover all indices, then rest of indices will be temporally folded over PE’s
2) Temporal_Map (size, offset) \( d \)

• Temporally distributing indices of the dimension \( d \) of an array to a PE across time
  • Mapping same indices to all PE’s in an iteration
  • Offset – Offset between indices in consecutive iterations

\[
\begin{align*}
\text{PE0:} & \quad \{O[0..2], O[3..5]\} \\
\text{PE1:} & \quad \{O[0..2], O[3..5]\} \\
\text{PE2:} & \quad \{O[0..2], O[3..5]\}
\end{align*}
\]

Where \( d \) refers to first dimension of the array \( O \)
Weight stationary – Data movement

- Weight is spatially distributed across PE’s for parallelization
  - $\rightarrow$ Spatial_Map(1,1) S

- Output is replicated to all PE’s in a given time step, but temporally distributed across time
  - $\rightarrow$ Temporal_Map(3,3) X
Weight stationary – Data movement order

Spatial Dimension (PEs)

Data movement Order: $S \rightarrow X'$

Spatial_Map(1,1) $S$
Temporal_Map(3,3) $X'$

$t = 0$

PE0
PE1
PE2

$O[0..2]$
$O[0..2]$
$O[0..2]$

$W[0]$
$W[1]$
$W[2]$

$I[0..2]$
$I[1..3]$
$I[2..4]$

$I[3..5]$
$I[4..6]$
$I[5..7]$

$t = 1$

$O[0..2]$
$W[2]$
$I[2..4]$

$I[0..2]$
$W[1]$
$I[3..5]$

$O[3..5]$
$W[2]$
$I[5..7]$

$O[3..5]$
$W[1]$
$I[4..6]$

Spatial Dimension

Temporal Dimension
Let's look at different orders and parallelism!

Spatial_Map(1,1) S
Temporal_Map(3,3) X'

Temporal_Map(3,3) X'
Spatial_Map(1,1) S

Spatial_Map(1,1) X'
Temporal_Map(3,3) S

Temporal_Map(3,3) S
Spatial_Map(1,1) X'

Weight stationary
Output stationary
Output stationary
Weight stationary
Representation of a dataflow

• In the output-centric notation of CNN’s, loop variables directly denote the dimensions of both output and weights.

• Data distribution of each dimension of both output & weight array should be specified if it is output-centric notation
  • E.g., Spatial_Map(size = 3, offset = 3) X’ → First dimension of Output
  • E.g., Temporal_Map(size = 3, offset = 3) S → First dimension of weight
  • If a dimension of two arrays are same (e.g., Filters for weight and Channels of output), then their distribution should be same!

• Data movement order is via the order of mappings
  • Size < Offset → Missing partial sums in the total computations
6D convolution – Sample weight stationary

Temporal_Map(1,1) K
Temporal_Map(1,1) C
Temporal_Map(3,1) Y
Spatial_Map(3,1) X
Temporal_Map(3,3) R
Temporal_Map(3,3) S
How to model the following dataflow?
Break
How to model the following dataflow?

Weights are not spatially distributed for all PE’s → No spatial!
Also, weights are not replicated for all PE’s → No temporal!
How we about we cluster?

Weights are spatially distributed within a cluster → Spatial within a cluster
Weights are replicated across clusters → Temporal across clusters
3) Cluster (size, type)

• Logical grouping of PE’s
  • Size – Number of PE’s or sub clusters to be grouped
  • Type – Elements of clusters are PE’s or logical clusters

• Used to specify different data distributions
  • Across clusters,
  • Within a cluster
Data-centric representation

Weights are replicated and inputs are spatially distributed across clusters.
Weights and inputs are spatially distributed within a cluster!

Spatial_Map(size=3, offset=3) X’,
Temporal_Map(size=9, offset=9) S
Cluster(size = 3, Type = PE’s)
Spatial_Map(size=3, offset=3) X’,
Spatial_Map(size=3, offset=3) S
Describing Dataflows in data-centric representation

- **Directives**
  - Temporal Map
  - Spatial Map
  - Cluster (PE grouping)

```c
for(int k = 0; k < K; k++)
  for(int c = 0; c < C; c++)
    for(int y = 0; y < Y; y++)
      for(int x = 0; x < X; x++)
        for(int r = 0; r < R; r++)
          for(int s = 0; s < S; s++)
            Output[k][y-r][x-s] += Weight[k][c][r][s] * Input[c][y][x]
```

Loop order can be changed depending on dataflow

---

NVDLA Accelerator

Global Buffer

PE Array

$K \times C$ MACs:
The number of kernels and input channel size must match array, otherwise MACs idle
Range:
- $C$ (16 to 128)
- $K$ (4 to 16)

Image Source: Nvidia

Released Sept 29, 2017

http://nvdla.org
NVDLA Accelerator (PE Array)

Number of Kernels (i.e. Filters) ‘Atomic-K’

Number of Input Channels ‘Atomic-C’
6D Convolution with NVDLA

- Temporal_Map(3,3) R
- Temporal_Map(3,3) S
- Temporal_Map(64,64) C
- Temporal_Map(1,1) X
- Temporal_Map(1,1) Y
- Cluster(64, PE)
- Spatial_Map(1,1) K
Input specification to MAESTRO

DNN Layer Sizes

HW Resources

Mapping (Dataflow)

1 | //Layer Description
2 | Layer CONV VGG16_C1
3 | K=64;C=3;R=3;S=3;Y=224;X=224
4 | endLayer

1 | //Hardware Resource Description
2 | L1Size 64
3 | L2 Size 1024
4 | NoCBW 64
5 | Multicast True
6 | NumPEs 256

1 | Temporal_Map(1,1) K
2 | Temporal_Map(1,1) C
3 | Temporal_Map(3,1) Y
4 | Spatial_Map(3,1) X
5 | Temporal_Map(3,3) R
6 | Temporal_Map(3,3) S
Demo
Summary

• Data-centric representations to represent dataflows
  • Spatial_Map
  • Temporal_Map
  • Cluster

• DNN model, dataflow, and hardware configuration is fed to MAESTRO to compute cost in terms of execution time and energy
Let's model Full Eyeriss dataflow now?

(b) Example dataflow

(c) Visualized data mapping over time