Marvel: A Heuristic-based Mapper for Mapping Space Exploration of DNN Operators onto Spatial Accelerators Using MAESTRO

Prasanth Chatarasi

MAESTRO Tutorial - MICRO 2020

October 17th, 2020

Georgia Tech School of Computer Science

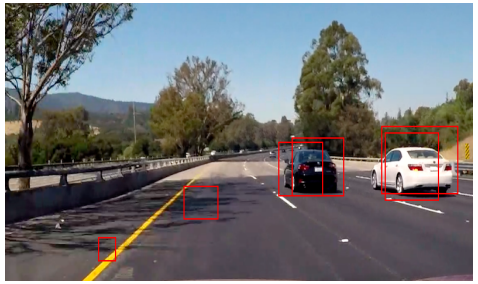
SYNERGY

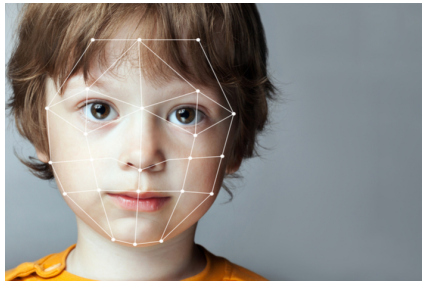
NVIDIA

"Marvel: A Data-centric Compiler for DNN Operators on Spatial Accelerators"

Prasanth Chatarasi, Hyoukjun Kwon, Natesh Raina, Saurabh Malik, Vaisakh Haridas, Angshuman Parashar, Michael Pellauer, Tushar Krishna, and Vivek Sarkar (ArXiv'20)

Deep Learning (DNN Models)

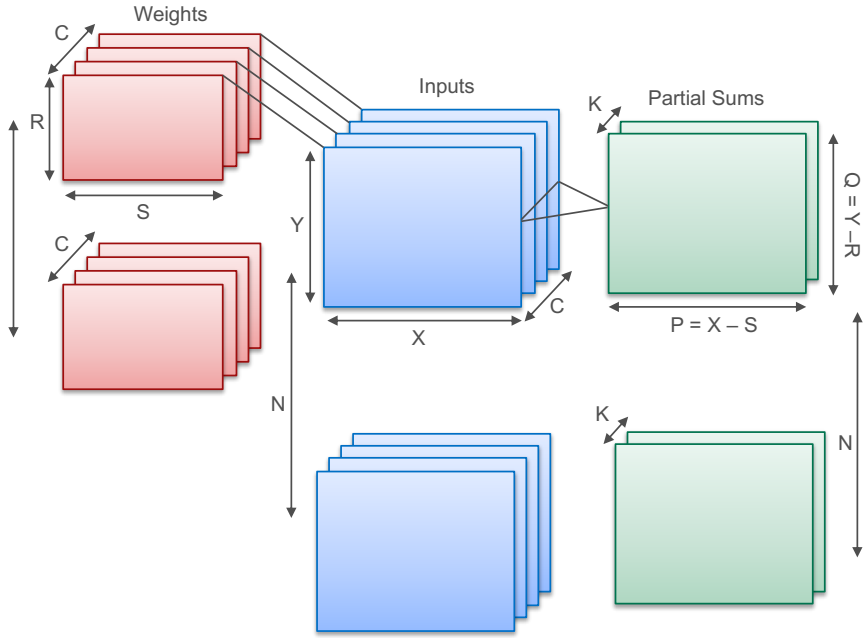





Examples of DNN Operators (Layers)

- Regular CONV1D
- Regular CONV2D
- Depth-wise CONV2D
- Transposed CONV2D
- Regular CONV3D
- Strided variants
- GEMM (MatMul)
- LSTM (RNNs)
- Element-wise
- Pooling
- Fully Connected/MLP
- .....

Regular CONV2D over 4D Tensors



Involves billions of computations

Parashar et al., ISPASS 2019

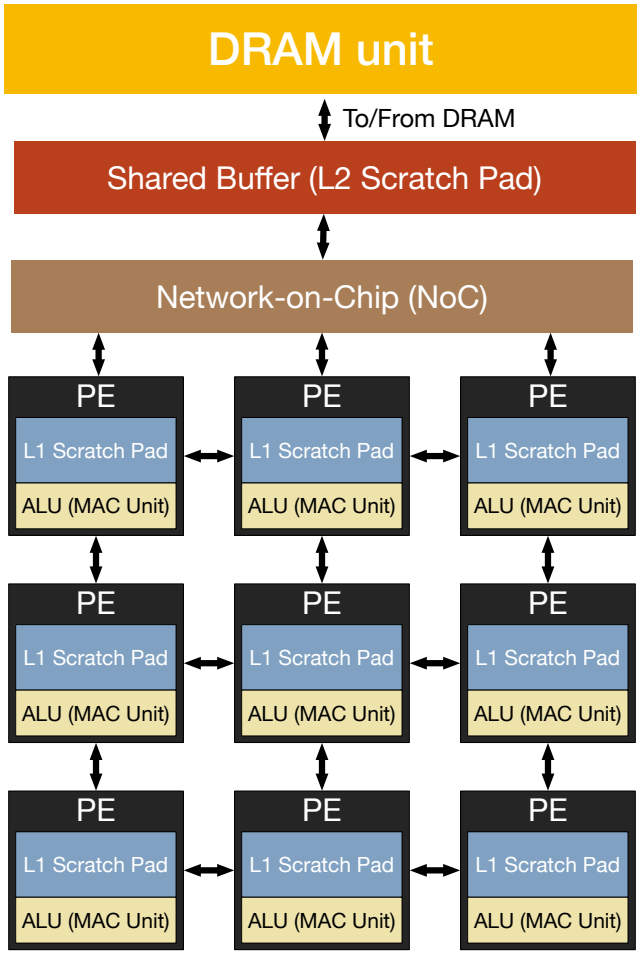
Mapping problem

DNN Operators

- Regular CONV1D
- Regular CONV2D
- Depth-wise CONV2D
- Transposed CONV2D
- Regular CONV3D
- Strided variants
- GEMM (MatMul)
- LSTM (RNNs)
- Element-wise
- Pooling
- Fully Connected/MLP
- …..

How to map for low latency, high energy efficiency?

Abstract overview



3-level accelerator

E.g., TPU, Eyeriss, NVDLA

Mapping involves

1) Parallelization onto compute resources,

2) Tiling across memory resources, and

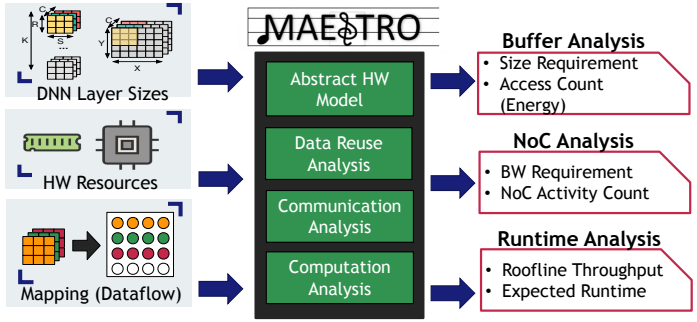
3) Exploitation of data reuse

Challenges

1. **Explosion of hardware choices in spatial accelerators**
   - Wide variety of hardware structures & data movement restrictions

2. **Rapid emergence of new DNN operators and shapes/sizes**
   - Various forms of algorithmic properties (e.g., reuses)

3. **Selection of optimized mapping from massive mapping space and also good cost models**
   - E.g., On average, O(10^{18}) mappings for CONV2D in MobileNetV2



"Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach"
Hyoukjun Kwon, **Prasanth Chatarasi**, Michael Pellauer, Angshuman Parashar, Vivek Sarkar, and Tushar Krishna,
In Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO'19)

Mapping space for a 3-level accelerator

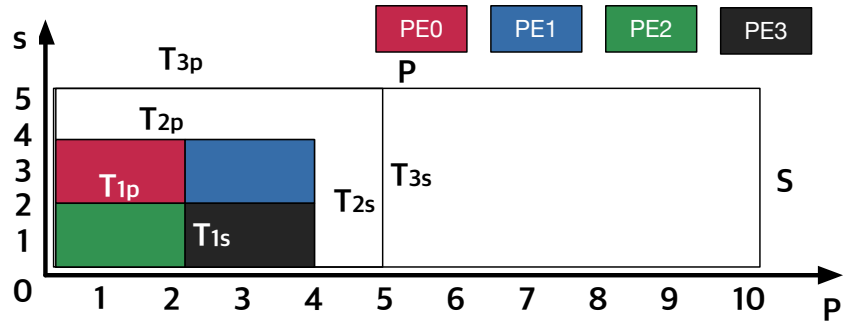
- **Multi-level tiling for memory hierarchy and for parallelization**
  1. Level-1 tiling for the L1 buffer
  2. Level-2 tiling for the PE array
  3. Level-3 tiling for the L2 buffer
- **Loop orders across tiles**
  4. Inter-tile level-3 loop order
  5. Inter-tile level-2 loop order
- **Data-layouts**
  6. Tensors on DRAM
- **Mapping is an unique 6D tuple in the 6-dimensional search space**

```
for(p=1; p < P; p++)
  for(s=1; s < S; s++)
    Output[p] += Weight[s] * Input[p+s]
```

(a) Plain 1D Convolution

```
for(t3p=0; t3p < P; t3p += T3p)                    ]  Level 3 Inter-tile loops
  for(t3s=0; t3s < S; t3s += T3s)                  ]
Lv3 tile ----------------------------------------------
    for(t2p=t3p; t2p < t2p+T3p; t2p+=T2p)          ]  Level 2 Inter-tile loops
      for(t2s=t3s ; t2s < t2s+T3s; t2s+=T2s)       ]
Lv2 tile ----------------------------------------------
        parallel_for(t1p=t2p; t1p < t1p + T2p; t1p += T1p)
          parallel_for(t1s=t1s; t1s < t1s + T2s; t1s += T1s)
Lv1 tile ----------------------------------------------
            for(t0p=t1p; t0p < t0p + T1p; t0p +=1)
              for(t0s=t1s; t0s < t0s + T1s; t0s +=1)
                Output[t0p] += Weight[t0s] * Input[t0p + t0s]
```

(b) Tiled 1D Convolution

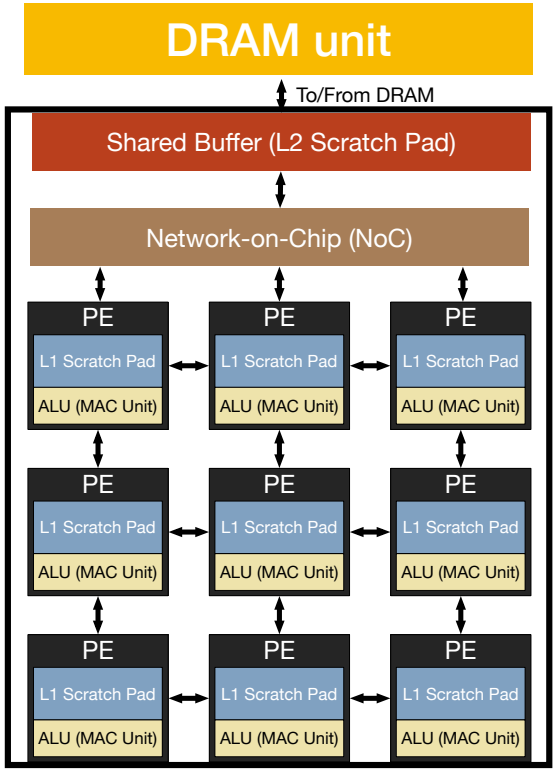


(c) Tiling Example

$O(10^{18})$ mappings on average for a single convolution layer in ResNet50 and MobileNetV2 models on Eyeriss-like accelerator

Our Intuition

Observation: Off-chip data movement is 2-3 orders of magnitude more expensive compared to on-chip data movement



Accessing DRAM unit: ~200x

Accessing L2 buffer: ~6x

Accessing non-local L1 buffer: ~2x

Accessing local L1 buffer: ~1x

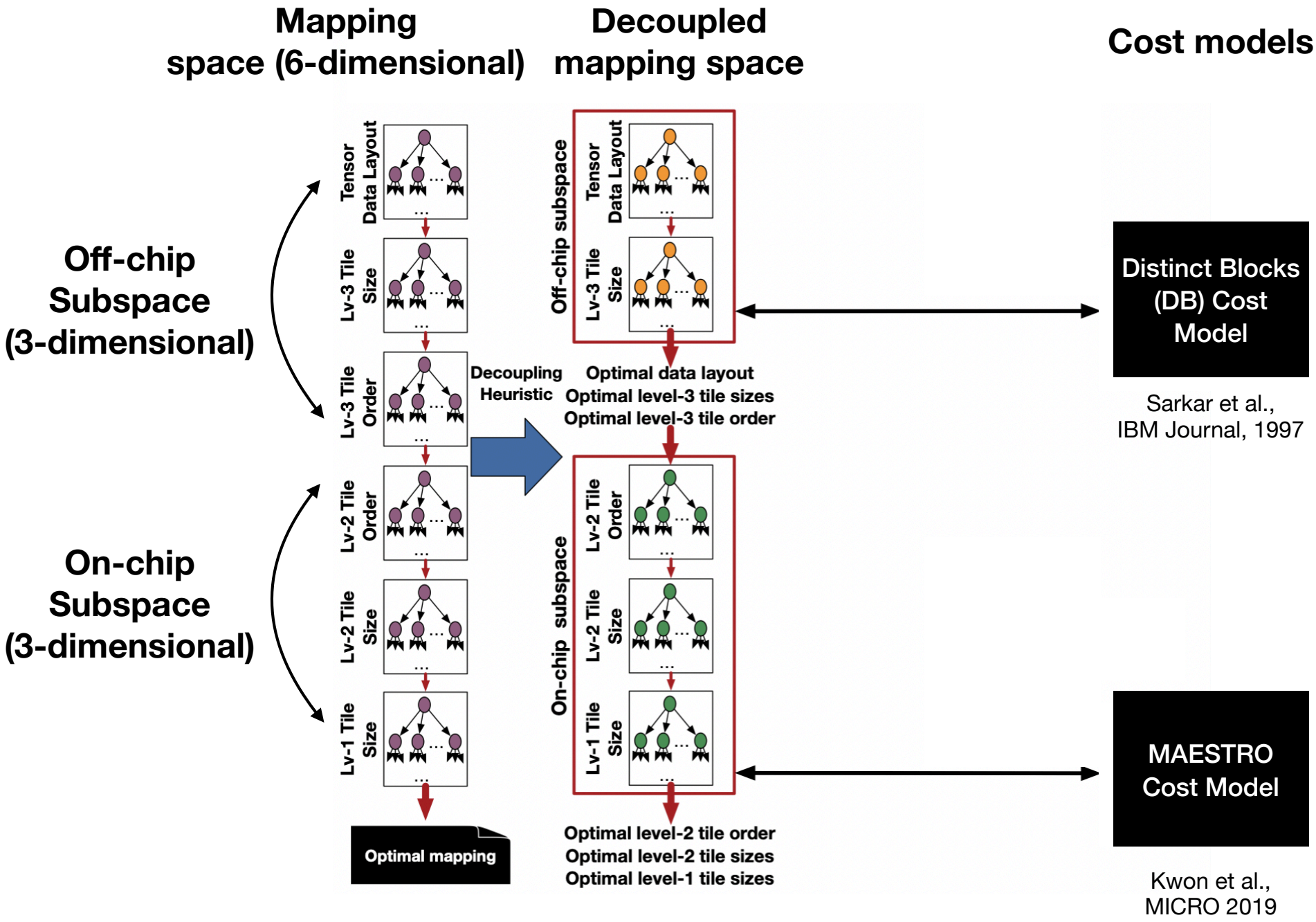
Compute: 1x

Data movement energy

Idea: Decouple the mapping space based on off-chip and on-chip data movement, and prioritize optimizing for off-chip data movement first?

Vivienne et al., Deep Learning Tutorial

Our approach (Marvel)



Step-1: Optimizing off-chip subspace

- **Input:** Workload, hardware configuration, and exploration options
- **Output:** Level-3 tile sizes & inter-tile order, and data-layouts
- **Distinct Blocks Model (DB Model)**
  - Given a parametric loop-nest and layout of tensors, the model measures distinct number of DRAM blocks for a computation tile

```
for(n=0; n<N; n++)
 for(k=0; k<K; k++)
  for(c=0; c<C; c++)
   for(p=0; p<P; p++)
    for(q=0; q<Q; q++)
     for(r=0; r<R; r++)
      for(s=0; s<S; s++)
       O[n][k][r][p] += W[k][c][r][s]
                      * I[n][c][q+r][p+s];
```

T_{3i} is the tile size for loop-i,
b is the DRAM block size

$$DB_W(T_3) \approx \left(\left\lceil \frac{T_{3S}}{b} \right\rceil\right) \times T_{3R} \times T_{3C} \times T_{3K}$$

$$DB_I(T_3) \approx \left(\left\lceil \frac{T_{3P} + T_{3S}}{b} \right\rceil\right) \times (T_{3Q} + T_{3R}) \times T_{3C} \times T_{3N}$$

$$DB_O(T_3) \approx \left(\left\lceil \frac{T_{3P}}{b} \right\rceil\right) \times T_{3Q} \times T_{3K} \times T_{3N}$$

$$DMC(T_3) \approx \frac{DB_W(T_3) + DB_I(T_3) + DB_I(T_3)}{T_{3N} \times T_{3K} \times T_{3C} \times T_{3X} \times T_{3Y} \times T_{3R} \times T_{3S}}$$

$$b \times DB_{Total}(T_3) \leq \frac{|L2|}{2}$$

Sarkar et al., IBM Journal, 1997

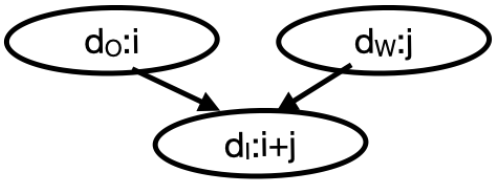
Step-2: Optimizing on-chip subspace

- **Input:** Level-3 tile sizes, Level-3 tile order, data-layouts
- **Output:** Level-2 tile sizes, Level-2 tile order, Level-1 tile sizes
- **Iterate over each on-chip mapping, translate into MAESTRO understandable format, and invoke MAESTRO cost model**

A) CONV1D operation

```
for(i = 0; i < M; i++)
  for(j = 0; j < N; j++)
    O[i] += I[i+j] * W[j]
```

B) DDG



C) A sample mapping in the loop-nest representation

Level-1 tile sizes: T_{1i}, T_{1j}
Level-2 tile sizes: T_{2i}, T_{2j}
Level-2 inter-tile order: t_{3i}, t_{3j}

```
L3 tile  for(t3i = t4i; t3i < t4i+T3i; t3i+=T2i)
         for(t3j = t4j; t3j < t4j+T3j; t3j+=T2j)
L2 tile    pfor(t2i = t3i; t2i < t3i+T2i; t2i+=T1i)
           pfor(t2j = t3j; t2j < t3j+T2j; t2j+=T1j)
L1 tile      for(t1i = t2i; t1i < t2i+T1i; t1i++)
             for(t1j = t2j; t1j < t2j+T1j; t1j++)
               O[t1i] += I[t1i+t1j] * W[t1j]
```

D) Mapping directives

#do, dw, di — index variables over tensor dimensions

#P2i = $\frac{T_{2i}}{T_{1i}}$, P2j = $\frac{T_{2j}}{T_{1j}}$

```
R4  TemporalMap(T2i,T2i) do
    TemporalMap(T2j,T2j) dw
    Cluster(1)
R3  SpatialMap(T1i,T1i) do
    TemporalMap(T2j,T2j) dw
    Cluster(P2i)
R2  TemporalMap(T1i,T1i) do
    SpatialMap(T1j,T1j) dw
    Cluster(1)
R1  TemporalMap(T1i,T1i) do
    TemporalMap(T1j,T1j) dw
```

Impact of our approach over CONV2D mapping space in Modern DNN models

Variants	Search space size		
	Min	**Avg**	**Max**
Original search space	2.7×10^{17}	9.4×10^{18}	1.8×10^{19}
Off-chip schedules search space after decoupling	7.3×10^{8}	3.6×10^{11}	1.3×10^{12}
On-chip schedules search space after decoupling	2.9×10^{7}	2.4×10^{10}	1.4×10^{11}
Off-chip schedules search space after decoupling + pruning	9.9×10^{5}	1.5×10^{8}	6.3×10^{8}
On-chip schedules search space after decoupling + pruning	3.8×10^{5}	5.9×10^{7}	2.4×10^{8}

TABLE IV: The statistics (min/avg/max) of the mapping space of convolution layers in our evaluation and the resultant mapping subspaces after decoupling and pruning strategies.

Because of our decoupled and pruning strategies, on an average, mapping space is reduced by O(10^{10})

Demo of Marvel

- **HW: 3-level edge class accelerator**
  - PEs = 1024, NoC (bi-directional) bandwidth = 25.6 GBps, L1 size = 512 Bytes, and L2 size = 108KB
- **Layer1: CONV2_2_2 (REGULAR CONV2D) from VGG16**
  - Mappings different for optimal latency, energy, and edp
  - Varying number of levels of parallelism
- **Layer2: Bottleneck6_2_2 (Depth-wise CONV2D) from MobileNetV2**
  - Mappings different for optimal latency, energy, and edp
  - Performance limited by NoC bandwidth

Summary and in-progress works

1. **Marvel — Decoupled off-chip/on-chip heuristic to efficiently explore the massive search space of mappings**
   - Reduced the search space on an average by $O(10^{10})$

2. **Existing exploration strategies in Marvel framework**
   - Off-chip/on-chip decoupling approach
   - Brute-force, Random Sampling,
   - Prior strategies for CONV2D such as Interstellar, dMazeRunner

3. **In-progress work**
   - Integration with MLIR framework
   - Integration with RL-based exploration strategies
   - Support for more DNN operators
   - Source code will be released soon!

Overall, we view Marvel as a research infrastructure to help compiler and micro-architecture research of accelerators!