ASA: Enabling DSE

Planning

Things to do (not in chro

Ideally: a DaCe program
In practice: NumPy Python Code that can be compiled to DaCe (e.g., DaCe does not support Collections and Recursion)

Use cases: be able to represent ap

- 5G use case: be more compliant with orig. appl — Needs your support
- ML: something more interesting than Lenet: Encoder. — December

Explore Application representations: be able to "compile" the application to Canonical DAGs

- Conveniently represent iterative algorithms — December
- Support considered use cases — January

Explore Architectures: be able to "consider" different macro-architectures

- We can change the number of PEs, this will affect the scheduling of the Canonical DAG — January
- Changing the PEs (but still under the homogenous PE assumption) supporting only certain type of operations — January

Space Exploration Goals and Optimization:

- Goals: optimize/minimize performance/power/area: we need way of estimating these
  - Performance is given by the scheduling makespan
  - Area: # of PEs, but also on-chip buffer space (e.g. for deadlock prevention) — First version on January
  - Power: directly proportional to the off-chip memory accesses — Then needs your support

Documentation

Explore Application Representations

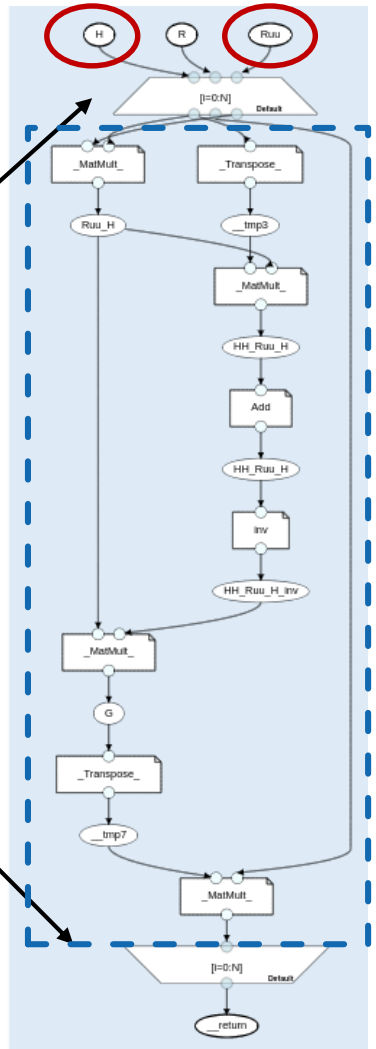
Iterative Computation – Partial Unrolling

Represent the iterative computation in DaCe, then partially unroll it in the Canonical DAG

ExDxAxU

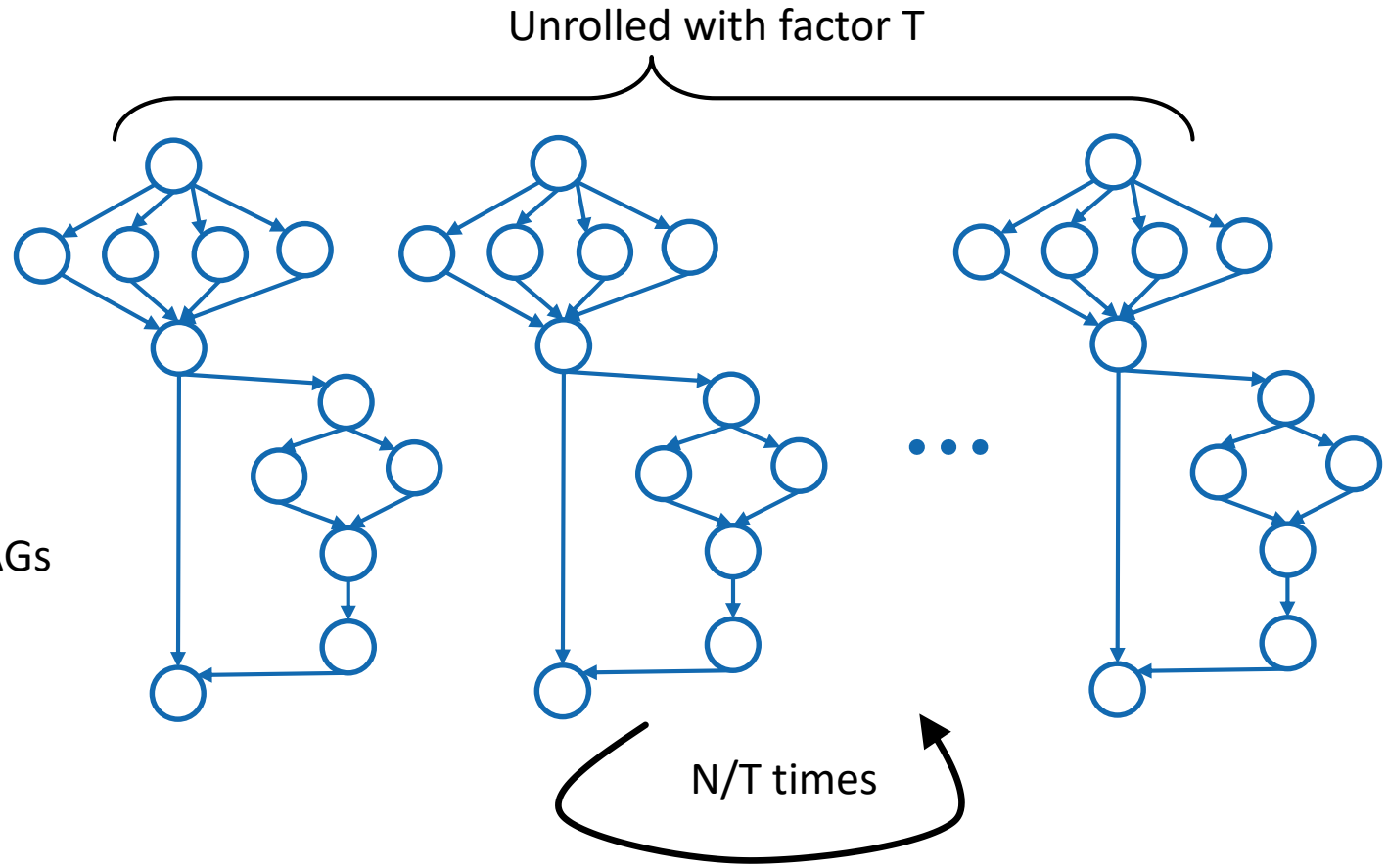
But, internally, we use single matrices

Map representing independent iterations



ExDxAxA

Canonical DAGs



Pro:

- Concise/cheap on the SDFG side
- Will allow us to capture inter-iteration parallelism

Cons: Unrolling factor is another dimension to explore (but we can keep it under control)

ASE

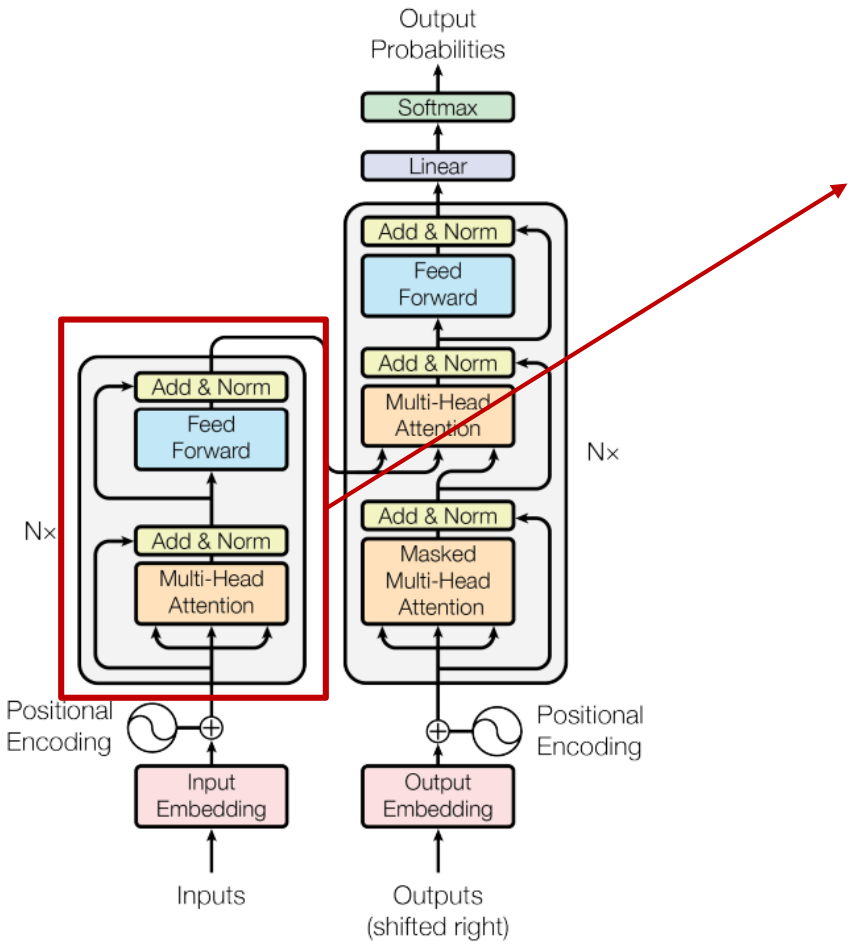
Performed Application Space Exploration + Scheduling. Partial unrolled 72 iterations (out of 32K). Analysis time: ~ 1 min per application variation

32 PEs	
Makespan (M)	I/O (M)
45.99	18.5
46.05	18.5
46.06	18.5
46.07	18
46.39	18.52
...	

What is in our experience a “reasonable” running frequency for a SoC? 1 GHz?

Use cases

ML use cases: Transformer Encoder Layer



Ashish Vaswani, et al. “Attention is All you Need”. Neurips 2017

48 ONNX operations

- MatMul: 6, (on large input matrices)
- Transpose: 8,
- Add: 10,
- Slice: 3,
- Mul: 3,
- Reshape: 4,
- Softmax: 1,
- ReduceMean: 4,
- Sub: 2,
- Pow: 2,
- Sqrt: 2,
- Div: 2,
- Relu: 1

Analyzed with the same approach

ML use cases: Transformer Encoder Layer

Preliminary results on small encoder layer (not all the expansions working ATM)

32 PEs		128 PEs	
Makespan (K)	I/O (M)	Makespan (K)	I/O (M)
1221.2	139.3	622.6	129.1
1232.3	143.3	632.8	130.3
1242	144.3	643.1	131.5
1242.6	142.1	653.3	132.5
1263.3	143.7	663.5	129.6
...		...	