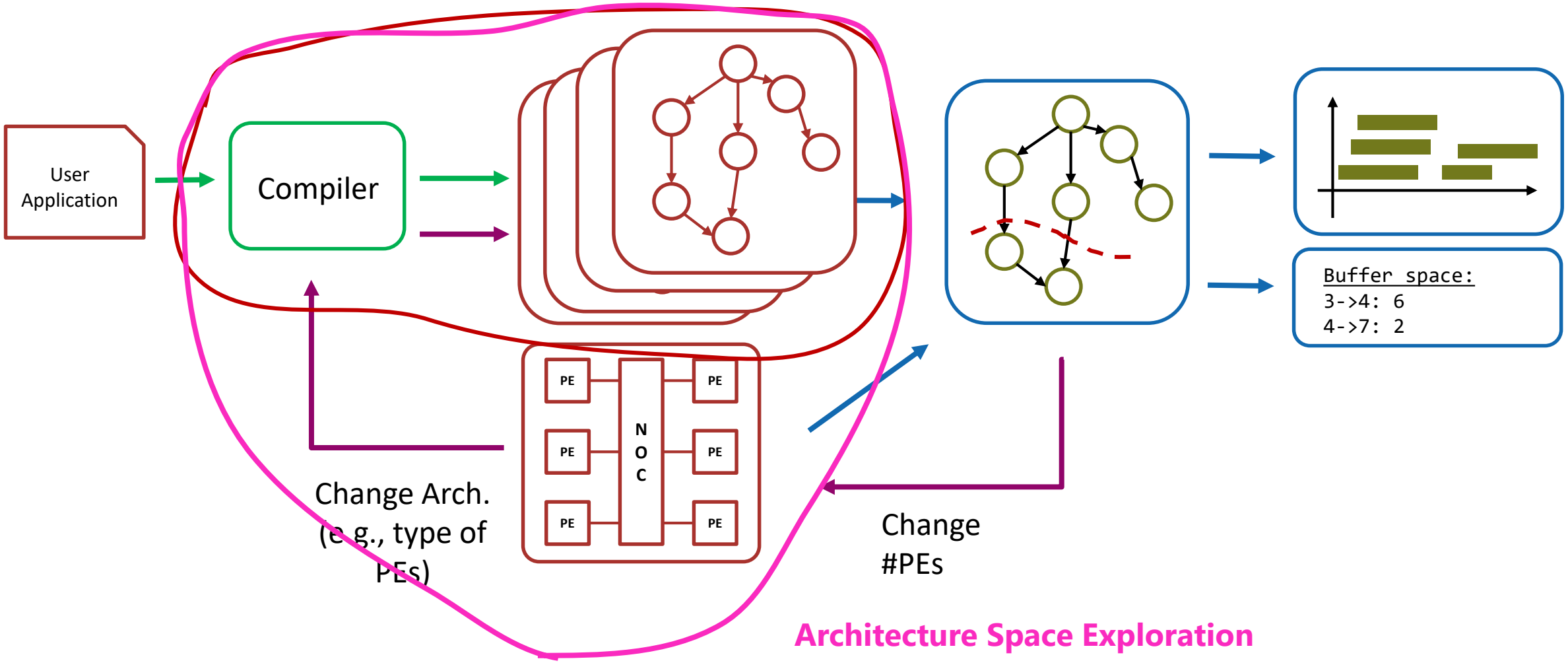
ASA: Enabling DSE

Different types of exploration

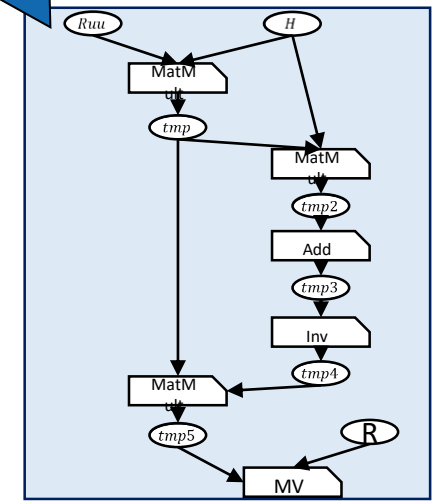


Goal: Find application representation and architecture that give max performance

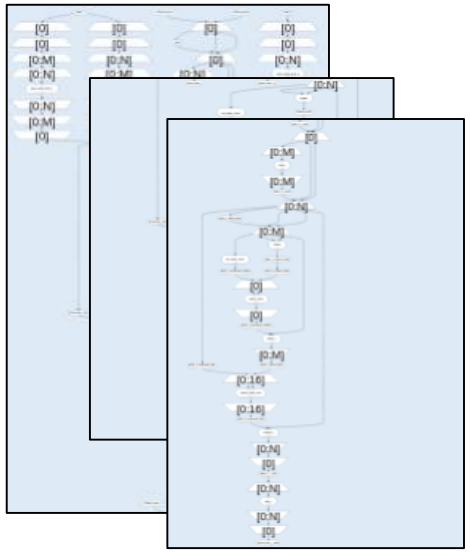
Approach

We want to use DaCe (IR, LibNode, Transformations) to enable all of this (“data-centric and compiler approach”)

User Application



Canonical Expansions



DaCe

Building Canonical DAGs

We want to build Canonical DAGs so that we can schedule on a data flow architecture (with streaming):

1. Each node receives/produces the same amount of data to/from all edges
2. We have Buffer nodes to represent non-streamable communications

We use DaCe:

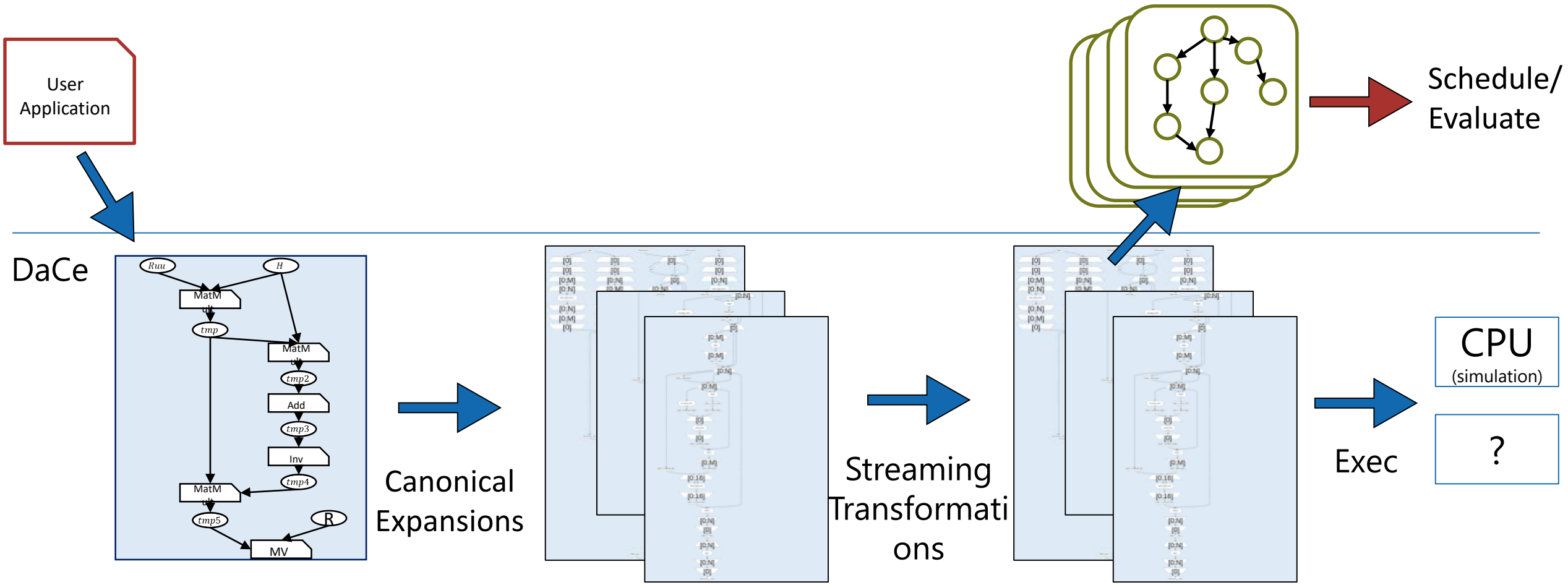
1. Proper Library Node expansions
2. … and Transformations to detect streaming/non-streaming opportunities

Note:

- Canonical DAGs are **a** way of assessing performance and schedule when running on a dataflow architecture
- DaCe by its own does not support mapping, but we will use it to enable mapping (scheduling) and, potentially, other optimization

Approach

We want to use DaCe (IR, LibNode, Transformations) to enable all of this (“data-centric and compiler approach”)

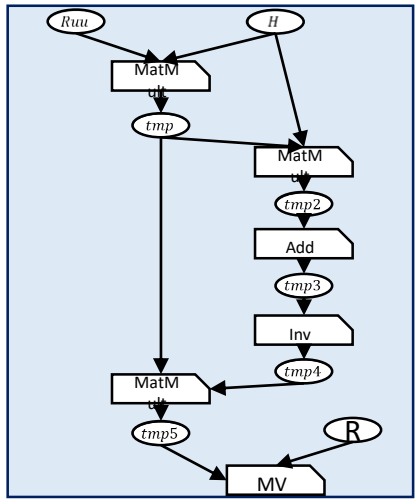


Approach

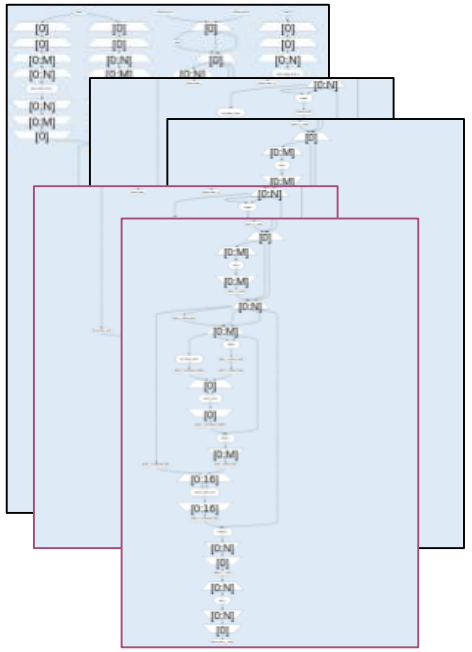
Task Granularity/Mapping? With this approach an SDFG subgraph is mapped to a task which is mapped to a PE

User Application

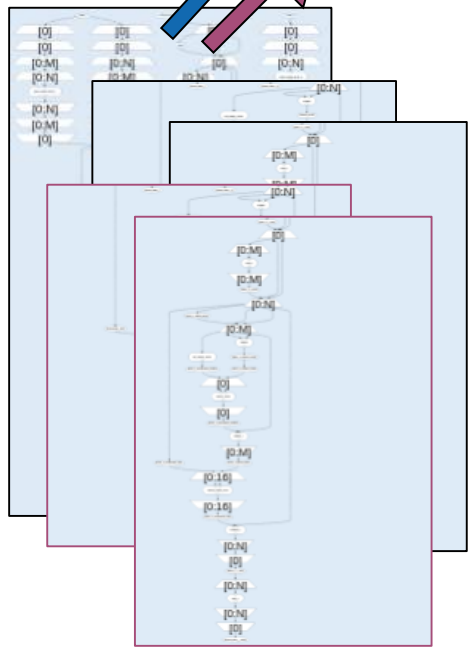
DaCe



Canonical Expansions



Streaming Transformations

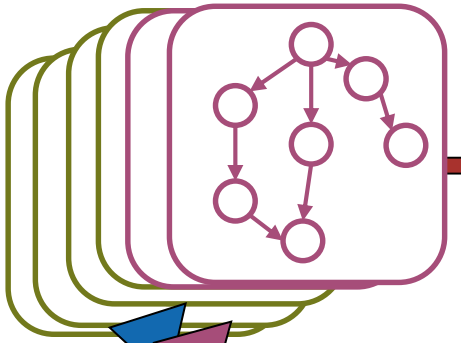


Exec

CPU (simulation)

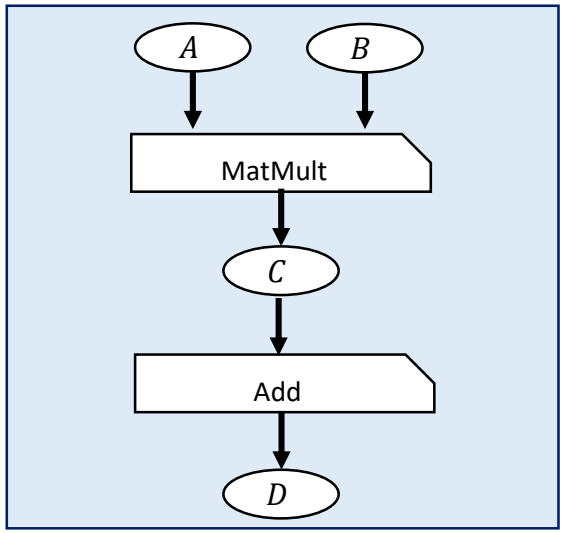
?

Schedule/ Evaluate

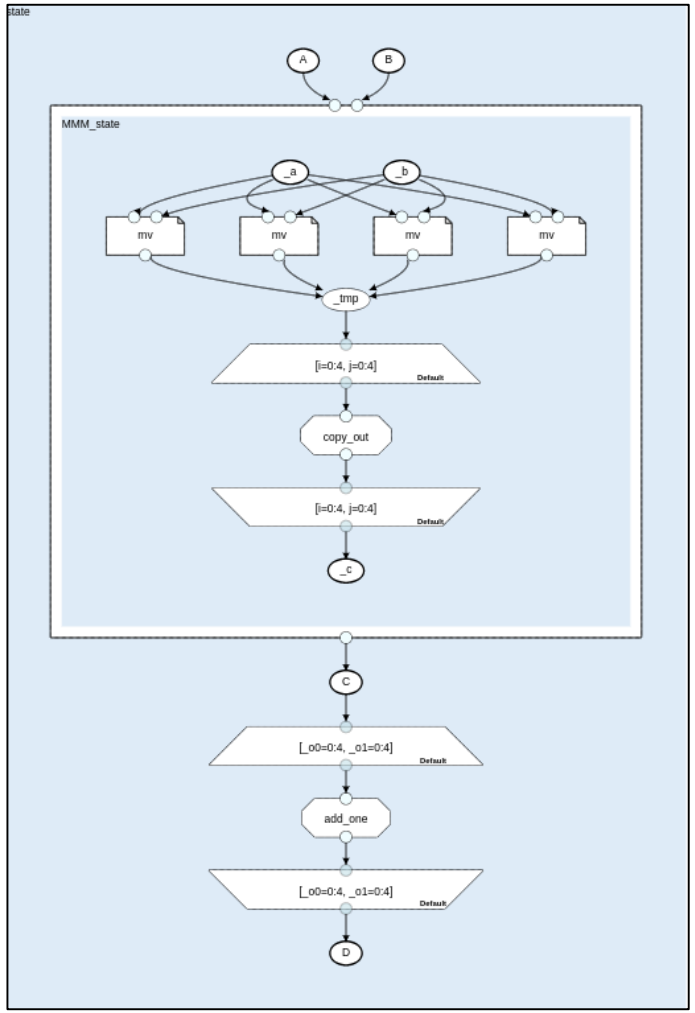


Example

For the sake of the example, let's assume that we have a MMM followed by an Addition on the result



Lowering:
MMM -> MV impl
(result produced by row)

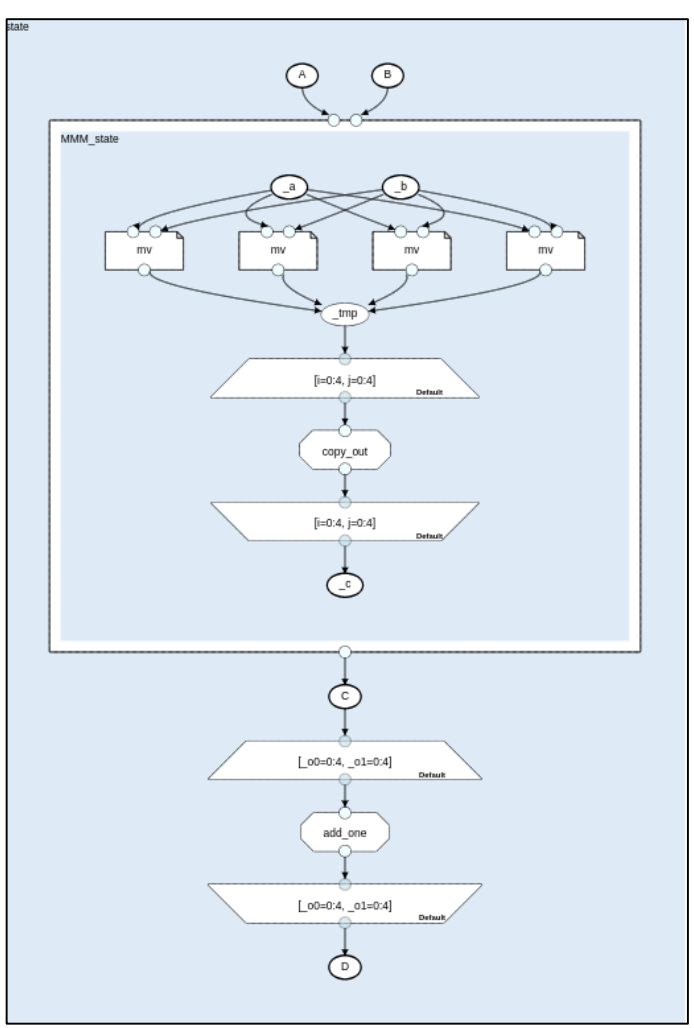


We want to:

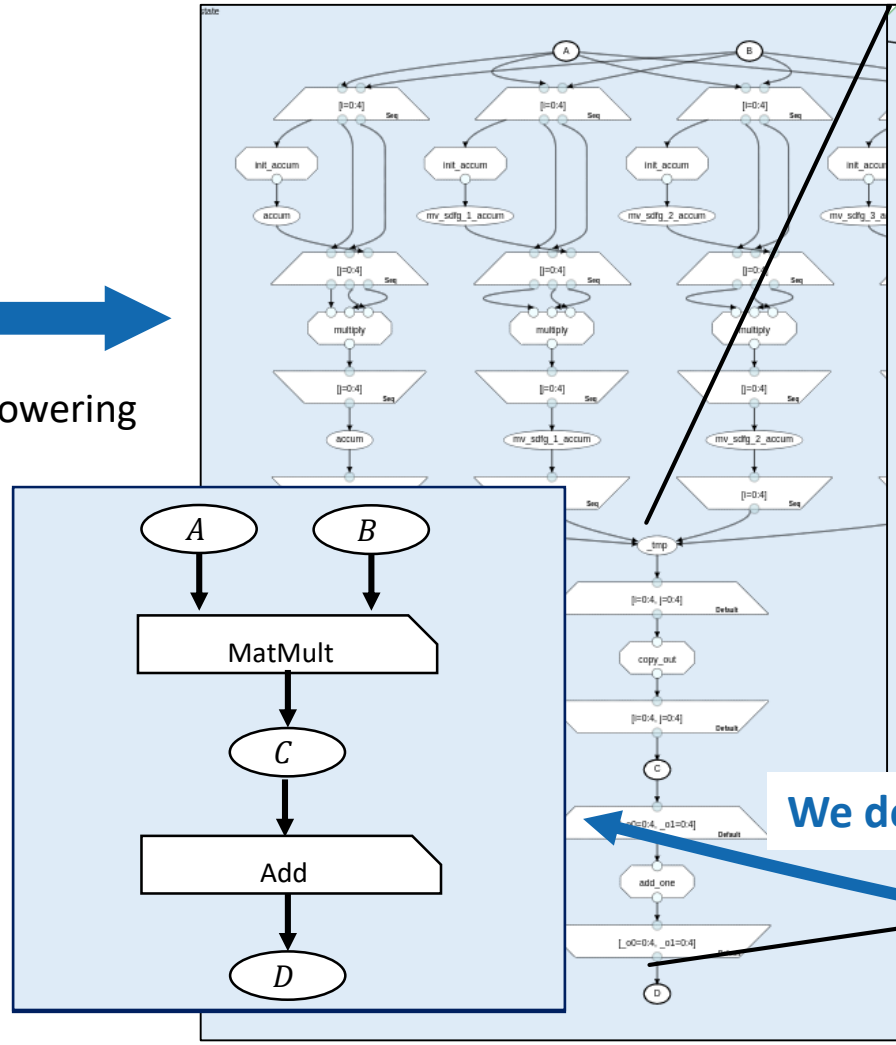
- Build the canonical DAG
- Understand if we can stream between MMM and Add

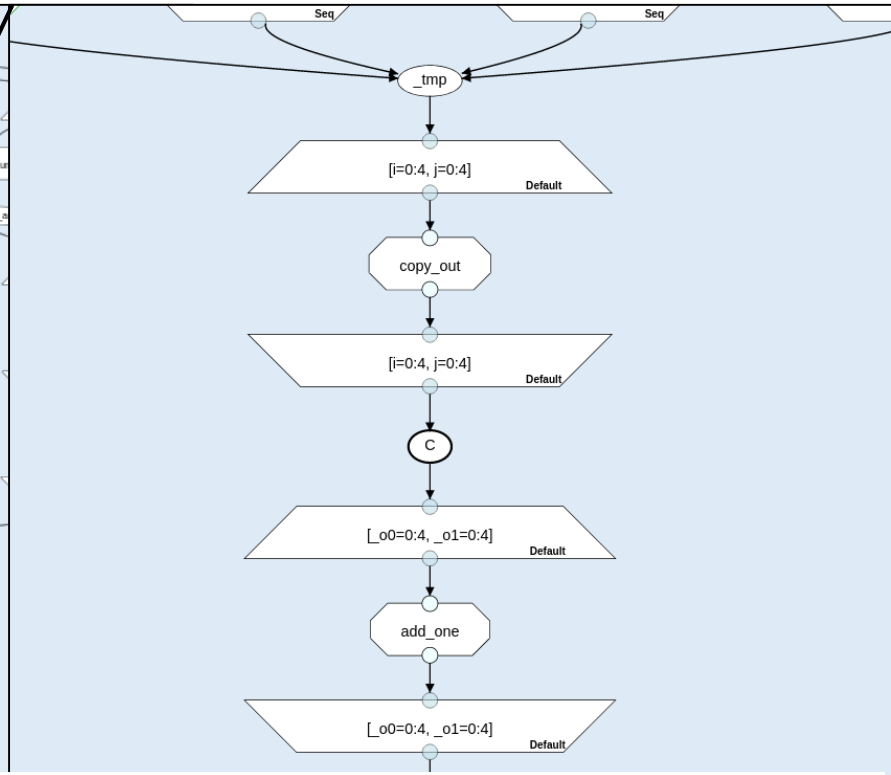
Example

For the sake of the example, let's assume that we have a MMM followed by an Addition on the result



Lowering



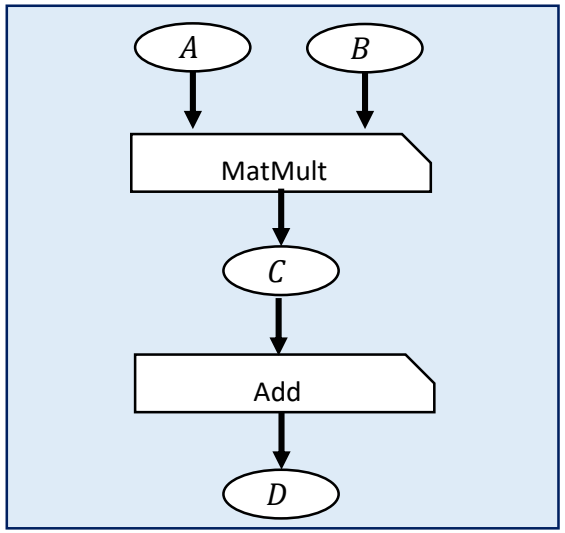


We don't need to buffer the result of the MMM

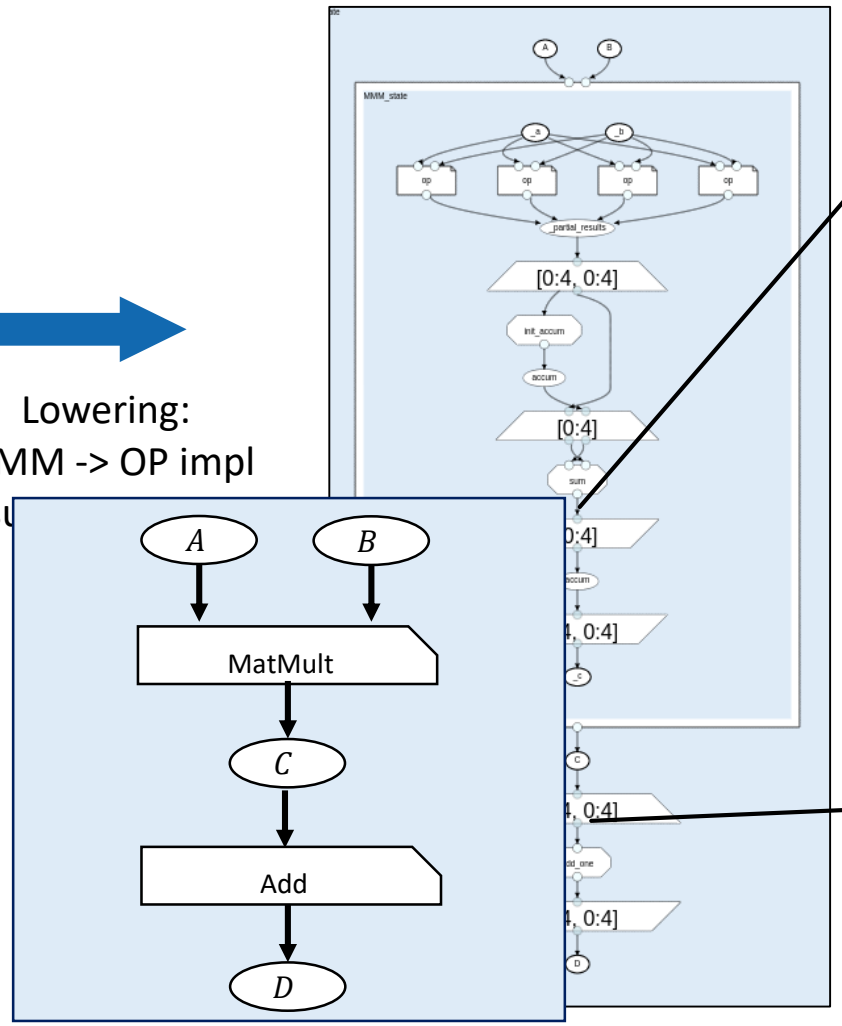
Conversion TBD

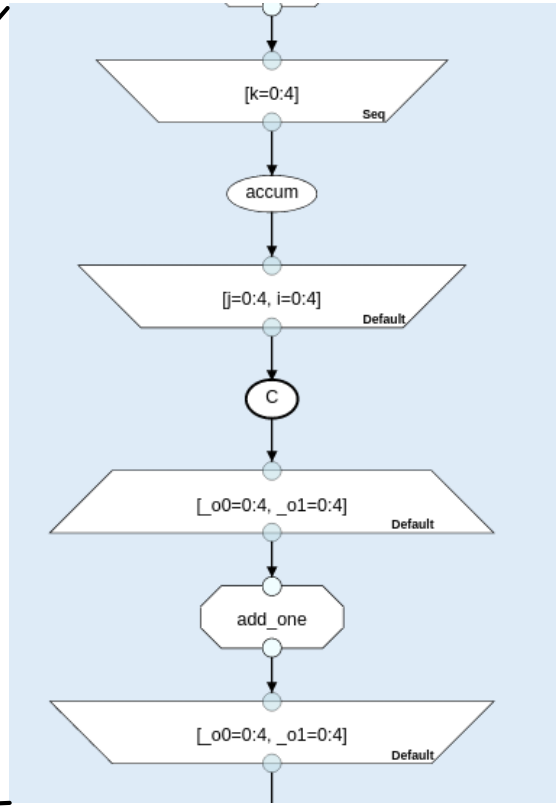
Example

For the sake of the example, let's assume that we have a MMM followed by an Addition on the result



Lowering:
MMM -> OP impl
(resu





We want to:

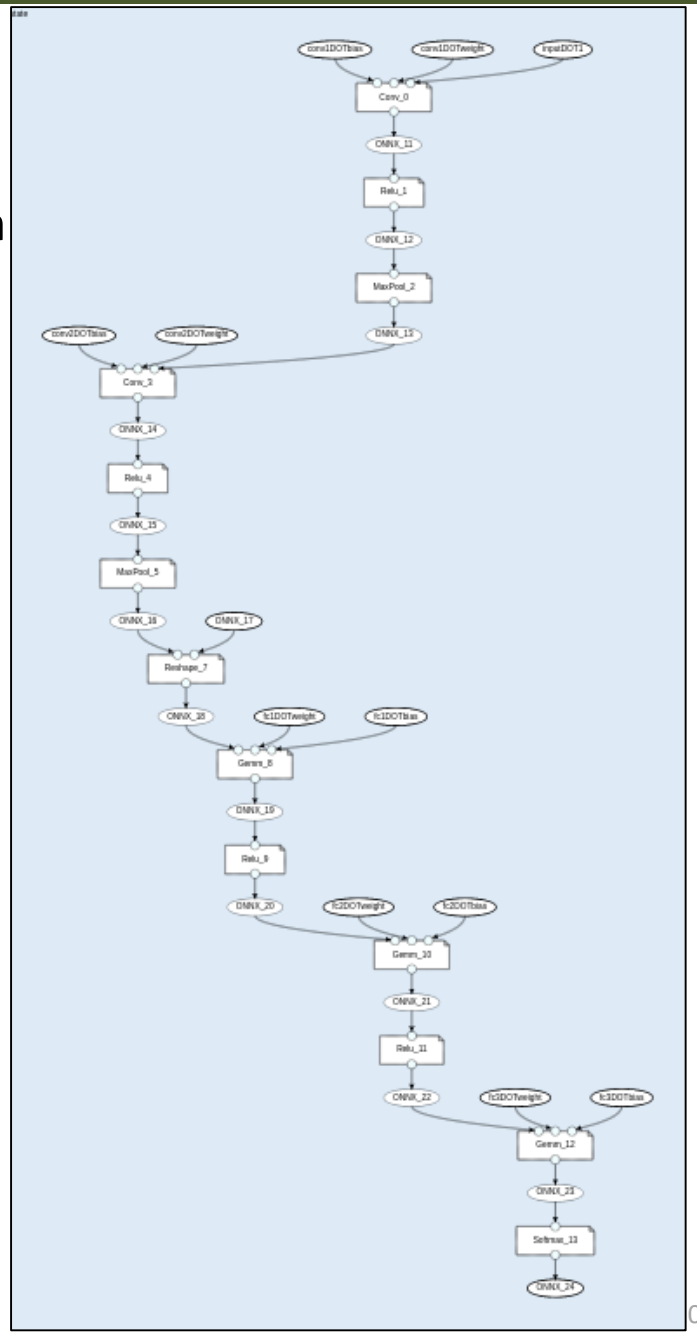
- Build the canonical DAG
- Understand if we can stream between MMM and Add

We need to buffer the result of the MMM

Application Space Exploration

By using library nodes, expansions (lowering) and data-movement analysis we can Exploration:

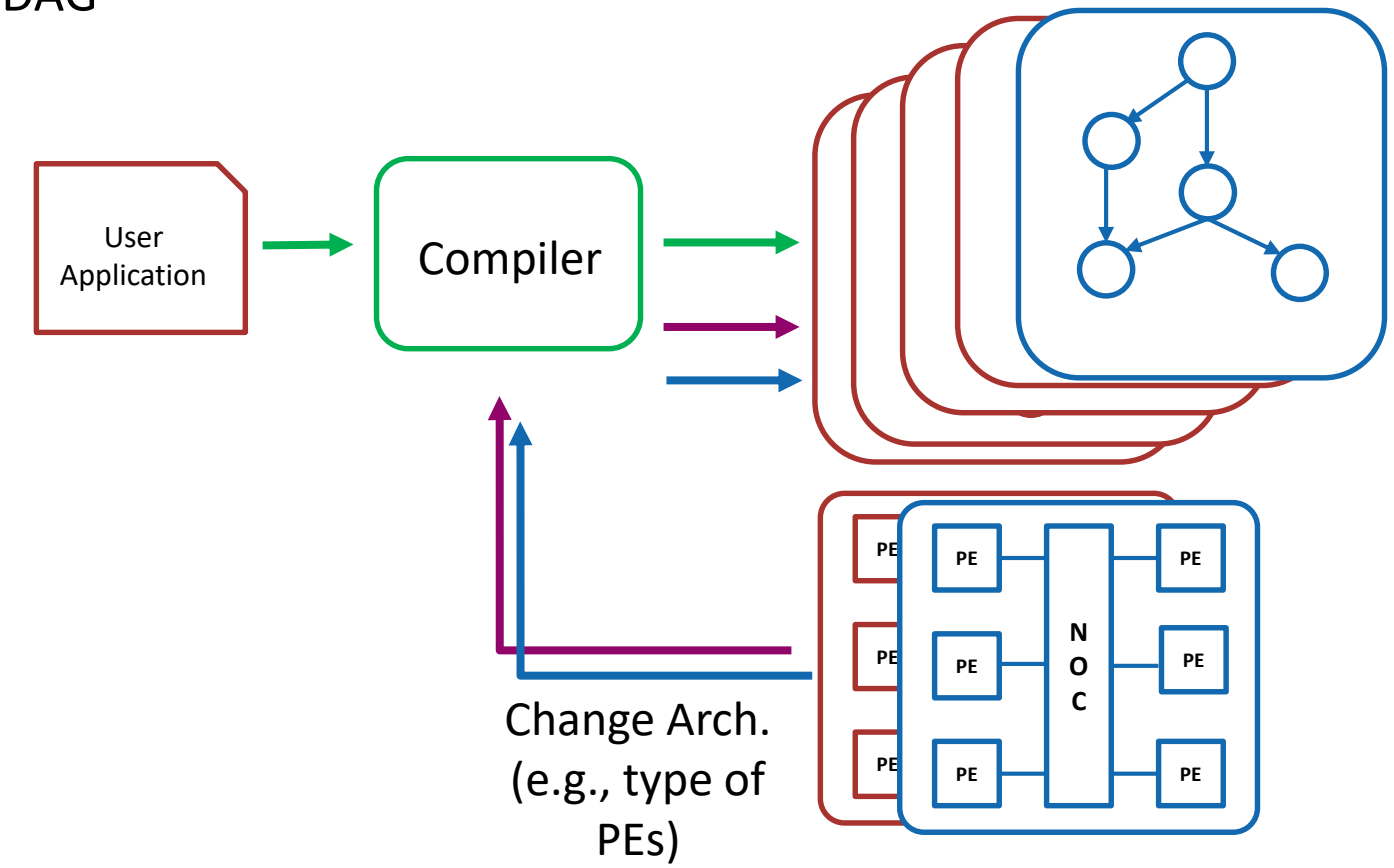
- The application is described by means of library nodes
- From this we create Canonical DAGs:
  - Each library node may have multiple expansions
  - We detect streaming opportunities
  - The scheduling algorithm will decide whether to exploit these or not
- The search space can be large -> we need to develop efficient search strategies
- Applicable to various application domains



Architecture Space Exploration

Architecture Space Exploration can be built on top of:

- **Scheduling/Mapping**, if we just change the # PEs
- **Application Space Exploration**: having a different architecture will let the compiler create a different Canonical DAG



Architecture Space Exploration

A first approach could be to leverage again library nodes.

For example: the architecture may support (or not) support certain operations, therefore we can lower (or not) to certain implementations

- If the architecture support Matrix-Vector operations, we will represent MM by means of Matrix-Vector multiplications
- If the architecture has an efficient MM implementation, we can use it directly

This would require to have an idea on the possible supported operations

Other ideas? (WIP)