Yakup Koray Budanaz

SoftHier Meeting September 13

SoftHier Simulator Workflow

Step	Implemented?
• Generate SoftHier Architecture Configuration	Yes
• Generate Layout for HBM Arrays, ELF binary to move data to the HBM-Files	Yes
• Duplicated Input Data and move data to HBM	No
• Run SoftHier SDFG	Yes
• Run NumPy Code (or. CPU reference SDFG)	No
• Copy back data from HBM-files to Host	No
• Compare the results	No

SoftHier Simulator Workflow

Open Implementation Tasks Needed for the E2E Verification and Development Pipeline

Step	Implemented?
• Generate SoftHier Architecture Configuration	Yes
• Generate Layout for HBM Arrays, ELF binary to move data to the HBM-Files	Yes
• Duplicated Input Data and move data to HBM	No
• Run SoftHier SDFG	Yes
• Run NumPy Code (or. CPU reference SDFG)	No
• Copy back data from HBM-files to Host	No
• Compare the results	No

SoftHier Simulator Workflow

Working on implementing these steps

Step	Implemented?
- Generate SoftHier Architecture Configuration	Yes
- Generate Layout for HBM Arrays, ELF binary to move data to the HBM-Files	Yes
- Duplicated Input Data and move data to HBM	~~No~~ Done
- Run SoftHier SDFG	Yes
- Run NumPy Code (or. CPU reference SDFG)	~~No~~ Done
- Copy back data from HBM-files to Host	No
- Compare the results	No

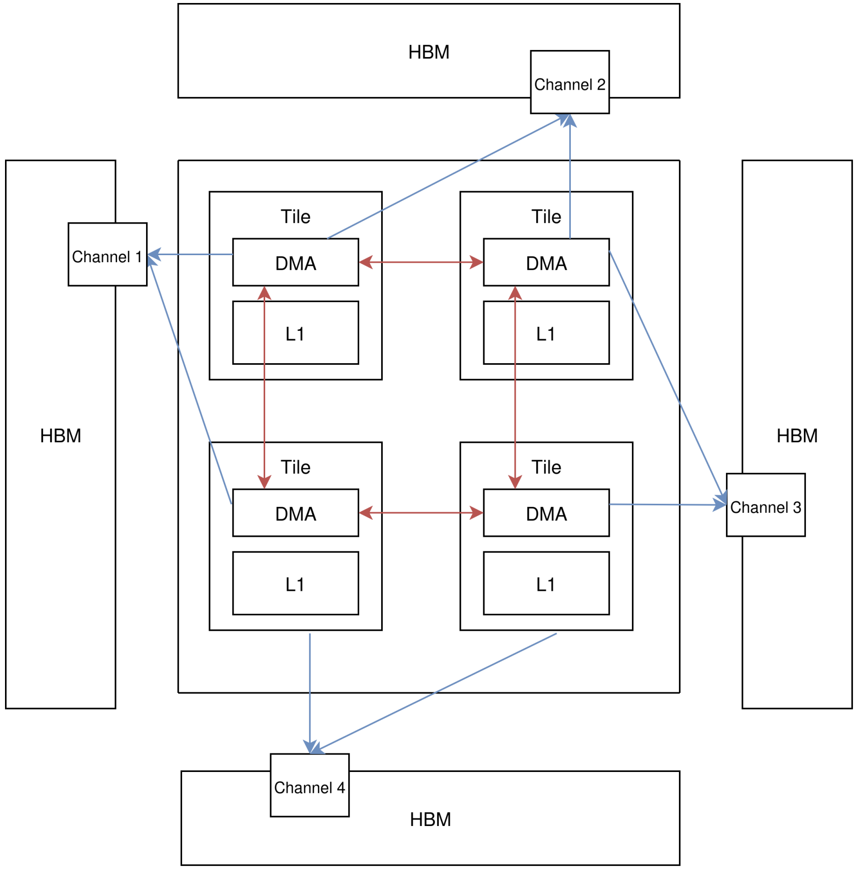
SoftHier Development Pipeline

- Verification of the results need to be automated
- This is also crucial to obtain the results for the hardware design space exploration paper

Applying Layout Transformations to SoftHier

The current layout implementations for SoftHier:

Cluster dimensions: (2, 2)
Cluster dimensions (DACE): (2, 2)



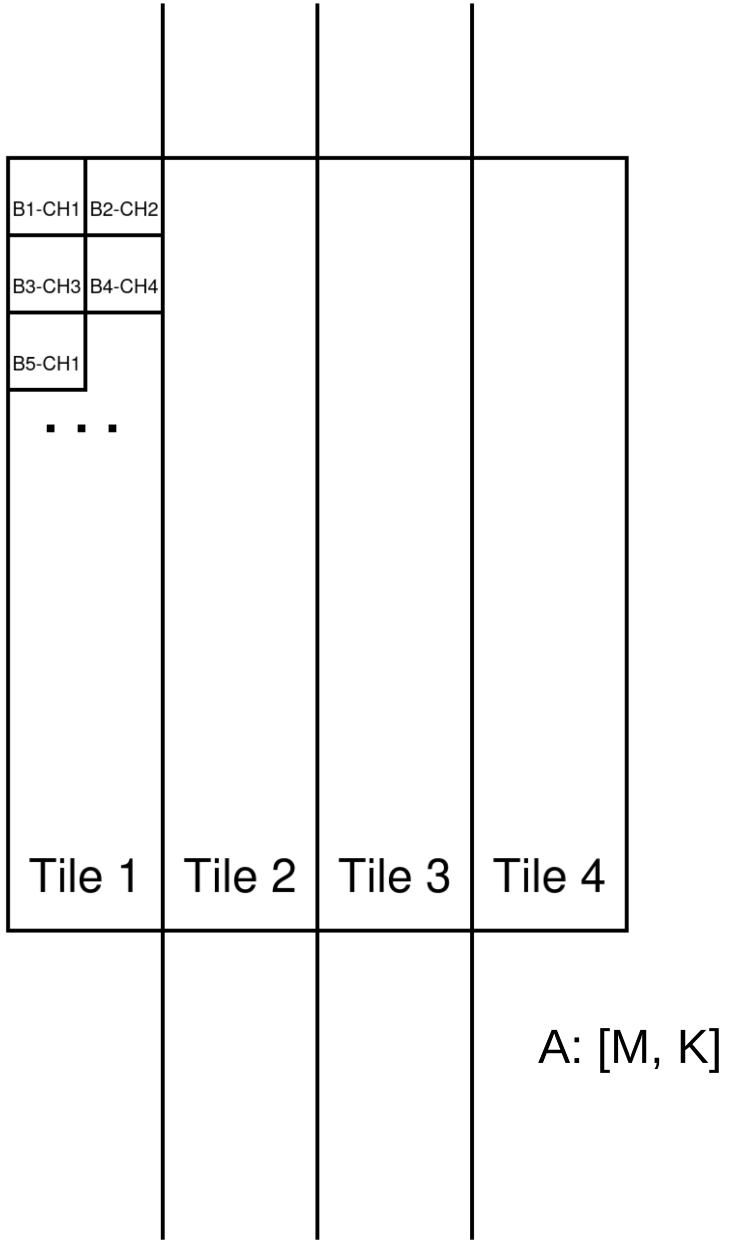
Applying Layout Transformations to SoftHier

The current layout implementations for SoftHier:
Example for “Vecrtical-Split” layout:

Block Shape: (1, 1) // C-based indexing (M, K)

Split Scheme: (1, 4) (we have 4 tiles over K)

Tiling Shape: (M, K / 4)



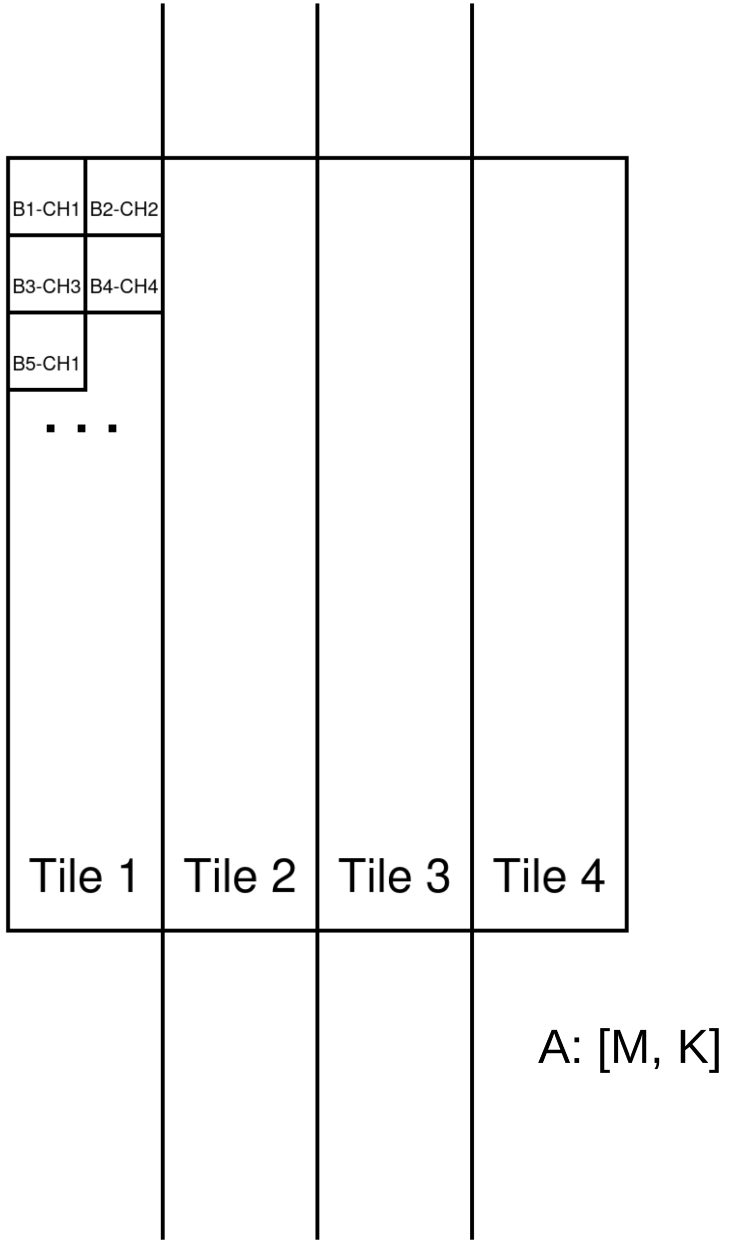
Applying Layout Transformations to SoftHier

The current layout implementations for SoftHier:
Example for “Vertical-Split” layout:

Block Shape: (1, 16)

Split Scheme: (1, 4)

Tiling Shape: (M, K / 4)



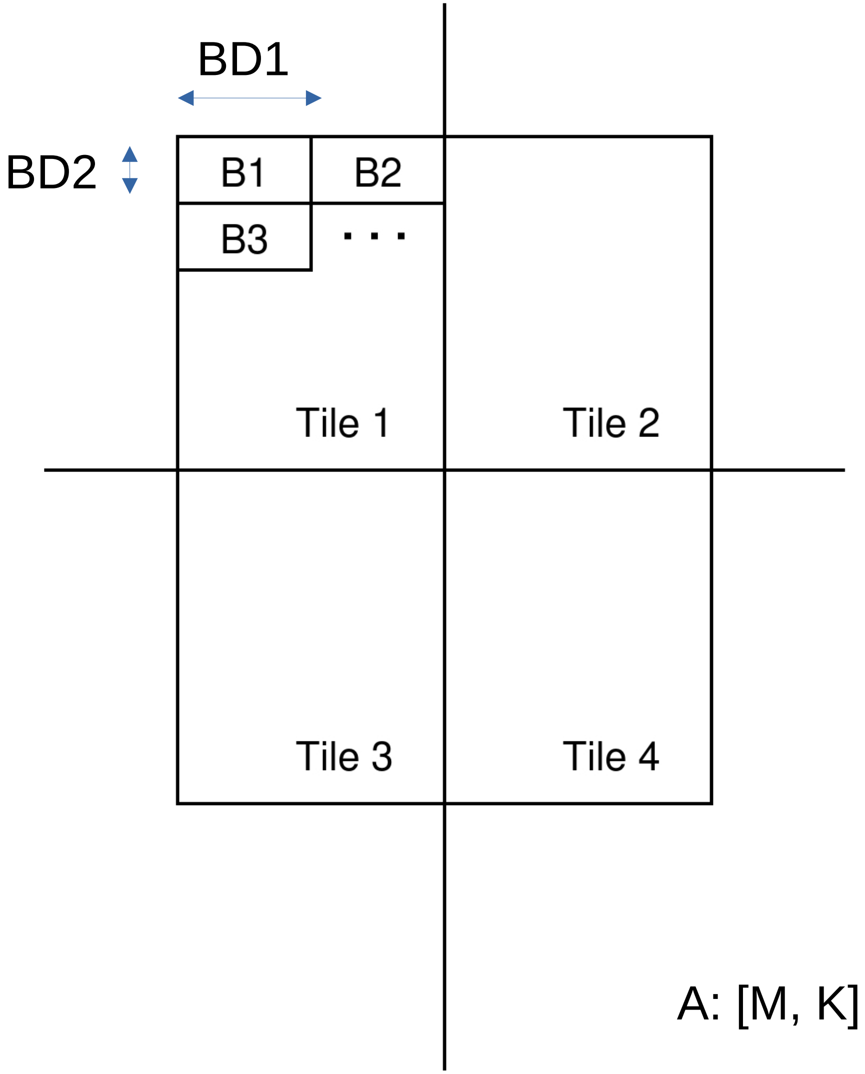
Applying Layout Transformations to SoftHier

The current layout implementations for SoftHier:
Example for a 2D-blocked layout:

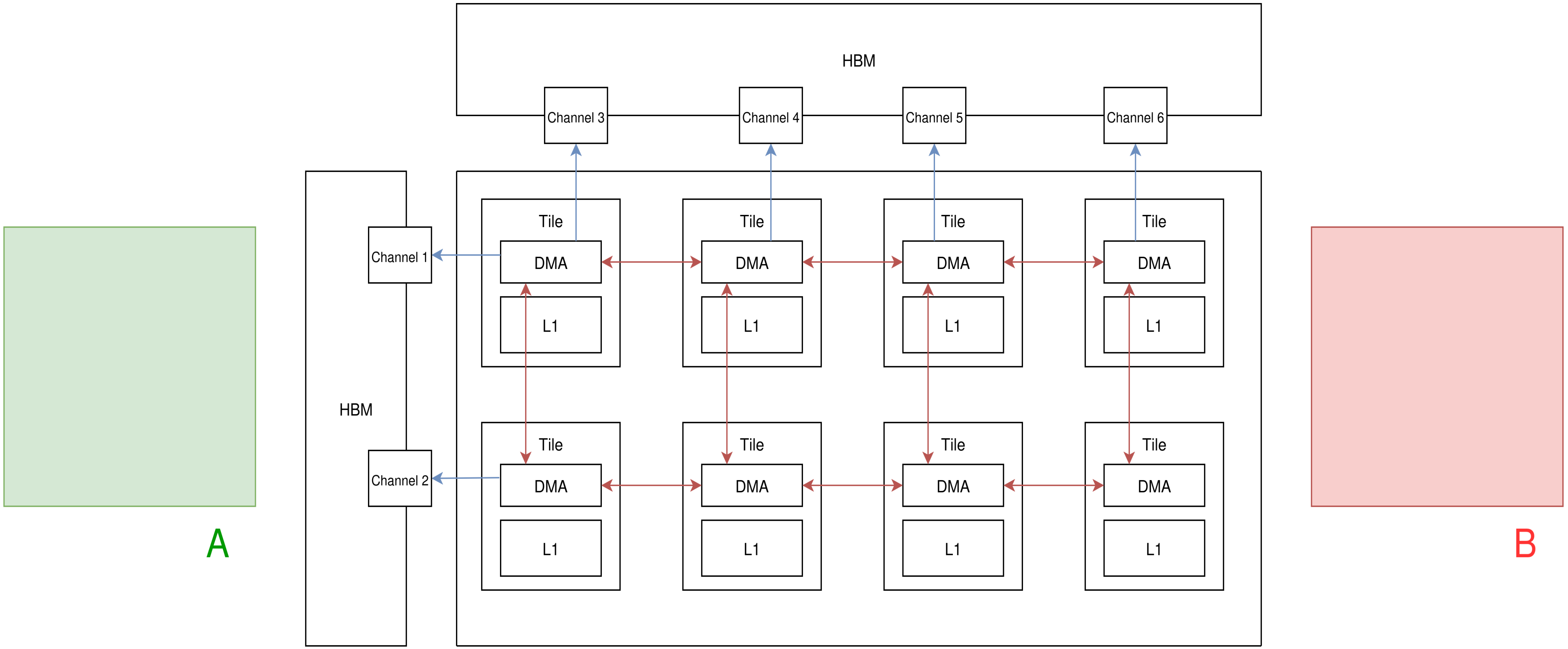
Block Shape: (BD1, BD2) // Corresponding to (M, K)

Split Scheme: (2, 2)

Tiling Shape: (M / 2, K / 2)

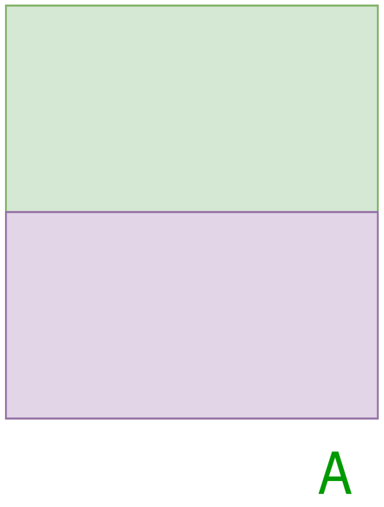


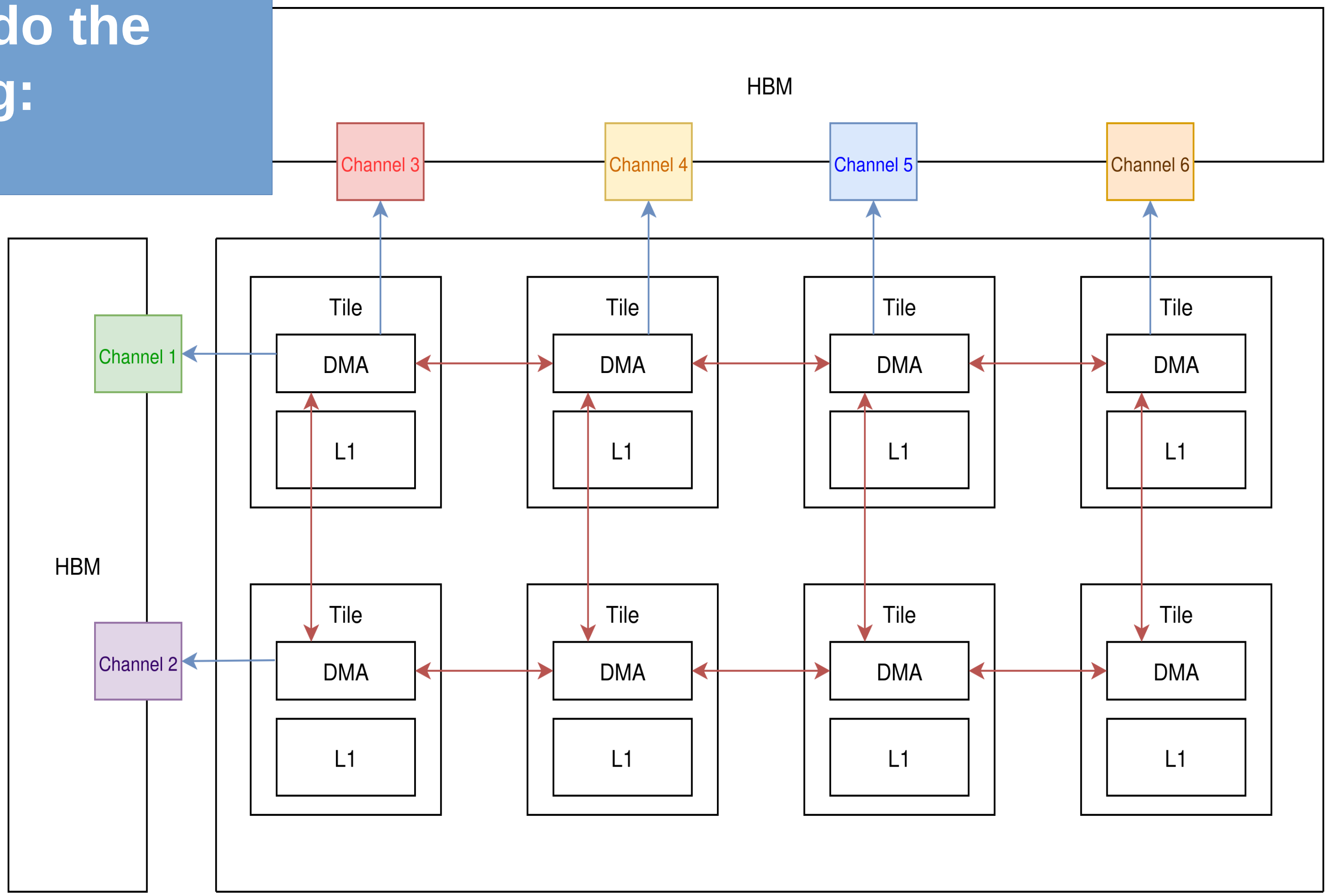
Automated Layout Mapping Derived From Hardware Topology

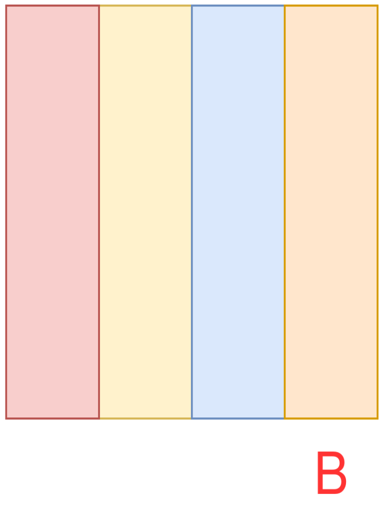


Automated Layout Mapping Derived From Hardware Topology

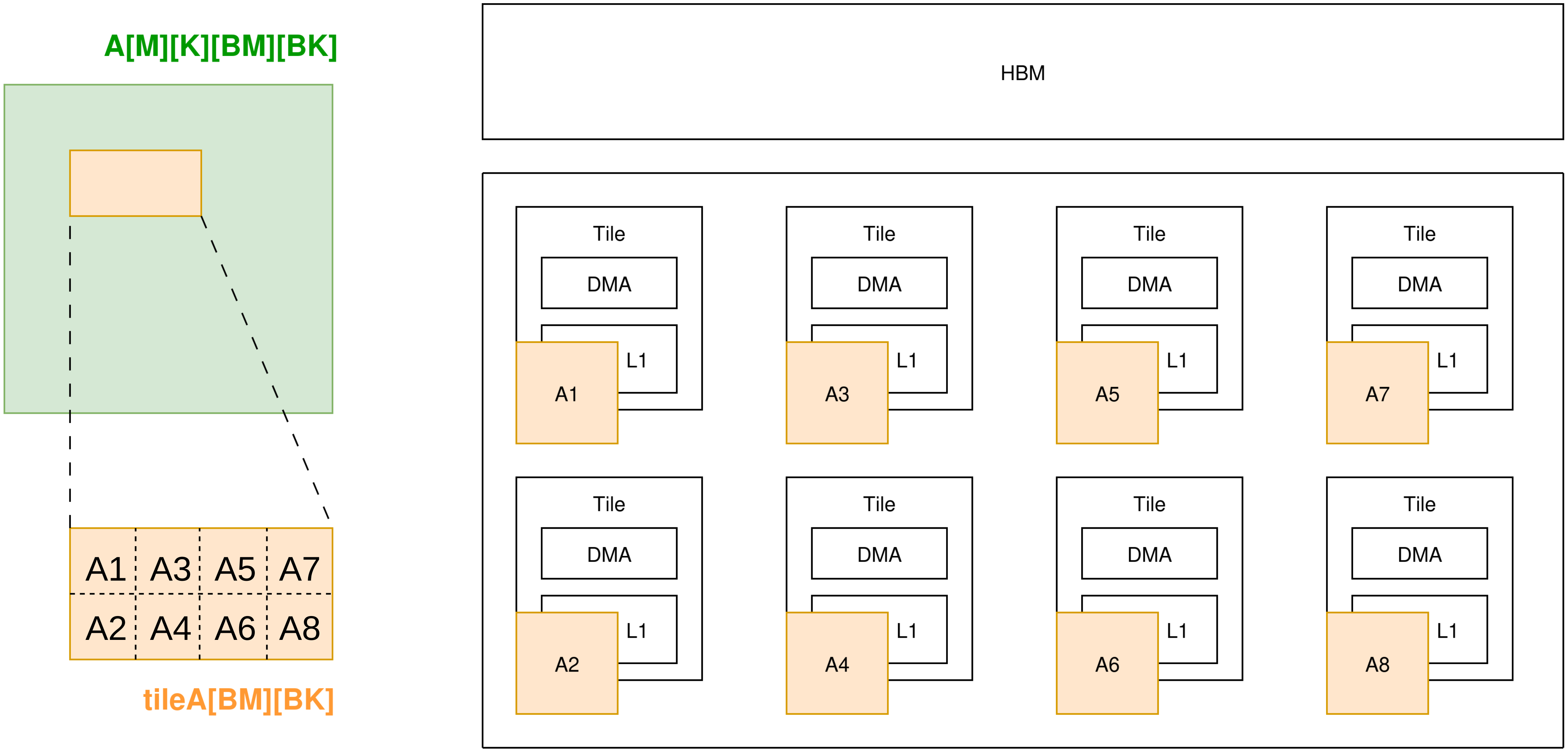
Horizontal and Vertical Split aim to do the following:



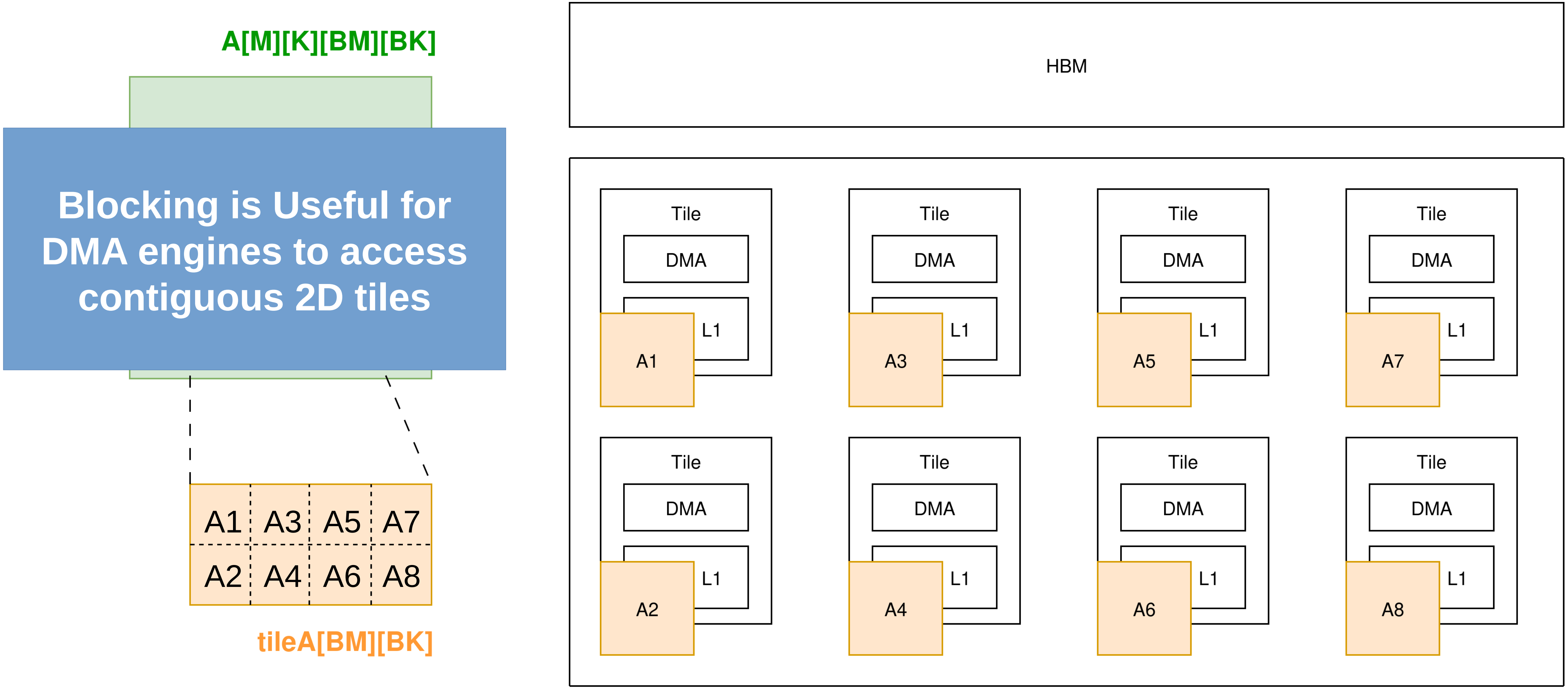




Automated Layout Mapping Derived From Hardware Topology



Automated Layout Mapping Derived From Hardware Topology



Blocking is Useful for DMA engines to access contiguous 2D tiles

Applying Layout Transformations to SoftHier

- The current layouts are implemented through a coupling of SDFG fields and the SoftHier Backend.
  - *Layouts only support two-dimensional tensors, any other shape needs to be transformed into a two-dimensional shape*

- The layout is not made visible to the SDFG IR (through dimensions or locations)
  - *Must not apply layout transformations on the SDFG IR, but on the SoftHier side*

Open Issues Regarding SoftHier Backend

- The layout implementation in SoftHier backend (Interleaver) is not teste
  - *By default GEMMs ran on arbitrary input and therefore the GEMM kernels and the interleaver lacks testing and rigorous numerical verificaiton*

- Arguments passed to the main function are hardcoded to be address at HBM-offsets: 0, 4, 8; names: A, B, C
  - *Different arguments need to be supported to run arbitrary programs*

- Print function combined with the interleaver does not work
  - *Need cooperation from Chi and Aofeng to fix the issue*

- The vector unit is supported by the backend
  - *Backend needs to be extended*

- The layout is not made visible to the SDFG IR (Through dimensionality of data or locations)
  - *Must not apply layout transformations on the SDFG IR, but on the SoftHier side for SoftHier SDFGs*

- SDFG is called from python to run a bash script that runs the \`output.elf\` and main function.
  - *Must ensure whole computation runs on the SoftHier device (no CPU-SoftHier hybrid computation). Due to structure of the \`output.elf\` a better solution might not be possible currently.*

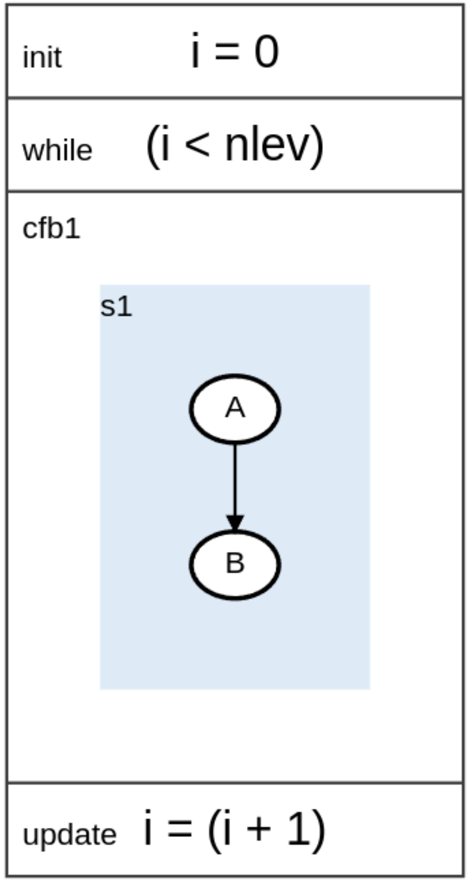
CloudSC on SoftHier – Next Steps for the Backend

- An Explicit Vectorization pass is necessary to get CloudSC to work on the vector units
- The whole computation needs to run on SoftHier (no hybrid CPU-SoftHier execution)
- Support for multi-kernel execution needs to be extended (improvement of hardcoded function arguments)

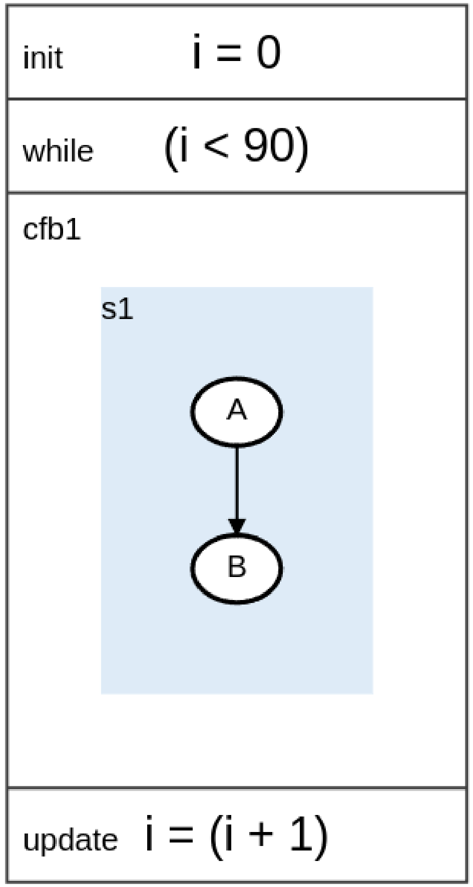
CloudSC on SoftHier

- An **Explicit Vectorization** pass and **Preprocessing Passes** to make computation more amenable for SoftHier is necessary to get CloudSC to work on the vector units

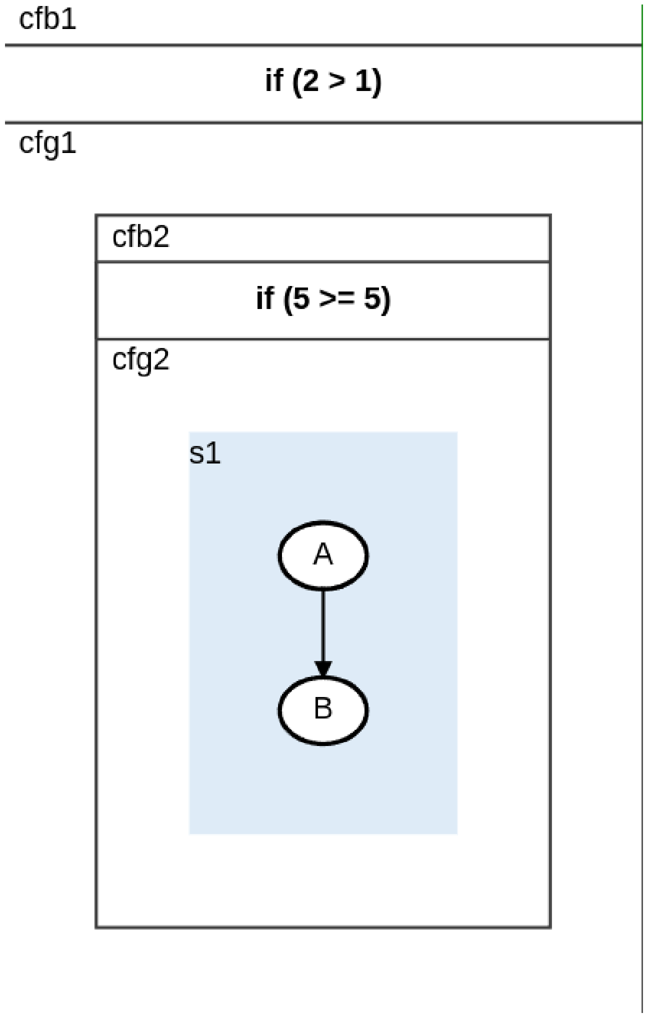
CloudSC on SoftHier – Specialize Scalar (#2139)



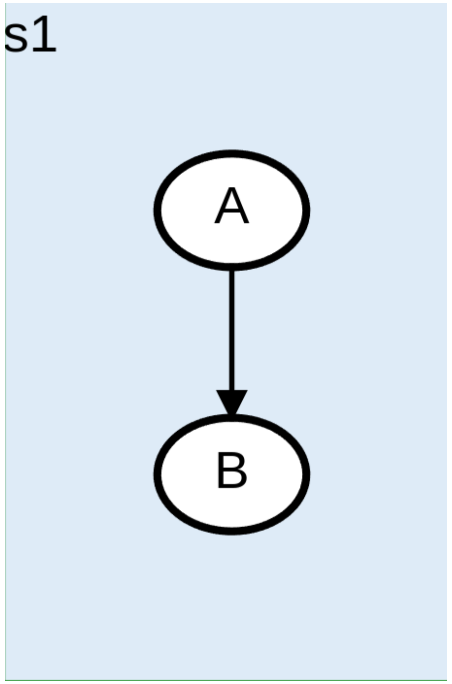
```
.specialize_scalar({
        “nlev”: 90
})
```



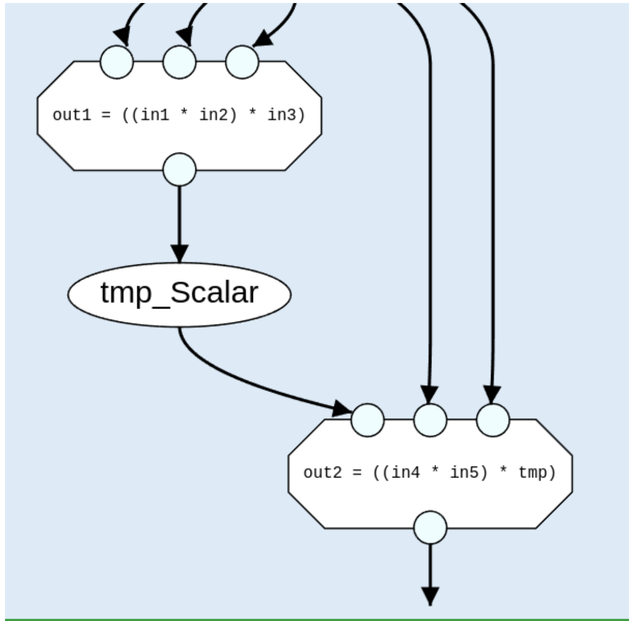
CloudSC on SoftHier – Lift Trivial Ifs (#2138)



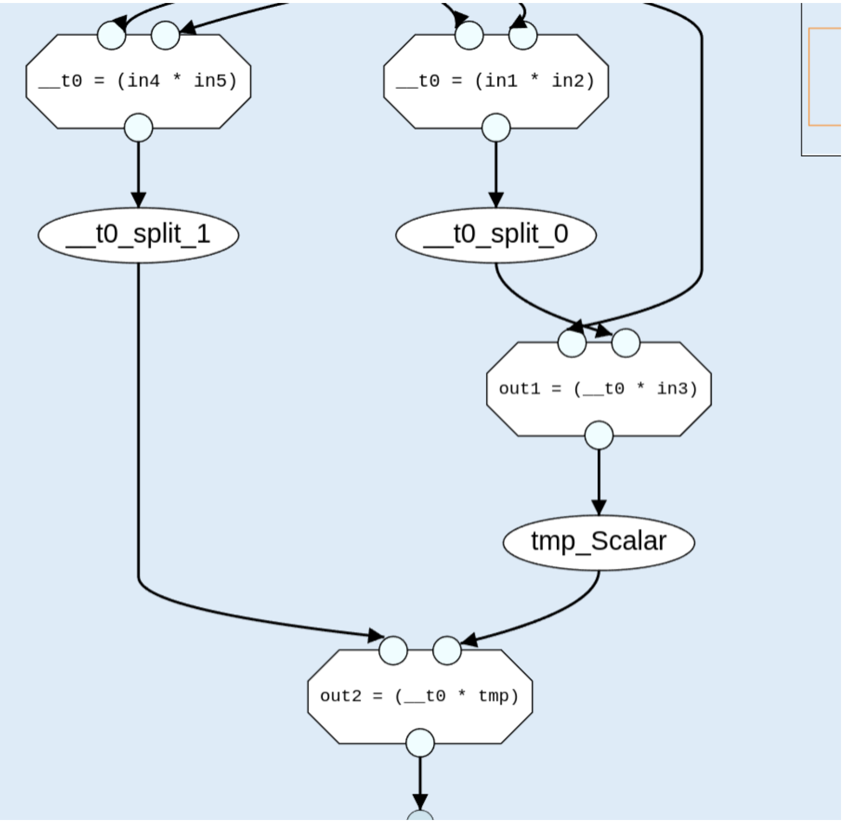
```
.sdfg.apply_pass(
        LiftTrivialIfs
)
```



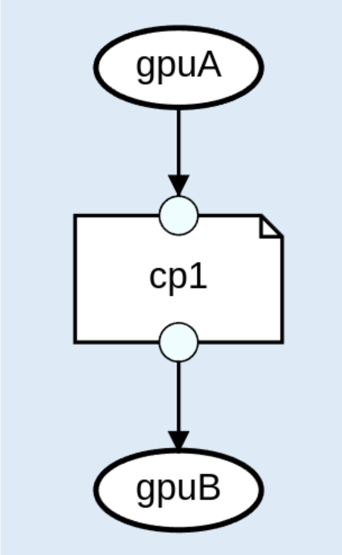
CloudSC on SoftHier – Split Tasklets (#2143)



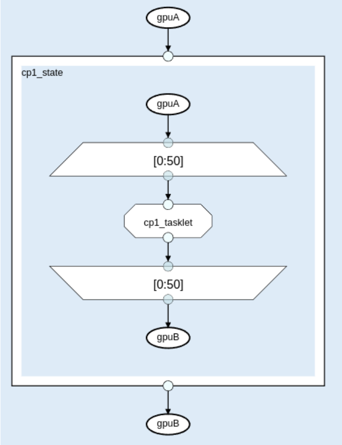
```
.sdfg.apply_pass(
        SplitTasklets
)
```



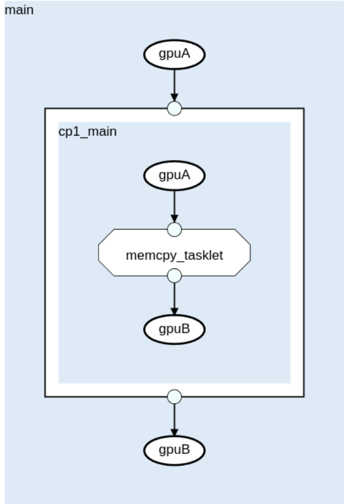
CloudSC on SoftHier – Memcpy / Memset Libraries (#2144, #2123)



```
.library.specialize(
       Pure
)
```



```
.library.specialize(
       CUDA
)
```



```
.library.specialize(
       SoftHier
) ?
```

?

CloudSC on SoftHier – F2DaCe (#2147)

- Working on merging **f2dace/dev** to main to move CloudSC SDFG from **f2dace/dev** branch to main and then to **softhier_backend**