ASA: Enabling DSE

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. — DONE

Probably we would need more simpler use cases

:he application to Canonical DAGs

- Conveniently represent iterative algorithms — DONE
- 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 — Ongoing
- 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) — Ongoing
  - Power: directly proportional to the off-chip memory accesses — Then needs your support

Documentation — Ongoing

Results streaming vs non-streaming

5G (full partition)

# PEs	32	64	128	256	512
Stream MKSP	46678450	24038560	12515230	6948760	3859310
Non Stream MKSP	46569250	24295180	13346060	8104460	5483660
Gain	1.00	1.01	1.07	1.17	1.42

Encoder Layer

# PEs	64	128	256	512
Stream MKSP	18492454	10103848	6473766	3997719
Non Stream MKPS	24483072	16618752	12686592	10720512
Gain	1.32	1.64	1.96	2.68

Can scale a bit further

PPA Estimation

Usually, hardware design tool flows rely on **synthesis** to estimate PPA (potentially re-iterating to meet given constraint) while performing DSE

Despite being accurate, this is time-consuming.

In our case:

- we are reasoning on an High-Level architecture, so synthesis is not immediate
- We want to quickly evaluate hundreds/thousands of solution, so synthesis is not a viable option
- We can be ok with a *qualitative* evaluation of some parameters rather than a *quantitative* (configuration X is better than configuration Y)
- Return a set of candidate solutions for successive refinements, rather than a single solution

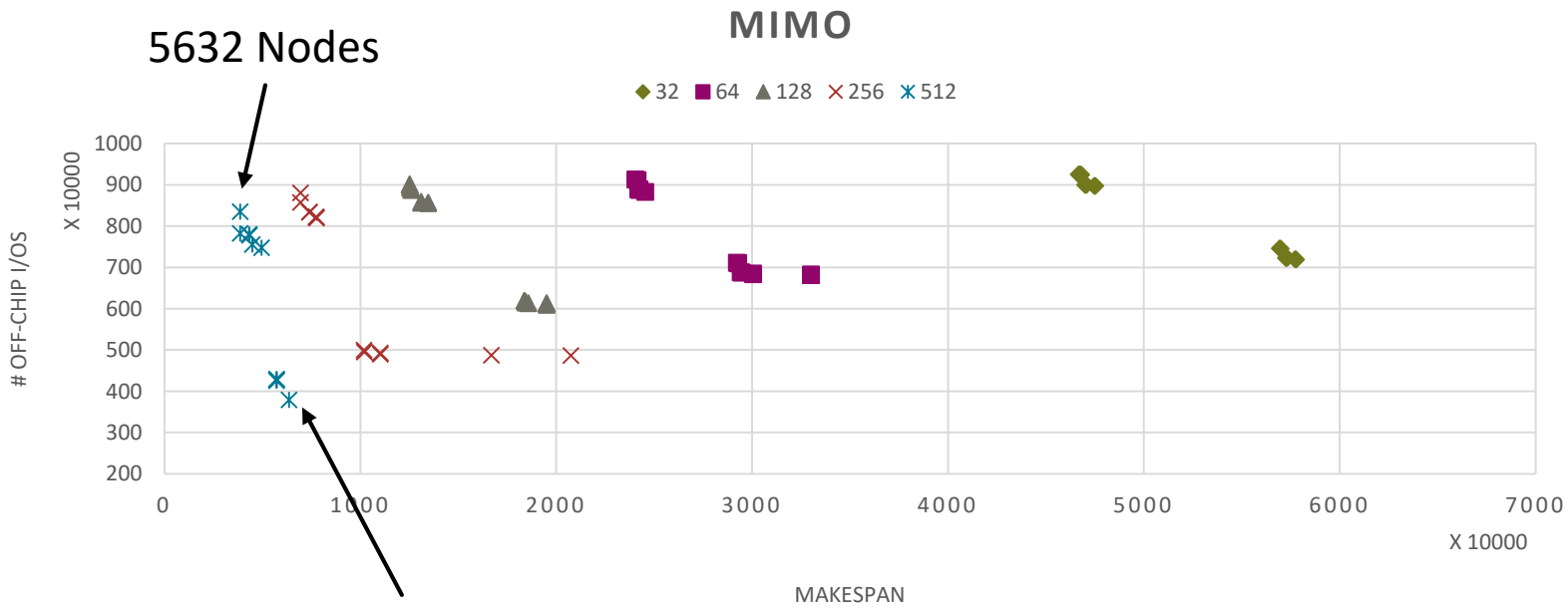
PPA (on going work)

Currently naïve models for PPA:

- Performance = Makespan
- Power = Number of off-chip I/Os
- Area = Number of PEs

Naïve models for the sake of creating the infrastructure.

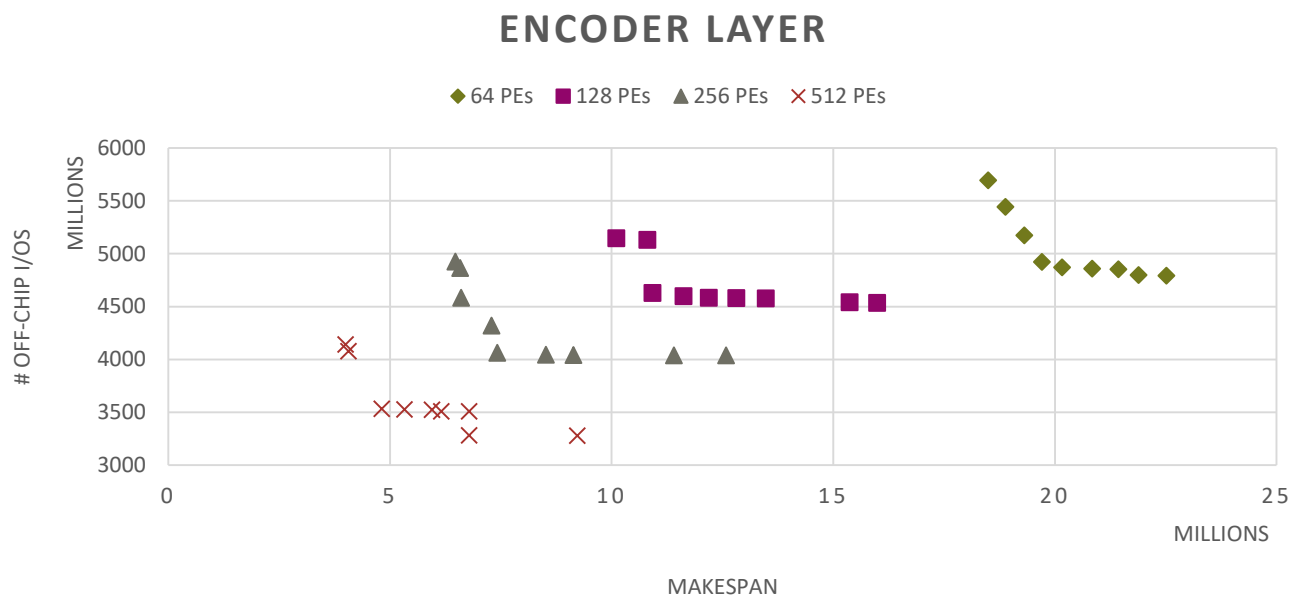
Need to be replaced with something more meaningful.



These are different applications

PPA (on going work)

Encoder



Analysis time improved by 2.5x by switching from *transformations* to *passes* (9 hrs vs 22 hrs)

TODOs:

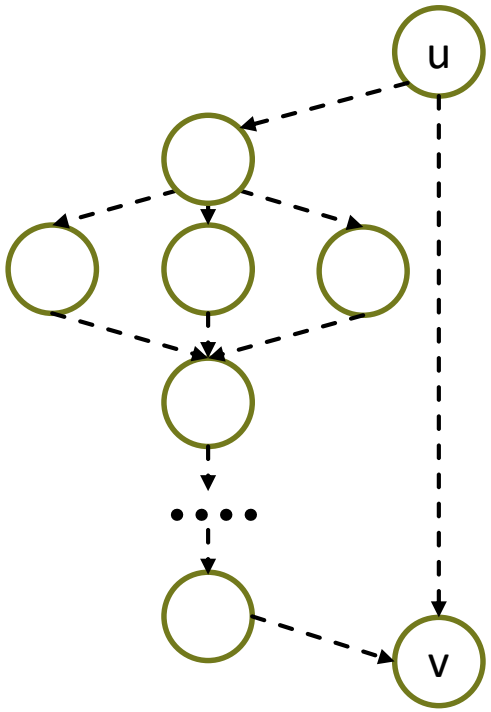
- incorporate buffer space in the Area estimation
- Consider an aggregate score?

Would like to have your feedbacks

Buffer space and spatial partitioning (Ongoing)

While looking at the buffer space requirements for Encoder, I realized that this was quite high (hundreds of MBs)

Solved a first semi-bug in buffer space detection (it was computing more buffer space than actually need)



To avoid deadlocks, the right streaming edges arriving to *v* must have enough buffer space to compensate for the delay on the left side:

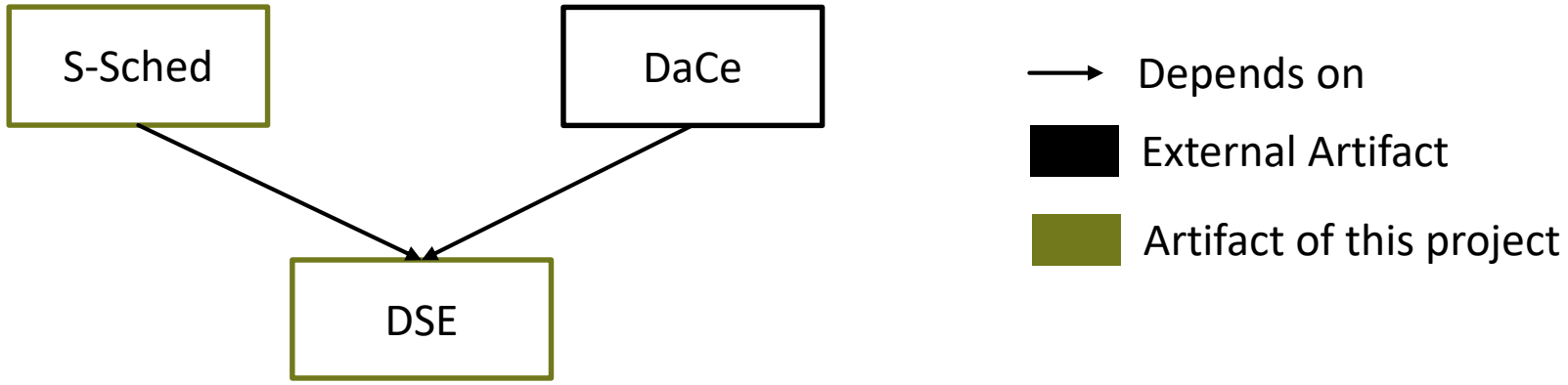
- This can be expensive (buffer space depends on the delay)
- There could be cases where there is no actual benefit in having that streaming edge

How to deal with this:

- Buffer space is computed only **after** the spatial partitioning and scheduling
- Clear cases can be detected afterward by looking at the computed buffer space and the edge volume
- How to deal with trade-offs?

Docs

We will have (at least) two software artifacts



Will be released as public projects:

- With documentation, testing, samples, …