VAADER IETR research team

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

IETR

HiPEAC/DASIP 2023: Workshop on Design and Architectures for Signal and Image Processing

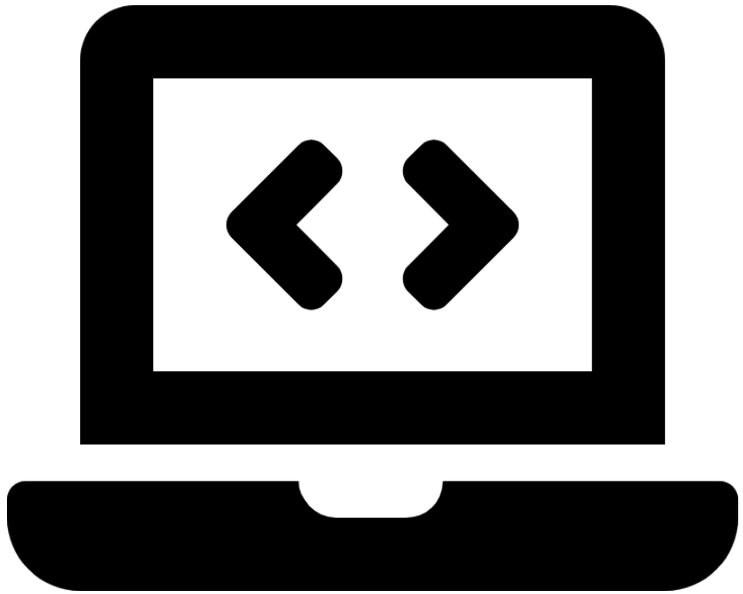
SCAPE: HW-Aware Clustering of Dataflow Actors for Tunable Scheduling complexity

Ophélie Renaud[*], Dylan Gageot, Karol Desnos[*], and Jean-François Nezan[*]

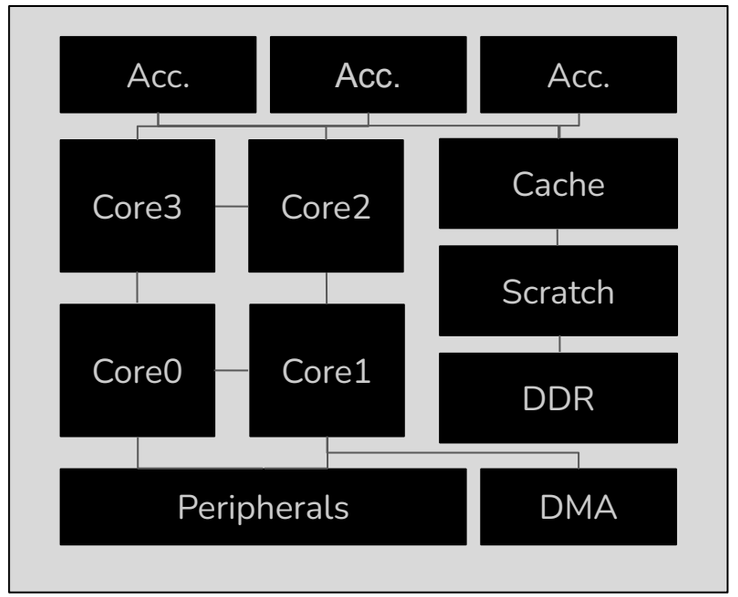
[*] Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes, France
first.last@insa-rennes.fr

DARK ERA

anr agence nationale de la recherche

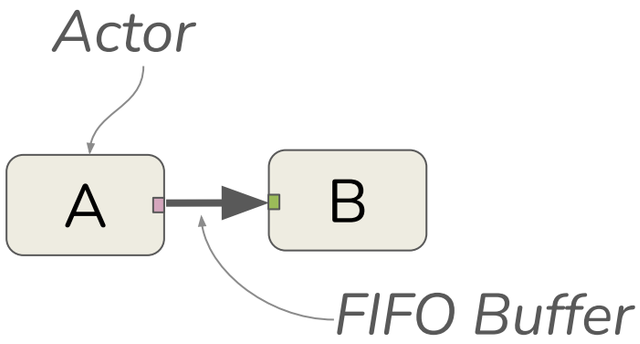


*Program **complex***



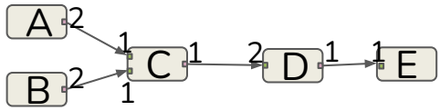
*Exploit all the **potential** of architectures*

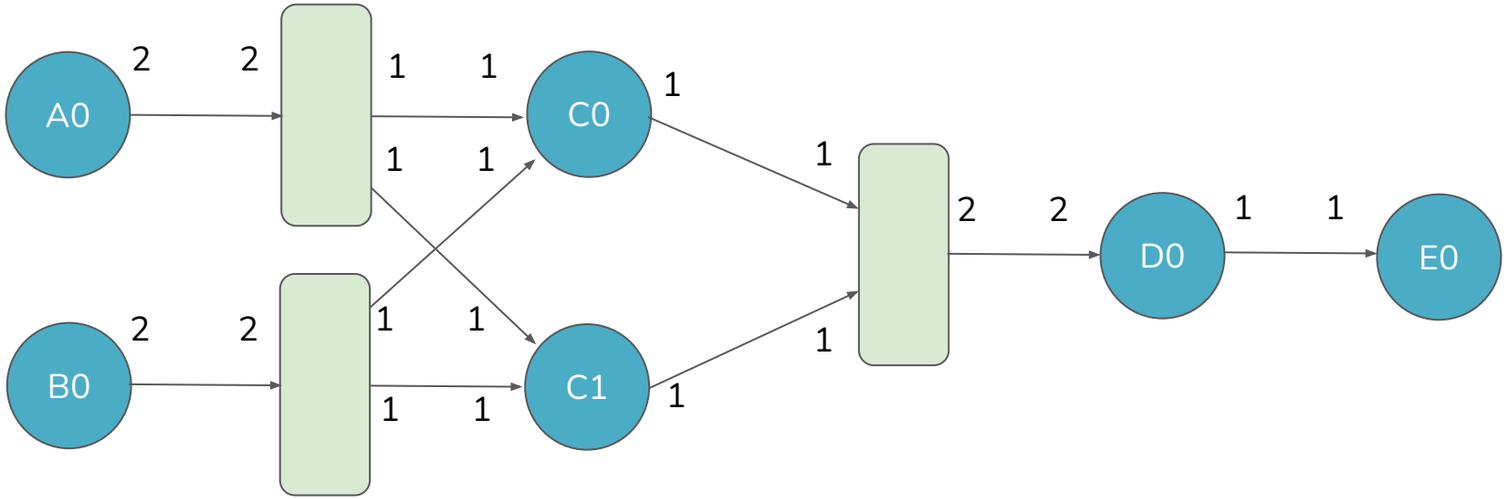
DataFlow

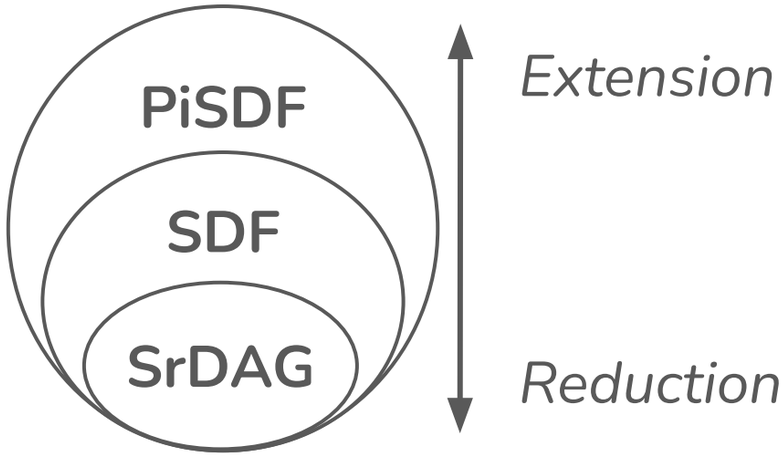


- *Reveals **dependencies** between **tasks***
- *Makes visible any missing **details***
- *Takes benefit of every available **resources***

Synchronous DataFlow (SDF)



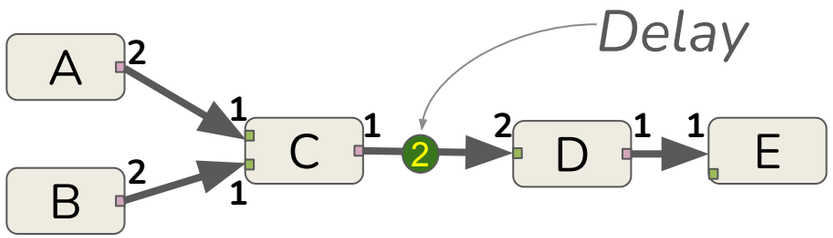


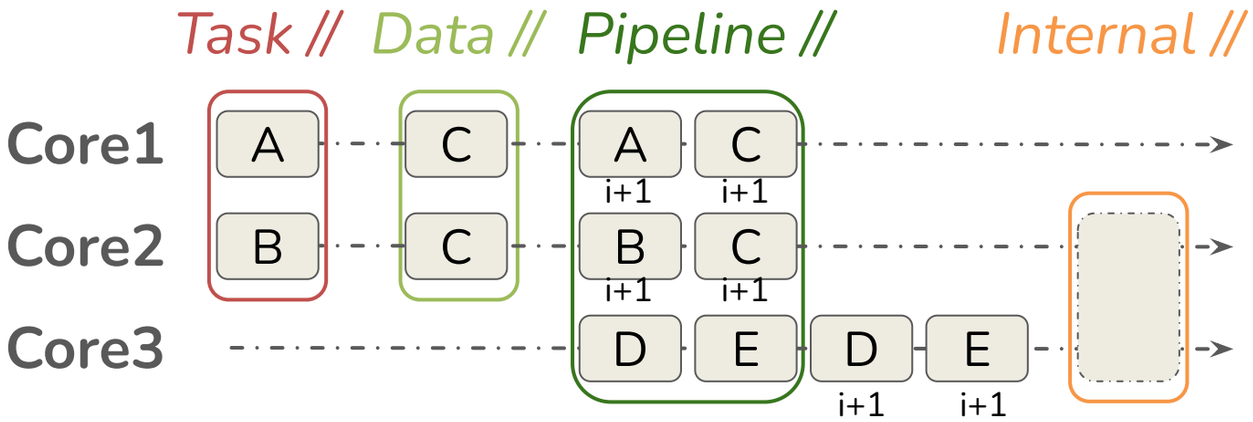


[1] E. Lee and D. Messerschmitt. “Static scheduling of synchronous data flow programs for digital signal processing”. Berkeley, 1987
[2] K.Desnos, , J.Heulot, : “PiSDF: Parameterized & Interfaced Synchronous Dataflow for MPSoCs Runtime Reconfiguration”.IETR, 2014
[3] G. Sih, E. Lee :” Scheduling to account for interprocessor communication within interconnection-constrained processor networks”. Berkeley, 1990

Why SDF?

➢ Expression of several types of parallelism

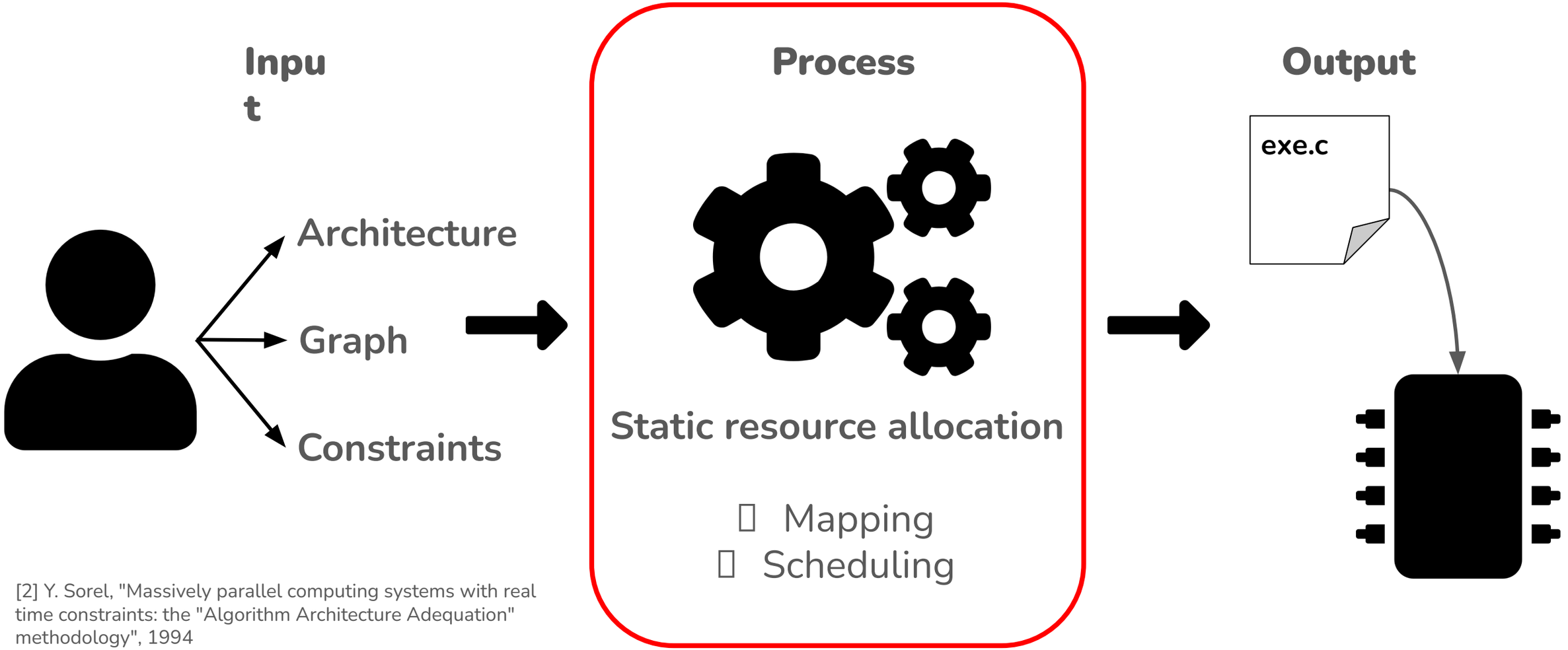




➢ Independent of the architecture

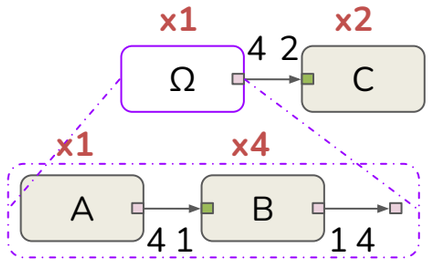
[1] E. Lee and D. Messerschmitt. “Static scheduling of synchronous data flow programs for digital signal processing”. Berkeley, 1987

Dataflow based design process

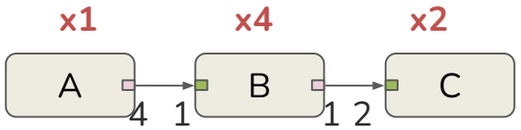


[2] Y. Sorel, "Massively parallel computing systems with real time constraints: the "Algorithm Architecture Adequation" methodology", 1994

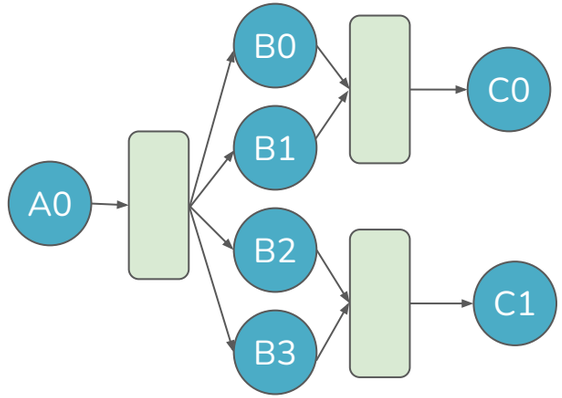
Classic static scheduling method



PiSDF Input Graph

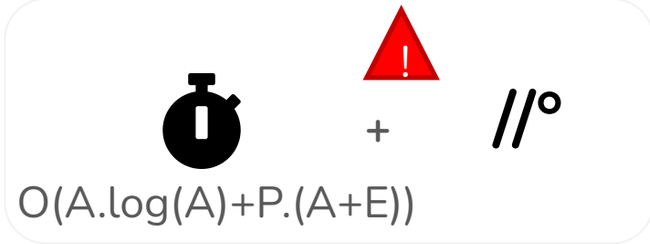


Flattening



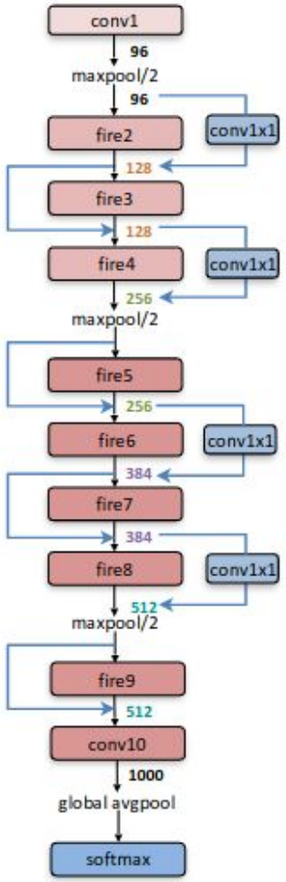
SrDAG Transformation

Mapping
Scheduling



[3] G.C. Sih, Lee. Scheduling to account for interprocessor communication within interconnection-constrained processor networks, Berkeley, 1990

Illustration of the problems



Squeezenet neural network

- 70 actors
- 84 fifos
- 1000 degree of parallelism

SrDAG

- 5452 actors
- 17665 fifos

Analysis 4 PEs

- 4 850 480 903 ms = 56d 3h 21m 20s

[4] F. N. Iandola , S. Han , M. W. Moskewicz , K. Ashraf , W. J. Dally , K. Keutzer. SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE, DeepScale Berkeley & Stanford, 2017

How to reduce the complexity of a program without compromising its parallelism?

- Reduce **analysis time**
- Not to compromise the **latency**

- Size of **SrDAG**
- Keep **parallelism**

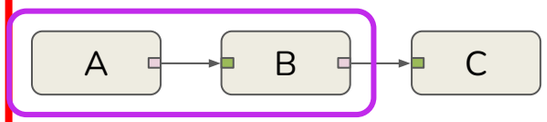
SCAPE method principle

Input Graph

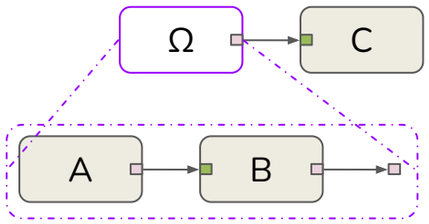
↓

Clustering Transformation → Flattening → SrDAG Transformation → Mapping Scheduling

Pattern identification



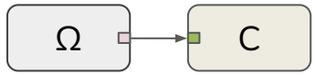
Subgraph generation



Schedule

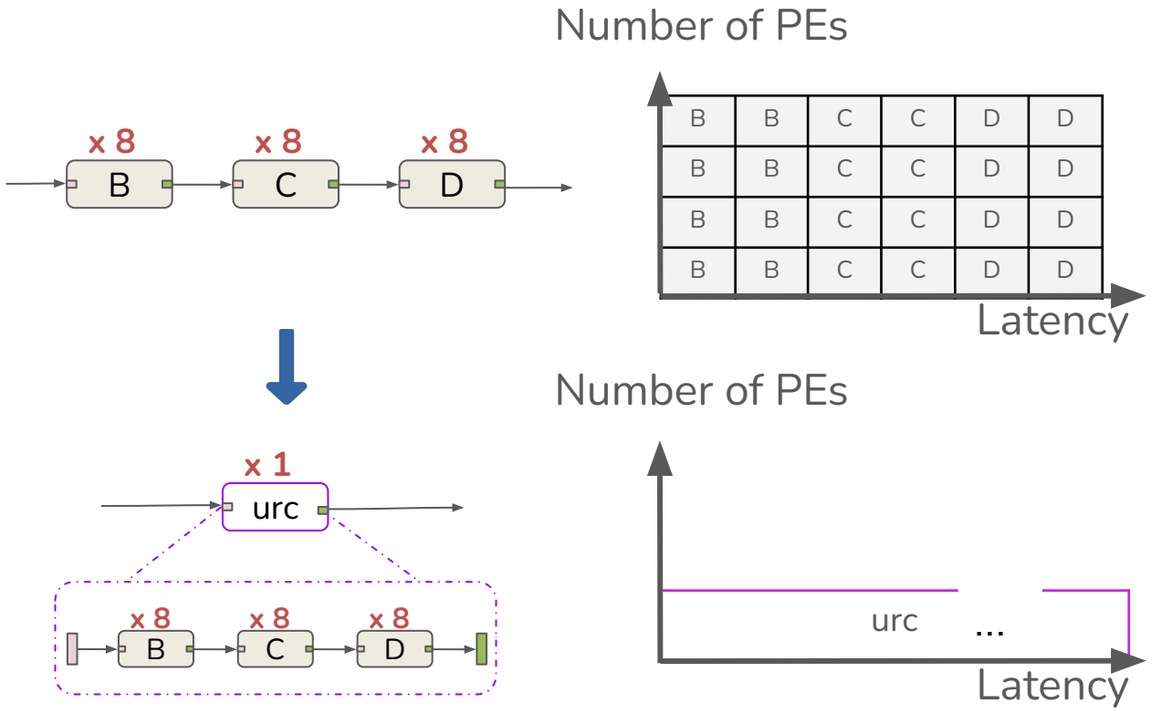
Code generation

Behavior modification

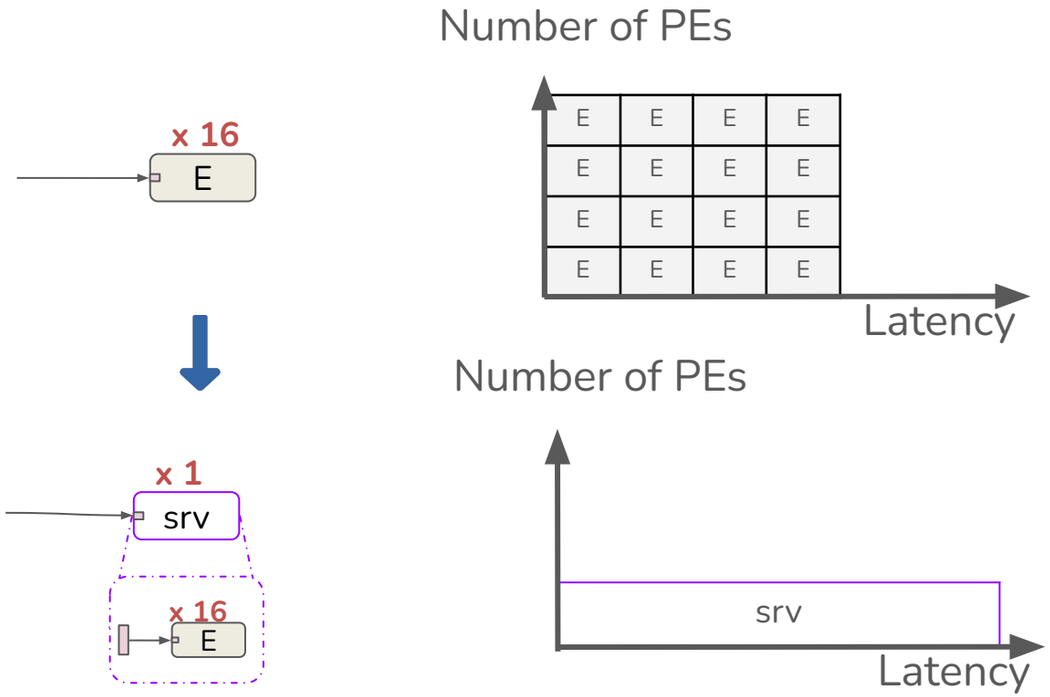


Step 1: Pattern to reduce analysis time

Unique Repetition Count

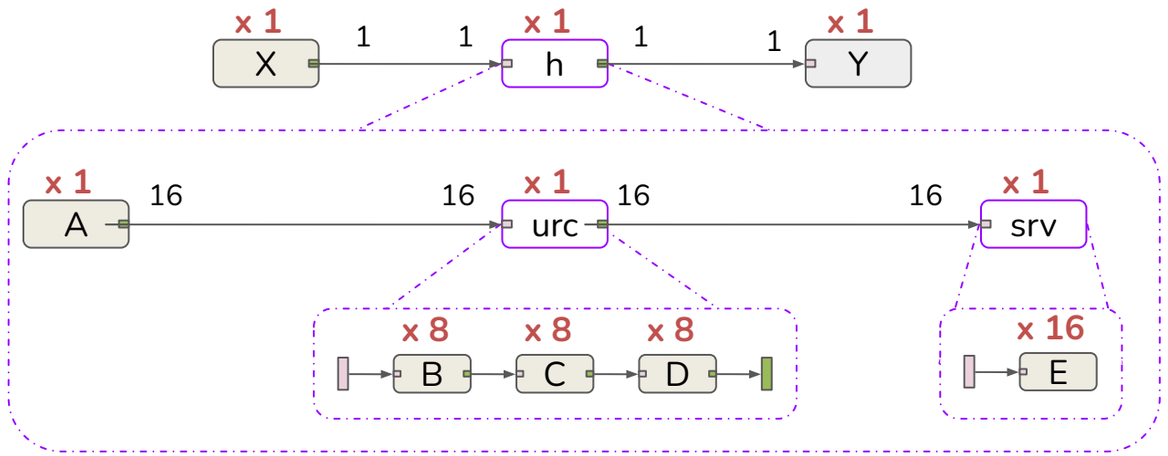


Single Repetition Vector

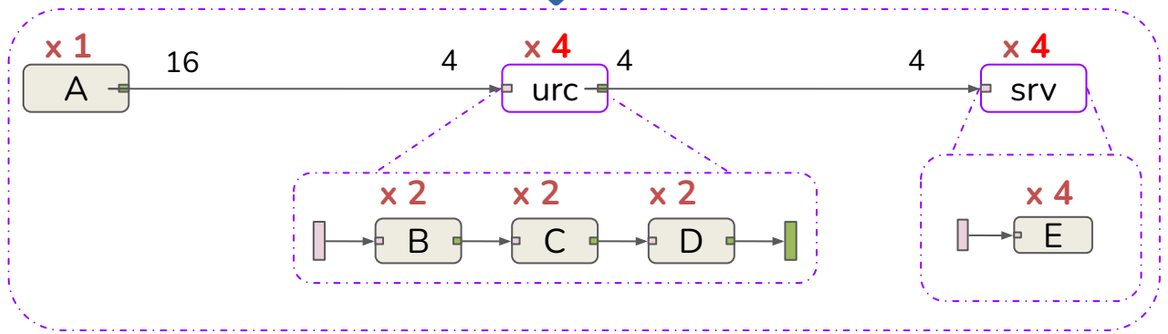


☑ SrDAG
☒ Parallelism

Step 2: Scaling up of Clusters on Processing Element

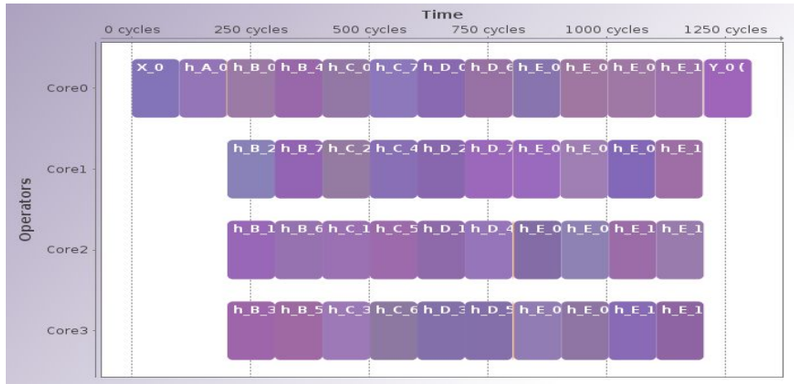


× **4** PEs



☑ SrDAG
☑ Parallelism

Without clustering



SrDAG : 53 actors

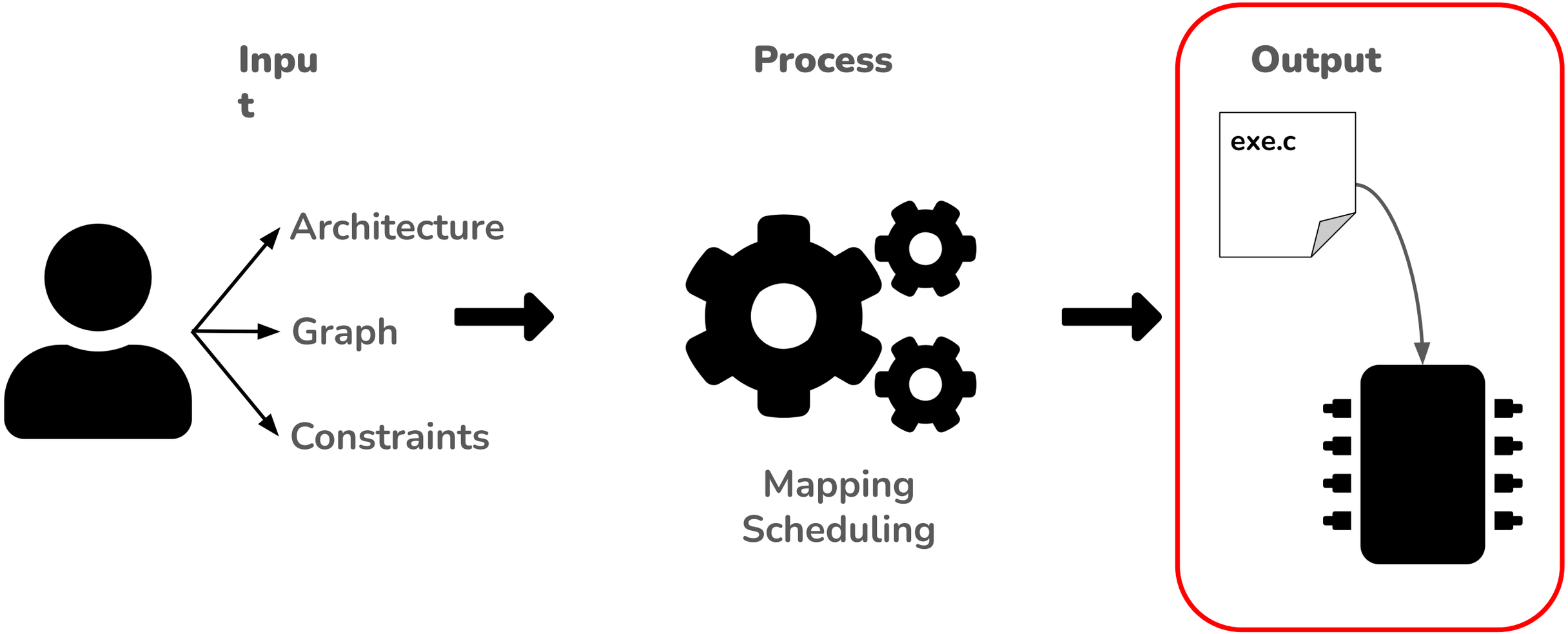
Latency : 1300 cycles

Optimized clustering

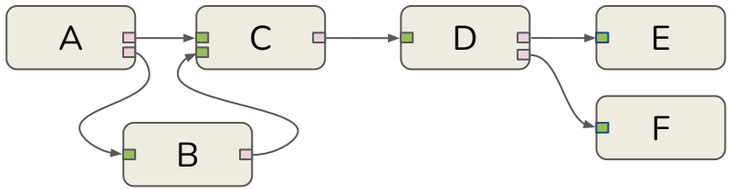
SrDAG : **13** actors

Latency : **1300** cycles

Dataflow based design process



Code generation with the classic method



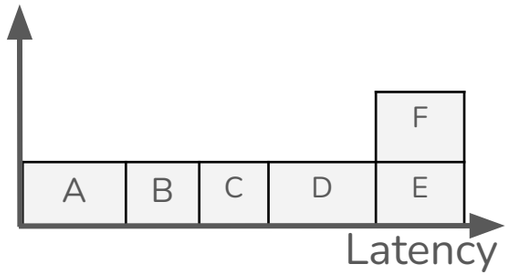
```
A(out0_A,out1_A);
B(out1_A,out0_B);
C(out0_A,out0_B,out0_C);
D(out0_C,out0_D,out1_D);
Send();
E(out0_D);
```

Core0.c

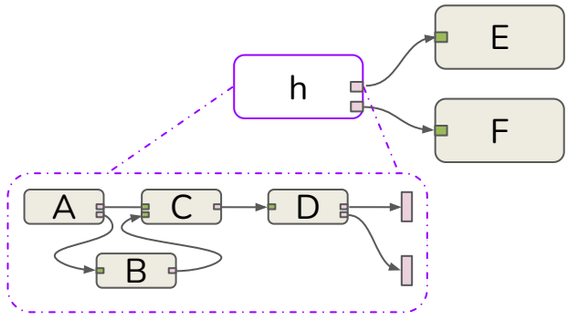
```
Receive();
F(out1_D);
```

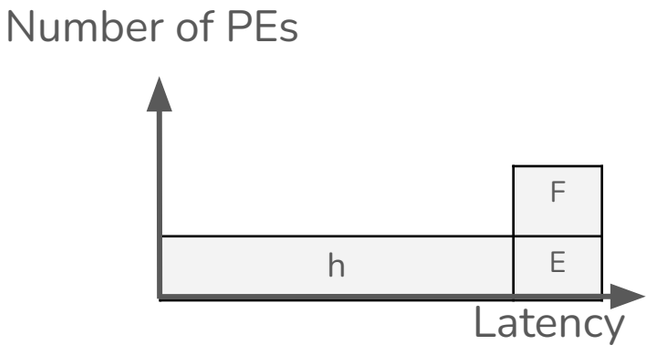
Core1.c

Number of PEs



Code generation with the clustering method






```
h(out0_h,out1_h);
Send();
E(out0_h);
```

Core0.c

```
Receive();
F(out1_h);
```

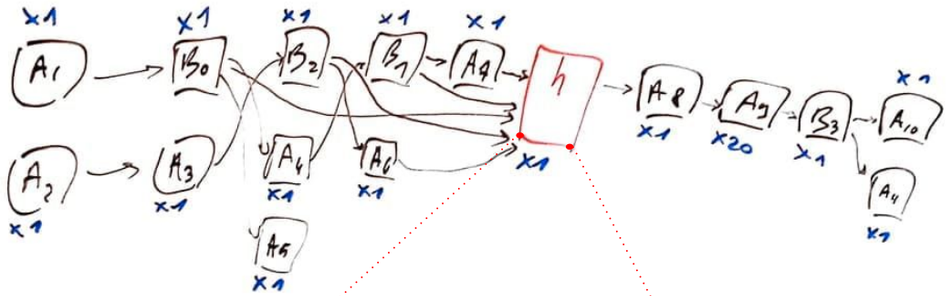
Core1.c

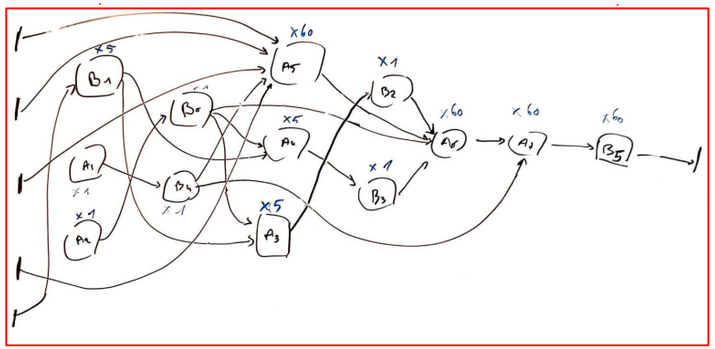
```
h(...){
    for(int index = 0; index < instance; index++){
        A(out0_A+ index *size, out1_A+ index *size);
        B(out1_A+ index *size,out0_B+ index *size);
        C(out0_A+ index *size,out0_B+ index *size,out0_C+ index *size);
        D(out0_C+ index *size,out0_h+ index *size,out1_h+ index *size);
    }
}
```

h.c

Stereo application

Picture/video

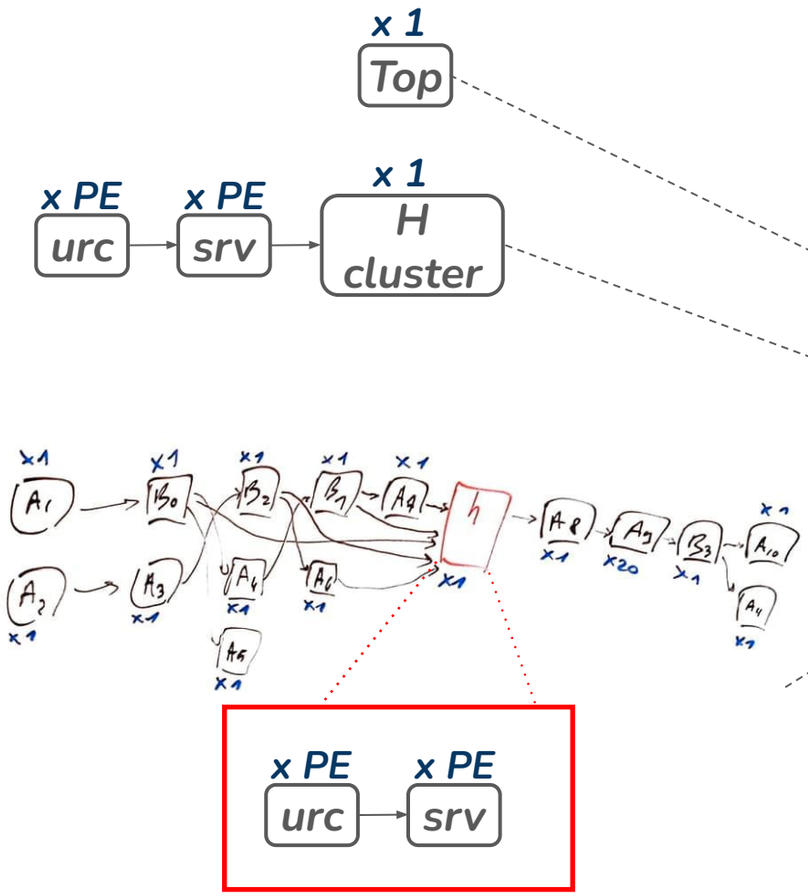




Depth map

- ☑ various forms of parallelism
- ☑ various firing instance
- ☑ hierarchy

Set up

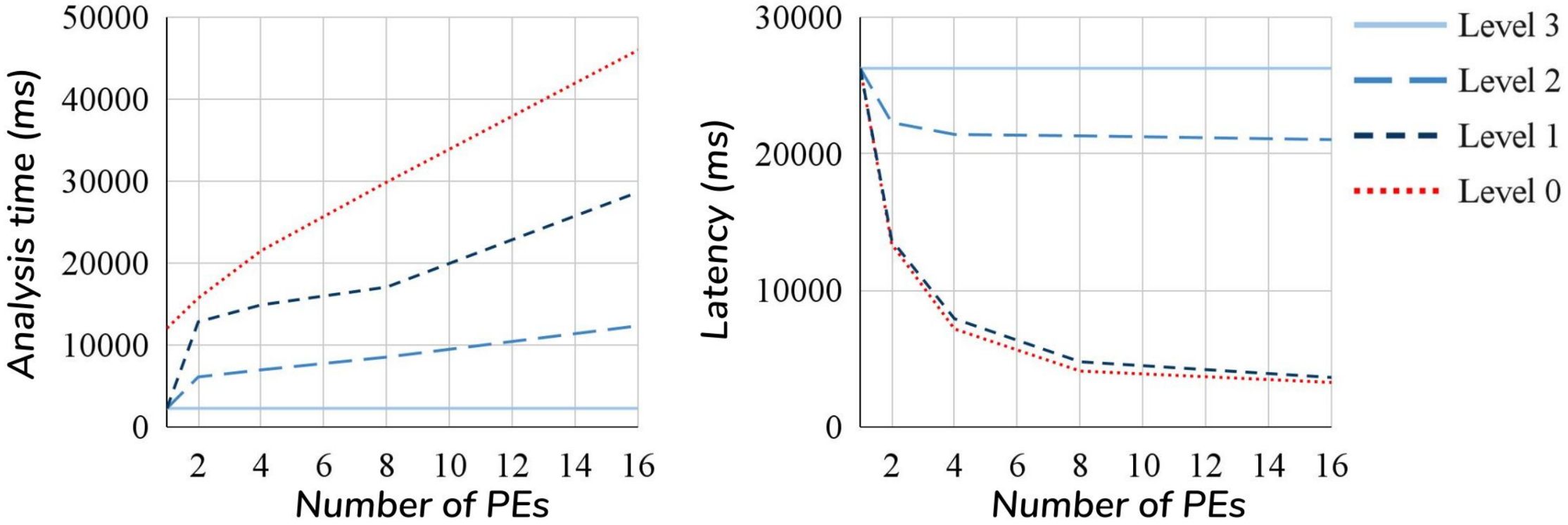


SrDAG

	Number of PEs				
	1	2	4	8	16
Level3			1		
Level2	1	14	16	22	32
Level1	1	70	76	88	108
Level0			187		

Configuration without clustering

Comparison of configuration with/without clustering



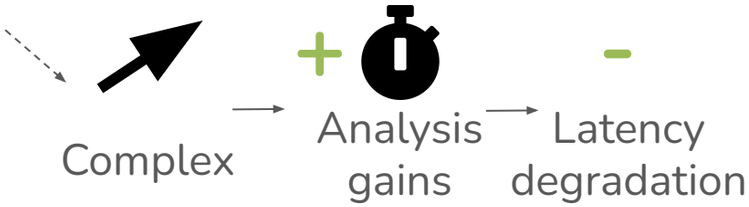
- The complexity of the **Level 0** configuration provides the best mapping opportunities
- The latency of the **Level 1** clustering configuration is slightly deteriorated for a significantly improved analysis time

Application	SDF	Level	SrDAG	Relative time	Number of PEs 1	2	4	8	16
Stereo	28	2	187	analysis	5.3	1.2	1.4	1.7	1.6
				latency	1.0	0.9	0.9	0.8	0.8
Stabilization	22	3	98	analysis	1.5	0.5	0.5	0.6	0.7
				latency	1.0	1.0	0.7	0.7	0.8
Squeezenet	98	3	5452	analysis*	203.5k	100.5	94.5	84.2	68.8
				latency*	1.0	1.0	1.0	1.0	1.0

*Estimated values

Table 1. Comparison of analysis time and latency between the classic flattening approach and best configurations of SCAPE method on three use-cases

Speedup: v > 1
Speeddown: v < 1



Summary conclusion

Scaling up of **C**lusters of **A**ctors on **P**rocessing **E**lements :

- ✓ Set of clustering configuration with different levels of granularity
- ✓ Tradeoff between analysis time and latency
- ✓ Implemented into PREESM and available on GitHub

Future work

- Identifying and clustering more complex patterns on each hierarchical levels
- Extending clustering to other types of parallelism

Thanks for your attention!
Any questions?