synchronous dataflow for free (SDF^3)

and other tool developments

Sander Stuijk, Marc Geilen, Twan Basten
m.c.w.geilen@tue.nl

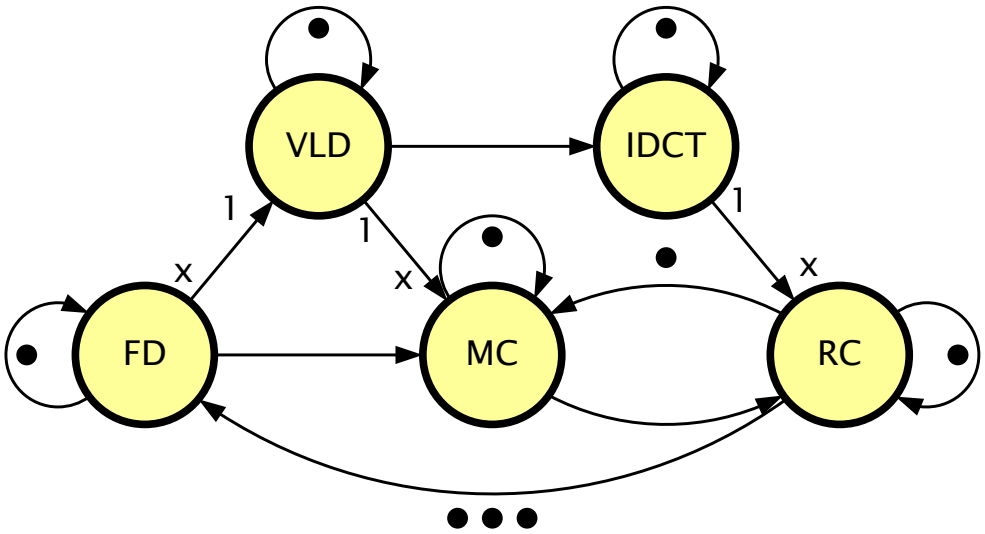
Map2MPSoC Workshop, November 28, 2008

overview

- tools TU/e
  - **SDF³** (extensions)
  - Pareto Calculator
  - MAMPS
- developments
  - dynamism, scenarios, SADF
  - memory mapping
- relation to other work
- SDF3 demo

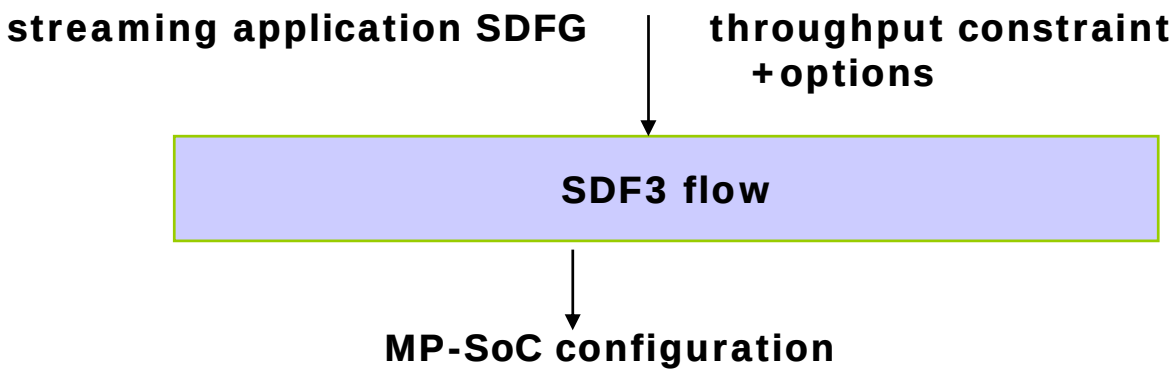
mocc

- model-based approach to mapping and scheduling with performance guarantees
- dataflow model of computation
- specifically (extensions of) synchronous dataflow (border of decidability in Christian's/Twan Basten's slide)

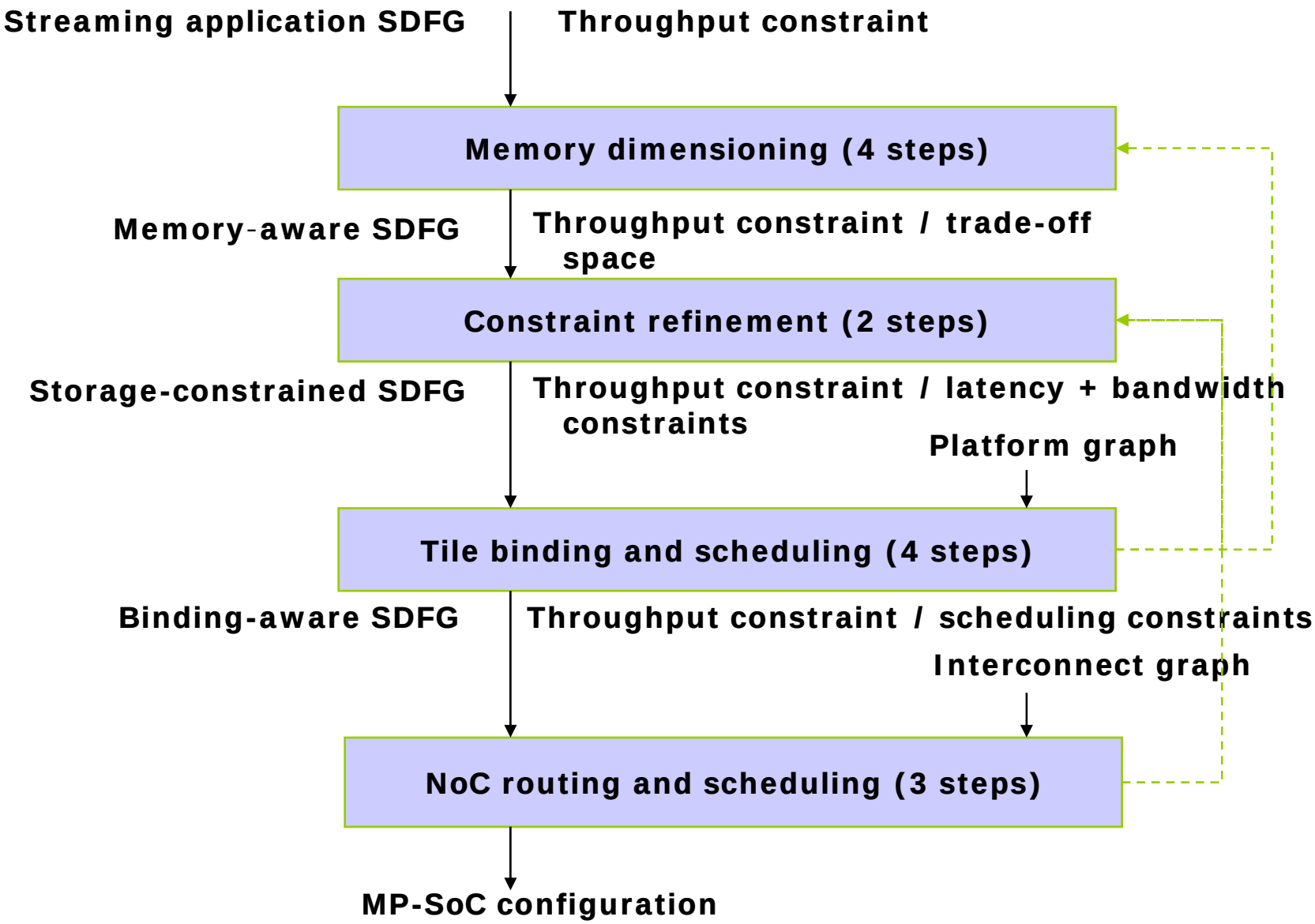


what do we have?

- yes, it's a 'model to model' flow…
- but a fairly realistic one…
- …and some generic dataflow analysis tools

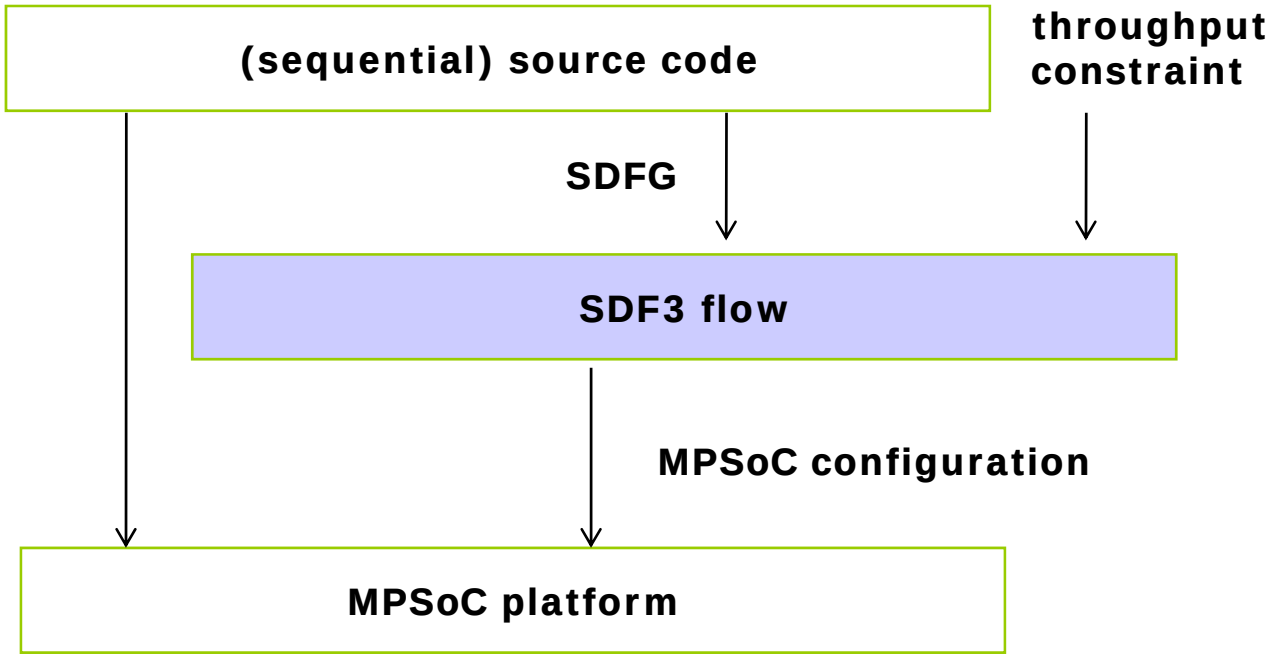


SDFG-based MP-SoC design-flow



what we want to get

- to use model and flow for source code to realisation flow
- to allow more dynamism in the model



SDF

- synchronous dataflow model of computation
- model applications conservatively
- model resources, fifos, NoCs
- model predictable arbitration, TDMA, latency-rate
- analysis algorithms
  - efficient throughput calculation
  - latency
  - deadlock, liveness, boundedness
  - parametric throughput analysis

SDF^3: SDF For Free

SDF3 offers

- SDFG transformation and analysis algorithms
- an SDFG random graph generator, more powerful than TGFF
- links to visualization and simulation software for SDFGs
- advanced MP-SoC binding and scheduling functions for SDFGs

SDF3 functionality is available as

- command-line tools
- C/C++ API

other TU/e tools

- **Pareto Calculator**
  - supports compositional calculation of multi-dimensional trade-offs
  - trade-offs are used for instance also by Erlangen, Zürich, Daedalos
  - based on concept of Pareto optimality
  - implements **Pareto Algebra**
  - on the web: www.es.ele.tue.nl/pareto

other TU/e tools

- **MAMPS** (work by Akash Kumar)
  - **Multi-Application Multi-Processor Synthesis**
  - profiling based automatic mapping of multiple applications onto an FPGA based multiprocessor system with Microblaze cores and point-to-point connections.
  - uses Leiden's KPNGen for paralellisation.
  - on the web: www.es.ele.tue.nl/mamps

relation to other work

- dataflow models used by several partners.
  Leiden, Bologna, NXP, Zürich, Erlangen, ...
- can use our generic SDF analysis algorithms
- **use SDF^3 instead of Task Graphs For Free (TGFF) ! :-)**
- Pareto analysis and trade-offs. Genereric multi-dimensional trade-off analysis and desing-space exploration .
  compute Pareto front, approximations, compositional calculations, fancy things like partially ordered objectives

scenarios

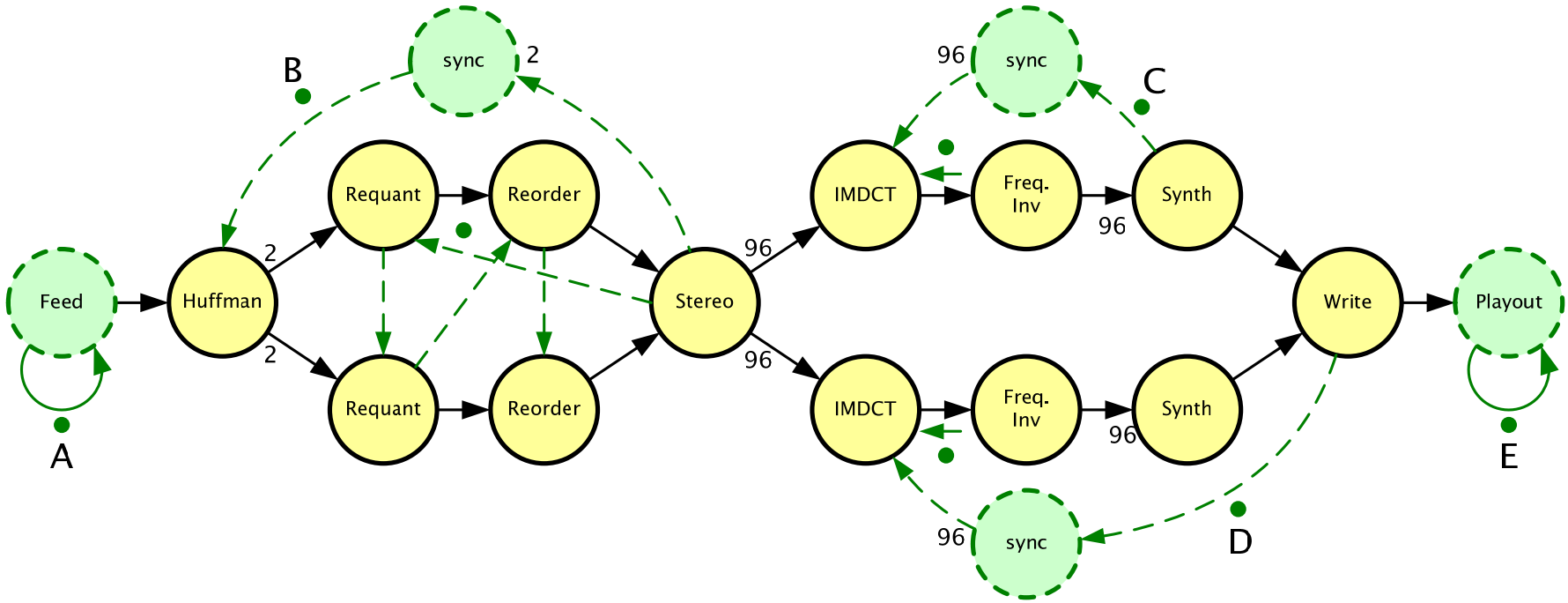
- scenarios: situations similar from a resource usage point of view
- for instance frame types or block types
- if we can detect scenarios, we can exploit them
- for dataflow models: every scenario has its own dataflow graph
  (sometimes that is the same graph structure, but with different execution times)
- Scenario Aware Dataflow
  synchronous dataflow graphs are (too?) static

scenario aware dataflow

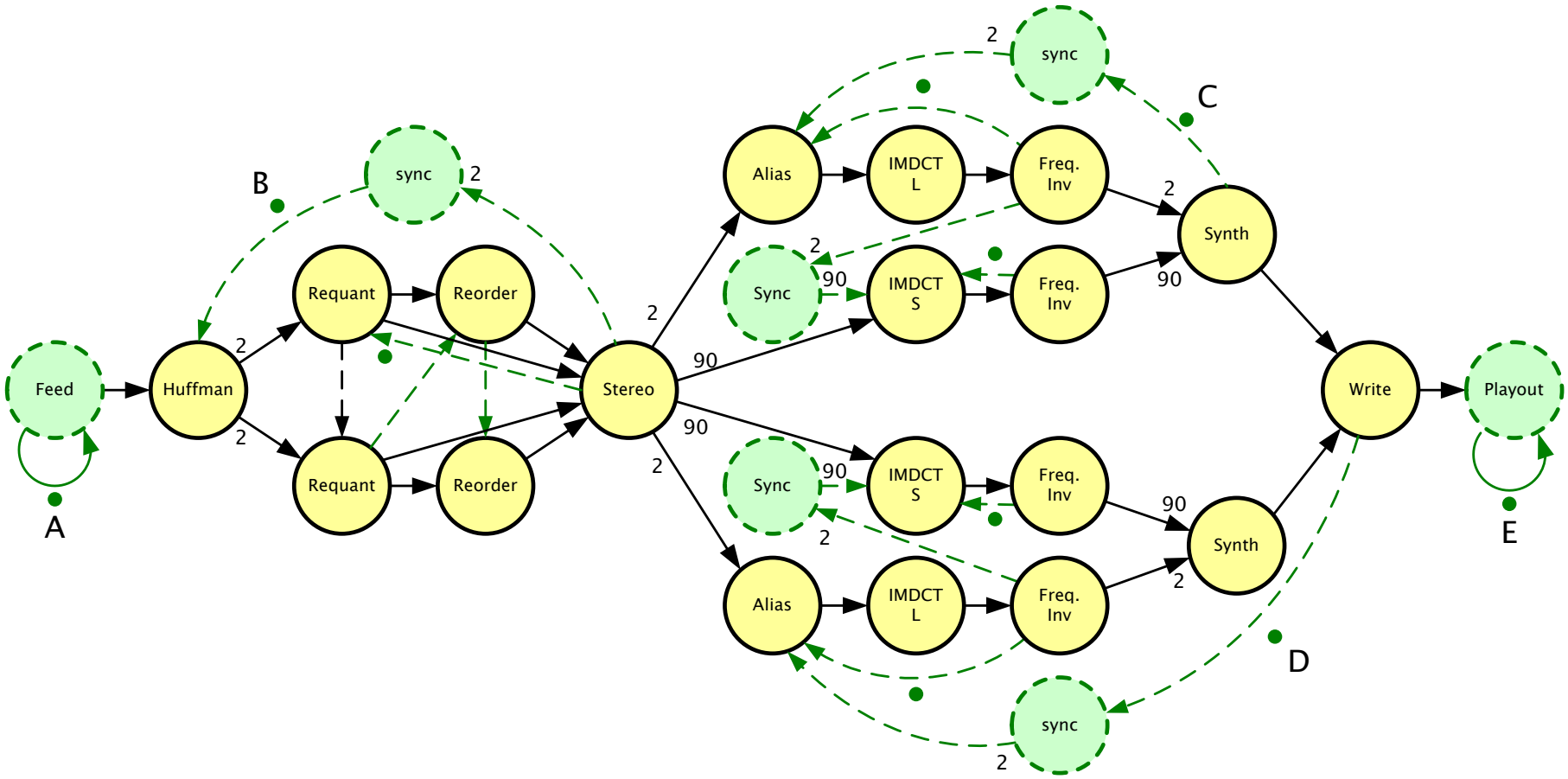
- Synchronous Data Flow is attractive for its analysability
- ...but it is often too static

- dynamism in SDF can be modelled as different **scenarios of ordinary SDF behaviour**
- **system switches between scenarios**
- there is no free lunch:
  analyse scenario switches
- we can analyse SADFs using Max-Plus algebra
  (not yet in SDf3)

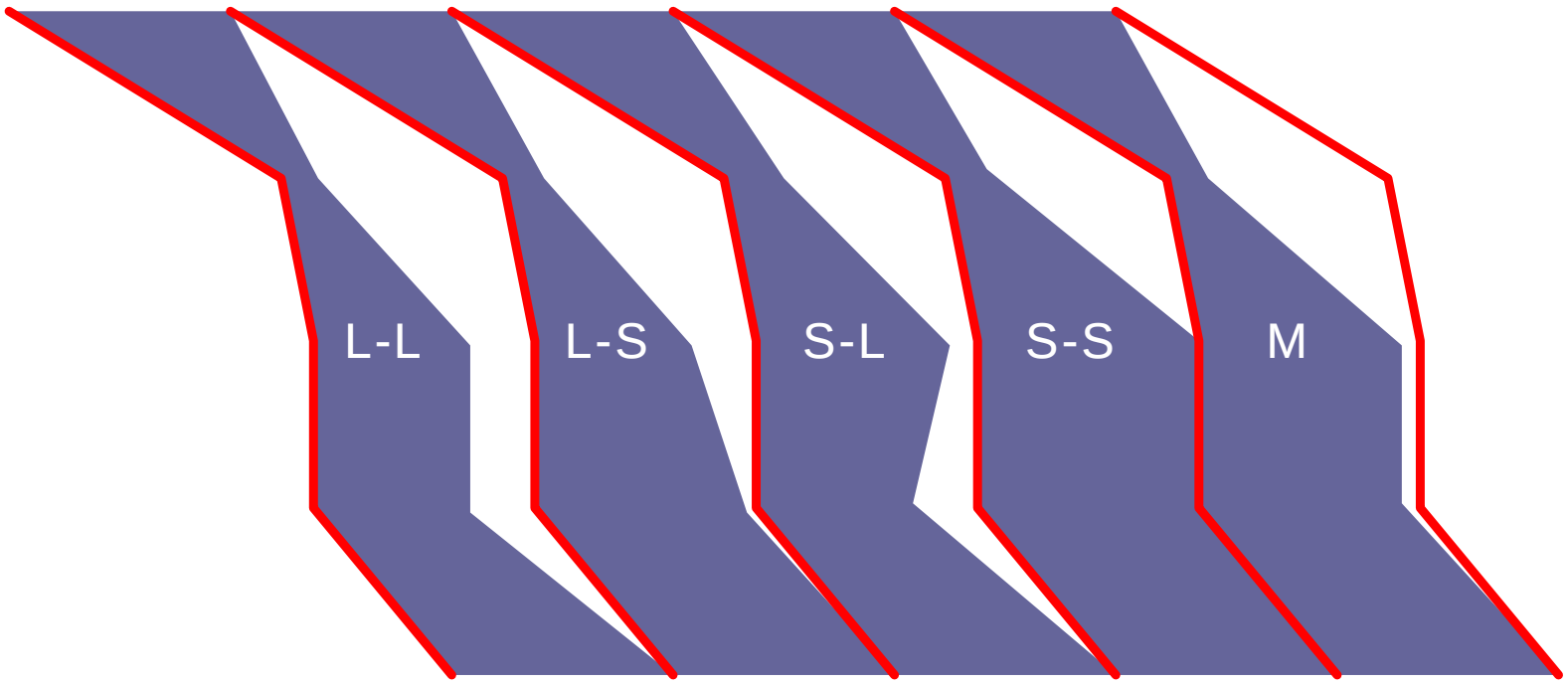
mp3 short block scenario



mp3 mixed block scenario



mp3 result



SDF^3: SDF For Free

Department of Electrical Engineering
Electronic Systems

SDF^3: SDF For Free

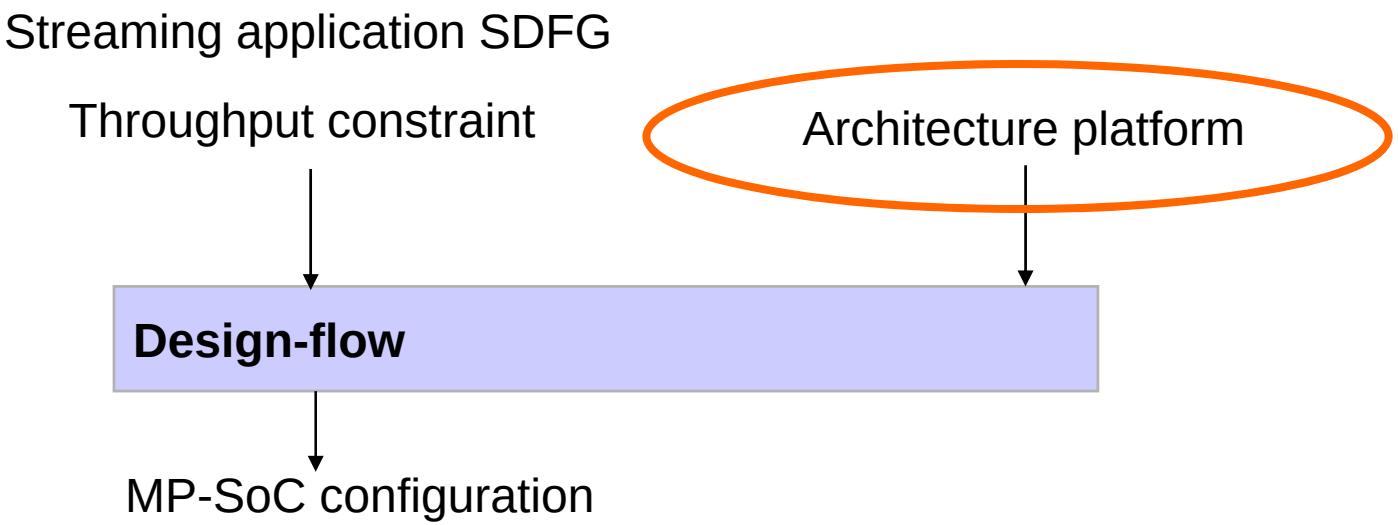
SDF3 offers

- SDFG transformation and analysis algorithms
- an SDFG random graph generator
- links to visualization and simulation software for SDFGs
- advanced MP-SoC binding and scheduling functions for SDFGs

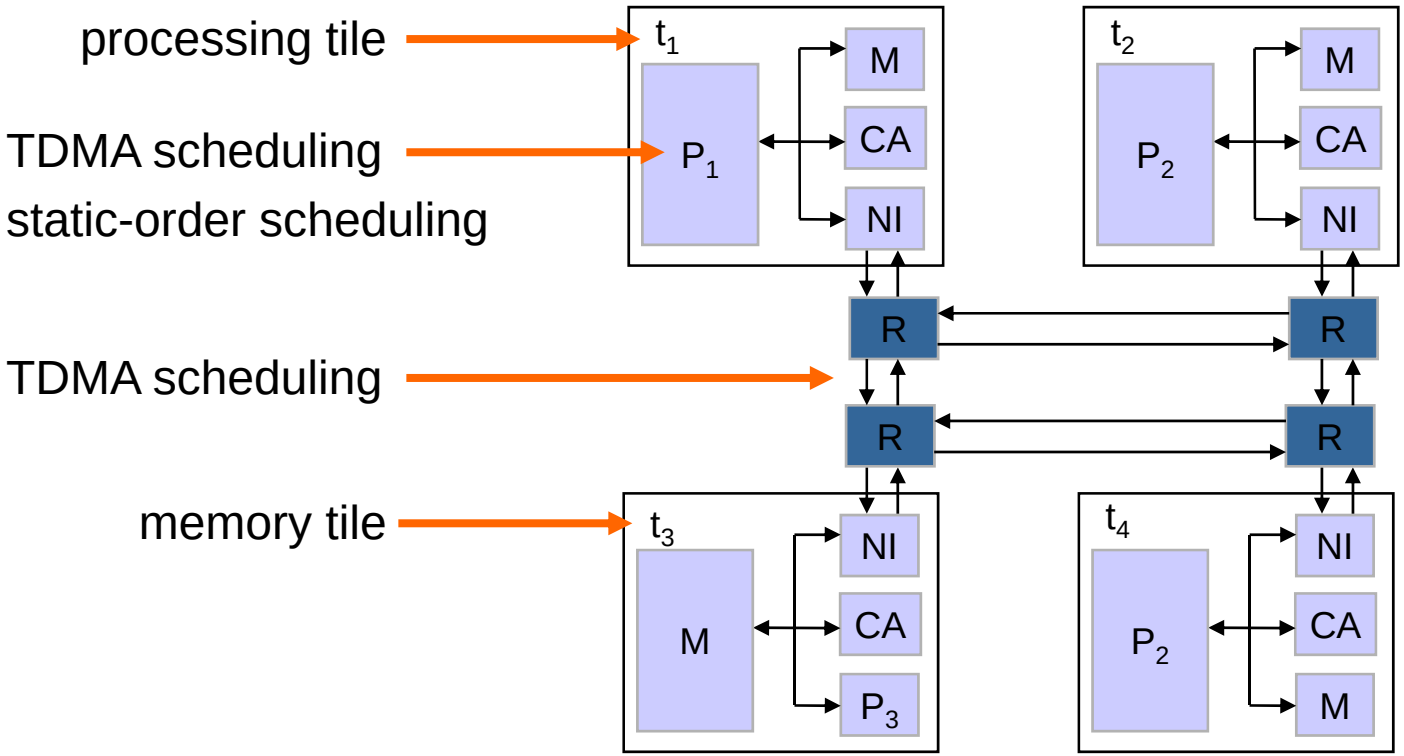
SDF3 functionality is available as

- command-line tools
- C/C++ API

SDFG-based MP-SoC design-flow (SDF^3)

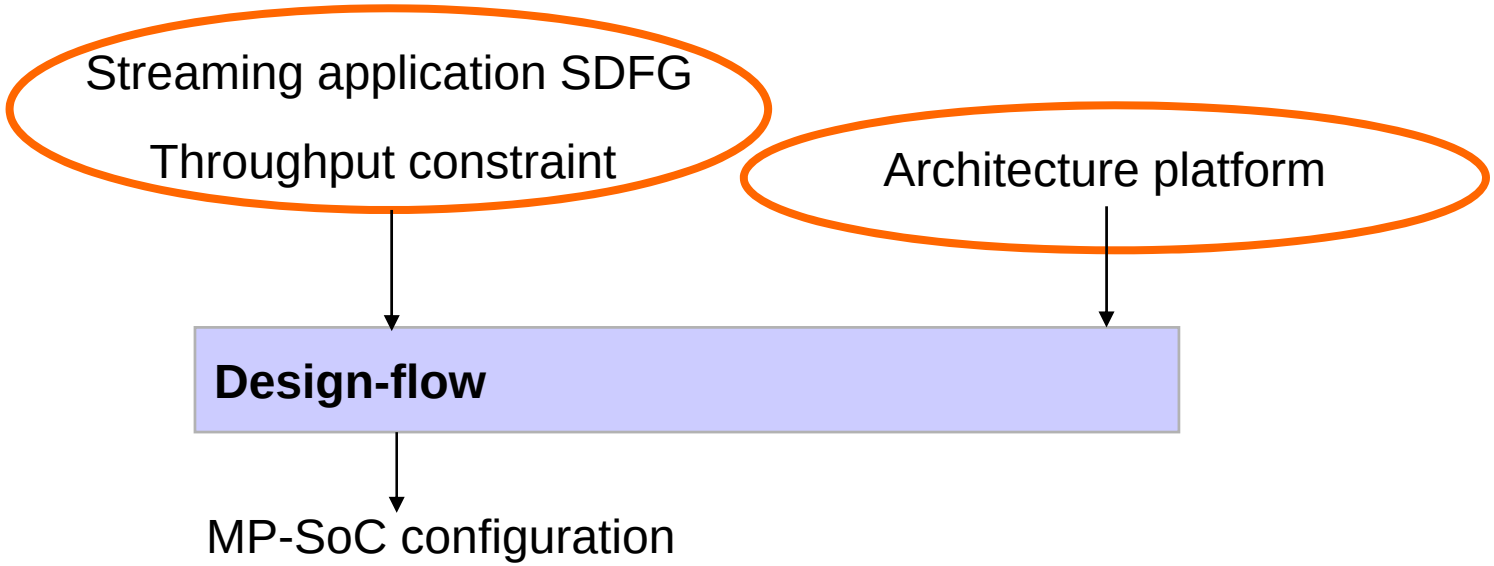


Predictable platform

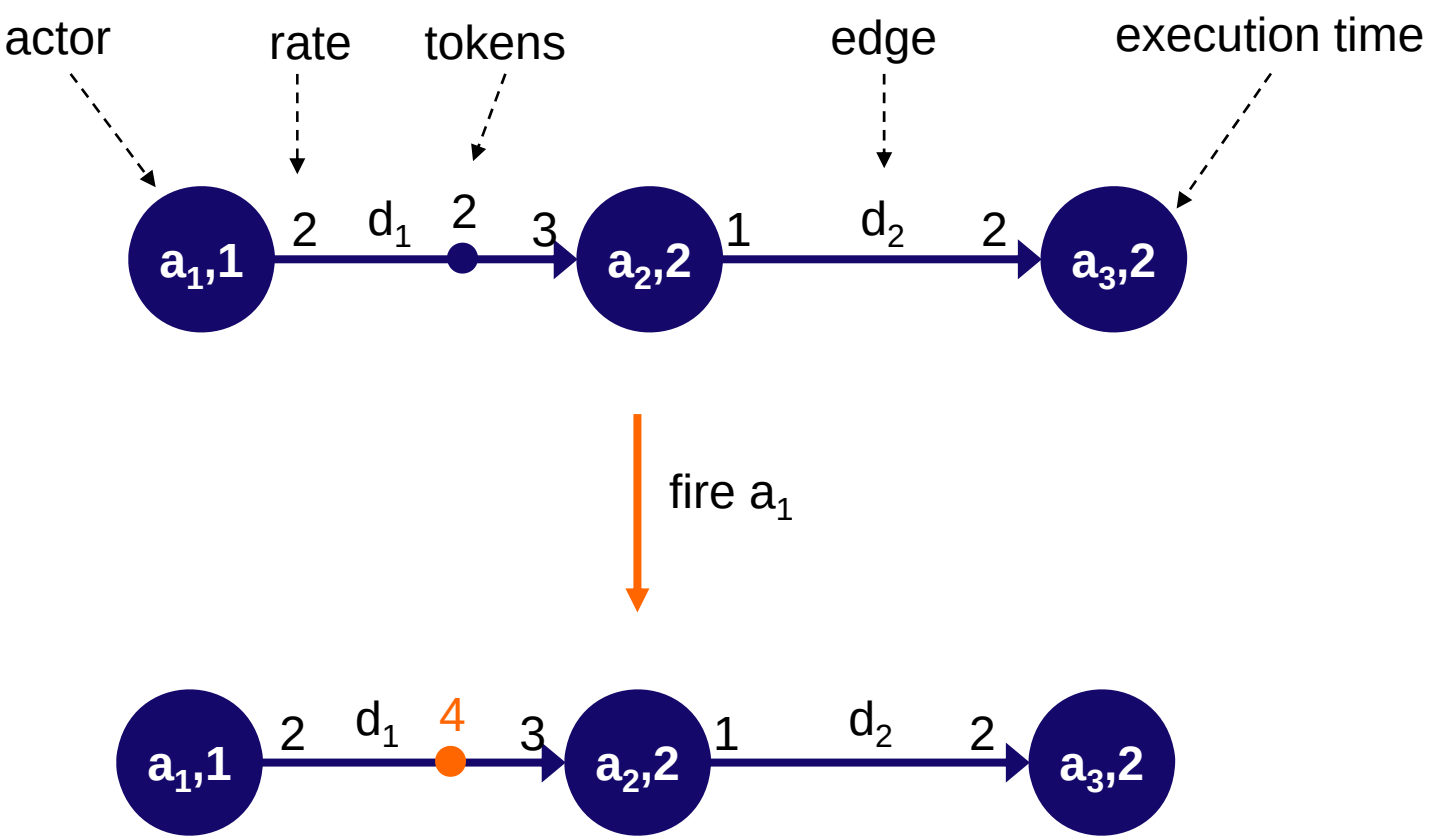


	processor		memory	network interface		
	type	wheelsize	size	#conn	in bw	out bw
t_1	P_1	100	1000	10	96	96
t_2	P_2	100	5000	10	96	96
t_3	P_3	100	10000	10	96	96
t_4	P_4	100	500	10	96	96

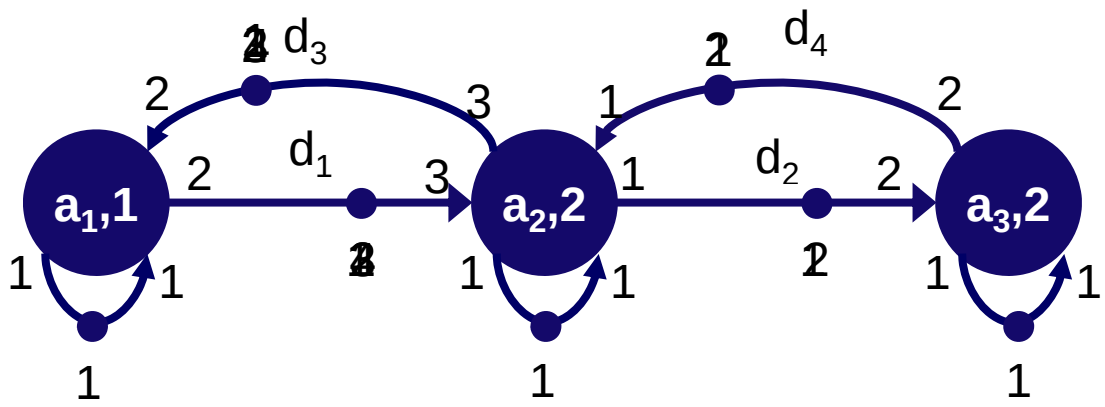
SDFG-based MP-SoC design-flow



Synchronous Dataflow Graphs

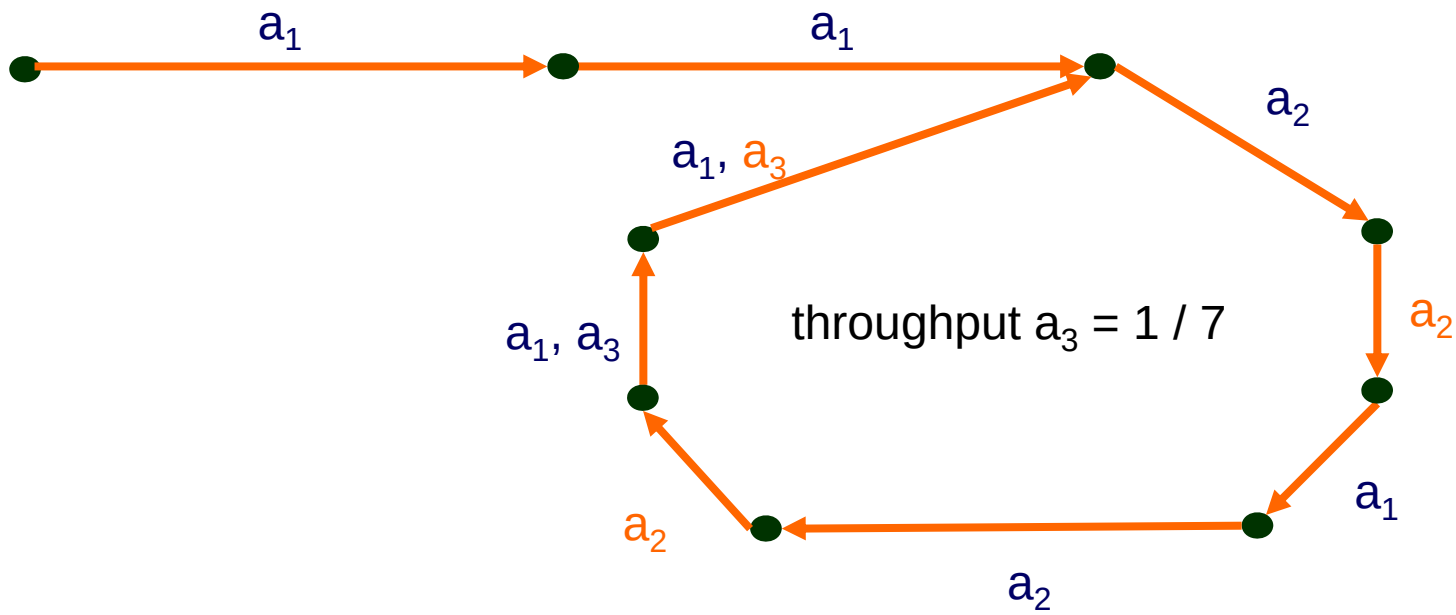


Timed SDF state space

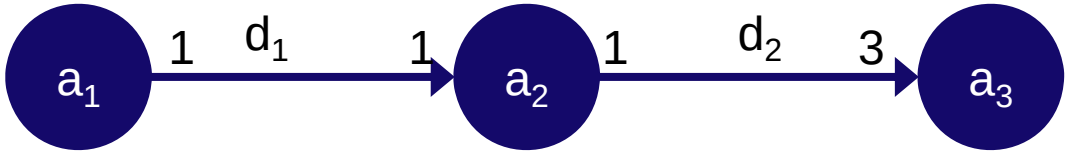


$\langle d_1, d_2 \rangle \rightarrow \langle 4,2 \rangle$

State: $((d_1, d_2, d_3, d_4), (\{a_1\}, \{a_2\}, \{a_3\}))$



Streaming application SDFG

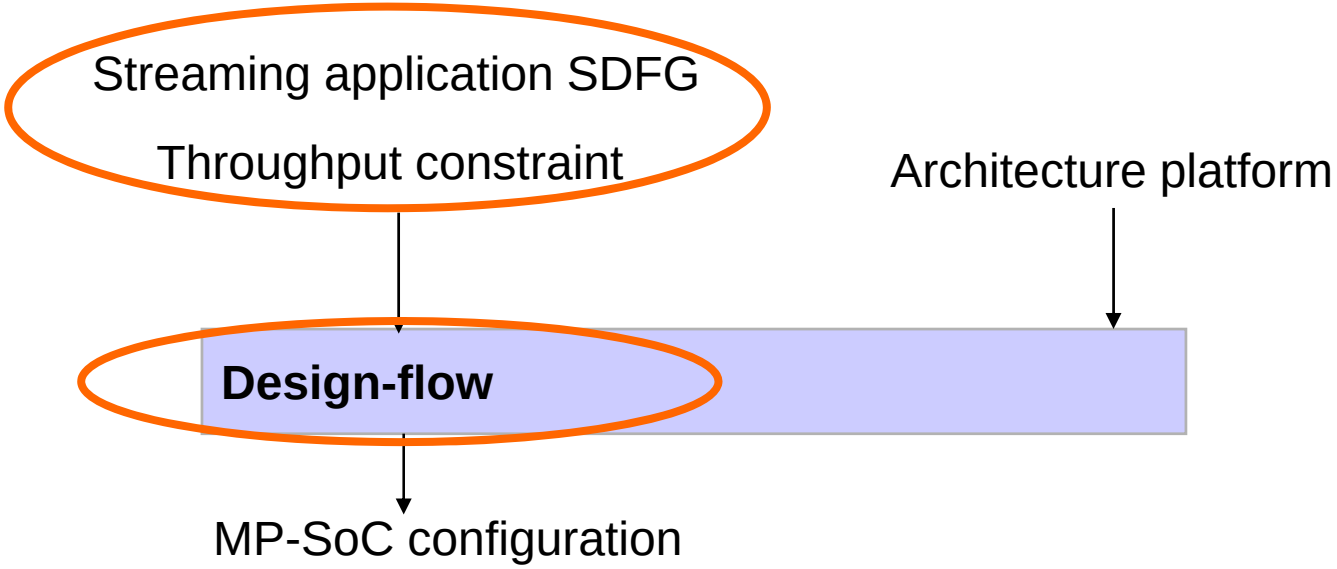


	execution time		memory	
	P_1	P_2	P_1	P_2
a_1	5	20	200	200
a_2	5	-	350	-
a_3	-	30	-	100

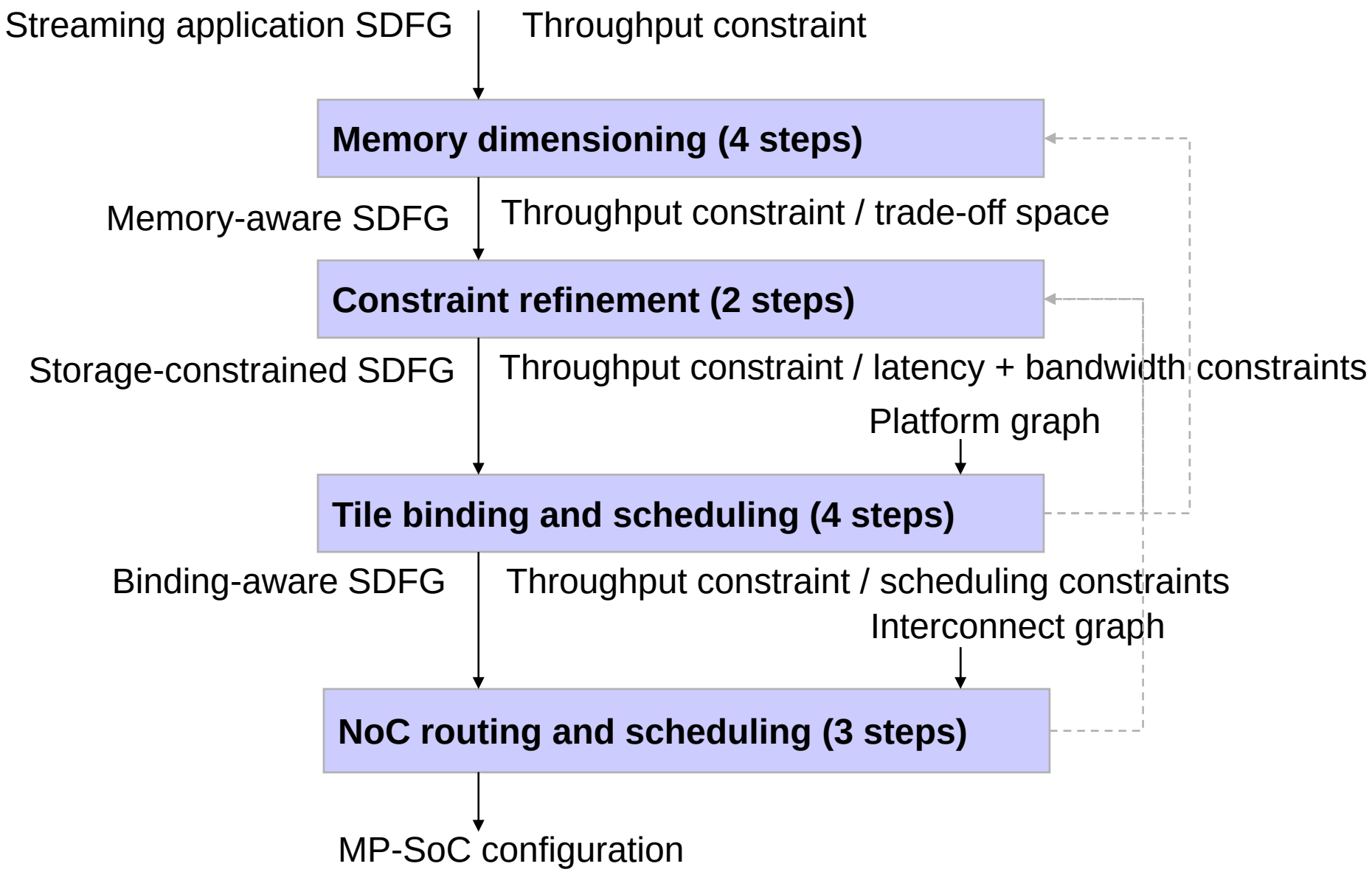
	token size
	sz
d_1	128
d_2	64

Throughput constraint: 0.0085 firings / time-unit

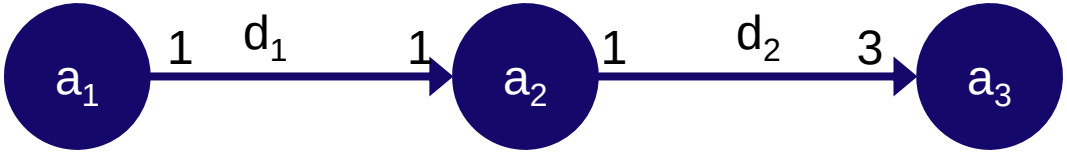
SDFG-based MP-SoC design-flow



SDFG-based MP-SoC design-flow

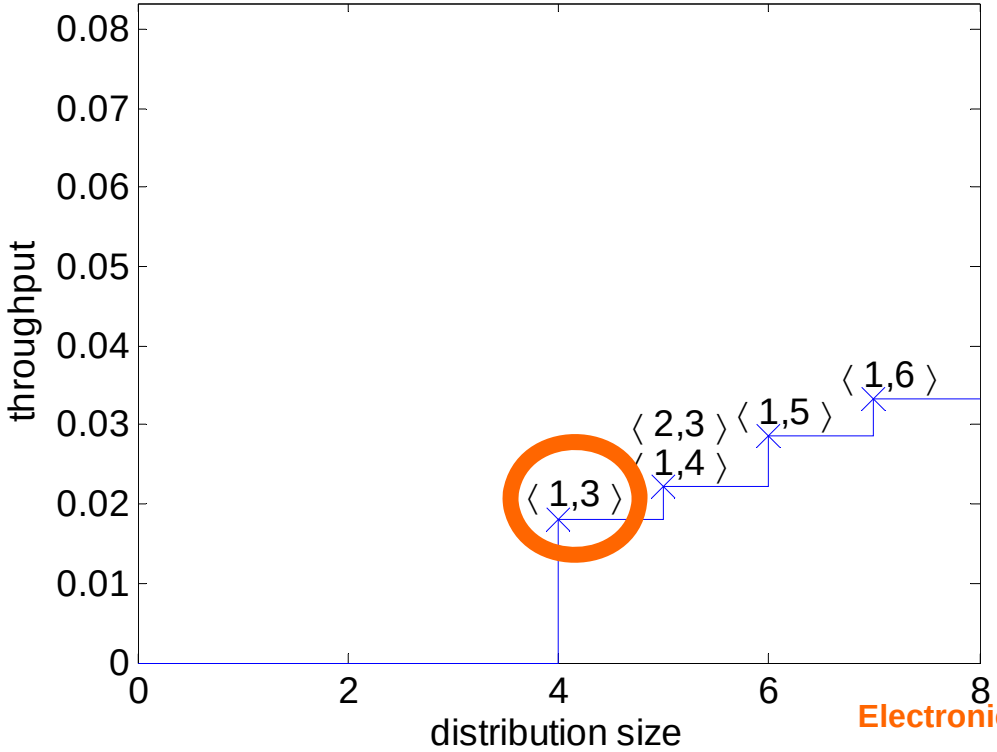


Memory dimensioning and constraint refinement

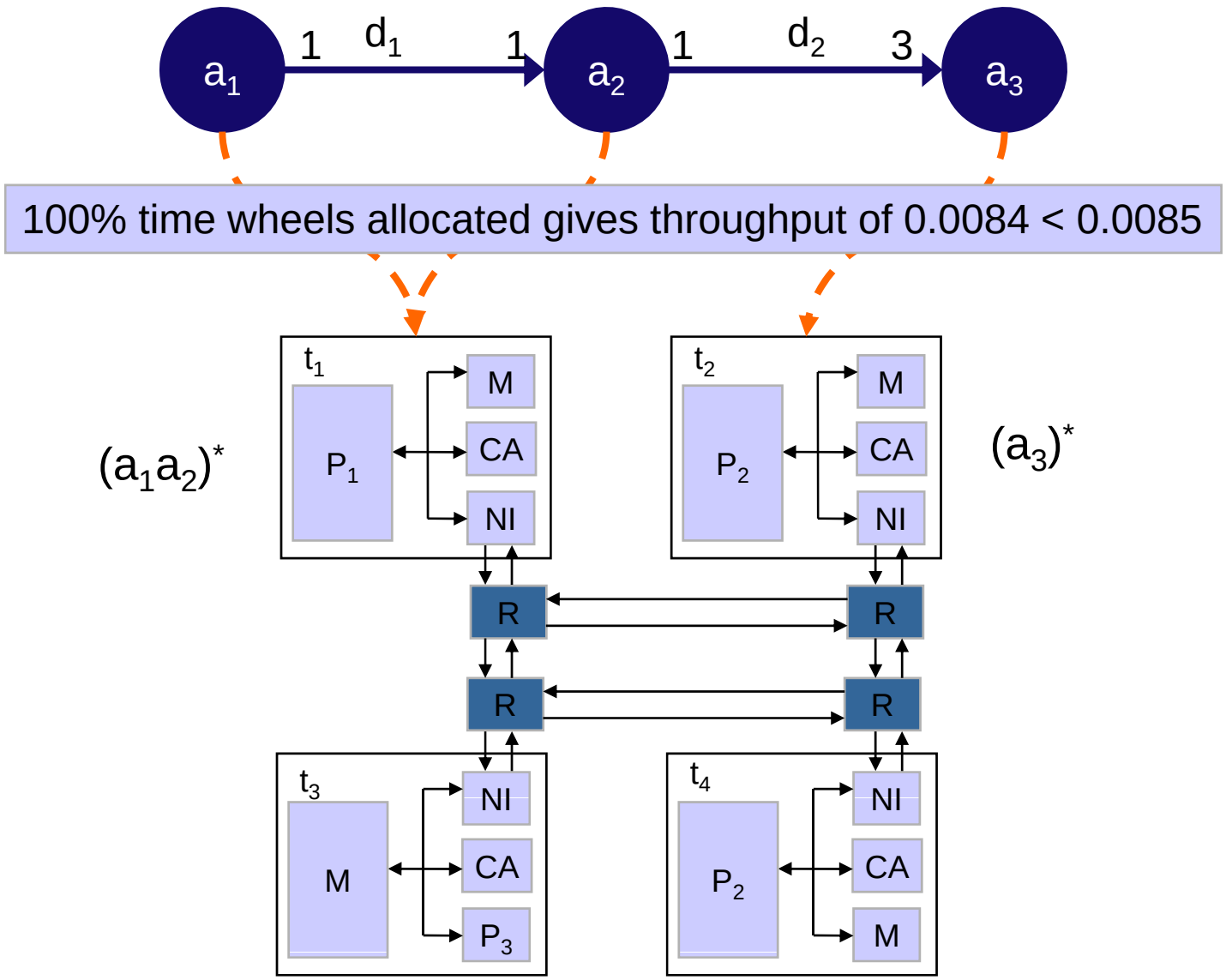


	token size	storage-space			latency	bandwidth
	sz	α_{mem}	α_{src}	α_{dst}	ρ	β
d_1	128	1	1	1	13	6.98
d_2	64	3	2	3	61	3.49

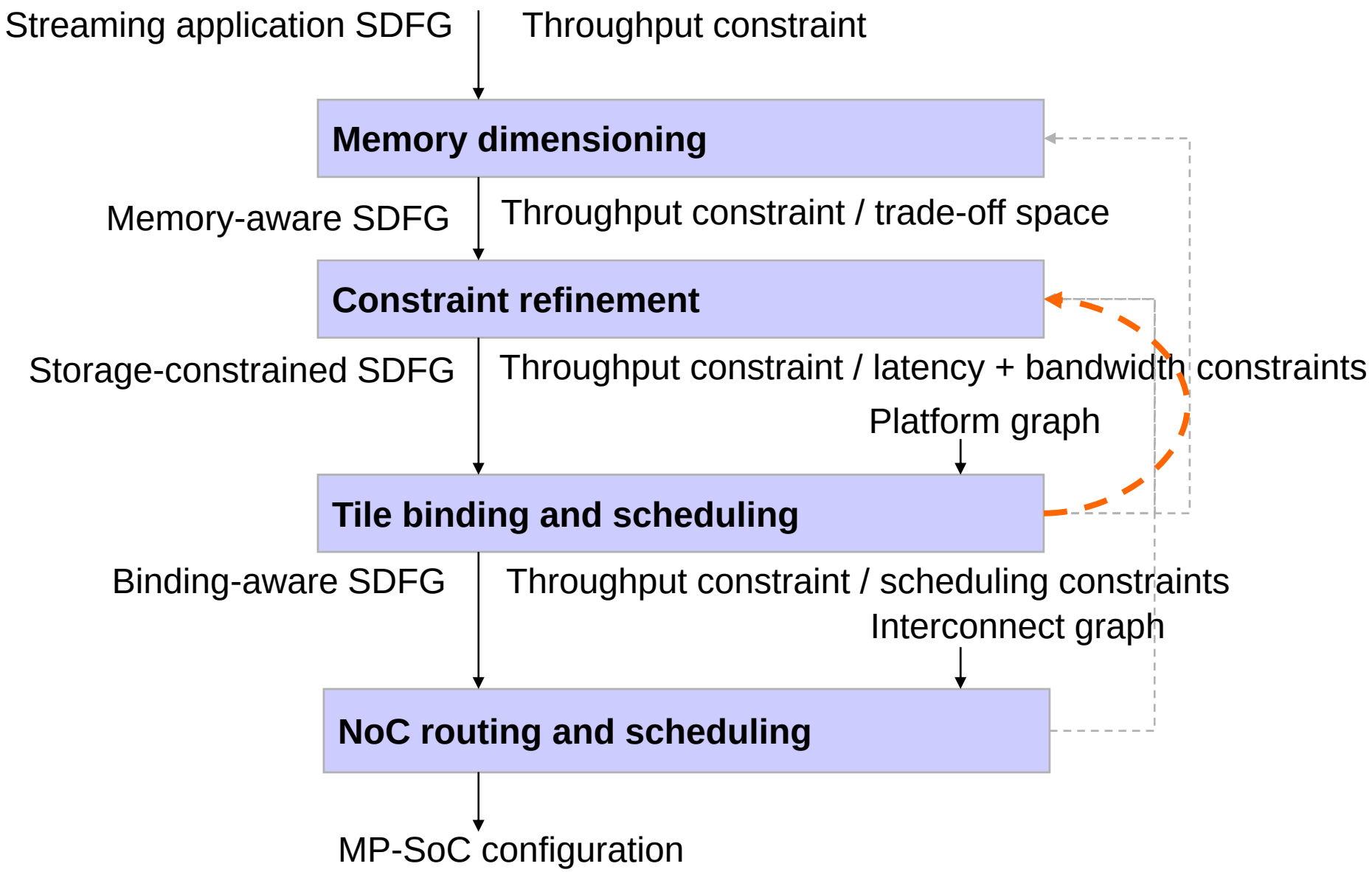
Throughput constraint:
0.0085 firings / time-unit



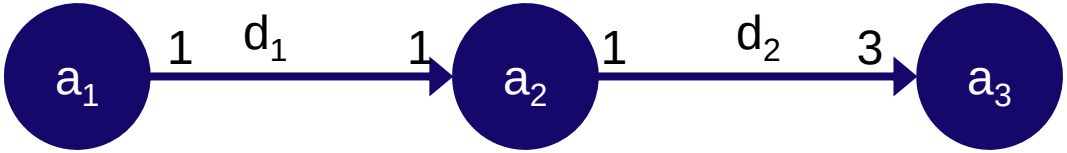
Tile binding and scheduling



SDFG-based MP-SoC design-flow

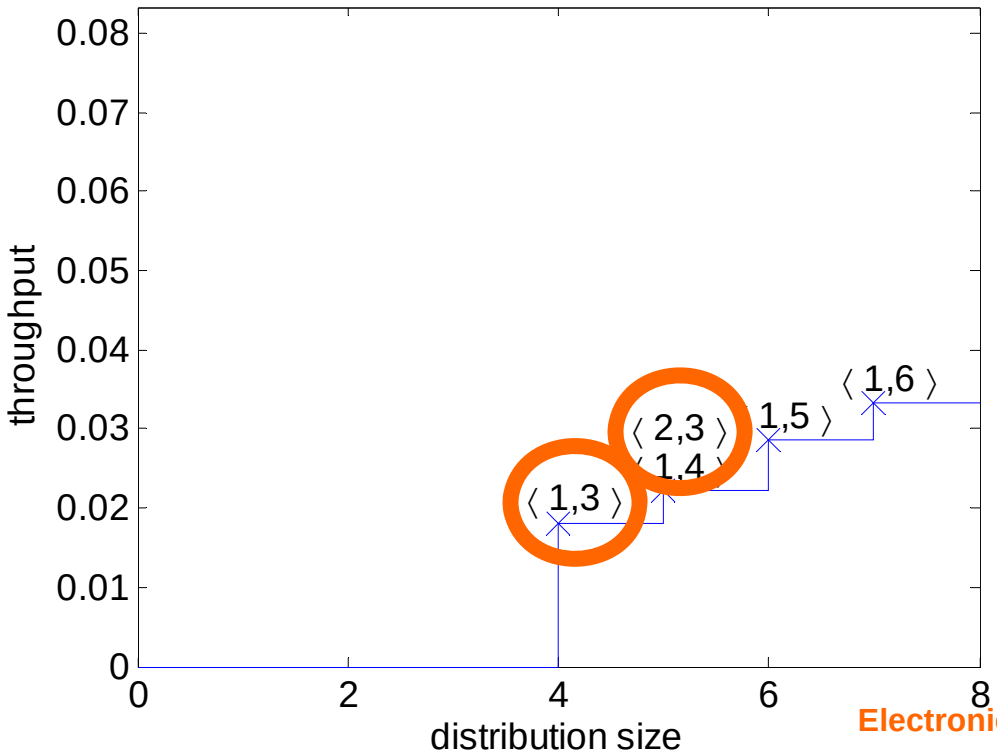


Memory dimensioning and constraint refinement

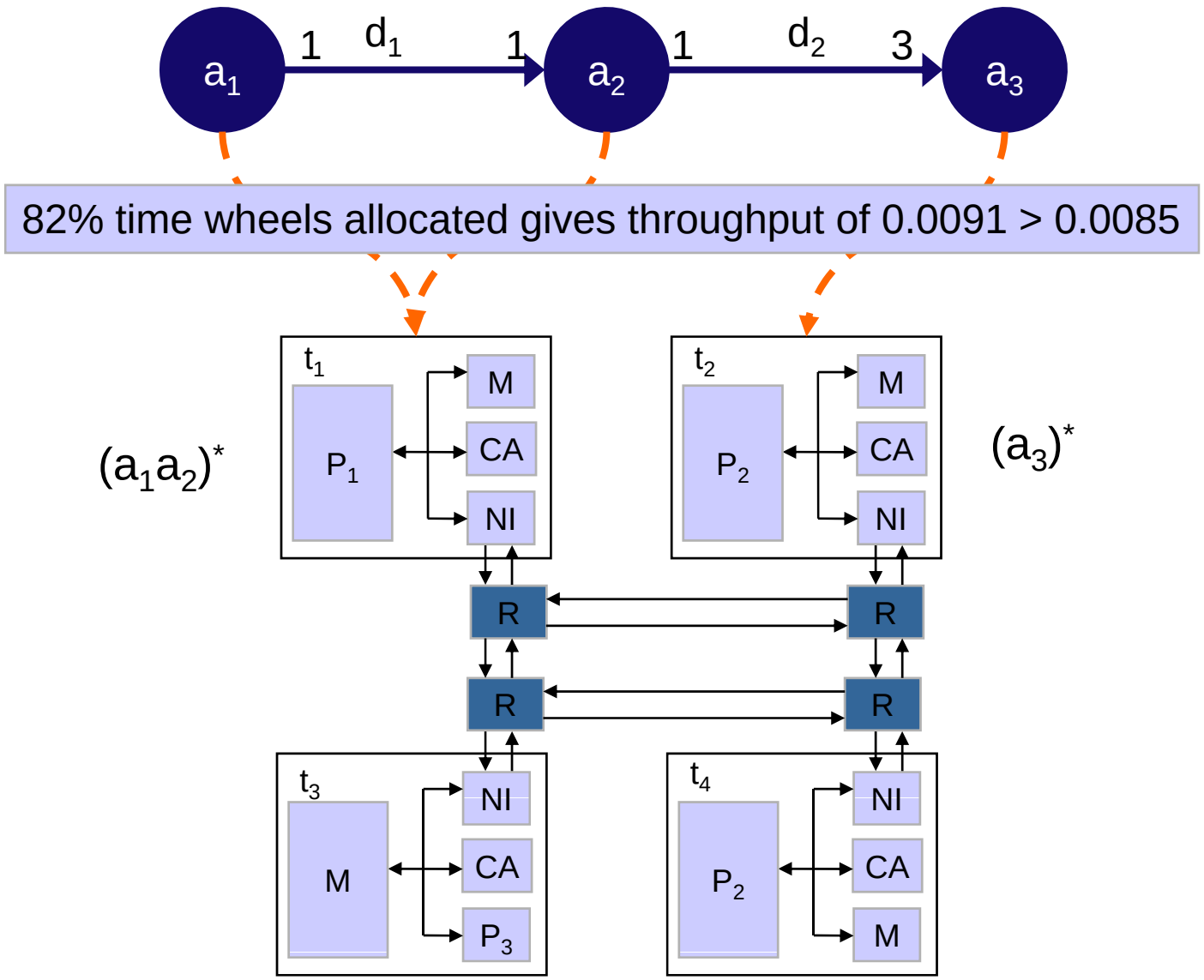


	token size	storage-space			latency	bandwidth
	sz	α_{mem}	α_{src}	α_{dst}	ρ	β
d_1	128	2	1	1	30	8.53
d_2	64	3	2	3	45	4.27

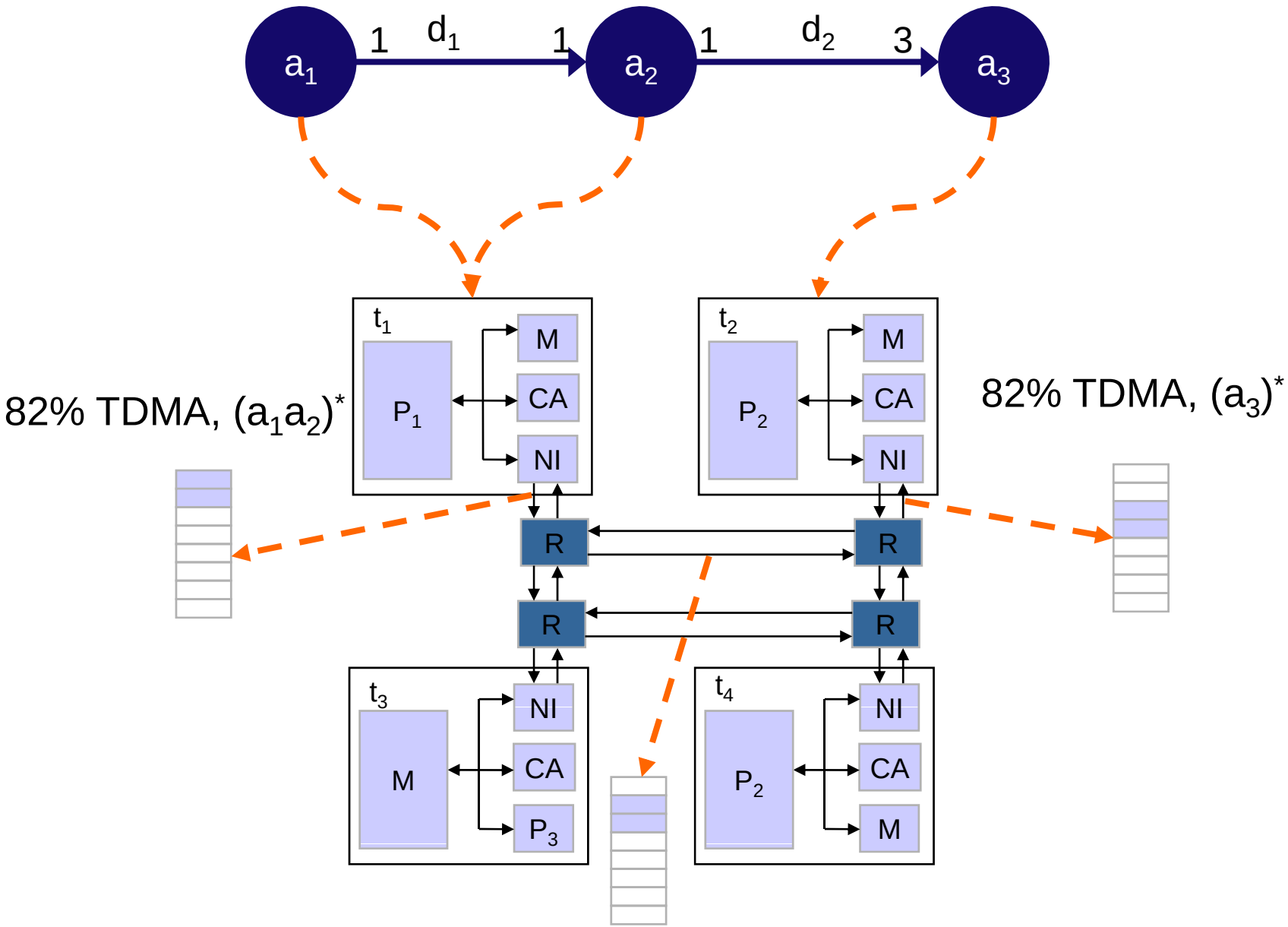
Throughput constraint:
0.0085 firings / time-unit



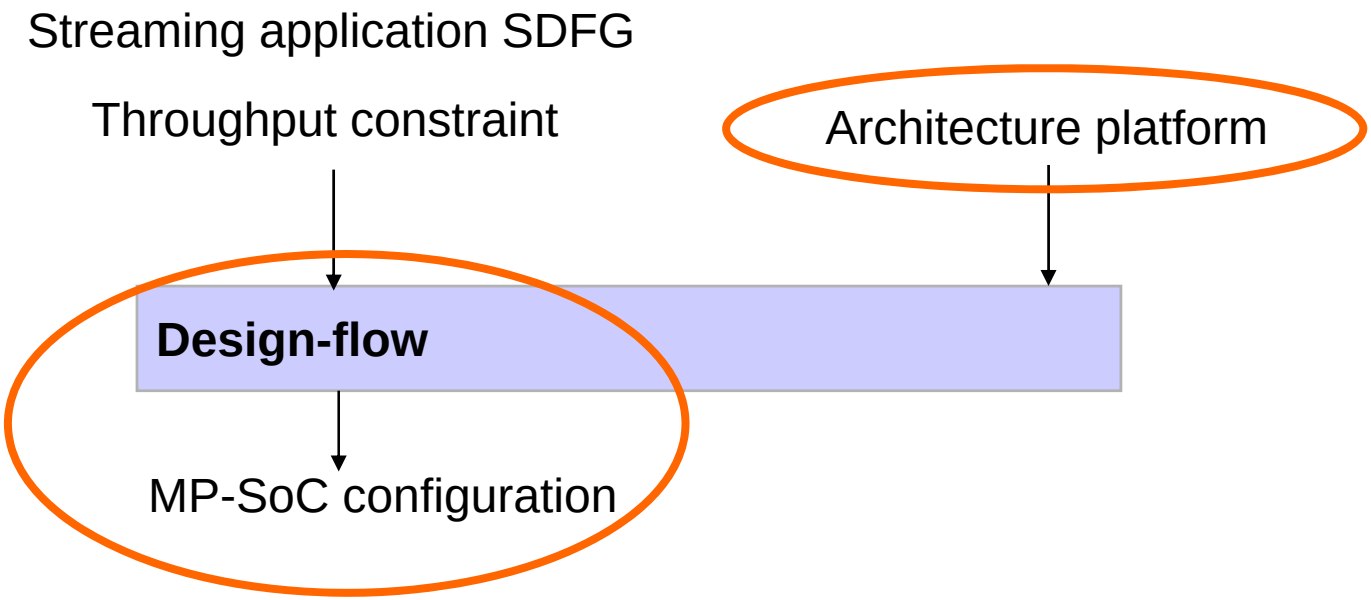
Tile binding and scheduling



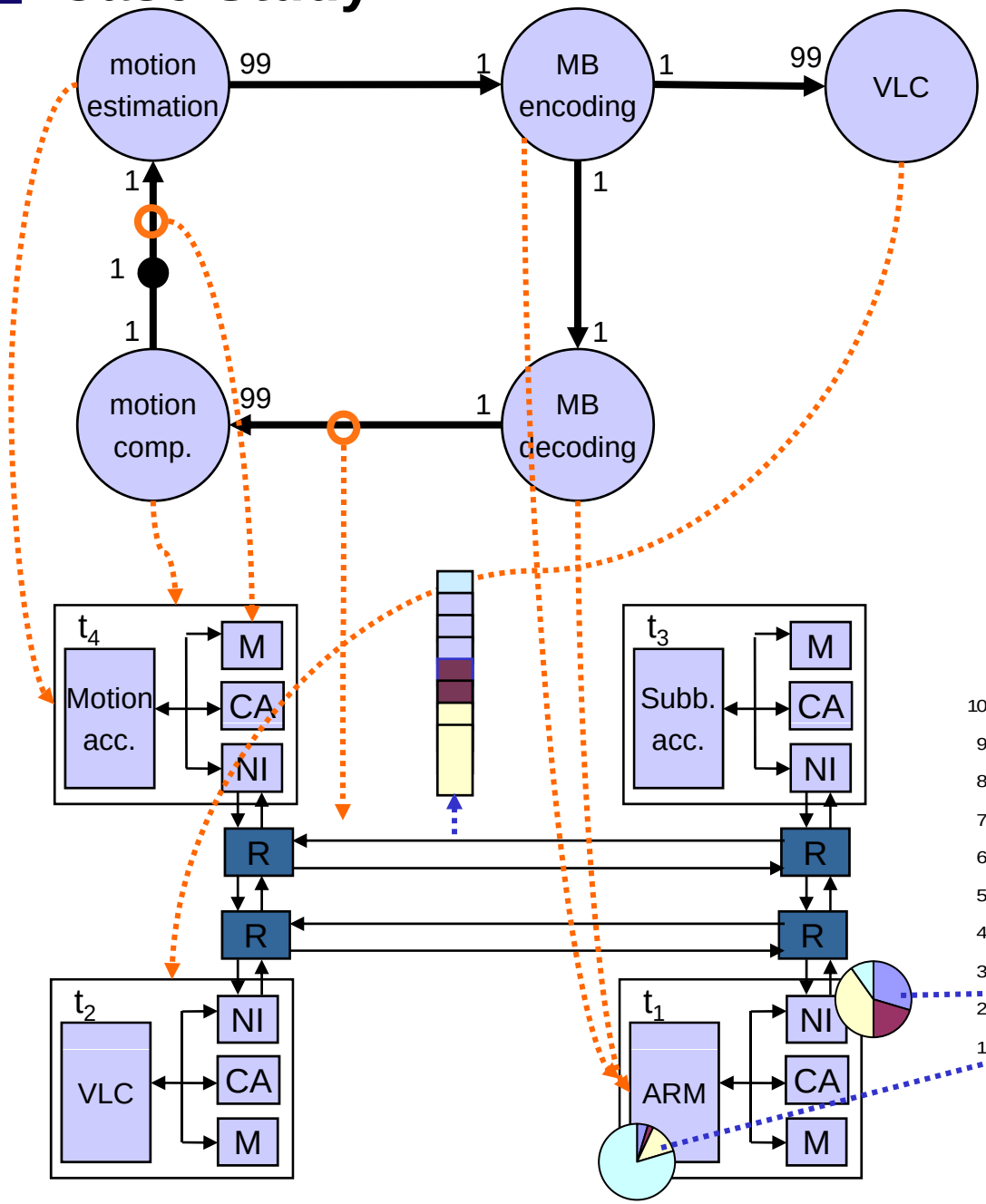
NoC routing and scheduling



SDFG-based MP-SoC design-flow



Case Study



Applications

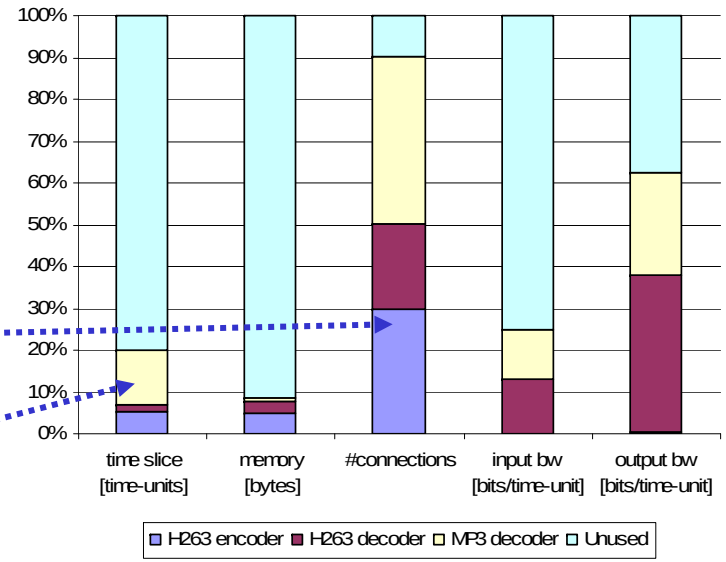
- H.263 encoder
- H.263 decoder
- MP3 decoder

Platform

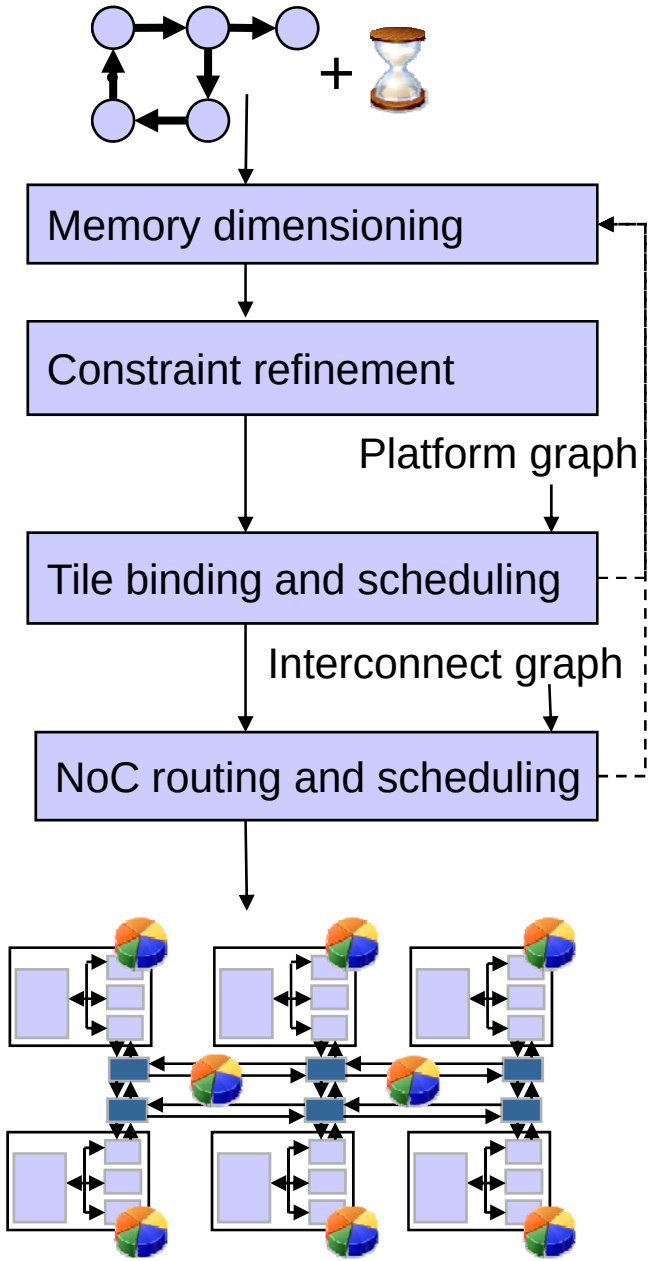
- ARM7 core
- Three accelerators

Views

- Mapping
- Resource budgets

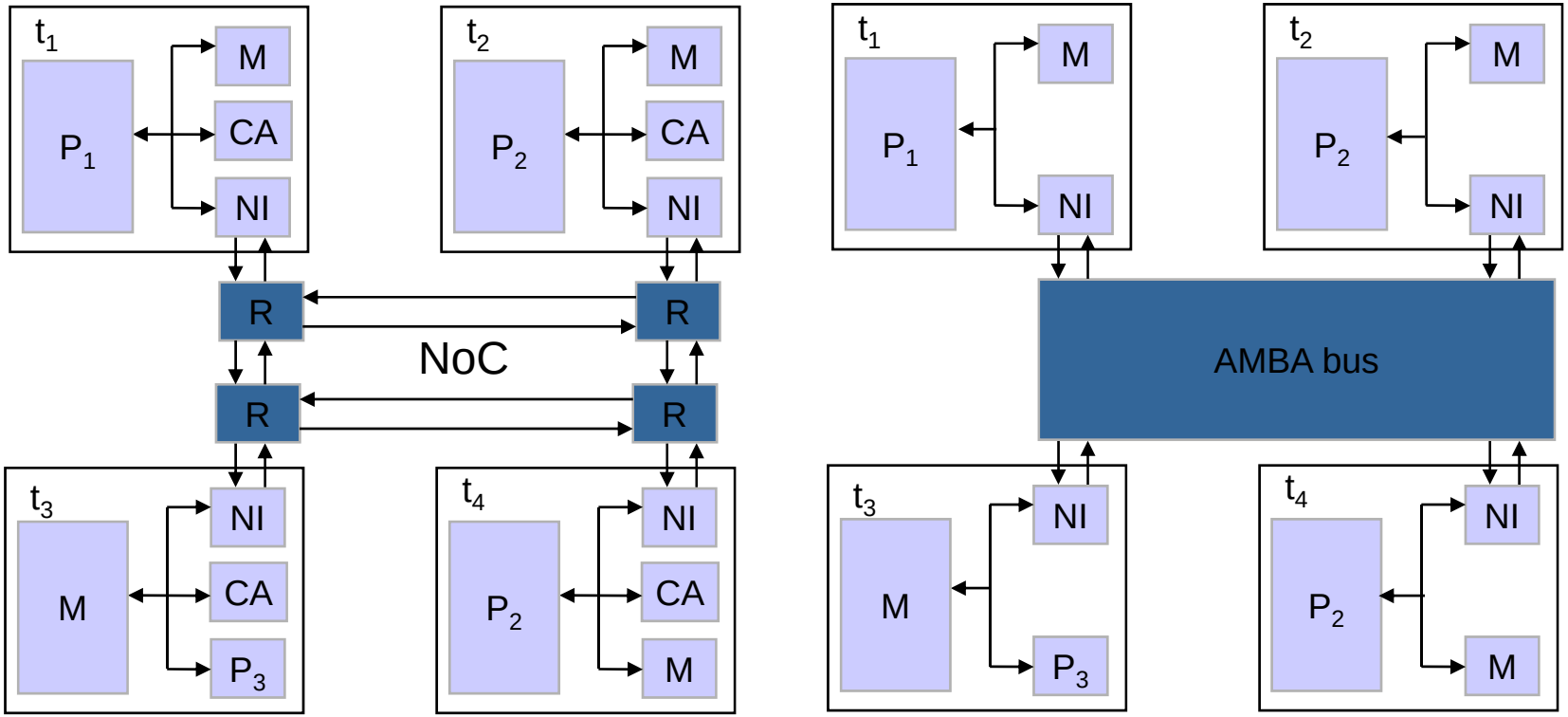


Case study



H.263 encoder	H.263 decoder	MP3 decoder
2ms	1611ms	143ms
1ms	0ms	1ms
287ms	816ms	55ms
125ms	261ms	5ms
415ms	**2688ms**	**203ms**

Predictable platform

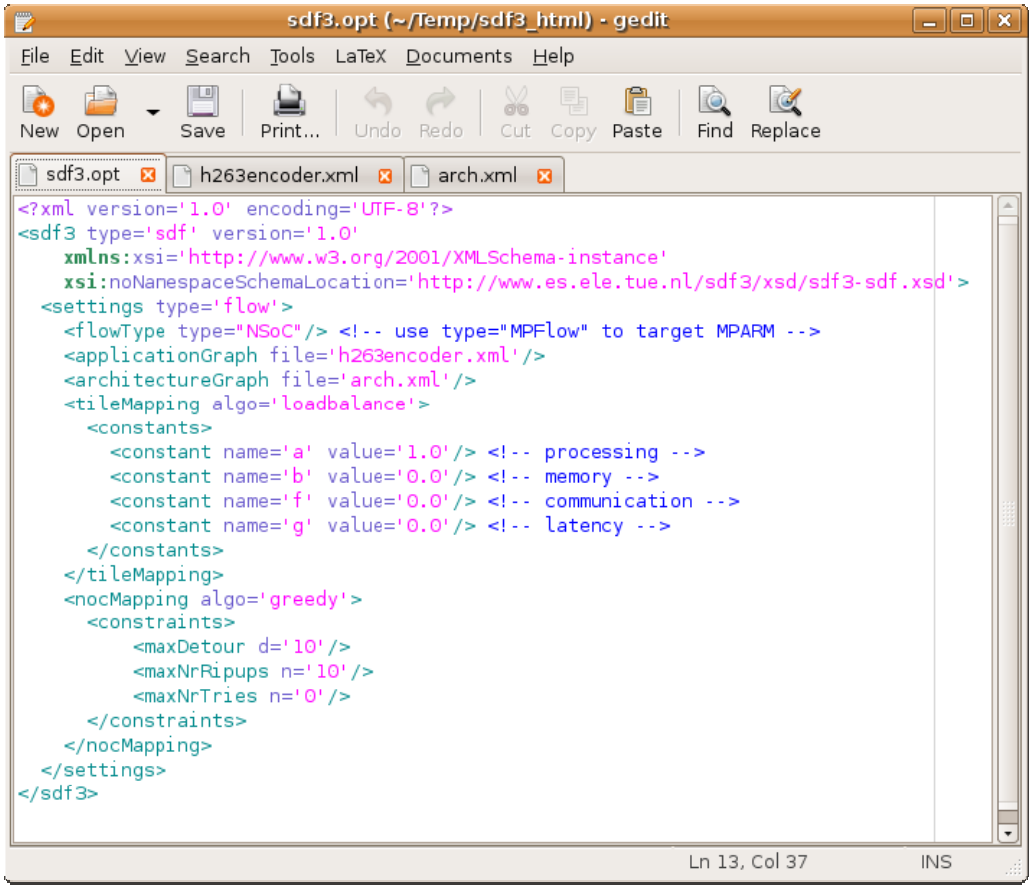


Design-flow supports two platforms:

- Network-on-Chip
- AMBA bus in combination with MP-Flow

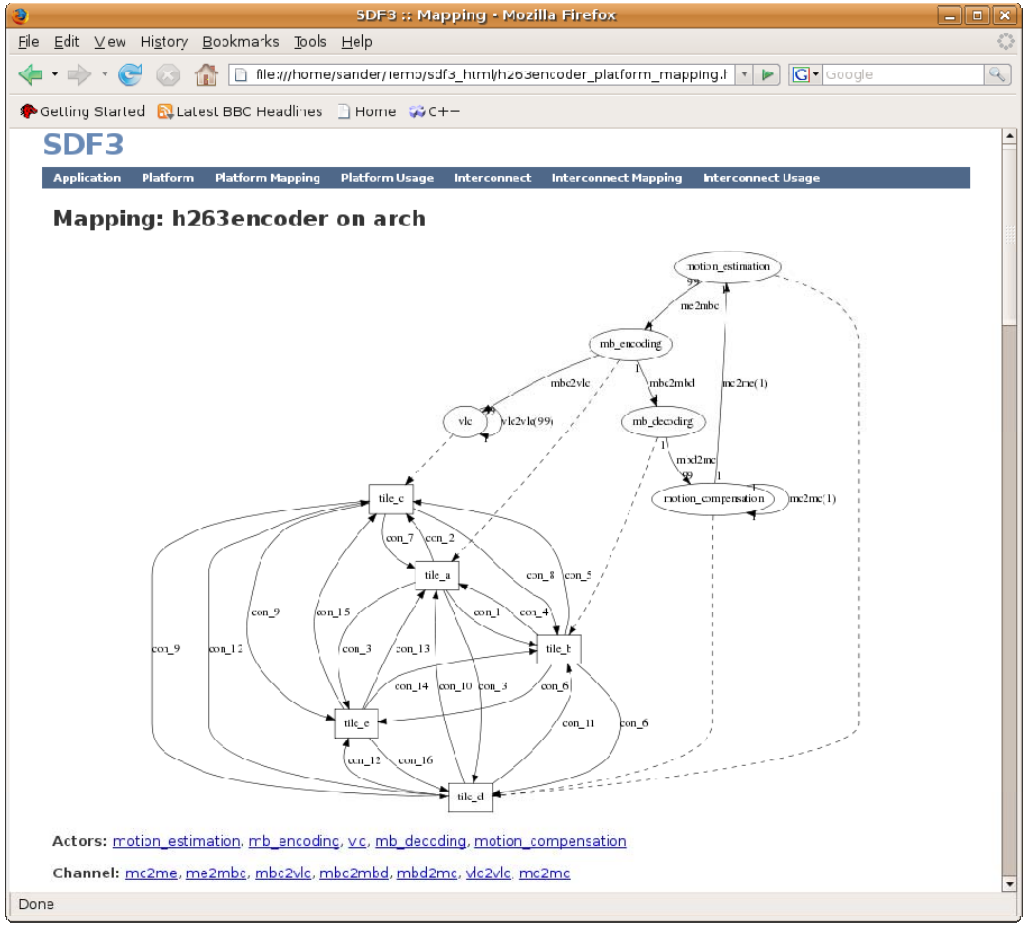
SDF³ interface

- Input/output of each step is described in XML
- XML can be transformed to HTML
- Command-line tool and C/C++ API available



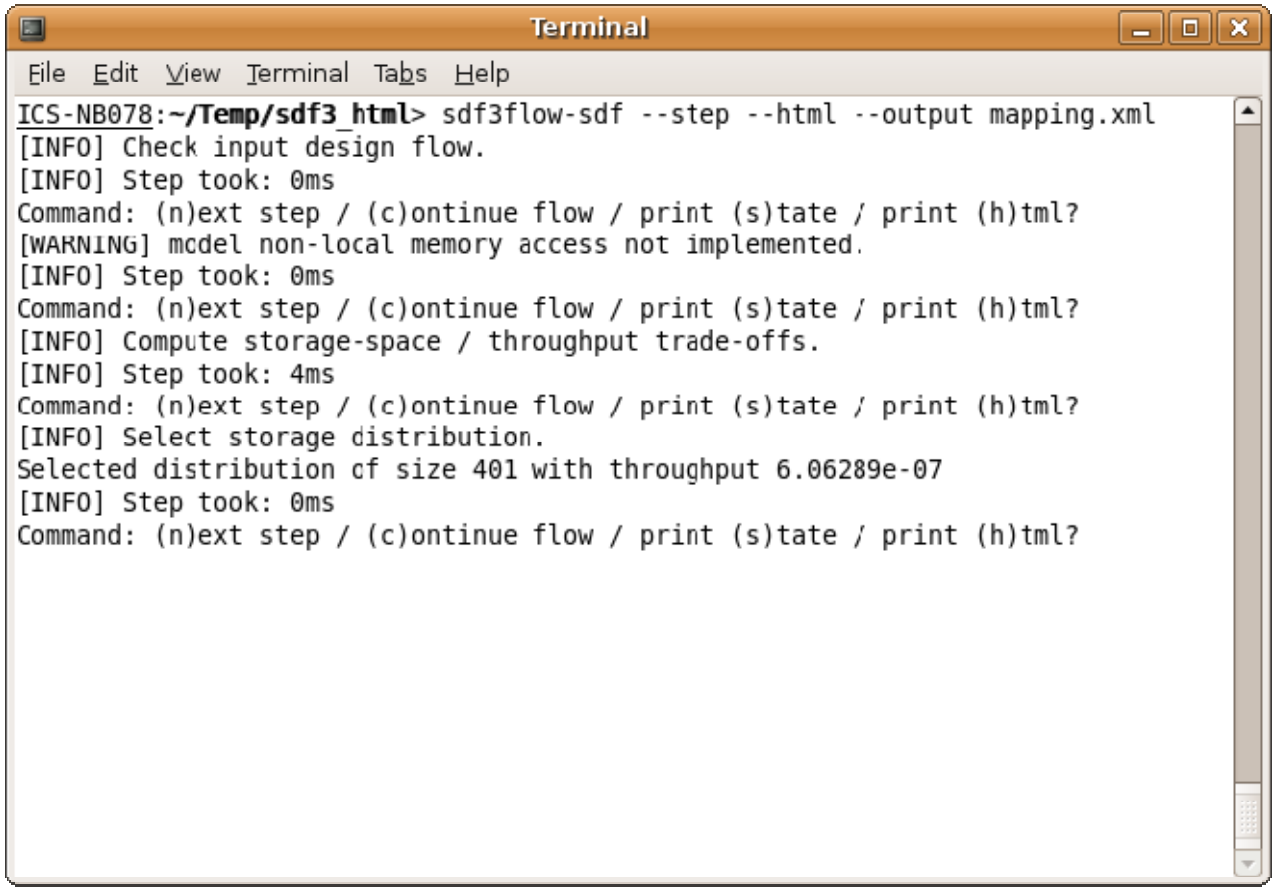
SDF³ interface

- Input/output of each step is described in XML
- XML can be transformed to HTML
- Command-line tool and C/C++ API available



SDF³ interface

- Input/output of each step is described in XML
- XML can be transformed to HTML
- Command-line tool and C/C++ API available



Concluding remarks

- MP-SoC design-flow and SDF^3 toolkit available at www.es.ele.tue.nl/sdf3
- First design-flow which maps SDFG to NoC-based MP-SoC
- Considers scheduling on processing, storage and communication resources
- Flow based on trade-offs between storage space, latency and bandwidth
- Most of the steps in the design-flow require milliseconds to complete for realistic applications