Computational Models for Concurrent Streaming Applications

Twan Basten

Based on joint work with

Marc Geilen, Sander Stuijk, and many others

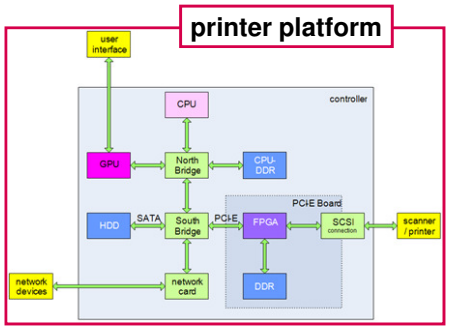
Department of Electrical Engineering
Electronic Systems

'Knowing is not understanding.'
Charles Kettering

The challenges of today

- Use-cases to be mapped onto a professional printer
  - Copying (black&white, color, various paper sizes, zoom factors, …)
  - Printing
  - Scanning
  - Simultaneous printing and scanning

- How many CPUs, GPUs ?
- Processor speeds ?
- How to achieve X images/min ?
- Double resolution ?
- Run-time changes in speed, job type, power budget … ?



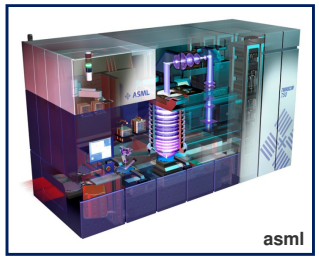
- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

5 Objectives

Embedded Systems INSTITUTE TU/e

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

High-tech professional systems

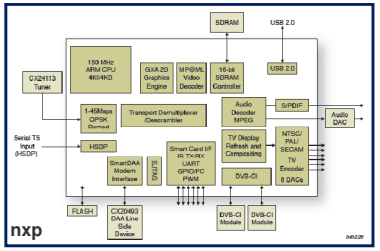


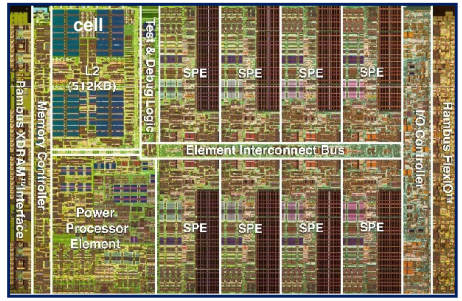
6 Objectives

Embedded Systems INSTITUTE TU/e

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

Systems-on-chip



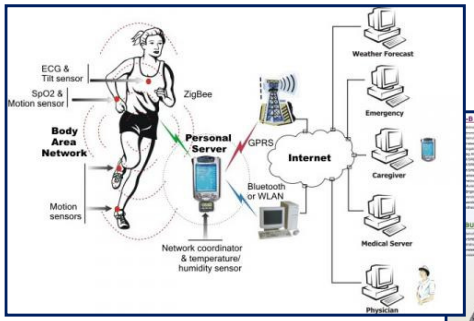


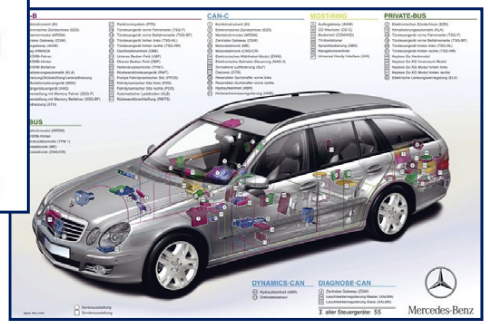
7 Objectives

Embedded Systems INSTITUTE TU/e

- Reliable, resource-efficient embedded systems
- Fast, predictable design trajectories

Networked embedded systems



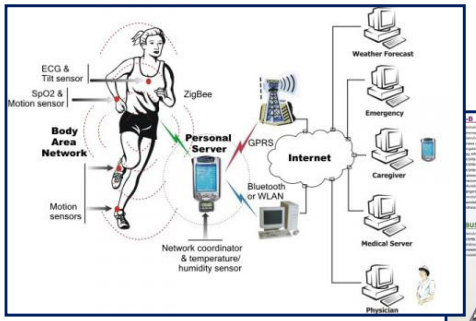


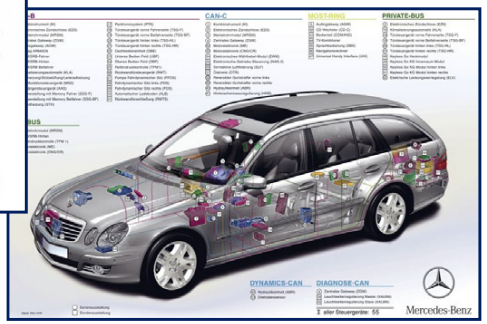
8 Trend: increasing complexity

Embedded Systems INSTITUTE TU/e

- Multiprocessor systems / networking / concurrency
- Application convergence / application evolution
- Mixed data- and event processing
- Variability / dynamism

Networked embedded systems





9 Scientific challenges

- Reliable, resource-efficient embedded systems
  - Computational modeling and analysis
  - Model-driven synthesis and run-time management
- Fast, predictable design trajectories
  - Complexity reduction
  - (Semi-)automatic system synthesis

'An understanding of the natural world and what's in it is a source of not only a great curiosity but great fulfillment.'
David Attenborough

10 Outline

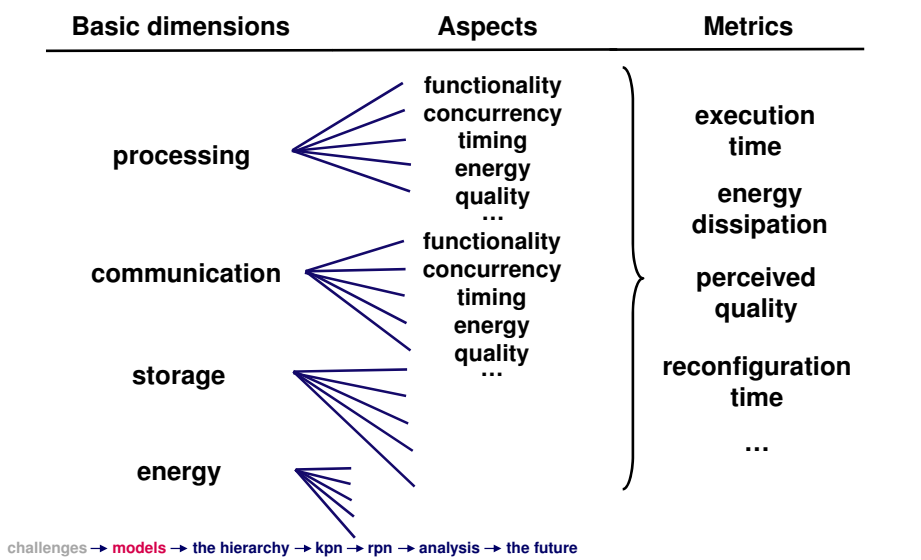
- The challenges of today
- Computational models
- The dataflow hierarchy
- Kahn Process Networks
- Reactive Process Networks
- Analysis
- Looking forward

11

Computational models

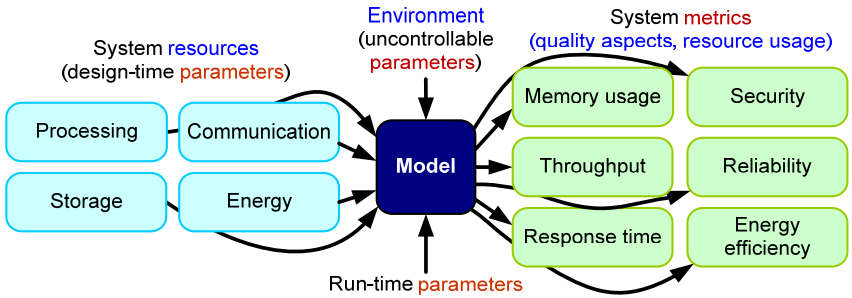
12 The essence: computational models

on the interface between science and engineering | essential for predictability



13 A model

… predicts system metrics given parameter values

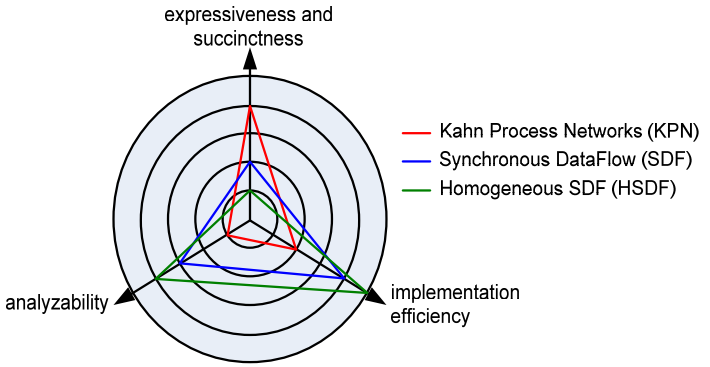


… should be targeted to the problem at hand

14 Essential challenges

predictability and efficiency

… are contradictory

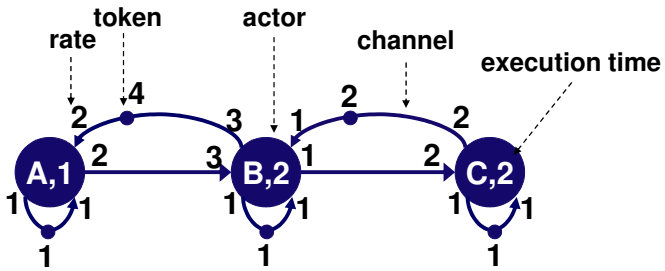


15

The dataflow hierarchy

16 Synchronous DataFlow

(SDF [Lee 1986])

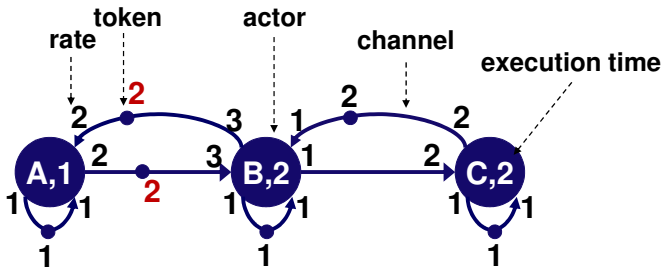


SDF: rates are fixed
(weighted marked graphs from Petri-net theory)

Homogeneous SDF: all rates 1
(marked graphs from Petri-net theory)

17 Synchronous DataFlow: actor firing

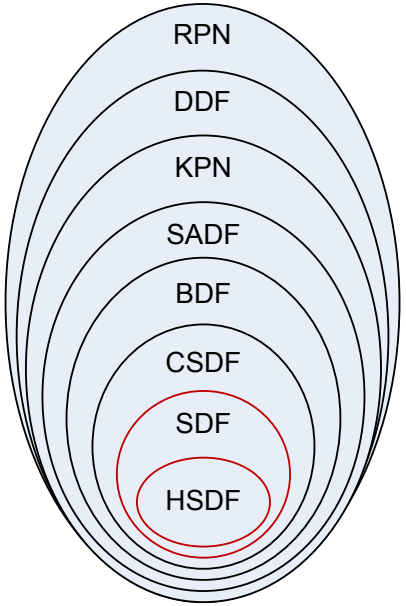
(SDF [Lee 1986])



SDF: rates are fixed
(weighted marked graphs from Petri-net theory)

Homogeneous SDF: all rates 1
(marked graphs from Petri-net theory)

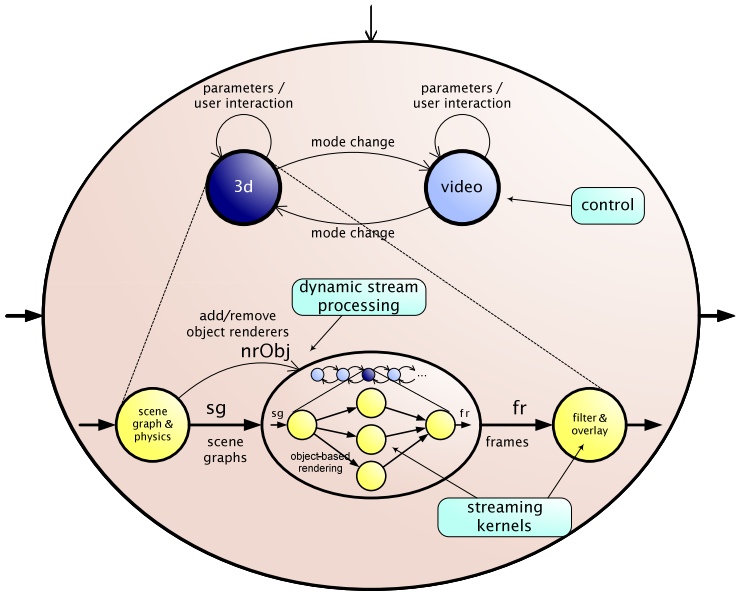
18 Dataflow expressiveness hierarchy



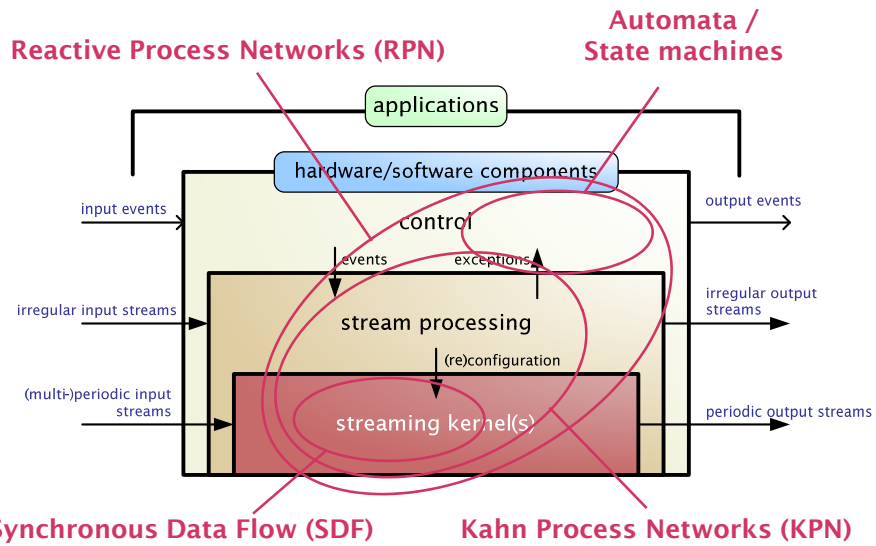
- **BDF and larger: Turing complete**
- **Better notions of expressiveness needed**
- **Unification needed**

RPN: Reactive Process Networks
KPN: Kahn Process Networks
DF: DataFlow
DDF: Dynamic DF
SADF: Scenario-Aware DF
BDF: Boolean DF
CSDF: Cyclo-Static DF
SDF: Synchronous DF
HSDF: Homogeneous SDF

19 A gaming example



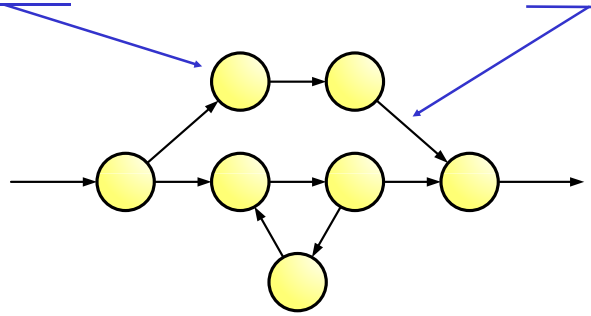
20 Application domain



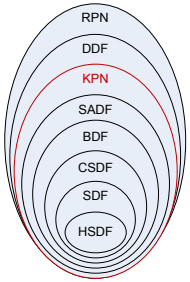
21 Kahn Process Networks

(KPN [Kahn 1974])

processes communicating via unbounded fifo queues

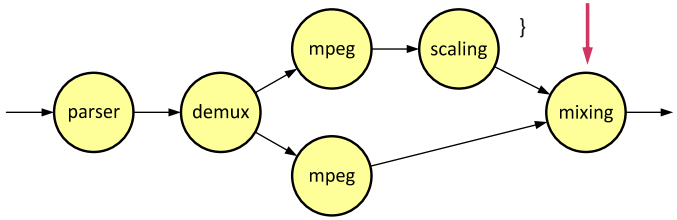


abstraction: only functionality, buffering
processes: need to be functional



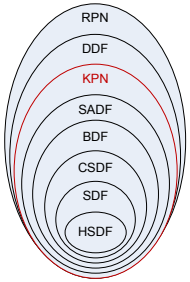
22 Example KPN

Picture in Picture




```
mixing()
while(true){
   read(in1, frame1);
   preprocess(frame1);
   read(in2, frame2);
   frame3 = combine(frame1, frame2);
   write(out, frame3);
}
```

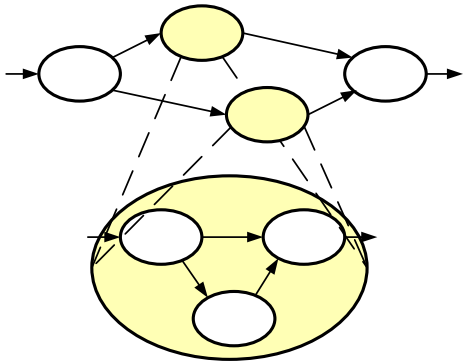
- reads block on empty queues
- writes may never block
- no global variables



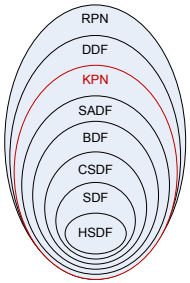
23 KPN: Denotational semantics

processes = (mathematical) functions
fifo queues = strings, sequences of data tokens

if functions are **continuous**, then a network is also a **continuous** function

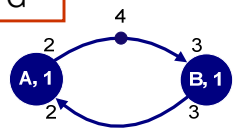


compositional semantics

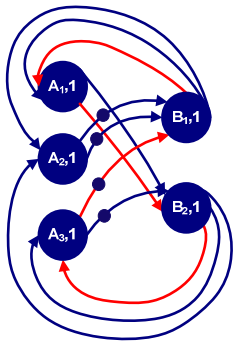


24 SDF 2 'equivalent' HSDF transformation

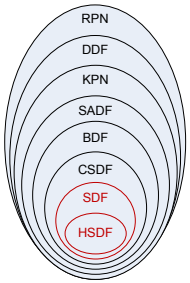
SDFG



HSDFG



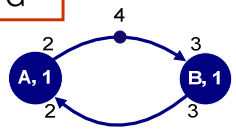
- Preserves actor firings, timing properties
- Does not preserve buffering aspects
- Algorithms operating directly on SDF may be more efficient and/or provide better results (or they may not ☺)



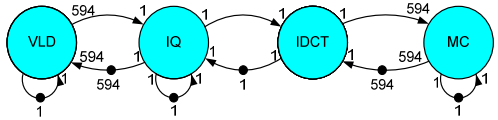
25 SDF 2 'equivalent' HSDF transformation

Embedded Systems INSTITUTE TU/e

SDFG

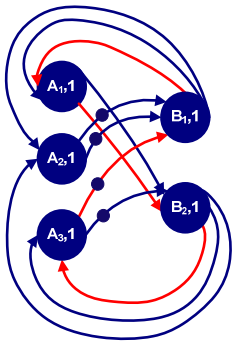


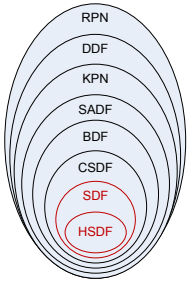
Dataflow graph H.263 decoder



HSDFG

HSDFG with 1190 actors

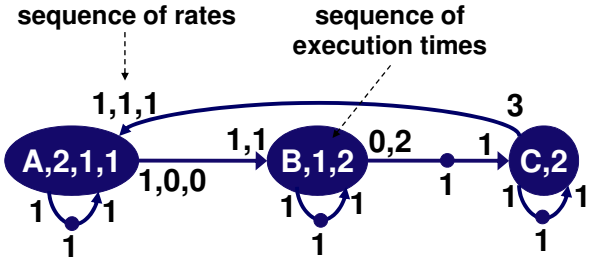




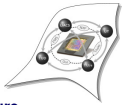
26 Cyclo-Static DataFlow

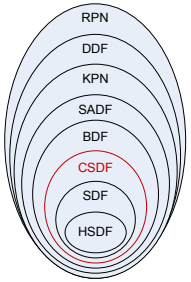
Embedded Systems INSTITUTE TU/e

(CSDF [Bilsen et al 1966])



- Allows the specification of certain iterations
- Can be transformed into an 'equivalent' SDF
- Algorithms operating directly on CSDF may be more efficient and/or provide better results
- The computational model selected in the NEST project





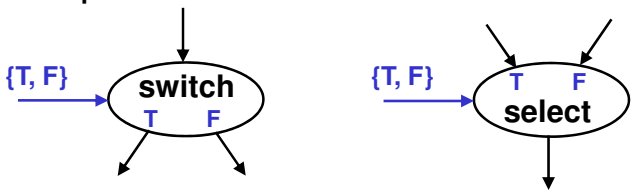
27 Boolean DataFlow

Embedded Systems INSTITUTE TU/e

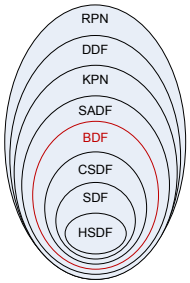
(BDF [Buck 1993])

the numbers of tokens produced or consumed in a firing can be a two-valued function of a control token

for example:



- Allows the specification of conditions and iterations
- Cannot, in general, be transformed into an 'equivalent' SDF
- Turing-complete

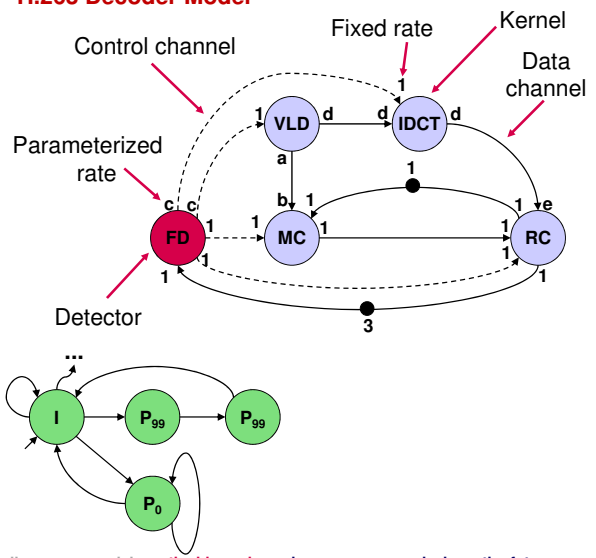


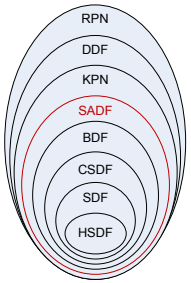
28 Scenario-Aware DataFlow

Embedded Systems INSTITUTE TU/e

(SADF [Theelen et al 2006])

H.263 Decoder Model

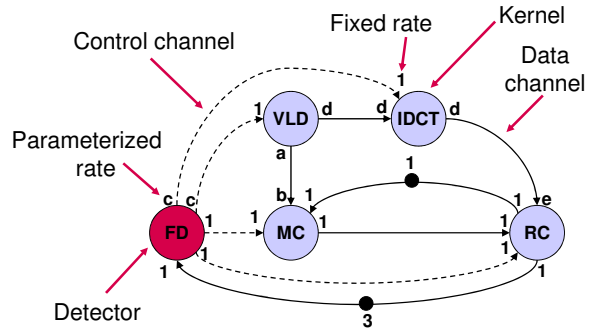




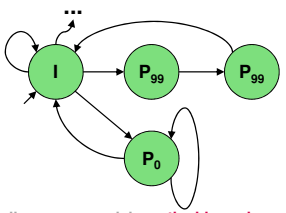
29 Scenario-Aware DataFlow

(SADF [Theelen et al 2006])

H.263 Decoder Model



Execution time		
VLD	P_0	0
	others	40
IDCT	P_0	0
	others	17
MC	I, P_0	0
	P_{30}	90
	P_{40}	145
	P_{50}	190
	P_{60}	235
	P_{70}	265
	P_{80}	310
	P_{99}	390
RC	I	350
	P_0	0
	P_{30}, P_{40}, P_{50}	250
	P_{60}	300
	P_{70}, P_{80}, P_{99}	320
FD	All	0

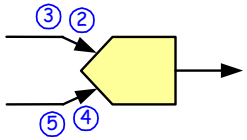


x={30,40,50,60,70,80,99}

Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

30 KPN: Fundamental limitation

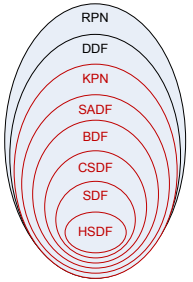
non-deterministic merge:



forwards tokens selecting input arbitrarily

outcomes: 5 4 3 2 **or**
3 2 5 4 **or**
3 5 4 2 **or …**

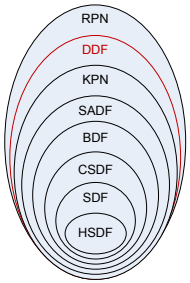
i.e., behavior is not a function !



31 Dynamic DataFlow

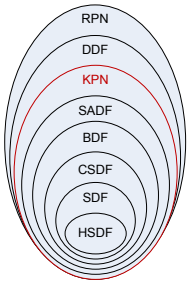
(DDF [Buck 1993])

the numbers of tokens produced or consumed in a firing may vary



32

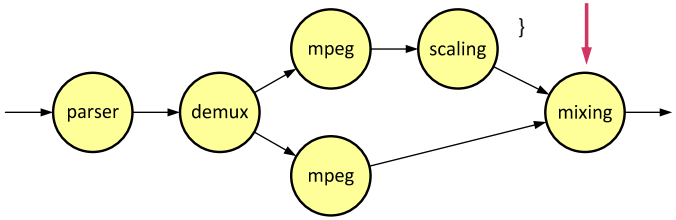
Kahn Process Networks



33 Example KPN


```
mixing()
while(true){
   read(in1, frame1);
   preprocess(frame1);
   read(in2, frame2);
   frame3 = combine(frame1, frame2);
   write(out, frame3);
}
```

Picture in Picture

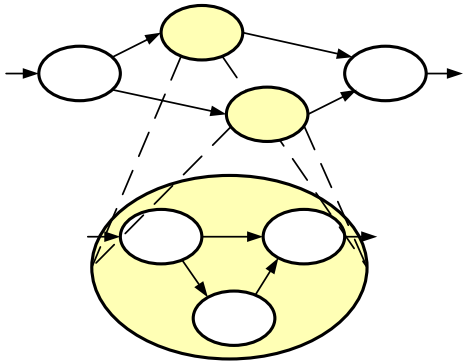


- reads block on empty queues
- writes may never block
- no global variables

34 Denotational semantics

processes = (mathematical) functions
fifo queues = strings, sequences of data tokens

if functions are **continuous**, then a network is also a **continuous** function



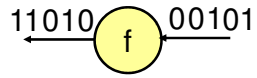
compositional semantics

35 Continuity

monotonicity

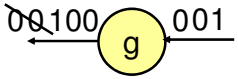
if input τ extends input σ
then output f(τ) extends output f(σ)

f(001)=110
f(00101)=11010



bit-wise invert

g(001)=100



reverse

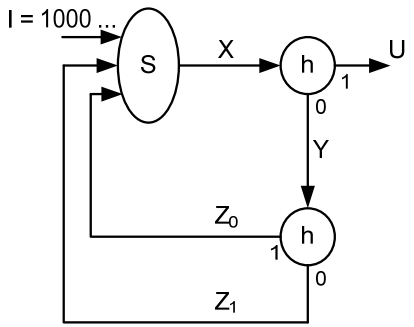
output must be taken back !! ☹

continuity

generalization of monotonicity to inputs of arbitrary, particularly infinite, length

" continuity = streaming "

36 Kleene iteration



S: elementwise sum
h_0: prefix a 0
h_1: copy

U = S(I,0U,00U)

the continuous function of a network
is the least fixpoint
of a set of fixpoint equations
and can be computed via a
Kleene iteration

$U_0 = \varepsilon$
$U_{i+1} = S(I,0U_i,00U_i)$

i =	0	1	2	3	4	5	6	7	8	9	10
U_i	ε	1	11	112	1123	… 5	… 8	… 13	… 21	… 34	… 55
		↑ U(0)	↑ U(1)	↑ U(2)	↑ U(3)	↑ U(4)	↑ U(5)	↑ U(6)	↑ U(7)	↑ U(8)	↑ U(9)

37 Strengths & weaknesses

- compositionality
- determinacy (execution order and timing)
- explicit concurrency
- explicit communication (via fifos; no shared variables)
- captures data-dependent streaming behavior
- high abstraction level
- needs run-time resource management when directly implemented
- cannot capture asynchronous reactive behavior
- undecidable, e.g., minimal buffer sizes

38 An operational view

denotational semantics: what output is computed

operational semantics: how the output is computed

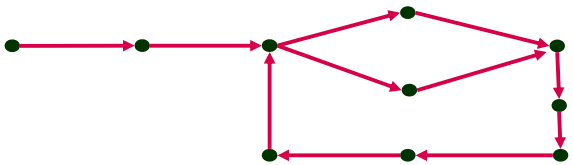
- strings are streams of tokens and fifo buffers of infinite capacity function as windows on those streams.
- an operational execution of functions conforms to (destructively) reading and writing tokens with internal computations in between.

39 Operational semantics

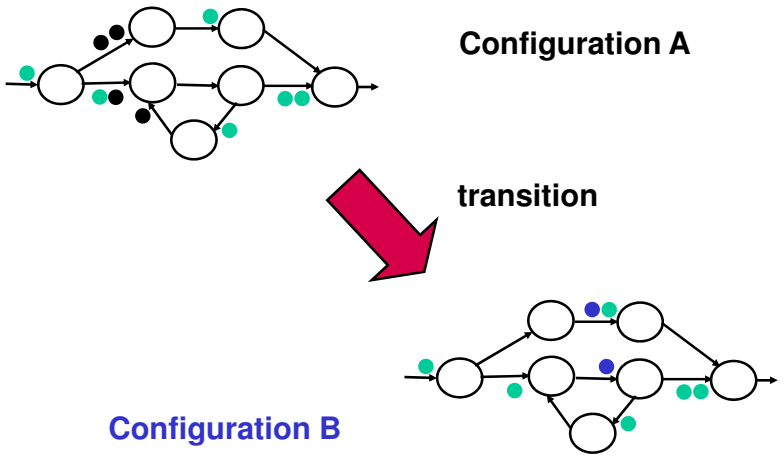
operational semantics captures all allowed ways to make behavior operational

based on
configurations
i.e., KPN + fifo fillings

and
transitions
from one configuration to another one



40 Operational semantics



41 The Kahn Principle

Embedded Systems INSTITUTE TU/e

" But hey, now we have two semantics of the model! Are they the same? "

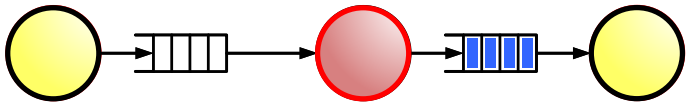
The Kahn Principle:
Any fair execution in the operational model realizes the denotational semantics, the mathematical function

This provides the boundaries for realizations of KPNs

42 Implementations of KPNs

Embedded Systems INSTITUTE TU/e

- **bounded FIFOs** combine aspects of data- and demand driven execution
- but **how large** should the FIFO buffers be?
- **undecidable**, FIFO sizes need to be **managed at run-time**



challenges → models → the hierarchy → kpn → rpn → analysis → the future

43 A run-time environment (RTE)

Embedded Systems INSTITUTE TU/e

requirements

boundedness: fifo bounds may not grow indefinitely

completeness: progress must be made on all outputs towards the output prescribed by the semantics

there exists an RTE satisfying these requirements

prerequisites (undecidable in general!)

boundedness: there exists a fair execution within finite fifo bounds

effectiveness: every token that is produced is eventually also consumed

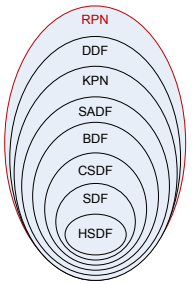
[M.C.W. Geilen and T. Basten. Requirements on the Execution of Kahn Process Networks. Proc ESOP 2003.]

challenges → models → the hierarchy → kpn → rpn → analysis → the future

44

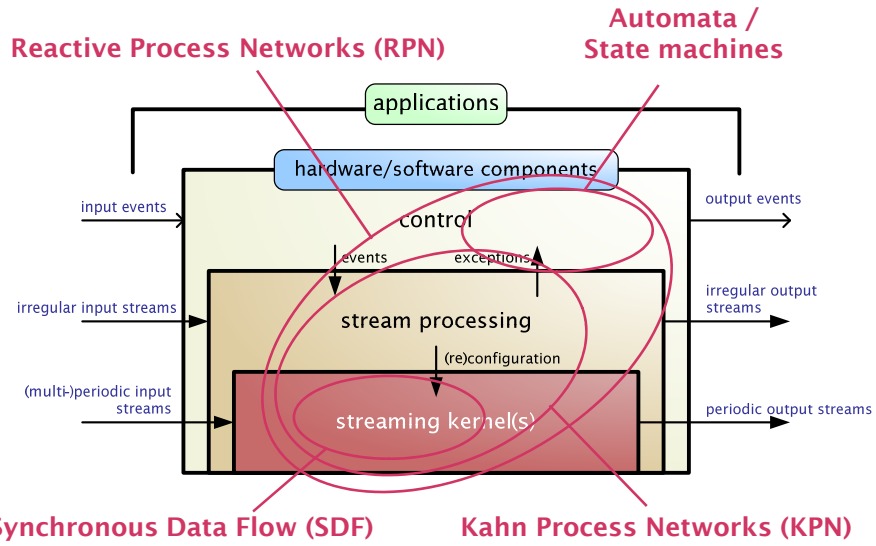
Embedded Systems INSTITUTE TU/e

Reactive Process Networks

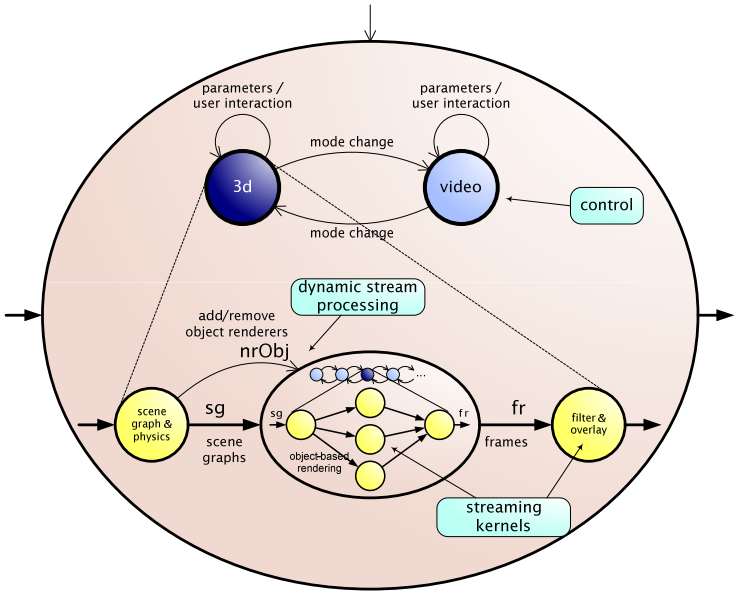


[M.C.W. Geilen and T. Basten. Reactive Process Networks. Proc EMSOFT 2004.]

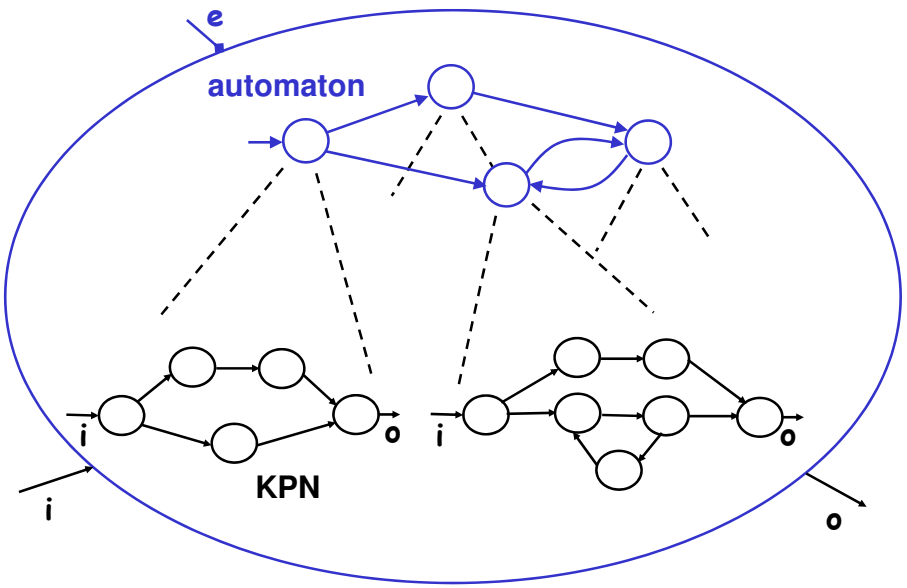
45 Application domain



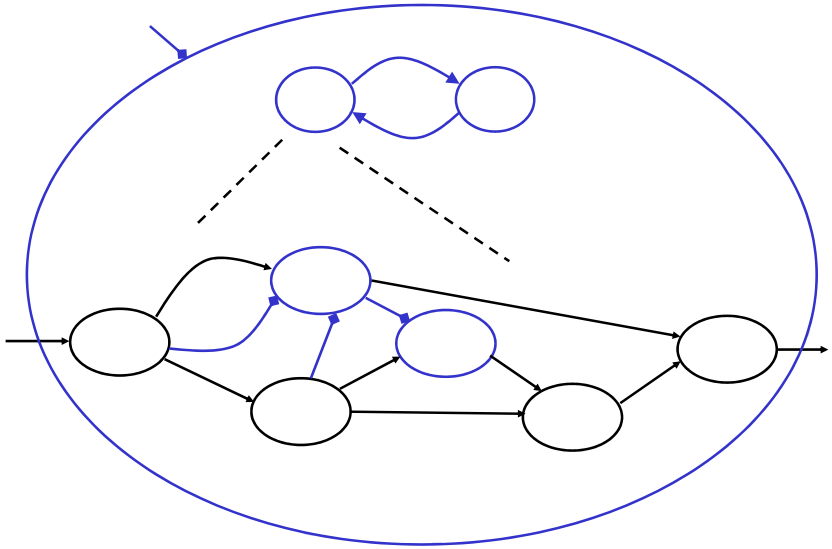
46 A gaming example



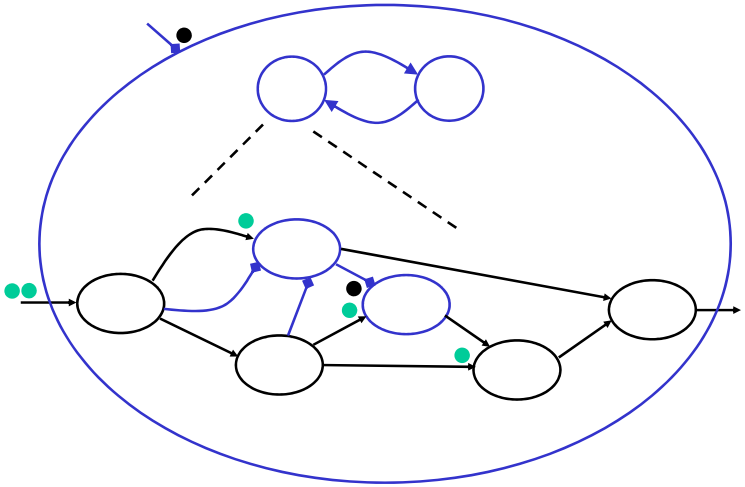
47 Reactive Process Networks



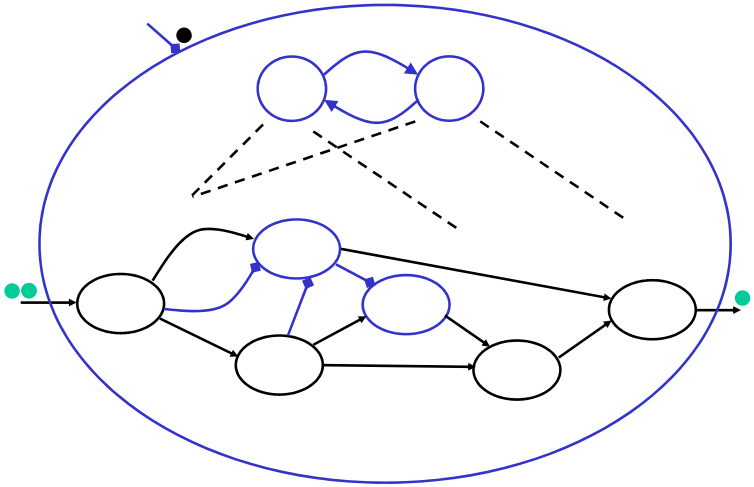
48 Reactive Process Networks



49 Operational semantics



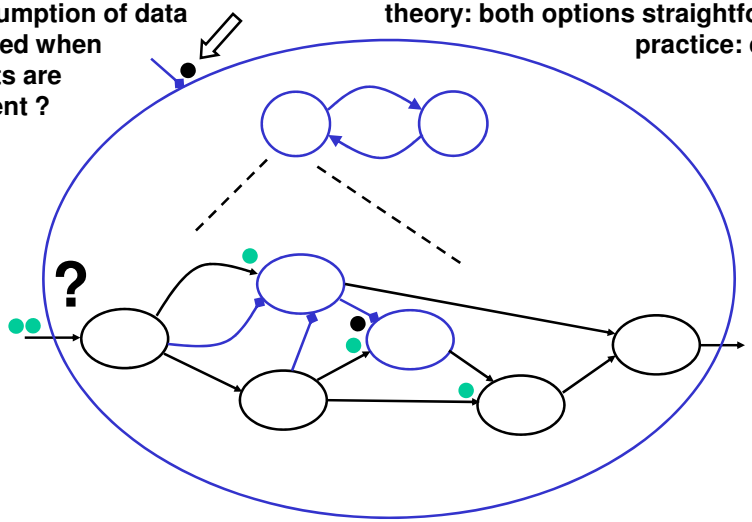
50 Operational semantics



51 Prioritizing events

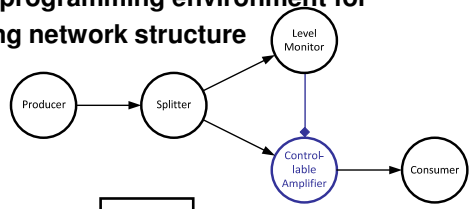
consumption of data allowed when events are present ?

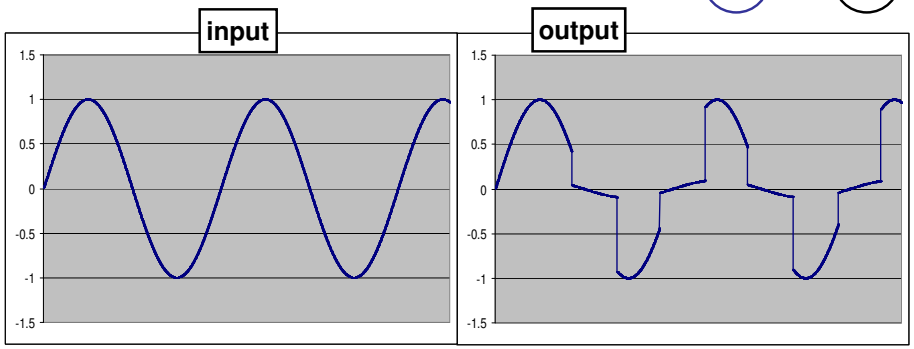
theory: both options straightforward
practice: choice



52 Prototype implementation

- prototype implementation of a programming environment for reconfigurations not affecting network structure
- events have priority over data

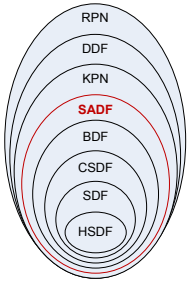




53 Open issues

- prototype implementation of arbitrary reconfigurations
- cases studies (programming styles, expressivity)
- timing
  - reconfiguration/reaction times for events
  - latency/throughput bounds on RPNs as a whole
- efficient implementation of reconfiguration
- fifo sizes, deadlock analysis
- distribution
- **analyzable subclasses** (BDF, **SDF**)

Scenario-Aware DataFlow



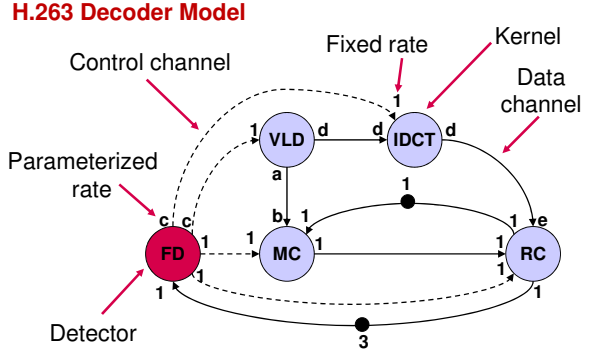
54

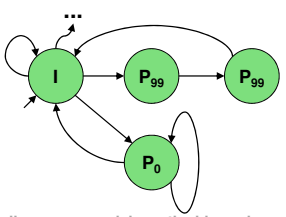
Analysis

55 Scenario-Aware DataFlow

(SADF [Theelen et al 2006])

H.263 Decoder Model



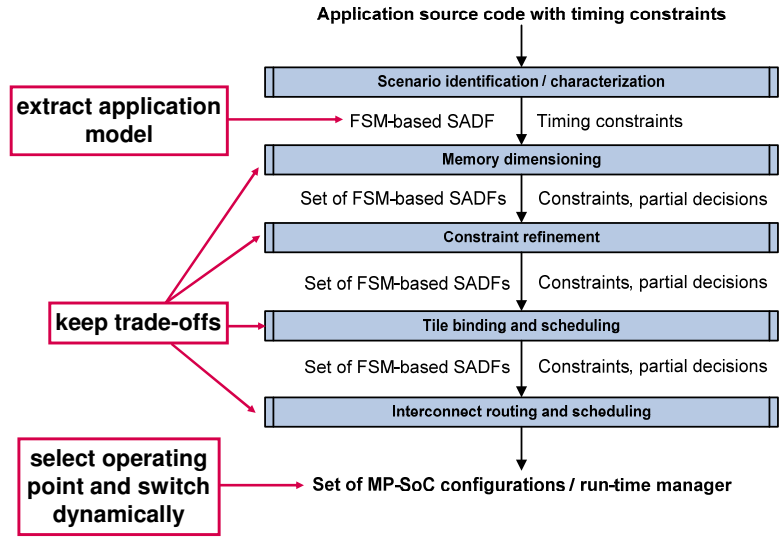


x={30,40,50,60,70,80,99}

Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

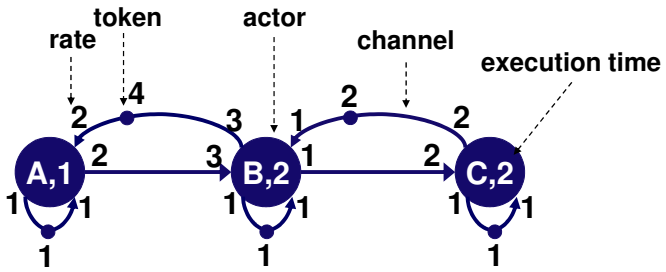
Execution time		
VLD	P_0	0
	others	40
IDCT	P_0	0
	others	17
MC	I, P_0	0
	P_{30}	90
	P_{40}	145
	P_{50}	190
	P_{60}	235
	P_{70}	265
	P_{80}	310
	P_{99}	390
RC	I	350
	P_0	0
	P_{30}, P_{40}, P_{50}	250
	P_{60}	300
	P_{70}, P_{80}, P_{99}	320
FD	All	0

56 A model-driven scenario-aware design flow



57 Synchronous DataFlow

(SDF [Lee 1986])



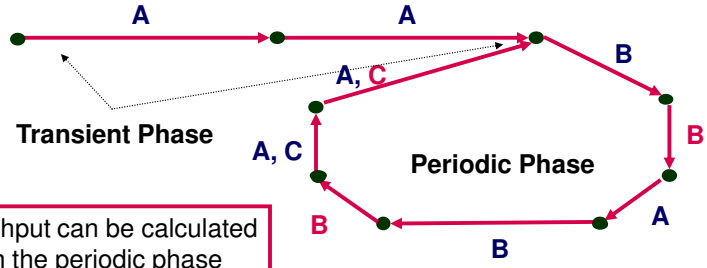
Throughput: average number of actor firings over time

Iteration: smallest non-empty set of actor firings that does not change the token distribution **(example: A:3, B:2, C:1)**

58 Throughput via state-space analysis

[Ghamarian et al 2006]

- Self-timed execution:
  - Each actor fires as soon as it can fire
  - Known to maximize throughput
- Execution state: distribution of tokens + remaining execution times
- Execution: transient + periodic phase



Throughput can be calculated from the periodic phase

Efficient implementation
Consider one designated firing per iteration only to detect a recurrent state

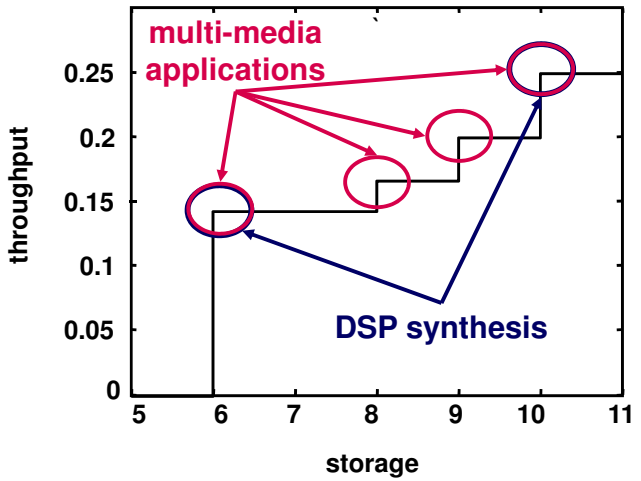
59 Throughput analysis results

Traditional methods, all using conversion to HSDF

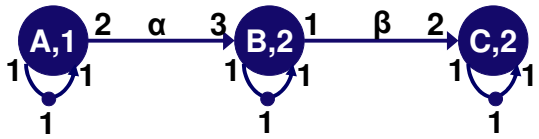
	State Space	Dasdan Gupta	Howard	Young Tarjan Orlin
MP3 dec.	$1.\ 10^{-3}$	$1.\ 10^{-3}$	$1.\ 10^{-3}$	$1.\ 10^{-3}$
Modem	1.10^{-3}	$82.\ 10^{-3}$	$81.\ 10^{-3}$	$81.\ 10^{-3}$
Sample Rate	$2.\ 10^{-3}$	>1800	>1800	>1800
Satellite	$56.\ 10^{-3}$	>1800	>1800	>1800
H.263 decoder	$10.\ 10^{-3}$	>1800	>1800	>1800

Runtimes of various throughput analysis methods (in seconds)

60 Trade-off: storage vs. throughput



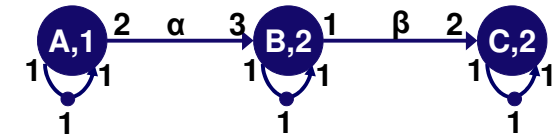
61 Storage distribution

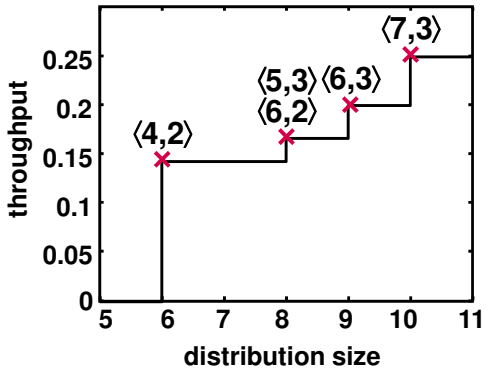


Tokens on channels must be stored in memory.

- Separate memory for each channel.
- Storage distribution, for example, ⟨α, β⟩ → ⟨4,2⟩

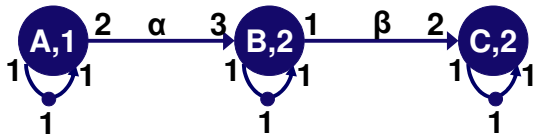
62 Problem definition



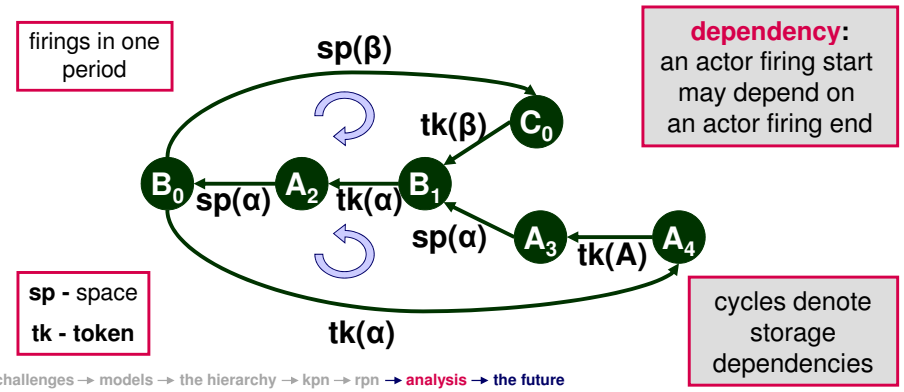


Find all minimal storage distributions for any possible throughput

63 Dependency graph

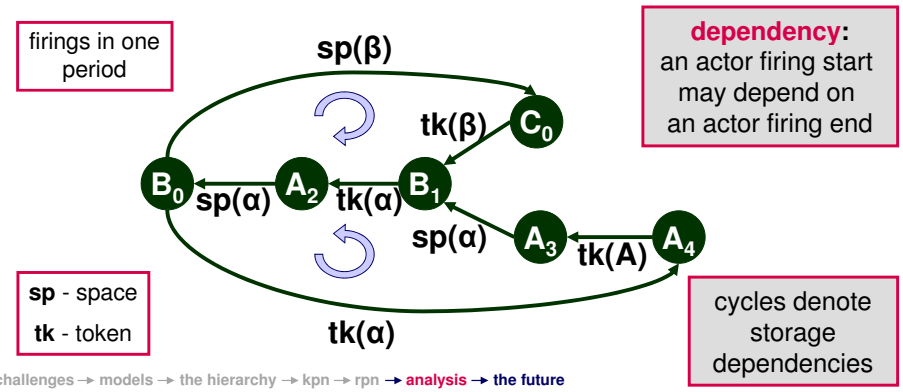


Storage distribution ⟨α, β⟩ → ⟨4,2⟩

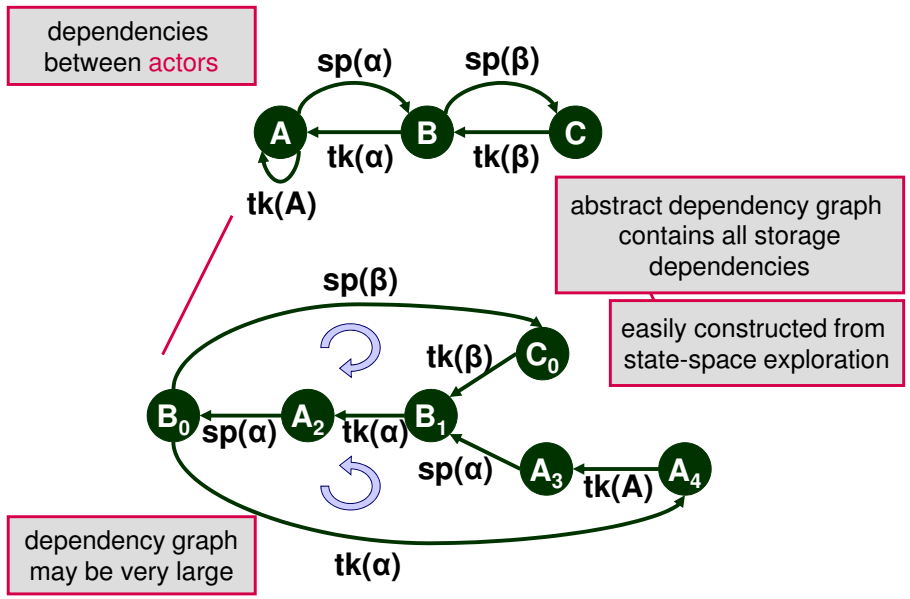


64 Dependency graph

resolving storage dependencies may increase throughput



65 Abstract dependency graph



66 Design-space exploration algorithm

Given an SDF graph G with storage distribution δ

1. Compute throughput and abstract dependency graph Δ for G with δ
2. For each channel c with a storage dependency in Δ
   Create new storage distribution by enlarging c
3. Repeat steps 1-2

All minimal storage distributions are found when starting from storage distribution ⟨0,...,0⟩

67 Experimental results

	Example	Satellite	MP3	H.263
#actors	3	22	13	4
#channels	2	26	12	3
min throughput > 0	1/7	0.18	1/137	1/21876
Distr. size	6	1542	12	4753
Max throughput				1/10000
Distr. size				8006
#Pareto points				3254
#Distributions	5	2	3	3254
Exec. time	1ms	7ms	2ms	53min

Approximation

increase buffers with n times the minimal step size

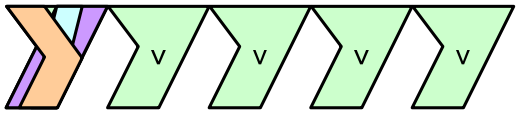
68 Extensions

- Technique applies also to CSDF [Stuijk et al, IEEE T Comp 2008]
- Technique can be generalized to other types of resources [Yang et al, DATE 2010]

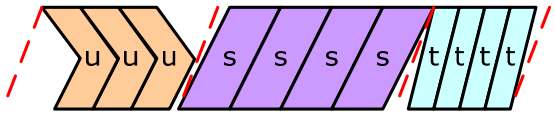
69 Scenario-aware performance analysis

[Geilen, ACM TECS 2010]

- SDF-based throughput analysis considers worst-case actor times



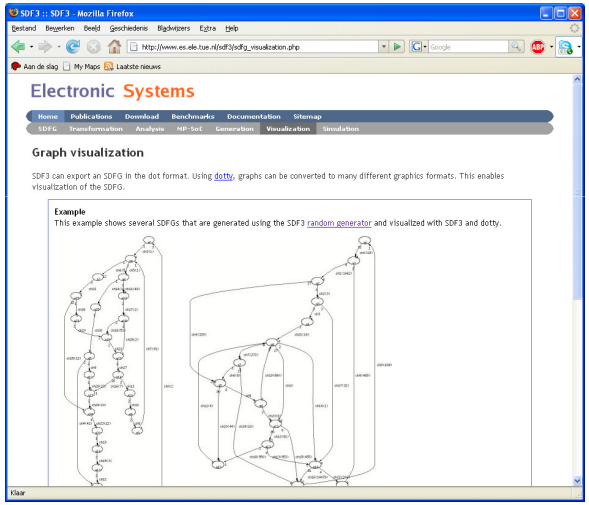
- Scenario-aware throughput analysis considers worst-case actor times per scenario, but it must consider scenario transitions



- Analyzing all scenario transitions separately can be avoided
- Separate scenario transitions with an invariant reference schedule

70 SDF3 toolkit

www.es.ele.tue.nl/sdf3



71

Looking Forward

72 Challenges

- Suitable notions of expressivity
- Expressivity while maintaining analyzability and synthesizability
- Abstraction without loosing accuracy

- Composability and compositionality
- MoCs for non-functional aspects
- Unification of MoCs

- Multi-objective trade-off analysis
- Parametric analysis

- Model-driven design and synthesis flows
- Model-driven run-time systems

Thank you !

Embedded Systems INSTITUTE TU/e

Questions ?

More info: www.es.ele.tue.nl/~tbasten/
SDF3 toolkit: www.es.ele.tue.nl/sdf3/

‘An understanding of the natural world and what's in it is a source of not only a great curiosity but great fulfillment.’
David Attenborough