Reliable Embedded Multimedia Systems?

Twan Basten

Joint work with
Marc Geilen, AmirHossein Ghamarian,
Hamid Shojaei, Sander Stuijk, Bart Theelen, and others

Funding:
NWO PROMES
EC FP6 Betsy, FP7 MNEMEE

'Knowing is not understanding.'
Charles Kettering

2 Overview

- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

3 Embedded Multi-media

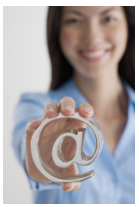
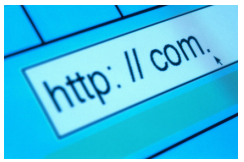



Trends
Concurrency
Connectedness
Interaction
Variability

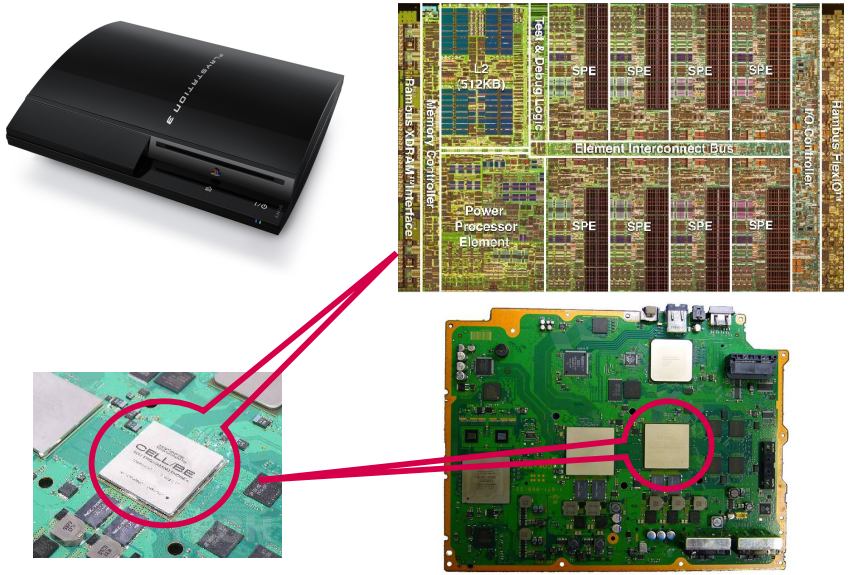
4 Embedded Multi-media





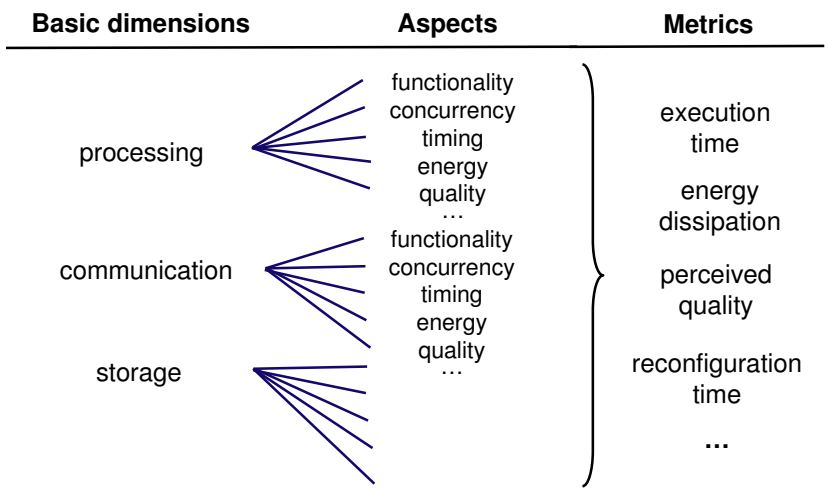
5 Embedded Multi-media



6 Embedded Multi-media

7 Approach: Models of Computation (MoCs)

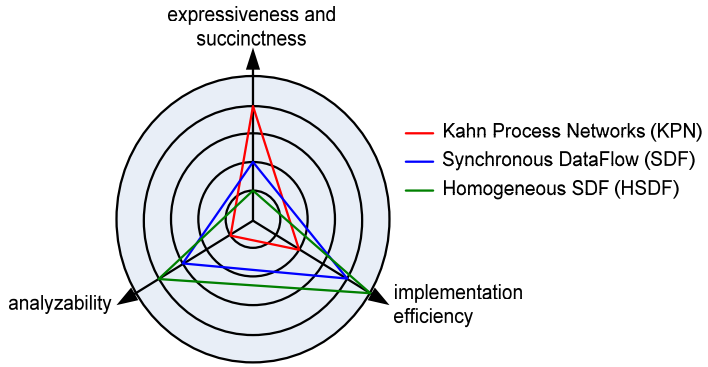
on the interface between science and engineering | essential for predictability



8 Essential Challenges

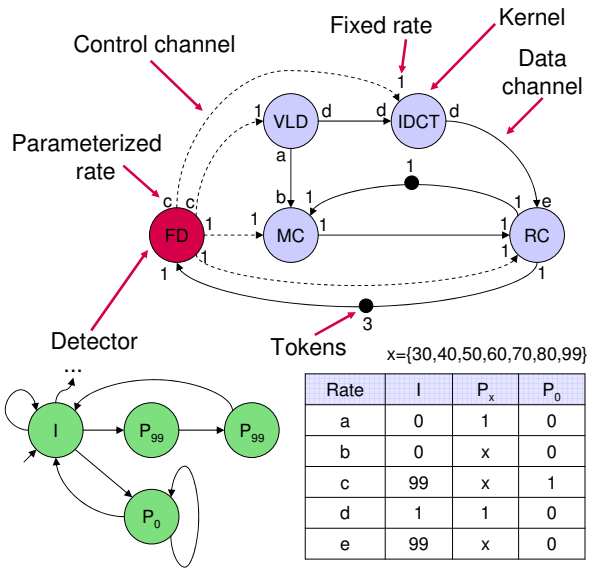
predictability and efficiency

… are contradictory



9 Scenario-aware H.263 Decoder Model

(FSM-based Scenario-Aware DataFlow (SADF))

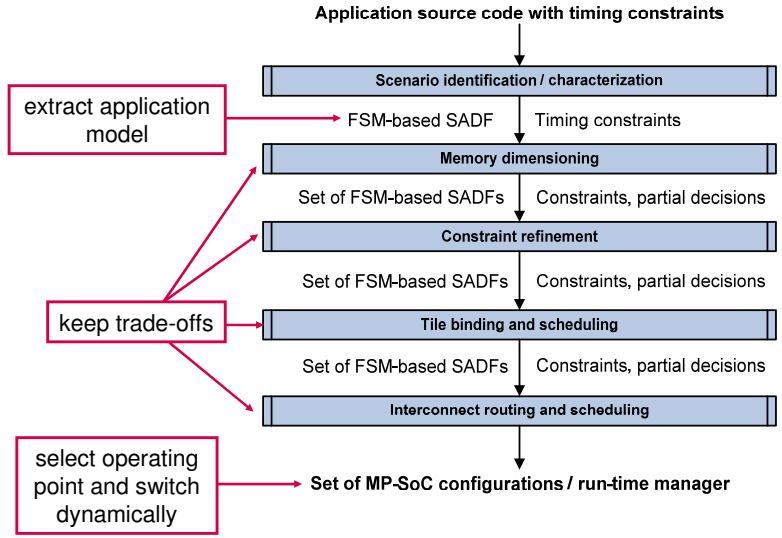


Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

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

MEMOCODE 2006

10 The Goal: Scenario-aware Flow



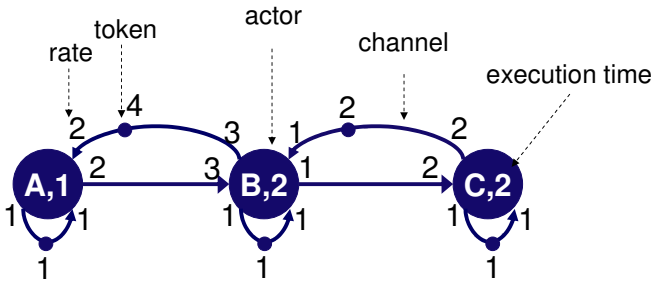
11 Overview

- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

ACSD 2006

12 Synchronous Data Flow Graphs

(SDFGs [Lee 1986])

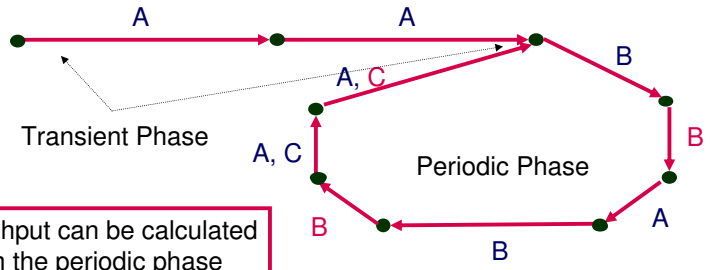


Throughput: average number of actor firings over time

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

13 Our Throughput Analysis Method

- 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

14 Throughput Analysis: Our Result

Traditional methods, all using conversion to Homogeneous SDFGs

	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

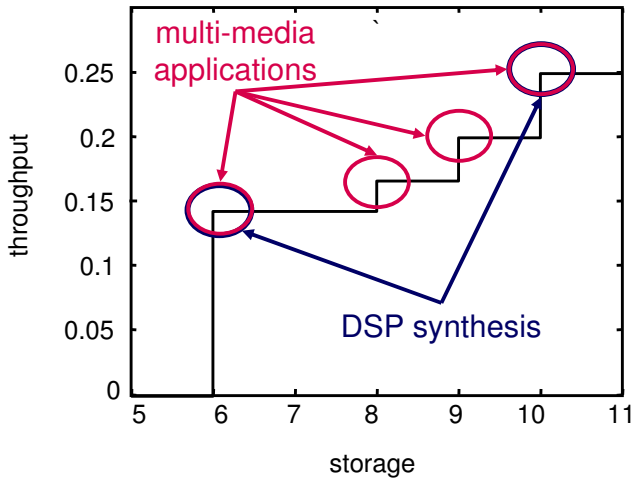
Runtimes of various throughput analysis methods (in seconds)

15 Overview

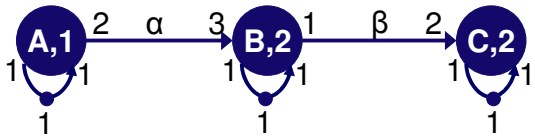
- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

DAC 2006, IEEE T on Comp 2008

16 Trade-off: Storage vs. Throughput



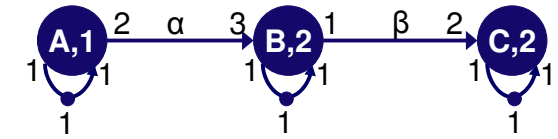
17 Storage Distribution

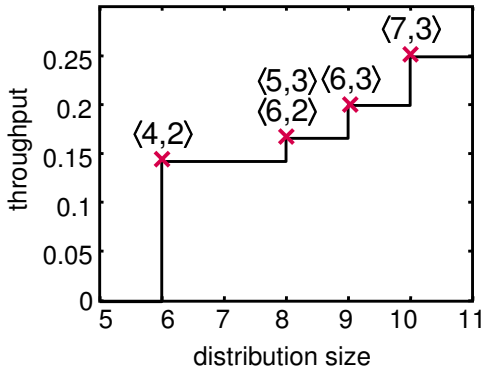


Tokens on channels must be stored in memory.

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

18 Problem Definition

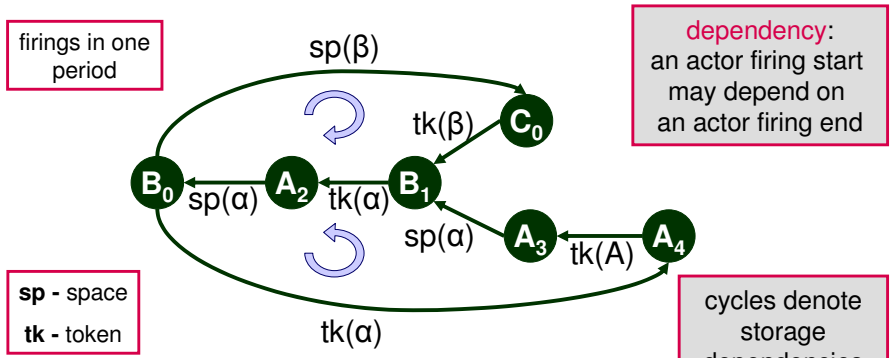




Find all minimal storage distributions for any possible throughput

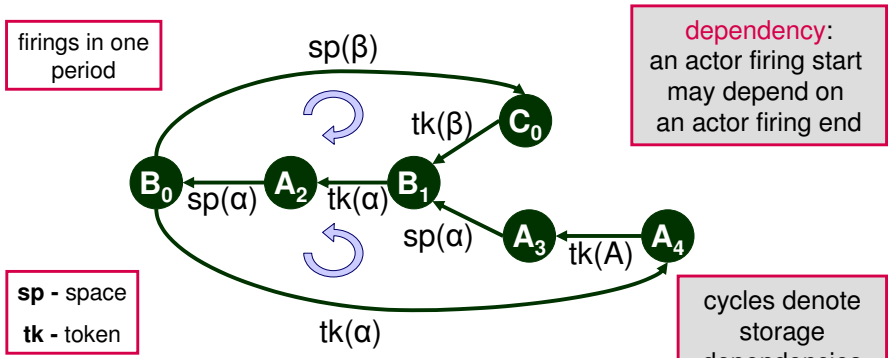
19 Dependency Graph

Storage distribution ⟨α, β⟩ → ⟨4,2⟩

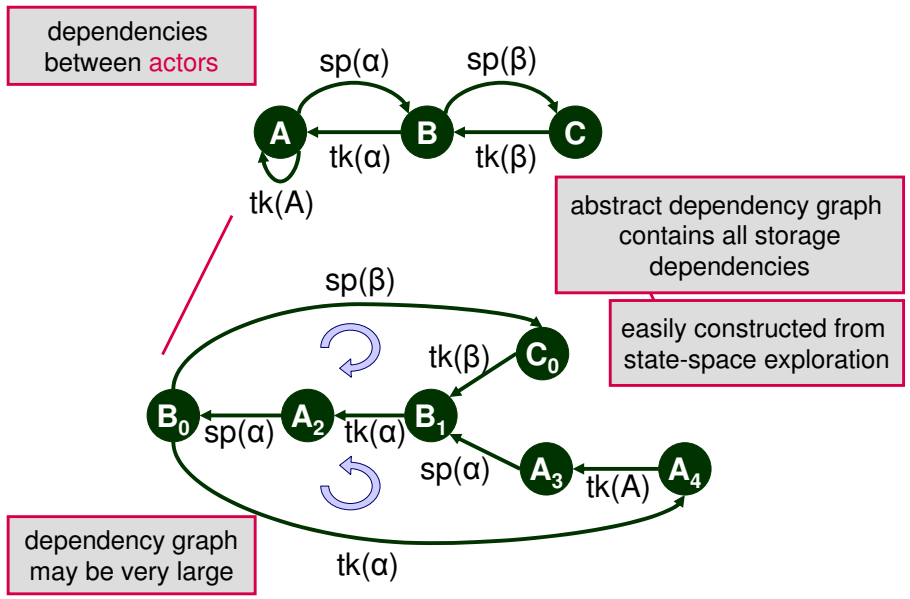


20 Dependency Graph

resolving storage dependencies may increase throughput



21 Abstract Dependency Graph



22 Design-Space Exploration Algorithm

Given an SDFG 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⟩

23 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

24 Conclusions Trade-off Analysis

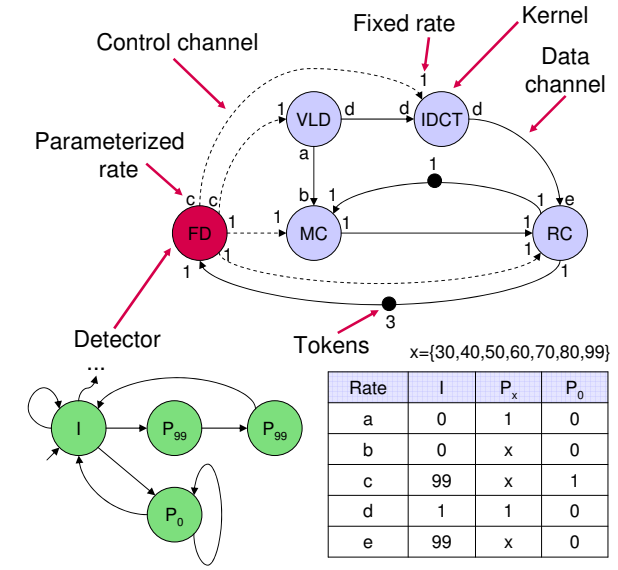
- Exact minimum memory requirements for any possible throughput
- Technique can be combined with heuristics to prune the search space if necessary
- Technique can be generalized to other types of resources and performance metrics

25 Overview

- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

26 Scenario-aware H.263 Decoder Model

(FSM-based Scenario-Aware DataFlow (SADF))



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

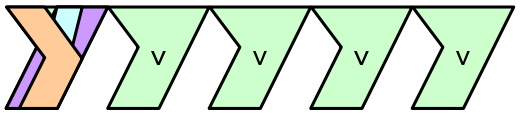
Rate	l	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

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

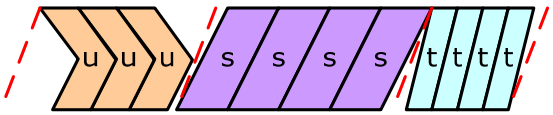
MEMOCODE 2006

27 Scenario-aware Performance Analysis

- 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

ACM TECS

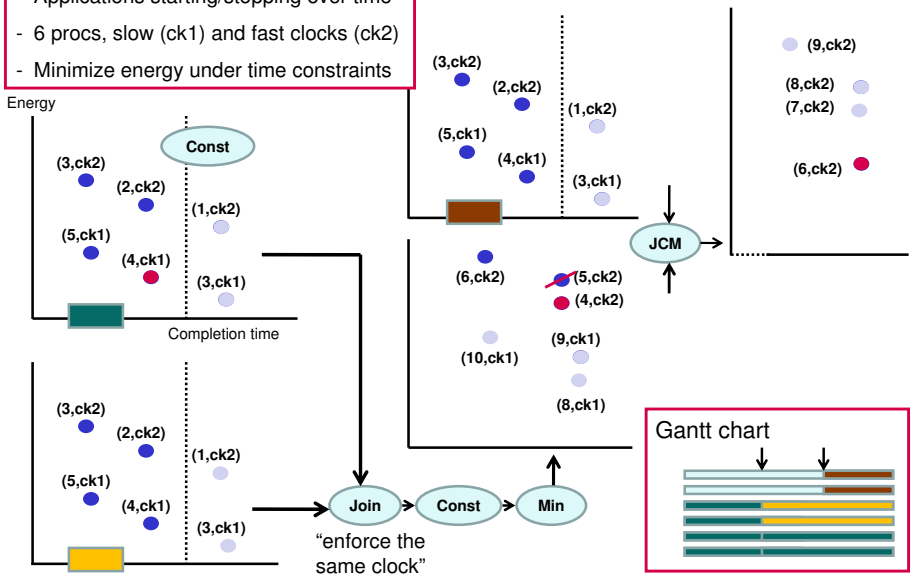
28 Overview

- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

DAC 2009

29 Run-time Application Management

- Applications starting/stopping over time
- 6 procs, slow (ck1) and fast clocks (ck2)
- Minimize energy under time constraints

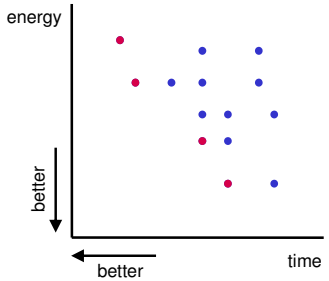


30 Pareto Algebra

Goal: Compositional computation of trade-offs

31 Pareto Algebra

- The elements: sets of configurations
- (Relevant) operators:
  - **Minimization**
    Gives the Pareto points (optimal trade-offs)
  - **Product**
    Cartesian product of configurations
    e.g. application and platform configurations
  - **Constraint**
    Selects solutions according to constraints
    e.g. all application configurations with some minimal quality
  - **Abstraction**
    Discards information about solutions
    e.g. bandwidth usage in bandwidth × energy × quality configurations
  - **Derived metric**
    Derive a new metric from other metrics
    e.g. total power from power of components



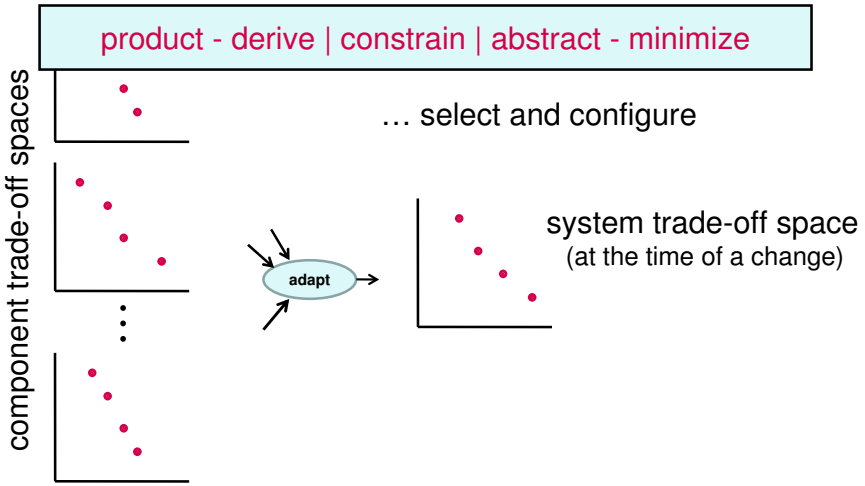
prerequisite
monotonicity

32 Pareto-Algebraic Characterization

… of run-time adaptation

product - derive | constrain | abstract - minimize

… select and configure



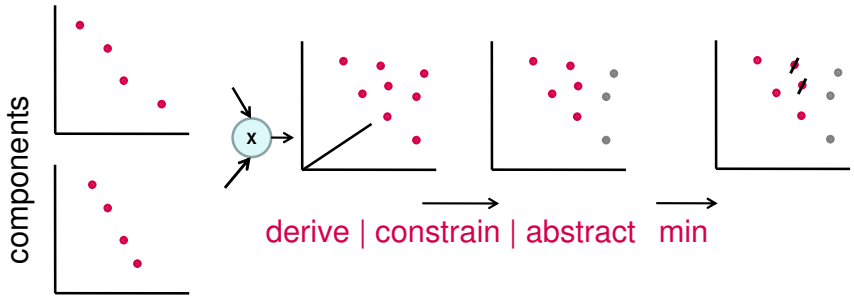
33 Pareto-Algebraic Characterization

Embedded Systems INSTITUTE TU/e

… of run-time adaptation

product - derive | constrain | abstract - minimize

… select and configure



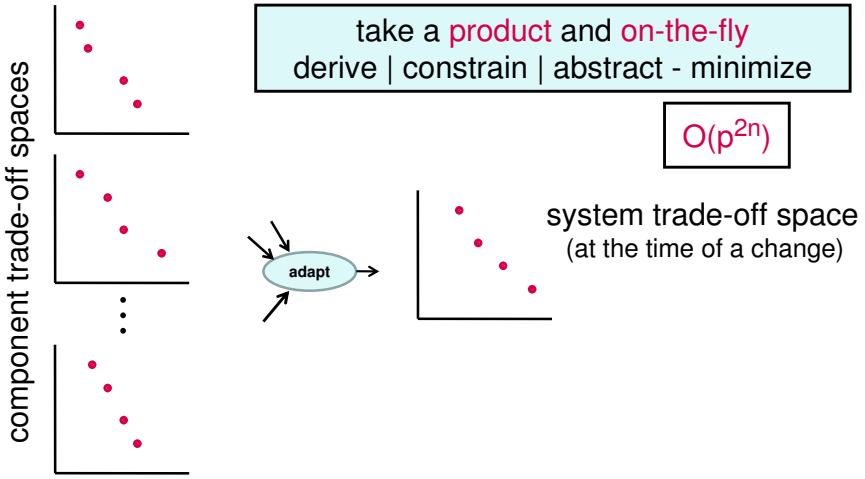
34 Complexity

Embedded Systems INSTITUTE TU/e

n components with p Pareto points each

take a product and on-the-fly
derive | constrain | abstract - minimize

$O(p^{2n})$



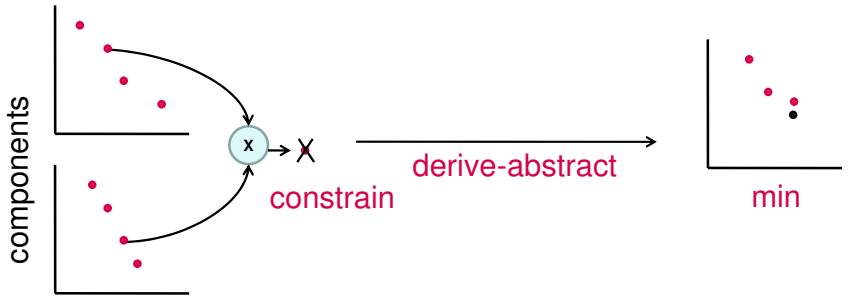
35 Complexity

Embedded Systems INSTITUTE TU/e

n components with p Pareto points each

take a product and on-the-fly
derive | constrain | abstract - minimize

$O(p^{2n})$

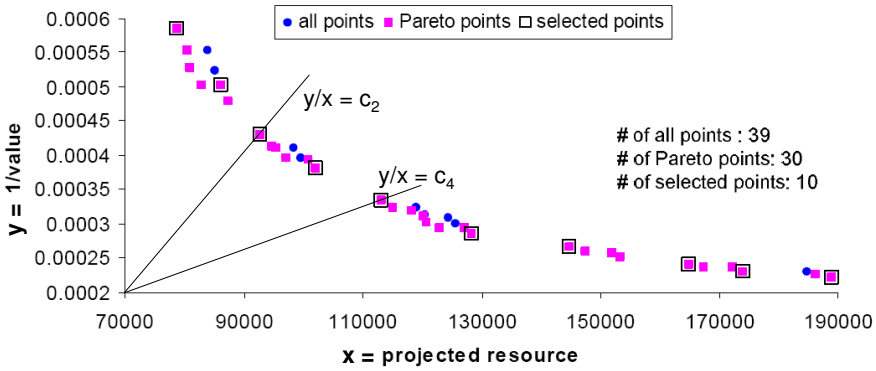


36 Complexity Control

Embedded Systems INSTITUTE TU/e

keep n and p small

- Compositional reasoning, among others
- Approximate Pareto sets



37 Multi-dimensional Multiple-choice Knapsack Problem

MMKP

NP hard

- One optimization objective (value)
- Multiple resource dimensions with capacity constraints
- Multiple independent applications
- Multiple independent configurations per application

Pick one configuration per application optimizing value within constraints

38 A Parameterized Compositional Heuristic

Embedded Systems INSTITUTE TU/e

- Project all resource dimensions into one dimension
- Approximate Pareto set in 2-dimensional space

product
constrain
derive
abstract
minimize

Compute product while on-the-fly

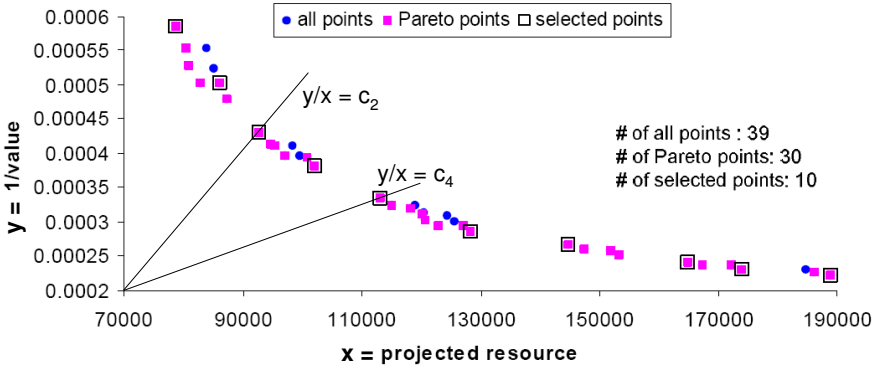
- Applying resource constraints
- Computing values, projecting all resource dimensions into one (taking simply the sum)
- Minimizing the configuration set in 2-dimensional space

product-constrain-derive-abstract-minimize

39 A Parameterized Compositional Heuristic

Embedded Systems INSTITUTE TU/e

- Project all resource dimensions into one dimension
- Approximate Pareto set in 2-dimensional space



product-constrain-derive-abstract-minimize

40 A Parameterized Compositional Heuristic

Embedded Systems INSTITUTE TU/e

- Project all resource dimensions into one dimension
- Approximate Pareto set in 2-dimensional space

parameter **p**: maintain (at most) **p** Pareto points

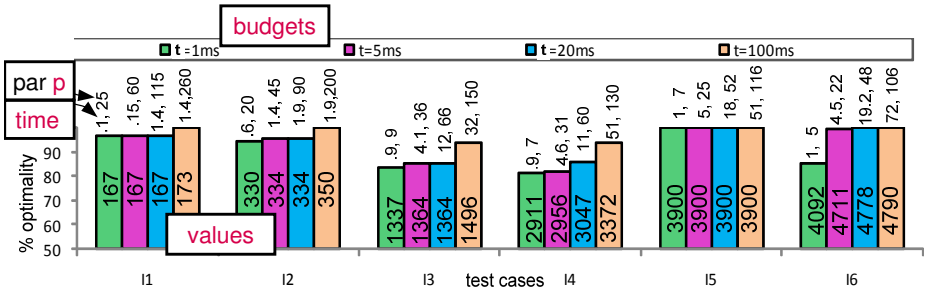
Allows to budget analysis time

analysis time ≤ **c** ▪ **n** ▪ p^2

(n number of applications; c platform constant)

product-constrain-derive-abstract-minimize

41 Controlling Time Budgets



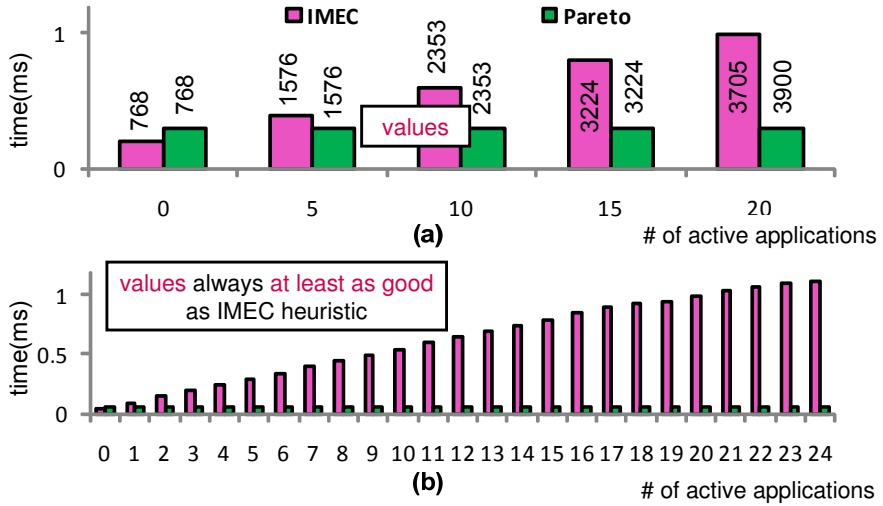
- MMKP benchmark: www.laria.u-picardie.fr/hifi/OR-Benchmark/MMKP
- Always within budget
- Good values
- Similar results to the IMEC, SOC 2006, non-compositional state-of-the-art heuristic

SimIt-ARM simulator
206 Mhz StrongARM

42 Compositionality

- Applications start at various times
- 5 at a time (a), 1 at a time (b)

what about applications stopping ?



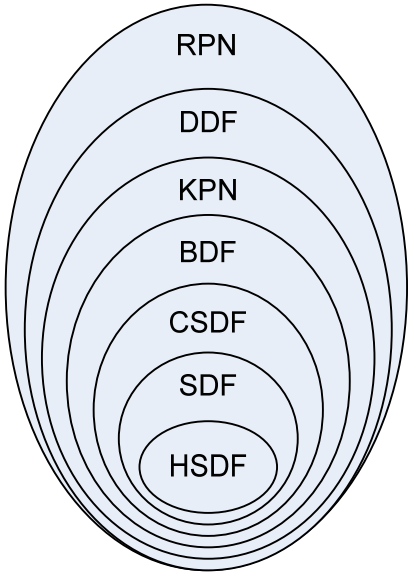
43 Conclusions Run-time Adaptation

- Run-time adaptation:
  product - derive | constrain | abstract - minimize
  … select and configure
- Parameterized compositional method
  - Allows to trade off quality with analysis time
  - Allows to bound analysis time
  - Open question: components leaving the composition ?
- Feasible for resource-constrained embedded systems
- Wide variety of applications

44 Overview

- Embedded Multi-media
- Analysis of Synchronous Dataflow Graphs
  - Throughput Analysis
  - Throughput-Storage Trade-off Analysis
  - Scenario-aware Analysis
- Run-time Adaptation
- Looking Forward

45 Dataflow MoCs 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
BDF: Boolean DF
CSDF: Cyclo-Static DF
SDF: Synchronous DF
HSDF: Homogeneous SDF

46 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

47 Thank you !

Questions ?

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

'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