Technische Universiteit Eindhoven University of Technology

InvasIC seminar, Erlangen, November 2014

Making Data Flow, Dynamically

Twan Basten

Eindhoven University of Technology & TNO Embedded Systems Innovation

Joint work with
Marc Geilen, Sander Stuijk, Bart Theelen
and many others

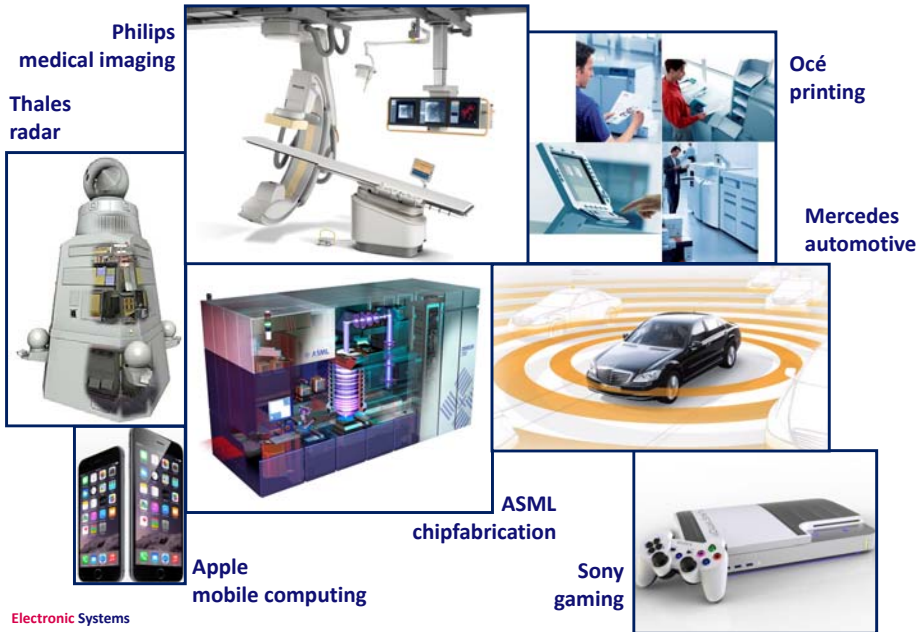
'Knowing is not understanding.'
Charles Kettering

TU/e

Predictable computing !?

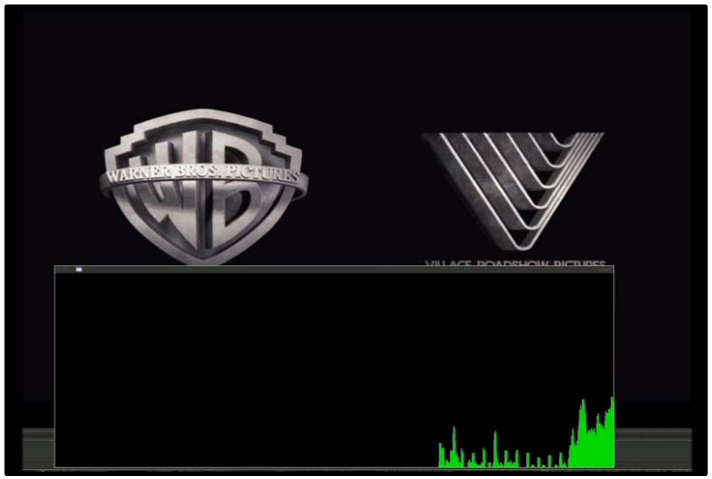
Data-intensive systems are everywhere!

TU/e



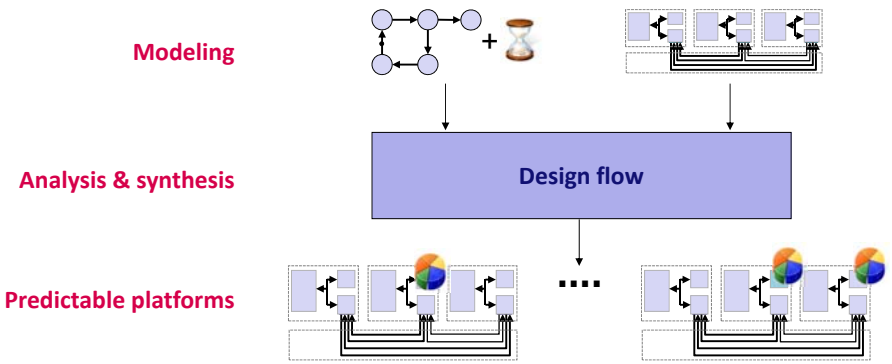
Challenge: Data-dependent data-processing workloads

TU/e



[Thx to Martijn Koedam]

What we need: Model-driven design, Predictable Platforms



Outline

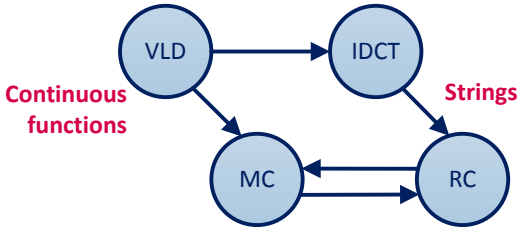
- Predictable computing !?
- Data flow models of computation
- SDF analysis and synthesis
- SADF analysis and synthesis
- Interaction
- Conclusions

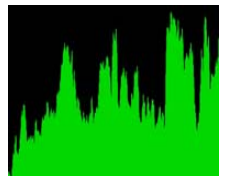
Data flow models of computation

Kahn Process Networks

(KPN [Kahn 1974])

MPEG-4 SP:



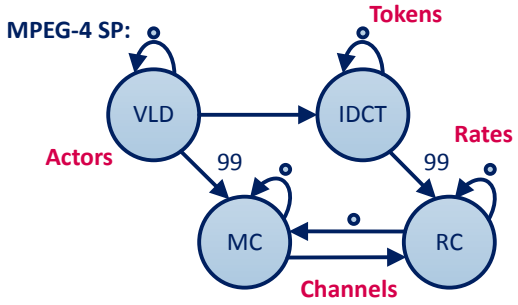


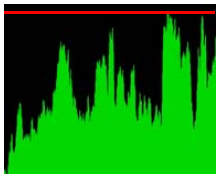
Adding interaction:
- Reactive Process Networks (RPN)
- Marc Geilen, 2004

- Can express any continuous function on strings
- Captures “precisely” all streaming data flow applications
- Doesn’t capture interaction with the environment
- No analysis & synthesis (other than functionality)

9 Synchronous Data Flow

(SDF [Lee 1986])

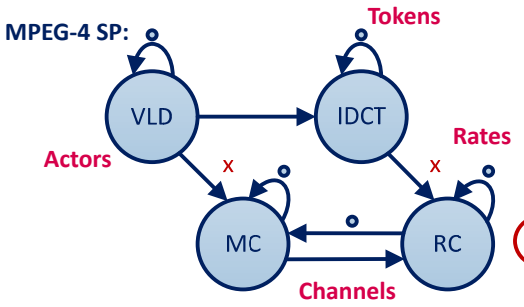




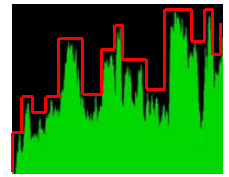
- Conservative, worst-case abstraction
- Strong analysis & synthesis support
- Low implementation overhead …
- … but may lead to over-allocation of resources

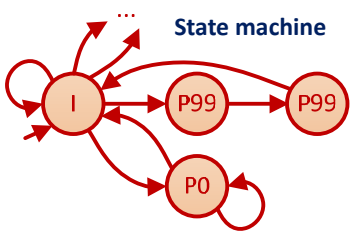
10 Scenario-Aware Data Flow

(SADF [Theelen 2006])



x = {0, 30, 40, 50, 60, 70, 80, 99}





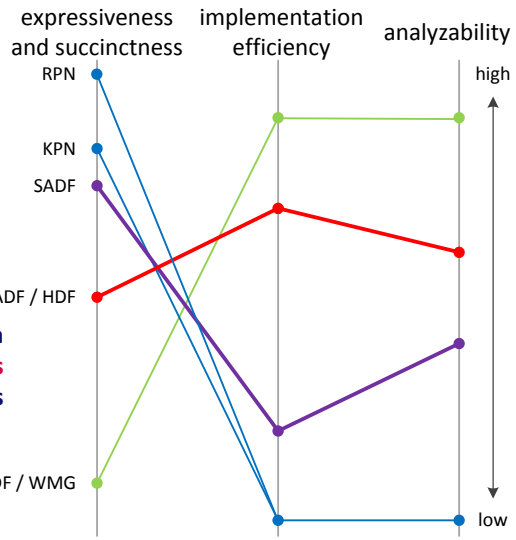
- Dynamics captured in scenarios
- Good compromise between expressivitiy, analysis & synthesis
- Efficient implementations

11 Comparison of data flow Models of Computation

implementation efficiency and analyzability vs. expressiveness are contradictory

but FSM-SADF provides a good compromise

SADF with Finite State Machines specifying scenario transitions



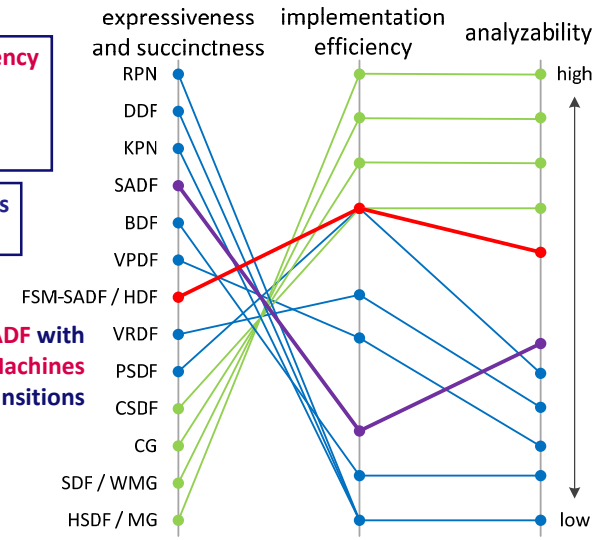
(SAMOS 2011)

12 Comparison of data flow Models of Computation

implementation efficiency and analyzability vs. expressiveness are contradictory

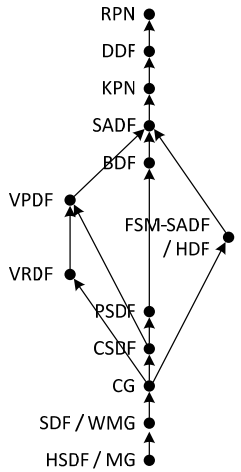
but FSM-SADF provides a good compromise

SADF with Finite State Machines specifying scenario transitions



(SAMOS 2011)

13 Data flow Models of Computation: Expressiveness hierarchy



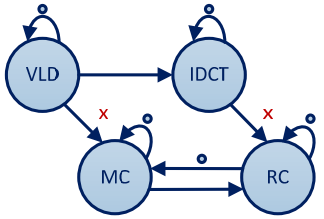
- Known or straightforward inclusion relations
- BDF and above: Turing complete
- Better expressiveness notions needed
- Unification needed

(SAMOS 2011)

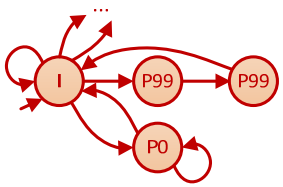
14 Targeting models to the problem at hand is important

2-processor implementation of MPEG4-SP

SADF



x = {0, 30, 40, 50, 60, 70, 80, 99}



SDF

Execution times	
VLD	33
IDCT	14
MC	325
RC	292

Data rate	
x	99

Actor worst-cases occur for different scenarios

Execution times		
VLD	P0	0
	other	33
IDCT	P0	0
	other	14
MC	I,P0	0
	P30	75
	P40	121
	P50	158
	P60	196
	P70	221
	P80	258
	P99	325
RC	I	292
	P0	0
	P30,40,50	208
	P60	250
	P70,80,99	267

15 Targeting models to the problem at hand is important

MPEG4-SP				MP3
metric	SDF	SADF	improvement	improvement
throughput	15 (frames/s)	15 (frames/s)	NA	NA
processor capacity	88 %	26 %	70 %	-
memory capacity	254 KB	254 KB	-	21 %
bandwidth	33 %	33 %	-	23 %

16 Outline

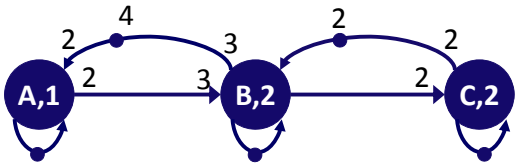
- Predictable computing
- Data flow models of computation
- SDF analysis and synthesis
- SADF analysis and synthesis
- Interaction
- Conclusions

TU/e

SDF
analysis and synthesis

 SDF: Iterations

TU/e



Iteration

smallest non-empty set of actor firings
that does not change the token distribution

(example: A:3, B:2, C:1)

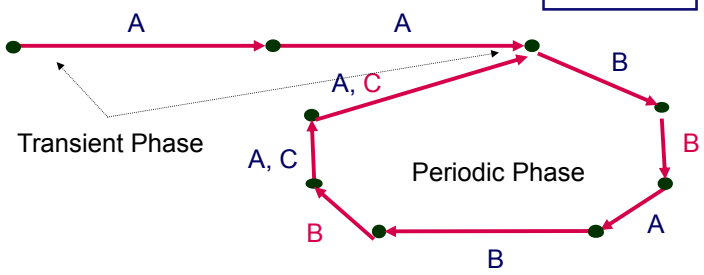
Crucial concept in data flow analysis

 SDF throughput analysis

TU/e

Self-timed execution:

- State space
- Time view



Throughput: average number of actor firings over time

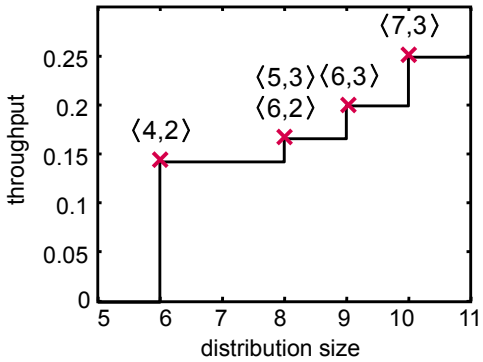
Throughput can be calculated from the periodic phase

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

(ACSD 2005)

 Storage vs. throughput trade offs

TU/e



- Separate buffer per channel
- Storage distribution, e.g., $\langle \alpha, \beta \rangle \rightarrow \langle 4,2 \rangle$

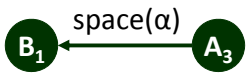
Find all minimal storage distributions for any possible throughput

(DAC 2006, TC 2008)

21 Storage vs. throughput: Dependency analysis

Storage distribution ⟨α, β⟩ → ⟨4,2⟩

dependency:
an actor firing start may depend on an actor firing end



e.g., the 3rd firing of A, A_3, depends on the first firing of B, B_1

space limitation

resolving space dependencies may increase throughput

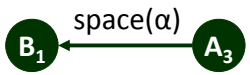
Efficient implementation
- capture dependencies during state-space exploration …
- at actor level

(DAC 2006, TC 2008)

22 Storage vs. throughput: Dependency analysis

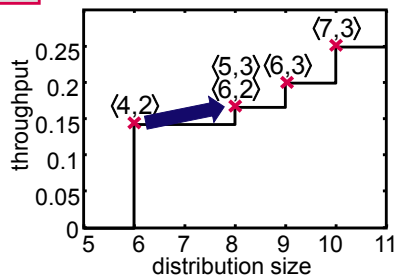
Storage distribution ⟨α, β⟩ → ⟨4,2⟩

dependency:
an actor firing start may depend on an actor firing end



e.g., the 3rd firing of A, A_3, depends on the first firing of B, B_1

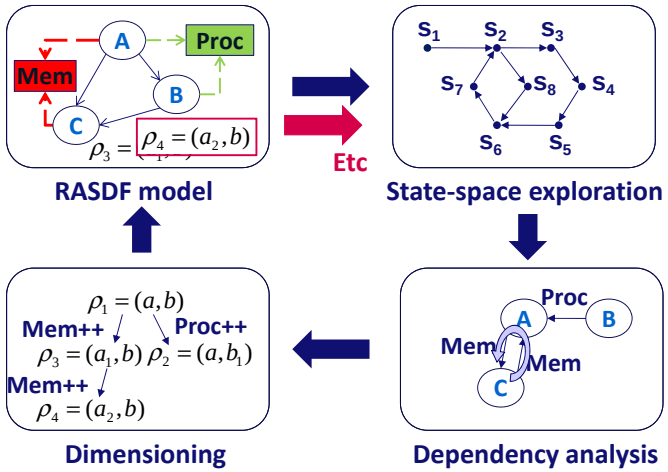
space limitation



(DAC 2006, TC 2008)

23 Trade-off analysis – resource sharing

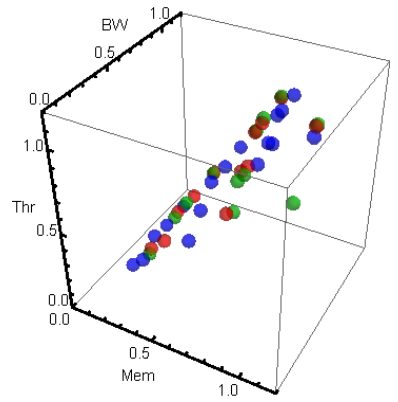
(Resource-Aware SDF, RASDF [Yang 2009])



(DATE 2010)

24 Trade-off analysis – resource sharing

(Resource-Aware SDF, RASDF [Yang 2009])

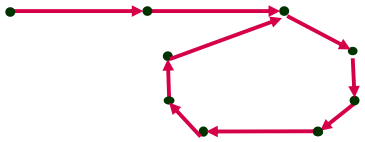


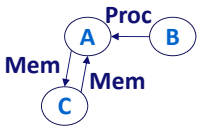
- Printer architecture exploration
- 3 architectures (colors)
- Throughput, memory, bandwidth trade offs

(DATE 2010)

25 Lessons learned: key concepts

- Iterations
- State-space exploration
- Dependency analysis





These concepts are re-usable !

26 Outline

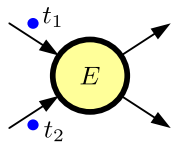
- Predictable computing
- Data flow models of computation
- SDF analysis and synthesis
- SADF analysis and synthesis
- Interaction
- Conclusions

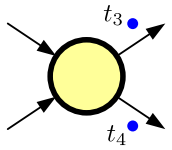
27

SADF analysis and synthesis

28 SADF throughput analysis

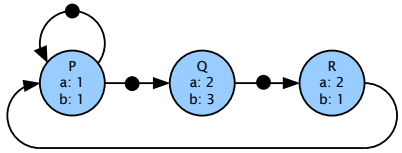
- Analysis based on (max, +)-algebra

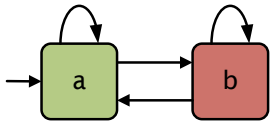




$$t_3 = t_4 = \max(t_1, t_2) + E$$

- SADF graph with two scenarios a and b
- Each iteration executes in one scenario

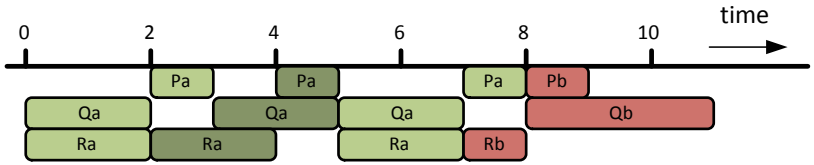


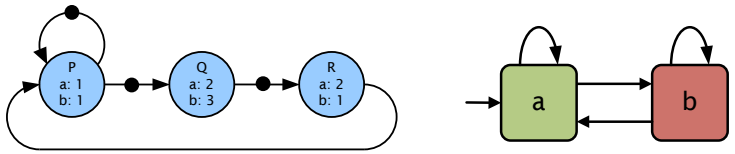


(CODES+ISSS 2010)

29 SADF throughput analysis

- Gantt chart for the scenario sequence aaab

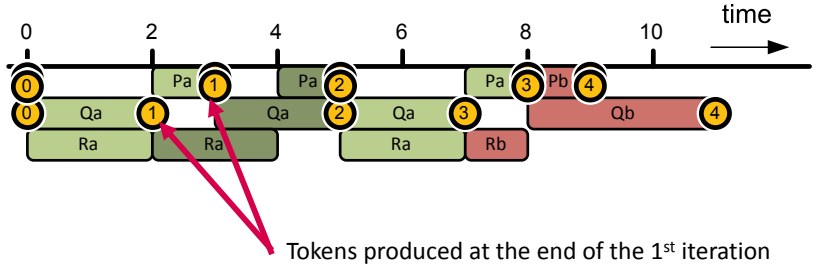


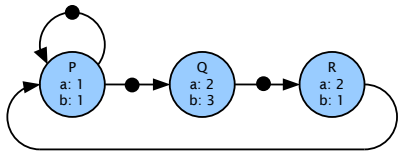


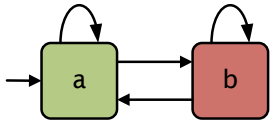
(CODES+ISSS 2010)

30 SADF throughput analysis

- Gantt chart for the scenario sequence aaab



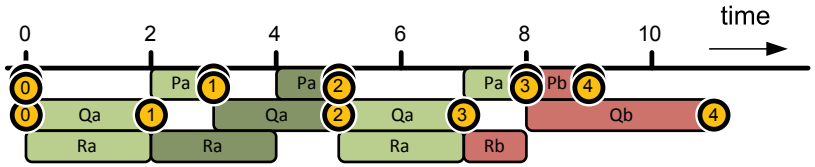




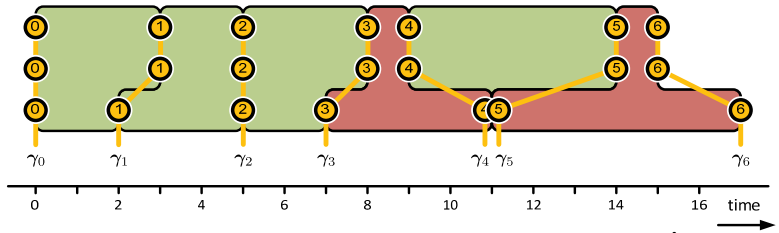
(CODES+ISSS 2010)

31 SADF throughput analysis: time-stamp vectors

- Gantt chart for the scenario sequence aaab



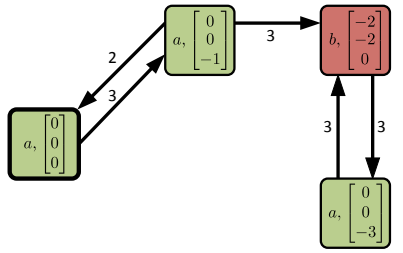
- Execution is a sequence of vector shapes
- Is nicely captured by matrix multiplication in (max,+)-algebra



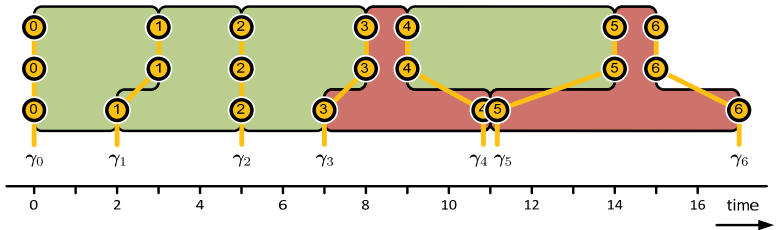
(CODES+ISSS 2010)

32 SADF throughput analysis: state-space analysis

- State space of the SADF



- Execution is a sequence of vector shapes
- Is nicely captured by matrix multiplication in (max,+)-algebra

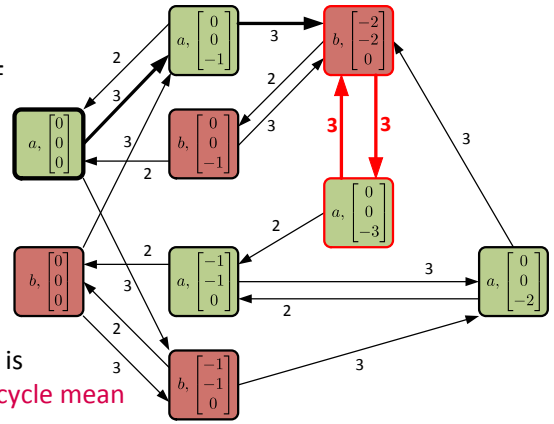


(CODES+ISSS 2010)

33 SADF throughput analysis: state-space analysis

TU/e

- State space of the SADF



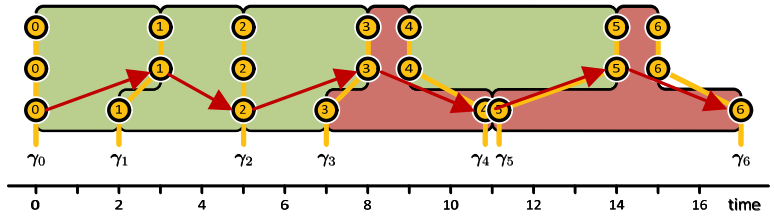
- Worst-case throughput is the inverse of the max cycle mean

(CODES+ISSS 2010)

34 SADF throughput analysis: (Max,+)-automata

TU/e

- Critical path is through token dependencies

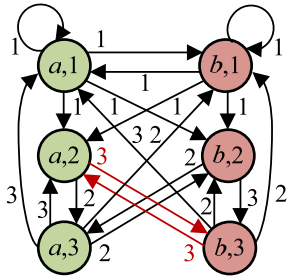


(CODES+ISSS 2010)

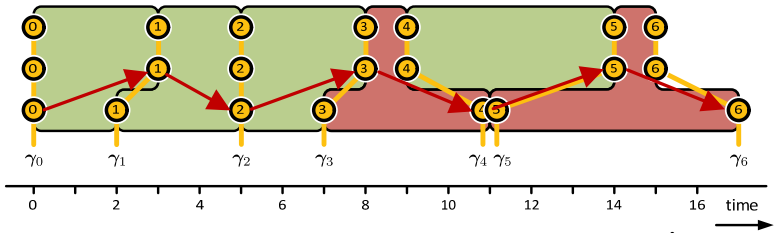
35 SADF throughput analysis: (Max,+)-automata

TU/e

- (Max,+)-automaton
- Worst-case throughput is again the inverse of the max cycle mean



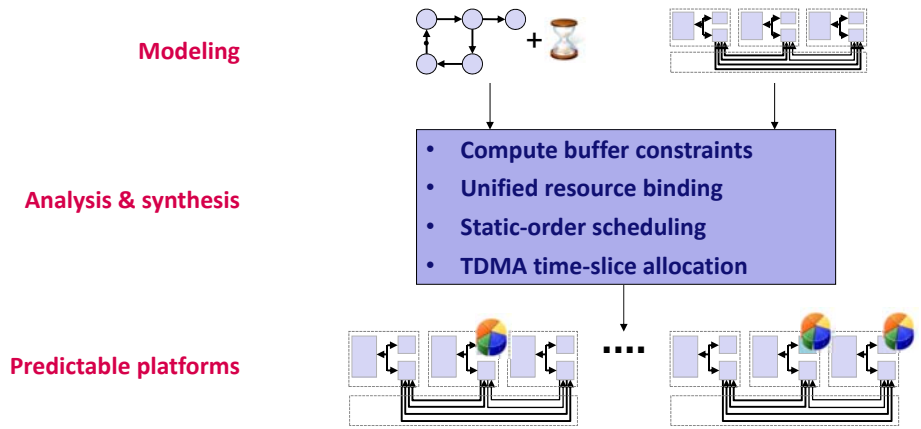
- Critical path is through token dependencies



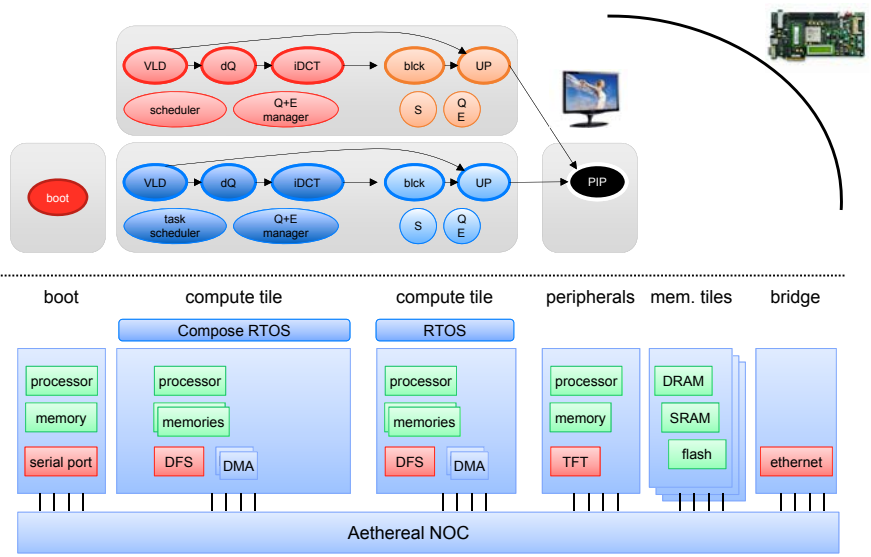
(CODES+ISSS 2010)

36 Model-driven SADF design flow: the SDF3 toolkit

TU/e



(DSD 2010, www.es.ele.tue.nl/sdf3)



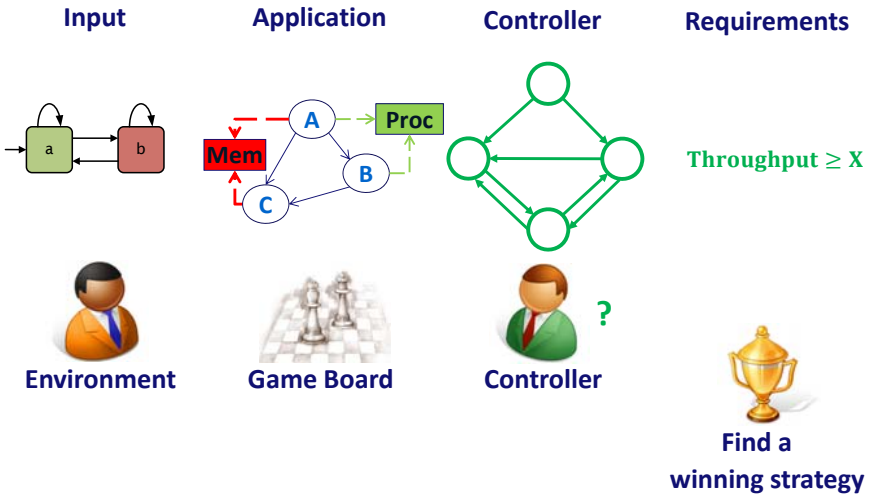
(SIGBED 2013, www.compsoc.eu)

- Predictable computing
- Data flow models of computation
- SDF analysis and synthesis
- SADF analysis and synthesis
- Interaction
- Conclusions

Interaction

- Can throughput be guaranteed ?
- Can energy usage be optimized under a throughput constraint ?

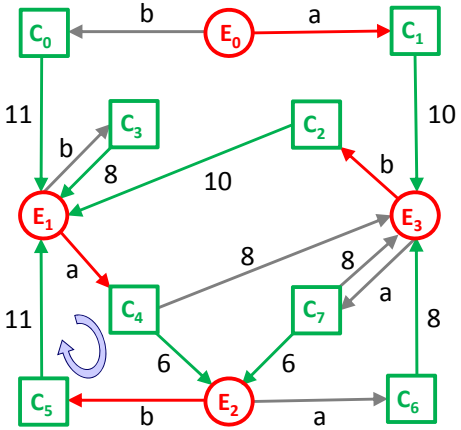
41 Realizing guaranteed throughput becomes a game



(DATE 2012)

42 Controller is constructed by solving a mean payoff game

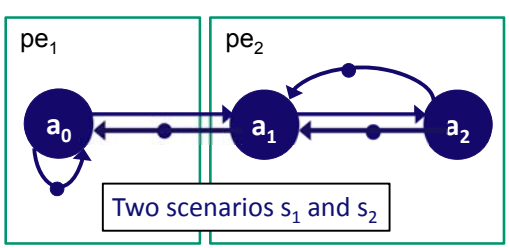
- Game graph can be derived from SADF graph
- For every input scenario, controller executes a pre-defined schedule with the indicated latency
- Solved by policy iteration
  - Red encodes the worst-case input
  - Green encodes the optimal controller
- Guaranteed throughput: inverse of the maximum cycle mean

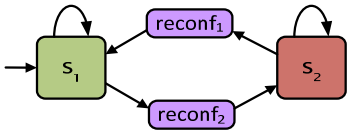


(DATE 2012)

43 Dynamic Frequency and Voltage Scaling

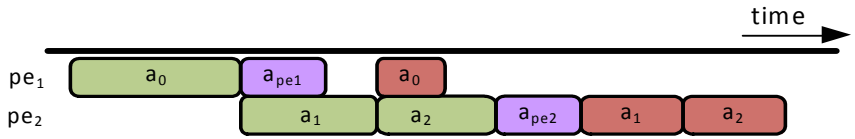
SADF model of mapped application, including reconfigurations





- Find frequency scale factor per processor per scenario
- within throughput constraint
- minimizing energy

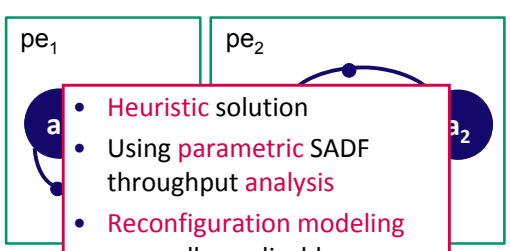
Reconfiguration SDF graphs



(RTAS 2013)

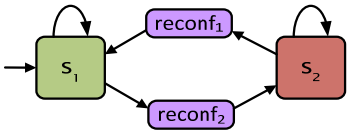
44 Dynamic Frequency and Voltage Scaling

SADF model of mapped application, including reconfigurations

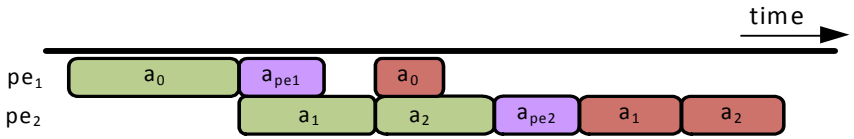


- Heuristic solution
- Using parametric SADF throughput analysis
- Reconfiguration modeling generally applicable
- Game-theoretic solution ?

Reconfiguration SDF graphs



- Find frequency scale factor per processor per scenario
- within throughput constraint
- minimizing energy



(RTAS 2013)

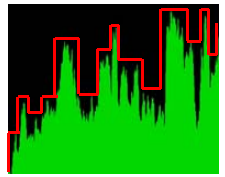
45 Outline

- Predictable computing
- Dataflow models of computation
- SDF analysis and synthesis
- SADF analysis and synthesis
- Interaction
- Conclusions

46 Conclusions

- Data flow MoCs:
  - Increasingly important
  - Good compromise between expressivity and analyzability
  - Synthesis feasible, leading to efficient implementations
- SDF3: Fully operational flow from SDF to CompSOC platform
- Challenges: Coping with dynamics
  - Expressivity, unification of MoCs
  - Modeling mapping decisions
  - Predictable reconfiguration
  - Buffer & memory analysis
  - Resource sharing
  - Controller synthesis
  - Multi-application synthesis
  - …

It's an exciting field !



47 Thank you !

Questions ?

More info: **www.es.ele.tue.nl/~tbasten/**

SDF3 toolkit: **www.es.ele.tue.nl/sdf3/**
CompSOC: **www.compsoc.eu**

Twan Basten

Sander Stuijk

Marc Geilen