



*Tutorial on
Predictability and Composability*

TU/e Technische Universiteit
Eindhoven
University of Technology

Kees Goossens <k.g.w.goossens@tue.nl>
Electronic Systems Group
Electrical Engineering Faculty

Where innovation starts

programme 1

13.00-13.30 Composable Timing and Energy in [CompSOC](#) (Kees Goossens).
13.30-14.15 Modeling [software defined radio](#) applications with dataflow (Orlando Moreira)
14.15-14.45 [Predictable](#) MPSoC architectures – [techniques](#) (Benny Akesson)
14.45-15.15 Predictability in the CoMPSoC platform – [processor tile](#) (Anca Molnos)
15.15-15.45 Break
15.45-16.30 Hands-on session using the [SDF3](#) dataflow analysis and mapping tool set
(Sander Stuijk)
16.30-17.00 SDF3/CompSOC [demonstration](#) and Q&A

all tutorial material will be available to you here & online

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

the CompSOC/SDF3 team

2

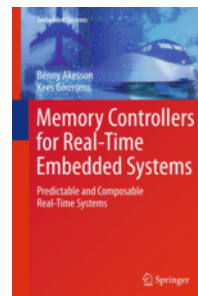
- Eindhoven university of technology
 - Benny Akesson
 - Martijn Koedam
 - Radu Stefan
 - Sven Goossens
 - Manil Dev Gomony
 - Shubhendu Sinha
- Delft university of technology
 - Anca Molnos
 - Arnaldo Azevedo
 - Karthik Chandrasekar
 - Davit Mirzoyan
 - Ashkan Beyranvand Nejad
 - Andrew Nelson
 - Pavel Zaykov
- in close collaboration with the dataflow research (SDF3) at TU/e
 - Sander Stuijk
 - Marc Geilen
 - and many others

This research is supported by
EU grants **T-CTREST**, **Cobra**
and NL grant **NEST**.
Parts of the platform were
developed in COMCAS,
Scalopes, TSAR, NEVA, MESA.

more information

3

- CompSOC, in Multiprocessor System-on-Chip. Huebner (ed), Springer, 2010
- Aethereal real-time NOC, DAC'10
- Predator real-time DRAM memory controller, DATE'11
- CompOSe RTOS, MICPRO'11
- composable power management, SAMOS'11
- SDF3, DAC'06 Stuijk, et al. <http://www.es.ele.tue.nl/sdf3/>



*Composable
Timing and Energy
in CompSOC*

Kees Goossens
& the CompSOC/SDF3 team




TU/e Technische Universiteit
Eindhoven
University of Technology

Kees Goossens <k.g.w.goossens@tue.nl>
Electronic Systems Group
Electrical Engineering Faculty

Where innovation starts

trend: embedded systems 5

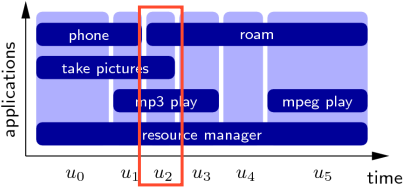
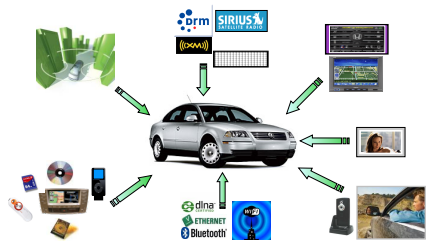
- phones, game consoles, cars, refrigerators, buildings, ...
- interaction with physical world → **real-time** requirements



trend: multiple applications on one device

6

- audio, video, graphics, games, artificial intelligence, internet, ...
- different **application domains** have diverse
 - requirements
 - methodologies
- independent software vendors
- **use cases**



© Kees Goossens
Electronic Systems

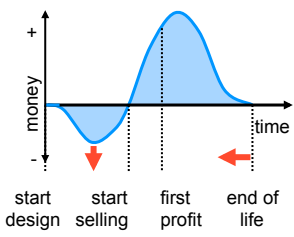
TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

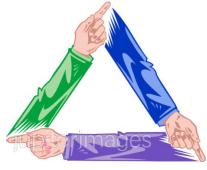
problem: design time

7

- systems are complex
- time is short
- it takes too long
- getting worse



- **monolithic verification after integration**
 - hardware, multiple applications
- circular verification
 - who to blame for errors?



© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

goal & approach

8


- reduce SOC design effort
- **independent design, verification, and deployment per application**
 - application as the unit of verification & re-use
- composability
- predictability

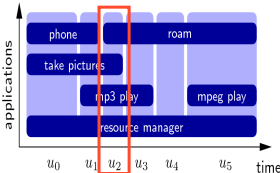
© Kees Goossens Electronic Systems TU/e tutorial 2012-01-27 TU/e Technische Universiteit Eindhoven University of Technology

composability

9

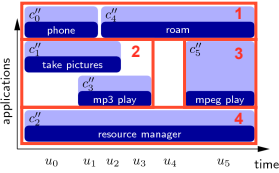
- **virtual platform** per application
- space, time, energy budgets
- design, verify, deploy in a virtual platform
- **no interference**
 - when integrating or switching use cases





inter-dependent
 $2^N \times 2^N \times 2^N \times 2^N$

→



independent
 $2^N + 2^N + 2^N + 2^N$

composability

10

- time-division multiplex virtual platforms
- **no interference**
 - space, time, energy

inter-application scheduling:
hypervisor, RTOS, ...

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

composability & predictability

11

- programming model, scheduling, power management per application
- for RT use predictable schedulers, and associated real-time formalism

intra-application scheduling
& power management

inter-application scheduling

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

composability

12

1. resources & users
2. no resource-resource dependencies
3. task \rightarrow resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling

© Kees Goossens
Electronic Systems

composability

13

- 1. resources & users
2. no resource-resource dependencies
3. task \rightarrow resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling

- processors, interconnect, memories
- applications, tasks, transactions

© Kees Goossens
Electronic Systems

composability

14

- resources & users
- no resource-resource dependencies
- task → resource binding is a function
- composable sharing
- unaligned scheduling intervals & periods
- optional: 2-level scheduling

- code & data fit in local memories
- no proc. – cache – NOC – memory dependencies, etc.
- NOC is one resource

© Kees Goossens
Electronic Systems

composability

15

- resources & users
- no resource-resource dependencies
- task → resource binding is a function
- composable sharing
- unaligned scheduling intervals & periods
- optional: 2-level scheduling

- no task migration
- use DMA for remote memory loads

© Kees Goossens
Electronic Systems

composability

16

1. resources & users
2. no resource-resource dependencies
3. task \rightarrow resource binding is a function
- 4. composable sharing**
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling

- preemption
- scheduling interval, period
- time-division multiplexing (TDM)
- **no interference** between applications

application scheduling

processor interconnect SRAM DRAM

© Kees Goossens
Electronic Systems

composability

17

1. resources & users
2. no resource-resource dependencies
3. task \rightarrow resource binding is a function
4. composable sharing
- 5. unaligned scheduling intervals & periods**
6. optional: 2-level scheduling

- resources have different
 - service units, periods
 - cost of preemption
- GALS, DVFS
- small service units \rightarrow non-determinism

processor interconnect SRAM DRAM

© Kees Goossens
Electronic Systems

composability

18

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
- 6. optional: 2-level scheduling

- on processor only
- separated scheduling & power management
 - trusted system inter-app
 - untrusted user-defined intra-app
 - time, energy, and power budgets

intra-application task scheduling

application scheduling

processor interconnect SRAM DRAM

© Kees Goossens
Electronic Systems

CompSOC

19

user space

application 1 application 2 application 1 application 3

task sch. PM task sch. PM task sch. PM task sch. PM

T1 T2 T1 T2 T3 T3 T4 T5 T6

.....

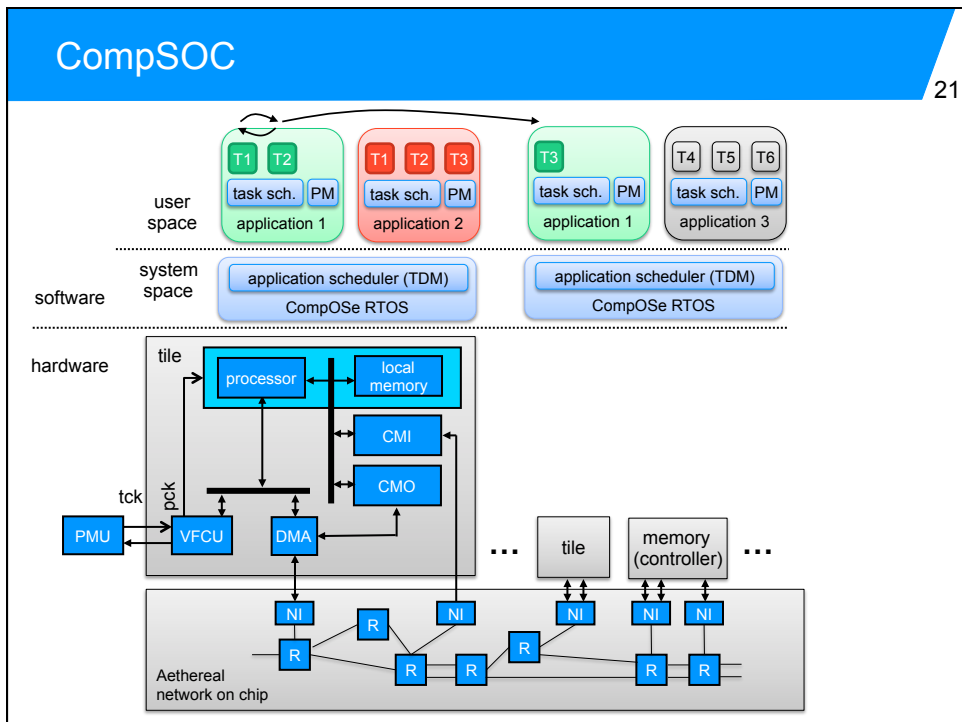
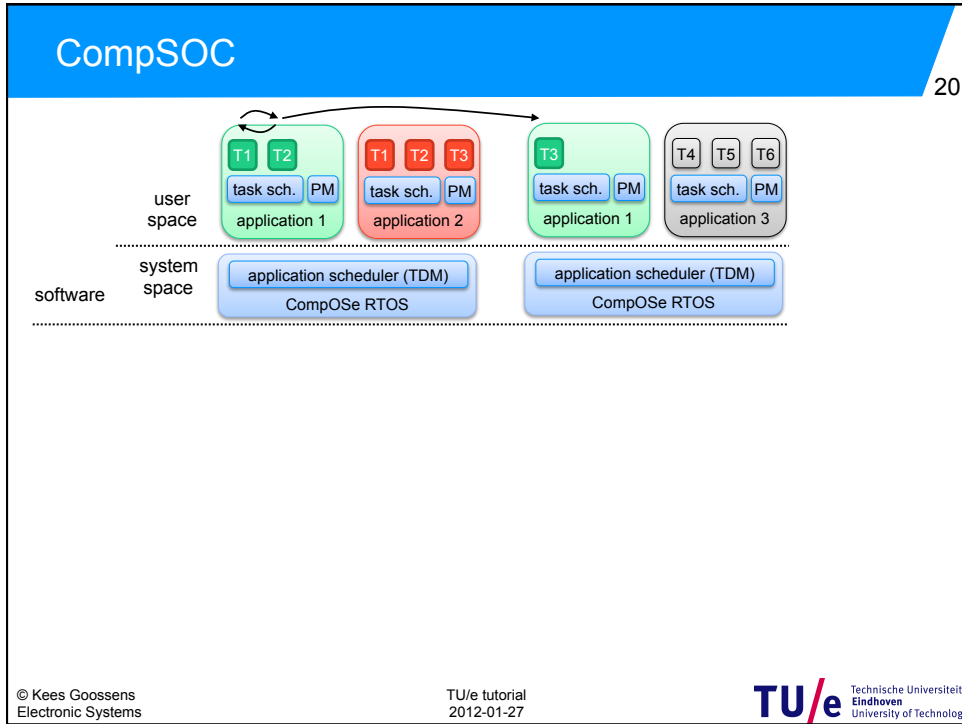
software

.....

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology



predictability

22

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
7. **formal analysis**

↑
model the application, resources, dependencies, binding, ...
↓

processor

interconnect

SRAM

DRAM

© Kees Goossens
Electronic Systems

predictability

23

1. resources & users
2. no resource-resource dependencies
- 3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
7. formal analysis

- worst-case execution time of a request on the unshared resource
 $WCET = fn(request, resource)$
- compositional

$WCET = fn(request, resource)$

processor

interconnect

SRAM

DRAM

© Kees Goossens
Electronic Systems

predictability

24

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
- 4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
7. formal analysis

- worst-case response time takes resource sharing into account
- WCRT ≈ WCET(t,r) * period / budget
- compositional

WCRT ≈ WCET(t,r) * 4/2

processor

interconnect

SRAM

DRAM

© Kees Goossens
Electronic Systems

predictability

25

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
- 6. optional: 2-level scheduling
7. formal analysis

- WCRT takes inter-app and intra-app resource sharing into account
- WCRT ≈ WCET(t,r) * app_period / app_budget * task_period / task_budget

WCRT ≈ WCET(t,r) * 4/1 * 4/2

processor

interconnect

SRAM

DRAM

© Kees Goossens
Electronic Systems

predictability

26

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
- 7. formal analysis

- use the WCRT of each actor
- cyclo-static **dataflow**
 - scenario aware
- SDF3 methods and tools

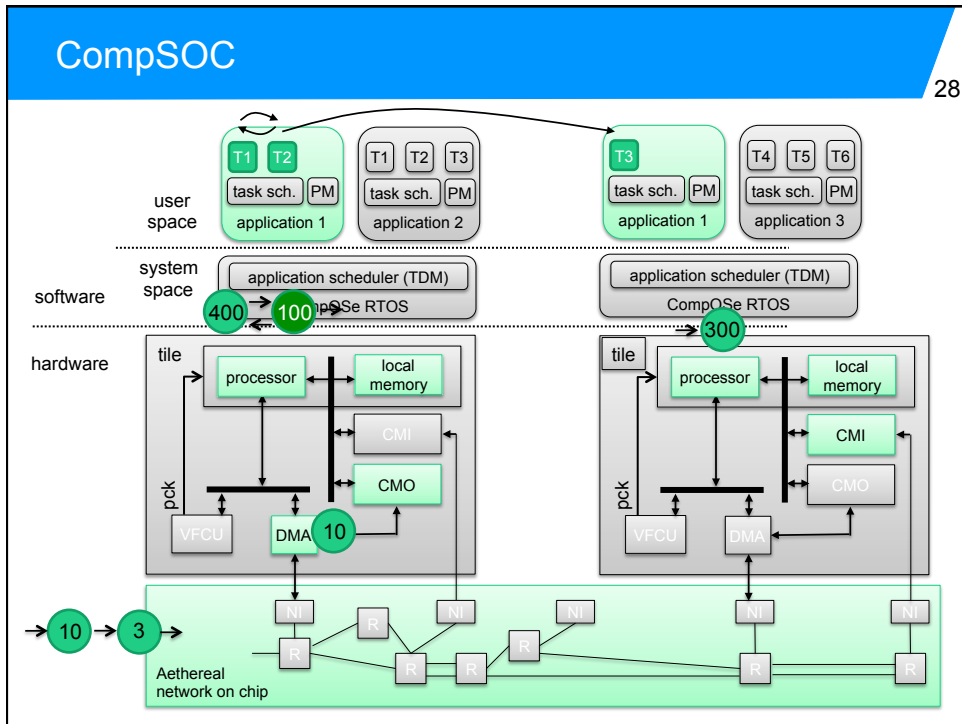
© Kees Goossens
Electronic Systems

CompSOC

27

The diagram is divided into three horizontal layers:

- user space:** Contains three application blocks. Application 1 has tasks T1 and T2; Application 2 has T1, T2, and T3; Application 3 has T3, T4, T5, and T6. Each application includes a task scheduler (task sch.) and a processor manager (PM).
- software:** Contains two application scheduler (TDM) blocks and two CompOSE RTOS blocks, which interface with the user space applications.
- hardware:** Shows two tiles. Each tile contains a processor, local memory, CMI, CMO, VFCU, and DMA. These tiles are interconnected via an Aethereal network on chip, which consists of nodes labeled N, R, NI, and P.



predictability

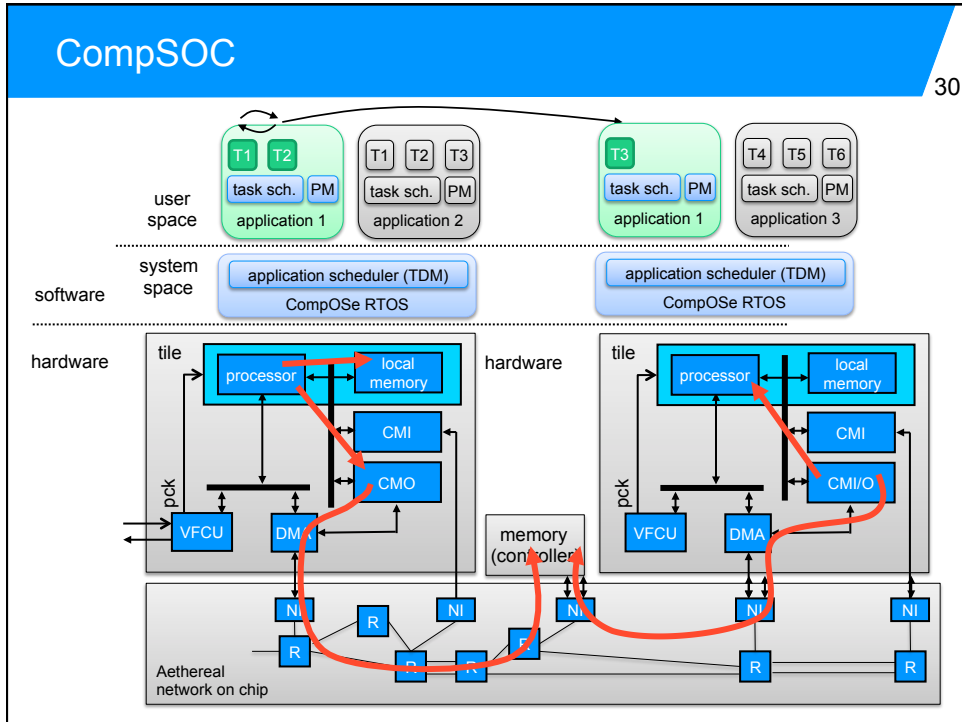
29

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
- 7. formal analysis

- buffer sizes & flow control
- latency-rate models of NOC, DRAM
- application throughput = $1/MCM$
 - MCM = max. cycle mean

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27



predictability

31

1. resources & users
2. no resource-resource dependencies
3. task → resource binding is a function
4. composable sharing
5. unaligned scheduling intervals & periods
6. optional: 2-level scheduling
- 7. formal analysis

- buffer sizes & flow control
- latency-rate models of NOC, DRAM
- application throughput = 1/MCM

The diagram shows a sequence of tasks and resources: task 1 (400) and task 2 (100) are connected to a DMA block. The DMA is connected to a NOC block, which is connected to a DRAM block. The DRAM is connected to task 3 (300). Arrows indicate the direction of data flow between these components.

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

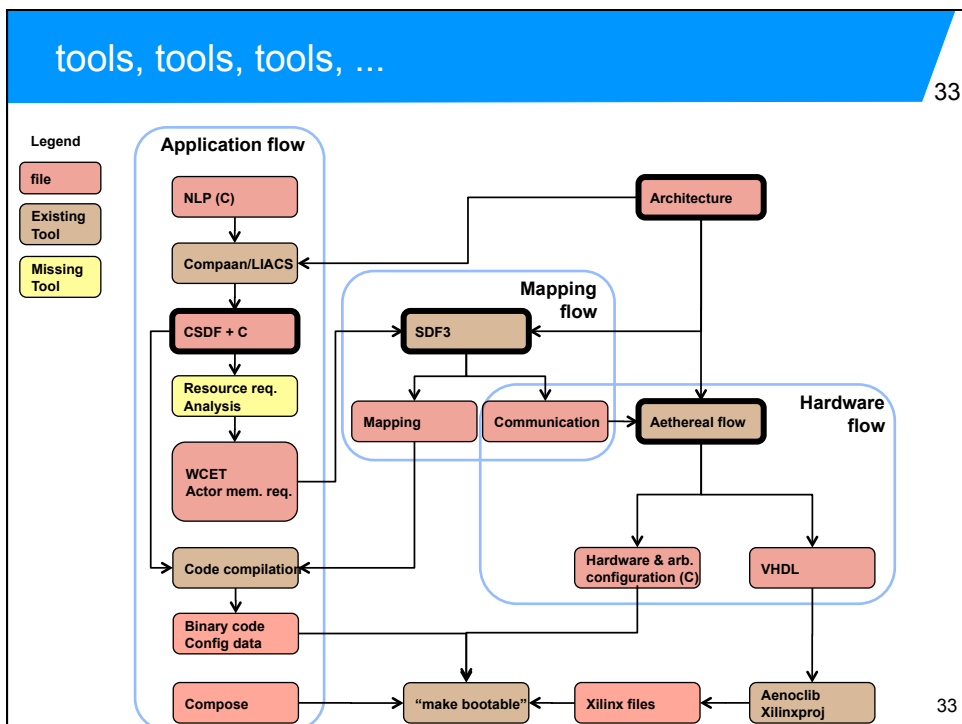
TU/e Technische Universiteit
Eindhoven
University of Technology

design flow

32

- the **SDF3 dataflow framework**
- automatic generation of
 - **hardware**: processor tiles, NOC, memory controllers
- for cyclo-static dataflow applications
 - **configurations**:
 - actor-resource binding, buffer sizes, RTOS scheduling interval, scheduler settings (TDM slots, CCSP priorities, etc.), ...
 - **software drivers**:
 - to load the configurations on the hardware at run time
 - end-to-end application throughput and latency **analysis**
- FPGA **prototype**

© Kees Goossens Electronic Systems TU/e tutorial 2012-01-27
 Technische Universiteit Eindhoven University of Technology



salient points: processor & CompOSE RTOS

34

- tracking of time, progress, energy, power, & slack
- fast DVFS, e.g. NXP [Pineda] or CEA/LETI [Vivet]
- constant scheduling interval

© Kees Goossens Electronic Systems TU/e tutorial 2012-01-27 **TU/e** Technische Universiteit Eindhoven University of Technology

salient points: Aethereal NOC

35

- global distributed scheduler (TDM)
 - single pipelined resource
 - fewer, smaller buffers
 - one level of scheduling
- best cost : performance trade-off
- latency-rate dataflow model, incl. end-to-end flow control

© Kees Goossens Electronic Systems TU/e tutorial 2012-01-27 **TU/e** Technische Universiteit Eindhoven University of Technology

salient points: Predator DRAM controller

36

1. predictable memory patterns
2. credit-controlled static priority (CCSP)
 - decoupled latency & rate
 - decoupled allocation granularity & latency
 - no over-allocation

The diagram shows a data flow from left to right. On the left, three orange arrows enter a trapezoidal block labeled '2'. Inside this block is a blue box labeled 'CCSP scheduler'. An orange arrow points from the scheduler to a blue box labeled '1' and 'pattern-based command generator'. From there, an orange arrow points to a blue box representing memory, which contains a grid of cells and a row of cells below it.

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

salient points: Predator DRAM controller

37

- predictable → composable
 - delay ET of responses to WCRT

The diagram is similar to the previous one but includes feedback loops. The '2' block (CCSP scheduler) has three orange arrows entering from the left. A blue arrow labeled 'arrival time' points from the scheduler to the '1' block (pattern-based command generator). From the '1' block, an orange arrow points to the memory block. From the memory block, an orange arrow points back to the scheduler. Below the scheduler, three orange arrows point to three red rectangular blocks. A blue arrow labeled 'release @ WCRT' points from these blocks back to the scheduler.

© Kees Goossens
Electronic Systems

TU/e tutorial
2012-01-27

TU/e Technische Universiteit
Eindhoven
University of Technology

(current) limitations

38

- multiple use cases fully supported by NOC only
- supported programming models
 - cyclo-static & variable-rate dataflow
 - Kahn process networks
- data and code must fit in local tile memories
 - no caches, or else flush on preemption
- no I/O virtualisation
- no external interrupts
- processor interrupt reserved for RTOS
 - pre-emptive intra-application task scheduling prototyped
- no memory protection
 - time, energy, and power budgets, but no space budget
- DVFS simulated on FPGA

conclusions

39

- reduce SOC design effort
- **independent design, verification, and execution per application**
 - application as the unit of verification & re-use
- composability
- predictability
- application-specific scheduling & power management
 - any mix of NRT, SRT, FRT
- design flow (SDF3, Aethereal)
- VHDL prototype
- used in teaching MSc embedded systems lab

end

40

for further information
Kees Goossens <k.g.w.goossens@tue.nl>
Electronic Systems Group
Electrical Engineering Faculty