Contracts

for

MIMOS Framework for Design and Update of Real-Time Embedded Systems

Susanne Graf

Verimag - Grenoble University

&

Uppsala University

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

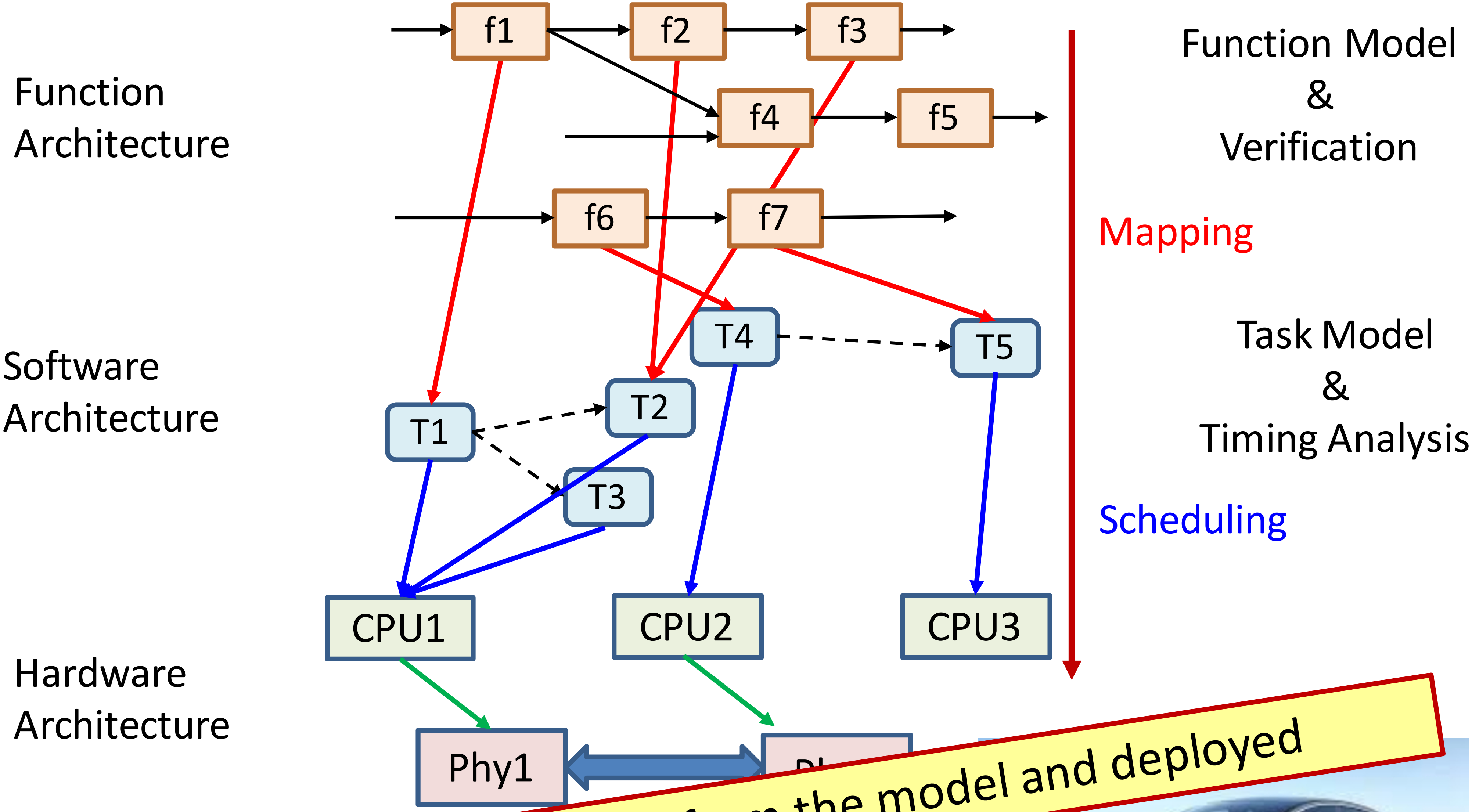
Supporting Projects

2 supporting projects:

CUSTOMER: Customizable Embedded Real-Time Systems: Challenges & Key Techniques (**Wang Yi**'s **ERC**)
UPDATE: Desgined for Update of Next-Generation Embedded Systems (Knut & Alice Wallenberg Foundation)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

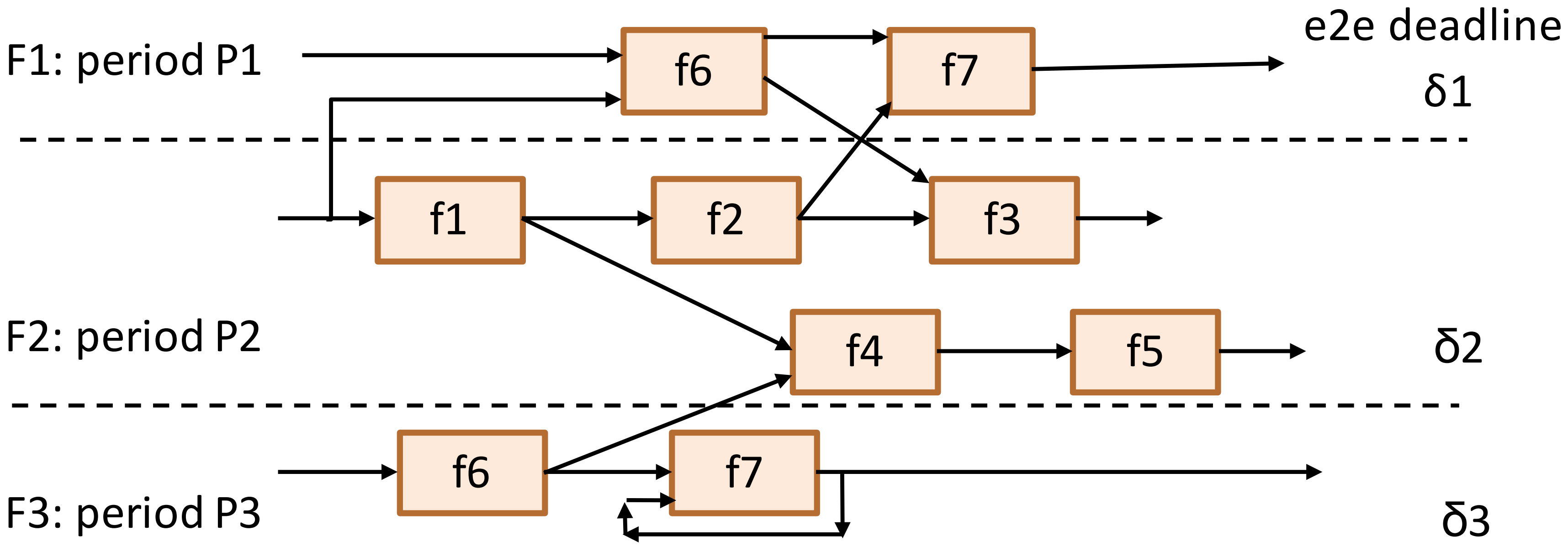
Model-Based Design of Real-Time Systems



Code running on HW: generated from the model and deployed

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Function Design and Specification



Functions: **Streams → Streams**

Requirements on the design:

- **Determinism** is fundamental
- Separation of Concern: **Independence of timing and function** (as much as possible)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Synchronous Languages and design frameworks

Many: Lustre, Esterel, Signal, SDF, SCADE …

Determinism, separation of time and function (eases verification), **semantic preserving compilation** to a **sequential C program**

Complications: Semantic preservation

- Multi-clock synchronous program as a **set of tasks on RTOS**
  - LET (logical Execution Time) in Giotto [HHK01]
- **Distributed** implementation
  - TTA (Timed Triggered Architecture) – single rate [K&al 95]
    key: det. Communication by clock-synchronization (TTP)

  More protocols for "virtually synchronous" protocols
  - PALS [M&al 09]

Determinism beyond strictly Synchronous:

- Lingua Franca Reactor Model [LL&al19+21]

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Verimag UPPSALA UNIVERSITET

OUTLINE

1. Motivations: Design of Real-time systems

2. MIMOS, a short introduction to the framework

3. The Analysis Challenges

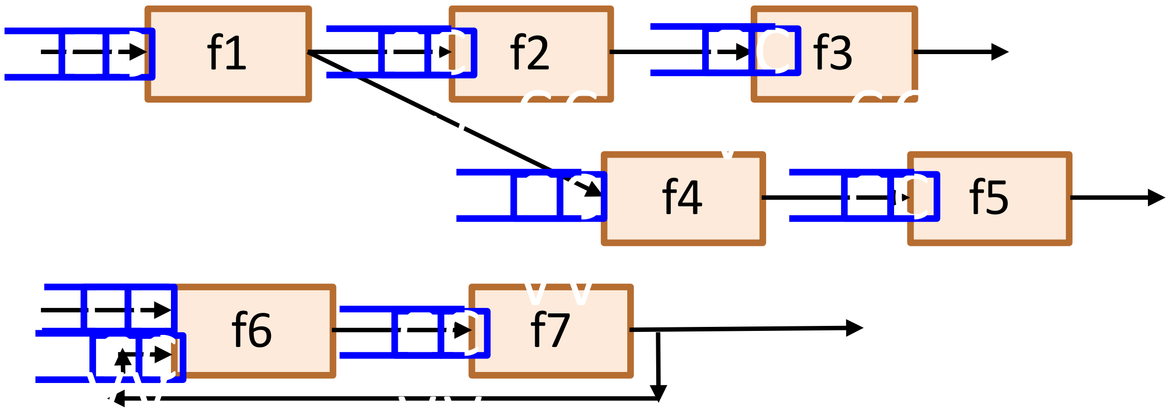
4. Contracts/Specifications for MIMOS

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Kahn Process Networks (KPN)

The semantics of a simple language for parallel programming, G. Kahn, 1974

Operational model for KPN



A network of processes

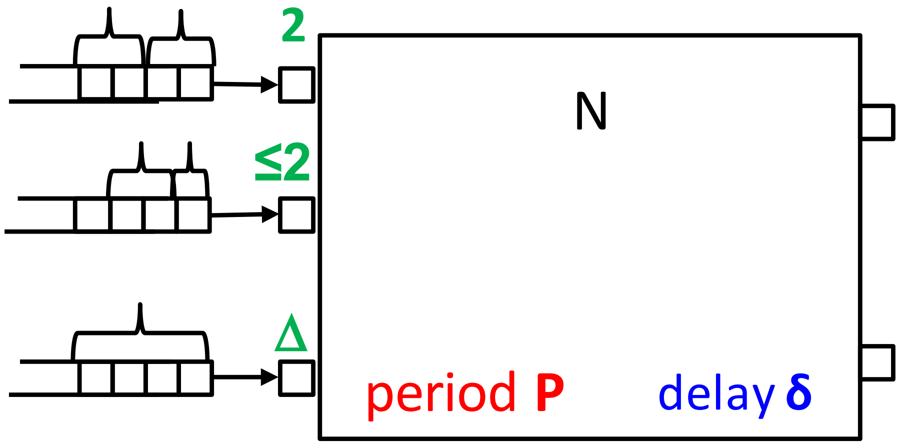
- Communicating through (potentially unbounded) FIFO buffers
- Read (and compute a step): when all required data is in the FIFOs (blocking, similar to PetriNets)
- Write: non-blocking (asynchronous)

Semantics of a KPN: a function from input to output streams (determinism) independent of execution order (scheduling) or delays (computation or communication)

SDF : nodes read & write fixed number of elements at each step

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

MIMOS for Real-time (semantics): an extension of Timed KPN



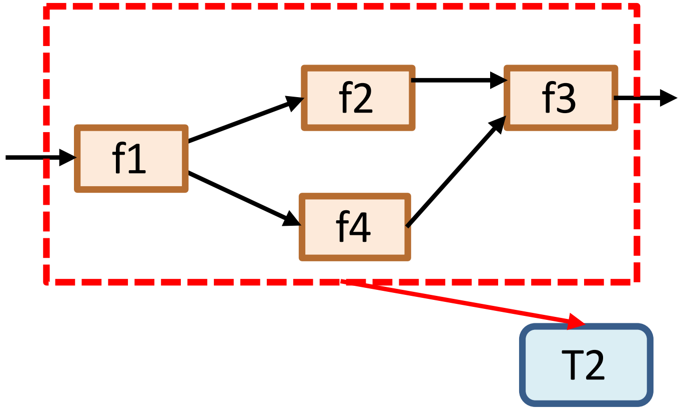
- Input ports read through FIFO from unique origin
- Output ports may have multiple destinations

- Specify a period **P** and execution delay **δ**:
  - each **P** : if sufficient input, read and apply node function (any program)
  - after delay **δ**: write output (LET)
- Each port specifies a read /write policy among:
  - read (write) exactly **k** elements
  - read (may be write) **≤k** elements (more generally **k∈[m,M]**)
  - read a time window Δ
- These strategies are meant to be deterministic on "***timed streams***" !

In [YMG20]: **MIMOS NW = KPN on timed streams**

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Different solutions (target depending)

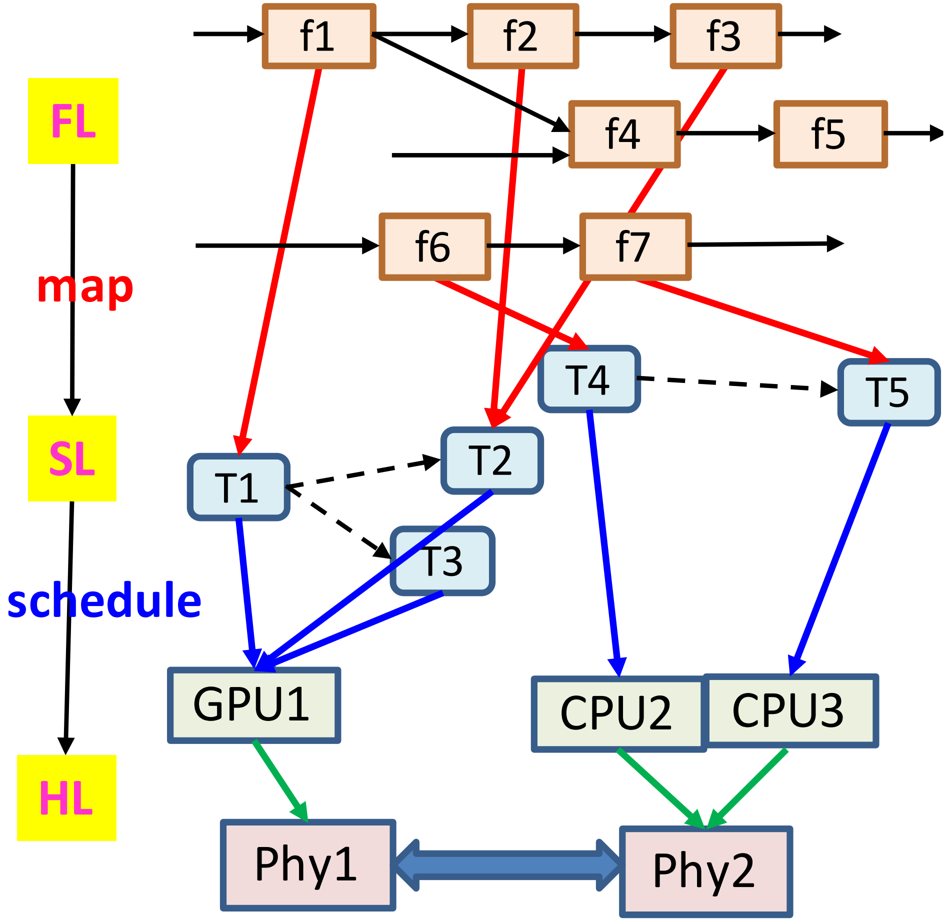


(1) a group of nodes on a single task

- back to synchronous (no FIFO)

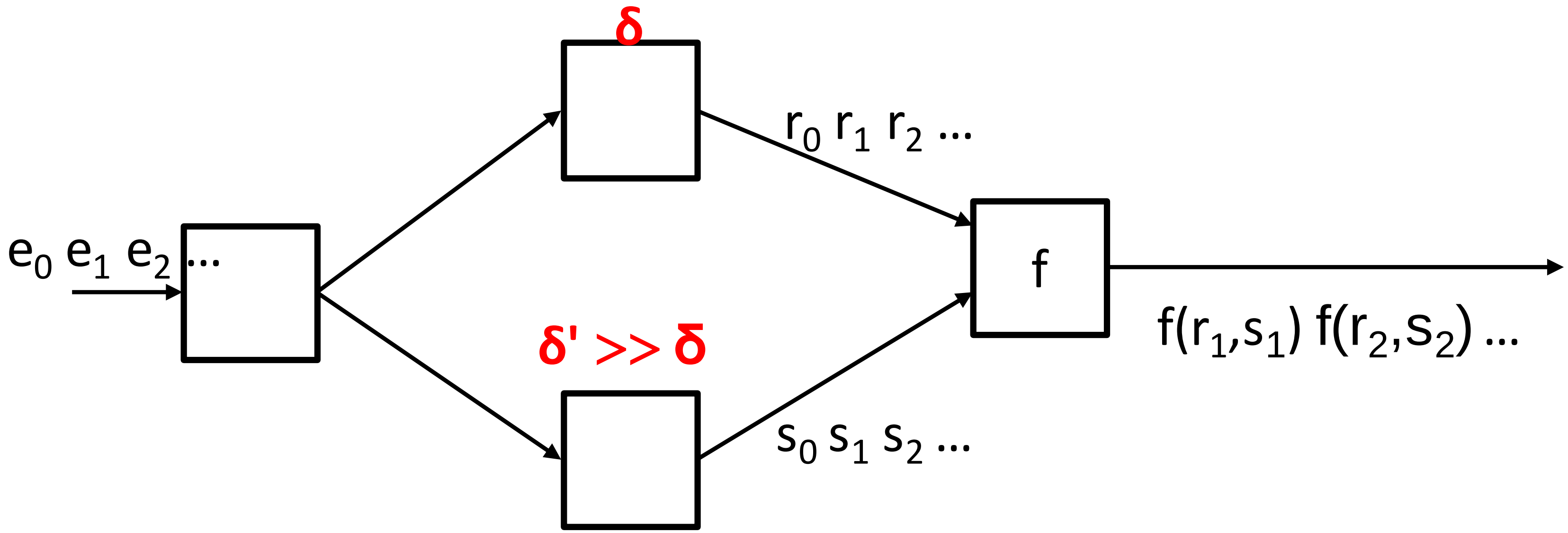
(2) Other cases:

- strict Kahn: determinism is guaranteed
- "timed read" strategies (register or sporadic streams): requires
  - time stamps
  - some timing guarantees



© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Back to a motivating example



MIMOS/Kahn computes f on the correct data independently of delay

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

OUTLINE

1. Motivations: Design of Real-time systems

2. MIMOS, a short introduction to the framework

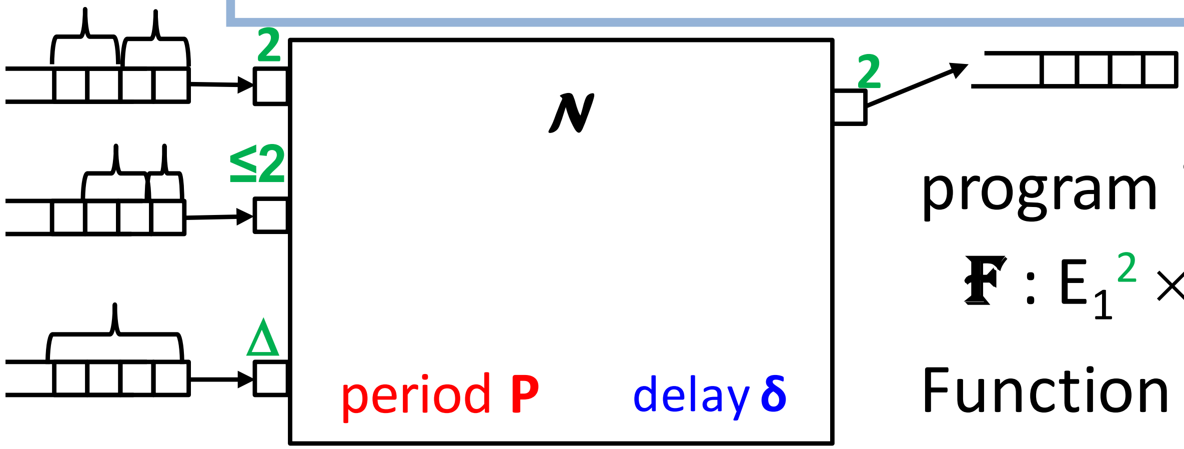
3. The Analysis Challenges

4. Contracts/Specifications for MIMOS

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Verimag
UPPSALA UNIVERSITET

The Analysis Challenge



program implementing node $\boldsymbol{N}$:

$\mathbf{F} : E_1^{2} \times E_2^{\leq 2} \times E_3^{*} \rightarrow E_4^{2}$

Function on streams represented by $\mathbf{F}$:

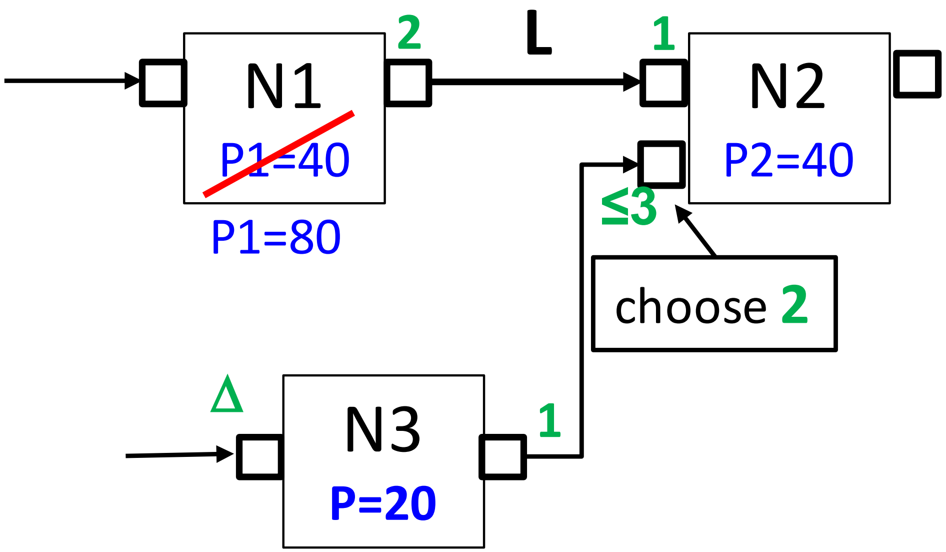
$\mathbf{F}^{\omega} : E_1^{\omega} \times E_2^{\omega} \times E_3^{\omega} \rightarrow E_4^{\omega}$

obtained by repeating $\mathbf{F}$ every period (***timed streams***)

1. Rate consistency: absence of Buffer under/overflow
   - for general KPN: is undecidable
   - for synchronous languages / SDF: easy
2. (E2E) function analysis: properties of $\mathbf{F}$ and $\mathbf{F}^{\omega}$:
   - which properties and how to express them
   - How to keep timing and function separate
   - How to derive global from local properties …

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Verimag UPPSALA UNIVERSITET



A link **L** must satisfy:

2/40 = 1/40

Let P be *minimum constraints*:

adapt P1 = 80:

2/80 = 1/40

All links must be consistent !

Flexible read/write policies make consistency easier:

- Links with “timed read strategy” are always consistent
- Flexible read/write strategies make finding a solution easier
- May be realized in different ways
- Adaptations for update are easier

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

OUTLINE

1. Motivations: Design of Real-time systems

2. MIMOS, a short introduction to the framework

3. The Analysis Challenges

4. Contracts/Specifications for MIMOS
   - ➢ Motivation for MIMOSA language: illustration on an example
     - Variety of properties to be expressed
     - The role of contracts
   - ➢ MIMOSA language

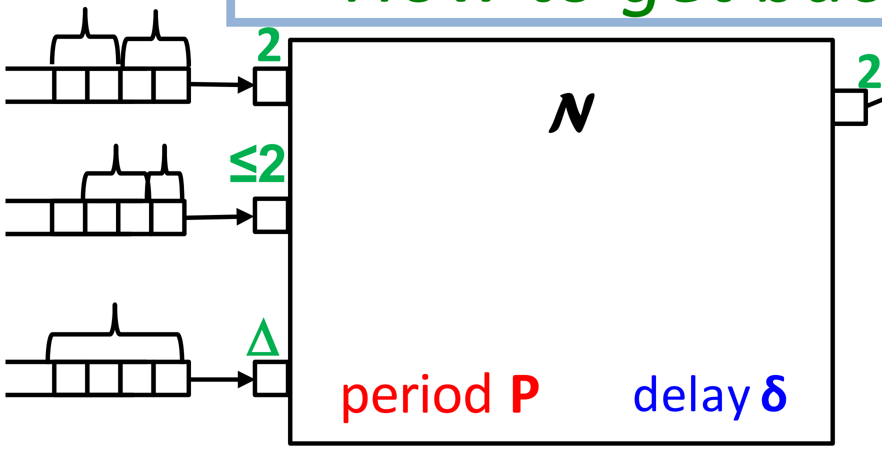
© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

General advantages of (and requirements on) the use of contracts:

1. Validate an implementation of a component in isolation (assumptions = what we need to know on environment)
2. Validate the consistency of a network should amounts to local consistency checks
3. Verification of global contracts (E2E requirements) from local ones does not require “contract composition”
4. Contracts to be used as placeholders for “not yet existing” components in a simulation
5. As we look at different “aspects” (function, timing, dependability …) separately, we may study “***inter-aspect contracts***”
6. Contracts required for “update”

How to get back separation of function and time



program implementing node ***N*** :

$\mathbf{F} : E_1^{2} \times E_2^{\leq 2} \times E_3^{*} \rightarrow E_4^{2}$

Function on streams represented by $\mathbf{F}$:

$\mathbf{F}^{\omega} : E_1^{\omega} \times E_2^{\omega} \times E_3^{\omega} \rightarrow E_4^{\omega}$

obtained by repeating $\mathbf{F}$ every period (***timed streams***)

Good News:

- Most disturbing feature is “timed read”
  - ➢ For validation: get read of it with artificial (semantic) “nil” elements
  - ➢ We get KPN, quite close to synchronous models (FIFOs will help)
  - ➢ We can build on existing contract-based approaches : e.g. Lustre-based CocoSpec [T&al2016]
- Multi-period (& feedback edges) make verification more difficult
  - ➢ A contract-based approach with assumptions will be of great help

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Contracts: 2 kinds of contracts

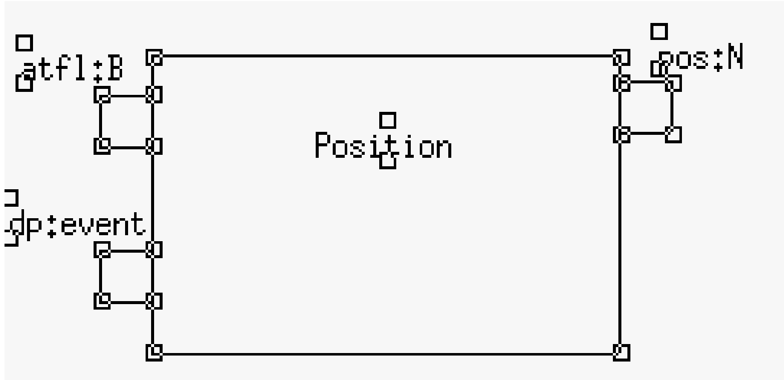
1. Contracts for the "step function" (from input segments to output segments):

   Pre/post conditions (on segments)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

2. Contracts on (complete) traces: several options
   a. Temporal properties (e.g. STL)
   b. (timed) automata
   c. We propose MIMOSA, a Lustre like functional language with modes

Example 1: a simple function



Informal spec:

- increments the position according to "increment events" dp
- corrects to nearby floor position if "atfl" event is present

1. Contract for step function

post condition: if !atfl & !dp then pos = pre(pos)
elseif !atfl & dp then pos = pre(pos) + incr
elseif atfl then pos =closestfloor(pos)

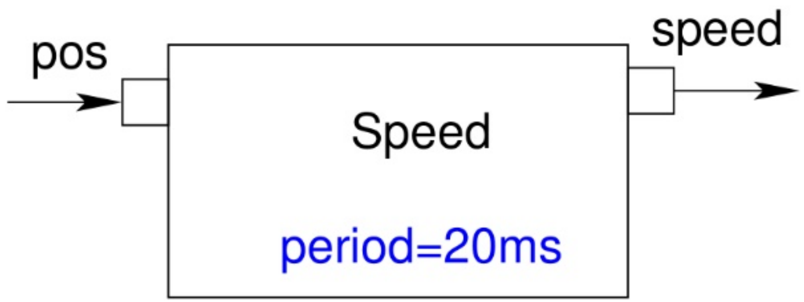
2. Temporal contract for expressing a stronger guarantee:
only small corrections are ever made

always(dist(pos,pre(pos)) ≤ 2*incr)

Requires an **assumption** (on consistency between input streams):

always (atfl ==> dist(pre(pos),closestfloor(pre(pos))) ≤ 2*incr)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects



Informal specification: compute the mean speed over the last 100ms.

1. postcondition for step function:

$$spd = (pos - pre^5(pos)) / 5*period$$

We would prefer :

$$spd = (pos - pre_{[100ms]}(pos)) / 100ms$$

2. A temporal contract

guarantee: **always(diff(spd,pre(spd)) ≤ δ)**

Requires **assumption:**

always (dist(pos),pre(pos))) ≤ d)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

MIMOSA a Lustre like programming and specification language

- Functional
- A notion of "mode" where the node function may be different in different modes
- A notion of "effect" :
  - A notion of (non controllable) environment for simulation and verification
  - "Interface to physics"
  - Boundary between software and HW/environment
  - Eases portability between HW platforms
- Contracts : assume / assert statements
  - Allows to easily express typical specification languages: automata, temporal logic ...
  - A MIMOSA node is a contract that can be executed
  - Allows to reuse/adapt state-of-art verification technologies

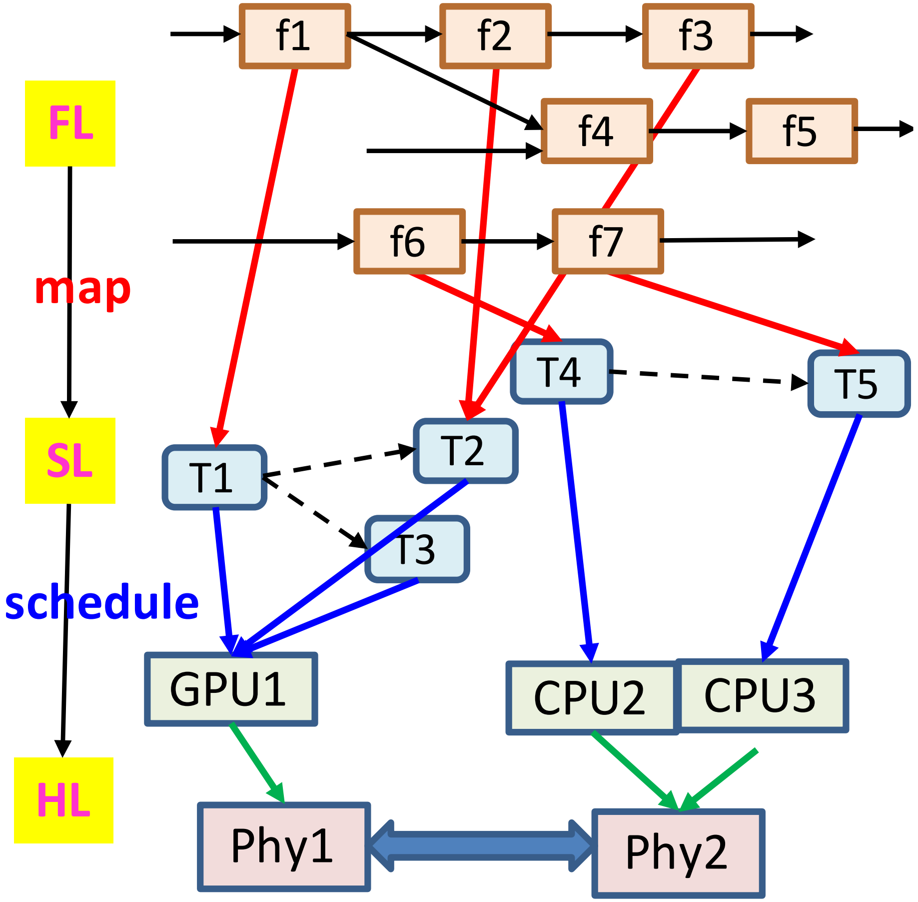
© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

MIMOS tools

Existing and under development:

Function layer:

- 2 simulation tools :
  - Java-based interface, functions in Java
  - web-interface, supports many progr languages, parallel simulation (integration of external simulators)
- “stream” consistency checker
- Compiler for MIMOSA language

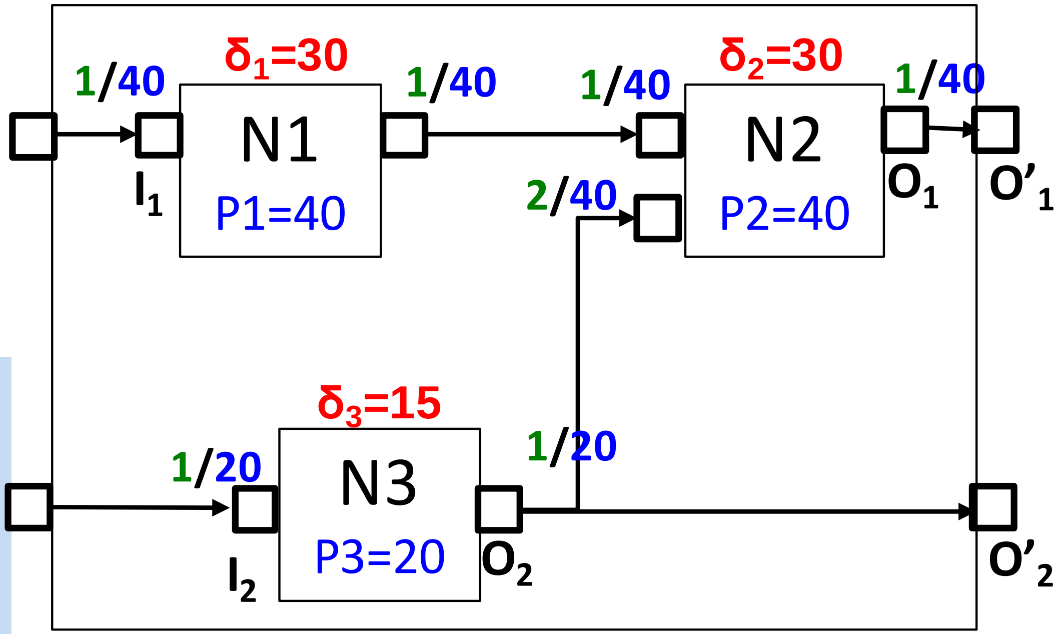


© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Software Layer:

- Some support for joint simulation of FL+SL
- Powerful methods under development for E2E-delay analysis, scheduling (main target: multicore), scheduling analysis

What about hierarchical nodes ?



Consider a (stream consistent) network NW of nodes.

Can we "draw a box around" and consider it as a node N (node = read/execute/write at port's deadline) ? Which produces the same (timed) output traces as the network ?

Some questions :

- What is period of N ?
- What is the memory of N ?
- What is δ of N (for each out port) ? δ > P possible ?
- Which read/write policies of N ?

What about refinement of N ?

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Hierarchical node (for the example)

Hypotheses: harmonic and aligned periods of NW

- Period P_N: 20, i.e. min(P_i)
- Memory M_N : X_i M_i x X $FIFOS_{NW}$.
- Delay δ_N: $\delta_{O2'}$ =15 = e2e(I_2,O_2),
  $\delta_{O1'}$ =70 = e2e(I_1,I_2,O_1) (delay per output port/function)
- Read/write policies of ports(N) wrt P_N=20:
  r(I'_1) = r(I_1) = 0.5/20, r(I'_2) = r(I_2) = 1/20
  w(O'_1) = w(O_1) = 0.5/20, w(O'_2) = w(O_2) = 1/20

Discussion:

- Traces(N) = Traces(NW) which is what we need
- Nevertheless: δ_N> P_N t
  thus: need to distinguish "basic nodes" from "non basic nodes"
  must be decomposed/decomposable s.th. comp(ref(N))=N
  (remind: N is periodic, thus reentrant)

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects

Verimag UPPSALA UNIVERSITET

(M)ANY QUESTIONS ?

© 2023 Susanne Graf -- CUSTOMER and UPDATE projects