CTL-Property Transformations Along an Incremental Design Process

Particularization to a Pipeline Flow Architecture

Cécile Braunstein and Emmanuelle Encrenaz

University of Paris 6 (UPMC)

Computer science Laboratory (LIP6)

Embeded system on-chip department (ASIM)

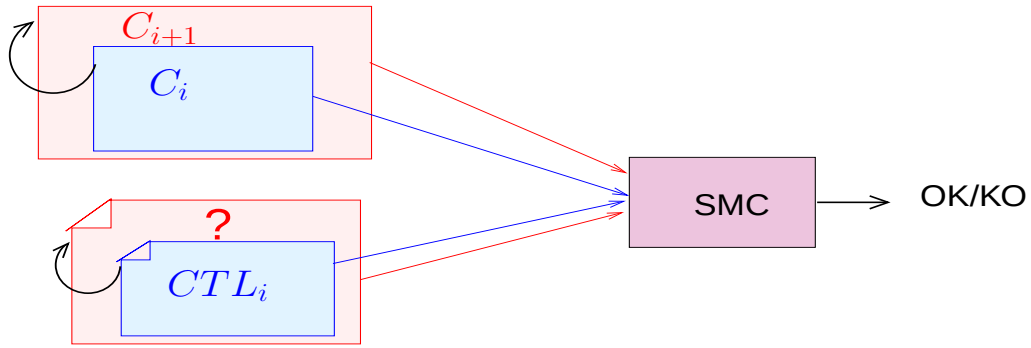
Outline

- The incremental design process
- CTL-property transformation
- Pipeline flow particularization
- Case study : VCI-PI protocol converter

Context

A design framework inspired by hardware designers :

- Successive additions of new behaviours
- Conservation of the existing behaviors



In a general case :

- ACTL Property preservation $C_{i+1} \Rightarrow C_i$ (Grumberg/Long 91)
- ECTL Property preservation $C_i \Rightarrow C_{i+1}$ (Loiseau and al. 95)

Incremental design :

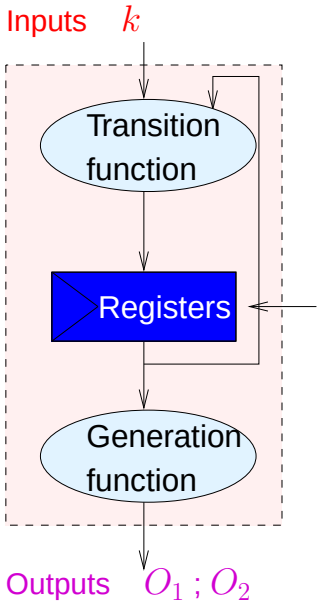
- CTL Property transformation $C_i \Leftrightarrow C_{i+1}$ (Braunstein/Encrenaz 04)

From Moore Machine to Kripke Structure

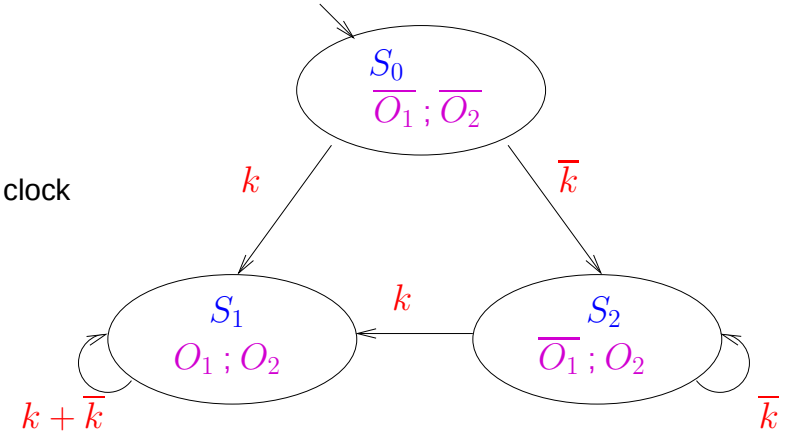
Moore Machine

The value of the registers defines the State.

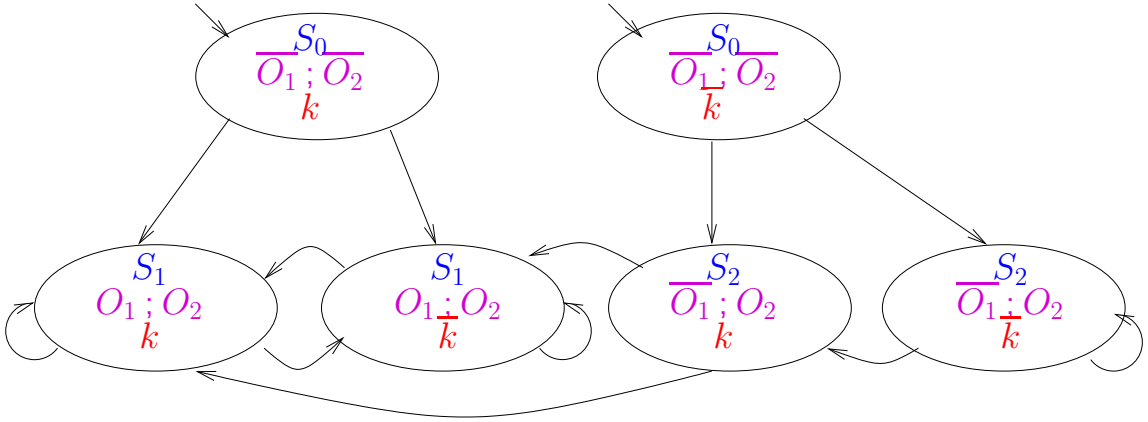
The output is a function of state only.



Complete deterministic and synchronous aut

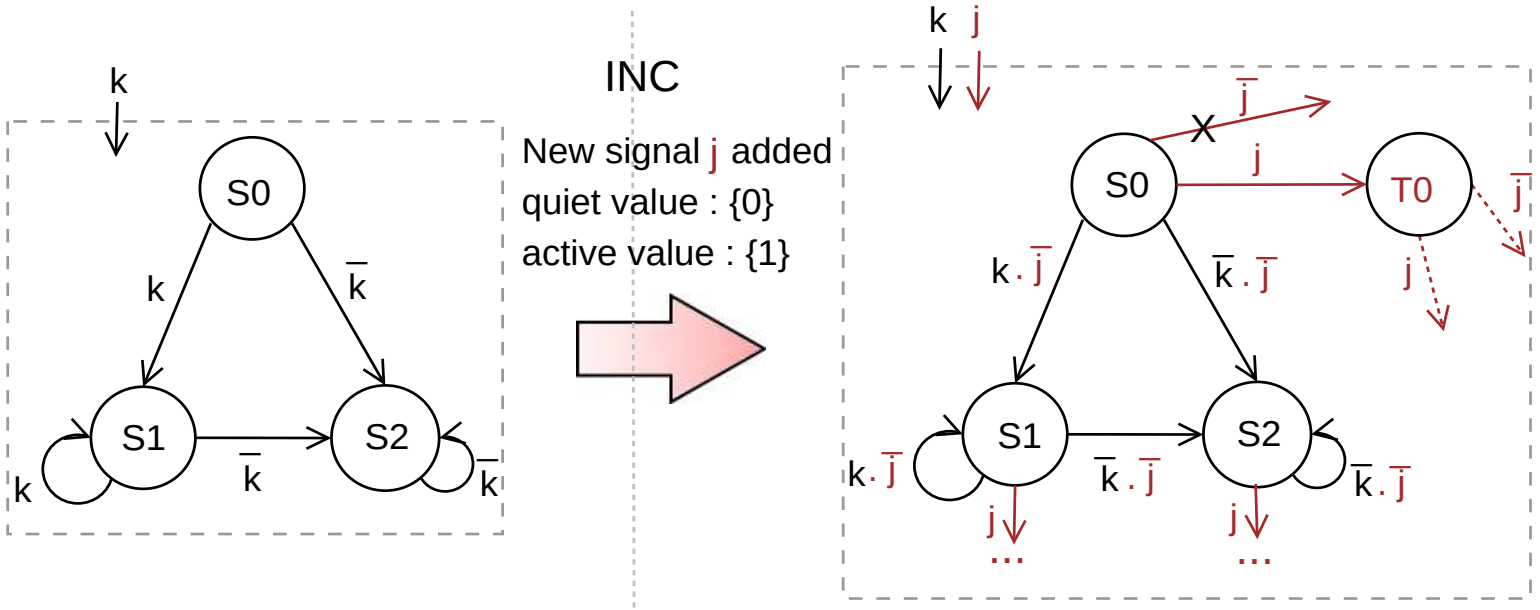


Kripke Structure



Increment Definition

- Increment INC is a set of new events at the interface
  - Each event has quiet values and active values
  - No new initial state, No behaviour overriding
  - C_{i+1} simulates C_i



Ci : complete deterministic Moore Machine

Ci+1 : complete deterministic Moore Machine

CTL-property transformations

Goal : Let be φ such that $C_i, s_0 \models \varphi$, what is φ' such that :

$$K(C_i), s_0 \models \varphi \Leftrightarrow K(C_{i+1}), s'_0 \models \varphi'$$

Principle : Reduction of the computational tree explored.

Example :

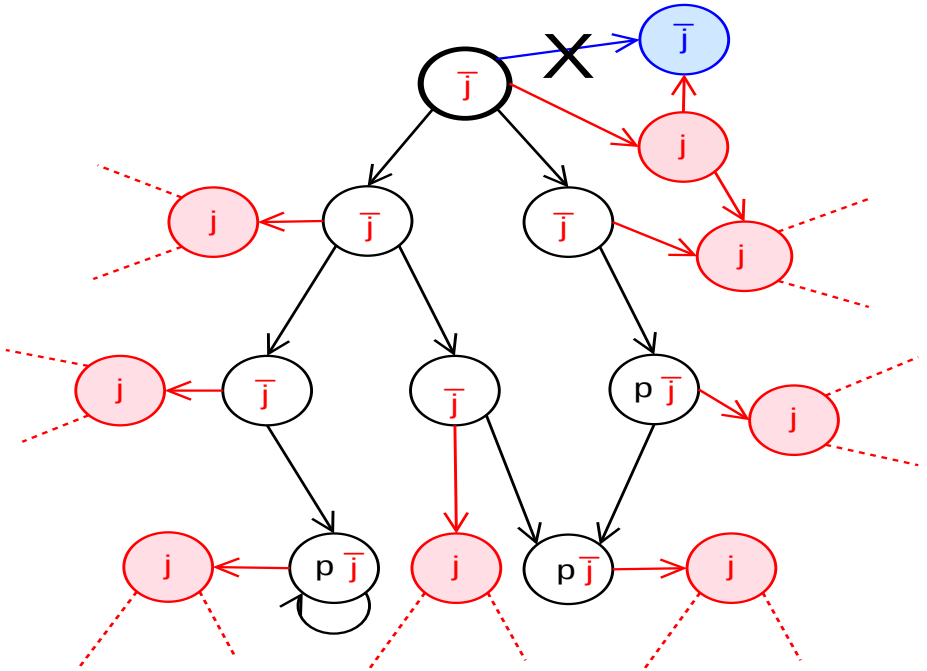
$K(C_i) \models AFp$

Increment : j

Quiet value : 0

Active value : 1

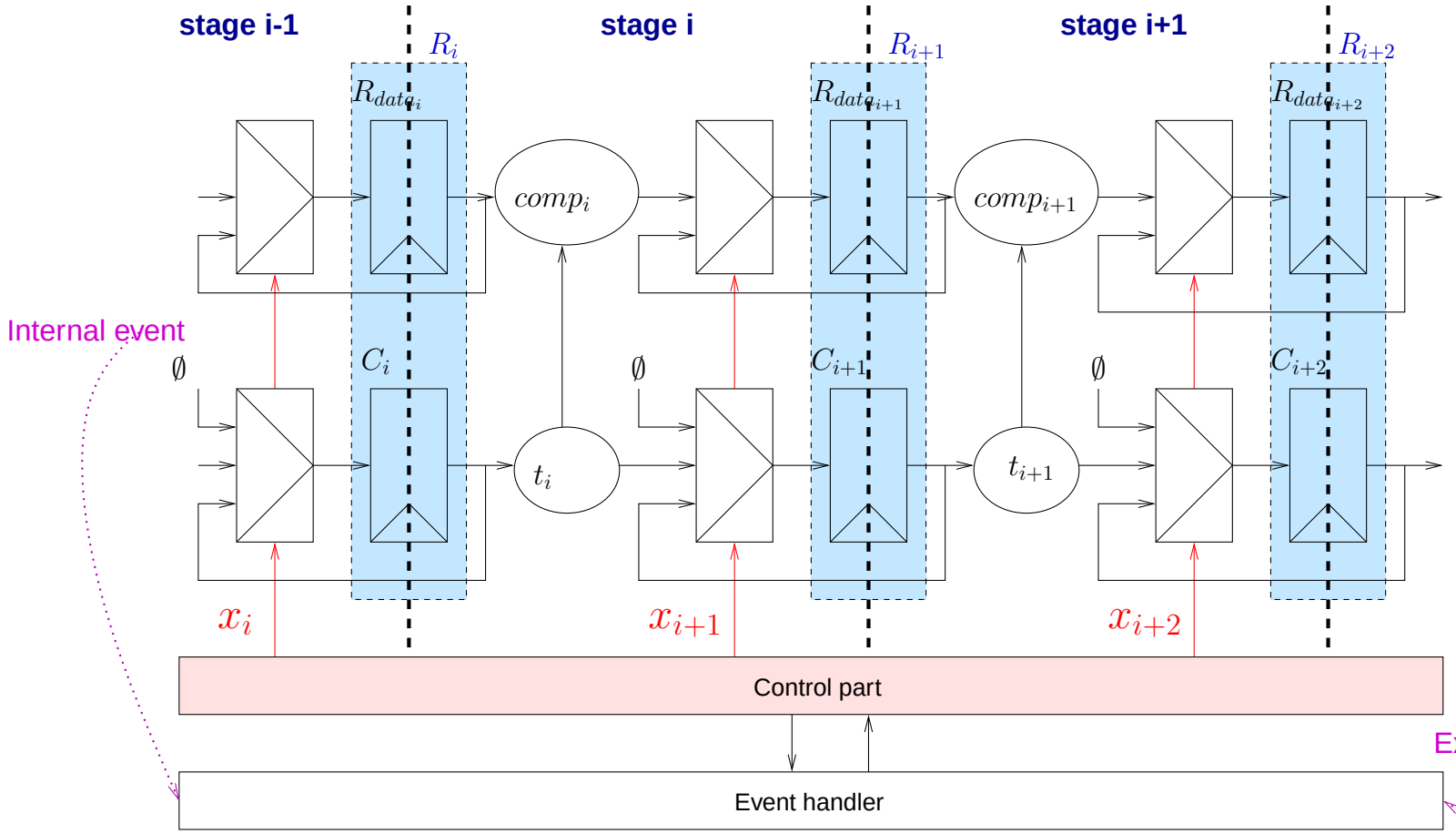
$K(C_{i+1}) \models AF(p \vee j)$



Results

- Transformation rules [accepted in STTT 2005]
  - All CTL operators are transformable (equivalence)
  - All CTL formulae are transformable
  - Transformed CTL formulae have same complexity as the initial ones
- Application to a concrete component design (VCI-PI protocol converter)
- Proves the non-regression of the specification by construction
- But we do not take into account the added behaviours !
  - More assumptions about the design and the increments extend our results

Pipeline Architecture



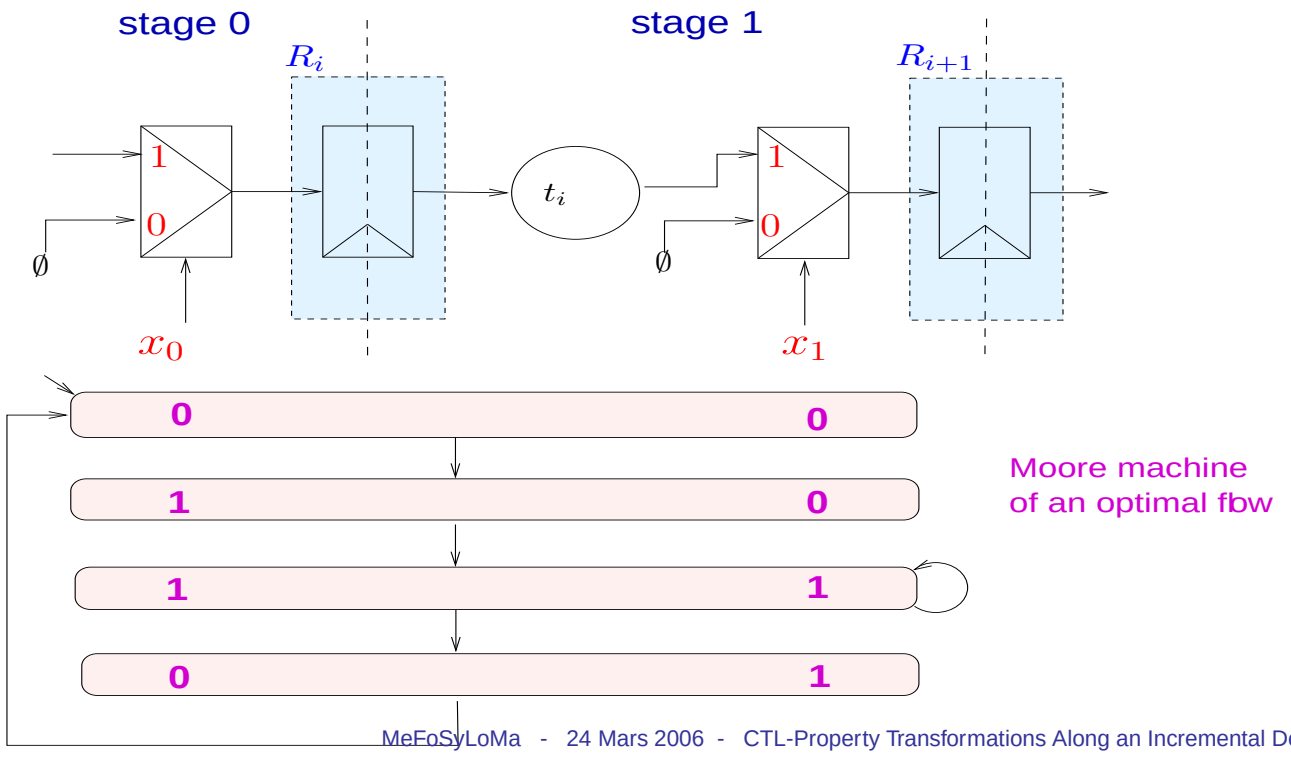
Control Part modeling

- Moore Machine : $(\#stage = n)$

  State : $(x_0 \ldots x_n)$; Transition : how the pipeline fills

- Basic case : optimal flow

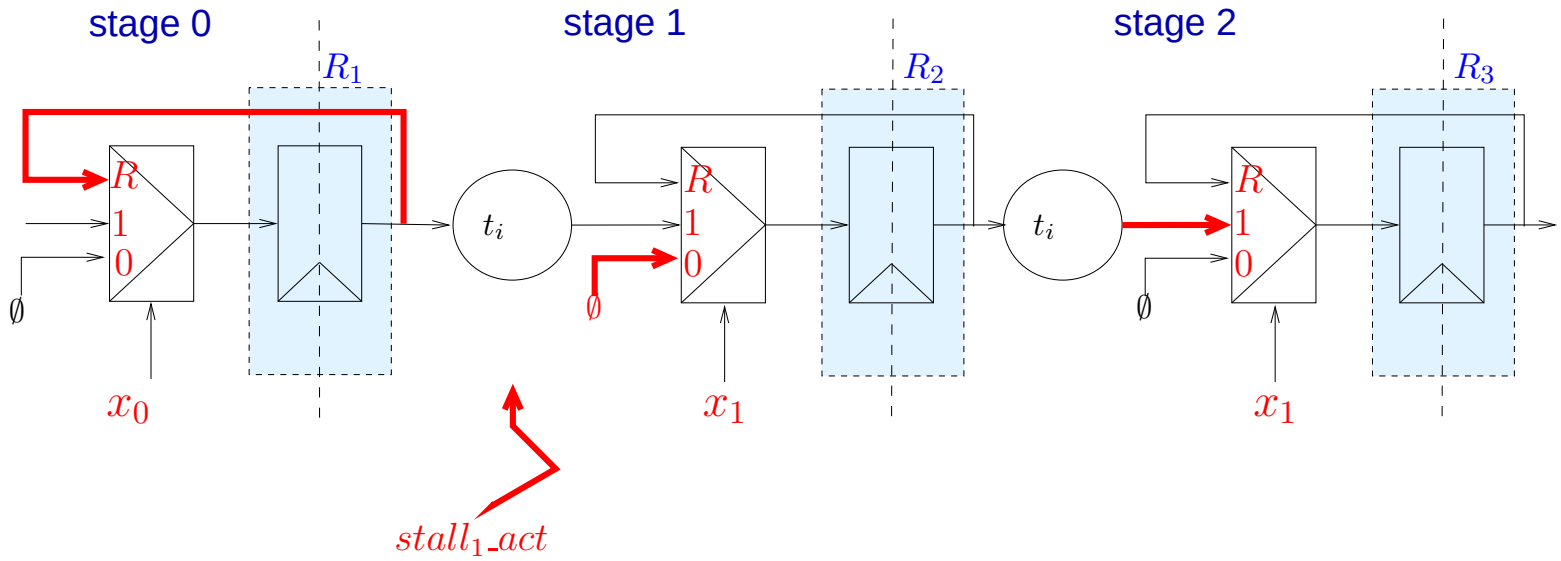
  No event disturbs the pipeline flow



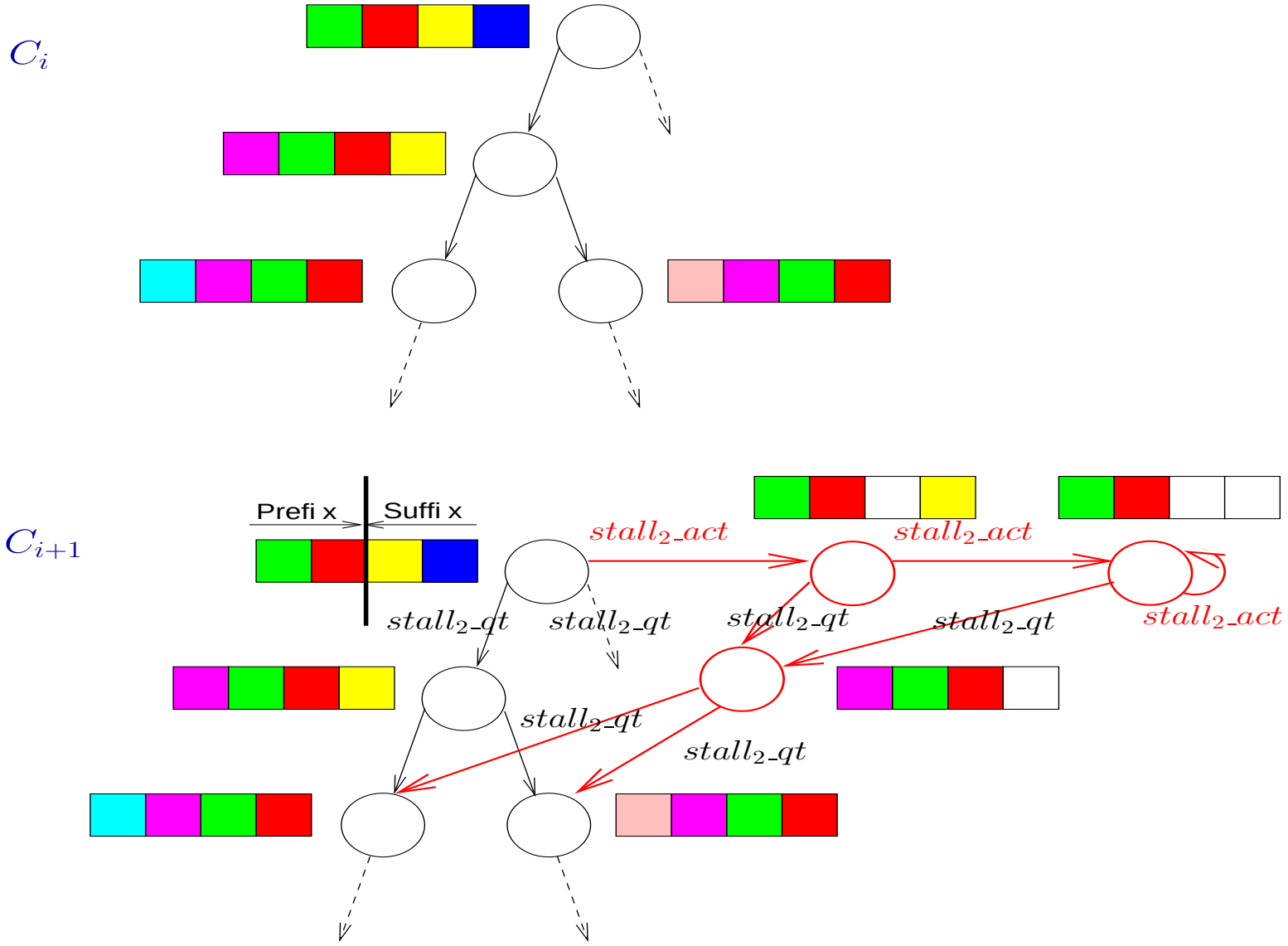
Stall Increment

Event $stall_k = \langle stall_k\_act, stall_k\_qt \rangle$ occurred at stage k

- All stages $i \leq k$ are stalled (stuttering progression of Prefi x)
- Stage $k+1$ executes an empty operation
- All stages $> k$ progress (normal progression of Suffi x)



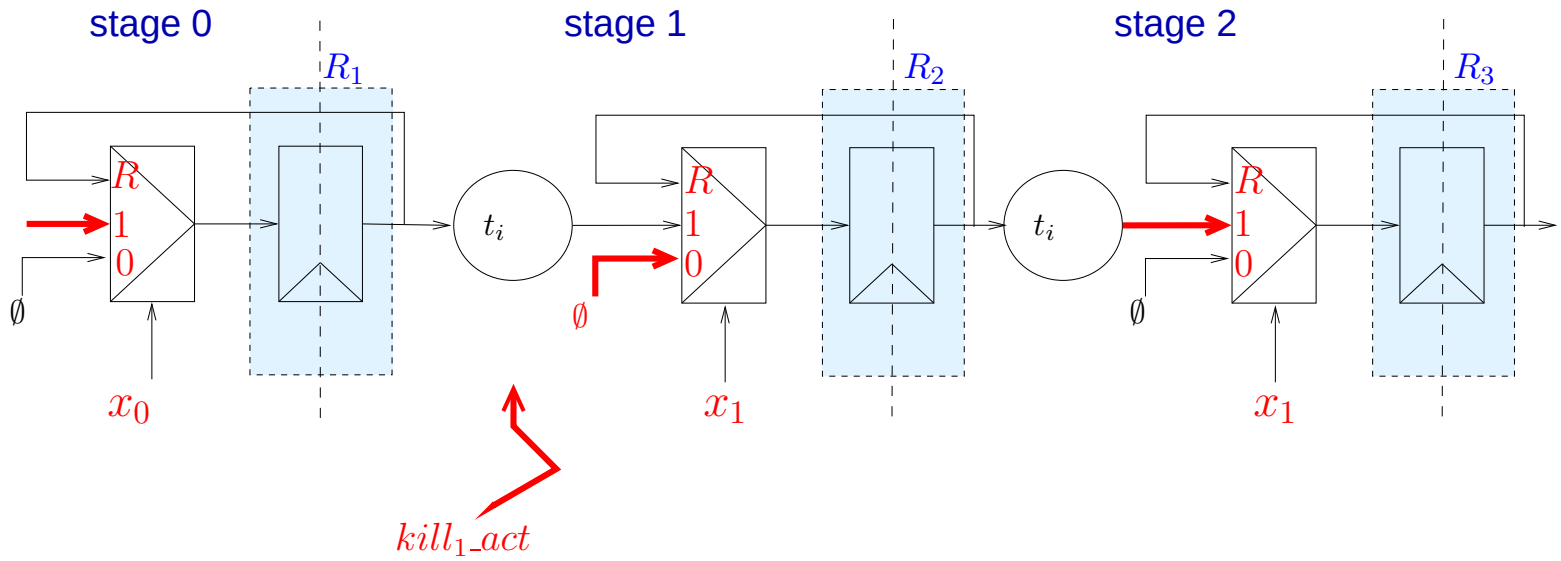
Stall Increment - Example



Kill increment

Event $kill_k = \langle kill_k\_act, kill_k\_qt \rangle$ occurred at stage k

- Treatment at stage k is destroyed
- The pipeline flow is not disrupted



Consequence on CTL Specification

Machine obtained by stall increments :

- Weak bisimulation of the prefix of states
- Stuttering progression of the prefix of states
- Normal progression of the suffix of states
  - Preservation of CTL\X formulae related to the outer part
  - Preservation of CTL\X formulae with one stage at a time
  - Transformation of CTL\X formulae with multiple stages

Machine obtained by kill increments :

  - Transformation of some CTL formulae related to the outer part

VCI-PI wrapper

- Virtual Component Interface design (VCI)
  - Standard communication protocol (OCB/VSIA 99)
  - General interface to connect system on-chip
  - Initiator/Target connections unidirectional
  - All requests from the initiator have to be acknowledge
- Peripheral interconnect Bus (PI-bus)
  - On-chip bus (ESPRIT OMI 1995)
  - Speed processor/memory bus
  - Pipeline behaviour

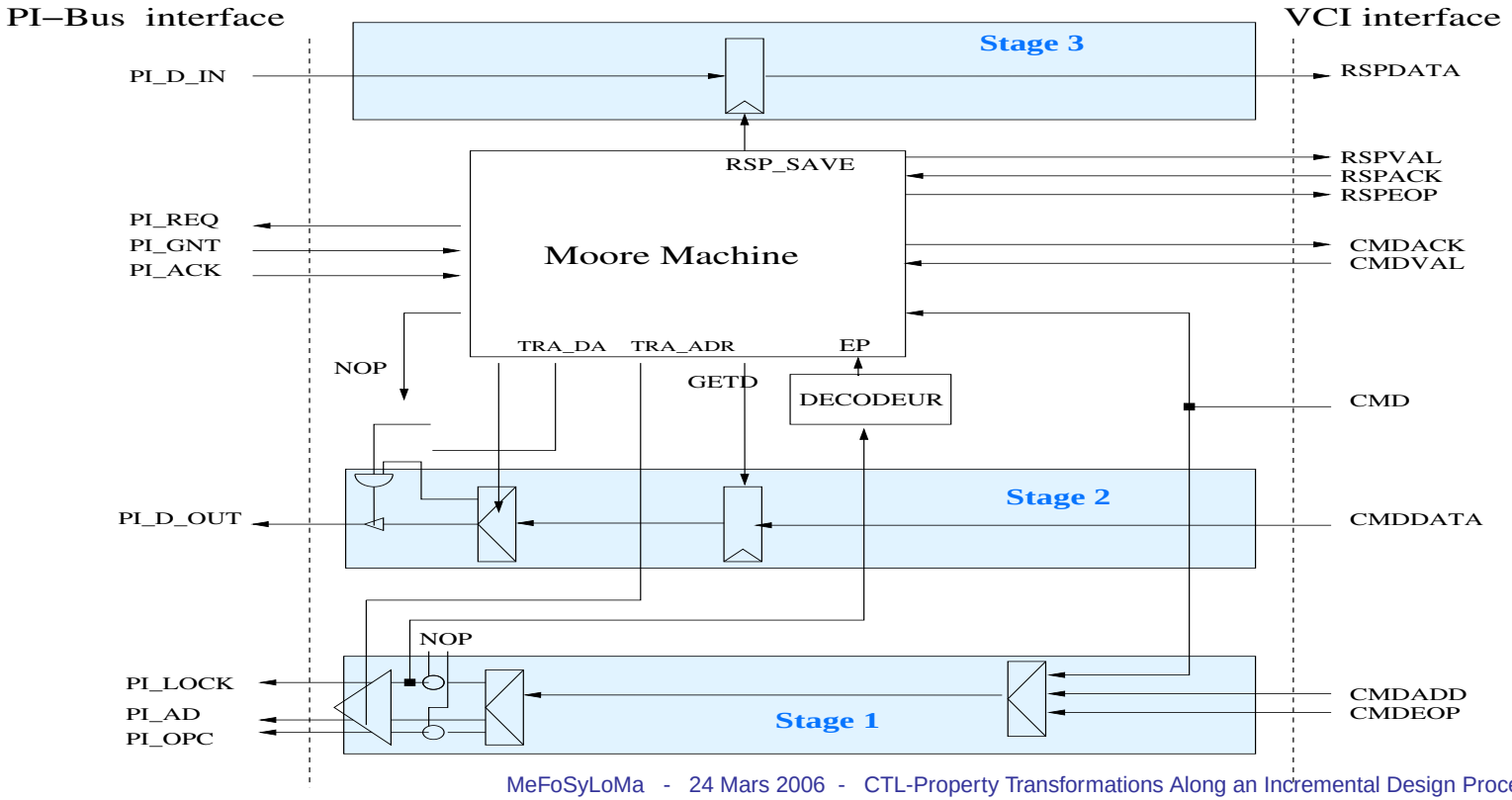
Incremental Design of VCI-PI Wrapper

3-stages pipeline

(stage 1) Accepting VCI request k.

(stage 2) Sending PI request ($k - 1^{th}$ VCI request).

(stage 3) Accepting PI response ($k - 2^{th}$ VCI request).



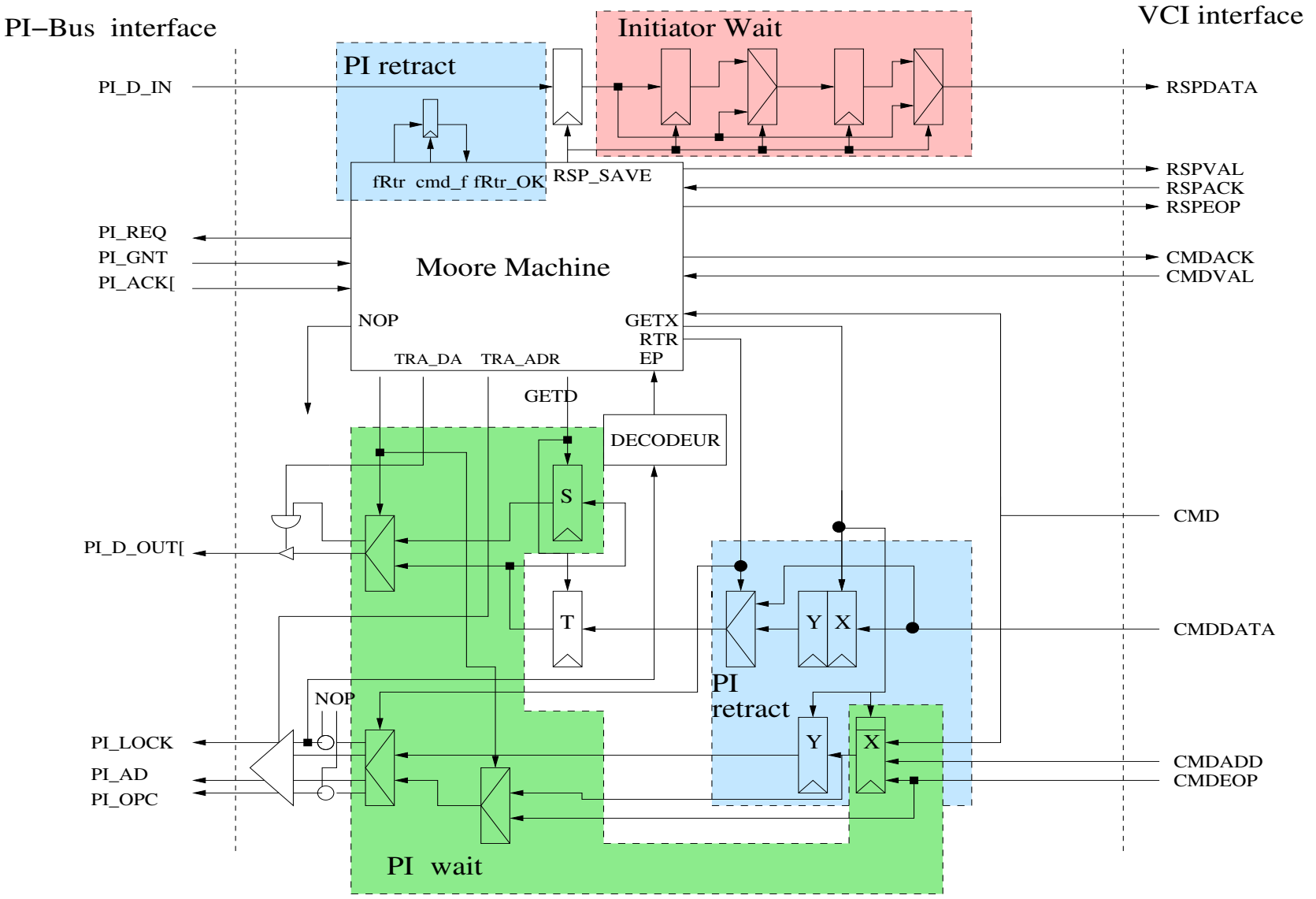
A set of 8 VCI-PI Wrapper

Type of event considered	Initiator is alway *ready* reset = {0} cmd_val={1}; rsp_ack={1}	Initiator may impose *wait states* reset = {0} cmd_val={0,1}; rsp_ack = {0,1}	Initiator may reset reset = {0,1} cmd_val={0,1}; rsp_ack = {0,1}
Target is always *ready* pi_rsp={RDY}	A	A'	A"
Target may impose *wait states* pi_rsp={RDY,WAIT}	B	B'	B"
Target may impose *retract* pi_rsp={RDY,WAIT,RTR}	C	C'	C"

80 CTL properties specify B (local and global)

By construction, they are part of the specifi cation for others wrappers

VCI-PI Wrapper C'



Conclusion

- Formalization of a framework to design component that insures non-regression
- Specification automatically derived from a simpler component
- Definition of a class of increments related to pipeline flow
- Used in the design of VCI-PI and VCI-AMBA protocol converters (Ossima Kéba 2002)
- Applicable for the design of systems with pipeline architecture

Transformations rules

$\Phi = p \quad \Leftrightarrow \Phi' = p$

$\Phi = \text{not } f \quad \Leftrightarrow \Phi' = \text{not } f'$

$\Phi = EXf \quad \Leftrightarrow \Phi' = e\_qt \text{ and } EXf'$

$\Phi = EFf \quad \Leftrightarrow \Phi' = E(e\_qt \text{ U } f')$

$\Phi = EGf \quad \Leftrightarrow \Phi' = EG(\ e\_qt \text{ and } f')$

$\Phi = Ef \text{ U } g \quad \Leftrightarrow \Phi' = E(\ (e\_qt \text{ and } f') \text{ U } g')$

$\Phi = Ef \text{ W } g \quad \Leftrightarrow \Phi' = E(\ (e\_qt \text{ and } f') \text{ W } g')$

$\Phi = AXf \quad \Leftrightarrow \Phi' = e\_qt \Rightarrow AXf'$

$\Phi = AFf \quad \Leftrightarrow \Phi' = AF(e\_act \text{ or } f')$

$\Phi = Af \text{ U } g \quad \Leftrightarrow \Phi' = A(\ f' \text{ U } (\ (e\_act \text{ and } f') \text{ or } g'))$

$\Phi = AGf \quad \Leftrightarrow \Phi' = A(\ f' \text{ W } (e\_act \text{ and } f')$

$\Phi = Af \text{ W } g \quad \Leftrightarrow \Phi' = A(f' \text{ W } ((e\_act \text{ and } f') \text{ or } g'))$