CTL-property transformations along an incremental design process

Cécile Braunstein, Emmanuelle Encrenaz

CTL-property transformations along an incremental design process

Cécile Braunstein and Emmanuelle Encrenaz

LIP6, Université Pierre et Marie Curie, Paris, France
<cecile.braunstein> <emmanuelle.encrenaz> @lip6.fr

Abstract. This paper formalizes an incremental approach to design VCI to PI protocol converters (VCI-PI wrappers) and presents a hierarchy of wrappers ranking from the simplest one up to the most complex one. In order to formally verify the correctness of a wrapper, a set of CTL properties is assigned to it. The purpose of the paper is to explore how, given a property that is true in a simple model, a new property, satisfied in a more complex model, can be derived from the first one, and reciprocaly. We propose some transformation rules to build new properties satisfied on more complex models. The properties transformation have been automated and applied in the context of non-regression analysis of VCI-PI wrappers.

Topics. System Design and Verification, Simulation Relation, Computational Tree Logic.

1 Introduction

The present paper stems from the observation of the way some hardware designers achieve the design of the device they have to build. They adopt an *incremental strategy* : after having defined the information flows of the design, the rough structure of the data-path and the control part, they proceed from the implementation of the simplest cases up to the most complex ones, by *adding* new functionalities to already existing ones, building a more and more complex device.

This design process is different from those applying a refining strategy as [4]. In refining strategies, the global information flows are initially defined, and all cases, according to their complexity, are obtained by incremental refinements of the initial model. The strength of these approches resides in the preservation of global properties along the refinement process : if a property is true on a given model, then, if the refinement conforms to a well-definition, the refined model will preserve the initial property. But this strategy excludes the addition of new functionalities during the design process : a refinement is a *specialization* of a pre-defined set of behaviours.

The incremental approach lacks of these good verification principles : once a behaviour is added to an initial model, a global property which was true on the initial model may be wrong on the extended one, due to this new behaviour.

In the incremental design process, a different set of property has to be assigned

to the component for each step of the design. Moreover, global properties (about the component plugged in a complex environment made of other components) have to be re-adapted for each step of the design. Often, the verification is performed by simulation of test cases, that have to produce the same results on the initial and extended models.

Alternative hardware verification methods include symbolic model checking [1] that is efficiently used to verify medium size hardware components (up to 10 Kgates). Classical results, concerning CTL property preservation, state that any CTL property that is true on an initial model may be either true or false on the extended one, (depending on the added behaviour), thus one cannot say much if one tries to preserve any property.

Focusing on the verification of CTL properties, we are interested in exploring the links between properties that are true on an initial model and these that are true on the extended one. If we can perform this, we can insure the extended model does not introduce regression with respect to the initial one.

We show that such a transformation is possible once the set of admissible extensions between an initial model and the extended one has been defined. The transformed CTL formulae obtained, applied to the extended model, restrict the verification state-space traversals to a sub-graph isomorphic to the one derived from the initial model.

The paper is organized as follows : In a first section we present how a new behaviour is added to an existing model. The difference between the initial and the extended model is named the increment. The increment is formally described. It is shown that given an initial model and an increment, the derived extended model simulates the initial one. The second section presents the Kripke structures derived from an initial model and the extended model (according to the increment), and characterizes the main properties of the latest, mainly that it includes the formest, tagged with a particular value. From these considerations, the subsequent section presents a set of transformations of CTL formulae, restricting the verification of CTL formulae in the Kripke structure of the extended model to the Kripke structure of the initial one it includes. Sketches of the proofs are given in this section while the complete proofs are given in annex. Then we briefly present the way these transformations were applied during the incremental design process of protocol converters (between VCI and Pi-bus), and conclude sketching some directions for future work.

2 Increment formalization

In this section, we formalize the component being designed and the increment. Then we characterize the extended component.

A component is viewed as a control part driving a data-path. Its state-space is modelled by a deterministic synchonous Moore Machine. The component presents an interface composed of typed signals.

2.1 Definitions of a signal and a configuration

Definition 1. *Each signal is defined by a name s and a definition domain $Dom(s)$.*

Definition 2. *Let E be a set of signals. A configuration $c(E)$ is a vector that associates to each signal in E one value of its definition domain. The set of all configurations $c(E)$ is named $C(E)$.*

2.2 Definition of a component

Definition 3. *A component $W_i =< S_i, I_i, O_i, T_i, L_i, s_i >$ is described as a deterministic Moore Machine. Its definition follows :*

S_i *: Finite set of states*
I_i *: Finite set of input signals with their finite definition domain. :* $\{(sig_{in}, Dom_i(sig_{in}))\}$
O_i *: Finite set of output signals with their finite definition domain. :* $\{(sig_{out}, Dom_i(sig_{out}))\}$
T_i *: Finite set of transitions* $\subset S_i \times C(I_i) \times S_i$
L_i *: Vector of generation functions* $= \{l_0, \ldots, l_{|O|-1}\}$ *each function defining the value of exactly one output signal in each state; for all output signal* o_j $0 \leq j \leq |O_i| - 1$ *we have* $l_j : S_i \rightarrow Dom(o_j)$,
$s_i \in S_i$ *: the initial state*

Remark 1. Applying the vector of generation functions to a given state of S_i produces a configuration $c(O_i)$.

2.3 Increment

An increment is the set of modifications applied to a component's architecture in order to build a more complex one. It reflects the carrying out of a new event at the component's interface. The architecture of W_i does not consider the occurrence of this new event, while the architecture W_{i+1} does.

An increment is a new event (or a set of new events), taken into account in a subset of states only, introducing new behaviors.
A new event can occur of two different manners :

- Either the definition domain of an existing signal is extended. The interfaces of the component are fixed, but the incremental design process takes into account values of these interfaces that were not previously considered.
- Or an new signal is added (with a definition domain). This is the case of an increasing complexity of the data-path of the component.

It may be considered in a subset of states only :

- If the signal previously existed, it appeared into the input configuration labelling some transitions exiting some states (at least one). The set of transitions leaving from these states has to be extended with the new potential value of the input signal.

– If the signal is a fresh one, its new value is supposed to be pertinent in all states, unless the user has tagged the states where this signal has to be taken into account.

It introduces new transitions and states :

– The new behaviors are represented as new transitions and states in the previous Moore Machine.
– Completeness and determinism of the Moore Machine are preserved.

It drives new actions :

– Either an existing output signal has its domain extended, this new value appears in the fresh states.
– Or a new output signal is created, it owns a quiet value (corresponding to a no-action) and an action value.
– The new signal is added in the states existing in the elder Moore Machine, driving its quiet value.
– The new signal appears in fresh states, driving either its quiet or action value.

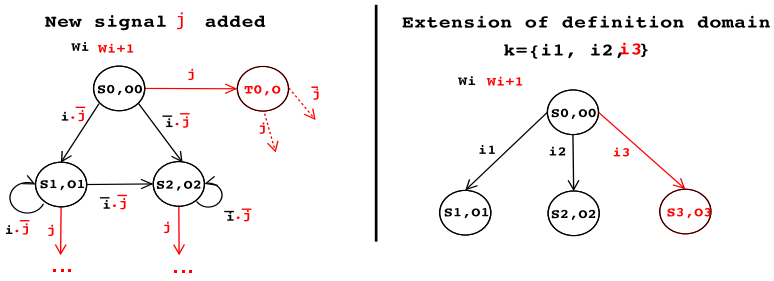

Fig. 1. Increment examples

Figure 1 presents two admissible increment. On the left, a new signal j is added to an initial component W_i. W_i is represented in with black lines and the increment applied is shown in grey. In this example, the quiet value of j equals 0. On the right, the increment consists in extending the definition domain of an existing signal k whose values were i_1 or i_2 in W_i. In W_{i+1} i_1 and i_2 are quiet values of signal k while i_3 is its active value.

Definition 4. *The increment from a component W_i to a component W_{i+1} is a 6-uplet $INC =< I_+, \Sigma_{in}, R_+, O_+, \Sigma_{out}, \Sigma_+ >$*

I_+ : The set of -possibly new- input signals and their -eventually- extended definition domain = {(e,Dom(e))} with:

- *if* $e \in I_i, Dom_{i+1}(e) = Dom_i(e) \cup \{newvalues\}$
- *if* $e \notin I_i, Dom_{i+1}(e) = \{\texttt{val\_qt}, \texttt{val\_act}\}$

$\Sigma_{in} \subseteq S_i$ *: The set of states modified by* I_+*. By default* $\Sigma_{in} = S_i$*.*
R_+ *: The set of new transitions.*
O_+ *: The set of new output signals and their -potentially- new definition domain = (o,Dom(o)) with :*

- *if* $o \in O_i$ $Dom_{i+1}(o) = Dom_i(o) \cup \{new\ values\}$
- *if* $o \notin O_i$ $Dom_{i+1}(o) = \{$ `val_qt` , `val_act` $\}$

Σ_{out} *: The set of states (belonging to* S_i *or successor of a state of* Σ_{in}*) modified by* O_+*.*
Σ_+ *: The set of new reachable states.*

We can now characterize the structure of W_{i+1} with respect to W_i and a given increment.

2.4 The component W_{i+1}

Definition 5. $W_{i+1} =< S_{i+1}, I_{i+1}, O_{i+1}, T_{i+1}, L_{i+1}, s_{i+1} >$ *:*

$S_{i+1} = S_i \cup \Sigma_+$
$I_{i+1} = I_i \cup I_+$
$O_{i+1} = O_i \cup O_+$
$T_{i+1} \subset S_{i+1} \times C(I_{i+1}) \times S_{i+1}$
$L_{i+1} \supset L_i$ *: generation functions:* $S_{i+1} \rightarrow C(O_{i+1})$*.*
$s_{i+1} = s_i$ *: initial states.*

Characterization of the transitions of W_{i+1} . The transitions that were present in W_i still exist in W_{i+1} but the input configuration that labels each of these may be extended (due to an extension of the definition's domain of the incremented signal). This is formaly defined below :

- If the increment signal $e \in I_+$ is a fresh one, having a quiet value `val_qt` and an active value `val_act`:
  $\forall (s, c_a, s') \in T_i$, $\exists c_b \in C(I_{i+1})$ such that $(s, c_b, s') \in T_{i+1}$ and $c_b{=}c_a.(e =$ `val_qt`$)$ (dot means vector concatenation)
  and $\exists c_c \in C(I_{i+1})$ and $s'' \in S_{i+1}$ such that $(s, c_c, s'') \in T_{i+1}$ and $c_c{=}c_a.(e =$ `val_act`$)$
- If the incremental signal $e \in I_+$ already existed in W_i but has an extended definition's domain in W_{i+1} $(Dom_i(e) \subset Dom_{i+1}(e))$:
  $\forall (s, c_a, s') \in T_i$, $\exists c_b \in C(I_{i+1})$ such that $(s, c_b, s') \in T_{i+1}$ and $c_b{=}c_a$
  and $\exists c_c \in C(I_{i+1}) \setminus C(I_i)$ and $s'' \in S_{i+1}$ such that $(s, c_c, s'') \in T_{i+1}$ and $c_c{=}c_{a_{e \leftarrow Dom_{i+1}(e) \setminus Dom_i(e)}}$

Characterization of the generation functions L_{i+1} . The output signals that were present in W_i remain in W_{i+1}. Their generation function is extended to the new states in Σ_+ and the new input configurations due to I_+. The newly introduced output signals own their generation functions.

- If the output signal o_j is a fresh one, having a quiet value `val_qt` and an active value `val_act`. The generation function of this signal is extended.
  $\forall s \in S_i \cap S_{i+1}, \forall c \in C(I_i \cap I_{i+1}) : l_{j_{i+1}}(s,c) = \texttt{val\_qt}$
  $\forall s \in S_{i+1} \setminus S_i, \forall c \in C_{i+1}(I_{i+1}) : l_{j_{i+1}}(s,c) \in Dom_{i+1}(o_j)$
- If the output signal o_j already existed in W_i but has an extended definition's domain in W_{i+1} ($Dom_i(o_j) \subset Dom_{i+1}(o_j)$):
  $\forall s \in S_i \cap S_{i+1}, \forall c \in C(I_i \cap I_{i+1} : l_{j_{i+1}}(s,c) = l_{j_i}(s,c)$
  $\forall s \in S_{i+1} \setminus S_i, \forall cinC_{i+1}(I_{i+1}) : l_{j_{i+1}}(s,c) \in Dom_{i+1}(o_j)$

A component W_{i+1} obtained by applying an increment to a component W_i preserves all behaviours that were present in W_i, assuming that, in W_{i+1}, the new input signals are maintained to their quiet value.

Property 1. **The initial state in W_{i+1} simulates the initial state in W_i**
Sketch of the proof : We build ρ_W a binary relation between the states of two consecutive components W_i and W_{i+1}, such that : $\rho_W :\subseteq S_i \times S_{i+1}$ with :
$\forall s \in S_i$, let be s' the name of s in $S_{i+1}, (s,s') \in \rho_W$. By construction, ρ_W is a simulation relation.

3 From Moore machine to Kripke structure

The verification of CTL properties is defined on the Kripke structure derived from the initial Moore Machine describing the component W_i. We formally define the Kripke structure $K(W_i)$ obtained from the component W_i.

Definition 6. *A Kripke structure is a 5-tuple $\langle S, s_0, AP, \mathcal{L}, R\rangle$ where*

S is a finite set of states,
$s_0 \subseteq S$ is the set of initial states,
AP is a finite set of atomic propositions,
$\mathcal{L} = \{l_0, \ldots, l_{|AP|-1}\}$ is a vector of $\mid AP \mid$ functions, each function defining the value of exactly one atomic proposition; for all $0 \leq i \leq \mid AP \mid -1$ we have $l_i : S \rightarrow \mathbb{B}$; for all $s \in S$, we have that $l_i(s)$ is true iff the atomic proposition associated to l_i is true in s,
$R \subseteq S \times S$ is the transition relation.

Definition 7. Translating a FSM by Putting the Inputs in the Source State *Given a FSM of a component $W_i =< S_i, I_i, O_i, T_i, L_i, s_i >$, we deduce the Kripke structure $K(W_i) = \langle S_{K(W_i)}, s_{K(W_i),0}, AP_{K(W_i)}, \mathcal{L}_{K(W_i)}, R_{K(W_i)}\rangle$ where:*

$S_{K(W_i)} = S_i \times C(I_i)$,
$s_{K(W_i),0} = s_i \times C(I_i)$,

$AP_{K(W_i)} = I_i \cup O_i$,
$\mathcal{L}_{K(W_i)} = \{l_{O0}, \ldots, l_{O|O_i|-1}\} \cdot \{l_{I0}, \ldots, l_{I|I_i|-1}\}$;
for all $0 \leq j \leq \mid O_i \mid -1$, *we have* $l_{0j} : S_{K(W_i)} \rightarrow Dom_{W_i}(o_j)$;
for all $0 \leq j \leq \mid I_i \mid -1$, *we have* $l_{Ij} : S_{K(W_i)} \rightarrow Dom_{W_i}(s_j)$;
$R_{K(W_i)} \subseteq S_{K(W_i)} \times S_{K(W_i)}$ *and* $\forall\,(s, c_i) \in S_{K(W_i)}, \forall\,(s', c_i') \in S_{K(W_i)}$, *we have* $((s, c_i), (s', c_i')) \in R_{K(W_i)}$ *iff* $(s, c_i, s') \in R$.

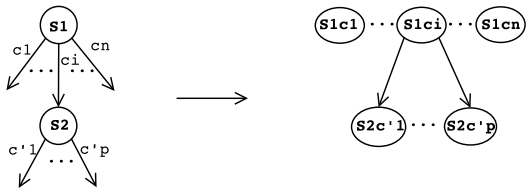

Fig. 2. Transformation of a Moore Machine into a Kripke structure

All the states s' in $K(W_i)$ obtained from the same state s in W_i are said to belong to the same brotherhood derived from s. The states belonging to a common brotherhood differ from the input configuration that labels them, and from their successor states.

Definition 8. Brotherhood *Let be a Kripke structure* $K(W_i)$ *obtained from a component* W_i, *we define the Brotherhood Relation* $B : S_{K(W_i)} \times S_{K(W_i)} \rightarrow \mathbb{B}$. *For all* $s = (s_1, c_1) \in S_{K(W_i)}$, *for all* $t = (t_1, c_2) \in S_{K(W_i)}$, *we have* $(s, t) \in B$ *iff* $s_1 = t_1$. *The* Brotherhood *is the function* $B_r : S_{K(W_i)} \rightarrow \mathcal{P}(S_{K(W_i)})$, *that, given a state* s *returns the maximal set of states* t *such that* $(s, t) \in B$.

Remark 2. The common part s of all states belonging to a Brotherhood $\{(s, c_1) \ldots (s, c_n)\} \in K(W_i)$ are said to be derived from s, that is a state in W_i.

Remark 3. The set of brotherhood in $K(W_i)$ is a partition of $S_{K(W_i)}$.

We are now interested in characterizing $\mathrm{K}(W_{i+1})$ with respect to $\mathrm{K}(W_i)$ and INC. $\mathrm{K}(W_{i+1})$ is obtained from the translation described in definition 7 of the component W_{i+1} obtained by applying the increment INC to the component W_i.

3.1 From $\mathbf{K}(W_i)$ to $\mathbf{K}(W_{i+1})$

We firstly define some relations between states of $\mathrm{K}(W_i)$ and $\mathrm{K}(W_{i+1})$, that expand the same state in W_i, but with an input configuration in $\mathrm{C}(I_i)$ or in $\mathrm{C}(I_{i+1})$. Diagram of figure 3 illustrates these relations between states in W_i, W_{i+1}, $\mathrm{K}(W_i)$ and $\mathrm{K}(W_{i+1})$.

Definition 9. Enrichment *For all state* $s_i = (s, c) \in K(W_i)$, *there exists* $s_i' = (s', c')$ *and* $s_i'' = (s'', c'') \in K(W_{i+1})$ *such that :*

- *if the increment is due to the extension of the definition domain of an imput signal* $e : s' = s, c' = c$ *and* $s'' = s, c'' = c_{|e \leftarrow Dom_{i+1}(e) \cap Dom_i(e)}$.
- *if the increment is due to a fresh signal* e *having a quiet value and an active value :* $s' = s, c' = c.(e = \texttt{val\_qt})$ *and* $s' = s, c' = c.(e = \texttt{val\_act})$.

In both cases, s'_i *is said to* enrich s_i *(with* $(e = \texttt{val\_qt})$ *in the second case).*

Definition 10. Incremental Brotherhood *Let be two Kripke structures* $K(W_i)$ *and* $K(W_{i+1})$ *obtained from an increment* INC*, we define the Incremental Brotherhood Relation* $B_{inc} : S_{K(W_i)} \times S_{K(W_{i+1})} \rightarrow \mathbb{B}$.
For all $s = (s_1, c_1) \in S_{K(W_i)}$*, for all* $t = (s'_1, c'_1) \in S_{K(W_{i+1})}$*, we have* $(s, t) \in B_{inc}$ *iff* $s_1 = s'_1$*. The* Incremental Brotherhood *is the function* $B_{r_{inc}} : S_{K(W_i)} \rightarrow \mathcal{P}(S_{K(W_i+1)})$*, that, given a state* s *returns the maximal set of states* t *such that* $(s, t) \in B_{inc}$.

We can describe the Kripke structure $\mathrm{K}(W_{i+1}) = < S_{K(W_{i+1})}, T_{K(W_{i+1})}, AP_{K(W_{i+1})}, L_{K(W_{i+1})}, s_{K(W_{i+1}),0} >$ derived from a component $W_i = < S_i, I_i, O_i, T_i, L_i, s_i >$, its derived Kripke structure $K(W_i) = < S_{K(W_i)}, T_{K(W_i)}, AP_{K(W_i)}, L_{K(W_i)}, s_{K(W_i),0} >$ and an increment $INC =< I_+, \Sigma_{in}, R_+, O_+, \Sigma_{out}, \Sigma_+ >$ and $I_+ = \{$ (e,`val_qt`, `val_act`)$\}$

$S_{K(W_{i+1})}$ **:** the finite set of states of $\mathrm{K}(W_{i+1})$ is composed of : - Each state s that was in $S_{K(W_i)}$ produces s' and s'' in $S_{K(W_{i+1})}$, such that s' is labelled with $e =$`val_qt` and s'' is labelled with $e =$`val_act`.
- The set of new states in W_{i+1} generates new brotherhoods in $K(W_{i+1})$ labelled with $\mathrm{C}(I_i \cup I_+)$

$T_{K(W_{i+1})}$ **:** the finite set of transitions of $\mathrm{K}(W_{i+1})$ is composed of : - Existing transitions in $\mathrm{K}(W_i)$ remain in $K(W_{i+1})$ and are extended to all successor states belonging to the incremental brotherhood of a successor in $K(W_i)$.
- For every new transitions in W_{i+1}, a new set of corresponding transitions in $K(W_{i+1})$ are added.

$AP_{K(W_{i+1})} \supset AP_{K(W_i)}$**:** the finite set of atomic propositions of $\mathrm{K}(W_{i+1})$

$L_{K(W_{i+1})}$ **:** the labelling function of $\mathrm{K}(W_{i+1})$. $L_{K(W_{i+1})}(s') = L_{K(W_i)}(s) \cup (e = val\_qt)$ if s' enriches s.

$s_{K(W_{i+1}),0}$ **:** the set of initial states corresponding to the $B_{r_{inc}}(s_i)$ for all $s_i \in s_{K(W_i),0}$.

Figure 3 illustrates the Brotherhood, incremental brotherhood, enrichment relations. Each state s in W_i induces a brotherhood $Br(s)$ in $K(W_i)$. The state s' that renames s in W_{i+1} induces an incremental brotherhood $Br_{inc}(s)$ in $K(W_{i+1})$ that extends $Br(s)$, since the input configuration part of a state in $K(W_{i+1})$ encompasses the one of a state in $K(W_i)$. the state sjk in $\mathrm{K}(W_{i+1})$ enriches the state sk in $\mathrm{K}(W_i)$.

3.2 Properties of $K(W_{i+1})$

By construction, the tree of behaviours of $K(W_i)$ is preserved in $K(W_{i+1})$, labelled with the quiet value of the increment signal. This preservation property

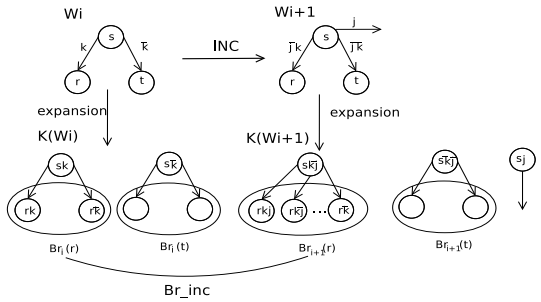

Fig. 3. Incremental Brotherhood

can be expressed as the existence of a simulation relation between the states of the Kripke structures. In fact, the enrichment relation captures more precisely the fact that the behaviours of the elder component are present in the newer one, but tagged with the increment signal assigned to its quiet value.

Property 2. **Each initial state of** $K(W_{i+1})$ **that enriches the initial state of** $K(W_i)$ **with** $(e = val\_qt)$ **simulates the latest.**
sketch of the proof : We define $\rho_{K_W} \subseteq \mathrm{K}(W_i) \times \mathrm{K}(W_{i+1})$, such that $\forall s \in S_{K(W_i)}$, $c \in C(I_i)$ and $\forall s' \in S_{K((W_{i+1})}$, $c' \in C(I_{i+1})$, if $s = s'(s' \in S_{K(W_i)} \cap S_{K(W_{i+1})})$ and if $c' = c$ or $c = c.(e = val\_qt)$ then ρ_{K_W} is a simulation relation.

Remark 4. From above, if s' enriches s with (e = val_qt) $\Rightarrow$ s' simulates s and s' simulates $s \not\Rightarrow s'$ enriches s

Corollary 1. *If there exists some infinite path in* $K(W_i)$*, then there exists some infinite path in* $K(W_{i+1})$ *along which the increment signal e has always its quiet value. Let be* $\sigma = s_0 \dots s_n \dots$ *in K(*W_i*),* $\exists \sigma' = s'_0 \dots s'_n \dots$ *in K(*W_i *such that all* s'_i *enriches* s_i *with* $(e =$ `val_qt`*).*

Corollary 2. $K(W_i)$ *is the maximal sub-graph in K(*W_{i+1}*), reachable from* s'_0*,(that enriches* s_0 *(in* $K(W_i)$*) with* $(e =$`val_qt` *) when e remains to the quiet value.*

Corollary 3. *The states in* $K(W_{i+1})$ *belonging to the brotherhood of a new state (a state added by the increment, i.e.* $\in \Sigma_+$*) are only reachable from the initial state* $s'_{K(W_{i+1}),0}$ *that enriches* $s_{K(W_i),0}$ *by a path along which* at least *one state is labelled by* $(e \neq val\_qt)$*.*

Corollary 4. *Let be* s' *that enriches* s *with* $(e =$ `val_qt` *), then for all* $t' \in K(W_{i+1})$ *such that* $s' \rightarrow t'$*, there exists* $t \in K(W_i)$ *such that* $t' \in B_{r_{inc}}(t)$ *and* $s \rightarrow t$*.*

Proof :

1. By induction on the length of σ.

2. By construction of K(W_{i+1}), we have $s'_{K(W_{i+1}),0}$ enriches $s_{K(W_i),0} \Rightarrow s'_{K(W_{i+1}),0}$ simulates $s'_{K(W_i),0}$ and for each state r' reachable from $s'_{K(W_{i+1}),0}$ that enriches a state r in K(W_i), such that r' $\rightarrow$ t' and t' does not enrich any state in $K(W_i)$, thus has no equivalent state in K(W_i), then t' $\models (e \neq val\_qt)$.
3. Let be $s' \in K(W_{i+1})$ that simulates $s \in K(W_i)$. Let e be a state in $K(W_{i+1})$ belonging to the brotherhood of a new state *new_s* added by the increment INC, and such that $s' \rightarrow new\_s$. There is no transition from s to e in $K(W_i)$ since there is no $B_r(e)$ in $K(W_i)$, hence, from $s'(\models (e =$ `val_qt`$))$, there is no transition towards e due to the corollary 2. On the opposite, from a state $s'' \in K(W_{i+1})$ belonging to the brotherhood of s' and such that $s'' \models (e \neq$ `val_qt`$)$,there exists a transition to e.
4. Directly from corollary 2.

Hence, K(W_{i+1}) includes K(W_i) and K(W_i) can be detected in K(W_{i+1}) since it is the maximal connected sub-graph tagged with e = `val_qt`. This is captured by the enrichment relation, that is a simulation. We now use this particularity to establish links between CTL properties verified on K(W_i) and others verified on K(W_{i+1}).

4 CTL-property transformations

[5] and [3] have stated some CTL property-preservation results between two Kripke structures ordered by any simulation relation. We recall their results in our particular context.

4.1 Preservation CTL formulae

Definition 11. Preservation of ECTL formulae from K(W_i) to K(W_{i+1}) [5] *Let be s a state of K(W_i) and s' a state of K(W_{i+1}) such that s' simulates s. Let Φ be an ECTL formulae : K(W_i),s $\models \Phi \Rightarrow$ K(W_{i+1}),s' $\models \Phi$.*

Definition 12. Preservation of ACTL formulae from K(W_{i+1}) to K(W_i)[3] *Let be s a state of K(W_i) and s' a state of K(W_{i+1}) such that s' simulates s. Let Φ be a ACTL formulae : K(W_{i+1}),s' $\models \Phi \Rightarrow$ K(W_i),s $\models \Phi$.*

4.2 Transformation of CTL formulae

Given a CTL formula Φ , we are going to set out the transformations rules to transform Φ in K(W_i),$s_{K(W_i),0}$ (shortly named s_0) into Φ' in K(W_{i+1}),$s'_{K(W_{i+1}),0}$ (shortly named s'_0) when s'_0 enriches s_0.

Theorem 1. *Let be $s \in S_{K(W_i)}$ and s' $\in S_{K(W_{i+1})}$ such that s' enriches s with (e = val_qt), let be p and q atomic propositions in $AP_{K(W_i)}$.*

K(W_i),s $\models p \Leftrightarrow$ K(W_{i+1}),s' $\models p$.
K(W_i),s $\models EXp \Leftrightarrow$ K(W_{i+1}),s' $\models (e = val\_qt) \Rightarrow EXp$.

$K(W_i),s \models EFp \Leftrightarrow K(W_{i+1}),s' \models E(\ (e = val\_qt)\ U\ p)$.
$K(W_i),s \models EGp \Leftrightarrow K(W_{i+1}),s' \models EG(\ (e = val\_qt) \wedge p)$.
$K(W_i),s \models EpUq \Leftrightarrow K(W_{i+1}),s' \models E(\ ((e = val\_qt) \wedge p)\ U\ q)$.
$K(W_i),s \models AXp \Leftrightarrow K(W_{i+1}),s' \models (e = val\_qt) \Rightarrow AXp$.
$K(W_i),s \models AFp \Leftrightarrow K(W_{i+1}),s' \models AF((e \neq val\_qt) \vee p)$.
$K(W_i),s \models ApUq \Leftrightarrow K(W_{i+1}),s' \models A(\ ((e = val\_qt) \wedge p)\ U\ (\ (e \neq val\_qt) \vee q))$.
$K(W_i),s \models AGp \Leftrightarrow K(W_{i+1}),s' \models A(\ ((e = val\_qt) \wedge p)\ W\ (e \neq val\_qt)\ )$.
$K(W_i),s \models ApW^1q \Leftrightarrow K(W_{i+1}),s' \models A(p\ W\ (q \vee (e \neq val\_qt)))$.

Sketch of proof : The transformations are based on the reduction of the computational tree explored in $K(W_{i+1})$ to the sub-tree along which the new value of the input signal is not considered. By corollary 2, this sub-graph represents $K(W_i)$. The transformation is proven for each CTL operator by including the $(e = val\_qt)$ constraint in its definition. The detailled proof of each case is given in annex.

Theorem 2. *For any CTL formulae Φ χ and Ψ (with all the atomic proposition in $AP_{K(W_i)}$), $K(W_i),s \models \Phi \Leftrightarrow K(W_{i+1}),s' \models \Phi'$, where Φ' is the formulae obtained by recusively applying the following transformations :*

$K(W_i),s \models EX\,\Psi \Leftrightarrow K(W_{i+1}),s' \models (e = val\_qt) \Rightarrow EX\Psi'$.
$K(W_i),s \models EF\Psi \Leftrightarrow K(W_{i+1}),s' \models E(\ (e = val\_qt)\ U\ \Psi')$.
$K(W_i),s \models EG\Psi \Leftrightarrow K(W_{i+1}),s' \models EG(\ (e = val\_qt) \wedge \Psi')$.
$K(W_i),s \models E\Psi U\chi \Leftrightarrow K(W_{i+1}),s' \models E(\ ((e = val\_qt) \wedge \Psi')\ U\ \chi')$.
$K(W_i),s \models AX\Psi \Leftrightarrow K(W_{i+1}),s' \models (e = val\_qt) \Rightarrow AX\Psi'$.
$K(W_i),s \models AF\Psi \Leftrightarrow K(W_{i+1}),s' \models AF((e \neq val\_qt) \vee \Psi')$.
$K(W_i),s \models A\Psi U\chi \Leftrightarrow K(W_{i+1}),s' \models A(\ ((e = val\_qt) \wedge \Psi')U((e \neq val\_qt) \vee \chi'))$.
$K(W_i),s \models AG\Psi \Leftrightarrow K(W_{i+1}),s' \models A(\ ((e = val\_qt) \wedge \Psi')\ W\ (e \neq val\_qt)\ )$.
$K(W_i),s \models A\Psi W\chi \Leftrightarrow K(W_{i+1}),s' \models A(\Psi'\ W\ (\chi' \vee (e \neq val\_qt)))$

Sketch of proof : The proof proceeds by induction on the length of the formula Φ.

5 Result

We implemented a tool that automates the transformation of the CTL formulae described on 4.3. This tool took a file with a set of CTL formulae and a file containing the definition of an increment and returns the set of transformed CTL formulae.

We applied this approach for the verification of a VCI-PI wrapper. A wrapper is a device wrapping around an IP core and implementing a given interface. In our context, the IP core are supposed to be VCI compliant and the considered wrappers are adapters between the VCI interface and the PI-bus protocol, thus we are able to connect these IP-cores through a PI-bus. Using an incremental

[1] W stands for "weak until"

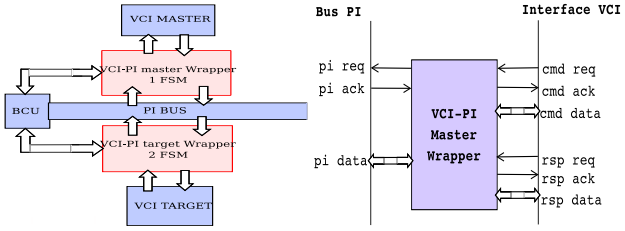

Fig. 4. Plateform ; Wrappers master interface

design process approach, we developed a set of six master VCI-PI wrappers, from a very simple one supposing that the VCI initiator and the PI target will always respond in one cycle, up to a more complex one supporting delays and retract events sent by the VCI initiator or the PI target.
The behavior of the simplest wrapper (model A) is a 3-stages pipeline, performing at the same time : accepting a VCI request k to be sent to PI from its VCI interface, sending the PI request corresponding to the k-1^{th} VCI request on its PI interface, accepting the PI response to the k-2^{th} VCI request on its PI interface. The further models (B to C') (fig.5) deal with external events disturbing the pipeline flow.

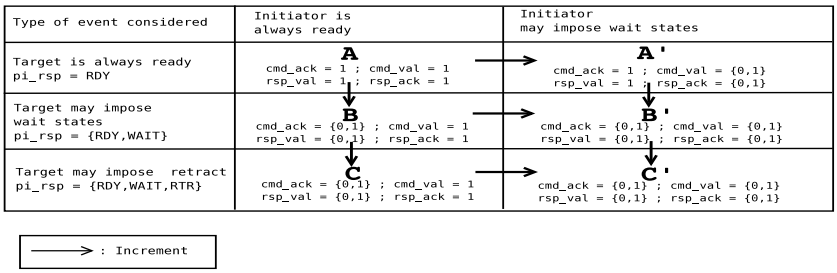

Fig. 5. Hierarchical VCI-PI wrapper

We implemented in synchronous Verilog a system containing a VCI initiator and a VCI target, connected to a PI-bus through a VCI-PI master wrapper and a VCI-PI slave wrapper. We verified this system with the VIS plateform [2].
We checked about 80 CTL properties for the master wrapper B, the slave wrapper B and the complete system (when the VCI initiator and target may generate delay events). Then, we checked the properties obtained by applying the CTL transformations on a system containing now VCI-PI master and slave belonging to model B'. Some examples of CTL properties checked is given in annex.
All properties that were successfully checked on the B plateform were translated to be accepted on plateform B', and the verification results were successfull. Of course, extra CTL formulae had to be added to the B'-platform in order to check

the behaviours added by the increment.
The whole system was composed of about 300-400 BDD variables, and the verification run from about a second to about a hour, depending on the complexity of the property under verification.

6 Concluding Remarks

The transformation rules of CTL formulae we propose is the basis to an approach to answer the "non-regression problem" encountered during the incremental design process of a component. We have shown that this approach can be used during the design of a concrete component, assuming the increment respects the rules we formalized, as we take advantage of the existence of a particular value tagging the initial part of a model included in an extended model. Nevertheless, the experimental results obtained have shown that the transformed CTL formula may become very complex (furthermore if the increment process is iterated), downgrading the verification process.

It is our intention to pursue this study towards the following directions :
Up to now, we did not take into account all the particularities of the increment; we considered only the existence of a particular value spliting the set of states with the ones that appeared in the initial model and the new ones. We did not take advantage of the graph structure of the increment; most of the time, it consists of the adding of a new state (or set of states) characterizing the freezing of the data-path. In these cases, a new set of CTL tranformations may be defined, simpler than the ones we proposed here in a more general context.
The opposite analysis can also be of interest : given a complex formula to be verified on a complex model, can we find an increment such that the complex model has been built from the application of this increment to a simpler model. If yes, can we transpose the complex formula to a simpler one to be verified.
We are also interested in studying the way this approach can be mixed with an Assume-Guaranty verification process.

References

1. J.R. Burch, E.M. Clarke, and K.L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation (Special issue for best papers from LICS90)*, 98(2):153–181, 1992.
2. The VIS group. Vis : A system for verification and synthesis. In *International Conference on Computer-Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 428–432. Springer-Verlag, 1996.
3. O. Grumberg and D.E. Long. Model checking and modular verification. In *International Conference on Concurrency Theory*, volume 527 of *Lecture Notes in Computer Science*, pages 250–263. Springer Verlag, 1991.
4. K.Lano. In *The B Language and Method, A guide to practical Formal Development*, FACIT. Springer-Verlag, 1996.
5. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. volume 6 of *Formal Methods in System Design*, pages 1–35. Kluwer, 1995.

Appendix

We present the proofs of each basic case of the CTL transformation. s' enriches s, K(W_i),s $\models$ p $\Leftrightarrow$ K(W_{i+1}), s' $\models$ p, p as an atomic proposition that is not concerned with the increment.

1. $\Rightarrow$ By definition, if s' enriches s, s' contains a greater set of atomic propositions than s. As s$\models$p, p is an atomic proposition of s, then p is an atomic proposition of s', hence s'$\models$p.
2. $\Leftarrow$ If p is not a property concerned with the increment and s'$\in$K(W_{i+1})enriches s$\in$K(W_i), then K(W_{i+1}),s'$\models$p $\Rightarrow$ K(W_i)$\models$p.

s' enriches s, K(W_i),s $\models$ EXp $\Leftrightarrow$ K(W_{i+1}), s' $\models$ (e = val_qt) $\Rightarrow$ EXp.

1. $\Rightarrow$ If s$\models$ EXp, there exists a state t $\in$ K(W_i) such that s$\rightarrow$ t and t $\models$ p. Let be a state s' enriches s with (e = val_qt), by corollary 2 there exists t' $\in$ K(W_{i+1}) such that s' $\rightarrow$ t' and t' $\in$ $B_{r_{inc}}(t)$. Hence K(W_{i+1}), t' $\models$p, K(W_{i+1}), s' $\models$ (e = val_qt) $\Rightarrow$EXp
2. $\Leftarrow$ Let be K(W_{i+1}),s'$\models$(e = val_qt)$\Rightarrow$EXp and s' enriches s with (e = val_qt), let be t' such that s' $\rightarrow$ t' and t'$\models$p. By the corollary 4, there exists t such that t'$\in$$B_{r_inc}$(t), then t$\models$p if p is not concerned with the increment and s' simulates s hence K(W_i),s $\models$ EXp.

s' enriches s, K(W_i),s $\models$ EFp $\Leftrightarrow$ K(W_{i+1}),s' $\models$ E((e = val_qt) U p).

1. $\Rightarrow$ If s$\models$ p, s'$\models$ p then s'$\models$ EFp.
   If s $\not\models$ p, there exists a path σ in K(W_i) : σ = s$\rightarrow$t $\rightarrow$ ... $\rightarrow$ r, such that r $\models$ p. By corollary 1, there exists a path σ' in K(W_{i+1}) : s'$\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r' such that s' enriches s with (e = val_qt), t' enriches t with (e = val_qt), ... , and r' $B_{r_{inc}}(r)$, then r$\models$p, hence s'$\models$E((e = val_qt) U p).
2. $\Leftarrow$ Let be K(W_{i+1}), s'$\models$E((e = val_qt) U p) and s' enriches s with (e = val_qt). There exists a path σ'in K(W_{i+1}) : s'$\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r' such that for all s'<u'<r' u'$\models$(e = val_qt) and r'$\models$p then there exists a path σ in K(W_i) : s$\rightarrow$t $\rightarrow$ ... $\rightarrow$ r such that for all s$\leq$u<r u' simulates u and r'$\in$ $B_{r_{inc}}$(r), then r$\models$p if p is not concerned by the increment and K(W_i), s $\models$EFp. No conclusion can be made in others cases.

s' enriches s, K(W_i),s $\models$ EGp $\Leftrightarrow$ K(W_{i+1}),s' $\models$ EG((e = val_qt) $\wedge$p).

1. $\Rightarrow$ If s $\models$ EGp there exists a path σ in K(W_i) s$\rightarrow$t $\rightarrow$ ... $\rightarrow$ r $\rightarrow$ ... , such that s $\models$ p, t $\models$ p, ... , r $\models$ p,... By corallary 1, there exists a path σ' in K(W_{i+1}) : s'$\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r'$\rightarrow$ ... , such that s' $\models$ p . (e = val_qt), t' $\models$ p . (e = val_qt), ... , r' $\models$ p . (e = val_qt), ... : s'$\models$ EG((e = val_qt) $\wedge$ p).
2. $\Leftarrow$ Let be K(W_{i+1}), s'$\models$E((e = val_qt)$\wedge$ p) and s' enriches s , s$\in$K(W_i). There exists a path in K(W_{i+1}) s'$\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r' $\rightarrow$ ... such for all state u' of this path u'$\models$p . (e = val_qt) then by corollary 4, ther exists a path σ $\in$ K(W_i) : s$\rightarrow$t $\rightarrow$ ... $\rightarrow$ r $\rightarrow$ ... such that s' enriches s, t' enriches t ... hence s$\models$p, t$\models$p if p is not concerned by the increment and K(W_i)$\models$EGp. No conclusion can be made in others cases.

s' enriches s, $\mathbf{K}(W_i)$,s $\models$ EpUq $\Leftrightarrow$ $\mathbf{K}(W_{i+1})$,s' $\models$ E(((e = val_qt)$\wedge$p) U q).

1. $\Rightarrow$ If s$\models$ q, then s'$\models$ q hence s'$\models$ E(((e = val_qt) $\wedge$ p) U q).
   If s$\models$ p and there exists a path σ : s$\rightarrow$t $\rightarrow$ ... $\rightarrow$ r $\rightarrow$ ... , such that s $\models$ p, t $\models$ p, ... , r $\models$ q, and s' enriches s with (e = val_qt) by corollary 1, there exists a path σ' in K(W_{i+1}) : s'$\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r' , such that s' $\models$ p . (e =val_qt), t' $\models$ p . (e = val_qt) , ... , r' $\models$ q : s'$\models$ E(((e = val_qt) $\wedge$ p) U q).
2. $\Leftarrow$ Let be K(W_{i+1}), s'$\models$ E(((e = val_qt)$\wedge$p) U q) and s' enriches s, s$\in$K(W_i). There exists a path in K(W_{i+1}) σ' = s' $\rightarrow$t' $\rightarrow$ ... $\rightarrow$ r'$\rightarrow$ ... such that s' $\models$ p . (e =val_qt), t' $\models$ p . (e = val_qt), ... , r' $\models$ q. By corollary 4, there exists a path in K(W_i) σ = s $\rightarrow$ t $\rightarrow$ ... $\rightarrow$ r $\rightarrow$ ... such that s' enriches s, t' enriches t, ..., r' $\in B_{r_{inc}}$(r) and σ satisfies pUq, if p and q are not concerned by the increment then K(W_i), s$\models$EpUq. No conclusion can be made in others cases.

s' enriches s, $\mathbf{K}(W_i)$,s$\models$ AXp $\Leftrightarrow$ $\mathbf{K}(W_{i+1})$,s'$\models$ (e = val_qt) $\Rightarrow$ AXp.

1. $\Rightarrow$ In K(W_i) all the successors r of s verify p. As s' enriches s with (e = val_qt) and there exists a set of the successors r' of s' which are in the incremental brotherhood of all the successors r of s. Let be r' $\in B_{r_{inc}}$(r), r' is represented as r' = r.(e = val_qt) or r'=r.(e $\neq$ val_qt). In the both cases r' $\models$p because r$\models$p and the atomic propositions are preserved in K(W_{i+1}). Hence all the successors of s' verify (e = val_qt) $\Rightarrow$ AXp.
2. $\Leftarrow$ Let be s' a state in K(W_{i+1}). s' can be a state in the brotherhood of s in K(W_i), or a state in the brotherhood of a new state.
   If s' $\in B_{r_{inc}}$(s), s$\in$K(W_i), then s' = s.(e = val_qt) ou s' = (e $\neq$ val_qt), moreover s' verify (e = val_qt) $\Rightarrow$ AXp, we have that s' = s.(e = val_qt) (s' enriches s with (e = val_qt)) verify AXp and s' = s.(e $\neq$ val_qt) can verify AXp or not, hence s$\in$K(W_i) verify AXp.
   If s' is a state in a brotherhood of a new state then s' = s.(e = val_qt) or s' = (e $\neq$ val_qt) but s' doesn't exist in K(W_i).

s' enriches s, $\mathbf{K}(W_i)$,s$\models$ AFp $\Leftrightarrow$ $\mathbf{K}(W_{i+1})$,s'$\models$ AF((e $\neq$ val_qt) $\vee$ p).

1. $\Rightarrow$ In K(W_i) for all path $\sigma = s_0, \ldots s_n \ldots$, there exists a state s_k, 0$\leq$k$\leq$n in which p is true. From the corallary 1, there exists some path in K(W_{i+1}) $\sigma' = s'_0, \ldots s'_n$, such that all the states labelled with (e = val_qt) and all states are in the incremental brotherhood of a state in K(W_i). Moreover, by constructing K(W_{i+1}) we have that there doesn't exist any transition t_k from a state in σ' to a state s'_{k+1} labelled with (e = val_qt) and which is not in $B_{r_{inc}}(s_{k+1})$ s_{k+1} in K(W_i).
   Induction hypothesis : If $s'_k \in \sigma'$ then $s'_k \models$AF(p + (e $\neq$ val_qt))
   From s'_0, there exists a path such that all the states verify (e = val_qt) and AF(p), hence $s'_0 \models$ (e = val_qt) and AF(p . (e = val_qt))
   Soit $s'_k \in \sigma'$, the set of these successors are as :
   $s'_{k+1} = s_{k+1}$.(e $\neq$ val_qt) and thus verify AF(p + (e $\neq$ val_qt))
   $s'_{k+1} = s_{k+1}$.(e = val_qt),the transition from $s_k \rightarrow s_{k+1}$ is in σ' and thus verify AF(p + (e $\neq$ val_qt)) (induction hypothesis).

2. $\Leftarrow$ By the corollary 2, the maximum computation tree in K(W_{i+1}) from s', s' enriches s with (e = val_qt) is K(W_i) in which e $\leftarrow$ val_qt. As K(W_{i+1})$\models$AF(p + (e $\neq$ val_qt)),and by the property preservation (definition 9 : preservation of ACTL-formulae), the sub-tree representing K(W_i) in K(W_{i+1}) verify A(p + (e = val_qt)), hence K(W_i), $\models$ AF(p).

s' enriches s, K(W_i),s$\models$ ApUq $\Leftrightarrow$ K(W_{i+1}),s'$\models$ A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_qt) $\vee$q)).

1. $\Rightarrow$ Let be $s_0 \in$ K(W_i), if $s_0 \models$q, then we have $s_0 \models$A(pUq). Let be $s'_0 \in$ K(W_{i+1}), s'_0 enriches s_0 with (e = val_qt) then $s'_0 \models$q.(e = val_qt), hence $s'_0 \models$ A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_qt) $\vee$q)).
   If $s_0 \not\models$q, $s_0 \models$p and $s_i \models$q, such that all s_i is a successor of s_0, then we have $s'_0 \models$p.(e = val_qt), by the corallary 4, all the successors of s'_0 are in the incremental brotherhood of a state $s_i \in$K(W_i), which is a successor of s_0. Then $s'_i \models$q, and we have K(W_{i+1}),s_0'$\models$A(p.(e = val_qt) U (e $\neq$ val_qt) $\vee$ q).
   Inductuon hypothesis : Let be r'$\in$K(W_{i+1}) such that r' enriches r with (e = val_qt) then we have r'$\models$ A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_qt) $\vee$q)).
   Let be s$\models$p we have for all r : s $\rightarrow$ r :
   r$\models$q $\Rightarrow$ r$\models$A(pUq) or
   r$\models$p and r$\models$A(pUq)
   Let be s' such that s' enriches s with (e = val_qt) we have s'$\models$p.(e = val_qt) and for all r' $\in B_{r_{inc}}$(r) such that s'$\rightarrow$ r' we have :
   r' = r.(e $\neq$ val_qt) hence r'$\models$A(((e = val_qt) $\wedge$ p) U ((e $_n eq$ val_qt) $\vee$q))
   r' = r.(e = val_qt) hence r' enriches r with (e = val_qt) and by induction r'$\models$A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_t) $\vee$q)).
2. $\Leftarrow$ Let be s' a state in K(W_{i+1}) such that s' verify A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_qt) $\vee$q)), if s' is in the incremental brotherhood of a state s in K(W_i), then s' = s.(e $\neq$ val_qt) and nothing can be said, or s' = s.(e = val_qt), in this case, by presevation of the ACTL-formula we have s$\models$ A(((e = val_qt) $\wedge$ p) U ((e $\neq$ val_qt) $\vee$q))$|_{i \leftarrow val\_qt}$= A(pUq).

s' enriches s, K(W_i),s$\models$ AGp $\Leftrightarrow$ K(W_{i+1}),s'$\models$ A(((e = val_qt) $\wedge$ p) W (e $\neq$ val_qt)).

1. $\Rightarrow$ Let be t'$\in$ $S_{K(W_{i+1})}$, t'$\models \neg$p.(e = val_qt) and $\sigma' = s' \ldots t'$. t' doesn't simulate a state in K(W_i) ($\forall$s$\in$ S_i s$\models$p). t' is a state in the brotherhood of a new state, t' is accessible only by a state where e$\neq$val_qt (corollary 4).
   Let be s', s' enriches s and $\sigma' =$ s'...r'...t' with s'$<$ r' $\leq$ t', if $\nexists$ u' labelled with (e $\neq$ val_qt) such that u'$<$r' then r'$\models$p.(e = val_qt) or r'$\models$(e $\neq$ val_qt).
   Moreover, by corollary 1, if there exists some infinite path in K(W_i), there exists some infinite path in K(W_{i+1}) labelled with (e = val_qt). Then there exists some path in K(W_{i+1}) which verify p.(e = val_qt).
   We have thus s'$\models$A(p.(e = val_qt) W (e $\neq$ val_qt))
2. $\Leftarrow$ By corollary 2, the maximum computation tree in K(W_{i+1}) from s', s' enriches s with (e = val_qt) and where all state are labelled with (e = val_qt)

is K(W_i) in which e is stucked to val_qt. As K(W_{i+1})⊨A(p.(e = val_qt) W i) in sub-tree representating K(W_i) e is always equal to val_qt, all the states of the sub-tree verify p.(e = val_qt) hence K(W_i)⊨ AGp.

s' enriches s, K(W_i),s⊨ ApWq ⇔ K(W_{i+1}),s'⊨ A((p ∧ (e = val_qt)) W (q ∨(e ≠ val_qt))).

1. ⇒ s⊨A(pWq) then all paths from s are such they verified pUq or Gp. In the first case, same reasoning as ApUq. In the second case, by corollary 1 these paths exist in K(W_{i+1}) labelled with (e = val_qt). The divergents behaviours are labelling with (e ≠ val_qt) (corollary 3). We have thus K(W_{i+1}),s'⊨ A((p ∧ (e = val_qt)) W (q ∨(e ≠ val_qt)))
2. ⇐ By the corollary 2, the maximum sub-graph in K(W_{i+1}) from s', s' enriches s with (e = val_qt) and where all path are labelled by (e = val_qt) is K(W_i), and is as p . . .pq ou p. . .p. We have thus K(W_i),s⊨ ApWq if p and q are not concerned by the increment.

Here are examples of CTL properties checked on models B :

```
AG ( (wrap0.state = R_REQ) -> (A( (m_pi_req = 1) U (m_pi_gnt = 1))));
Ckeck the interface between the inititor VCI and the master wrapper.

!EF((wrap_cible.cmd_cible.state = CMD_IDLE) *
!(wrap_cible.rsp_cible.state = RSP_IDLE));
Check the behavior of the slave wrapper.

AG( (m_cmd_plen[6:0] = 8 * m_cmd[0] = 1 ) ->
% A ( (A (
(A( (m_cmd_plen[6:0] = 8 * m_cmd[0] = 1 * m_cmd_eop = 0 * m_cmd_val = 1)
 U  (m_cmd_ack = 1)))
  U   ( A( (m_cmd_eop = 1 * m_cmd_val=1)
   U (m_cmd_ack = 1)))))
 U (m_cmd_val = 0) ));
Check the behavior of the complete system, check the number of packet send
with the number of acknowledgement recieved.
```