1. INTRODUCTION

Formal verification of hardware designs is an important area of research [Borrione 96]. Among the possible approaches, symbolic model checking [McMillan 93] has already been used to verify behavioral equivalence or CTL properties of hardware designs : VFORMAL [CLSI 93], SMV [McMillan 93], VIS [BR95], SEVERO [CCP 93], VPN-VMC [Encrenaz 95a, Bawa 96]. Of course formal verification techniques do not apply to real hardware design, but to a model representing the behaviour of the design. In [BE 96] we present a platform devoted to the verification of hardware designs described in VHDL, which extracts a symbolic transition relation[1] from an elaborated VHDL program [Encrenaz 95b], and then performs symbolic model checking in order to verify CTL properties or behavioral equivalence of two different VHDL descriptions. The performances obtained with this platform allow us to verify medium complexity designs, as the DLX [HP80] or 8086 microcontrollers.

One way to treat more complex designs consists in considering their structural information : complex systems can be seen as a collection of interacting simpler components (e.g. a structural VHDL program is composed of entities linked by signals), each one represented by a transition relation. The verification platform can verify some properties concerning each module, but one has to combine these modules to verify global properties. A first way to verify a global property is to compute the synchronized product of the elementary transition relations, and then to apply symbolic model checking techniques, but in this case we obtain a unique transition relation as big as the one extracted from the "flat" description of the system.

Another approach consists in reducing each transition relation, according to the property to be verified and the transition relations interconnections, before combining them in a global transition relation, thus leading to a smaller global transition relation easier to manipulate. The reduction of each component is based on the identification of equivalent states aggregated into equivalence classes. Strong bisimulation [Milner 89] is known to be the coarsest[2] equivalence relation that preserves all CTL formulae. In the context of hardware verification, the specification to be verified is usually composed of a restricted number of CTL formulae to be verified, and thus instead of preserving all CTL formulae, we present an equivalence relation which depends on the property to be verified, and preserves the CTL formula to be verified and the composition of components : the π-bisimulation. This equivalence relation is even coarser than the strong bisimulation in the general case, leading to a better reduction of each transition relation.

In [ASSS-V 94] a compositional, property-based model checking technique is already presented, but our approach differs from theirs in the sense that (1) we do consider an initial state, as VHDL programs have an initial state, hence the component composition is based on the synchronized product with respect to an initial state. The example given in our paper highlights the wrong results of

[1] the symbolic transition relation expresses the set of reachable states in one VHDL simulation cycle, from a given set of states.

[2] the coarsest relation is the one that renders equivalent the biggest number of states (according to a criterion), hence this is the one that offers the biggest reduction power.

the reduction proposed by [ASSS-V 94] in our context. (2) Our approach consists in reducing the transition relation according to atomic propositions of CTL formulae, thus we can compute the reduced FSM independently of all the other FSM of the system interacting with this one, contrary to [ASSS-V 94]. This reduction over atomic propositions makes the component composition simpler. The work of [SR 94] in the MEIJE project concerns also the compositional verification of parallel systems based on an *a priori* reduction of each component by bisimulation. They verify that a component is bisimilar to another one, simpler, representing the formula to be verified over this component. In our knowledge, they do not reduce each component according to a global property, taking into account the synchronisation between components, in order to verify the global property on the simplified system, as we propose. [LL 95] also presented a compositional model checking approach, but their work is based on the quotienting of the formula instead of the quotienting of the components.

The paper is organised as follows : the preliminary section (2) recalls definitions concerning FSMs. Section 3 defines the reduction of each component by bisimulation and presents the relation we propose to reduce each VHDL component. It is shown that it is equivalent to verify an atomic property over the synchronized product of the initial transition systems and of the reduced transition systems. Section 4 extends the result to all CTL operators. Section 5 recalls the computation of the coarsest bisimulation algorithm [BFH 94] and presents the modifications to be made in order to consider the relation we defined. Section 6 details the reduction of a VHDL entity and provides experimental results of the modular verification of VHDL programs, and then (section 7) we conclude and suggest some directions for future work.

2. PRELIMINARIES

2.1. From VHDL to transition relations

An elaborated VHDL program is represented by a transition relation which mimics the VHDL simulation cycle. As in [BE 96], a state of a VHDL entity is composed of the program counter of each process, the value of each variable, the value of each signal driver, the effective value of each signal, and the value of the event attribute associated to each signal. The way the transition relation is built from the VHDL program is detailed in [Encrenaz 95]. Roughly, for each state variable v_k, the equation modelling the evolution of this variable along the $i+1^{th}$ simulation cycle is built from the state of the system at the end of the i^{th} simulation cycle :

$$v_k[i+1] = f_k(v_1[i], ..., v_n[i])$$

The transition relation is then a conjunction of the equation of all state variables.

$$TR = \bigwedge_k v_k[i+1] = f_k(v_1[i], ..., v_n[i])$$

The VHDL program has an initial state, as defined in [LRM 87] : each state variable is initialised to the leftmost value of its definition interval, except when they are explicitly initialised by the programmer.

Sets of states and transition relation are represented by BDD structures, and one can perform state space traversals, either to verify temporal logic properties or to check the equivalence of two descriptions of a same system, that is to find the set of reachable states in which a property holds. Various improvements have been proposed which speed up the verification process and reduce the memory used, for example external stimuli representation, partitioned transition relation, intermediary variables simplification are discussed in [BE 96, Bawa 96].

Another direction to be studied is the compositional verification, which starts from a structural VHDL program instead of an elaborated one. In this case, we distinguish components linked together by signals, and associate a transition relation per component. A conjunction of these transition relations exactly represents the flat VHDL program, and hence no improvement can be expected, except if we reduce the transition relations before their conjunction : each reduced transition relation may be expressed with a smaller number of variables, reducing the size of the transition relation of the whole program, and simplifying the downstream verification process.

2.2. Definitions

Definition 1: Finite State Machine

A Finite State Machine (FSM) is a 6-tuple $\boldsymbol{M=(S,I,J,T,O,S_o)}$ *with :*
$\boldsymbol{S}$ *the finite set of states,* $\boldsymbol{I}$ *the finite set of input values,* $\boldsymbol{J}$ *the finite set of output values,* $\boldsymbol{T}$ *the transition function,* $t \in \boldsymbol{T}$ *is a 3-tuple (s,i,s') such that : s and s' of* $\boldsymbol{S}$ *are resp. source and destination states of the transition and* $i \in \boldsymbol{2^I}$ *is the label of t, hence* $\boldsymbol{T} \subseteq \boldsymbol{S} \times \boldsymbol{\Sigma} \times \boldsymbol{S}$ *with* $\boldsymbol{\Sigma} = 2^{\boldsymbol{I}}$*,* $\boldsymbol{O}$ *is the output function :* $\boldsymbol{S} \rightarrow \boldsymbol{2^J}$ *and* $\boldsymbol{S_o}$ *is the initial state.*

The FSM is complete, i.e. : $\forall s \in \boldsymbol{S}$, $\forall i \in \boldsymbol{I}$, $\exists s' \in \boldsymbol{S}$ *such that* $(s,i,s') \in \boldsymbol{T}$.
If the FSM is deterministic and complete, the transition relation is a function $\boldsymbol{T} : \boldsymbol{S} \times \boldsymbol{\Sigma} \rightarrow \boldsymbol{S}$

Definition 2 : reachable state

Let $\boldsymbol{M=(S,I,J,T,So)}$ *a FSM, a state* $s' \in \boldsymbol{S}$ *is a reachable state from a state* $s \in \boldsymbol{S}$ *iff :*
$\exists (i_1,i_2.....i_j.....i_n)$ *s.t.* $\forall j \in [1,n]$, $i_j \in 2^{\boldsymbol{I}}$ *and* $\boldsymbol{T}(s_{k-1}, i_k) = s_k$ *for* $1 \le k \le n$ *and* $s_n = s'$ *and* $s_0 = s$

Definition 3 : Cartesian product of automata

Let $\boldsymbol{M_1=(S_1,I_1,J_1,T_1,O_1,S_{o1})}$ *and* $\boldsymbol{M_2=(S_2,I_2,J_2,T_2,O_2,S_{o2})}$ *be two FSMs, the Cartesian product* $\boldsymbol{M_1} \times \boldsymbol{M_2}$ *is the FSM* $\boldsymbol{M(S,I,J,T,O,So)}$ *such that :* $\boldsymbol{S} = \boldsymbol{S_1} \times \boldsymbol{S_2}$, $\boldsymbol{T} = \boldsymbol{T_1} \times \boldsymbol{T_2}$, $\boldsymbol{I} = \boldsymbol{I_1} \times \boldsymbol{I_2}$, $\boldsymbol{J} = \boldsymbol{J_1} \times \boldsymbol{J_2}$, $\boldsymbol{S_o} = (\boldsymbol{S_{o1}},\boldsymbol{S_{o2}})$, $\boldsymbol{O} : \boldsymbol{S} \rightarrow \boldsymbol{J}$, *that maps (s) into* $\boldsymbol{O}(s)$ *such that : if* $s=(s_1,s_2)$ *then* $\boldsymbol{O}(s,i) = (\boldsymbol{O_1}(s_1) , \boldsymbol{O_2}(s_2))$.

This Cartesian product represents the behaviour of two components not connected, having either synchronous or asynchronous execution (if the null transition exists), and with the assumption of an initial state. The Cartesian product can be restricted when components are linked together and when a synchronous or asynchronous semantics is defined, as in Figure 2. This leads to the definition of the synchronised product given below. To make notations simpler, we provide the definition only in the case of deterministic and complete FSMs.

Definition 4 : FSM connection and synchronisation constraint

Let $\boldsymbol{M_1=(S_1,I_1,J_1,T_1,O_1,S_{o1})}$ *and* $\boldsymbol{M_2=(S_2,I_2,J_2,T_2,O_2,S_{o2})}$ *be two linked FSMs (i.e. some outputs of* M_1 *are inputs for* M_2 *and some outputs of* M_2 *are inputs for* M_1 *:*

$J_{12} = J_1 \cap I_2 \neq \emptyset$ *is the set of output variables of* M_1 *connected to* M_2

$J_{21} = J_2 \cap I_1 \neq \emptyset$ *is the set of output variables of* M_2 *connected to* M_1

$I_{11} = I_1 \setminus J_{21}$ *is the set of primary input variables of* M_1

$I_{22} = I_2 \setminus J_{12}$ *is the set of primary input variables of* M_2.

The transition function of M_1 is defined as :

$T_1 : S_1 \times 2^{I_{11}} \times 2^{J_{21}} \rightarrow S_1$ *which given a state* s_1 *of* S_1*, a set of values* i_1 *of* $2^{I_{11}}$ *and a set of values* j_{21} *of* $2^{J_{21}}$ *associates* $T_1(s_1, i_1, j_{21}) = s'_1 \in S_1$

The transition function of M_2 : $T_2(s_2, i_2, j_{12}) = s'_2$ is defined similarly.

We define the restriction of the output function to the connected outputs by :

$O_{21} : S_2 \rightarrow 2^{J_{21}}$ *that associates to any state* s_2 *of* S_2 *the value of the output variables of* M_2 *connected to* M_1.

Similarly we define O_{12} by :

$O_{12} : S_1 \rightarrow 2^{J_{12}}$ *that associates with any state* s_1 *of* S_1 *the value of the output variables of* M_1 *connected to* M_2.

Let (s_1,s_2) *be a state of* $\boldsymbol{S_1} \times \boldsymbol{S_2}$*. The set of reachable states from* (s_1,s_2) *in one global transition is the set of states* (s'_1,s'_2) *that verify the constraints :*

$T_1(s_1, i_1, O_{21}(s_2)) = s'_1$ *and* $T_2(s_2, i_2, O_{12}(s_1)) = s'_2$

where i_1 *is any value of the primary input variables of* M_1 *and* i_2 *any value of the primary input variables of* M_2.

These two constraints are called synchronisation constraints over transitions of $\mathbf{M_1} \times \mathbf{M_2}$.

Definition 5 : synchronized product

Let be $\boldsymbol{M_1=(S_1,I_1,J_1,T_1,O_1,S_{o1})}$ *and* $\boldsymbol{M_2=(S_2,I_2,J_2,T_2,O_2,S_{o2})}$*, two FSMs, and their Cartesian product* $M=(S,I,J,T,O,So)$*. Let* (U_n) *be a serie of elements of* $S_1 \times S_2$ *defined by :*

- $U_o = (S_{o1},S_{o2})$ [3]
- $U_n = \{$ *reachable states in one global transition from* $U_{n-1} \} \cup U_{n-1}$

then the serie (U_n) *admits a limit* Q *included in* $\boldsymbol{S_1} \times \boldsymbol{S_2}$*. The FSM* M' *included in* M *containing only states of* S *belonging to* Q *is the synchronized product of* M_1 *and* M_2*, denoted* M_1 ***o*** M_2.

[3]The synchronized product used by [ASSS-V 94] does not assume an initial state : it is defined as the limit of the serie given in definition 5 with $U_o = S_1 \times S_2$.

Example : synchronized product

Let be two components M_1 and M_2 given on Figure 1. Component M_1 has an input signal a and an output signal p. Its behaviour is represented by the FSM M_1. Component M_2 has an input signal p (which is the output of M_1), and two output signals a (which is the input signal of M_1) and q. Its behaviour is represented by the FSM M_2. Values of the input signals are transition labels, and values of the output of the FSMs are represented near each corresponding state.

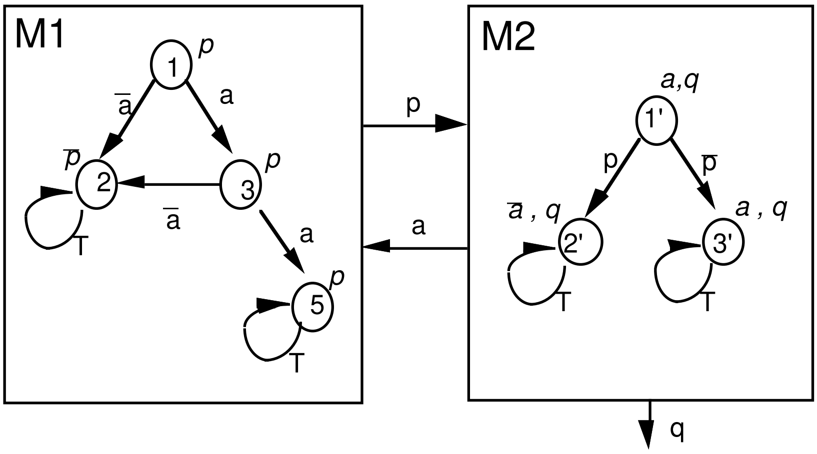

Figure 1. A system composed of two components M_1 and M_2.

Two synchronized products are depicted on figure 2. They represent the behaviours of the system when the components are connected and synchronous. The graph on the left represents the synchronized product without assumption of initial state, this is the composition used by [ASSS-V 94] , while the graph on the right is the synchronized product as defined above, assuming a specified initial state (the initial states of M_1 and M_2 are resp. state 1 and 1'). The synchronized product with initial state is a subgraph of the one defined by [ASSS-V 94]. In the following, we will consider systems having an initial state (as we work on VHDL programs).

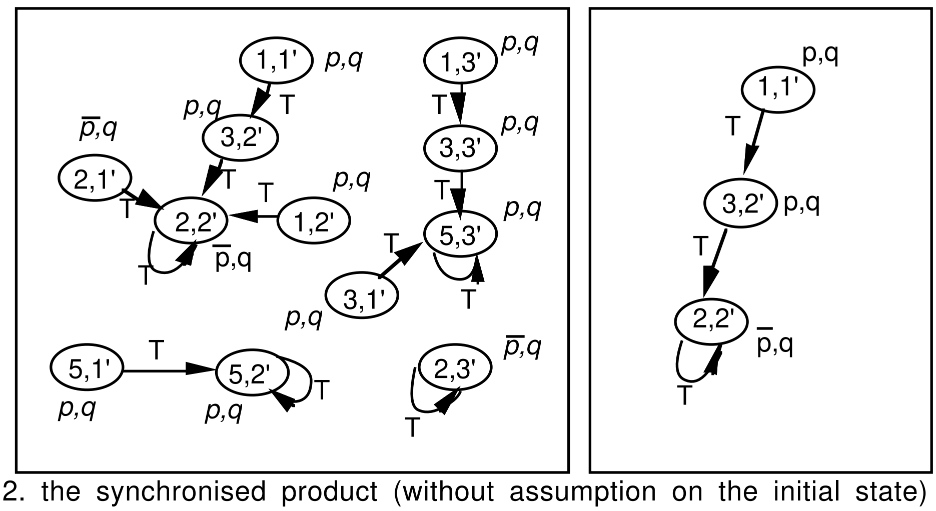

Figure 2. the synchronised product (without assumption on the initial state) and the synchronised product (M_1 o M_2) (with a specified initial state). T means that for any input configuration, the label in the transition function is true .

3. COMPONENT REDUCTION BY BISIMULATION

Our approach consists in identifying an equivalence relation $\mathcal{R}$ between states of a FSM, and then using the quotient FSM with respect to $\mathcal{R}$ for the verification instead of the initial FSM. As the quotient FSM will be smaller than the initial FSM, so will be the synchronized product, therefore simplifying the verification process. The equivalence relation $\mathcal{R}$ must preserve the synchronized product, i.e. $\mathcal{R}$ must be a congruence with respect to this composition operation since it must be equivalent to verify the property over the synchronized product of the initial FSMs and over the synchronized product of the quotient FSMs. There are two reasons for this :

- Most of the properties to be verified are global properties, hence preserving the composition is necessary.
- Even if the property is local to a module, one has to take the environment into account as the reachability graph of a component is not the same if the component is connected to the others or not, due to the synchronisation constraints.

remark 1 : In this paper for simplicity of presentation, we will restrict ourselves to a system composed of two components. The results presented would easily extend to modular symbolic model checking of systems with more than two interconnected FSMs.

3.1. Bisimulation is a necessary condition

In this section, we shall exhibit that a necessary condition for the equivalence relation to preserve the property and the composition is that it is a bisimulation. This is first illustrated by the following counter-example.

Let us consider again the system of Figure 1, over which we would like to verify the global property "$p \wedge q$", i.e. to find the states in which p and q are true. The intuitive idea would lead to the reduction of M_1 (resp. M_2) distinguishing the states where "p" (resp. "q") is true, from those were "p" (resp. "q") is false. The relation $\mathcal{R}_1$: $\mathcal{R}_1 \subseteq S_1 \times S_1$: $x \mathcal{R}_1 y \Leftrightarrow (x \models p \Leftrightarrow y \models p)$ (resp. $\mathcal{R}_2$) aggregates states of M_1 (resp. M_2) having the same truth value for p (resp. q).

The reduction of M_1 (resp. M_2) by $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) is illustrated on Figure 3. Notice that this reduction is similar to the one proposed by [ASSS-V 94] for the property $\pi = p \wedge q$[4], but here each component can be reduced independently of the others, while the definition of [ASSS-V 94] imposes the construction of the product of M_1 and M_2, without assumption of an initial state, to determine the equivalent states in each module[5].

[4] The formula-dependent equivalence given by [ASSS-V 94] for the property $\pi = p \wedge q$ is :
two states (x,y) of M_1 are equivalent iff :$\forall\ s_2 \in M_2$, $(x,s_2) \models \pi$ iff $(y,s_2) \models \pi$

[5] To be totally fair, [ASSS-V 94] would synchronise $M_{1/R1}$ and $M_{2/R2}$ without assumption of initial state, then the states (C_1,C_2'), (C_2,C_1') and (C_2,C_2') would be represented, invalidating the formula. But these classes are not reachable from the initial state, and we want to find the reachable states that satisfy or not the formula. [ASSS-V 94] can not answer to this question.

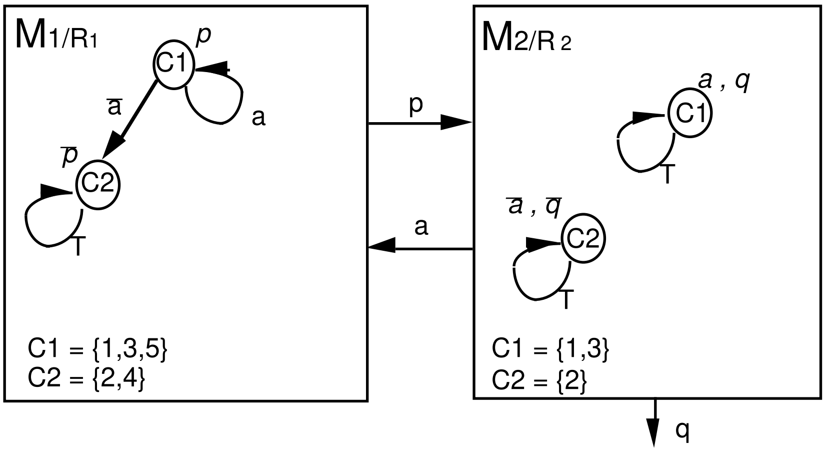

Figure 3. Each module is reduced with respect to its equivalence relation : Concerning M_{1/R_1} (resp. M_{2/R_2}), state 1 (resp. 3) was chosen as representative of the class C_1.

Each reduction (M_1 by $\mathcal{R}_1$) and (M_2 by $\mathcal{R}_2$) preserves the property, by definition, but the synchronized product is not respected : the synchronized product of the reduced components M_{1/R_1} and M_{2/R_2} is shown on figure 4. All its states verify "$p \wedge q$", hence the verification of "$p \wedge q$" over this graph would conclude that "$p \wedge q$" is always true. This assertion is wrong, as there exists some states in the initial synchronized product where "$p \wedge q$" is false (i.e. state (2,2') on figure 2, right). The problem is due to the non preservation of paths (1) validating or invalidating the formulae to be verified, and (2) along which interface signal values are modified: in M_{1/R_1}, the path σ = a a !a leads to a state where "p" is false, whilst in M_1 it leads in a state where "p" holds.

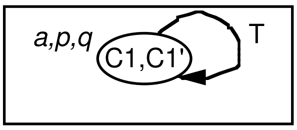

Figure 4 : synchronized product of M_{1/R_1} and M_{2/R_2}

A reduction relation verifying the path-preserving-property garantees that the synchronized product is preserved. Informally, two states x and y equivalent by an equivalence relation $\mathcal{R}$ verifying the path-preserving property iff all the next states of x and y for the same inputs are still equivalent by $\mathcal{R}$.

Definition 6 : path-preserving property

*An equivalence relation $\mathcal{R}$ between states of a FSM **M(S,I,J,T,O,So)** verifies the path-preserving property iff :*

$\forall (x, y) \in \mathbf{S},\ x\,\mathcal{R}\,y \Rightarrow$

1) $\forall i \in \mathbf{I}, \forall t \in \mathbf{S}$ *such that* $(x, i, t) \in T$ *then* $\exists w \in \mathbf{S}$ *such that* $(y, i, w) \in T$ *and* $\underline{t\,\mathcal{R}\,w}$

2) $\forall i \in \mathbf{I}, \forall t \in \mathbf{S}$ *such that* $(y, i, t) \in T$ *then* $\exists w \in \mathbf{S}$ *such that* $(x, i, w) \in T$ *and* $\underline{t\,\mathcal{R}\,w}$

Theorem 1 :

Let $\mathcal{R}$ be an equivalence relation included in $N \subseteq S \times S$: $x\,N\,y \Leftrightarrow (x \models \pi \Leftrightarrow y \models \pi)$, then a necessary condition for $\mathcal{R}$ to preserve the synchronized product is : $\mathcal{R}$ verifies the path-preserving property.

Proof : Without loss of generality, we will prove the theorem assuming that the property π refers to the outputs of a component $\mathbf{M_1(S_1,I_1,J_1,T_1,O_1,S_{o1})}$. Let $\mathcal{R}$ be a relation included in N $\subseteq S \times S$: x N y $\Leftrightarrow$ (x $\models \pi \Leftrightarrow$ y $\models \pi$), and not verifying the path-preserving property.
we will show that it is not equivalent to verify the property π over M_1 **o** M_2 and over $M_{1/R}$ **o** M_2, where M_2 is a any FSM, and $M_{1/R}$ is M_1 quotiented by $\mathcal{R}$.

Let s_1 and s_2 be two states of S_1 reachable from S_{o1}, and $\mathcal{R}$-equivalent :
$\exists\ \sigma_1 \in \mathbf{I_1} \times .... \times\ \mathbf{I_1}$, S_{o1} -$\sigma_1\rightarrow s_1$ and $\exists\ \sigma_2 \in \mathbf{I_1} \times .... \times\ \mathbf{I_1}$, S_{o1}-$\sigma_2\rightarrow s_2$,
and $s_1 \models \pi$ and $s_2 \models \pi$ or $s_1 \models \neg\pi$ and $s_2 \models \neg\pi$.
without loss of generality, assume $s_1 \models \pi$ and $s_2 \models \pi$.

If $\mathcal{R}$ does not verify the path property, then :
$\exists$ i $\in \mathbf{I_1}$ and $\exists$ $s'_1 \in \mathbf{S_1}$, such that (s_1 , i , s'_1) $\in$ $\mathbf{T_1}$ and
$\exists$ $s'_2 \in \mathbf{S_2}$, such that (s_2 , i , s'_2) $\in$ $\mathbf{T_2}$ and
s'_1 and s'_2 are not $\mathcal{R}$-equivalent : $s'_1 \models \pi$ and $s'_2 \models \neg\pi$ or $s'_1 \models \neg\pi$ and $s'_2 \models \pi$.
without loss of generality, assume $s'_1 \models \pi$ and $s'_2 \models \neg\pi$.

As s_1 and s_2 belong to the same equivalence class C, each of them may be chosen for representing C.

- If s_1 is chosen, then C is reachable from S_{o1} by a path σ_1, and the successor of C by the transition labelled i is s'_1, in which the property is true.
- If s_2 is chosen, then C is reachable from S_{o1} by a path σ_2, and the successor of C by the transition labelled i is s'_2, in which the property is false.

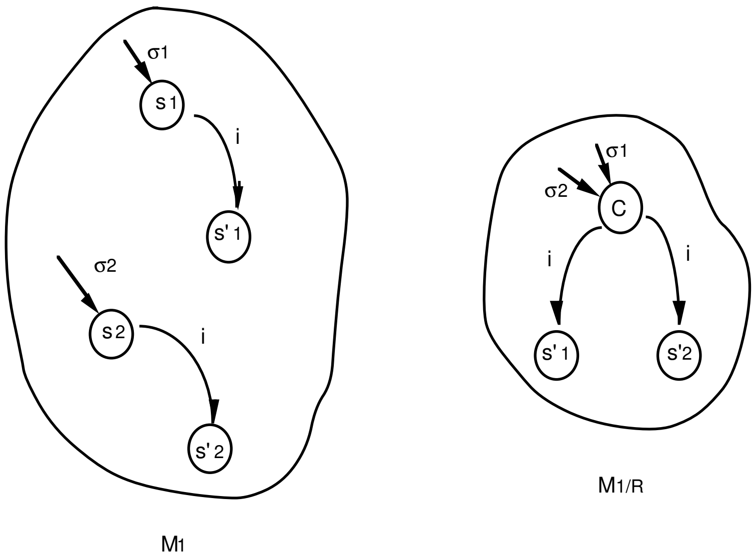

Figure 5. Definition of states and transitions in M_1 and the corresponding ones in $M_{1/R}$.

Then $M_{1/R}$ is non deterministic, as it contains the paths σ_1.i falsifying π, and σ_2.i validating π from S_{o1}, which are not in M_1.

Let us assume s_2 not to be reachable in M_1 o M_2, while s_1 is reachable : the synchronisation constraints of M_2 over M_1 eliminate all the paths from S_{o1} to s_2, but has preserved a path from S_{o1} to s_1, for example σ_1. Then s'_2 is not reachable in M_1 o M_2 from the path σ_1.i, while it is reachable in $M_{1/R}$ o M_2 since s_1 (reachable) and s_2 (unreachable) are equivalent. Thus, the set of states verifying π is different in M_1 o M_2 and in $M_{1/R}$ o M_2 : A relation that does not satisfy the path-preservation property does not preserve the synchronized product. ❑

As we shall see in the following sections, an equivalence relation satisfying the path-preservation property is a bisimulation. The most common bisimulation is the strong bisimulation introduced by [Milner 89], it is recalled in section 3.2. The general bisimulation will then be defined as [Arnold 92] does in section 3.3, and our equivalence relation is introduced in section 3.4.

3.2. Strong bisimulation

Definition 7 : strong bisimulation

Let $\boldsymbol{M(S,I,J,T,O,S_o)}$ *be a FSM, two states x and y of S are equivalent in the sense of strong bisimulation iff the three conditions hold :*

(i) $\boldsymbol{O}(x) = \boldsymbol{O}(y)$

(ii) $\forall i \in \boldsymbol{I}$ *, such that* $(x, i, x') \in T$, $\exists y' \in \boldsymbol{S}$, *such that* $(y,i,y') \in \boldsymbol{T}$ *and x' equivalent to y'*

(iii) $\forall i \in \boldsymbol{I}$ *, such that* $(y, i, y') \in T$, $\exists x' \in \boldsymbol{S}$, *such that* $(x,i,x') \in \boldsymbol{T}$ *and y' equivalent to x'.*[6]

Definition 8 : quotient automaton

If we denote by $\approx f$ *the strong bisimulation relation, then the strongly bisimilar FSM of* $\boldsymbol{M=(S,I,J,T,O,So)}$ *is the FSM* $\boldsymbol{M'=(S_{/\approx f},I,J,T,O,So)}$ *where* $\boldsymbol{S_{/\approx f}}$ *is the quotient of* $\boldsymbol{S}$ *by the equivalence relation* $\approx f$.

This strong bisimulation has the property to preserve all CTL formulae (i.e. it is equivalent to verify a CTL formula over the initial FSM and over the quotient FSM) [CLM 89]. In fact, [CLM 89] has shown that it is the coarsest equivalence relation preserving all CTL formulas. It also preserves the synchronized product [CLM 89] (i.e. it is equivalent to verify a CTL formula over the synchronized product of the initial FSMs and the synchronized product of the quotient FSMs). Moreover [BFH94] proposes an algorithm to build the strongly bisimilar FSM of a given FSM.

Nevertheless, the strong bisimulation has a low reducing power as it only aggregates redundant states (i.e. having the same outputs and all transitions leading to equivalent states).

As the strong bisimulation is the coarsest equivalence relation preserving all CTL formulae, we can not expect to find a coarser equivalence relation preserving all CTL formulae and the synchronized product. But as a matter of fact, if we focus on the verification of a unique property, then we can built an equivalence relation that preserves this CTL formula and the synchronized product and that is coarser than the strong bisimulation. This approach makes sense since for the formal verification of hardware devices, we only verify a restricted number of CTL formulae, and for

---

[6] The strong bisimulation is an equivalence relation such that x R y iff O(x) = O(y) and verifies the path-preserving property.

a given formula, it may be more efficient to manipulate a small graph preserving the formula only rather than a big graph preserving all CTL formulae. On the other hand, we will have to build a new reduced graph for each property to be verified.

In the following of the section, we present the general bisimulation and how we can derive our equivalence relation preserving an atomic property and the synchronized product from it.

3.3. The general bisimulation

Definition 9 : bisimulation and autobisimulation

Let $\boldsymbol{M_1=(S_1,I_1,J_1,T_1,O_1,S_{o1})}$ *and* $\boldsymbol{M_2=(S_2,I_2,J_2,T_2,O_2,S_{o2})}$*, two FSMs such that* $I_1=I_2$*, a bisimulation between* M_1 *and* M_2 *is a binary relation* $\mathcal{R}$ *beetween* S_1 *and* S_2 *iff :*

- *(i)* $\forall s_1 \in S_1, \exists s_2 \in S_2$ *such that* $s_1 \mathcal{R} s_2$
- *(ii)* $\forall s_2 \in S_2, \exists s_1 \in S_1$ *such that* $s_2 \mathcal{R} s_1$
- *(iii)* $\forall (s_1, s_2) \in S_1 \times S_2$*, such that* $s_1 \mathcal{R} s_2$ *and* $s_2 \mathcal{R} s_1$*,*
  $\forall i \in I_1=I_2, \exists s'_1 \in S_1$ *and* $\exists s'_2 \in S_2$ *such that :*
  $(s_1, i, s'_1) \in T_1$ *and* $(s_2, i, s'_2) \in T_2$ *then* $s'_1 \mathcal{R} s'_2$ *and* $s'_2 \mathcal{R} s'_1$*.*

An autobisimulation is a bisimulation between an automaton and itself.

[Arnold 92] defines an application E($\mathcal{R}$) in order to built the biggest bisimulation included in a relation $\mathcal{R}$. E($\mathcal{R}$) is defined as follows :

Definition 10 : E($\mathcal{R}$)

Let $M(S,I,J,T,O,S_o)$ *a FSM, and E the application :*

$\S(S \times S) \quad \rightarrow \quad \S(S \times S)$ *(* $\S(S \times S)$ *is the set of lattices of* $S \times S$ *)*

$\mathcal{R} \quad \rightarrow \quad \mathcal{E}(\mathcal{R})$

such that : $\forall (s_1, s_2) \in S, (s_1, s_2) \in \mathcal{E}(\mathcal{R})$ *iff :*

(i) $(s_1, s_2) \in \mathcal{R}$

(ii) $\forall i \in I, \forall s'_1 \in S$ *such that* $(s_1, i, s'_1) \in T, \exists s'_2 \in S$ *such that* $(s_2, i, s'_2) \in T$ *and* $s'_1 \mathcal{R} s'_2$

(iii) $\forall i \in I, \forall s'_2 \in S$ *such that* $(s_2, i, s'_2) \in T, \exists s'_1 \in S$ *such that* $(s_1, i, s'_1) \in T$ *and* $s'_2 \mathcal{R} s'_1$

Then $\mathcal{E}(\mathcal{R}) \subseteq \mathcal{R}$ *is monotonic decreasing, and admits a biggest fixed-point* $\bigcap_{n\geq 0} \mathcal{E}^n(\mathcal{R})$*.*

[Arnold 92] gives the following proposition :

Proposition 1: Let $\mathcal{R}$ an arbitrary relation included in S × S and let $\mathcal{R}f$ the relation defined by $\mathcal{R}f = \bigcap_{n\geq 0} \mathcal{E}^n(\mathcal{R})$, then the two following statements are equivalents :

(i) There exists a bisimulation $\mathcal{R}'$ included in $\mathcal{R}$

(ii) $\mathcal{R}f$ is the biggest bisimulation included in $\mathcal{R}$

From this proposition, one can derive an algorithm to built the biggest bisimulation included into a given relation. We will present the algorithm proposed by [BFH 94] in section 5.

From these results, we can build an equivalence relation preserving the property and the synchronized product. This is the object of the following sub-section.

3.4. property-dependent bisimulation : an equivalence relation preserving a property and the synchronized product

3.4.1. The property to be verified

First of all we focus on global properties that are *static* : they do not contain temporal operator. Moreover, they can be expressed as a conjunction or a disjunction of Boolean propositions depending exclusively of variables of a given module. Let us denote π the global property, π_1 the Boolean proposition concerning the variables of the module M_1 and π_2 the one concerning the variables of the module M_2. Then $\pi = \pi_1 . \pi_2$ or $\pi = \pi_1 + \pi_2$.

In Section IV, we will show that the global properties can be extended to CTL formulae, since we reduce each module according to the atomic propositions of the formula.

Remark 2 : The reduction of each module will be made according to the atomic propositions of the CTL formula that concern this module. In the above example (cf. Figure 1), if the global property to be verified was "p+q", M_1 was reduced according to π_1 = p and M_2 according to π_2 = q

3.4.2. The property-dependent bisimulation

Our equivalence relation must preserve the property π. This means that we can aggregate two states into the same equivalence class if the property is either true in both states, either false in both states :

$s_1 \mathcal{R} s_2$ implies $s_1 \models \pi \Leftrightarrow s_2 \models \pi$

On the other hand, we must preserve the interface behaviour of the component in order to insure the preservation of the synchronized product. This means that we can not aggregate two states having different values on the output signals. Thus

$s_1 \mathcal{R} s_2$ implies O[7]$_{interface}(s_1) = O_{interface}(s_2)$

If we take a bisimulation included in the two constraints above, then we preserve all sequences in the bisimilar FSM modifying the value of π and of each output signal value. This is what we will prove in the following of this section.

Definition 11 : property-dependent relation

Let two FSMs M_1 and M_2 linked together, let $O_{interface}$ the set of outputs of M_1 that are input for M_2, let π a static property, we call Q the equivalence relation between states of the FSM M_1 defined by :
$\forall (x,y) \in M_1 \times M_1$, $x \, Q \, y$ iff

(i) $x \models \pi \Leftrightarrow y \models \pi$

(ii) $O_{12}(x) = O_{12}(y)$

[7] $O_{interface}$ means O_{12} for M_1 and O_{21} for M_2.

Proposition 2 : property-dependent bisimulation

There exists a biggest bisimulation included in $\mathcal{Q}$, denoted $\mathcal{B}(\mathcal{Q})$.

Proof : The strong bisimulation is included in $\mathcal{Q}$ as it preserves all CTL formulae and in particular it preserves π; moreover, as strong bisimulation imposes the values of the outputs of the FSM to be equal, it imposes it in particular for the outputs linking M_1 to M_2. Hence there exists a bisimulation included in $\mathcal{Q}$, it follows from proposition 2 that there exists a biggest bisimulation included in $\mathcal{Q}$, equals to $\mathcal{B}(\mathcal{Q}) = \bigcap_{n\geq 0} \mathcal{E}^n (\mathcal{Q})$. ❑

By definition of a bisimulation, $\mathcal{B}(\mathcal{Q})$ preserves the property π. Proposition 3 is necessary to prove that $\mathcal{B}(\mathcal{Q})$ preserves the synchronized product.

Definition 12 : sequence of values of linking signals

Let $\boldsymbol{M_1=(S_1,I_1,J_1,T_1,O_1,S_{o1})}$ *and* $\boldsymbol{M_2=(S_2,I_2,J_2,T_2,O_2,S_{o2})}$ *be two FSMs, the sequence of values of linking signals from* M_1 *to* M_2 *associated to a serie of reachable states* $c=(s_1 , s_2 , ...s_n)$ *(i.e.* s_{i+1} *is reachable from* s_i *and* s_1 *is reachable from* S_0 *in one transition) is the serie* $P(c) = (O_{12}(s_1) , O_{12}(s_2) , ..., O_{12}(s_n))$.

Proposition 3 : Let **M= (S,I,J,T,O,So)** be a FSM and π a static property over this FSM. $\mathcal{B}(\mathcal{Q})$ is the bisimulation induced by $\mathcal{Q}$ as defined in definition 10. Let $c=(s_1,....,s_i,......s_n)$ a path of **M** and $c'= (s'_1,......,s'_i,......s'_n)$ its equivalent path by $\mathcal{B}(\mathcal{Q})$ (i.e. such that $s_1\ \mathcal{B}(\mathcal{Q})\ s'_1$), then $P(c) = P(c')$.

Proof : As $s_1\ \mathcal{B}(\mathcal{Q})\ s'_1$, we have $O_{12}(s_1) = O_{12}(s'_1)$ and $s_2\ \mathcal{B}(\mathcal{Q})\ s'_2$ by definition of $\mathcal{B}(\mathcal{Q})$.
Hence by induction, if we assume $s_i\ \mathcal{B}(\mathcal{Q})\ s'_i$ then $O_{12}(s_i) = O_{12}(s'_i)$ and $s_{i+1}\ \mathcal{B}(\mathcal{Q})\ s'_i$.
Finally for all i $O_{12}(s_i) = O_{12}(s_{i+1})$. ❑

Corollary : Let M_1 be a FSM linked to a FSM M_2, and $M_{1/\mathcal{B}(\mathcal{Q})}$ its equivalent FSM induced by an equivalence relation $\mathcal{Q}$ as defined in definition 10, then M_1 and $M_{1/B(Q_1)}$ produce the same sequences of outputs connected to M_2.

Theorem 2:

$\mathcal{B}(\mathcal{Q})$ is an equivalence relation that preserves the property π and the synchronized product.

Proof : Let M_1 and M_2 be two FSMs linked together, and two static properties π_1 and π_2 such that π_1 concerns variables of M_1 and π_2 concerns variables of M_2. $\mathcal{B}(\mathcal{Q}_1)$ is the bisimulation induced by π_1 over M_1 and $\mathcal{B}(\mathcal{Q}_2)$ the bisimulation inducted by π_2 over M_2, then we will demonstrate that it is equivalent to verify the property $\pi_1 . \pi_2$ or $\pi_1 + \pi_2$ over M_1 **o** M_2 and over $M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)}$. To achieve this, we will demonstrate that there exists a bisimulation included in $\mathcal{R}_1 \subseteq S_1 \times S_1$: $x\ \mathcal{R}_1\ y \Leftrightarrow (x \models \pi_1 \Leftrightarrow y \models \pi_1)$ and $\mathcal{R}_2 \subseteq S_2 \times S_2$: $x\ \mathcal{R}_2\ y \Leftrightarrow (x \models \pi_2 \Leftrightarrow y \models \pi_2)$, between M_1 **o** M_2 and $M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)}$.

(1) Let $\mathcal{R}$ be an equivalence relation between states of M_1 **o** M_2 and $M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)}$ such that :

$$\forall (s_1,s_2) \in M_1 \mathbf{o} M_2 \text{ and } (s'_1 , s'_2) \in M_{1/B(Q_1)} \mathbf{o} M_{2/B(Q_2)}),$$
$$(s_1,s_2)\ \mathcal{R}\ (s'_1,s'_2) \text{ iff } s_1\ \mathcal{B}(\mathcal{Q}_1)\ s'_1 \text{ and } s_2\ \mathcal{B}(\mathcal{Q}_2)\ s'_2.$$

Let us prove that $\forall$ $(s_1,s_2) \in (M_1$ **o** $M_2)$, $\exists$ $(s'_1,s'_2) \in (M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)})$ such that :

$(s_1,s_2)\ \mathcal{R}\,(s'_1,s'_2)$:

a) $\forall\ s_1 \in M_1\ \exists\ s'_1 \in M_{1/B(Q_1)}$ such that $s_1\ \mathcal{B}(Q_1)\ s'_1$

b) $\forall\ s_2 \in M_2\ \exists\ s'_2 \in M_{2/B(Q_2)}$ such that $s_2\ \mathcal{B}(Q_2)\ s'_2$

c) Moreover, if (s_1, s_2) is reachable then (s'_1,s'_2) is reachable also, because of proposition 4 : the behaviours of the outputs of M_1 connected to M_2 or of M_2 connected to M_1 are the same in M_1 and $M_{1/B(Q_1)}$ and in M_2 and $M_{2/B(Q_2)}$)

Hence $\forall$ $(s_1,s_2) \in M_1$ **o** $M_2\ \exists\ (s'_1,s'_2) \in M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)})$ such that $(s_1,s_2)\ \mathcal{R}\,(s'_1,s'_2)$.

(2) Let $s = (s_1\ , s_2) \in M_1$ **o** M_2 and $sf = (sf_1\ , sf_2) \in M_1$ **o** M_2 such that $\exists\ a \in I_1$ **o** I_2 and $(s,a,sf) \in T_1$ **o** T_2 (i.e. there exists a transition labelled a_1 going from s_1 to sf_1 and a transition labelled a_2 going from s_2 to sf_2) . Le be $s' = (s'_1\ , s'_2) \in M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)}$) such that $s\ \mathcal{R}\,s'$, then $\exists$ $s'f \in M_{1/B(Q_1)}$ **o** $M_{2/B(Q_2)}$ such that $\exists\ a' \in I_1$ **o** I_2 and $(s',a',s'f) \in T_1$ **o** T_2.

s'f can be split into $(s'f_1\ , s'f_2)$ with $s'f_1 \in M_{1/B(Q_1)}$ and $s'f_2 \in M_{2/B(Q_2)}$ such that $a = (a_1, a_2)$ and $s'_1 \text{ -}a_1\!\rightarrow s'f_1$ and $s'_2 \text{ -}a_2\!\rightarrow s'f_2$.

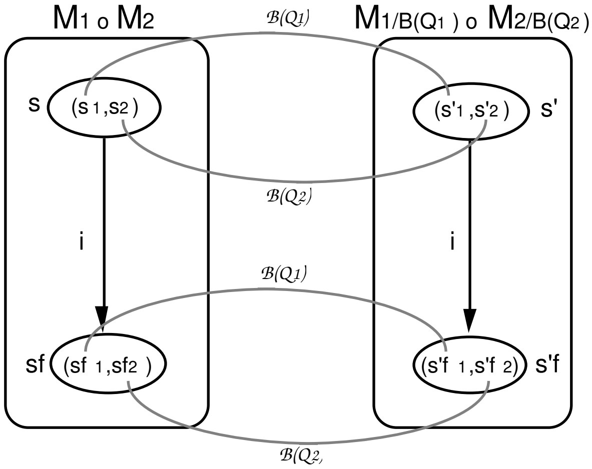

Figure 5. Definition of states and relations.

Indeed, as the behaviours of the outputs of M_1 linked towards M_2 and of M_2 linked towards M_1 are the same resp. for M_1 and $M_{1/B(Q_1)}$ and for M_2 and $M_{2/B(Q_2)}$, and if $s_1\ \mathcal{B}(Q_1)\ s'_1$ and there exists a_1 such that $s_1 \text{ -}a_1\!\rightarrow sf_1$, then there exists a'_1 : $s'_1 \text{ -}a'_1\!\rightarrow s'f_1$ and $a_1 = a'_1$. Similarly, if there exists a transition a_2 : $s_2 \text{ -}a_2\!\rightarrow sf_2$ then there exists a transition a'_2 : $s'_2 \text{ -}a'_2\!\rightarrow sf'_2$ and $a_2 = a'2$. Moreover, as $s_1\ \mathcal{B}(Q_1)\ s'_1$ and $s_2\ \mathcal{B}(Q_2)\ s'_2$ then $sf_1\ \mathcal{B}(Q_1)\ s'f_1$ and $sf_2\ \mathcal{B}(Q_2)\ s'f_2$.

To conclude, for all transitions t going from a state s of M_1 **o** M_2 to a state sf of M_1 **o** M_2, and for all states s' of $M_{1/B(Q_1)}$ o $M_{2/B(Q_2)}$ such that $s\ \mathcal{R}\,s'$, then there exists a state s'f of $M_{1/B(Q_1)}$ o $M_{2/B(Q_2)}$ reachable in one transition from s' and $sf\ \mathcal{R}\,sf'$. As $\mathcal{R}$ is an equivalence relation, this is sufficient to show that there exists a bisimulation $\mathcal{R}$ between M_1 o M_2 and $M_{1/B(Q_1)}$ o $M_{2/B(Q_2)}$.

Moreover, it is clear that $\mathcal{R}$ is included in $\mathcal{R}_1 \subseteq S_1 \times S_1$: $x\ \mathcal{R}_1\ y$ iff ($x \models \pi_1$ iff $y \models \pi_1$) and $\mathcal{R}_2 \subseteq S_2 \times S_2$: $x\ \mathcal{R}_2\ y$ iff ($x \models \pi_2$ iff $y \models \pi_2$), thus it is equivalent to verify $\pi_1 . \pi_2$ or $\pi_1 + \pi_2$.over M_1 o M_2 or over $M_{1/B(Q_1)}$ o $M_{2/B(Q_2)}$. ❑

3.4. Gain over the strong bisimulation

A question may subsist : what distinguishes the biggest bisimulation included in the defined equivalence relation from the strong bisimulation defined by [Milner 89] ?

We prove that in general $\mathcal{B}(Q)$ is coarser than the strong bisimulation, and even that $\mathcal{B}(Q)$ is the coarsest bisimulation preserving π (if π is static) and the synchronized product.

proposition 4 : In the general case, the strong bisimulation is strictly included in the equivalence relation we defined.

Proof : Let be $\mathcal{SB}$ the equivalence relation defined by : x $\mathcal{SB}$ y $\Leftrightarrow$ O(x) = O(y), Q defined as in definition 9, and the application $\mathcal{E}$ defined as in definition 11, then, in the general case :

	$\mathcal{SB} \subset Q$
$\Rightarrow$	$\mathcal{E}(\mathcal{SB}) \subset \mathcal{E}(Q)$ (because $\mathcal{E}$ is monotonic and strictly decreasing),
then	$\lim_{n\to\infty} \mathcal{E}(\mathcal{SB}) \subset \lim_{n\to\infty} \mathcal{E}(Q)$ as sets have finite topology,
and	$\lim_{n\to\infty} \mathcal{E}(\mathcal{SB})$ is the strong bisimulation, while $\lim_{n\to\infty} \mathcal{E}(Q)$ is $\mathcal{B}(Q)$ ❑

Instead of preserving all paths modifying a value of any output of a FSM, as strong bisimulation does, the relation we defined preserves the paths modifying the truth values of the static property to be verified and the interconnection outputs. Thus if we want to verify the property "x + y" over a component M with J = {x,y,z,t}, and t is a connection signal, the strong bisimulation will distinguish all the configurations of x, y, z and t (i.e. sixteen configurations) along the executing paths, while the relation we defined will distinguish the configurations where "x + y" is true or false, with the value of t (i.e. four configurations are considered[8] : (x + y).t, (x+y).!t, !(x + y).t and !(x+y).!t).

4. EXTENSION TO ALL CTL OPERATORS

Up to now we had been considering the global property π to be a static property of the form $\pi = \pi_1 . \pi_2$ or $\pi = \pi_1 + \pi_2$, with π_1 a Boolean proposition referring to outputs of M_1 and π_2 to these of M_2. The following result extend the type of property that are preserved by our reduction.

Theorem 3:

Let M_1 and $M_{1/B(Q1)}$ two FSMs equivalent by $\mathcal{B}(Q_1)$. Let π_1 a static property concerning variables of M_1, then for all CTL operators, (i.e. AX, EX, AF, EF, AG, EG, AU, EU), it is equivalent to verify the property C(π_1) over M_1 or over $M_{1/B(Q1)}$.

Sketch of the proof : Let P the set of states of M_1 verifying C(π_1) and let P' the set of states of $M_{1/B(Q1)}$=(S'$_1$,I'$_1$,J'$_1$,T'$_1$,O'$_1$,S'$_{o1}$) verifying C(π_1) also. We prove by contradiction and for each operator : $\forall$ s $\in$ S_1, such that s $\in$ P, then s' $\in$ S'$_1$ such that s $\mathcal{B}(Q_1)$ s iff s' $\in$ P'. ❑

Corollary :

Let $\mathbf{M_1 = (S_1,I_1,J_1,T_1,O_1,S_{o1})}$, $\mathbf{M_2 = (S_2,I_2,J_2,T_2,O_2,S_{o2})}$ two FSMs, and π_1 and π_2 two static properties concerning resp. variables of M_1 and M_2. Let $\mathcal{B}(Q_1)$ the equivalence relation induced

[8] the symbol ! represents the boolean negation of the expression

by π_1 and $\mathcal{B}(Q_2)$ the one induced by π_2. Then for each CTL operator, it is equivalent to verify C(π_1 . π_2) or C($\pi_1 + \pi_2$) over M_1 **o** M_2 or over $M_{1/B(Q1)}$ **o** $M_{2/B(Q2)}$.

5. ALGORITHM FOR THE REDUCTION OF A COMPONENT

5.1. Computation of the biggest bisimulation of a FSM induced by a relation

Let π be a static property (cf. 3.4.1) over the outputs of a FSM M, and a relation $\mathcal{R}$ over the states of M such that : x $\mathcal{R}$y iff : ($x \models \pi \Leftrightarrow y \models \pi$).

The algorithm presented by [BFH 94] computes the transition relation representing the quotient FSM by the biggest bisimulation induced by the relation $\mathcal{R}$, from the transition relation of the initial FSM. The algorithm exactly computes the relation $\mathcal{R}f = \bigcap_{n\geq 0} \mathcal{E}^n (\mathcal{R})$. The method is based on an iterative refinement of a partition of the states of the FSM, separating the non bisimilar states into disjoint equivalence classes, up to the stabilization of the classes. The advantages of their algorithm is that it computes on the fly the quotient FSM, it only refines reachable classes, and symbolic representation of sets of states and transition relation can be applied.

In their paper, [BFH 94] only focus on the reduction of a single FSM according to a static property π. Thus the initial partition of the set of states is composed of two classes : the one containing the states verifying π and the one containing the states not verifying π. Then the class containing the initial state is refined : its states leading to states of different classes are distinguished, and then the refinement of this class induces new classes; the reachable ones are, in their turn, refined ... and so on, up to the stabilization of all classes.

5.2. Extension of the algorithm to handle property-dependent bisimulation

This algorithm can be applied to compute the biggest bisimulation included in Q defined as in definition 10 of a given FSM : the only change concerns the initial partition.

If $\pi = \pi_1 . \pi_2$ or $\pi = \pi_1 + \pi_2$, where π_1 contains variables of M_1 and π_2 contains variables of M_2, and M_1 and M_2 are connected through J_{12} outputs from M_1 connected to M_2 and through J_{21} outputs from M_2 connected to M_1, then the reduction of M_1 and M_2 can be built by computing the biggest bisimulation $M_{1/B(Q1)}$ and $M_{2/B(Q2)}$ with :

For the module M_1, $\forall$ (x,y) $\in S_1 \times S_1$, x Q_1 y $\Leftrightarrow$

(i) $x \models \pi_1 \quad \Leftrightarrow y \models \pi_1$

(ii) $O_{12}(x) = O_{12}(y)$

The initial partition of the computation algorithm of $M_{1/B(Q1)}$ is then $\bigcap C_i$, where :

$C_i = \{ s \in M_1 \mid s \models$ configuration[9][10]$(\pi_1, o_1,....o_n)$ such that $o_j \in J_{12}\}$

---

[9] configuration(x_1,...,x_n) is the set of all tuples (x_1,...x_n) for all values of x_1, ... x_n. Its cardinality is 2^n.

[10] if the formula contains p terms refering to variables of M_1, all these terms are considered in the initial partition by extending the set of configurations to configuration(π_{11}, ..., π_{1p}, o_1, ..., o_n)

Once the components are reduced, we can form their synchronized product and prove a CTL property whose atomic proposition is π.

We can extend the type of CTL property to be verified once its atomic propositions are identified and combined as sum or products of static propositions of M_1 and M_2.

<u>Example</u> : Assume $\pi = (\pi_{11} . \pi_{21}) + AF(\pi_{12}+ \pi_{21})$, with π_{11} and π_{12} referring to outputs of M_1 and π_{21} and π_{22} referring to M_2, then

two states (x_1,y_1) of M_1 are bisimilar by π iff they are bisimilar by π_{11} , π_{12} and J_{12}.

two states (x_2,y_2) of M_2 are bisimilar by π iff they are bisimilar by π_{21} , π_{22} and J_{21}.

As one can see, the initial partition of the algorithm for the reduction of a given component grows exponentially with (1) the number of interconnection signals going from this component to the others implicated in the global formula, and (2) the number of terms referring to variables of this component in the atomic propositions of the formula.

<u>6. APPLICATION TO VHDL AND EXPERIMENTAL RESULTS</u>

We implemented a compositional reduction routine for interacting VHDL structural programs and applied it to few number of academic examples. The software tool is still a prototype developed to experimentally confirm the correctness of our approach. The experimental results below indeed provide such a confirmation. For these examples, it is equivalent to verify the property on the product of the interacting VHDL modules or on the product of reduced interacting VHDL modules where each module was reduced by our formula-dependent equivalence.

6.1. VHDL example

Let be the following VHDL entity :

```
entity prgm_grenoble is
port(
       ck    : in bit;
       a     : in bit
        );
end prgm_grenoble;

architecture behaviour of prgm_grenoble is

begin

prgm :  process

       variable y  : bit ;
       variable z  : bit ;
       variable x  : bit := '1';
       variable w  : bit := '1';

       begin
             z  := y;
              y  := (x and w) or a;
              x  := not z;
              w  := (not w and x) or y;
              wait on ck;
       end process prgm ;

end behaviour;
```

At the end of each simulation cycle, a state of the entity is composed of :

- the signal effective values **ck** and **a**,
- the event attribute of the signal ck : **evt_ck**,
- the values of the variables of the process prgm : **y**, **z**, **x**, **w**.

The initial state is the one after the initialisation cycle (once the initial values were given, an execution phase has occurred): **<!evt_ck.!ck.!a.!z. y. x. w>**

The transition relation is computed by VPN :

R1 := evt_ck'.(ck ≡ ck') + !evt_ck'.(ck = ck');

R2 := a.evt_ck.w' + a.!evt_ck.w.w' + a.!evt_ck.!w.!w' + !a.evt_ck.(w.w'.x + w.!w'.!x) +!a.evt_ck.(!w.w'.!y + !w.!w'.y) + !a.!evt_ck.(w = w');

R3 := evt_ck.y.!x' + evt_ck.!y.x' + !evt_ck.x.x' + !evt_ck.!x.!x' ;

R4 := a.evt_ck.y' + a.!evt_ck.y.y' + a.!evt_ck.!y.!y' + !a.evt_ck.w.y'.x + !a.evt_ck.w.!y'.!x + !a.evt_ck.!w.!y' + !a.!evt_ck.y.y' + !a.!evt_ck.!y.!y';

R5 := evt_ck.z'.y + evt_ck.!z'.!y + !evt_ck.z.z' + !evt_ck.!z.!z';

R := R1 . R2 . R3 . R4 . R5;

The reachable state space is computed by symbolic forward traversal. It contains 32 states. The initial state is the one below located on the left.

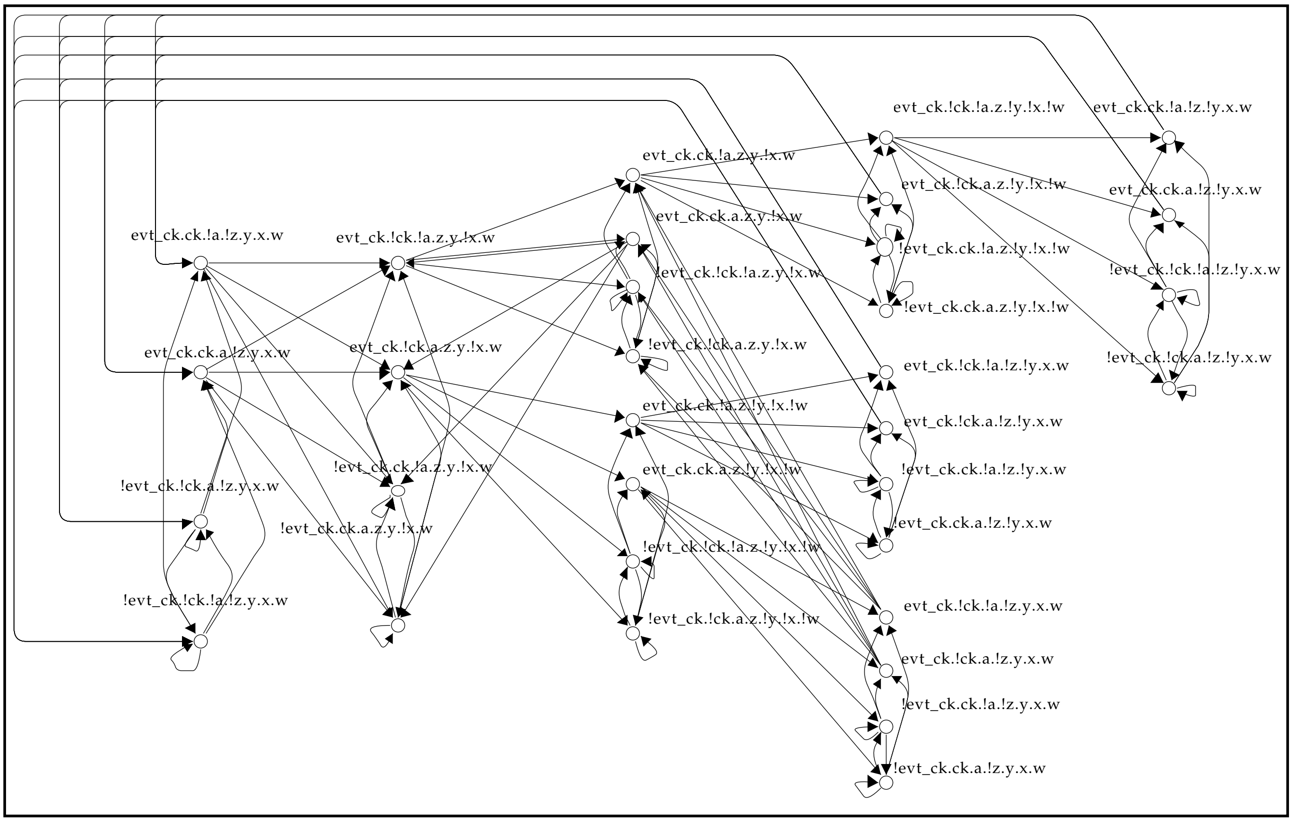

Figure 6. The reachability graph of the VHDL program.

Assume we want to verify a CTL property whose atomic proposition is "x + y". The transition relation can be reduced by computing the biggest bisimulation included in $\mathcal{R}$:

$\forall\ (s_1,s_2) \in S \times S\ ,\ s_1\ \mathcal{R}\, s_2\ \Leftrightarrow (s_1 \models x + y\ \Leftrightarrow s_2 \models x + y)$

This leads to a FSM of seven classes, depicted below.

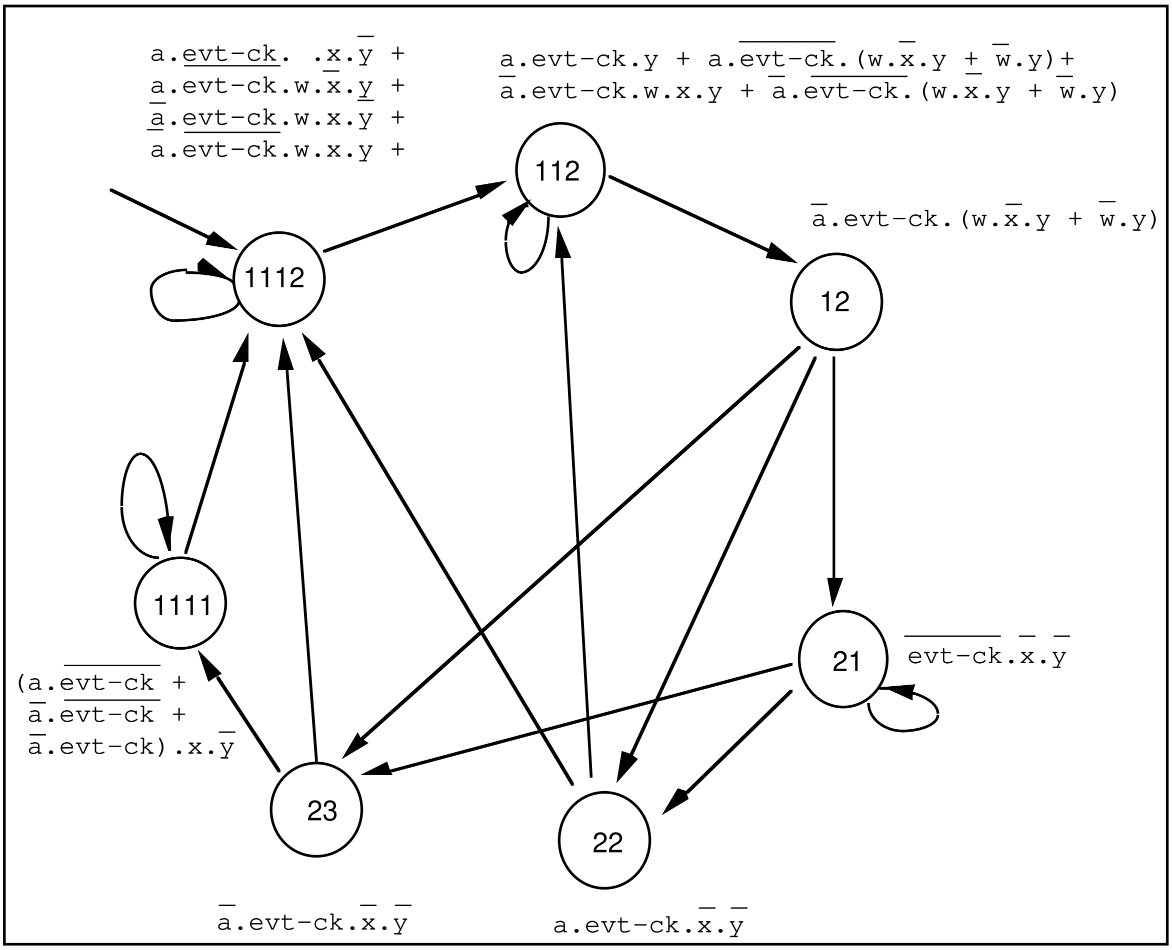

Figure 7. The reduced graph of the VHDL program.

Each reachable state belongs to a given class. The states in whom "x + y" is true are in class 1111, 1112, 112 or 12, while the states in whom "x + y" is false are in class 21, 22 or 23. Notice that all classes represented are reachable, but a class may contain unreachable states. Reachable and unreachable states in a given class are equivalent for the property.

6.2. Experimental results

These experiments were performed using BDDs. We used the information about the equivalent states to simplify the BDDs representing the transition relation but we were unable to compute our approach to large industrial designs, may be in part due to the fact that BDD minimisation routines were not used. Nevertheless we have found large reduction in the number of states for small design. More experiments need to be performed to evaluate the contribution of each parameter in the reduction factor.

name of design	number of states	number of variables implied in the formula	number of connected outputs	number of equivalence classes in the quotient automaton	reduction factor
prgm_grenoble	32	1	0	5	6.4
prgm_grenoble	32	2	0	5	6.4
prgm_grenoble	32	5	0	9	3.2
prgm_grenoble	32	2	1	7	4.3
prgm_grenoble	32	2	2	9	3.2
part	640	1	2	7	90
part	640	2	2	23	29
part	640	5	2	54	12
arbiter2	2260	1	0	13	174
arbiter2	2260	2	0	19	114
arbiter2	2260	2	1	20	110
arbiter2	2260	2	2	27	88

Table 1. Size of the initial and reduced FSMs of VHDL entities.

The initial automaton contains n states encoded over p Boolean variables. Depending on the formula and on the interconnection, its quotient automaton is composed of q equivalences classes, each one grouping several indistinguishable states. Thus the quotient automaton may be encoded on a smaller number of Boolean variables since q << n, depending on the number of equivalence classes, simplifying the composition and further verification.

7. CONCLUSIONS AND PERSPECTIVES

This report presents a step towards compositional model checking based on the reduction of each component before its connection. The equivalence relation we define in order to reduce each component preserves enough information to insure the correctness of the verification : it preserves the property to be verified and the synchronised product, which is the basic operation of combining VHDL components. We have shown that the formula-dependent equivalence relation defined by [ASSS-V 94] in not adequate for VHDL programs, since (i) it cannot handle specified initial states and (ii) it is not a bisimulation. In addition, with our approach, we can compute each reduced FSM independently and reuse this reduced FSM even if the other components connected to this one have been modified (but assuming the same interface).

The component reduction procedure has been implemented and tested. Our experimental results confirm the fact that significant reduction factors may be obtained, while suggesting that they depend a lot on the size of the FSMs and on the structure of the property to be checked. We therefore plan to extend our existing verification platform [BE 96b] by introducing the *a priori* reduction of each component for model cheking and equivalence verification purposes.

A fruitful avenue for future research is the study of the partitioning of a given structural VHDL program in order to yield a structural program where each module is as loosely connected to the other modules as possible, in other words to reduce the number of interconnection between FSMs. Another subject of future research concerns the introduction of component reduction in the architectural synthesis process.

REFERENCES

[Arnold 92] A. Arnold, *"Systèmes de transitions finis et sémantique des processus communicants", Masson* , France, 1992 (book in french)

[ASSS-V 94] A. Aziz, T. Shiple, V. Singhal, A.Sangiovani-Vincentelli, *"Formula-Dependent Equivalence for Compositional Model Checking* ", proceedings of CAV 94, CA 1994.

[Bawa 96] R.K. Bawa, "*Un environnement intégré pour la vérification formelle et l'analyse des systèmes décrits en VHDL*", PhD, University Paris VI, dec 96.

[BE 96] R.K. Bawa, E. Encrenaz, *"A Platform for the Formal Verification of VHDL Programs"*, Actes de SMACD'96 (4th International Workshop on Symbolic Methods and Applications in Circuit Design), Heverlee, Belgique, oct. 1996.

[BFH 94] A. Bouajjani, J-C. Fernendez, N.Halbwachs, P. Raymond, and C. Ratel "*Minimal state graph generation",* Science of Computer Programming, 18(3), pp 247-271, 1994.

[Borrione 96] D. Borrione, "*Research on VHDL in France, Italy and Switzerland*", in Proceedings of the VHDL International Users' Forum, spring Conference, feb-mar 1996, Santa Clara, CA.

[CCP 93] P. Camurati, F. Corno, P. Prinetto, "*An efficient tool for system-level verification of behaviors and temporal properties*", in proc of EURO-DAC'93 : IEEE European Design Automation Conference, sept. 93, Hamburg, Germany.

[CLM 89] E. Clarke, D. Long, K. McMillan, " *Compositional Model Checking*", proc 4th Symp on Logic in Computer Science, Asilomar, CA, jun 89.

[CLSI 93] CLSI Solutions, " *VFORMAL Users' Manual*", May 93.

[Encrenaz 95a] E. Encrenaz, "*Une méthode de vérification de propriétés de programmes VHDL basée sur des modèles formels de réseaux de Petri"*, PhD, University Paris VI, dec 95.

[Encrenaz 95b] E. Encrenaz, "*A symbolic Relation for a subset of VHDL'87 description and it's application to symbolic model checking"*, proc. of CHARME'95 , Frankfurt, Germany , oct.95, LNCS 987.

[HP 90] J. Hennesy, D. Patterson, "Computer Architecture, a Quantitative approach", Morgan Kaufman Publishers Inc, San Mateo, CA,1990.

[LL 95] F. Laroussinie, K. G. Larsen, "*Compositionnal model checking of real-time systems*", proceedings of CONCUR'95, LNCS 962.

[LRM 87] "*IEEE Standard VHDL Language Reference Manual*" IEEE Std 1076-1987.

[McMillan 93] K. McMillan, *"Symbolic Model Checking",* Kluwer Academic Publisher, Norwell Massachusetts, 1993.

[SR 94] R. de Simone, A. Ressouche,*"Compositional semantics of ESTEREL and verification by compositional reductons".* in proc. of CAV 94, CA, jun. 1994, pp 441-454.

[Milner 89] R.Milner, " *Communication and Concurency* ", Prentice Hall, New York, 1989.

[VIS 95] R.K.Brayton et. al. "*Vis : a System for Verification and Synthesis* " , UC Berkeley Electronics Research Lab, Technical Report No: UCB/ERL M95/104, dec. 1995.