Hierarchical Gate-Level Verification of Speed-Independent Circuits

Oriol Roig, Jordi Cortadella and Enric Pastor *
Department of Computer Architecture
Universitat Politècnica de Catalunya
Gran Capità s/n, Mòdul D6, 08071-Barcelona, Spain

Abstract

This paper presents a method for the verification of speed-independent circuits. The main contribution is the reduction of the circuit to a set of complex gates that makes the verification time complexity depend only on the number of state signals (C elements, RS flip-flops) of the circuit.

Despite the reduction to complex gates, verification is kept exact. The specification of the environment only requires to describe the transitions of the input/output signals of the circuit and is allowed to express choice and non-determinism. Experimental results obtained from circuits with more than 500 gates show that the computational cost can be drastically reduced when using hierarchical verification.

1 Introduction

Asynchronous circuits can be considered as a practical alternative to face some of the critical problems that appear when designing complex, low power, high performance digital systems.

The clock signal in synchronous circuits enables to introduce a level of abstraction in the time domain and overlook most temporal relations among the signals of the circuit. Only the concept of *critical path* is relevant for the performance of the system but not for its functional correctness. Unfortunately for the designer, the absence of a clock in asynchronous circuits makes their design an error-prone task. Most difficulties come from the need to ensure that all signals are free of undesirable transitions, *hazards*, that can produce circuit malfunctions.

The additional complexity introduced by the analysis of the temporal relations makes verification essential for asynchronous circuits. But while only the outputs of memory elements, e.g. flip-flops, are required to represent the state of a synchronous circuit, the output of all nodes (gates and memory elements) must be probed to define the state of an asynchronous circuit. Given that, in the worst case, the size of the state space can be $O(2^n)$, n being the number of signals to define the state, this space can become extremely large even for moderate size asynchronous circuits.

Several authors have proposed verification techniques to avoid the explicit enumeration of all the states: unfoldings [11], partial orders [14], symbolic model checking [4] and trace theory [6] among others.

This paper presents sufficient conditions to automatically reduce the complexity of the circuit to be verified for speed-independence. The proposed method aims at the reduction of the number of variables required for verification. It has been combined with symbolic model checking techniques to efficiently represent the state space of the circuit.

1.1 Contributions

The method presented in this paper aims at the verification of gate-level speed-independent circuits. Beerel et al. [2] observed that, if the verifier were told by some oracle that the circuit is hazard-free, checking its correctness against its specification could be reduced, roughly speaking, to perform a verification à la synchronous with only the outputs of the memory elements (e.g. C-elements or RS flip-flops) as state variables. Based on this observation, our approach verifies correctness in two steps: (1) satisfiability of the specification assuming the absence of hazards and (2) hazard detection. The major contributions of this paper are the following:

1. The circuit, a flat netlist of gates, is *automatically reduced* to a set of complex gates. The time complexity of verification is made dependent on the number of memory elements rather than on the number of signals of the circuit.

2. Even with the reduction to complex gates, verification is kept exact, i.e. neither false positive nor false negative verification results are possible.

3. The environment is described by a state graph that only needs to contain the transitions of the input/output signals of the circuit. Choice and non-determinism of the environment are allowed.

The paper is organized as follows. Section 2 discusses the basic ideas of hierarchical gate-level verification by means of an example. Section 3 presents some basic definitions used along the paper. Section 4 analyzes the conditions under which exact hierarchical verification can be performed. Section 5 discusses the most significant implementation issues of our verifier. Comparative results between flat and hierarchical verification are presented in Section 6. Finally, Section 7 concludes the paper.

*Work supported by ACiD-WG (Esprit 7225), CYCIT TIC 94-0531-E and Departament d'Ensenyament de la Generalitat de Catalunya.

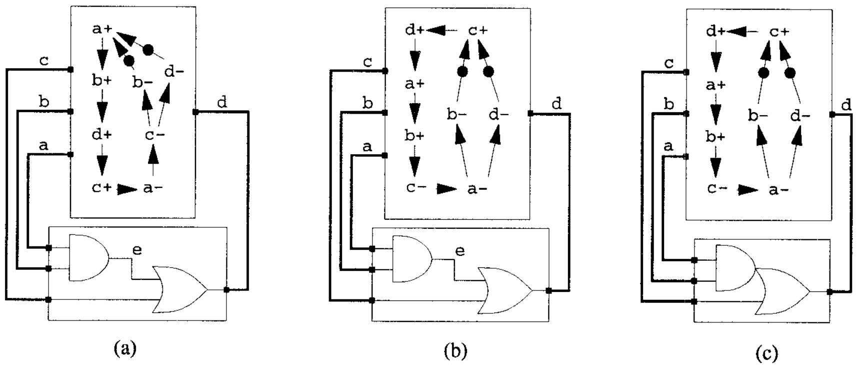


Figure 1: (a) Circuit with a hazard-free behavior, (b) The same circuit with a hazardous behavior, (c) Equivalent hazard-free complex-gate circuit.

2 Hierarchical verification: overview

This section presents hierarchical verification by means of two examples. In this section, speed-independence will be considered equivalent to hazard-freeness under the unbounded gate delay model. More precise definitions will be given in Section 3.

Speed-independence is not a property of a circuit by itself but of the behavior of a circuit under a certain environment. In our framework, the behavior of the environment will be represented by a *Signal Transition Graph* [5] in which the output signals will be inputs of the circuit and vice versa. Figures 1.(a) and 1.(b) depict a circuit excited by two different environments. The circuit is hazard-free with environment (a), but hazardous with environment (b). In the latter case, a static hazard can be produced on signal d when, in the state $(abcde) = (11110)$, the event $c-$ arrives before e has switched to 1. However, note that an equivalent complex-gate implementation of the same circuit (Figure 1.(c)) can be hazard-free.

Figure 2 shows the state graph obtained by a reachability analysis of the system in Figure 1.(a). In the worst case, the number of states can be as large as 2^n, n being the number of signals of the circuit. Verification through reachability analysis [4] would simply check that each transition produced at the outputs of the circuit can be accepted by the environment, i.e. a transition with the same label is enabled in the STG.

2.1 Functional and behavioral correctness

Two important concepts must now be introduced to set up the basis of verification: *functional correctness* and *behavioral correctness.*

A circuit is said to be *functionally correct* if an appropriate combination of the delays of its components can produce the behavior expected by the environment.

A functionally correct circuit is said to be *behaviorally correct* if it produces the behavior expected by the environment regardless the delay assigned to each component, provided that the delays are within the margins assumed for the delay model and the technology. For speed-independent circuits, delays are in the range $(0, \infty)$.

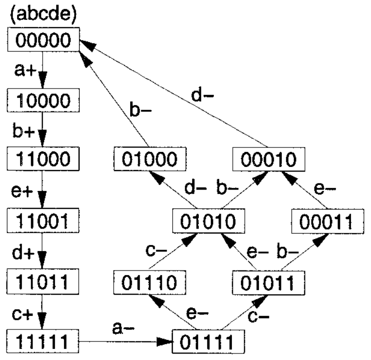


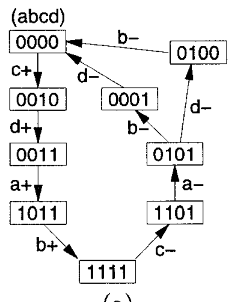
Figure 2: State Graph of the circuit in Figure 1.(a).

We can say now that the circuit in Figure 1.(b) is functionally correct, since by assigning the AND gate a delay shorter than the delay between the transitions $b+ \rightarrow c-$, the generated behavior is the one expected by the environment. However, this circuit is behaviorally incorrect, since long delays on the AND gate may produce a static hazard on d. The circuit in Figure 1.(c) is both functionally and behaviorally correct (the complex-gate architecture basically assumes zero delay for the AND gate).

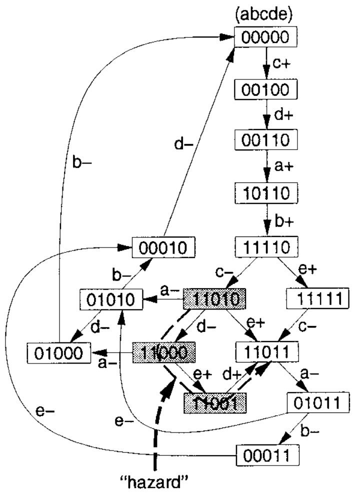
2.2 Verification of functional correctness

A speed-independent circuit must behave correctly for any finite delay of its components. A particular case consists in "moving" the delay of a gate to its fan-out gates. Let us take as example the circuit in Figure 1.(b). We can move the delay of gate e to its fan-out gate d, and we obtain the complex gate in Figure 1.(c). We refer to this kind of gate clustering as *collapsing.* Since all the delays are in the range $(0, \infty)$, the sum of the delays of d and e is still in the same range. The behavior of the output of a complex gate is included in the behavior of the original circuit (prior to collapsing).

In our framework, functional correctness is verified



(a)



(b)

Figure 3: (a) State graph after verification of functional correctness. (b) State graph after verification of behavioral correctness.

by collapsing some of the gates of the circuit. In this way, multiple gates can be collapsed into one complex gate and, thus, internal signals eliminated for the verification. Only for memory elements (e.g. C elements) or outputs of complex gates, the signals cannot be eliminated.

Verification of functional correctness becomes simpler and faster because of the elimination of internal signals. Moreover, design errors that do not depend on the delays of the gates can be detected soon, without requiring an exhaustive verification of the temporal relations among all signals.

In the example of Figure 1.(b), functional correctness is verified by first collapsing the AND and OR gates into one complex gate and eliminating signal e. Next, the state graph of the circuit/environment is built and verified for correctness (Figure 3.(a)).

2.3 Verification of behavioral correctness

In general, large circuits will be collapsed into several complex gates. As illustrated in Figure 4, this can be done hierarchically according to efficiency criteria for verification.

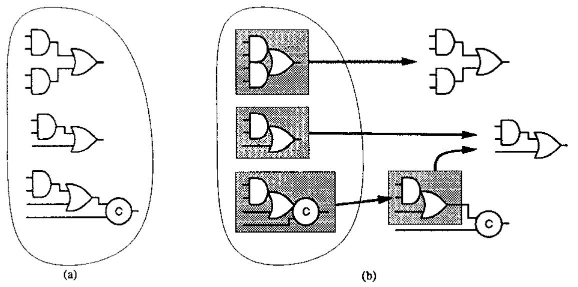


Figure 4: (a) Flat circuit (8 gates). (b) Hierarchical complex-gate organization (3 complex gates).

The second step of verification is devoted to detect hazards inside the complex gates. Intuitively, this is performed as follows. Given a complex gate, the state graph of the collapsed circuit is projected onto the input/output signals of the complex gate. This projection maps all states with the same values for the input/output signals of the complex gate (even if they are semantically different) onto the same state. We will show that this apparent loss of environmental information is not relevant for the verification of hazard-freeness. Finally, the complex gate is verified to be hazard-free under the projected graph as environment.

Isomorphic groups of gates can be mapped onto the same complex gate. In the example in Figure 4 there is a pattern repeated twice: an OR gate which inputs are an AND gate and a primary input. We will show in Section 4 that we can project the environment of several complex gates onto one single state graph and verify they hazard-freeness at a time. This hierarchy allows us to verify isomorphic subcircuits together.

It is important to notice that the environment of each complex gate is calculated as if it were hazard-free. In Section 4 we will show that, even with this restricted environment, hazard-freeness can be *exactly* verified.

Figure 3.(a) shows the state graph of the collapsed circuit. By chance, this graph coincides with its projection onto the signals $\{a, b, c, d\}$ as the whole circuit has been collapsed into one complex gate. When generating the state graph of the complex gate (Figure 3.(b)), an unexpected transition ($d-$) is detected in state 11010, since the corresponding state of the environment (1101) can only accept transition $a-$.

In case the complex gate were verified to be hazard-free, its corresponding state graph would be projected onto the input/output signals of its components and the same operation would be performed at the next level of the hierarchy. This is illustrated in Figure 5 that depicts the environment of the AND gate after projecting the graph of Figure 3.(b) onto the signals $\{a, b, e\}$[1].

[1]This environment is only depicted as an example, since there is not need to derive it for simple gates or for gates contained in hazardous complex gates.

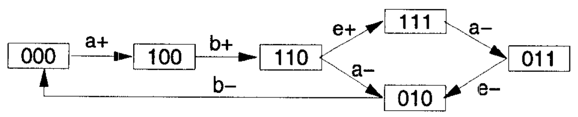


Figure 5: Environment of the AND gate after the projection of the state graph onto the signals a, b and e.

Only one question remains to be answered: why is hierarchical verification exact? In our framework, the absence of hazards is proved by verifying that the circuit is *semi-modular*, i.e. no gate can be disabled by changing the value of its inputs. Let us assume that C is a circuit and $\widehat{C}$ is an equivalent circuit in which some gates have been collapsed into complex gates and the corresponding internal signals eliminated. In Section 4 we will prove that:

a) if $\widehat{C}$ is not semi-modular, then C is not semi-modular either.

b) if $\widehat{C}$ is semi-modular but C is not semi-modular, there is a complex gate of $\widehat{C}$ for which the behavior of the corresponding decomposed gate, under the projection of the state graph of $\widehat{C}$ onto the input/outputs of the gate, is not semi-modular.

Conjecture a) guarantees *no false negatives*, whereas conjecture b) guarantees no *false positives*.

2.4 Why is hierarchical verification more efficient ?

A critical factor that determines the complexity of verification is the number of signals of the circuit. With hierarchical verification the number of signals relevant at each step of the verification is drastically reduced: during verification of functional correctness only the input/output signals of the complex gates are required; during verification of behavioral correctness of a complex gate only the interface and internal signals of the gate are required.

There is only one limit to the minimum number of variables required for functional verification: the number of output signals of the memory elements, such as C-elements or RS flip-flops.

2.5 Related work

In this section, some of the most relevant efforts related with the verification of speed-independence and closest to the approach described in this paper are presented.

Hierarchical verification

In his thesis [6], Dill already proposed hierarchical verification of speed-independence: if a component *conforms to* a trace structure, the behavior of that component can be safely substituted by the trace structure.

However, this approach requires the designer to identify the basic components of the circuit and know their expected behavior in advance. This makes the approach impractical when a flat netlist of gates, with no explicit hierarchical organization, must be verified.

Following Dill's approach, a subset of gates of the flat circuit (potentially substitutable by a complex gate) should be substituted by an equivalent trace structure. Not knowing how the environment of the complex gate will be inside the circuit, the trace structure should consider all possible input/output transitions and, therefore, include the state of all internal signals, which would preclude the subset of gates to be handled as a complex gate.

Conservative verification

Beerel et al. [2] also propose a two-step approach. After verifying the circuit is *complex-gate equivalent* to its specification, hazard-freeness is verified by subsequently checking the monotonicity and acknowledgment of all signal transitions. A cube approximation that overestimates the set of states of the circuit is proposed to conservatively prove the absence of hazards. Although never found in the examples presented by the authors, *false negatives* are theoretically possible. Other limitations of this approach are that it is limited to externally-cut circuits (all memory elements must appear in the specification) and that the specification of the circuit is not allowed to express output choice (arbitration).

Polynomial methods for signal graphs

Kishinevsky et al. [7] presented a polynomial algorithm to verify *distributivity* (a subclass of speed-independence) from circuit behaviors described by *signal graphs*. The main limitation of their approach is that the signal graph must specify the transitions of all signals of the circuit and that neither choice nor non-determinism are allowed in the signal graphs.

3 Definitions

We will consider a circuit to be a set of gates connected to an environment. The behavior of the environment will be modeled by means of a state graph. In our verifier, the state graph is derived from a Signal Transition Graph that describes the interaction of the environment with the input/output signals of the circuit. Thus, environments with choice and non-determinism are allowed.

Definition 3.1 (Circuit) *A* circuit *is a pair $C = \langle A, F\rangle$, where $A = \{a_1, ..., a_n\}$ is a set of signals ($n = |A|$) and F maps each signal $a_i \in A$ to a boolean function f_i of arity n, that represents the function computed by the gate that drives a_i.*

Definition 3.2 (Fan-in and fan-out of a signal) *The* fan-in *of signal $a_i \in A$, $fanin(a_i) \subseteq A$, is the set of signals that f_i depends on. For gates that hold state, $a_i \in fanin(a_i)$. The* fan-out *of signal $a_i \in A$, $fanout(a_i) \subseteq A$, is the set of signals that depend on a_i, i.e. $fanout(a_i) = \{a_k \in A | a_i \in fanin(a_k)\}$.*

Definition 3.3 (State graph) *A* state graph *(SG) is a 4-tuple, $\langle A, S, E, \lambda \rangle$, where $A = \{a_1, \ldots, a_n\}$ is the set of signals, S is the set of states, $E \subseteq S \times S$ is the set of transitions and λ is the labeling function for states that maps each state with a bit-vector over A.*

The fact that $(s, s') \in E$ will be also denoted by sEs'. E^* denotes the transitive closure of E, and sE^*s' denotes that there is a path from state s to state s' in the state graph. In those cases in which the labeling function is the identity, the state graph will be denoted simply as $\langle A, S, E \rangle$.

Definition 3.4 State graph of a circuit *The state graph of a circuit $C = \langle A, F \rangle$ with initial state s^0 is a state graph, $SG(C, s^0) = \langle A, S, E \rangle$, such that S and E are strictly defined by the following recursion:*

1. $s^0 \in S$.
2. $[(s \in S) \wedge (\forall_{i \neq k} s'_i = s_i) \wedge (s'_k \neq s_k) \wedge (s'_k = f_k(s))] \Longrightarrow [(s' \in S) \wedge (s, s') \in E]$.

Relation E can be partitioned into n subsets as follows:

$$\begin{aligned} E_i &= \{(s, s') \in E | s_i = \overline{s}'_i\} \ , \\ E &= \bigcup_{a_i \in A} E_i \ . \end{aligned}$$

Note that the labeling function λ is the identity. This means that each state $s \in S$ is a bit-vector over A such that the ith element of s, denoted by s_i, specifies the value of signal a_i in state s.

Given a state $s \in S$, if there exists $s' \in S$ such that sE_is' we will say that signal a_i is *excited* in state s. Otherwise we will say that a_i is *stable* in s.

Definition 3.5 (Projection of the state graph of a circuit) *Given the state graph of a circuit, $SG(C, s^0) = \langle A, S, E \rangle$, and a subset of signals $X \subseteq A$, the projection of $SG(C, s^0)$ onto X is a state graph, $proj_X(SG(C, s^0)) = \langle X, proj_X(S), proj_X(E) \rangle$ defined as follows:*

$\forall s = (s_1, \ldots, s_n) \in S$, $proj_X(s) = (s_1, \ldots, s_k)$, i.e. the sub-vector of s containing only the signals in X (we assume $|X| = k$ and X to be the first k elements of A),

$$\begin{aligned} proj_X(S) &= \{s' | \exists s \in S : proj_X(s) = s'\} \ , \\ proj_X(E) &= \{(proj_X(s), proj_X(s')) | sE^*s' \\ &\quad \text{and only one signal in } X \\ &\quad \text{transitions from } s \text{ to } s'\} \ . \end{aligned}$$

Note that the definition of a state as a bit-vector implies that semantically different states can be projected onto the same state (i.e. $proj_X(s) = proj_X(s') = \widehat{s}$ and $s \neq s'$).

The following proposition is a result of the previous definition.

Proposition 3.1 *Let $C = \langle A, F \rangle$ be a circuit, $SG(C, s^0) = \langle A, S, E \rangle$ its state graph, and $fanin(a_i) \cup \{a_i\} \subseteq X \subseteq A$. Let $proj_X(SG(C, s^0)) = \langle X, \widehat{S}, \widehat{E} \rangle$ be the projection of $SG(C, s^0)$ onto X. Let $s, s' \in S$ and $\widehat{s} \in \widehat{S}$ such that $proj_X(s) = proj_X(s') = \widehat{s}$. Then*

$$a_i \text{ excited in } s \Leftrightarrow a_i \text{ excited in } s' \Leftrightarrow a_i \text{ excited in } \widehat{s} \ .$$

Proposition 3.1 is crucial for our method, since it states that the excitation/stability of a complex gate (and subsequently semi-modularity) can be locally checked by only knowing the values of the input/output signals of the gate and regardless the state of the rest of the circuit.

Without loss of generality and for the sake of simplicity, we will consider ***autonomous circuits***, i.e. with no interface, for verification. The obtained results can be naturally extended to circuits with interface.

Next, observational equivalence [12] is defined. This is a concept that establishes an equivalence among those circuits that produce the same events on a given set of signals. For simplicity, we will use a restricted definition, since we are only interested in circuits in which the signals of one of them is a subset of the signals of the other.

Definition 3.6 (Observational equivalence between two circuits) *Let $C = \langle A, F \rangle$ and $\widehat{C} = \langle X, \widehat{F} \rangle$ be two circuits with $X \subseteq A$, and let $SG(C, s^0) = \langle A, S, E \rangle$ and $SG(\widehat{C}, \widehat{s}^0) = \langle X, \widehat{S}, \widehat{E} \rangle$ be their state graphs. C and $\widehat{C}$ are* observationally equivalent *from s^0 and $\widehat{s}^0$ respectively iff:*

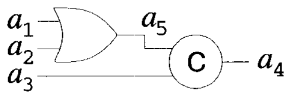
1. $\widehat{s}^0 = proj_X(s^0)$.
2. *$\forall s \in S, \widehat{s} \in \widehat{S}$ such that $\widehat{s} = proj_X(s)$ and $\forall a_i \in X$:*
   a) *if sE_is' then $\exists \widehat{s}' \in \widehat{S}$ such that $\widehat{s}\widehat{E}_i\widehat{s}'$ and $\widehat{s}' = proj_X(s')$.*
   b) *if $\widehat{s}\widehat{E}_i\widehat{s}'$ then $\exists s' \in S$ such that $sE^*_{\overline{X}}E_iE^*_{\overline{X}}s'$ and $\widehat{s}' = proj_X(s')$.*

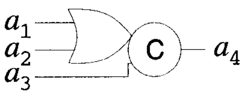
where $E^*_{\overline{X}}$ denotes any sequence of non-observable transitions.

In this paper we propose to verify semi-modularity rather than speed-independence. Semi-modularity is more robust than speed-independence and both concepts are tightly related for most practical cases, as subsequently explained (see [17] for further details).

Definition 3.7 (Semi-modularity) *A signal a_i is* semi-modular *with respect to signal $a_k \in fanin(a_i)$ ($a_i \neq a_k$) if the gate that drives a_i, having been excited, cannot become stable by changing the value of a_k. In terms of the SG of the circuit, a signal a_i is semi-modular with respect to a_k in $SG(C, s^0) = \langle A, S, E \rangle$ iff*

$$sE_ks' \Longrightarrow [s_i \neq f_i(s) \Longrightarrow s'_i \neq f_i(s')] \ .$$





$$f_4 = a_3 \cdot a_5 + a_4 \cdot (a_3 + a_5)$$
$$f_4 = a_3 \cdot (a_1 + a_2) + a_4 \cdot (a_1 + a_2 + a_3)$$

Figure 6: OR and C gates collapsed into a complex gate.

A signal a_i is semi-modular if it is semi-modular with respect to all its fan-in signals. A circuit is semi-modular if all its signals are semi-modular.

Definition 3.8 (Strongly-live circuit [17]) *A circuit is* strongly live *iff its state graph is strongly connected and for each signal a_i there exists a state $s \in S$ in which a_i is excited.*

Theorem 3.1 ([17]) *If a circuit is* strongly live, *then the circuit is* speed-independent *iff it is* semi-modular.

4 Reduction to complex gates

This section provides the means that enable to eliminate some signals of a circuit to simplify its verification. We propose to collapse several gates into one complex gate with the same functional behavior and eliminate the internal signals.

Let us assume we have a circuit $C = \langle A, F \rangle$ with signal a_n being driven by a combinational gate, i.e. $a_n \notin fanin(a_n)$. Let us build a new circuit $\widehat{C} = \langle X, \widehat{F} \rangle$, with $X = A - \{a_n\}$. Let $\widehat{s} = proj_X(s)$ and the boolean expressions of the gates of $\widehat{C}$ defined as follows:

$$\widehat{f}_i(\widehat{s}) = \begin{cases} f_i(s) & \text{if } a_n \notin fanin(a_i) \ , \\ f_i(s_1, \ldots, s_{n-1}, 0) \cdot \overline{f_n}(s) + & \\ f_i(s_1, \ldots, s_{n-1}, 1) \cdot f_n(s) & \text{if } a_n \in fanin(a_i) \ . \end{cases}$$

Note that the above expression substitutes s_n by $f_n(s)$ and, therefore, $\widehat{f}_i(\widehat{s})$ does not depend on s_n, as $a_n \notin fanin(a_n)$.

Figure 6 shows how the boolean expression of a complex gate is derived from the expressions of the simple gates. In case $|fanout(a_n)| > 1$, multiple complex gates will be created, as illustrated in Figure 7.

Theorem 4.1 *Given two circuits $C = \langle A, F \rangle$ and $\widehat{C} = \langle X, \widehat{F} \rangle$, with $X = A - \{a_n\}$ and $\widehat{F}$ defined as above, and their state graphs, $SG(C, s^0) = \langle A, S, E \rangle$ and $SG(\widehat{C}, proj_X(s^0)) = \langle X, \widehat{S}, \widehat{E} \rangle$. C and $\widehat{C}$ are observationally equivalent from s^0 and $proj_X(s^0)$ respectively iff all signals in $fanout(a_n)$ are semi-modular with respect to a_n in $SG(C, s^0)$.*

Proof
Condition 1 of definition 3.6 holds by construction. Let $s \in S, \widehat{s} \in \widehat{S}, \widehat{s} = proj_X(s)$ and $a_i \in X$. In those cases where we prove that $f_i(s) = \widehat{f}_i(\widehat{s})$, it immediately follows that observational equivalence holds. More precisely, $f_i(s) = \widehat{f}_i(\widehat{s}) = s_i$ implies that a_i is stable in both s and $\widehat{s}$ and, therefore, conditions 2.a and 2.b hold. If $f_i(s) = \widehat{f}_i(\widehat{s}) = \bar{s}_i$ there exist s' and $\widehat{s}'$ such that sE_is' and $\widehat{s}\widehat{E}_i\widehat{s}'$ and $\widehat{s}' = proj_X(s')$, since the same signal transitions from s and $\widehat{s}$. Therefore, conditions 2.a and 2.b also hold.

If $a_n \notin fanin(a_i)$ then $\widehat{f}_i(\widehat{s}) = f_i(s)$ and, therefore, observational equivalence holds.

If $a_n \in fanin(a_i)$ then

$$\widehat{f}_i(\widehat{s}) = f_i(s_1, \ldots, s_{n-1}, 0) \cdot \overline{f_n}(s) + f_i(s_1, \ldots, s_{n-1}, 1) \cdot f_n(s).$$

$\boxed{\text{semi-modularity} \Longrightarrow \text{observational equivalence}}$

Since a_i is semi-modular with respect to a_n, a change on signal a_n cannot disable a_i. Hence, $f_i(s)$ does not depend on s_n when signals a_i and a_n are simultaneously excited, i.e.

$$[f_i(s) \neq s_i \ \wedge \ f_n(s) \neq s_n] \implies$$
$$[f_i(s_1, \ldots, s_{n-1}, 0) = f_i(s_1, \ldots, s_{n-1}, 1)] \ .$$

For the previous predicate to hold we need

$$[f_i(s) = s_i] \ \vee \ [f_n(s) = s_n] \ \vee$$
$$[f_i(s_1, \ldots, s_{n-1}, 0) = f_i(s_1, \ldots, s_{n-1}, 1)] \ .$$

If $f_n(s) = s_n$, we have (by Shannon's expansion)

$$\begin{aligned} \widehat{f}_i(\widehat{s}) &= f_i(s_1, \ldots, s_{n-1}, 0) \cdot \bar{s}_n + f_i(s_1, \ldots, s_{n-1}, 1) \cdot s_n \\ &= f_i(s) \ . \end{aligned}$$

If $f_i(s_1, \ldots, s_{n-1}, 0) = f_i(s_1, \ldots, s_{n-1}, 1)$ it immediately follows that

$$\widehat{f}_i(\widehat{s}) = f_i(s_1, \ldots, s_{n-1}, 0) = f_i(s_1, \ldots, s_{n-1}, 1) = f_i(s) \ .$$

It only remains the case

$$[f_i(s) = s_i] \ \wedge \ [f_n(s) = \bar{s}_n] \ \wedge$$
$$[f_i(s_1, \ldots, s_{n-1}, 0) = \overline{f_i}(s_1, \ldots, s_{n-1}, 1)] \ .$$

which describes the situation in which a_i is stable, a_n is excited, and $f_i(s)$ depends on the value of signal a_n. Hence,

$$\begin{aligned} \widehat{f}_i(\widehat{s}) &= \overline{f_i}(s_1, \ldots, s_{n-1}, 1) \cdot s_n + f_i(s_1, \ldots, s_{n-1}, 1) \cdot \bar{s}_n \\ &= \overline{f_i}(s) \ = \ \bar{s}_i \ . \end{aligned}$$

Clearly, condition 2.a holds for state s, since a_i is not excited in s. To prove 2.b, let us take $\widehat{s}'$ such that

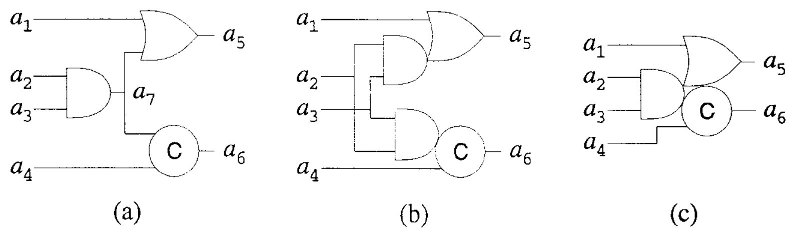


Figure 7: (a) Gate with multiple-fan-out. (b) Complex gate considered for functional correctness (c) and for behavioral correctness.

$\hat{s}\hat{E}_i\hat{s}'$. We will prove that there exist $s', s'' \in S$ such that $sE_n s'' E_i s'$ and $\hat{s}' = proj_X(s')$.

Since a_n is excited in s then we have $s'' \in S$ such that $sE_n s''$. But now, a_i is also excited in s'' as

$$f_i(s_1,\ldots,s_{n-1},0) = \overline{f_i}(s_1,\ldots,s_{n-1},1) \ ,$$

and thus there exists $s' \in S$ such that $s''E_i s'$. Finally, s and s' only differ in the ith and nth elements and therefore $\hat{s}' = proj_X(s')$.

$\boxed{\neg \text{ semi-modularity} \Longrightarrow \neg \text{ observational equivalence}}$

If a_i is not semi-modular with respect to a_n, then $\exists s, s', s'' \in S$ such that $sE_i s'$, $sE_n s''$ and a_i is not excited in s''.
Since only a_n changes between s and s'', we have that $\hat{s} = proj_X(s) = proj_X(s'')$. Thus,

$$\hat{f_i}(\hat{s}) = f_i(s_1,\ldots,s_{n-1},0)\cdot\overline{f_n}(s) + f_i(s_1,\ldots,s_{n-1},1)\cdot f_n(s) \,.$$

Since a_i is excited in s and stable in s'' (after a transition of a_n) then

$$f_i(s_1,\ldots,s_{n-1},0) = \overline{f_i}(s_1,\ldots,s_{n-1},1) \ .$$

Moreover, $f_n(s) = \bar{s}_n$ and $f_i(s) = \bar{s}_i$, as a_n and a_i are excited in s. Therefore,

$$\begin{aligned}\hat{f_i}(\hat{s}) &= \overline{f_i}(s_1,\ldots,s_{n-1},1)\cdot s_n + f_i(s_1,\ldots,s_{n-1},1)\cdot \bar{s}_n \\ &= \overline{f_i}(s) \ = \ s_i \ ,\end{aligned}$$

which means that a_i is not excited in $\hat{s}$ and, therefore, condition 2.a does not hold. □

Theorem 4.1 is the basis to prove that hierarchical verification is exact. This is the purpose of the next corollaries.

Corollary 4.1 *$\hat{C}$ not semi-modular from $\hat{s}^0 \Longrightarrow C$ not semi-modular from s^0.*

Proof This immediately follows from the fact that the state graph of $\hat{C}$ is the projection of the state graph of C. □

Corollary 4.1 guarantees that hierarchical verification will not give false negatives.

Corollary 4.2 *If $\hat{C}$ is semi-modular from $\hat{s}^0$ and C is not semi-modular from s^0, then either a_n or some signal $a_i \in fanout(a_n)$ are not semi-modular in C.*

Proof (by contradiction) Assume that a_n and all its fanout signals are semi-modular. Then, by theorem 4.1, C and $\hat{C}$ should be observationally equivalent. Since $\hat{C}$ is semi-modular and C is not semi-modular, then a_n (the only non-observable signal) should be non-semi-modular, which contradicts the initial assumption. □

Corollary 4.2 shows that hierarchical verification does not produce false positives. Consider a complex gate that drives $a_i \in fanout(a_n)$, and that X_i is the set of input/output signals of the gate, i.e.

$$X_i = \{a_i\} \cup (fanin(a_i) - \{a_n\}) \cup fanin(a_n) \ .$$

It can be derived that, by taking $proj_{X_i}(SG(\hat{C}, \hat{s}^0))$ as the environment of the complex gate, and s_n^0 as the initial value for signal a_n, non-semi-modularity of a_i and a_n in $SG(C, s^0)$ is also detected in $proj_{X_i}(SG(\hat{C}, \hat{s}^0))$ (by proposition 3.1).

Intuitively it can be proved by showing there is always one state s of C in which non-semi-modularity is manifested for the first time from s^0. Because of the observational equivalence while semi-modularity holds from s^0, the projection of s onto X_i will also belong to the set of states of $proj_{X_i}(SG(\hat{C}, \hat{s}^0))$.

4.1 Environment of a complex gate and circuits with environment

Complex gates obtained from collapsing can be seen as externally-cut circuits [1]. An important property of externally-cut circuits is that they have no hidden state. The state of such circuits is completely captured by the values of the interface signals, i.e. the values of the interface signals uniquely define the value to which all internal signals would eventually settle if the interface were held fixed [1]. This follows from the fact that memory elements in externally-cut circuits can be regarded as combinational gates when given an interface state. For example, a C-element will operate as an AND gate in those states in which the output is zero, but as an OR gate if the output is one.

The projection of the state graph onto the interface signals will keep the edges involving interface signal switches (see proposition 3.1). This projection, however, may fold semantically different states onto the same state, thus introducing additional nondeterminism (choice). Nevertheless, input choice is not a problem because in the second verification step we are dealing with externally-cut circuits. Since there are no hidden variables, the circuit reaction will depend only on the state and on the signal that has switched. Therefore, the behavior of an externally-cut circuit in such cases will be the same independently of whether there is a state with nondeterministic choice or two different states (with deterministic choice).

Let us assume that a circuit has several instances of the same (complex) gate. Figure 8 shows two AND gates of the same circuit with a different environment for each. As previously mentioned, the environment of a complex gate is calculated as the projection of the state graph onto X_i. In this is example the states labeled with 010 of the environment of G2 result from the projection of two different states[2].

To verify the semi-modularity of each AND gate, we calculate the union of the environments of all AND gates of the circuit (environment for the generic gate G in Figure 8). This many-to-one mapping may introduce choice and/or non-determinism not manifested in the initial state graph. In fact, the set of sequences of transitions accepted by the union of projected state graphs can be larger than the union of the sets of sequences generated by each individual gate. However, semi-modularity is a local property of a gate that needs to be checked only between adjacent states of its environment. Since any transition of the projected state graph results from at least one projection of the original state graph, verification is not pessimistic but exact.

Interestingly, if the union of the projected state graphs produces a semi-modular behavior of G (the generic gate), it also describes a set of sequences of events that, if applied to each gate individually, would produce a semi-modular behavior.

Needless to say that, with the previous considerations, the presented approach allows to verify circuits against an environment described by a state graph, possibly containing choice, non-determinism and/or state variables that do not correspond to values of input/output signals.

5 Implementation issues

A verifier based on symbolic model checking has been implemented. Its inputs are a Signal Transition Graph, describing the behavior of the environment, and a netlist of gates. The environment only needs to specify transitions of the interface signals of the circuit. Input/output choice and non-determinism are allowed.

The markings (states) of the Signal Transition Graph are symbolically represented by using encoding techniques such as the ones presented in [8]. Disjunctively partitioned transition relations and breadth first search algorithms for symbolic traversal [4] have been used to calculate the set of reachable states. Next, some implementation issues are discussed.

[2]For the sake of clearness, they are depicted as different states in the figure

5.1 Reduction to complex gates

The algorithm currently implemented is very simple. Each combinational gate is collapsed with its fanout gates. Only when the output of a combinational gate is one of its inputs (feedback loop), the reduction is not possible.

At the end of the reduction step, only one signal for each memory element and combinational loop is kept. These signals are the ones used for functional verification.

5.2 Output choice

Circuits with output choice (arbitration) can also be verified with our method. The non-semi-modularity of arbitration signals (e.g. outputs of a mutex) is considered hidden inside the gate and not manifested externally. This requires a special ad-hoc description of arbitration elements in the library of gates. For example, a mutex element with two inputs (R1,R2) and two outputs (A1,A2) can be modeled by two boolean equations:

$$\mathtt{A1} = \mathtt{R1} \wedge \overline{\mathtt{A2}}; \qquad \mathtt{A2} = \mathtt{R2} \wedge \overline{\mathtt{A1}}$$

In this case, non-semi-modularity is allowed for A1 and A2 with respect to A2 and A1 respectively.

5.3 Isochronic forks

Verification of speed-independence assumes that wire delays are negligible with regard to gate delays. As shown in Figure 7, gates with multiple fan-out are split into several instances, each one collapsed with one of the fan-out gates. However, forks must be considered isochronic during the detection of hazards on the internal signals. Therefore, gates that share some input signals must be simultaneously verified for behavioral correctness, with only one common instance of the multiple-fan-out internal gates. As it is shown in Figure 7.(c), signals a_5 and a_6 must be simultaneously verified with only one instance of the gate that drives a_7. This would not be necessary if delay-insensitiveness were verified, since forks are not assumed to be isochronic.

6 Experimental results

Table 1 reports the results obtained from running several experiments on our verifier. All the examples are scalable, i.e. they can be enlarged by simply increasing the number of instances of the basic cells. However, their intrinsic regularity has not been exploited to verify the circuit.

The examples used are the following: master-read (obtained from automatic synthesis tools), a Distributed Mutual Exclusion (DME) circuit [9, 6], a tree arbiter [16], an asynchronous FIFO [10], a register file [13] and a demultiplexer [3].

Results on flat (no reduction to complex gates) and hierarchical verification are shown[3]. The number of signals for hierarchical verification corresponds to the

[3]For the DME, results are comparable to those presented in [4] when multiple initial states are used. Here, we only present results obtained with one initial state (to avoid taking advantage of the regularity of the circuit)

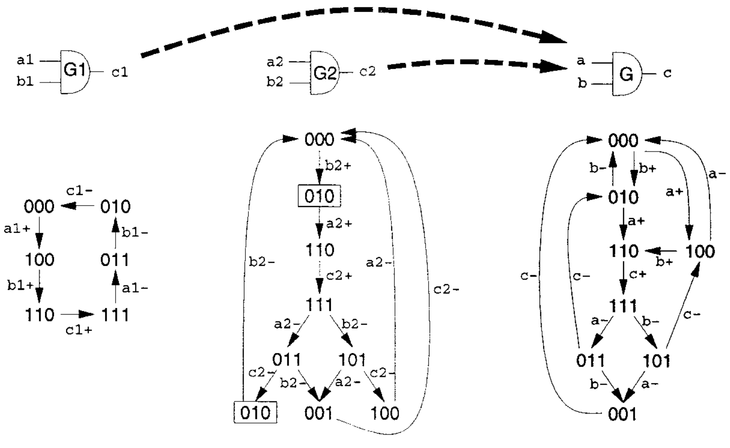


Figure 8: Union of environments for different instances of the same gate.

number of states signals of the circuit, since all combinational gates are eliminated.

All the circuits are dominated by memory elements. The one with most combinational gates is the DME (half of the signals). The FIFO is a peculiar case, as all the signals are outputs of memory elements and, therefore, no difference exists between flat and hierarchical verification. The reported BDD sizes are the largest ones encountered during the traversal of the circuit. The CPU time of hierarchical verification is mostly dominated by the first step (functional verification). The number of states of hierarchical verification is the one obtained during functional verification.

The size of the BDDs, often crucial to avoid running out of space, is reduced by the fact that many variables are eliminated when reducing to complex gates. The significant improvements in CPU time are basically due to two factors: 1) the reduction of the size of the BDDs and 2) the reduction of the logic depth of the circuit, which directly influences on the number of iterations required to reach a fixed point during the traversal.

The presented results confirm that hierarchical verification makes time complexity depend on the number of state signals of the circuit, rather than the number of gates. We believe that even better results can be obtained for circuits generated by automatic synthesis techniques, in which the ratio of combinational gates may be higher. However, at this moment there are no examples large enough to be considered critical for verification (the largest ones can be verified in roughly a dozen of seconds). As tools for synthesis and composition of circuits become mature, the complexity of the circuits will increase significantly.

7 Conclusions

The complexity of formal verification of asynchronous circuits fundamentally depends on the size of the circuit, i.e. the number of gates. Reducing the size of a circuit by collapsing gates into complex gates only allows a partial verification in which a false positive might be given as result.

In this paper, sufficient conditions for hierarchically verifying speed-independence have been presented. It has been shown that an *exact verification* can still be done if the circuit is reduced to complex gates and the environment of each complex gate is calculated during the verification of functional correctness. Circuits are allowed to be verified against an environment that may specify input/output choice and non-determinism.

A verifier based on symbolic model checking has been implemented and several experiments with large circuits reported. It has been shown that, by reducing the number of relevant variables during verification, both the size of the BDDs and the computational cost drastically drop.

As future work, techniques for regularity extraction will be explored [15]. They should allow to further reduce the computational cost of those circuits in which combinational gates dominate over memory elements.

Acknowledgments

We would like to thank Luciano Lavagno, Alex Yakovlev, Michael Kishinevsky and Alex Kondratyev for numerous insightful discussions on improving the clarity and presentation of this work.

References

[1] P. A. Beerel. *CAD Tools for the Synthesis, Verification, and Testability of Robust Asynchronous Circuits.* PhD thesis, Stanford Univ., Aug. 1994.

[2] P. A. Beerel, J. R. Burch, and T. H.-Y. Meng. Sufficient conditions for correct gate-level speed-independent circuits. In *Proc. Int. Symp. on Advanced Research in Asynchronous Circuits and Syst.*, pages 33–43. IEEE Computer Society Press, Nov. 1994.

[3] P. A. Beerel and T. H.-Y. Meng. Semi-modularity and testability of speed-independent circuits. *Integration, the VLSI journal*, 13(3):301–322, Sept. 1992.

example	signals		states		BDD size		iterations		CPU (sec.)		speed-
	flat	hier.	flat	hier.	flat	hier.	flat	hier.	flat	hier.	up
master-read	27	14	1.7×10^5	2.1×10^3	6432	1091	13	13	42	5	8.4
4 DME	72	36	7.5×10^4	4.1×10^3	1963	1082	34	19	39	7	5.6
8 DME	144	72	8.0×10^8	2.1×10^6	5175	3035	58	31	428	86	5.0
16 DME	288	144	4.5×10^{16}	2.8×10^{11}	11599	7181	106	55	3890	799	4.9
32 DME	576	288	7.0×10^{31}	2.4×10^{21}	24751	16391	202	103	28365	5788	4.9
6 Tree arb.	42	28	2.3×10^5	6.3×10^4	3253	2457	22	22	40	25	1.6
8 Tree arb.	58	38	1.4×10^7	2.3×10^6	9084	6239	30	27	261	132	2.0
10 Tree arb.	74	48	7.9×10^8	8.4×10^7	20789	9184	39	37	1425	512	2.8
12 Tree arb.	90	58	4.5×10^{10}	3.0×10^9	26717	14703	49	45	4316	1318	3.3
5 FIFO	18	–	2.7×10^3	–	573	–	12	–	8	–	–
10 FIFO	33	–	1.2×10^6	–	1765	–	22	–	130	–	–
15 FIFO	48	–	5.8×10^8	–	3662	–	32	–	634	–	–
20 FIFO	63	–	2.7×10^{11}	–	6214	–	42	–	2200	–	–
4-bit REG	74	50	7.6×10^7	1.2×10^6	14000	10412	15	7	388	171	2.3
6-bit REG	109	73	2.3×10^{11}	4.6×10^8	40346	31008	15	7	2156	681	3.2
8-bit REG	144	96	7.1×10^{14}	1.7×10^{11}	93913	70133	15	7	7246	1763	4.1
10-bit REG	179	119	2.2×10^{18}	6.7×10^{13}	188040	149250	15	7	11330	4650	2.4
6-bit DEMUX	91	49	1.6×10^{17}	8.2×10^9	19057	3083	9	5	662	95	7.0
8-bit DEMUX	121	65	6.9×10^{22}	1.3×10^{13}	26473	4117	9	5	1268	181	7.0
10-bit DEMUX	151	81	2.9×10^{28}	2.1×10^{16}	33889	5151	9	5	2027	292	6.9
12-bit DEMUX	181	97	1.3×10^{34}	3.4×10^{19}	41305	6185	9	5	2995	444	6.7

Table 1: Experimental results.

[4] J. R. Burch, E. M. Clarke, D. E. Long, K. L. McMillan, and D. L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Trans. on CAD*, 13(4):401–424, 1994.

[5] T.-A. Chu. *Synthesis of Self-timed VLSI Circuits from Graph-theoretic Specifications.* PhD thesis, MIT, June 1987.

[6] D. L. Dill. *Trace Theory for Automatic Hierachical Verification of Speed-Independent Circuits.* ACM Distinguished Dissertations. MIT Press, 1989.

[7] M. Kishinevsky, A. Kondratyev, A. Taubin, and V. Varshavsky. Analysis and identification of speed-independent circuits on an event model. *Formal Methods in System Design*, 4(1):33–75, Jan. 1994.

[8] A. Kondratyev, J. Cortadella, M. Kishinevsky, E. Pastor, O. Roig, and A. Yakovlev. Checking signal transition graph implementability by symbolic BDD traversal. In *Proc. EDAC-ETC-EuroASIC*, pages 325–332, Paris, Mar. 1995.

[9] A. J. Martin. The design of a self-timed circuit for distributed mutual exclusion. In H. Fuchs, editor, *Proc. of the Chapel Hill Conf. on VLSI*, pages 245–260. Computer Science Press, 1985.

[10] A. J. Martin. Self-timed FIFO: An exercise in compiling programs into VLSI circuits. In D. Borrione, editor, *From HDL Descriptions to Guaranteed Correct Circuit Designs*, pages 133–153. Elsevier Science Publishers, 1986.

[11] K. L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In G. v. Bochman and D. K. Probst, editors, *Proc. Int. Workshop on CAV*, volume 663 of *LNCS*, pages 164–177. Springer-Verlag, 1992.

[12] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, 1980.

[13] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, and A. Takamura. TITAC: Design of a quasi-delay-insensitive microprocessor. *IEEE Design & Test of Comp.*, 11(2):50–63, 1994.

[14] D. K. Probst and H. F. Li. Using partial-order semantics to avoid the state explosion problem in asynchronous systems. In R. P. Kurshan and E. M. Clarke, editors, *Proc. Int. Workshop on CAV*, volume 531 of *LNCS*, pages 146–155. Springer-Verlag, 1990.

[15] D. Rao and F. Kurdahi. On clustering for maximal regularity extraction. *IEEE Trans. on CAD*, 12(8):1198–1208, Aug. 1993.

[16] C. L. Seitz. Ideas about arbiters. *Lambda*, 1(1, First Quarter):10–14, 1980.

[17] A. Yakovlev, L. Lavagno, and A. Sangiovanni-Vincentelli. A unified signal transition graph model for asynchronous control circuit synthesis. In *Proc. of the IEEE Int. Conf. on Computer Aided Design*, pages 104–111. IEEE Computer Society Press, Nov. 1992.