[11] J. Kozhaya and F. N. Najm, "Accurate power estimation for large sequential circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1997, pp. 448–493.

[12] L. Ljung, *System Identification: Theory for the User*. Engelwood Cliffs, NJ: Prentice-Hall, 1987.

Sequential Equivalence Checking Based on Structural Similarities

C. A. J. van Eijk

***Abstract*—Checking the functional equivalence of sequential circuits is an important practical problem. Because general algorithms for solving this problem require a state-space traversal of the product machine, they are computationally expensive. In this paper, we present a new method for sequential equivalence checking which utilizes functionally equivalent signals to prove the equivalence of both circuits, thereby avoiding the state-space traversal. The effectiveness of the proposed method is confirmed by experimental results on retimed and optimized ISCAS'89 benchmarks.**

***Index Terms*—Equivalence checking, finite-state machine verification, formal verification, sequential circuits, structural similarity.**

I. Introduction

With the increasing use of sequential optimizations during logic synthesis, sequential equivalence checking is becoming an important practical verification problem. Conventional algorithms for solving this problem first build the so-called product machine, which is the parallel composition of the two circuits being verified. Then they check that the corresponding outputs of the two circuits are identical in every state of the product machine reachable from its initial state. To determine which states in the product machine are reachable, a state-space traversal of the product machine is required. This state-space traversal is a significant bottleneck in these algorithms, which limits their applicability to relatively small circuits. Impressive progress has been made in this respect by the introduction of so-called symbolic algorithms, which are based on the application of binary decision diagrams (BDD's) to traverse the state-space (see, e.g., [3], [8], [26]). Although these techniques can conceivably handle large circuits and are still being improved (see, e.g., [6], [13], and [20]), they cannot be expected to scale well with circuit size for many types of circuits.

For combinational equivalence checking, the state-of-the-art verification methods combine a powerful base verification algorithm with techniques to exploit the structural similarities of the circuits under verification (see, e.g., [2], [5], [11], [16]–[19], and [21]). These similarities typically occur in practical problem instances because of the incremental nature of the design process. The techniques to capture them are based on functional equivalences, indirect implications, or permissible functions. It has clearly been shown that the utilization of structural similarities is extremely important for the efficient verification of synthesized circuits. Because sequential optimizations such as retiming only have a limited impact on the structure of a circuit, this approach of exploiting structural similarities is also likely to be applicable to sequential equivalence checking. In this paper, we present a new method for proving the equivalence of sequential circuits that exhibit structural similarities. The method is not based on a state-space traversal, and can handle larger circuits than existing methods.

When verifying combinational circuits, structural similarities can be identified before they are used to simplify the verification problem. When dealing with sequential circuits, this is clearly more difficult. In the presence of sequential feedback, it is necessary to combine the detection and utilization of similarities to really benefit from them. In this paper, we solve this problem by proposing a fixed point iteration which gradually filters sets of potentially equivalent functions until the actual equivalences remain. This filtering process only requires combinational verification techniques. Therefore, the proposed method can be viewed as a way to extend the applicability of the state-of-the-art combinational techniques to sequential equivalence checking.

The organization of this paper is as follows. Section II gives an overview of related work on sequential equivalence checking. Section III presents the theory on which the detection of equivalent signals is based. The resulting verification method is described in Section IV. Implementation issues are discussed in Section V. Section VI gives empirical results and Section VII concludes the paper.

II. Related Work

In this paper, we focus on the utilization of structural similarities to enable the verification of large sequential circuits. As mentioned in Section I, such similarities are likely to exist between a specification and a synthesized implementation; their existence can be insured by putting restrictions on the synthesis process, such as the complete-1–distinguishability (C-1-D) property proposed in [1], or the data-enable decomposition condition of [23]. These restrictions are coupled to sequential equivalence techniques that avoid state-space traversal of the product machine, the same intention as the method proposed in this paper.

Conventional symbolic algorithms for sequential equivalence checking do not attempt to benefit from the structural similarities of the circuits under verification. Several techniques are proposed in literature to improve them in this respect. In [10], the use of functional dependencies is proposed to exploit the relation between the state encodings of both circuits during the state-space traversal of the product machine. In [22], a method is described to incrementally re-encode one of the circuits to factor out their differences.

When a sequential circuit is only optimized with combinational synthesis techniques, the correctness of the implementation can be verified with a combinational verification method if it is known which registers in the two circuits correspond. An efficient technique to automatically identify this register correspondence without calculating the reachable state-space of the product machine is proposed independently in [9] and [12]. In [4], this approach is extended to also locate bugs in incorrect circuits and to take don't care conditions into consideration. The detection of corresponding registers also forms the basis for the utilization of structural similarities in the verification method proposed in [15]. A preprocessing step for handling retimed circuits is described in [14], which relies on three-valued equivalence and name correspondences. Because most combinational synthesis techniques do not preserve the behavior of a circuit with respect to three-valued equivalence, the applicability of this step is limited in practice.

Recently, a new sequential verification method called "Record & Play" was proposed in [25]. This method uses recursive learning in combination with a so-called structural fixed point iteration to find equivalent signals. By applying retiming transformations, the two circuits are made more similar. The concept of instruction queues is introduced to capture the equivalent signals.

Manuscript received December 16, 1998; revised December 14, 1999. This paper was recommended by Associate Editor D. Dill.

The author is with the Eindhoven University of Technology, 5600 MB Eindhoven, the Netherlands (e-mail: C.A.J.v.Eijk@ele.tue.nl).

Publisher Item Identifier S 0278-0070(00)05862-0.

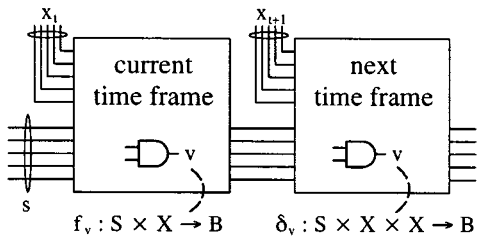


Fig. 1. Time frame model of the product machine.

Based on the work outlined above, the following important observation can be made: In many practical cases, sequential equivalence of circuits can be proved by determining the correct relation between their state encodings rather than by calculating the entire reachable state-space of the product machine. The method we propose in this paper relies on this observation. It uses a greatest fixed point iteration to identify functionally equivalent signals, which is a generalization of the techniques used in [4], [9], and [12] for determining corresponding registers.

III. The Signal Correspondence Relation

Our basic model of a sequential circuit is a deterministic Mealy-type finite-state machine (FSM) with a specified initial state. We assume that we are given two circuits, which are combined into a product machine with input space X, state-space S, initial state $s_0 \in S$, next-state function Δ: $S \times X \to S$, and output function Λ: $S \times X \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ denotes the set of Boolean values. The output function Λ is one for a given state and input vector, if all pairwise corresponding outputs of the two circuits evaluate to the same value. The objective of sequential equivalence checking is to prove that Λ is one for every reachable state and input vector.

The basis of our verification method is a greatest fixed point iteration, which works on the set of all functions associated with the signals (or nets) in both circuits. It gradually partitions this set into equivalence classes, such that all functions in the same equivalence class are sequentially equivalent, meaning that for any reachable state and input vector, they have the same value. In each step of the iteration, two consecutive time frames of the product machine are considered which are called the *current time frame* and the *next time frame*, as shown in Fig. 1. Each time frame consists of the combinational logic of the product machine. The current time frame is used to model the potential equivalences between the signals found thus far. The correctness of these equivalences is verified in the next time frame.

With each signal v, we associate a current-state function f_v: $S \times X \to \mathbb{B}$ which expresses the value of v in the current time frame as a function of the current state and the current input vector, and a next-state function δ_v: $S \times X \times X \to \mathbb{B}$ which expresses the value of v in the next time frame as a function of the current state, the current input vector and the next input vector. Note that $\delta_v(s, x_t, x_{t+1}) = f_v(\Delta(s, x_t), x_{t+1})$.

Before we describe the fixed point iteration in detail, we first introduce the theory on which it is based. The set of all signals in the product machine is denoted V. Based on this set, we construct the following set of functions F. We take the initial state s_0 and a randomly selected input vector $x_0 \in X$ as a reference point. For each signal $v \in V$, we consider the value of $f_v(s_0, x_0)$. If it is one, then we add the function f_v to the set F and, otherwise, its complement $\overline{f}_v$. This procedure normalizes each signal in the product machine with respect to its polarity, which is important for detecting not only equivalent, but also antivalent signals (i.e., signals having opposite values in every reachable state).

Informally, the detection of structural similarities corresponds to identifying signals with sequentially equivalent current-state functions. This can be formalized as the calculation of an equivalence relation on F, such that all functions equivalent with respect to this relation are also sequentially equivalent. Note that this still allows a function to correspond with several other functions. Before defining the conditions an equivalence relation on F has to comply with to represent a correct signal correspondence, we first introduce the notion of a state compatibility condition to associate a set of states with an equivalence relation on F. More specifically, this state compatibility condition is a function which evaluates to true for all states conforming to that relation.

Definition 1: Given an equivalence relation $\overset{sc}{\equiv}$ on the set F. The *state compatibility condition* is the function $Q_{\overset{sc}{\equiv}}$: $S \times X \to \mathbb{B}$ that defines whether a state conforms to the relation $\overset{sc}{\equiv}$, i.e., whether all functions in the same equivalence class of $\overset{sc}{\equiv}$ indeed have the same value

$$Q_{\overset{sc}{\equiv}}(s, x) = \prod_{f_m, f_n \in F \wedge f_m \overset{sc}{\equiv} f_n} f_m(s, x) = f_n(s, x).$$

Note that the state compatibility condition may also depend on the current input vector. This is a technicality: The theory is not influenced by adding a universal quantification over the input space in the definition of the condition.

We can now define the conditions an equivalence relation $\overset{sc}{\equiv}$ on F has to satisfy to represent a correct signal correspondence. We impose the following two conditions. The first condition is that if two functions correspond according to $\overset{sc}{\equiv}$, then they must have the same value in the initial state s_0. This guarantees that both circuits start in a state in which $Q_{\overset{sc}{\equiv}}$ holds. The second condition we impose is that if we consider two functions that correspond according to $\overset{sc}{\equiv}$, and a state in which $Q_{\overset{sc}{\equiv}}$ holds, then the associated next-state functions have to be equivalent in this state. This condition guarantees that if the two circuits are in a state for which $Q_{\overset{sc}{\equiv}}$ holds, then $Q_{\overset{sc}{\equiv}}$ will also hold for every next state of the circuits. If both these conditions are satisfied, it can directly be concluded that all functions in the same equivalence class necessarily are sequentially equivalent. We introduce the term *signal correspondence relation* to denote an equivalence relation on F that complies with the two conditions described above.

Definition 2: An equivalence relation $\overset{sc}{\equiv}$ on F is a *signal correspondence relation* iff for each pair of functions $f_m, f_n \in F$ with $f_m \overset{sc}{\equiv} f_n$:

- for all $x \in X$: $f_m(s_0, x) = f_n(s_0, x)$;
- for all $s \in S$, $x_t, x_{t+1} \in X$: $Q_{\overset{sc}{\equiv}}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})$.

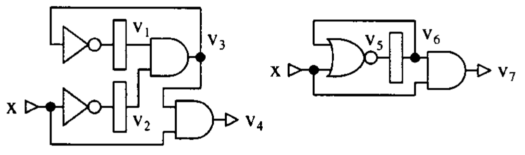
We will use the example of Fig. 2 to illustrate the notion of a signal correspondence relation and the associated state compatibility condition. An example of a correct signal correspondence relation for this example is given by the partition $\{\{f_1\}, \{f_2\}, \{f_3, f_6\}, \{f_4, f_7\}, \{f_5\}\}$. This relation states that the signals v_3 and v_6 are sequentially equivalent, as are the signals v_4 and v_7. The associated state compatibility condition is

$$Q_{\overset{sc}{\equiv}} = (v_1 v_2 \equiv v_6)(v_1 v_2 x_t \equiv v_6 x_t) \tag{1}$$

which can be simplified to $Q_{\overset{sc}{\equiv}} = (v_1 v_2 \equiv v_6)$. Using the information given in the table of Fig. 2, it can be checked that this signal correspondence relation satisfies both conditions stated in Definition 2. For example, applying the second condition of this definition to the functions f_3 and f_6, gives

$$v_1 v_2 \equiv v_6 \Rightarrow \overline{v_1 v_2}\ \overline{x_t} \equiv \overline{x_t + v_6} \tag{2}$$

which is a tautology. Similarly, it can be proved that the functions of the outputs v_4 and v_7 are equivalent and, thus, that the two circuits are equivalent.



init.:
$v_1 = 1$,
$v_2 = 1$,
$v_6 = 1$,
$x_t = 1$

v_i	init.	f_i	δ_i
v_1	1	v_1	$\overline{v_1 v_2}$
v_2	1	v_2	$\overline{x_t}$
v_3	1	$v_1 v_2$	$\overline{v_1 v_2}\,\overline{x_t}$
v_4	1	$v_1 v_2 x_t$	$\overline{v_1 v_2}\,\overline{x_t} x_{t+1}$
v_5	0	$x_t + v_6$	$x_{t+1} + \overline{x_t + v_6}$
v_6	1	v_6	$\overline{x_t + v_6}$
v_7	1	$v_6 x_t$	$\overline{x_t + v_6}\, x_{t+1}$

Fig. 2. Example of a retimed and logically optimized circuit.

For a given pair of circuits, there may exist several signal correspondence relations. We state without proof that the set of all signal correspondence relations is a partially ordered set in which each pair of elements has a least upper bound. As a consequence, there is a unique maximum relation $\overset{\text{msc}}{=}$. This maximum signal correspondence relation can also be characterized by the property that for any other signal correspondence relation $\overset{\text{sc}}{=}$ and for all $f_m, f_n \in F$

$$f_m \overset{\text{sc}}{=} f_n \Rightarrow f_m \overset{\text{msc}}{=} f_n. \tag{3}$$

Our verification method uses the following theorem to prove the equivalence of sequential circuits. This theorem follows from the observation that the conditions imposed by Definition 2 are sufficient to guarantee that the state compatibility condition of the maximum signal correspondence relation is an overestimation of the product machine's reachable state-space.

Theorem 1: Let $\overset{\text{msc}}{=}$ denote the maximum signal correspondence relation. If for all $s \in S$, $x \in X$

$$Q_{\overset{\text{msc}}{=}}(s, x) \Rightarrow \Lambda(s, x) = 1$$

then the two circuits are sequentially equivalent.

Note that Theorem 1 states a *sufficient* and not a *necessary* condition for the equivalence of sequential circuits. It is useful because it offers a different tradeoff between performance and accuracy than methods that require a state-space traversal: As will be shown in the next section, the maximum signal correspondence relation can be calculated using *combinational* verification techniques and, therefore, it provides improved efficiency in comparison to state-space traversal methods, with the weakness that the method may suffer from false negatives.

IV. Verification Method

In the previous section we have shown how the maximum signal correspondence relation $\overset{\text{msc}}{=}$ can be used to prove the sequential equivalence of two circuits. We will now describe the greatest fixed point iteration to calculate this relation. A sequence of equivalence relations T_i is calculated, which converges to the maximum signal correspondence relation. The first relation T_0 compares the functions in F with respect to the initial state

$$f_m\, T_0\, f_n \Leftrightarrow \forall x \in X\colon f_m(s_0, x) = f_n(s_0, x). \tag{4}$$

Starting with T_0, a sequence of relations is calculated by applying the second condition of Definition 2

$$\begin{aligned} f_m\, T_{i+1}\, f_n \Leftrightarrow f_m\, T_i\, f_n \wedge (\forall s \in S,\ x_t,\ x_{t+1} \in X\colon \\ Q_{T_i}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})). \end{aligned} \tag{5}$$

Since there is only a finite number of functions in F and the sequence of relations T_i is monotone nonincreasing, a fixed point is reached after a finite number of iterations, i.e., at some point $T_i = T_{i+1}$. This T_i is the maximum signal correspondence relation. The number of iterations is at most $|F| + 1$, because in every iteration, except the last one, the number of equivalence classes increases by at least one.

Theorem 2: Given the sequence of equivalence relations as defined by (4) and (5). If $T_i = T_{i+1}$, then T_i is the maximum signal correspondence relation.

Proof: We first show that T_i is indeed a signal correspondence relation as defined in Definition 2. Because of the structure of (5), it is easy to see that each relation T_{j+1} is contained in the relation T_j, with $0 \le j \le i$: This implies that $f_m\, T_i\, f_n \Rightarrow f_m\, T_0\, f_n$ and, thus, that

$$f_m\, T_i\, f_n \Rightarrow (\forall x \in X\colon\ \ f_m(s_0, x) = f_n(s_0, x)). \tag{6}$$

Therefore, we may conclude that T_i satisfies the first condition of Definition 2. If $T_i = T_{i+1}$, we can rewrite (5) to

$$\begin{aligned} f_m\, T_i\, f_n \Leftrightarrow f_m\, T_i\, f_n \wedge (\forall s \in S,\ x_t,\ x_{t+1} \in X\colon \\ Q_{T_i}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})). \end{aligned} \tag{7}$$

From (7), it follows that

$$\begin{aligned} f_m\, T_i\, f_n \Rightarrow (\forall s \in S,\ x_t,\ x_{t+1} \in X\colon \\ Q_{T_i}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1})) \end{aligned} \tag{8}$$

which proves that T_i also satisfies the second definition of 2; hence, if $T_i = T_{i+1}$, then this T_i is a signal correspondence relation. We will now prove that T_i equals $\overset{\text{msc}}{=}$ by showing that it satisfies Equation (3). Given a signal correspondence relation $\overset{\text{sc}}{=}$. It is easy to see that

$$f_m \overset{\text{sc}}{=} f_n \Rightarrow f_m\, T_0\, f_n. \tag{9}$$

Now assume that there are functions $f_m, f_n \in F$ such that f_m and f_n are equivalent according to $\overset{\text{sc}}{=}$ but not with respect to T_i, i.e., $f_m \overset{\text{sc}}{=} f_n \wedge \neg(f_m\, T_i\, f_n)$. Then there has to be a T_j, with $0 \le j < i$, such that:

$$\forall f_m, f_n \in F\colon\quad f_m \overset{\text{sc}}{=} f_n \Rightarrow f_m\, T_j\, f_n \tag{10}$$

$$\exists f_m, f_n \in F\colon\quad f_m \overset{\text{sc}}{=} f_n \wedge \neg(f_m\, T_{j+1}\, f_n) \tag{11}$$

From (10), it follows that $\overset{\text{sc}}{=}$ is included in T_j and, thus, that $Q_{T_j} \Rightarrow Q_{\overset{\text{sc}}{=}}$, while from (11), it follows that there are $f_m, f_n \in F$ such that

$$\begin{aligned} \forall s \in S,\ x_t,\ x_{t+1} \in X\colon \\ Q_{\overset{\text{sc}}{=}}(s, x_t) \Rightarrow \delta_m(s, x_t, x_{t+1}) = \delta_n(s, x_t, x_{t+1}) \end{aligned} \tag{12}$$

and

$$\begin{aligned} \exists s \in S,\ x_t,\ x_{t+1} \in X\colon \\ Q_{T_j}(s, x_t) \wedge \delta_m(s, x_t, x_{t+1}) \ne \delta_n(s, x_t, x_{t+1}). \end{aligned} \tag{13}$$

and This results in a contradiction and, therefore, we conclude that for every signal correspondence relation $\overset{\text{sc}}{=}$ and for all $f_m, f_n \in F$: $f_m \overset{\text{sc}}{=} f_n \Rightarrow f_m\, T_i\, f_n$. This completes the proof. ■

We use the example of Fig. 2 to illustrate the fixed point calculation. The model of Fig. 3(a) is used to determine T_0. In bold, the function of each signal is shown for the initial state. By putting all functions that are equivalent with respect to the intial state in the same equivalence class, we obtain

$$T_0 = \{\{f_1, f_2, f_3, f_5, f_6\}, \{f_4, f_7\}\}. \tag{14}$$

Now the model of Fig. 3(b) is used to refine this partition. Note that the model only contains combinational logic. The registers shown in the figure have no formal meaning; they are only used to denote the boundaries of the time frames. The state compatibility condition associated with T_0 is

$$Q_{T_0} = (f_1 \equiv f_2)(f_1 \equiv f_3)(f_1 \equiv f_5)(f_1 \equiv f_6)(f_4 \equiv f_7). \tag{15}$$

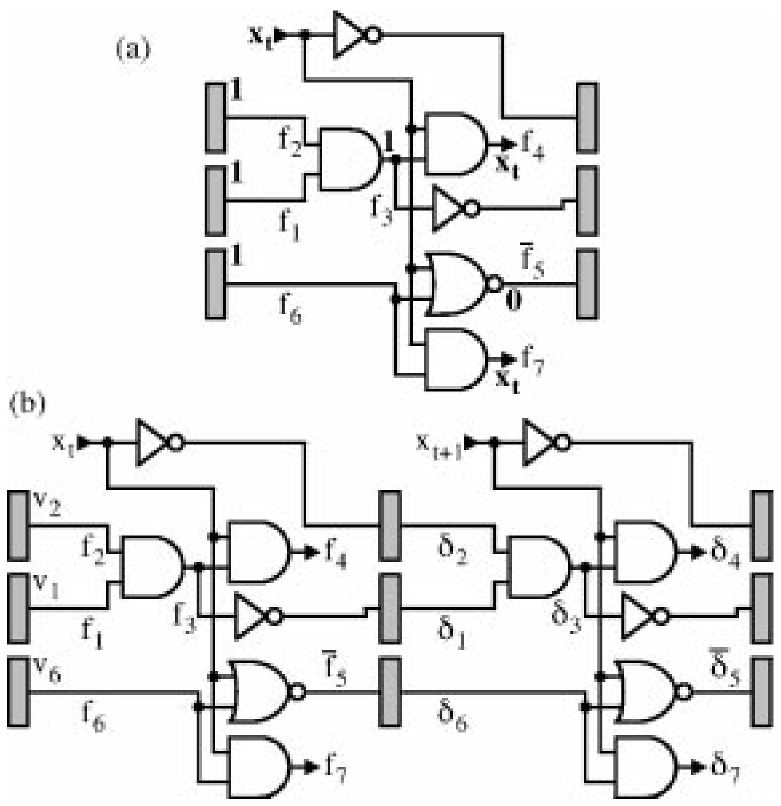


Fig. 3. Example circuit models for calculating; (a) T_0 and (b) T_{i+1}.

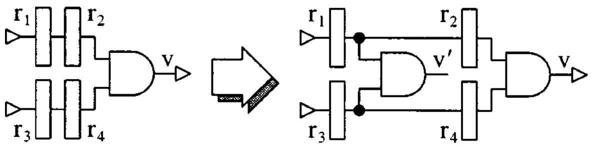


Fig. 4. Retiming with lag -1 to generate additional logic.

By applying (5), T_0 is refined to T_1

$$T_1 = \{\{f_1\}, \{f_2\}, \{f_3, f_6\}, \{f_5\}, \{f_4, f_7\}\}. \qquad (16)$$

If we apply (16) again, no further refinement occurs, i.e., $T_1 = T_2$. Therefore, this partition represents the maximum signal correspondence relation for this example.

The "scope" of the fixed point iteration explained above is determined by the functions in the set F. Its accuracy can, therefore, be improved by extending this set with extra functions. In our verification method, we do this by applying forward retiming transformations to the product machine. Note that this differs from the way retiming transformations are used in [14] and [25]: we do not actually move latches (and, thus, we avoid the problems related to maintaining a valid initial state [24])but rather add the extra combinational logic that would result from the retiming transformations. We only consider retimings with a lag of -1, meaning that at most one register is moved from every input to every output of a gate. This is illustrated in Fig. 4. In the circuit at the left, the AND gate can be retimed with a lag of -1 by moving the registers r_2 and r_4 to the output of the gate. We model the effects of this retiming move by introducing an extra AND gate connected to the registers r_1 and r_3.

The outline of the resulting verification method is shown in Fig. 5. First the maximum signal correspondence relation is calculated. If the current-state functions of all pairwise corresponding outputs of the two circuits are equivalent according to this relation, then the sequential equivalence of both circuits is proved and we can stop. Otherwise it is checked whether the set F can be extended using retiming transformations as explained above. Note that because this step may be applied more than once, also retiming transformations with a lag smaller than -1 are considered. If the retiming generates new combinational logic and, thus, results in an extension of the set F, the method continues with the calculation of the maximum signal correspondence relation for this larger set of functions.

The proposed verification method can easily be extended to also take sequential don't cares due to the nonreachable state-space into account. For example, the reachable state-space of the specification can be used to strengthen the state compatibility condition, i.e., instead of using the state compatibility condition $Q_{\underline{\underline{\mathrm{sc}}}}$, the condition $Q_{\underline{\underline{\mathrm{sc}}}} \wedge S_{\mathrm{reach}}$ can be used, where S_{reach} denotes the characteristic function of the specification's reachable state-space. Note that by combining the specification's reachable state-space with the state compatibility condition, this information is also applied to the implementation. Instead of using the exact reachable state-space, it is also possible to use an upper bound approximation of the reachable state-space, which can be calculated efficiently using techniques as, e.g., described in [7], [13], and [20].

V. Implementation Issues

When implementing the method of Section IV, we have to choose an appropriate data structure for storing the relations T_i that are calculated during the refinement process. Because every T_i is an equivalence relation, it can be represented by its equivalence classes. Therefore, the choice of an appropriate data structure is not difficult: We can simply store the equivalence classes of T_i explicitly, resulting in a space complexity of $\mathcal{O}(F)$.

In every iteration of the fixed point computation, a new relation T_{i+1} is derived from the previous relation T_i by splitting some equivalence classes into a number of smaller classes. This is done by evaluating (5).

The complement of the state compatibility condition is basically used as a don't care set when comparing the next-state functions. We can use functional dependencies [10] of the state compatibility condition to better exploit this don't care set. To illustrate this, consider the example based on the circuits of Fig. 2 again. In this example, the state compatibility condition is $v_1 v_2 \equiv v_6$. This condition can be taken into account by actually replacing state variable v_6 by the function $v_1 v_2$. We use a greedy heuristic based on the structure of the product machine to select the state variables that can be written as a function of other state variables before the state compatibility condition is actually calculated.

Sequential simulation of the product machine with random input vectors can be used to partition the set F into sets of potentially equivalent functions; if two functions have different values during simulation, it can directly be decided that they are not equivalent. This results in a better initial approximation of the maximum signal correspondence relation and, thus, reduces the required number of iterations.

VI. Experimental Results

This section reports the results of some experiments performed with the proposed verification method. Our current implementation constructs BDD's expressed over the input and state variables to represent the state compatibility condition and the next-state functions without introducing extra variables for intermediate signals. It is based on the BDD package developed at Eindhoven University. Dynamic variable ordering is used to control the BDD variable ordering. All tests are performed on a 99 MHz HP9000/755 workstation with a memory limit of 100 (Mb)imposed on the BDD package and a time limit of 3600 (s). The verification method is tested on circuits from the ISCAS'89 benchmark set.

In the first set of experiments, we evaluate the performance of the proposed method by comparing it with the symbolic verification method of [10]; this method uses functional dependencies to capture the relation between the state encodings of both circuits. The original benchmark descriptions are verified against the synthesized versions of these circuits from [25], which have been optimized by kerneling and retiming. To make these circuits more difficult to verify, we have further optimized them using script.rugged of SIS. Table I shows the experimental results. The first two columns show the name of the benchmark and the number of registers in the circuits before and after synthesis. The following columns list the run time, the maximum number of BDD nodes during verification, and the required number of iterations for both verification methods. For the proposed method, the number between parentheses in the column #its denotes the number of times the retiming procedure is invoked. The last column shows the percentage of signals in the specification for which a corresponding signal in the implementation is found. The average percentage of equivalences is 54%; without running script.rugged on the circuits of [25], the percentage of equivalences is 85%.

The experimental results clearly show that the proposed verification method can handle larger circuits than a symbolic method which uses BDD's to traverse the state-space, even if the latter method exploits functional dependencies. Only the circuits s3384 and s6669 cannot be handled, because the BDD's become too large. This problem is however more related to the combinational verification technique used than to the proposed method.

To investigate the false negative problem, we have also verified 15 of the original benchmark descriptions against versions optimized by extracting reachability don't cares and then running script.rugged of SIS. The method of this paper can verify 80% of the examples, and only suffers from false negatives in three cases. In comparison, the method of [11], which only tries to relate the registers of both circuits and not the other signals, can only verify 33% of the circuits; in all other cases, it suffers from false negatives.

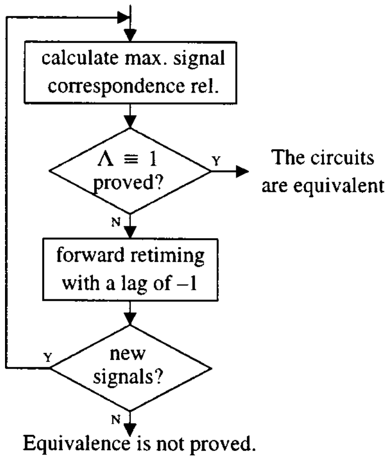

calculate max. signal correspondence rel.
$\Lambda \equiv 1$ proved?
Y
The circuits are equivalent
N
forward retiming with a lag of −1
new signals?
Y
N
Equivalence is not proved.

Fig. 5. Outline of the verification method.

TABLE I
EXPERIMENTAL RESULTS FOR RETIMED AND LOGICALLY OPTIMIZED CIRCUITS

Circuit	#regs	Symbolic traversal			Proposed method			
	orig. / opt.	time (s)	nodes	#its	time (s)	nodes	#its	eqs (%)
s208.1	8 / 14	0.7	1604	256	0.4	325	8 (0)	55
s298	14 / 29	3.8	6149	19	0.8	682	5 (0)	57
s344	15 / 38	21.0	18962	7	0.6	1061	2 (0)	49
s349	15 / 38	19.2	15944	7	0.7	1272	2 (0)	49
s382	21 / 36	8.9	11339	151	3.3	1103	21 (0)	60
s386	6 / 43	7.9	8876	8	2.2	1905	5 (0)	53
s420.1	16 / 34	73.3	11152	65536	5.3	1725	32 (0)	55
s444	21 / 36	14.3	11035	151	3.6	1172	22 (0)	58
s510	6 / 64	2683	708940	47	2.0	2473	1 (1)	42
s526	21 / 58	62.7	44942	151	6.7	1375	27 (0)	61
s641	19 / 17	2.9	5667	3	2.1	4178	2 (0)	83
s713	19 / 17	4.7	6667	3	2.5	4196	2 (0)	83
s820	5 / 53	228	165895	11	9.6	4443	8 (0)	42
s832	5 / 57	173	115889	11	11.8	4617	8 (0)	42
s838.1	32 / 74	—	—	—	55.6	4548	80 (0)	55
s953	29 / 76	130	85756	10	5.7	4225	3 (0)	51
s1196	18 / 18	4.5	8236	1	3.0	4152	2 (0)	33
s1238	18 / 18	5.5	5820	1	2.9	4015	2 (0)	30
s1423	74 / 85	—	—	—	676	100830	12 (0)	53
s1512	57 / 101	—	—	—	34.7	9339	13 (0)	60
s3271	116 / 189	—	—	—	808	28499	6 (4)	26
s3330	132 / 114	—	—	—	1316	1094772	3 (1)	64
s3384	183 / 506	—	—	—	—	—	—	-
s4863	104 / 152	—	—	—	55.6	9547	2 (0)	56
s5378	179 / 263	—	—	—	801	34117	17 (1)	71
s6669	239 / 267	—	—	—	—	—	—	-

VII. Conclusion

We have proposed a new verification method for sequential equivalence checking which proves equivalence by detecting and utilizing structural similarities rather than performing a state-space traversal of the product machine. A greatest fixed point iteration is used to determine sequentially equivalent signals. Because the method only requires combinational verification techniques, it is more efficient than symbolic verification methods requiring a state-space traversal. We expect that the performance of the method on larger circuits can be significantly improved by applying techniques that introduce cutpoints. Although the proposed method is sound, it is not a complete method, i.e., there are pairs of equivalent circuits for which it cannot prove equivalence. The method can be used as an effective preprocessing step for a general method such as [10]. It is interesting to note that for some synthesis steps, the method is complete. This is, e.g., the case for circuits optimized with combinational synthesis techniques, and also for retimed circuits.

The proposed method assumes that an initial state is designated for both circuits. This initial state is used in two ways: It acts as a reference point to allow the detection of antivalent signals, and it is used to calculate the initial partition T_0 of the set F. The approach of [4] shows that the assumption of a designated initial state can be weakened. It should be possible to extend their work such that it also applies to the method presented in this paper.

ACKNOWLEDGMENT

The author would like to thank W. Kunz and D. Stoffel for providing the circuits of [25].

REFERENCES

[1] P. Ashar, A. Gupta, and S. Malik, "Using complete-1-distinguishability for FSM equivalence checking," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 346–353.
[2] D. Brand, "Verification of large synthesized designs," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 534–537.
[3] J. R. Burch *et al.*, "Symbolic model checking for sequential circuit verification," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 401–424, Apr. 1994.
[4] J. R. Burch and V. Singhal, "Robust latch mapping for combinational equivalence checking," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 563–569.
[5] ——, "Tight integration of combinational verification methods," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 570–576.
[6] G. Cabodi *et al.*, "Disjunctive partitioning and partial iterative squaring: An effective approach for symbolic traversal of large circuits," in *Proc. 34th Design Automotation Conf.*, 1997, pp. 728–733.
[7] H. Cho *et al.*, "Algorithms for approximate FSM traversal," in *Proc. 30th Design Automation Conf.*, 1993, pp. 25–30.
[8] O. Coudert, C. Berthet, and J. C. Madre, "Verification of synchronous sequential machines based on symbolic execution," *Proc. Workshop Automatic Verification Methods for Finite State Machines*, vol. 407, pp. 365–373, 1989.
[9] C. A. J. van Eijk and J. A. G. Jess, "Detection of equivalent state variables in finite state machine verification," *Workshop notes Int. Workshop Logic Synthesis*, pp. 3.35–3.44, 1995.
[10] ——, "Exploiting functional dependencies in finite state machine verification," in *Proc. European Design& Test Conf.*, 1996, pp. 9–14.
[11] C. A. J. van Eijk, "Formal methods for the verification of digital circuits," Ph.D. dissertation, Eindhoven Univ. Technol., Eindhoven, The Netherlands, Sept. 1997.
[12] T. Filkorn, "Symbolische methoden für die verifikation endlicher zustandssysteme," Ph.D. dissertation, Institut für Informatik der Technischen Universität München, Munich, Germany, 1992.
[13] S. G. Govindaraju and D. L. Dill, "Verification by approximate forward and backward reachability," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 366–370.
[14] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, "On verifying the correctness of retimed circuits," in *Proc. Great Lakes Symp. VLSI*, 1996, pp. 277–281.
[15] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen, "AQUILA: An equivalence verifier for large sequential circuits," in *Proc. Asian South Pacific Design Automation Conf.*, 1997, pp. 455–460.
[16] J. Jain, R. Mukherjee, and M. Fujita, "Advanced verification techniques based on learning," in *Proc. 32nd Design Automation Conf.*, 1995, pp. 420–426.
[17] A. Kuehlmann and F. Krohm, "Equivalence checking using cuts and heaps," in *Proc. 34th Design Automation Conf.*, 1997, pp. 263–268.
[18] W. Kunz and D. Stoffel, *Reasoning in Boolean Networks—Logic Synthesis and Verification Using Testing Techniques*. Amsterdam, The Netherlands: Kluwer, 1997.
[19] Y. Matsunaga, "An efficient equivalence checker for combinational circuits," in *Proc. 33th Design Automation Conf.*, 1996, pp. 629–634.
[20] I.-H. Moon *et al.*, "Approximate reachability don't cares for CTL model checking," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 351–358.
[21] D. K. Pradhan, D. Paul, and M. Chatterjee, "VERILAT: Verification using logic augmentation and transformations," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 88–95.
[22] S. Quer *et al.*, "Incremental re-encoding for symbolic traversal of product machines," in *Proc. Eur. Design Automation Conf.*, 1996, pp. 158–163.
[23] R. K. Ranjan *et al.*, "Using combinational verification for sequential circuits," in *Proc. Design, Automation Test Europe Conf.*, 1999, pp. 138–144.
[24] L. Stok, I. Spillinger, and G. Even, "Improving initialization through reversed retiming," in *Proc. Eur. Design Test Conf.*, 1995, pp. 150–154.
[25] D. Stoffel and W. Kunz, "Record & play: A structural fixed point iteration for sequential circuit verification," in *Proc. Int. Conf. Computer-Aided Design*, 1997, pp. 394–399.
[26] H. J. Touati *et al.*, "Implicit state enumeration of finite state machines using BDD's," in *Proc. Int. Conf. Computer-Aided Design*, 1990, pp. 130–133.

Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations

Hai Zhou, D. F. Wong, I-Min Liu, and Adnan Aziz

***Abstract*—During the routing of global interconnects, macro blocks form useful routing regions which allow wires to go through but forbid buffers to be inserted. They give restrictions on buffer locations. In this paper, we take these buffer location restrictions into consideration and solve the simultaneous maze routing and buffer insertion problem. Given a block placement defining buffer location restrictions and a pair of pins (a source and a sink), we give a polynomial time exact algorithm to find a buffered route from the source to the sink with minimum Elmore delay.**

***Index Terms*—Buffers, integrated circuit interconnections, layout, routing.**

I. INTRODUCTION

With the evolution of very large scale integrated (VLSI) circuit fabrication technology, interconnect delay, especially global interconnect delay, has become the dominant factor in deep submicrometer design. Many techniques are employed to reduce interconnect delay; among them, buffer insertion has been shown to be an effective approach [1].

During routing process, especially that for global nets, there are macro blocks placed within the area. These blocks form useful routing regions because wires are allowed to run over them. But since buffers are implemented by transistors, a buffer "over" a macro block must be

Manuscript received March 4, 1999; revised January 4, 2000. This paper was recommended by Associate Editor C.-K. Cheng.

H. Zhou is with Advanced Technology Group, Synopsys, Inc., Mountain View, CA 94043 USA.

D. F. Wong, is with the Department of Computer Sciences, University of Texas, Austin, TX 78712 USA.

I-M. Liu, and A. Aziz are with the Department of Electrical and Computer Engineering, University of Texas, Austin, TX 78712 USA.

Publisher Item Identifier S 0278-0070(00)05863-2.