# Mining Complex Boolean Expressions for Sequential Equivalence Checking

Neha Goel*, Michael S. Hsiao*, Naren Ramakrishnan† and Mohammed J. Zaki‡

* Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, 24061.
† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24061.
‡ Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York, USA 12180.
Emails: {nehagoel, mhsiao}@vt.edu, naren@cs.vt.edu, zaki@cs.rpi.edu

***Abstract*—We propose a novel technique to mine powerful and generalized boolean relations among flip-flops in a sequential circuit for sequential equivalence checking. In contrast to traditional learning methods, our mining algorithm can detect inductive invariants as well as illegal state cubes. These invariants can be arbitrary boolean expressions and can thus prune a large don't care space during equivalence checking. Experimental results demonstrate that these general invariants can be very effective for sequential equivalence checking of circuits with no or very few equivalent signals between them, with low computational costs.**

**Keywords:** Sequential Equivalence Checking, BLOSOM, Re-descriptions, Induction based proof

## I. Introduction and Motivation

Integrated circuit design has progressed significantly over the last few decades. Synthesis and optimization techniques have helped improve performance, area, delay, and other measures. The success of combinational equivalence checking (CEC) has contributed to aggressive combinational logic synthesis and optimizations for circuits with millions of logic gates. However, without powerful sequential equivalence checking (SEC) techniques, the potential and extent of sequential optimization is quite limited. In other words, the success of SEC can unleash a plethora of aggressive sequential optimizations that can take the circuit design to the next level. Currently, SEC remains extremely difficult compared to CEC, due to the huge search space of the problem.

In the literature, several definitions of SEC exist that differ in the assumption(s) made w.r.t the operation of sequential circuit and the surrounding environment. For circuits with reset states, there exists a notion of reset equivalence, where two circuits are equivalent iff they are equivalent after reset. However, this definition is a strict and narrow form of equivalence and requires that a global reset sequence exists for both circuits.

For circuits without reset, different notions of SEC exist such as sequential hardware equivalence (SHE), safe replaceability, and 3-valued safe replaceability. Pixley proposed the notion of SHE [1] where two designs are equivalent if there exists a universal alignment sequence that takes them to an equivalent state pair. The theory of SHE is motivated by the desire to prove two designs equivalent without reference to any intended environment, presence of reset states or reset sequences. In this paper, we assume that the two circuits can be brought to a known unique, specified state by applying a synchronizing sequence. In the past few years, several methods have been proposed to solve the problem of SEC. Binary decision diagrams (BDDs) [2] have been used for symbolic techniques [3], [4]. These symbolic finite state machine (FSM) traversal techniques are possible only for small to medium sized circuits but become vulnerable to memory explosion for designs with hundreds to thousands of registers.

To reduce the complexity of SEC, structural similarity used earlier for identifying equivalent internal signal pairs (EISP) for incremental verification of combinational circuits have been explored for FSM [5], [6]. In [5], sequential automatic test pattern generation (ATPG) is used to identify equivalent flip-flops and EISP as constraints together with an induction based proof to reduce the search space. For portions of designs with different state encodings and not much similarity, symbolic techniques are then applied, together with the constraints learned earlier. Thus, incremental verification and FSM traversal are integrated to verify designs which cannot be verified by each of these approaches separately. In [6], greatest fixed point iteration is used to identify functionally equivalent signals (ES) without explicit symbolic state space traversal. However, with increasing use of sequential optimization techniques, the probability of finding ES between two designs will reduce, leading to a large don't care space and increasing problem complexity.

In this paper, we propose a novel low-cost approach for computing general legal & illegal constraints to prune the search space effectively. For example, consider the original design $C_1$ with $n$ flip-flops ($a_1$ to $a_n$) and an optimized design $C_2$ with $m$ flip-flops ($b_1$ to $b_m$) as shown in Figure 1. If there exists a relation among flip-flops $(a_1, a_2, b_1, b_2)$ such that if $a_1 = 1$ and $a_2 = 0$ in $C_1$ then $b_1 = 0$ and $b_2 = 1$ in $C_2$ and vice versa (i.e. $(a_1 \wedge \overline{a_2}) \equiv (\overline{b_1} \wedge b_2)$). Then, adding this relation as a constraint will prune the search space by preventing those states that violate it. The illegal states blocked from this constraint are shown in Table I. Note that conventional ways of finding internal signal equivalences will not uncover such general, arbitrary relationships among signals. Our approach uses data mining

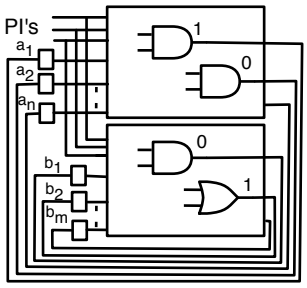


Fig. 1. Example circuit

to obtain complex yet general boolean relations shown in Equation 1. However, with $n+m$ variables, there is an exponential number of possible boolean expressions, far too many to enumerate. To make the search space tractable, one approach is to mine all frequent boolean expressions, but instead of mining frequent expressions, we mine minimal generators of the lossless subset called the closed boolean

TABLE I
EXAMPLE OF UNREACHABLE STATES (ASSUME $n < m$)

| $a_1$ | $b_1$ | $a_2$ | $b_2$ | $\cdots$ | $a_n$ | $b_n$ | $\cdots$ | $b_m$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | $\cdots$ | X | X | $\cdots$ | X |
| 1 | 1 | 0 | 0 | $\cdots$ | X | X | $\cdots$ | X |
| 1 | 1 | 0 | 1 | $\cdots$ | X | X | $\cdots$ | X |
| 0 | 0 | 0 | 1 | $\cdots$ | X | X | $\cdots$ | X |
| 0 | 0 | 1 | 1 | $\cdots$ | X | X | $\cdots$ | X |
| 1 | 0 | 1 | 1 | $\cdots$ | X | X | $\cdots$ | X |

expressions that retains the complete frequency information of the dataset. The reason for using the minimal generators is that they are the simplest expressions that represent the same information as the closed expressions. In addition to this we also discuss a mining approach for finding illegal constraints based on low threshold value.

$$(a_1 \vee b_2) \wedge (a_1 \vee b_4) \equiv (b_3 \vee a_2) \wedge (a_1 \vee a_4) \tag{1}$$

As per our observation, current circuit optimizations perform limited sequential optimizations and thus there is a lot of structural similarity between the original and optimized design. So, traditional methods based on ES may be sufficient to prove them equivalent (e.g., one circuit is a retimed version of the other). However, in this paper we propose an approach that finds invariants capturing the functionality of FSM and are not just restricted to equivalences. We postulate that if our technique is available, more powerful sequential optimizations can be applied and existing methods based on equivalences will not be sufficient to prove the two designs equivalent. We note that due to lack of the availability of these powerful optimizations, the benchmarks we can use to test our approach are limited. We generated few benchmarks with different state encodings (gray & one-hot encoding) for a given design using Synopsys Design Compiler. There was none or very few one-to-one mapping among the flip-flops in the gray and one-hot encoded designs and also limited structural similarity internally. Some of them cannot be proved by equivalent signals whereas after adding the constraints found by our mining approach they can be proved equivalent. We also tested the effectiveness of our approach on the currently available retimed benchmark circuits (where there is still significant structural similarity), without considering internal signal equivalences to see if the constraints found by our approach are effective enough. We observed that for all benchmarks our approach is effective in proving the circuits equivalent. However, we only report a few of those since existing methods as well as our method can prove them equivalent and thus they are not the main focus of the paper.
The rest of paper is organized as follows. Section 2 provides preliminaries and background. Section 3 details the two proposed approaches based on data mining and the overall proposed framework. Section 4 analyzes the experimental results. Section 5 concludes the paper.

## II. PRELIMINARIES

### A. Background on invariants

In recent years, boolean satisfiability (SAT) has improved significantly and SAT solver tools such as zChaff [7] and MiniSat [8] have been developed which can efficiently handle large industrial designs. In the context of SEC, SAT solvers have been used to compute inductive invariants, such as internal equivalent signals [9]. Lu & Cheng [10] proposed a framework for SEC based on $k^{th}$ invariants. The framework has three complementary components: bounded model checker (BMC), invariant checker and sequential SAT. BMC is used to identify and merge $k^{th}$ invariants for circuit simplification and an inductive invariant is used to find true $k^{th}$ invariants. If one of the true $k^{th}$ invariant is the miter circuit output equal to 1, then circuits are equivalent. If not, then these $k^{th}$ invariants are added as constraints to prune the search space of the Seq-SAT.

Another approach is using logic implications to capture relationships among signals in the circuit. They are used for several applications in electronic design automation (EDA) problems such as design verification [11], multi-level logic optimization [12], etc. Previous works are based on computing different kinds of implications such as static, indirect, extended backward [13] and so on. To extend the learning, methods have been proposed to extract relationships among two or more variables in the form of multi-variable relationships. But computing such relations may require an exponential number of combinations and thus low cost computational methods have been investigated. A method for finding static multi-node implications based on circuit structure information [14] has also been proposed.

A data mining approach has recently been proposed [15] using Apriori [16], a popular methodology for association rule mining to compute potential three node invariants among signals. To further reduce the mining cost, domain knowledge of circuit is used to improve the quality of the discovered rules [17]. These static and dynamic multi node implications can significantly enhance bounded SEC.

### B. Overview of Data Mining

Data mining is the process of discovering knowledge from complex and information rich data sets. In particular the boolean pattern mining problem can be potentially helpful for applications such as market based analysis, bioinformatics, etc.

TABLE II
DATASET D

| $t_{id}$ | set of items |
|---|---|
| 1 | ACD |
| 2 | BC |
| 3 | ABCD |
| 4 | ADE |

Let $T$ denote the set of transaction identifiers and $I$ denote a set of items. Given a transaction $(t, t.X) \in D$, where $t \in T$ and $t.X \subseteq I$, a transaction $t$ satisfies an item/literal $i \in I$ if $i \in t.X$ and similarly $t$ satisfies the negation of a item/literal $\bar{i}$ if the item $i \notin t.X$. For a literal $l$, the truth value of $l$ in a transaction $t$ is defined as $V_t(l)$ where

$$V_t(l) = \begin{cases} 1 & \textit{if t satisfies l} \\ 0 & \textit{if t does not satisfy l} \end{cases} \tag{2}$$

A boolean expression E is the logical AND or OR of one or more clauses where each clause is either the logical AND or logical OR of one or more literals. A transaction $t$ satisfies a boolean expression E if the truth value of E $(V_t(E))$ evaluates to true when each literal $l$ in E is replaced with $V_t(l)$. The support $s$ of $E$ in the dataset $D$ is the number of transactions that satisfy E, i.e $|t(E)|$. A boolean expression E is frequent if its support is more than or equal to a user specified minimum support $(min\_sup)$ value, i.e. $|t(E)| \geq min\_sup$. One can also define a maximum support threshold $(max\_sup)$ to disallow any expression with too high a support or mine infrequent expressions. For example, in dataset D in table II, there are a total of five items I = $\{A, B, C, D, E\}$ and four transaction ids T = $\{1, 2, 3, 4\}$. The transaction $t_1$ contains the set

of items $\{A, C, D\}$ which is a subset of set of all the items in the dataset. The transactions $t_1$ and $t_3$ satisfy the boolean expression (E) $\{A \wedge C \wedge D\}$ since $V_t(E)$ evaluates to be true in both the transactions. Also, if the $min\_sup$ is 2 then E is a frequent boolean expression.

### C. Mining minimal boolean expressions using BLOSOM

BLOSOM [18] allows for mining frequent boolean AND/OR/CNF/DNF expressions and their minimal generators. The closure operator on partially ordered set $(P)$ is a function $C : P \rightarrow P$ such that $C$ is monotone, idempotent, and extensive. A set $(y)$ is considered closed with respect closure operator $(C)$ if it satisfies the property $C(y) = y$. A set $(x)$ is called a minimal generator of a closed set $(y)$ if it satisfies two properties.

1) The *closure* of set $(x)$ is set $(y)$.
2) No proper subset of $(x)$ generates $(y)$.

The minimal generators can be viewed as succinct descriptors of data. When more than one minimal generator exists for the same subset of data, we refer to them as redescriptions [19]. Redescriptions can be thought of as a generalization of association rule mining from finding implications to equivalences. For example, $\{A, C, D\}$ is a closed AND clause in Table II and the minimal generators for this closed expression are $\{A, C\}$ and $\{C, D\}$. So, they can be represented in terms of redescription (Equation 3):

$$A \wedge C \equiv C \wedge D \tag{3}$$

### D. SAT-based sequential equivalence checking

In SEC, the aim is to prove that the miter output $(o)$, $o = 0$ (the outputs of two circuits are the same) for all states reachable from the alignment state. We note that SEC is a special case of unbounded model checking (UMC) where the property is to verify if the miter output is a constant '0' starting from any reachable state. To benefit from a SAT engine, we regard the problem setup as follows: The unrolled miter circuit is first translated into a boolean formula, with its initial state unconstrained, and the objective is to see if the miter output can be set to '1'. We note that inductive analysis can be applied in this setup so that only the miter output in the last time-frame makes up the objective. More details on this will be given later. If the formula is unsatisfiable, there exists no sequence of states (starting from any arbitrary state) that can make the miter output '1'. However, due to a large don't care space of the system, there may exist a counterexample that can violate the property from some arbitrary initial state and thus one cannot conclude whether the property holds or not. Thus, mining those key and effective constraints in terms of minimal boolean expressions and adding them to prevent false counterexamples can help in proving the two circuits equivalent as quickly as possible.

## III. Proposed Sequential Equivalence Checking Framework

### A. Mining multi node illegal constraints

*1) Constraint mining framework:* First a miter circuit is constructed and simulated for $M$ random input vectors. The logic values on the flip-flops in the circuit are recorded in the mining database such that one dimension lists the flip-flops and the other dimension lists the random input vector number. Unlike previous works on finding two or three node relations, which might be potentially true in the database, we mine potential one, two, three and up to four node relations among flip-flops which are missing from the database.

Given the mining database for the circuit in Table III, the first step is to compute the probability of each flip-flop $f_i$ having logic 1 $({P_i}^1)$ or logic 0 $({P_i}^0)$ using equation (4) or (5).

TABLE III
Mining Database example

| Vector# | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 1 |
| 2 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 1 | 0 |
| 4 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 |

$$ {P_i}^0 = \sum_{i=0}^{M} (f_i = 0)/M \tag{4}$$

$$ {P_i}^1 = \sum_{i=0}^{M} (f_i = 1)/M \tag{5}$$

If $({P_i}^1)$ or $({P_i}^0)$ for flip-flop $f_i$ is 0 than it is a potential 1-node illegal constraint. To compute potential 2 node illegal constraint, if the probability of getting a logic '0' or '1' on both flip-flops ($f_i$ and $f_j$) is below threshold then that logic value combination on the flip-flop pair is searched in the mining database. If it is missing, then it is a potential 2-node illegal constraint. Similarly, for $n$ node illegal constraints, the probability of not getting any of the $(2^n)$ combinations (filtered by threshold criteria) for the flip-flops are computed. For instance, in Table III, assuming the threshold as 0.4, the probability of getting a '0' on $f_2$ and a '1' on $f_3$ is 0.3 ($<$ threshold). Thus, if "01" on $f_2 f_3$ is missing from the database then it is a potential 2-node illegal constraint. The benefit of this approach is its low computational cost because the database is searched only for combinations with probability lower than threshold instead of all the exponential number of combinations possible. To further reduce the mining cost, we propose a two step approach for computing these constraints only among subset of flip-flops out of total $(n)$ flip-flops. In the first step, we compute all the equivalent flip-flop pairs $(m)$. In the second step we create a mining database consisting of the non equivalent flip-flops$(n-m)$ and only one flip-flop from the equivalent pair $(m/2)$ having a low observability value. Lower observability value means it is easier to observe and thus most likely to affect the primary outputs of the miter circuit. So, the constraints are computed only among subset of flip-flops $(n - m/2)$ thus reducing the number of constraints and the time to verify them.

For example, for a circuit with $(n)$ flip-flops $(f_i, f_j, f_k, f_l, f_m, \cdots, f_n)$, if flip-flops $f_i$ and $f_j$ are equivalent, then $f_i$ with higher observability value (less observable) compared to $f_j$ can be removed from the mining database. If there exists an illegal constraint $f_i f_j f_k f_m f_n = 00110$ in the circuit, then the combination 0110 on $f_j f_k f_m f_n$ will be missing from the database and will be captured as a potential illegal constraint. However, if $f_i$ is not removed from the database then two additional and redundant illegal constraints 01110 & 10110 will also be captured, leading to an increase in the count of constraints.

*2) Validity check of mined constraints:* Each mined illegal constraint is verified using the assume-then-verify approach until a fixed point is reached. The miter is unrolled for two time-frames and all the constraints are assumed true in the first time-frame. The initial state variables (pseudo-primary inputs) are unconstrained in the first time-frame and each potential invariant is verified one by one in the second time-frame. If the constraint is a true illegal constraint then there exists no sequence of states that can satisfy the instance and the SAT solver returns UNSAT else it returns SAT. For example, if the potential illegal constraint is $a\bar{b}c$, then $\bar{a_0} + b_0 + \bar{c_0}$ (a,b,c are flip-

flops and subscripts indicate time-frame) is added to the original CNF formula in the first time frame and three clauses $(a_1), (\neg b_1), (c_1)$ are added in the second time frame. If SAT solver returns a SAT solution that satisfies the constraint, then it is dropped and the satisfying assignment is used to discard those potential constraints yet to be verified. However, if the solver returns UNSAT, then it is a true illegal constraint and any constraint that is a superset of it is dropped. For example, if $f_i = 0$ and $f_j = 0$ is a true illegal constraint in the current iteration of induction, then all the potential three/four node constraints such as $f_i = 0$ & $f_j = 0$ & $f_k = 1$, which are a superset of it are also dropped. This incremental verification step significantly reduces the time needed to verify them.

We use zChaff [7] as the underlying SAT solver because it supports incremental SAT solving such that clauses can be added and deleted from the database after each run of the solver based on their group ID. These true global illegal constraints are added in each time frame during equivalence checking and can help in proving the circuits equivalent as quickly as possible.

### *B. Mining redescriptions using BLOSOM*

*1) Constraint mining framework:* Most potential constraints found today are restricted to either two or three node boolean relations in conjunctive form. One of the reasons is that it becomes computationally expensive to exhaustively find all possible relations among flip-flops in circuits. So, we propose to mine complex minimal boolean expression with BLOSOM. The mined relations are general constraints among flip-flops that can constrain a bigger subset of the boolean space then those simple conjunctive-type constraints. First, the mining database is constructed such that flip-flops with higher observability value from each equivalent pair are removed from the database. Then BLOSOM is run on the database with different minimum and maximum support values to obtain potential constraints in the form of redescriptions/equivalences among boolean expressions. For example, the redescription (equation 6) represents equivalences between minimal generators $(f_i \wedge f_j \wedge f_k)$ and $(f_l \wedge f_m \wedge f_n)$ for the same closed set in the dataset. If it is proved as a true constraint then the following relations $1110XX$, $111X0X$, $111XX0$, $0XX111$, $X0X111$ and $XX0111$ on these six flip-flops represent the don't care set and constrain a large subset of boolean space.

$$f_i \wedge f_j \wedge f_k \equiv f_l \wedge f_m \wedge f_n \tag{6}$$

$$(f_i \vee f_j) \wedge (f_k \vee f_i) \equiv (f_l \vee f_m) \wedge (f_n \vee f_m) \tag{7}$$

The second form of redescription (equation 7) represents equivalences between minimal generators in minimal boolean CNF expressions form. Thus if $f_i$ is 1 or $f_j$ is 1 and $f_k$ is 1 or $f_i$ is 1 then $f_l$ is 1 or $f_m$ is 1 and $f_n$ is 1 or $f_m$ is 1 and vice versa. These kind of constraints cannot be obtained by the previous data mining techniques. Interestingly, we observed that some of these constraints may be redundant or unnecessary for our purposes. We categorize them into four types that can be discarded to verify as small set of potential constraints as possible yet powerful enough to prune the search space effectively. It also reduces the time needed to verify these mined relations.

**Type 1:**
If two redescriptions (eg. 8) and (eg. 9) for the two closed sets are proven true then any redescription which is a conjunction of these two redescriptions (eg. 10) is discarded.

$$f_1 \vee f_2 \equiv f_3 \vee f_4 \tag{8}$$

$$f_5 \vee f_1 \equiv f_4 \vee f_6 \tag{9}$$

$$(f_1 \vee f_2 \equiv f_3 \vee f_4) \wedge (f_5 \vee f_1 \equiv f_4 \vee f_6) \tag{10}$$

**Type 2:**
Given the redescription (eg. 12) for the closed set, any redescription (eg. 11) that contains the same variables, but with some or all of them are repeated on either side of the redescription, can be discarded.

$$f_1 \wedge f_2 \equiv f_1 \wedge f_2 \wedge f_3 \tag{11}$$

$$f_1 \wedge f_2 \equiv f_3 \tag{12}$$

**Type 3:**
If a redescription containing $n$ variables is proven true then any redescription containing the same $n$ variables, but with some of them swapped on either side of the redescription, can be discarded. For instance in (eg. 13) $f_3$ and $f_6$ are swapped as shown in (eg. 14).

$$f_1 \wedge f_2 \wedge f_3 \equiv f_4 \wedge f_5 \wedge f_6 \tag{13}$$

$$f_1 \wedge f_2 \wedge f_6 \equiv f_4 \wedge f_5 \wedge f_3 \tag{14}$$

**Type 4:**
If a redescription of size $n$ ($n$ is the count of variables in the equivalence relation) covers less than $(n-1)$ different variables then it is discarded to keep the count of potential equivalences low. In example (15) the redescription size is six but it only covers four flip-flops so it is discarded.

$$f_1 \wedge f_2 \wedge f_3 \equiv f_1 \wedge f_2 \wedge f_4 \tag{15}$$

Removing these four types of redescriptions reduces the mined constraints significantly and the remaining constraints are powerful enough to prove the two circuits equivalent as can be observed from the experimental results. We also provide the count for these four types of redescriptions in the experimental results. For large circuits, running BLOSOM with lower minimum support might lead to significant number of redescriptions due to exponential complexity with increasing count of variables. So, the database is divided into multiple small databases and BLOSOM is run on each of these databases, once with a lower minimum support and a lower maximum support and the second time with a higher minimum support and a higher maximum support. However, constraints computed from a small database that violate transactions in the remaining small databases (called the filtering databases) are dropped.

*2) Validity check of mined boolean relations:* To reduce the time to verify the constraints an approach similar to verification of potential illegal constraints is used. When using assume-then-verify based proof, if the SAT solver returns the SAT solution then it is used to discard any constraint that satisfies it. However, if the SAT solver returns UNSAT then the constraint is true for a subset of boolean space in the current iteration of induction and any constraint that is at least true in that boolean space is also true in the current iteration and thus skipped from verification.

The miter circuit is unrolled for two time frames and assume-then-verify is used to verify the mined relations in conjunctive normal form (CNF). These mined relations cannot be directly converted to CNF clauses. So, extra AND and OR gates are added for each relation being verified. For example, if the potential constraint is $(\bar{f_i} \vee \bar{f_j}) \wedge (\bar{f_m}) \rightarrow (f_k) \wedge (\bar{f_l} \vee f_p)$ then OR gates $(o1)$, $(o2)$, $(o3)$, $(o4)$ are added for $\bar{f_i} \vee \bar{f_j}$ and $\bar{f_l} \vee f_k$ in the first and second time-frames and their corresponding OR clauses are added to the original CNF database. Next $(o1)$ is ANDed with flip-flop $\bar{f_m}$ and an AND gate (a1) is added in the first time-frame. Similarly, $(o2)$ is ANDed with

flip-flop ($f_k$) and an AND gate (a2) is added in the first time-frame. The corresponding AND gates ($a3$), ($a4$) are added in the second time-frame. The AND clauses for ($a1$), ($a2$), ($a3$), ($a4$) are added to the original CNF database. This is also represented by figure 2. The gates ($a1$) and ($a2$) represent the boolean expressions on left and right hand side of the implication in the first time-frame. We then add the clause $(\bar{a1} + a2)$ to the original CNF formula for the first time frame and add the two clauses ($a3$), ($\bar{a4}$) for the second time-frame. If the SAT solver returns UNSAT, then it is a true global constraint else it is dropped.

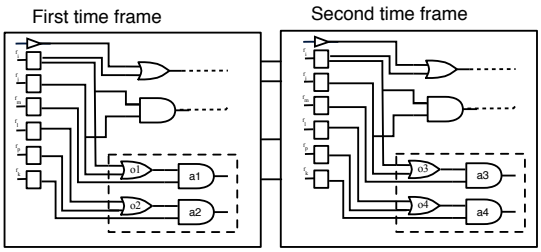


Fig. 2. Two time frame unrolled miter with extra gates for implication

### C. Application to sequential equivalence checking

The true constraints learned are now applied to (unbounded) sequential equivalence checking based on an inductive reasoning framework by adding them in all time-frames. For instance, if an illegal constraint is $\bar{a_0} + b_0 + \bar{c_0}$ (a,b,c are flip-flops and subscripts indicate time-frame) and the miter is unrolled for two time-frames then it is replicated to second time-frame as $\bar{a_1} + b_1 + \bar{c_1}$. In the SEC model, the miter circuit in each time frame consists of both the original and the optimized circuits and is unrolled for $k$ time-frames. First, the base case of induction is done where the pseudo primary inputs are constrained to the initial state and the miter output is checked for '1.' If the SAT solver returns SAT then the circuits are not equivalent. The assignment to the primary inputs serves as a witness to their inequality. If it is UNSAT, then the induction step is done, where the miter is unrolled for $k$ time-frames with the initial state left unconstrained. The miter output is assumed to be '0' in the first $k-1$ time-frames and the output is checked for 1 in the last time-frame. The proven mined relations by our approach are added to all the $k$ time-frames. If the SAT solver returns UNSAT, then the two circuits are equivalent for all reachable states. However, if it returns SAT, then the SAT solution could be a false or true counterexample and nothing can be concluded. So, the miter is unrolled deeper for $k+1$ time-frames and the process is repeated until SAT solver returns UNSAT (the two circuits are equivalent) or an upper bound is reached (assumed 20 in our experiments).

As observed from our experimental results, we are able to prove all the circuits equivalent by adding the true relations found by our method. Thus, we conclude that these general invariants are very powerful and effective in constraining a large portion of the search space and preventing false counterexamples in the inductive step of our SEC.

## IV. Experimental Results

The proposed technique was implemented in C++. All tests were performed on a Intel Core i7 processor, 3.33GHz, with 6GB of RAM, running Linux Ubuntu v9. The ISCAS'89 benchmarks along with their retimed and optimized version obtained either using our script or ABC package [20] are used in our framework. For some of these benchmarks adding equivalent flip-flops as constraints was sufficient to prove them equivalent and so we have not included them in the experimental results and only report few of hard-to-verify instances. Please note that although these benchmarks can also be proved with internal equivalent signals, we avoid injection of these internal equivalences to show the power of our approach. For the retimed circuits that can be proved by previous works and our approach, we only report three of them since that is not the focus of the paper. We were also able to generate few ITC99 benchmarks in our framework where the original design was optimized using gray & one hot encoding followed by state minimization using Synopsys design compiler. These optimized designs have the same functionality as the original design, but had very limited structural similarity. Currently, due to limited availability of these benchmarks that employ powerful sequential optimizations (beyond retiming and simple optimizations), we resort to such an experimental setup. However, should there be more circuits with few or no internal equivalences, we believe our method will offer tremendous value to the current state of art.

Table IV shows the results of using mined general boolean relations for SEC on two different kinds of benchmarks as described above. The first column lists the benchmarks. Those benchmarks marked in bold are optimized ITC99 benchmarks with different state encodings. The second column reports the flip-flops in the original versus optimized circuit. The third column reports the true equivalent flip-flop pairs. The fourth column reports the true internal equivalent signals found. The fifth column reports the subset of flip flops in the mining database after removing one flop from equivalent pair. The sixth column reports if the circuit is provable by internal equivalent signal and flip flop pairs. The seventh column reports if circuit is provable by ABC tool [20]. The eighth and ninth column reports the minimum and maximum support used for mining constraints from the mining database. The tenth column reports the maximum count of flip-flops included in a term in the redescription. The eleventh column reports if the four types of redescriptions (section III B) are removed from all the redescriptions obtained from blosom. The twelfth column reports the count of these redescriptions. The thirteenth column reports the count of potential constraints and the fourteenth column reports the true invariants. The last column reports the total time for mining and proving the potential constraints, adding them to the unrolled miter circuit, and the final SEC checking of the miter output for 1 after adding these constraints. We were able to prove all benchmarks equivalent using our approach.

The results show that some of the ITC99 benchmarks could not be proved equivalent just by internal equivalences and equivalent flip-flop pairs, but could be proved equivalent by our approach. This reflects the effectiveness of our approach based on general boolean expressions where a small number of constraints were effective in proving the two circuits equivalent. For example, in b08_gray_onehot, the b08 circuit with gray-code state encoding was verified against b08 with one-hot encoding. In this instance, one circuit had 21 FFs while the other had 23. There were no equivalent flip-flop pairs. Including the equivalent outputs, there were a total of 683 equivalent signal pairs (none of which involved flip-flops). Because there were no equivalent flip-flops, all 44 (21+23) flip-flops were used in our mining framework. The two circuits could be not proved equivalent using equivalent flip-flops and internal signals, as well as using ABC as reported under **P1** and **P2** columns respectively. Next, a min-support of 300 and max support of 10000 were used to mine potential relations among flip-flops. We had used 4 types of redescription filter as indicated under the column **R2**, with which 4109 implications were found redundant and removed from those found by BLOSOM. The remaining 756 potential implications were validated using an assume-

TABLE IV
MINING AND VERIFICATION RESULTS USING BLOSOM (USING SUBSET OF FLIP-FLOPS.)

| Benchmark | FFs (x/y) | # Equiv FF pairs | # Internal Equiv pairs | Subset of flops | P1 | P2 | Min sup-port | Max support | M | R2 | Red impl | Poten-tial impl | True impl | Time (sec) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| s444_2_3 | 25/30 | 2 | 1746 | 37 | Yes | Yes | 1 | 1000 | 4 | Yes | 184 | 606 | 64 | **8.8** |
| s526_orig_3 | 21/49 | 8 | 529 | 66 | Yes | Yes | 350 | 1000 | 6 | Yes | 17481 | 32166 | 4450 | 432.7 |
| s5378_orig_opt | 179/203 | 620 | - | 321 | Yes | Yes | 1 | 1000 | 4 | Yes | 82250 | 164843 | 135453 | **12036** |
| **b01_gray_hot** | **5/10** | **0** | 195 | **15** | **No** | **No** | **1** | **100** | **6** | **Yes** | **161** | **144** | **92** | **0.23** |
| **b02_gray_hot** | **4/8** | **0** | 54 | **12** | **No** | **No** | **1** | **100** | **6** | **Yes** | **88** | **112** | **81** | **0.19** |
| **b03_gray_hot** | **30/31** | **0** | **47639** | **61** | **Yes** | **Yes** | **100** | **200** | **6** | **Yes** | **294** | **410** | **122** | **2.52** |
| **b04_gray_hot** | **69/70** | **10** | **917** | **134** | **Yes** | **Yes** | **710** | **10000** | **6** | **Yes** | **7604** | **1660** | **395** | **26.3** |
| **b05_gray_hot** | **32/34** | **3** | - | **65** | **No** | **No** | **1** | **100** | **6** | **Yes** | **206781** | **3452** | **1105** | **139.4** |
| **b06_gray_hot** | **9/13** | **2** | **2205** | **20** | **Yes** | **No** | **1** | **100** | **6** | **Yes** | **16208** | **686** | **411** | **1.59** |
| **b07_gray_hot** | **51/53** | **110** | **118880** | **49** | **No** | **No** | **1** | **100** | **6** | **Yes** | **3746** | **18968** | **4344** | **814.1** |
| **b08_gray_hot** | **21/23** | **0** | 683 | **44** | **No** | **No** | **300** | **10000** | **6** | **Yes** | **4109** | **756** | **215** | **3.22** |
| **b09_gray_hot** | **28/30** | **0** | 18161 | **58** | **Yes** | **No** | **1** | **10000** | **6** | **Yes** | **31351** | **189702** | **4407** | **2298** |
| **b10_gray_hot** | **17/24** | **0** | 1150 | **41** | **No** | **No** | **100** | **10000** | **8** | **Yes** | **123013** | **801246** | **73924** | **31841** |
| **b11_syn_gray** | **30/30** | **0** | - | **60** | **Yes** | **Yes** | **85** | **10000** | **6** | **Yes** | **28447** | **93350** | 14538 | **197.3** |
| **b13_gray_hot** | **29/34** | **16** | 2084 | **55** | **No** | **No** | **25** | **10000** | **6** | **Yes** | **44432** | **122640** | **19965** | **8202** |

P1 - Provable by internal equivalent signals and equivalent flip-flop pairs. R2 - Count of four types of redescriptions obtained from BLOSOM.
P2 - Provable by ABC tool from Berkley. [20] M - Maximum number of flip-flops in each boolean term. '-' - indicates time out in 12 hours.

then-verify flow, out of which 215 were found to be true implications. Then, these relations were used in a SEC setup. The total execution time, including mining all the way to SEC was just 3.22 seconds!

Next, for those ISCAS89 benchmarks (original vs. retimed versions), we were able to prove all the benchmarks equivalent. For example, consider s444_2_3, with 25 and 30 flip-flops, there were 2 equivalent flip-flop pairs. Note that a flip-flop $FF_i$ in one circuit could be equivalent to multiple flip-flops in the other circuit. Also, there were 1746 internal equivalent signals, including the corresponding output signals of the two circuits. We note that internal equivalences and flip-flop equivalences were sufficient to prove the two circuits equivalent, as shown under the **P1** column. The remaining columns could be interpreted in a similar manner as we had done earlier. Finally, the two circuits could be proved equivalent in only 8.8 seconds including the time to mine the relations, add then to the SEC framework and prove the two circuits equivalent. Note that in this case, we did not use any internal equivalences during SEC, so that we can test if general boolean expressions were powerful enough. Moreover, in most of benchmarks, setting a higher minimum support reduces the implication count and these implications are sufficient to prove the circuits equivalent in a significantly less time. Note: The benchmark s35932 is not included in the results since the original and optimized version obtained by ABC package [20] can be proved equivalent just with equivalent flip-flop pairs alone.

## V. CONCLUSION

We presented a novel technique for mining complex multi-node boolean relations among flip-flops in sequential circuits. Both illegal constraints among a subset of flip-flops and complex boolean relationships among flip-fops were mined. To the best of our knowledge, this is first of such mining of general boolean expressions for SEC. These relationships can constrain the search space much more effectively than existing approaches. Experimental results show the potential and effectiveness of our mining approach. Our future work is based on extracting quality constraints to reduce the count of these implications which will further reduce the verification time for larger designs.

## REFERENCES


[1] C. Pixley, "A theory and implementation of sequential hardware equivalence," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 11, no. 12, pp. 1469–1478, 1992.

[2] R. Bryant, "Graph-based algorithms for boolean function manipulation," *Computers, IEEE Transactions on*, vol. C-35, no. 8, pp. 677 –691, August 1986.

[3] O. Coudert and J. Madre, "A unified framework for the formal verification of sequential circuits," November 1990, pp. 126 –129.

[4] J. H. Jiang and R. Brayton, "On the verification of sequential equivalence," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 6, pp. 686 – 697, June 2003.

[5] S.-Y. Huang, K.-T. Cheng, K.-C. Chen, C.-Y. Huang, and F. Brewer, "Aquila: an equivalence checking system for large sequential designs," *Computers, IEEE Transactions on*, vol. 49, no. 5, pp. 443 –464, May 2000.

[6] C. van Eijk, "Sequential equivalence checking based on structural similarities," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 7, pp. 814 –819, July 2000.

[7] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: engineering an efficient sat solver," 2001, pp. 530 – 535.

[8] N. Een and N. Sörensson, "An extensible sat-solver [ver 1.2]," 2003.

[9] C. A. J. van Eijk, "Sequential equivalence checking without state space traversal," in *DATE '98: Proceedings of the conference on Design, automation and test in Europe*. Washington, DC, USA: IEEE Computer Society, 1998, pp. 618–623.

[10] F. Lu and K.-T. Cheng, "SEChecker: A sequential equivalence checking framework based on kth invariants," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 6, pp. 733 –746, June 2009.

[11] W. Kunz, D. Pradhan, and S. Reddy, "A novel framework for logic verification in a synthesis environment," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 15, no. 1, pp. 20 –32, January 1996.

[12] W. Kunz, D. Stoffel, and P. Menon, "Logic optimization and equivalence checking by implication analysis," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 16, no. 3, pp. 266 – 281, March 1997.

[13] J.-K. Zhao, E. Rudnick, and J. Patel, "Static logic implication with application to redundancy identification," April 1997, pp. 288 –293.

[14] K. Gulrajani and M. Hsiao, "Multi-node static logic implications for redundancy identification," 2000, pp. 729 –733.

[15] W. Wu and M. Hsiao, "Mining global constraints for improving bounded sequential equivalence checking," 2006, pp. 743 –748.

[16] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," pp. 307–328, 1996.

[17] W. Wu and M. Hsiao, "Mining global constraints with domain knowledge for improving bounded sequential equivalence checking," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 1, pp. 197 –201, January 2008.

[18] L. Zhao, M. J. Zaki, and N. Ramakrishnan, "Blosom: a framework for mining arbitrary boolean expressions," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2006, pp. 827–832.

[19] M. J. Zaki and N. Ramakrishnan, "Reasoning about sets using redescription mining," in *KDD '05: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. New York, NY, USA: ACM, August 2005, pp. 364–373.

[20] (2006). [Online]. Available: Berkeley Logic Synthesis and Verification Group, Berkeley, CA, ABC: A system for sequential synthesis and verification, 2006. [Online]. Available: http://www.eecs.berkeley.edu/ alanmi/abc/