Preprint from Proceedings of GI/ITG/GMM-Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen", Dresden, Germany, February 2010

SAT Modulo BDD
A Combined Verification Approach for Incomplete Designs

Tobias Nopper Christian Miller Matthew Lewis
Bernd Becker Christoph Scholl
{nopper,millerc,lewis,becker,scholl}@informatik.uni-freiburg.de
Institute of Computer Science, Albert-Ludwigs-University, D-79110 Freiburg, Germany

We introduce a novel method that integrates BDD-based symbolic model checking into SAT-based bounded model checking in the presence of unknowns. By incorporating both distinctively different methods, our new hybrid verification tool can take advantage of what each type of model checking has to offer. We present a case study for which both the BDD-based method as well as the SAT-based method are unable to provide a proof that the considered property fails, but that can be solved using the integrated approach.

1 Introduction

Model checking [20, 9] is an important method for verifying whether a sequential circuit satisfies a given temporal property. There are two main approaches to solve the model checking problem. First, unbounded *symbolic model checking* (SMC) [7, 18] makes use of binary decision diagrams (BDDs) [6], which are often able to represent state sets and state traversals that occur in model checking in an efficient form. Secondly, in the last few years, SAT-based techniques like *bounded model checking* (BMC) [4, 3] have also attracted much interest. BMC makes use of a SAT-solver for the satisfiability problem, and due to the enormous performance increase of SAT-solvers over the last few years, SAT-based BMC is now applicable to large industrial designs.

Typically, when performing verification on a circuit, the design is considered to be complete. However, in this work we focus on *incomplete* designs for which some parts of the circuit design are not yet implemented. These unknown parts are combined into so-called blackboxes. There are many applications for model checking incomplete designs, such as early verification checks on unfinished designs, error localization in faulty designs and the abstraction of complex parts of a design in order to simplify the model checking task. Symbolic model checking for incomplete designs [19] is able to provide proofs that a given property is satisfied for all replacements of the blackboxes ('validity') and proofs that the property is not satisfied for any replacement of the blackboxes ('unrealizability'). However, since SMC is a backward-oriented approach only considering backward traversals, SMC may run into difficulties when it explores huge state spaces, even when certain states are not reachable. Incomplete designs can also be processed by bounded model checking [13, 15, 14], providing proofs for unrealizability. However, we will show that BMC for incomplete designs is not as accurate as SMC for incomplete designs.

To combine the strengths of the respective approaches, we investigate a deeply integrated combination (using an approach similar to SAT-modulo-theory solvers) of both for disproving the realizability of invariants: First we perform a number of backward steps by using SMC to gain a set of states from which it is possible to reach states violating the invariant, regardless of the implementation of the blackboxes. Then, BMC is used to check whether it is possible to reach this set of states starting from the initial state independently of the implementation of the blackboxes. If this is possible, unrealizability of the property is proven. Figure 1 illustrates the cooperation of symbolic and bounded model checking. In the following sections, we will show that the combined approach is able to provide proofs for examples which could be solved neither by SMC nor by BMC alone.

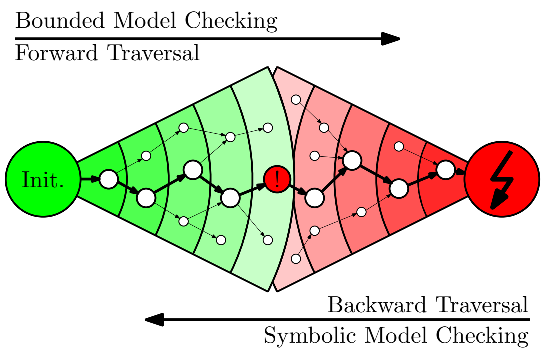

Figure 1: Concept of SAT-modulo-BDD.

Related Work

Several researchers have tried to leverage the advantages of both SAT and BDD before. Gupta et al. [12] performed BDD-based reachability analysis using SAT methods in symbolic image computation. They used BDDs to represent state sets, and a CNF formula to represent the transition relation. This was done by allowing the SAT-solver to find a partial solution, and then invoking a BDD method to map the possible solution space from the current branch in the search tree. The authors extended this idea for BMC in [11] using BDD-based forward reachability analysis to generate conflict clauses to prune the search space of the BMC problem. In other work, [8] used BDD methods to generate an over-approximation of the traces connecting the initial state set to the target state. This estimate was then translated into CNF form to restrict the search space of the SAT-solver working on the BMC problem. Similarly, Bischoff et al. [5] presented a method where they first approximated both enlarged initial and target state sets using BDD-based methods, converting this information into CNF form as a pre-processing step. The SAT-solver would then be run on this BMC problem with fixed enlarged initial and target state sets.

All these methods however, have a few drawbacks. For one, [12] and [11] only used selected variables to build a BDD representation of the possible solution space. Furthermore, [8] and [5] are static approaches. In contrast, the method we introduce here works on the entire problem, is fully dynamic, and we examine its behavior on incomplete designs.

Outline

The paper is structured as follows: After giving a brief review of SMC and BMC for incomplete designs in Section 2, we will introduce a new concept to combine the strengths of SMC and BMC for incomplete designs in Section 3. In Section 4 we will present a case study that is hard to solve for the independent approaches, but that can be solved by the combined method. We confirm the effectiveness and feasibility of our method with preliminary results. Section 5 concludes the paper.

2 Preliminaries

In this section we will give a brief introduction of both SMC and BMC for incomplete designs by means of a small sequential example given in Fig. 2. This circuit has a single input a and two storage elements q_1 and q_0, both initially holding the value 0. The blackbox computes a boolean-valued output Z_0. For the remainder of this paper, we abbreviate the state names for this example as follows: We use (00) for $(q_1 = 0, q_0 = 0)$, (01) for $(q_1 = 0, q_0 = 1)$, etc. Likewise, we use (0) for $a = 0$ and (1) for $a = 1$. The state space is $Q = \{(00), (01), (10), (11)\}$ and the set of possible input values is $A = \{(0), (1)\}$. In this paper, we will consider invariant properties of the form $AG\varphi$, where φ is a boolean expression without any temporal operators. The invariant we will use for this example is $AG(\overline{q}_0 \vee \overline{q}_1)$ stating that q_0 and q_1 never hold the value 1 at the same time. Since this invariant can be proven to be unsatisfied for all substitutions of the blackbox, the property is *unrealizable*.

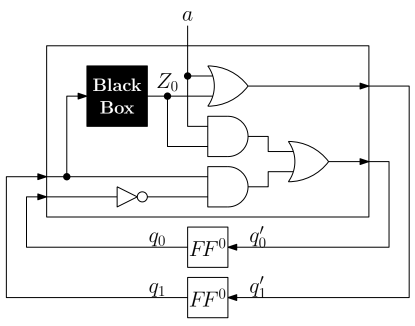

Figure 2: An exemplary sequential circuit with one blackbox. Initially, $q_1 = q_0 = 0$.

Symbolic Model Checking for Incomplete Designs When SMC for incomplete designs [19] is applied to an invariant, first the set of *possible transitions* between the states is computed. For instance, for state (00) and input $a = 1$ the next state may be either (10) (in case that the blackbox computes $Z_0 = 0$ in this situation) or (11) (in case that the blackbox computes $Z_0 = 1$ in this situation). We call both transitions $((00),(1),(10))$ and $((00),(1),(11))$ *possible transitions*, since they may or may not be present depending on the implementation of the blackbox. The complete set of possible transitions $R_E \subseteq Q \times A \times Q$ in this example is depicted in Fig. 3. Starting from the states directly violating the invariant, SMC then iteratively computes the set of 'bad' states for which, independently of the blackbox behavior, there is an input sequence leading into a state violating the property.

For our example, the initial set of bad states for which this holds is $\{(11)\}$, since this is the only state directly violating the property. In the next step, (10) can be added to the bad states, since there is an input value ($a = 1$) that leads from (10) to (11), independent of the value the blackbox computes. In the next step, both (00) and (01) can be added to the bad state set, since for both, there is an input value ($a = 1$) that leads either to (10) (in case that the blackbox computes $Z_0 = 0$ in this situation) or (11) (in case that the blackbox computes $Z_0 = 1$). Since we have shown that for initial state (00), there is an input sequence for each blackbox substitution such that a state not satisfying the invariant is reached, the unrealizability of $AG(\overline{q}_1 \vee \overline{q}_0)$ has been proven.

There are several possibilities to model the behavior of blackboxes in their environment, from a symbolic version of ternary $(0,1,X)$-simulation [2] to using distinct Z_i-variables (as already shown above for the small example in Fig. 2). The different methods to handle the blackbox outputs lead to different approximations of the set of possible transitions. Based on this, [19] defines a symbolic model checking procedure (for arbitrary CTL formulas) that is able to falsify realizability problems and to prove validity problems for designs with blackboxes.

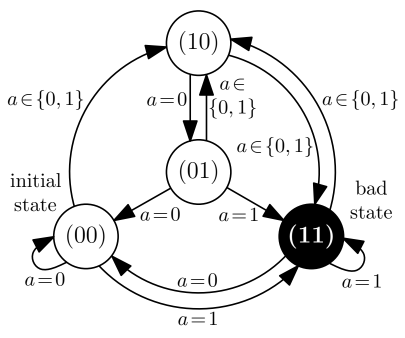

Figure 3: Possible transitions for the incomplete design in Fig. 2 as considered by SMC.

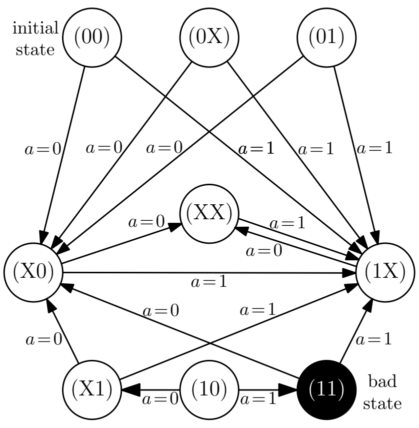

Figure 4: Transitions for the incomplete design in Fig. 2 in Q_X as considered by BMC.

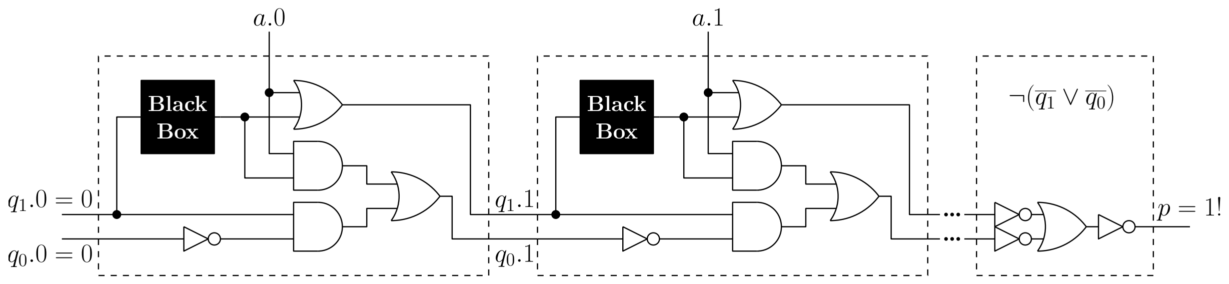

Figure 5: Unfolding of incomplete design in Fig. 2.

Bounded Model Checking for Incomplete Designs BMC for incomplete designs [13, 15, 14] iteratively 'unfolds' the circuit under consideration and checks whether the given property can be disproved for the current unfolding. Figure 5 illustrates an unfolding of the circuit given in Fig. 2. The state signals of the first unfolding $\vec{q}.0$ are set to their initial value and a representation of the negated invariant is connected to the final state bits $\vec{q}.k$.

To handle unknown signals as blackbox outputs, the Boolean logic is extended to include an unknown value X [13, 15, 14] [1]. For handling the three logical values 0, 1, and X, we use the encoding as proposed by Jain et al. [16], applying a binary encoding and extending this encoding to the operators. The encoded circuit and the property is Tseitin-transformed [22], introducing additional variables for internal signals. The Tseitin-transformation preserves satisfiability and results in a CNF whose size is linear with respect to the number of unfoldings. The final CNF is satisfiable iff there is an assignment to the inputs such that the last state of the unfolding violates the invariant. In that case, unrealizability is proven. For this a SAT-solver can be used to evaluate the CNF.

Since the values of the state variables in the j-th unfolding $\vec{q}.j$, are allowed to hold the value X (e.g. $a.0 = 0$ implies $q_1.1 = X$), one can consider the state space to include X values as well:

$$Q_X = \{0, 1, X\}^{|\vec{q}|}$$

All assignments to the state variables that are potentially considered by BMC together with the according transitions are depicted in Fig. 4. Additionally, to the abbreviations introduced above, we use $(1X)$ for $(q_1 = 1, q_0 = X)$, $(X0)$ for $(q_1 = X, q_0 = 0)$ etc. Note that for this example, it can be seen that BMC will never consider (11) to be a reachable state since there is no path from the initial state to the bad state. Thus, conventional BMC for incomplete designs is unable to prove the unrealizability of this example.

2.1 Comparison of SMC and BMC

As shown by the example, BMC is not as exact as SMC, in the sense that it is not able to prove unrealizability in all cases that SMC can. One reason for this is that BMC for incomplete designs considers the question "Is there an input sequence such that for all substitutions of the blackboxes a state violating the invariant is reached?". In contrast to that, SMC considers the unrealizability question: "Does for all blackbox substitutions exist an input sequence leading to a state violating the invariant?". In the latter case the input sequence can 'react' to the behavior of the blackbox and does not necessarily have a fixed length as for BMC, and thus SMC is more accurate. Furthermore, the state set $\{(00), (01), (10)\}$ can only be described as (XX) in BMC. This can have the effect that for incomplete designs it is unknown whether a property is violated or not (due to Xs appearing in the state vector). Although, in these cases, SMC is sometimes able to provide a result, since it does not approximate sets of states using X-values (even if $(0, 1, X)$-logic is used for approximating transitions in the system).

[1] We do not consider more advanced modelings like Z_i-modeling as in [15], since this results in QBF problems that are hard to solve.

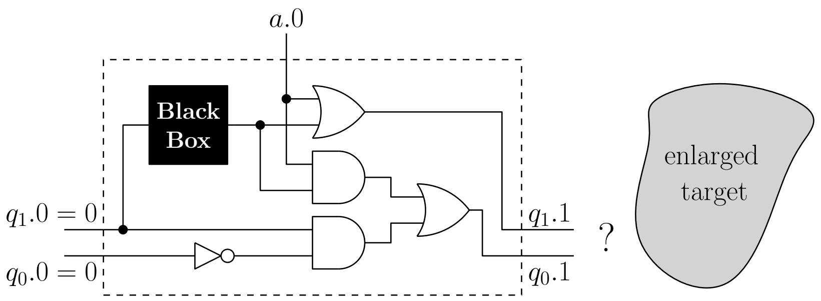

Figure 6: Example in Fig. 2 after one unfolding.

3 SAT Modulo BDD

In this section we consider how the advantages of the two methods can be combined such that the resulting approach is both faster and more accurate than the respective methods alone.

First, SMC performs a number of backward traversals as described in Section 2 until a given time bound is reached. For all states in the computed state set the following holds: For all substitutions of the black-boxes, there is a input sequence such that, starting from the current state, a state violating the invariant is reached. In analogy to [5], we call this set of states $ET \subseteq Q$ the 'enlarged target' which we want to 'hit' using BMC. In a second step, BMC iteratively builds unfoldings of the sequential circuit, yet *without* including the property. Each unfolding is Jain-encoded and Tseitin-transformed. The resulting CNF is obviously satisfiable for all possible assignments to the inputs. To prove the unrealizability of the invariant, it is necessary to find an assignment to the inputs such that the resulting assignment to the final state variables $\vec{q}.k$ is within ET (i.e. the state variables after the last unfolding k hits the enlarged target). Thus, the SAT-solver that processes the CNF provided by BMC, has to be aware of the enlarged target state set and has to check each satisfying assignment of the CNF formula whether the states represented by the assignment of the final state variables $\vec{q}.k$ lie inside the enlarged target.

The current assignment $\vec{\epsilon} \in Q_X$ to the final state variables $\vec{q}.k$ represents a set of states $F(\vec{\epsilon}) \subseteq Q$:

$$F(\vec{\epsilon}) := \left\{ (q_{n-1}, \ldots, q_0) \;\middle|\; q_i \in \begin{cases} \{1\} & : \epsilon_i = 1 \\ \{0\} & : \epsilon_i = 0 \\ \{0,1\} & : \text{else} \end{cases} \right\}$$

This set $F(\vec{\epsilon})$ can be compared against the enlarged target set ET. If $F(\vec{\epsilon})$ lies completely inside the enlarged target ($F(\vec{\epsilon}) \subseteq ET$), the unrealizability has been proven. Else, the current assignment does not (completely) hit the enlarged target and we have to perform a backtrack. Additionally, we add a new clause prohibiting that the assignment $\vec{\epsilon}$ is considered again. If all satisfying assignments to the CNF formula have been considered without finding a resulting assignment to the final state variables that hits the enlarged target, the number of unfoldings is increased by 1.

Note that in case that SMC is able to prove that there is an initial state in the enlarged target, the unrealizability is already proven. On the other side, if SMC reached a fixed point while computing the enlarged target and there is no initial state in that set, then BMC will not be able to find an input sequence leading into the enlarged target and thus the unrealizability cannot be proven. In both cases, it is unnecessary to invoke BMC.

Example We again use our example given in Fig. 2 together with the invariant $AG(\overline{q}_1 \vee \overline{q}_0)$. We assume that SMC performs 1 backward traversal step. As shown in Section 2, the set known to violate the property for all blackbox replacements in at most one step is $\{(10), (11)\}$. Accordingly, the enlarged target in this case is $ET = \{(10), (11)\}$. Figure 6 depicts the example circuit after one unfolding. Assuming that the SAT-solver decides to set $a.0$ to 1, the final state variables $q_1.1$ and $q_0.1$ hold the values 1 and X, respectively. The state set represented by this assignment is

$$F(1,X) = \{(10),(11)\} \subseteq \{(10),(11)\} = ET$$

and thus, the property is proven to be unrealizable.

3.1 Building Conflict Clauses

In the case that the SMC evaluation returns the result that the current assignment $\vec{\epsilon}$ does not completely lie inside the enlarged target ET, i.e. the state set $F(\vec{\epsilon}) \not\subseteq ET$, a conflict clause that forbids this assignment has to be created. In the most naive approach we could produce a conflict clause forbidding this exact assignment. This could be achieved by adding a clause modeling the following condition:

$$\bigvee_{i=0}^{|\vec{q}|-1} \begin{cases} (q_i = 0) \vee (q_i = X) & : \epsilon_i = 1 \\ (q_i = 1) \vee (q_i = X) & : \epsilon_i = 0 \\ (q_i = 0) \vee (q_i = 1) & : \epsilon_i = X \end{cases} \tag{1}$$

However, the conflict clause based on (1) would not be very general. To produce a more general conflict clause, we can perform the steps illustrated in Fig. 7. The goal of this routine is to truncate as much of the remaining search space as possible. This is done by first replacing some of the X values in $\vec{\epsilon}$ by values 0 or 1, until the modified state space $F(\vec{\epsilon}^A)$ lies completely *out*side of ET. As few as possible X values should be replaced by 0 or 1, since then, the state set represented by this assignment is larger an thus, more of the search space can be truncated. Next, a more general assignment $\vec{\epsilon}^B$ is computed from $\vec{\epsilon}^A$ while still maintaining the property that $F(\vec{\epsilon}^B) \subseteq Q \setminus ET$. This is accomplished by changing the value of some variables in $\vec{\epsilon}^A$ which were 0 or 1 to X, and then testing to see if $F(\vec{\epsilon}^B)$ is still completely outside of ET. Again, the goal is to have as many X's in the assignment as possible. For both steps, the selection of the values to change is made by a heuristic based on the symbolic representation of F; details follow in section 3.3.

The resulting assignment ϵ^B represents a significantly larger part of the state space than $\vec{\epsilon}^A$. In some cases, we observed $\vec{\epsilon}^B$ with an order of magnitude less of assigned variables then $\vec{\epsilon}^A$. Using $\vec{\epsilon}^B$ instead of $\vec{\epsilon}^A$ to generate conflict clauses result in clauses that are substantially smaller, making them more useful in aiding the SAT-solvers search.

Obviously, every assignment $\vec{\epsilon}'$ that lies *in*side ET has to differ in *at least one* value compared to $\vec{\epsilon}^B$. However, an assignment $\vec{\epsilon}''$ with $\epsilon''_i = \epsilon^B_i$ except $\epsilon^B_j = 1$, $\epsilon''_j = X$, will not lie inside ET due to:

$$F(\vec{\epsilon}^B) \subset F(\vec{\epsilon}'') \text{ and } F(\vec{\epsilon}^B) \subseteq S \setminus ET$$

Same holds for $\epsilon^B_j = 0$, $\epsilon''_j = X$. Since these assignments cannot be used to prove unrealizability, they are not of interest, and will be forbidden. Accordingly, for all $\vec{q}$ with $F(\vec{q}) \subseteq ET$ the following condition (2) has to hold:

$$\bigvee_{i=0}^{|\vec{q}|-1} \begin{cases} q_i = 0 & : \epsilon^B_i = 1 \\ q_i = 1 & : \epsilon^B_i = 0 \\ - & : \epsilon^B_i = X \end{cases} \tag{2}$$

The original assignment to the final state variables $\vec{\epsilon}$ does not satisfy this condition, since no single ϵ_i has the value that is requested in the condition: If $\epsilon_i = 0$, then $\epsilon^A_i = 0$ and $\epsilon^B_i \in \{0, X\}$. Due to that, q_i will either be required to be 1, or it will not appear in the condition at all. $\epsilon_i = 0$ satisfies neither. Analogously for $\epsilon_i = 1$. If $\epsilon_i = X$, then $\epsilon^A_i \in \{0,1,X\}$ and $\epsilon^B_i \in \{0,1,X\}$. Thus, q_i will either be required to be 1 in the condition, will be required to be 0 in the condition or it does not occur in the condition at all; the assignment $\epsilon_i = X$ satisfies neither. Due to that, the clause that can be build by the translation of condition (2) into Jain's encoding, is a conflict clause that can be added to the clause set and will invoke a backtrack.

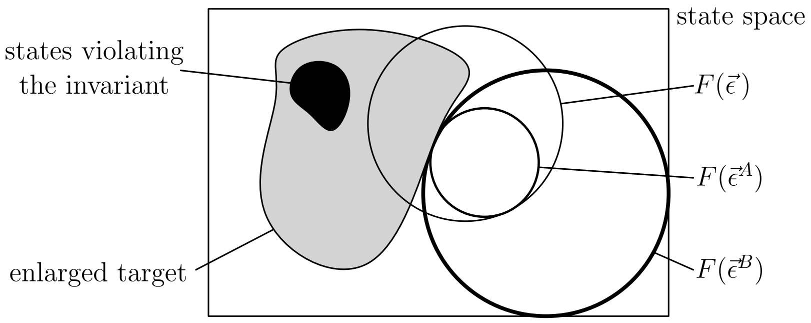

Figure 7: Illustration of conflict clause generation.

Example Again, consider the example in Fig. 6. We assume that the SAT-solver decided to assign 0 to $a.0$, leading to the final state assignments $\vec{\epsilon} = (X, 0)$ with $F(X, 0) = \{(00), (10)\}$. As we have seen above, the enlarged target after one preimage computation is $ET = \{(10), (11)\}$. Since $F(0, 0) \not\subseteq ET$, the unrealizability is not proven and a conflict clause has to be computed. First, the X's in $\vec{\epsilon} = (X, 0)$ are replaced with a boolean value such that $\vec{\epsilon}^A$ lies outside ET. In this case, $\vec{\epsilon}^A = (0, 0)$. Then, all values in $\vec{\epsilon}^A$ not equal to X are checked whether they can be replaced with X such that the resulting state set still lies outside ET. For this example, the second 0 can be replaced by X and thus $\vec{\epsilon}^B = (0, X)$. Following this consideration, a conflict clause can be added that forces q_1 to value 1.

3.2 Early comparison against the enlarged target

It is not necessary to restrict the comparison of an assignment to the final state to the case that the complete CNF is satisfied (i.e. after a valid path of length k has been found). Instead it is reasonable to additionally query the symbolic model checker about *incomplete* final state assignments, since it may already be possible to prove that the state set represented by the incomplete assignment does not intersect with the enlarged target and thus, the current candidate assignment can be discarded. In our implementation, the check was invoked every second time a variable in $\vec{q}.k$ was assigned to a new value.

3.3 Symbolic Representation

We make use of the fact that the enlarged target ET is computed by a symbolic model checker and thus represented by a symbolic function χ_{ET} and all the operations for evaluating a final state variable assignment and for conflict clause computation can be mapped to efficient BDD operations. The set of states represented by the assignment $\vec{\epsilon}$ to the final state variables $\vec{q}.k$ can be represented by a cube of state variables:

$$\chi_{F(\vec{\epsilon})}(\vec{q}) = \bigwedge_{i=0}^{|\vec{q}|-1} \begin{cases} q_i & : \epsilon_i = 1 \\ \overline{q}_i & : \epsilon_i = 0 \\ 1 & : \text{else} \end{cases}$$

Since $\chi_{F(\vec{\epsilon})}$ is a cube, the BDD's cofactor operation can be used for fast comparison of $\chi_{F(\vec{\epsilon})}$ against χ_{ET}: If $\chi_{ET}|_{\chi_{F(\vec{\epsilon})}} = 1$, unrealizability is proven; if $\chi_{ET}|_{\chi_{F(\vec{\epsilon})}} \neq 1$, a conflict clause has to be added and a backtrack has to be performed. In the latter case, we first compute the set of states that are in $F(\vec{\epsilon})$ but not in ET:

$$A(\vec{q}) := \overline{\chi_{ET}} \cdot \chi_{F(\vec{\epsilon})}$$

Every implicant B of A represents a set $F(\vec{\epsilon}^A)$ with properties as described in Section 3.1. It is natural to pick the 'biggest' such implicant containing the least variables. The removing of variables assignments to rule out an even larger state set corresponds to the existential quantification of variables in cube B, so as many state variables as possible are removed from B until B is a prime implicant of $\overline{\chi_{ET}}$. Finally, the conflict clause can be generated according to the values of the state variables as implied by B in analogy to Section 3.1.

4 Case Study

The case study we will use to evaluate our approach is based on the rail segment control from the Funk-FahrBetrieb (FFB) specification of the Deutsche Bahn [10], which is closely related to the European Train Control System level 2/3 Movement Authority [1]. The case study models a network of segments, gates and trains that move on these segments according to a given schedule.

The main concept of this benchmark is as follows: Each train sends requests to the current and subsequent segments (determined by the train's schedule). Some time later, the segments will grant permission to the train. A train only moves on segments it has permission to. If a train has moved from a segment, it no longer sends a request. Gates are raised and lowered as required. Figure 8 illustrates the communication between trains and segments or segments and gates, respectively. The schedules of the two trains, depicted in Fig. 9, are identical. Both trains follow the route depicted below, where SB, SBOL,..., FR represent the different track segments.

$$\text{Holding track} \rightarrow \text{SB} \rightarrow \text{SBOL} \rightarrow \text{OL} \rightarrow \text{SB} \rightarrow \text{SBFR} \rightarrow \text{FR} \rightarrow \text{SB} \rightarrow \text{SBOL} \rightarrow \ \dots$$

The implementation of the process communication uses messages containing sender and receiver IDs as well as the actual information to be sent. In each process, queues are used to store messages and internal information. Due to these queues, only a small fraction of the state space is reachable, even when there are blackboxes with unknown behavior in the design. This turns out to be a disadvantage for backward-oriented methods like SMC, since they are unaware of the set of reachable states and thus compute *all* states satisfying a given property. We considered an invariant stating that two trains will never be on the same segment for four scenarios:

FFB-A-1 and -2: The SB segment contains the following error: After a train has requested a permission for the segment SB, and this request was granted, then normally the *reserved* flag of this rail segment is set. In the erroneous case, this flag is not set properly. For FFB-A-1, the flag is set to the constant 0, and for FFB-A-2, the value of the flag computed by an subcircuit with a blackbox. In both cases, two trains may move to the same rail segment which would result in a collision. All rail segments besides SB, and all gates are put into blackboxes.

FFB-B-1 and -2: The SBOL segment fails to set its reserved flag properly as in FFB-A-1 and -2; all segments except SB and SBOL are blackboxed, as well as the gate connected to SBFR. Again, this could result in a serious collision.

4.1 Results

Table 1 gives preliminary results for the FFB benchmarks just described. We combined the DPLL-based SAT-solver *Mira* [17] with *MIND* [19], a CUDD-based [21] symbolic model checker for incomplete de-

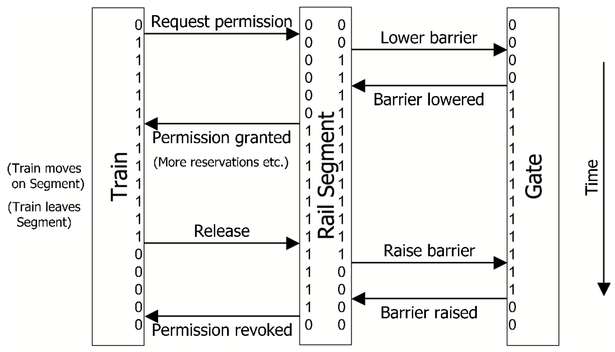

Figure 8: Illustration of the communication between trains, segments and gates.

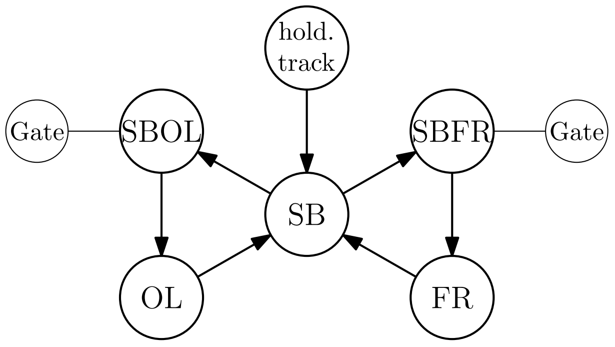

Figure 9: Illustration of the trains' schedule in case study.

signs. The evaluation was performed on a 2.6 GHz AMD Opteron with 4 GB main memory and a timeout limit of 12000 CPU seconds. For SMC only, the table gives the number of backward steps, the peak number of BDD nodes, and the required CPU time in seconds. For BMC only, the table gives the number of forward steps, the peak number of clauses and the CPU time in seconds, respectively. Finally, for the combined approach, the proportionate values for SMC and BMC as well as the overall CPU time in seconds are given.

When standard SMC is applied to this case study, it is not able to prove the unrealizability for FFB-A-1 and FFB-A-2 before exceeding the timeout limit of 12000 seconds. Due to that, we first computed an overapproximation of the possibly reachable states and used this state set to restrict the backward model checking procedure. By doing so, we were able to prove the unrealizability of the considered property in 31.23 seconds and 23.22 seconds, respectively. However, the more difficult scenarios FFB-B-1 and FFB-B-2 are still out of reach (in Table 1 indicated by *TO*). On the other hand, BMC was not able to successfully provide a proof of unrealizability for the FFB-A-2 and FFB-B-2 benchmarks (in Table 1 indicated by *NS*). This is due to the broader abstraction used by 01X-BMC as compared to SMC. Using the combined approach FFB-A-2 and FFB-B-2 can be solved in reasonable time.

The best division of SMC and BMC in our tool was determined experimentally, with the basic goal of using SMC as much as possible to maintain exactness. As can be seen from the combined method's results, the number of nodes explodes after around 5 time steps, making any additional expansion of the enlarged target very expensive. This is why the SMC approach alone is not able to solve the last two problems. As for the BMC part of the combined technique, the number of clauses is significantly higher than that when using only BMC. The difference is the clauses that the combined approach has learnt from the BDD comparisons of the enlarged target. These clauses decidedly aid the BMC search. Overall, the combined method shows much promise.

bench-mark	SMC only			BMC only			SMB+BMC						
							SMC part			BMC part			overall
	#steps	max #nodes	time	#steps	max #clauses	time	#steps	max #nodes	time	#steps	max #clauses	time	time
FFB-A-1	10	252469	31.23	10	25305	1.37	4	170273	4.31	6	31883	1.39	5.70
FFB-A-2	10	191798	23.22	-	-	NS	4	170560	4.45	6	32691	1.35	5.80
FFB-B-1	-	-	TO	25	99141	10772.16	5	1353759	221.12	20	142052	375.46	596.58
FFB-B-2	-	-	TO	-	-	NS	5	2078530	205.57	22	162477	861.37	1066.94

Table 1: Experimental results.

5 Conclusions and Future Work

In this paper we presented a combined approach using SMC and BMC for verifying incomplete designs. This is done by first computing a set of states from which the target states can be reached independently of the blackbox behavior using BDD-based methods. Then SAT-based BMC is invoked to try and 'hit' this enlarged target supported by an advanced BDD/SAT-based conflict clause computation. This technique combines the speed of BMC and the accuracy of SMC allowing us to solve benchmarks for which both SMC and BMC fail.

In the future, we would like to extend our method to falsify properties $AG\varphi$ with φ being an arbitrary CTL formula. Here the enlarged target is a set of states violating φ for any blackbox substitution.

References

[1] ERTMS/ETCS - Class 1 — system requirements specification, chapter 3 (principles), subset-026-3, issue 2.2.2. 2002.

[2] M. Abramovici, M. Breuer, and A. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[3] A. Biere, A. Cimatti, E. M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *IEEE Design Automation Conf.*, 1999.

[4] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *LNCS*. Springer Verlag, 1999.

[5] G. P. Bischoff, K. S. Brace, G. Cabodi, S. Nocco, and S. Quer. Exploiting target enlargement and dynamic abstraction within mixed bdd and sat invariant checking. *Electr. Notes Theor. Comput. Sci.*, 119, 2005.

[6] R. Bryant. Graph - based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[7] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, 98(2):142–170, 1992.

[8] G. Cabodi, S. Nocco, and S. Quer. Improving sat-based bounded model checking by means of bdd-based approximate traversals. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, page 10898, Washington, DC, USA, 2003. IEEE Computer Society.

[9] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite–state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, 1986.

[10] A. Deutsche Bahn AG. *Betriebliches Lastenheft FunkFahrBetrieb (FFB) 413.9210*, version 2.0, 01.10.1996 edition, 1997.

[11] A. Gupta, M. Ganai, C. Wang, Z. Yang, and P. Ashar. Learning from bdds in sat-based bounded model checking. In *DAC '03: Proceedings of the 40th annual Design Automation Conference*, pages 824–829, New York, NY, USA, 2003. ACM.

[12] A. Gupta, Z. Yang, P. Ashar, and A. Gupta. Sat-based image computation with application in reachability analysis. In *FMCAD '00: Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design*, pages 354–371, London, UK, 2000. Springer-Verlag.

[13] M. Herbstritt and B. Becker. On SAT-based Bounded Invariant Checking of Blackbox Designs. In *Proc. of Microprocessor Test and Verification Workshop (MTV)*, Austin (TX), USA, November 2005. IEEE Computer Society.

[14] M. Herbstritt and B. Becker. On Combining 01X-Logic and QBF. In *Proceedings of 11th International Conference on Computer Aided Systems Theory (EUROCAST)*, volume 4739 of *LNCS*, Las Palmas de Gran Canaria, Canary Islands, Spain, February 2007. Springer-Verlag.

[15] M. Herbstritt, B. Becker, and C. Scholl. Advanced SAT-techniques for bounded model checking of blackbox designs. In *Proc. of Microprocessor Test and Verification Workshop (MTV)*, Austin (TX), USA, December 2006. IEEE Computer Society.

[16] A. Jain, V. Boppana, R. Mukherjee, J. Jain, M. Fujita, and M. Hsiao. Testing, Verification, and Diagnosis in the Presence of Unknowns. In *VLSI Test Symp.*, pages 263–269, 2000.

[17] M. Lewis, T. Schubert, and B. Becker. Multithreaded SAT Solving. In *12th Asia and South Pacific Design Automation Conference*, 2007.

[18] K. McMillan. *Symbolic Model Checking.* Kluwer Academic Publisher, 1993.

[19] T. Nopper and C. Scholl. Approximate symbolic model checking for incomplete designs. In *Formal Methods in Computer-Aided Design*, pages 290–305, Nov 2004.

[20] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[21] F. Somenzi. *CUDD: CU Decision Diagram Package Release 2.4.0.* University of Colorado at Boulder, 2004.

[22] G. Tseitin. On the complexity of derivations in propositional calculus. In A. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logics*. 1968.