A Symbolic Approach for Mixed-Signal Model Checking

Alexander Jesser Lars Hedrich
Department of Computer Science
J. W. Goethe University Frankfurt a.M.
D-60325 Frankfurt a.M., Germany
{jesser,hedrich}@em.cs.uni-frankfurt.de

Abstract— In this paper we firstly introduce a novel symbolic model checker (*MScheck*) for mixed-signal circuits. *MScheck* is capable to conflate the continuous behavior, typical for analog designs, and the discrete behavior in the digital domain for formal verification. Timing information of both systems will be symbolically stored within multi terminal binary decision diagrams (MTBDDs) for the entire verification procedure. The effectiveness of our approach is demonstrated on a phase locked loop (PLL) by formal verification of the locking property[1].

I. Introduction

Todays microelectronic systems are characterized by an increasing level of integration complexity. In recent years multi circuit boards have been evermore integrated in few or ideally one single chip. Such systems are called System on Chip (SoC) and consist usually of a collection of digital and analog subcircuits. Generally, circuits that are divided into an analog and a digital part are often called mixed-signal systems.

One characteristic of mixed-signal systems is that each subsystem interacts with each other by internal connections and reacts to inputs coming from extern. Another characteristic is different state change behavior in both domains. Digital systems behavior usually exhibits discrete changes in time and values, whereas analog circuits usually exhibit continuous changes. The integration of both sub-circuits in one chip is an advanced but widely used method in chip design.

Because of this increase in complexity, the likelihood of subtle functional errors is much greater. Today mixed-signal circuit validation can be done by several analog/mixed-signal (AMS) simulators that only give an input pattern dependent, incomplete correctness of the circuit. To overcome todays design complexity formal proof techniques that guarantee total correctness must be evaluated. One common formal verification technique is model checking. Model checking tools prove the model of a design against a number of stated properties that the system must fulfill under all possible input variations.

A. Previous Work

The basic methodologies used by todays digital formal verification tools based upon symbolic representations like binary decision diagrams (BDD) or applying satisfiability (SAT) solvers mostly used in bounded model checkers (BMC). However, analog and mixed-signal formal verification is much more crucial. Nevertheless, in recent years several approaches for verification of mixed- and hybrid systems were proposed. Most of them base on the finite state discrete abstraction by partitioning the continuous state space, whose dynamic is described by differential algebraic equations (DAE), into hypercubes using a fixed grid. The most popular tool is called *HyTech* and was introduced in [1]. *HyTech* is a symbolic model checker for linear hybrid automata that can be analyzed automatically using polyhedral state sets. In [2] the tool *d/dt* is proposed which computes the reachable states for hybrid systems by discrete time integration. Set of states are represented with orthogonal polyhedra, which guarantees the reachable state enclosures. *PHAVer* (Polyhedral Hybrid Automaton Verifier) [3] is another tool that do an on-the-fly overapproximation of piecewise affine dynamics and by partitioning the state space based on user definable constraints of the systems.

All the above mentioned tools are restricted to circuit properties like the linearity of the analog part or can only do reachability analysis. Furthermore, they suffer from explicitly handling and describing each digital state. In this work we introduce our tool *MScheck* that can verify mixed-signal circuits by using symbolic model checking techniques which represent states implicitly. We are capable to treat non-linear analog circuits integrated with a sequential digital circuit part at gate level. We used multi terminal binary decision diagrams (MTBDD) to efficiently represent time behavior within the circuit. Additionally, we used CTL-AT for defining mixed-signal properties.

The following sections discuss the verification flow we used for mixed-signal model checking. Section II gives a formal definition of the specification language CTL-AT we used for characterizing necessary circuit behavior. In section III a short introduction to the symbolic representation of timed delayed transition relations and state declarations will be given. The principles of the verification flow will be given in sections IV and V. In section VI we demonstrate our verification flow by verifying the behavior of a phase locked loop (PLL). Finally, a conclusion will be given in section VII.

II. Specification Language

In digital circuit verification some common formal specification languages exists. All these languages are based on *linear*

[1]This work has been supported by BMBF/edacentrum project FEST under No. 01M3072C.

time logic (LTL) or *computation tree logic* (CTL). The upto now proposed specification languages are able to analyze timing behavior for digital circuits based on a state transition graph called Kripke structure [4]. To specify dynamic system behavior with a CTL syntax, it is necessary to introduce time-constrained temporal operators that additionally constrain the scope of the operations. To extend Kripke structures with transition delay time information and interface variables we introduce the *time delayed transition structure*.

Definition 1 (Time Delayed Transition Structure) *Let $AP = \{p_1, p_2, \dots\}$ be a set of atomic propositions. A time delayed transition structure (TDTS) is a six tuple $\mathcal{M} = (S, S_0, \delta, G, T, L)$ where*

- *S is an infinite set of states.*
- *$S_0 \subseteq S$ is the set of initial states.*
- *G is a finite set of input symbols.*
- *$\delta \subseteq S \times S \times G$ is a dependent transition relation that must be total, that is, for every state $s \in S$ there is a successor state $s^+ \in S$ such that $\delta(s, s^+, g)$ with $g \in G$.*
- *$T : \delta \longrightarrow \mathbb{R}$ is a function that labels each transition from state s to s^+ with the delay time using this transition.*
- *$L : S \longrightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state.*

A path π in a TDTS is an infinite sequence of states $s_0, s_1, \dots$ in $\mathcal{M}$ with $(s_i, s_{i+1}, g_i, t_i) \in \delta \times T$ for every $i \geq 0$. With $\pi[i]$ we denote the part of π starting at s_i. In the following, each state s_i can be encoded by a binary coding vector $\vec{z} \in \mathbb{B}^n$ which gives each state a unique representation.

In [5] a timed constraint CTL (CTL-AT) specification language especially for analog properties was introduced. At this point we want to define CTL-AT more formally. We write $(\mathcal{M}, s_0) \models p$ to express that in a time delayed transition structure $\mathcal{M}$ formula p is true for starting state s_0. The formal semantics of CTL-AT will be given in Definition 2.

Definition 2 (Time Constrained CTL Semantics) *Let $\mathcal{M}$ be a TDTS, π be a path starting with the state s_0 in $\mathcal{M}$, $[t_l, t_h]$ be a time interval with the boundaries $t_l, t_h \in \mathbb{R}^+$. Hence, ϕ and ψ be CTL-AT formulas and $p_i \in AP$ be an atomic proposition. Then we will define the following Time Constrained CTL model relations:*

$$\begin{aligned}
(\mathcal{M}, s_0) &\models p_i && \Leftrightarrow p_i \in L(s) \\
(\mathcal{M}, s_0) &\models \neg\phi && \Leftrightarrow \pi \not\models \phi \\
(\mathcal{M}, s_0) &\models EF_{[t_l,t_h]}(\phi) && \Leftrightarrow \exists\pi\ \exists i : t_i \in [t_l, t_h], \pi[i] \models \phi \\
(\mathcal{M}, s_0) &\models EG_{[t_l,t_h]}(\phi) && \Leftrightarrow \exists\pi\ \forall i : t_i \in [t_l, t_h], \pi[i] \models \phi \\
(\mathcal{M}, s_0) &\models AF_{[t_l,t_h]}(\phi) && \Leftrightarrow \forall\pi\ \exists i : t_i \in [t_l, t_h], \pi[i] \models \phi \\
(\mathcal{M}, s_0) &\models AG_{[t_l,t_h]}(\phi) && \Leftrightarrow \forall\pi\ \forall i : t_i \in [t_l, t_h], \pi[i] \models \phi \\
(\mathcal{M}, s_0) &\models E(\psi U_{[t_l,t_h]}\phi) && \Leftrightarrow \exists\pi\ \exists i : t_i \in [t_l, t_h], \pi[i] \models \phi \wedge \\
& && \quad \forall j < i : \pi[j] \models \psi \\
(\mathcal{M}, s_0) &\models A(\psi U_{[t_l,t_h]}\phi) && \Leftrightarrow \forall\pi\ \exists i : t_i \in [t_l, t_h], \pi[i] \models \phi \wedge \\
& && \quad \forall j < i, \pi[j] \models \psi
\end{aligned}$$

A. Past Time CTL-AT

For many property circumstances it is useful to apply past time CTL-AT expressions. Past temporal operators can be described as a mirror image of future operators. Hence, we introduce past time operators (P,H, and S) which are mirror images of future operators ($F \leftrightarrow P$, $G \leftrightarrow H$, $U \leftrightarrow S$). In this way, e.g. $EP(\phi)$ means that ϕ was true at some past instant.

III. Symbolic Representation

The key breakthrough in the digital verification domain came by using symbolic representation for states to avoid the state explosion problem. Rather than explicitly enumerating every single state symbolic model checkers use BDDs to specify sets of states in the form of Boolean functions. To store timing behavior we used *multi terminal binary decision diagrams* (MTBDDs) as an extension of BDDs. MTBDDs represent pseudo-Boolean functions by mapping a vector of Boolean values into a real number. With BDDs as well as with MTBDDs we can get a formal description of transition relations and state sets using characteristic functions [6].

A. Characteristic Transition Function

In digital model checking theory transition relations and states are often denoted by characteristic functions. Characteristic functions are functional representations that indicate if a variable from a set A is contained in a subset B. To denote transition relations with delay time according to TDTS we define the following *characteristic transition function*.

Definition 3 (Characteristic Transition Function) *Let A and B be sets and $s \in A$, $s^+ \in B$. $\tau \in \mathbb{R}$ be a real number characterizing the transition delay time between the states s and s^+. The characteristic transition function (CTF) $\chi_\delta : \delta \to \mathbb{R}$ related to TDTS is then defined as:*

$$\chi_\delta(s, s^+, g) = \begin{cases} \tau & : \quad (s, s^+, g) \in \delta \\ \infty & : \quad (s, s^+, g) \notin \delta \end{cases}$$

Considering the binary coded vector $\vec{z} \in \mathbb{B}^n$ representing each present state, the binary coded vector $\vec{z}^+ \in \mathbb{B}^n$ representing each successor state, and the binary input coded vector $\vec{g} \in \mathbb{B}^m$. The overall CTF can now be written as a concatenation of all pseudo Boolean transition relations between each states $s_i, s_j \in S$.

$$\chi_\delta = \bigvee_{s_i, s_j} \big(\tau_{i,j} \cdot (\vec{z} \equiv \vec{z}(s_i))(\vec{z}^+ \equiv \vec{z}(s_j))(\vec{g} \equiv \vec{g}(s_i, s_j))\big) \tag{1}$$

The symbol $\tau_{i,j}$ denotes the according transition delay time between the present state s_i and the successor state s_j under the related input $\vec{g}(s_i, s_j)$. For example, Figure 1(a) shows an illustration of a TDTS structure. The transitions, displayed as edges between the discrete states, are tagged with the delay time and the valid input value $g \in G$. The related MTBDD representation of the transition relation is given in Figure 1(b). Each non-terminal value represents a state coding bit $\vec{z} = [z_1, z_2]$, $\vec{z}^+ = [z_1^+, z_2^+]$ or an input g. The terminal value denotes the delay

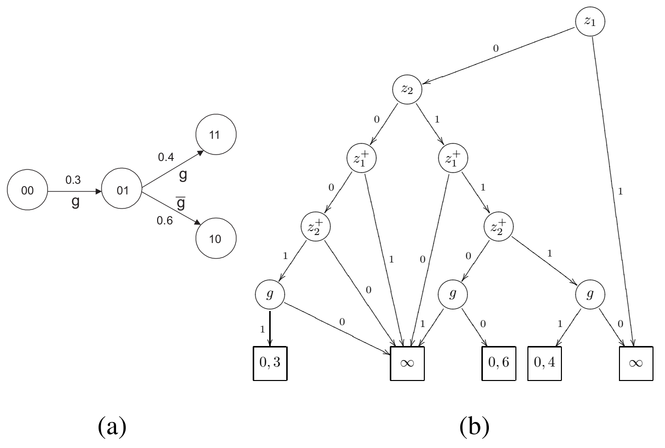

Fig. 1. *(a) Time delayed transition structure, (b) transition relation MTBDD.*

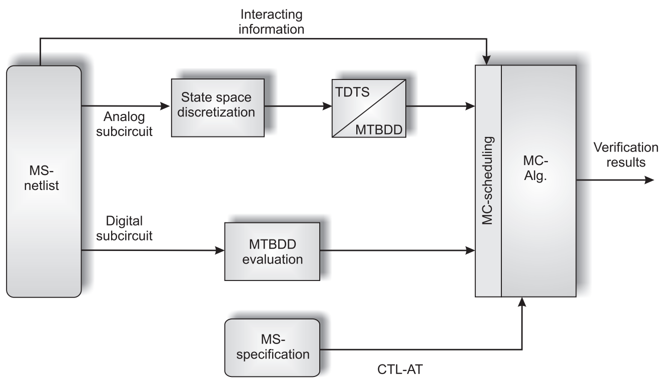

Fig. 2. Mixed-signal verification framework.

time using the transition represented by the path to its terminal node. Related to definition 3, the infinity symbol ∞ as a terminal value characterizes that there is no valid transition between two states.

Similar to the symbolic representations in the digital manner, sets of states can also be given by MTBDDs. The terminal nodes in these MTBDDs are labeled only with 1 for a valid state or 0 for no valid state. For this reason MTBDDs which represent states can be considered as BDDs. In the remain of this paper we will denote the characteristic function for a set of states with $\phi = \chi(\vec{z})$.

IV. Analog State Space

Our approach is basically superimposed on the analog state space discretization that was presented in [7]. Based on a modified nodal analysis a system of nonlinear differential-algebraic equations (DAE) is setup up for the circuit. The energy storing quantity (like voltage at capacitances and currents through inductors) and inputs span an extended continuously state space. This infinity state space is then bounded to a finite space, limited by e.g. supply voltages, and automatically divided into a finite number of homogeneous n-dimensional hyperboxes that cover the limited state space. Each hyperbox is treated as a discrete state and the dynamics can be interpreted as transitions which labels the delay time. The transition delay time between the discrete states is obtained by numerically solving the nonlinear DAE-systems of the circuit for a finite number of randomly chosen points within the predecessor hyperbox. The obtained discrete model describes the dynamics of the analog systems by a TDTS.

V. Symbolic Mixed-Signal Model Checking

Figure 2 gives the framework of the verification flow we used. The mixed-signal netlist contains both analog and digital circuit parts, which is split into a digital and an analog subcircuit. The interfaces between these two parts are modeled by simple 1-bit quantizer on the analog side (AD) and an 1-bit converter on the digital side (DA). The converter transforms the output value to a voltage source which supplies the analog circuit. To get a discrete time delayed transition structure for the analog circuit we applied the above described discretization method. The discretized state space is transformed into a corresponding transition relation MTBDD which contains all transition delay times.

From the sub-netlist of the digital part the corresponding transition MTBDD is directly determined. Both transition MTBDDs are inputs of the model checking algorithm. Additionally, the mixed-signal specification of the system, which should be proven, is needed. For evaluating a CTL-AT formula simultaneously on the MTBDDs of both systems, we implemented a scheduling algorithm which is strongly dependent on the interface signals within the mixed-signal system.

A. Mixed-Signal CTL-AT Analysis

Considering mixed-signal systems, the strong concurrencies between the analog and digital subsystems have to be applied carefully. Hence, such systems consist of two kinds of input signals. One input ($\vec{e}$) is the immediate consequence to the output of the intercommunicating system and can be therefore considered as a state variable. The other input ($\vec{w}$) comes from extern and is independent of the system states. Both input signals are included in the transition MTBDD. The basic quantification functions for the *EX* and *AX* operations has to be modified as follows.

$$EX(\phi_{st}) = \exists \vec{z}^+ \, \exists \vec{e} \, \exists \vec{w} : \chi_\delta(\vec{z}, \vec{z}^+, \vec{e}, \vec{w}) \wedge \chi_{st}(\vec{z}^+) \tag{2}$$

$$AX(\phi_{st}) = \exists \vec{z}^+ \, \exists \vec{e} \, \forall \vec{w} : \chi_\delta(\vec{z}, \vec{z}^+, \vec{e}, \vec{w}) \wedge \chi_{st}(\vec{z}^+) \tag{3}$$

Furthermore, the time constraints and the different delay time in both circuit domains restrict the verification to a crucial procedure. Hence, we developed a scheduling algorithm that sequentially decides which subsystem has to be evaluated by the CTL-AT formula and the time restriction. The time interval included in the CTL-AT formula is automatic divided into subintervals that guarantee that evaluation in the actual subsystem without having any affect to the other system part causing a state change. Algorithm 1 shows the basic structure of the recursive scheduling procedure for analog and digital CTL-AT evaluation. The initial arguments ϕ_A and ϕ_D are the analog and digital starting state MTBDDs declared in the

Algorithm 1 CTLschedul($\phi_A, \phi_D, \varphi_A, \varphi_D, \chi_{\delta_A}, \chi_{\delta_D}, [t_l, t_h]$)

```
$A_{Min}$ = getMinTransition($\phi_A, \chi_{\delta_A}$)
$D_{Min}$ = getMinTransition($\phi_D, \chi_{\delta_D}$)
**if** $A_{Min} \leq t_h \vee D_{Min} \leq t_h$ **then**
    **if** $A_{Min} < D_{Min}$ **then**
        $(\Theta_A, \phi_{AI})$ = CheckCTL($\phi_A, \varphi_A, \chi_{\delta_A}, [t_{tl}, D_{Min}]$)
        $\varphi_D$ = CheckImpact($\phi_{AI}, \phi_D$)
        $(\Theta_D, \phi_{DI})$ = CheckCTL($\phi_D, \varphi_D, \chi_{\delta_D}, [t_{tl}, D_{Min}]$)
        $\varphi_A$ = CheckImpact($\phi_{DI}, \phi_{AI}$)
        $t_l = t_l$ - $D_{Min}$
        $t_h = t_h$ - $D_{Min}$
    **else**
        $(\Theta_D, \phi_{DI})$ = CheckCTL($\phi_D, \varphi_D, \chi_{\delta_D}, [t_l, A_{Min}]$)
        $\varphi_A$ = CheckImpact($\phi_{DI}, \phi_A$)
        $(\Theta_A, \phi_{AI})$ = CheckCTL($\phi_A, \varphi_A, \chi_{\delta_A}, [t_l, A_{Min}]$)
        $\varphi_D$ = CheckImpact($\phi_{AI}, \phi_{DI}$)
        $t_l = t_l$ - $A_{Min}$
        $t_h = t_h$ - $A_{Min}$
    **end if**

    $(\tilde{\Theta}_A, \tilde{\Theta}_D)$ = CTLschedul($\phi_{AI}, \phi_{DI}, \varphi_A, \varphi_D, \chi_{\delta_A},$
                                        $\chi_{\delta_D}, [t_l, t_h]$)
    $\Theta_A = \Theta_A \cup \tilde{\Theta}_A$
    $\Theta_D = \Theta_D \cup \tilde{\Theta}_D$
    **return** $\Theta_A, \Theta_D$
**end if**
```

CTL-AT formula. χ_{δ_A} and χ_{δ_D} are the corresponding transition relation MTBDDs. The interval $[t_l, t_h]$ declares the desired time delay boundaries. First, corresponding to the starting states of both systems the least outgoing transition delay time for each subsystem has to be determined (*getMinTransition*). Whichever transition delay time is the lesser, the origin upper bound within the interval is modified by this value and the corresponding CTL-AT analysis for the appropriate subsystem starts (*CheckCTL*). After that, the last obtained state (ϕ with index I) in fixpoint iteration is responsible for the interaction to the other subsystem. For this reason an impact checking (*CheckImpact*) procedure has to be done. The function *CheckImpact* determines the corresponding set of input values $\varphi_A = \{\vec{e}_{A1}, \vec{e}_{A1}, \dots\}, \varphi_D = \{\vec{e}_{D1}, \vec{e}_{D1}, \dots\}$ to the other subsystem and modifies the relating MTBDD entry within the start state MTBDD.

The impact checker modifies the MTBDD input entry within the start state corresponding to the output value of the other subsystem. According to the new start state the CTL-AT analysis and an impact checking for the other subsystem can start. The function *CTLschedul* is then be recursively executed with the time interval decreased by the elapsed time. The function *CheckCTL* is calling the essence CTL-AT analysis procedure. As an example we will represent the algorithm *CheckDigitalTimeEF* and *CheckAnalogTimeEF* in Algorithms 2 and 3 which are evaluating the time restricted CTL-AT operation for a given initial state and time interval. The reason why we distinguish between the analog and digital EF operation, is that we can have different delay times within the analog system in contrary to the digital part. The Algorithm *CheckDigitalTimeEF* applies fixpoint iteration according to the time restriction given by the time interval $[t_l, t_h]$. The value τ_{clk} denotes the digital clock time. The function *CheckEX* is computed by using

Algorithm 2 CheckDigitalTimeEF($\phi_D, \chi_{\delta_D}, [t_l, t_h]$)

```
$\Omega_i = false, \Omega_{i+1} = \phi_D, \Upsilon = \phi_D, \Theta = false$
**if** $t_l < \tau_{clk}$ **then**
    $\Theta = \phi_D$
**end if**
**while** $t_h \geq \tau_{clk}$ **do**
    **if** $\Omega_i \neq \Omega_{i+1}$ **then**
        $t_h = t_h - \tau_{clk}, t_l = t_l - \tau_{clk}$
        $\Omega_i = \Omega_{i+1}$
        $\Upsilon$ = CheckEX($\Upsilon, \chi_{\delta_D}$)
        **if** $t_l < \tau_{clk}$ **then**
            $\Theta = \Theta \cup \Upsilon$
        **end if**
        $\Omega_{i+1} = \Upsilon \cup \Omega_i$
    **else**
        $\Theta = \Theta \cup \Omega_{i+1}$
        $t_h = 0$
    **end if**
**end while**
**return** $\Theta, \Upsilon$
```

the quantification procedure in equation (2) after mapping the MTBDD to a BDD. This can be done, because all terminal nodes excluding the infinity nodes are equal. The algorithm returns two MTBDDs Θ and Υ, while Θ represents the result of the EF operation and Υ gives the last state in iteration needed for the impact checking in Algorithm 1 (ϕ_{DI}).

Algorithm 3 applies a recursive call to evaluate the EF operation in the analog domain. We used a recursive algorithm, because all transitions are considered under time restriction explicitly. For every transition $\lambda(\phi_A, \chi_{\delta_A}) \in \delta$ outgoing from the actual state ϕ_A the delay time τ_{tr} and the iteration step is determined (*CheckEX*) by extracting the related transition relation and transforming it to a transition BDD before the recursive call is executed. The result Θ is a collection of all interim results during the time interval $[t_l, t_h]$. Whereas Υ gives the last state needed for the impact checking in Algorithm 2 (ϕ_{AI}). Figure 3 demonstrates the scheduling algorithm by using the CTL-operation $EF[t_l, t_h](D_1 \wedge A_4)$. To ease the understanding we consider the product automata of an interacting set of analog and digital states. The big dashed circles depict digital states ($D_1, \cdots, D_4$) and the smaller circles depict analog states

Algorithm 3 CheckAnalogTimeEF($\phi_A, \chi_{\delta_A}, [t_l, t_h]$)

```
$\Theta = false, \Upsilon = false$
**for** all transitions $\lambda(\phi_A, \chi_{\delta_A}) \in \delta$ **do**
    $\tau_{tr}$ = getTransitionTime($\lambda(\phi_A, \chi_{\delta_A})$)
    **if** $t_l < \tau_{tr}$ **then**
        $\Theta = \phi_A$
    **end if**
    **if** $\tau_{tr} \leq t_h$ **then**
        $\phi_A$ = CheckEX($\phi_A, \chi_{\delta_A}, \tau_{tr}$)
        $(\tilde{\Theta}, \tilde{\Upsilon})$ = CheckAnalogTimeEF($\phi_A, \chi_{\delta_A}, [t_l$-$\tau_{tr}, t_h$-$\tau_{tr}]$)
        $\Theta = \Theta \cup \tilde{\Theta}, \Upsilon = \Upsilon \cup \tilde{\Upsilon}$
    **else**
        $\Upsilon = \phi_A$
    **end if**
**end for**
**return** $\Theta, \Upsilon$
```

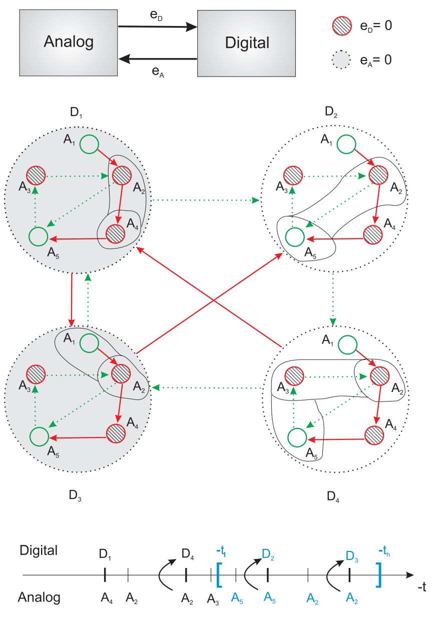

Fig. 3. $EF[t_l,t_h](D_1 \wedge A_4)$ verification flow.

$(A_1,\cdots,A_5)$. Without loss of generality, we assume that the digital clock, i.e. the digital transition time, is bigger than each analog transition delay. Each set of states can be divided into states that produce a Boolean value 0 and a Boolean value 1 at the interacting output ports. The digital states with the grey background depict states that produce an $e_A = 0$ and the states with white background an $e_A = 1$. The red hatched analog states generate an $e_D = 0$ and the green ones an $e_D = 1$ at the output. The green dashed and the red edges between the digital states indicate which transition has to be applied dependently to the output of the analog subsystem and vice versa. All transitions between the digital states are labeled with the digital delay time t^D. Further, we assume that the delay transition time t_i^A between the analog states can be different. The interval $[t_l, t_h]$ is partioned into subintervals constrained by the digital clock cycle t^D.

First the CTL-operation EF is applied using fixpoint iteration until the time reaches the upper bound t^D. The obtained analog path is A_4, A_2 related to the digital state D_1. The function *CheckImpact* modifies the input of the digital subsystem (e_A) to a 0. Next, the digital part takes one step using the digital red transition to state D_4. The boundaries in the time interval are then decreased by t^D ($[t_l - t^D, t_h - t^D]$). The new digital state generates an output of 1, therefore the actual analog state changes after a delay to state A_3. With the new time boundaries the algorithm repeats while the origin upper bound of the time interval is not reached. The result is a set of analog and digital states collected during the iterations fulfilling the time restrictions. For the results of the $EF[t_l,t_h](D_1 \wedge A_4)$ formula we obtained the set of states $\{(D_4, A_3), (D_4, A_5), (D_2, A_5), (D_2, A_2), (D_3, A_2)\}$.

VI. Experimental Results

To demonstrate our symbolic approach we consider the phase locked loop (PLL) in Figure 4. The PLL consists of an analog voltage controlled oscillator (VCO), a charge pump and a digital phase detector. The voltage at capacity C_1 drives the input of the digital part using an 1-bit quantizer (A/D-Interconnection). The output v_D of the digital block drives the input of the analog part by converting the Boolean value into a high voltage $U(v_D = 1) = 2.25V$ or a low voltage $U(v_D = 0) = 0.25V$ value. Within the digital part we added a decimation block to divide the digital clock into the desired reference clock T_{ref}. We used a clock frequency of $f_{clk} = \frac{1}{T_{clk}} = 250Hz$ and a decimation factor of 128 $(T_{ref} = 128 \cdot T_{clk})$. This means that we used a 7-bit counter, where the most significant bit (MSB) belongs to the clock level. The phase detecting is done by a simple XOR-gate. The synchronous output of the phase detector set the output latch W_0 which defines the output of the digital subcircuit. As a direct consequence the state MTBDD exhibits 9 variables, 7 for the counter, 1 for the output latch and 1 for defining the input.

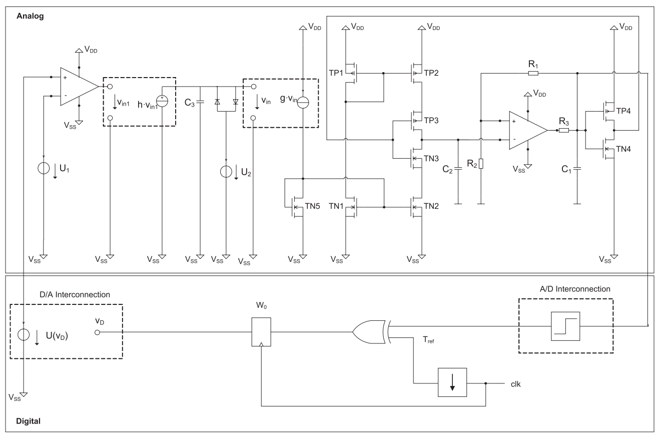

Fig. 4. Phase locked loop (PLL) consists of a voltage controlled oscillator (VCO) and a digital phase detector.

For the analog part we got a four dimensional state space spanned by the three capacitance voltages $(U_{C_1}, U_{C_2}, U_{C_3})$ and the input voltage $(U(v_D))$ within the D/A-Interconnection. We got 2.048 analog states which are symbolic encoded by 11 MTBDD variables. In Figure 5 the sub-state space spanned

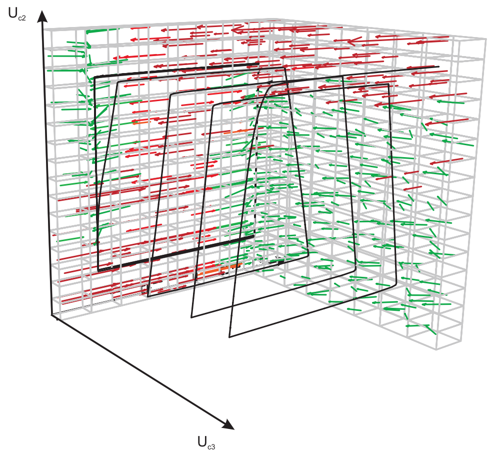

Fig. 5. VCO sub-state space.

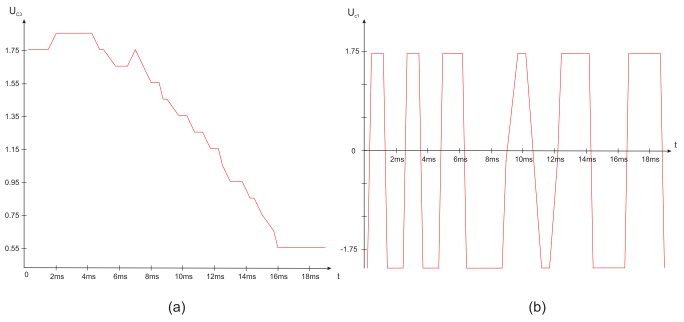

Fig. 6. PLL reachability analysis.

by the three capacitance voltages U_{C_1}, U_{C_2}, and U_{C_3} for input voltage $U(v_D) = 0.25V$, corresponded to the digital output zero, is depicted. The figure indicates the discretized sub-state space and a simulation trajectory as a black spiral line showing the dynamic behavior of the PLL.

A. Reachability Analysis

First we did reachability analysis by using the $EP[0.0ms, 20ms](\phi_D \wedge \phi_A)$ CTL operation starting with a randomized single state ϕ_D and ϕ_A for the digital and analog subsystems to get a simulation comparable progress showing phase locking behavior. Figure 6 (a) illustrates the voltage U_{C3} at the capacity C_3 extracted from the MTBDDs we got by stepwise traversal using the EP operation. The voltage U_{C3} at C_3 represents the smoothed input voltage dependent to the digital phase detector output. After 16 ms the voltage achieved 0.55 V that has the effect that the output oscillation frequency at capacity C_1 remains at 250 Hz (see Figure 6 (b)) which is identical to the digital reference frequency f_{clk}. The set of states the EP operation remained after 16 ms will be below denoted as $\phi_{lock} = U_{C3} > 0.5 \wedge U_{C3} < 0.6$. We used the following property to verify reachability behavior.

$$\phi_{lock} \models EP[0.0ms, 20ms](\phi_D \wedge \phi_A) \tag{4}$$

This property means, that starting from the given analog and digital states ϕ_A and ϕ_D the locking states ϕ_{lock} will be achieved within 20 ms by at least one path.

B. Locking Analysis

Furthermore we verify locking behavior of the PLL. To get all states that lead to the locking states ϕ_{lock} we used the AF formulation. We defined the locking property

$$true \models AF[0.0ms, 30ms](\phi_{lock}) \tag{5}$$

to verify that all states lead to the locking states ϕ_{lock} within the next 30 ms. This property is the main behavior of the PLL for phase synchronization of the input signal. For the above assumptions we got what we expected, that all starting states ϕ_D and ϕ_A lead to the locking states ϕ_{lock} within 30 ms. For this verification example our general purpose single CPU PC consumed 3.15 minutes for the discretization of the analog state space and 50 seconds for the verification procedure.

C. Discussion

Upto now every verification method applying mixed-signal or hybrid systems suffers from an explicit representation of digital states. Furthermore, they can only analyze linear systems or can only do reachability analysis. In this contribution we firstly introduced a verification method for mixed-signal systems with a non-linear behavior. Our method represents both analog and digital states as well as timed transitions implicitly by MTBDDs. This approach is able to overcome different transition times in the analog circuit part in combination with the digital clock. Hence, interactions between the analog and digital circuit parts can now be formal verified. The proposed example gives an advanced mixed-signal circuit with non-linear behavior and a strong dependency between the analog and digital parts. The resulted state variables for both, the analog and digital part, did not achieved practical limits when considering todays BDD based verification tools. But it shows that different time restrictions and interactions between analog and digital parts can now be overcome for formal verification issues.

VII. Conclusion

This paper introduces a novel approach for symbolic mixed-signal circuit model checking. To treat the different time properties that characterize the analog and digital subcircuits we used MTBDDs for efficient storing time behavior. We extend common digital quantification procedures to handle timing dependency between the circuits. For specifying mixed-signal properties that will be checked we used CTL-AT as the specification language. To demonstrate our approach we verified the reachability and the locking behavior of a PLL.

References

[1] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, “HYTECH: A Model Checker for Hybrid Systems,” *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 110–122, 1997.

[2] E. Asarin, T. Dang, and O. Maler, “d/dt: A Tool For Reachability Analysis Of Continuous And Hybrid Systems,” in *5th IFAC Symposium Nonlinear Control Systems (NOLCOS'01)*, July 2001.

[3] G. Frehse, “PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech,” *Lecture Notes in Computer Science*, vol. 3414, pp. 258–273, February 2005.

[4] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, Cambridge, Massachusetts, London, 1999.

[5] D. Grabowski, D. Platte, L. Hedrich, and E. Barke, “Time Constrained Verification of Analog Circuits using Model-Checking Algorithms,” in *Proceedings of the First Workshop on Formal Verification of Analog Circuits (FAC'05)*, ser. 3, vol. 153, April 2005, pp. 37–52.

[6] E. M. Clarke, M. Fujita, P. C. McGeer, K. McMillan, J. C.-Y. Yang, and X. Zhao, “Multi Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation,” in *International Workshop on Logic Synthesis (IWLS'93)*, May 1993.

[7] W. Hartong, L. Hedrich, and E. Barke, “Model Checking Algorithms for Analog Verification,” in *Proceedings of the 39th Conference on Design Automation (DAC'02)*, June 2002, pp. 542–547.