

# Analysis of a Boost Converter Circuit Using Linear Hybrid Automata

Ulrich Kühne

LSV ENS de Cachan, 94235 Cachan Cedex, France,  
kuehne@lsv.ens-cachan.fr

## 1 Introduction

Boost converter circuits are an important component in electrical and electronic systems. Serving as energy converters, they increase the voltage of a source. In mobile devices, high voltages can be achieved in this way, with a relatively small number of accumulator cells. Without this or a similar technique, it would be difficult to supply electric cars with the necessary voltages of about 500 V.

During the design and optimization of such analog systems, formal methods are rarely applied, in contrast to the computer aided design and verification of digital systems. The choice of the system parameters – capacitors, inductors and resistors – is mainly based on the expertise of the designers. This process is supported by simulation tools.

Results on the modeling and formal verification of analog oscillator circuits have been reported [DDM04,FKR06]. A promising approach is the modeling of analog circuits using hybrid automata.

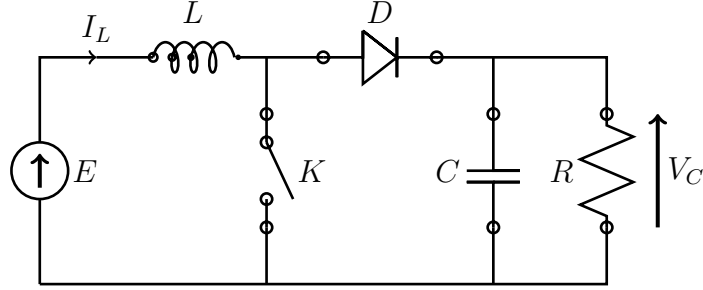
## 2 Preliminaries

### 2.1 Hybrid Automata

The definitions in this section have been adapted from [Fre08]. Given a set of variables  $Var$ , a valuation is a function  $v : Var \rightarrow \mathbb{R}$ . Let  $V(Var)$  be the set of valuations of  $Var$ . An activity is a function  $f : \mathbb{R}^{\geq 0} \rightarrow V(Var)$ , the set of activities over variables  $Var$  is denoted as  $act(Var)$ . A set  $S$  of activities is time-invariant if  $\forall f \in S, d \in \mathbb{R}^{\geq 0} : f(t + d) \in S$ .

**Definition 1.** A hybrid automaton  $H = (Loc, Var, Lab, \rightarrow, Act, Inv, Init)$  consists of the following

- a finite set of locations  $Loc$
- a finite set of variables  $Var$
- a finite set of synchronization labels  $Lab$
- discrete transitions  $l \xrightarrow{a, \mu} l'$ , where  $l, l' \in Loc$ ,  $a \in Lab$  and  $\mu \in 2^{V(Var) \times V(Var)}$
- a mapping  $Act : Loc \rightarrow 2^{act(Var)}$  from locations to sets of time-invariant activities, describing the change of the variables over time



**Fig. 1.** Schematics of boost converter circuit

- a mapping  $Inv : Loc \rightarrow 2^{V(Var)}$  from locations to sets of valuations, constraining the valuations of variables for each location
- a set of initial states  $Init \subseteq Loc \times V(Var)$ .

In a *linear hybrid automaton* (LHA), the invariants are given by linear constraints on the variables, and the activities as linear constraints over the time derivatives of the variables.

## 2.2 Boost Converter Circuit

The circuit that has been investigated is shown in Figure 1. The behavior of the system depends on the state of a switch  $K$ . The switch is controlled by a clock signal with period  $T_d$ . During the on-phase, state  $Q_1$  is active, while  $Q_2$  is active during the off-phase. The duration of the clock phases is controlled by a parameter  $\alpha$ , such that the on-phase lasts for  $\alpha \cdot T_d$ , and the off-phase lasts for  $(1 - \alpha) \cdot T_d$ .

The state of the system is given by the the current  $I_L$  the voltage  $V_C$ , which will be represented by the variables  $x$  and  $y$ , respectively. Each of the two clock phases is associated with a system of differential equations

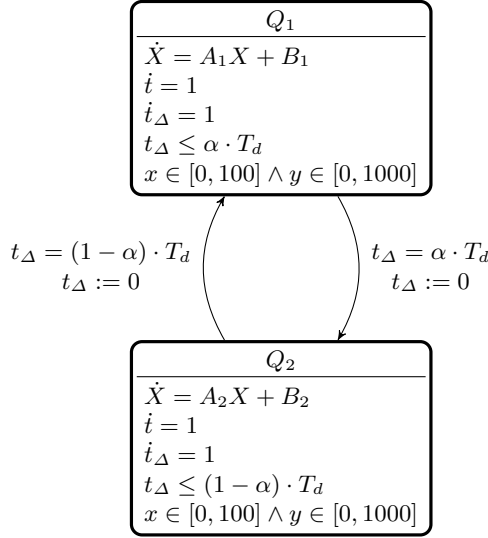
$$\dot{X} = AX + B, \quad \text{where } X = \begin{pmatrix} x \\ y \end{pmatrix}. \quad (1)$$

The concrete values for the matrices in the two clock phases is given below. The matrices  $A_1$  and  $B_1$  describe the on-phase, while  $A_2$  and  $B_2$  model the off-phase.

$$A_1 = \begin{pmatrix} -\frac{r_K}{L} & 0 \\ 0 & -\frac{1}{RC} \end{pmatrix}, \quad B_1 = \begin{pmatrix} \frac{E}{L} \\ 0 \end{pmatrix}, \quad (2)$$

$$A_2 = \begin{pmatrix} -\frac{r_D}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{pmatrix}, \quad B_2 = \begin{pmatrix} \frac{E}{L} \\ 0 \end{pmatrix}, \quad (3)$$

where in all experiments the following values are used:  $E = 100 \text{ V}$ ,  $C = 10 \mu\text{F}$ ,  $L = 500 \mu\text{H}$ ,  $R = 75 \Omega$ ,  $R_K = 10^6 \Omega$ ,  $r_K = r_D = 0.1 \Omega$ . The frequency is set to



**Fig. 2.** Hybrid automaton  $H_1$  of boost converter circuit

$Freq = 10 kHz$ , resulting in a period of  $T_d = 0.1 ms$ , and the clock parameter is set to  $\alpha = \frac{13}{16} = 0.8125$ .

### 3 System Model

#### 3.1 Hybrid Automaton Model

The system is modeled by means of hybrid automata. The first model is shown in Figure 2. According to the clock phases of the system, there are two locations  $Q_1$  and  $Q_2$ . The system has four (continuous) state variables,  $x$  and  $y$  for current and voltage, a global clock  $t$ , and an auxiliary clock  $t_\Delta$ , which is used to switch between the locations according to the clock phase.

The invariants of both locations limit the values of  $x$  and  $y$  to the intervals  $x \in [0, 100]$  and  $y \in [0, 1000]$ . Additionally,  $t_\Delta$  is limited to the duration of the corresponding clock phase ( $\alpha \cdot T_d$  and  $(1 - \alpha) \cdot T_d$ , respectively). Whenever the maximum value of  $t_\Delta$  is reached, a transition is fired to the other location, and  $t_\Delta$  is reset to zero.

The derivatives of the state variables  $x$  and  $y$  are constrained as given by equations (2) and (3), while the two clocks  $t$  and  $t_\Delta$  evolve uniformly. The initial states of the automaton are given as  $Init = (Q_1, x = y = t = t_\Delta = 0)$ .

#### 3.2 Implementation with PHAVer

The tool PHAVer [Fre08] is used to implement the above model and to compute the set of reachable states. PHAVer models *linear hybrid automata* based on a

polyhedral representation of states. Due to the LHA semantics, the activities must be described by linear formulas over the time derivatives. Thus, the automaton in Figure 2 cannot be verified directly, as the derivatives depend on the variables  $x$  and  $y$ . But, PHAVer supports such *affine dynamics* by automatic partitioning of locations. According to the differential equations, a linear approximation of the derivatives is computed for each location.

The partitioning can be done on-the-fly, in order to avoid splitting unreachable locations or locations where the linear approximation is sufficiently precise. The minimum and maximum size of the partitions is controlled via input parameters to PHAVer. In the following, we refer to the maximum number of partitions for each dimension as parameter  $P$ . Given a location  $Q$  with variable  $\varphi$  and the invariant  $\varphi \in [l, u]$ , the location can be split up during the analysis until the invariant of the resulting locations has reached the minimum size  $\frac{|u-l|}{P}$ .

In order to reduce the complexity of the computations, PHAVer allows to limit the number of constraints per formula as well as the number of bits used for representing the coefficients. Whenever a user specified limit is exceeded, an overapproximation is performed on the polyhedra to reduce the memory requirements. A further method to speed up reachability analysis is to join the reachable states after each step by their convex hull. In total, these features offer rich possibilities to trade precision with speed, that can be adjusted with regard to each application.

The implementation of  $H_1$  in the PHAVer input language is shown in Figure 3. The invariants of the locations are given after the `while` keyword, while the derivatives are defined in the block after the `wait` keyword. The bounds on  $x$  and  $y$  are given by constants  $X_l, X_u, Y_l, Y_u$ . The constant  $T_{on}$  and  $T_{off}$  are assigned the values  $\alpha \cdot T_d$  and  $(1 - \alpha) \cdot T_d$ , respectively. The transition from  $Q_1$  to  $Q_2$  is fired at  $t = T_{on}$ , and the automaton returns to location  $Q_1$  after  $t_\Delta = T_{off}$ . The blocks after the `do` keyword contain the assignments to the variables when the transition is fired. For both transitions, all variables keep their values, except  $t_\Delta$  which is reset to zero.

## 4 System Exploration

In the traditional design process, the system is simulated using tools like Matlab. There, a numeric solution of the differential equations is used to obtain a single simulation trace.

### 4.1 Forward Reachability Analysis

In order to explore the behavior of the boost converter circuit, the reachable states of the automaton are computed with PHAVer. However, the computation for the automaton as given in Figure 2 will not terminate. Since there is no bound on  $t$ , there is no fix-point. A fix-point can only be reached when removing the global clock  $t$ , or adding a bound on  $t$  to the invariants of the locations  $Q_1$  and  $Q_2$ .

```

1  automaton H1
2  contr_var : x, y, t, tD;
3  synclabs  : S, P;
4
5  loc Q1:
6      while x >= X_l & x <= X_u & y >= Y_l & y <= Y_u &
7          tD <= T_on wait
8          {x' == A1_11*x + A1_12*y + B1.1 &
9           y' == A1_21*x + A1_22*y + B1.2 &
10          t' == 1 & tD' == 1}
11     when tD == T_on sync S do
12         {x' == x & y' == y & t' == t & tD' == 0}
13         goto Q2;
14
15  loc Q2:
16     while x >= X_l & x <= X_u & y >= Y_l & y <= Y_u &
17         tD <= T_off wait
18         {x' == A2_11*x + A2_12*y + B2.1 &
19          y' == A2_21*x + A2_22*y + B2.2 &
20          t' == 1 & tD' == 1}
21     when tD == T_on sync S do
22         {x' == x & y' == y & t' == t & tD' == 0}
23         goto Q1;
24
25  initially : Q1 & x == 0 & y == 0 & t == 0 & tD == 0;
26  end

```

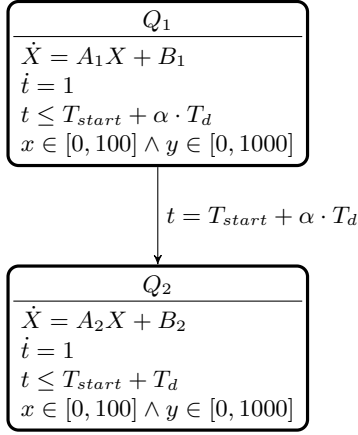
**Fig. 3.** PHAVer code for the circuit model  $H_1$

When removing the variable  $t$  from the model, a simplified automaton is obtained, for which the reachable states can possibly be computed in a finite number of steps. But, for a reasonable precision, the resources needed for the reachability analysis are very high. In fact, even for a very low resolution with  $P = 4$ , the computation does not terminate in less than two hours, unless the convex hull overapproximation is used. When activating this option (as well as limiting the number of constraints and the bitsize of the coefficients), the reachable states with respect to  $x$  and  $y$  can be computed.

But, due to the overapproximation, the computed states cover a large portion of the observed state space. This does not allow for any precise predictions of the system behavior. Furthermore, for  $P > 16$ , no results could be obtained within a timeout of two hours. In order to overcome these limitations, a different approach is explored.

## 4.2 Iterative Reachability Analysis

In a second approach, the reachable states of the system are computed step by step, one period at a time. The reachable states are saved to a file after each



**Fig. 4.** Bounded hybrid automaton  $H_2$  of boost converter circuit

iteration. Before the next iteration, the automaton is initialized with the saved states. In this way, unnecessary computations are avoided, as only the post-image of the last intermediate result needs to be computed. Furthermore, only one clock is needed, as the system is limited to one period only and does not need to return to state  $Q_1$ . The bounded model of the boost converter circuit is shown in Figure 4.

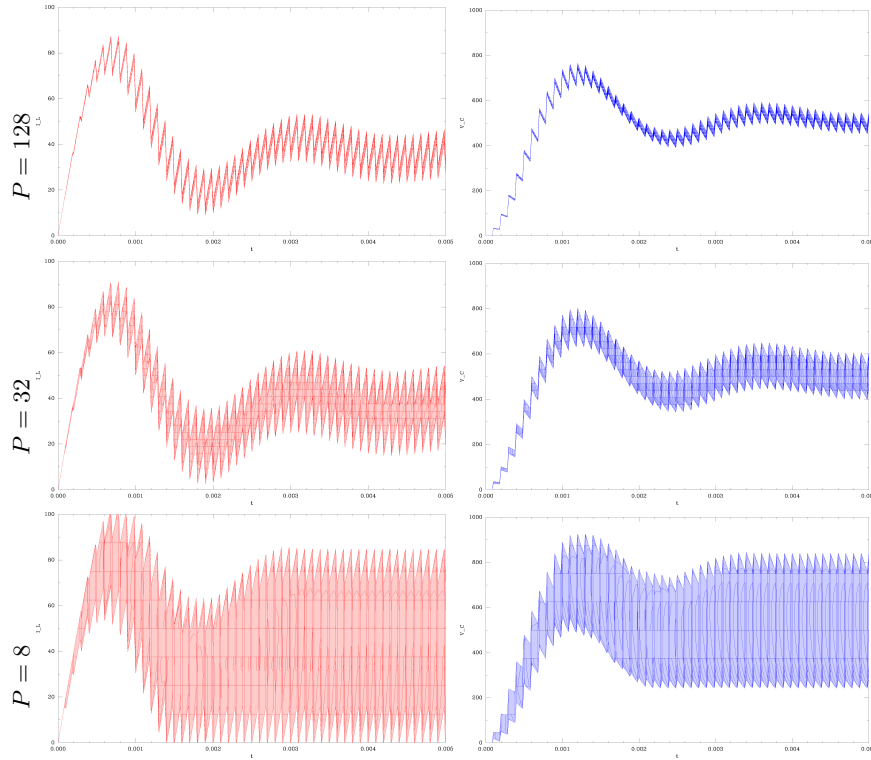
The advantage of this approach is that the behavior of the circuit can be observed with respect to the global time. For the boost converter circuit, this is interesting as it goes through a phase of high amplitude before stabilizing at a certain current and voltage. With the iterative approach, the behavior of this phase can be observed with a relatively high precision. After reaching the stable phase, the reached states can be used to perform further analyses, as reported in the following sections.

The iterative reachability analysis proceeds as follows:

1. Set  $T_{start} = 0$ .
2. Assign states  $I = (Q_1, x = y = 0 \wedge t = T_{start})$ .
3. Initialize automaton with states  $I$ .
4. Compute and save reachable states for one period.
5. Compute final states  $F$  at time  $t = T_{start} + T_d$ .
6. Assign  $I = F$ .
7. Increase  $T_{start}$  by  $T_d$  and go to 3.

Note that in order to restart the automaton in the correct location, the states  $F$  are restricted to  $Q_1$  before saving them for the next iteration.

With the above algorithm, the reachable states of the system have been computed for 50 iterations, corresponding to 5 ms. The experiment has been performed with different partitioning by adjusting the parameters for the minimum size of the partition. With the parameter  $P$ , the minimum size of the

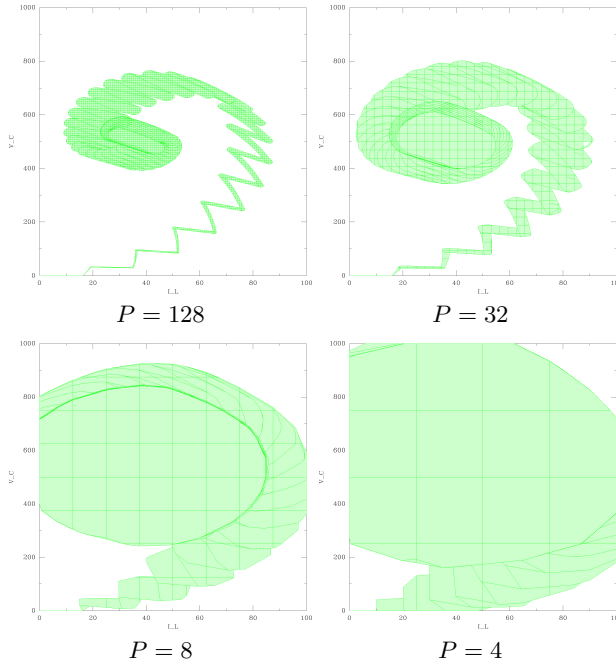


**Fig. 5.** Plots for  $I_L$  and  $V_C$  with different partitionings

location's invariants is restricted to  $\frac{|X_u - X_l|}{P}$  and  $\frac{|Y_u - Y_l|}{P}$ . The plots for  $I_L$  and  $V_C$  over  $t$  are shown in Figure 5. The projections on  $I_L$  and  $V_C$  are shown in Figure 6. Run-time and maximal memory requirements are reported in Table 1.

**Table 1.** Resources needed for 50 iterations of reachability analysis

$P$	time	memory
4	118.4 s	5.6 MB
8	350.7 s	13.8 MB
16	715.7 s	24.2 MB
32	1099.8 s	35.8 MB
64	1637.5 s	57.9 MB
128	2662.8 s	104.6 MB



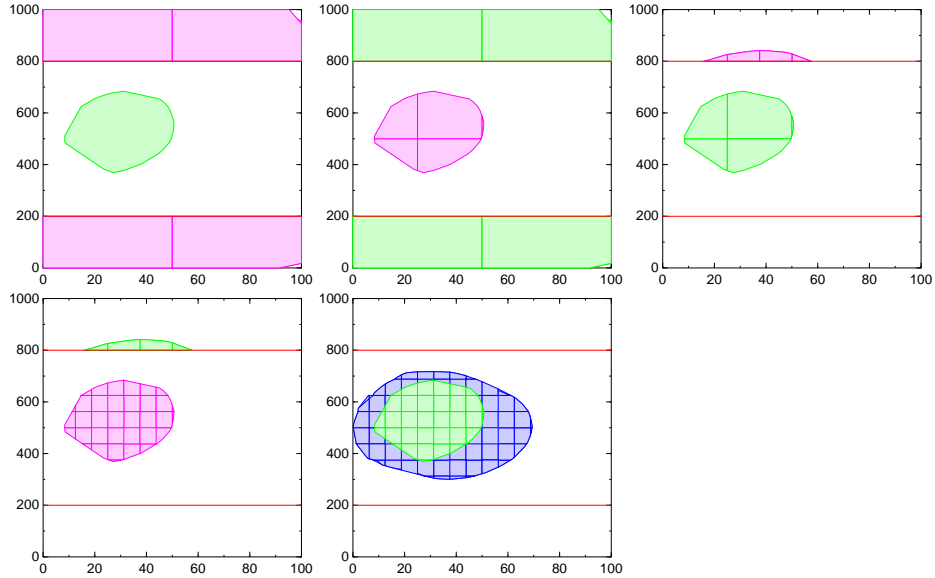
**Fig. 6.** Plots for  $I_L$  and  $V_C$  with different partitionings

## 5 Formal System Analysis

### 5.1 Proving Properties

Looking back at Figure 5, consider the plots for  $P = 8$ . Even if  $I_L$  and  $V_C$  cover a wide range of values, the curve seems to stabilize after about 35 cycles. Safety properties can be checked by analyzing the reachability of the system. This corresponds to proving that a set of bad states  $F$  are never reachable. As computing the reachable states is expensive, it is desirable to use an abstract model that allows to prove the property, while reducing the costs wrt. the full model. For the system at hand, this boils down to finding the coarsest partitioning for which the states  $F$  are unreachable. An automatic way of finding this abstraction is *forward-backward-refinement* (FBR), which is implemented in PHAVer.

In principal, during FBR, forward and backward reachability are used alternately, with increasing precision. During the procedure, PHAVer checks the reachability of  $F$ , starting with a very coarse abstraction and computing the reachable states  $R_1$ . If  $F$  is reachable from the initial states  $I$ , then the automaton is reversed. The reachability analysis is now performed from the found bad states towards the initial states. Note that not all bad states are used, but rather the intersection with the reachable states for the first step. More precisely, it is checked if from the reached bad states  $B_1 = R_1 \cap F$ , the initial states  $I$  are reachable, where the precision (the number of partitions per variable) is increased. If



**Fig. 7.** Refinement steps in reachability analysis

the initial states are unreachable, the property has been proven. If the reachable states  $R_2$  hit a part of  $I$ , then the procedure is repeated: the automaton is reversed, and it is checked if the bad states  $B_1$  are reachable from  $I \cup R_2$  with increased precision. The algorithm stops as soon as no bad states are reachable any more.

In an experiment, it was proven that starting from a set of states  $I$ , the voltage can never drop under 200 V and will never exceed 800 V. Note that for this purpose, the full model  $H_1$  (see Figure 2) is used rather than the bounded model. As initial states  $I$ , the states reached by the iterative analysis with  $P = 8$  after exactly 43 cycles were used. The property could be proven after four refinement steps. After the last refinement step, all reachable states fulfill the property.

The reachable bad states during FBR are shown in Figure 7. The initial states are plotted in green, while the reached bad states are shown in magenta. In the first step, almost all bad states are reachable. After reversing the automaton, all the original initial states are reachable. In the third step, only a fraction of the bad states are reachable. However, when reversing the automaton, again all initial states are reachable in the fourth step. In the last step, the reachable states in blue do not intersect the bad states any more. In this experiment, the FBR method did not provide any advantages over simple forward reachability. If one would have started with a partitioning given by  $P = 16$ , the property could have been proven directly, without having to perform steps one to four.

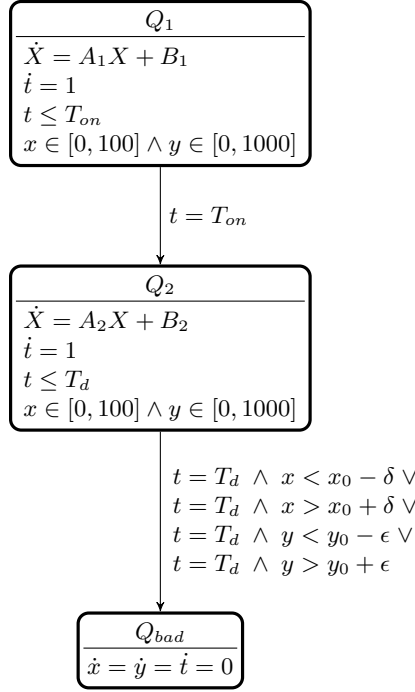


Fig. 8. Hybrid automaton  $H_3$  with monitor location

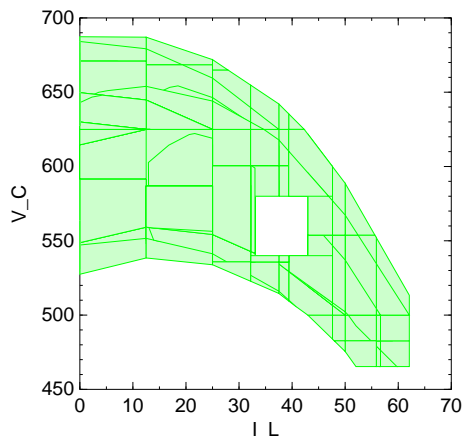
## 5.2 Parameter Synthesis

**Synthesizing  $\alpha$**  The parameter  $\alpha$  controls the level of  $I_L$  and  $V_C$ , on which the system stabilizes. In order to meet specific requirements for  $I_L$  and  $V_C$ , one has to find out the right  $\alpha$  by simulation (or an automatic regulation of  $\alpha$  has to be added to the system). With the LHA model of the circuit,  $\alpha$  can be synthesized.

For the given system,  $\alpha$  is not synthesized directly, but the duration of  $T_{on}$ , which triggers the transition to location  $Q_2$ . For this purpose,  $T_{on}$  is declared as a parameter for the bounded LHA model, and it is restricted to  $0 < T_{on} < T_d$ .

The first idea is to find the good  $T_{on}$  such that the system stays within a stable region. In a simple approach, this is expressed by the requirement that the system returns to a specified rectangle after one period. In order to distinguish the good and the bad traces in the system, a monitor location  $Q_{bad}$  is added to the automaton, as can be seen in Figure 8. At the end of the period the transition from  $Q_2$  to  $Q_{bad}$  can be fired if either of the state variables is outside the region  $x \in [x_l, x_u] \wedge y \in [y_l, y_u]$ , where  $x_l = x_0 - \delta$ ,  $x_u = x_0 + \delta$ ,  $y_l = y_0 - \epsilon$ ,  $y_u = y_0 + \epsilon$  and  $X_0 = (x_0, y_0)^T$  is the reference point for the analysis.

The *bad* values of parameter  $T_{on}$  can now be computed by means of a reachability analysis. The initial states are restricted to  $(Q_1, x \in [x_l, x_u] \wedge y \in [y_l, y_u])$ . Then the reachable states  $R$  are computed. When restricting  $R$  to location  $Q_{bad}$  and projecting the result to  $T_{on}$ , the values for  $T_{on}$  can be obtained for which



**Fig. 9.** Reachable states in location  $Q_{bad}$

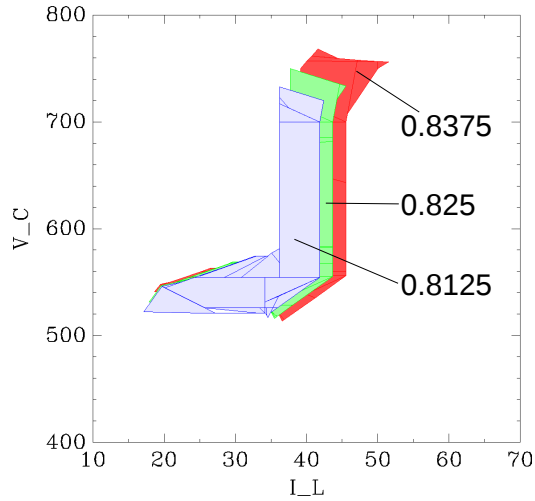
the system leaves its initial region after one period. In Figure 9 all the reachable states in  $Q_{bad}$  are shown, projected on  $x$  and  $y$ . For this experiment, the starting point  $X_0 = (38, 560)^T$  has been chosen with  $\delta = 5$  and  $\epsilon = 20$ . The white window in the plot corresponds to the good target region.

Unfortunately, there exist no values for  $T_{on}$  (or  $\alpha$ ), such that the system returns to the region around this  $X_0$  after one period. There are two possible reasons for this. First,  $X_0$  might not represent a valid stable state of the system, i.e. for any value of  $\alpha$ , it will deviate towards a different fixpoint. Second, due to the linear approximation, the computed reachable states diverge such that the final states at  $t = T_d$  do not fit into the target rectangle. The results in Figure 9 have been obtained for a partitioning with  $P = 8$ . The reachability analysis is expensive, already for  $P = 16$ , more than 300 MB of memory are needed.

Good values for  $\alpha$  can be found if the automaton is initialized with the single point  $X_0$ . For the example above, the interval  $\alpha \in [0.8393, 0.8525]$  was computed with partitioning  $P = 8$ , and with  $P = 16$  the larger interval  $\alpha \in [0.8254, 0.8576]$ . For  $P = 4$ , the interval is empty, meaning that the divergence is too large for the final states to fit into the target region. But even if a set of good values can be found, it cannot be guaranteed that  $X_0$  is a stable system state for any of the  $\alpha$  obtained in this way.

**Finding Stable States** Another idea is to find all the points  $X_0$  such that for a given  $\alpha$  the system stays within a bounded interval around  $X_0$  after one period. For this purpose, the same automaton as before is used, but with constant  $T_{on}$  and parameters  $x_0$  and  $y_0$ . The automaton is initialized with point  $X_0 = (x_0, y_0)^T$ . All bad parameters can again be obtained from the location  $Q_{bad}$  after a reachability analysis. The complement of these states represent the good states.

For two parameters, the reachability analysis is even more expensive. Thus, only for a very coarse partitioning with  $P = 4$ , the reachable states could be



**Fig. 10.** Stable states for different  $\alpha$

computed. As the states tend to diverge strongly at this precision, a large target interval with  $\delta = 10$  and  $\epsilon = 50$  was chosen. For these values, the resulting good states are shown in Figure 10, for three different  $\alpha$ . Again, due to the linear approximation and the fact that the target interval is reached from a single point, the existence of a real stable state cannot be guaranteed. But, if it exists, it must lie within the found region.

## References

- [DDM04] T. Dang, R. Donz, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 21–36, 2004.
- [FKR06] G. Frehse, B.H. Krogh, and R.A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In *Design, Automation and Test in Europe (DATE)*, pages 257–262, 2006.
- [Fre08] G. Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.