Circuit Verification by Projectagon Based Reachability Analysis

Chao Yan

June 3, 2010

Contents

ACTL	A Universal Fragment of CTL
AnaCTL	Analog CTL
AMS	Analog or Mixed Signal
APR	Arbitrary Precision Rational
AQTS	Approximated QTS
ASL	Analog Specification Language
BDD	Binary Decision Diagram
BP	Bisimulation Procedure
CDD	Clock Difference Diagram
CTL	Computation Tree Logic
CTL-AT	Analog and Timed CTL
DBM	Difference Bound Matrix
DI	Differential Inclusion
DTTS	I Discrete-Trace Transition System
ET	Ellipsoidal Toolbox
HA	Hybrid Automaton
HACDD	Hybrid Automaton with Continuous-Time and Discrete-Time Dynamics
HIOA	Hybrid Input Output Automaton
HLTS	Hybrid Labeled Transition System
ICTL	Integrator CTL
LDHA	Linear Dynamical Hybrid Automaton
LDI	Linear Differential Inclusion
LHA	Linear Hybrid Automaton
LHPN	Labeled Hybrid Petri Net
LP	Linear Programming
LTI	Linear Time Temporal Logic
MILT	Metric Interval Temporal Logic
NHA	Nonlinear Hybrid Automaton
ODE	Ordinary Differential Equation
ORH	Oriented Rectangular Hull
PCA	Principal Component Analysis
PCD	Piece-Wise Constant Dynamic
PDE	Partial Differential Equation
PIHA	Polyhedral Invariant Hybrid Automaton
PLL	Phase-Locked Loop
PSL	Accellera Property Specification Language
QTS	Quotient Transition System

RA	Rectangular Automaton
RTCTL	Real Time CTL
SAV	Simulation Aid Verification
SDHA	Sampled-Data Hybrid Automaton
SMT	Satisfiability Modulo Theory
STL	Signal Temporal Logic
STTS	Sample Trace Transition System
TA	Timed Automaton
TCTL	Timed CTL
TEDHS	Threshold-Event-Driven Hybrid System
THPN	Timed Hybrid Petri Net
TTS	Timed Transition System

Table 1: Abbreviations

1 Introduction

1.1 Motivation

The high cost of integrated circuit fabrication requires circuit designers to have a high degree of confidence in the correctness of a design before tape-out. The customary method for verifying a circuit design is to test a model of the circuit through extensive simulations. However, the correctness of complex designs cannot be ensured by simulating the model under a finite set of operating conditions. In fact, simulation coverage decreases as system complexity increases. Furthermore, simulation lacks well-defined criteria to determine if a given simulation result corresponds to a correct behaviour or not. Often the decision of whether or not a simulation result exposes a "bug" is left to the intuition of circuit designers. Simulation based verification is also expensive because a huge number of simulations are required to validate a complex circuit. The problem is more severe when the process variations of circuits fabricated in deep sub-micron processes are concerned.

Formal verification attempts to solve these problems in a mathematically rigourous way. It formally models a design, specifies correct behaviors, and automatically determines if all possible behaviors of the model are correct. Formal methods such as model checking have been well-studied and applied in industry for digital circuits. These techniques typically use a gate-level model as an abstraction of underlying physical transistors. However, such digital models ignore details of the underlying circuit behaviors thus they can not sufficiently represent behaviors of the actual circuit in some cases. For example, a circuit-level model must be used when precise timing information is required or "metastability" can occur. Furthermore, errors from analog circuits account for a growing percentage of total errors, and it is more and more common that analog components such as PLLs, A/D converters are embedded in digital designs. However, conventional formal verification methods can not be applied to analog or mixed signal (AMS) designs directly. Therefore, new algorithms and techniques are necessary to analyze digital circuits using an analog model or verify AMS circuits.

The difficulties of analog formal verification mainly originate from two aspects. First, it is difficult to represent a continuous region both efficiently and accurately. Operations on high-dimensional objects have exponentially complexity. Therefore, computing all possible states of a circuit is very expensive and is only practical for small circuits. Second, there does not exist a general way to precisely specify analog signals and analog properties. The verification of analog circuits is usually *ad hoc* and requires a large amount of human effort.

Some tools such as d/dt, CheckMate, etc have been developed in the hybrid system community. However, these tools are only suitable for very small hybrid

systems and can not be applied to verify circuits directly. Therefore, it is attractive to develop an efficient tool to formally model circuits, specify desired circuit behaviors, and automatically verify if the model satisfies these specifications.

1.2 Contribution

My thesis statement is:

> Projectagon based reachability analysis can formally verify the behaviour of digital and analog circuits using nonlinear, ordinary differential equation models.

With the reachability analysis tool COHO, we extend formal verification to synchronous or asynchronous digital circuits, and analog circuits.
The main contributions are

- A robust and efficient implementation of a projection based reachability analysis tool COHO for hybrid systems.
  - A projection based representation, projectagon, is applied to represent high-dimensional, non-convex regions with small approximation error and efficient operations.
  - New reachability algorithms based on projectagons are designed and implemented to compute the forward reachable regions for circuits where continuous dynamics are modeled by linear differential inclusions.
  - Interval computation and arbitrary precision rational (APR) arithmetic are applied to make the algorithm numerically stable. A set of approximate algorithms are developed to improve performance and reduce error.
- A framework of modeling and verifying circuits directly using COHO.
  - A circuit is modeled as a system of non-linear ordinary differential equation (ODE) system, which is locally approximated by linear differential inclusion to compute all possible circuit states by COHO.
  - Brockett's annulus construction is applied to represent a family of analog signals and specify analog properties of circuits.
  - We present a simple, table-based method for modeling transistors. These models closely match the more complicated models of commercial

simulators such as HSPICE. By using the same model for both simulation and verification, we can find simple bugs through simulation before expending the effort required for verification. Furthermore, the gap between the simulation and reachability results indicates places to increase simulation coverage and/or reduce approximation errors.

  - Techniques to reduce approximation error during reachability computation are developed, including multiple models, slicing, changing variables, decomposition, etc.

- Examples of practical circuit verification including synchronous/asynchronous digital circuits, and analog circuits.

  - We verified that a toggle circuit's reachable space is contained by an invariant set and its output signal satisfies the same specification of input.
  - We computed the reachable space of a latch circuit and a flip-flop and proved that the output of flip-flop is stable after clock-to-q delay.
  - We verified that a two-input asynchronous arbiter circuit satisfies several safety and liveness properties and the metastability filter works as an Brockett annulus transformer.
  - We showed that the 2-stage Rambus oscillator has only one oscillatory mode.

1.3 Organization

This proposal consists of six sections:

- Section 2 describes related research including reachability analysis methods and related tools.
- Section 3 describes our reachability analysis algorithm and our methods to speed up computation and reduce approximation error.
- Section 4 presents our verification framework and how COHO is applied during the verification.
- Section 5 describes digital and analog circuits that we have verified.
- Section 6 describes research plan and time line for the final dissertation.

2 Related Work

This section first introduces several reachability analysis tools from both the computer science and the control theory communities. Then we examine related algorithms and techniques used to solve four important problems for the reachability analysis of hybrid systems:

1. how to construct a mathematical model for the system to be verified,
2. how to formally specify properties,
3. how to represent a high dimensional region efficiently and accurately,
4. how to compute the reachable region of the system.

Finally, we explore some digital or AMS circuits that have been successfully verified. Surveys of formal verification methods and tools for hybrids systems or AMS circuits can be found in [SSKE01, ADF+06, ZTB08, BGG+09].

2.1 Reachability Tools

Formal verification of AMS circuits is a new area, and there are not many tools that work directly on formally verifying circuits. However, reachability analysis tools for hybrid systems as well as some other related tools have been used to verify nonlinear circuits. This section first introduces some tools for real-time systems, including UPPAAL and KRONOS. Then several tools for more complicated systems are described, including HYTECH, PHAVer, CheckMate, d/dt and LEMA. Finally, some tools from the control theory community are surveyed including VeriSHIFT and level set methods. More details of these tools will be discussed in the following sections. These tools and their features are compared and summarized in Table 2, 3 (on page 11 and 11).

UPPAAL [BDL04] is an integrated tool environment for modeling, simulation and verification of real-time systems, developed jointly by Uppsala University, Sweden and Aalborg University, Denmark. It consists of three main parts: a description language to describe systems as networks of timed automata, a simulator, and a model-checker to check invariant, reachability, safety, and liveness properties. Diagnostic traces are generated in case verification of a particular system fails. It has been applied successfully in case studies ranging from communication protocols to multimedia applications, but we have not found its application to circuits.

KRONOS [Yov97] is a software tool for formally checking whether a real-time system meets its requirements, developed at IMAG, France. It supports forward analysis, backward analysis, generation of counter-examples and time-abstracting

bisimulation checking. It has been used to verify real-time communication protocols, hybrids systems and timed asynchronous circuits [MY96, BMPY97].

HYTECH (The HYbrid TECHnology Tool) [HHWt95b] is an automatic tool for the analysis of embedded systems developed by Tom Henzinger *et al.* at UC Berkeley. It is one of the earliest tools for verifying hybrid systems modeled by linear hybrid automata. There have been four versions of HYTECH. The very earliest prototype [AHH96] was written entirely in the symbolic computation tool Mathematica and the algorithm is based on symbolic manipulation of expressions. However, symbolic operations were expensive. Therefore, the second version [HH95a] represented reachable regions by convex polyhedra to avoid the bottleneck in Mathematica, which is about 10 times faster than the first version. The third generation [HHWt95a] reimplemented the whole system in C++ and improved the performance by two or three orders of magnitude. However, HYTECH uses limited precision rational numbers for exact computations which can easily cause an overflow problem. The fourth version [HHMWt00], renamed as HYPERTECH, applied an interval based ODE solver to solve the problem and support nonlinear functions directly. Generation of error traces and parametric analysis [HHWt97] are also implemented in HYTECH. Numerous hybrid system examples have been verified by HYTECH and HYPERTECH.

Goran Frehse *et al.* from IMAG, France extended the idea of HYTECH and implemented the PHAVer tool [Fre08]. It uses arbitrary precision integer and rational arithmetic to solve the numerical over-flow problem of HYTECH. In its polyhedral computation, PHAVer uses the Parma Polyhedra Library [BRZH02], which is a library for exact computations of non-convex polyhedra. It also implemented a separate engine for compositional and assume-guarantee reasoning [FHK04, Fre04, Fre05]. PHAVer has been used to verify several examples including an oscillator circuit using a forward/backward refinement technique [FKR06].

CheckMate [CK01, CK03, SRKC00, SK00a, SK00b, SK01, KK02, GKR04] is a Matlab based tool for modeling, simulating and verifying properties of hybrid dynamic systems, developed by Bruce Krogh *et al.* from CMU. The bisimulation based verification algorithm is implemented in Matlab with a GUI interface using Matlab's Simulink and Stateflow toolboxes. It has been applied to verify several hybrid systems and some simple circuits.

d/dt [Dan00] is a tool for reachability analysis of continuous and hybrid systems with linear differential inclusions, developed by Thao Dang *et al.* from IMAG, France. It can solve reachability, safety verification and safety switching controller synthesis problems. It developed a technique to represent a region by a finite unions of hyper-rectangles. It has been used to verify several examples including both hybrid systems and control systems.

LEMA (LHPN Embedded/Mixed-signal Analyzer) [Lit08] is a verification tool

for AMS circuits developed by Scott Little, *et al.* from University of Utah. It models AMS circuits as Petri net based models which are compiled from VHDL-AMS or automatically generated from simulation traces. It implemented three engines to verify safety properties of AMS circuits: a DBM based model checker, a BDD based model checker and a SMT bounded model checker. It has been applied to hybrid systems and AMS circuits.

HySat [Her09] is a satisfiability checker, developed at University of Oldenburg, Germany, for combinations of boolean atoms, nonlinear arithmetic constraints and nonlinear ODEs. It can be used to analyze hybrid systems by bounded model checking.

There are some tools that support hybrids systems by introducing discrete variables into tools from the control community. VeriSHIFT [BT00] is a bounded time reachability analysis tool for hybrid automata with linear differential inclusions, based on ellipsoidal techniques [KV96, KV00a, KV00b, KV01]. It was developed by Botchkarve and Tripakis from UC Berkeley. Zonotope based analysis [Gir05, GGM06, GG08] has been applied to hybrid systems with uncertain linear systems. It uses a zonotope representation (see section 2.4) of the reachable region which can represent high-dimensional space efficiently. The level set toolbox [MT00, TMB$^+$03] extends level set methods, which are a class of numerical algorithms for simulation of dynamic surfaces, to support nonlinear hybrid systems.

2.2 Modeling the System

To formally verify properties of a design requires a formal model for the system and a formal statement of the specification. This section introduces some commonly used models for hybrid systems or AMS circuits, including hybrid automata, transition systems, and hybrid Petri nets. Formal methods of specifications are described in Section 2.3

A formal model for hybrid systems is hybrid automata (HA) [ACH$^+$95,HHWt96]. Hybrid automata have several definitions from different research groups. Informally, a hybrid automaton is a finite state machine augmented with continuous variables and dynamic equations. It consists of a graph in which each vertex, also called location, or mode, is associated with a set of differential equations (or inclusions) that defines the time driven evolution, referred as rate, derivative or flow, of continuous variables. A state consists of a location and values for all continuous variables. The edges of the graph, also called transitions, allow the system to jump between locations, thus changing the dynamics, and instantaneously modify variable values according to a jump condition. The jump may only take place when variable values satisfy a certain condition, specified by a guard, associated with

Table 2: Comparison of Hybrid System Verification Tools

Tool	Model	Spec	Space Representation	Space Approximation	ODE (directly)	ODE (approximation)	Algorithm
UPPAAL	TA	subset of TCTL	CDD	N/A	clock ODE	abstraction	compositional & symbolic model checking
KRONOS	TA	TCTL	symbolic expressions, DBM	N/A	clock ODE	N/A	on-the-fly symbolic model checking
HYTECH	LHA	ICTL	symbolic expressions, convex polyhedron	convex hull, extrapolation	$A\dot{x} \leq b$	clock translation, linear phase portrait approximation	symbolic computation, polyhedron operation, fixpoint
HYPERTECH	NHA	ICTL	interval	interval methods	polynomials, exponential, trigonometric	N/A	interval ODE integrator
PHAVer	linear HIOA	ICTL	convex polyhedron	limit the number of constraints and number bits	$A\dot{x} \leq b$	On-the-fly approximation	polyhedron operation, fixpoint
d/dt	LDHA	safety property	orthogonal polyhedron	face lifting	linear DI	face-lifting	integrator, optimal control
CheckMate	TEDHS, PIHA, DTTS, SDHA	ACTL	flow pipe	N/A	constant, linear, nonlinear ODE	N/A	optimization, error analysis
LEMA	THPN, LHPN	TCTL	zone, DBM	zone wrapping	constant ODE, constant DI	N/A	linear algebra, discrete model checking
HySat	NHA	safety property	interval	coordinate transformation	nonlinear ODE	N/A	interval methods, Taylor expansion, BMC
VeriSHIFT	LDHA	safety property	ellipsoid	N/A	linear DI	N/A	ellipsoid calculus
Zonotope	LDHA	safety property	zonotope	limit the order	linear DI	N/A	zonotope operation
Level Set	NHA	safety property	level set	N/A	nonlinear ODE	N/A	level set methods

Table 3: Comparison of Hybrid System Verification Tools (continued)

Tool	Hybrid Examples	Circuits Examples	Others
UPPAAL	Fischer's protocol, Philips Audio-Control Protocol, Train gate controller, Manufacturing plant, steam generator, Mine-Pump controller, Water tank	N/A	simulation, diagnostic trace
KRONOS	FDDI protocol, Fischer's protocol [DOTY96] CSMA/CD protocol [Yov97]	a 4-input circuit [MY96], XOR, 4-input AND [BMPY97]	counter example
HYTECH	generic railroad crossing [HHWt95a], Audio Control Protocol [HWt95],Steam Boiler [HWt96], etc.	N/A	diagnostic error trace, parametric analysis
PHAVer	navigation benchmark [Fre08], level monitor [FHK04]	TDO [Fre08], VCO [FKR06]	assume-guarantee reasoning
d/dt	collision avoidance, double pendulum [Dan00]	2^{nd} order Biquad lowpass filter, $\Delta\Sigma$ modular [DDM04]	
CheckMate	switched linear system [Chu99], batch evaporator [CK01].cutoff control problem [SRKC00], etc	TDO [GKR04], $\Delta\Sigma$ modulator [GKR04]	quick verification [SRKC00]
LEMA	level monitor, temperature controller, billiard game [Lit08]	TDO [LW04, LSW+06, Lit08], PLL model [LW04, Lit08], switched capacitor integrator [LSW+06, WLS+07, WLM07, LWJM07, Lit08], 2-stage Rambus Oscillator [Lit08]	construct LHPN from VHDL-AMS code or simulation data, error traces
HySat	ETCS level 3 [HEFT08], heater, bouncing ball, moving heaters [EFH08]	N/A	nonlinear constraints
VeriSHIFT	train-gate system [BT00]	N/A	bounded time verification
Zonotope	thermostat [GG08],200 dimensional random linear system [GGM06], two tank system [Gir05], etc	N/A	
Level Set	landing aircraft [TMB+03]	N/A	

Table 4: Continuous Dynamics Supported by Tools

Tool	ODE	DI	Approx. Tech
UPPAAL	$\dot{x} = 1$		abstraction (from $\dot{x} \in [c_l, c_h]$ to $\dot{x} = 1$)
KRONOS	$\dot{x} = 1$		
HYTECH	$\dot{x} = c$	$A\dot{x} \leq b$	clock translation (from clock translatable system to $\dot{x} = 1$), linear phase portrait approximate (from $\dot{x} = f(x)$ to $A\dot{x} \leq b$)
PHAVer	$\dot{x} = c$	$A\dot{x} \leq b$	on-the-fly approximation (from $A[x;\dot{x}] \leq b$ to $A\dot{x} \leq b$)
HYPERTECH	polynomials, exponential, trigonometric	$\dot{x} = Ax + Bu, u \in U, U$ is convex set	
d/dt	$\dot{x} = Ax + b$	$\dot{x} = Ax + u, u \in U, U$ is convex set	face lifting (from $\dot{x} = f(x)$ to $\dot{x} \in [c_l, c_h]$)
CheckMate	$\dot{x} = c, \dot{x} = Ax + b, \dot{x} = f(x)$		
LEMA	$\dot{x} = 1$	$\dot{x} \in [c_l, c_h]$ (BDD and SMT engines only)	
HySAT	$\dot{x} = f(x)$		
VeriSHIFT		$\dot{x} = Ax + u, u \in U$, U is convex set	
Ellipsoidal Toolbox		$\dot{x} = A(t)x(t) + B(t)u(t) + C(t)v(t), x_0 \in X, u \in U, v \in V, X, U$ is ellipsoid or hyper-rectangle, V is convex set	
Zonotope		$\dot{x} = Ax + Bu, x_0 \in X, u \in U, X, U$ is zonotope	
Level Set	$\dot{x} = f(x)$		

each transition. The modified values for continuous variables after the transition are also referred to as the reset map. The system starts from locations labeled as initial and may only remain in a location as long as the variable values are in a region called the invariant associated with the location.

Timed automata (TA) [AD94] are a simple class of hybrid automata proposed for modeling real-time systems. All continuous variables are clocks whose values change with the constant rate one (e.g. $\dot{x} = 1$). Singular automata [Hen00] are an extension of timed automata which allow the rate of variables to be any constant (e.g. $\dot{x}_i = c_i$). Rectangular automata (RA) introduce uncertainty to the dynamics using constant differential inclusion of the form $\dot{x} \in [c_l, c_h]$, the initial, invariant, and jump conditions for RA are all rectangles. A more powerful model is linear hybrid automata (LHA) [Hen00], where the system dynamics are linear differential inequalities of the form $A\dot{x} \leq b$, and all initial, invariant, and jump conditions are boolean combinations of linear inequalities. Multi-rectangular and triangular automaton are proposed in [Hen00]. Linear dynamical hybrid automata (LDHA) are hybrid automata with linear dynamics, such as linear ODEs or linear differential inclusions. A more generalized model nonlinear hybrid automata (NHA) use nonlinear ODEs to model nonlinear dynamics of systems. The reachability problem is to determine if a target state is reachable from an initial state. It is not decidable even for quite simple automata such as LHA [AD94]. Rectangular automata are the maximal class of hybrid automata with a decidable model-checking algorithm [HM00]. See [HKPV95, AD94, PV94, LPY98] for more details about decidability and undecidability of hybrid automata.

KRONOS [DOTY96] models real-time systems as timed automata. Similarly,

UPPAAL [LPY97] models a real-time system as a network of timed automata. It also developed an abstraction technique [BLL+96], implemented as a translator hs2ta, to transform LHA to timed automata.

HYTECH uses linear hybrid automata to model hybrid systems. Complex systems can be constructed using parallel composition. HYTECH [HHWt96] developed two techniques for approximating nonlinear systems by a linear hybrid automaton. The first clock transition method replaces continuous variables by clock variables and translates all invariant, initial and jump conditions accordingly. This technique only applies to clock-translatable systems, which require all variables to be independent of each other and variable values can be easily computed from the initial value and the elapsed time. This approach may duplicate discrete modes during the translation. The other linear phase-portrait approximation method approximates a nonlinear ODE by its lower and upper bounds. The state space is partitioned to make the approximation error small, but this may result in a large number of partitions.

PHAVer uses linear hybrid input output automata (HIOA) [Fre08] which impose additional structure on hybrid automata by declaring certain variables as inputs and outputs. The HIOA model supports PHAVer's reachability analysis engine and assume-guarantee reasoning engine. PHAVer developed an on-the-fly overapproximation technique [Fre08] which is a variation of the phase-portrait approximation method from HYTECH [HHWt96]. The method approximates affine hybrid automata, whose dynamics are affine systems with the form of $A[x;\dot{x}] \leq b$, by LHA. It has two methods to remove the x variables from the constraints. The projection method projects the space defined by constraints onto the space of derivatives. For very complex constraints, a constraint based method finds the bounds of variables x and transforms linear dynamics to linear constraints over $\dot{x}$. The constraint based method is more efficient than the projection method but its approximation error is usually larger. To make the error small, it partitions each reachable location into two or more smaller locations according to one or more user provided hyper-planes. Similar partition techniques have been used in [SK99, HHB02b].

However, TA or LHA are generally not expressive enough to accurately model systems with complex dynamics, especially nonlinear AMS circuits. LDHA have been applied to model linear systems or linear systems with inputs by some tools. For exmaple, d/dt models linear systems with inputs as LDHA with dynamic systems of the form $\dot{x} = Ax + U$, where the input set U is a convex and compact region. While computer scientists introduce progressively more complex continuous dynamics into traditional finite-state automata, control theorists approach hybrid systems by incorporating discrete behaviors into continuous dynamics. For example, ellipsoidal techniques [KV00a, KV00b, KV01, KV07] support forward or

backward reachability analysis for linear control systems. The algorithms for ellipsoids have been extended to support LDHA in VeriSHIFT [BT00]. An algorithm for computing the union of ellipsoids and the intersection of an ellipsoid and a polytope has been developed to calculate discrete jumps between locations of hybrid automata. The zonotope based method [Gir05, GGM06] supports LDHA with uncertain linear systems of the form $\dot{x} = Ax + Bu, x(0) \in I, u(t) \in U$, where both I and U are zonotopes. However, the guards of LDHA are restricted to be simple switching hyper-planes and the reset is always an identity map, because computing the intersection of zonotopes is usually very expensive. An efficient algorithm is developed in [GG08] for computing an over-approximation of the intersection of a zonotope and a hyper-plane, which will be described in more details in section 2.4.

NHA is rarely used because of the lack of efficient algorithms for solving nonlinear dynamics. [MT00, TMB$^+$03] supports NHA using the level-set method which finds the solution of a nonlinear ODE by solving a Hamilton-Jacobi partial differential equation. HySat [EFH08] computes an interval approximation of nonlinear ODEs using Taylor expansions.

Transition systems are a very general class of dynamical systems. A transition system consists of a set of finite or infinite states, a transition relation and a set of initial states. Bisimulation based model checking algorithms [Lyg03] work on transition systems which abstract away continuous behaviors of hybrid automata. For example, CheckMate models hybrids systems by discrete-trace transition systems (DTTS).

CheckMate supports the verification of threshold-event-driven hybrid systems (TEDHS) [CK01, Chu99]. A TEDHS has three subsystems: a switched continuous system (SCS), a threshold event generator(TEG) and a FSM. The SCS has digital inputs from the FSM and continuous outputs to the TEG; the TEG generates digital events for the FSM based on the continuous inputs. The front-end of CheckMate is built on top of Matlab's Simulink and Stateflow toolboxes, and has three important blocks: a switched continuous system block (SCSB), a polyhedral threshold block (PTHB) and a finite state machine block (FSMB), corresponding to SCS, TES, and FSM respectively. The TEDHS front-end model is translated to a polyhedral invariant hybrid automaton (PIHA) for the purpose of verification. A PIHA is a hybrid automaton with following constraints: 1) the invariant set of each location is a convex polyhedron; 2) discrete state transitions occur immediately when trajectories reach the boundary of the invariant set; and 3) the reset map is an identity map which implies there are no discontinuities in the continuous state space. A DTTS is constructed from the PIHA model for the bisimulation based model checking algorithm.

Furthermore, CheckMate supports the verification of computer controlled processes. A computer controlled process is a hybrid system where the continuous dynamic

process is controlled by a discrete controller and state transitions only occur at valid sampling times. CheckMate uses a sampled-data hybrid automata (SDHA) [SK00b] to model the computer controlled process. A SDHA is a hybrid automata augmented with a clock structure, which defines the valid sampling sequence by specifying the range of clock initial phases, clock periods and clock jitters. Of course, the clock events can be modeled in a standard hybrid automata by an additional continuous variable. However, compared with SDHA, this solution increases the order of dynamics and adds large number of states from sampling times when no event occurs. [SK01] extended the SDHA model to support clocked or unclocked events, and [KK02] presented a hybrid automaton with continuous -time and discrete-time dynamics (HACDD) to include continuous variables that evolve by difference equations in the controller. Both SDHA and HACDD are converted to transition systems for verification.

Hybrid Petri nets are an alternative model for hybrid systems. [LW04] presented a timed hybrid Petri net (THPN), which combines discrete Petri nets and continuous Petri nets, to model AMS circuits. The rates of continuous variables in the THPN is a constant. To model complex dynamics, the state space is partitioned into uniform hyper-rectangles, in which a THPN is constructed assuming the dynamic is constant in each region. The THPN model is enhanced to labeled hybrid Petri nets(LHPN) [LSW$^+$06] in LEMA. A key component of LHPN are the labels that label each transition with enabling conditions, bounds of delay, range of derivatives, etc. Another improvement of LHPN is that the derivatives of its variables are bounded by constant intervals. In LEMA, LHPN models are either automatically constructed from a subset of VHDL-AMS codes by a compiler [Lit08], or generated from simulation data by a simulation aided verification (SAV) method proposed in [LWJM07]. The SAV algorithm partitions the state space into bins based on a user provided threshold and calculates bounds of dynamics from the simulation data. This method does not ensure all system behaviors are covered especially when the number of simulations is not big enough or simulations are not performed properly.

2.3 Specification

Temporal logic is the most popular formalism for specifying properties of digital circuits. Linear Time Temporal Logic (LTL) and Branching Time Temporal Logic such as CTL are two of the most commonly used temporal logics. They have been used directly in reachability tools to specify properties of hybrid systems. For example, CheckMate [CK01] checks universal properties specified using ACTL [GL91], a universal fragment of CTL obtained by removing existential paths.

A behavior of a system can be viewed as a function from a time domain to its state space. Unlike digital systems, the state space is continuous and time is dense in analog or hybrid systems. The conventional temporal logic should be extended to address two conceptual difficulties: dense metric time and uncountable state space. The "untimed" temporal logic is extended for timed automata and real-time systems by putting constraints on temporal operators to limit their scope in time. Several researchers have explored "timed" logics with continuous time and discrete state. For example, Real Time CTL (RTCTL) [EMSS92] uses superscripts to bound the maximum number of permitted transitions along a path. Timed CTL (TCTL) [ACD90] puts subscripts on the temporal operators to limit the lower or upper bound of accumulated time over paths. Metric Interval Temporal Logic (MITL) [AFH96] constrains the LTL temporal operators with time intervals. It also drops the Next operator because it is meaningless for dense time domain. These extensions are used by many timed automata tools. LEMA [WLS+07] and KRONOS [Yov97] use TCTL to specify safety properties. In LEMA, TCTL specifications are either generated from assert statements of VHDL-AMS code by the compiler or provided by users for complicated circuits. TCTL statements are translated to T_μ calculus formulae to be verified by a BDD-based model checker [WLS+07] or a SMT bounded model checker [WLM07]. UPPAAL does not support the full version of TCTL. The subset of TCTL [LPY95] does not allow nesting of path formulae, but it is still sufficiently expressive for safety and bounded liveness properties.

Furthermore, logics have been defined where the state space is continuous. This allows, continuous variables to be used to specify analog properties. Analog CTL (AnaCTL) [DC05] replaces boolean variables with real-valued variables. It also introduces several special propositions including waveform proposition for analog systems. However, it does not support dense time, therefore, it has only been applied to verify transient response of analog circuit by discretizing state spaces and constructing a FSM from SPICE simulations. Analog and Timed CTL (CTL-AT) [GPHB06, HHB02b] extends RTCTL with analog comparison operators (greater and smaller operators) that allow sets of states to be defined as continuous regions of an analog circuit's state space. CTL-AMS [Jes08] is an extension of CTL-AT advanced by real-valued time interval and Boolean statements describing digital behavior. Integrator CTL (ICTL) [Hen00] supports both dense time and continuous space by three kinds of variables: continuous variables, control mode (discrete variable) and integrators. Continuous variables are used to represent continuous state spaces and integrators variables account for accumulated time over a path. HYTECH [Hen00] uses ICTL to express safety properties, liveness proper-

ties[1], time-bounded liveness properties and duration properties [AHH96]. PHAVer also uses ICTL to specify safety properties and timed-bounded liveness properties which are formulated as reachable problems by adding monitoring components [Fre08].

Accellera Property Specification Language (PSL) [FMW05] is a standard assertion language, which supports both LTL and CTL semantics. It is defined in four layers: the boolean layer is the foundation; the temporal layer specifies behaviors over time; the verification layer consists of directives which guide tools to check specifications; the and modeling layer defines the environment of a system. PSL is widely used in both assertion based verification and formal verification. An extended PSL is proposed in [SZDT07] for AMS systems. It replaces the boolean layer by a basic property which constrains circuit signals by linear equalities. However, it does not support dense time, thus ODEs of AMS systems are translated to difference equations during verification. To support both real time and analog properties, Signal Temporal Logic (STL/PSL) is proposed in [MP05]. It borrows syntax from MITL to support dense time and uses partial functions to specify analog signals. It also supports large classes of constraints with different complexity to define subsets of continuous spaces. The language is used in an analog systems monitoring tool, Analog Monitoring Tool (AMT) [NM07].

All CTL or PSL based languages are not friendly to analog circuit developers. To solve the problem, a new Analog Specification Language (ASL) is defined in [SH08]. It represents analog properties by continuous values and their alternation over time. It can specify complex static and dynamic circuit properties like oscillation and startup time. It is used on an analog model checking tool Amcheck [HHB02a].

2.4 Representation of Reachable Regions

Representation of reachable regions is crucial for reachability computation and determines the balance between accuracy of results and efficiency of algorithms. While Section 2.5 describes these algorithms; we describe several commonly used representations in this section along with the commonly used operations from these algorithms. These operations include intersection, union, and bloating.

Polyhedra or polytopes (bounded polyhedra) can represent a region with arbitrary accuracy. However, space and time complexity of operations on polytopes are exponential with the number of dimensions. Therefore, they are only suitable for low dimensional (2-3) regions, and general (non-convex) polytopes are rarely used in practice. Convex polytopes have a relatively simple representations:

[1]Liveness Properties are no longer supported since the second version of HYTECH because reachable regions are over approximated.

the inequality representation represents a region by a system of linear inequalities in the form of $Px \leq b$; the frame representation represents a region by its extreme points $\{x_1, \cdots, x_n\}$ and extreme rays $\{r_1, \cdots, r_m\}$ as: $P = \{x \in R^d | x = \sum_{i=1}^{n} \lambda_i x_i + \sum_{j=1}^{m} \mu_j r_j,\ \lambda_i, \mu_j \geq 0,\ \sum_{i=1}^{n} \lambda_i = 1\}$. Different operations may require different representations. There are several algorithms that translate each representation to the other [Che68, LV92]. Both representations are supported by Halbwachs' polyhedron-manipulation library [Hal93, HRP94].

For systems such as timed automata or linear hybrid automata, where the continuous dynamics are given by linear constraints on the derivatives of continuous variables of the form $A\dot{x} \leq b$, an exact reachability analysis is possible using standard linear algebra. Thus, convex polytopes are usually used to represent convex regions without approximation error, such as in the second [HH95a] and third [HHWt95a] versions of HYTECH as well as in PHAVer [Fre08]. HYTECH uses Halbwachs' library and computes reachable regions by geometric manipulation using both representations. Convex hulls are used to bound the union of two regions if they are bounded; otherwise, widening [ACH+95] or extrapolation methods [HH95b] are used. Both widening and extrapolation operations are based on the frame representation, and the latter method usually produces smaller approximation errors. Because HYTECH uses limited precision rational number for exact computation, overflow prevents it from solving large problems. To overcome this limitation, PHAVer uses the Parma Polyhedra Library [BRZH02] which employs arbitrary precision rational numbers. To limit space and time complexity, PHAVer simplifies a polytope by dropping linear inequalities from the inequality representation and reducing the number of bits by rounding rational numbers in the proper direction to guarantee conservative, over-approximations of the reachable space.

For more complex dynamical systems, reachable regions can not be represented exactly; additional space approximation error is inevitable on each computation step. The propagation of space approximation errors, known as the wrapping effect, generally has a large impact on the total error. To minimize the wrapping effect, convex polytopes are widely used to represent convex regions at a cost of requiring a large number of faces and expensive operations. CheckMate uses a flow pipe representation [CK01], which is essentially a convex polytope with the inequality representation, to represent the reachable region during a time interval $[t_k, t_{k+1}]$. It samples several points in the initial region, runs simulations from these points until time points t_k and t_{k+1}, and computes a convex hull of all simulation points as an under approximation of the real reachable region. The space complexity of a flow pipe is controlled by the number of sampled points: when fewer simulation points are used, a simpler flow pipe is obtained, but the over-approximation is larger. To contain all possible reachable points, the convex hull is bloated outward by a distance computed by solving a set of optimization problems. One benefit

of the simulation based approach is that it avoids the wrapping effect because all simulations start from the initial region. Furthermore, it allows parallel computation because all simulations are independent. However, simulation time becomes longer and longer and the number of simulations required to produce an accurate approximation increases exponentially with the number of dimensions. The flow pipe approximation has been extended to discrete time flow analysis (DTFA) to support difference equations in [KK02].

At the other extreme, a hyper-rectangle is one of the simplest possible representations of a reachable region. Its space complexity is linear with the number of dimensions, and the time complexities of operations on hyper-rectangles are typically small. Furthermore, each dimension can be handled separately. However, the biggest disadvantage is the lack of accuracy. Interval variables are another approach equivalent to hyper-rectangles. For example, HYPERTECH [HM00] uses an interval based ODE solver to solve the overflow problem of HYTECH. Although the space approximation error is much larger than that of convex polytope, the authors claim that the total error of HYPERTECH is much smaller than that of HYTECH because the interval ODE solver can solve nonlinear ODEs directly without model approximation error. HySat [FHT+07] also applies interval based methods to solve nonlinear constraints and nonlinear ODEs.

Several variations of hyper-rectangles have been developed to improve accuracy. [PKWtH98] uses face regions to represent reachable regions of rectangular hybrid automata [Pet96] and thereby solve the overflow problems of HYTECH. A face region over approximates an n-dimensional region by the convex hull of a set of $n-1$ dimensional hyper-rectangular faces with one dimension fixed to a constant value. The overflow problem is solved by rounding vertices of these faces outward.

The Oriented rectangular hull (ORH) representation was proposed in [SK03] to reduce approximation error by rotating a rectangle to a better orientation. The new orientation is computed by principal component analysis (PCA). The PCA method finds dominating correlations from singular value decomposition of a sample covariance matrix which is built from a set of sampled points. The space complexity of ORH is $O(n^2)$ where n is the number of dimensions. However, the number of sampled points required to find a good orientation increases exponentially with n. Furthermore, ORH are not closed under union, intersection, etc.

Another method to increase the accuracy of hyper-rectangle representation is to partition the state space into small pieces and represent a region by a union of hyper-rectangles. For example, d/dt represents a regions with orthogonal polyhedra [BMP99]. An orthogonal polyhedron is a finite union of uniform or non-uniform hyper-rectangles. This representation allows arbitrarily small approximation error, but the number of hyper-rectangles increases exponentially with the number of dimen-

sions. Therefore, it is only suitable for low (e.g. 2-3) dimensional spaces.

By adding difference inequalities in the form of $x_i - x_j \leq b$ into rectangles, LEMA introduces zone [LW04,LSW+06] to represent reachable regions. A zone is a simple convex polygon with only 45 and 90 degree angles, which is more accurate than rectangles by allowing dependence between variables but is still much more efficient than convex polytopes. Because of its special structure, a zone can be represented efficiently by a difference bound matrix (DBM) which represents each polygon edge by the constant term of its corresponding inequality.

The Minkowski sum of two sets A and B in Euclidean space is the result of adding every element of A to every element of B, i.e. the set $A \oplus B = \{a+b | a \in A, b \in B\}$. It is an important operation for solving linear systems with uncertain inputs, which will be discussed in more detail in the next section. This is exploited by the zonotope representation which is closed under the Minkowski sum operation. A zonotope is a polytope which can be represented as the Minkowski sum of segments. A zonotope has the form $Z = (u, (v_1, \cdots, v_m)) = \{u + \sum_{i=1}^{m} \alpha_i v_i | \alpha_i \in [-1,1]\}$, where u is the center, v_i is the generator, and $p = \frac{m}{n}$ is the order of the zonotope. Particularly, a hyper-rectangle is a special zonotope and a parallelepiped is a zonotope with order of 1. Zonotopes have many benefits. First, they are closed under linear transformations Φ and Minkowski sums, and there are efficient algorithms for these operations: $\Phi Z = (\Phi u, \langle \Phi v_1, \ldots, \Phi u_m \rangle)$ and $Z \oplus Z' = (u+u', \langle u_1, \cdots, u_m, v'_1, \ldots, v'_m \rangle)$. Second, the representation is very compact. For a zonotope encoded in $n(m+1)$ numbers, the number of its faces is up to $\binom{m}{n-1}$ and the number of its vertices is more than $(2p)^{n-1}/\sqrt{n}$. However, zonotopes have two main drawbacks. First, the order increases after each Minkowski sum operation. To reduce the order of a zonotope, [Gir05] presents an algorithm to compute an approximation of the zonotope. The algorithm first sorts all generators by the difference of first and infinity norms, then it replaces the first $2n$ generators with n new ones. To avoid wrapping effects introduced by the approximation, [GGM06] presents an algorithm which uses approximation for only a part of the zonotopes in a recurrent relation. Second, although the intersection of a zonotope Z and a hyperplane $n^T \cdot x = r$ can be easily detected in linear time, it is expensive to compute the result because the zonotope has to be converted to a polytope first. [GG08] presents an efficient algorithm which implementers the intersection operation. The algorithm first bounds the zonotope by several halfplanes with normal vectors in a given set $D = \{l_1, \cdots, l_k\}$. Then it projects Z onto each (n, l_i) plane. The projected 2D zonotope intersects the line $x = r$ and the intersection segment gives a lower and upper bound on the direction l_i. Finally an approximated zonotope is easily constructed from these constraints.

There are some representations from the control community. The ellipsoid

representation is used in ellipsoidal techniques [KV00a, KV00b, KV01] and VeriSHIFT [BT00]. An ellipsoid is defined as $\mathscr{E}(q,Q) = \{u : (u-q)^T Q^{-1}(u-q) \leq 1\}$, where q is the center and Q defines the axes. The space complexity for the ellipsoidal representation is quadratic in the number of dimensions, and the time complexity of ellipsoidal operations is also polynomial. Furthermore, a reachable region can be approximated with arbitrary small error through intersection (union) of a family of external (internal) ellipsoids. However, the ellipsoidal representation is not free of the wrapping effects because it is not closed under common operations such as intersection, union, etc. Ellipsoidal calculus is implemented in a Matlab toolbox, the Ellipsoidal Toolbox (ET) [KV06]. ET supports operations including affine transformation, Minkowski sums and differences, intersection and distance of ellipsoid with ellipsoid, hyperplane, halfspace, or polyhedra. An algorithm to compute the union of ellipsoids is presented in [BT00].

A level set [TMB$^+$03, MT00] of a real-valued function with n variables is the set where the function takes on a given constant value. When n is two, the level set is a level curve, when n is three, it is a level surface and for higher dimension, it is a level hypersurface. The function value on a point is its distance to the boundary of a region. If the value is negative, the point is inside the region, otherwise, it is outside the region. The zero level of the function is the boundary of region. Level sets are used to exactly represent surfaces of regions in level-set methods. However, the number of grids points required during the computation increases exponentially as the number of dimensions increases.

There are several tools that have used symbolic data structures to represent reachable regions. For example, LEMA uses BDDs in its BDD-based engine and SMT-based engine. Symbolic expressions, such as boolean combination of linear inequalities, are used to represent constraint systems by many tools, such as KRONOS, the first verion of HYTECH. UPPAAL generalized the ideas of BDDs and integer decision diagrams and developed clock difference diagrams (CDD) [ABB$^+$00] to represent a zone. The nodes of CDD represent variable differences. Each node can have several outgoing edges which are labeled with integral intervals. The CDD representation is more compact the standard DBM representation, but operations on CDD is a little more expensive.

2.5 Reachability Algorithm

Several commonly used methods have been developed to formally verify hybrid systems or AMS circuits. The first discretization approach converts a reachability problem in the continuous space into a problem in the discrete space which can be verified by conventional algorithms or tools. The reachability analysis method computes reachable regions by solving both continuous dynamics and discrete

dynamics. For a system with infinite states, the simulation (resp. bisimulation) method constructs and verifies a finite state system which simulates (resp. bisimulates) the original one. The compositional method partitions a complex system into several smaller subsystems and verifies each subsystem individually. This section discusses these techniques and their implementations in reachability tools.

2.5.1 Discretization

The discretization method constructs a discrete state model by partitioning the continuous state space into hyper-rectangular regions and computing the transition relation between these regions. All continuous variables including state variables, input variables and time have to be transformed to discrete states. This method has been applied to verify digital and analog circuits using circuit level models.

The first effort in applying model checking for circuit-level models is the work by Kurshan and McMillan [KM91], where the authors proposed a semi-algorithm for verifying digital circuits at the transistor level. The algorithm first partitions the continuous state space representing the characteristics of transistors into fixed size hyper-cubes and divides continuous time into uniform time steps. Input signals are divided similar but only logic low and high regions are used with the assumption of instantaneous transitions. Second, the algorithm computes the transition relation between these hyper-cubes using the lower and upper bounds of the continuous dynamics. The final constructed model is verified against properties defined by ω-language using a language containment tool, COSPAN [HHK96]. The partition is refined manually and the procedure is repeated if the verification fails. The method has been applied to verify that an arbiter presented by Seitz citeSeitz79 satisfies mutual exclusion, starvation freeness, and fairness properties.

The simple approach for discretization proposed by Kurshan and McMillan has been generalized for nonlinear analog systems by Hartong *et al.* [HHB02a, HHB02b] The extended work has several improvements. First, it uses a varied time step rather than a constant one. Second, it first divides the state space uniformly and then performs an automatic subdivision strategy to react on different system dynamics. The subdivision procedure is continued recursively until the variation of behaviors in a hyper-rectangle is smaller than a given threshold. Third, it proposed three algorithms for computing the transition relation between boxes. The first method computes an overestimated solution by interval analysis. The second approach chooses a number of test points and calculates the dedicated target points. However, the result is not always an over approximation unless all corner points are used. The method used in Kurshan's work is a special case of this approach, which assumes the current function is monotonic and uses the lower and upper corner values to bound the dynamics. The third approach makes the second process rigorous

using Lipschitz constants of nonlinear functions. Fourth, it introduces the CTL-AT logic as described in section 2.3 to specify analog properties. A modified model checker has been developed to check these specifications. This method has been applied to verify a Schmitt trigger and a tunnel diode oscillator. However, neither work presented a general solution to the problem of partitioning input functions. Like Kurshan and McMillan's earlier work, Hartong's algorithm assumes that the values of input signals do not change at all or change instantaneously over the whole input value range.

In [HC97], Hendricx and Claesen proposed a boolean based approach for AMS circuit verification. It represents analog signals including input variables by symbolic vectors and models an analog component by its functional behavior in terms of the symbolic representations. The symbolic model is verified by the Signal-Flow-Graph Tracing method developed at IMEC. The usefulness of this method has been demonstrated by verifying a computer input device *SmartPen*^{TM}.

The discretization method can use the well-studied conventional model checking techniques and tools. However, it has two problems: first, the abstraction ignores some analog characteristics; second, it is only suitable for small circuits because the number of grids is usually huge and increases exponentially with the number of dimensions.

2.5.2 Solving dynamic functions

Computing all possible system states is one of the fundamental tasks for formal verification. A fixed-point algorithm is the basic for reachability analysis of hybrid systems. In each iteration, a new reachable region is computed by applying $post_c$ and $post_d$ operators to the current reachable region S, where $post_d(S)$ is the set of states reachable by taking a transition from a state in S, and $post_c(S)$ is the set of states that result by letting time elapse without state transitions. The algorithm terminates when no new reachable region is found. Backward reachability analysis is performed similarly using pre_c and pre_d operators. However, termination is not guaranteed for a hybrid automaton which is more powerful then the rectangular hybrid automata [HKPV95, PV94].

Solving continuous dynamics of the $post_c$ operator is, in general, much more difficult than solving discrete dynamics of the $post_d$ operator. Thus, this section focuses on the solution of continuous dynamics. The evolution of deterministic dynamical systems is usually described by ordinary differential equations (ODEs). The commonly used ODEs include clock ODE of the form $\dot{x} = 1$, constant ODE of the form $\dot{x} = c$, linear ODE of the form $\dot{x} = Ax + b$, and nonlinear ODE of the form $\dot{x} = f(x)$. For example, UPPAAL and KRONOS support clock ODEs, HYTECH, PHAver and LEMA support constant ODEs, d/dt supports linear ODEs,

and CheckMate supports nonlinear ODEs directly. Differential inclusions (DIs) are used to model nondeterministic dynamical systems such as control systems, hybrid systems with inputs or disturbances. Many tools also use piece-wise differential inclusions to enclose complex ODEs. Two commonly used differential inclusions are constant DIs of the form $\dot{x} \in [c_l, c_h]$ and linear DIs of the form $\dot{x} \in Ax + U$, where U is the input (disturbance) set. For example, HYTECH and PHAver approximate linear and nonlinear ODEs by constant DIs. HYTECH also supports a generalized constant DI of the form $A\dot{x} \leq b$. d/dt supports linear DIs directly and approximates nonlinear ODEs by constant DIs. Table 4 lists the supported dynamics of each tool.

Mathematical solutions exist for some simple dynamics, including clock ODEs, constant ODEs, and constant DIs. Therefore, reachable region are generally computed exactly for timed automata and LHAs by either symbolic manipulation or geometry operations.

Alur, Courcoubetis and Dill [ACD90] developed one of the earliest modeling checking algorithms for real-time systems. It uses TCTL to specify properties and models the system as a timed graph which is similar with timed automata. It represents a signal state as a region and constructs a finite partition of the state space as a region graph. The major drawback is that the algorithm is enumerative and the size of the region graph is exponential on the number of clocks. In order to circumvent this problem, two alternative solutions have been proposed. The first symbolic method [HNSY94] represents a set of regions as a boolean combinations of linear inequality constraints. The second on-the-fly technique [HKV96] explores only the portion of the region graph required by the verification procedure. UPPAAL [LPY95] uses a symbolic model checking algorithm which represents the constraint systems by a compact data structure called a (Clock Decision Diagram) (CDD) [ABB+00]. It also solves the state explosion problem in the space of control nodes for parallel composition of timed automata by a compositional quotient construction which moves components from the parallel system into the specification. However, the compositional model checking only supports a subset of TCTL where nesting of path formulae is not allowed. KRONOS [BTY97] combines the on-the-fly and the symbolic approaches which is similar with the timed model checking (TMC) algorithm proposed by Sokolsky in [SS95]. However, KRONOS uses simulation graph to represent the state space which is more accurate than the symbolic graph used din TMC which constructs a quotient as coarse as possible.

LEMA analyzes LHPN models using three different model checkers: a DBM-based model checker, a BDD-based model checker and a SMT-based bounded model checker. The DBM based model checker [LSW+06] only supports clock variables. Therefore, constant ODEs and constant DIs are translated to clock ODEs. Constant DIs are first approximated by piece-wise constant ODEs. These

constant ODEs are converted to clock ODEs by a variable substitution technique which has the effect of warping the zone represented as a DBM in a given dimension. If the rate of a variable is negative, the DBM is warped into the negative space. This step converts all 45 degree angles into 225 degree angles which requires further approximation. A higher dimension problem can be reduced to several 2D problems [LSW+06]. The BDD-based model checker and SMT bound model checker convert the LHPN model to a symbolic model. The model has three components: an invariant for all system states, a set of possible rates to represent the dynamics, and a set of guarded commands. Safety properties represented by TCTL are translated to T_μ calculus formulae. The BDD-based model checker [WLS+07] uses a symbolic analysis algorithm from [SBB03] to solve the problem. The SMT bounded model checker [WLM07] uses the Yices [DM06] tool to solve the SMT formula translated from the symbolic model. If the formula is satisfiable, there is an error in the system; otherwise, the property can not be violated in the user specified number of iterations.

HYTECH supports both forward and backward analysis. In the first version, the *post* and *pre* operators were implemented by manipulating symbolic formulae in Mathmatica. However, the quantifier elimination operation of symbolic formulae is extremely inefficient. Therefore, the second version and the third version represent regions as convex polytopes [Ho95]. The *post* and *pre* operators are implemented as manipulation on convex polytopes using the Halbwachs' library. Several techniques have been developed to force the algorithm to terminate. First, forward and backward analysis are applied simultaneously [HH95b] to remove non-reachable regions introduced by approximation errors. Second, the reachable region is over approximated at each step. The overflow problem caused by limited precision rational numbers is solved using face regions for RHA as described in section 2.4. For LHA, HYPERTECH [HHMWt00] computes interval ranges of all variables using an interval based ODE solver: the ADIODES library [Sta97].

The reachable algorithm of PHAVer are nearly identical to the second and the third versions of HYTECH. It also implemented a forward/backward refinement algorithm [FKR06], which iteratively partitions the state space into smaller regions and trims the reachable region to generate more accurate results.

For more complicated dynamical systems, exact solutions are generally unavailable. Therefore, approximation methods or numerical techniques must be applied to solve these continuous dynamics. For linear DIs of the form $\dot{x} \in Ax + U$, two approaches have been developed for computing an over approximation of the reachable region R. The first optimal control method is based on the maximum principle of the optimal control theory [Var98]. The method finds the optimal input $u^*(t)$ which leads to the boundary of R. The optimal input for a hyper-plane

with normal vector d is of the form $u^*(t) \in \arg\max\{\left\langle e^{-A^T t}d, u\right\rangle | u \in U\}$, where $\langle a,b\rangle$ denotes the inner product of vectors a and b. The second Minkowski sum method approximates the reachable region by the Minkowski sum of a region $\hat{R}$ and an error region E as the equation $R \in \hat{R} \oplus E$. The region $\hat{R}$ is computed using the autonomous dynamics $\dot{x} = Ax$ and the error region E accounts the influences of inputs or disturbances. By integration, the reachable region $\hat{R}_{i+1}$ at time t_{i+1} is computed from a region R_i at time t_i using the autonomous dynamics: $\hat{R}_{i+1} = e^{A\delta t}R_i$. The error region can be bounded using error analysis. For example, for any nonlinear ODE with Lipschitz constant L, the radius r of the error region E is bounded by $r = ||R_{i+1} - e^{A\delta t}R_i|| \leq \frac{\mu}{2}(e^{L\delta t} - 1),\ \ \mu = \max_{u\in U}||u||$, which is proved by the fundamental inequality theorem from the theory of dynamical systems [ADG03]. The reachable space $R_{[i,i+1]}$ during time interval $[t_i, t_{i+1}]$ can be approximated by the convex hull of R_i and R_{i+1}. The convex hull should be bloated outward to contain all possible reachable states during the time interval. Therefore, the time step should be small enough to reduce the approximation error at each step. However, the time step can not be arbitrary small because the fixed-point algorithm also suffers from the wrapping effect which is more severe as the number of time steps increases. Therefore, finding an appropriate time step is crucial to reduce the total error.

d/dt applies the first optimal control method to solve linear DIs [Var98]. It integrates the optimal input $u^*(t)$ using a numerical integrator. Note that if the input set U is a bounded convex polytope, then $u^*(t)$ must be a vertex of the polytope. The algorithm only works on the boundary of an orthogonal polyhedron which is sufficient to find the extreme trajectories. It computes the intersection of all time-advanced half-planes as an approximation of the forward reachable region. The reachable region $R_{[i,i+1]}$ is over approximated by an orthogonal polyhedron at the end of each step.

The zonotope based reachability analysis method [Gir05, GGM06, GG08] uses the Minkowski sum approach to solve linear systems with input of the form $\dot{x} = Ax + Bu$, where the initial region and the input region are bounded by zonotopes. The error region E is a hyper-cube and consequently a zonotope with radius $r = \frac{e^{||A||\delta t}-1}{||A||}\mu$. The reachable region R^{i+1} is computed by the recurrent relation $R_{i+1} = e^{A\delta t}R_i \oplus E(r)$. Obviously, R_{i+1} is also a zonotope because the zonotope representation is close under Minkowski sum and linear transformation operations. The reachable space $R_{[i,i+1]}$ is approximated by a non-tight zonotope rather the convex hull of R_{i+1} and R_{i+1} because the convex hull of two zonotopes is generally not a zonotope. The non-tight zonotope encloses R_{i+1} and R_i with $2m+1$ generators where m is the number of generators of R_i. However, the order of the zonotope R_i increases by one after each step. The method described in section 2.4 can be

applied to reduce the order of the zonotope. However, the algorithm suffers from the wrapping effect. Therefore, an new algorithm is developed in [GGM06] based on rewritting the recurrence relation as

$$
\begin{aligned}
&X_0 = R_0, V_0 = E, S_0 = \{0\} \\
&X_{i+1} = e^{A\delta t} R_i, V_{i+1} = e^{A\delta t} V_i \\
&S_{i+1} = S_i \oplus V_i, R_{i+1} = X_{i+1} \oplus S_{i+1}
\end{aligned}
$$

where X_{i+1} is the reachable region of the autonomous systems from the initial region, and S_{i+1} is the reachable region of the system with inputs from the empty set $\{0\}$. The algorithm limits the order of the zonotope by over-approximating R_{i+1} by $\tilde{R}_{i+1}$ as

$$
\begin{aligned}
\tilde{S}_{i+1} &= \tilde{S}_i \oplus Hull(V_i) \\
\tilde{R}_{i+1} &= Hull(X_{i+1}) \oplus \tilde{S}_{i+1}
\end{aligned}
$$

where $Hull(Z)$ computes an over-approximation of a zonotope by either a hyper-rectangle or a zonotope with given generators [GGM06]. Therefore, the order of $\tilde{R}_{i+1}$ is a constant. The algorithm does not suffer from the wrapping effect because the $Hull$ operator is always applied to exact zonotopes.

The Minkowski sum method is also employed in the Ellipsoid toolbox [KV00a, KV00b], and VeriSHIFT [BT00]. The ellipsoidal technique uses this method for linear control systems with time-varying coefficients of the form $\dot{x}(t) = A(t)x(t) + B(t)u(t)$, where the initial region and control region are either ellipsoids [KV00a] or hyper-rectangles [KV00b]; and linear control systems with disturbances of the form $\dot{x}(t) = A(t)x(t) + B(t)u + C(t)v(t)$ where $v(t)$ is the disturbance. The reachable region is usually not an ellipsoid because the representation is not closed under the Minkowski sum operation. However, it can be approximated exactly as the intersection (union) of a family of external (internal) ellipsoids. The parameters of ellipsoidal approximation are described by a fairly simple ODE and can be solved efficiently by the algorithm in [KV00a, KV00b]. VeriSHIFT [BT00] uses the same algorithm with ET to solve linear DIs. An approximation of the reachable region $R_{[i,i+1]}$ is computed by enlarging R_{i+1} by a ball of radius r which is the maximum Hausdorff semi-distance between R_{i+1} and $R_{[i,i+1]}$.

For nonlinear ODEs, finding an approximated solution with reasonable error is generally very expensive. Several methods have been proposed to tackle this difficult problem. The hybridization method encloses a nonlinear ODE by a simpler differential inclusion within a small region. Interval methods are used for computing a safe numeric approximation in several tools. The error analysis approach bounds the solution using the Lipschitz constant. The level set method translates

a nonlinear ODE to a partial differential equation (PDE) which can be solved numerically.

The hybridization method splits the entire state space into small hyper-rectangles, calculates an over approximation of the dynamics on the partition of the state space, and computes the reachable region based on this approximated model. The piecewise model is a differential inclusion of the form $\dot{x} \in g_k(x) + U_k$, where the dynamic of $g_k(x)$ is chosen as simpler, e.g. constant or linear, functions. The methods described above can be used to solve these constant DIs or linear DIs. The balance between the approximation error and the number of regions should be considered to decide the size of hyper-rectangle regions. Many tools, including d/dt, LEMA, HYTECH and PHAVer, use the hybridization technique for complex dynamic systems which are not supported directly. d/dt presented a face lifting method [DM98] which encloses nonlinear ODEs with piece-wise constant dynamics (PCD) in each hyper-rectangle of an orthogonal polyhedron. The maximum derivative is used to calculate the largest distance that any point in the hyper-rectangle can move. This method is simple and also maintains the shape of the orthogonal polyhedron. The idea of approximating nonlinear ODEs by PCDs is employed similarly by the linear phase-portrait approximation method [HHWt96] of HYTECH as well as the BDD-based and SMT based engines of LEMA. PHAVer developed a on-the-fly approximation method [Fre08] which approximates affine systems of the form $A[x;\dot{x}] \leq b$ by linear systems of the form $A\dot{x} \leq b$. The approximation is computed by either projecting the affine system or finding the bounds of x variables.

Moore, Lohner and Stauning [Sta97, HW04, RMC08] proposed an interval-based, safe numeric approximation of nonlinear ODEs. The algorithm computes a Taylor expansion of the nonlinear ODE and an enclosure that contains the truncation error. It uses an interval method to calculate a safe interval approximation to contain all possible trajectories. This approach usually has large space approximation error and suffers from the wrapping effect because of its hyper-rectangle representations of reachable regions. A coordinate transformation method has been developed to minimize the wrapping effect, but it usually does not work well for nonlinear ODEs. HYPERTECH [HHMWt00] uses this interval based algorithm to support polynomial, exponential, and trigonometric functions directly using the ADIODES library. Although the error in approximating the reachable space is large, the authors claimed that the result of HYPERTECH is more accurate than that of HYTECH because the algorithm is free of model approximation error. HySat [FHT$^+$07, FH07, EFH08] also integrates the algorithm and uses the coordinate transformation method to minimize the wrapping effect. It constructs a formula which is satisfiable if and only if there exists a trace that starts from an initial state and ends in a target state in at most k steps. The satisfiable of the formula is checked by the iSAT algorithm, which is the algorithmic core of HySAT base on

the Davis-Putnam-Logemann-Loveland (DPLL) algorithm. The iSAT algorithm performs backtrack search to prune the search space until it is left with a sufficiently small portion of the space. The initial search space is the cartesian product of the interval ranges of all variables, which is pruned by alternating operations between decision step and deduction step. The decision step selects a variable, splits its interval range, and processes one part of them. The other parts are considered later by backtracking. The deduction step applies deduction rules to carve away portions of the search space that contain non-solution only. The algorithm terminates either the space is small enough or the progress is little.

CheckMate developed a flow pipe based approach for nonlinear ODEs. A flow pipe is a convex polytope which is constructed by bloating the convex hull of simulation points. The bloat distance is computed by solving an optimization problem which finds the extreme point of all possible trajectories along a given direction. If the dynamics is a linear ODE, the optimized problem in the time interval $[t, t+\delta t]$ is converted to the problem in the time interval $[0, \delta t]$. Then, the convex optimization problem is solved efficiently in Matlab [CK98, CK03]. However, for a nonlinear ODE, there is no efficient method to find the global optimal solution. Given the assumption that the ODE is a Lipschitz continuous function, CheckMate computes the largest distance r between any two possible reachable points. Then it uses a ball with diameter r as a conservative approximation of the reachable region [CK03, Chu99]. The space approximation error is generally very large unless the time interval is tiny and the initial region is very small. The algorithm is also very expensive by embedding large amount of numerical simulations into the routine for computing the objective function.

The level set method [MT00] can solve nonlinear ODEs directly and exactly. This method is based on the theorem that the solution of a particular Hamilton Jacobian PDE corresponds exactly the boundary of the reachable region of an nonlinear ODE. However, PDEs are even more difficult to solve than ODEs, and the number of grid points increases exponentially with the number of dimensions. Thus, it has so far only been applied to low (e.g. ≤ 5) dimensional examples.

2.5.3 Bisimulation

The bisimulation [Lyg03] method is an alternative approach for model checking of labeled transition systems with infinite states. It constructs a finite-state transition system that is equivalent to the original system according to a labeling function. The finite-state system can be verified by model checking. A standard approach to finding a bisimulation of a transition system is based on a quotient transition system (QTS). A QTS is constructed from a partition of the state space which must be consistent with the labeling function. The partition is computed by a bisimulation

procedure (BP) which refines the partition until all points in the same region have the same behaviors according to the labeling function.

CheckMate adopts this technique to verify the PIHA model and the SDHA model [CK00,CK99a,CK99b,CK01]. It constructs a discrete-trace transition system (DTTS) from a PIHA model to abstract away the continuous dynamics of the PIHA model. It computes a labeling function from its ACTL specifications which assigns to each state a set of satisfiable atomic propositions. Furthermore, it improves the standard BP algorithm to tackle two limitations. First, the standard BP algorithm only terminates for a limited class of transition systems [Hen95, LPY98, LPS99]. CheckMate solves the termination problem by verifying the QTS in the iteration of the BP algorithm rather than waiting for the algorithm to terminate. Although the QTS in the iteration is not a bisimulation, it is a simulation of the original system. Therefore, universal properties can be checked by verifying the QTS. Second, the exact pre-condition and post-condition sets required by the BP algorithm can not be computed for complicated systems. CheckMate uses an approximated QTS (AQTS) rather than the QTS for this problem. The construction of an AQTS only requires an approximation of the pre-condition set or the post-condition set. There exists a sufficient condition for the AQTS using the post-condition set to be a simulation of the DTTS; therefore, the post-condition set is used in CheckMate. An approximation of the post-condition set is computed based on the transition between elements of the partition, which can be calculated from the reachable region computed by the flow pipe method described in section 2.5.2. The SDHA model is converted to a sampled trace transition system (STTS) which is handled similarly. The principle difference of STTS, compared with DTTS, is that discrete state transitions happen at sampling time, which may not be the boundary of guard conditions.

2.5.4 Compositional reasoning

Compositional reasoning exploits the modular structure of a system and infers knowledge about the composed system. Assume-guarantee reasoning [HQR98] is a form of compositional proof, which analyzes a subsystem using assumptions about the rest of the system. Given a simplified model of each module, one original model is composed with the rest of the simplified system and its behaviors are verified. If the simplified model is a conservative abstraction of the original module, the proof is sound. Otherwise additional conditions are required to ensure that no undetected violations can occur.

Compositional reasoning with simulation relations was employed by Grumberg and Long [GL91] for discrete systems. Alur and Henzinger [AH97, HMP01] extended the assume-guarantee reasoning to hybrid systems and presented an suf-

ficient condition for assume-guarantee rules to hold. The condition is that every module must be receptive which requires that the module is not blocked by any of its inputs. PHAVer [Fre05] has a separate engine that supports assume-guarantee reasoning for the HIOA model with [FHK04] or without [Fre04] shared variables. It supports both non-cyclic assume-guarantee reasoning and cyclic assume-guarantee reasoning. It developed a less restricted sufficient assume/guarantee condition [FHK04] based on simulation relation, which is computed by a new semi-algorithm. The algorithm uses a hybrid labeled transition system (HLTS) and a timed transition system (TTS) [FHK04] to carve away the continuous activities of the HIOA model.

2.6 Application

The tools and methods described in the above sections have been applied to a large number of hybrid systems and control problems. For example, there are more than 20 hybrid system examples verified by HYTECH [HWt95, HWt96], including a generic railroad crossing problem which has 320 discrete states and 7 continuous variables [HHWt95a]. Table 3 lists some verified examples of each tool. However, most examples are low-dimensional, simple systems. The highest dimensional system is a 200 dimensional random generated linear system using the zonotope representation in [GGM06].

These tools have been extended to verify circuit designs. KRONOS has verified a 4-input 8-transistor circuit [MY96], a cascade of XOR gates and a 4-input AND gate [BMPY97]. However, timed automata are generally not sufficiently to model physical realities of circuits. For example, KRONOS uses a 3-state time automaton to model a transistor in [MY96] and a 4-state time automaton to model a XOR gate in [BMPY97]. More complicated models such as LHA, LDHA or even NHA are required to verify AMS circuits with accurate models. The tunnel-diode oscillator (TDO) has been verified by many tools, including PHAVer [Fre08], CheckMate [GKR04], the DBM-based model checker of LEMA [Lit08,LW04,LSW$^+$06], etc. However, it can not be verified by HYTECH because of the overflow problem. Another simple example is a 2^{nd} order Biquad lowpass filter. The reachable region of the filter was computed in [HHB02b] by partitioning the state space, and the property of absence of overshoots has been checked by d/dt [DDM04]. However, the state space of these circuits is only two-dimensional which is easy to represent and manipulate. Circuits with higher dimensional reachable regions have also been verified. PHAVer has computed the invariant set of a voltage controlled oscillator (VCO), which is a 3-dimensional system, using the forward/backward refinement technique [FKR06] CheckMate [GKR04] has verified a delta-sigma modulator which has four continuous variables. The stability analysis of the mod-

ulator is also performed by the d/dt tool [DDM04]. LEMA has verified several AMS circuits using the SAV method, including a switched capacitor integrator [Lit08, LSW+06, WLS+07, WLM07], a PLL phase detector [Lit08, LW04], and a two-stage Rambus oscillator [Lit08]. The integrator circuit has been verified by all of LEMA's three model checkers. The BDD-based and SMT-based engines require significantly more time to complete the verification and the BDD-based model checker produces a false negative result on the VHDL-AMS version of the corrected switched capacitor integrator. For the oscillator circuit, LEMA only measured the stability of oscillation for two simulation traces which is not enough to verify any general property of the circuit. Several circuits have been verified using the discretization method as described in section 2.5.1. For example, [HHB02b] computed the reachable regions of the tunnel-diode oscillator and Schmitt trigger circuit. [KM91] has verified the mutual exclusion, starvation freeness and fairness properties of a Seitz interlock circuit which is a 4-dimensional system. The functional correctness of a prototype of the SmartPen^{TM} device is verified in [HC97] using a pre-defined symbolic library. As we can see, the current verification tools either work on low-dimensional (e.g, ≤ 4) circuit systems or use simple, abstracted circuit models. Therefore, there is a need for more powerful reachability analysis tools for verifying practical AMS circuits.

3 Reachability Analysis Tool COHO

COHO is a reachability analysis tool for dynamic systems. The basic algorithm for computing forward reachable regions is described in section 3.1. To make the algorithm numerically stable, several techniques are presented in section 3.2. Finally, section 3.3 describes an extended algorithm to improve performance and reduce approximation errors.

3.1 Reachability Algorithm

As described in Section 2, reachability tools require a representation of the reachable space and a method for computing the reachable region at time $t+u$ given the reachable region at time t. As described in Section 3.1.1, Coho uses "projectagons" to represent reachable regions. The forward reachability computation is based on an Euler integrator applied to faces of the projectagon with care taken to ensure that all approximation are over approximations. Coho's integration algorithm is described in Section 3.1.2.

3.1.1 Projectagons

As discussed in section 2.4, the representation of reachable regions strongly affects the efficiency and accuracy of a reachability analysis algorithm. For example, approximating a region by the minimum hyper-rectangle that contains it as used in HYPERTECH and HySat is quite simple. The number of vertices/faces of hyper-rectangles increases linearly with the number of dimensions, and operations on hyper-rectangles are straightfoward. However, this representation introduces unacceptable approximation errors for most applications. At the other extreme, general non-convex high-dimensional polyhedra can represent arbitrary regions with relatively small errors. However, manipulations of general n dimensional polyhedra typically have time and space complexities with exponents of n or $n/2$. Accordingly, we are not aware of any reachability tools that use general high-dimensional polyhedra.

In COHO, reachable sets are represented as projectagons. For the purpose of this thesis, a projectagon is the high dimensional polyhedron formed by the intersection of a collection of prisms. Each prism is unbounded in all but two dimensions, and in those two dimensions the cross-section of the prism is a bounded polygon, which is called projection polygon. The projection polygons are not required to be convex; thus, non-convex, high-dimensional objects can be represented by projectagons. For example, Figure 1 shows how a three-dimensional object (the "anvil") can be represented by its projection onto the *xy*, *yz*, and *xz* planes. To ensure that the projectagon is bounded, each dimension of the full-dimensional polyderon must be in the basis of the two-dimensional subspace for at least one projection polygon. Therefore, the number of projection polygons of a n-dimensional projectagon is in the interval $[n-1, \frac{n(n-1)}{2}]$.

The high-dimensional object represented by a projectagon is the largest set of points that satisfies the constraints of each projection. It can be obtained from its projections by back-projecting each projection polygon into a prism in R^n and computing the intersection of these prisms. Obviously, this can lead to an overapproximation of the polyhedron. Thus, not all polyhedra can be represented exactly using projectagons. For example, the projectagon representation of an object with one or more indentations on its faces is an projectagon that has these indentations filled in.

Compared with other representations, projectagons offer several advantages. First, the representation is space efficient because it only tracks two-dimensional projection polygons rather than any full dimensional object. Rather than performing exponential-time operations on high-dimensional objects, Coho performs a linear number of polynomial time (typically $O(n \log n)$) operations on the projection polygons. This has the added benefit of letting Coho draw on the large body of al-

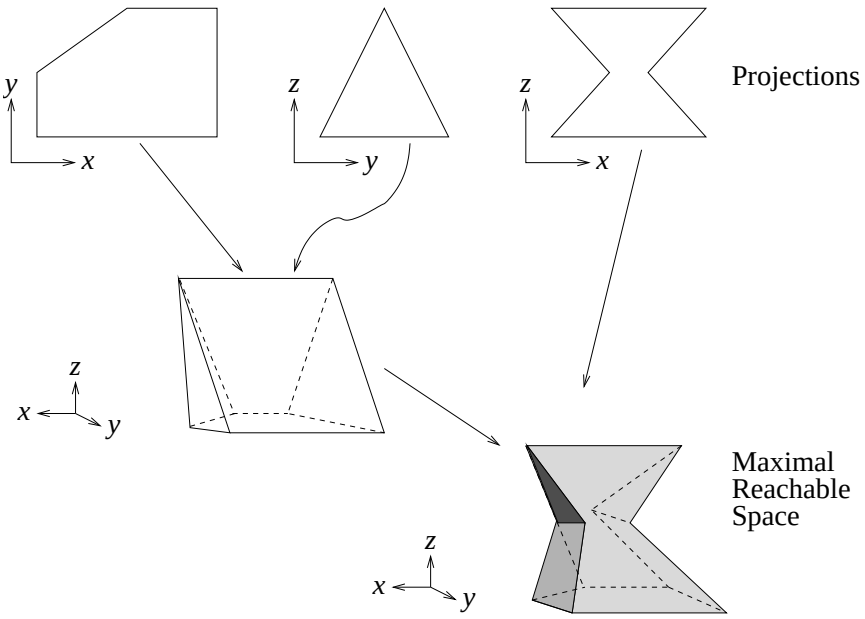


Figure 1: A Three Dimensional "Projectagon"

gorithms developed for two-dimensional computational geometry [PS85]. Second, ignoring degeneracies, faces of the object represented by a projectagon correspond to edges of its projection polygons. This allows many operations on faces including calculations of their trajectories to be carried out as simple operations on polygon edges. Furthermore, projectagons can represent non-convex objects efficiently, which is not supported by most of other representation methods as described in section 2.4. At the same time, the convex hull of a non-convex object is also recorded in the projectagon representation for efficient manipulation. The convex hulls of projection polygons can be represented by linear inequality constraints of the form

$$P \cdot x \leq q. \tag{1}$$

The matrix P has one or two non-zero elements in any row because each constraint corresponds to an edge of a two-dimensional projection polygon. The special property is used to develop an efficient solver for linear systems in [Laz01]. From equation 1, we can see that the convex hull of a projectagon corresponds to the feasible region of a linear program which has efficient and well-developed algorithms and tools. Accordingly, COHO makes extensive use of linear programming (LP) to mainpulate projectagons.

3.1.2 Computing forward reachable regions

Computing reachable regions is an essential step for the reachability analyses that are performed by most verification tools for continuous systems. Given a region $S \in R^n$ and a dynamic system X, the reachability problem is to compute a successor or advanced region $\delta_{t_1 \to t_2}(S)$ that contains all trajectories at time t_2 given that they started in S at time t_2. For simplicity, we write $\delta_t(S)$ to denote $\delta_{t_1 \to (t_1+t)}$ when t_1 is implied by context. COHO uses projectagons described above to represent the state region S, and supports all dynamic systems X that can be modeled by ordinary differential equations (ODEs).

To avoid the expensive numerical integration and guarantee the result is conservative, COHO over-approximates a non-linear ODE by a linear differential inclusion (LDI) of the form:

$$\dot{x} = Ax + b \pm u, \tag{2}$$

which is a special case of linear systems with uncertain inputs. A linear system with uncertain inputs is of the form $\dot{x} = Ax + u, u \in U$, where the input region U is a convex and compact set. An optimal control method is presented in [Dan00] for computing over-approximated advanced regions for such linear systems. For a face represented as $\langle \vec{n}, x \rangle = v$, where $\vec{n}$ is the normal vector of the face and $\langle a, b \rangle$ denotes the inner product of vectors a and b, they proved that the most outward face after time t is

$$\begin{aligned} \left\langle e^{-A^T t}\vec{n}, x \right\rangle &= v + \int_0^t \left\langle e^{-A^T s}\vec{n}, u^*(s) \right\rangle ds \\ u^*(t) &\in \arg\max\{\left\langle e^{-A^T t}\vec{n}, u \right\rangle | u \in U\}. \end{aligned}$$

It is well known that extremal trajectories are those emanating from faces as long as the dynamic system has bounded derivatives. Therefore, COHO advances a projectagon by computing the successor of each face of the projectagon in turn. The projectagon face is computed by intersecting the corresponding edge and the convex hull of the projectagon, thus can be represented by a set of linear inequality constraints of the form $Px \leq q$. By applying the method from [Dan00], the successor of the face with the LDI model in equation 2 is

$$\begin{aligned} P \cdot E \cdot x &\leq \hat{q} \\ E &= e^{-At} \\ \hat{q} &= q + P(I-E)A^{-1}b + P(I-E)A^{-1} \cdot * sign(P)u, \end{aligned} \tag{3}$$

where $\cdot *$ is the element-wise matrix product operator. Here we assume that the optimal input $u^*(t)$ for a face does not change during the time interval $[0,t]$ and

compute the values of $u^*(0)$ for all faces by the $sign(P)u$ operation. Because the input set U is a hyper-rectangle in our model and the optimal input $u^*(t)$ is the optimal point along the direction $e^{-A^T t}\vec{n}$, large time step is required to force the optimal point to move from one vertex of U to another. Therefore, this assumption is reasonable as the step size is usually very small during practical computation. This algorithm is quite efficient as it only uses matrix computations. The projectagon successor $\delta_t(S)$ for a projectagon S is approximated by the intersection of all face successors. And COHO over-approximates the reachable region during a time interval $[0,t]$ as

$$\delta_{[0,t]}(S) \quad \in \quad bloat(conv(S,\delta_t(S)),t\cdot||\dot{x}||_{max}),$$

where *conv* is the convex hull operator and *bloat* function moves each face of the convex hull outward by a given distance. Figure 2 illustrates the relation between a projectagon S, its successor $\delta_t(S)$ and the approximated region $\delta_{[0,t]}(S)$.

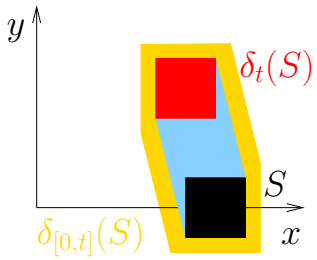


Figure 2: Successor of S by a continuous system X

Algorithm 1 shows the fundamental function of our COHO tool. It repeatly computes the successor of the current projectagon until no new reachable region is found. During each iterations, the algorithm first computes a LDI model for each face and then calculates the step size Δt. To make sure the LDI model is valid during this step, COHO assumes that any point in the projectagon moves along any direction by at most a user provided distance Δd, and approximates the system dynamic by a LDI in a bloated face computed by bloating the face outward by Δd. The step size Δt is determined by the maximum derivative of the LDI model over the bloated face. The conservative strategy guarantees that the LDI model always overapproximates the non-linear ODE during the time interval $[0,\Delta t]$. Therefore, our algorithm is sound. Given LDI models and the step size, COHO advances all faces of the projectagon by equation 3. To maintain the shape of projectagon, all advanced faces are projected onto their own two-dimensional subspace where the results are named as projected polygons. The face of the new projectagon is

constructed as the union of all projected polygons on the same subspace. Finally, the projection polygons of new the projectagon are simplified to save space and improve performance. The approximated projectagon is further clipped to ensure that its projection polygons are mutually feasible with each other.

Algorithm 1: Original COHO Algorithm

Input: S: current reachable region represented as a projectagon
Input: Δd: maximum moving distance allowed
Output: $\delta_t(S)$: the advanced projectagon

```
1  begin
2      S^b = lp_bloat(S, Δd);
3      for each projection polygon p do
4          for each face f do
5              f^b = lp_bloat(lp_bloat(f, −2Δd), Δd);
6              f.modelLP = lp_and(f^b, S^b);
7              f.model = model_create(f.modelLP);
8          end
9      end
10     Δt = +∞;
11     for each projection polygon p do
12         for each face f do
13             ||ẋ||_max = lp_max(f.model, f.modelLP);
14             Δt = min(Δt, Δd / ||ẋ||_max);
15         end
16     end
17     for each projection polygon p do
18         for each face f do
19             f^b = lp_bloat(f, −Δd);
20             f^b = lp_and(f^b, S);
21             δ_t(f) = int_forward(f^b, f.model, Δt);
22             poly_[f] = lp_project(δ_t(f));
23         end
24         δ_t(p) = poly_union(poly);
25         δ_t(p) = poly_simplify(δ_t(p));
26     end
27     δ_t(S) = ph_create(δ_t(p));
28     δ_t(S) = ph_clip(δ_t(S));
29 end
```

3.2 Numerical Issues

3.2.1 COHO LP solver

COHO makes extensive use of linear programs (LPs) to represent projectagons, compute advanced projectagon, calculate projected polygons, etc. The linear constraints correspond to either convex hulls of projection polygons as shown in equation 1 or an advanced projectagon as shown in equation 3, thus, these LPs are naturally written with the form:

$$\begin{aligned} \min_x & d^T \cdot x, \text{ s.t.} \\ & P \cdot E \cdot x \leq q \end{aligned} \tag{4}$$

where $P \cdot x \leq q$ are the constraints corresponding to the convex hull of a projectagon, d is the cost vector, and the optional matrix E is the backward time step operator for linearized model as shown in equation 2. We call such a LP a COHO LP in this dissertation. The dual [PS82] of a COHO LP is a standard form LP:

$$\begin{aligned} \min_u & -c^T \cdot E \cdot u, \text{ s.t.} \\ & P^T \cdot u = d \\ & u \geq 0 \end{aligned} \tag{5}$$

We implemented an efficient solver for a COHO LP (or its dual) based on the traditional Simplex algorithm. By exploiting the property that there are only one or two non-zero elements in any row of P, a linear-time algorithm for solving linear systems that arise in the Simplex algorithm is presented in [Laz01, Yan06]. In this algorithm, the tableau entries are computed from the original data at each pivot step; the only data carried forward from one pivot to the next is the set of columns in the basis and the original LP. By avoiding rank-one updates of the tableau from the traditional formulation of Simplex, our algorithm avoids error propagation from one pivot step to the next. The linear-time linear system solver makes this approach as efficient (to within constant factors) as traditional Simplex.

However, COHO LPs are occasionally extremely ill-conditioned, which makes the Simplex algorithm using floating point numbers fail to find an optimal solution. We solved the problem by applying interval computation and arbitrary precision rational (APR) arithmetic. Most computations are performed using interval arithmetic, and the APR package is used when it is numerically difficult to determine a favorable pivot. Thus our solver eliminates numerical stability problems and guarantees finding the exact optimum for a linear program in all cases.

3.2.2 LP Project Function

As shown in algorithm 1 (line 22), advanced faces need to be projected onto a two dimensional subspace to construct projection polygons for the new projectagon at each time step. The projection problem is to compute the projected polygon of a convex projectagon of the form $P \cdot v \leq q$ onto a subspace defined by basis vectors $\vec{x}, \vec{y}$. The idea behind our projection algorithm is to solve COHO LPs

$$\max_{v \in \mathbb{R}^n} (\vec{x}\cos\theta + \vec{y}\sin\theta) \cdot v \quad \text{s.t. } Pv \leq q \tag{6}$$

for all θ from 0 to 2π and use the optimal points to construct the projected polygon. Of course, it is impossible to solve equation 6 for every possible θ. In fact, COHO only solves one LP for one edge of the projection polygon, where the optimal direction $\vec{x}\cos\theta + \vec{y}\sin\theta$ is also the normal vector of the edge.

Our Simplex based solver works on the dual of equation 6:

$$\min_{u \in \mathbb{R}^{+m}} q \cdot u \quad \text{s.t. } P^T u = \vec{x}\cos\theta + \vec{y}\sin\theta. \tag{7}$$

When the solver finds a solution to equation 7, it also finds an optimal basis $\mathscr{B}$ and an optimal point $u = P_{\mathscr{B}}^{-T}(\vec{x}\cos\theta + \vec{y}\sin\theta)$ whose elements are all non-negative. By increasing the value of θ to some critical value, the basis $\mathscr{B}$ will no longer be optimal for the optimization direction $\vec{x}\cos\theta + \vec{y}\sin\theta$. The critical value of θ is the one at which u acquires a negative element, and the corresponding direction is orthogonal to the polygon edge that is from the current optimal vertex to the new optimum. Each successive value of θ can be determined by a single linear system solve in linear time. Similarly, the normal vector of the edge from previous optimal vertex to current one can be computed by decreasing θ.

I generalized the algorithm described above to project a time advanced projectagon of the form $P \cdot E \cdot x \leq q$ where E is the linear operator for moving a point at the end of a time step back to the corresponding point at the beginning of the time step (see Equation 3). Although the advanced projectagon is not a projectagon in the standard coordinate $\mathscr{C}$, it can be represented as a projectagon of the form $P \cdot u \leq q$ in the new coordinate $E\mathscr{C}$. Therefore, the problem is converted to project $P \cdot u \leq q$ on to a subspace with basis vectors $E\vec{x}$, and $E\vec{y}$.

3.2.3 Polygon operations

As shown in algorithm 1 (lines 24,25), many polygon operations are performed in COHO. To construct a projection polygon of the advanced projectagon, the union of projected polygons on the same two-dimensional subspace needs to be computed. Our algorithm first computes intersection points of all projected polygons using the

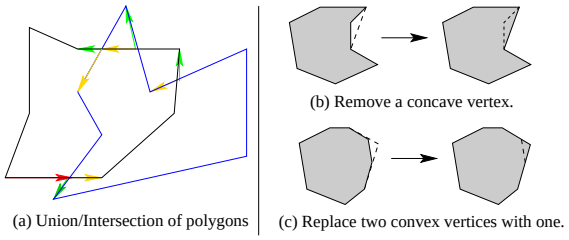


Figure 3: Polygon Operations

sweep-line algorithm in [PS85], then finds the union of these polygons by walking from the lower-left-most point in the anti-clock wise order and always selecting the right most edge on each intersection point. An example is shown as green arrows in figure 3(a). The intersection of polygons is computed similarly except the algorithm chooses the left most edge on each intersection point, illustrated as yellow arrows in figure 3(a).

COHO simplifies projection polygons at the end of each step for two purposes. First it keeps the number of constraints in the LP for the convex hull of a projectagon small. Second, it reduces the number of faces to be advanced at each step. To produce an over-approximated result, COHO can either delete a concave vertex or replace two consecutive, convex vetices with a single vertex as shown in figure 3(b) and figure 3(c) separately.

Similar with the LP solver, there are numerical problems when solving linear systems during the geometry computation. For example, the numerical errors are huge when computing the intersection point of two nearly parallel lines. We applied a solution similar with the one in the LP solver to the problem: polygons are represented by interval numbers and most computation are performed using interval arithmetic, when the condition number of a linear system is large, the APR package is used to compute the exact solution.

3.3 Performance and Accuracy

The algorithm described above is numerically stable; however, the reachability computation can be very expensive especially for high dimensional systems. Furthermore, the over-approximated result required for soundness may have too large error to verify a correct system successfully. To improve the performance and accuracy, I developed several new algorithms. Section 3.3.1 presents our methods for increasing the time step size and reducing the linearization error. The approximated algorithms for LP solvers and projection problems are presented in section 3.3.2

and 3.3.3. Section 3.3.4 describes algorithms to represent a projectagon face more precisely. The complete reachability algorithm with these improvements is shown in algorithm 2.

3.3.1 Step size and bloat amount

Choosing a good pair of step size and bloat amount for computing a LDI model is important to obtain good performance and small error. If the bloat amount is too small, COHO would take very small time steps resulting in long execution times and reachable regions that are overly conservative because of the erorr from the projection phase. Conversely, if the bloat amount is too large, then the non-linearity error (u in equation 2) will be large, causing another kind of over approximation and small time steps. In algorithm 1, the step size is computed using the ℓ_∞ norm of the derivatives. Therefore, it is usually very pessimistic and nearly always much smaller than what would actually be safe for the given bloat amount. When a face is advanced, the successor of the face at the end of time step would lie well inside the bloated face.

Noting the fact that the pair of step size and bloat amount is valid as long as the advanced face lies inside the bloat region used to create the LDI model[2], the new algorithm tries to improve performance by guessing a pair of step size and bloat amount based on the data from previous steps. At the end of each step, COHO checks that the estimated bloat is sufficient for the estimated step size. If not, COHO updates the bloat amount and/or step size and repeats the computation. In addition to enabling larger time steps, the guess-verify strategy speeds up the computation of each step by eliminating the step-size calculation phase.

In Algorithm 1, all variables are bloated equally. However, in a dynamic system, it is common that some variables change faster than others. For example, in digital circuits, a few signals will be in transition at any given time and the others will be relatively stable. This results in excessive bloating. To achieve an acceptable step size, the bloat for fast changing signals must be relatively large. When the same bloat is used for all variables, the bloat for slow changing signals is excessive, leading to much larger error terms in the differential inclusion than necessary. Likewise, when a signal is changing, it is generally either clearly rising or clearly falling. Thus, a large bloat is only needed in one direction, allowing the total bloat for these variables to be reduced by nearly a factor of two.

[2]The soundness requires that every trajectory from any point on the face at the beginning of the time step must remain in the bloated region throughout the time step. Otherwise, there is a potential unsoundness if a trajectory goes outside the bloat region and re-enters the bloat by the end of the time step. However, we ignore this detail and only check the advanced polygon at the end for efficiency in the implementation.

Algorithm 2: Revised COHO algorithm.

Input: S: reachable space represented as a projectagon
Input: $\Delta d, \Delta t$: bloat amount (step size) of previous step (optional)
Output: $\delta_t(S)$: the advanced projectagon
Output: $\Delta d, \Delta t$: bloat amount (step size) of this step

```
1  begin
2      repeat
3          S^b = lp_bloat(S, Δd);
4          for each projection polygon p do
5              for each face f do
6                  f^b = lp_bloat(lp_bloat(f, −Δd), Δd) ;
7                  f^b = lp_and(f^b, S^b) ;
8                  f.heightLP = intervalClosure(S, p, f);
9                  f.modelLP = intersect(f^b, f.heightLP);
10                 f.model = model_create(f.modelLP);
11             end
12         end
13         for each projection polygon p do
14             for each face f do
15                 f^b = lp_and(f, f.heightLP);
16                 δ_f(f) = int_forward(f^b, f.model, Δt);
17                 poly_[f] = lp_project(δ_t(f));
18             end
19             δ_t(p) = poly_union(poly);
20             δ_t(p) = poly_simplify(δ_t(p));
21         end
22         δ_t(S) = ph_create(δ_t(p));
23         δ_t(S) = ph_clip(δ_t(S));
24         Δd' = ph_bloatAmt(S, δ_t(S));
25         if Δd' ≤ Δd then Δd and Δt are valid;
26         else update Δd and Δt;
27     until Δd and Δt are valid ;
28 end
```

We implemented asymmetric and anisotropic bloating. Asymmetric bloating allows the positive and negative bloats for a variable to be different. Thus, bloating can adapt to the direction in which a signal is making a transition. Anisotropic bloating allows each variable to have its own bloat amount. Thus, bloating can adapt according to which variables are changing and which are stable. This approach allowed a significant increase in the typical step size. As an added benefit, the smaller total bloat reduced the error terms in the differential inclusion, allowing COHO to compute much tighter bounds on the reachable regions.

3.3.2 Approximated LP solver

Solving LPs is one of the most expensive computations in COHO. Although our LP solver uses an $O(n)$ linear system solver and only a small fraction of computation is based on APR number, the interval computation is much slower than ordinary double-precision arithmetic, and the APR calculations dominate the execution time of the solver.

To speed up the computation, the new algorithm uses ordinary double-precision arithmetic for each pivot. It then verifies that each pivot succeeded in reducing the cost function first by using interval arithmetic, and in the infrequent event that this fails, COHO uses APR. If the pivot failed to reduce the cost, COHO repeats the pivot step with interval arithmetic or APR. Likewise, at the end of the algorithm, COHO tests the optimality of the solution by verifying that it is feasible in both the primal and dual LPs, again using interval arithmetic first and APR if the result from the interval calculation is inconclusive. In this way, we obtain the certainty of APR while performing nearly all calculations using ordinary, double-precision arithmetic.

Noting that most of the LPs to solve occur in the projection algorithm, I further improved the LP solver by taking advantages of the special properties of these LPs. As shown in section 3.2.2, when θ in equations 6 and 7 is increased to force a pivot to the next edge of a projected polygon, the standard form LP becomes infeasible. Traditional formulations of Simplex assume a feasible basis; thus, the original algorithm restarted the LP solver to establish feasibility for each edge of the projection of each face. However, only a single pivot is required to re-establish feasibility in the absence of degeneracies. Accordingly, we modified our LP solver to try each column of P^T to determine if its introduction into the optimal basis of previous LP achieves optimality. This requires a single linear-system solve for each column tried which can be performed in $O(n)$ time due to the special structure of COHO's LPs. We found that this optimization works for about 80% of the projection polygon edges which resulted in a significant improvement in performance.

3.3.3 Approximated LP project algorithm

The projection of an advanced face at the end of a time step can have clusters of very closely spaced vertices separated by much larger gaps. These clusters arise from near degeneracies in the COHO LPs. To avoid a rapid growth in the number of vertices in the projection polygon, COHO performs a simplification step where the projection polygon is replaces by an enclosing polygon of smaller degree. Consequently, every vertex but one in a cluster will be discarded by the simplification process, but the projection algorithm expended a significant amount of computation time to determine these vertices. We avoided this extra work by enforcing a lower bound on the change of θ at each step of the projection algorithm as shown in algorithm 3.

The approximated algorithm can skip over vertices if the normals of the consecutive polygon edges are nearly parallel. Thus, the polygon obtained from the revised projection algorithm could be an under approximation which would violate the soundness requirement for COHO. Conversely, we can use each vertex from the projection algorithm to define a half plane, and construct the polygon defined by the intersection of these half-planes. The resulting polygon is an over-approximation. COHO computes both polygons. If their areas differ by more than a preset tolerance, COHO reverts to computing the exact projection polygon. Otherwise it uses the over-approximation.

3.3.4 Interval closure

In algorithm 1, COHO derives an LP for each projectagon face by intersecting the constraints for the face's edge with the bloated convex hull for each of the other projection polygons. If these polygons are non-convex, then this can produce a large over approximation of the face. Our solution is to perform an interval closure calculation. We can view each projection polygon edge as defining interval bounds for the two variables of the projection. The closure algorithm then applies these intervals to other polygons that include one of these variables in their basis to obtain bounds for other variables. This process continues until no further tightening of the interval bounds is possible or the process is below a threshold. The algorithm is simple, fast and greatly reduces approximation error when the projection polygons are non-convex. However, some care must be taken to preserve the soundness of the reachability algorithm. The problem is that although all extremal trajectories originate from faces of the projectagon, an extremal trajectory for one projection may emanate from a face whose edge only appears in another projection. Our solution is to bloat the bounding rectangle for each edge by the bloat amount Δd, intersect the resulting rectangle with the edge's projection polygon, and use the

Algorithm 3: Approximated projection algorithm.

Input: S: a convex projectagon of the form $P \cdot x \leq q$
Input: $\vec{x}, \vec{y}$: basis vectors of projection plane
Output: *poly*: projected polygon

```
1  for θ = 0; θ ≤ 2π do
2  |  [x_pt, B] = lp_opt(S, x⃗ cos θ + y⃗ sin θ);
3  |  poly_u = poly_u + x_pt;
4  |  planes = planes + plane_create(x_pt, x⃗ cos θ + y⃗ sin θ);
5  |  θ_n = θ;
6  |  repeat
7  |  |  θ_n = nextCritical(B, θ_n);
8  |  until θ_n − θ ≥ eps ;
9  |  θ = θ_n;
10 end
11 poly_o = plane_intersect(planes);
12 if poly_o − poly_u ≥ errTol then
13 |  call the exact algorithm
14 end
```

bounding box for this intersection as the starting point for the interval closure calculation. Algorithm 4 shows pseudocode for our interval closure calculation. Then COHO uses the intersection of hyper-rectangle R and the bloated face to describe region to compute the LDI model, and uses the intersection of R and the current edge to obtain an LP that contains all points on the current face. The soundness of the algorithm is proved in [YG08].

4 Verification of AMS circuits

To apply COHO to verify analog and mixed-signal circuits, a complete verification framework need to be developed. The section describes our methods to model a circuit by a ODE system, formally specify analog signals and properties, integrate COHO to compute circuit states, and check the properties at the end.

4.1 Modeling circuits as ODE systems

To get the dynamics of circuit systems, we model transistors as voltage controlled current sources and all capacitors as having fixed values with one terminal connected to ground. For a transistor, let $ids(v_s, v_g, v_d)$ be the current that flows from

Algorithm 4: Interval closure algorithm.

Input: S: a projectagon S of the form $P \cdot x \leq q$
Input: p: a projection polygon of S; e: an edge of polygon p
Input: Δd: the bloat amount for the current time step
Output: R: The hyper-rectangle of interval closure bounds for e

```
1 r = lp_bloat(e, 2Δd);
2 r = poly_intersect(r,p);
3 h = poly_box(r);
4 repeat
5 |   for each slice s do
6 |   |   h = intersect(h,s);
7 |   end
8 until h does not decrease ;
9 R = lp_bloat(h,2Δd);
```

the drain to the source when the voltages on the source, gate and drain are v_s, v_g and v_d respectively. We obtained our *ids* functions by tabulating data on a 0.01 volt grid for $(v_s, v_g, v_d) \in [-0.3, 2.25]^3$ for transistors in the TSMC 180nm, 1.8 volt, bulk CMOS process. By Kirchoff's current law, the total current flowing out of each node through the capacitors connected to the node must equal the total current flowing into the node through the transistors. The current through a capacitor is $c\dot{v}$ where c is the capacitance of the capacitor and $\dot{v}$ is the time derivative of the voltage across the capacitor. This yields:

$$\dot{V} \quad = \quad C^{-1} M \, ids(V) \tag{8}$$

where V is the vector of node voltages (one element for each node of the circuit); $ids(V)$ is the vector of drain-to-source currents (one element for each transistor); M maps transistors to nodes with $M(i, j) = 1$ if the source of transistor j is connected to node i, $M(i, j) = -1$ if the drain of transistor j is connected to node i, and $M(i, j) = 0$ otherwise. C is a diagonal matrix of node to ground capacitances. In the physically reasonable case that all capacitances are non-negative and every node has some capacitance to ground, C is positive definite, and thus invertible. We note that these assumptions can be violated by common macro-modeling methods, but don't address such macro-models in our current work.

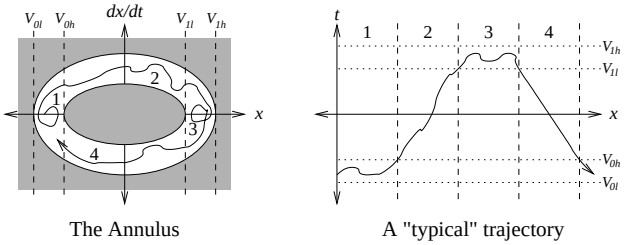


Figure 4: Brockett's Annulus

4.2 Brockett-based Abstraction

To specify the desired continuous behaviors of signals in AMS circuits, we use the Brockett annulus construction [Bro89] as shown in Figure 4. When a variable is in region 1 (referred to as B_1, likewise B_2, B_3, and B_4 for the other regions), its value is constrained but its derivative may be either positive or negative. Thus, region 1 of the annulus specifies a logically low signal: it may vary in a specified interval around the nominal value for low signals. When the variable leaves region 1, it must be increasing; therefore, it enters region 2. Because the derivative of the variable is positive in region 2, it makes a monotonic transition leading to region 3. Regions 3 and 4 are analogous to regions 1 and 2 corresponding to logically high and monotonically falling signals respectively. Because transitions through regions 2 and 4 are monotonic, traversals of these regions are distinct events.

This provides a topological basis for discrete behaviors. Furthermore, the horizontal radii of the annulus define the maximum and minimum high and low levels of the signal (i.e. V_{0l}, V_{0h}, V_{1l}, and V_{1h} in figure 4). The maximum and minimum rise time for the signal correspond to trajectories along the upper-inner and upper-outer boundaries of the annulus respectively. Likewise, the lower-inner and lower-outer boundaries of the annulus specify the maximum and minimum fall times. Note that a signal may remain in regions 1 or 3 arbitrarily long. In addition to the constraints captured by the geometry of the annulus, we add constraints as in [Gre96] for the minimum time that ϕ must remain in region 1 before entering region 2, and likewise for region 3. This construction allows a large class of input signals to be described in a simple and natural manner.

The mapping from continuous to discrete region provided by Brockett annulus allows methods for specifying digital circuits such as temporal logic to be extended to analog properties. Of course, human expertise is usually required to write the specification because the details of the Brockett annuli must correspond to particular designs, and often additional properties are needed to fully capture the intent

of the specification.

4.3 Integration with COHO

With the circuit model and specification, we first use MSPICE, a Matlab package developed for circuit simulation and analysis, to simulate the circuit before attempting verification to find any obvious errors. While the simulation speed is an order-of-magnitude slower than dedicated simulator such as HSPICE, MSPICE gives the user much greater flexibility and access to the numerical computations of the simulation. This allows, for example, inputs to be generated as random trajectories that satisfy their given Brockett annuli. Then the reachable region is computed by COHO, and the specification is checked for all possible reachable states at the end. If it succeeds, the property is guaranteed to be correct; otherwise, the reachable states that violate the specification are reported for further analysis.

From equation 8, we can see that it suffices to linearize *ids* to create a linear model required by the COHO algorithm as described in section 3.1.2. COHO implements two methods to linearize the current function of transistors. Given a linear program that defines a region for v_s,v_g, and v_d, the least squares method computes a linear fit for all points in the bounding box of this region. The algorithm uses constant time to compute linear coefficients using pre-computed sums of tabulated data. Noting that the *ids* function has exactly one inflection (along the $v_s = v_d$ plane) enables efficient computation of the worst-case errors of the least-square model. Coho then adjusts the constant term of the linear-regression to balance the positive and negative worst-case errors. However, the table of simulation data generated by HSPICE uses too much memory for higher dimensional (e.g. more than three) systems. The quadratic interpolation method uses smaller tables of which each entry holds the coefficients for a quadratic polynomial approximation of the current functions. It uses a higher-order polynomial to allow coarser grid, for example, 0.1 and support circuit components with a larger number of dimensions. A linear model is computed by performing a least squares fit to the polynomial model. The region where the linear model is computed can be specified by linear programs rather than a bounding box, therefore, its linearization error is usually smaller than that of the least squares method if the region is large and cannot be well approximated by a hyper-rectangle. However, the value computed by adjacent cells do not necessarily agree at the boundaries. To make the model continuous, we smooth our interpolated values by a weighted cosine window. For example, let g_0, $g_1, \ldots$ be a set of quadratic functions where g_i is the function for $i-1 \le x \le i+1$, the approximation of function $f(x)$ is

$$\begin{array}{rcl} f(x) & \approx & w_m(x-\lfloor x \rfloor) g_{\lfloor x \rfloor}(x-\lfloor x \rfloor) + w_m(x-\lceil x \rceil) g_{\lceil x \rceil}(x-\lceil x \rceil) \\ w_m(x) & = & \frac{1}{2} + \frac{1}{16}\left(9\cos(\pi x) - \cos(3\pi x)\right). \end{array} \tag{9}$$

This smooth model can be used by the integrator of MSPICE.

To verify an analog circuit successfully, it is important to control approximation errors from COHO. The linearization step described above may introduce large errors during the computation of advance projectagons as shown in equation 3. There are two methods to reduce the linearization error: the multiple models method and the slicing method. First, the intersection of multiple linear differential inclusions provides a tighter bound of the non-linear dynamics. For example, if the drain voltage is greater than the source voltage, the current of NMOS transistor is always positive, but negative current may be introduced during the linearization process, resulting in non-physical behaviors. The problem can be solved by adding a transistor model that simply determines the minimum and maximum drain-to-source currents. Of course, the disadvantage of using multiple models is that each face should be advanced and projected several times. The second method, slicing, avoids computing LDI models for large regions. It is quite common that the reachable region of a circuit tends to have long skinny projection polygons with a clear flow along the long axis of these projections. This allows us to slice the reachable space into several ordered phases. Then COHO computes the reachable region in the first phase and the intersected slab that is the intersection of the reachable region and the hyper-plane between the first and the second phases. The reachable regions for other phases are computed similarly with an initial region as the intersected slab from the previous phase. For example, there are four discrete regions of a signal specified by Brockett annulus, but the rising and falling regions are usually divided into several small phases to avoid large intervals of the signal voltages.

An additional benefit of slicing is it is possible to run the reachability computations for all of the phases in parallel. To make the computation of each phase independent, the assume-guarantee strategy is applied. For each phase i, we assume that the circuit state is in a hyper-rectangle R_{i-1} at the end of phase $i-1$, solve the reachability problem for this phase with initial region R_{i-1}, and show that the circuit state is enclosed by the hyper-rectangle R_i at the end of phase i. This strategy is crucial for parallel computations which is an important method to speed up the computation of reachable regions in COHO. It also makes it easier to find critical phases which have large approximation errors, and consequently apply more expensive algorithms to these phases to obtain a more accurate result.

Noting the smoothness of the cosine window model, we can apply the same model to both simulation and verification and compare their results. Obviously, the simulation result is an under-approximation of all possible circuit states, and the reachable region computed by COHO is an over-approximation. We use the gap between the results from MSPICE and COHO to manually guide MSPICE to increase coverage and COHO to reduce errors. The simulation result is also a good starting point for guessing a hyper-rectangle R_i for each phase in the assume-guarantee

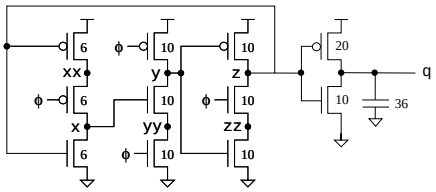
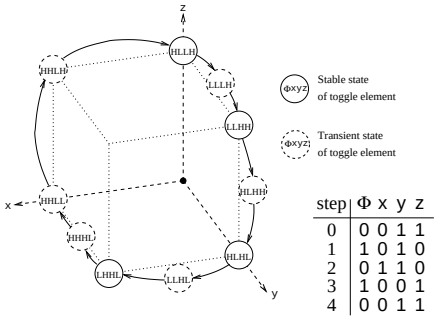


Figure 5: Toggle Circuit



step	Φ	x	y	z
0	0	0	1	1
1	1	0	1	0
2	0	1	1	0
3	1	0	0	1
4	0	0	1	1

Figure 6: State Transition Diagram

method.

Another general idea for verifying large systems is to decompose a complicated circuit system into several subsystems. The method can greatly improve performance,because running time grows rapidly with the dimension of the system. For example, during the verification of the toggle circuit as shown in section 5.1, we first compute the reachable region of the subsystem which ignores the output inverter, and then perform a separate reachability analysis for the output inverter. Using Brockett annuli for both input signals and output signals makes it sound to decompose a circuit and perform reachability analysis for these smaller systems.

5 Examples

We applied the reachability analysis tool COHO and the verification methods described in section 4 to several circuits, including a synchronous toggle circuit in section 5.1, a flip-flop circuit in section 5.2, an asynchronous arbiter in section 5.3 and an analog differential ring oscillator in section 5.4. The successful verifications of these examples show the effectiveness of verifying analog circuits by projection based reachability analysis.

5.1 The Yuan-Svensson Toggle

Figure 5 shows the toggle circuit from [YS89]. Transistors are labeled with their shape factors and the capacitor on the *q* output represents a load equivalent to the gate capacitance of transistors with the a total shape factor of 36; this is the load that the toggle places on its clock input. We use this load to verify that the output of one toggle can drive the clock input of another to implement a ripple counter.

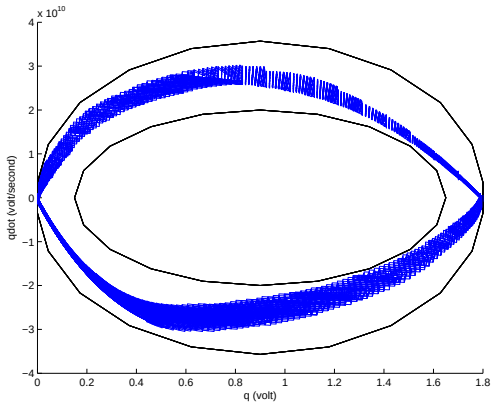


Figure 7: The Brockett Annulus for the output of the Yuan-Svensson toogle

The operation of this circuit can be understood by using a simple switch model starting from a state where the ϕ input is low. In this case, y is driven high, z is floating, and x is the logical negation of z. Figure 6 shows the state transition diagram for the toggle starting from the state where z is high when ϕ is low – the other case, with z high, is reached on step 2 of the figure. Note that from step 2 to 3 in the figure, all three of x, y and z change values. This is a critical race for the toggle.

We specify the behavior of the toggle as a safety property. In particular, we use COHO to find an invariant subset of $\mathbb{R}^d$ such that all trajectories in this set have a period twice that of the clock signal. Accordingly, our reachability calculation is carried out for two periods of ϕ. We break each of these periods into two phases: one for the rising transition of ϕ and the time that ϕ is high; and the other for the falling transition and low time. Using the assume-guarantee strategy, the task of verifying the toggle is divided into four separate reachability problems. The bounding hyper-rectangle for the end of each phase is estimated based on simulation results. By showing that the last phase leads to a hyper-rectangle that is contained in the initial hyper-rectangle of the initial phase, we establish that the reachable set that we have computed is invariant.

The Brockett annulus of the output signal is computed from the invariant set and shown in figure 7. Clearly, the output satisfies the same Brockett annulus that we used for the input. Thus, these toggles can be composed to form an arbitrarily large ripple-counter as desired.

5.2 Latch and Flip Flop

A latch is a circuit that has two stable states and thereby is capable of serving as one bit of memory. Figure 8 shows a static, transparent pass gate latch that we verify. The input signal can not change when the clock falls because of the metastability issue. The property to check is that the output signal is stable when the clock is clearly low.

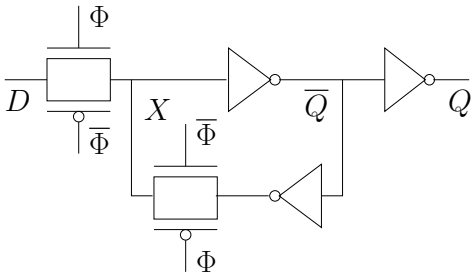


Figure 8: The pass gate latch circuit

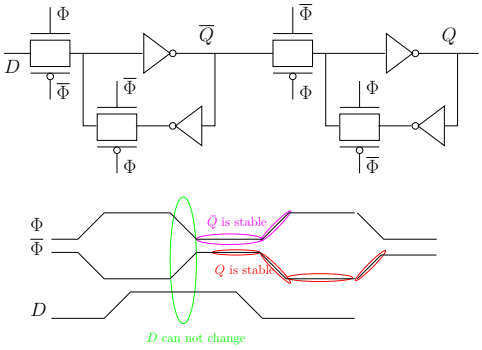


Figure 9: The flip flop

Unlike the toggle circuit, a latch has both a clock signal and a data input signal. Both of them are specified by Brockett annuli which can be mapped to four digital regions. Therefore, there are sixteen regions for the combinations of these two inputs as shown in figure 10. Of course, the states $\langle \Phi_{fall}, D_{rise} \rangle$ and $\langle \Phi_{fall}, D_{fall} \rangle$ can be omitted because of the input specification. Due to the requirement of a minimum stay time T in the stable regions, there are two kinds of transistion for each stable state. As an example, consider when the inputs are in discrete state $\langle \Phi_{low}, D_{low} \rangle$ and Φ has been low for at least T time units, but D has been low for less than T. From this condition, the inputs can transition to $\langle \Phi_{rise}, D_{low} \rangle$ but not to $\langle \Phi_{low}, D_{rise} \rangle$ or $\langle \Phi_{rise}, D_{rise} \rangle$ because only the clock is allowed to change at this point. These kinds of transitions where only one signal satisfies the time requirement are shown as dotted lines in figure 10. When both input signal and clock satisfy the time requirement, transitions are shown as solid lines in figure 10. Noting that the reachable regions also depend on the initial states of internal nodes, two reachability problems with the same input signal and clock are solved for different latch states. Therefore, there are six independent phases to compute the reachable regions using the assume-guarantee strategy.

1. From $\langle \Phi_{low}, D_{low} \rangle$ to $\langle \Phi_{high}, D_{high} \rangle$ with $x = low$.
2. From $\langle \Phi_{low}, D_{low} \rangle$ to $\langle \Phi_{high}, D_{high} \rangle$ with $x = high$.

3. From $\langle \Phi_{low}, D_{high} \rangle$ to $\langle \Phi_{high}, D_{low} \rangle$ with $x = low$.
4. From $\langle \Phi_{low}, D_{high} \rangle$ to $\langle \Phi_{high}, D_{low} \rangle$ with $x = high$.
5. From $\langle \Phi_{high}, D_{low} \rangle$ to $\langle \Phi_{low}, D_{high} \rangle$ with $x = low$.
6. From $\langle \Phi_{high}, D_{high} \rangle$ to $\langle \Phi_{low}, D_{low} \rangle$ with $x = high$.

As noted above, the specification for the inputs forbids scenarios where Φ and D are both falling at the same time. Thus, for the 6^{th} phase, COHO considers the two sub-cases: $\langle \Phi_3, D_3 \rangle \rightarrow \langle \Phi_4, D_3 \rangle \rightarrow \langle \Phi_1, D_3 \rangle \rightarrow \langle \Phi_1, D_4 \rangle \rightarrow \langle \Phi_1, D_1 \rangle$ and $\langle \Phi_3, D_3 \rangle \rightarrow \langle \Phi_3, D_4 \rangle \rightarrow \langle \Phi_3, D_1 \rangle \rightarrow \langle \Phi_4, D_1 \rangle \rightarrow \langle \Phi_1, D_1 \rangle$ in a single run. It is similar for the 5^{th} phase.

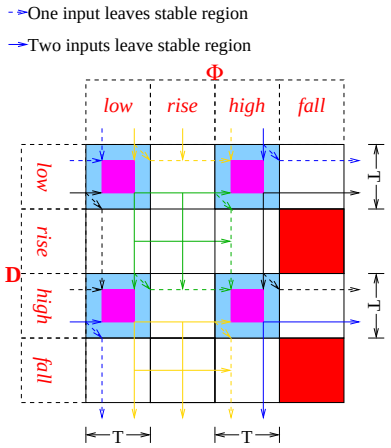


Figure 10: State transition

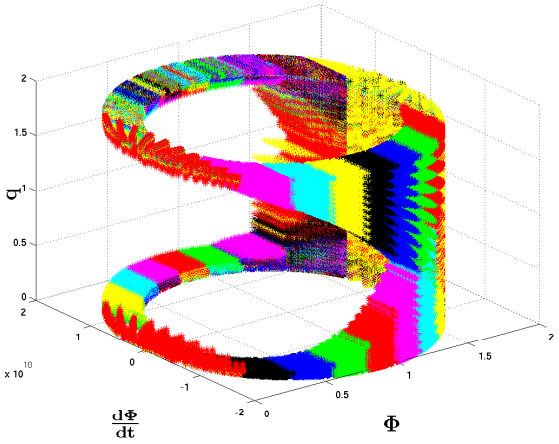


Figure 11: The output specification

Figure 11 shows the value of $\overline{Q}$ respect to the clock. The plot shows that if the input signal is stable when the clock is falling, then the output signal is stable when the clock is clearly low or rising. Based on the reachable regions, it is also measured that the minimum time T can be as small as $0.27ns$. Therefore, the highest frequency is about 1.3GHz considering the rising/falling time as defined by the input specification. However, there is still a big gap between the result from COHO and the result from the MSPICE simulator which finds an upper bound of the clock frequency around 4GHz[3].

A flip-flop can be made by connecting two latches as shown in figure 9. For the first latch, the output $\overline{Q}$ is stable when Φ is rising, which also indicates the input signal of the second latch does not change when $\overline{\Phi}$ is rising. Therefore,

[3]4GHz is very fast for a 180nm process. The capacitance values in our MSPICE models might be too small and generate this high value. We will solve this problem in the actual thesis.

the output Q of the second latch is stable when $\overline{\Phi}$ is low or rising. However, $\overline{Q}$ usually becomes stable before the next rising edge of Φ as shown in figure 9. As measured, the delay from the time when Φ starts to fall to the time when Q is stable is about $200ps$, which is an upper bound of the clock-to-q delay of the flip flop. A similar method can be applied to measure the upper bound of circuit delay for static analysis which is widely used in the digital design.

5.3 Arbiter Circuit

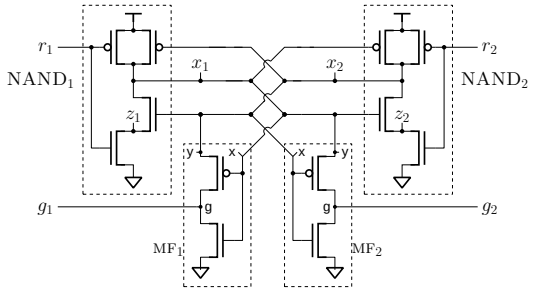


Figure 12: A two-inputs arbiter

An arbiter is a circuit that provides mutually exclusive access to a resource for some set of clients. We consider an asynchronous arbiter with two clients as shown in figure 12. Figure 13 gives a specification for the two-input arbiter. The two clients interact with the arbiter using a four-phase handshake protocol: client i raises r_i to request the privilege; the arbiter raises g_i to grant client i the privilege; when the client is done with the privilege, it lowers r_i; and finally the arbiter lowers g_i to complete the handshake. The arbiter must guarantee mutual exclusion; in other words, signals g_1 and g_2 may not both be high at the same time. Obviously, it would be desirable if the arbiter were guaranteed to eventually issue a grant when a request is pending. It is well-known that a real arbiter cannot satisfy this requirement along with the safety requirements described above (see, for example, [MS92]). Thus, we do not give a liveness requirement for traces with contested requests.

The arbiter shown in figure 12 is implemented based on a SR-latch using a pair of cross-coupled NAND gates (see [Mar89, Fig. 5]). If r_1 and r_2 are asserted at roughly the same time, then signals x_1 and x_2 will both start to fall. This results in a falling input for each NAND-gate and the consequent possibility of metastability if the two NAND-gates reach a balance point with their outputs at an intermediate

Initially:	$\forall i \in \{1,2\}.\ \neg r_i \wedge \neg g_i$
Assume:	$\forall i \in \{1,2\}.\ (\Box r_i U\, g_i) \wedge (\Box \neg r_i U \neg g_i)$
Guarantee:	
Handshake:	$\forall i \in \{1,2\}.\ (\Box \neg g_i U\, r_i) \wedge (\Box g_i U \neg r_i)$
Mutual Exclusion:	$\Box \neg (g_1 \wedge g_2)$
Liveness:	$(\Box (r_1 \oplus r_2) \Rightarrow \Diamond (g_1 \oplus g_2) \vee (r_1 \wedge r_2))$ $\wedge (\Box \neg r_i \Rightarrow \Diamond \neg g_i)$
Note:	because metastability is unavoidable, no arbiter can guarantee $\Box (r_1 \wedge r_2) \Rightarrow \Diamond (g_1 \vee g_2)$.

Figure 13: Discrete specification

level between the power supply voltage and ground. This condition can persist for an arbitrarily long time [CM73, Sei79, Mar81]. The metastability filters MF_1, MF_2 prevent the outputs of the arbiter from changing until metastability resolves. Unlike a traditional inverter, the source of the PMOS pull-up of MF_1 is connected to x_2. With this configuration, the pull-up transistor remains in cut-off until x_1 is at least the PMOS threshold voltage below x_2. This is to prevent g_1 from moving any significant amount above ground until client 1 has clearly won the arbitration.

To compute the all possible reachable space, we sub-divided the Brocket annulus for each request signal into sixteen disjoint regions: one region for logical low values; one for a logical high values; seven for rising transitions; and seven for falling signals. We modeled the concurrent behaviors of the two request signals using cross-product of these partitions. By exploiting the symmetry of the arbiter and its clients, we reduce the number of reachability problems from 256 to 136, and further down to 108 by noting that there must be a failure of the arbiter or its clients if both requests are falling at the same time – this would imply that either the arbiter had violated the mutual-exclusion requirement or that at least one client had violated the handshake protocol. These 108 reachability problems are divided into three phases: the first phase starts with both requests in region B_1 and ends with at least one in B_3. The second phase starts with at least one request in region B_3 and ends with that request back in B_1. The final phase starts with one request having just entered region B_1 and the other in region B_3, and ends when the second request returns to region B_1. The assume-guarantee approach is used similarly for each phase to establish an invariant set that are used to verify the arbiter.

The mutual exclusion and handshake properties as shown in figure 13 are verified using the computed circuit states. We also verified some liveness properties such as fairness, contested requests, and more in [YG08]. Figure 14 and figure 15 show the Brockett annulus for x_1 and g_1. The bulge in the lower right part of

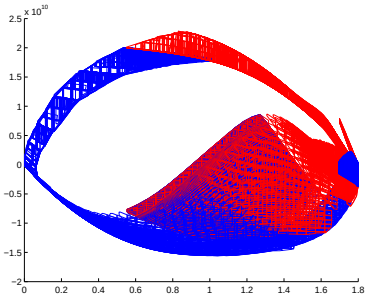


Figure 14: Brockett Annulus for x

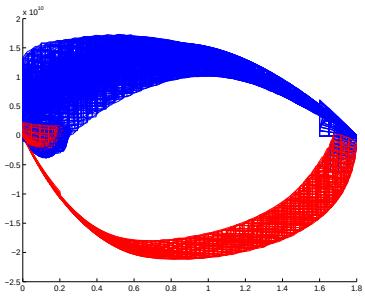


Figure 15: Brockett Annulus for g

the annulus in figure 14 is a consequence of metastability in the arbiter: x_1 may drop to a value near the metastable voltage and remain their for an arbitrarily long time. Comparing the Brockett annuli for x_1 and g_1, we clearly see the role of the metastability filter as a Brockett annulus transformer.

However, we encountered problems with stiffness when attempting to verify this property. Let $\dot{x} = f(x)$ be an ODE, and let $Jac_f(x)$ be the Jacobian matrix for f at x. The ODE is stiff at point x if $\|\lambda_{\max}/\lambda_{\min}\|$ is large where $\lambda_{\max}$ is the largest magnitude eigenvalue of $Jac_f(x)$, and $\lambda_{\min}$ is the smallest magnitude eigenvalue. From a circuit's perspective, stiffness of the ODE model indicates that there is a large ratio between the settling times of the fastest settling node in the circuit and the slowest settling node. This is exactly what happens by the introduction the z_i nodes. Therefore, large time steps result in large linearization error for the z_i nodes, and small time steps lead to large projection and simplification error at the end of each for many timesteps. The stiffness problem makes COHO fail to show that the g_i nodes satisfy the same Brockett annulus used for inputs because of large approximation error.

We implemented two methods to solve this problem. The first one is to ignore the capacitance of z_i nodes. With this assumption the voltage on these nodes is always exactly the value that balances the currents flowing through the upper and lower n-channel transistors, and we model each pair of transistors as a single four-terminal devices. This simplification reduces the ODE model from six dimension down to four and allows us to ignore the stiffness problem of the full ODE model. The second solution involves two main steps to control the over approximation. First, we replaced variables z_1 and z_2 in the ODE with u_1 and u_2 where u_i is the difference between z_i and the equilibrium value for z_i. The value of u_i is quite close

to zero most of the time because z_i nodes settle quickly. Secondly, we added a set of externally verified invariants to reduce the approximation error. With either of these two methods, it is showed that there is no metastability behavior of the output signal.

5.4 Rambus Ring Oscillator

Figure 16 shows a differential ring oscillator. The oscillator consists of an even number of stages, n; and each stage has two forward (labeled fwd in the figure) inverters connected by a pair of cross-coupling (labeled cc) inverters. If the forward inverters are much larger than the cross-coupling inverters, then the circuit acts like a ring of $2n$ inverters and will settle to one of two states:

$$\begin{array}{ll} \text{State 1:} & \mathsf{x(0,0)} \text{ and } \mathsf{x(1,0)}, \text{ are high, and } \mathsf{x(0,1)} \text{ and } \mathsf{x(1,1)} \text{ are low.} \\ \text{State 2:} & \mathsf{x(0,0)} \text{ and } \mathsf{x(1,0)}, \text{ are low, and } \mathsf{x(0,1)} \text{ and } \mathsf{x(1,1)} \text{ are high.} \end{array} \tag{10}$$

Conversely, if the cross-coupling inverters are much larger than the forward ones, then the circuit acts like two separate static latches and has four stable states. If the forward and cross-coupling inverters have comparable strength, then the circuit should oscillate in a stable fashion.

The circuit was proposed by researchers at Rambus [JKK08] as a verification challenge, and they noted that some implementations of the circuit had failed in real, fabricated chips. Therefore, they posed the problem of showing that the oscillator starts from all initial conditions for a particular choice of transistor sizes. They posed a further problem of determining the range of transistor sizes for which proper start-up is guaranteed. Although [GY08] establishes the condition to ensure that the oscillator is free from lock-up, it remains a problem to show that there is only one stable oscillatory mode. We solved these problems for a two-stage Ramubs oscillator as shown in figure 16. In the circuit, all inverters have the same size and signal nodes are denoted as x_1, x_2, x_3, x_4. With the circuit states computed by COHO, we have shown that the two stage oscillator with any possible initial condition oscillates in the specified mode with probability one.

Our verification proceeds in three main phases:

1. The oscillator shown in Figure 16 is a differential design: nodes X_1 and X_3 form a "differential pair" and likewise for nodes X_2 and X_3. The first phase of the verification shows that each of these differential pairs can be treated as a single signal. This symmetry reduction of the state space simplifies the subsequent analysis.

2. Any oscillator must have an equilibrium point with an associated stable manifold. Any starting state on this manifold leads to a non-oscillating behaviour.

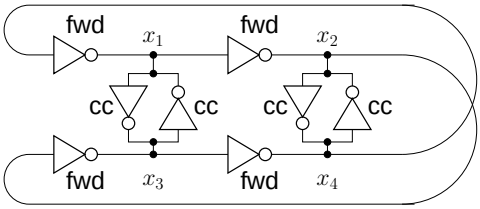


Figure 16: The Rambus oscillator

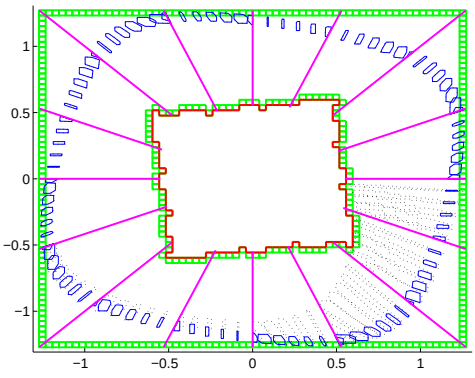


Figure 17: Computing the invariant set

The second phase of the verification shows that this occurs with probability zero.

3. The first two phases show that most initial conditions lead to a fairly small subset of the full phase space. In the final phase, we divide the remaining space into small regions, and use existing reachability methods to show that the oscillator starts up properly from each such region.

This verification phase starts by changing the coordinate system to one based on the differential and common mode representation of signals. Let u be the circuit state in differential coordinates:

$$\begin{aligned} u &= M^{-1}x \\ M &= \frac{\sqrt{2}}{2}\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \end{aligned} \tag{11}$$

To establish differential operation, we divide the range of each of the u variables into n intervals, creating n^4 cubes. We construct a graph, $G = (V,E)$ to represent the reachability relationship between these cubes. Let $v_{i,j,k,\ell}$ be a vertex corresponding to the i^{th} interval for u_1, the j^{th} interval for u_2 and so on. There is an edge from v to w if f allows a flow out of the cube for v directly into the cube for w, and there is a self-loop for v if each component of f is zero somewhere in v. The key idea is that if vertex G has no incoming edges, then any trajectory that starts in the corresponding cube will eventually leave that cube, and no trajectories will ever enter the cube. Such a cube can be eliminated from further consideration. Thus, we

only need to consider cubes whose vertices are members of cycles. These vertices can be identified in $O(V+E) = O(n^4)$ time.

However, the cube that contain the metastable point prevents COHO from finding the oscillatory trajectory because the circuit can stay there for an arbitrary long time. We generalized the conclusion in [Mit96] and found a sufficient condition based on the Jacobean matrix for ensuring all trajectories leave a cube with probability one. Therefore, it is not necessary to solve reachability problem for cubes that satisfy the sufficient condition. At the end of this phase, the number of cubes to consider for the final reachaility analysis has been reduced to a small fraction of the original.

Noting that the common mode voltages u_3 and u_4 are restricted to a small region as shown in Figure 18, we eliminate these two variables by replacing the differential equation model for the circuit with a differential inclusion. This reduces the state space from four dimensions to two which enables efficient reachability computation. Figure 19 shows the region that remains to be verified. We divide this region into its inner and outer boundaries, and a collection of "spokes" as shown in Figure 17. The computation has three parts:

1. Starting from each "spoke", show that all trajectories starting at that spoke eventually cross then next spoke.

2. Show that all trajectories starting from the inner or outer boundary eventually cross the next spoke.

3. Starting from one spoke, compute the reachable set until it converges to a limit set.

The first two show that all trajectories converge to the same attractor. This means that all initial conditions lead to a unique mode of oscillation. The final step tightens the bound on this unique mode.

With this method, we verified that the oscillator starts up properly with probability one for all initial conditions if $0.875 \leq r \leq 2$ where r is the ratio of the cross-coupling inverter size to the forward inverter size.

6 Research Plan and Time Line

The work that we have done to date has largely validated the claim that it is possible to veify non-trivial properties of circuits using projection based verification. To complete the thesis, I plan to

1. clean up and document the code such that it can be released to public research community.

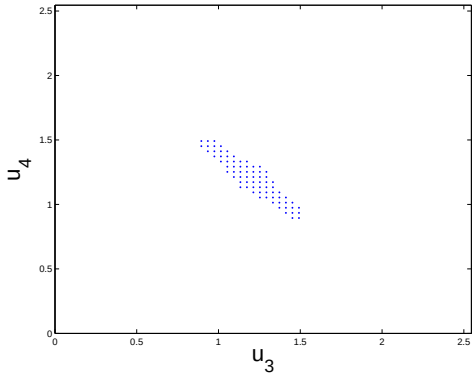


Figure 18: Common-mode convergence to $V_{dd}\sqrt{2}/2$

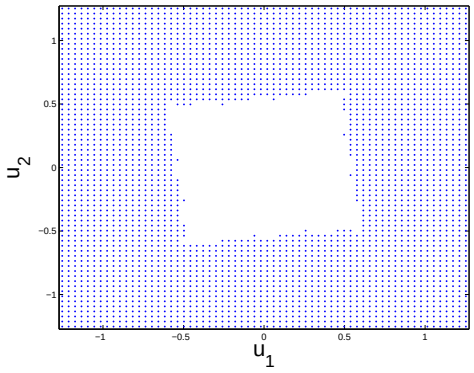


Figure 19: Eliminating the unstable equilibrium

2. make some generalization to the Rambus oscillator verification to other ring oscillators.

If time permits, I plan to

1. verify one more truly analog problem, for example a phase interpolator or a AMS circuit.
2. apply COHO to a hybrid system example and compare COHO with other tools described in section 2.
3. use parallel computation to speedup the computation of COHO.
4. find a general solution for the stiffness problem and apply it to other circuits.

The timeline until graduation is

Date	Milestones and Activities
Apr 30, 2010	Thesis proposal defended.
May - Jun, 2010	Release COHO
Jul - Aug, 2010	Verify other AMS circuits or hybrid system examples
May - Aug, 2010	Thesis writing.
Nov 1, 2010	Thesis defended

Table 5: Thesis Milestones and Activities

References

[ABB+00] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim Guldstrand Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. UPPAAL - now, next, and future. In Franck Cassez, Claude Jard, Brigitte Rozoy, and Mark Dermot Ryan, editors, MOVEP, volume 2067 of Lecture Notes in Computer Science, pages 99–124. Springer, 2000.

[ACD90] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In LICS, pages 414–425. IEEE Computer Society, 1990.

[ACH+95] Rajeev Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138(1):3 – 34, 1995. Hybrid Systems.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. Theoretical Computer Science, 126(2):183 – 235, 1994.

[ADF+06] Eugene Asarin, Thao Dang, Goran Frehse, Antoine Girard, Colas Le Guernic, and Oded Maler. Recent progress in continuous and hybrid reachability analysis. In In Proc. IEEE International Symposium on Computer-Aided Control Systems Design. IEEE Computer. Society Press, 2006.

[ADG03] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In In Oded Maler and Amir Pnueli, editors, Hybrid Systems: Computation and Control, LNCS 2623, pages 20–35. Springer-Verlag, 2003.

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. J. ACM, 43(1):116–146, 1996.

[AH97] Rajeev Alur and Thomas A. Henzinger. Modularity for timed and hybrid systems. In Antoni W. Mazurkiewicz and Józef Winkowski, editors, CONCUR, volume 1243 of Lecture Notes in Computer Science, pages 74–88. Springer, 1997.

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering, 22:181–201, 1996.

[BDL04] Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on Uppaal. In Marco Bernardo and Flavio Corradini, editors, SFM, volume 3185 of Lecture Notes in Computer Science, pages 200–236. Springer, 2004.

[BGG+09] Erich Barke, Darius Grabowski, Helmut Graeb, Lars Hedrich, Stefan Heinen, Ralf Popp, Sebastian Steinhorst, and Yifan Wang. Formal approaches to analog circuit verification. In DATE, pages 724–729. IEEE, 2009.

[BLL+96] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Uppaal—a tool suite for automatic verification of real-time systems. In Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control, pages 232–243, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.

[BMP99] Olivier Bournez, Oded Maler, and Amir Pnueli. Orthogonal polyhedra: Representation and computation. In Schuppen (Eds.), Hybrid Systems: Computation and Control, LNCS 1569, pages 46–60. Springer, 1999.

[BMPY97] Marius Bozga, Oded Maler, Amir Pnueli, and Sergio Yovine. Some progress in the symbolic verification of timed automata. In Orna Grumberg, editor, CAV, volume 1254 of Lecture Notes in Computer Science, pages 179–190. Springer, 1997.

[Bro89] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In Hendrik Nijmeijer and Johannes M. Schumacher, editors, Three Decades of Mathematical Systems Theory: A Collection of Surveys at the Occasion of the 50th Birthday of J. C. Willems, volume 135 of Lecture Notes in Control and Information Sciences, pages 19–30. sv, 1989.

[BRZH02] Roberto Bagnara, Elisa Ricci, Enea Zaffanella, and Patricia M. Hill. Possibly not closed convex polyhedra and the parma polyhedra library. In SAS '02: Proceedings of the 9th International Symposium on Static Analysis, pages 213–229, London, UK, 2002. Springer-Verlag.

[BT00] Oleg Botchkarev and Stavros Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In HSCC '00: Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control, pages 73–88, London, UK, 2000. Springer-Verlag.

[BTY97] Ahmed Bouajjani, Stavros Tripakis, and Sergio Yovine. On-the-fly symbolic model checking for real-time systems. In IEEE Real-Time Systems Symposium, pages 25–. IEEE Computer Society, 1997.

[Che68] N. V. Chernikov. Algorithms for discovering the set of all solutions of a linear programming problem. Computational Mathematics and Mathematical Physics, pages 283–293, 1968.

[Chu99] Alongkrit Chutinan. Hybrid System Verification Using Discrete Model Approximations. PhD thesis, Carnegie Mellon University, 1999.

[CJ08] Alessandro Cimatti and Robert B. Jones, editors. Formal Methods in Computer-Aided Design, FMCAD 2008, Portland, Oregon, USA, 17-20 November 2008. IEEE, 2008.

[CK98] Alongkrit Chutinan and Bruce H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In In Proceedings of the 37rd IEEE Conference on Decision and Control. IEEE Press, 1998.

[CK99a] Alongkrit Chutinan and Bruce H. Krogh. Computing approximating automata for a class of linear hybrid systems. In Hybrid Systems V, pages 16–37, London, UK, 1999. Springer-Verlag.

[CK99b] Alongkrit Chutinan and Bruce H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In HSCC '99: Proceedings of the Second International Workshop on Hybrid Systems, pages 76–90, London, UK, 1999. Springer-Verlag.

[CK00] Alongkrit Chutinan and Bruce H. Krogh. Computing approximating automata for a class of hybrid systems. In Mathematical Modeling of Systems: Special Issue on Discrete Event Models of Continuous Systems, submitted, March 2000.

[CK01] A. Chutinan and B. H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. Automatic Control, IEEE Transactions on, 46(9):1401–1410, September 2001.

[CK03] A. Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. Automatic Control, IEEE Transactions on, 48(1):64–75, January 2003.

[CM73] T. J. Chaney and C. E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. IEEETC, C-22(4):421–422, April 1973.

[Dan00] Thao Dang. Verification and Synthesis of Hybrid Systems. PhD thesis, Institut National Polytechnique de Grenoble, 2000.

[DC05] Tathagato Rai Dastidar and P. P. Chakrabarti. A verification system for transient response of analog circuits using model checking. In VLSI Design, pages 195–200. IEEE Computer Society, 2005.

[DDM04] Thao Dang, Alexandre Donzé, and Oded Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In Alan J. Hu and Andrew K. Martin, editors, FMCAD, volume 3312 of Lecture Notes in Computer Science, pages 21–36. Springer, 2004.

[DM98] Thao Dang and Oded Maler. Reachability analysis via face lifting. In HSCC '98: Proceedings of the First International Workshop on Hybrid Systems, pages 96–109, London, UK, 1998. Springer-Verlag.

[DM06] Bruno Dutertre and Leonardo De Moura. The yices smt solver. Technical report, Computer Science Laboratory, SRI International, 2006.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool kronos. In Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control, pages 208–219, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.

[EFH08] Andreas Eggers, Martin Fränzle, and Christian Herde. SAT modulo ODE: A direct SAT approach to hybrid systems. In Sung Deok Cha, Jin-Young Choi, Moonzoo Kim, Insup Lee, and Mahesh Viswanathan, editors, ATVA, volume 5311 of Lecture Notes in Computer Science, pages 171–185. Springer, 2008.

[EMSS92] E. Allen Emerson, Aloysius K. Mok, A. Prasad Sistla, and Jai Srinivasan. Quantitative temporal reasoning. Real-Time Systems, 4(4):331–352, 1992.

[FH07] Martin Fränzle and Christian Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. Formal Methods in System Design, 30(3):179–198, 2007.

[FHK04] Goran Frehse, Zhi Han, and Bruce H. Krogh. Assume-guarantee reasoning for hybrid i/o-automata by over-approximation of continuous interaction. In Decision and Control, 2004. CDC. 43rd IEEE Conference on, volume 1, pages 479–484 Vol.1, December 2004.

[FHT+07] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. Journal on Satisfiability, Boolean Modeling and Computation, 1:209–236, 2007.

[FKR06] Goran Frehse, Bruce H. Krogh, and Rob A. Rutenbar. Verifying analog oscillator circuits using forward/backward abstraction refinement. In DATE '06: Proceedings of the conference on Design, automation and test in Europe, pages 257–262, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[FMW05] Harry Foster, Erich Marschner, and Yaron Wolfsthal. IEEE 1850 PSL: The next generation, 2005.

[Fre04] Goran Frehse. Compositional verification of hybrid systems with discrete interaction using simulation relations. In Computer Aided Control Systems Design, 2004 IEEE International Symposium on, pages 59–64, September 2004.

[Fre05] Goran Frehse. Compositional Verification of Hybrid Systems Using Simulation Relations. PhD thesis, Radboud Universiteit Nijmegen, October 2005.

[Fre08] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. Int. J. Softw. Tools Technol. Transf., 10(3):263–279, 2008.

[GG08] Antoine Girard and Colas Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In HSCC '08: Proceedings of the 11th international workshop on Hybrid Systems, pages 215–228, Berlin, Heidelberg, 2008. Springer-Verlag.

[GGM06] Antoine Girard, Colas Guernic, Le, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In João P. Hespanha and Ashish Tiwari, editors, HSCC, volume 3927 of Lecture Notes in Computer Science, pages 257–271. Springer Berlin / Heidelberg, 2006.

[Gir05] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In Manfred Morari and Lothar Thiele, editors, HSCC, volume 3414 of Lecture Notes in Computer Science, pages 291–305. Springer Berlin / Heidelberg, 2005.

[GKR04] Smriti Gupta, Bruce H. Krogh, and Rob A. Rutenbar. Towards formal verification of analog designs. In Proceedings of 2004 IEEE/ACM International Conference on Computer Aided Design, pages 210–217, November 2004.

[GL91] Orna Grumberg and David E. Long. Model checking and modular verification. In Jos C. M. Baeten and Jan Friso Groote, editors, CONCUR, volume 527 of Lecture Notes in Computer Science, pages 250–265. Springer, 1991.

[GPHB06] Darius Grabowski, Daniel Platte, Lars Hedrich, and Erich Barke. Time constrained verification of analog circuits using model-checking algorithms. Electr. Notes Theor. Comput. Sci., 153(3):37–52, 2006.

[Gre96] Mark R Greenstreet. Verifying safety properties of differential equations. In Proceedings of the 1996 Conference on Computer Aided Verification, pages 277–287, New Brunswick, NJ, July 1996.

[GY08] Mark R Greenstreet and Suwen Yang. Verifying start-up conditions for a ring oscillator. In GLSVLSI '08: Proceedings of the 18th ACM Great Lakes symposium on VLSI, pages 201–206, New York, NY, USA, 2008. ACM.

[Hal93] Nicolas Halbwachs. Delay analysis in synchronous programs. In CAV '93: Proceedings of the 5th International Conference on Computer Aided Verification, pages 333–346, London, UK, 1993. Springer-Verlag.

[HC97] S. Hendriex and L. Claesen. A symbolic core approach to the formal verification of integrated mixed-mode applications. European Design and Test Conference, 0:432, 1997.

[HEFT08] Christian Herde, Andreas Eggers, Martin Fränzle, and Tino Teige. Analysis of hybrid systems using HySAT. In ICONS '08: Proceedings of the Third International Conference on Systems, pages 196–201, Washington, DC, USA, 2008. IEEE Computer Society.

[Hen95] Thomas A. Henzinger. Hybrid automata with finite bisimulatioins. In ICALP '95: Proceedings of the 22nd International Colloquium on Automata, Languages and Programming, pages 324–335, London, UK, 1995. Springer-Verlag.

[Hen00] T. A. Henzinger. The theory of hybrid automata. In Verification of Digital and Hybrid Systems, volume Vol.170, pages 265–292. Springer, 2000.

[Her09] Christian Herde. HySat User Guide, November 2009.

[HH95a] Thomas A. Henzinger and Pei-Hsin Ho. HyTech: The Cornell HYbrid TECHnology tool. In Hybrid Systems II, LNCS 999, pages 265–293. Springer-Verlag, 1995.

[HH95b] Thomas A. Henzinger and Pei-Hsin Ho. A note on abstract-interpretation strategies for hybrid automata. In Hybrid Systems II, volume 999 of LNCS, pages 252–264. Springer-Verlag, 1995.

[HHB02a] Walter Hartong, Lars Hedrich, and Erich Barke. Model checking algorithms for analog verification. In DAC '02: Proceedings of the 39th annual Design Automation Conference, pages 542–547, New York, NY, USA, 2002. ACM.

[HHB02b] Walter Hartong, Lars Hedrich, and Erich Barke. On discrete modeling and model checking for nonlinear analog systems. In CAV '02: Proceedings of the 14th International Conference on Computer Aided Verification, pages 401–413, London, UK, 2002. Springer-Verlag.

[HHK96] R. H. Hardin, Z. Har'EI, and R. P. Kurshan. Cospan. Lecture Notes in Computer Science, Computer Aided Verification, 1102/1996:423–427, 1996.

[HHMWt00] Thomas A. Henzinger, Benjamin Horowitz, Rupak Majumdar, and Howard Wong-toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In in HSCC, pages 130–144. Springer, 2000.

[HHWt95a] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. HyTech: The next generation. In In Proceedings of the 16th IEEE Real-Time Systems Symposium, pages 56–65. IEEE Computer Society press, 1995.

[HHWt95b] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. A user guide to HyTech, 1995.

[HHWt96] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. Algorithmic analysis of nonlinear hybrid systems. IEEE Transactions on Automatic Control, 43:225–238, 1996.

[HHWt97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-toi. HyTech: A model checker for hybrid systems. Software Tools for Technology Transfer, 1:460–463, 1997.

[HKPV95] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In Journal of Computer and System Sciences, pages 373–382. ACM Press, 1995.

[HKV96] Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In Ugo Montanari and Vladimiro Sassone, editors, CONCUR, volume 1119 of Lecture Notes in Computer Science, pages 514–529. Springer, 1996.

[HM00] Thomas A. Henzinger and Rupak Majumdar. Symbolic model checking for rectangular hybrid systems. In TACAS '00: Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems, pages 142–156, London, UK, 2000. Springer-Verlag.

[HMP01] Thomas A. Henzinger, Marius Minea, and Vinayak S. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, HSCC, volume 2034 of Lecture Notes in Computer Science, pages 275–290. Springer, 2001.

[HNSY94] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. Information and Computation, 111(2):193–244, 1994.

[Ho95] Pei-Hsin Ho. Automatic Analysis of Hybrid Systems. PhD thesis, Cornell University, Ithaca, NY, USA, August 1995.

[HQR98] Thomas A. Henzinger, Shaz Qadeer, and Sriram K. Rajamani. You assume, we guarantee: Methodology and case studies. In Alan J. Hu and Moshe Y. Vardi, editors, CAV, volume 1427 of Lecture Notes in Computer Science, pages 440–451. Springer, 1998.

[HRP94] Nicolas Halbwachs, Pascal Raymond, and Yann-eric Proy. Verification of linear hybrid systems by means of convex approximations. In SAS, pages 223–237. Springer-Verlag, 1994.

[HW04] Timothy J. Hickey and David K. Wittenberg. Rigorous modeling of hybrid systems using interval arithmetic constraints. In Rajeev Alur and George J. Pappas, editors, HSCC, volume 2993 of Lecture Notes in Computer Science, pages 402–416. Springer, 2004.

[HWt95] Pei-Hsin Ho and Howard Wong-toi. Automated analysis of an audio control protocol. In Proceedings of the 7th International Conference on Computer Aided Verification, pages 381–394, London, UK, 1995. Springer-Verlag.

[HWt96] T. A. Henzinger and Howard Wong-toi. Using HyTech to synthesize control parameters for a steam boiler. In In Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control, LNCS 1165, pages 265–282. Springer-Verlag, 1996.

[Jes08] Alexander Jesser. Mixed-Signal Circuit Verification Using Symbolic Model Checking Techniques. PhD thesis, University of Frankfurt a.M., Germany, ISBN 978-3-89963-841-7, October 2008.

[JKK08] Kevin D. Jones, Jaeha Kim, and Victor Konrad. Some “real world” problems in the analog and mixed-signal domains. In Proc. Workshop on Designing Correct Circuits, April 2008.

[KK02] J. Kapinski and B. H. Krogh. Verifying switched-mode computer controlled systems. IEEE Conference on Computer-Aided Control System Design, pages 98–103, September 2002.

[KM91] R. P. Kurshan and K. L. McMillan. Analysis of digital circuits through symbolic reduction. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 10(11):1356–1371, November 1991.

[KV96] Alexander B. Kurzhanski and Istvan Valyi. Ellipsoidal Calculus for Estimation and Control. Birkhäuser Boston, 1 edition edition, September 1996.

[KV00a] Alexander B. Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In HSCC ’00: Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control, pages 202–214, London, UK, 2000. Springer-Verlag.

[KV00b] Alexander B. Kurzhanski and Pravin Varaiya. On ellipsoidal techniques for reachability analysis. Optimization Methods and Software, 17:177–237, 2000.

[KV01] Alexander B. Kurzhanski and Pravin Varaiya. Reachability analysis for uncertain systems - the ellipsoidal technique. Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms, 9(347-367), 2001.

[KV06] Alex A. Kurzhanskiy and Pravin Varaiya. Ellipsoidal toolbox. Technical Report UCB/EECS-2006-46, EECS Department, University of California, Berkeley, May 2006.

[KV07] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal techniques for reachability analysis of discrete-time linear systems. Automatic Control, IEEE Transactions on, 52(1):26–38, Jan. 2007.

[Laz01] Marius Laza. A robust linear program solver for projectahedra. Master's thesis, Univerisity of British Columbia, 2001.

[Lit08] Scott Little. Efficient Modeling and Verification of Analog/Mixed-Signal Circuits Using Labeled Hybrid Petri Nets. PhD thesis, University of Utah, 2008.

[LPS99] Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. Hybrid systems with finite bisimulations. In Hybrid Systems V, pages 186–203, London, UK, 1999. Springer-Verlag.

[LPY95] K. G. Larsen, P. Pettersson, and Wang Yi. Compositional and symbolic model-checking of real-time systems. In RTSS '95: Proceedings of the 16th IEEE Real-Time Systems Symposium, page 76, Washington, DC, USA, 1995. IEEE Computer Society.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. STTT, 1(1-2):134–152, 1997.

[LPY98] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Decidable hybrid systems. Technical report, Department of Mathematical Science, Portland State University, Portland, OR, 1998.

[LSW+06] Scott Little, Nicholas Seegmiller, David Walter, Chris Myers, and Tomohiro Yoneda. Verification of analog/mixed-signal circuits using labeled hybrid Petri nets. In ICCAD '06: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design, pages 275–282, New York, NY, USA, 2006. ACM.

[LV92] H. Le Verge. A note on chernikov's algorithm. Technical report, IRISA, 1992.

[LW04] Scott Little and David Walter. Verification of analog and mixed-signal circuits using timed hybrid Petri nets. Automated Technology for Verification and Analysis, 3299 of LNCS:426–440, November 2004.

[LWJM07] Scott Little, David Walter, Kevin Jones, and Chris Myers. Analog/mixed-signal circuit verification using models generated from simulation traces. Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, 4762:114–128, 2007. Springer, Berlin.

[Lyg03] John Lygeros. Lecture notes on hybrid systems, January 2003. Department of Engineering, University of Cambridge.

[Mar81] L. R. Marino. General theory of metastable operation. IEEETC, C-30(2):107–115, February 1981.

[Mar89] Alain J. Martin. Programming in vlsi: From communicating processes to delay insensitive circuits. In C. A. R. Hoare, editor, University of Texas Year of Programming Institute on Concurrent Programming. Addison-Wesley, 1989.

[Mit96] Ian Mitchell. Proving newtonian arbiters correct, almost surely. Master's thesis, The University of British Columbia, 1996.

[MP05] Oded Maler and Amir Pnueli. Extending PSL for analog circuits. Technical report, PROSYD: Property-Based System Design, 2005.

[MS92] Michael Mendler and Terry Stroup. Newtonian arbiters cannot be proven correct. In Proceedings of the 1992 Workshop on Designing Correct Circuits, January 1992.

[MT00] Ian Mitchell and Claire Tomlin. Level set methods for computation in hybrid systems. In HSCC '00: Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control, pages 310–323, London, UK, 2000. Springer-Verlag.

[MY96] O. Maler and S. Yovine. Hardware timing verification using KRONOS. Israeli Conference on Computer-Based Systems and Software Engineering, 0:23, 1996.

[NM07] Dejan Nickovic and Oded Maler. Amt: A property-based monitoring tool for analog systems. In Jean-françois Raskin and P. S. Thiagarajan, editors, FORMATS, volume 4763 of Lecture Notes in Computer Science, pages 304–319. Springer, 2007.

[Pet96] W. Kopke Peter. The Theory of Rectangular Hybrid Automata. PhD thesis, Cornell University, August 1996.

[PKWtH98] Joerg Preussig, Stephan Kowalewski, Howard Wong-toi, and T. A. Henzinger. An algorithm for the approximative analysis of rectangular automata. In Anders P. Ravn and Hans Rischel, editors, FTRTFT, volume 1486 of Lecture Notes in Computer Science 1486, pages 228–240. Springer, 1998.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice Hall, Englewood Cliffs, NJ, 1982.

[PS85] Franco P. Preparata and Michael I. Shamos. Computational Geometry: An Introduction. Texts and Monographs in Computer Science. Springer, 1985.

[PV94] Anuj Puri and Pravin Varaiya. Decidability of hybrid systems with rectangular differential inclusion. In CAV '94: Proceedings of the 6th International Conference on Computer Aided Verification, pages 95–104, London, UK, 1994. Springer-Verlag.

[RMC08] Nacim Ramdani, Nacim Meslem, and Yves Candau. Reachability of uncertain nonlinear systems using a nonlinear hybridization. In Magnus Egerstedt and Bud Mishra, editors, HSCC, volume 4981 of Lecture Notes in Computer Science, pages 415–428. Springer, 2008.

[SBB03] Sanjit A. Seshia, Randal E. Bryant, and Al E. Bryant. Unbounded, fully symbolic model checking of timed automata using boolean methods. In In Proc. of CAV03. LNCS 2725. Springer-Verlag, 2003.

[Sei79] Charles L. Seitz. System timing. In Introduction to VLSI Systems (Carver Mead and Lynn Conway), chapter 7, pages 218–262. Addison Wesley, 1979.

[SH08] Sebastian Steinhorst and Lars Hedrich. Model checking of analog systems using an analog specification language. In DATE '08: Proceedings of the conference on Design, automation and test in Europe, pages 324–329, New York, NY, USA, 2008. ACM.

[SK99] O. Stursberg and S. Kowalewski. Approximating switched continuous systems by rectangular automata. In In Proc. ECC'99, 1999.

[SK00a] B. I. Silva and Bruce H. Krogh. Formal verification of hybrid system using CheckMate: A case study. In American Control Conference, volume 3, pages 1679–1683, June 2000.

[SK00b] B. I. Silva and Bruce H. Krogh. Modeling and verification of sampled-data hybrid systems. ADPM, September 2000.

[SK01] B. I. Silva and Bruce H. Krogh. Modeling and verification of hybrid system with clocked and unclocked events. 40th Conference on Decision and Control, December 2001.

[SK03] Olaf Stursberg and Bruce H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In In HSCC2003, LNCS 2289, pages 482–497. Springer, 2003.

[SRKC00] B. I. Silva, K. Richeson, Bruce Krogh, and Alongkrit Chutinan. Modeling and verifying hybrid dynamical systems using CheckMate. In Proceedings of the 4^{th} International Conference on Automation of Mixed Processes (ADPM 2000), pages 323–328, September 2000.

[SS95] Oleg Sokolsky and Scott A. Smolka. Local model checking for real-time systems (extended abstract). In Pierre Wolper, editor, CAV, volume 939 of Lecture Notes in Computer Science, pages 211–224. Springer, 1995.

[SSKE01] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. 40th Conference on Decision and Control, December 2001.

[Sta97] O. Stauning. Automatic Validation of Numerical Solutions. PhD thesis, Danmarks Tekniske Universitet, Kgs., Lyngby, Denmark, 1997.

[SZDT07] Ghiath Al Sammane, Mohamed H. Zaki, Zhi Jie Dong, and Sofiène Tahar. Towards assertion based verification of analog and mixed signal designs using PSL. In FDL, pages 293–298. ECSI, 2007.

[TMB+03] Claire J. Tomlin, Ian Mitchell, Alexandre M. Bayen, Re M. Bayen, and Meeko Oishi. Computational techniques for the verification and control of hybrid systems. In Proceedings of the IEEE, pages 986–1001, 2003.

[Var98] Pravin Varaiya. Reach set computation using optimal control. In Proc. KIT Workshop, pages 377–383, 1998.

[WLM07] David Walter, Scott Little, and Chris Myers. Bounded model checking of analog and mixed-signal circuits using an SMT solver. Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, 4762:66–81, 2007. Spring, Berlin.

[WLS+07] D. Walter, S. Little, N. Seegmiller, C. J. Myers, and T. Yoneda. Symbolic model checking of analog/mixed-signal circuits. In ASP-DAC '07: Proceedings of the 2007 Asia and South Pacific Design Automation Conference, pages 316–323, Washington, DC, USA, 2007. IEEE Computer Society.

[Yan06] Chao Yan. Coho: A verification tool for circuit verification by reachability analysis. Master's thesis, The University of British Columbia, August 2006.

[YG08] Chao Yan and Mark R Greenstreet. Verifying an arbiter circuit. In Cimatti and Jones [CJ08], pages 1–9.

[Yov97] Sergio Yovine. Kronos: A verification tool for real-time systems. International Journal on Software Tools for Technology Transfer, 1:123–133, 1997.

[YS89] J. Yuan and C. Svensson. High-speed cmos circuit technique. Solid-State Circuits, IEEE Journal of, 24(1):62–70, February 1989.

[ZTB08] Mohamed H. Zaki, Sofiène Tahar, and Guy Bois. Formal verification of analog and mixed signal designs: A survey. Microelectronics Journal, 39(12):1395 – 1404, 2008.