Formal Verification of Mixed-Signal Systems Using Real Number Modeling with SMT-based Model Checking

Augusto Mafra, Haniel Barbosa
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
✉{augustomafra, hbarbosa}@dcc.ufmg.br



***Abstract*—Formal verification of digital hardware designs is a well established practice in the semiconductor industry. However, designs combining digital and analog components (mixed-signal designs) remain with limited application of formal verification due to the complexity of automating proofs on the continuous state space of analog systems.**

In this work we perform a case study of formal verification for mixed-signal designs where we instrumentalize Yosys, an industrial-grade SystemVerilog compiler, and Pono, a state-of-the-art word-level model checker, to implement a formal verification flow on a set of 31 realistic mixed-signal benchmarks that we crafted from testbenches for analog blocks using real number modeling. As the practice of abstracting the analog domain with real-valued variables in SystemVerilog is already common place on simulation, we propose an interpretation of such verification problems on the theory of real arithmetic, which enables order of magnitude verification speed-ups using SMT solvers relative to the traditional bit-blasting interpretation on the theory of floating-point arithmetic. Despite the semantic differences between the two theories, our benchmarks indicate that mixed-signal hardware verification problems are not dependent on floating-point arithmetic details in practice, which favors the adoption of real arithmetic solvers. To demonstrate the practicality of this method, we perform a comparative analysis of the results from different theories over our benchmark set and discuss the technical trade-offs that arise when introducing formal verification to the mixed-signal verification domain.

***Index Terms*—Formal Verification, Hardware Verification, Model Checking, Satisfiability Modulo Theories, Mixed-Signal Circuits, Real Number Modeling.**

I. Introduction

Verification of digital hardware designs applies formal methods extensively to ensure strict correctness requirements on custom Integrated Circuits (ICs) in the semiconductor industry. Formal Verification (FV) is a mature and practical methodology used regularly on VLSI (Very-Large-Scale Integration) design, achievable by any engineer familiar with RTL (Register Transfer Level) design concepts. As RTL is inherently compatible with Boolean logic, effective model checking algorithms based on Satisfiability (SAT) solvers and Binary Decision Diagrams (BDD) have been developed to solve industrial-sized problems [1].

However, the increasing trend of designing mixed-signal Systems-on-Chip (SoCs), i.e., systems that combine digital and analog components in a single chip, imposes a challenge for such methodologies, because the common SAT or BDD solvers are not able to reason natively on the continuous domain of analog systems. Verification becomes even harder with the industry trend toward mobile computing and Internet of Things, which requires advanced low-power design techniques that introduce interdependence between analog and digital circuits [2]. As a result, mixed-signal systems can only achieve complete verification if both domains are modeled simultaneously, since digital logic may be driven by the analog blocks and vice-versa.

In the mixed-signal simulation domain, a successful trade-off for reducing run times is Analog Real Number Modeling (RNM), consisting of abstracting analog physical variables by floating-point variables in the RTL representation. Despite not capturing all transient physical behavior with accuracy, RNM describes the analog system as a signal-flow model that can be executed by a digital simulator with significant speedup over traditional analog simulation [2]. RNM designs in SystemVerilog HDL (Hardware Description Language) model analog variables using the `real` data type defined by the IEEE Std 1800™-2012 standard [3], which defines it as the 64-bit floating-point type according to IEEE Std 754™-2019 standard [4].

This work implements a full formal verification flow on SystemVerilog RNM mixed-signal hardware systems. To the best of our knowledge, no other practical FV methodology has been developed before on this domain. We achieve an effective formal verification methodology suitable to SoC scale by encoding SystemVerilog RNM verification problems to an extended version of Pono [5], a state-of-the-art Satisfiability Modulo Theory (SMT) model checker that supports systems on the theories of real arithmetic [6] and has been modified in this work to support floating-point arithmetic [7]. Translation of the input SystemVerilog RNM code is implemented by extending Yosys [8], an industrial-grade Verilog and SystemVerilog synthesis suite.

The core thesis in this work is that floating-point arithmetic, despite being the standard interpretation for SystemVerilog RNM, is not the best encoding for formal verification of mixed-signal systems. Solving in the floating-point theory

https://doi.org/10.34727/2026/isbn.978-3-85448-093-8_53

This article is licensed under a Creative Commons Attribution 4.0 International License

often requires order of magnitude longer run times than in real arithmetic due to the size and complexity of the formulas generated after bit-blasting. Furthermore, floating-point arithmetic has its own semantic differences from real arithmetic, such as non-associativity, rounding modes, and ∞ and `NaN` (not-a-number) values [7]. Even though these differences are critical for encoding machine operations correctly, they are not meaningful for the domain represented by SystemVerilog RNM: physical quantities in the continuous domain of analog circuits.

The contributions of this work are summarized as follows:

- We implement a formal verification flow on mixed-signal SystemVerilog RNM designs leveraging SMT model checking suitable to SoC scale, including an extended version of Yosys to compile SystemVerilog RNM designs to the SMV format [9] and an extension of Pono and its underlying `Smt-Switch` API [10] to support the SMV verification problems on the theories of real and floating-point arithmetic on three SMT solvers: cvc5 [11], MathSAT [12] and Bitwuzla [13].
- We create a set of 31 realistic FV benchmarks of mixed-signal RNM verification problems with meaningful properties covering the specification of the designs.
- We perform a comparative analysis of the semantics and performance of solving mixed-signal FV problems over the theories of real and floating-point arithmetic using different SMT solvers, achieving speed-ups up to 870x on the real arithmetic run time relative to floating-point arithmetic while finding only a single example containing a semantic mismatch between the two theories.

a) Software: From the benchmarks developed in this work, 17 are publicly available under the GPL 3.0 license at [14]; the remaining ones were built on proprietary Cadence® Design Systems models and testbenches, as detailed in Section IV.

The modified version of the tools used in this implementation are public forks of Pono and Yosys, both available at https://github.com/augustomafra/pono[1] and https://github.com/augustomafra/yosys,[2] respectively. Pono also requires a custom version of the `Smt-Switch` API, which is also available as a public fork of the original repository at https://github.com/augustomafra/smt-switch.[3]

A. Related Work

Application of formal verification to Analog and Mixed-Signal (AMS) designs has been topic of systematic review [15], [16], showing that existing work still hits scalability gaps inherent of hybrid systems and is hard to integrate into the existing verification languages and tooling [17]. Most of the literature on the topic dates from the period between 2007 and 2013, correlated with the increasing interest in mixed-signal design due to the trend of mobile computing in industry. However, SMT model checking research has advanced significantly since then, making it possible that challenges that were not approachable on the past decade now can be explored by instrumentalizing current state-of-the-art solvers.

In general, prior mixed-signal verification methods [18]–[21] obtain hard verification problems due to being applied to low-level system representations with nonlinear circuit components and feedback loops. Other approaches have already applied SMT solving to RTL abstraction level [22], [23], but complete SMT model checking was not consolidated by the time they have been published (2007). This work avoids the verification complexity by focusing on higher-level behavioral RNM mixed-signal representation, which is already an established simplification for analog systems in the hardware simulation domain. Even when the RNM model uses nonlinear constraints, they are usually a more abstract representation of the original circuit, which greatly reduces verification complexity.

In the digital RTL verification domain, SMT-based model checkers have already been successfully applied. Irfan et al. [24] designed Verilog2SMV, a translator from SystemVerilog to the nuXmv model checker [9]. Our work also uses the SMV language as a backend, but while we target arithmetic theory solvers, [24] leverages the SMT theory of arrays to verify designs with large memories, a known complexity bottleneck on bit-level checkers. That domain was also a highlight for Pono, the solver adopted in this work, which received the gold medal in the track for word-level with arrays on the 2025 Hardware Model Checking Competition [25], solving 49 instances more than the competing solvers.

More recently, Mohanty and Gadde [26] adapted state-of-the-art digital verification methodologies to work on industrial-grade SystemVerilog RNM designs. Their strategy was to abstract analog variables as digital wires, which does not fully model the semantics of the continuous values but suffices for verifying connectivity between analog and digital components using existing digital verification apps on Cadence Jasper®.

II. Background

A. Mixed-Signal Hardware Simulation

The trend in semiconductor industry for the last two decades has largely focused on mobile communication and networking applications, which has increased the demand for low-power SoCs requiring high-performance analog and mixed-signal designs [2]. This demand introduces the need for large and complex analog components within digital blocks.

Moreover, the analog systems often include their own digital control logic, which introduces logical feedback loops between both domains. As a result, when doing sign-off for a mixed-system SoC, a verification engineer may take advantage of the automated and metric-driven methodologies suited for digital simulation, but the analog verification flow remains as a bottleneck due to the compute-intensive circuit simulation methods such as SPICE [2].

Analog behavioral modeling is a trade-off widely used in industry to improve simulation speed on mixed-signal

[1]Pono git commit 3e1453cfa5d18616b05424c8af00d74aa41b4292.
[2]Yosys git commit a726634e9e9883709f94c5e479ce8558af3da704.
[3]Smt-Switch git commit 5815833ed9b553ef594a8236b1b0ca323fc6c1f5.

hardware. While SPICE allows an accurate modeling for the physical electronic circuits and their dynamic behavior, simulating it requires solving the entire system matrix for each time step. Hardware designers can significantly reduce this cost by developing a sufficiently accurate behavioral model in an RTL language that mimics the output of the modeled analog circuit. A language commonly used for developing such models is SystemVerilog RNM, an extension of the SystemVerilog language with the `real` type, modeling mixed-signal systems with discrete-time semantics.

By leveraging SystemVerilog RNM models in the design process, verification engineers can take advantage of the event-driven signal-flow simulation adopted for digital hardware with performance suitable for verification at the SoC scale. As a trade-off, not all transient behavior from the analog system may be captured by the discrete time model. Furthermore, the verification of the SoC-level integration is often only concerned with the analog values as seen from the digital interfaces. As exemplified by the results in Sapsanis et al. [27], since these values are almost always sampled at the edges of the digital clock, temporal transients would not matter to the final system behavior, making the RNM output effectively an exact representation of the analog system to the point of view of the digital domain.

B. Hardware Formal Verification

Formal verification of digital hardware gained industrial relevance in the past decades following the significant evolution of SAT solvers [1, ch. 2]. These technology advancements allowed solving SAT instances with millions of variables since the 1990s, enabling the application of SAT solvers in industry-scale verification problems.

One important factor of such widespread commercial adoption is that there is a straightforward encoding of digital RTL models, where all variables assume binary values `0` or `1`, into Propositional Logic, the language of the SAT problem, comprising formulas of Boolean variables `true` and `false`.

As discussed in this work, that encoding is not as straightforward on mixed-signal hardware due to the addition of variables ranging over values in some background theory, such as with real or floating-point arithmetic. The introduction of theories in the reasoning is the core of SMT solvers (presented in Section II-C), which motivates the adoption of SMT-based model checkers like Pono to perform formal verification on this kind of design.

C. SMT Solvers

Despite the usefulness and efficiency of SAT solvers, many interesting computer science problems cannot be naturally formulated or cannot be formulated at all as satisfiability problems in propositional logic. These problems are better formulated in a more expressive language including non-Boolean variables, such as first-order logic (FOL). When restricting the interpretation of certain symbols to models of some logical background theory (e.g. the theory of integer arithmetic or real arithmetic), this class of problems is called *Satisfiability Modulo Theories* (SMT) [28].

Research and industry practice in SMT has grown in over 20 years building on top of the development of decision procedures for first-order reasoning and of the technological advancements in SAT solvers [28]. This trend lead to the development of advanced SMT solvers, such as cvc5 [11], MathSAT [12] and Z3 [29], covering problems formulated in different background theories including real arithmetic, bit vectors, arrays and floating-point arithmetic. These solvers are built on top of a general framework over CDCL SAT solvers called CDCL(T) [30], where T stands for *theory*.

1) Real Arithmetic Theory: SMT solvers achieve good performance when solving practical problems in real arithmetic by leveraging multiple decision procedures. Linear Real Arithmetic (LRA) formulas can often be solved efficiently using procedures based on the Simplex method. Nonlinear Real Arithmetic (NRA) formulas, on the other hand, can be solved with iterative procedures such as incremental linearization [31] or algorithms based on Cylindrical Algebraic Decomposition [32]. Other procedures have been proposed for checking satisfiability in the nonlinear arithmetic theory augmented with transcendental functions (i.e. exponential and trigonometric), which is undecidable [33].

2) Floating-Point Arithmetic Theory: The theory of Floating-Point (FP) arithmetic encodes IEEE-754 floating-point numbers [4], being mainly applied for enabling automated reasoning over machine arithmetic on software and hardware verification applications.

A common method for solving SMT Floating-Point arithmetic problems is to encode the theory formulas into SAT via bit-blasting. However, problems with a big number of arithmetic operations are frequently difficult for bit-blasting solvers due to the large quantity of Boolean variables and the inherent complexity of floating-point arithmetic encodings. Modern SMT solvers adopt efficient bit-vector encodings such as the SymFPU library [7], used in cvc5, or advanced rewrites, lemma generation and local search procedures as state-of-the-art solver Bitwuzla [13].

III. Handling Real and Floating-Point Arithmetic in Hardware Model Checking

This section describes the implementation extending Yosys, the SystemVerilog compiler, and Pono, the SMT-based model checker, into the proposed toolchain, which takes advantage of the arithmetic theories supported by SMT-based model checkers to solve verification problems on mixed-signal RNM designs. Our implementation is summarized in the block diagram in Fig. 2

The SystemVerilog RNM code is compiled by the Yosys frontend into its RTLIL (Register Transfer Level Intermediate Language) representation and processed by Yosys' backend into an SMV model checking problem for the SystemVerilog assertions (Section III-A). Pono then parses the SMV file and encodes it in either real arithmetic (RA) or floating-point arithmetic (FP) using `Smt-Switch` terms (Section

```
module main(input clk, rst, input real incr,
            output real q);
   real past_q;
   real past incr;
   always @(posedge clk) begin
       if (rst) begin
           q <= 0.0;
           past_q <= 0.0;
           past_incr <= 0.0;
       end else begin
           q <= q + incr;
           past_q <= q;
           past_incr <= incr;
       end
   end

   wire increasing = past_incr > 0.0;

   assert property (!increasing || past_q <= q);
endmodule
```

```
MODULE main
 IVAR
   _clk : unsigned word[1];
  _incr : real;
   rst : unsigned word[1];
 VAR
  _q : real;
   past incr : real;
   _past_q : real;
 DEFINE
  add := q + incr;
  _increasing := resize(word1(_past_incr > 0.0), 1);
  _logic_not := resize(word1((_increasing = 0ub1_0)), 1);
   _le := resize(word1(_past_q <= _q), 1);
   _logic_or := resize(word1((_logic_not != 0ub1_0)
                             | (_le != 0ub1_0)), 1);
   _next_past_incr := bool(_rst) ? 0.0 : _incr;
   _next_past_q := bool(_rst) ? 0.0 : _q;
   _next_q := bool(_rst) ? 0.0 : _add;
 ASSIGN
  next(_q) := _next_q;
   next(_past_incr) := _next_past_incr;
   next(_past_q) := _next_past_q;
INVARSPEC bool(_logic_or);
```

Figure 1: Example SystemVerilog RNM code (left) and the SMV generated from Yosys (right). Dashed arrows indicate the SMV construct generated for the SystemVerilog element.

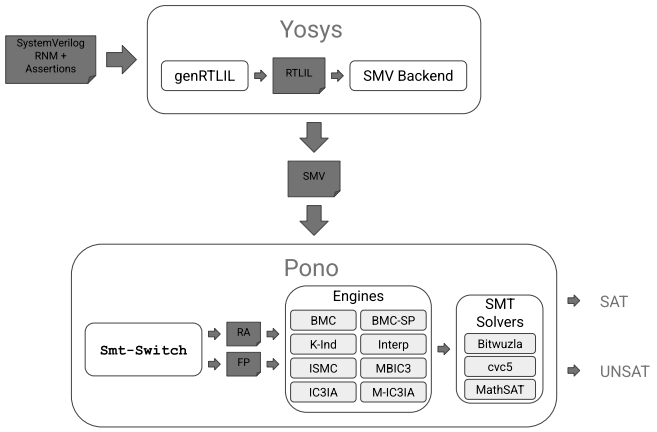


Figure 2: Block diagram of the proposed mixed-signal formal verification toolchain.

III-B), applying SMT model checking algorithms that delegate the underlying satisfiability queries to Bitwuzla [13], cvc5 [11] or MathSAT [12]. A property is returned as SAT (a counterexample was found) or UNSAT (the property is valid).

A. Extending Yosys to SystemVerilog RNM

Yosys is an industrial-grade and actively maintained Verilog and SystemVerilog synthesis suite, being employed alongside other open-source software to perform complete hardware projects from the early design stages up to the final silicon implementation. It provides a comprehensive frontend for Verilog HDL, covering most SystemVerilog features, and is also used for formal verification of digital RTL designs, with backends to encode designs and assertions into model checking languages such as SMV [34]. We encode all verification problems using the SMV representation, similarly to the Verilog2SMV work from Irfan et al. [24].

Despite being originally designed for digital hardware, the SMV format has been extended in the nuXmv model checker [9] to also encode infinite-state model checking using real and integer arithmetic sorts. The main extension in Yosys' SMV backend performed in this work was the translation of SystemVerilog RNM types and arithmetic operations into native SMV real arithmetic expressions. This required extending Yosys' parser and RTLIL representation to account for non-constant real variables and expressions and the implicit casts between real and binary variables, as well as adjusting Yosys' optimization passes to preserve real expressions that its digital transformations could otherwise invalidate. The generated representation is theory-agnostic: the interpretation in either the real or the floating-point theory is performed only in the model checker, as detailed in Section III-B. Fig. 1 shows an example SystemVerilog RNM design and the generated SMV, with dashed edges relating the corresponding constructs.

B. Extending Pono to Floating-Point Arithmetic Theory

The model checker used in this work is Pono [5], a word-level model checker supporting the theories of bit-vectors, arrays, and real and integer arithmetic, and integrating multiple SMT solvers for the underlying satisfiability queries.

We apply a comparative methodology to evaluate the performance and functionality of SystemVerilog RNM formal verification with two distinct interpretations for each verification problem:

- **Real Arithmetic:** Real variables are interpreted as infinite-precision real numbers, with the usual interpretation of the Real Arithmetic theories. This does not conform to the language standard [3], but, as we discuss, it provides a useful and meaningful framework for solving the verification problems.
- **Floating-Point Arithmetic:** Real variables are interpreted as double-precision floating-point variables with IEEE-754 [4] arithmetic semantics, conforming to the SystemVerilog standard.

Pono already supports SMV problems in Real Arithmetic without further modifications, so this work extends it to

Floating-Point Arithmetic by implementing an SMT-LIB-conforming floating-point interface [35] in `Smt-Switch` [10], the common API used by Pono to encode formulas to the different SMT solvers.

The same theory-agnostic SMV files are reused for floating-point problems through a new setting in Pono that models SMV `real` types with `Smt-Switch`'s 64-bit float sort, encoding arithmetic operations such as `+` and `*` to their floating-point counterparts. Since neither SystemVerilog nor SMV provide a standard way to represent rounding modes, all operations use the *round to nearest, ties to even* mode, the default in the SystemVerilog floating-point reference library VGM [36].

IV. A Benchmark for SystemVerilog RNM Model Checking

The benchmarks consist of 31 formal verification properties defined over 23 RNM designs, with some designs containing multiple properties. 17 benchmarks are publicly available at [14], and the remaining 14 were developed on top of proprietary Cadence RNM examples representing realistic industrial models. To the best of our knowledge, there is no earlier benchmark set for this kind of hardware verification problem.

Several modeling techniques were applied to develop practical formal verification testbenches for realistic SystemVerilog RNM designs. Many existing simulation models rely on asynchronous event-driven `always` blocks executed with a simulation time step defined by SystemVerilog `` `timescale `` directive, usually in the order of nanoseconds or picoseconds. However, this approach would be highly inefficient for model checking: if every nanosecond were a state transition, unrolling the transition system into any meaningful bound would be unfeasible.

We converted these asynchronous analog signals to synchronous versions controlled by global clock definitions, introducing a trade-off on the choice of a suitable sampling frequency that preserves the information of the analog signals at the same time it avoids the complexity penalty of high sampling rates. As a general rule, the sampling frequency should be at least the Nyquist rate, i.e. twice the highest frequency in the system. However, in our benchmarks, the asynchronous signals are either low-frequency or direct-current (DC), thus the digital clock frequency already used in the model is already fast enough to sample the signals correctly.

Furthermore, equality assertions have been weakened to check only for values up to a sufficiently small tolerance value ϵ, which prevents spurious counterexamples due to floating-point rounding errors. Floating-point values such as `NaN` and infinity have also been constrained when the variable represents a physical voltage or current so that trivial counterexamples with these values are not considered during verification.

a) Public Benchmarks: The 17 public benchmarks [14] include a discrete-time approximation of an analog RC circuit, several ADC (analog-to-digital converter) and DAC (digital-to-analog converter) models, and hybrid system problems converted from the KeYmaera X theorem prover [37]. The ADC and DAC models used implementations available in the literature, such as Maurice's [38] RNM models with verification testbenches for assertion-based equivalence checking.

We also implemented formal verification testbenches for Kundert and Zinke's [39] and Sapsanis et al.'s [27] digital and analog converter designs. Even though these models do not provide properties, comprehensive testbenches for known ADC and DAC behavior can be implemented by verifying that round-tripping a signal through ADC-DAC and DAC-ADC chains preserves the original signal.

On the hybrid system problems, we created 6 benchmarks by translating KeYmaera X' damped oscillator, ping-pong ball, driving car and bouncing ball examples to SystemVerilog RNM. These benchmarks represent simple models of physical systems and encode their expected properties for verification.

A key difference introduced in these tests is that SystemVerilog RNM is only able to model discrete-time dynamic systems, whereas the proof assistant's language models continuous-time differential equations. Therefore, these benchmarks were adapted to sample the values after each state transition with a fixed clock frequency using a technique similar to the sampling described in Section IV.

b) Proprietary RNM Models: The 14 proprietary benchmarks were based on Cadence's SystemVerilog Real Number Modeling (SV-RNM) Based Advanced Verification Training.[4] Designs in this set include common mixed-signal blocks such as clock generators, PLL (phase-locked loop) circuits, current sources, diodes, and a second-order analog filter. Their verification conditions range from simple range checks on the power supply setup to custom assertions for more interesting behaviors described in the specification documents, such as ensuring that the PLL locks after a specified number of cycles and that clock generators produce an oscillating pattern.

A block-level methodology was also applied to Cadence's Verification of the Power Intent of a Mixed Signal SoC,[5] which provides a low-power SoC with a processor and an off-chip voltage regulator. Our toolchain proves properties from the specification document for the VREG (voltage regulator) and PCM_ANA (Analog Power Control Module) blocks: the VREG verification ensures a stable supplied voltage after the reset sequence, and we developed a testbench with 3 assertions checking that PCM_ANA power state transitions follow the Multi-Supply Voltage (MSO) rules when the supply is reduced to low-power mode or restored to full power.

V. Evaluation

This section presents the experimental evaluation of the formal verification toolchain developed in this work. The experiments assess the following research questions:

RQ1 What are the differences in the verification results when interpreting the problems with real arithmetic or floating-point arithmetic semantics?

[4]Access can be requested at https://www.cadence.com/en_US/home/training/all-courses/86274.html.

[5]Access can be requested at [40].

Theory	SMT Solver	Engine	Proven	Solver Total	Theory Total
FP	Bitwuzla	BMC-SP	2	**24**	**26**
		K-Ind	20		
		MBIC3	2		
	cvc5	K-Ind	1	**1**	
	MathSAT	K-Ind	1	**1**	
RA	cvc5	K-Ind	5	**5**	**28**
	MathSAT	IC3IA	1	**23**	
		K-Ind	19		
		Interp	1		
		MBIC3	2		
		Total			**54**

(a) Number of benchmarks proven for each theory, engine and solver setting.

Theory	Benchmark	SMT Solver	Bound
FP	VREG	Bitwuzla	11
	Damped Oscillator	Bitwuzla, cvc5, MathSAT	1
	Short Bouncing Ball	Bitwuzla	7
	Short Bouncing Ball with Invariant	Bitwuzla	7
RA	Damped Oscillator	cvc5, MathSAT	1
	Short Bouncing Ball	MathSAT	51
	Short Bouncing Ball with Invariant	MathSAT	49

(b) Maximum bound reached by BMC on unsolved benchmarks.

Table I: Results of proven benchmarks and BMC on unsolved instances.

RQ2 How effective is SMT-based model checking in solving our SystemVerilog RNM benchmarks?

RQ3 How does solving the benchmarks in real arithmetic or floating-point arithmetic affect run time?

The experiments were performed on a cluster machine with Intel® Xeon® E5-2697 v3 14-core CPUs at 2.6 GHz and 256 GB RAM, running Red Hat Enterprise Linux 8.4. We consider a total of 62 benchmarks (the 31 benchmarks encoded in both RA and FP), each with a 100s timeout, using 8 model checking algorithms in Pono and 3 SMT solvers. The results shown are taken from the engine+solver setting with the best run time for each benchmark.

a) RQ1: On the Differences Between Arithmetic Theories: The benchmarks were constructed expecting that all properties are proven safe regardless of the arithmetic interpretation provided. The evaluation confirms that this is indeed the case for most benchmarks. However, one semantic mismatch between RA and FP occurs in the proprietary SystemVerilog RNM model representing a diode. The property encodes the specification that an input voltage of at least $V_d = 10\ mV$ must produce a positive voltage equal to V_d in the output pin, considering a tolerance of $\epsilon = 10\ \mu V$, i.e. the assertion $V_d - \epsilon \leq v \leq V_d + \epsilon$ over the output voltage v.

While this property is proven for the model in real arithmetic, Pono finds a counterexample for the version encoded in floating-point arithmetic. The counterexample trace assigns the value $V_d + \epsilon + 2 \times 10^{-18}$ to the output voltage v, exceeding the specification range by a small factor introduced by floating-point rounding. Since ϵ comes from the actual diode model specification, this counterexample could affect the expected verification outcomes in traditional simulation methodologies.

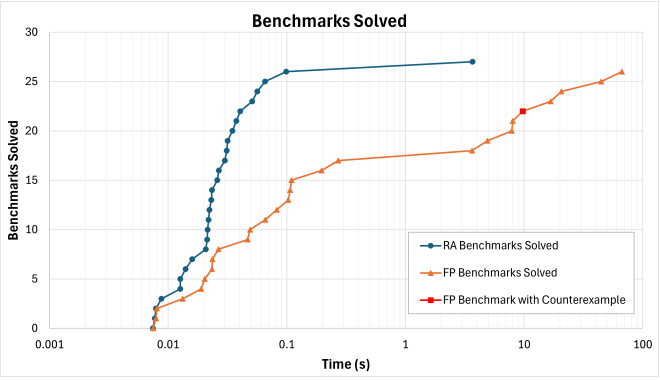
Benchmark	UnrollingDepth RA	UnrollingDepth FP	#Terms	#AnalogStateVars
adc_dac	1	1	237	0
maurice2024_adc	1	-	223	0
dac_adc	1	1	215	0
damped_oscillator	-	-	209	7
PCM_ANA_prop0*	1	1	206	0
PCM_ANA_prop1*	1	1	206	0
PCM_ANA_prop2*	1	1	206	0
VREG*	1	-	188	4
PLL*	-	-	186	1
dac_adc_formal	3	3	181	3
adc	3	3	154	2
fwd_driv_car	2	1	136	4
current_src0*	3	3	135	11
current_src1*	3	3	135	11
ev_trigg_ball	-	11	113	4
ev_trigg_ball_invar	-	11	113	4
sh_bounc_ball	-	-	105	6
sh_bounc_ball_invar	-	-	105	6
rnm	1	1	91	0
dac_prop1	4	4	89	3
dac_prop0	3	3	89	3
diode*	3	3	88	2
maurice2024_dac	1	1	82	0
4*	4	4	78	0
1*	1	1	78	0
2*	1	1	78	0
3*	1	1	78	0
low_pass_2nd_order*	4	-	61	7
5*	6	6	54	8
flop_rnm	-	2	46	3
capacitor	3	3	30	2

Table II: Characterization of the benchmarks by size and unrolling depth.

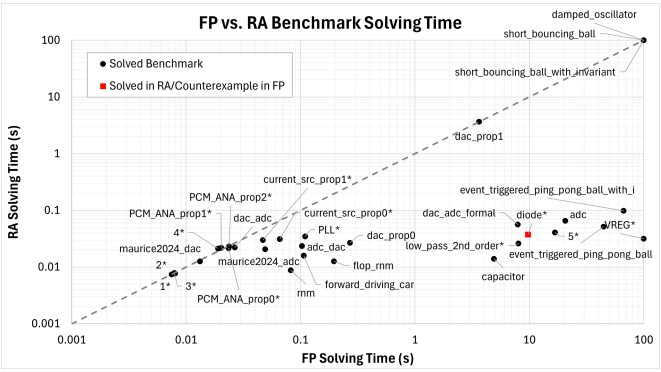
b) RQ2: On the Effectivity of SMT-Based Model Checking: The solved benchmarks are presented in Table Ia, categorized by theory. Only 7 benchmarks were not solved under 100s. 3 of the theorem proving problems [37] timed out in both theories, due to nonlinear arithmetic terms including quadratic and fourth-degree polynomials. The VREG benchmark [40] was not solved only in FP, due to the complexity of the floating-point formula obtained after unrolling the reset sequence. Even for the unsolved benchmarks, the maximum bound reached by BMC for each setting can be used to evaluate the effectivity of formal verification, as shown in Table Ib.

A detailed characterization of the benchmarks by size and k-induction unrolling depth is shown in table II. Unsolved benchmarks (or benchmarks solved by an algorithm other than k-induction) have their unrolling depths marked with a "-", and proprietary benchmark names are marked with an asterisk. The benchmarks are displayed in a decreasing order of number of SMT terms, with 12 out of 31 properties proven directly with induction after unrolling a single transition.

c) RQ3: On the Performance of Solving with Different Theories: The total number of solved benchmarks throughout time is plotted in Fig. 3a for both theories. Almost all real arithmetic problems are proven under 0.1s, while the timescale for the floating-point benchmarks is significantly larger, with proofs still found for instances around 8s, 20s, 44s and 67s and none after until the 100s timeout.



(a) Survival Plot of the total number of solved benchmarks per time period.



(b) Comparison of run time of benchmarks on RA and FP. Proprietary benchmark names are marked with an asterisk.

Figure 3: Run time comparison of the benchmarks solved in real (RA) and floating-point (FP) arithmetic.

We also evaluate how the choice of the background theory affects the run time for each benchmark individually. Fig. 3b plots each benchmark as a point with coordinates (t_{FP}, t_{RA}) where t_{FP} represents its run time when solved in FP and t_{RA} represents its run time in RA. Proprietary benchmark names are marked with an asterisk. Most samples lie below the identity line, and the complex benchmarks form a cluster grouping proprietary, public and KeYmaera X problems with the biggest difference between run times, where the RA solution reaches speed-ups ranging from 140x to 870x faster than FP. In particular, the VREG benchmark discussed in RQ2 was proven in 32ms in RA but was not solved in FP.

Discussion: The experiments show that the SMT-based model checking methodology is successful in solving our benchmark collection, with the engine+solver setting having a significant impact: the best results were favorable to Bitwuzla in FP and MathSAT in RA, which also got the largest bounds in BMC on the unsolved tests.

Even though the benchmarks exercise realistic SystemVerilog RNM patterns, most of the properties considered are simple enough to be fully proven by inductive methods with a small unrolling depth. This indicates that our benchmarks do not represent yet the hardest problems that could be handled by Pono: the solved properties were mostly k-inductive, not requiring use the use of state-of-the-art IC3-based engines. Moreover, most of the benchmarks are small and have few analog state variables, with 12 out of 31 tests having either no state element or having only digital state variables, and nonlinear terms appear in few designs, with only damped_oscillator, low_pass_2nd_order, short_bouncing_ball and short_bouncing_ball_with_invariant covering nonlinear arithmetic problems.

The BMC evaluation highlights the improvement from choosing real arithmetic: even without a complete proof, the performance gain of RA over FP allows an order of magnitude increase in the bound checked.

On the functional side, the results support applying the RA model in spite of the standards-conforming FP interpretation: analog variables modeled as `real` values represent physical quantities such as voltage or current, so floating-point rounding or `NaN`/infinity values are an incidental complexity not required when introducing the formal verification interpretation for such models. Careful modeling is still required, as the diode example showed the RA model might diverge from the FP model, thus the verification results must be carefully pondered against the constraints, tolerances and parameters of the modeled system. Despite providing performance gains, the real arithmetic verification model may disagree with simulation results (which follow strictly the FP semantics as specified by SystemVerilog). As a general guideline, FP should be used when verification must exactly match RNM simulation, with RA available for problems that cannot be solved in FP.

VI. Conclusion

We have implemented an effective formal verification toolchain for mixed-signal hardware designs, validated with realistic SystemVerilog RNM testbenches and built by leveraging model checking algorithms lifted to the SMT arithmetic theories implemented in Pono. Despite SystemVerilog RNM being widely adopted for mixed-signal hardware for over a decade, to the best of our knowledge this is the first formal verification flow proposed for this domain.

Solving speed-ups of up to 870x were obtained by exploring the core thesis of our work: mixed-signal hardware systems can be more naturally encoded using real arithmetic theory. Even though this interpretation requires weighing the modeling trade-offs involved, we claim that real arithmetic is a straightforward and more scalable encoding for formal verification.

Finally, we expect our benchmarks to support future research as validation for model checker development and as a modeling reference for adopting formal verification. As future work, we intend to target harder problems with IC3-based engines and extend the toolchain to temporal and liveness properties and transcendental arithmetic.

Acknowledgments: This work was partially supported by a gift from Amazon Web Services and by the Defense

Advanced Research Projects Agency (DARPA) under contract FA8750-24-2-1001. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of DARPA.

DATA AVAILABILITY

The artifact(s) associated with this paper is/are available at Zenodo under the following DOI: 10.5281/zenodo.19902956.

REFERENCES

[1] E. Seligman, T. Schubert, and M. V. A. K. Kumar, *Formal Verification: An Essential Toolkit for Modern VLSI Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2015.

[2] S. Balasubramanian and P. Hardee, "Solutions for mixed-signal SoC verification using real number models," tech. rep., Cadence Design Systems, 2013.

[3] IEEE, "IEEE standard for SystemVerilog–unified hardware design, specification, and verification language," *IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009)*, pp. 1–1315, 2013.

[4] IEEE, "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, 2019.

[5] M. Mann, A. Irfan, F. Lonsing, Y. Yang, H. Zhang, K. Brown, A. Gupta, and C. Barrett, "Pono: A flexible and extensible SMT-based model checker," in *Computer Aided Verification* (A. Silva and K. R. M. Leino, eds.), (Cham), pp. 461–474, Springer International Publishing, 2021.

[6] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani, "Incremental linearization for satisfiability and verification modulo nonlinear arithmetic and transcendental functions," *ACM Trans. Comput. Logic*, vol. 19, Aug. 2018.

[7] M. Brain, F. Schanda, and Y. Sun, "Building better bit-blasting for floating-point problems," in *Tools and Algorithms for the Construction and Analysis of Systems* (T. Vojnar and L. Zhang, eds.), (Cham), pp. 79–98, Springer International Publishing, 2019.

[8] C. Wolf, "Yosys open SYnthesis suite." https://yosyshq.net/yosys/, 2013.

[9] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuXmv symbolic model checker," in *Computer Aided Verification* (A. Biere and R. Bloem, eds.), (Cham), pp. 334–342, Springer International Publishing, 2014.

[10] M. Mann, A. Wilson, C. Tinelli, and C. Barrett, "Smt-switch: A solver-agnostic C++ API for SMT solving (extended abstract)," in *Proceedings of the 18th International Workshop on Satisfiability Modulo Theories (SMT '20)*, jul 2020.

[11] H. Barbosa, C. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, and Y. Zohar, "cvc5: A versatile and industrial-strength SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems* (D. Fisman and G. Rosu, eds.), (Cham), pp. 415–442, Springer International Publishing, 2022.

[12] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani, "The MathSAT5 SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems* (N. Piterman and S. A. Smolka, eds.), (Berlin, Heidelberg), pp. 93–107, Springer Berlin Heidelberg, 2013.

[13] A. Niemetz and M. Preiner, "Bitwuzla," in *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II* (C. Enea and A. Lal, eds.), vol. 13965 of *Lecture Notes in Computer Science*, pp. 3–17, Springer, 2023.

[14] A. Mafra, "augustomafra/sv_rnm_formal_verification: v2.0.5 (v2.0.5). zenodo. https://doi.org/10.5281/zenodo.19902956," 2026.

[15] N. Gurushankar, "Functional verification methodologies for mixed signal designs," *Journal of Engineering and Applied Sciences Technology*, vol. 4, pp. 1–4, 06 2022.

[16] M. H. Zaki, S. Tahar, and G. Bois, "Formal verification of analog and mixed signal designs: A survey," *Microelectronics Journal*, vol. 39, no. 12, pp. 1395–1404, 2008.

[17] C. Zivkovic, C. Grimm, M. Olbrich, O. Scharf, and E. Barke, "Hierarchical verification of AMS systems with affine arithmetic decision diagrams," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 10, pp. 1785–1798, 2019.

[18] A. Jesser and L. Hedrich, "A symbolic approach for mixed-signal model checking," in *2008 Asia and South Pacific Design Automation Conference*, pp. 404–409, 2008.

[19] E. Barke, D. Grabowski, H. Graeb, L. Hedrich, R. Popp, S. Steinhorst, and Y. Wang, "Formal approaches to analog circuit verification," in *Proceedings -Design, Automation and Test in Europe, 2009*, pp. 724–729, 04 2009.

[20] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson, "Formal verification of analog designs using MetiTarski," in *2009 Formal Methods in Computer-Aided Design*, pp. 93–100, 2009.

[21] L. Yin, Y. Deng, and P. Li, "Verifying dynamic properties of nonlinear mixed-signal circuits via efficient SMT-based techniques," in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '12, (New York, NY, USA), p. 436–442, Association for Computing Machinery, 2012.

[22] D. Walter, S. Little, N. Seegmiller, C. J. Myers, and T. Yoneda, "Symbolic model checking of analog/mixed-signal circuits," in *2007 Asia and South Pacific Design Automation Conference*, pp. 316–323, 2007.

[23] D. Walter, S. Little, and C. Myers, "Bounded model checking of analog and mixed-signal circuits using an SMT solver," in *Automated Technology for Verification and Analysis* (K. S. Namjoshi, T. Yoneda, T. Higashino, and Y. Okamura, eds.), (Berlin, Heidelberg), pp. 66–81, Springer Berlin Heidelberg, 2007.

[24] A. Irfan, A. Cimatti, A. Griggio, M. Roveri, and R. Sebastiani, "Verilog2SMV: A tool for word-level verification," in *2016 Design, Automation & Test in Europe Conference & Exhibition (2016)*, pp. 1156–1159, 2016.

[25] A. Biere, N. Froleyks, and M. Preiner, "Hardware model checking competition 2025," in *Proceedings of the 25th Conference on Formal Methods in Computer-Aided Design - FMCAD 2025* (A. Irfan and D. Kaufmann, eds.), pp. 7–7, TU Wien Academic Press, 2025.

[26] H. Mohanty and D. N. Gadde, "Analogous alignments: Digital "formally" meets analog," in *Design and Verification Conference and Exhibition - DVCON 2024, Munich, Germany, October 15-16, 2024, Proceedings*, 2024.

[27] C. Sapsanis, M. Villemur, and A. G. Andreou, "Real number modeling of a SAR ADC behavior using SystemVerilog," in *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pp. 1–4, 2022.

[28] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking* (E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, eds.), pp. 305–343, Cham: Springer International Publishing, 2018.

[29] L. de Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems* (C. R. Ramakrishnan and J. Rehof, eds.), (Berlin, Heidelberg), pp. 337–340, Springer Berlin Heidelberg, 2008.

[30] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Abstract DPLL and abstract DPLL modulo theories," in *Logic for Programming, Artificial Intelligence, and Reasoning* (F. Baader and A. Voronkov, eds.), (Berlin, Heidelberg), pp. 36–50, Springer Berlin Heidelberg, 2005.

[31] A. Cimatti, A. Griggio, A. Irfan, M. Roveri, and R. Sebastiani, "Invariant checking of NRA transition systems via incremental reduction to LRA with EUF," in *Tools and Algorithms for the Construction and Analysis of Systems* (A. Legay and T. Margaria, eds.), (Berlin, Heidelberg), pp. 58–75, Springer Berlin Heidelberg, 2017.

[32] E. Ábrahám, J. H. Davenport, M. England, and G. Kremer, "Deciding the consistency of non-linear real arithmetic constraints with a conflict driven search using cylindrical algebraic coverings," *Journal of Logical and Algebraic Methods in Programming*, vol. 119, p. 100633, 2021.

[33] A. Cimatti, A. Griggio, E. Lipparini, and R. Sebastiani, "Handling polynomial and transcendental functions in SMT via unconstrained optimisation and topological degree test," in *Automated Technology for Verification and Analysis* (A. Bouajjani, L. Holík, and Z. Wu, eds.), (Cham), pp. 137–153, Springer International Publishing, 2022.

[34] K. L. McMillan, *Symbolic model checking - an approach to the state explosion problem*. PhD thesis, Carnegie Mellon University, 1992.

[35] C. Barrett, P. Fontaine, and C. Tinelli, "The SMT-LIB standard: Version 2.7," tech. rep., Department of Computer Science, The University of Iowa, 2025.

[36] W. E. Ferguson, Jr. and F. L. Andersen, "Verilog Golden Model (VGM) library," 2019.

[37] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer, "KeYmaera X: An axiomatic tactical theorem prover for hybrid systems," in *CADE* (A. P. Felty and A. Middeldorp, eds.), vol. 9195 of *LNCS*, pp. 527–538, Springer, 2015.

[38] M. Maurice, "Functional verification of analog devices modeled using SV-RNM," in *Design and Verification Conference and Exhibition - DVCON 2024, San Jose, CA, USA, March 4-7, 2024, Proceedings*, 2024.
[39] O. Z. Kenneth S. Kundert, *The Designer's Guide to Verilog-AMS*. Springer New York, NY, 1st ed., 2004.
[40] C. XTeam, "RAK attack: Verifying power intent for low power mixed signal SoCs." https://community.cadence.com/cadence_blogs_8/b/fv/posts/rak-attack-verifying-power-intent-for-low-power-mixed-signal-socs, 2018.