Charge Pump Phase-Locked Loops and Full Wave Rectifiers for Reachability Analysis (Benchmark Proposal)

Omar Ali Beg, Ali Davoudi, and Taylor T. Johnson

University of Texas at Arlington, USA

Abstract

Analog-mixed signal (AMS) circuits are widely used in various mission-critical applications necessitating their formal verification prior to implementation. We consider modeling two AMS circuits as hybrid automata, particularly a charge pump phase-locked loop (CP-PLL) and a full-wave rectifier (FWR). We present executable models for the benchmarks in SpaceEx format, perform reachability analysis, and demonstrate their automatic conversion to MathWorks Simulink/Stateflow (SLSF) format using the HyST tool. Moreover, as a next step towards implementation, we present the VHDL-AMS description of a circuit based on the verified model.

Category: academic **Difficulty:** medium

1 Context and Origins

Many analog-mixed signal (AMS) circuits are widely used in various mission cicritical applications and require formal verification prior implementation. Formal verification methods construct a mathematical model $\mathcal{M}$ with precise semantics, provide extensive analysis with respect to some correctness requirement $\mathcal{P}$, and verify that $\mathcal{M} \models \mathcal{P}$ [2]. This can be ascertained through reachability analysis [1]. As an example of circuitry that can benefit from formal verification prior to field implementation and deployment, we provide two potential benchmarks for hybrid verification research community, i.e., charge pump phase-locked loop (CP-PLL), and full-wave rectifier (FWR).

CP-PLL integrated circuits are widely used in modern mobile, radio, and wireless communication applications to synchronize a high-frequency signal with a low-frequency reference signal. In [8], the auhtors use SpaceEx model checking tool [6] to verify the global convergence with respect to phase and frequency lock for a digital PLL. An FWR converts an AC electric input signal to a DC output signal, and formal verification through reachability analysis has been reported using different model checking tools in [5], except SpaceEx. We develop hybrid automaton models of CP-PLL and FWR, and used SpaceEx [6], a reachability analysis tool, to compute the over-approximated sets of reachable states [1]. This a classical fixed point computation tool that operates on symbolic states.

We also use HyST (Hybrid Source Transformer) [3] to automatically convert the hybrid automaton models developed in SpaceEx to MathWorks Simulink/Stateflow (SLSF) models [2]. It is a source-to-source translation tool that takes input in the SpaceEx model format, and translates it to the formats of HyCreate,Flow*, dReach, C2E2, Passel 2.0, and HyComp. Additional tool support is being added from time to time. Verification and validation research community may use HyST to automatically transform the hybrid automaton models in SpaceEx format to

[1] The tool is available online from the SpaceEx website at: http://spaceex.imag.fr/.

[2] The executable models are included on the ARCH website and are also available online from the HyST website at: http://verivital.com/hyst/.

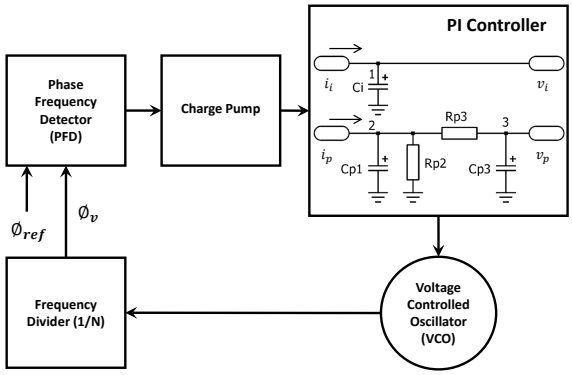

Figure 2.1: Block diagram of the PLL circuit with a PI controller.

other formats and perform reachability analysis using aforesaid model checking tools. Finally, we present VHDL-AMS description of an FWR.

2 Hybrid Automaton Modeling of CP-PLL and FWR

In this section, we present the hybrid automaton modeling of CP-PLL and FWR.

2.1 CP-PLL Modeling

We consider a third-order CP-PLL as described in [1]. It consists of a reference frequency signal generator, a phase frequency detector (PFD), a charge pump, a proportional-integral (PI) controller, a voltage- controlled oscillator (VCO) and a frequency divider as shown in Figure 2.1. The state variables are defined by the voltages across the capacitors C_i, C_{p1}, and C_{p3}, i.e., v_i, v_{p1}, and v_p respectively. Two more state variables are defined by the dynamics of VCO and reference frequencies, i.e., ϕ_v and ϕ_{ref}, respectively. CP-PLL is designed such that ϕ_v locks on to ϕ_{ref}, that may constitute the property of CP-PLL to be verified. This locking is ensured by PFD using the phase difference of ϕ_{ref} and ϕ_v to generate 'UP' or 'DN' signal for the charge pump.

The ODEs from the CP-PLL circuit diagram can be readily formed using the traditional circuit analysis techniques, i.e., Kirchoff's voltage law (KVL) and Kirchoff's current law (KCL). We apply KCL at node 1 of the circuit used to implemented the analog PI controller shown in Figure 2.1

$$i_i = i_{C_i} \tag{2.1}$$

We can write the above equation in terms of voltage across capacitor C_i as

$$C_i.\dot{v}_i = i_i. \tag{2.2}$$

Rearranging the above equation, we obtain

$$\dot{v}_i = \frac{i_i}{C_i} \tag{2.3}$$

We apply KCL at node 2 of the the circuit used to implemented the analog PI controller in Figure 2.1 to get

$$i_p = i_{C_{p1}} + i_{R_{p2}} + i_{R_{p3}} \tag{2.4}$$

Replacing the current terms with voltage terms in right hand side of above equation, we get

$$i_p = C_{p1}\dot{v}_{p1} + \frac{v_{p1}}{R_{p2}} + \frac{(v_{p1} - v_p)}{R_{p3}}. \tag{2.5}$$

Rearranging the above equation for $\dot{v}_{p1}$, we get

$$\dot{v}_{p1} = -\frac{v_{p1}}{C_{p1}}\left(\frac{1}{R_{p2}} + \frac{1}{R_{p3}}\right) + \frac{v_p}{C_{p1}R_{p3}} + \frac{i_p}{C_{p1}}. \tag{2.6}$$

Next, we may apply KCL at node 3 to get

$$i_C p3 = i_{R_p 3} \tag{2.7}$$

Re-writing the above equation in terms of voltages, we get

$$C_{p3}\dot{v}_p = \frac{v_{p1} - v_p}{R_{p3}} \tag{2.8}$$

Rearranging the above equation leads to

$$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}. \tag{2.9}$$

For the VCO, the output phase ϕ_v is the integral of the frequency and the input voltages, i.e., v_i, and v_p [7]. We also include the frequency division factor N to obtain the ODE as

$$\dot{\phi}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0 \tag{2.10}$$

and

$$\dot{\phi}_{ref} = 2\pi f_{ref}. \tag{2.11}$$

Here, K_i and K_p are the voltage-to-frequency gains for v_i and v_p respectively, and f_0 is the frequency of VCO. These ODEs depict the continuous dynamics within each discrete location. The input to the PI controller, i.e., $[i_i, i_p]^T$, is generated by the charge pump depending upon the relative phase of ϕ_v and ϕ_{ref}. This phase difference is measured by PFD, which generates an 'UP' signal if ϕ_{ref} leads ϕ_v, and 'DN' signal if ϕ_v leads ϕ_{ref}. An 'UP' signal will charge the capacitors, hence increasing the voltages across the capacitors of the proportional and integrator channel, i.e., v_p and v_i, respectively, leading to an increased VCO frequency. On the other hand, a 'DN' signal from PFD will tend the charge pump to produce current in reverse direction to discharge the capacitors, hence reducing the voltages in the PI channel. The reduced v_p and v_i voltages will result in a reduced ϕ_v to make it track ϕ_{ref}. Depending upon the status of Up/Down signals, there may be four discrete locations (i.e., the input varies for each discrete location) as follows:

1 $Both_0$ (i.e. Both OFF): The input vector is given by $[i_i, i_p]^T = [0, 0]^T$

2 Up_1 (i.e. UP ON): The input vector is given by $[i_i, i_p]^T = \left[I_i^{up}, I_p^{up}\right]^T$

3 $Both_1$ (i.e. Both ON): The input vector is given by $[i_i, i_p]^T = \left[I_i^{up} + I_i^{dn}, I_p^{up} + I_p^{dn}\right]^T$

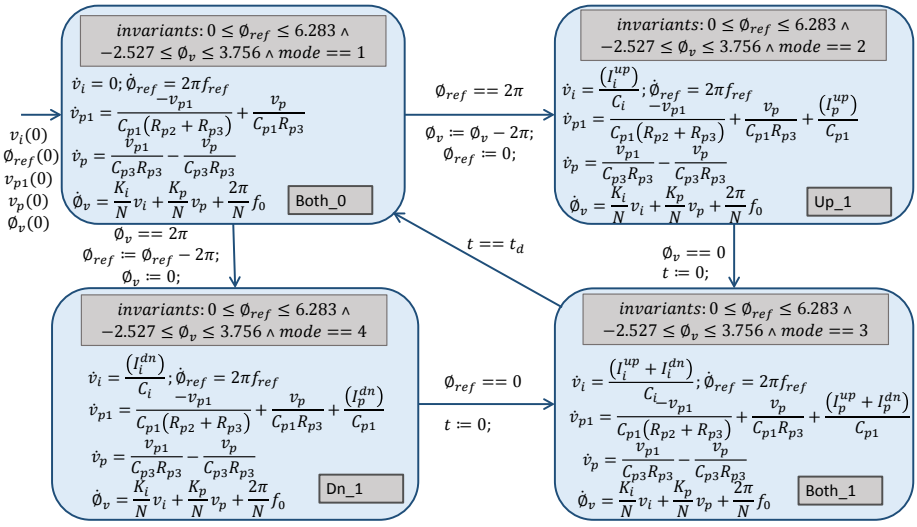

Figure 2.2: Hybrid automaton model for CP-PLL system.

4 Dn_1 (i.e. DN ON): The input vector is given by $[i_i, i_p]^T = \left[I_i^{dn}, I_p^{dn}\right]^T$

Accordingly, using the above ODEs and the inputs defined, a hybrid automaton is shown in Figure 2.2. The component values used in the model are as per Table 1 of [1]. Moreover, the input values are: $I_i^{up} = 10.1\mu A, I_i^{dn} = -10.1\mu A$, and $I_p^{up} = 505\mu A, I_p^{dn} = -505\mu A$. The guard conditions for discrete transitions are formed depending upon ϕ_{ref} and ϕ_v. As discussed earlier, the PFD output depends on whether ϕ_{ref} leads or lags with respect to ϕ_v. If the initial discrete location is $Both_0$, the automaton jumps to Up_1 if ϕ_{ref} leads as $\phi_{ref} = 2\pi$, otherwise it jumps to Dn_1 if ϕ_v leads as $\phi_v = 2\pi$. There is a design requirement to introduce a time delay, t_d, required to switch off both the charge pumps. This is represented by the location $Both_1$. Once the lagging signal reaches zero, the automaton jumps to this location and, once $t = t_d$, the automaton transitions back to $Both_0$.

2.2 FWR Modeling

We consider an FWR as described in [5]. It is basically a full-wave diode bridge, that consists of two diodes D_1 and D_2, a capacitor C and the load resistor R as shown in Figure 2.3. An AC input signal is supplied to the circuit through a center-tapped transformer. For the modeling purpose, and without the lack of generality, we use two AC sources as shown in Figure 2.3. This circuit converts the input AC voltage V_{in} to a DC voltage V_o, at its output measured across R. We may need to verify that V_o is stable within $\pm 1\% V_{max}$ for the steady-state operation, where V_{max} is the maximum value of the input AC signal.

For modeling purposes, we consider R_d as the forward resistance of each diode. Let the current through R_d, C, and R be i_{Rd}, i_C, and i_R, respectively. The input sinusoidal voltage be $V_{in} = V_{max} sin(2\pi f t)$, and the output voltage across the load resistor R be V_o, where, V_{max} is the maximum amplitude of the sinusoidal signal and f is its frequency. For model checking purposes, we use SpaceEx that requires hybrid automaton model with linear dynamics, so we model the input AC signal using a second-order differential equation [5]. We define another

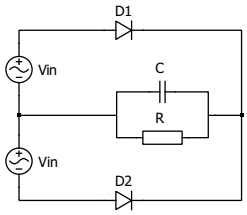

Figure 2.3: Schematic diagram of FWR.

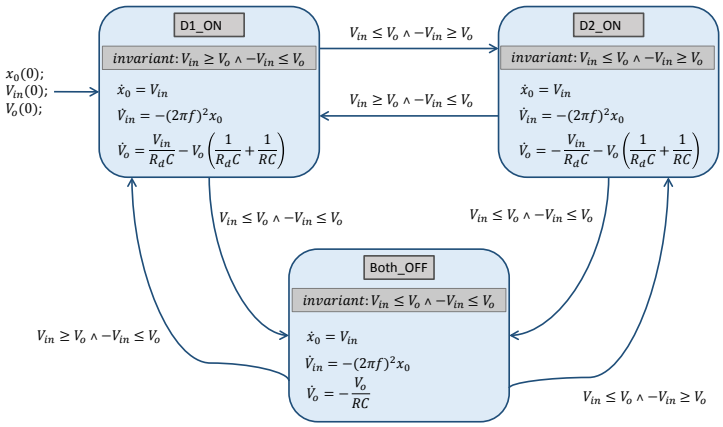

Figure 2.4: Hybrid automaton model for FWR system.

state variable x_0 and model the AC input by ODEs defined as

$$\dot{x}_0 = V_{in} \tag{2.12}$$

and

$$\dot{V}_{in} = -\left(2\pi f\right)^2 x_0 \tag{2.13}$$

The solution of above system is $V_{in} = V_{max} sin(2\pi ft)$ such that the initial conditions are $x_0 = \dfrac{-V_{max}}{2\pi f}$ and $V_{in} = 0$. Next, we consider the FWR circuit dynamics to form ODE for V_o. The circuit dynamics depend upon the operation of diodes D_1 and D_2. Accordingly, we may form three different topological instances, i.e., D_1 ON and D_2 OFF, D_1 OFF and D_2 ON, and both the diodes OFF when $V_{in} \leq V_o$. There could be a fourth topological instance, i.e., both the diodes ON at the same time, but this is not practical due to the nature of the sinusoidal input. Therefore, we may consider three topologies one by one to form the ODEs and start with the topology with D_1 ON and D_2 OFF. The invariants for this topological instance are $V_{in} \geq V_o \wedge -V_{in} \leq V_o$. Applying KCL at the node joining C and R in Figure 2.3, we get

$$i_{Rd} = i_C + i_R \tag{2.14}$$

and we can express the above equation in terms of voltages as

$$\frac{V_{in} - V_o}{R_d} = C\dot{V}_o + \frac{V_o}{R}. \tag{2.15}$$

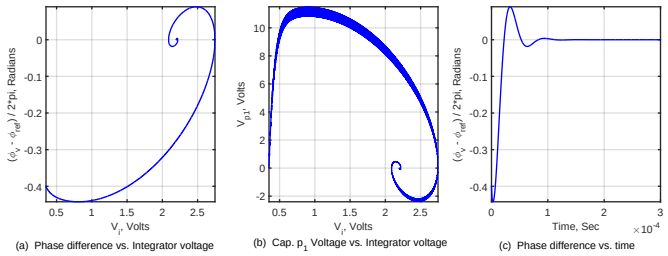

Figure 3.1: SLSF plots for PLL showing stable limit cycles and ϕ_v locking onto ϕ_{ref} within 0.2 mSec.

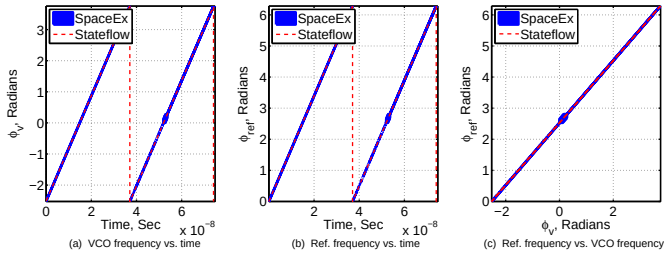

Figure 3.2: Comparison of SpaceEx reach sets and SLSF trajectories for PLL.

Rearranging the above equation provides

$$\dot{V_o} = \frac{V_{in}}{R_d C} - V_o \left(\frac{1}{R_d C} + \frac{1}{RC} \right). \tag{2.16}$$

By the same token, for D_1 OFF and D_2 ON with invariants $V_{in} \leq V_o \wedge -V_{in} \geq V_o$, we use KCL at the same node in Figure 2.3 to get

$$\dot{V_o} = -\frac{V_{in}}{R_d C} - V_o \left(\frac{1}{R_d C} + \frac{1}{RC} \right). \tag{2.17}$$

For the topology when both D_1 and D_2 are OFF, the sinusoidal input signal is cut off from the entire circuit and the load voltage is only provided by the capacitor. The invariants for this topological instance are $V_{in} \leq V_o \wedge -V_{in} \leq V_o$. Therefore, we get

$$\dot{V_o} = -\frac{V_o}{RC}. \tag{2.18}$$

Accordingly, the hybrid automaton model of FWR is shown in Figure 2.4. In addition, we consider the VHDL-AMS description of FWR in Section A, where the circuit is externally supplied by V_{in}.

3 SLSF Simulations and Reachability Analysis

Formal verification of CP-PLL constitutes verifying its frequency-locking property, i.e., whether ϕ_v locks onto ϕ_{ref}. For this purpose, we need to compute the phase difference between ϕ_v and

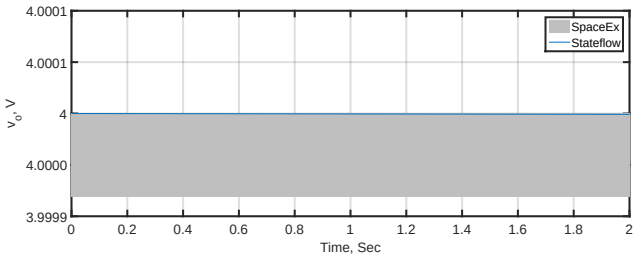

Figure 3.3: Comparison of SpaceEx and SLSF for the output voltage of FWR in the steady state, showing the simulation trace containment within overapproximated sets of reachable states.

ϕ_{ref}. The SLSF plots for the phase difference vs. integrator voltage, voltage of capacitor p_1 vs. integrator voltage v_i, and the phase difference versus time are shown in Figure 3.1. The first two plots depict a stable limit cycle highlighting stability properties of CP-PLL. In the third plot, we show that the phase difference between ϕ_{ref} and ϕ_v reaches zero within 0.2 mSec., signifying that ϕ_v locks onto ϕ_{ref} within such time intervals.

We also analyze the hybrid automaton using SpaceEx, and a comparison of the first few iterations for SpaceEx and SLSF is shown in Figure 3.2. We show that SLSF simulation traces, and the over-approximated sets of reachable states computed using SpaceEx, match for the first five iterations. CP-PLL requires thousands of cycles to lock, hence there will be thousands of discrete transitions for the switching logic resulting inaccuracy due to SpaceEx overapproximations [1]. It is evident from comparing the first five iterations in Figure 3.2 that SLSF simulation traces are contained within the over-approximated sets of reachable states. We also conclude that the SLSF traces exhibit stable limit cycles, and that frequency locking is achieved within 0.2 mSec.

As evident from this benchmark, the performance of reachability analysis tools is not satisfactory due to the high number of discrete transitions (practically being in order of thousands). It is pertinent to highlight that in [4], the authors have used a variant of continuization [1] to address this problem for the design of a yaw damper system for a 747 jet aircraft. Continuization is a process whereby the abstraction of a hybrid system having large number of discrete transitions is obtained by a continuous system with an extra non-deterministic input. The authors use HyST to automatically transform the model and perform reachability analysis using Flow* and SpaceEx to display satisfactory results in [4]. A similar approach can be used for this benchmark so as to perform reachability analysis using SpaceEx and Flow*.

We perform the reachability analysis using SpaceEx under the steady-state conditions for FWR, i.e., $V_{max} = 4V$, $V_o(0) = 4V$, and $f = 50Hz$, as shown in Figure 3.3. The steady-state SLSF time traces for the output voltage are contained within the over-approximated sets of reachable states computed using SpaceEx.

During conversion from SpaceEx to SLSF using HyST, the conversion time noted for CP-PLL is 1.633077 seconds and that for FWR is 1.936676 seconds. We used MATLAB Release 2015a on a Windows 7, 64 bit operating system with Intel Core i7-2600 CPU at 3.40 GHz and 16 GB RAM.

4 Key Observations

Hybrid automaton modeling and reachability analysis of CP-PLL using traditional model checking tools, such as SpaceEx, is an extensive challenge. This is due to the reason that CP-PLL requires thousand of cycles to lock, resulting in thousand of discrete transitions in the switching logic. Therefore, the SpaceEx analysis did not produce accurate reachability results if the analysis is run for an extended duration of time. This requires some advanced techniques, such as continuization [1] that is demostarted in [4] using HyST, SpaceEx, and Flow*. For FWR, SpaceEx produced a run-time error due to non-affine dynamics as the model had pure sinusoidal time-dependent signal as an input. Therefore, we have modeled the sinusoidal input signal using the second-order ODEs to successfully compute the reachability analysis results.

5 Benchmark Outlook

Overall, these verification benchmarks have medium difficulty level, and can serve as a first step towards a benchmark library to evaluate reachability and verification methods for AMS circuits. These benchmarks are open to the continuous and hybrid systems verification community to evaluate their methods and tools.

Acknowledgments The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] Matthias Althoff, Akshay Rajhans, Bruce H. Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. *Commun. ACM*, 56(10):97–104, October 2013.

[2] Rajeev Alur. Formal verification of hybrid systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 273–278. IEEE, 2011.

[3] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. Hyst: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.

[4] Stanley Bak and Taylor T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *36th IEEE Real-Time Systems Symposium (http://2015.rtss.org/RTSS 2015/)*, San Antonio, Texas, December 2015. IEEE Computer Society.

[5] Luca P Carloni, Roberto Passerone, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. *Languages and tools for hybrid systems design*. now Publishers Inc, 2006.

[6] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In Shaz Qadeer Ganesh Gopalakrishnan, editor, *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[7] Jaewook Kim and Seonghwan Cho. A time-based analog-to-digital converter using a multi-phase voltage controlled oscillator. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 3934–3937, May 2006.

[8] Jijie Wei, Yan Peng, Ge Yu, and M. Greenstreet. Verifying global convergence for a digital phase-locked loop. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 113–120, Oct 2013.

A Appendix: VHDL-AMS Description of FWR

As discussed in Section 2, the FWR circuit behavior depends upon the state of the diodes being ON or OFF due to the input sinusoidal signal. We assume that this signal is supplied externally, and form the description as per Equation 2.16, Equation 2.17, and Equation 2.18. It should be mentioned that, in VHDL-AMS, we must minimize the use of the division operation. VHDL-AMS models are typically comprised of two sections, i.e., an entity and an architecture. Entity describes the model interface to the outside world, whereas, architecture describes the function or behavior of the model. A VHDL-AMS description is given below:

```
library ieee;
use ieee.electrical_systems.all;
use ieee.math_real.all;
entity fwr is
   port ( terminal input: electrical;
          terminal output: electrical );
end entity fwr;
-----------------------------------------------------------------
architecture dot of fwr is
   quantity vin across input to electrical_ref;
   quantity vout across output to electrical_ref;
   constant r : real := 1000; -- load resistance
   constant rd : real := 0.1; -- diode forward resistance
   constant cap : real := 0.001; -- capacitance
begin
   if vin >= vout and -vin <= vout use
      vin == vout'dot * r * rd + vout + vout * rd / r; -- diode D1 ON
   elseif vin <= vout and -vin >= vout use
      - vin == vout'dot * r * rd + vout + vout * rd / r; -- diode D2 ON
   elseif vin <= vout and - vin <= vout use
      vout == - vout'dot * r * cap; -- Both OFF
   end if;
end architecture dot;
```