Symbolic Model Checking of Analog/Mixed-Signal Circuits

David Walter\textsuperscript{1} Scott Little\textsuperscript{1} Nicholas Seegmiller\textsuperscript{1} Chris Myers\textsuperscript{1} Tomohiro Yoneda\textsuperscript{2}

\textsuperscript{1}University of Utah
Salt Lake City, UT 84112, USA

\textsuperscript{2}National Institute of Informatics
Tokyo, Japan

January 24, 2007
ASP-DAC 2007
Motivation

- System on a chip (SoC) designs are becoming commonplace.
- They include digital and analog/mixed-signal (AMS) circuits.
- Validation is typically divided since digital requires long time steps while analog requires short time steps.
- Strict divisions in SoC validation, however, are rarely possible.
- New functional validation methods are needed to support the heterogeneous nature of SoC designs.
Formal verification has shown advantages for digital circuits.
Research has begun in formal methods for AMS circuits.
AMS verification is complicated by the need to accurately track continuous quantities such as voltages and currents.
Alur et al. developed *hybrid automata* symbolic model-checking procedure for embedded systems.

Seshia/Bryant developed timed automata symbolic model-checking procedure using BDDs.

This work describes a hybrid Petri net symbolic model-checking procedure that uses BDDs for verifying analog/mixed-signal systems.

Crucial to the acceptance of new AMS formal methods is the use of a familiar description language.

The *labeled hybrid Petri net* (LHPN) model was developed to facilitate generation from a VHDL-AMS subset.

We have developed a compiler that generates LHPNs given a circuit description using this subset of VHDL-AMS.
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments

\[ \langle \max : T \wedge \dot{x} := 2 \rangle \]

\[ p_0 \overset{t_0}{\rightarrow} p_1 \overset{t_1}{\rightarrow} \]

\[ \{ \text{open} \} \langle x := 0 \wedge \dot{x} := -1 \rangle \]

\[ \{ l \geq 18 \} \]
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments

\[
\begin{align*}
\langle \text{max} := & \ T \land \dot{x} := 2 \rangle & \qquad [1, 2] \\
\{l \geq 18\} & \quad [1, 2] \\
\{\text{open}\} & \quad [1, 2]
\end{align*}
\]
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.

- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments

```
\langle \text{max} := T \land \dot{x} := 2 \rangle [1, 2]
\langle \text{open} \rangle [1, 2]
\{l \geq 18\}
\langle x := 0 \land \dot{x} := -1 \rangle [1, 2]
```
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments

\[
\begin{align*}
\{l \geq 18\} & \quad t_0 \\
\langle \text{max} := T \land x := 2 \rangle & \quad [1, 2] \\
\{\text{open}\} & \quad t_1 \\
\langle x := 0 \land \dot{x} := -1 \rangle & \quad [1, 2]
\end{align*}
\]
Labeled Hybrid Petri Nets (LHPNs)

- Composed of a Petri net and labels operating on continuous variables and Boolean signals.
- Label types are:
  - Enablings
  - Delay bounds
  - Boolean assignments
  - Value assignments
  - Rate assignments

\[
\begin{align*}
\langle \text{max} := T \land \dot{x} := 2 \rangle & \quad [1, 2] \\
\{ l \geq 18 \} & \\
\end{align*}
\]
Switched Capacitor Integrator Circuit

\[ \frac{dV_{out}}{dt} = \pm 20 \text{ mV/µs} \]

\[ \text{freq}(V_{in}) = 5 \text{ kHz} \]

\[ V_{in} = \pm 1000 \text{ mV} \]

\[ C_1 = 1 \text{ pF} \]

\[ C_2 = 25 \text{ pF} \]

\[ \text{freq}(\Phi_1) = \text{freq}(\Phi_2) = 500 \text{ kHz} \]

\[ dV_{out}/dt = \pm 20 \text{ mV/µs} \]
Switched Capacitor Output Waveform

Typical Simulation

![Graph showing the output voltage (Vout) over time (us)].

- **Vout (mV)**
  - Values range from -2000 to 0 to 2000 mV.
- **Time (us)**
  - Range from 0 to 5000 us.

The graph illustrates a periodic waveform with a consistent oscillation pattern.
Switched Capacitor Integrator Circuit

\[ \frac{dV_{out}}{dt} = \pm (18 \text{ to } 22) \text{ mV/µs} \]

\[ f_{\text{freq}(\Phi_1)} = f_{\text{freq}(\Phi_2)} = 500 \text{ kHz} \]

\[ V_{in} = \pm 1000 \text{ mV} \]

\[ f_{\text{freq}(V_{in})} = 5 \text{ kHz} \]

\[ C_1 = 1 \text{ pF} \]

\[ C_2 = 25 \text{ pF} \]
Switched Capacitor Integrator Circuit

$V_{in} = \pm 1000 \text{ mV}$
$freq(V_{in}) = 5 \text{ kHz}$
$freq(\Phi_1) = freq(\Phi_2) = 500 \text{ kHz}$
$dV_{out}/dt = \pm (18 \text{ to } 22) \text{ mV/\mu s}$

Does $V_{out}$ saturate? (i.e. $-2000 \text{ mV} \leq V_{out} \leq 2000 \text{ mV}$)
Switched Capacitor Output Waveform
Random Simulation

![Graph showing a periodic waveform with Vout in mV on the y-axis and Time in us on the x-axis. The waveform oscillates between -2000 and 2000 mV with a period of approximately 2000 us.](image)
Switched Capacitor Output Waveform
Worst Case Simulation

![Graph showing the switched capacitor output waveform with amplitude and time axes.]
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; --Initial value
  if Vin='0' use
    Vout'dot == span(18.0, 22.0);
  elsif Vin = '1' use
    Vout'dot == span(-22.0, -18.0);
  end use;
  process begin
    assign(Vin,'1',100,100);
    assign(Vin,'0',100,100);
  end process;
  assert (Vout'above(-2000.0) and
    not Vout'above(2000.0))
    report "error"
    severity failure;
end switchCap;
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; --Initial value
  if Vin='0' use
    Vout'dot == span(18.0, 22.0);
  elsif Vin = '1' use
    Vout'dot == span(-22.0, -18.0);
  end use;
  process begin
    assign(Vin,'1',100,100);
    assign(Vin,'0',100,100);
  end process;
  assert (Vout'above(-2000.0) and
              not Vout'above(2000.0))
    report 'error'
    severity failure;
end switchCap;
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
    signal Vin:std_logic := '0';
    quantity Vout:real;
begin
    break Vout => -1000.0; --Initial value
    if Vin='0' use
        Vout'dot == span(18.0, 22.0);
    elsif Vin = '1' use
        Vout'dot == span(-22.0, -18.0);
    end use;
    process begin
        assign(Vin,'1',100,100);
        assign(Vin,'0',100,100);
    end process;
    assert (Vout'above(-2000.0) and not Vout'above(2000.0))
        report "error"
        severity failure;
end switchCap;
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; -- Initial value
  if Vin='0' use
    Vout'dot == span(18.0, 22.0);
  elsif Vin = '1' use
    Vout'dot == span(-22.0, -18.0);
  end use;
  process begin
    assign(Vin,'1',100,100);
    assign(Vin,'0',100,100);
  end process;
  assert (Vout'above(-2000.0) and
          not Vout'above(2000.0))
    report 'error'
    severity failure;
end switchCap;
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; --Initial value
  if Vin='0' use
    Vout'dot == span(18.0, 22.0);
  elsif Vin = '1' use
    Vout'dot == span(-22.0, -18.0);
  end use;
  process begin
    assign(Vin,’1’,100,100);
    assign(Vin,’0’,100,100);
  end process;
  assert (Vout’above(-2000.0) and
          not Vout’above(2000.0))
    report ‘‘error’’
    severity failure;
end switchCap;
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; --Initial value
  if Vin='0' use
    Vout'dot == span(18.0, 22.0);
  elsif Vin = '1' use
    Vout'dot == span(-22.0, -18.0);
  end use;
  process begin
    assign(Vin,'1',100,100);
    assign(Vin,'0',100,100);
  end process;
  assert (Vout'above(-2000.0) and
    not Vout'above(2000.0))
    report "error"
    severity failure;
end switchCap;
Create Boolean variables for each of the following:

- **Marking:** $p_0, p_1, \ldots, p_4$
- **Boolean signals:** $Vin, fail$
- **Boolean rate variables:** $\dot{V}_{out}[18, 22], \dot{V}_{out}[-22, -18]$
- **Clock active variables:** $a_{t_0}, a_{t_1}, \ldots, a_{t_4}$
Create Boolean variables for each of the following:

- **Marking:** $p_0, p_1, \ldots, p_4$
- **Boolean signals:** $Vin, fail$
- **Boolean rate variables:** $\dot{V}_{out}[18,22], \dot{V}_{out}[-22,-18]$
- **Clock active variables:** $a_{t_0}, a_{t_1}, \ldots a_{t_4}$
Create Boolean variables for each of the following:

- **Marking:** \( p_0, p_1, \ldots, p_4 \)
- **Boolean signals:** \( Vin, fail \)
- **Boolean rate variables:** \( \dot{V}_{out}[18,22], \dot{V}_{out}[−22,−18] \)
- **Clock active variables:**
  - \( at_0, at_1, \ldots, at_4 \)
Create Boolean variables for each of the following:

- **Marking:** $p_0, p_1, \ldots, p_4$
- **Boolean signals:** $Vin, fail$
- **Boolean rate variables:** $\dot{V}out_{[18,22]}, \dot{V}out_{[-22,-18]}$
- **Clock active variables:** $a_{t_0}, a_{t_1}, \ldots a_{t_4}$
Maintain relationships among real values including system variables (\(V_{out}\)) and clock variables (\(c_{t_0}, c_{t_1}, ... c_{t_4}\)) using inequalities of the form \(c_i x_i \geq c_j x_j + c\).

Restrictions are placed upon inequalities to ensure canonicity.

Boolean encoding:
- Replace \(c_i x_i \geq c_j x_j + c\) with a Boolean variable.
- Note that \(c_i x_i > c_j x_j + c\) is represented as \(c_j x_j \geq c_i x_i + -c\).

Hybrid Separation Logic (HSL):

\[
\phi ::= \text{true} | \text{false} | b | \neg \phi | \phi \land \phi | c_i x_i \geq c_j x_j + c
\]
Boolean Constraints

Boolean constraints are periodically created as new inequalities are generated to maintain relationships among inequalities:

- **Same variables, different constants:**

  \[ 2x \geq 3y + 5 \Rightarrow 2x \geq 3y + 4 \]

- **Same variables, same constants:**

  \[ 2x > 3y + 5 \Rightarrow 2x \geq 3y + 5 \]

- **Transitivity:**

  \[ 2x \geq 3y + 5 \land 4y \geq 5z + 5 \Rightarrow \]
Boolean constraints are periodically created as new inequalities are generated to maintain relationships among inequalities:

- **Same variables, different constants:**

  \[ 2x \geq 3y + 5 \Rightarrow 2x \geq 3y + 4 \]

- **Same variables, same constants:**

  \[ 2x > 3y + 5 \Rightarrow 2x \geq 3y + 5 \]

- **Transitivity:**

  \[ 2x \geq 3y + 5 \land 4y \geq 5z + 5 \Rightarrow 2x \geq 3y + 5 \]

  \[ \frac{2}{3}x \geq y + \frac{5}{3} \land y \geq \frac{5}{4}z + \frac{5}{4} \Rightarrow \]
Boolean Constraints

Boolean constraints are periodically created as new inequalities are generated to maintain relationships among inequalities:

- **Same variables, different constants:**
  
  \[ 2x \geq 3y + 5 \Rightarrow 2x \geq 3y + 4 \]

- **Same variables, same constants:**

  \[ 2x > 3y + 5 \Rightarrow 2x \geq 3y + 5 \]

- **Transitivity:**

  \[
  \begin{align*}
  2x & \geq 3y + 5 \land 4y \geq 5z + 5 \\
  \Rightarrow \\
  \frac{2}{3}x & \geq y + \frac{5}{3} \land y \geq \frac{5}{4}z + \frac{5}{4} \\
  \Rightarrow \\
  \frac{2}{3}x & \geq \frac{5}{4}z + \frac{5}{3} + \frac{5}{4}
  \end{align*}
  \]
Boolean Constraints

Boolean constraints are periodically created as new inequalities are generated to maintain relationships among inequalities:

- Same variables, different constants:

\[ 2x \geq 3y + 5 \Rightarrow 2x \geq 3y + 4 \]

- Same variables, same constants:

\[ 2x > 3y + 5 \Rightarrow 2x \geq 3y + 5 \]

- Transitivity:

\[
\begin{align*}
2x & \geq 3y + 5 \land 4y \geq 5z + 5 \\
\frac{2}{3}x & \geq y + \frac{5}{3} \land y \geq \frac{5}{4}z + \frac{5}{4} \\
\Rightarrow \quad \frac{2}{3}x & \geq \frac{5}{4}z + \frac{5}{3} + \frac{5}{4} \\
\Rightarrow \quad 8x & \geq 15z + 35
\end{align*}
\]
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]
Invariant

A statement that must always be satisfied.

$$\phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)})$$
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\bar{a}_t \Rightarrow \bar{\bullet} t \lor \bar{E}(t)) \]

\[ p_0 \quad \bar{p}_1 \quad p_2 \quad \bar{p}_3 \quad p_4 \quad \bar{V}in \quad \bar{fail} \quad \dot{V}out[\bar{-22},\bar{-18}] \quad \dot{V}out[18,22] \lor \]

\[ p_0 \quad \bar{p}_1 \quad p_2 \quad \bar{p}_3 \quad p_4 \quad V\text{in} \quad \bar{fail} \quad \dot{V}out[\bar{-22},\bar{-18}] \quad \dot{V}out[18,22] \lor \]

\[ \bar{p}_0 \quad p_1 \quad \bar{p}_2 \quad p_3 \quad p_4 \quad \bar{Vin} \quad \bar{fail} \quad \dot{V}out[\bar{-22},\bar{-18}] \quad \dot{V}out[18,22] \lor \]

\[ \bar{p}_0 \quad p_1 \quad \bar{p}_2 \quad \bar{p}_3 \quad p_4 \quad \bar{Vin} \quad \bar{fail} \quad \dot{V}out[\bar{-22},\bar{-18}] \quad \dot{V}out[18,22] \lor \]
A statement that must always be satisfied.

\[
\phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)})
\]
A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]

\[ \begin{aligned} &p_0 \quad \overline{p_1} \quad p_2 \quad p_3 \quad p_4 \quad \overline{Vin} \quad \overline{fail} \quad \overline{\dot{V}_{out}\left[-22, -18\right]} \quad \dot{V}_{out}[18, 22] \lor \\ &p_0 \quad \overline{p_1} \quad p_2 \quad p_3 \quad p_4 \quad \overline{Vin} \quad \overline{fail} \quad \overline{\dot{V}_{out}\left[-22, -18\right]} \quad \dot{V}_{out}[18, 22] \lor \\ &\overline{p_0} \quad p_1 \quad \overline{p_2} \quad p_3 \quad p_4 \quad \overline{Vin} \quad \overline{fail} \quad \overline{\dot{V}_{out}\left[-22, -18\right]} \quad \dot{V}_{out}[18, 22] \lor \\ &\overline{p_0} \quad p_1 \quad p_2 \quad p_3 \quad p_4 \quad \overline{Vin} \quad \overline{fail} \quad \overline{\dot{V}_{out}\left[-22, -18\right]} \quad \dot{V}_{out}[18, 22] \lor \end{aligned} \]
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]
Invariant

A statement that must always be satisfied.

$$\phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)})$$

Pseudo-Inverse:

$$a \land \overline{x} \leq 2000 = \overline{a} \lor x \geq 2000$$
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \not\bullet t \lor \overline{E(t)}) \]

\[ (a_{t_0} \Rightarrow p_0 \land \text{Vin} \land c_{t_0} = 0) \land (\overline{a_{t_0}} \Rightarrow \overline{p_0} \lor \overline{\text{Vin}}) \]
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]

\[
\begin{align*}
(a_{t_0} & \Rightarrow p_0 \land Vin \land c_{t_0} = 0) \land (\overline{a_{t_0}} & \Rightarrow \overline{p_0} \lor \overline{Vin}) \end{align*}
\]
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]

\[
(a_{t_0} \Rightarrow p_0 \land \overline{Vin} \land c_{t_0} = 0) \land (\overline{a_{t_0}} \Rightarrow \overline{p_0} \lor \overline{Vin}) \land \\
(a_{t_1} \Rightarrow p_1 \land \overline{Vin} \land c_{t_1} = 0) \land (\overline{a_{t_1}} \Rightarrow \overline{p_1} \lor Vin)
\]
A statement that must always be satisfied.

\[
\phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \bullet \overline{t} \lor \overline{E(t)})
\]

\[
\begin{aligned}
(a_{t_0} \Rightarrow p_0 \land \overline{Vin} \land c_{t_0} = 0) \land (\overline{a_{t_0}} \Rightarrow \overline{p_0} \lor \overline{Vin}) \land \\
(a_{t_1} \Rightarrow p_1 \land \overline{Vin} \land c_{t_1} = 0) \land (\overline{a_{t_1}} \Rightarrow \overline{p_1} \lor Vin) \land \\
(a_{t_2} \Rightarrow p_2 \land 0 \leq c_{t_2} \leq 100) \land (\overline{a_{t_2}} \Rightarrow \overline{p_2})
\end{aligned}
\]
Invariant

A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]

\[
\begin{align*}
(a_{t_0} \Rightarrow p_0 \land \text{Vin} \land c_{t_0} = 0) \land (\overline{a_{t_0}} \Rightarrow \overline{p_0} \lor \overline{\text{Vin}}) \land \\
(a_{t_1} \Rightarrow p_1 \land \overline{\text{Vin}} \land c_{t_1} = 0) \land (\overline{a_{t_1}} \Rightarrow \overline{p_1} \lor \text{Vin}) \land \\
(a_{t_2} \Rightarrow p_2 \land 0 \leq c_{t_2} \leq 100) \land (\overline{a_{t_2}} \Rightarrow \overline{p_2}) \land \\
(a_{t_3} \Rightarrow p_3 \land 0 \leq c_{t_3} \leq 100) \land (\overline{a_{t_3}} \Rightarrow \overline{p_3})
\end{align*}
\]
A statement that must always be satisfied.

\[ \phi_I = \Phi \land \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \land E(t) \land 0 \leq c_t \leq u(t)) \land (\overline{a_t} \Rightarrow \overline{\bullet t} \lor \overline{E(t)}) \]

\[
\begin{align*}
    (a_{t_0} &\Rightarrow p_0 \land \text{Vin} \land c_{t_0} = 0) \land (\overline{a_{t_0}} \Rightarrow \overline{p_0} \lor \overline{\text{Vin}}) \land \\
    (a_{t_1} &\Rightarrow p_1 \land \overline{\text{Vin}} \land c_{t_1} = 0) \land (\overline{a_{t_1}} \Rightarrow \overline{p_1} \lor \overline{\text{Vin}}) \land \\
    (a_{t_2} &\Rightarrow p_2 \land 0 \leq c_{t_2} \leq 100) \land (\overline{a_{t_2}} \Rightarrow \overline{p_2}) \land \\
    (a_{t_3} &\Rightarrow p_3 \land 0 \leq c_{t_3} \leq 100) \land (\overline{a_{t_3}} \Rightarrow \overline{p_3}) \land \\
    (a_{t_4} &\Rightarrow p_4 \land c_{t_4} = 0) \land \\
    (\overline{\text{Vout}} \leq -2000 \lor \overline{\text{Vout}} \geq 2000) \land \\
    (\overline{a_{t_4}} \Rightarrow \overline{p_4} \lor (\text{Vout} \geq -2000 \land \text{Vout} \leq 2000))
\end{align*}
\]
Fires transition causing marking update and assignments.

For each transition $t_i$:

$$C_P = \bigcup_{t \in T} \{ \langle \phi_{GP}(t), A_P(t) \rangle \}$$

where

$$\phi_{GP}(t) = (\bullet t \land t \bullet - \bullet t \land E(t) \land a_t \land c_t \geq l(t))$$

$$A_P(t) = \{(\bullet t - t \bullet) := F, (t \bullet) := T, a_t := F, c_t := [-\infty, \infty], BA(t), VA(t), RA(t)\}$$
Primary Guarded Command for Integrator Example

\[
\phi_{GP}(t_2) = p_2 \land \overline{p_3} \land a_{t_2} \land c_{t_2} \geq 100
\]

\[
A_P(t_2) =
\]

\[
\{ Vin \} [0, 0]
\]
\[
\langle \dot{V}_{out} := [-22, -18] \rangle
\]

\[
\{ \neg Vin \} [0, 0]
\]
\[
\langle V_{out} := [18, 22] \rangle
\]
\[
[100, 100]
\]
\[
\langle Vin := T \rangle
\]

\[
\{ V_{out} \leq -2000 \lor V_{out} \geq 2000 \}
\]
\[
[0, 0] \langle fail := T \rangle
\]

\[
\{ Vin \} [0, 0]
\]
\[
\langle \dot{V}_{out} := [-22, -18] \rangle
\]

\[
\{ \neg Vin \} [0, 0]
\]
\[
\langle V_{out} := [18, 22] \rangle
\]
\[
[100, 100]
\]
\[
\langle Vin := T \rangle
\]
Primary Guarded Command for Integrator Example

\[ \phi_{GP}(t_2) = p_2 \land \overline{p_3} \land a_{t_2} \land c_{t_2} \geq 100 \]

\[ \mathcal{A}_P(t_2) = \{ p_2 := F, p_3 := T, a_{t_2} := F, \\
\quad c_{t_2} := [-\infty, \infty], \\
\quad Vin := T \} \]
Secondary Guarded Command Set

- Activates or deactivates clock based on marking, enabling condition, and clock value.
- For each transition $t_i$:

$$C_S = \bigcup_{t \in T} \{ \langle \phi_{G_{SA}}(t), A_{SA}(t) \rangle, \langle \phi_{G_{SD}}(t), A_{SD}(t) \rangle \}$$

where

$$\phi_{G_{SA}}(t) = \bullet t \land E(t) \land \overline{a_t}$$

$$A_{SA}(t) = \{ a_t := T, c_t := [0, 0] \}$$

$$\phi_{G_{SD}}(t) = (\overline{\bullet t} \lor E(t)) \land a_t$$

$$A_{SD}(t) = \{ a_t := F, c_t := [-\infty, \infty] \}$$
Activating Guarded Command:

\[ \phi_{GS_A}(t_0) = p_0 \land \text{Vin} \land \overline{a_{t_0}} \]
\[ A_{SA}(t_0) = \]

Deactivating Guarded Command:

\[ \phi_{GS_D}(t_0) = \]
\[ A_{SD}(t_0) = \]
Activating Guarded Command:
\[
\phi_{G_{SA}}(t_0) = p_0 \land Vin \land \overline{a_{t_0}} \\
A_{SA}(t_0) = \{ a_{t_0} := T, c_{t_0} := [0, 0] \}
\]

Deactivating Guarded Command:
\[
\phi_{G_{SD}}(t_0) = \\
A_{SD}(t_0) = 
\]
Activating Guarded Command:

\[ \phi_{GA}(t_0) = p_0 \land \text{Vin} \land \overline{a_{t_0}} \]

\[ A_{SA}(t_0) = \{ a_{t_0} := T, c_{t_0} := [0, 0] \} \]

Deactivating Guarded Command:

\[ \phi_{GS}(t_0) = (\overline{p_0} \lor \overline{\text{Vin}}) \land a_{t_0} \]

\[ A_{SD}(t_0) = \]
Secondary Guarded Commands for Integrator Example

Activating Guarded Command:

\[ \phi_{GA}(t_0) = p_0 \land Vin \land \overline{a_{t_0}} \]
\[ A_{SA}(t_0) = \{a_{t_0} := T, c_{t_0} := [0, 0]\} \]

Deactivating Guarded Command:

\[ \phi_{GS}(t_0) = (\overline{p_0} \lor \overline{Vin}) \land a_{t_0} \]
\[ A_{SD}(t_0) = \{a_{t_0} := F, c_{t_0} := [-\infty, \infty]\} \]
Firing a transition may activate or deactivate clocks associated with other transitions, therefore primary and secondary guarded commands must be merged.

Create new guarded commands when a primary assignment modifies a secondary guard.
Merging Guarded Commands for Integrator Example

\[ \phi_G(t_2, t_0) = p_0 \land p_2 \land \overline{p_3} \land \overline{a_{t_0}} \land a_{t_2} \land c_{t_2} \geq 100 \]

\[ A(t_2, t_0) = \{ p_2 := F, p_3 := T, Vin := T, a_{t_0} := T, c_{t_0} := [0, 0], a_{t_2} := F, c_{t_2} := [-\infty, \infty] \} \]
Dense real-time CTL is translated to a $T\mu$ calculus formula.

In the integrator example, the $T\mu$ property is

$$
\phi_{init} \implies \neg \mu Y. [fail \lor (\text{true} \triangleright Y)]
$$

where $\phi_{init}$, the initial state, is:

$$
\phi_{init} = p_0 \overline{p_1} p_2 \overline{p_3} p_4 \overline{Vin} \overline{Fail} \overline{Vout}_{[-22,-18]} \overline{Vout}_{[18,22]} \overline{a_t_0} \overline{a_t_1} \overline{a_t_2} \overline{a_t_3} \overline{a_t_4} \land c_{t_2} = 0 \land Vout = -1000
$$

If a state in which $fail$ is true cannot be reached from the initial state, then the formula evaluates to true.
Analysis proceeds by starting with a set of states that violate a property and finding all possible states that could have reached that state via time elapsing and/or firing transitions. If the initial state is encompassed within the calculated region, property is violated.
Time Precondition

\[ p_0 \xrightarrow{t_0} p_1 \]
\[ p_1 \]

\[ \{Vin\} [0, 0] \\
\langle Vout := [22, 18] \rangle \\
\]

\[ \{Vin\} [0, 0] \\
\langle Vout := [18, 22] \rangle \\
\]

\[ p_2 \xrightarrow{t_2} p_3 \]
\[ p_3 \]

\[ \{Vin\} [100, 100] \\
\langle Vin := T \rangle \\
\]

\[ \{Vin\} [100, 100] \\
\langle Vin := F \rangle \\
\]

\[ Vout = 2000 \]
\[ c_{t_2} = 100 \]
Time Precondition

\[ V_{out} \leq 18c_{t2} + 200 \]

\[ (\dot{V}_{out} := [18, 22]) \]

\[ 0 \leq c_{t2} \leq 100 \]

\[ V_{out} \geq 22c_{t2} - 200 \]

\[ V_{out} \leq 18c_{t2} + 200 \]

\[ (\dot{V}_{out} := [-22, -18]) \]

\[ \{ Vin \} [0, 0] \]

\[ (\dot{V}_{out} := [18, 22]) \]

\[ \{ Vin \} [0, 0] \]

\[ \{ Vin \} [0, 0] \]

\[ V_{out} = 2000 \]

\[ c_{t2} = 100 \]

\[ (\dot{V}_{out} := [-22, -18]) \]

\[ \{ Vin \} [0, 0] \]

\[ (\dot{V}_{out} := [18, 22]) \]

\[ \{ Vin \} [0, 0] \]

\[ (\dot{V}_{out} := [18, 22]) \]
Transition Precondition

\[ \begin{align*}
V_{out} &\leq 18c_{t_2} + 200 \\
0 &\leq c_{t_2} \leq 100 \\
\{Vin\} &\leq 0 \\
\langle V_{in} := T \rangle &\leq [100, 100] \\
\langle V_{out} := [18, 22] \rangle &\leq 2000 \\
\{Vin\} &\geq 0 \\
\langle V_{in} := F \rangle &\geq [100, 100] \\
\langle \dot{V}_{out} := [-22, -18] \rangle &
\end{align*} \]
Transition Precondition

\[ \{ \text{Vin} \} [0, 0] \]
\[ \langle \dot{V}_{\text{out}} := [-22, -18] \rangle \]

\[ p_0 \]
\[ t_0 \]
\[ t_1 \]
\[ p_1 \]

\[ \{ \text{Vin} \} [0, 0] \]
\[ \langle \dot{V}_{\text{out}} := [18, 22] \rangle \]

\[ -200 \leq V_{\text{out}} \leq 200 \]
\[ c_{t_3} = 100 \]
Experimental Results

- Expermented with different ranges of rates for $V_{out}$ in switched capacitor integrator circuit.
- With equal lower and upper rates, property is verified to be satisfied within a few seconds.
- With ranges of rates, property is verified to fail within a few seconds.
Corrected Switched Capacitor Integrator

\[ dV_{out} \text{ } dt = \begin{cases} 
22 \text{ to } 24 \text{ mV} / \mu s \text{ when } V_{out} \leq -1000 \\
16 \text{ to } 22 \text{ mV} / \mu s \text{ when } V_{out} > -1000 \\
-(22 \text{ to } 24) \text{ mV} / \mu s \text{ when } V_{out} > 1000 \\
-(16 \text{ to } 22) \text{ mV} / \mu s \text{ when } V_{out} \leq 1000 
\end{cases} \]
Conclusions

- Hybrid system verification methods are necessary to verify analog/mixed-signal systems.
- It is possible to convert a subset of VHDL-AMS to LHPNs to encourage acceptance by AMS designers.
- Developed symbolic model checking method which uses BDDs for labeled hybrid Petri nets.
- Separation predicates are mapped to BDD variables.
Future Work

- Provide a SPICE-deck front-end to further improve the ability of AMS designers to use our tool.
- Use abstraction methods to reduce number of BDD variables.
- Support handling of false negatives when applying abstraction.
- Implement algorithm using an SMT solver such as Yices.