Symbolic Model Checking of Analog/Mixed-Signal Circuits

David Walter[1] Scott Little[1] Nicholas Seegmiller[1]
Chris Myers[1] Tomohiro Yoneda[2]

[1]University of Utah
Salt Lake City, UT 84112, USA

[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

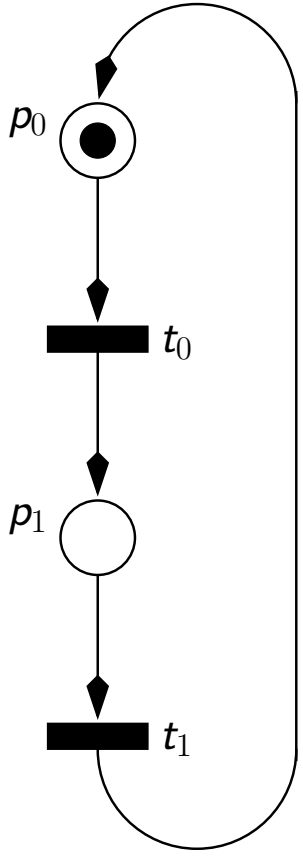
- Formal verification has shown advantages for digital circuits.
- Research has begun in formal methods for AMS circuits.
- AMS verification is complicated by the need to accurately track continuous quantities such as voltages and currents.

Background

- 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.

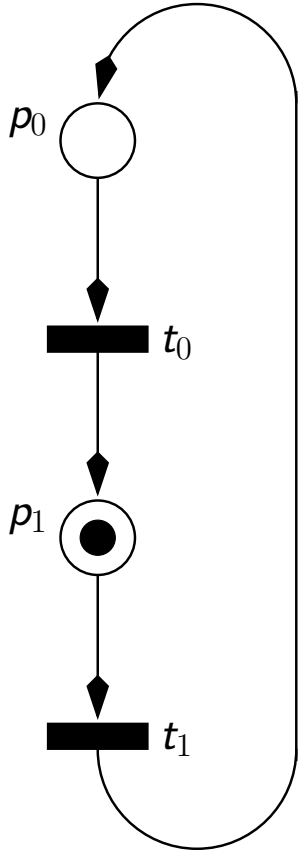
Labeled Hybrid Petri Nets (LHPNs)

- 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



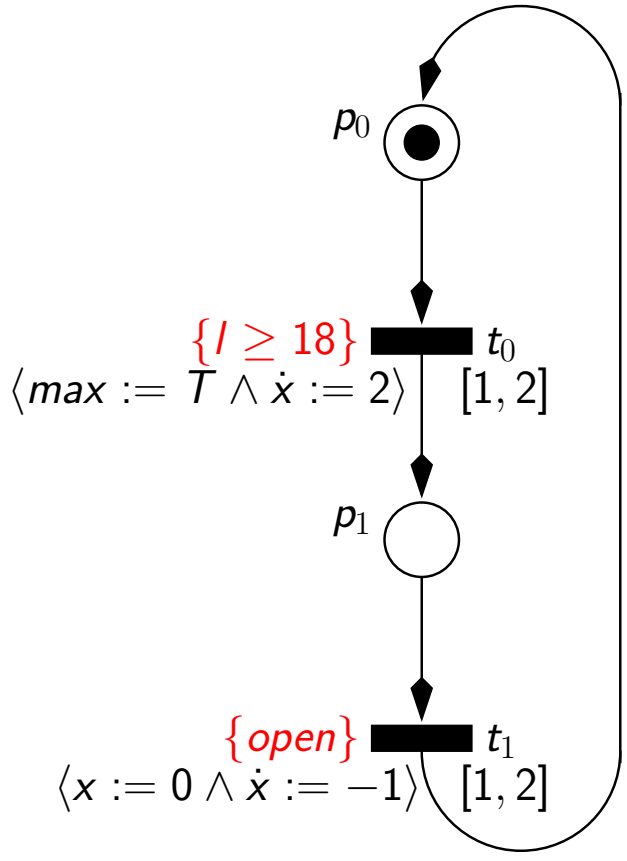
Labeled Hybrid Petri Nets (LHPNs)

- 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



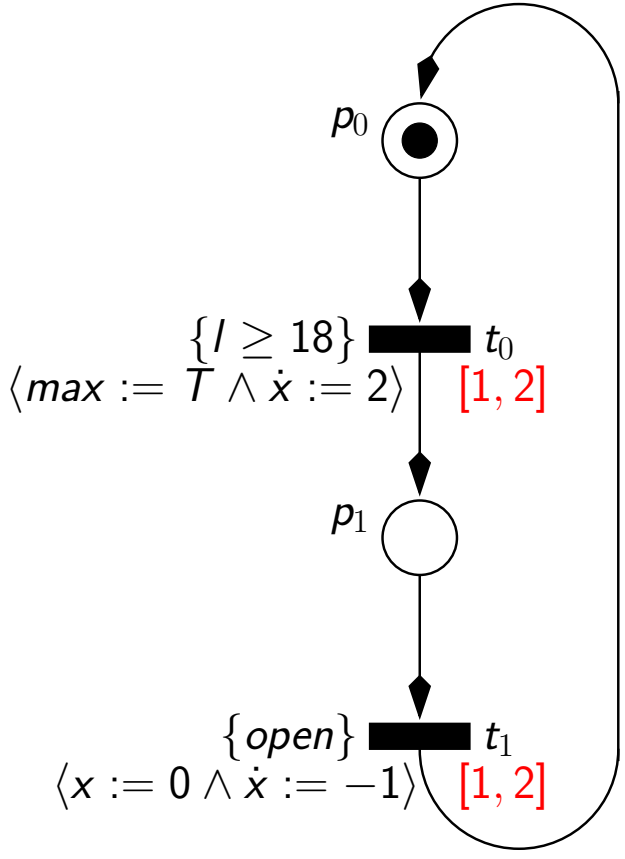
Labeled Hybrid Petri Nets (LHPNs)

- 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



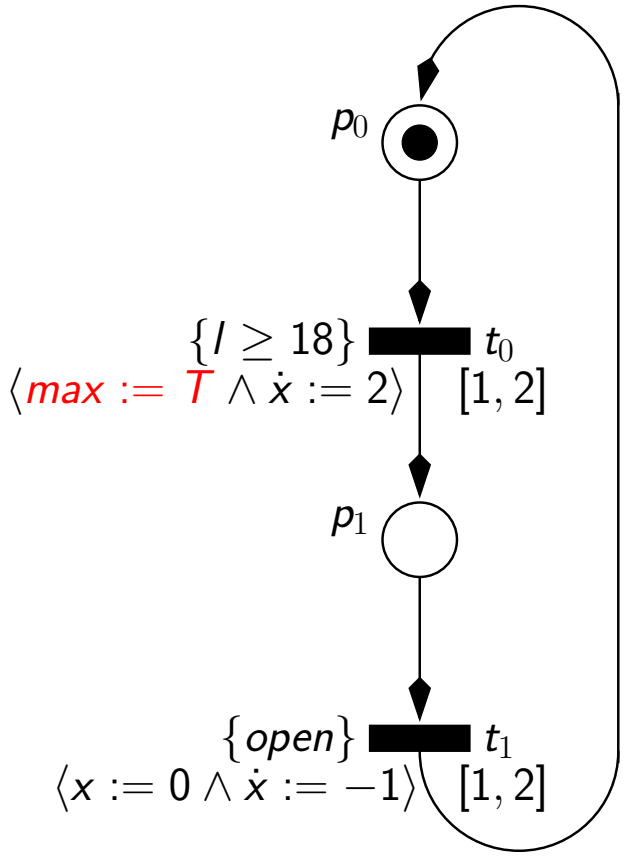
Labeled Hybrid Petri Nets (LHPNs)

- 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



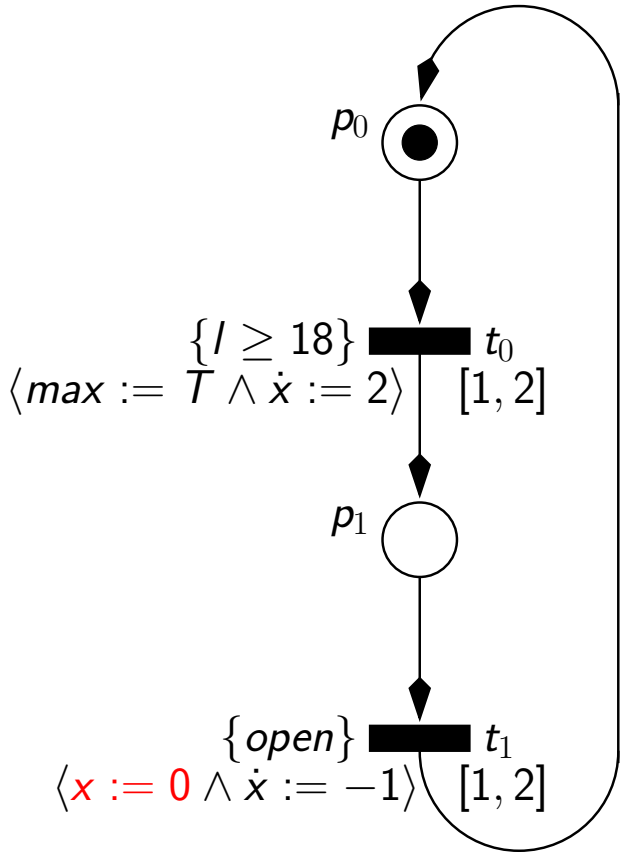
Labeled Hybrid Petri Nets (LHPNs)

- 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



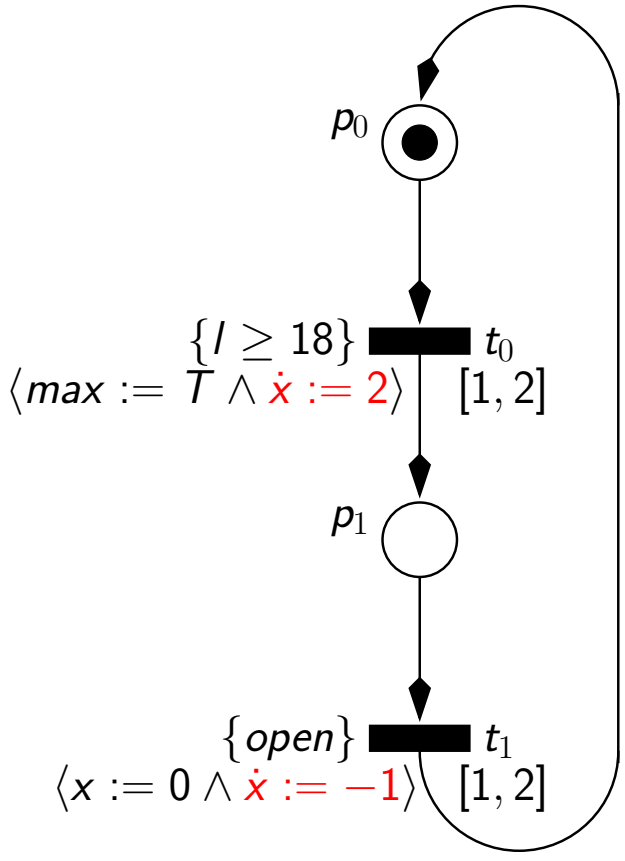
Labeled Hybrid Petri Nets (LHPNs)

- 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

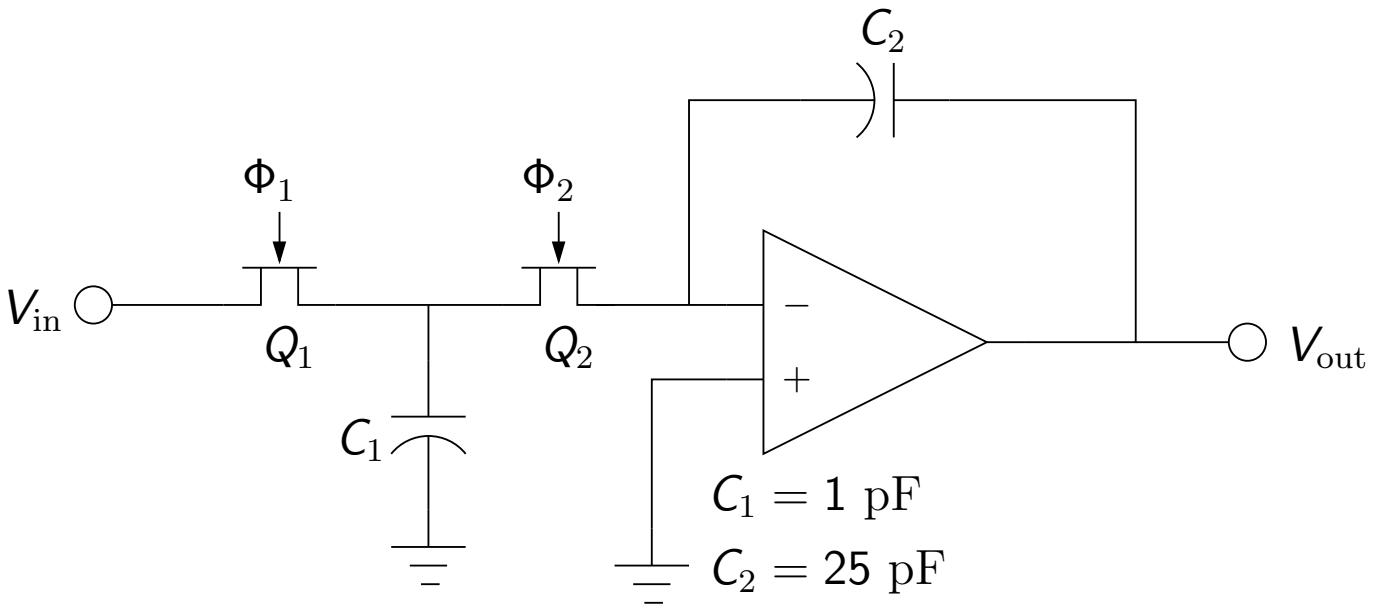


Labeled Hybrid Petri Nets (LHPNs)

- 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



Switched Capacitor Integrator Circuit



$C_1 = 1$ pF

$C_2 = 25$ pF

$freq(\Phi_1) = freq(\Phi_2) = 500$ kHz

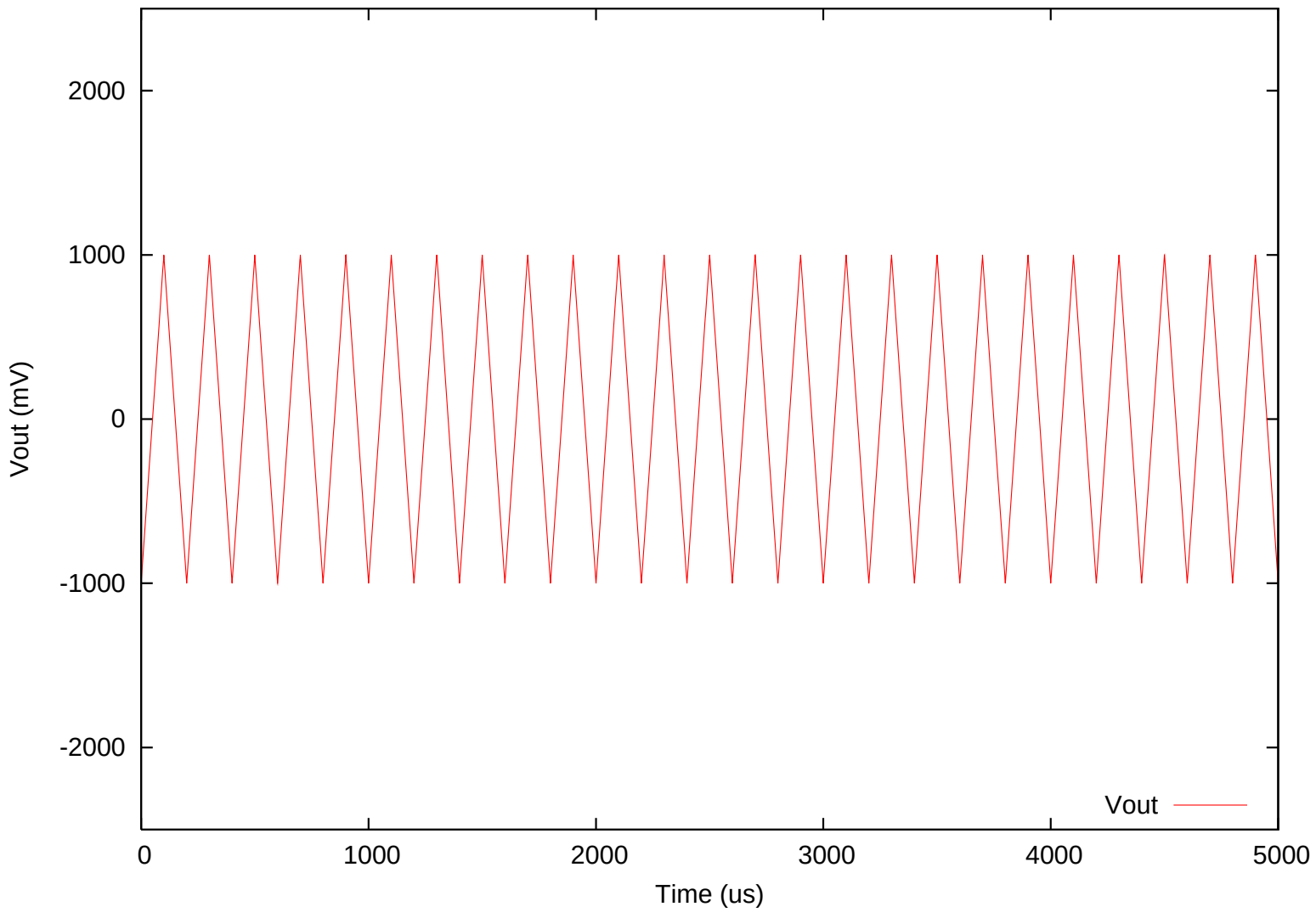
$dV_{\text{out}}/dt = \pm 20$ mV/μs

$V_{\text{in}} = \pm 1000$ mV

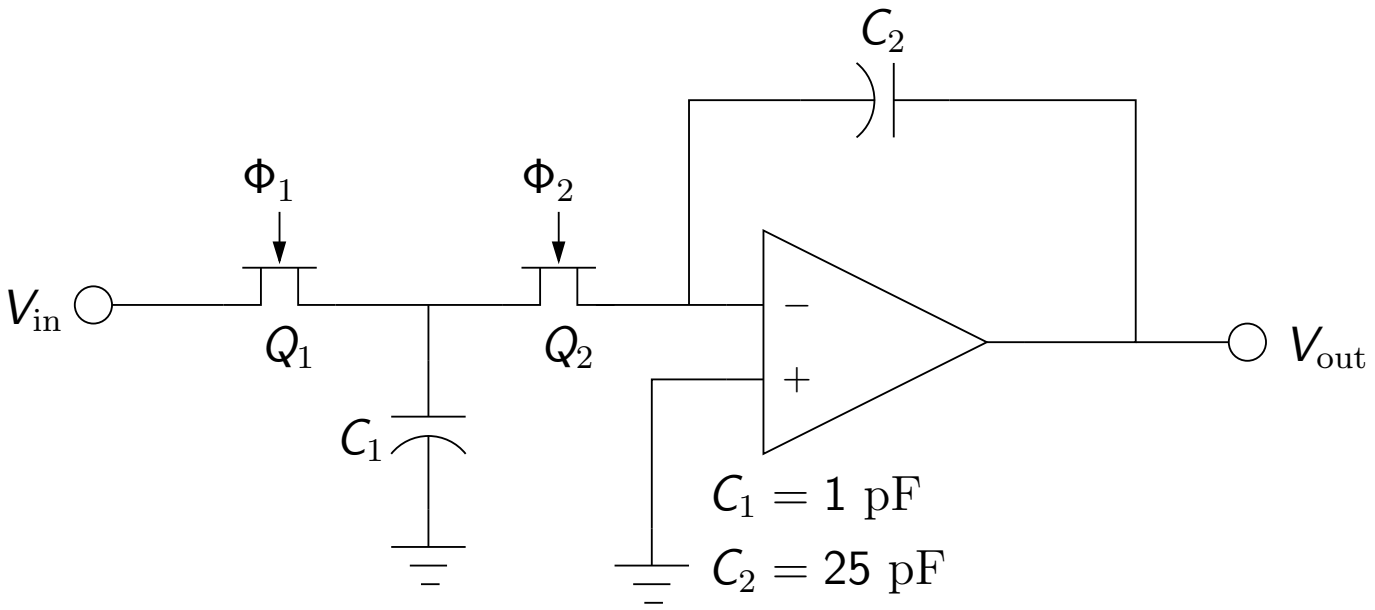
$freq(V_{\text{in}}) = 5$ kHz

Switched Capacitor Output Waveform

Typical Simulation



Switched Capacitor Integrator Circuit



$C_1 = 1$ pF

$C_2 = 25$ pF

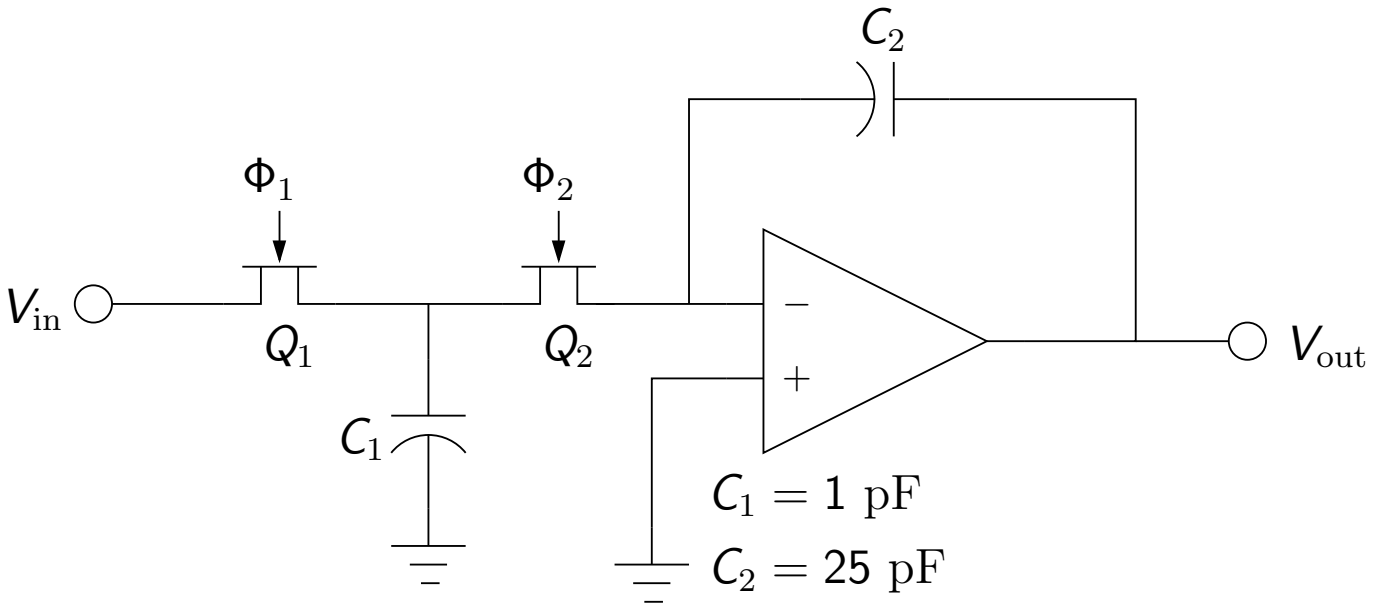
$V_{\text{in}} = \pm 1000$ mV

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

$freq(\Phi_1) = freq(\Phi_2) = 500$ kHz

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

Switched Capacitor Integrator Circuit



$C_1 = 1$ pF

$C_2 = 25$ pF

$V_{\text{in}} = \pm 1000$ mV

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

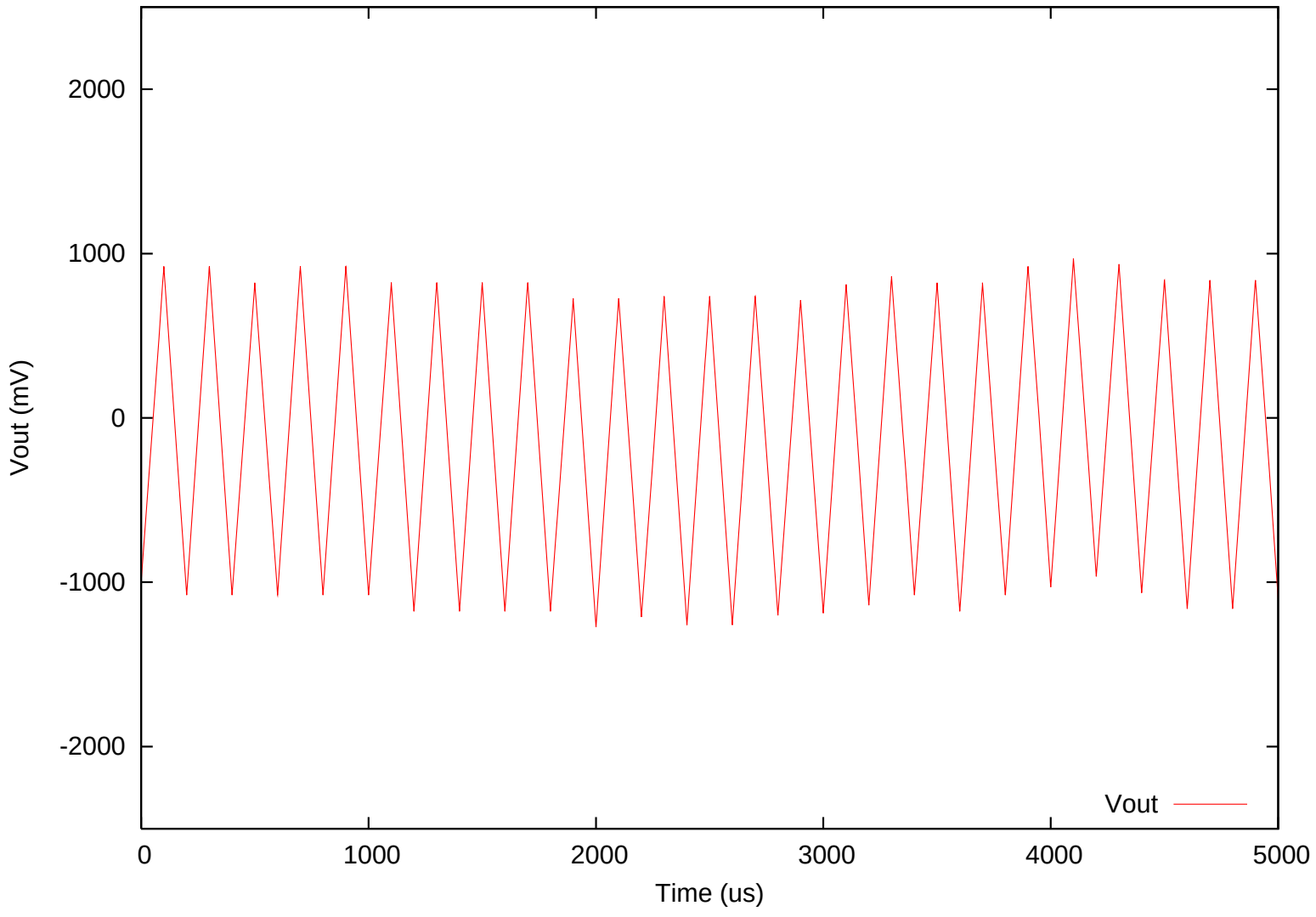
$freq(\Phi_1) = freq(\Phi_2) = 500$ kHz

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

Does V_{out} saturate? (i.e. $-2000mV \leq V_{out} \leq 2000mV$)

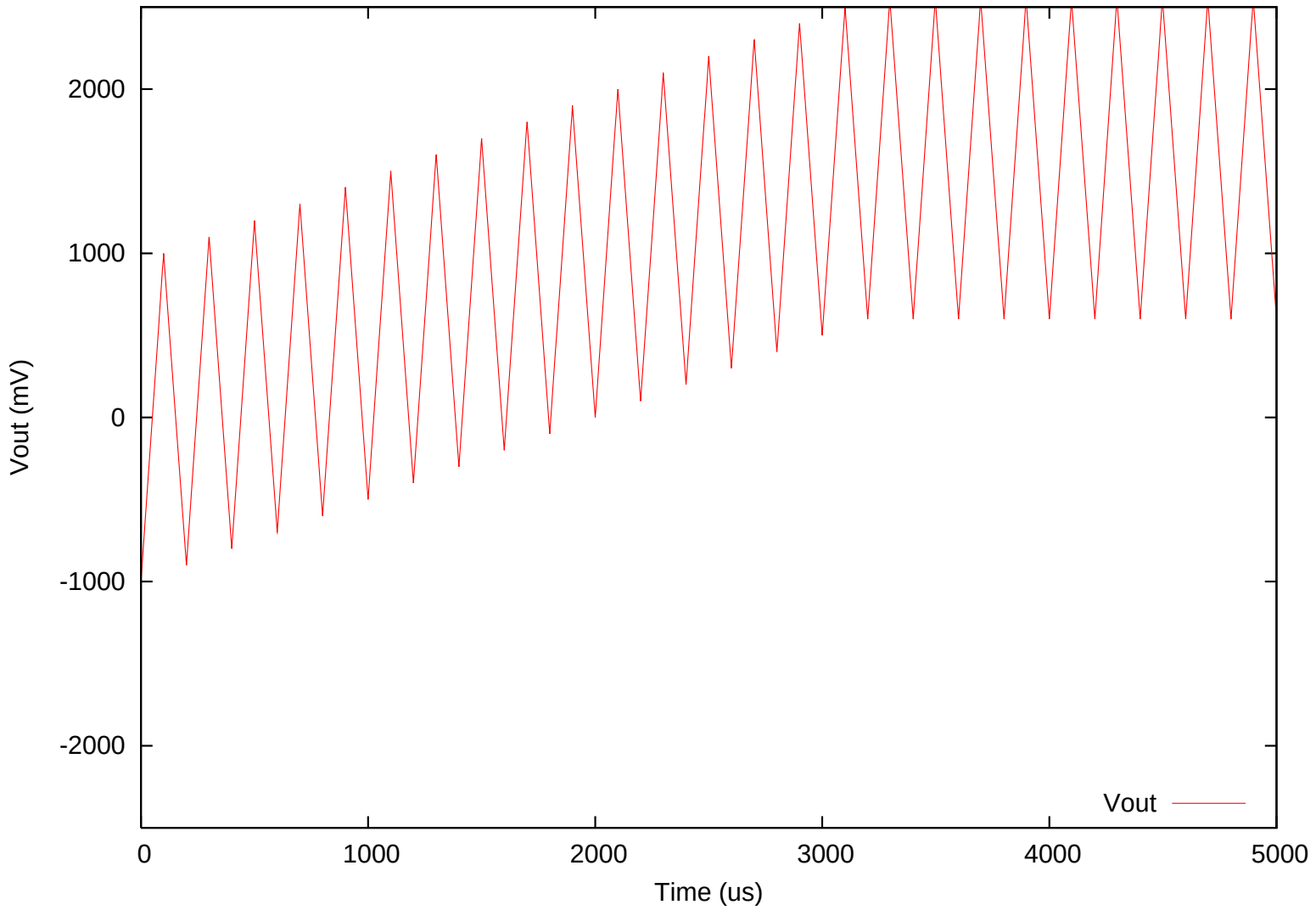
Switched Capacitor Output Waveform

Random Simulation



Switched Capacitor Output Waveform

Worst Case Simulation

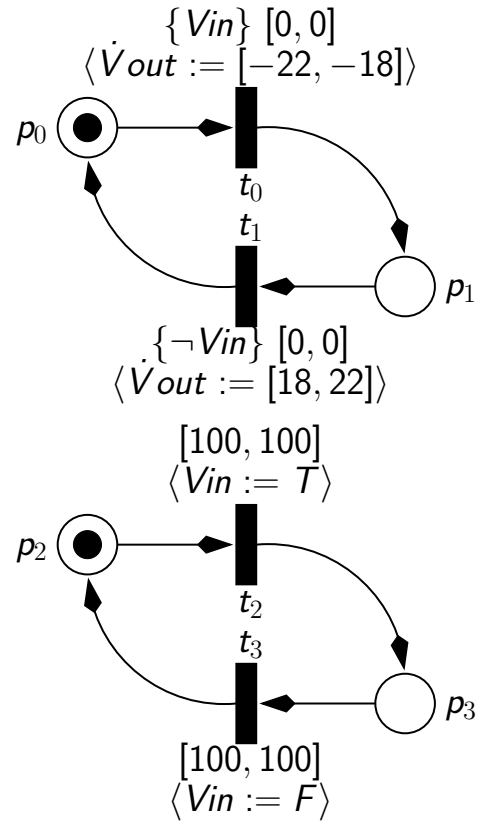


Switched Capacitor Integrator Model

```
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;
```

Switched Capacitor Integrator Model

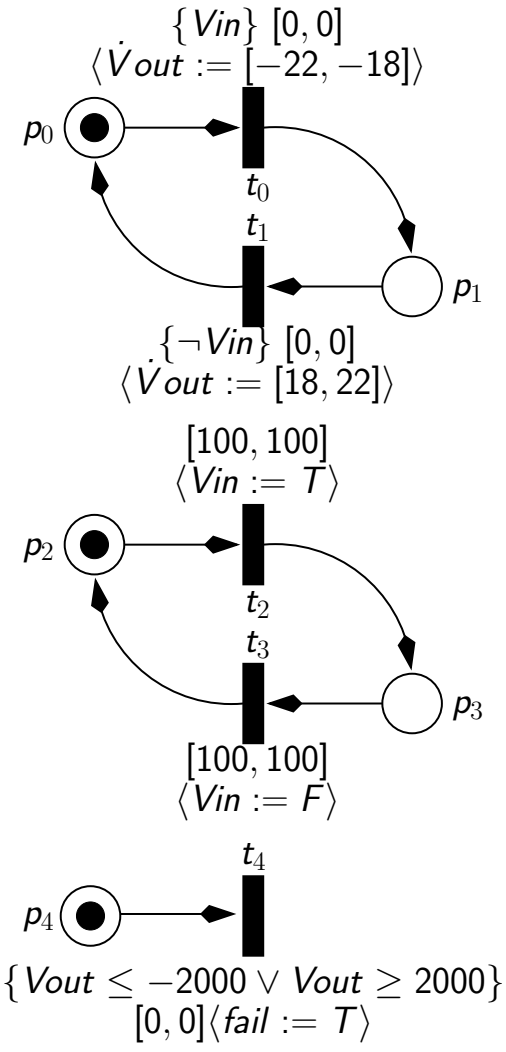
```
library IEEE;
use IEEE.std_logic_1164.all;
use work.handshake.all;
use work.nondeterminism.all;
entity integrator is
end integrator;
architecture switchCap of integrator is
  signal Vin:std_logic := '0';
  quantity Vout:real;
begin
  break Vout => -1000.0; --Initial value
  if Vin='0' use
    Vout'dot == 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;
```



$$Q_0 = \{Vout = -1000\}$$
$$R_0 = \{\dot{V}out = [18, 22]\}$$
$$S_0 = \{\neg Vin, \neg fail\}$$

Switched Capacitor Integrator Model

```
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;
```



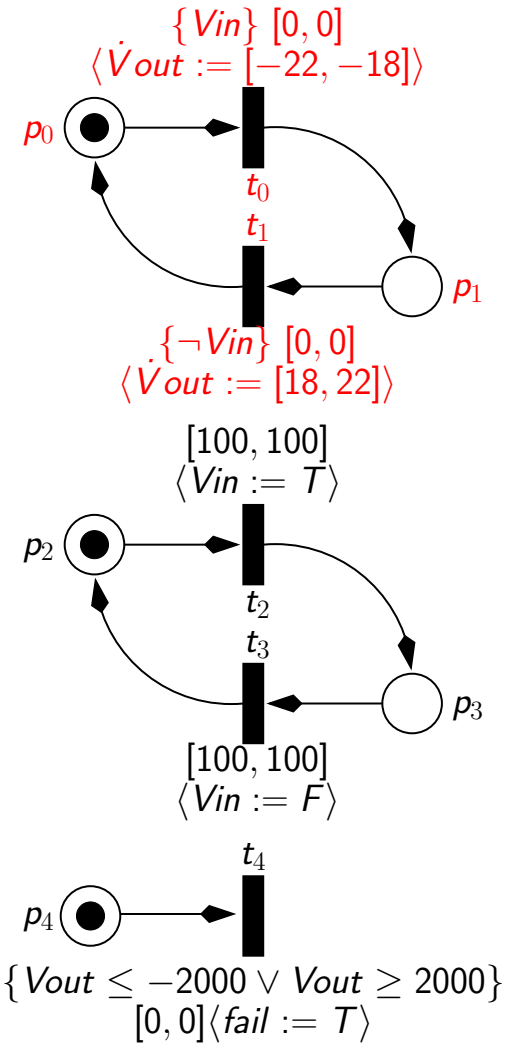
$Q_0 = \{Vout = -1000\}$

$R_0 = \{\dot{V}out = [18, 22]\}$

$S_0 = \{\neg Vin, \neg fail\}$

Switched Capacitor Integrator Model

```
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;
```



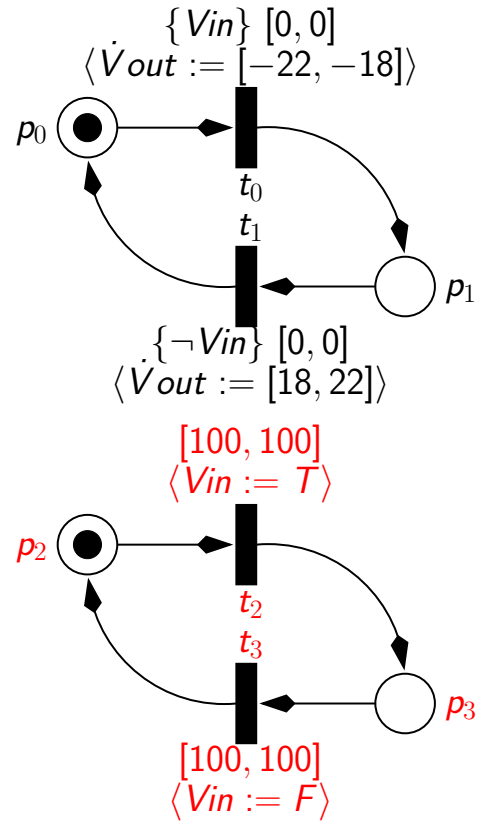
$Q_0 = \{Vout = -1000\}$

$R_0 = \{\dot{Vout} = [18, 22]\}$

$S_0 = \{\neg Vin, \neg fail\}$

Switched Capacitor Integrator Model

```
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;
```



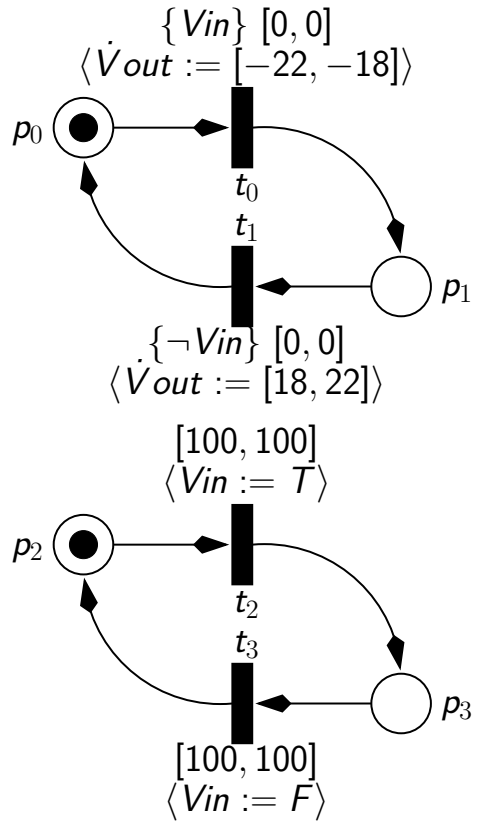
$Q_0 = \{Vout = -1000\}$

$R_0 = \{\dot{V}out = [18, 22]\}$

$S_0 = \{\neg Vin, \neg fail\}$

Switched Capacitor Integrator Model

```
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;
```

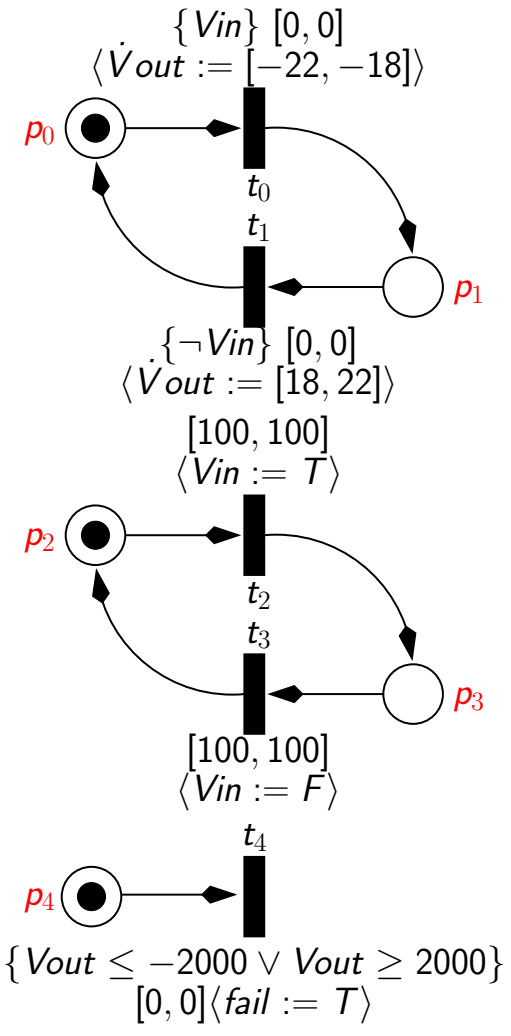


$$Q_0 = \{Vout = -1000\}$$
$$R_0 = \{\dot{V}out = [18, 22]\}$$
$$S_0 = \{\neg Vin, \neg fail\}$$

Boolean Representation

Create Boolean variables for each of the following:

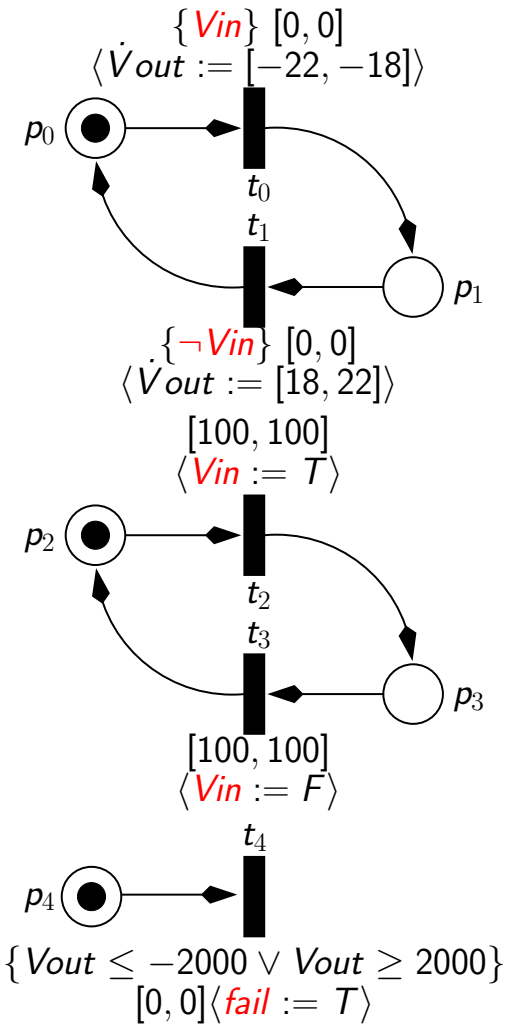
- Marking: $p_0, p_1, \ldots, p_4$
- Boolean signals: Vin, $fail$
- Boolean rate variables: $\dot{V}out_{[18,22]}, \dot{V}out_{[-22,-18]}$
- Clock active variables: $a_{t_0}, a_{t_1}, \ldots a_{t_4}$



Boolean Representation

Create Boolean variables for each of the following:

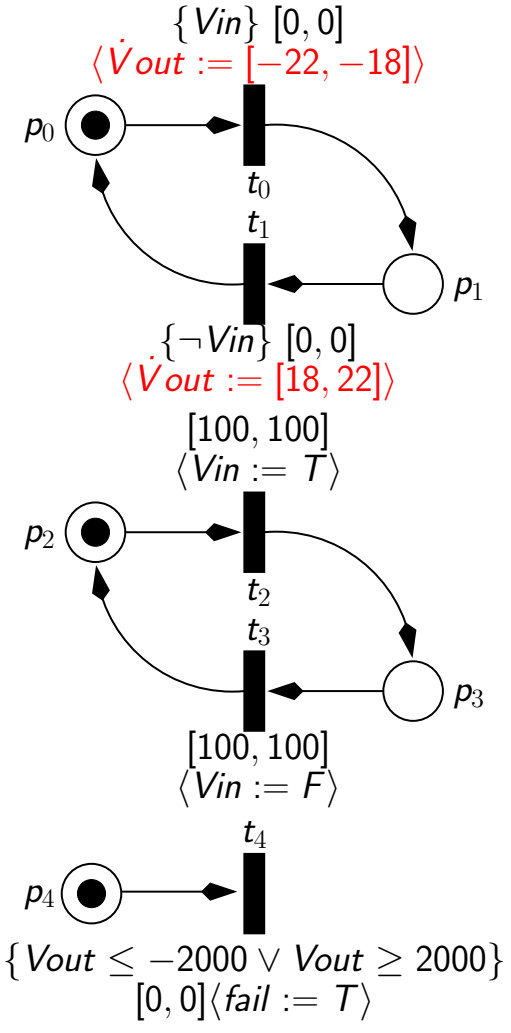
- Marking: $p_0, p_1, ..., p_4$
- Boolean signals: $Vin, fail$
- Boolean rate variables: $\dot{V}out_{[18,22]}, \dot{V}out_{[-22,-18]}$
- Clock active variables: $a_{t_0}, a_{t_1}, ...a_{t_4}$



Boolean Representation

Create Boolean variables for each of the following:

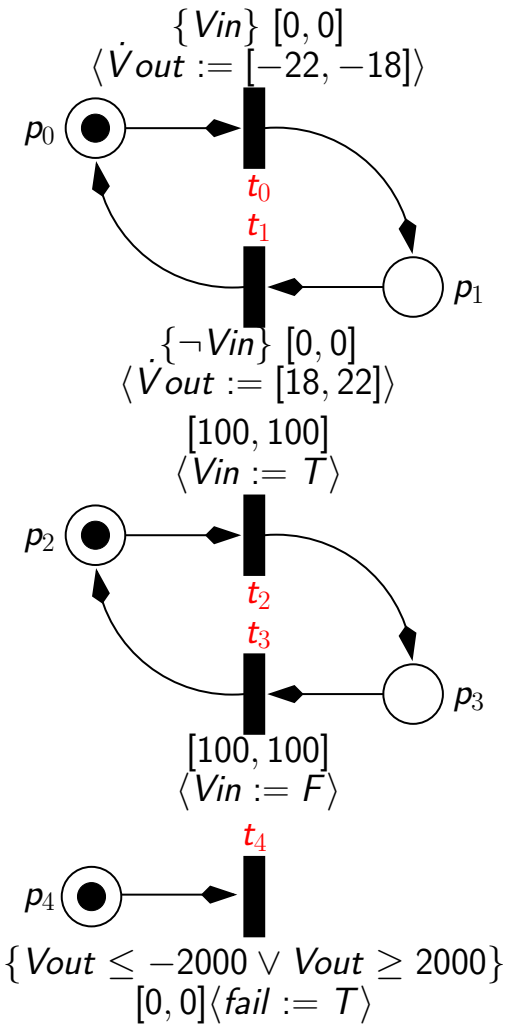
- Marking: $p_0, p_1, ..., p_4$
- Boolean signals: Vin, $fail$
- Boolean rate variables: $\dot{V}out_{[18,22]}, \dot{V}out_{[-22,-18]}$
- Clock active variables: $a_{t_0}, a_{t_1}, ...a_{t_4}$



Boolean Representation

Create Boolean variables for each of the following:

- Marking: $p_0, p_1, ..., p_4$
- Boolean signals: Vin, $fail$
- Boolean rate variables: $\dot{V}out_{[18,22]}, \dot{V}out_{[-22,-18]}$
- Clock active variables: $a_{t_0}, a_{t_1}, ...a_{t_4}$



Hybrid Separation Logic

- Maintain relationships among real values including system variables ($Vout$) 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 $\overline{c_j x_j \geq c_i x_i + -c}$.
- Hybrid Separation Logic (HSL):

$$\phi ::= \textbf{true} \mid \textbf{false} \mid b \mid \neg\phi \mid \phi \wedge \phi \mid c_i x_i \geq c_j x_j + c$$

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:

$$2x \geq 3y + 5 \wedge 4y \geq 5z + 5 \quad \Rightarrow$$

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:

$$\begin{aligned} & 2x \geq 3y + 5 \wedge 4y \geq 5z + 5 \quad \Rightarrow \\ & \frac{2}{3}x \geq y + \frac{5}{3} \wedge y \geq \frac{5}{4}z + \frac{5}{4} \quad \Rightarrow \end{aligned}$$

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:

$$\begin{aligned} & 2x \geq 3y + 5 \wedge 4y \geq 5z + 5 \quad \Rightarrow \\ & \frac{2}{3}x \geq y + \frac{5}{3} \wedge y \geq \frac{5}{4}z + \frac{5}{4} \quad \Rightarrow \quad \frac{2}{3}x \geq \frac{5}{4}z + \frac{5}{3} + \frac{5}{4} \end{aligned}$$

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:

$$2x > 3y + 5 \Rightarrow 2x \geq 3y + 5$$

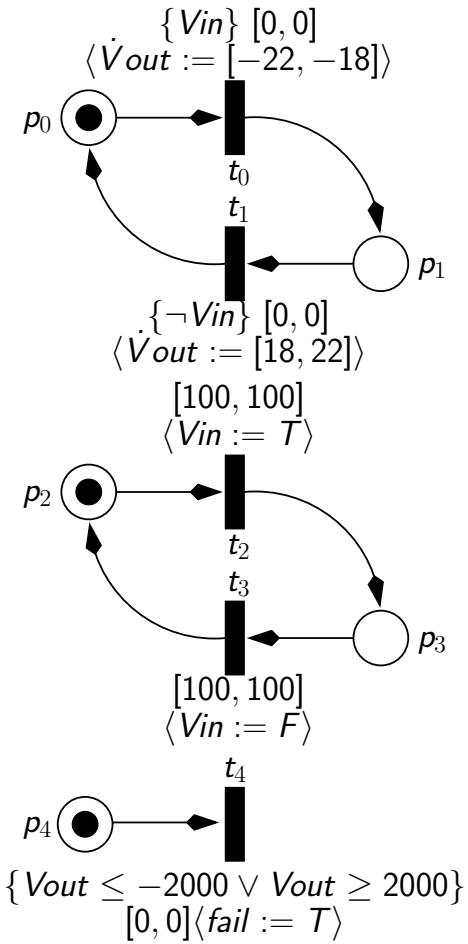
- Transitivity:

$$\begin{aligned} 2x \geq 3y + 5 \wedge 4y \geq 5z + 5 \quad &\Rightarrow \\ \frac{2}{3}x \geq y + \frac{5}{3} \wedge y \geq \frac{5}{4}z + \frac{5}{4} \quad &\Rightarrow \quad \frac{2}{3}x \geq \frac{5}{4}z + \frac{5}{3} + \frac{5}{4} \\ &\Rightarrow \quad 8x \geq 15z + 35 \end{aligned}$$

Invariant

A statement that must always be satisfied.

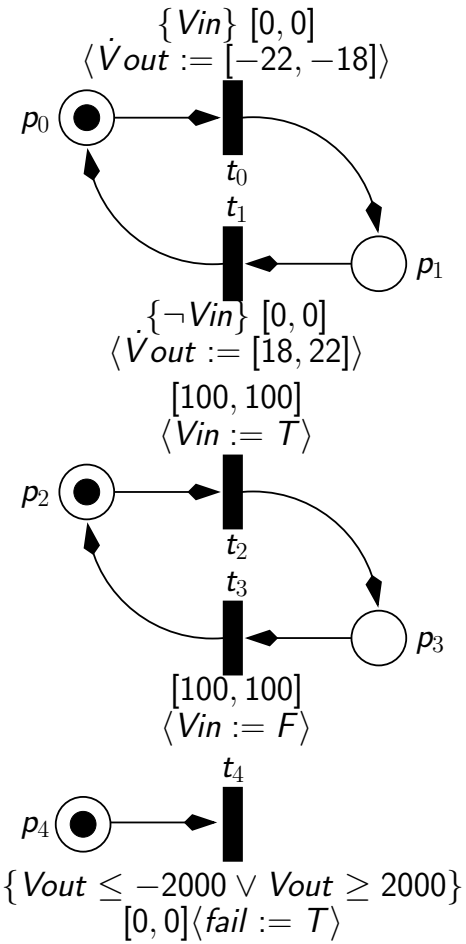
$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

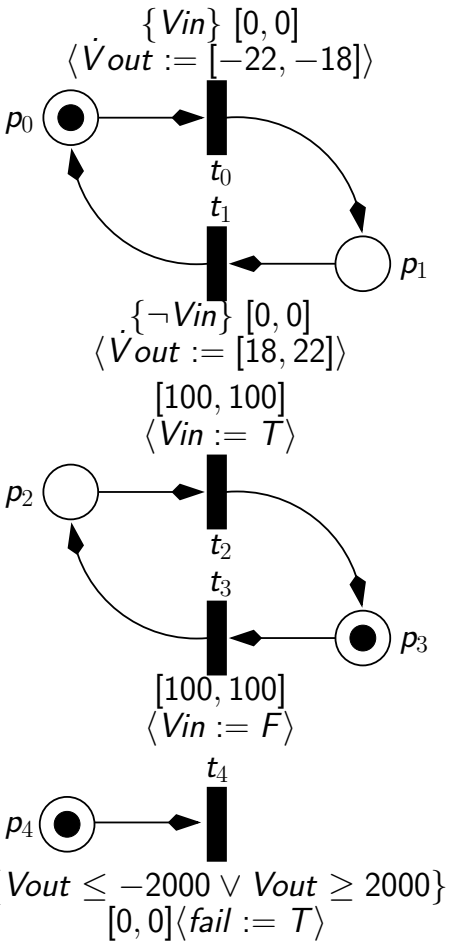


$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$
$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}}$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

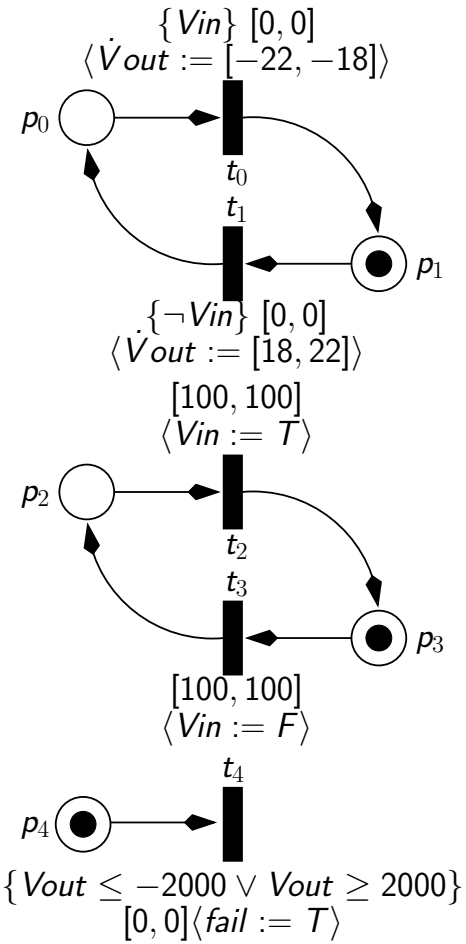
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$

$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}}$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

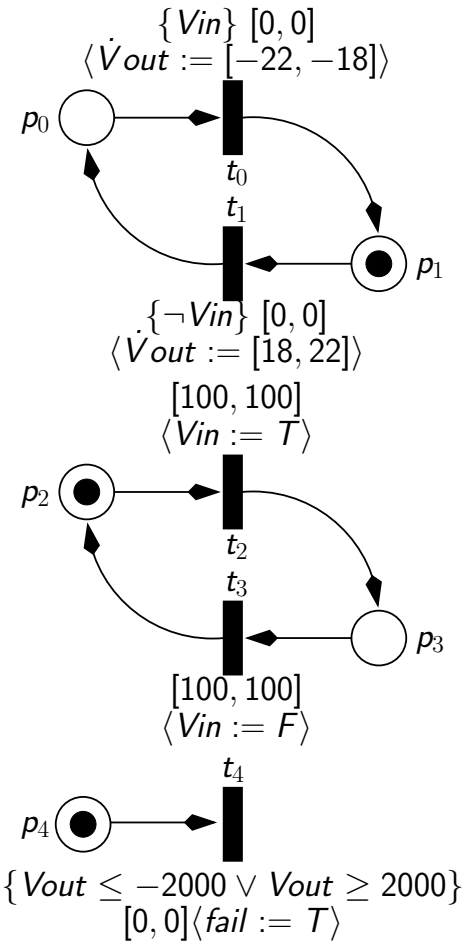
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$

$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}}$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$

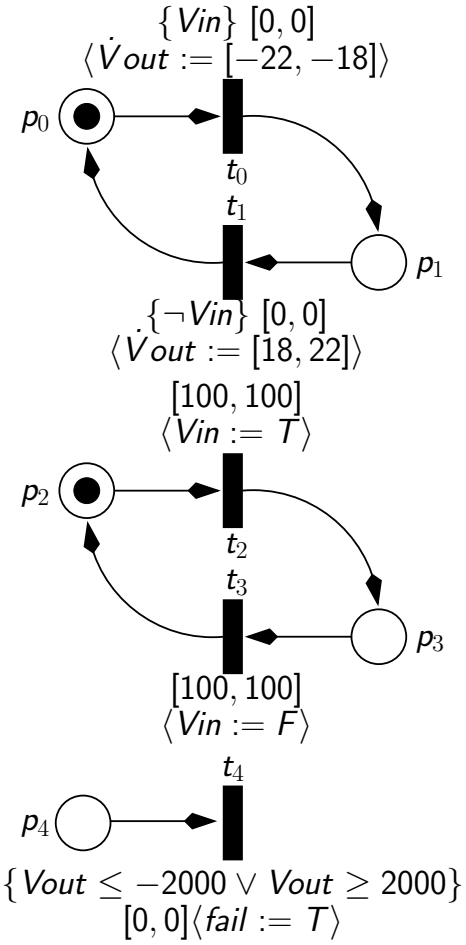
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$

$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}}$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

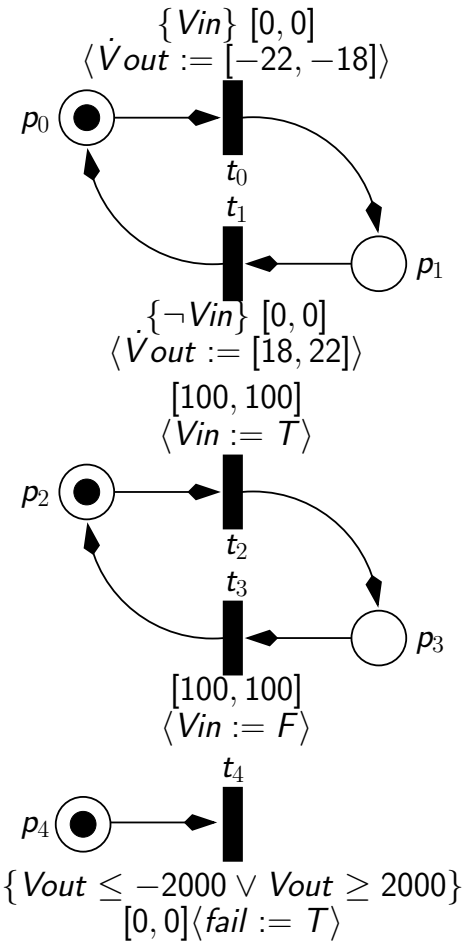


$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ p_4\ Vin\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$
$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ p_4\ \overline{Vin}\ \overline{fail}\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$
$$p_0\ \overline{p_1}\ p_2\ \overline{p_3}\ \overline{p_4}\ \overline{Vin}\ fail\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$p_0\ \overline{p_1}\ \overline{p_2}\ p_3\ \overline{p_4}\ Vin\ fail\ \overline{\dot{V}out_{[-22,-18]}}\ \dot{V}out_{[18,22]} \vee$$
$$\overline{p_0}\ p_1\ \overline{p_2}\ p_3\ \overline{p_4}\ Vin\ fail\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}} \vee$$
$$\overline{p_0}\ p_1\ p_2\ \overline{p_3}\ \overline{p_4}\ \overline{Vin}\ fail\ \dot{V}out_{[-22,-18]}\ \overline{\dot{V}out_{[18,22]}}$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



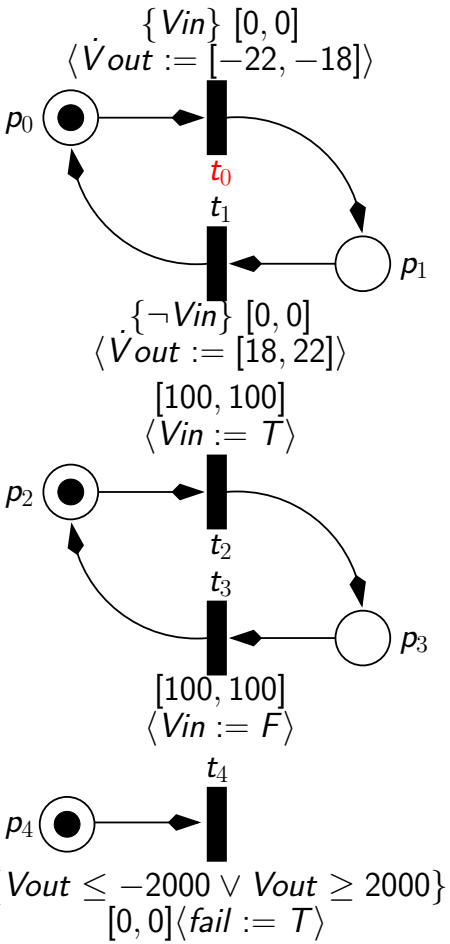
Pseudo-Inverse:

$$a \wedge \widetilde{x \leq 2000} = \overline{a} \vee x \geq 2000$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

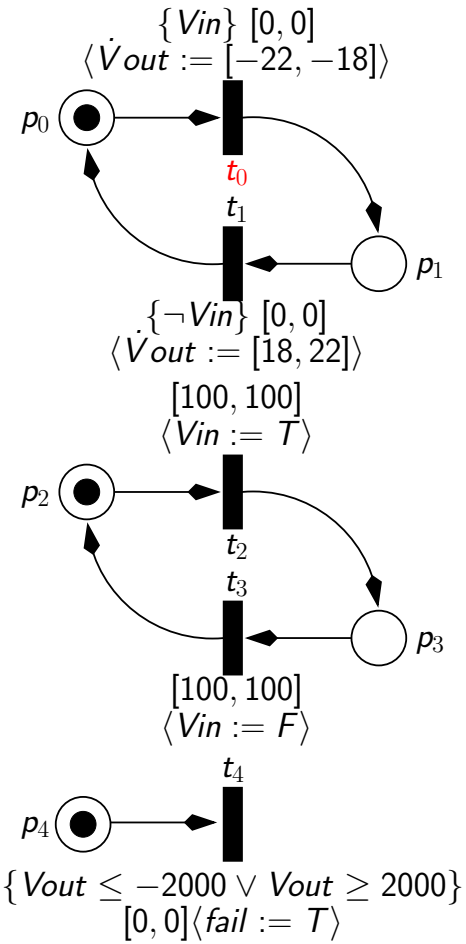


$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin})$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

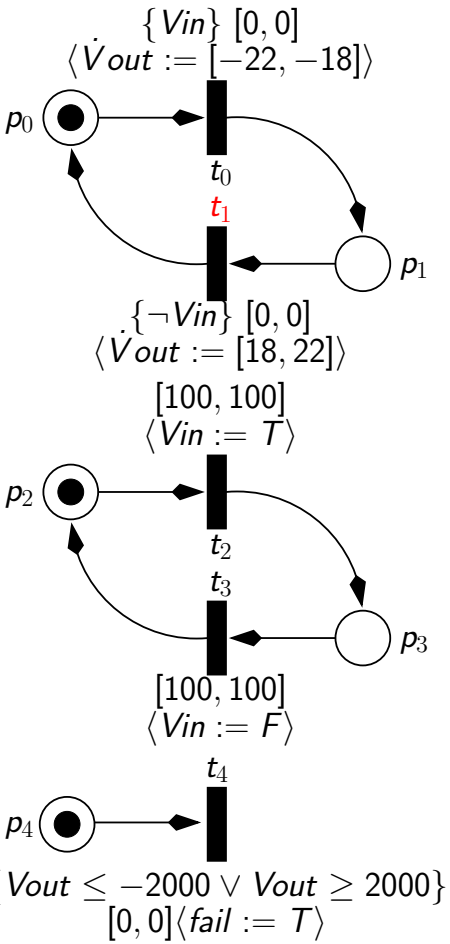


$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin})$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

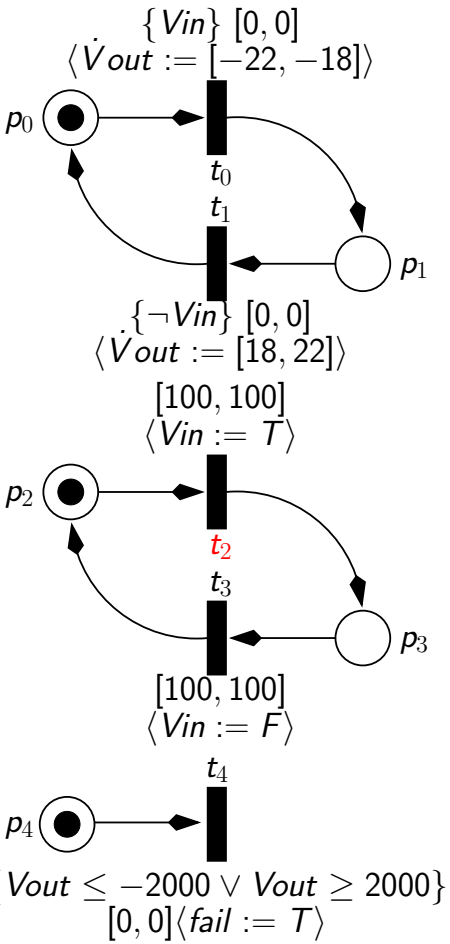


$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin}) \wedge$$
$$(a_{t_1} \Rightarrow p_1 \wedge \overline{Vin} \wedge c_{t_1} = 0) \wedge (\overline{a_{t_1}} \Rightarrow \overline{p_1} \vee Vin)$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

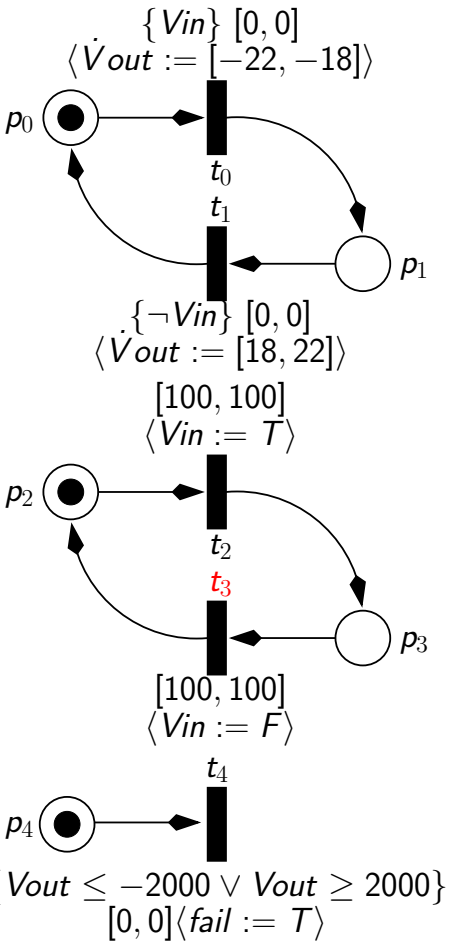


$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin}) \wedge$$
$$(a_{t_1} \Rightarrow p_1 \wedge \overline{Vin} \wedge c_{t_1} = 0) \wedge (\overline{a_{t_1}} \Rightarrow \overline{p_1} \vee Vin) \wedge$$
$$(a_{t_2} \Rightarrow p_2 \wedge 0 \leq c_{t_2} \leq 100) \wedge (\overline{a_{t_2}} \Rightarrow \overline{p_2})$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$

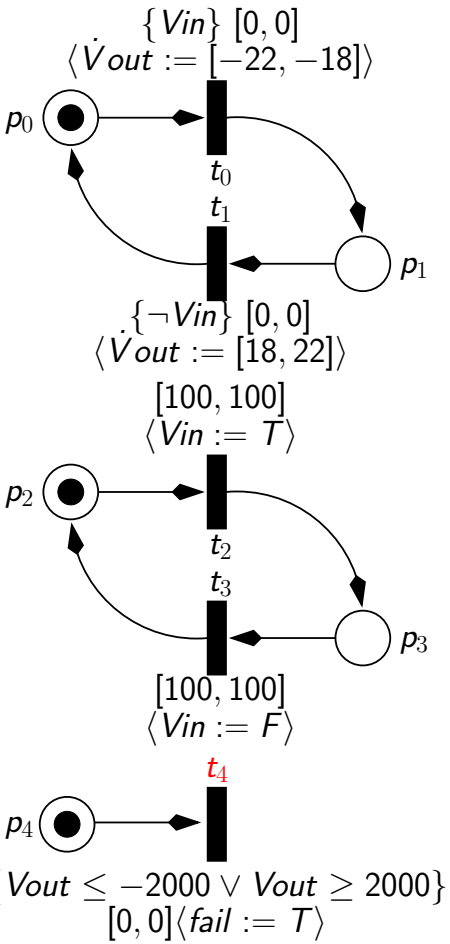


$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin}) \wedge$$
$$(a_{t_1} \Rightarrow p_1 \wedge \overline{Vin} \wedge c_{t_1} = 0) \wedge (\overline{a_{t_1}} \Rightarrow \overline{p_1} \vee Vin) \wedge$$
$$(a_{t_2} \Rightarrow p_2 \wedge 0 \leq c_{t_2} \leq 100) \wedge (\overline{a_{t_2}} \Rightarrow \overline{p_2}) \wedge$$
$$(a_{t_3} \Rightarrow p_3 \wedge 0 \leq c_{t_3} \leq 100) \wedge (\overline{a_{t_3}} \Rightarrow \overline{p_3})$$

Invariant

A statement that must always be satisfied.

$$\phi_{\mathcal{I}} = \Phi \wedge \bigwedge_{t \in T} (a_t \Rightarrow \bullet t \wedge E(t) \wedge 0 \leq c_t \leq u(t)) \wedge (\overline{a_t} \Rightarrow \overline{\bullet t} \vee \widetilde{E(t)})$$



$$(a_{t_0} \Rightarrow p_0 \wedge Vin \wedge c_{t_0} = 0) \wedge (\overline{a_{t_0}} \Rightarrow \overline{p_0} \vee \overline{Vin}) \wedge$$
$$(a_{t_1} \Rightarrow p_1 \wedge \overline{Vin} \wedge c_{t_1} = 0) \wedge (\overline{a_{t_1}} \Rightarrow \overline{p_1} \vee Vin) \wedge$$
$$(a_{t_2} \Rightarrow p_2 \wedge 0 \leq c_{t_2} \leq 100) \wedge (\overline{a_{t_2}} \Rightarrow \overline{p_2}) \wedge$$
$$(a_{t_3} \Rightarrow p_3 \wedge 0 \leq c_{t_3} \leq 100) \wedge (\overline{a_{t_3}} \Rightarrow \overline{p_3}) \wedge$$
$$(a_{t_4} \Rightarrow p_4 \wedge c_{t_4} = 0 \wedge$$
$$(Vout \leq -2000 \vee Vout \geq 2000)) \wedge$$
$$(\overline{a_{t_4}} \Rightarrow \overline{p_4} \vee (Vout \geq -2000 \wedge Vout \leq 2000))$$

Primary Guarded Command Set

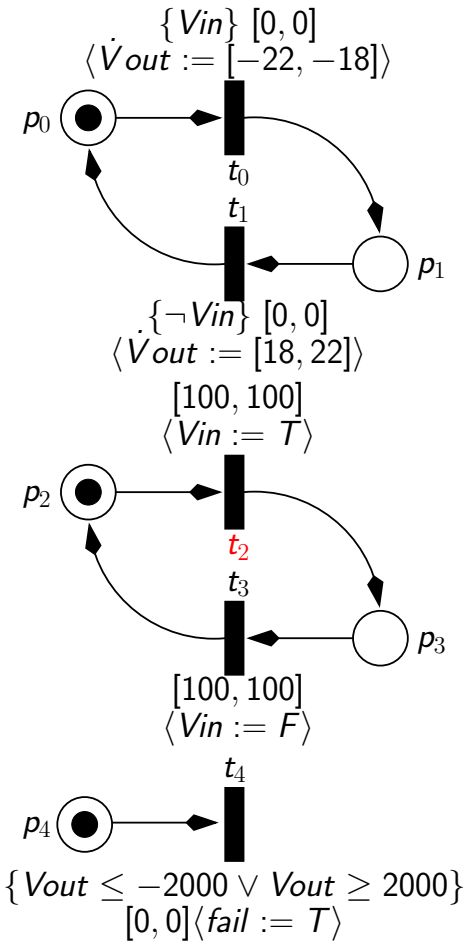
- Fires transition causing marking update and assignments.
- For each transition t_i:

$$\mathcal{C}_P = \bigcup_{t \in T} \{\langle \phi_{G_P}(t), \mathcal{A}_P(t) \rangle\}$$

where

$$\begin{aligned} \phi_{G_P}(t) &= (\bullet t \wedge \overline{t\bullet - \bullet t} \wedge E(t) \wedge a_t \wedge c_t \geq I(t)) \\ \mathcal{A}_P(t) &= \{(\bullet t - t\bullet) := F, (t\bullet) := T, a_t := F, c_t := [-\infty, \infty], \\ &\quad BA(t), VA(t), RA(t)\} \end{aligned}$$

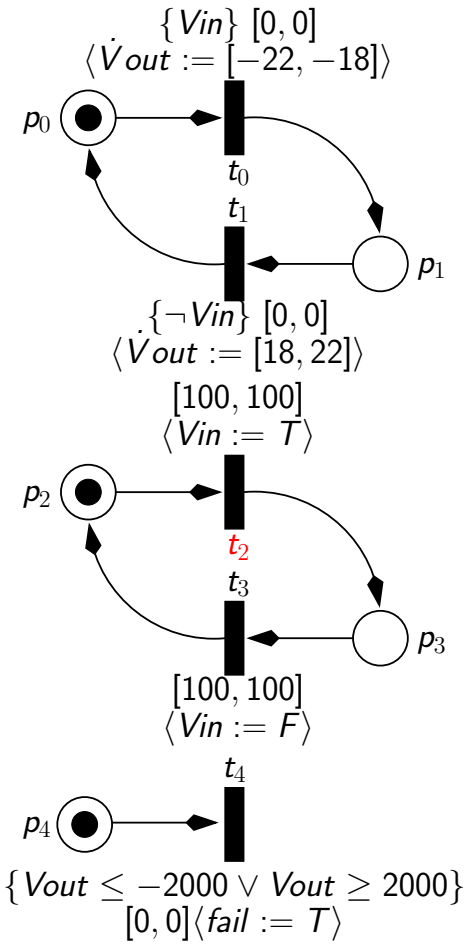
Primary Guarded Command for Integrator Example



$$\phi_{G_P}(t_2) = p_2 \wedge \overline{p_3} \wedge a_{t_2} \wedge c_{t_2} \geq 100$$

$$\mathcal{A}_P(t_2) =$$

Primary Guarded Command for Integrator Example



$$\phi_{G_P}(t_2) = p_2 \wedge \overline{p_3} \wedge a_{t_2} \wedge c_{t_2} \geq 100$$

$$\mathcal{A}_P(t_2) = \{p_2 := F, p_3 := T, a_{t_2} := F, c_{t_2} := [-\infty, \infty], Vin := T\}$$

Secondary Guarded Command Set

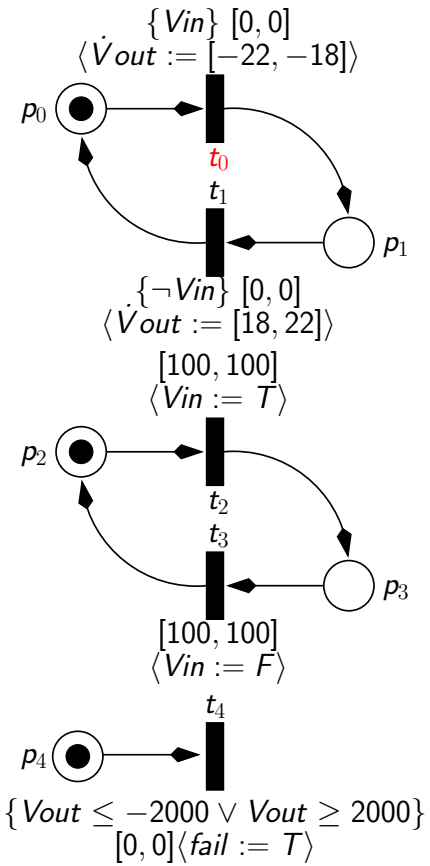
- Activates or deactivates clock based on marking, enabling condition, and clock value.
- For each transition t_i:

$$\mathcal{C}_S = \bigcup_{t \in T} \{\langle \phi_{G_{SA}}(t), \mathcal{A}_{SA}(t) \rangle, \langle \phi_{G_{SD}}(t), \mathcal{A}_{SD}(t) \rangle\}$$

where

$$\begin{aligned}
\phi_{G_{SA}}(t) &= \bullet t \wedge E(t) \wedge \overline{a_t} \\
\mathcal{A}_{SA}(t) &= \{a_t := T, c_t := [0, 0]\} \\
\phi_{G_{SD}}(t) &= (\overline{\bullet t} \vee \widetilde{E(t)}) \wedge a_t \\
\mathcal{A}_{SD}(t) &= \{a_t := F, c_t := [-\infty, \infty]\}
\end{aligned}$$

Secondary Guarded Commands for Integrator Example



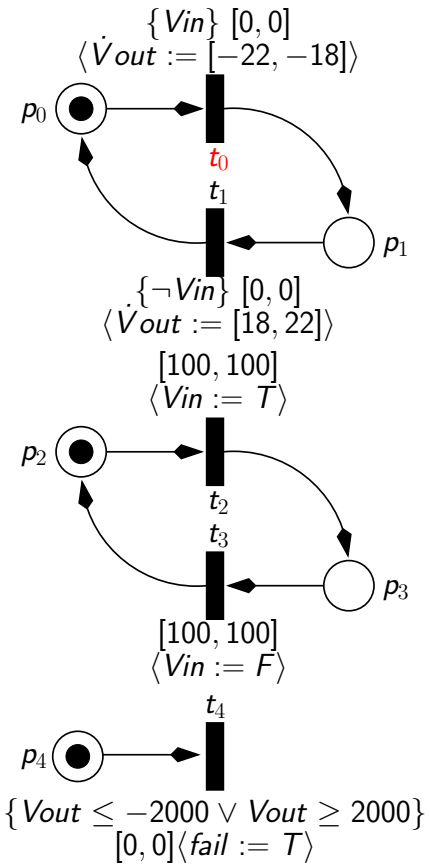
Activating Guarded Command:

$$\phi_{G_{SA}}(t_0) = p_0 \wedge Vin \wedge \overline{a_{t_0}}$$
$$\mathcal{A}_{SA}(t_0) =$$

Deactivating Guarded Command:

$$\phi_{G_{SD}}(t_0) =$$
$$\mathcal{A}_{SD}(t_0) =$$

Secondary Guarded Commands for Integrator Example



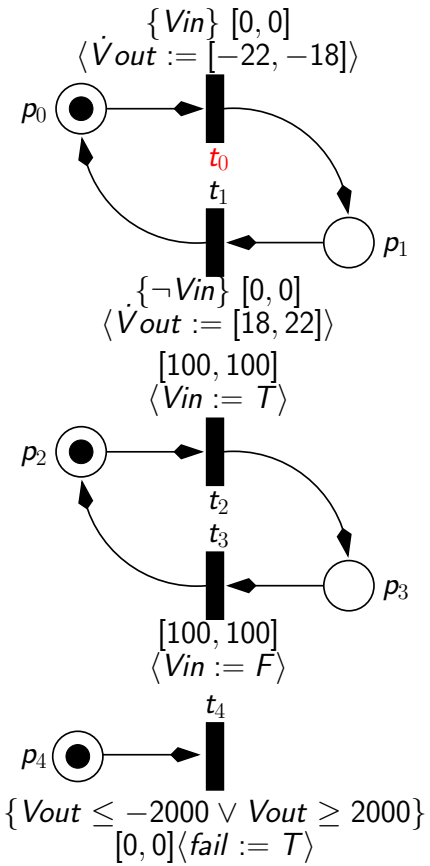
Activating Guarded Command:

$$\phi_{G_{SA}}(t_0) = p_0 \wedge Vin \wedge \overline{a_{t_0}}$$
$$\mathcal{A}_{SA}(t_0) = \{a_{t_0} := T, c_{t_0} := [0,0]\}$$

Deactivating Guarded Command:

$$\phi_{G_{SD}}(t_0) =$$
$$\mathcal{A}_{SD}(t_0) =$$

Secondary Guarded Commands for Integrator Example



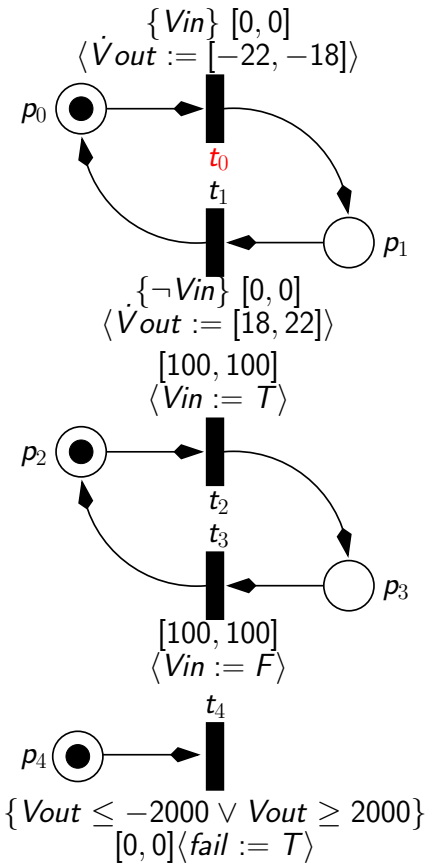
Activating Guarded Command:

$$\begin{aligned} \phi_{G_{SA}}(t_0) &= p_0 \wedge Vin \wedge \overline{a_{t_0}} \\ \mathcal{A}_{SA}(t_0) &= \{a_{t_0} := T, c_{t_0} := [0,0]\} \end{aligned}$$

Deactivating Guarded Command:

$$\begin{aligned} \phi_{G_{SD}}(t_0) &= (\overline{p_0} \vee \overline{Vin}) \wedge a_{t_0} \\ \mathcal{A}_{SD}(t_0) &= \end{aligned}$$

Secondary Guarded Commands for Integrator Example



Activating Guarded Command:

$$\begin{aligned} \phi_{G_{SA}}(t_0) &= p_0 \wedge Vin \wedge \overline{a_{t_0}} \\ \mathcal{A}_{SA}(t_0) &= \{a_{t_0} := T, c_{t_0} := [0,0]\} \end{aligned}$$

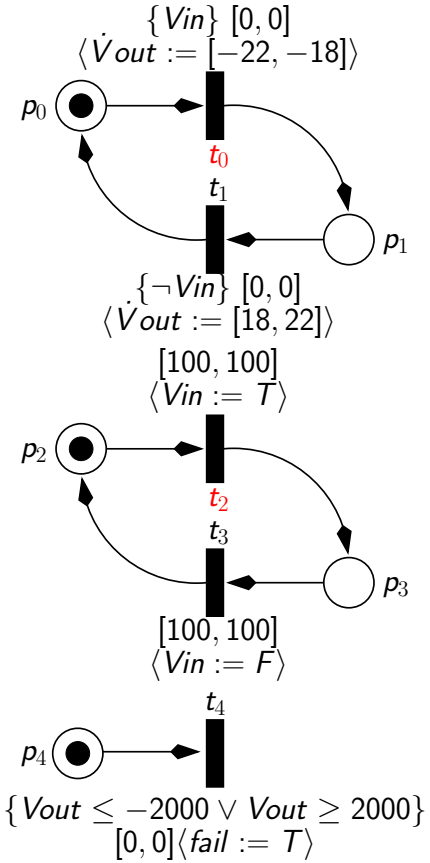
Deactivating Guarded Command:

$$\begin{aligned} \phi_{G_{SD}}(t_0) &= (\overline{p_0} \vee \overline{Vin}) \wedge a_{t_0} \\ \mathcal{A}_{SD}(t_0) &= \{a_{t_0} := F, c_{t_0} := [-\infty, \infty]\} \end{aligned}$$

Merging Primary and Secondary Guarded Commands

- 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



$$
\begin{aligned}
\phi_G(t_2, t_0) &= p_0 \wedge p_2 \wedge \overline{p_3} \wedge \overline{a_{t_0}} \wedge a_{t_2} \wedge \\
&\quad\; c_{t_2} \geq 100 \\
\mathcal{A}(t_2, t_0) &= \{p_2 := F, p_3 := T, Vin := T, \\
&\quad\; a_{t_0} := T, c_{t_0} := [0, 0], \\
&\quad\; a_{t_2} := F, c_{t_2} := [-\infty, \infty]\}
\end{aligned}
$$

Specifying Properties

- Dense real-time CTL is translated to a $\mathbf{T}\mu$ calculus formula.
- In the integrator example, the $\mathbf{T}\mu$ property is

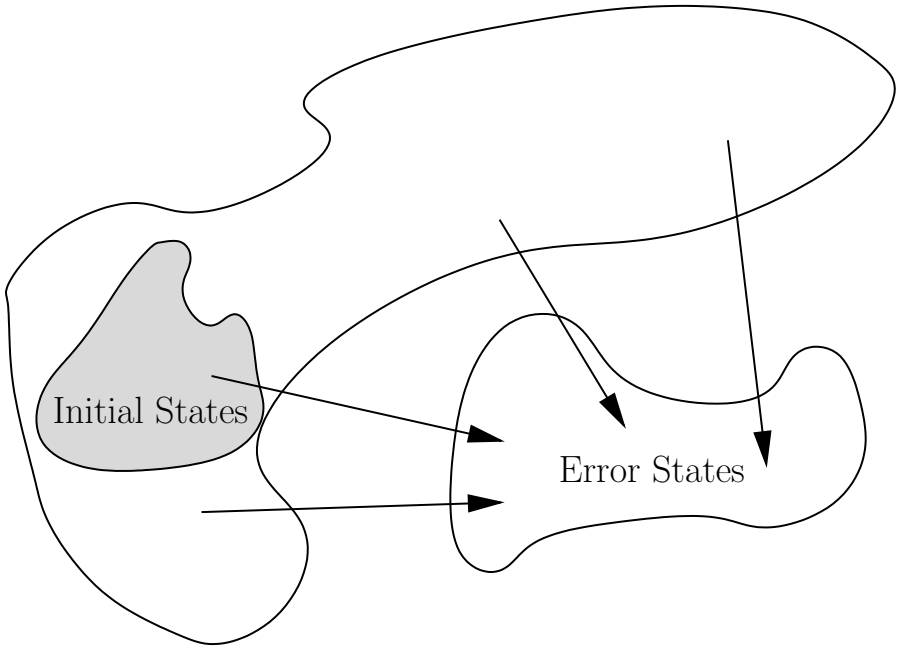
$$\phi_{init} \Longrightarrow \neg \mu Y.[fail \vee (\mathbf{true} \triangleright Y)]$$

where ϕ_{init}, the initial state, is:

$$\begin{aligned} \phi_{init} \quad = \quad & p_0\, \overline{p_1}\, p_2\, \overline{p_3}\, p_4\, \overline{Vin}\, \overline{Fail}\, \overline{\dot{V}out_{[-22,-18]}}\, \dot{V}out_{[18,22]} \\ & \overline{a_{t_0}}\, \overline{a_{t_1}}\, a_{t_2}\, \overline{a_{t_3}}\, \overline{a_{t_4}} \wedge c_{t_2} = 0 \wedge Vout = -1000 \end{aligned}$$

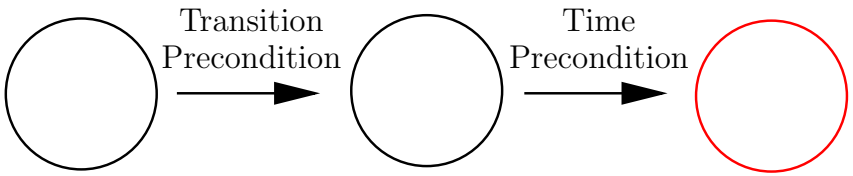
- If a state in which *fail* is true cannot be reached from the initial state, then the formula evaluates to true.

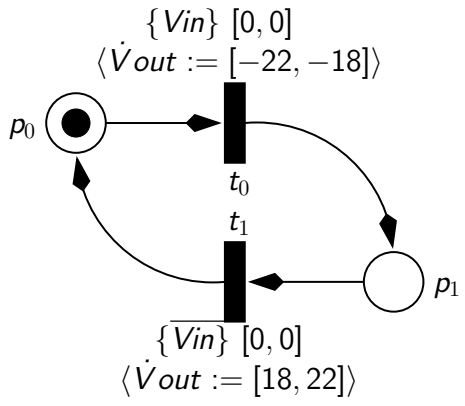
Model Checking Algorithm



- 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.

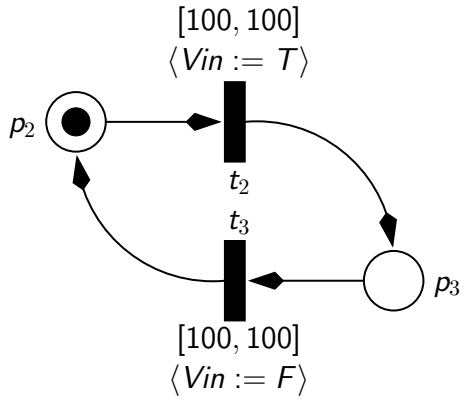
Time Precondition



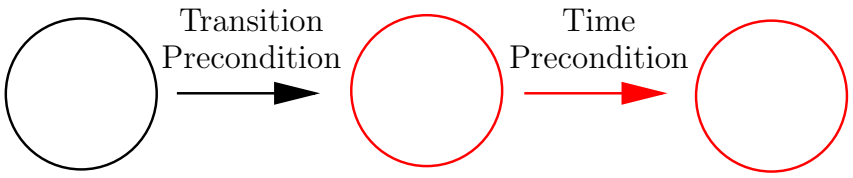


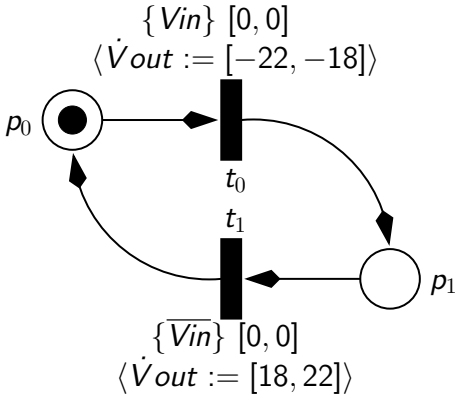
$Vout = 2000$

$c_{t_2} = 100$

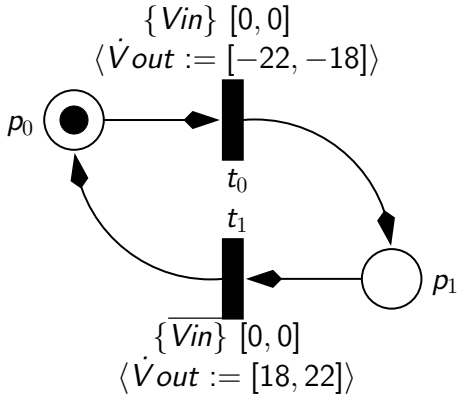


Time Precondition

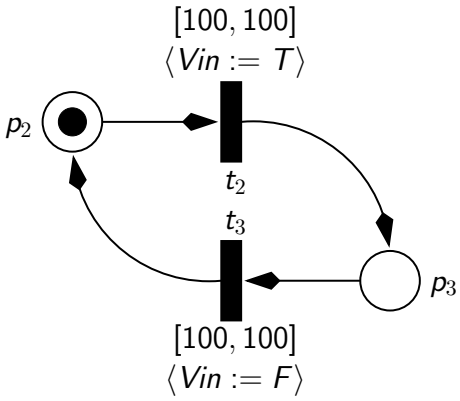


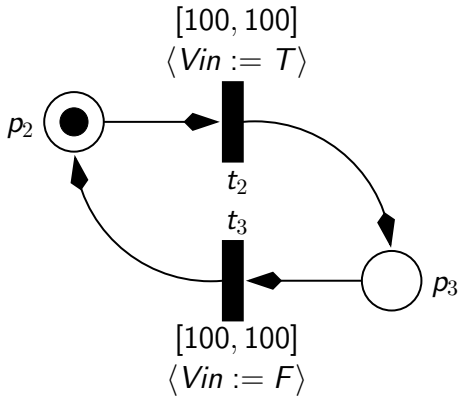


$-200 \leq Vout \leq 2000$
$0 \leq c_{t_2} \leq 100$
$Vout \geq 22c_{t_2} - 200$
$Vout \leq 18c_{t_2} + 200$

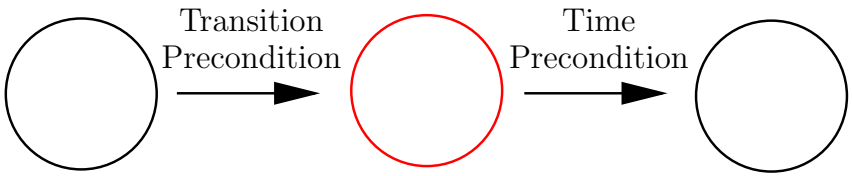


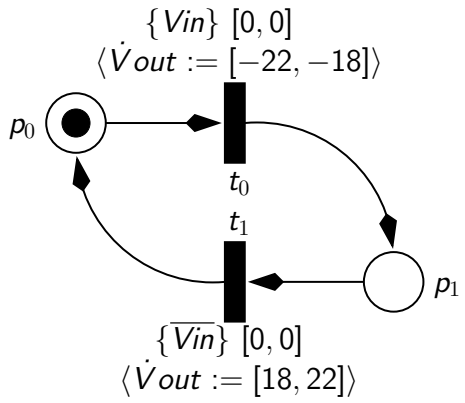
$Vout = 2000$
$c_{t_2} = 100$



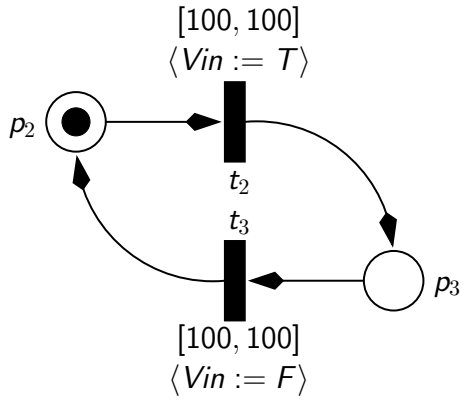


Transition Precondition

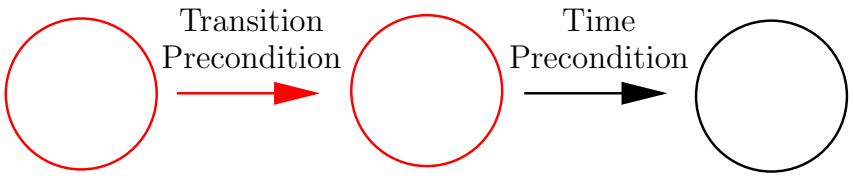


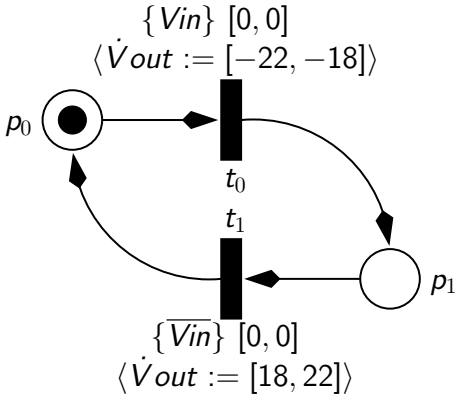


$-200 \leq Vout \leq 2000$
$0 \leq c_{t_2} \leq 100$
$Vout \geq 22c_{t_2} - 200$
$Vout \leq 18c_{t_2} + 200$



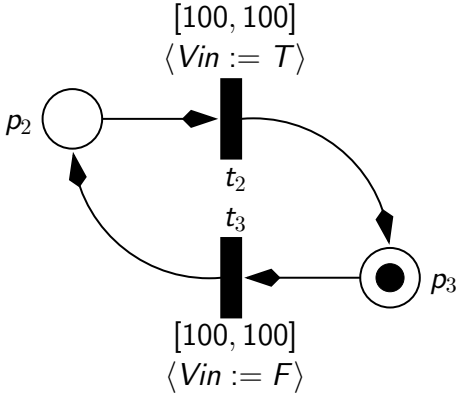
Transition Precondition

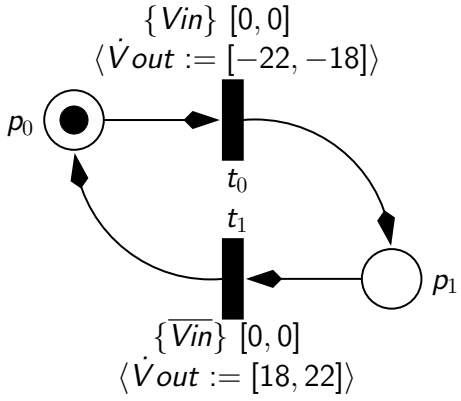




$-200 \leq Vout \leq 200$

$c_{t_3} = 100$



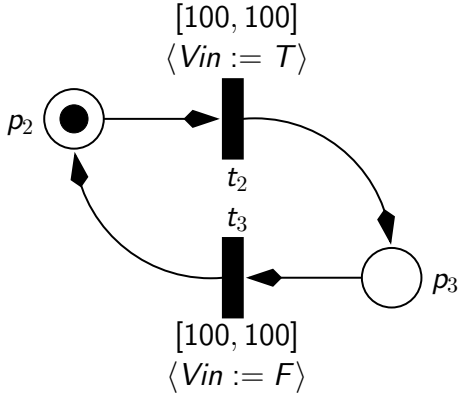


$-200 \leq Vout \leq 2000$

$0 \leq c_{t_2} \leq 100$

$Vout \geq 22c_{t_2} - 200$

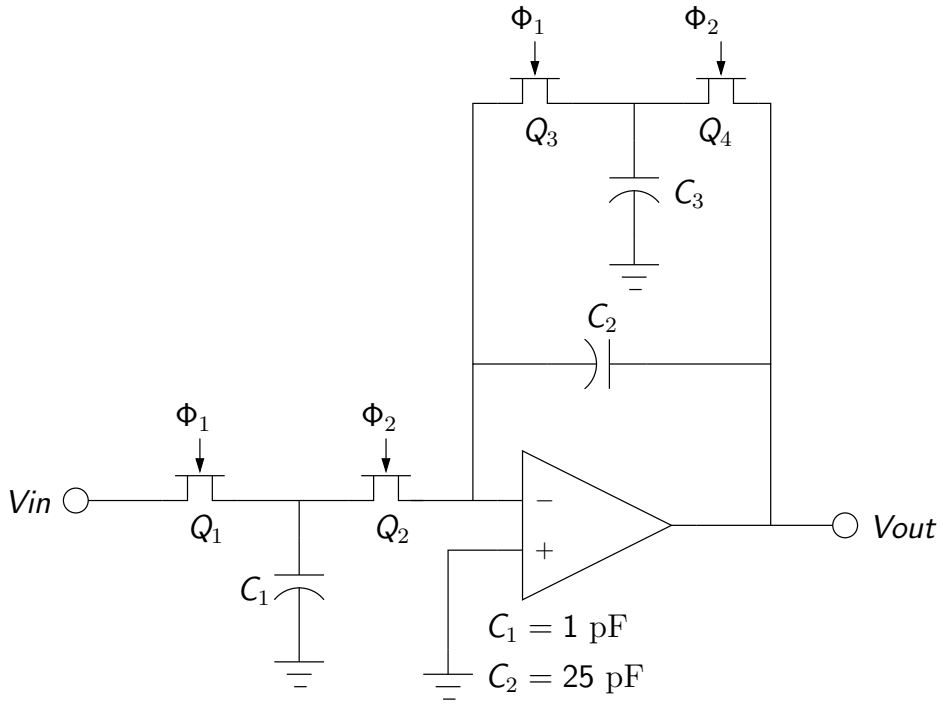
$Vout \leq 18c_{t_2} + 200$



Experimental Results

- Experimented with different ranges of rates for *Vout* 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



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

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.