A New Assertion Property Language for Analog/Mixed-Signal Circuits

Dhanashree Kulkarni, Andrew N. Fisher, Chris J. Myers

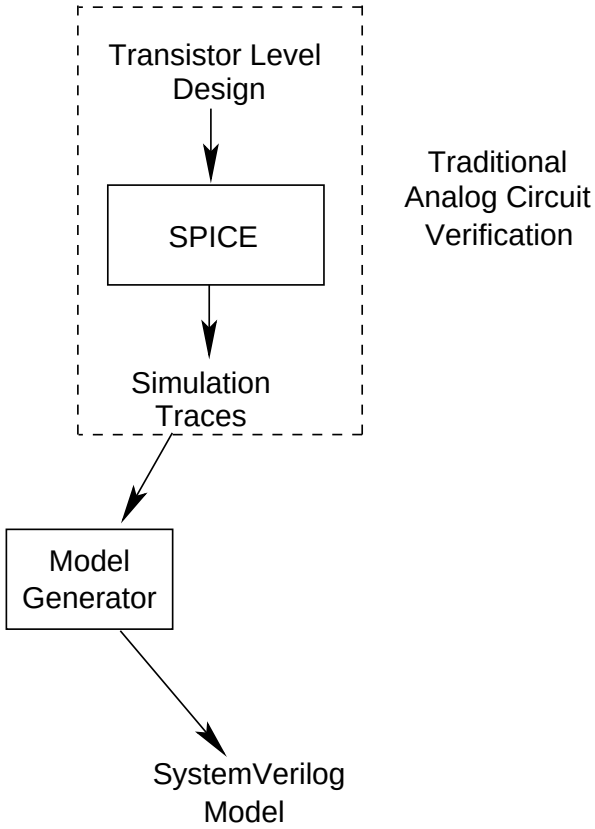
Electrical and Computer Engineering Department
University of Utah

Frontiers in Analog CAD
February 15, 2013

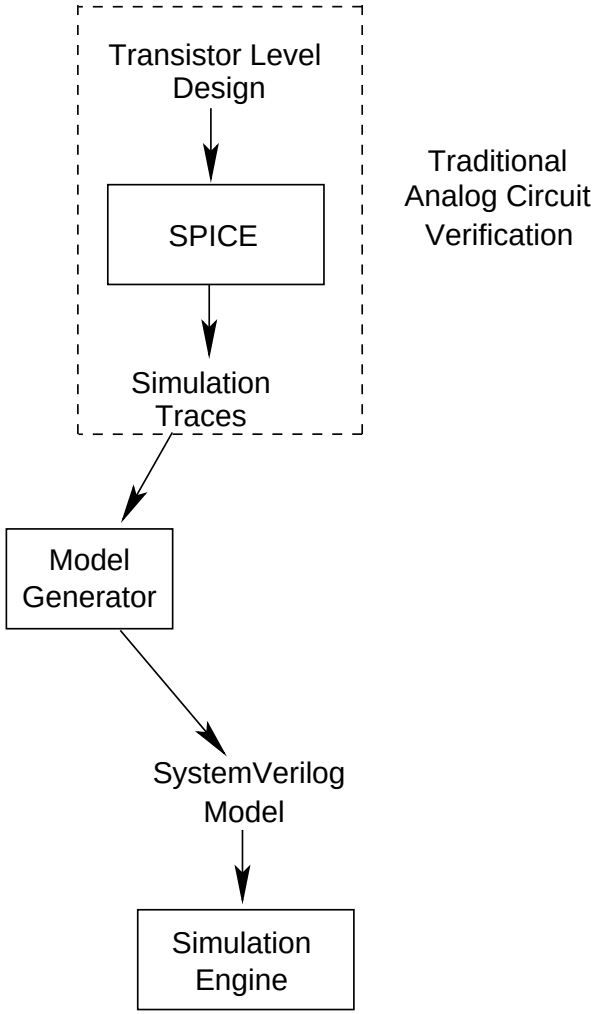
Motivation

- *Analog/mixed-signal* (AMS) verification uses detailed transistor-level (SPICE) simulations.
- SPICE simulation of a PLL can take weeks or even months.
- Long simulation time makes system-level simulation difficult.
- Functional bugs can be missed resulting in catastrophic failures.
- *Model checking* uses non-determinism and state exploration to formally verify designs over all possible behaviors.
- Has had tremendous success for verifying of both digital hardware and software systems (now routinely used at Intel, IBM, Microsoft, etc.).
- For AMS circuits, it is a promising mechanism to validate designs in the face of noise and uncertain parameters and initial conditions.

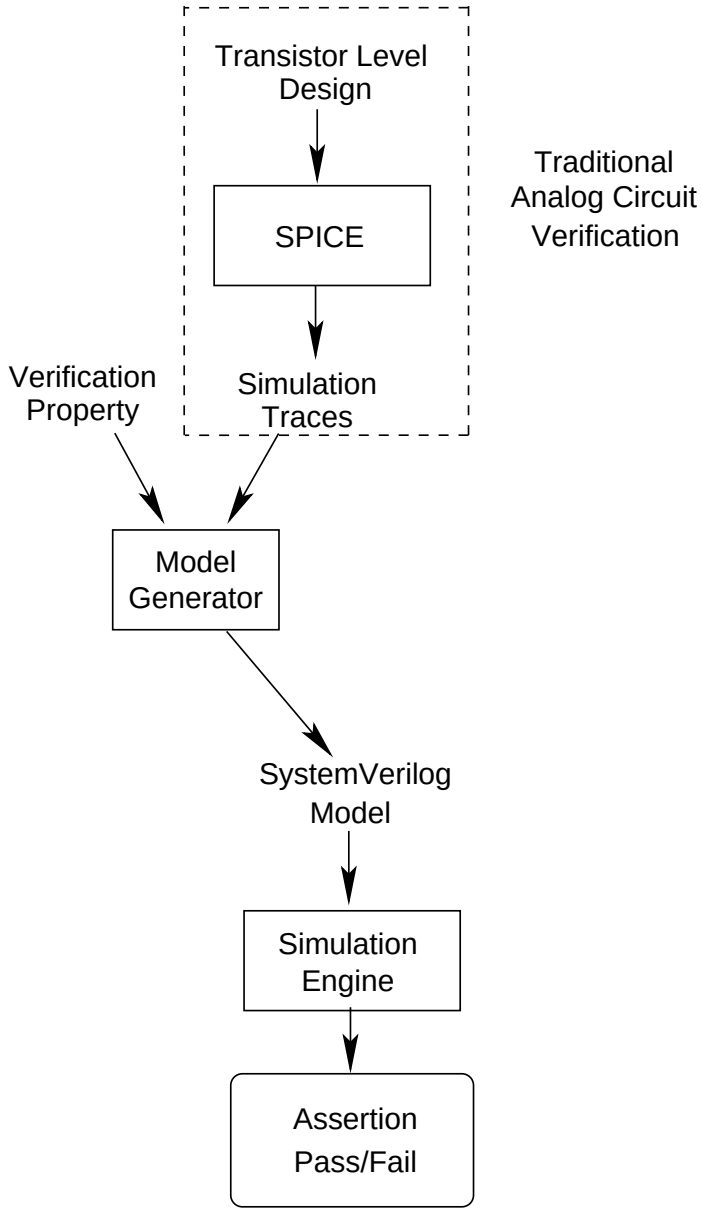
LEMA: LPN Embedded Mixed-Signal Analyzer



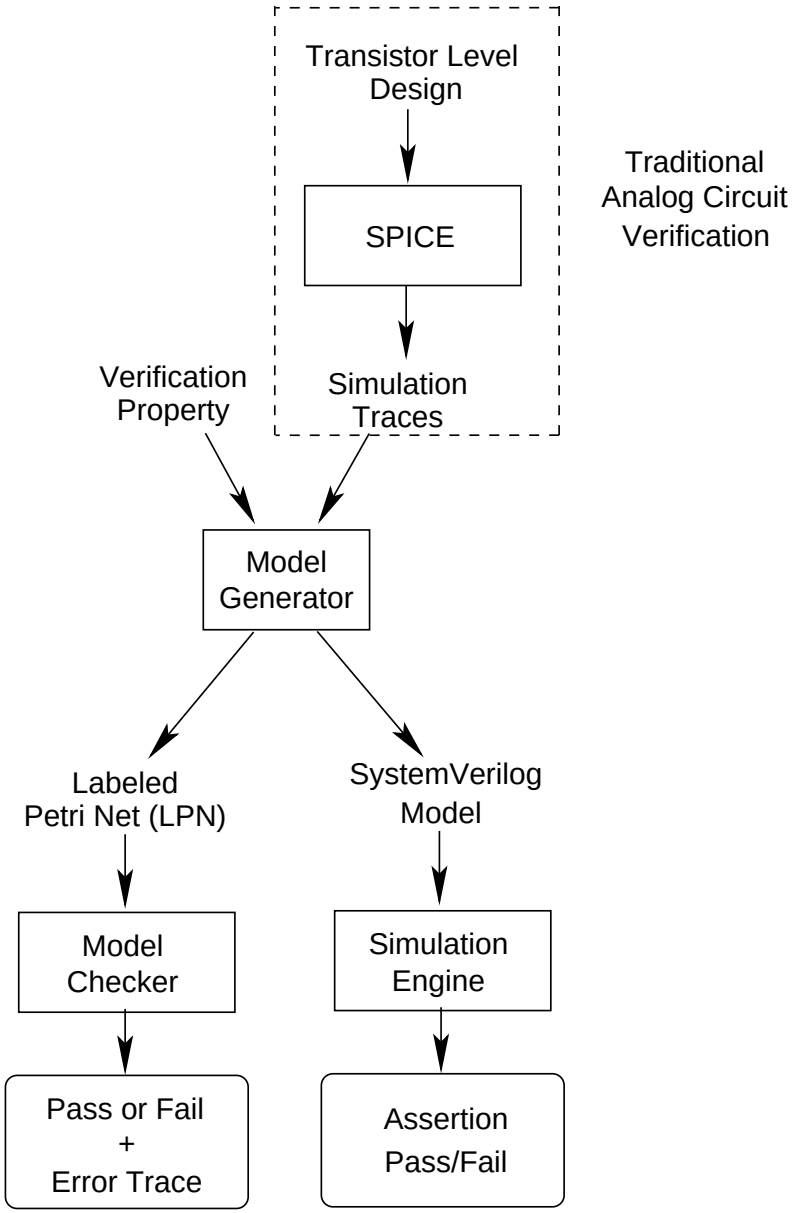
LEMA: LPN Embedded Mixed-Signal Analyzer



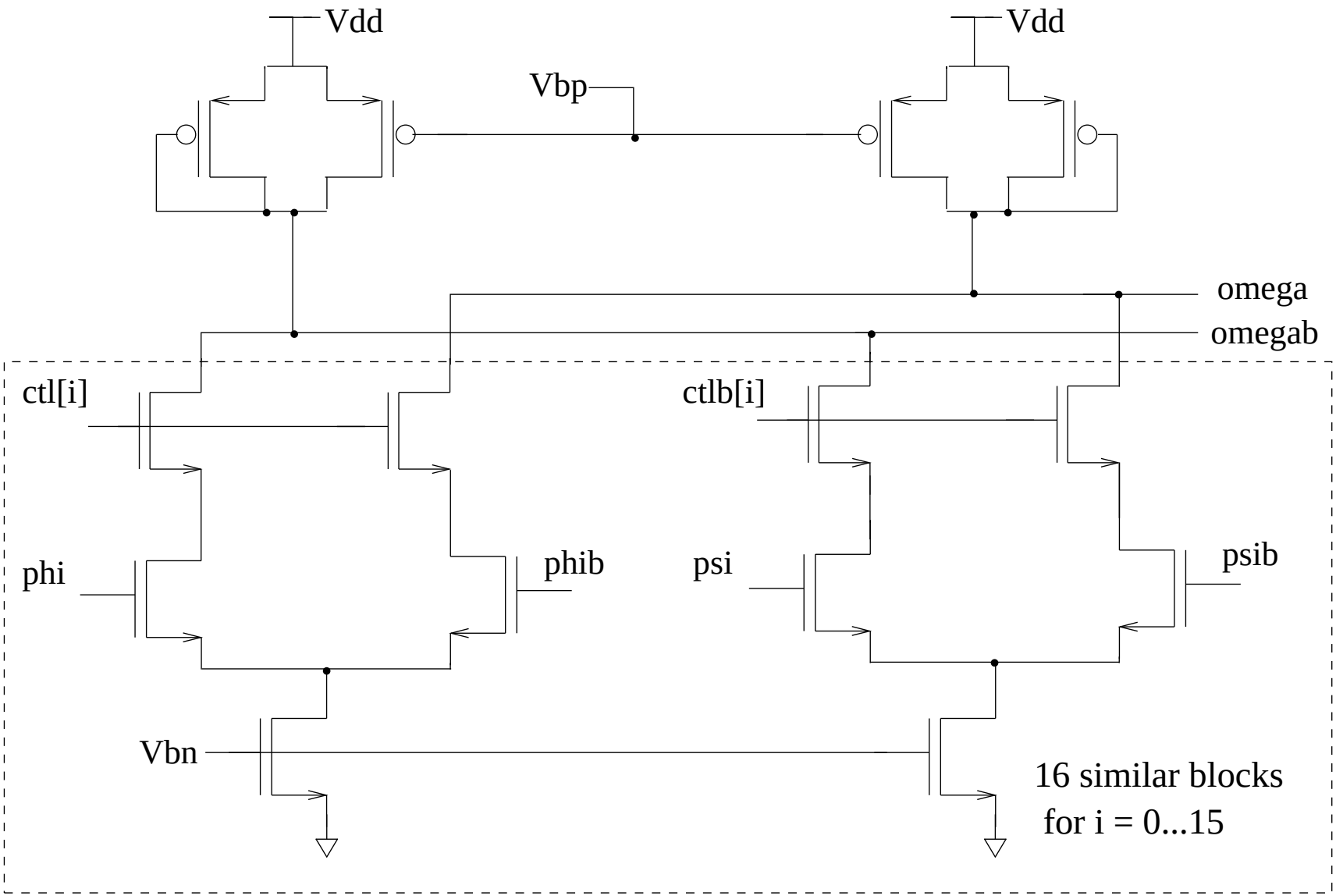
LEMA: LPN Embedded Mixed-Signal Analyzer



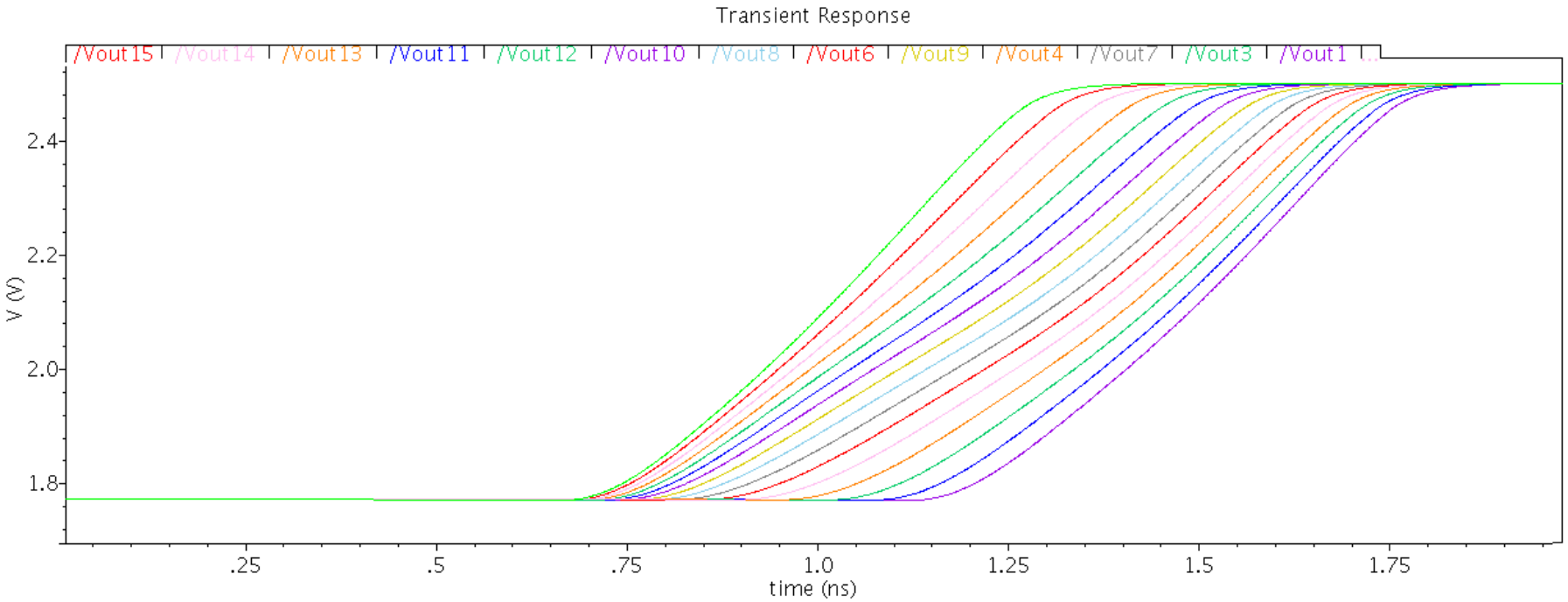
LEMA: LPN Embedded Mixed-Signal Analyzer



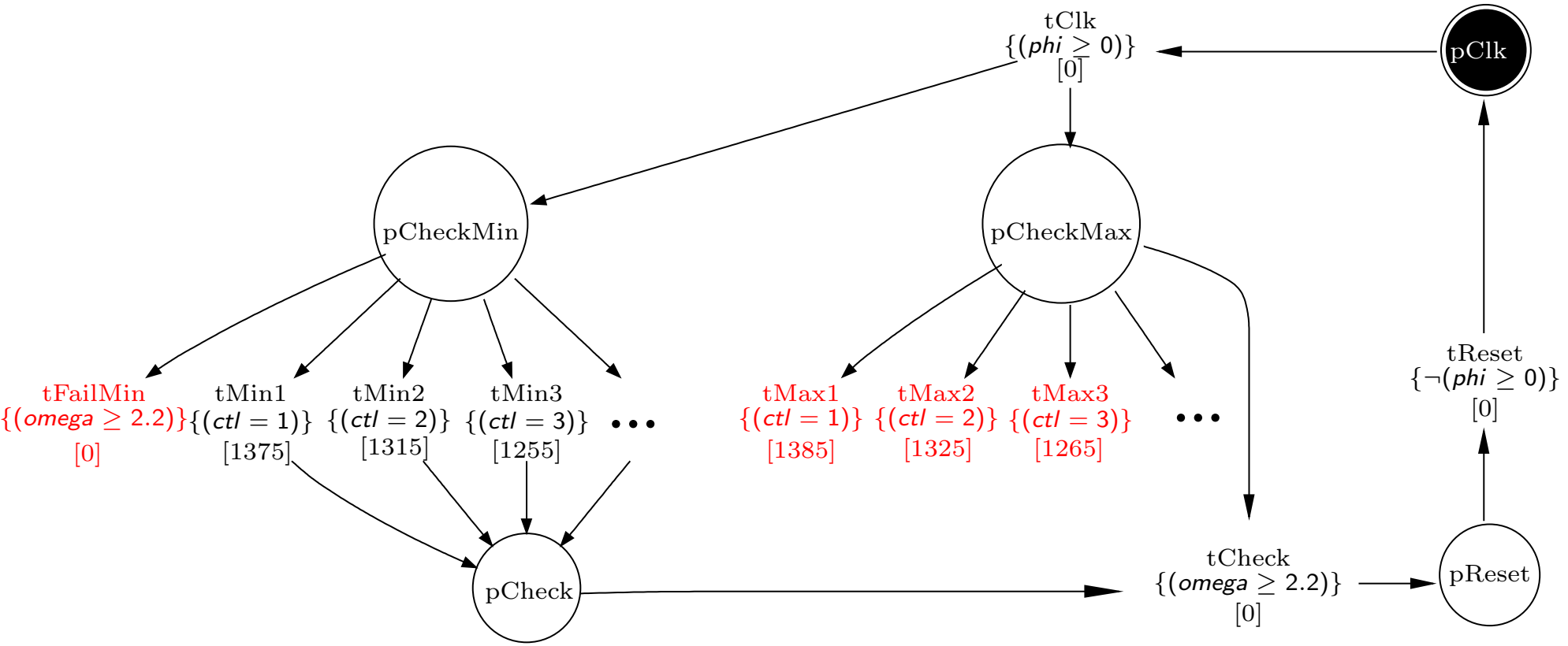
Phase Interpolator



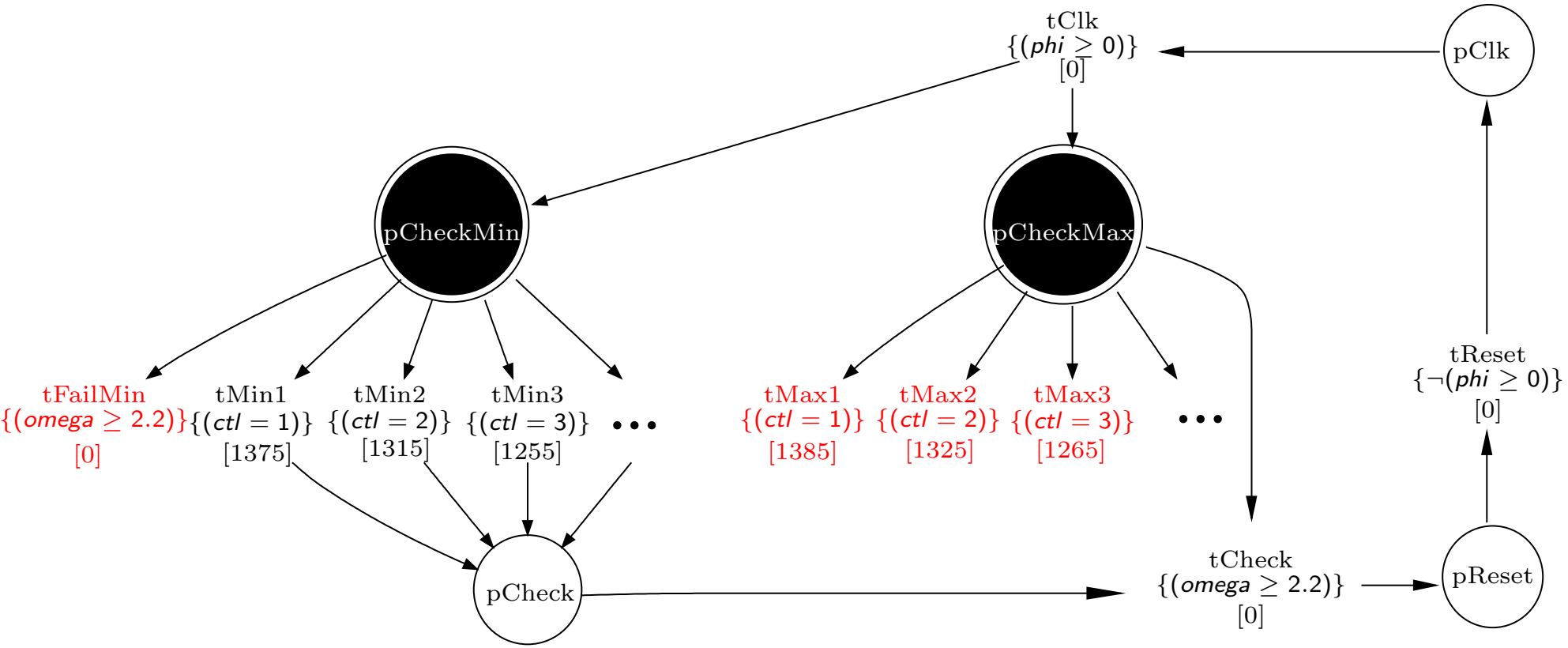
Phase Interpolator Simulation



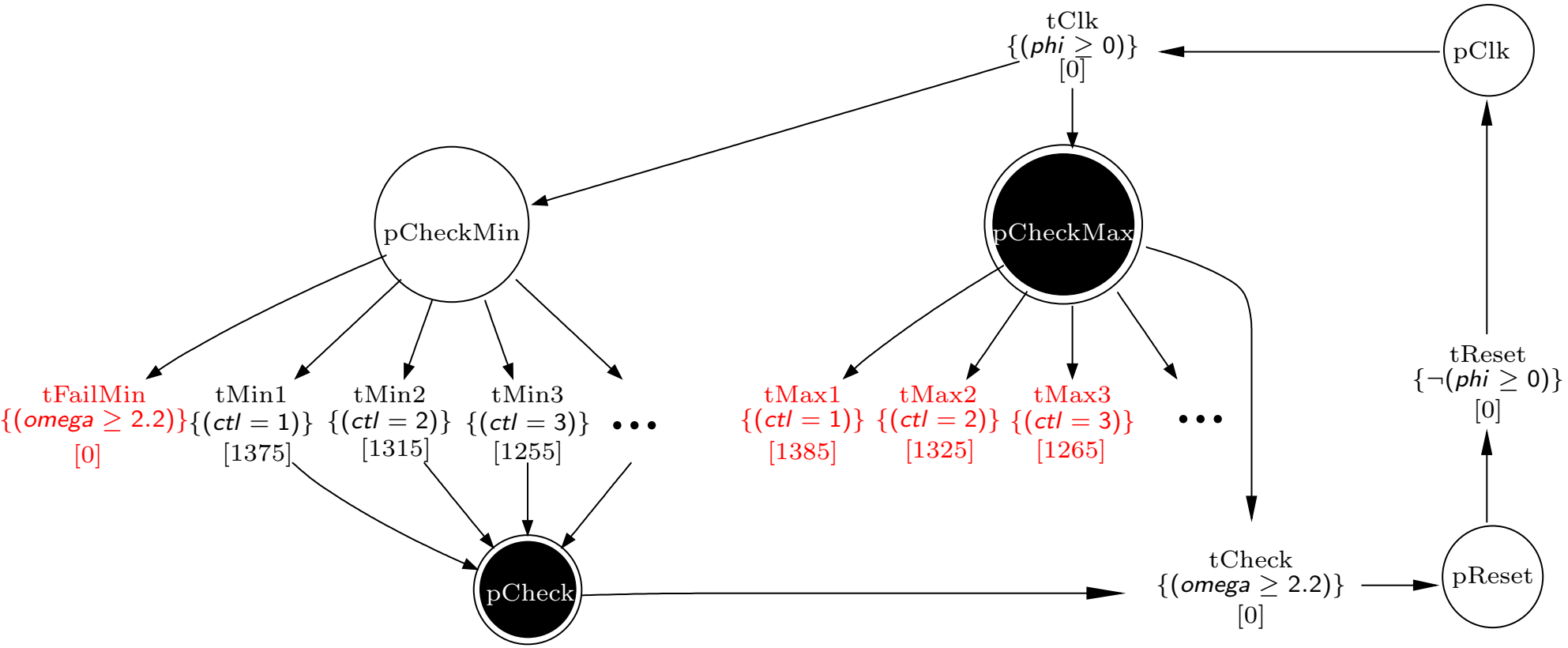
Phase Interpolator (Property LPN)



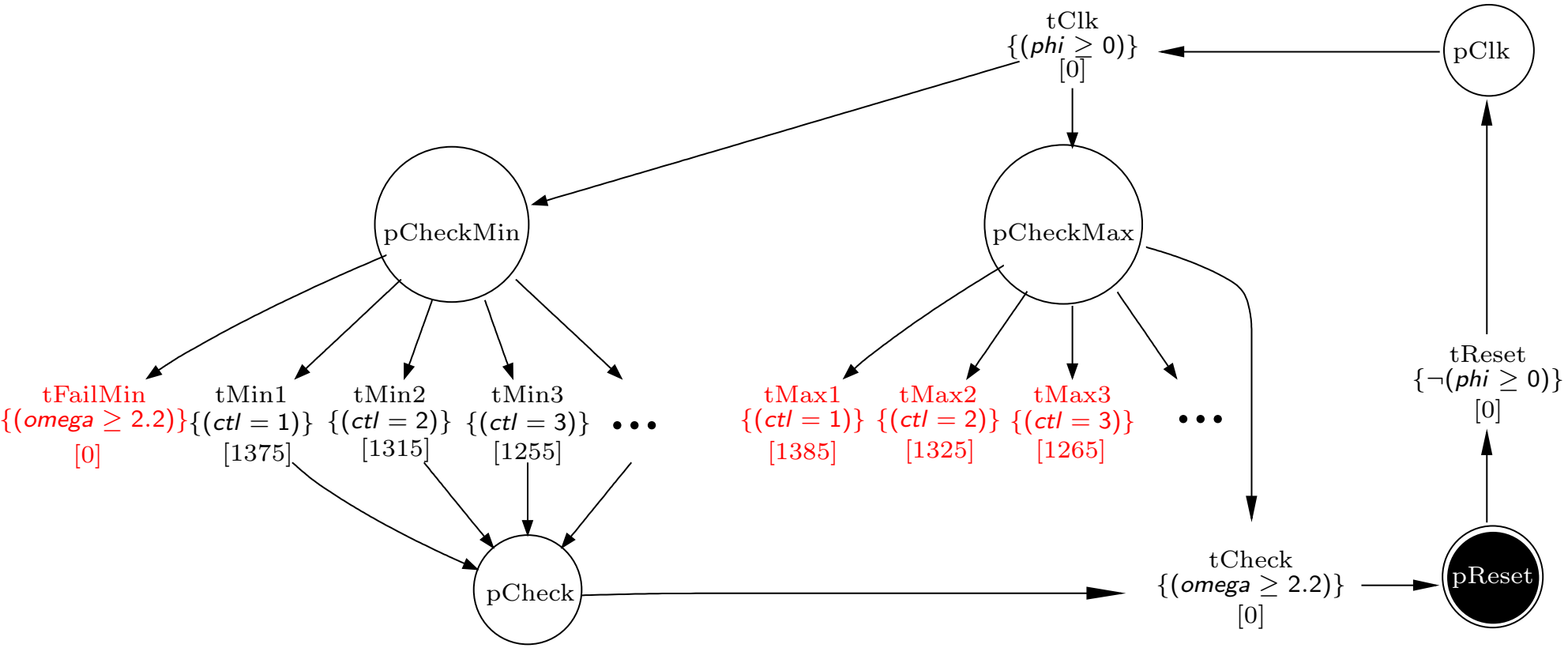
Phase Interpolator (Property LPN)



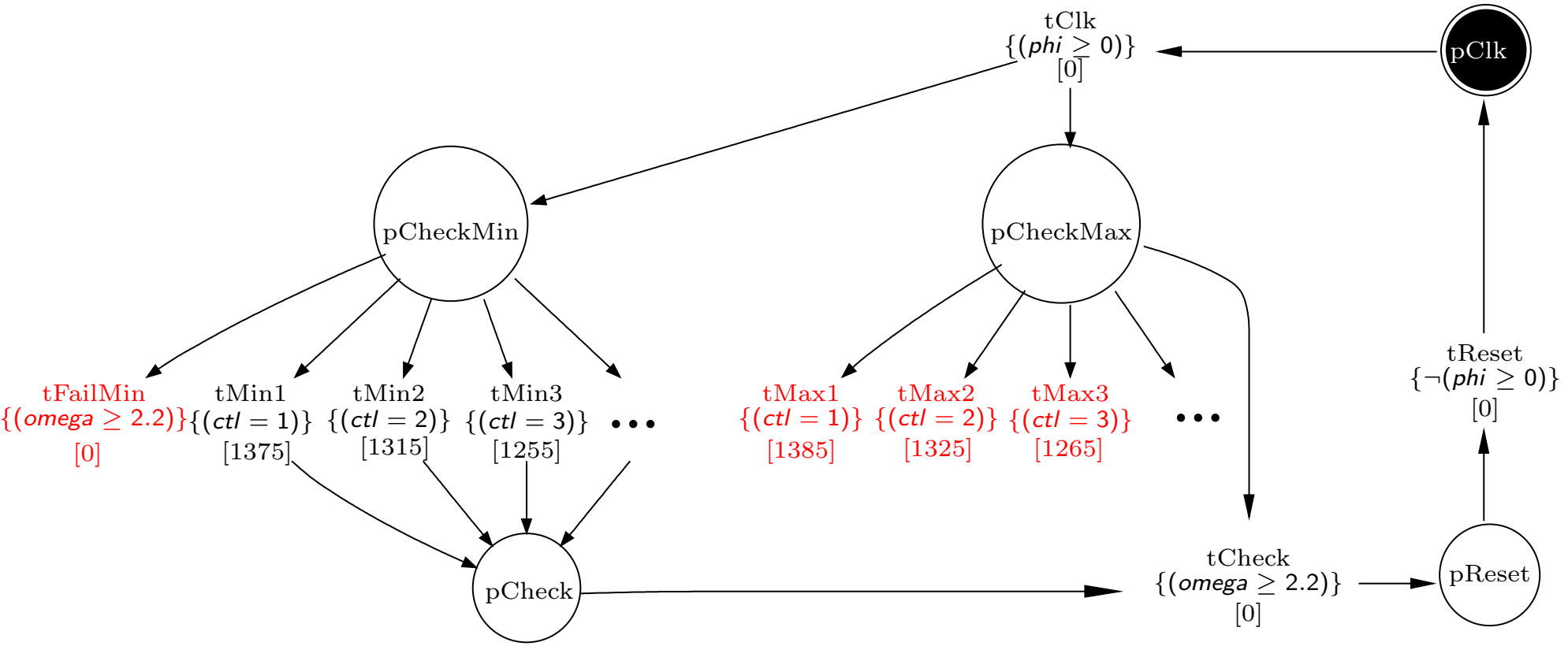
Phase Interpolator (Property LPN)



Phase Interpolator (Property LPN)



Phase Interpolator (Property LPN)



Property Language Translator

- Building property net is a tedious process.
- Requires user to have considerable familiarity with the tool.
- A new simple, intuitive property language is needed.

SystemVerilog Assertions (SVA)

- assert (A == B);
- assert property (@(posedge Clock) Req $\mapsto$ ## [10:20] Ack);

Real-time SVA

- R ::= @(κ)(b) | R ##1 R' | R ##0 R' | R or R' | R intersect R' | R[*0] | R[+] | b | b[*α [+] : β [-]]

$$(phi \geq 0)[\sim> 1] \;\#\#1$$

$$\Big(((ctl == 1)\ \&\&\ !(omega \geq 2.2))[*1375 : 1385] \;\#\#1\; (omega \geq 2.2))$$

$$or$$

$$(((ctl == 2)\ \&\&\ !(omega \geq 2.2))[*1315 : 1325] \;\#\#1\; (omega \geq 2.2))$$

$$or$$

$$(((ctl == 3)\ \&\&\ !(omega \geq 2.2))[*1255 : 1265] \;\#\#1\; (omega \geq 2.2))\Big)$$

$$\#\#1\ !(phi \geq 0)[\sim> 1]$$

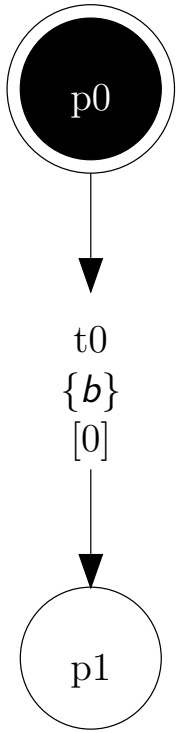
where $b[\sim> 1] \equiv !b[*0.0 : \$] \;\#\#1\; b$.

Our New Property Language

- wait(b) - wait until boolean expression, b, becomes true.
- wait(b,d) - wait at most d time units for b to become true.
- assert(b,d) - ensure that b remains true for d time units.
- assertUntil(b1,b2) - ensure that b1 remains true until b2 is true.
- waitPosedge(b) - wait for a positive edge on b.
- always and if-else constructs for control flow.

Property Language: *wait*(b)

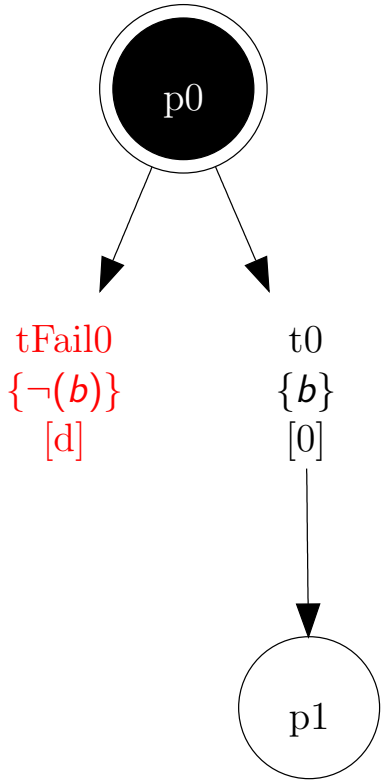
RT-SVA: $b[\sim> 1]$
LPN:



Property Language: $wait(b, d)$

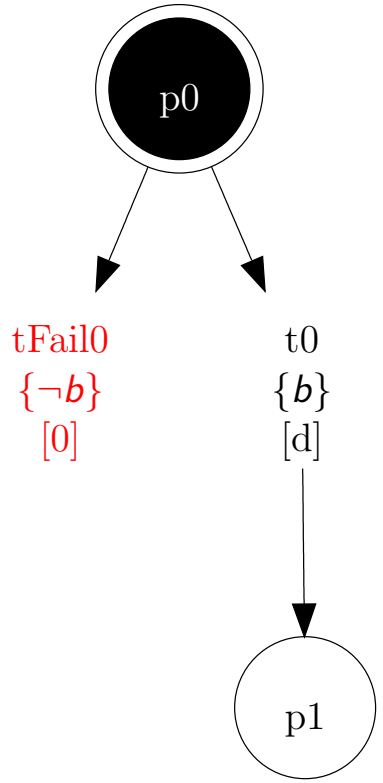
RT-SVA: $!b[*0 : d]$ $\#\#1$ b

LPN:



Property Language: $assert(b, d)$

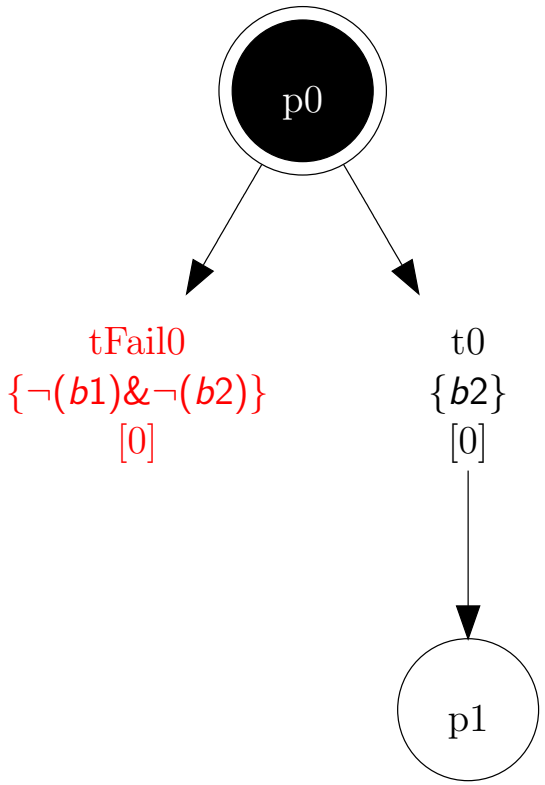
RT-SVA: $b[*d : d]$
LPN:



Property Language: *assertUntil*($b1, b2$)

RT-SVA: $((b1 \;\&\&\; !b2)[*0 : \$] \;\#\#1\; b2)$ *or* $b2$

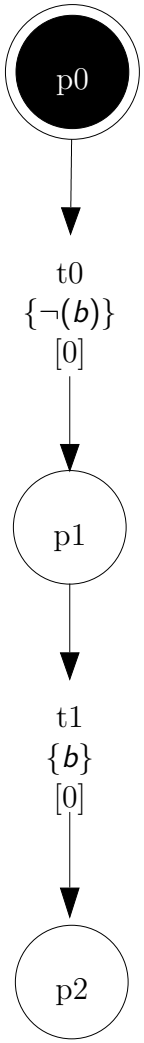
LPN:



Property Language: *waitPosedge*(b)

RT-SVA: $!b[\sim> 1]$ ##1 $b[\sim> 1]$

LPN:



Property Language: *if – else*

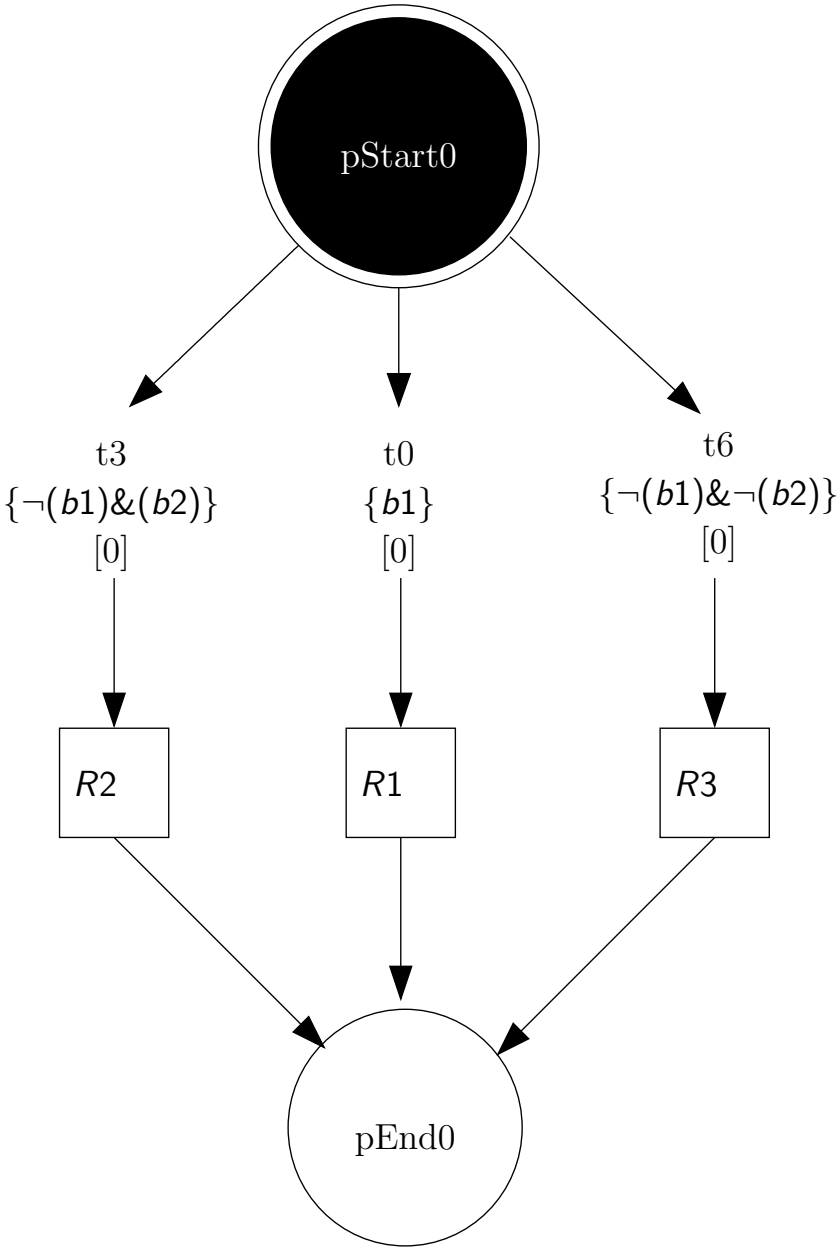
Function:

if (b1) {
    R1
}
else if (b2) {
    R2
}
else {
    R3
}

RT-SVA :

$b1$ ##0 $R1$
or
($b2$ && !$b1$) ##0 $R2$
or
(!$b1$ && !$b2$) ##0 $R3$

Property LPN: *if − else*



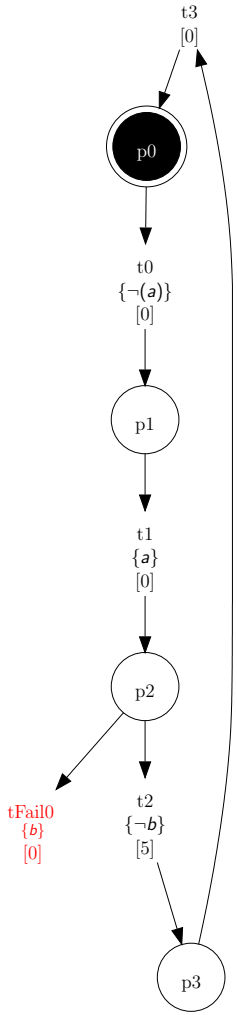
Example 1: Property Language

Whenever a goes from zero to one, b remains low for at least $5ms$.

```
property Example1 {
    boolean a;
    boolean b;
    always{
        waitPosedge (a);
        assert(!b, 5);
    }
}
```

Example 1: Conversion to RT-SVA and LPN

$!a[\sim> 1]\ \#\#1\ a[\sim> 1]\ \#\#1\ !b[*5:5]$



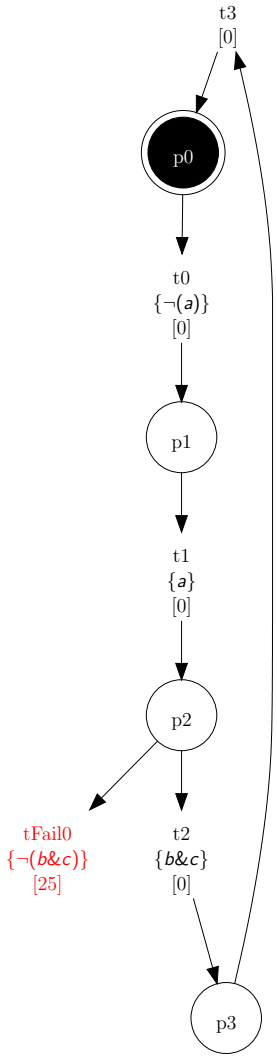
Example 2: Property Language

After a goes high, b and c must be true simultaneously within $25ns$.

```
property Example2{
    boolean a;
    boolean b;
    boolean c;
    always{
        waitPosedge (a);
        wait(b&c, 25);
    }
}
```

Example 2: Conversion to RT-SVA and LPN

$!a[\sim> 1]\ \#\#1\ a[\sim> 1]\ \#\#1\ !(b\ \&\&\ c)[*0:25]\ \#\#1\ (b\ \&\&\ c)$



Example 3: Property Language

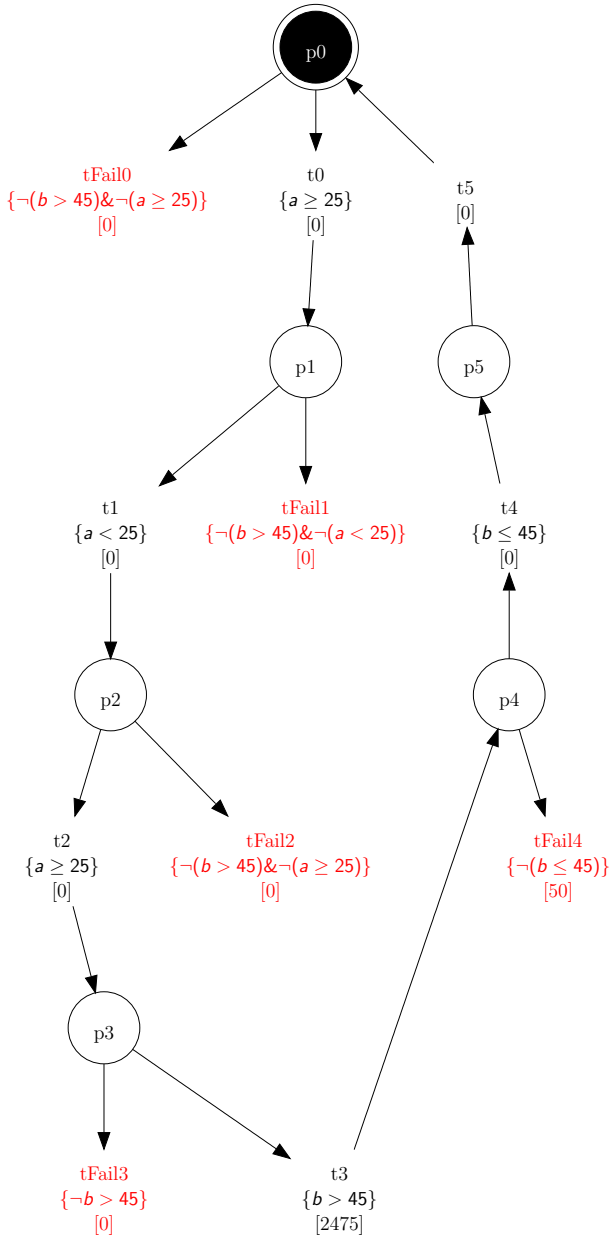
The delay between the second rising crossing of a at $2.5V$ and the first falling crossing of b at $4.5V$ is $250.0ns$ with a tolerance of $2.5ns$.

```
property Example3 {
    real b;
    real a;
    always{
        assertUntil(b > 45, a >= 25);
        assertUntil(b > 45, a < 25);
        assertUntil(b > 45, a >= 25);
        assert(b > 45, 2475);
        wait(b <= 45, 50);
    }
}
```

Example 3: Conversion to RT-SVA

$$(((b > 45)\ \&\&\ !(a \geq 25))[*0 : \$]\ \#\#1\ (a \geq 25))\ \#\#1$$
$$(((b > 45)\ \&\&\ !(a < 25))[*0 : \$]\ \#\#1\ (a < 25))\ \#\#1$$
$$(((b > 45)\ \&\&\ !(a \geq 25))[*0 : \$]\ \#\#1\ (a \geq 25))\ \#\#1$$
$$((b > 45)[*2475 : 2475])\ \#\#1$$
$$(!(b \leq 45)[*0 : 50]\ \#\#1\ (b \leq 45))$$

Example 3: Conversion to LPN



Phase Interpolator Property Using Property Language

```
property PhaseInterpolator {
    real ctl;
    real omega;
    real phi;
    always{
        wait(phi >= 0);
        if(ctl=1){
            assert(!(omega >= 22), 1375);
            wait(omega >= 22,10);
        }
```

continued.....

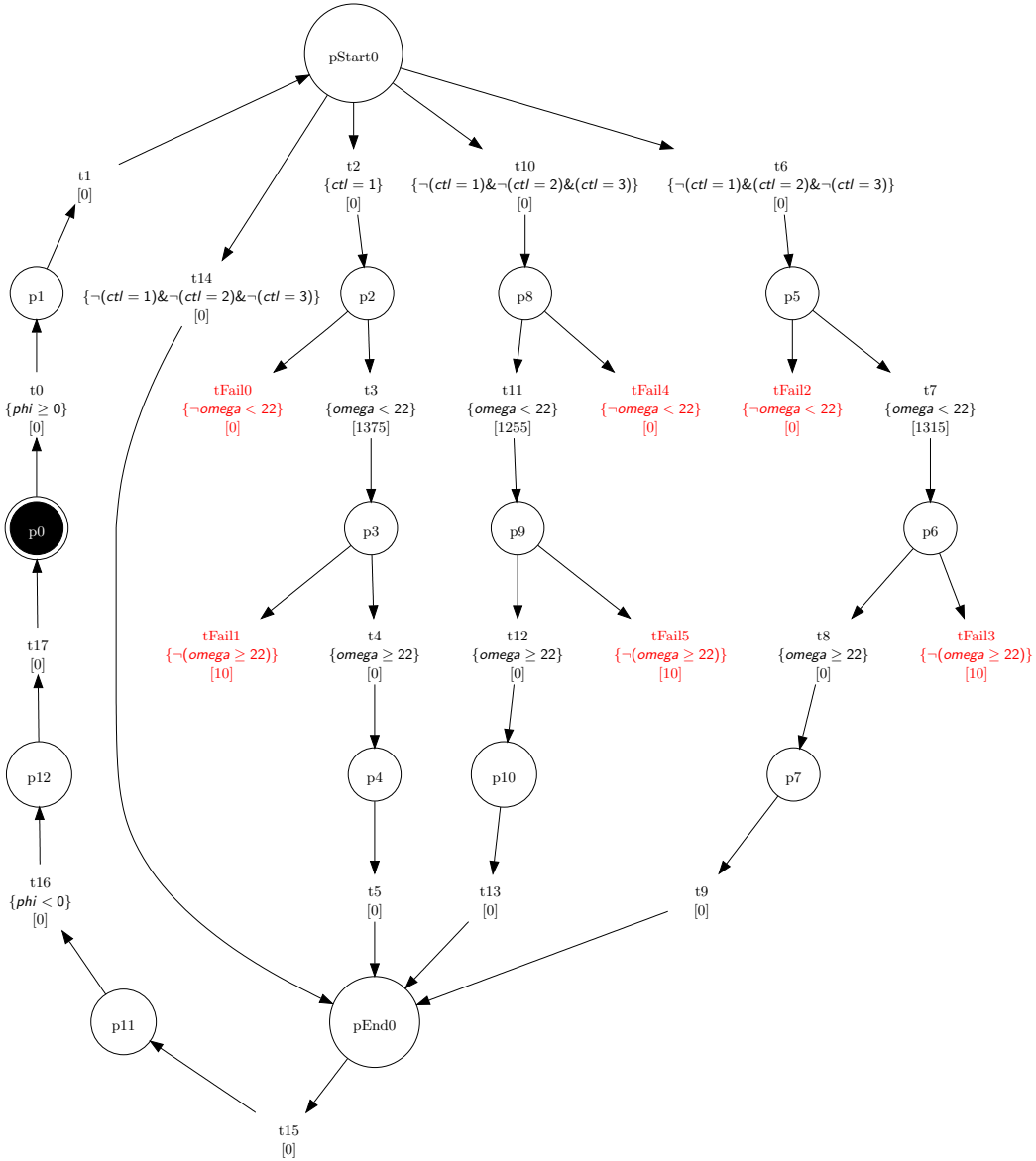
Phase Interpolator Property Using Property Language

```
        else if(ctl=2){
            assert(!(omega >= 22), 1315);
            wait(omega >= 22,10);
        }
        else if(ctl=3){
            assert(!(omega >= 22), 1255);
            wait(omega >= 22,10);
        }
        else {
        }
        wait(phi < 0);
    }
}
```

Phase Interpolator Using Real-Time SVA

$$
\begin{aligned}
&(phi \geq 0)[\sim> 1] \;\#\#1 \\
&\Big(((ctl == 1) \;\#\#0 \\
&\qquad (!(omega \geq 22)[*1375, 1375] \;\#\#1 \\
&\qquad\; !(omega \geq 22)[*0:10] \;\#\#1 \; (omega \geq 22))) \\
&\qquad\qquad \text{or} \\
&(((ctl == 2) \;\&\&\; !(ctl == 1)) \;\#\#0 \\
&\qquad (!(omega \geq 22)[*1315, 1315] \;\#\#1 \\
&\qquad\; !(omega \geq 22)[*0:10] \;\#\#1 \; (omega \geq 22))) \\
&\qquad\qquad \text{or} \\
&(((ctl == 3) \;\&\&\; !(ctl == 2) \;\&\&\; !(ctl == 1)) \;\#\#0 \\
&\qquad (!(omega \geq 22)[*1255, 1255] \;\#\#1 \\
&\qquad\; !(omega \geq 22)[*0:10] \;\#\#1 \; (omega \geq 22))) \\
&\qquad\qquad \text{or} \\
&(!(ctl == 3) \;\&\&\; !(ctl == 2) \;\&\&\; !(ctl == 1)) \Big) \\
&\#\#1 \; (phi < 0)[\sim> 1]
\end{aligned}
$$

Property Language Using Property LPN



Future Work

- Prove the equivalence of RT-SVA automata and property LPNs.
- Determine to what extent LPNs can express RT-SVA.
- Expand the property language to include more constructs.

Acknowledgements

Dhanashree Kulkarni
U. of Utah

Chris J. Myers
U. of Utah

This work has been supported by the National Science Foundation, the Semiconductor Research Corporation, and Intel Corporation.