Symbolic Simulation of Clocks

Guillaume Baudart  Timothy Bourke  Marc Pouzet

Work In Progress
Example: Quasi-Periodic Architectures

- A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

$$0 < T_{\text{min}} \leq T^n \leq T_{\text{max}} \quad \text{or} \quad T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

- Buffered communication without message inversion or loss

“Cooking Book” [Caspi 2000]
Example: Quasi-Periodic Architectures

- A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

\[ 0 < T_{\text{min}} \leq T^n \leq T_{\text{max}} \quad \text{or} \quad T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon \]

\[(\kappa_i)_{i \in \mathbb{N}} \text{ clock activations} \]

- Buffered communication without message inversion or loss
Example: Quasi-Periodic Architectures

• A set of “quasi-periodic” processes with local clocks and nominal period $T^n$ (jitter $\varepsilon$)

$$0 < T_{\text{min}} \leq T^n \leq T_{\text{max}} \quad \text{or} \quad T^n - \varepsilon \leq \kappa_i - \kappa_{i-1} \leq T^n + \varepsilon$$

$(\kappa_i)_{i \in \mathbb{N}}$ clock activations

• Buffered communication without message inversion or loss
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ x \leq T_{\text{max}} \]
\[ \dot{x} = 1 \]

\[ T_{\text{min}} \leq x \]
\[ x := 0, \quad \text{tic!} \]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

$$x \leq T_{\text{max}}$$

$$\dot{x} = 1$$

$$T_{\text{min}} \leq x \Rightarrow x := 0, \text{ tic!}$$

let hybrid metro (tmin, tmax) = tic where
  rec clock x reset tic()
  and tic = present (tmin <= x <= tmax) -> ()
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
x \leq T_{\text{max}} \\
\dot{x} = 1 \\
T_{\text{min}} \leq x / x := 0, \text{ tic!}
\]

let hybrid metro (tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
x \leq T_{\text{max}} \\
\dot{x} = 1 \\
T_{\text{min}} \leq x / x := 0, \text{ tic!}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ x \leq T_{\text{max}} \]
\[ \dot{x} = 1 \]
\[ T_{\text{min}} \leq x / x := 0, \text{ tic!} \]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ \begin{align*}
  x & \leq T_{\text{max}} \\
  \dot{x} & = 1
\end{align*} \]

\[ T_{\text{min}} \leq x / x := 0, \text{ tic!} \]

let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

[x = 0]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
\begin{align*}
  x &\leq T_{\text{max}} \\
  \dot{x} &\overset{\text{der}}{=} 1 \\
  T_{\text{min}} &\leq x / x := 0, \text{ tic!}
\end{align*}
\]

let hybrid metro (in\_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0 
  and tic = present in\_x on (tmin \leq x \leq tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\([x = 0] \rightarrow [x = 2.4]\)
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
\begin{align*}
&x \leq T_{\text{max}} \\
&\dot{x} = 1
\end{align*}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[x = 0 \rightarrow [x = 2.4] \xrightarrow{\text{in}_x} [x = 2.4]\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

let hybrid metro (in_x, tmin, tmax) = tic where
    rec der x = 1.0 init 0.0 reset c() -> 0.0
    and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[ [x = 0] \rightarrow [x = 2.4] \xrightarrow{\text{in}_x/} [x = 2.4] \rightarrow [x = 3.5] \]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
x \leq T_{\text{max}} \quad \dot{x} = 1 \quad T_{\text{min}} \leq x \quad x := 0, \text{ tic!}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin \leq x \leq tmax) -> ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[
[x = 0] \xrightarrow{\text{in}_x} [x = 2.4] \xrightarrow{\text{in}_x/} [x = 2.4] \xrightarrow{\text{tic}} [x = 3.5] \xrightarrow{\text{in}_x/\text{tic}} [x = 0]
\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
x \leq T_{\text{max}} \\
\dot{x} = 1
\]

\[
T_{\text{min}} \leq x / x := 0, \text{ tic!}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[
[x = 0] \rightarrow [x = 2.4] \xrightarrow{\text{in}_x/} [x = 2.4] \rightarrow [x = 3.5] \xrightarrow{\text{in}_x/\text{tic}} [x = 0]
\]

[x = 0]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

let hybrid metro (in_x, tmin, tmax) = tic where
  rec der x = 1.0 init 0.0 reset c() -> 0.0
  and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[
\begin{align*}
[x = 0] & \rightarrow [x = 2.4] & [x = 2.4] & \rightarrow [x = 2.4] & \rightarrow [x = 3.5] & \rightarrow [x = 0] \\
[x = 0] & \rightarrow [x = 1.1]
\end{align*}
\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
\begin{align*}
\dot{x} &= 1 \\
T_{\min} \leq x / x := 0, \text{ tic!}
\end{align*}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() \rightarrow 0.0
and tic = present in_x on (tmin <= x <= tmax) \rightarrow ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[
\begin{align*}
[x = 0] &\rightarrow [x = 2.4]^{\text{in}_x/} \rightarrow [x = 2.4] \rightarrow [x = 3.5]^{\text{in}_x/\text{tic}} \rightarrow [x = 0] \\
[x = 0] &\rightarrow [x = 1.1]^{\text{in}_x/} \rightarrow [x = 1.2]
\end{align*}
\]

\[
t_{\min} = 3 \\
t_{\max} = 5
\]
Example: Quasi-Periodic Architectures

**Fuzzy Metronome**

\[ \begin{align*}
  x &\leq T_{\text{max}} \\
  \dot{x} &\hspace{1em} = 1 \\
  T_{\text{min}} &\leq x \\
  x &\hspace{1em} : = 0, \text{ tic!}
\end{align*} \]

let hybrid metro (in_x, tmin, tmax) = tic where 
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[ \begin{align*}
  [x = 0] &\rightarrow [x = 2.4]^{\text{in}_x/} \rightarrow [x = 2.4] \rightarrow [x = 3.5]^{\text{in}_x/\text{tic}} \rightarrow [x = 0] \\
  [x = 0] &\rightarrow [x = 1.1]^{\text{in}_x/} \rightarrow [x = 1.2] \rightarrow [x = 4.2]
\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
\begin{align*}
x &\leq T_{\text{max}} \\
\dot{x} & = 1
\end{align*}
\]

\[
T_{\text{min}} \leq x \Rightarrow x := 0, \text{ tic!}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin \leq x \leq tmax) -> ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[
\begin{align*}
[x = 0] &\rightarrow [x = 2.4] \xrightarrow{\text{in}_x/} [x = 2.4] \rightarrow [x = 3.5] \xrightarrow{\text{in}_x/\text{tic}} [x = 0] \\
[x = 0] &\rightarrow [x = 1.1] \xrightarrow{\text{in}_x/} [x = 1.2] \rightarrow [x = 4.2] \xrightarrow{\text{in}_x/\text{tic}} [x = 0]
\end{align*}
\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ \begin{align*}
& x \leq T_{\text{max}} \\
& \dot{x} = 1
\end{align*} \]

\[ T_{\text{min}} \leq x / x := 0, \text{ tic!} \]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs
Example: Quasi-Periodic Architectures

**Fuzzy Metronome**

\[
\begin{align*}
&x \leq T_{\text{max}} \\
&\dot{x} = 1 \\
&T_{\text{min}} \leq x / x := 0, \text{ tic!}
\end{align*}
\]

let hybrid metro (in_x, tmin, tmax) = tic where

rec der x = 1.0 init 0.0 reset c() -> 0.0

and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[0 \leq x \leq 3\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[
\begin{align*}
x & \leq T_{\text{max}} \\
\dot{x} & = 1
\end{align*}
\]

\[
T_{\text{min}} \leq x \rightarrow x := 0, \text{ tic!}
\]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[0 \leq x \leq 3 \xrightarrow{\text{in}_x/} [0 \leq x \leq 3]\]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ T_{\text{min}} \leq x / x := 0, \text{tic!} \]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() -> 0.0
and tic = present in_x on (tmin <= x <= tmax) -> ()

Clock can be implemented as a simple ODE

Non-determinism is handle via additional inputs

\[ 0 \leq x \leq 3 \] \[ \text{in}_x/ \rightarrow \] \[ 0 \leq x \leq 3 \] \[ \text{wait} \rightarrow \] \[ 3 \leq x \leq 5 \]
Example: Quasi-Periodic Architectures

Fuzzy Metronome

\[ \begin{align*}
  x &\leq T_{\text{max}} \\
  \dot{x} &= 1 \\
  T_{\text{min}} &\leq x / x := 0, \text{ tic}!
\end{align*} \]

let hybrid metro (in_x, tmin, tmax) = tic where
rec der x = 1.0 init 0.0 reset c() \rightarrow 0.0
and tic = present in_x on (tmin \leq x \leq tmax) \rightarrow ()

Clock can be implemented as a simple ODE
Non-determinism is handle via additional inputs

\[\begin{align*}
  [0 \leq x \leq 3] &\xrightarrow{\text{in}_x/} [0 \leq x \leq 3] \xrightarrow{\text{wait}} [3 \leq x \leq 5] \xrightarrow{\text{in}_x/\text{tic}} [0 \leq x \leq 3]
\end{align*}\]
Symbolic Simulation

Definition: Two simulation states \((q, s)\) and \((q', s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)
Symbolic Simulation

Definition: Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse
**Symbolic Simulation**

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse
Symbolic Simulation

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse
Symbolic Simulation

Definition: Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\).

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse
Symbolic Simulation

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each state, there is a set of enabled actions:

- Transition on inputs
- Non-determinism (additional inputs + boolean constraint)
- Time elapse

Simulation Box

- Regular inputs: \(x\) and \(y\)
- Additional inputs for non determinism: \(\text{in}_x\), \(\text{in}_y\)
- Time elapse: \(\text{wait}\)
Symbolic Simulation
Comparison with existing tool
Australian Walk
Symbolic Simulation
Comparison with existing tool

Australian Walk
Symbolic Simulation
Comparison with existing tool

Australian Walk

Safe
Symbolic Simulation

Comparison with existing tool

Australian Walk

Safe
Symbolic Simulation

Comparison with existing tool

Australian Walk

Safe
Symbolic Simulation

Comparison with existing tool

Australian Walk

Safe
Symbolic Simulation

Comparison with existing tool

Australian Walk

Demo: Uppaal

[Larsen et al. 1997]
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

Safe
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

Safe
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

Safe
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

Safe
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk

Safe
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk
Symbolic Simulation

Our proposal: keep a notion of time elapsing

Australian Walk
Definition: Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

Increment all clocks at the same time: slide along direction \((1, 1, ..., 1)\)

[Halbwachs et al. 1994]
Definition: Two simulation states \((q, s)\) and \((q’, s’)\) are equivalent if \(q = q’\) and \(\text{actions}(s) = \text{actions}(s’)\)
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q, s)\) and \((q', s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: **entering** and **leaving**
**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: entering and leaving
Definition: Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q, s)\) and \((q', s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: *entering* and *leaving*
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: entering and leaving.
Time Elapse... A New Horizon

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: entering and leaving
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q,s)\) and \((q’,s’)\) are equivalent if \(q = q’\) and \(\text{actions}(s) = \text{actions}(s’)\)

For each action, two horizons: **entering** and **leaving**
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: entering and leaving
**Time Elapse... A New Horizon**

**Definition:** Two simulation states \((q,s)\) and \((q',s')\) are equivalent if \(q = q'\) and \(\text{actions}(s) = \text{actions}(s')\)

For each action, two horizons: entering and leaving.
Language Restrictions
A subset of Zélus

Clocks: \texttt{der } x = 1.0

Clock constraints: \((x - y) \bowtie e\)
with \(\bowtie \in \{\leq, <, >, \geq\}\) and e: float

Operations on clocks:

- Reset: \(x = v\)
- Translation: \(x = x + v\)
- Synchronization: \(x = y\)
Language Restrictions
A subset of Zélus

Clocks: \( \text{der} \ x = 1.0 \)

Clock constraints: \((x - y) \bowtie e\)
with \( \bowtie \in \{\leq, <, >, \geq\} \) and \( e: \text{float} \)

Operations on clocks:
• Reset: \( x = v \)
• Translation: \( x = x + v \)
• Synchronization: \( x = y \)

[Difference Bound Matrices (DBM)] [Dill 1990]
Compilation

**Modularity:** Each block returns the guard of the enabled transitions.

**Global State:** A DBM represents clock constraints of the entire system (current clock domain).

**Simulation:** At each step, we return the next horizon and compute the next clock domain.
Compilation

**Modularity:** Each block returns the guard of the enabled transitions.

**Global State:** A DBM represents clock constraints of the entire system (current clock domain).

**Simulation:** At each step, we return the next horizon and compute the next clock domain.

A way to discretize continuous systems
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

\[
\text{let hybrid \ qp\_archi (in\_x, in\_y) = ticA, ticB where}
\]
\[
\text{rec ticA = metro (in\_x, 3, 5)}
\]
\[
\text{and ticB = metro (in\_x, 3, 5)}
\]
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

```plaintext
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```

\[
\begin{align*}
0 &\leq x < 3 \\
0 &\leq y < 3 \\
x - y & = 0
\end{align*}
\]
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)

\[
\begin{align*}
3 &\leq x \leq 5 \\
3 &\leq y \leq 5 \\
x - y & = 0
\end{align*}
\]
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

```ocaml
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```

```
{ 0 \leq x < 2
  3 \leq y \leq 5
  3 \leq y - x \leq 5 } 
```
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

\[
\text{let hybrid } \text{qp_archi} (\text{in}_x, \text{in}_y) = \text{ticA}, \text{ticB} \text{ where } \\
\text{rec } \text{ticA} = \text{metro} (\text{in}_x, 3, 5) \\
\text{and } \text{ticB} = \text{metro} (\text{in}_x, 3, 5)
\]

\[
\begin{align*}
0 \leq x &< 3 \\
0 \leq y &\leq 1 \\
0 \leq x - y &\leq 2
\end{align*}
\]
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

```plaintext
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```

In this symbolic simulation, we define a pair of quasi-periodic clocks, `ticA` and `ticB`, with period parameters `(in_x, 3, 5)` for `ticA` and `ticB`. The clocks' behavior is determined by their periodicity in the range defined by the inequality:

\[
\begin{align*}
1 &\leq x \leq 5 \\
1 &\leq y < 3 \\
0 &\leq x - y \leq 2
\end{align*}
\]

This set of conditions specifies the allowed values for `x` and `y`. The simulation includes inputs `in_x`, `in_y`, and a `wait` signal, indicating the synchronization points and transition times between the clocks.
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```

\[
\begin{aligned}
3 & \leq x \leq 5 \\
3 & \leq y \leq 5 \\
0 & \leq x - y \leq 2
\end{aligned}
\]
Example: Quasi-Periodic Architectures
Symbolic Simulation of a pair of quasi-periodic clocks?

```
let hybrid qp_archi (in_x, in_y) = ticA, ticB where
  rec ticA = metro (in_x, 3, 5)
  and ticB = metro (in_x, 3, 5)
```
Example: Quasi-Periodic Architectures

Symbolic Simulation of a pair of quasi-periodic clocks?

let hybrid qp_archi (in_x, in_y) = ticA, ticB where
rec ticA = metro (in_x, 3, 5)
and ticB = metro (in_x, 3, 5)

\[
\begin{align*}
3 \leq x & \leq 5 \\
0 \leq y & \leq 2 \\
3 \leq x - y & \leq 5
\end{align*}
\]

No more than two ticks of one clock between two ticks of the other.
Future Work

Prototype implementation in zélus
Source to source transformation and runtime

More complex clock domains
octagon, polyhedron, …

Under-approximation / Over-approximation
safety vs. precision

Generate discrete controllers
for instance quasi-synchronous controllers

Improve test coverage
see [Alur et al 2008]