TimeSquare : A multiform time simulation environment

C. André, B. Ferrero, F. Mallet
AOSTE - I3S/Inria
Université de Nice Sophia Antipolis
INRIA Sophia Antipolis Méditerranée

Outline

- MARTE Time model
- CCSL
- TimeSquare
- Examples

MARTE motivation

- In the real world, SW and RTE designers
  - Use UML to draw graphs, vertices and edges, with fancy adornments
  - Perform model transformations to their proprietary language that makes its own assumptions and give its own semantics
  - ➢ Models are not merged but only stored in the same bundle

- MARTE defines a common ground (and semantics?) for building RTE models with UML
  - The MARTE Time model relies on CCSL to define interactions among *clocks* (processes, actors, …)
  - MARTE should be extended for domain-specific purposes

Where to put the semantics itself ? In the OMG specification ?

Time model - Clocks

- Any event (start/end of actions; send/receive of messages; transition being fired; …) is a **Clock**
  - When the *distance* between two successive occurrences of the event is meaningful (like in Physical time) => Chrono**metric clocks**
  - Otherwise => **Logical clocks** => **Multiform time**

- More formally, a clock is a five-tuple $\langle \mathcal{I}, \prec, \mathcal{D}, \lambda, u \rangle$
  - $\mathcal{I}$ is a set of instants (possibly infinite);
  - $\prec$ is a strict quasi-order relation on $\mathcal{I}$;
  - $\mathcal{D}$ is as set of labels;
  - $\lambda : \mathcal{I} \rightarrow \mathcal{D}$ is a labeling function ;
  - u is the unit.

- Clocks can be
  - *discrete* ($\mathcal{I}$ is a discrete set) - idx : $\mathcal{I} \rightarrow \mathbb{N}^*$, idx is order-preserving
  - or *dense*.

Today, focus on discrete logical clocks

Time model – Time structure

- Several interdependent clocks are gathered within a **time structure**

- A *time structure* is a pair $\langle \mathcal{C}, \preccurlyeq \rangle$
  - $\mathcal{C}$ is a finite set of clocks;
  - $\preccurlyeq$ is a partial order relation on $\bigcup_{c \in \mathcal{C}} \mathcal{I}_c$

- From $\preccurlyeq$ we derive four *instant relations*:
  - *Coincidence*: $\equiv \triangleq \preccurlyeq \cap \succcurlyeq$
  - *Strict precedence*: $\prec \triangleq \preccurlyeq \setminus \equiv$
  - *Independence*: $\parallel \triangleq \overline{\preccurlyeq \cup \succcurlyeq}$
  - *Exclusion*: $\# \triangleq \prec \cup \succ$

Time model – Clock relations

- *Clock relations* define (infinitely) many instant relations
- Four categories of clock relations
  - **Coincidence-based** (synchronous)
    - isSubClock, discretizedBy, **isPeriodicOn**, filteredBy …
  - **Precedence-based** (asynchronous)
    - isFasterThan (precedes), **alternatesWith** …
  - **Mixed** (asynchronous => synchronous)
    - **sampledOn**, delayedFor, timer, inf, sup …
  - **Quantitative** (related to chronometric clocks)
    - hasStability, hasOffset, hasJitter, hasDrift …
- **C**lock **C**onstraint **S**pecification **L**anguage = concrete syntax
  - Non-normative annex of MARTE

TimeSquare purpose

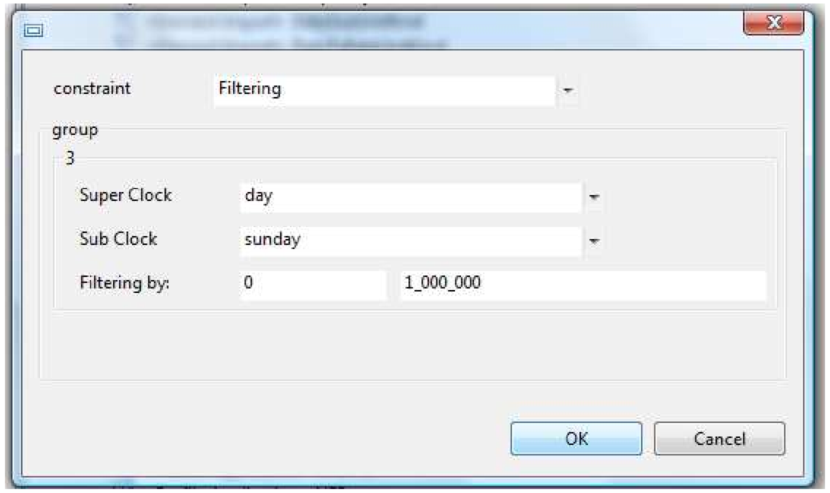
- Modeling and Analysis of timed systems

- Fully supports MARTE Time model
  - UML Profile for Modeling and Analysis of RTE systems
  - Logical and multiform time
- **C**lock **C**onstraint **S**pecification **L**anguage
  - Formal Timed extension to OCL

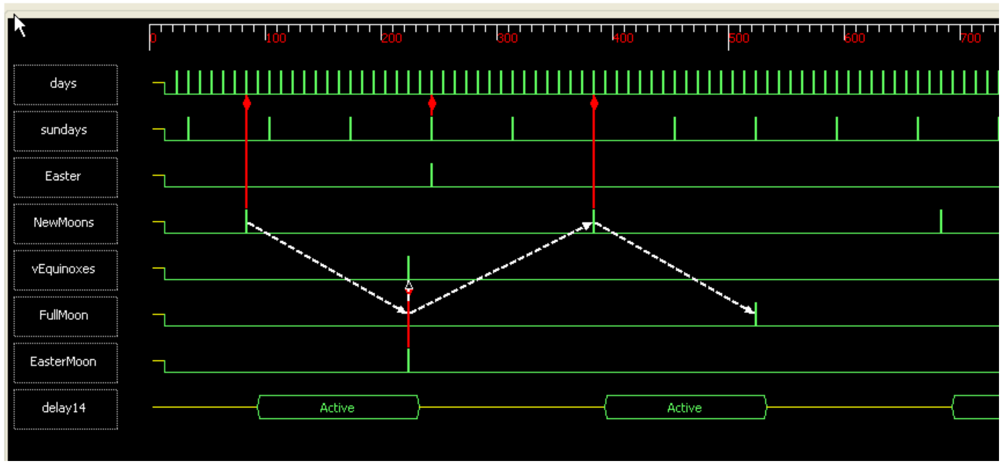
- Detects requirement inconsistencies
- Exhibits one valid behavior (simulation)

TimeSquare functionality

1. Interactive clock-related specifications
2. Clock constraint checking
3. Generation of sequences of steps
4. Displaying & exploring waveforms



constraint	Filtering
group	
3	
Super Clock	day
Sub Clock	sunday
Filtering by:	0 | 1_000_000

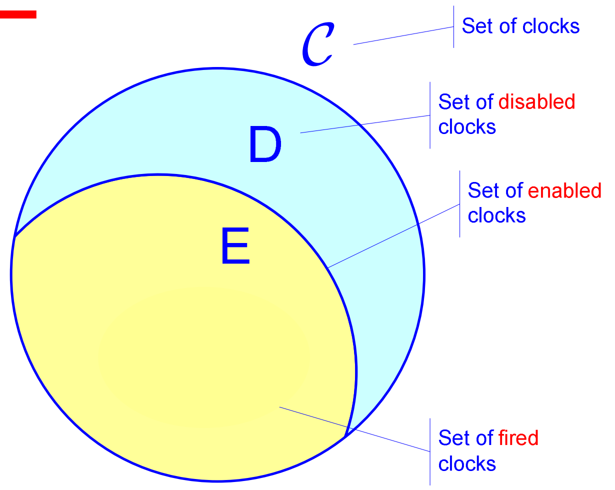


VCD-compliant

From constraints to behavior (1/2)

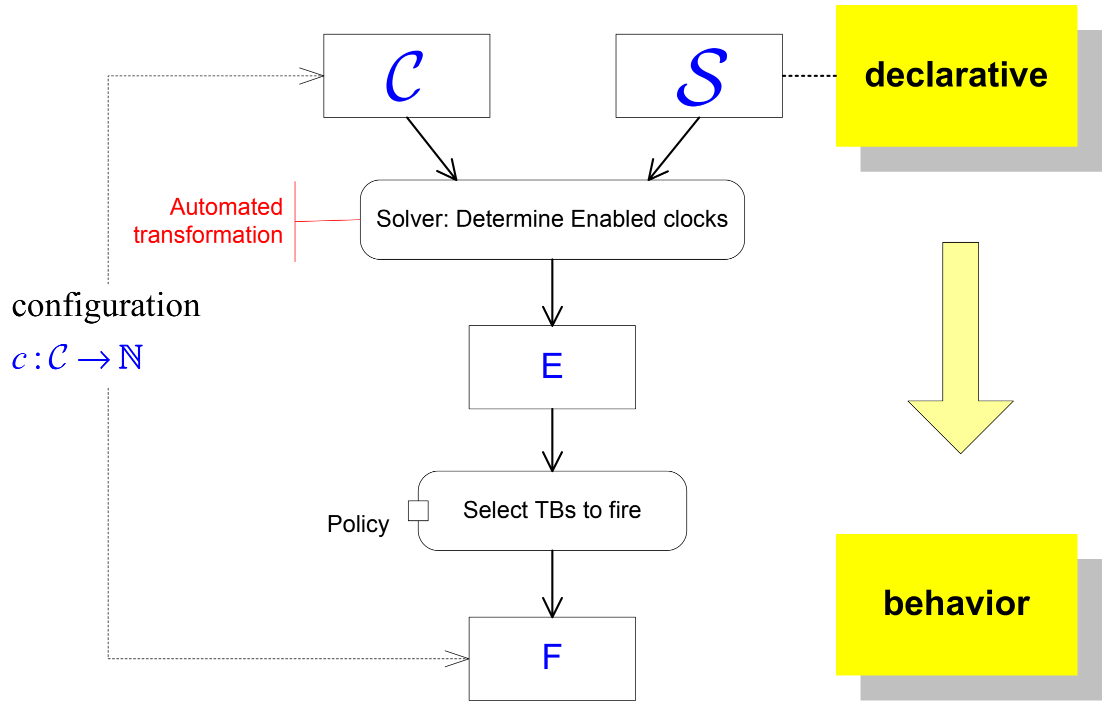
- At each simulation step, three phases:
  1. For each constraint determine the set of implied "Boolean equations"
  2. When all constraints are analyzed, determine the set of enabled clocks (E)
  3. Select the set of clocks to fire (F)

Phase 3 is not necessarily deterministic
=> simulation policy (random, min, max/asap, …)



For a given configuration c

From constraints to behavior (2/2)



Objective: build sequences of steps that respect $\mathcal{S}$

Solution: SOS for a Kernel of CCSL. $\mathcal{S}, c \xrightarrow{F} \mathcal{S}', c'$

User's viewpoint: standard CCSL library provided

+ facility for user's defined constraints + stochastic parameters

CCSL kernel (1/2)

Clock & Instant **Expressions**

Terminating CExpr

$\tau^0 ::= !1$	// forcing
$\mid !0$	// inhibition
$\mid t \wedge n$	// waiting next nth t
$\mid t \square_{\bullet} t$	// (strict) sampling
$\mid t \square_{\circ} t$	// non-strict sampling
$\mid t ↯ t$	// t upto t

Non-terminating CExpr, c-independent

$\tau^1 ::= t$	// simple time base reference
$\mid t \bullet t$	// concatenation
$\mid t + t$	// union = coarsest finer $\left(\mathrm{Sup}_{\subseteq}\right)$
$\mid t * t$	// intersection
$\mid t, \sigma \rightsquigarrow t$	// defer

Non-terminating CExpr, c-dependent

$\tau^2 ::= t \vee t$	// sup = fastest slower $\left(\mathrm{Sup}_{\preccurlyeq}\right)$
$\mid t \wedge t$	// inf = slowest faster $\left(\mathrm{Inf}_{\preccurlyeq}\right)$

IExpr

$\iota ::= t @ n$	// n^{th} instant of t
$\mid t @ \mathrm{end}$	// last instant of a finite time base

CExpr

$\tau ::= \tau^0 \mid \tau^1 \mid \tau^2$	// simple clock expressions
$\mid \text{if } b \text{ then } \tau \text{ else } \tau$	// conditional clock expression

CCSL kernel (2/2)

Clock & Instant **Relations**

Definition CRel

$\rho^d ::= t \square \tau$ // clock standard definition

$\mid t \overset{\circ}{=} t \bullet t$ // clock recursive definition

Basic CExpr, c-independent

$\rho^1 ::= t \left(\boxed{\subset} \mid \boxed{\multimap} \mid \boxed{\#} \right) t$ // basic constraints on clocks, cnt-independent

Basic CExpr, c-dependent

$\rho^2 ::= t \left(\boxed{\equiv} \mid \boxed{\prec} \mid \boxed{\preccurlyeq} \right) t$ // basic constraints on clocks, cnt-dependent

$\mid \iota \left(\equiv \mid \multimap \mid \# \mid \prec \mid \preccurlyeq \right) \iota$ // basic constraints on instants

Clock Constraint

$\gamma ::= \gamma \mid \gamma$ // constraint conjunction

$\mid \rho^d \mid \rho^1 \mid \rho^2$ // simple relation

$\mid \rho \text{ if } b$ // conditional relation

Example of semantics rule

Structure $\rightarrow$
Boolean Eqn System
$\Rightarrow$ E

Boolean equations implied by t_1 under c

$$\frac{t_1, c \vdash b, a \qquad \beta \,\square\, (n = 1)}{t \cong t_1 \,\hat{}\, n, c \vdash b \wedge \text{ite}(\beta, t = t_1, \neg t), a} \;(\text{next})$$

behaves like

Boolean equations implied by t under c

F $\Rightarrow$ Rewritten Structure

$$t \cong t_1 \,\hat{}\, 1 \xrightarrow{t_1 \in F} t \cong 1 \quad (\text{RWnext 1})$$

$$\frac{n > 1 \quad t_1 \xrightarrow{t_1 \in F} t_2}{t \cong t_1 \,\hat{}\, n \xrightarrow{t_1 \in F} t \cong t_2 \,\hat{}\, (n-1)} \quad (\text{RWnext 2})$$

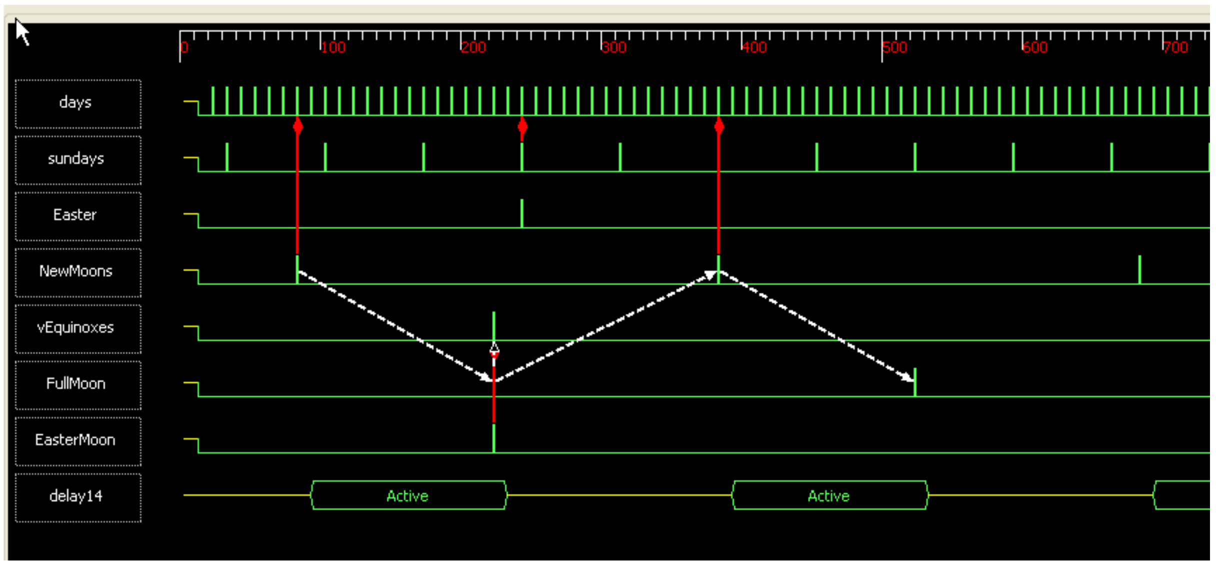
Conditional rewriting rules

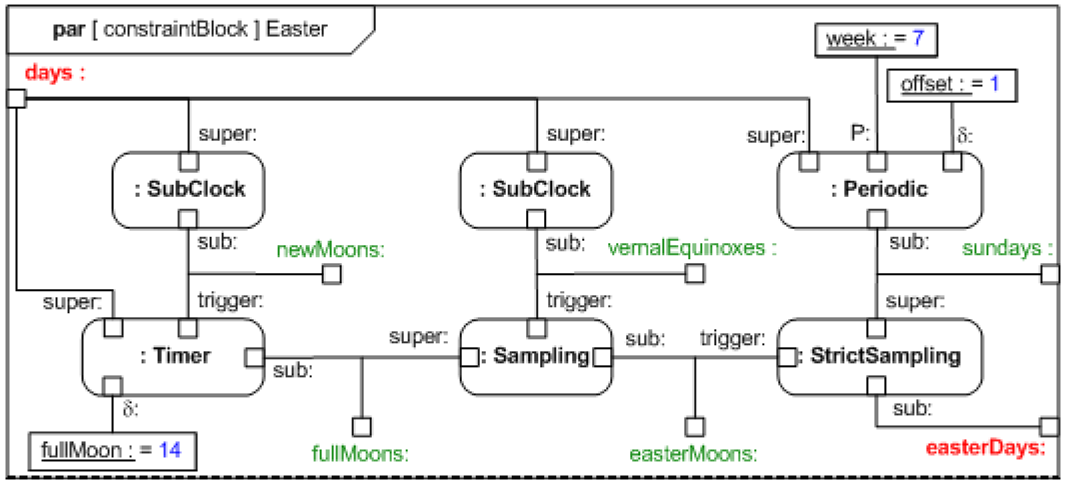
Implementation

- Plug-in developed with Ganymede Eclipse Modeling Tools
- CCSL parser: ANTLR 2.7
- Solver: JavaBDD
- Waveforms: VCD compliant (IEEE Std1364)
- Available at:

http://www-sop.inria.fr/aoste/dev/time_square/

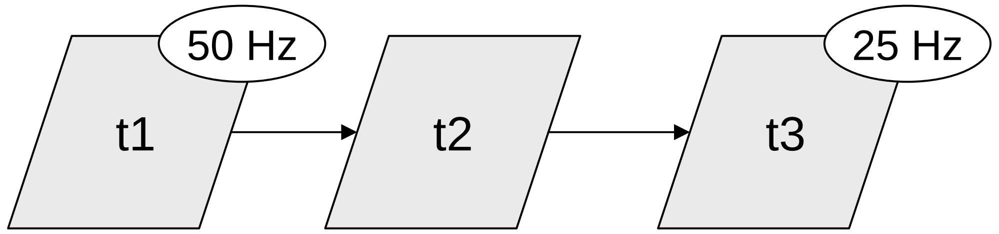
Example 1: Easter

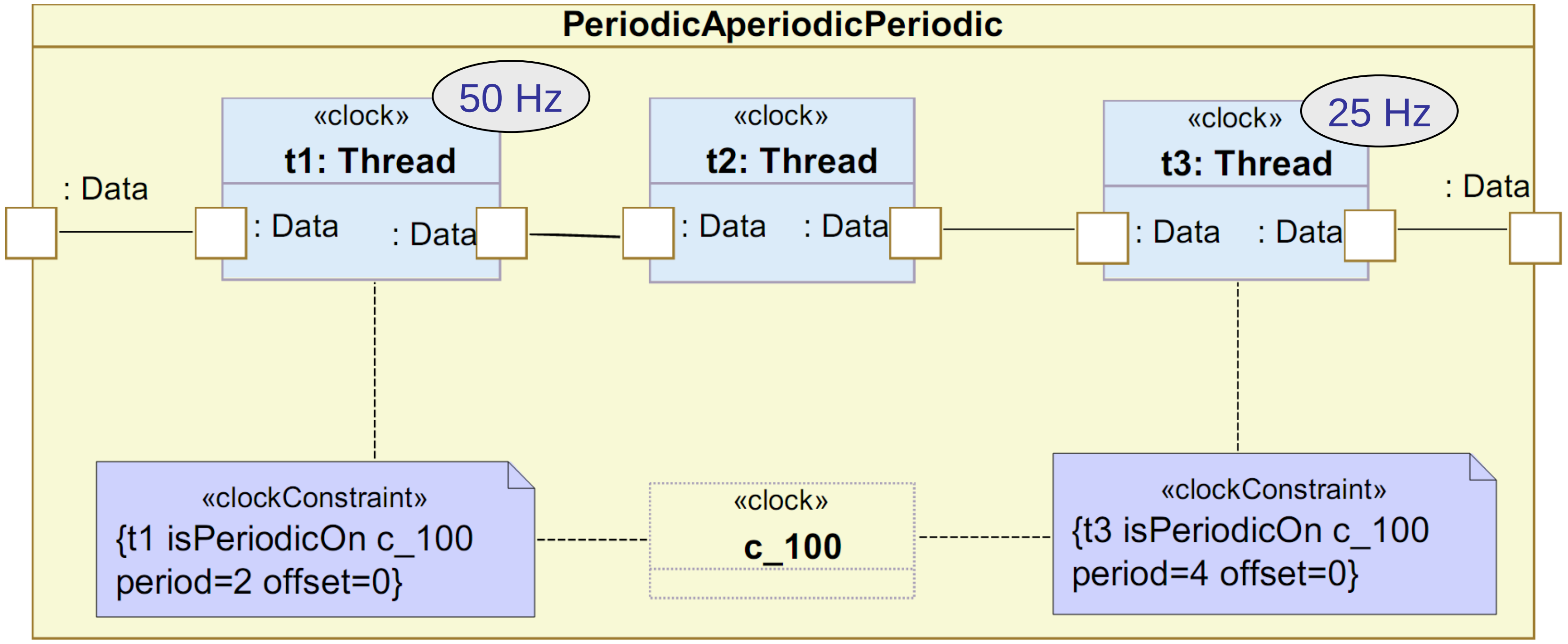




Visual representation ?

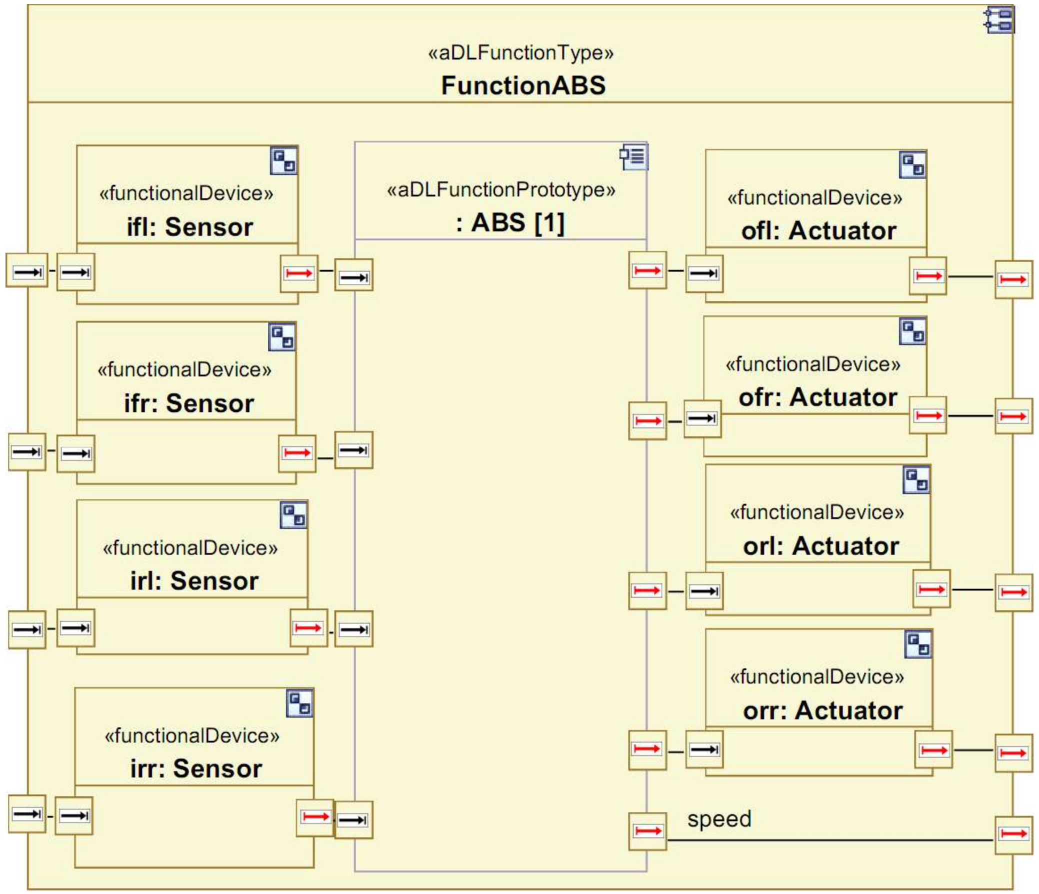
Example 2: AADL





c_100 = IdealClk **discretizedBy** 0.01

Example 3: ABS and East-ADL2

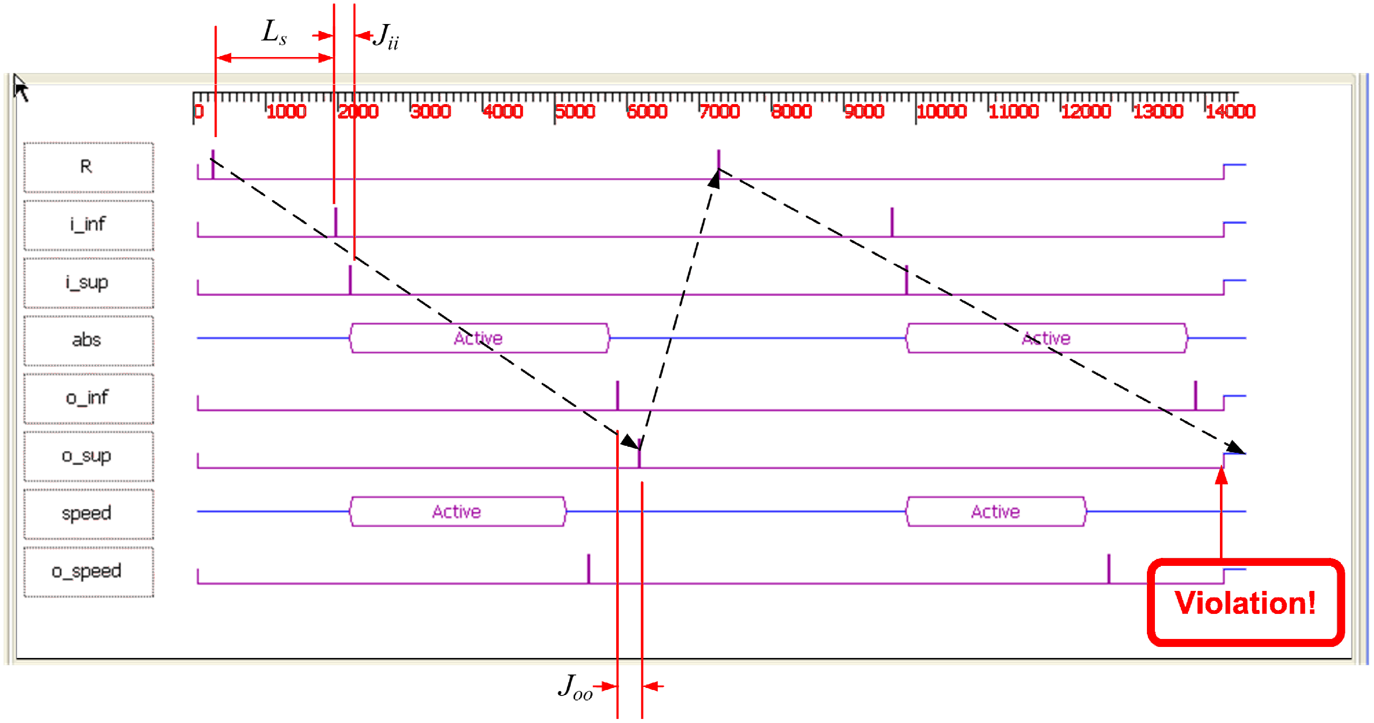


Clocks associated with ports

Stochastic durations for communications and executions

Example 3: ABS and East-ADL2

- Gives a formal semantics to East-ADL2 timing requirements
  - Make the requirements executable



Example 4: Digital Filter (1/2)

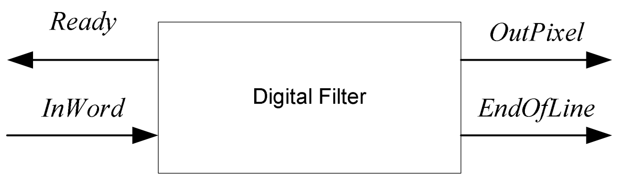
Specification

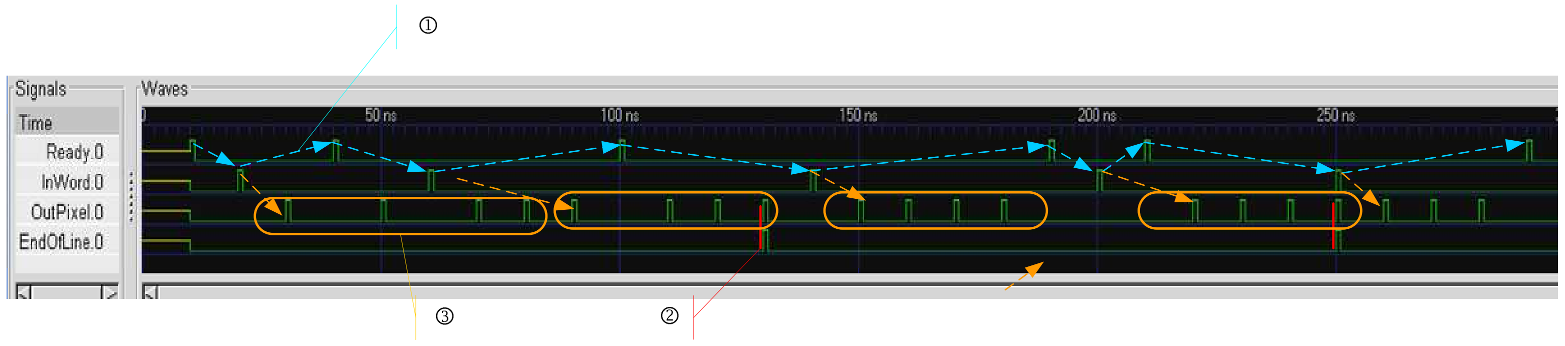
① $Ready \,\square\, InWord$

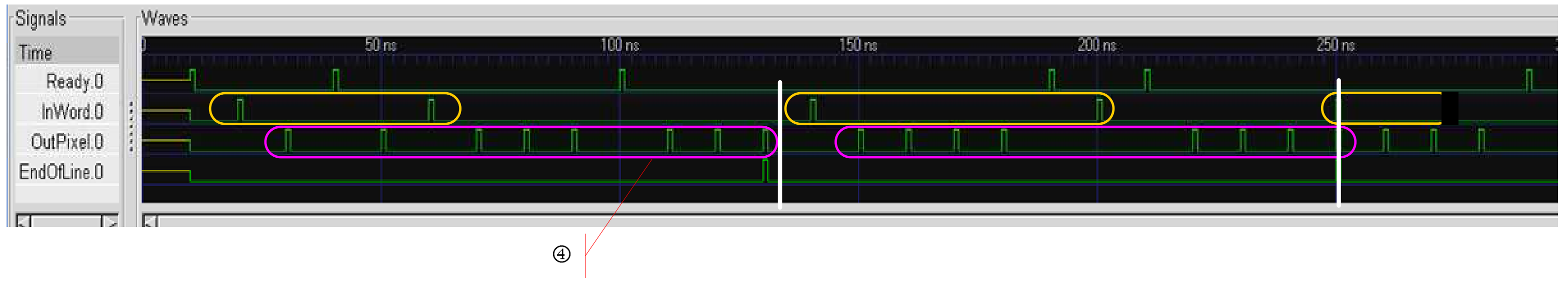
② $EndOfLine = OutPixel \blacktriangledown \left(0^{LINE\_LENGTH-1}.1\right)^{\omega}$

③ $InWord \prec OutPixel / PIXELS\_PER\_WORD$

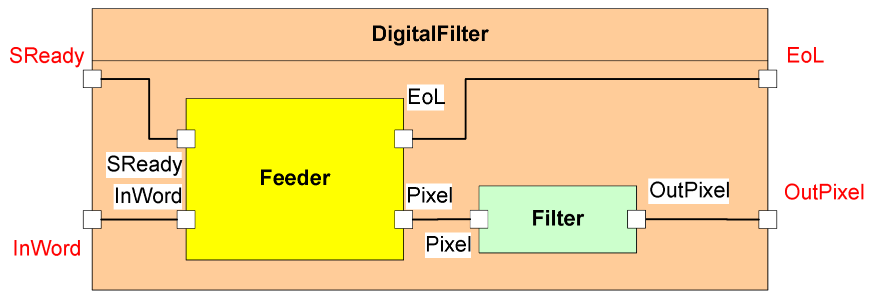
④ $InWord / WORDS\_PER\_LINE \rhd\lhd_{\equiv} OutPixel / LINE\_LENGTH$



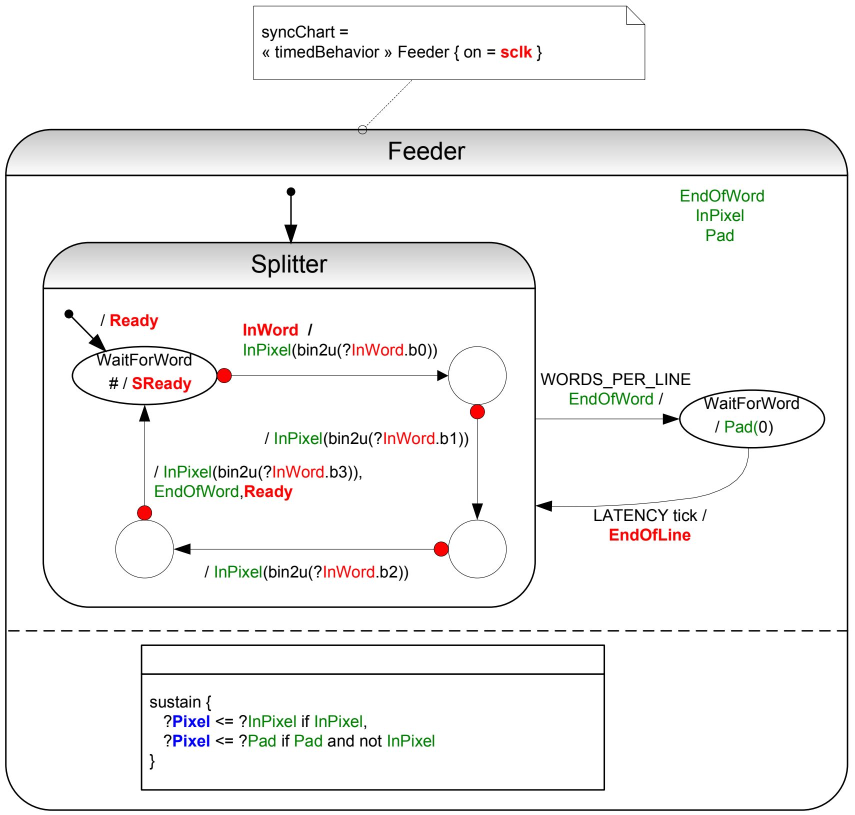




Example 4: Digital Filter (2/2)



$$
\begin{aligned}
\{\ & \text{Clock } \mathit{InPixel}, \mathit{Pad}, \mathit{EndOfWord}, \mathit{FirstInstant}, \mathit{Pixel} \\
& \mathit{FirstInstant} = \mathit{sclk} \blacktriangledown 1 \\
& \mathit{InWord} = \mathit{InPixel} \blacktriangledown \left(1.0^{3}\right)^{\omega} \\
& \mathit{EndOfWord} = \mathit{InPixel} \blacktriangledown \left(0^{3}.1\right)^{\omega} \\
& \mathit{InPixel} = \mathit{Pixel} \blacktriangledown \left(1^{8}.0^{2}\right)^{\omega} \\
& \mathit{Pad} = \mathit{Pixel} \blacktriangledown \left(0^{8}.1^{2}\right)^{\omega} \\
& \mathit{EndOfLine} = \mathit{Pixel} \blacktriangledown \left(0^{9}.1\right)^{\omega} \\
& \mathit{OutPixel} = \mathit{InPixel} \text{ delayedFor 2 on } \mathit{Pixel} \\
& \mathit{Ready} = \mathit{FirstInstant} + \mathit{EndOfWord} \\
& \mathit{SReady} = \text{sustain } \mathit{Ready} \text{ upto } \mathit{InWord} \\
\}\ &
\end{aligned}
$$



syncChart =
« timedBehavior » Feeder { on = sclk }

```
sustain {
   ?Pixel <= ?InPixel if InPixel,
   ?Pixel <= ?Pad if Pad and not InPixel
}
```

Glitch detected