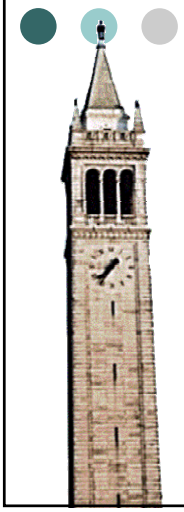


Concurrent Models of Computation for Embedded Software

Edward A. Lee
Professor, UC Berkeley
EECS 290n – Advanced Topics in Systems Theory
Fall, 2004

Lecture 22: Clocks in Synchronous Languages

Synchronous Languages

- Esterel
- Lustre
- SCADE (visual editor for Lustre)
- Signal
- Statecharts (some variants)
- Ptolemy II SR domain

The model of computation is called *synchronous reactive* (SR). It has strong formal properties (many key questions are decidable).

The Synchronous Abstraction

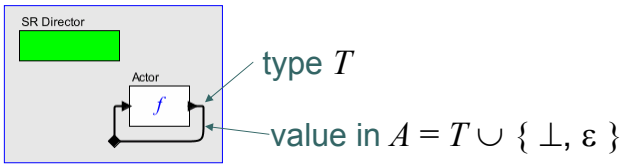
- "Model time" is discrete: Countable ticks of a clock.
- WRT model time, computation does not take time.
- All actors execute "simultaneously" and "instantaneously" (WRT to model time).
- There is an obviously appealing mapping onto real time, where the real time between the ticks of the clock is constant. Good for specifying periodic real-time tasks.

Simple Execution Policy

At each tick, start with all signals "unknown." Evaluate non-strict actors and source actors. Then keep evaluating any actors that can be evaluated until all signals become known or until no further progress can be made.

Note that signals will resolve to a value or to "absent" if there are no causality loops.

Fixed Point Semantics



At each tick of the clock

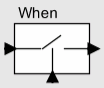
- Start with signal value $\perp$ (unknown)
- Evaluate $f(\perp)$
- Evaluate $f(\ f(\perp))$
- Stop when a fixed point is reached

  A fixed point is always reached in a finite number of steps (one, in this case).

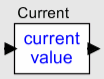
Synchronous/Reactive Actors

Key SR Actors

Pre: When the input is present, the output is the previous present input value.

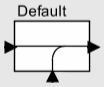


When: When the bottom input is present and true, the output equals the input. Otherwise, the output is absent.



Current: The output equals the most recent present input value.

NonStrictDelay: The output is equal to the input in the previous clock tick.



Default: The output equals the left input, if it is present, and the bottom input otherwise.

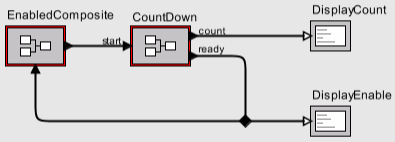
EnabledComposite: Composite actor whose internal clock ticks only when the bottom input is present and true.

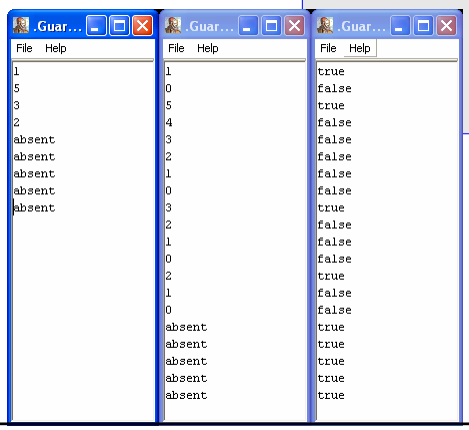
Use of some of these can be quite subtle.

Design in SR: Example

SR Director

This model illustrates the use of SR primitive actors to make a CountDown actor. This (composite) actor outputs a true on the ready port when it is ready to count. In the same tick of the clock, the Sequence actor provides it with a starting number. It then counts down to zero on each subsequent tick of the clock, emitting true on ready when it again reaches zero.





The three displays show (left to right):

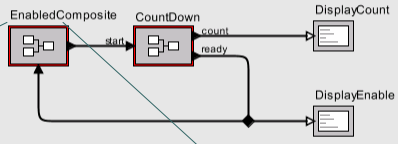
- Requested numbers to count down from.
- The count down for these numbers.
- The enable signal for the EnabledComposite actor.

In this example, the CountDown composite issues a “ready” signal to the EnabledComposite, which then issues a number. The CountDown composite counts down from that number to 0, then issues another ready.

Design in SR: Example

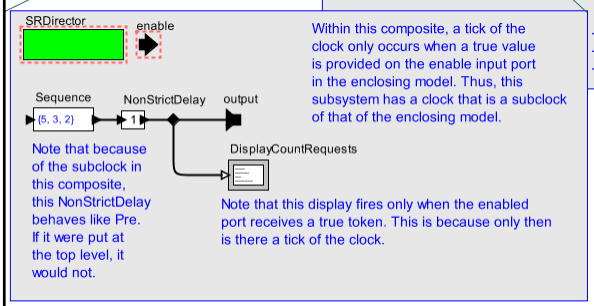
SR Director

This model illustrates the use of SR primitive actors to make a CountDown actor. This (composite) actor outputs a true on the ready port when it is ready to count. In the same tick of the clock, the Sequence actor provides it with a starting number. It then counts down to zero on each subsequent tick of the clock, emitting true on ready when it again reaches zero.



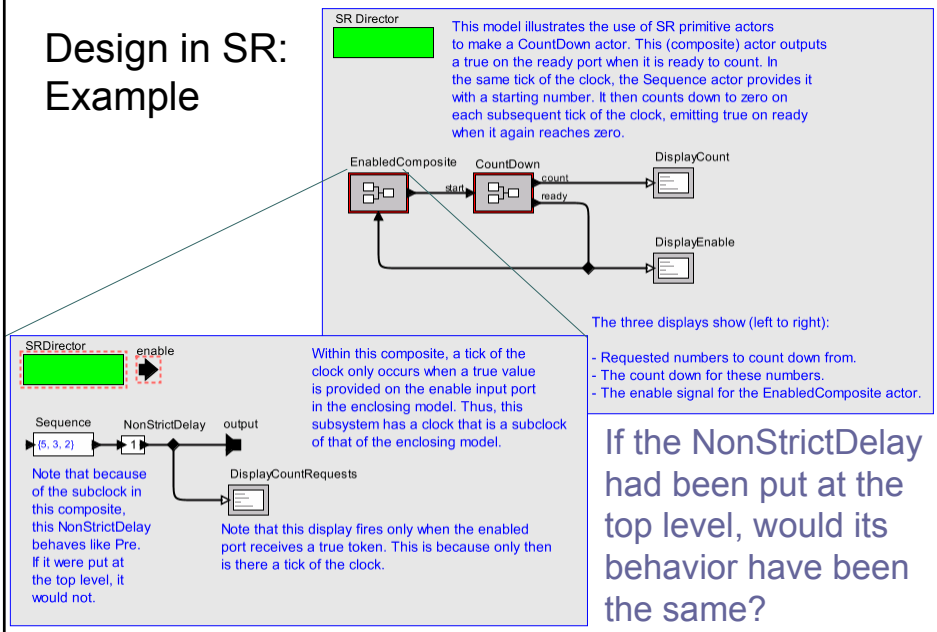
The three displays show (left to right):

- Requested numbers to count down from.
- The count down for these numbers.
- The enable signal for the EnabledComposite actor.



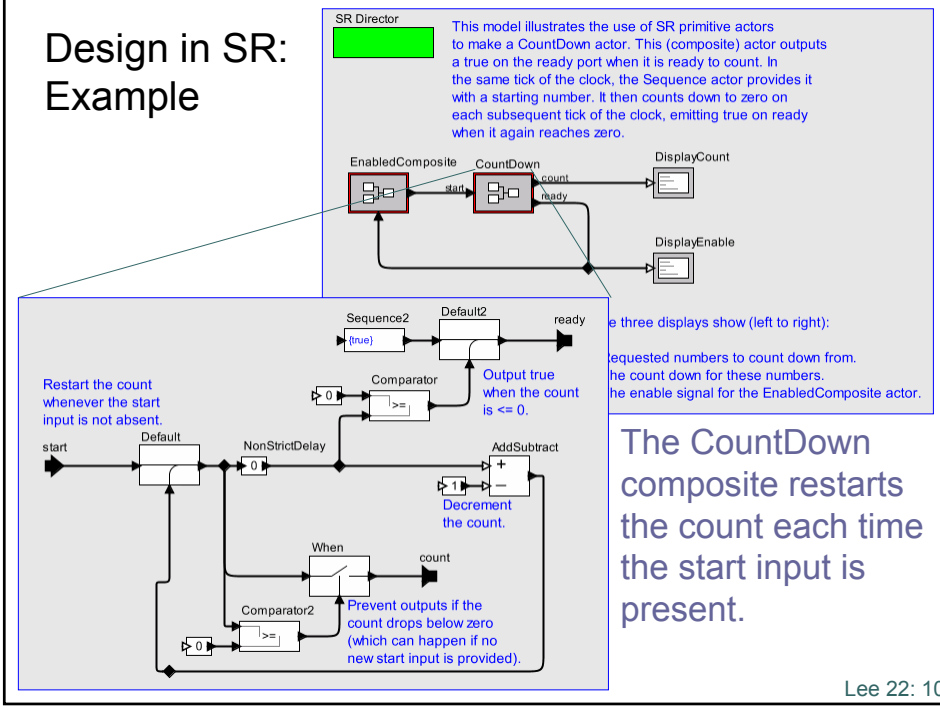
The EnabledComposite has a clock that ticks only when the enable input is present and true. It issues the sequence 1, 5, 3, 2, followed by absent henceforth.

Design in SR: Example



If the NonStrictDelay had been put at the top level, would its behavior have been the same?

Design in SR: Example



The CountDown composite restarts the count each time the start input is present.

Subtleties: Pre vs. NonStrictDelay

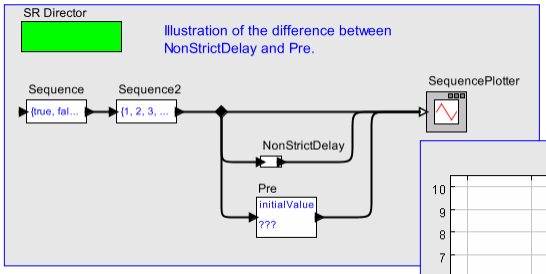
Pre: When the input is present, the output is the previous present input value.

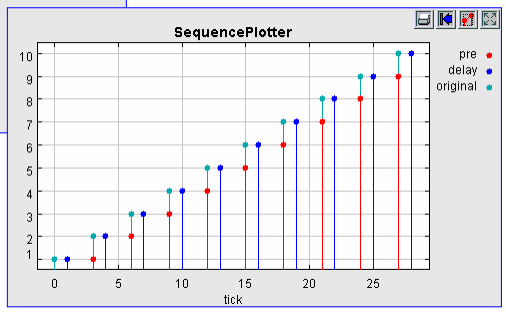
NonStrictDelay: The output is equal to the input in the previous clock tick.

Pre: True one-sample delay. The behavior is not affected by insertion of an arbitrary number of ticks with "absent" inputs between present inputs.

NonStrictDelay: One-tick delay (vs. one-sample). The output in each tick equals the input in the previous tick (whether absent or not).

Illustration of this Subtlety





In this example, the original signal is present only if every third tick of the clock. The output of the NonStrictDelay is delayed by one click, whereas the output the Pre actor is delayed by one (present) sample.

Consequences: Pre vs. NonStrictDelay

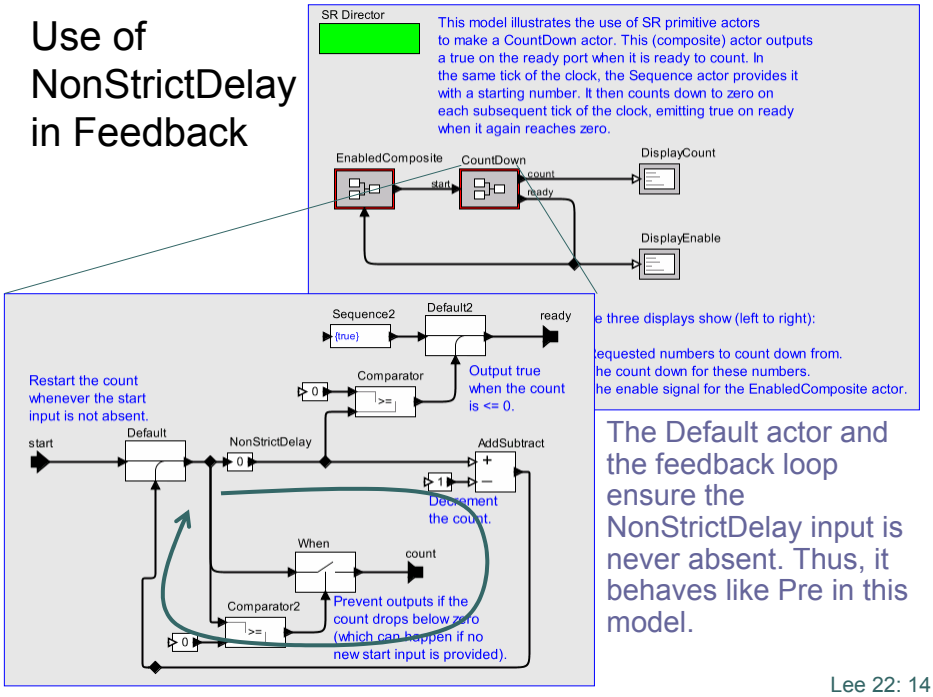
Pre: When the input is present, the output is the previous present input value.

NonStrictDelay: The output is equal to the input in the previous clock tick.

Pre: This actor is *strict*. It must know whether the input is present before it can determine the output. Hence, it cannot be used to break feedback loops.

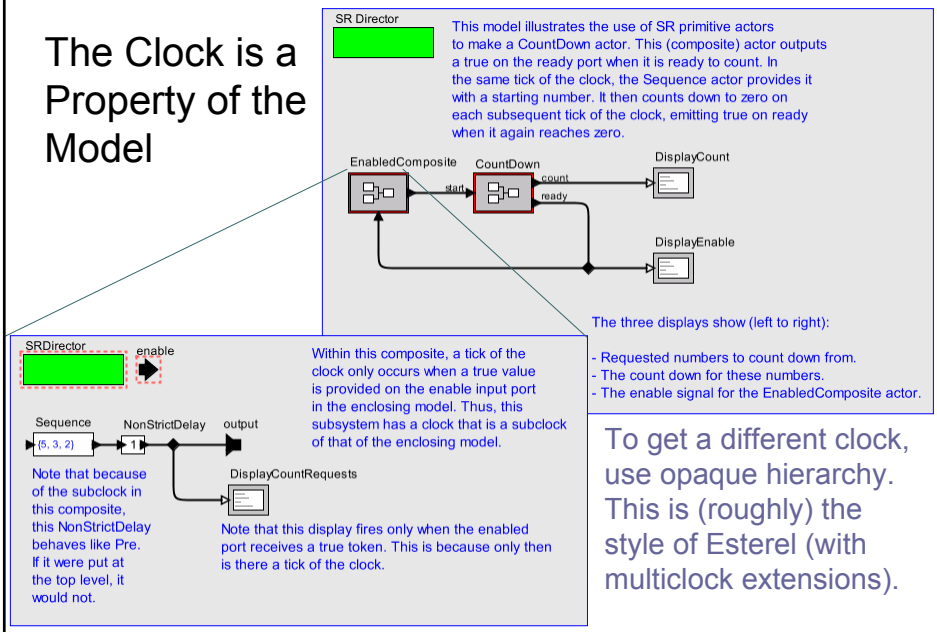
NonStrictDelay: This actor is *nonstrict*. It need not know whether the input is present nor what its value is before it can determine the output. Hence, it can be used to break feedback loops.

Use of NonStrictDelay in Feedback



The Default actor and the feedback loop ensure the NonStrictDelay input is never absent. Thus, it behaves like Pre in this model.

The Clock is a Property of the Model



To get a different clock, use opaque hierarchy. This is (roughly) the style of Esterel (with multiclock extensions).

Hierarchical Clock Domains

Opaque hierarchy can do:

- Conditioning an internal tick on an external signal
  - Like a conditional
  - If the internal component is an instance of the external, then this amounts to recursion
- Multiple internal ticks per external tick
  - Like a do-while
- Iterated internal ticks over a data structure (use IterateOverArray higher-order actor)
  - Like a for

Alternative Semantics: The Clock is a Property of the Signal

In Lustre and Signal, a clock is a property of a signal, and Pre and NonStrictDelay could (in theory) behave identically. They would only "tick" when the clock of the input signal ticked.

However, this model has problems with decidability. Clocks cannot always be inferred.

Clock Calculus

- Let T be a well founded totally ordered set of tags.
- Let $s: T \rightarrow V \cup \{\varepsilon\}$ be a signal of type V, where ε means "absent."
- Let $c: T \rightarrow \{-1, 0, 1\}$ be a *clock* associated with s where

$$s(t) = \varepsilon \Rightarrow c(t) = 0$$
$$s(t) = true \Rightarrow c(t) = 1$$
$$s(t) = false \Rightarrow c(t) = -1$$

If V is not boolean, then when $s(t)$ is present, $c(t)$ has value or 1 or -1 (we will make no distinction).

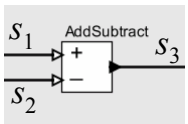
Operations on Clocks

Arithmetic on clocks is in GF-3 (a Galois field with 3 elements), as follows:

$$0 + x = x \qquad 0 \cdot x = 0$$
$$1 + 1 = -1 \qquad 1 \cdot x = x$$
$$-1 + -1 = 1 \qquad -1 \cdot x = -x$$
$$-1 + 1 = 0$$

Clock Relations: Simple Synchrony

Most actors require that the clocks on all signals be the same. For example:

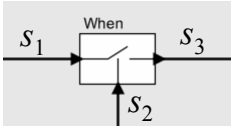


$$\forall t \in T, \quad c_1^2(t) = c_2^2(t) = c_3^2(t)$$

This means that either all are present, or all are absent.

Clock Relations: When Operator

Assuming that s_1 is a boolean-valued signal (which it must be), the clocks on signals interacting through the when operator are related as follows:



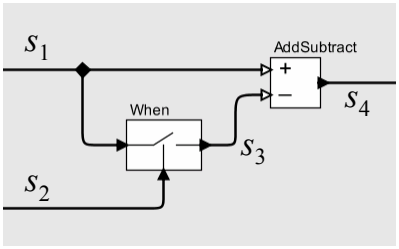
$$\forall t \in T, \quad c_3(t) = c_1(t)(-c_2(t) - c_2^2(t))$$

This means:

- If s_1 is absent, then s_3 is absent.
- If s_2 is false, then s_3 is absent.
- If s_2 is true, then s_3 is the same as s_1.

Consistency Checking

Consider the following model:



$$\forall t \in T, \quad c_1^2(t) = c_4^2(t) = c_3^2(t)$$

$$\forall t \in T, \quad c_3(t) = c_1(t)(-c_2(t) - c_2^2(t))$$

These two together imply that:

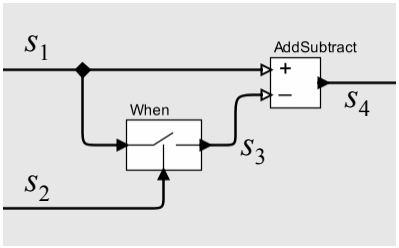
$$\forall t \in T, \quad c_2^2(t)(1 + c_1^2(t)) = -c_2(t)c_1^2(t)$$

where we have used the fact that:

$$(-c_2(t) - c_2^2(t))^2 = (-c_2(t) - c_2^2(t))$$

Interpretation of Consistency Result

Consistency check implies that:



$$\forall t \in T, \quad c_2^2(t)(1+c_1^2(t)) = -c_2(t)c_1^2(t)$$

This means:

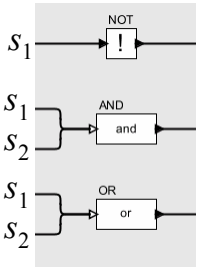
s_1 is absent if and only if s_2 is absent.

if s_2 is present, then s_2 is true.

Logic Operators Affect Clocks

The output of the When actor has a clock that depends on the Boolean control signal. Clocks of Boolean-valued signals reflect the signal value as follows:

$$\forall t \in T,$$



$$c_2(t) = -c_1(t)$$

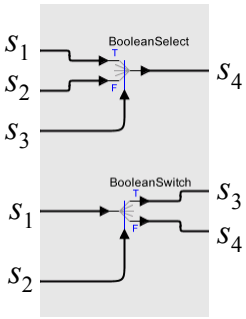
$$c_3(t) = (c_1(t)c_2(t))^2(-(c_1(t)+1)(c_2(t)+1)-1)$$

$$c_3(t) = (c_1(t)c_2(t))^2((c_1(t)-1)(c_2(t)-1)+1)$$

Token Routing Also Affects Clocks

Switch and Select affect the clocks as follows:

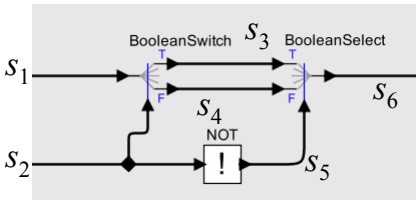
$$\forall t \in T,$$



$$c_4(t) = c_3(t)(c_2(t)(1-c_3(t)) - c_1(t)(1+c_3(t)))$$
$$-(c_3(t)+1)c_3(t) = c_1^2(t)$$
$$-(c_3(t)-1)c_3(t) = c_2^2(t)$$

$$c_3(t) = -c_2(t)(c_2(t)+1)c_1(t)$$
$$c_4(t) = c_2(t)(1-c_2(t))c_1(t)$$
$$c_2^2(t) = c_1^2(t)$$

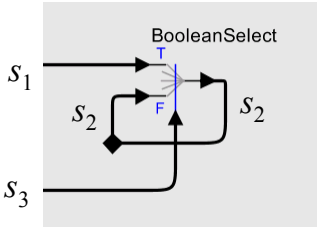
Example 1 Using Switch and Select



What can you infer about the clock of s_6?

$$c_6(t) = 0$$

Example 2 Using Switch and Select

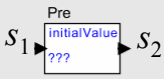


What can you infer about the clocks?

$$c_1(t) = 0 \quad \text{and}$$
$$\text{either} \quad c_3(t) = 0 \quad \text{or} \quad 1 + c_3(t) = 0$$

This means that s_1 is absent and s_3 is either absent or false.

What About Delays?



Pre: When the input is present, the output is the previous present input value.

NonStrictDelay: The output is equal to the input in the previous clock tick.

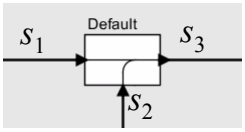
Clock relations across the delays become dependent on the tags. E.g., if T is the natural numbers, then we get a nonlinear dynamical system:

$$c_1^2(t) = c_2^2(t) \quad \text{and}$$
$$c(0) = \text{initial state}$$
$$c(t+1) = (1 - c_1^2(t))c(t) + c_1(t)$$
$$c_2(t) = c_1^2(t)c(t)$$

This makes clock analysis very difficult, in general.

Default Operator

Default: The output equals the left input, if it is present, and the bottom input otherwise:



$$\forall t \in T, \quad c_3(t) = c_1(t) + c_2(t)(1 - c_1^2(t))$$

This means the clock of s_3 is equal to the clock of s_1, if it is present, and to the clock of s_2 otherwise.

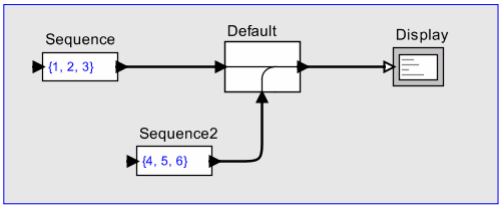
SIGNAL Clock System

In the SIGNAL language, the clock system is richer:

- Let T be a partially ordered set of tags.
- A signal s: $T \rightarrow V \cup \{\varepsilon\}$ of type V is a *partial function* defined on a totally ordered subset of T, where again ε means "absent."

Default Operator in SIGNAL is Nondeterministic

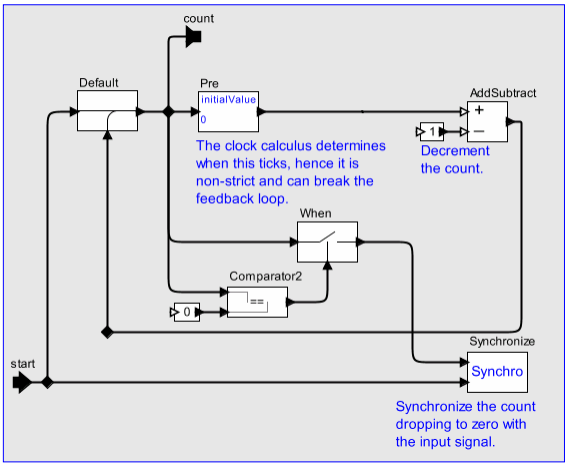
In SIGNAL semantics, the following model has many behaviors:



The two generated sequences have independent clocks (defined over incomparable values of $t \in T$), and the output sequence is any interleaving that preserves the ordering.

Guarded Count in SIGNAL

Instead of generating a "ready" signal, in SIGNAL, the count hitting zero can be synchronized with the input being present.



Conclusion and Open Issues

- When clocks are a property of the model, the result is structured synchronous models, where differences between clocks are explicit and no consistency checks are necessary.
- When clocks are a property of a signal, the result is similar to Boolean Dataflow (BDF). It is arguable that clock operators like “when,” “default,” “switch,” and “select” become analogous to unstructured gotos. Clock consistency checking becomes undecidable.
- When further extended as in SIGNAL to partially ordered clock ticks, models easily become nondeterministic.