Lustre
Semantics of Synchronous Systems

Caspi D. Pilaud, N. Halbwachs, J.A. Plaice
Presented by Gregory M. Malecha

Rice University

March 11, 2008

Motivation

- Uccello can be used for circuit design
- To be able to evaluate Uccello programs, we need to understand stage 0 computation
- What are the semantics for cycles in Uccello?
- Can we model clocks in some way?

Lustre

- Lustre is a programming language for time-driven systems
- Primarily used for automatic control and signal processing systems
- Similar programming paradigm to Uccello (nodes and wires)
- A program is a system of equations
- Order of evaluation based on dependencies between variables

Streams

- Constants are streams

$$1 \Rightarrow \{1, 1, 1, ...\}$$

- Operators operate piecewise on stream elements

$$\begin{aligned} 1 + 2 &\Rightarrow \{1, 1, 1, ...\} + \{2, 2, 2, ...\} \\ &\Rightarrow \{1 + 2, 1 + 2, 1 + 2, ...\} \\ &\Rightarrow \{3, 3, 3, ...\} \end{aligned}$$

- We'll denote the element of stream X at time t as X_t.

Memory/State

- Streams can be shifted forward

$$pre(X) = \{\text{nil}, X_0, X_1, X_2, ...\}$$

- Assume that

$$X \Rightarrow \{1, 2, 3, 4, 5, ...\}$$

  then

$$pre(X) \Rightarrow \{\text{nil}, 1, 2, 3, 4, ...\}$$

- nil represents an undefined value
- All operations on nil fail

Initial Conditions

- $pre(X)_0 \Rightarrow$ nil
- We can think of nil as undefined initial conditions
- We need to eliminate nil
- Use the "followed by" operator (->)

$$0- > X \Rightarrow \{0, X_1, X_2, X_3, ...\}$$

- *Replaces* first element in right stream with left stream, e.g.

$$(X\text{->}Y)_k = \begin{cases} X_0 & k = 0 \\ Y_k & k > 0 \end{cases}$$

Simple Example

- Construct $\{0, 1, 2, 3, ...\}$ (count up)

```
COUNT_UP = 0 -> (1 + pre(COUNT_UP))
```

- Construct $\{0, 1, 1, 2, 3, 5, ...\}$ (Fibonacci)

```
FIB = 0 -> (1 -> (pre(FIB) + pre(pre(FIB))))
```

- pre introduces memory
- -> sets an initial condition

Clocking Streams

- Clocks are boolean streams
- Use when to associate a clock with a stream
- Streams are undefined except for when their clock is true
- Example (a clock which samples on even cycles)

```
EVEN = true -> (not pre(EVEN))
```

- Example:

$$\begin{array}{rcl} B & \Rightarrow & \{\mathit{true}, \quad \mathit{false}, \quad \mathit{true}, \quad \mathit{false}, \quad \mathit{true}, \quad \ldots\} \\ \mathit{FIB}\ \texttt{when}\ B & \Rightarrow & \{0, \quad\quad , \quad\quad 1, \quad\quad , \quad\quad 1, \quad \ldots\} \end{array}$$

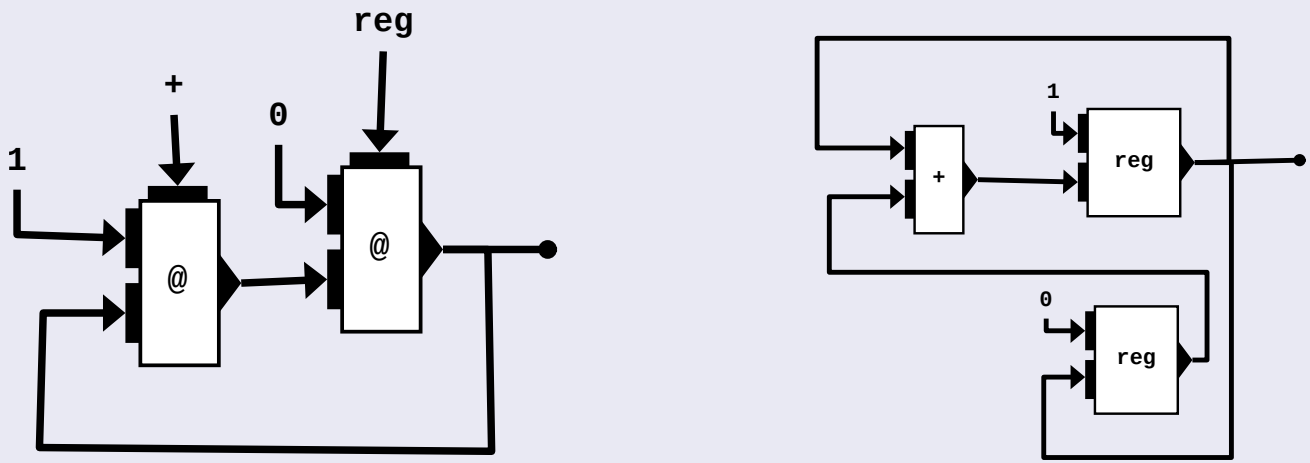
Clock Consistency

- Operations are only well defined if they are on the same clock
- Otherwise values become "available" at different times which causes problems
- Since we can't, in general, decide the equivalence of two infinite binary streams, static checking requires using the same expression for both clocks

Connections to Uccello

- Uccello renditions of COUNT-UP and FIB

Registers



- *Lustre's* notion of clocks and variables could be a starting point for converting VPP semantics to Uccello semantics
- As time approaches ∞, semantics converge to lazy semantics

$$\textbf{let}\ \ x = 1 + x\ \ \textbf{in}\ \ x$$

Summary

- Lustre is a synchronous language (deals with clocks)
- Computation is modeled on streams of values
- Streams can be delayed (with pre) or shifted (with ->)
- These ideas are a starting point for understanding Uccello cycles (based on VPP semantics)