EE249 Discussion: Synchronous Modeling

David Burnett
Wei Yang Tan

Synchronous Approach to Reactive and Real-Time Systems

Some Definitions

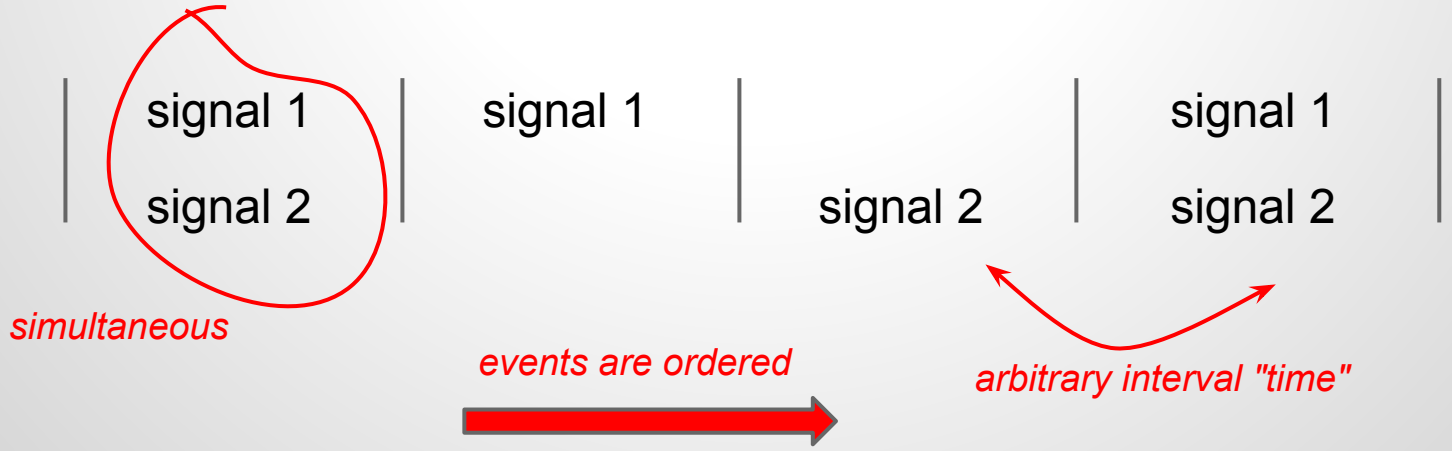
- Reactive systems
  - maintains a permanent interaction with its environment
  - for e.g. classical communication protocols
- Real-Time systems
  - *reactive systems* that are also subject to externally defined timing constraints
  - for e.g. car, air-traffic control

Inadequacies in Classical Techniques

- Lack of support for concurrency
  - e.g. finite-state machine (FSM)
- No modularity in structure -> not scalable
  - e.g. Petri Nets, FSM
- Not deterministic
  - e.g. Petri Nets, OS primitives, classical concurrent programming language (ADA)
- No formal techniques for specifications / verifications
  - e.g. using OS primitives for communications

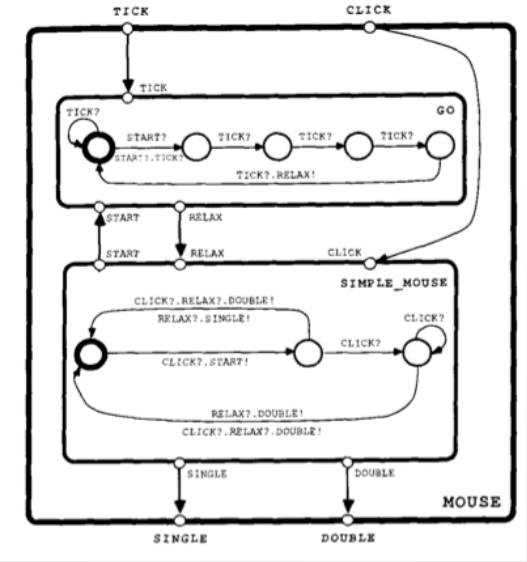
New Synchronous Modeling Approach

- Output is synchronous with input
- Internal actions are instantaneous
- Communications are performed via instantaneous broadcasting
- Environment signals is modeled in a form of global interleaving:



State-based Formalism

- E.g. Statecharts, ESTEREL
- Easy to adopt when control flow is prevalent
- But defining behavior of a concurrent composition is difficult



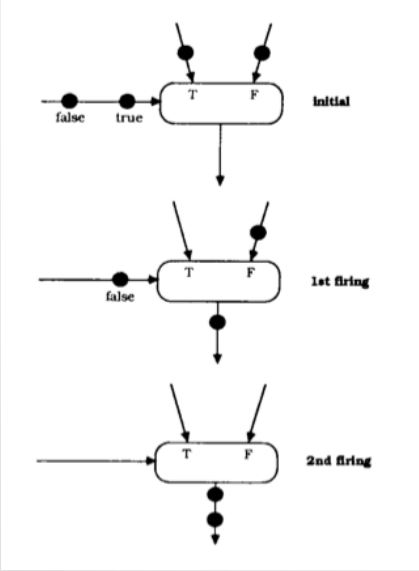
Dataflow-based Approach

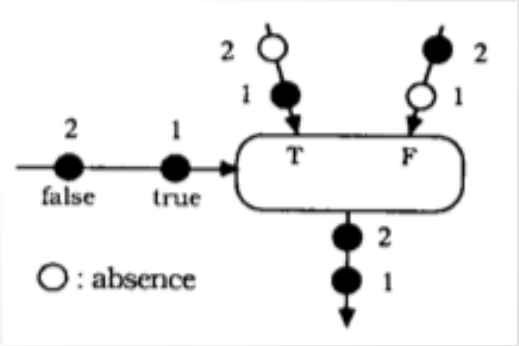
- *Multiple Clocked Recurrent Systems* modeling, which uses different time indices
- E.g. LUSTRE
- Easy to adopt when data flow is prevalent
- But difficult to model functioning mode changes

$$
\begin{aligned}
\boldsymbol{N} &= \boldsymbol{C} \cup \boldsymbol{R} \\
X_n &= \text{if } n \in \boldsymbol{R}, \text{ then } 0 \text{ else } \min\{2, X_{n-1} + 1\} \\
M_{R_k} &= \text{if } R_k \in \boldsymbol{C}, \text{ then } \min\{2, X_{R_k - 1} + 1\} \\
&\quad \text{else } X_{R_k - 1} \\
&\quad \text{if } R_k \in \boldsymbol{C}, \text{ then } X_{R_k - 1} \neq 0
\end{aligned}
$$

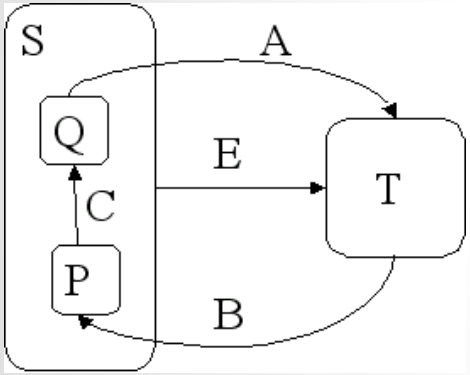
Synchronous Models to Asynchronous Systems

- We can use synchronous approach to validate asynchronous execution
- For example, for data-flow asynchronous execution:





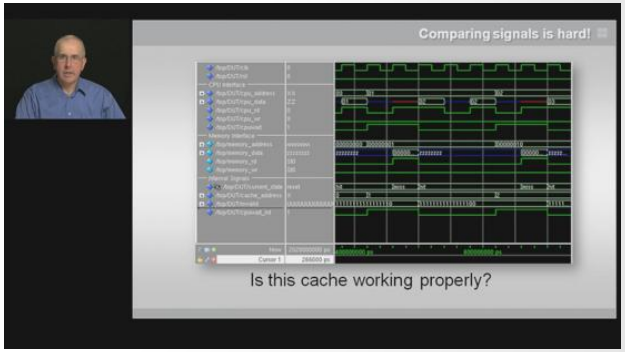
Conclusion



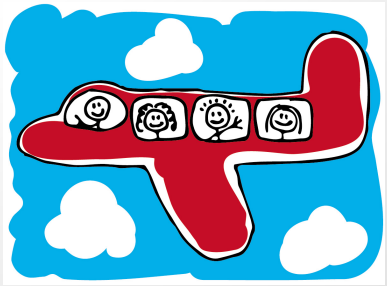
Synchronous programming approach

analyze specification

verify



Implement



Reactive / real-time systems

Implementing Synchronous Models on Loosely Time Triggered Architectures

Overview

- Designing with a synchronous model is simpler, easier to analyze & verify
- Ensuring synchrony in implementation is difficult
- Can we design synchronously but implement something that executes asynchronously?

LTTA improvements over TTA

- Time-triggered architecture (TTA) is decent but has some limitations in complex configurations and long delays
- Loose TTA (LTTA) is more flexible
- Paper discusses Finite FIFO Platforms (FFPs), which include even more flexibility

Synchronous system boundaries

- No self-loops without a unit delay (UD)
- Leads to set of equations to model functions
- Equations executed following any partial order sequence

Loosely Time-Triggered Architecture

- Each node runs one process
- Communicates via Communication by Sampling (CbS), i.e., one-way buffer
- Paper adds features to standard CbS to aid deduplication and message ordering
- Assume each process completes before being triggered again

Finite FIFO Platforms (FFPs)

- Directed, point to point, lossless FIFO queues between sequential processes
- Non-blocking R/W
- API implemented appropriately
- Same execution length assumption as LTTAs

Map Synchronous Models on FFP

- Queue size of 1-2, depending on unit delay
- Code mapping from synch model to FFP described
- Skipping introduced to handle overflow
- Deadlock guarantee given
- Existing proof re-use performed via relating the Synchronous FFP to a Marked Directed Graph (MDG) or Kahn Process Network (KPN)
- Queues then allowed to grow without check

Implementation of FFP on LTTA

- Using LTTA operations to complete each FFP API command is described
  - Each FFP command is implemented with a finite number of LTTA operations
  - LTTA operations are nonblocking

Throughput & Latency

- Worst-case analyzed
  - Processes trigger asynchronously or on top of one another, causing skipping
- Special topologies: chains, loops
- Synchronous models
  - Analysis with non-negligible delays

Closing

- Criticism regarding skip feature, data loss via overwrite
- Many extensions possible
  - Jitter
  - Multirate
  - Multicast
  - Average-case

Thank you!