Computational Models for Embedded Systems

Vescan Andreea, PHD, Assoc. Prof.

Faculty of Mathematics and Computer Science
Babeș-Bolyai University
Cluj-Napoca

2025-2026

Lecture 4a: Synchronous Reactive Models

Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

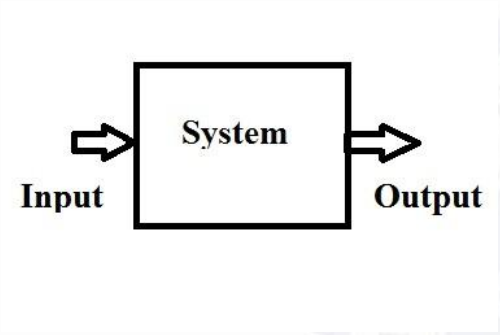
(Benjamin Franklin)

Outline

- Functional vs. Reactive Component
- Synchronous Reactive Models
  - Perfect Synchrony Hypothesis
  - The Synchronous Model of Time
  - Reactive embedded systems
- Synchronous Reactive Component
  - Definition
  - Executions
  - Properties of Components (Finite-state, CombinationaL, Event-triggered, Nondeterministic, Input-enabled, Task Graphs and Await Dependencies)
  - Composing Components (Block Diagrams, Input/Output Variable Renaming, Parallel Composition, Output Hiding)
- Synchronous Design - Cruise control system
  - Inputs and outputs
  - decomposing into subsystems
  - Tracking speed
  - Tracking cruise settings

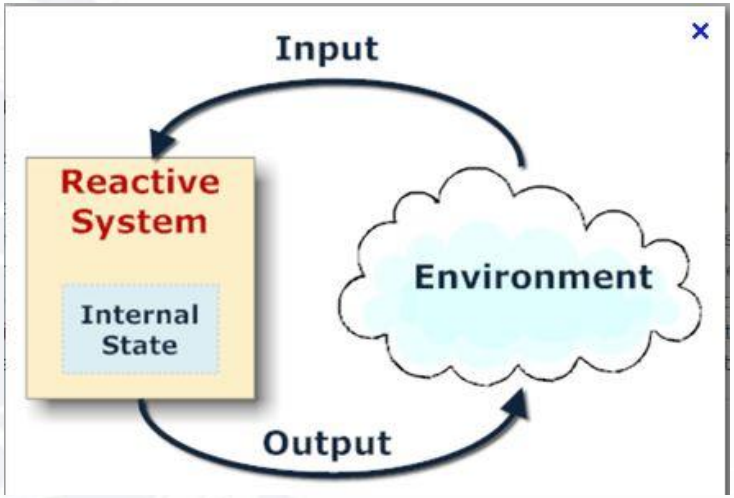
Functional vs. Reactive

Functional component



- produces outputs when supplied with inputs
- its behavior can be mathematically described using a mapping between input values and output values.

Reactive component



- interacts with other components via inputs and outputs in an ongoing manner

Synchronous Reactive Models

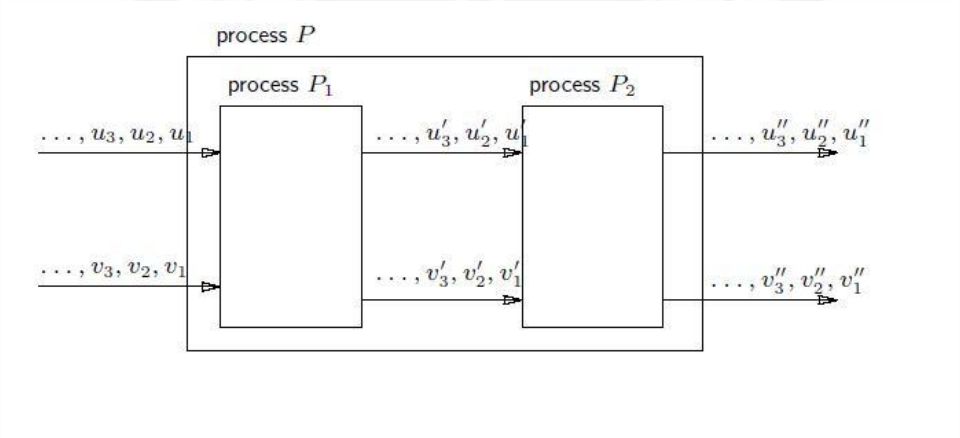
Synchronous model of reactive computation

- All components execute in a sequence of rounds.
- In each round, a reactive component reads its inputs, and based on its current state and inputs, it computes outputs, and updates the internal state.

Synchronous Reactive Models

Perfect Synchrony Hypothesis

- Neither computation nor communication takes time.

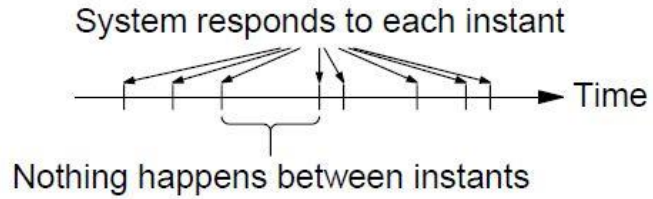


- The synchrony hypothesis:
  - Things compute instantaneously.
  - The external inputs do not change during a round, and when the inputs do change, a new round is initiated with all the tasks ready to process the new inputs.

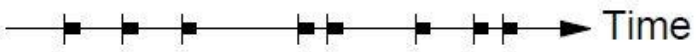
Synchronous Reactive Models

The Synchronous Model of Time

- Synchronous: time is an ordered sequence of instants.
- *Reactive: Instants initiated by environmental* events



- A system only needs to be “fast enough” to simulate synchronous behavior.

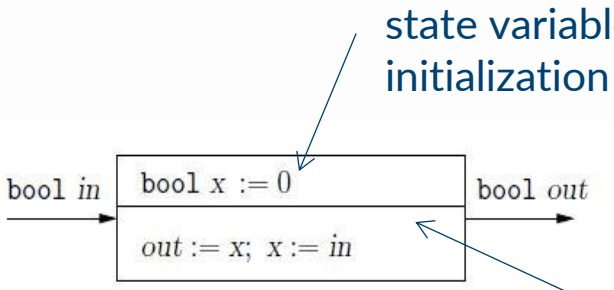


Synchronous Reactive Models

Reactive embedded systems

- Run at the speed of their environment
  - ***When*** *as important as* ***what***
  - *Concurrency for controlling the real world*
  - *Determinism desired*
  - *Limited resources (e.g., memory)*
  - *Discrete*
- Examples:
  - **Systems with user interfaces**
    - *Digital Watches*
    - *CD Players*
  - **Real-time controllers**
    - *Anti-lock braking systems*
    - *Industrial process controllers valued, time-varying*

Synchronous Reactive Component



Delay

Comp.

- I = {in} – set of typed input variables
- O = {out} - typed output variables
- S = {x} -typed state variables
- The set of all possible
  - Inputs Q_I={0,1}
  - Outputs Q_O
  - States Q_S

- To describe the dynamics of the component – we specify:
  - Initial states.
  - How reacts to a given input in each round.
- Reaction description - the computation of a component in response to an input in each round.
- If the component in state s, when supplied with input i, can produce output o and update its state to t, we write

$$s \xrightarrow{i/o} t$$

- Such a response is called a reaction.
- Delay component reactions:

$$0 \xrightarrow{0/0} 0;\quad 0 \xrightarrow{1/0} 1;\quad 1 \xrightarrow{0/1} 0;\quad 1 \xrightarrow{1/1} 1.$$

Synchronous Reactive Component

Definition

- *A synchronous reactive component C* is described by a finite set *I* of typed input variables defining the set Q_I of inputs, a finite set *O* of typed output variables defining the set Q_O of outputs, a finite set *S* of typed state variables defining the set Q_S of states, an initialization *Init* defining the set $[Init] \in Q_S$ of initial states, and a reaction description React defining the set [*React*] of reactions of the form $s \xrightarrow{i/o} t$

  , where *s*, *t* are states, *i* is an input and *o* is an output.

Synchronous Reactive Component

Executions

- An execution of a synchronous reactive component S consists of a finite sequence of the form:

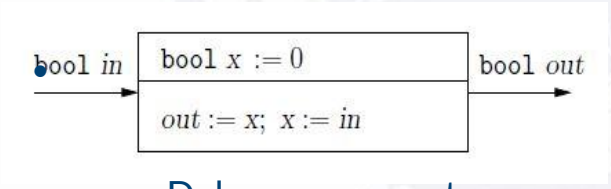
Where:

$$s_0 \xrightarrow{i_0/o_0} s_1 \xrightarrow{i_1/o_1} s_2 \ldots s_{k-1} \xrightarrow{i_{k-1}/o_{k-1}} s_k$$

a) For $0 \le j \le k$, each s_j is a state of C, and for $0 \le j < k$ each i_j is an input of C and each o_j is an output of C;

b) s_0 is an initial state of C;

c) For $0 \le j < k, s_j \xrightarrow{i_j/o_j} s_{j+1}$ is a reaction of C.



Delay component

$0 \xrightarrow{0/0} 0;\quad 0 \xrightarrow{1/0} 1;\quad 1 \xrightarrow{0/1} 0;\quad 1 \xrightarrow{1/1} 1.$

Delay reactions

One possible execution:

$$0 \xrightarrow{1/0} 1 \xrightarrow{1/1} 1 \xrightarrow{0/1} 0 \xrightarrow{1/0} 1 \xrightarrow{1/1} 1 \xrightarrow{1/1} 1.$$

Synchronous Reactive Component

Properties of Components

- Finite-state Components
- Combinational Components
- Event-triggered Components
- Nondeterministic Components
- Input-enabled Components
- Task Graphs and Await Dependencies

Synchronous Reactive Component

Properties of Components

Finite-state Components

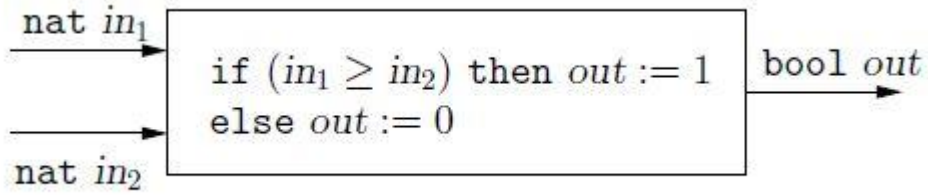
- In many embedded applications, it suffices to consider types with only finitely many values.
- Such components are called **finite-state components**, and are amenable to powerful automated analysis.

- A synchronous reactive component C is said to be *finite-state* if the type of each of its input, output, and state variables are finite.

Synchronous Reactive Component

Properties of Components

Combinational Components



Comparator component

-has no state variables

- When the set S of state variables is empty, formally there is a unique valuation for S, $s_\emptyset$.
- For every pair of natural numbers m, n, if m≥n, Comparator has a reaction $S_\Phi \xrightarrow{(m,n)/1} S_\Phi$ and if m<n it has a reaction $S_\Phi \xrightarrow{(m,n)/0} S_\Phi$
- A possible execution of the component is:

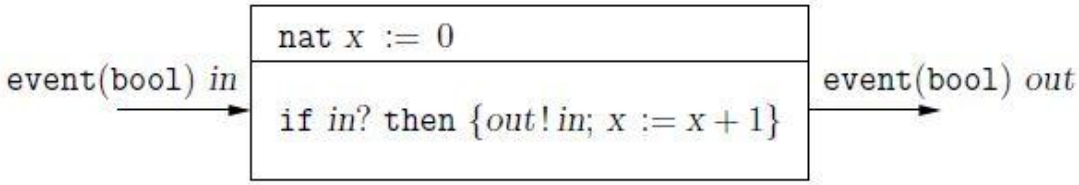
$$S_\Phi \xrightarrow{(3,7)/0} S_\Phi \xrightarrow{(9,2)/1} S_\Phi \xrightarrow{(20,20)/1} S_\Phi$$

- A synchronous reactive component C is said to be *combinational* if the set of its state variables is empty.

Synchronous Reactive Component

Properties of Components

Event-triggered Components



TriggeredCopy component

- type event - enumerated type
  - T - denotes that the event is present; (x?, $\{T, \perp\}$) $x \neq \perp$
  - $\perp$ - denotes that the event is absent.
- the event type to be parameterized by another type: event (bool) has three values: 1,0, $\perp$.
- *out:=in* is abbreviated by *out!in* – the event out is issued.
- For every natural number n, the component has 3 reactions:

$$n \xrightarrow{\perp/\perp} n; n \xrightarrow{0/0} n+1; n \xrightarrow{1/1} n+1.$$

- Sample execution:

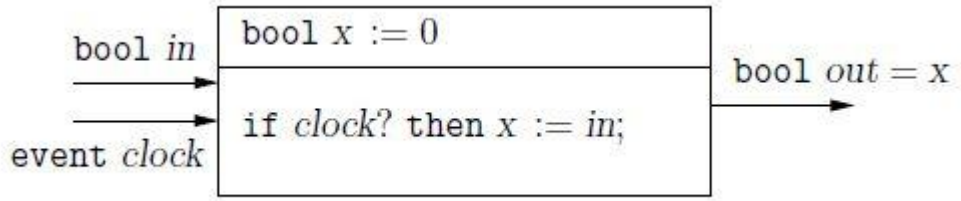
$$0 \xrightarrow{\perp/\perp} 0 \xrightarrow{0/0} 1 \xrightarrow{1/1} 2 \xrightarrow{\perp/\perp} 2 \xrightarrow{1/1} 3.$$

- Stuttering reaction - the output is absent and state stays unchanged.

Synchronous Reactive Component

Properties of Components

Event-triggered Components (cont).



ClockedCopy component

- An output variable is said to be ***latched*** if there exist a state variable x such that in every reaction $s \xrightarrow{i/o} t$ of the component the value of the output variable y is the updated value of the state variable x: o(y)=t(x).
- Sample execution:

$$0 \xrightarrow{(1,\perp)/0} 0 \xrightarrow{(1,T)/1} 1 \xrightarrow{(0,\perp)/1} 1 \xrightarrow{(0,\perp)/1} 1 \xrightarrow{(0,T)/0} 0.$$

Synchronous Reactive Component

Properties of Components

Event-triggered Components (cont.)

- For a synchronous reactive component C=(I,O,S,Init,React), a set $J \subseteq I$ of input variables is said to be a **trigger** if:

a) Every input variable in J is of type event;

b) Every output variable either is latched, or is of type event, and

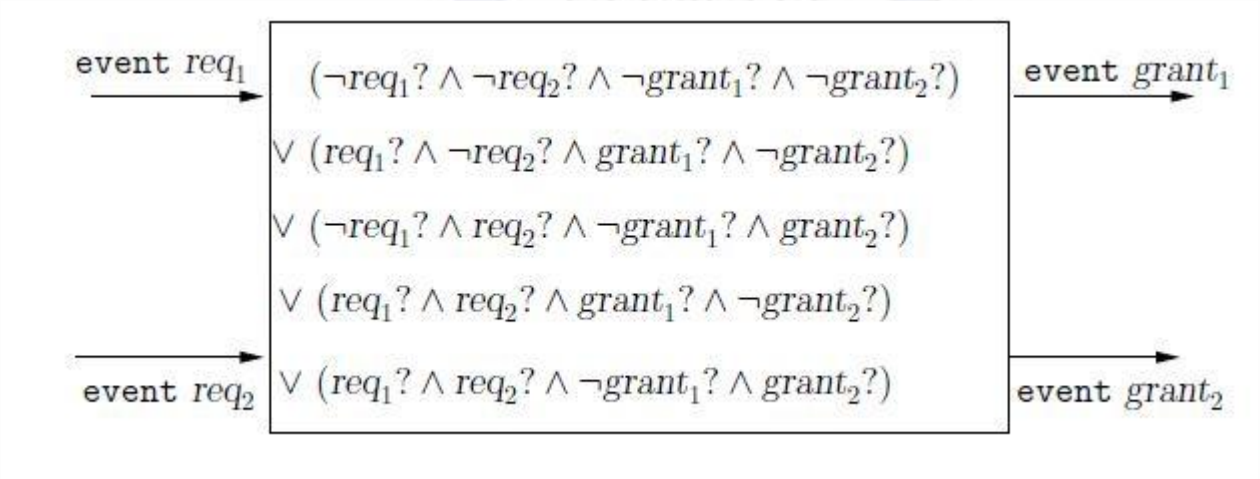
c) If i is an input with all events in J absent (that is, for all input variables $x \in J, i(x) = \perp$), then for all states s, if $s \xrightarrow{i/o} t$ is a reaction then s=t and $o(y) = \perp$ for all event output variables y.

A component C is said to be **event-triggered** if there exists a subset $J \subseteq I$ of its input variables such that J is a trigger for C.

Synchronous Reactive Component

Properties of Components

Nondeterministic Components



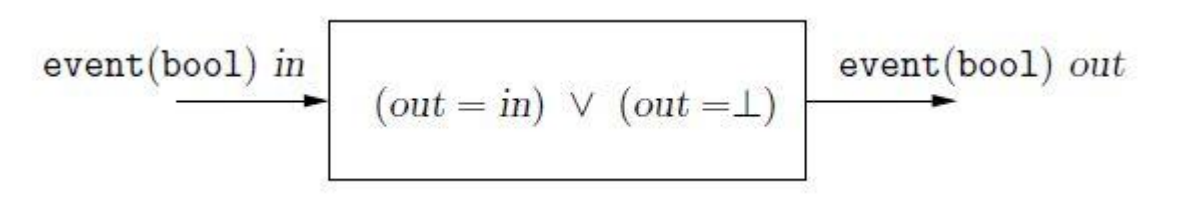
Arbiter component

- resolve contention among incoming requests
- when only one of the requests is present, the component issues the corresponding grant output
- if both requests are absent, then both the outputs are absent
- if both requests are present, then there are two possible computations of the component

Synchronous Reactive Component

Properties of Components

Nondeterministic Components (cont.)



LossyCopy component

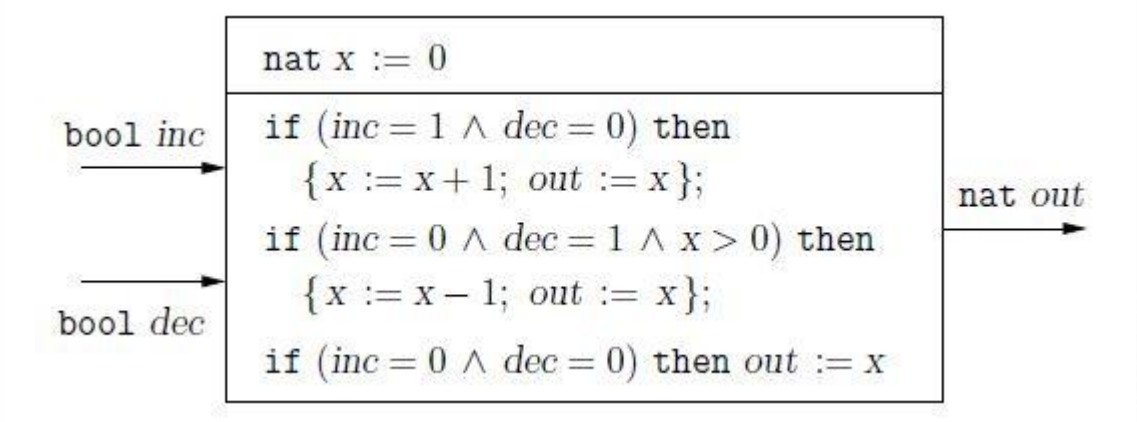
- Sample execution

$$s_\phi \xrightarrow{0/0} s_\phi \xrightarrow{1/\perp} s_\phi \xrightarrow{\perp/\perp} s_\phi \xrightarrow{1/1} s_\phi \xrightarrow{\perp/\perp} s_\phi \xrightarrow{0/\perp} s_\phi .$$

Synchronous Reactive Component

Properties of Components

Input-enabled Components



Counter component

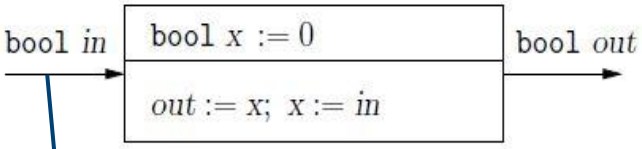
- The counter does not expect both input variable inc and dec to be 1 simultaneously, nor does expect the counter to be decremented when the counter value is 0.
- When both input variables inc and dec are 1, and when dec is 1 with the state x equal to 0, the reaction description does not assign any value to the output, and thus, there is no corresponding reaction.

- For a synchronous reactive component *C*, an input *i* is said to be enabled in a state *s* if there exists an output *o* and a state *t* such that $s \xrightarrow{i/o} t$ is a reaction of C. The component C is said to be input-enabled if every input is enabled in every state.

Synchronous Reactive Component

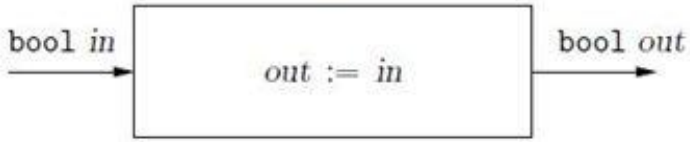
Properties of Components

Task Graphs and Await Dependencies



Delay component

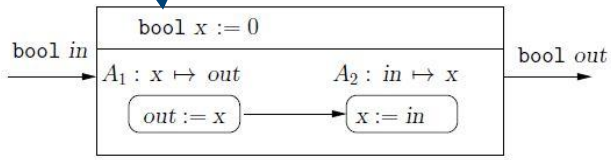
- the output does not depend on its input in that round



Relay component

- the output must await its input

The description hides the intra-round independence of the output on the input.



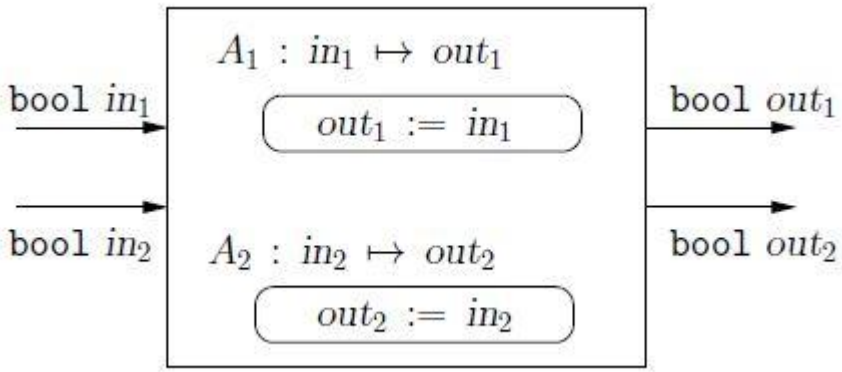
SplitDelay component

- to specify constraints on the order in which the tasks should be executed

Synchronous Reactive Component

Properties of Components

Task Graphs and Await Dependencies (cont.)



ParallelRelay component

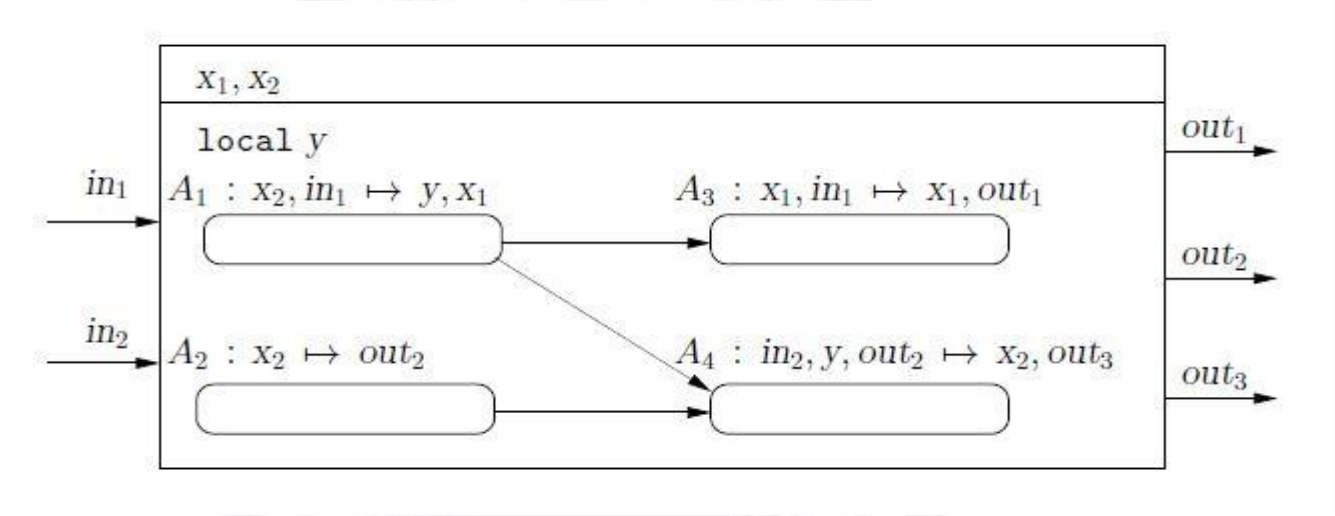
- out_1 awaits in_1
- out_2 awaits in_2

- **task schedule** - a linear ordering of all the tasks that is consistent with the precedence relation
- SplitDelay: A1; A2
- ParallelRelay: A1;A2 and A2;A1

Synchronous Reactive Component

Properties of Components

Task Graphs and Await Dependencies (cont.)



How many possible schedules?

- Ax, Ay, Az, At?

Illustrative task graph

- there are five possible schedules:
  - A1;A2;A3;A4;
  - A1;A2;A4;A3;
  - A1;A3;A2;A4;
  - A2;A1;A3;A4;
  - A2;A1;A4;A3:

Synchronous Reactive Component

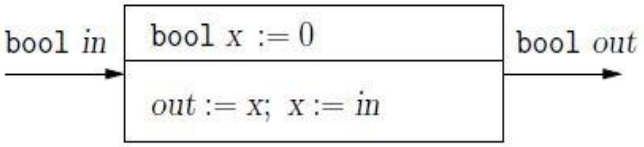
Composing Components

- Block Diagrams
- Input/Output Variable Renaming
- Parallel Composition
- Output Hiding

Synchronous Reactive Component

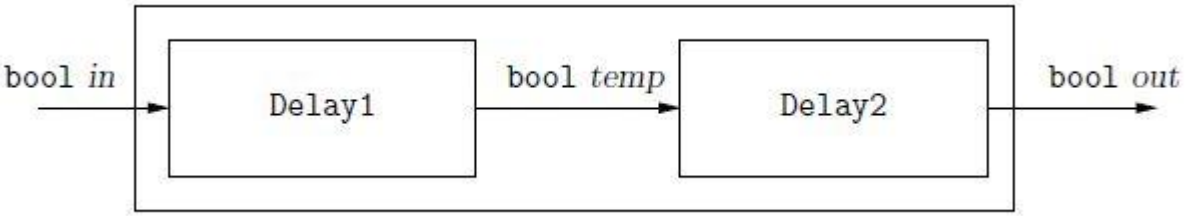
Composing Components

Block Diagrams



Delay component

- Block diagram for DoubleDelay from two Delay components



In the same round

- The component DoubleDelay is textually defined as
  (Delay[out ⟼ temp] || Delay[in ⟼ temp]) \temp
- Operations
  - parallel composition ||
  - renaming ⟼
  - hiding \

Synchronous Reactive Component

Composing Components

Input/Output Variable Renaming

- Let C = (I,O, S, Init,React) be a synchronous reactive component, x be an input or an output variable, and y be a fresh variable (that is, y is not a state, input, or output variable of C) such that the types of x and y are the same.

- Then the component obtained by renaming x to y in C, denoted C [x $\longmapsto$ y], is the synchronous reactive component obtained by substituting the variable name x by y in the description of C.

Synchronous Reactive Component

Composing Components

Parallel Composition

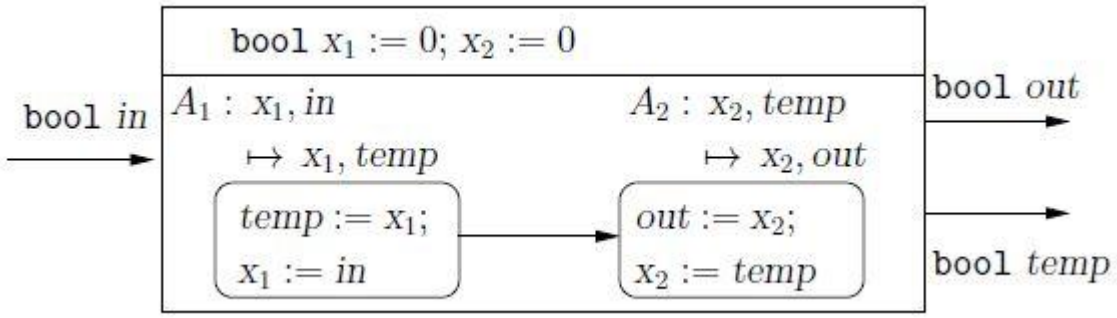
- The **parallel composition operation** combines two components into a single component whose behavior captures the synchronous interaction between the two components running concurrently.
- **Compatibility in variable names**
  - First, there should be no name conflicts concerning state variables.
    - That is, the set S1 should be disjoint from each of the sets I2, O2, and S2, and symmetrically, S2 should be disjoint from each of I1, O1, and S1.
  - Second, a variable can be an input variable to both the components, and an output variable of one component can be an input variable to the other, but a variable cannot be an output variable of both the components.
    - That is, the sets O1 and O2 should be disjoint.

Synchronous Reactive Component

Composing Components

Parallel Composition (cont)

- Product variables
- Product states
- Reaction description of the product



Parallel composition of Delay1 and Delay2

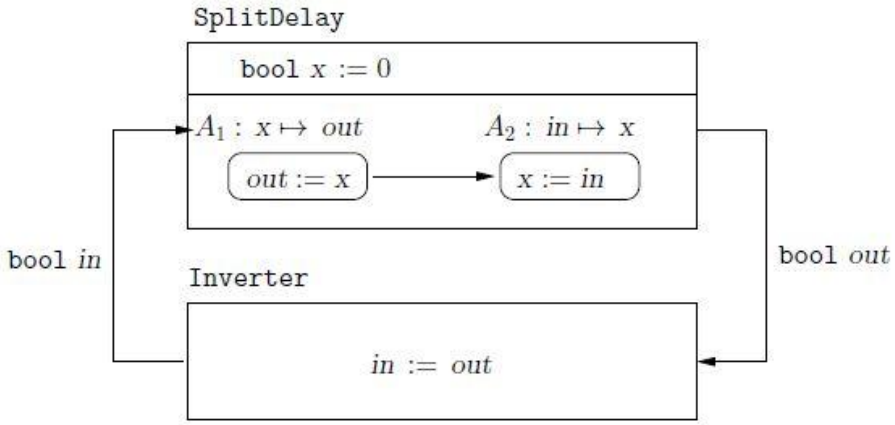
Synchronous Reactive Component

Composing Components

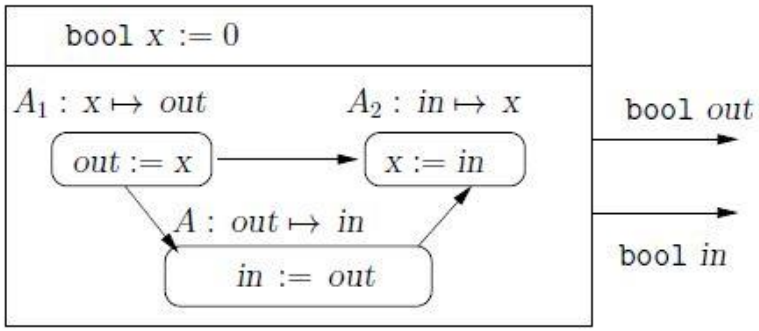
Parallel Composition (cont)

Composing task graphs

Feedback composition of a SplitDelay with an Inverter



Parallel composition of SplitDelay and Inverter



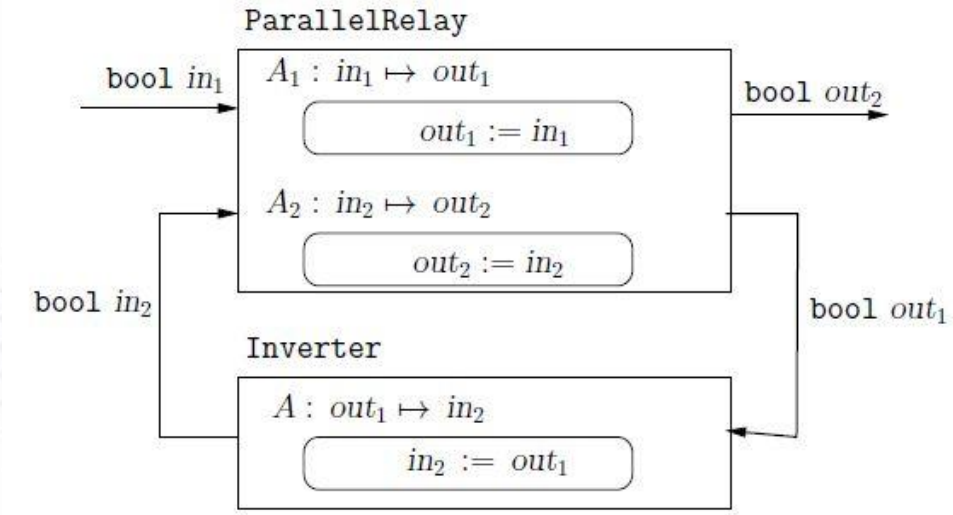
- In the first round, out is 0 and in is 1,
- - in every subsequent round, both these values toggle.
- sequence of outputs produced by the product (in ,out)

10; 01; 10; 01; 10

Synchronous Reactive Component

Composing Components

Parallel Composition (cont)



- Parallel composition of ParallelRelay and Inverter
  - The two tasks A1 and A2 are independent in ParallelRelay.
  - Since the component Inverter reads out1 and writes in2, we get cross-component edges from A1 to A and from A to A2.
  - ➔a new transitive precedence constraint
    ➔ the task A1 must be executed before A2 in the product.

Synchronous Reactive Component

Composing Components

Parallel Composition (cont)

- Component Compatibility

Component Compatibility

The components C_1 with input variables I_1, output variables O_1, and input/output await-dependency relation $\succ_1 \subseteq O_1 \times I_1$, and C_2 with input variables I_2, output variables O_2, and input/output await-dependency relation $\succ_2 \subseteq O_2 \times I_2$, are said to be *compatible* if (1) O_1 and O_2 are disjoint, and (2) the relation $(\succ_1 \cup \succ_2)$ is acyclic.

Synchronous Reactive Component

Composing Components

Parallel Composition (cont)

COMPONENT COMPOSITION

Let $C_1 = (I_1, O_1, S_1, Init_1, React_1)$ and $C_2 = (I_2, O_2, S_2, Init_2, React_2)$ be compatible synchronous reactive components. Suppose the reaction description $React_1$ is given using local variables L_1 by a task graph with the set $\mathcal{A}_1$ of tasks and the precedence relation $\prec_1$, and the reaction description $React_2$ is given using local variables L_2 by a task graph with the set $\mathcal{A}_2$ of tasks and the precedence relation $\prec_2$. Then the *parallel composition* $C_1 \| C_2$ is a synchronous reactive component C such that

- the set S of state variables is $S_1 \cup S_2$;
- the set O of output variables is $O_1 \cup O_2$;
- the set I of input variables is $(I_1 \cup I_2) \setminus O$;
- the initialization for a state variable x is given by $Init_1$ for $x \in S_1$ and by $Init_2$ for $x \in S_2$;
- the reaction description of C uses the local variables $L_1 \cup L_2$, and is given by the task graph whose set of tasks is $\mathcal{A}_1 \cup \mathcal{A}_2$ and the precedence relation is the union of $\prec_1$ and $\prec_2$ and task pairs (A_1, A_2) such that A_1 and A_2 are tasks of different components with some variable occurring in both the write-set of A_1 and read-set of A_2.

Synchronous Reactive Component

Composing Components

Output Hiding

- The final operation needed to define the semantics of block diagrams is hiding of output variables. If y is an output variable of a component C, the result of hiding y in C, denoted C \ y, gives a component that behaves exactly like the component C, but y is no longer an output that is observable outside. This is achieved by removing y from the set of output variables, and declaring y to be a local variable in the reaction description.

Synchronous Design

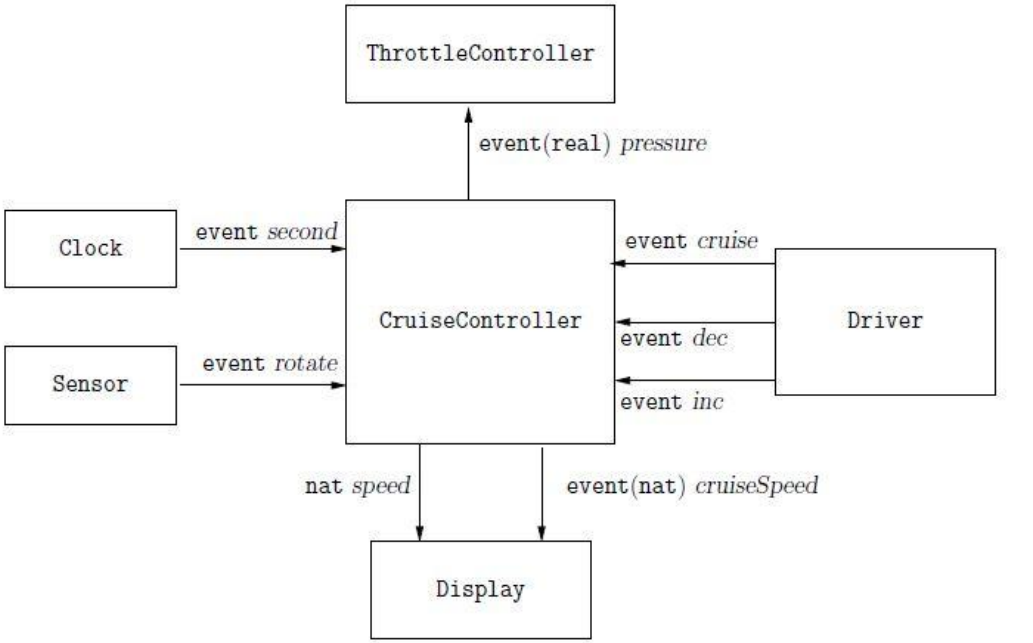
Cruise control system

- Inputs and outputs of the cruise control system
- Decomposing into subsystems
- Tracking speed
- Tracking cruise settings

Synchronous Design

Cruise control system

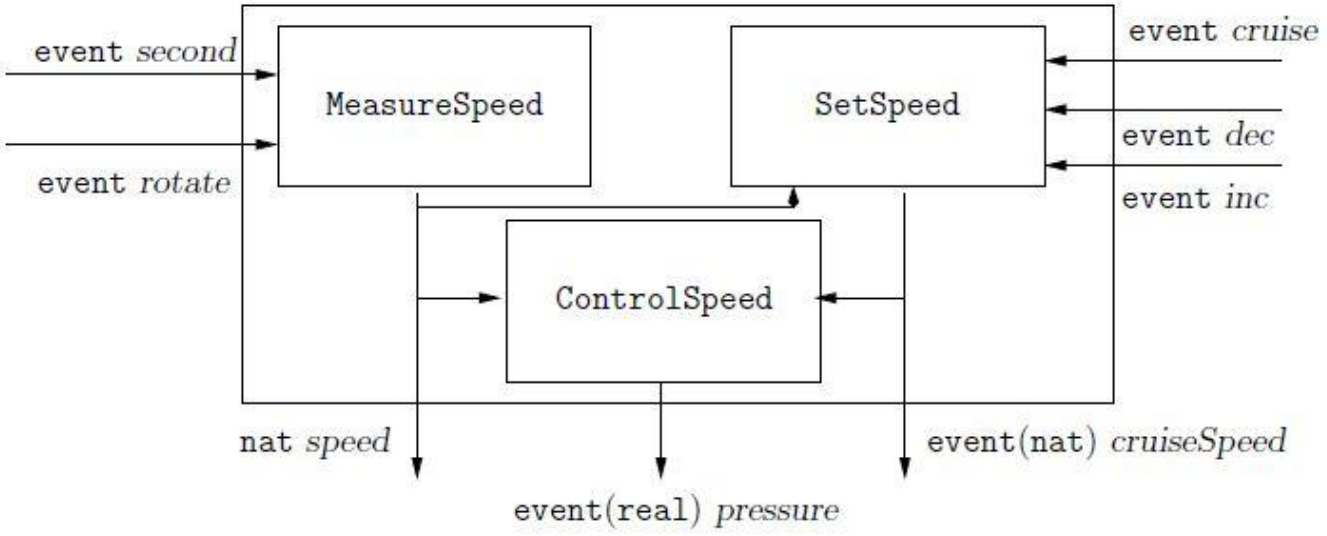
Inputs and outputs of the cruise control system



Synchronous Design

Cruise control system

Decomposing into subsystems

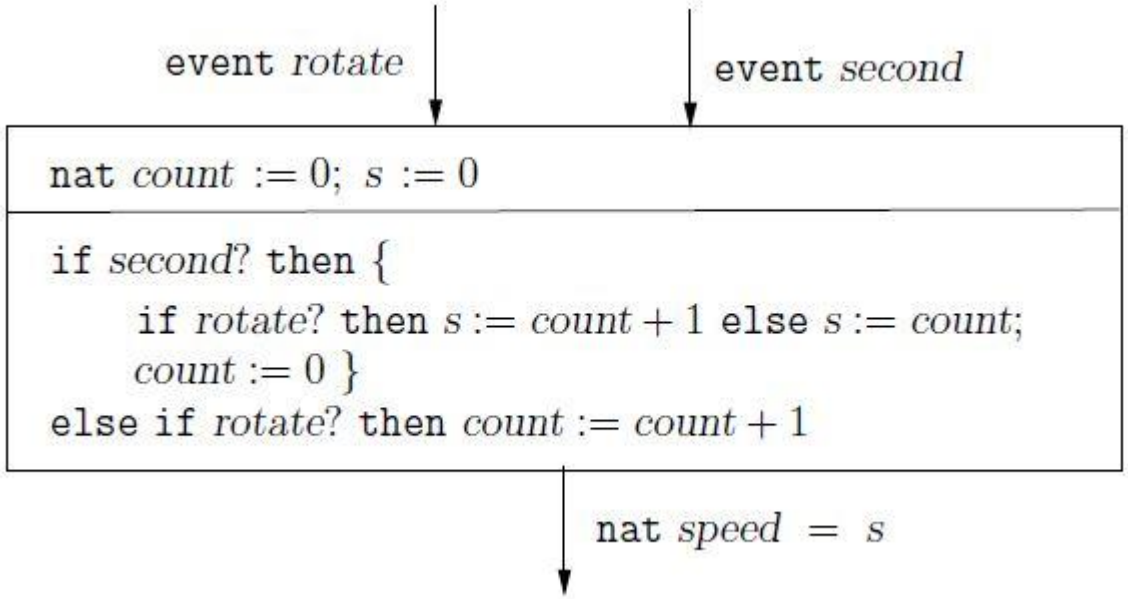


CruiseController

Synchronous Design

Cruise control system

Tracking speed

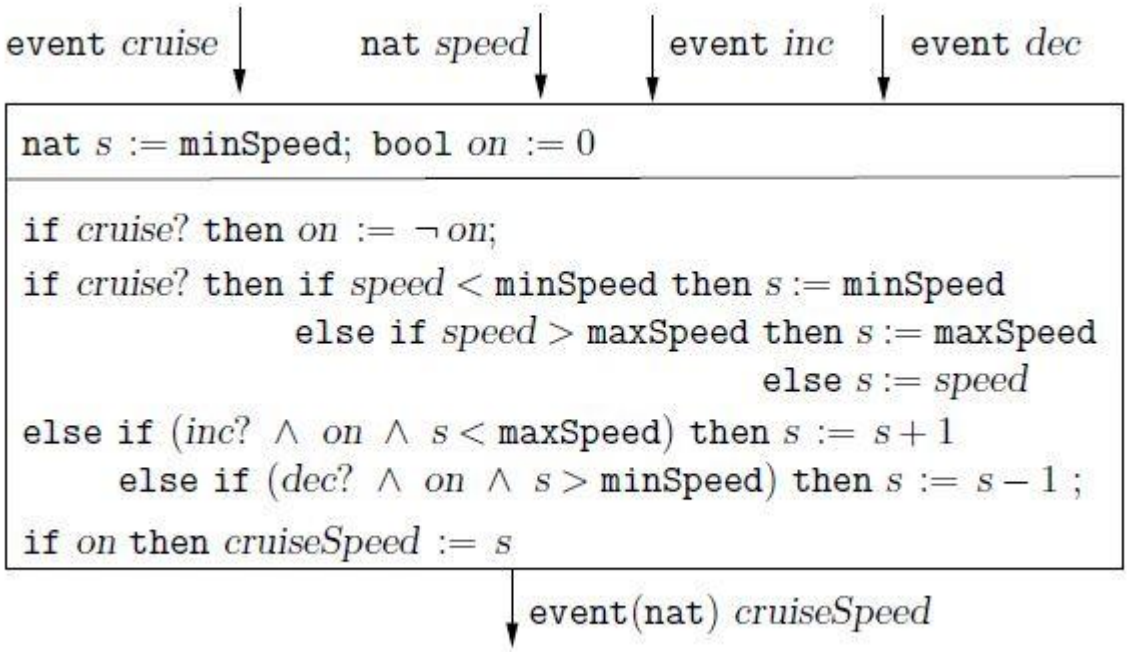


MeasureSpeed

Synchronous Design

Cruise control system

Tracking cruise settings



Who Uses This Stuff?

- Virtually all digital logic designed this way

- In software,
  - ***Dassault*** (French aircraft manufacturer) builds avionics with synchronous software.
  - ***Polis*** (Berkeley HW/SW codesign project) uses Esterel for specifying EFSMs.
  - ***Cadence built product*** (Cierto VCC) based on Polis.
  - TI (Texas Instruments) exploring using synchronous software for specifying/simulating DSPs (Digital signal processors).

Suitable Applications for the Synchronous Model

- Reactive systems receive inputs, react to them by computing outputs and wait for the next inputs to arrive.
- Embedded control systems connected to sensors and actuators.
- Wireless communication devices receiving samples with a fixed, predefined frequency.
- Telecom switches reading all input data packets before re-emitting al packets.
- Synchronous hardware.

References

- [1] Principles of Embedded Computation, Rajeev Alur http://www.seas.upenn.edu/~cis540/
- [2] Design of Embedded Systems: Models, Validation and Synthesis, Alberto Sangiovanni-Vincentelli
- https://inst.eecs.berkeley.edu/~ee249/fa07/

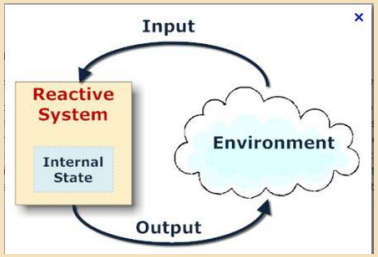
- [3] Rajeev Alur, Principles of Cyber-Physical Systems, MIT Press, 2015
- (Biblioteca de Matematica si Informatica, UBB,
  - Rajeev Alur, Principles of Cyber-Physical Systems, 2015
    COTA:9491
  - https://www.bcucluj.ro/ro/despre-noi/filiala/biblioteca-de-matematic%C4%83-%C5%9Fi-informatic%C4%83)

CMES – Today

Bring it All Together

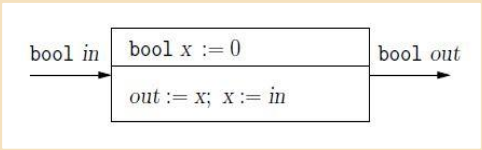
Synchronous Model

Reactive Component



Synchronous Reactive Component

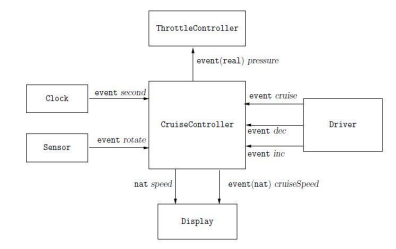
- Delay component



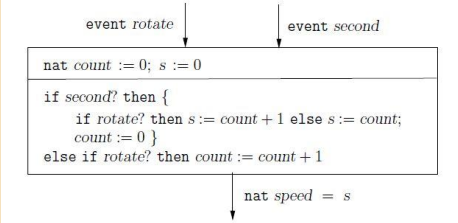
Properties of Components

- Finite-state Components
- Combinational Components
- Event-triggered Components
- Nondeterministic Components
- Input-enabled Components
- Task Graphs and Await Dependencies

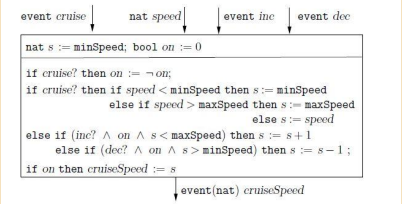
Cruise control system



Tracking speed

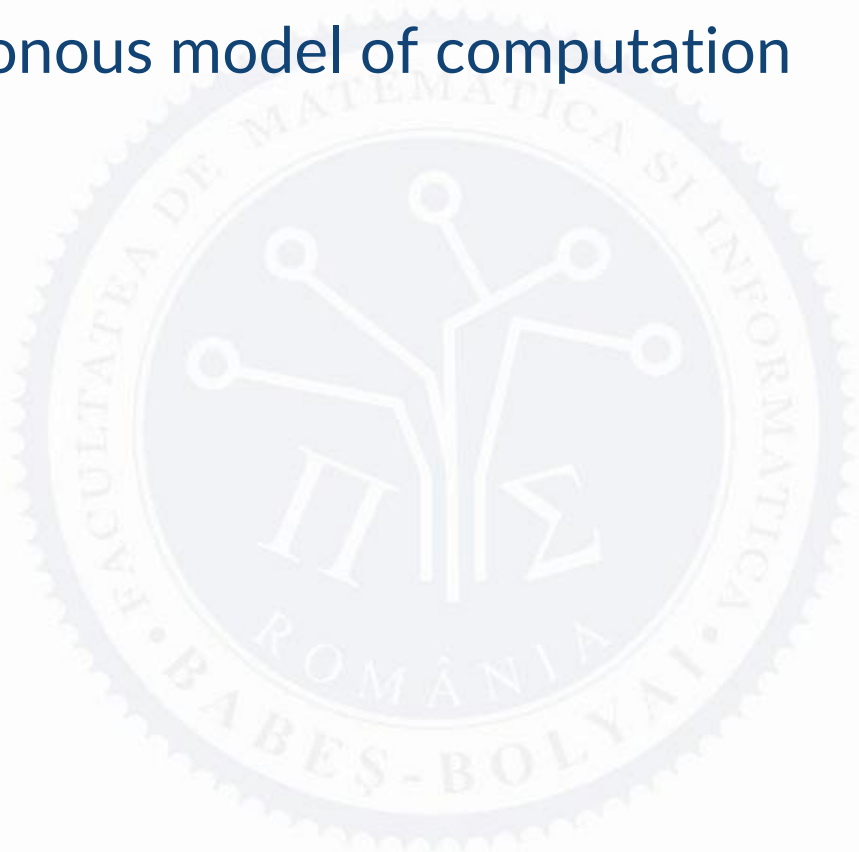


Tracking cruise settings



Next Lecture

- Asynchronous model of computation



Software Systems Verification and Validation

"Tell me and I forget, teach me and I may remember, involve me and I learn."

(Benjamin Franklin)