Introduction to Discrete Event Simulation

John Colley

12th December 2011

Southampton

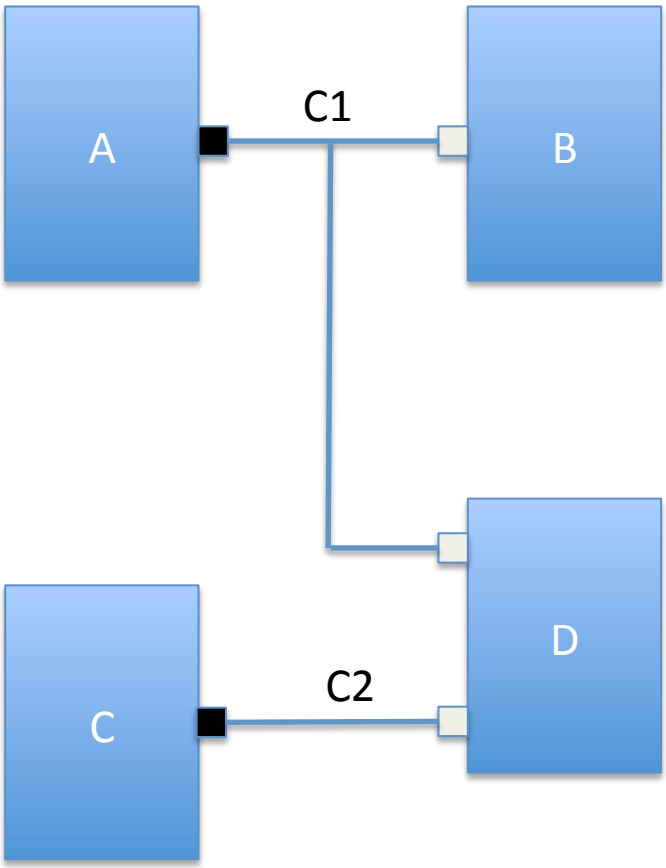
Introduction

- Motivation
- Discrete Event Simulation Principles
- Single-thread/Multi-thread Implementations
- Continuous Simulation Principles
- Hybrid Continuous/Discrete Simulation
- Summary

Discrete Event Simulation: Motivation

- Dominant Verification Technology for Synchronous Digital Hardware and Systems on Chip
  - $300 million per year Market
  - Mature Tools (30 years of development)
  - High Price Tools ($30k per seat + maintenance)
- Single Kernel, Multi-Language Tools for Model *Re-use*
  - VHDL
  - SystemVerilog
  - SystemC (C++)
  - C

Discrete Event Simulation



COMPONENT VIEW

Components: A, B, C, D (processes)

Connections: C1, C2 (unidirectional)

Ports: IN □ OUT ■

SIMULATOR API

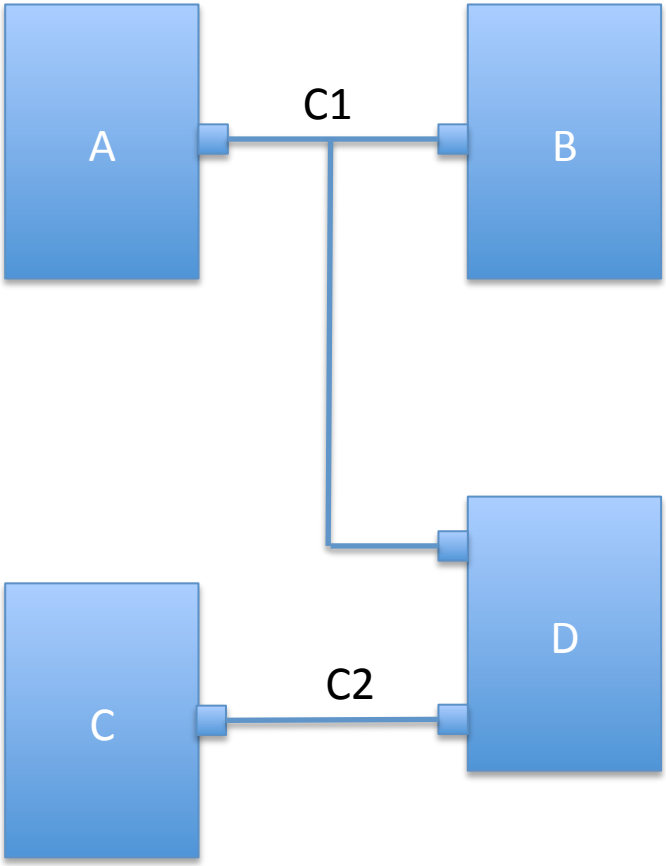
GetValue(port)

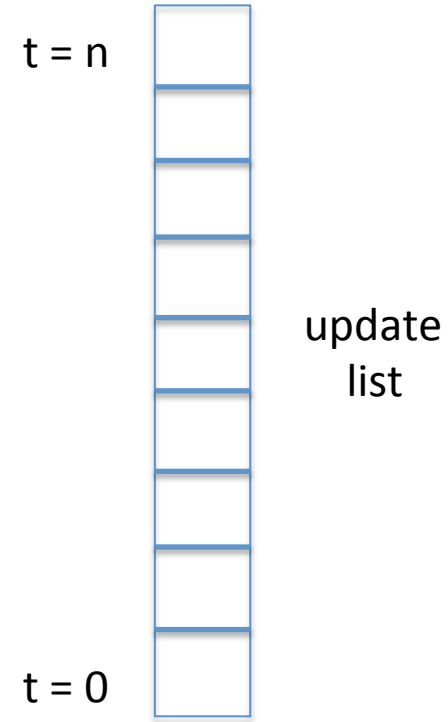
HasChanged(port)

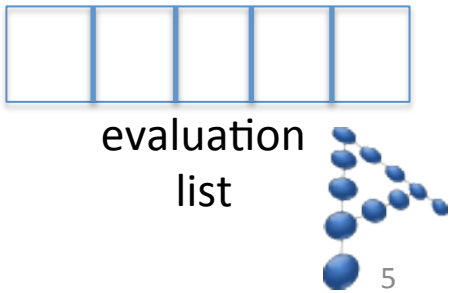
SetValue(OUT port, val, delay)

ScheduleEval(component, delay)

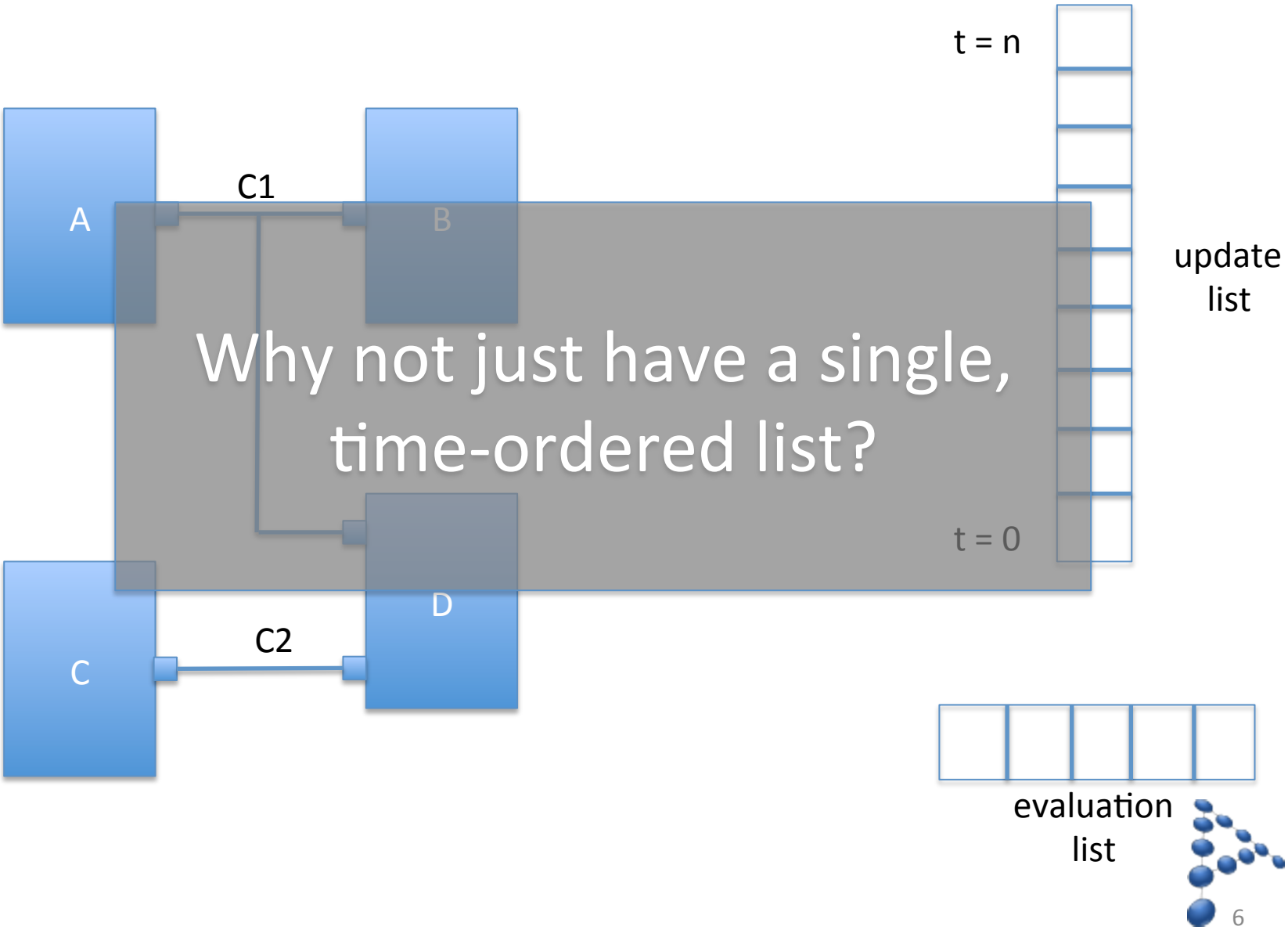
The Two-list Simulation Algorithm



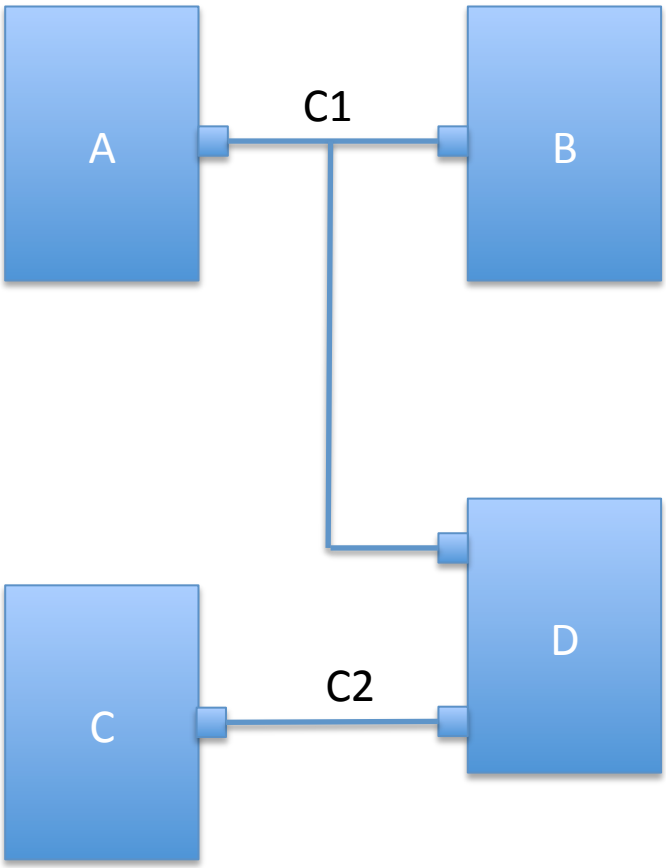


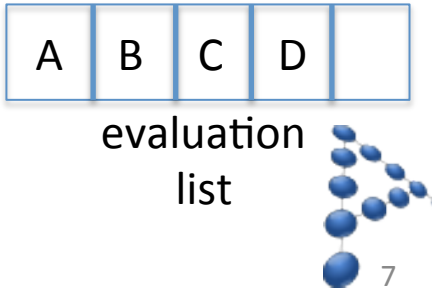


The Two-list Simulation Algorithm

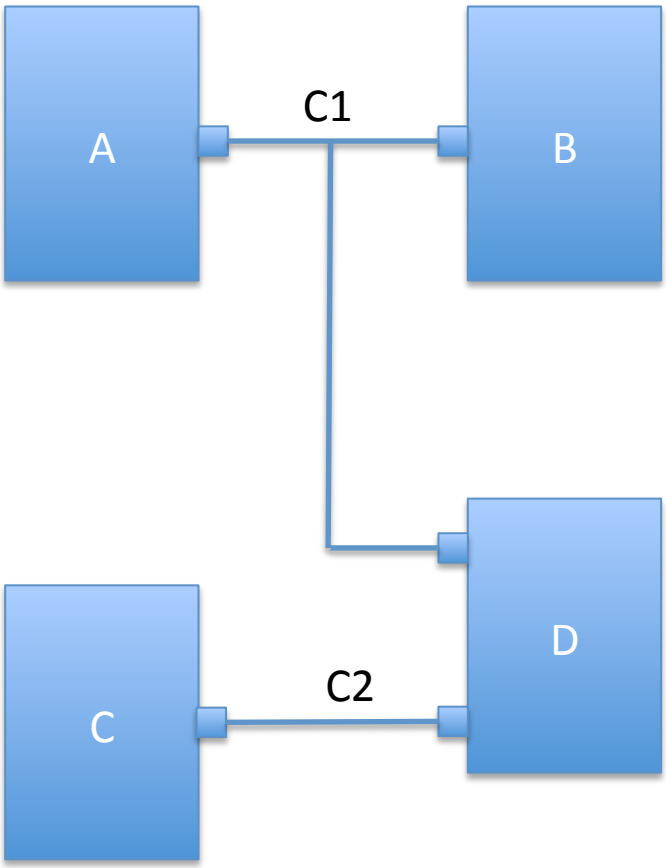


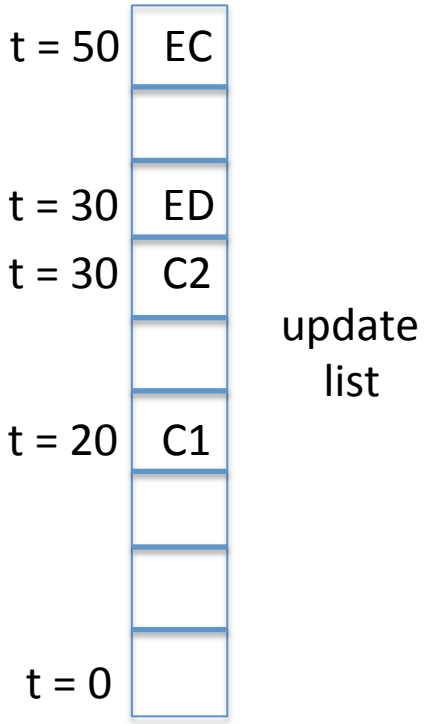
Time Zero Initialisation: Evaluate all Components

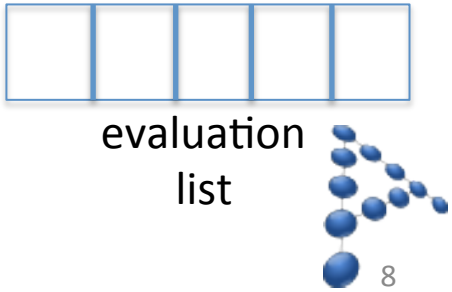




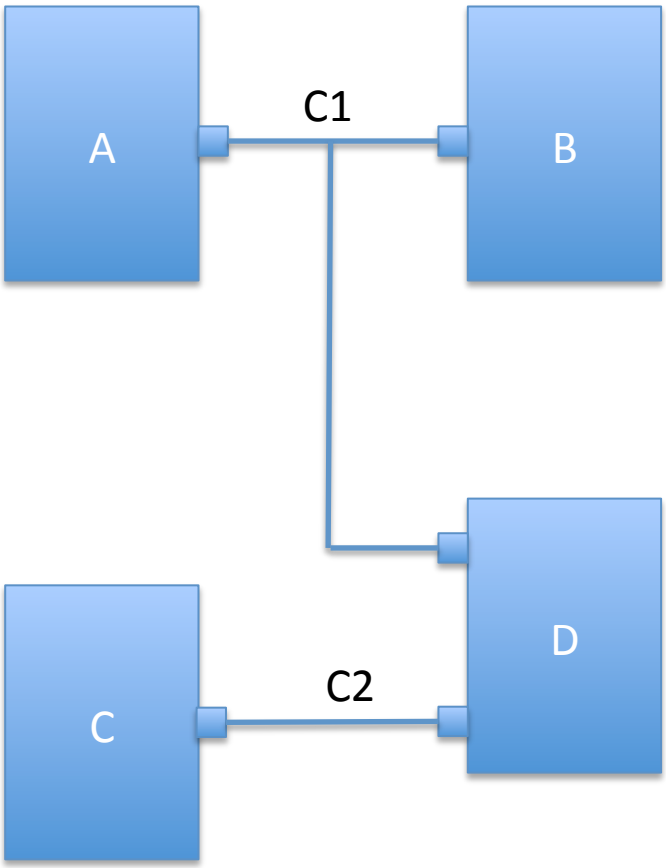
Component evaluations call *SetValue*, *ScheduleEval*

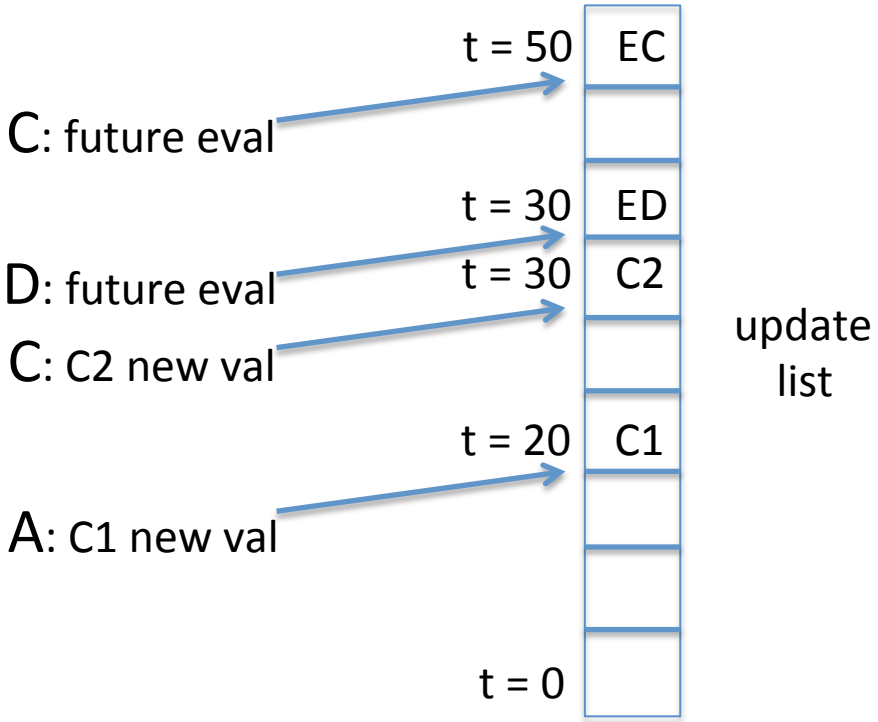


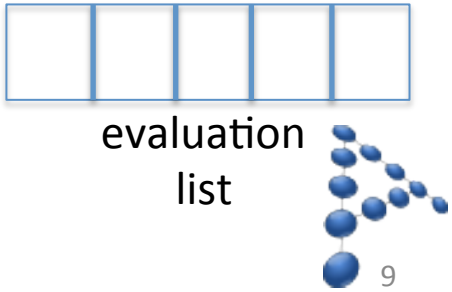




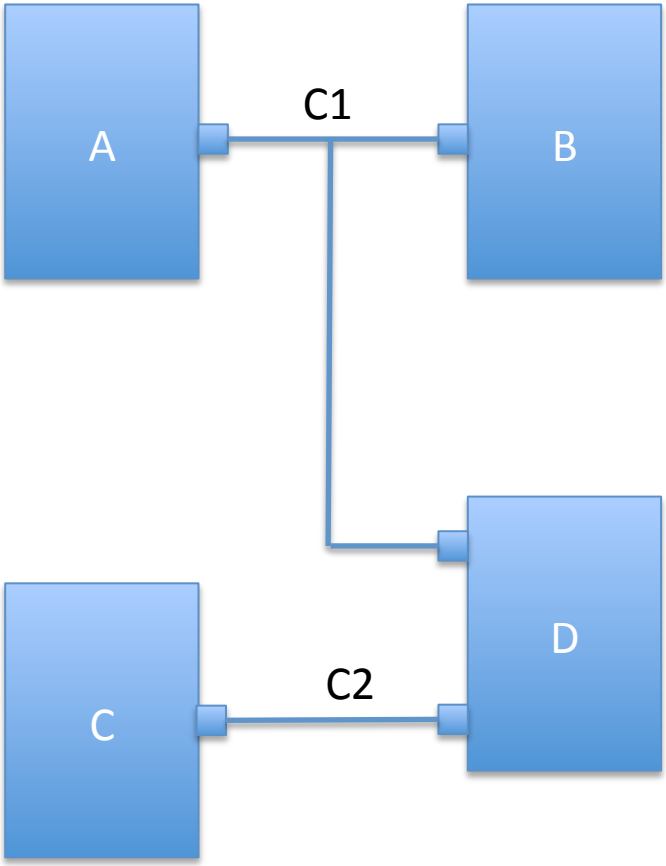
Component evaluations call *SetValue*, *ScheduleEval*

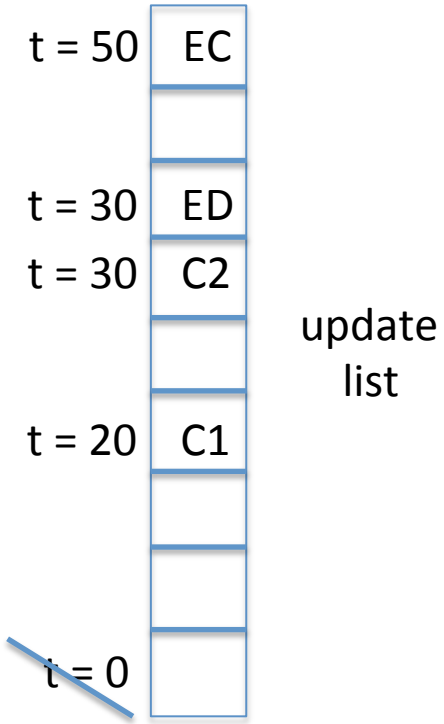


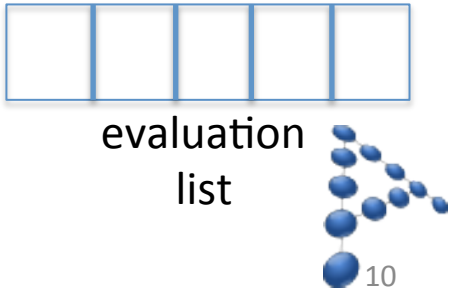




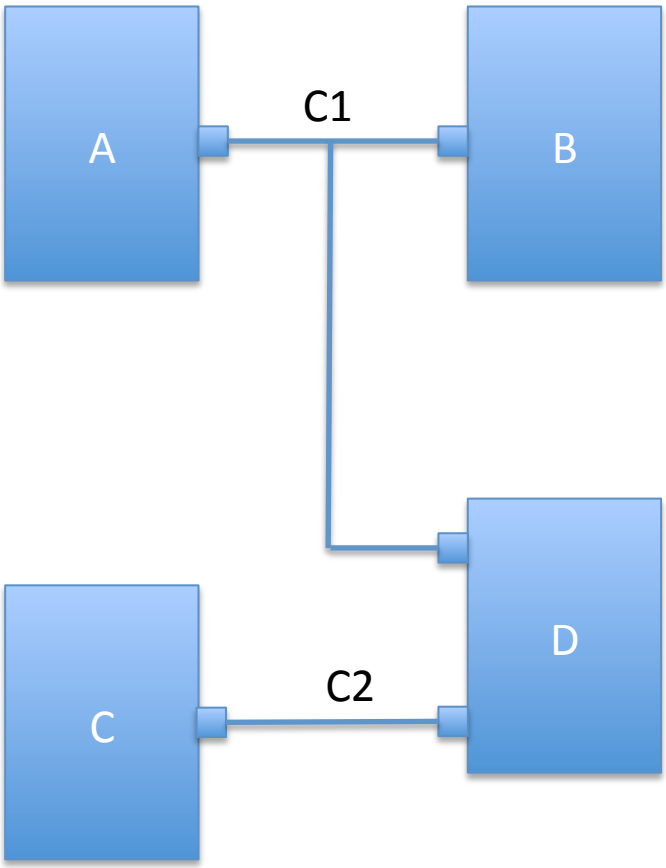
Global Time is Advanced to t = 20

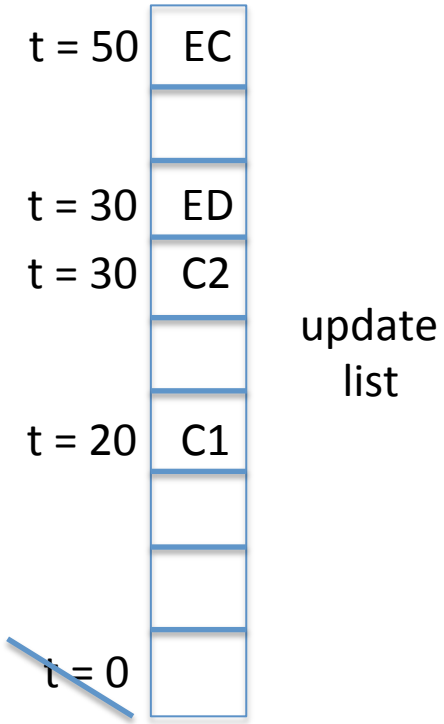


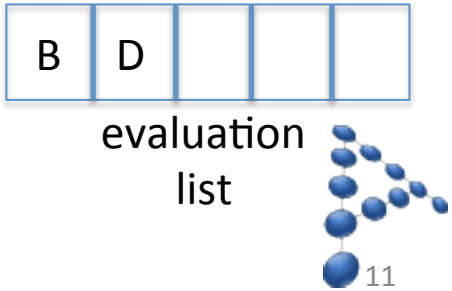




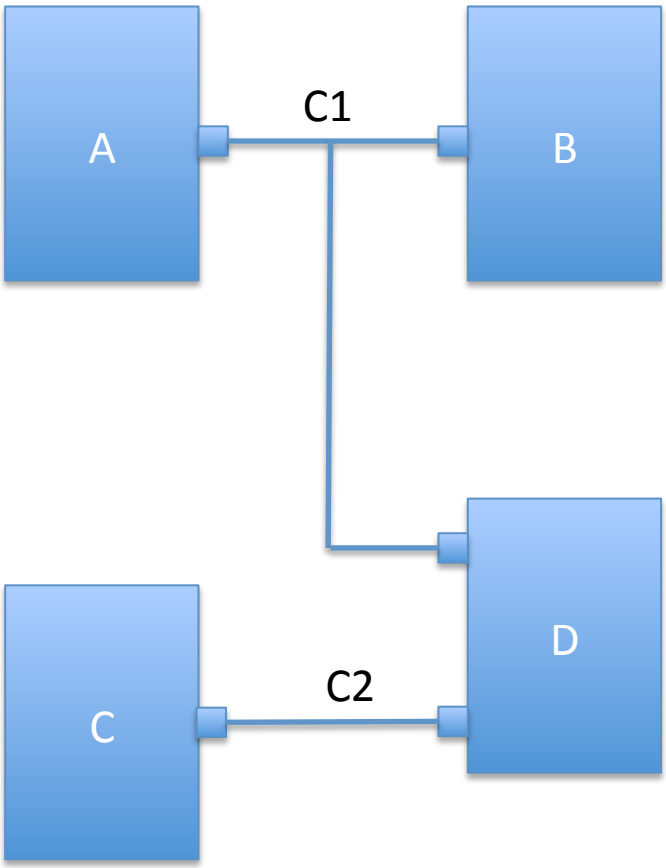
Add to *eval list* each component on C1 *fanout*

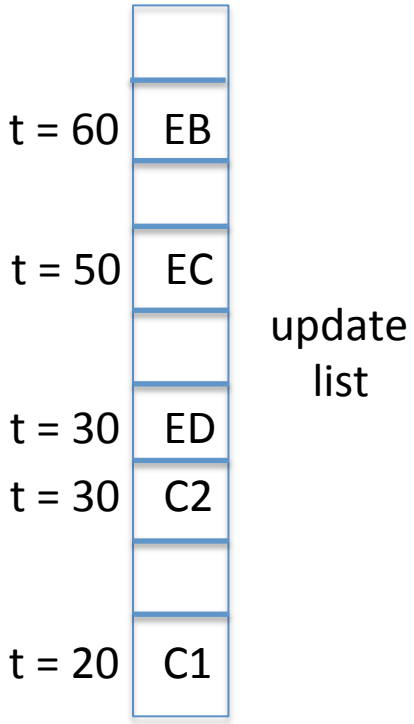


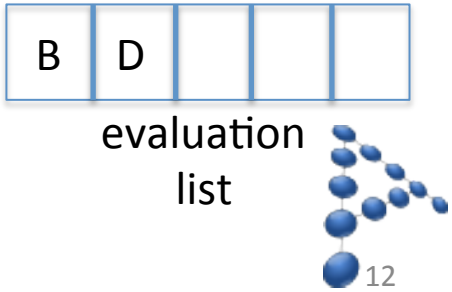




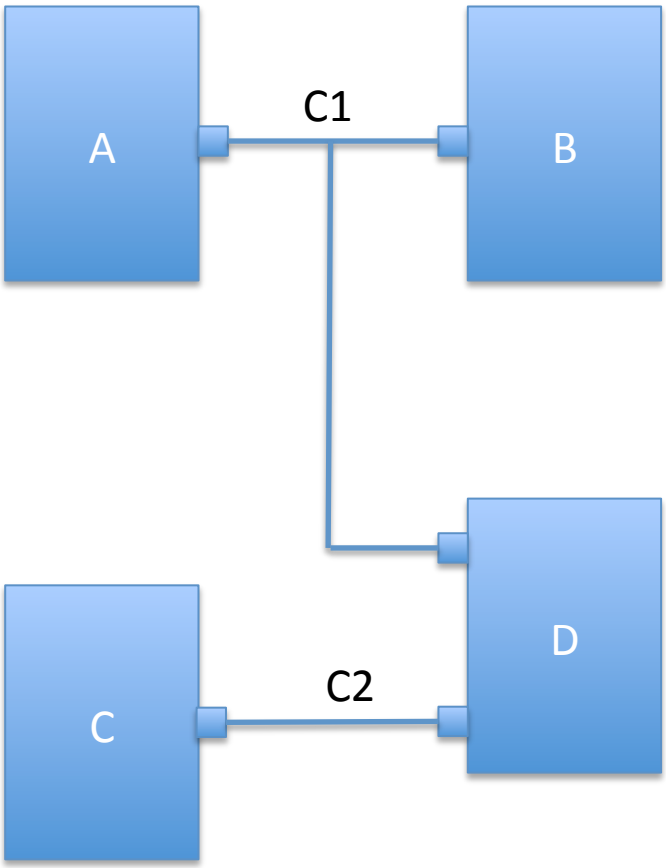
B calls *ScheduleEval* with delay 40

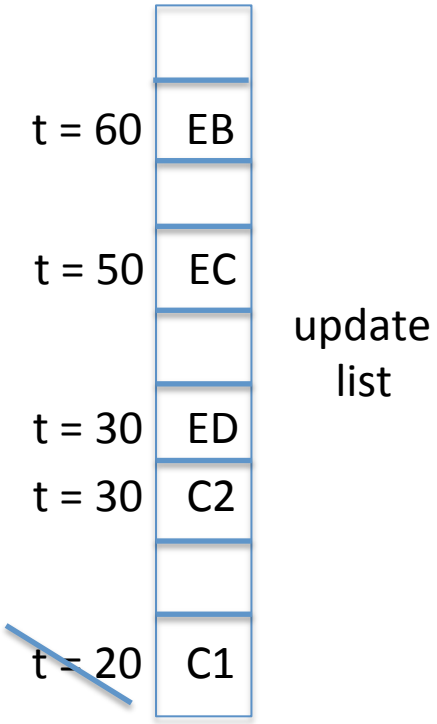


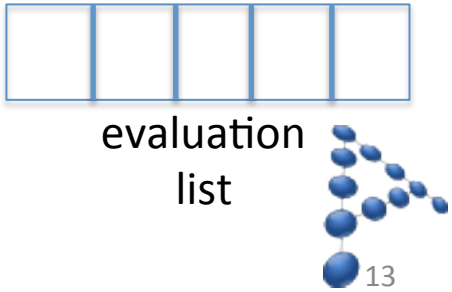




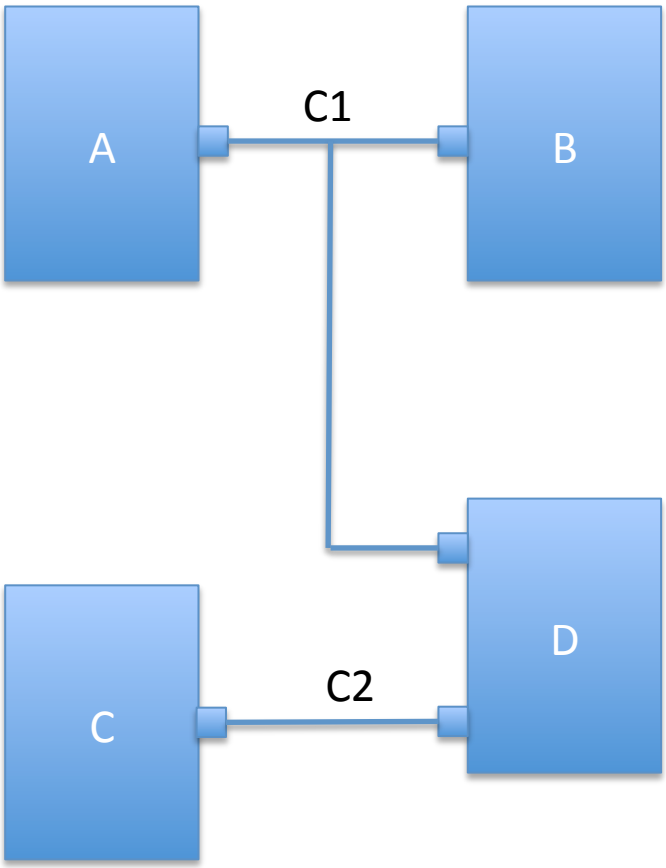
Time advances to t = 30: *two* updates

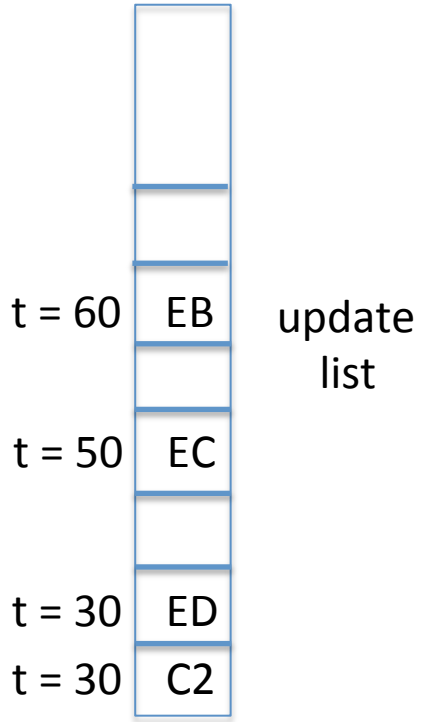


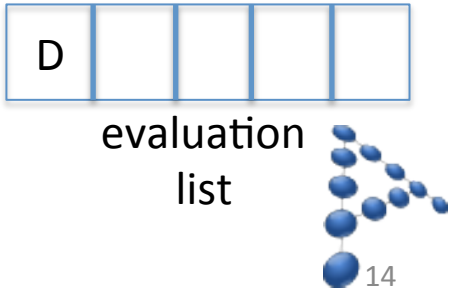




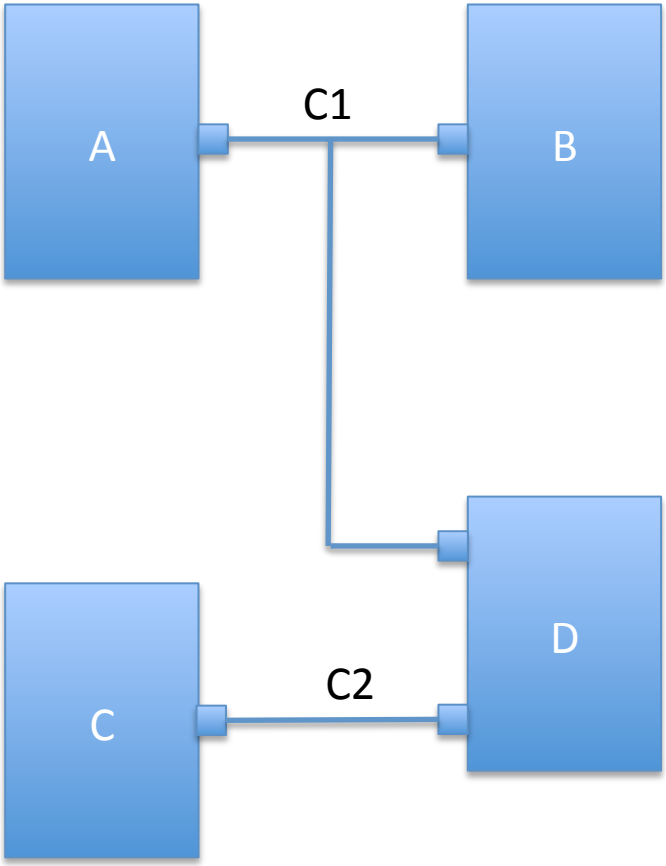
Time advances to t = 30: *two* updates, *one* eval

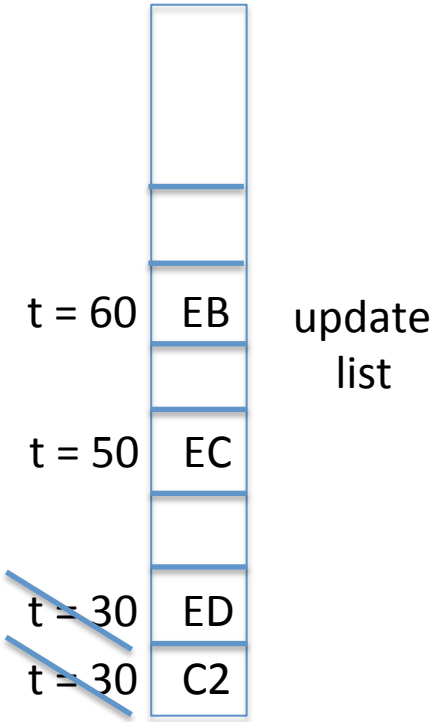


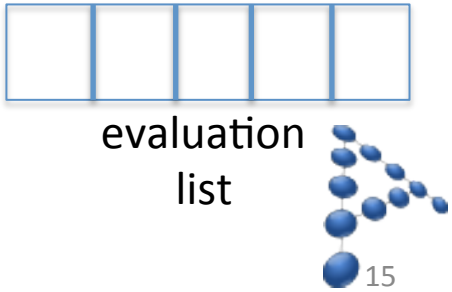




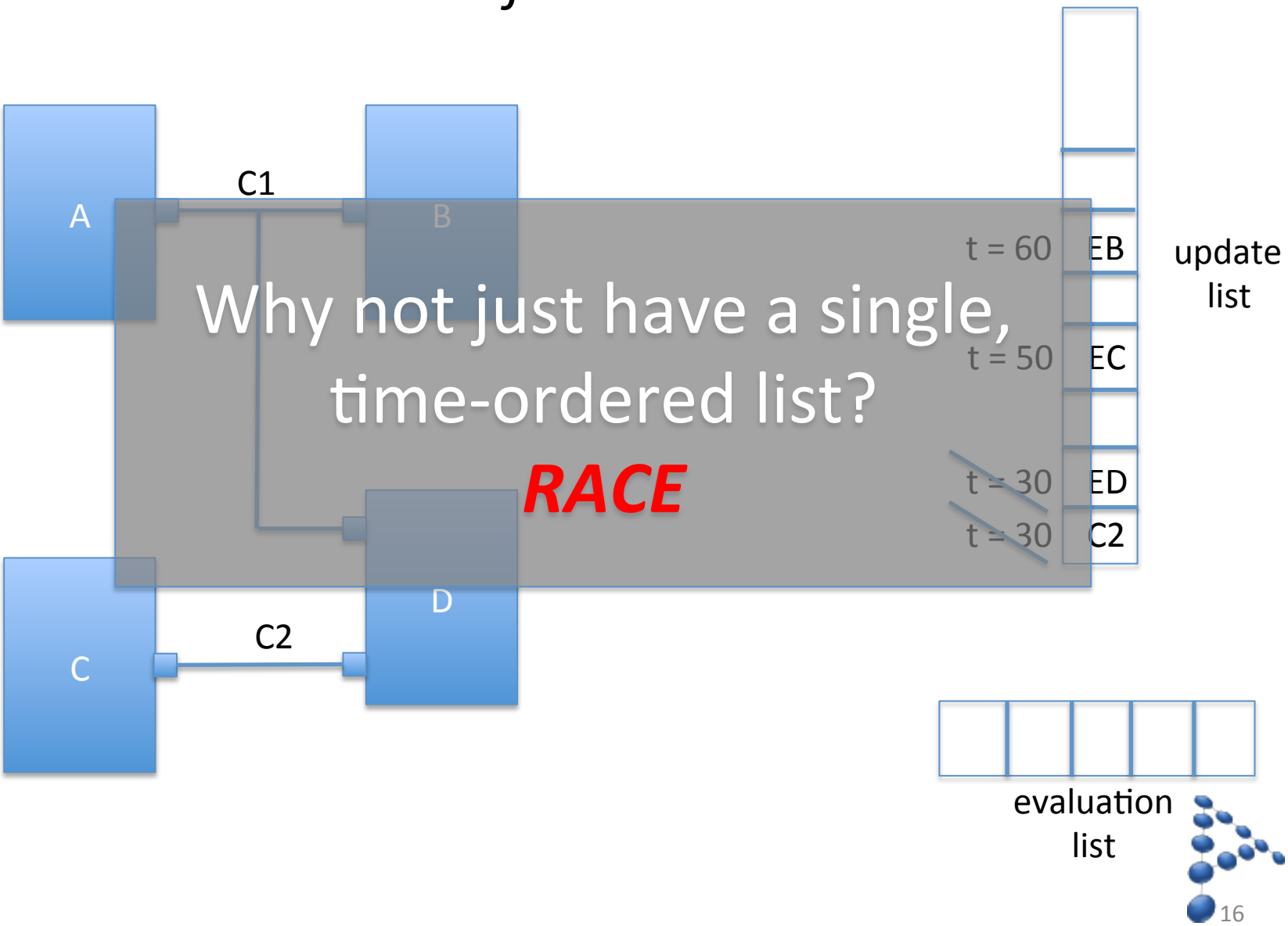
Simple Algorithm, Complex Implementation for *Performance*



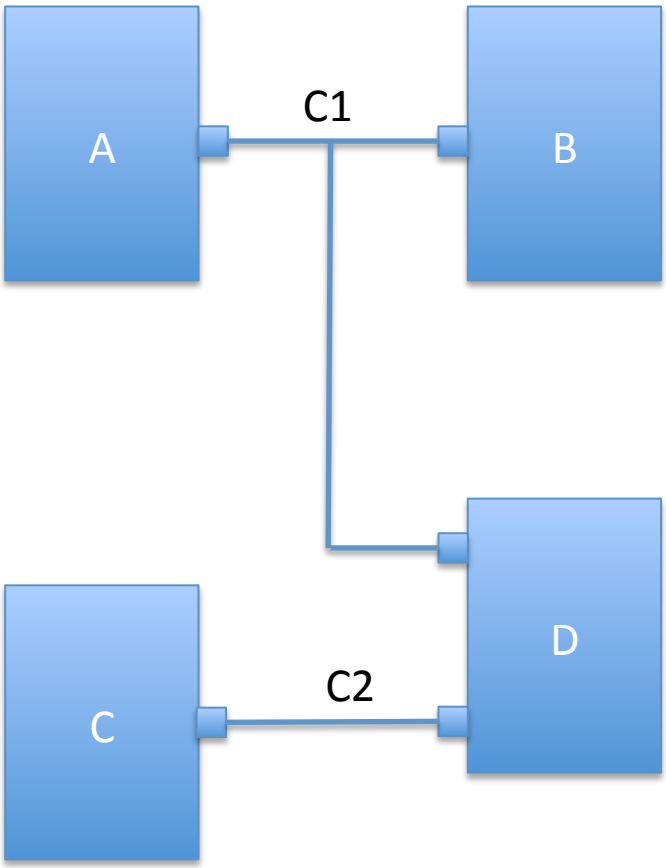




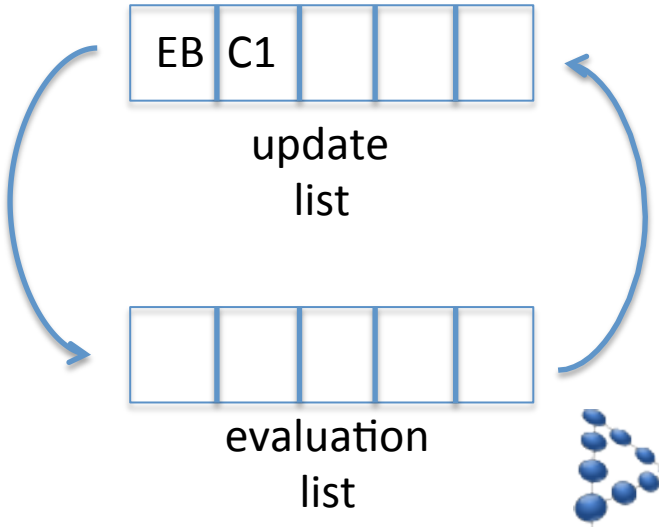
Simple Algorithm, Complex Implementation for *Performance*



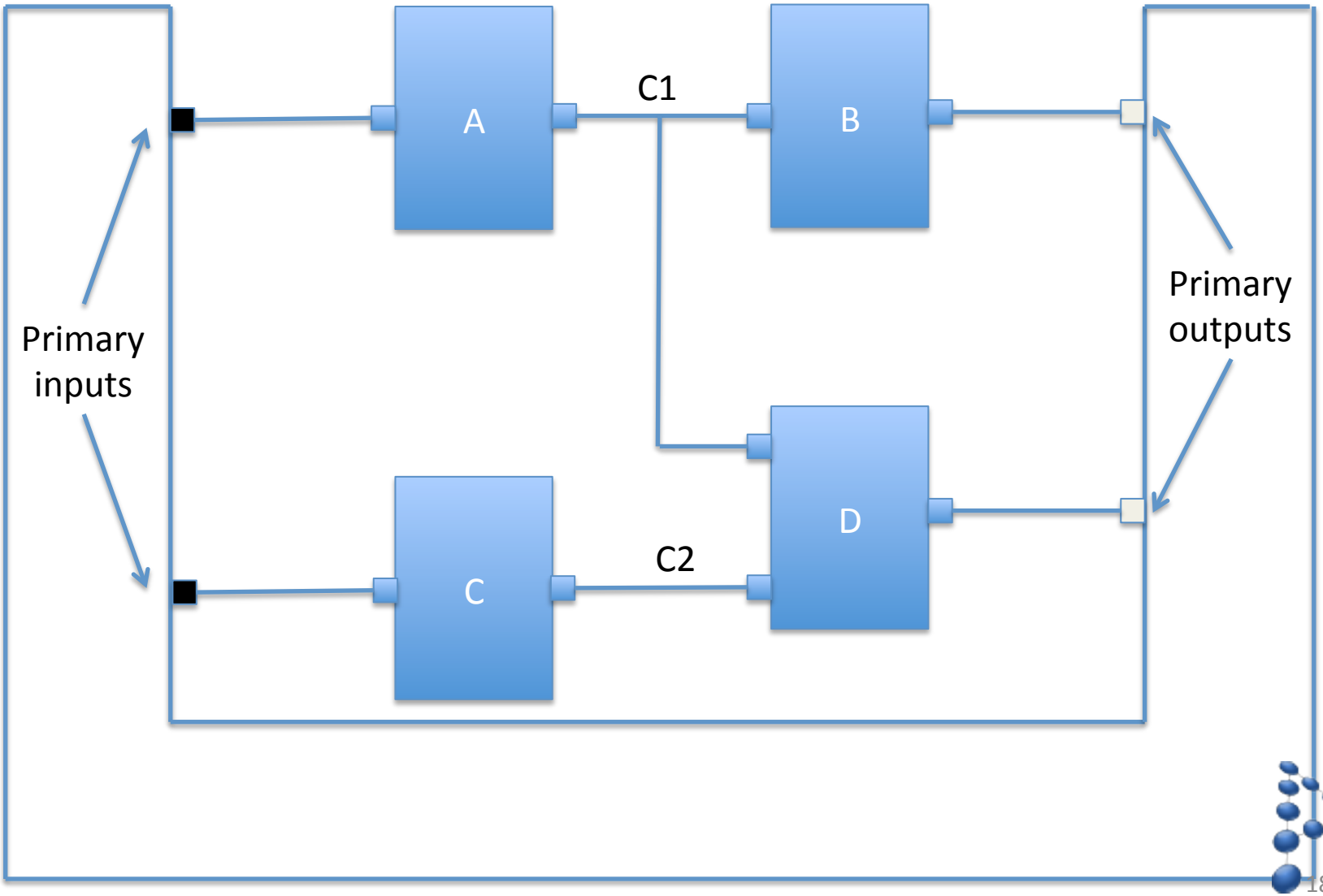
Simulating Models without Discrete Delays – *Unit Delay*



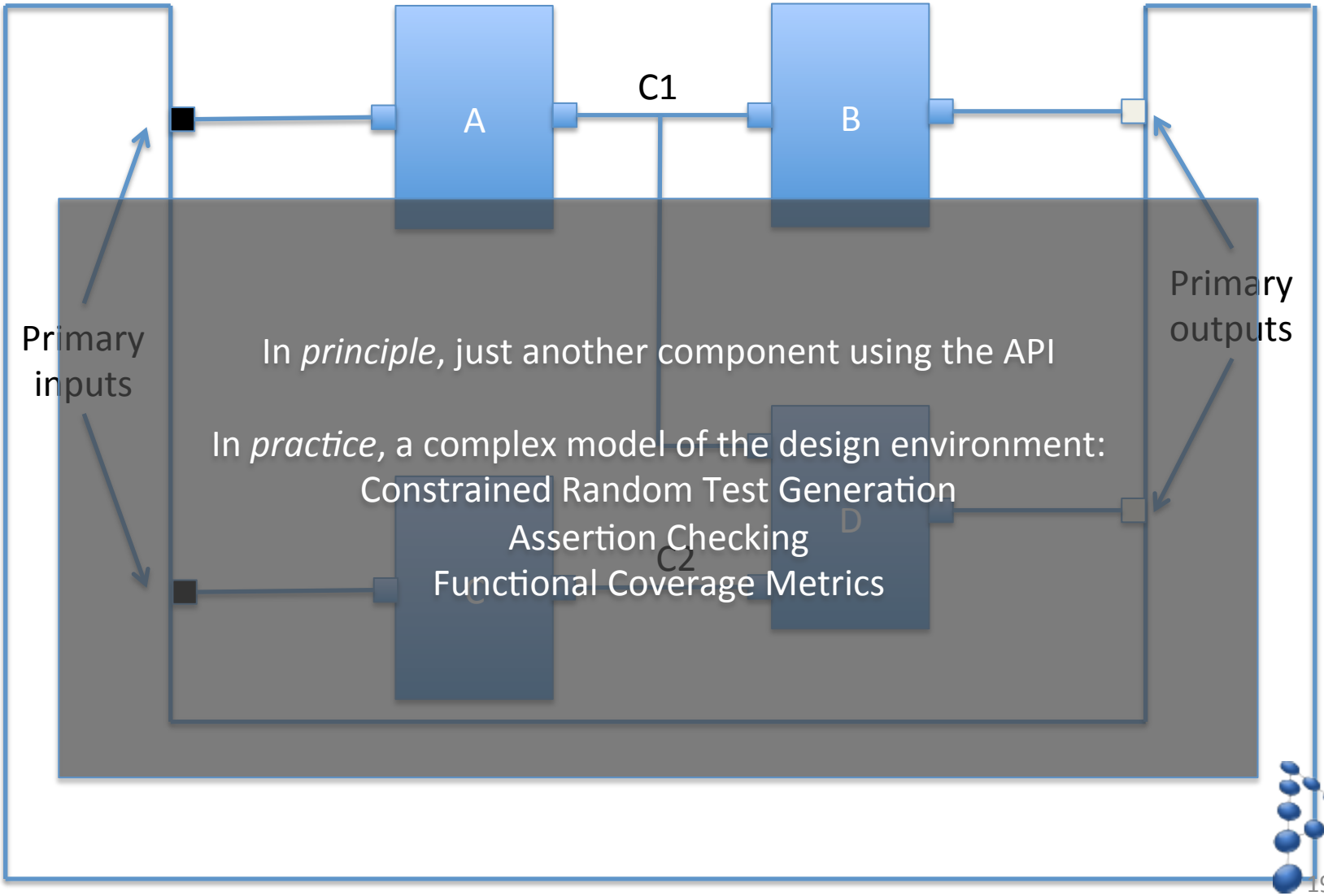
Each evaluate/update cycle advances time by one *tick*



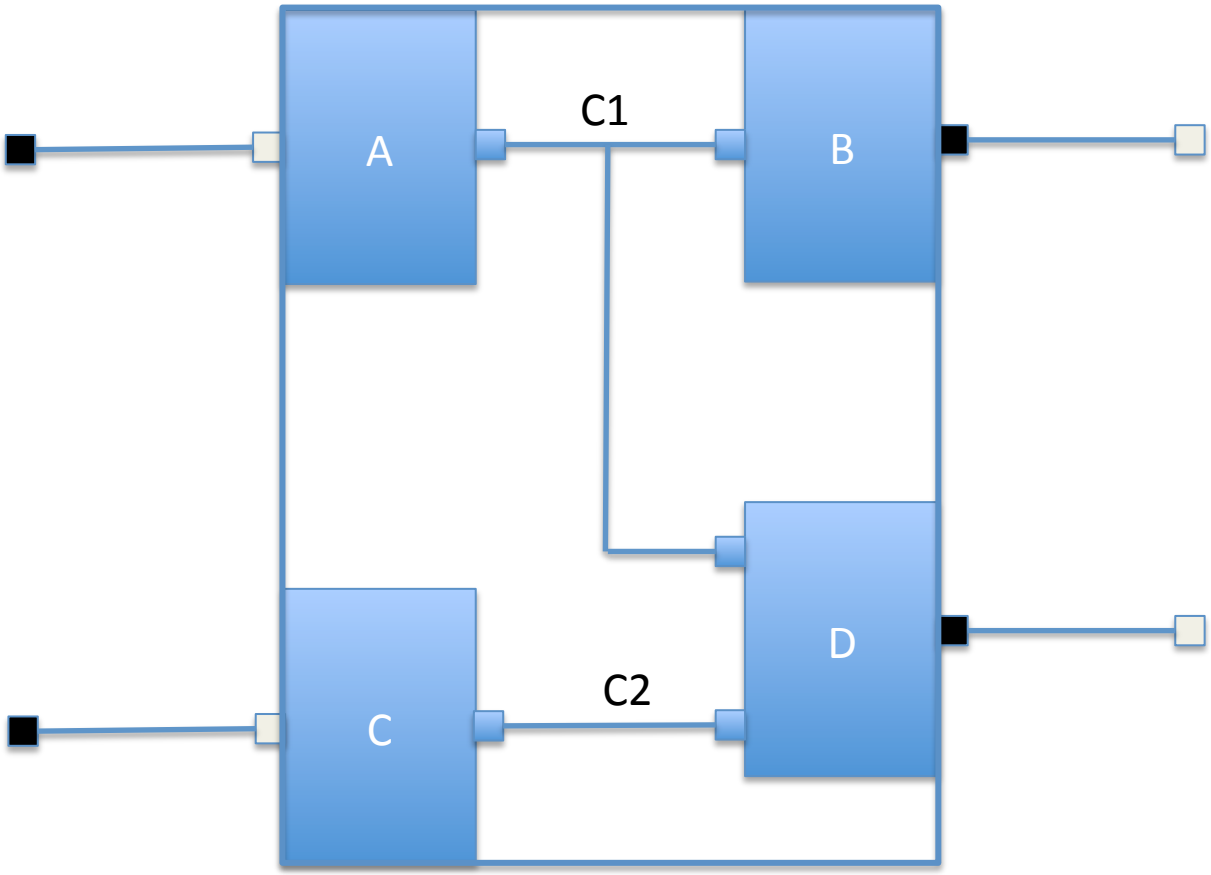
The Simulation Testbench



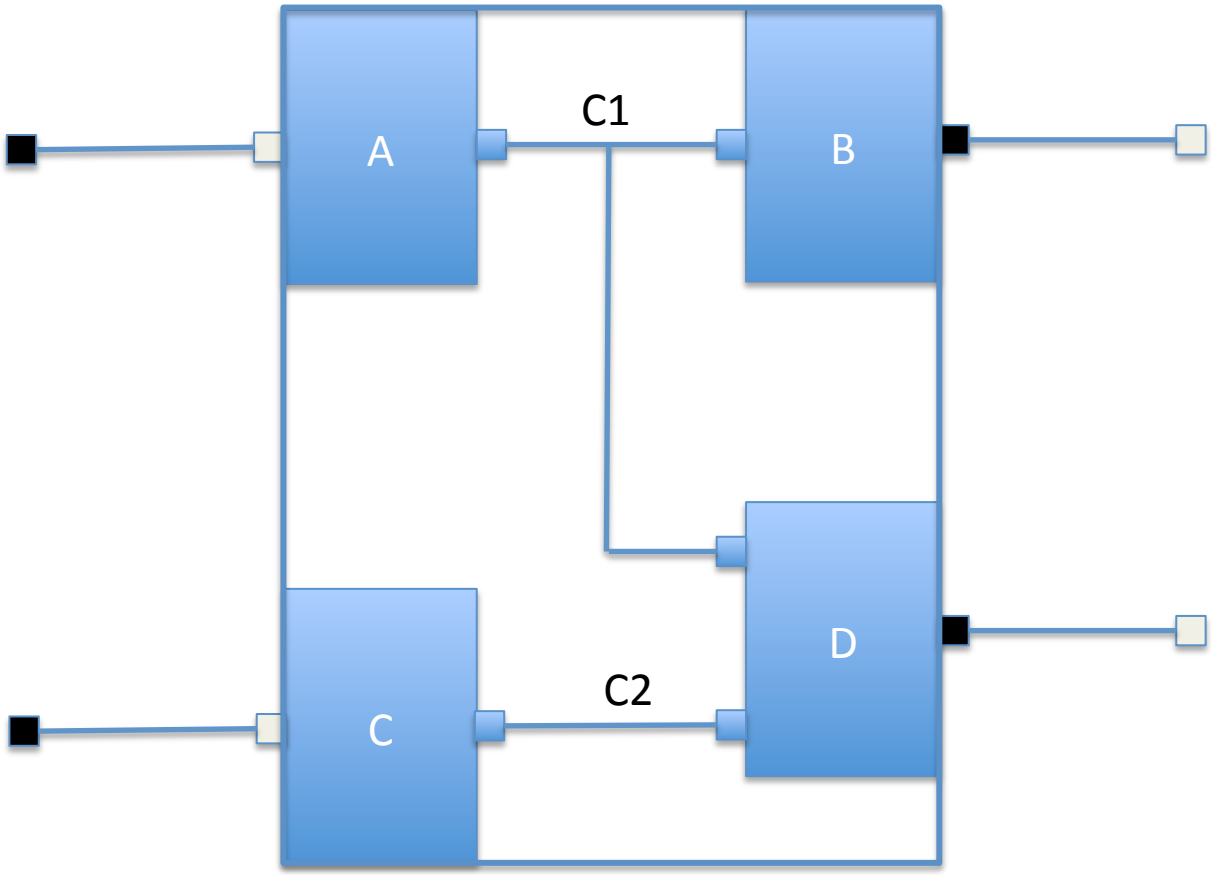
The Simulation Testbench



Discrete Event Simulation Languages: Hierarchy



Discrete Event Simulation Languages: Hierarchy

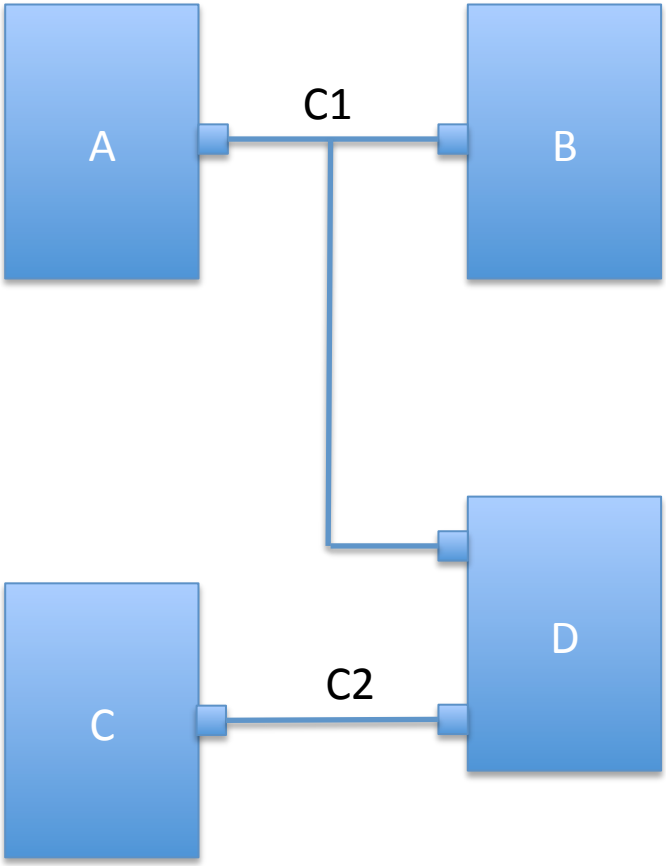


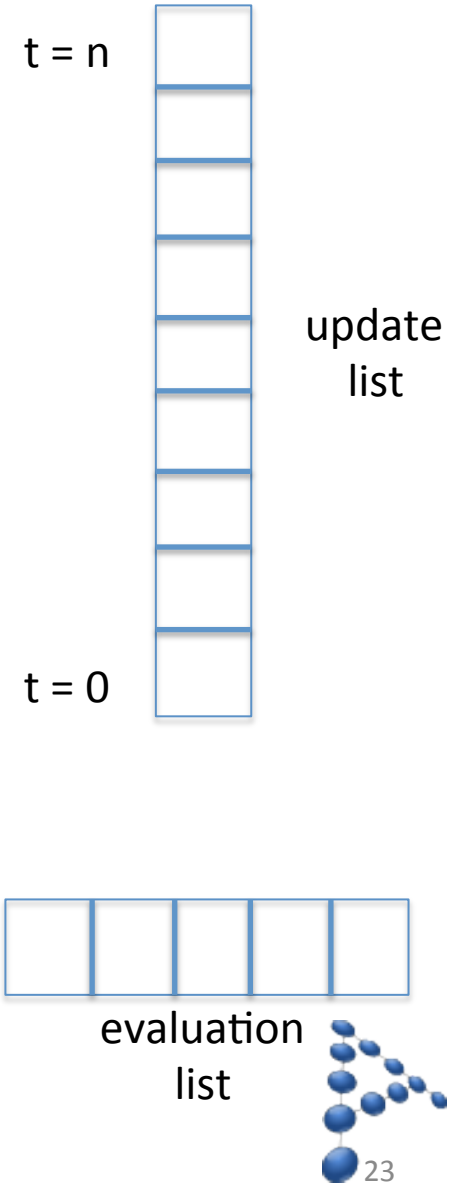
Hierarchy is *flattened* for simulation

Discrete Event Simulation Languages: Function

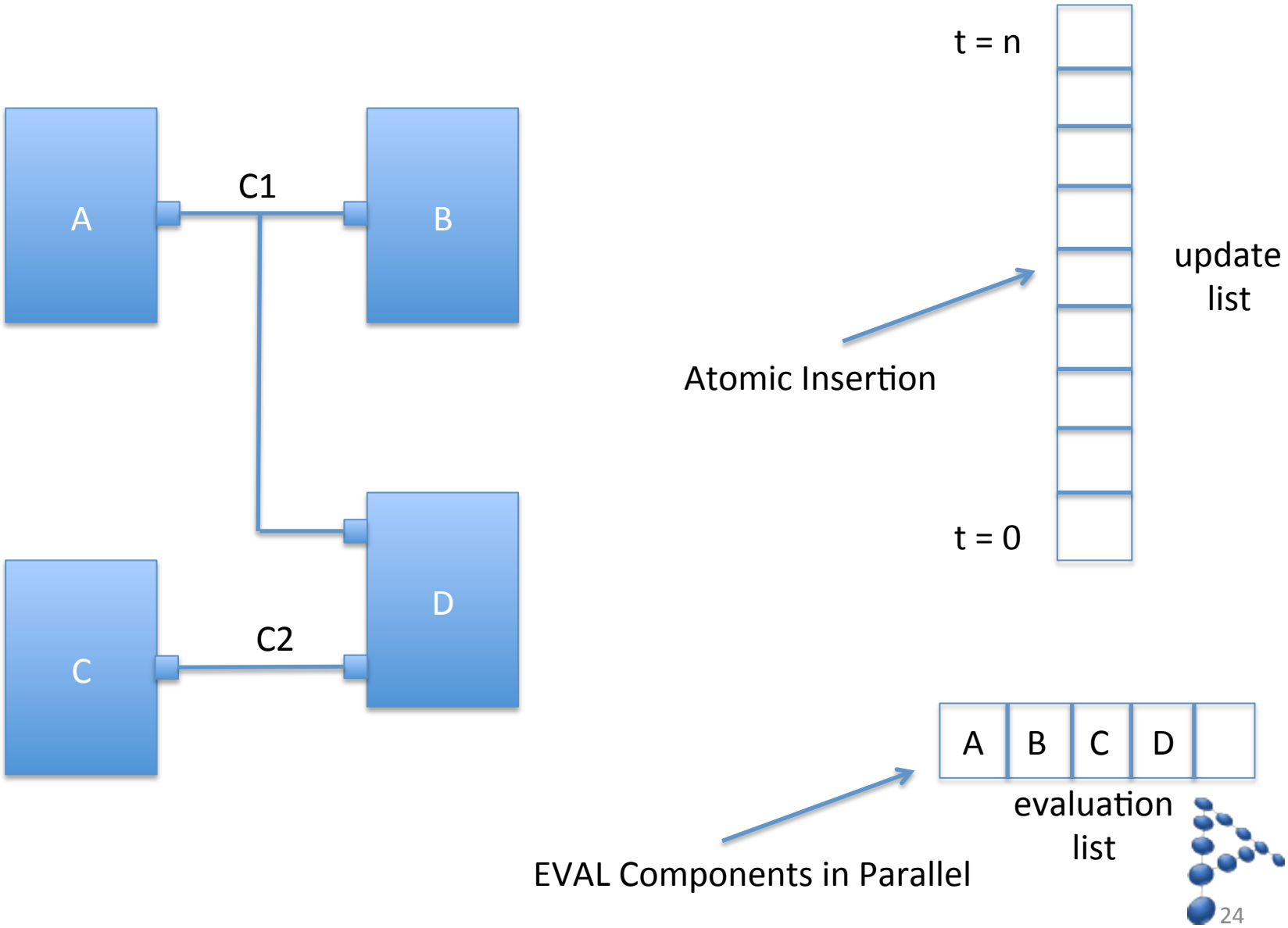
- There must be an Entry Point to implement the *Eval* API call
- *Eval* call must execute to completion
  - Method call
  - Actor
- Languages with Embedded *Wait*
  - *Eval* call must resume at the Wait Point
  - Implement as Finite State Machine
    - Stored Current State represents the next Entry Point

A Concurrent Simulator Implementation?

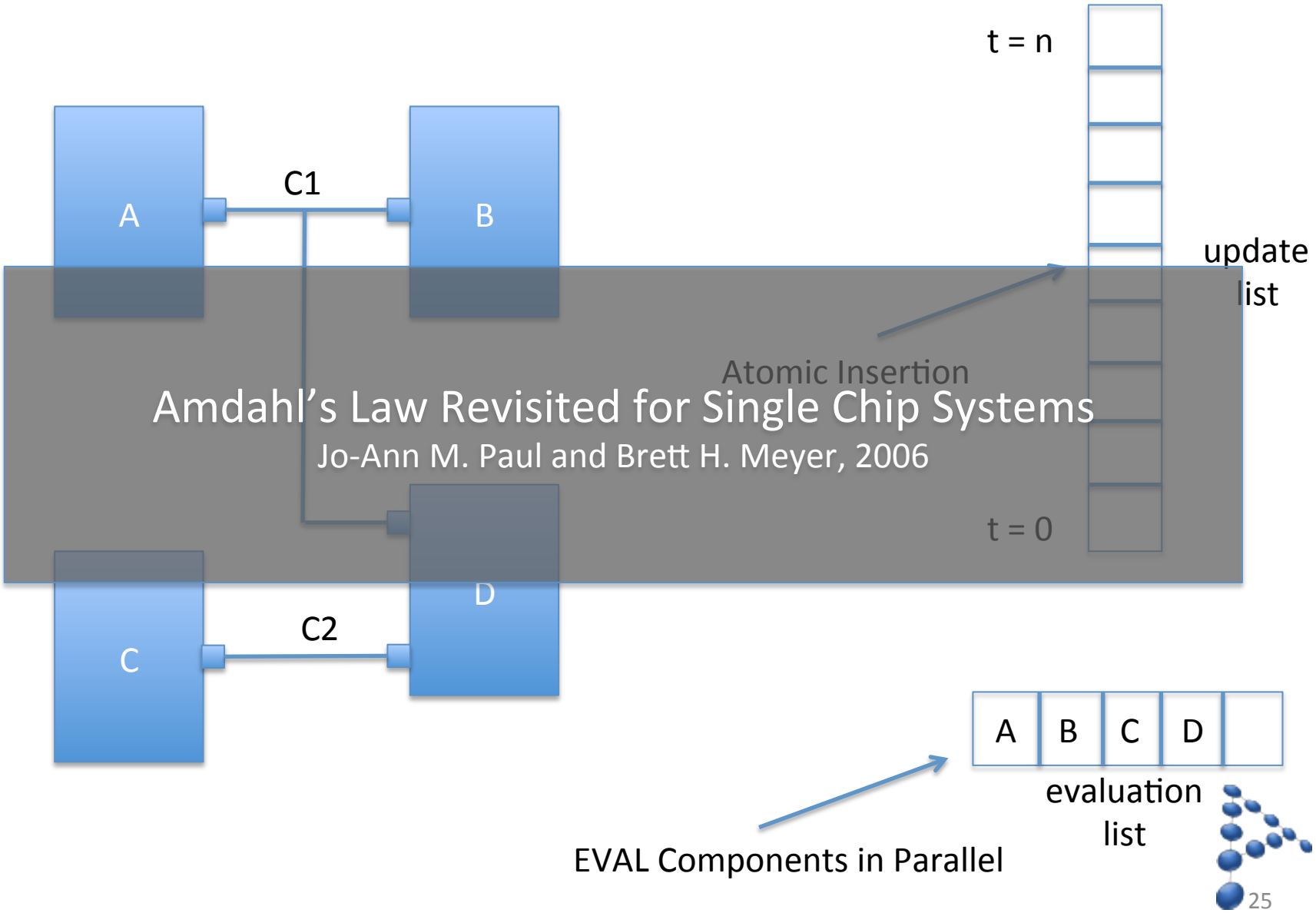




A Concurrent Simulator Implementation



A Concurrent Simulator Implementation



Event-Driven vs Process-Driven

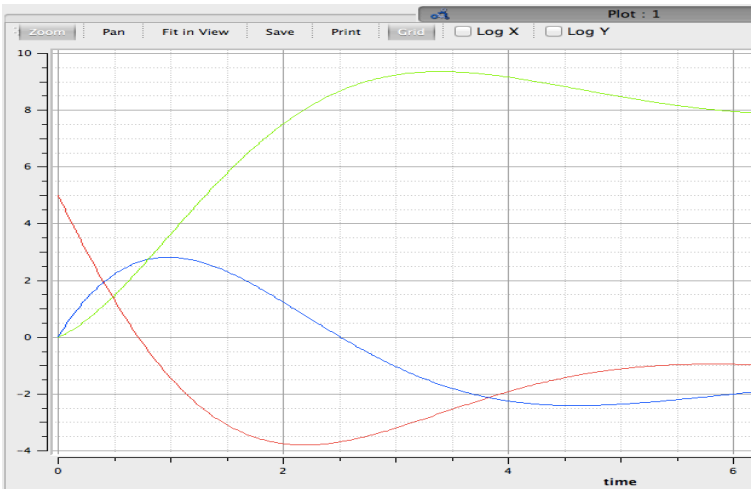
- Event-Driven
  - Nominally Single core, single process, *BUT*
  - OpenMP implementation
    - Portable
    - Optimised for multi-core
- Process-Driven
  - Each process runs as a concurrent Thread , *BUT*
    - Threads must manage communication/synchronisation
- Python Generators (coroutines)
    - Single core only (SimPy)

Continuous v Discrete Simulation

Continuous

- Equation-based

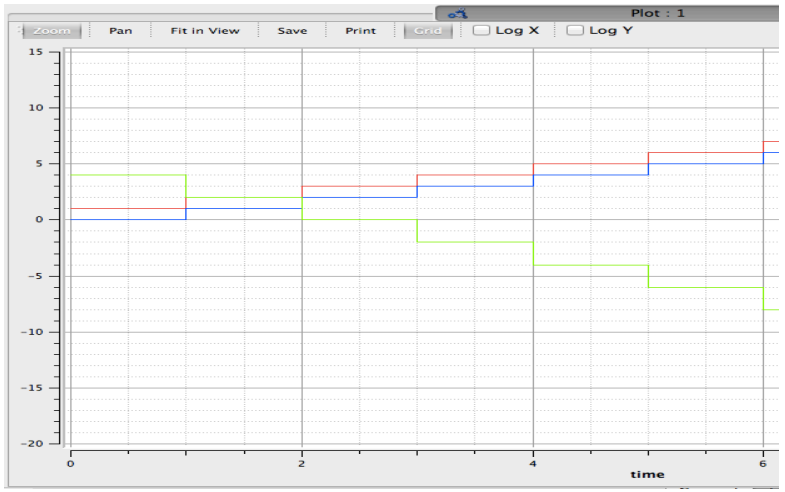
```
model cont01
  Real x, y, z;
equation
  x + y + z = 5.000;
 der(y) = x + 1.365;
  y = der(z) – 1.875;
end cont01;
```



Discrete

- Assignment-based

```
always @ (posedge clk)
begin
  x <= x + 1
  y <= x - 1
end
```



Hybrid Modeling

- Continuous-time + Discrete-time Modeling
  - Modelica "***when***"

```
model t03
  Real x, y, z;
equation
  x + y + z = 5;
  when sample(0, 1) then
    x = x + 1;
    y = delay(x – 1, 20);
  end when;
end t03;
```

Summary

- Discrete Event Simulation
  - Two-List Algorithm for Deterministic Execution
  - "Event-driven" or "Process-driven"
  - Hardware/Software
  - Multi-thread implementation with OpenMP
- Continuous Simulation
  - Modelica, Matlab/Simulink
- Hybrid Continous/Discrete Simulation
    - Discrete Interfaces