Sitar : Simulation Tool for Architectural Research

A framework for modeling cycle-based systems

System description language

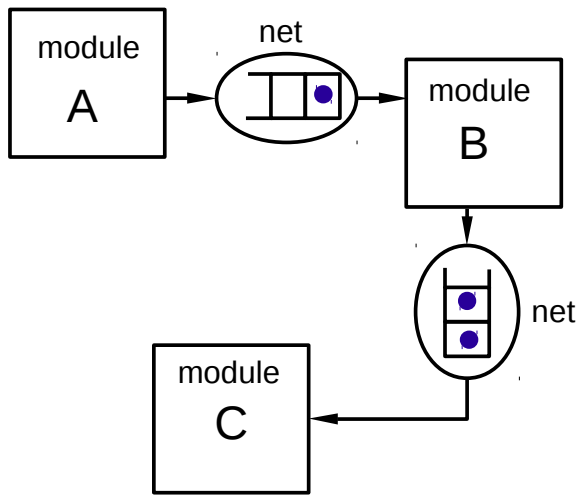
+

cycle-based simulation kernel

- Language: Constructs for modeling time-delays, structure, and concurrency on top of C++
- Simulation kernel: Lightweight, easy to parallelize
- V2.0 is open source (https://nehakaranjkar.github.io/sitar/)

Sitar: Simulation Kernel

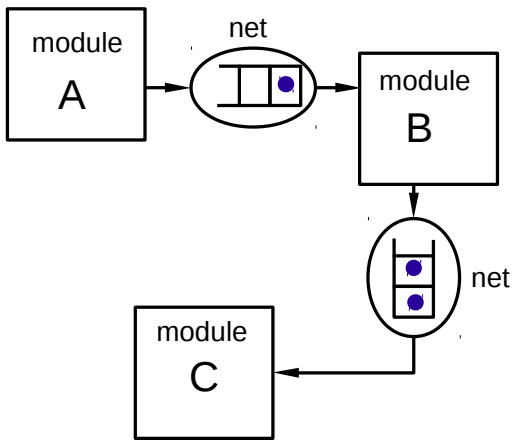
- A system consists of *modules* communicating over FIFO channels (*nets*) by transfer of data-tokens

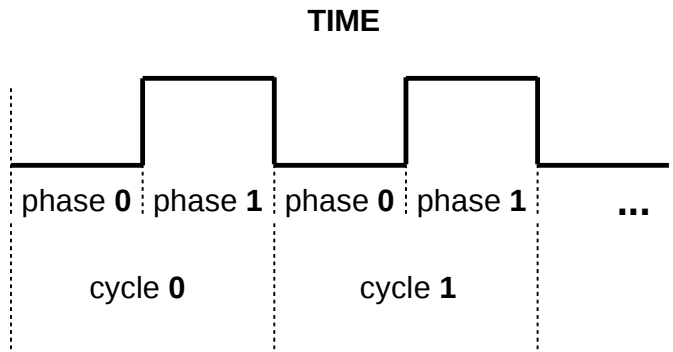


- All modules operate on a single clock.

Sitar: Simulation Kernel

- Two-phase execution (deterministic) : input in phase 0, output in phase 1





- Simple simulation algorithm, parallelized using OpenMp:

```
cycle = 0;
while (cycle<simulation_length)
{
        phase 0 : execute behavior of each module;
        phase 1 : execute behavior of each module;
        cycle = cycle + 1
}
```

Sitar: System Description Language

- Structure (interconnection of modules):
  - Hierarchy, Generics (templated modules), For-loops to create regular structures

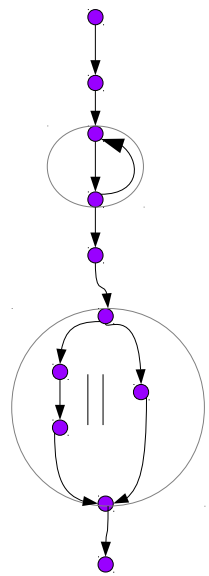
- Module Behavior: described in an imperative manner as a **sequence** of statements

Atomic statements (instantaneous)

- `wait (time)`, `wait until (condition)`
- C++ code blocks
- simulation control, logging

Compund statements (in-turn contain a sequence)

- `do-while`, `if-else`
- Parallel blocks (fork-join concurrency)
- Procedures



Sitar: System Description Language

- Each module description gets translated to a C++ class
- The translated classes can be linked with the simulation kernel to get a single simulation executable
- Systematic logging support