ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

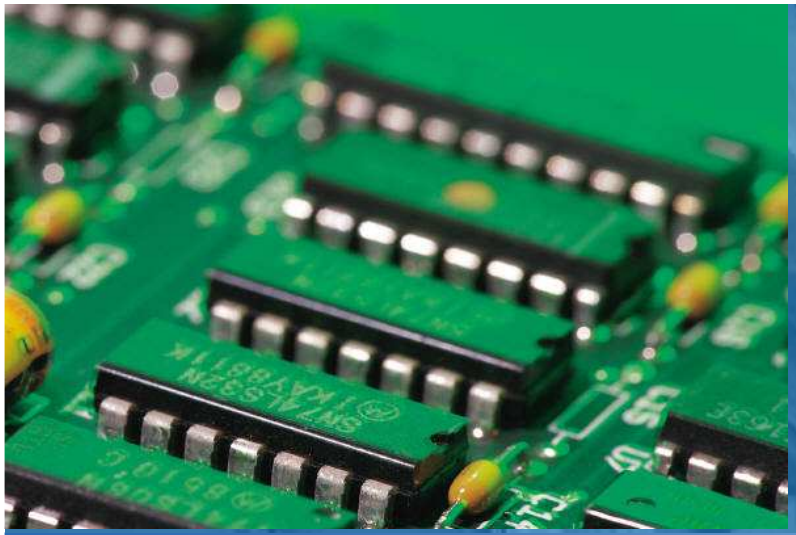
inf | Informatik
Computer Science

Fast Simulation of SystemC Designs with Scoot

Nicolas Blanc, Daniel Kroening

www.cprover.org/scoot

Supported by Intel and SRC



Theme of the Presentation

Utilization of the semantics of SystemC for Simulation speedup.

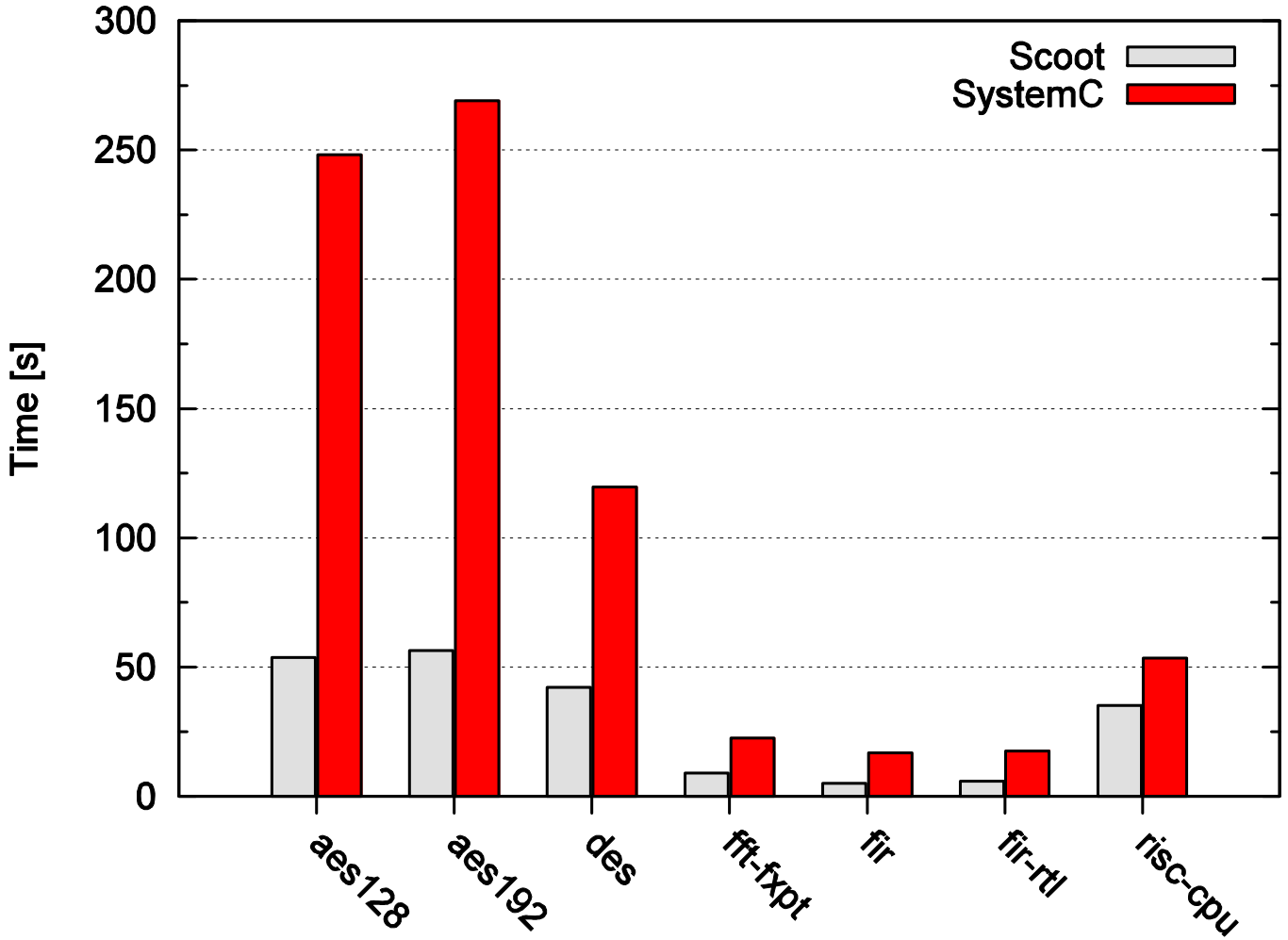
SYSTEM C

- System Description Language
- Based C++
- Compilation using g++
- IEEE Std.

Scoot

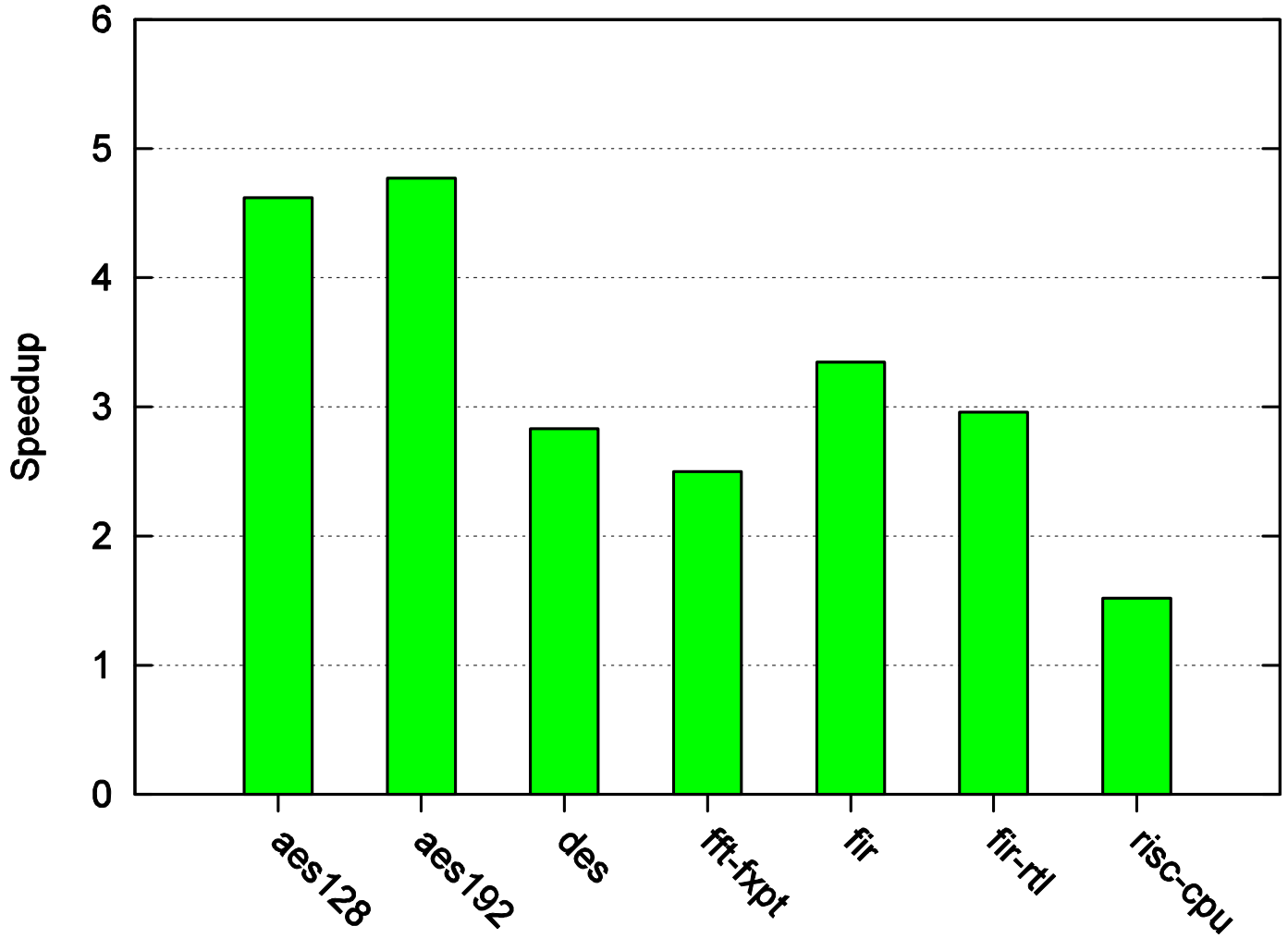
- Compiler for SystemC
- www.cprover.org/scoot

Execution Time (Oct 09, Linux 3Ghz, gcc 4.2.4, Linux)



17 Oct 09

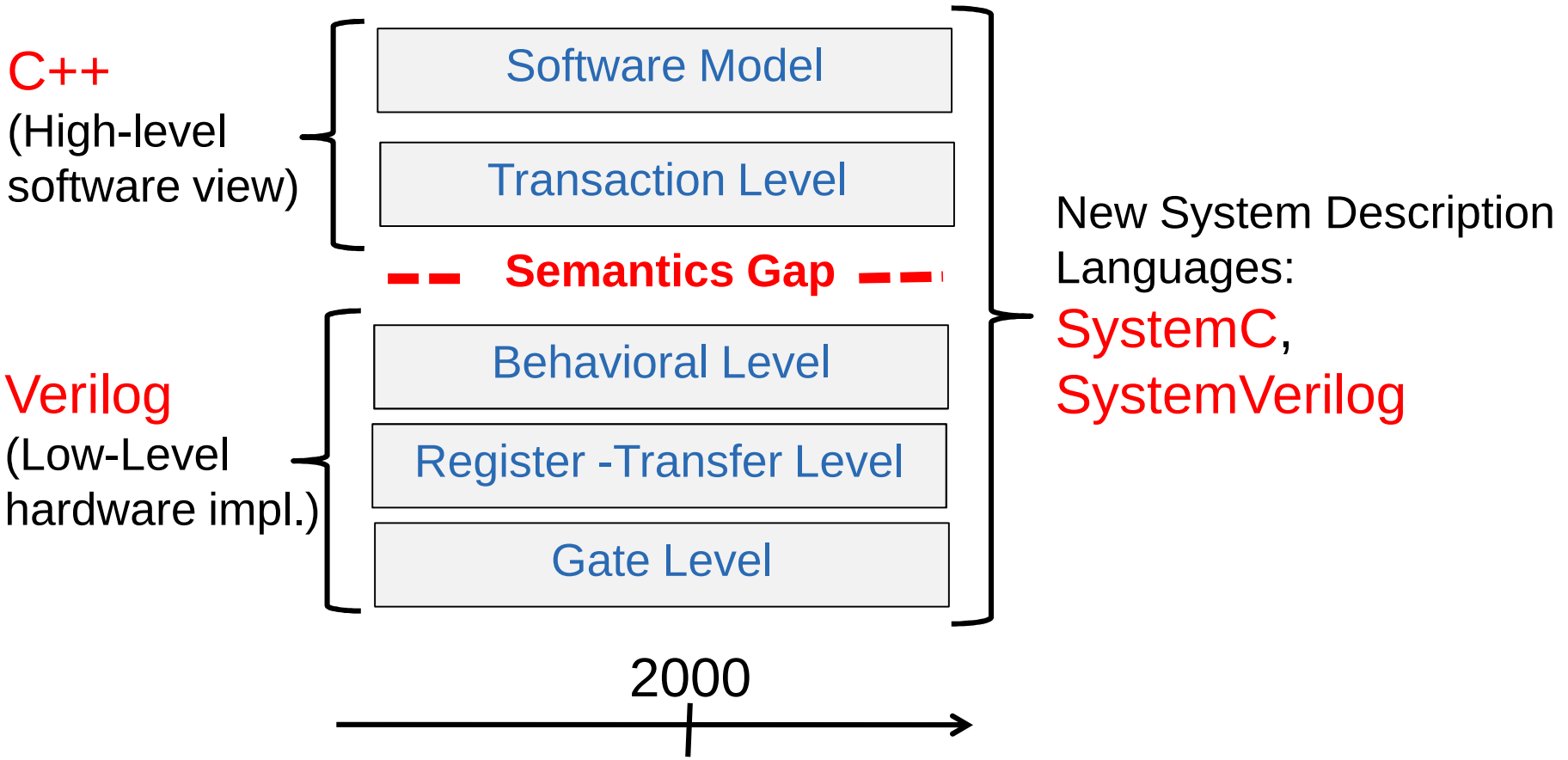
Simulation Speedup



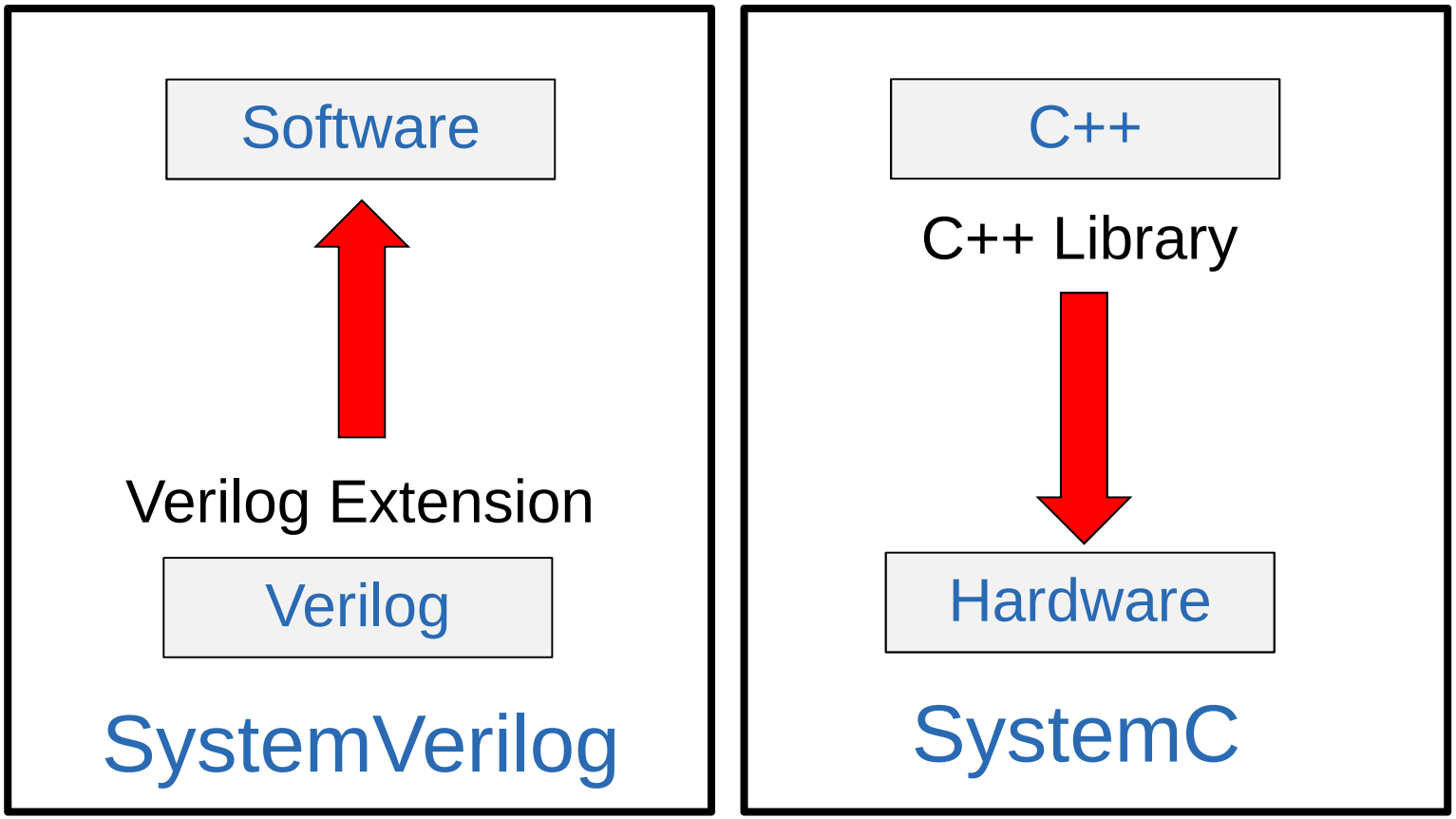
Outline

- Overview of SystemC
- Overview of Scoot
- Demo AES128

System Description Languages



SystemC versus SystemVerilog



Example: Memory Module

```
#include <systemc.h>

SC_MODULE (Memory) {
  sc_in<bool>           clk, cs, we, re;
  sc_in<sc_uint<32>  > data_in;
  sc_in<sc_uint<32>  > addr1, addr2;
  sc_out<sc_uint<32> > data_out;

  SC_CTOR(Memory){
    SC_METHOD(read());
    senstive << clk.pos();
    SC_METHOD(write());
    senstive << clk.pos()

  }
  void read();
  void write();
};
```

Inherits from sc_module

Module Interface

Module Constructor

Processes

```
int sc_main(int argc, char* argv[])
{
  sc_clock clk;
  sc_signal<bool> cs, re, we;
  ...

  Memory mem(« MEMORY »);
  Testbench tb(« TEST BENCH »);

  mem.clk(clk);
  mem.cs(cs); mem.re(re); mem.we(we);
  ...

  sc_start(10,SC_US);
  return 0;
}
```

Signals

Modules

Port Binding

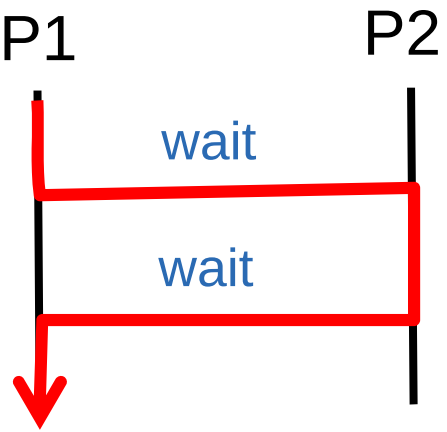
Start Simulation

SystemC Simulation:

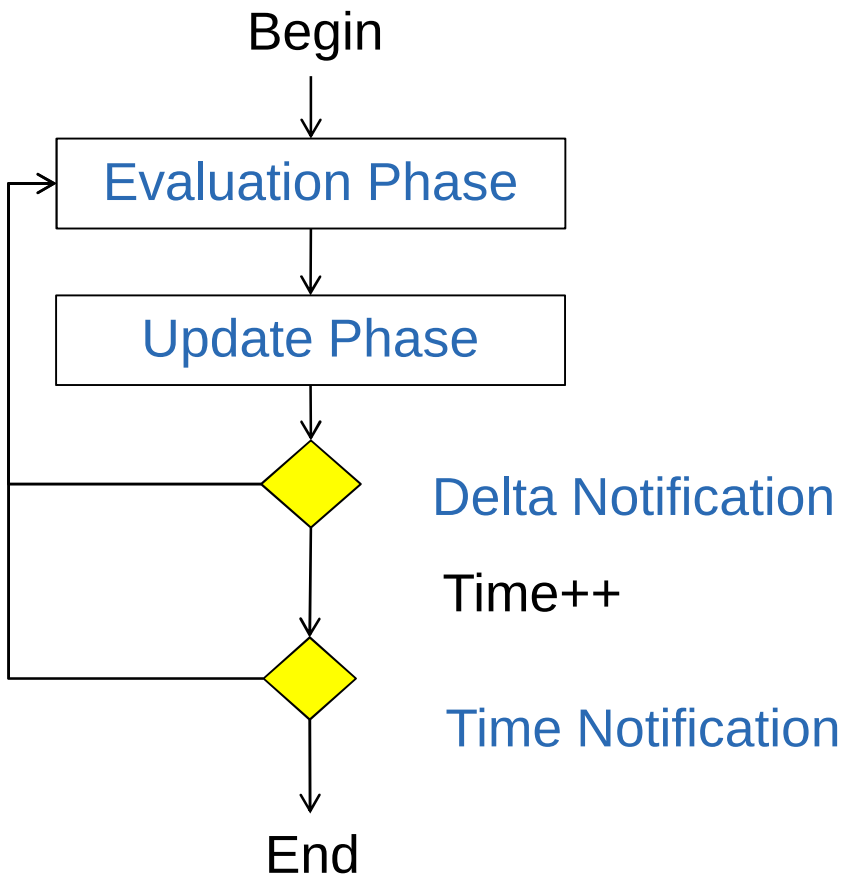
```
g++ main.cpp memory.cpp tb.cpp –lsystemc –o  simulator
```

The Concurrency Model

- Execution driven by events
- Cooperative Multitasking Model:
  - Only one process running at a time
  - **No preemption!**



The SystemC Scheduler

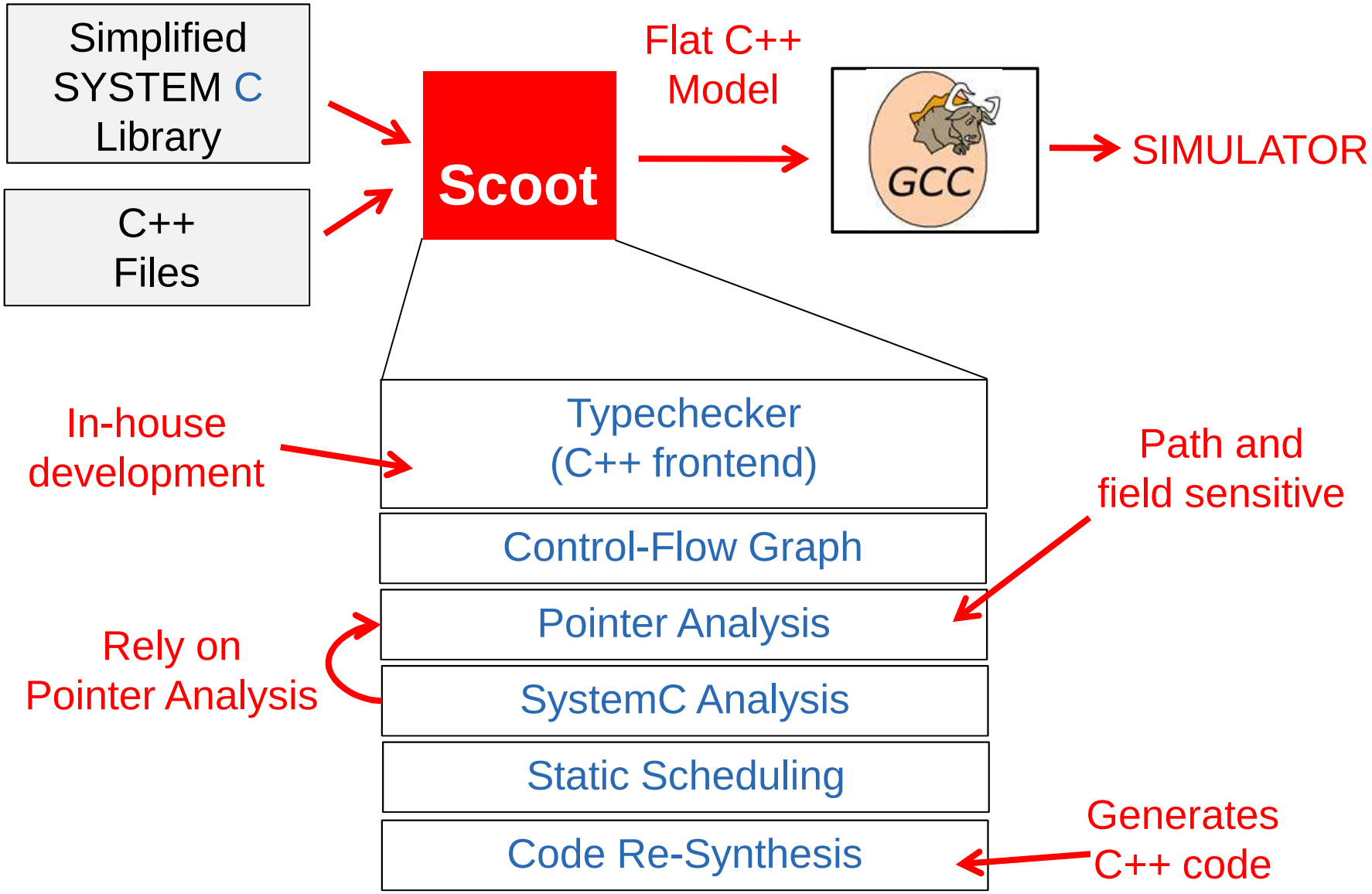


Observations

- Elaboration of the Module Hierarchy at Runtime:
  - Modules, processes, port binding,... The approach is flexible!
- C++ is fast: Fast execution of the processes!
- Yes ... but, GCC is not taking advantage of SystemC information!
  - module hierachy, processes, and port binding.

Overview of Scoot

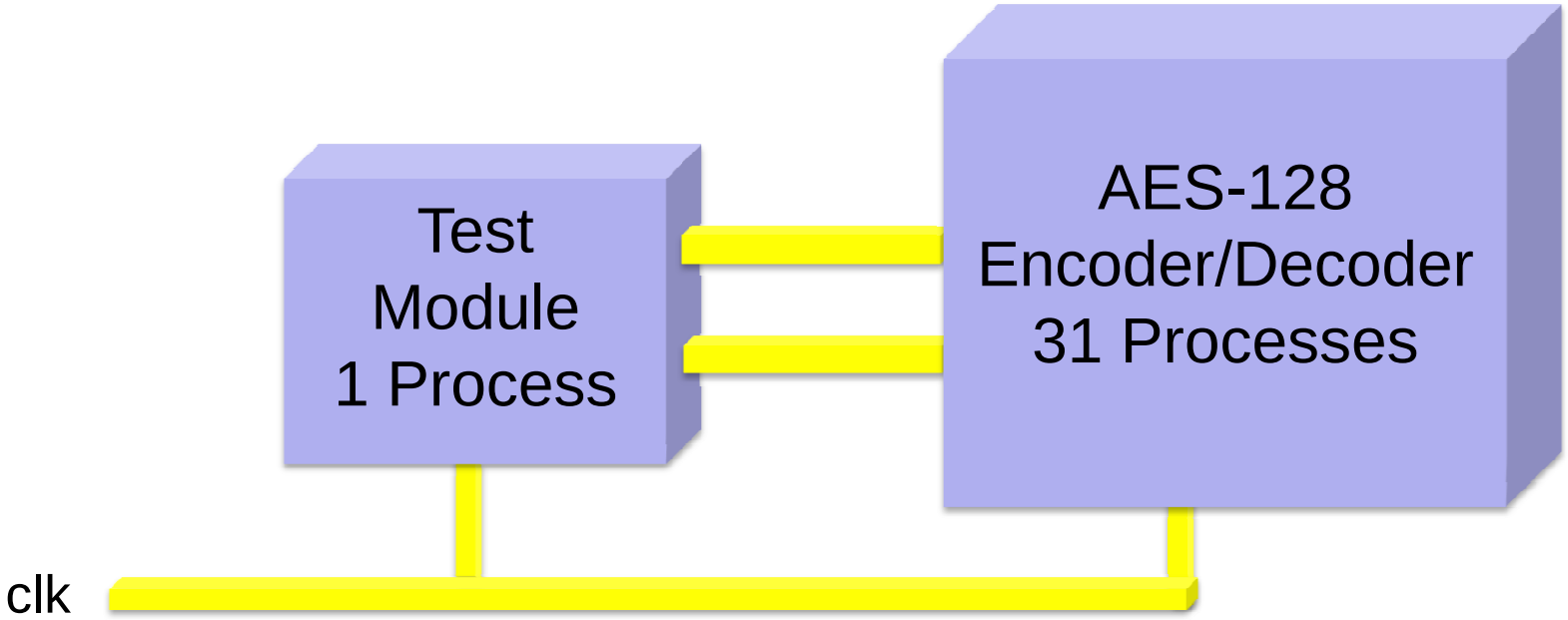
- Scoot statically discovers:
  - Module hierarchy, port binding, processes, and sensitivity lists.
- Simulation benefits from:
  - Resolution of dynamic calls (static-scheduling)
  - Suppression of dynamic data structures in the scheduler (lists, sets).
  - Propagation of port binding information (pointers).


Simplified SYSTEM C Library
C++ Files
Scoot
Flat C++ Model
GCC
SIMULATOR
In-house development
Typechecker (C++ frontend)
Control-Flow Graph
Pointer Analysis
SystemC Analysis
Static Scheduling
Code Re-Synthesis
Path and field sensitive
Rely on Pointer Analysis
Generates C++ code

Demo

Scoot

Benchmark: Encrypt, decrypt, and then display 128-bit vectors .
Simulation Time: 800 Microseconds.



Conclusion

Scoot

- Elaboration of the Module Hierarchy at Compile Time:
  - We sacrifice some flexibility in exchange for
  - significant simulation speedup, and
  - we can now reason about SystemC models statically!
    - Formal Verification, e.g., previous talk about static race analysis.

Thank You!