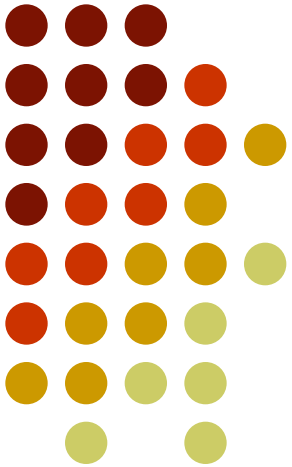
SISTEME DE CALCUL DEDICATE

Curs 1



Cuprins

- SystemC
  - Istoric
  - ESL
  - Privire de ansamblu
  - Example
- Bibliografie

Istoric

- ***SystemC is a system design language based on C++.***
- SystemC started out as a very restrictive cycle-based simulator and “yet another” RTL language
- the Open SystemC Initiative (OSCI) was formed in 1999

Date	Version	Notes
Sept 1999	0.9	First version; cycle-based
Feb 2000	0.91	Bug fixes
Mar2000	1.0	Widely accessed major release
Oct 2000	1.0.1	Bug fixes
Feb 2001	1.2	Various improvements
Aug 2002	2.0	Add channels & events; cleaner syntax
Apr 2002	2.0.1	Bug fixes; widely used
June 2003	2.0.1	LRM in review
Spring 2004	2.1	LRM submitted for IEEE standard
Dec 2005	2.1v1	IEEE 1666-2005 ratified
July 2006	2.2	Bug fixes to more closely implement IEEE 1666-2005

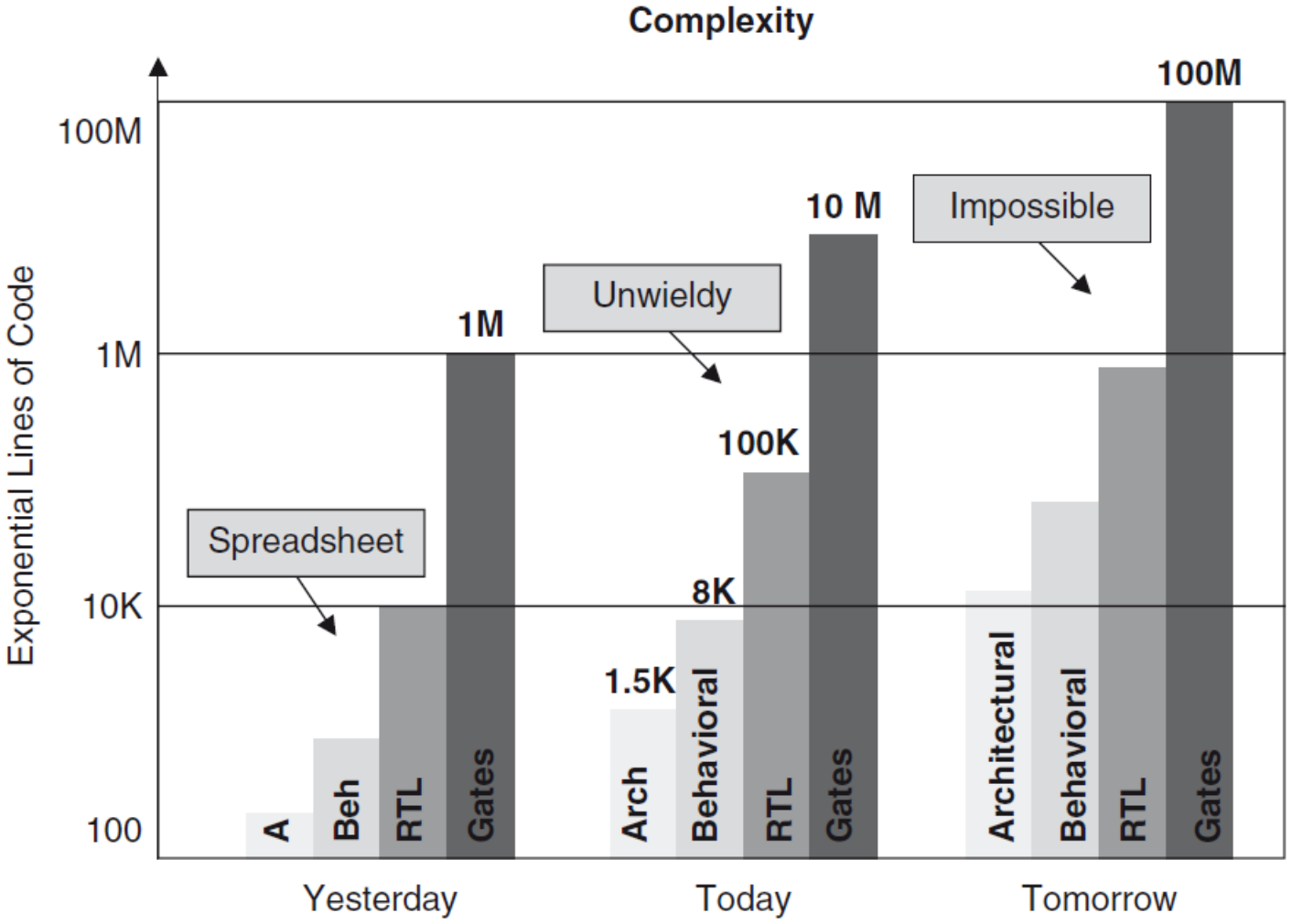
SystemC
electronic system-level design

- SystemC is a system design and modeling language.
- The prevailing name for the concurrent and multidisciplinary approach to the design of complex systems is ***electronic system-level design*** or ESL:
  - ESL happens by modeling systems at higher levels of abstraction
  - Portions of the system model are subsequently iterated and refined, as needed
  - A set of techniques has evolved called ***Transaction-Level Modeling*** or TLM to aide with this task

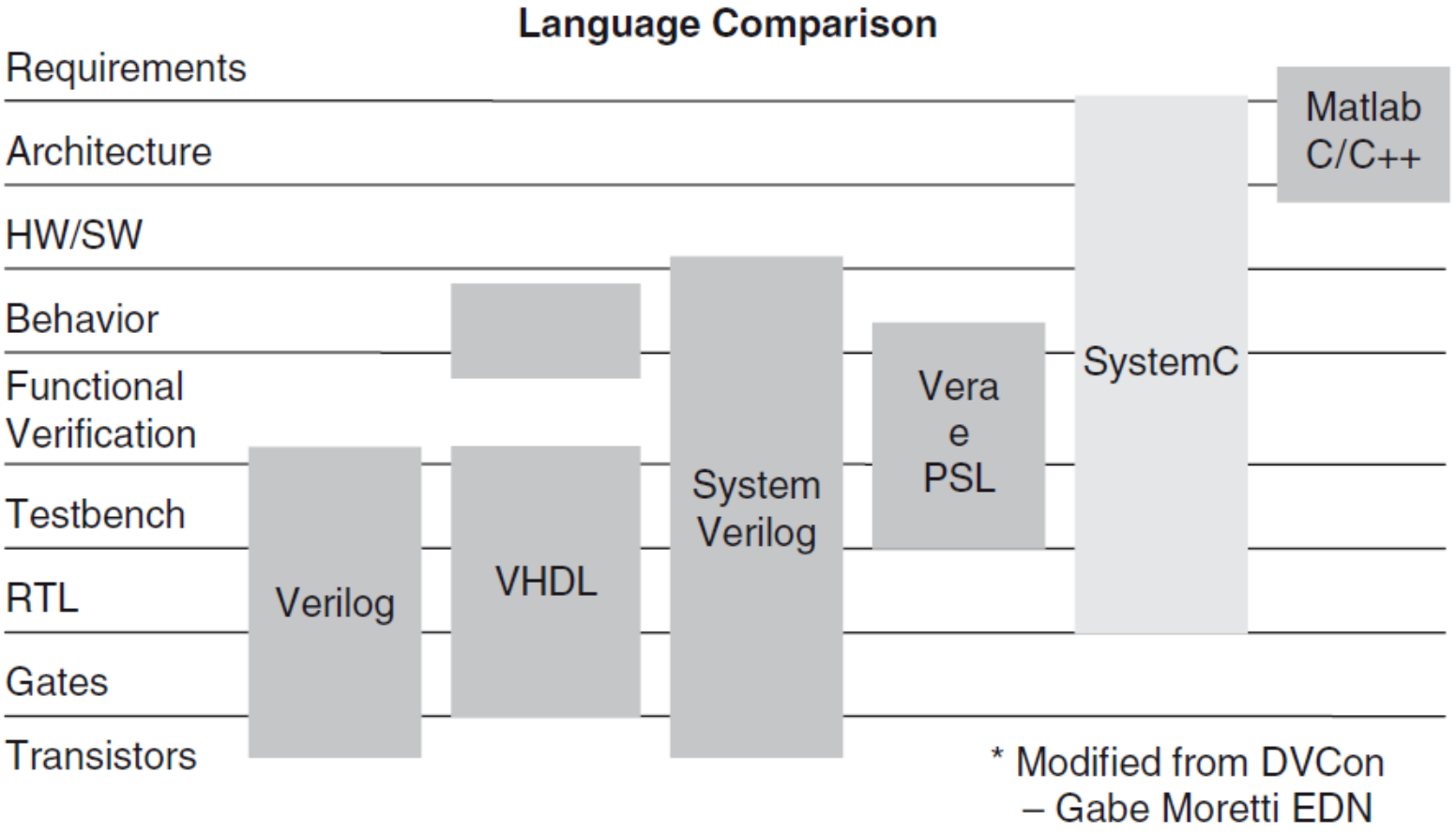
SystemC
electronic system-level design

- ESL techniques evolved in response to increasing design complexity and increasingly shortened design cycles
- System development teams find themselves asking questions like:
  - Should this function be implemented in hardware, software, or with a better algorithm?
  - Does this function use less power in hardware or software?
  - Do we have enough interconnect bandwidth for our algorithm?
  - What is the minimum precision required for our algorithm to work?
- ***Shortened Design Cycle = Need For Concurrent Design***

SystemC
electronic system-level design

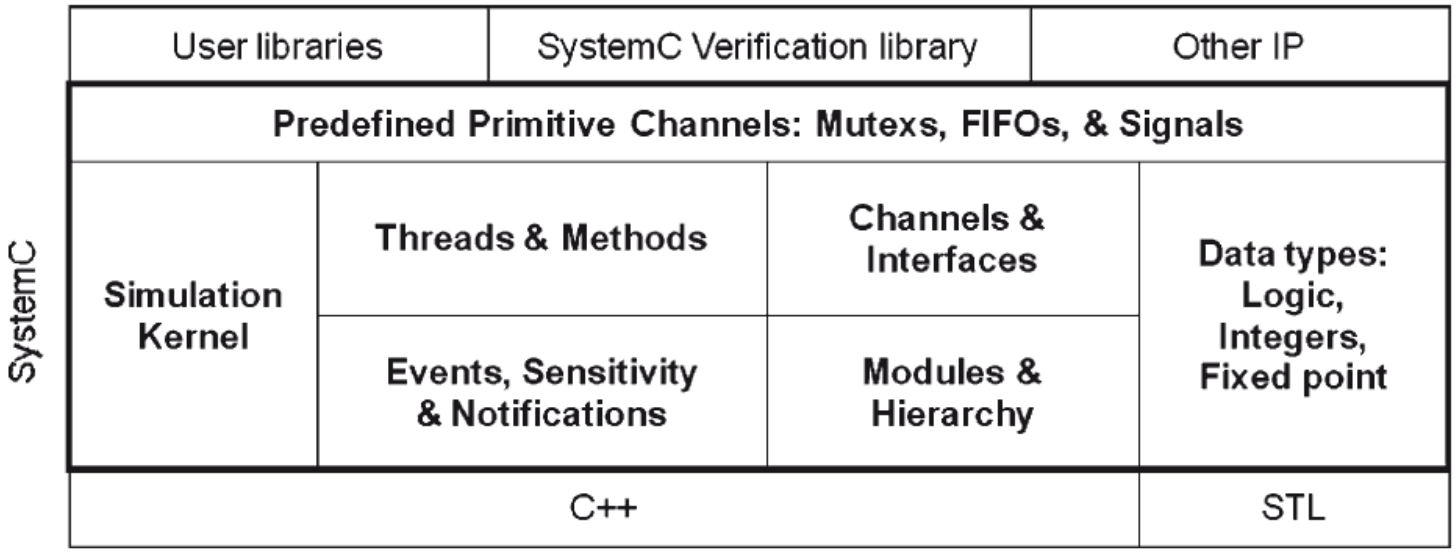


SystemC
electronic system-level design

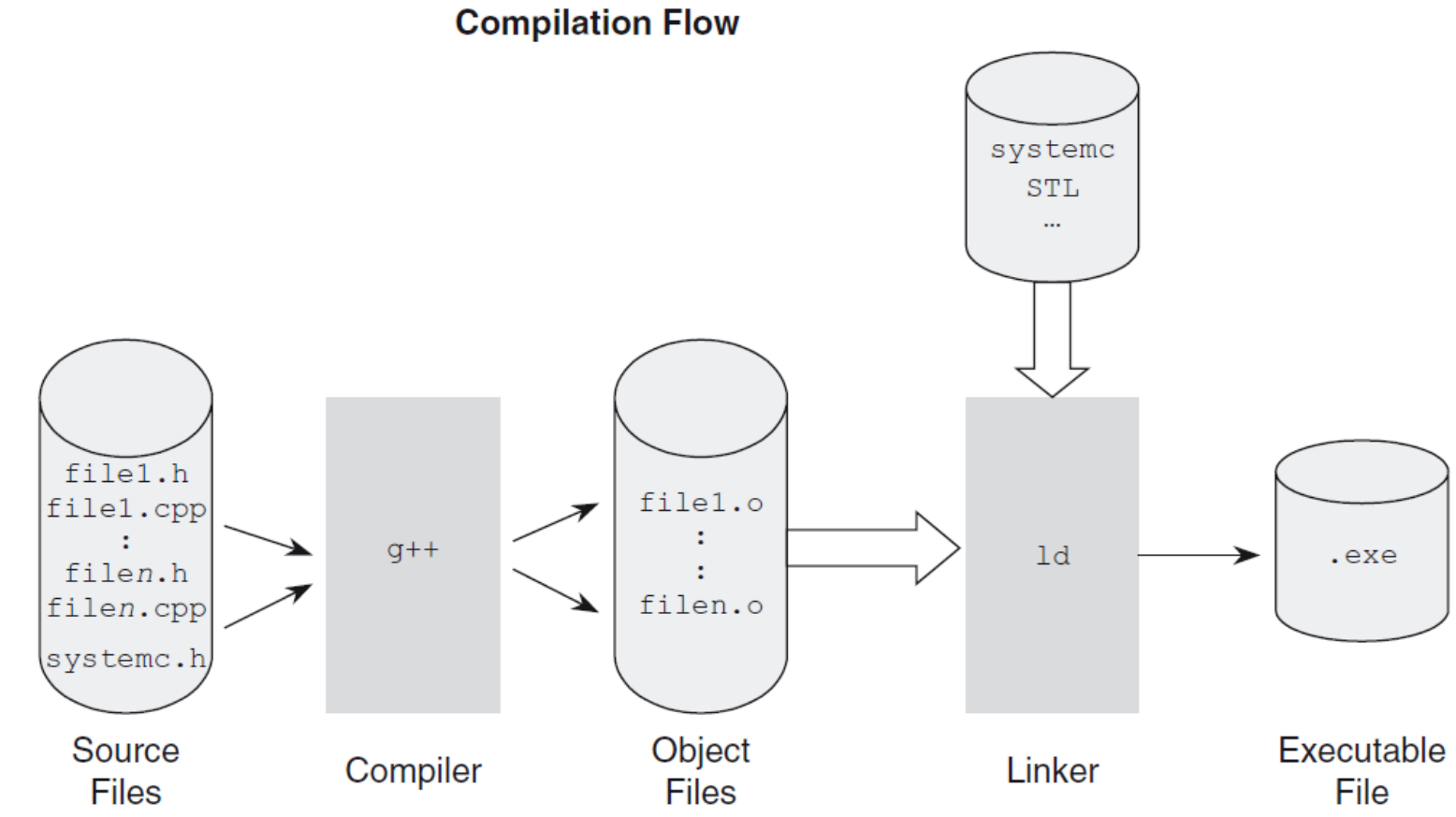


SystemC Overview

- SystemC addresses the modeling of both hardware and software using C++.
- The components of a SystemC environment include a:
  - SystemC-supported platform
  - SystemC-supported C++ compiler
  - SystemC library (downloaded and compiled for your C++ compiler)
  - Compiler command sequence make file or equivalent



SystemC Overview



SystemC Overview

```
#include <systemc>
SC_MODULE(Hello_SystemC) { // declare module class

  SC_CTOR(Hello_SystemC) { // create a constructor
    SC_THREAD(main_thread);// register the process
  }//end constructor

  void main_thread(void) {
    SC_REPORT_INFO(" Hello SystemC World!");
  }
};

int sc_main(int sc_argc, char* sc_argv[]) {

  //create an instance of the SystemC module
  Hello_SystemC HelloWorld_i("HelloWorld_i");

  sc_start(); // invoke the simulator

  return 0;
}
```

SystemC Overview

- The major hardware-oriented features implemented within SystemC include:
  - Time model
  - Hardware data types
  - Module hierarchy to manage structure and connectivity
  - Communications management between concurrent units of execution
  - Concurrency model

SystemC Overview

- **Time Model**
  - SystemC tracks time with 64 bits of resolution using a class known as **sc_time**.
  - Global time is advanced within the kernel.
  - For those models that require a clock, a class called **sc_clock** is provided.
- **Hardware types**
  - SystemC provides hardware-compatible data types that support explicit bit widths for both integral and fixed-point quantities.
  - Digital hardware requires non-binary representation such as tri-state and unknowns, which are provided by SystemC.

SystemC Overview

- **Hierarchy and Structure**
  - For modeling hardware hierarchy, SystemC uses ***the module entity*** interconnected to other modules using channels.
  - The hierarchy comes from the instantiation of module classes within other modules.
- **Communications management**
  - The SystemC channel provides a powerful mechanism for modeling communications.
  - Channels can represent complex communications schemes that eventually map to significant hardware such as the AMBA bus.
  - At the same time, channels may also represent very simple communications such as a wire or a FIFO (first-in first-out queue).

SystemC Overview

- **Concurrency**
  - SystemC provides a simulation kernel
  - Concurrency in a simulator is always an illusion.
  - The simulator uses a cooperative multitasking model.
  - The simulator merely provides a kernel to orchestrate the swapping of the various concurrent elements, called simulation processes.

SystemC Overview

- **Modules and Hierarchy**
  - Hardware designs typically contain hierarchy to reduce complexity.
  - Design components are encapsulated as “modules”.
  - Modules are classes that inherit from the ***sc_module*** base class.
  - Modules may contain other modules, processes, and channels and ports for connectivity.

SystemC Overview

- **SystemC Threads and Methods**
  - Simulation processes are simply member functions of **sc_module** classes that are “registered” with the simulation kernel.
  - They need no arguments and they return no value.
  - From a software perspective, processes are simply threads of execution.
  - From a hardware perspective, processes provide necessary modeling of independently timed circuits.
  - The **SC_METHOD** and **SC_THREAD** are the basic units of concurrent execution.
  - The simulation kernel invokes each of these processes.

SystemC Overview

- **Events, Sensitivity, and Notification**
  - Events, sensitivity, and notification are very important concepts for understanding the implementation of concurrency
  - Events are implemented with the SystemC ***sc_event*** and ***sc_event_queue*** classes
  - SystemC has two types of sensitivity: static and dynamic

SystemC Overview

- **SystemC Data Types**
  - Several hardware data types are provided in SystemC
  - These data types are implemented using templated classes and generous operator overloading
  - Non-binary hardware types are supported with four-state logic (0,1,X,Z) data types (e.g., sc_logic)

SystemC Overview

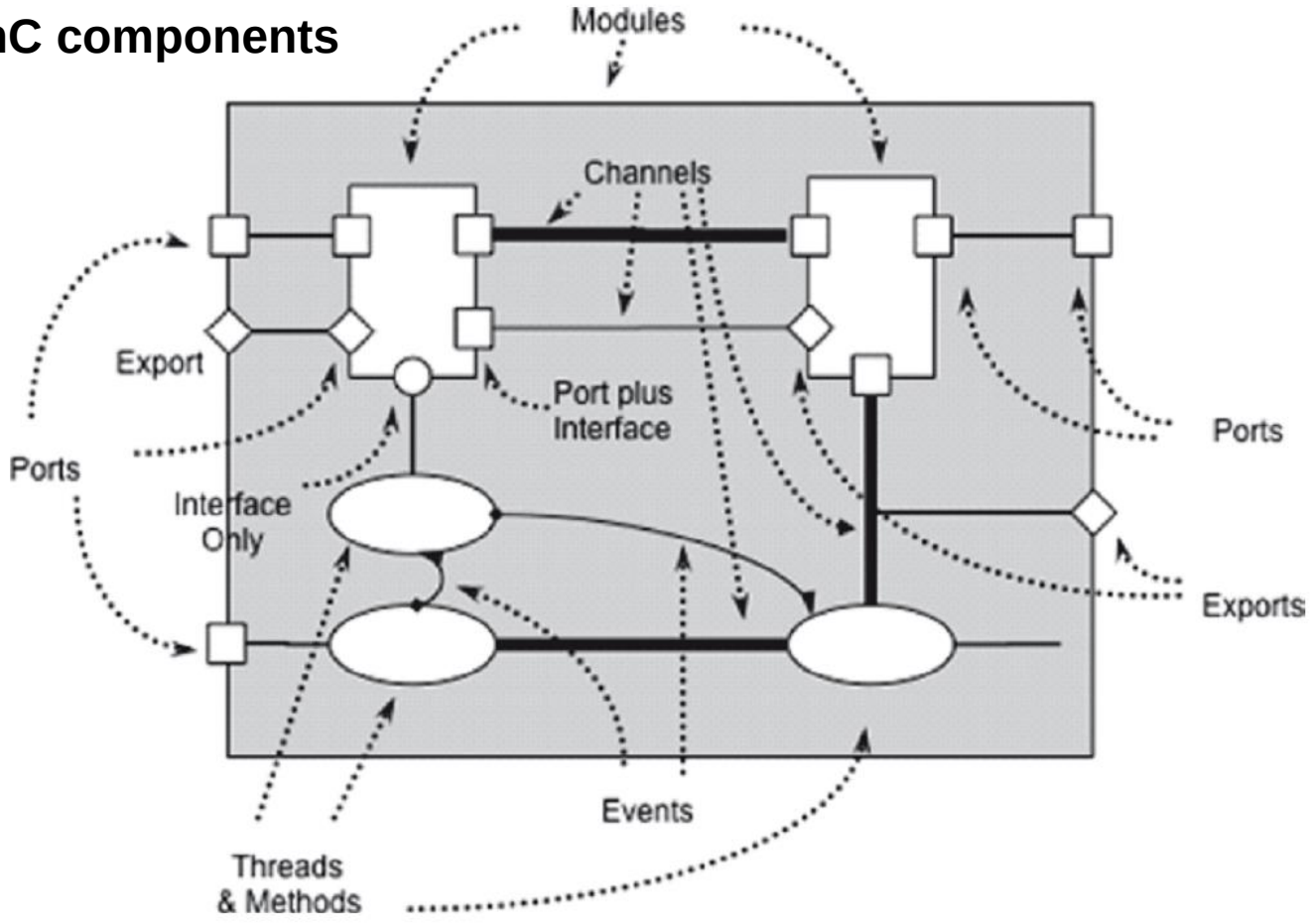
- **SystemC Data Types**
  - Several hardware data types are provided in SystemC
  - These data types are implemented using templated classes and generous operator overloading
  - Non-binary hardware types are supported with four-state logic (0,1,X,Z) data types (e.g., sc_logic)

SystemC Overview

- **Ports, Interfaces, and Channels**
  - Processes need to communicate with other processes both locally and in other modules.
  - In SystemC, processes communicate using channels or events
  - Modules may interconnect using channels, and connect via ports
  - Module interconnection happens programmatically in SystemC during the elaboration phase

SystemC Overview

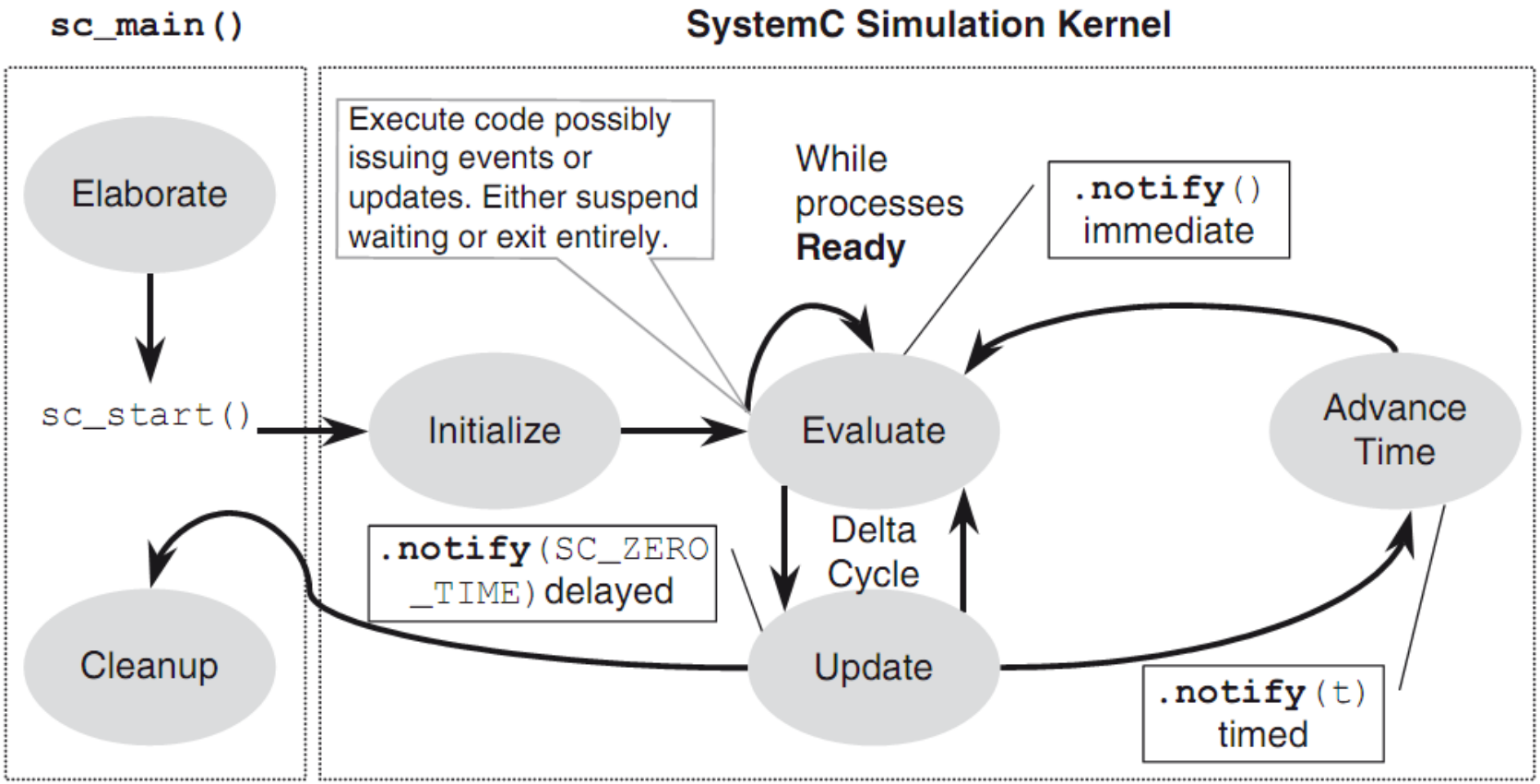
SystemC components



SystemC Overview

- **SystemC Simulation Kernel**
  - The SystemC simulator has two major phases of operation: ***elaboration*** and ***execution***.
  - A third, often minor, phase occurs at the end of execution; this phase could be characterized as ***post-processing*** or ***cleanup***.
  - Execution of statements prior to the ***sc_start()*** function call are known as the elaboration phase.
    - This phase is characterized by the initialization of data structures, the establishment of connectivity, and the preparation for the second phase, execution.
  - The execution phase hands control to the SystemC simulation kernel:
    - orchestrates the execution of processes to create an illusion of concurrency.

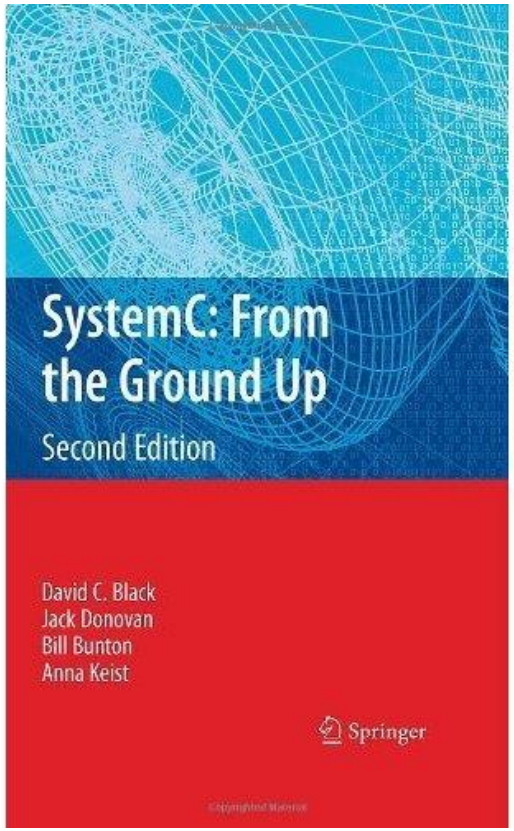
SystemC Overview



Example

- **Model and simulate a simple logic gate.**
  - **Hints (for Visual Studio) – project properties**
    - **C/C++, General, Additional Include Directories**
      - **$(SYSTEMC)\..\src**
    - **C/C++, Code generation, Runtime Library:**
      - **Multi-threaded Debug (/MTd)**
    - **C/C++, Language, Enable Run-Time Type Information:**
      - **Yes (/GR)**
    - **C/C++, Command Line, Additional Options:**
      - **/vmg**
    - **Linker, General, Additional Library Directories**
      - **$(SYSTEMC)\SystemC\Debug**
    - **Linker, Input, Additional Dependencies:**
      - **SystemC.lib**

Bibliografie



- David C. Black, Jack Donovan, Bill Bunton, Anna Keist, ***SystemC:From the Ground Up,*** Springer Science+Business Media, LLC 2010
  - *"*The authors designed this book primarily for the student or engineer new to SystemC. This book's structure is best appreciated by reading sequentially from beginning to end."