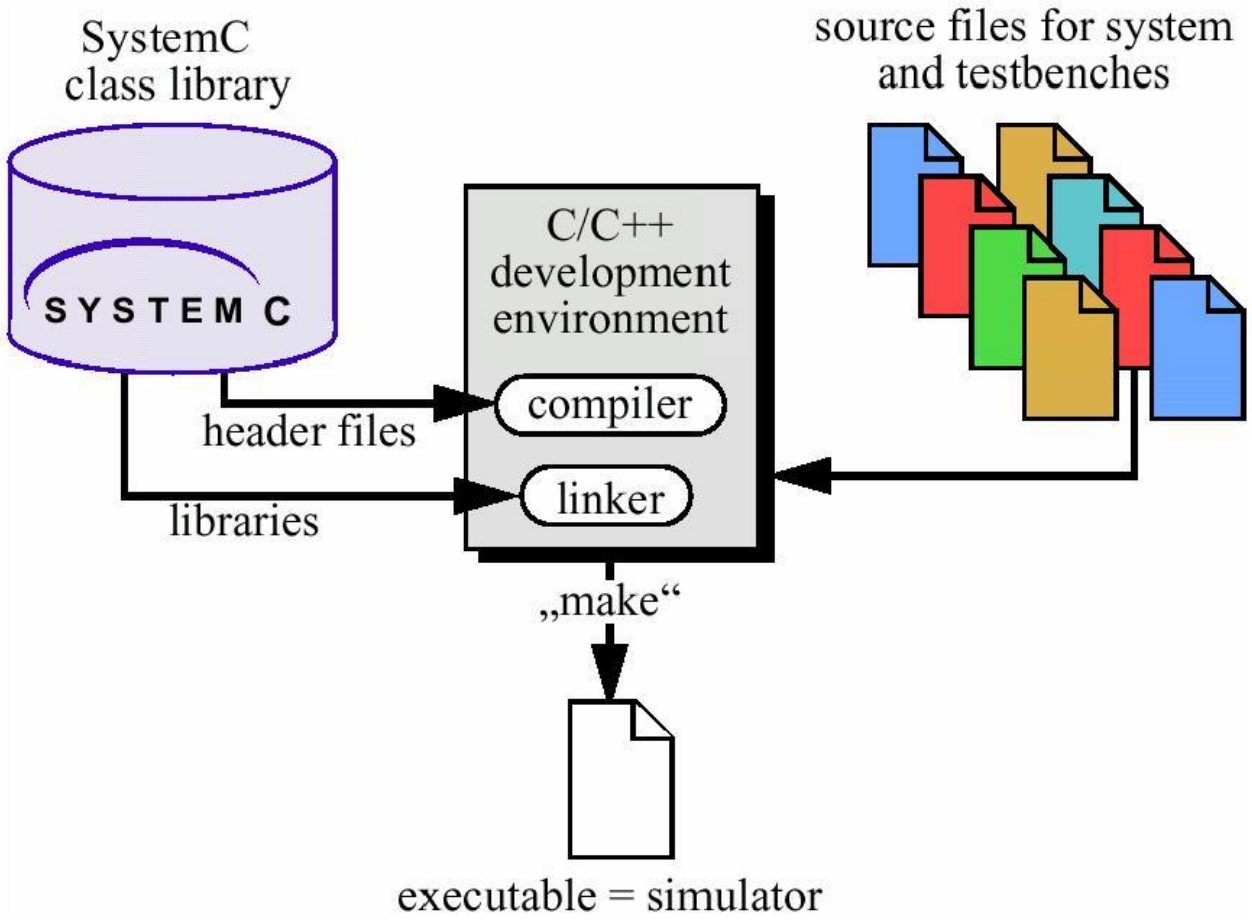
TUHH
Technische Universität Hamburg

SystemC Introduction

Prof. Dr.-Ing. Sohan Lal

Massively Parallel Systems Group
School of Electrical Engineering, Computer Science and Mathematics
Technische Universität Hamburg

SystemC Design Flow



Using Software to Simulate Hardware Behavior

TUHH
Technische Universität Hamburg

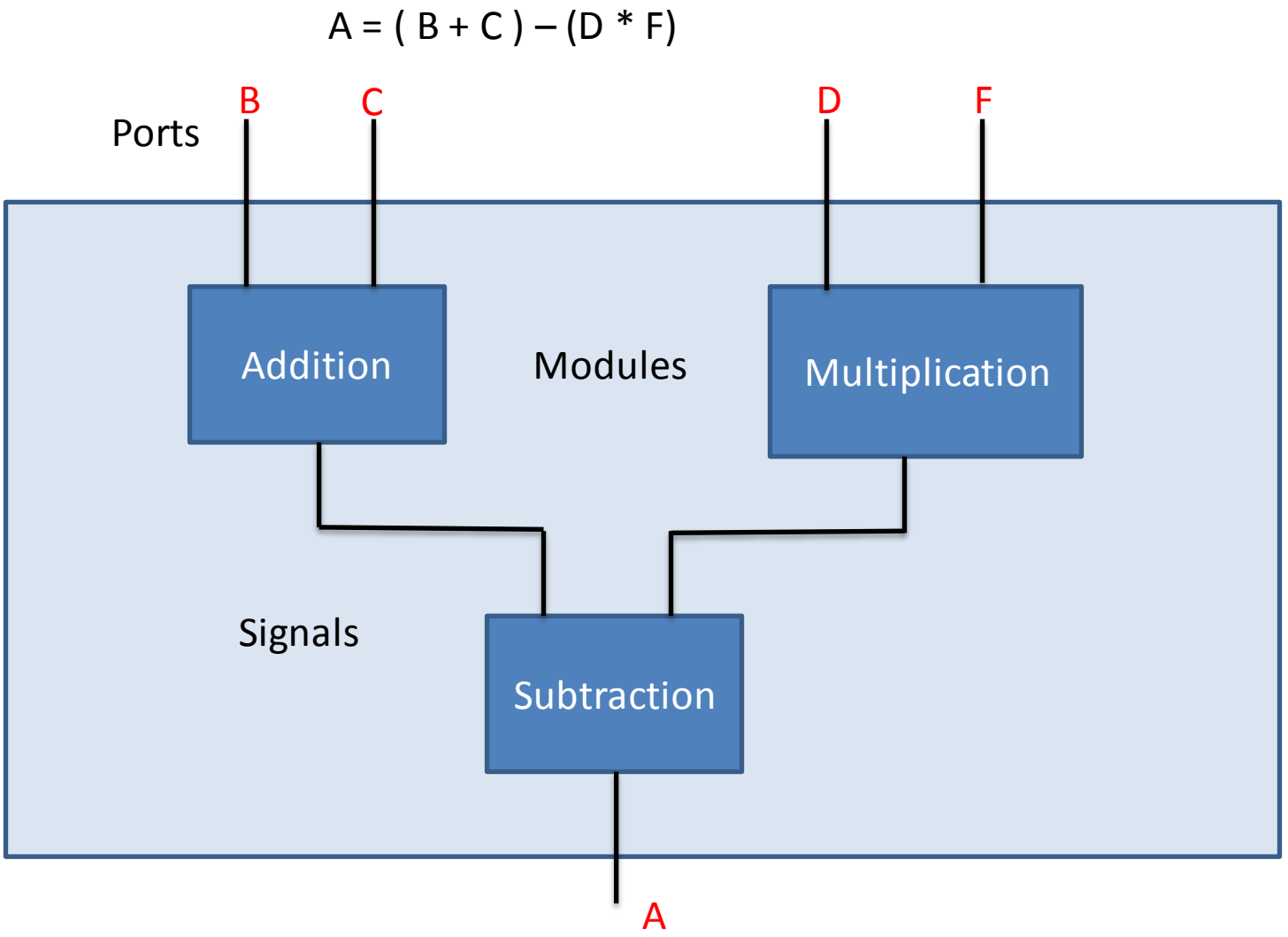
- Library of C++ that supports hardware modeling, design and synthesizable code
  - Hardware notion of time → clocks
  - Hardware communication → signals
  - Port mapping
  - Data types used in hardware (bit-vectors, multi-valued logic types)
  - Concurrency → different operations in parallel

SystemC Key Components

TUHH
Technische Universität Hamburg

- Modules (corresponds to a C++ class)
  - Functionality → processes which are the methods in C++
  - Hierarchy → when it contains sub-modules
- Ports
  - External interface of the modules
- Signals
  - Local to module and convey information between different modules
- Processes
  - Basis of concurrent execution
  - Include functionalities and have sensitive lists
- Clocks

Example SystemC Model



Example: Half Adder

```
#include "systemc.h"

SC_MODULE(half_adder) {
  sc_in<bool>a, b;
  sc_out<bool>sum, carry;

  void proc_half_adder();
  SC_CTOR(half_adder) {
    SC_METHOD (proc_half_adder);
    sensitive << a << b;
  }
};

void half_adder::proc_half_adder() {
  sum = a ^ b;
  carry = a & b;
}
```

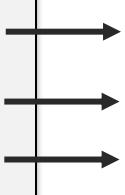
Basic Block: Module

```
SC_MODULE (module_name) {
        input/output declaration
        internal variable
        constructor (computation block)
      };
```

- Input : sc_in<type> var1, …;
- Output: sc_out<type> var2, …;
- Type
  - C++ primitive type : int, float, char, …
  - hardware type : sc_int, sc_uint, …
  - user defined type

SystemC Constructor

```
SC_CTOR (module_name) {
  SC_METHOD (function name);
  sensitive  << a << b << c;
             …
}
```



C++ constructor

Computation function name

Sensitivity list

Example: 4-bit Counter (I)

```
#include "systemc.h"

SC_MODULE (first_counter) {
  sc_in_clk     clock ; //Clock input of the design
  sc_in<bool>   reset ; //active high, synchronous Reset input
  sc_in<bool>   enable; //Active high enable signal for counter
  sc_out<sc_uint<4> > counter_out; //4 bit vector output of the
  sc_uint<4>  count;                //counter
  void incr_count () {
    if (reset.read() == 1) {
      count =  0;
      counter_out.write(count);
    } else if (enable.read() == 1) {
      count = count + 1;
      counter_out.write(count);
      cout<<"@" << sc_time_stamp() <<" :: Incremented Counter "
        <<counter_out.read()<<endl;
    }
  } // End of function incr_count
```

Example: 4-bit Counter (II)

```
SC_CTOR(first_counter) {
    cout<<"Executing new"<<endl;
    SC_METHOD(incr_count);
    sensitive << reset;
    sensitive << clock.pos();
  } // End of Constructor
}; // End of Module counter
```

Demo

TUHH
Technische Universität Hamburg