EEL 4783: HDL in Digital System Design

Lecture: SystemC Language and Its Usage

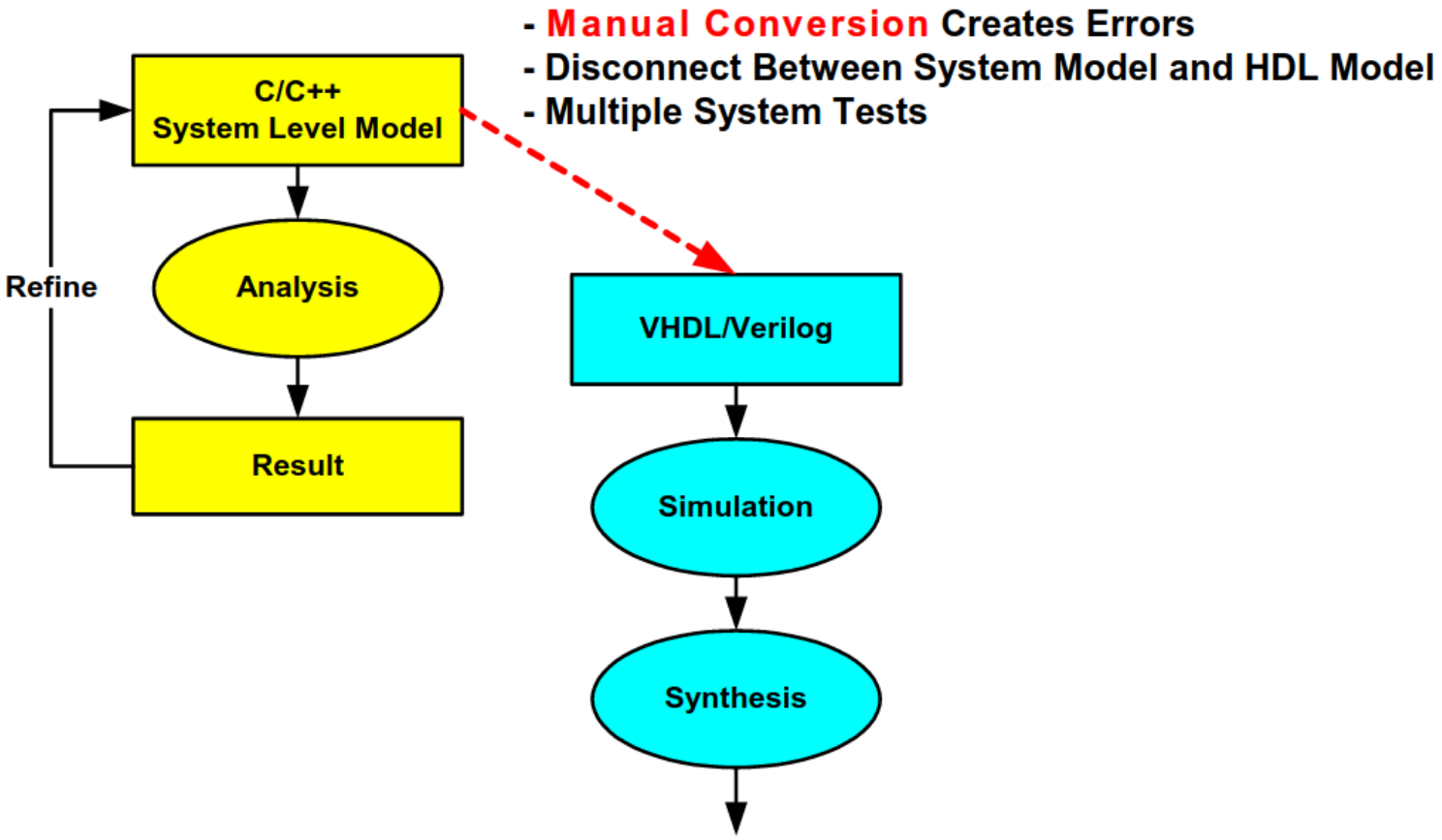
Part 2

Prof. Mingjie Lin

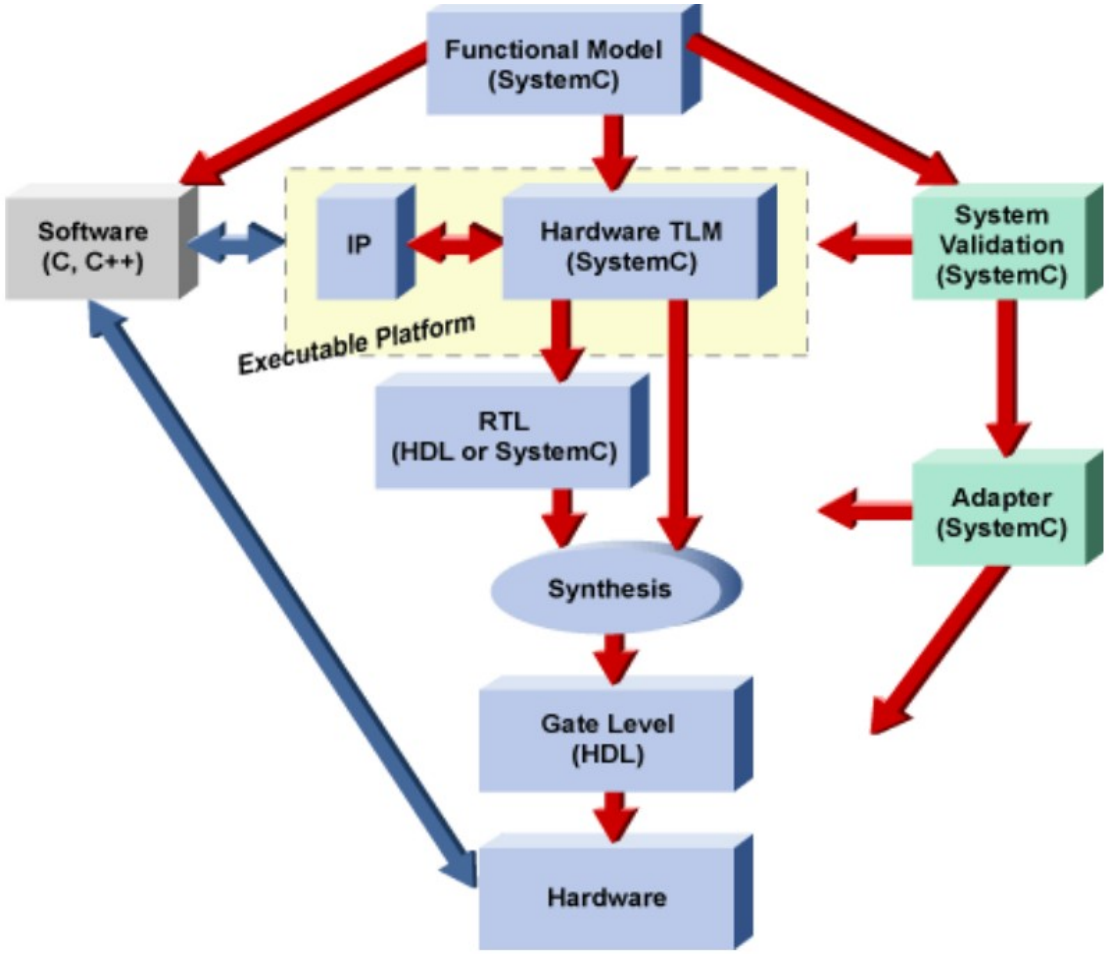
System Design Methodology

- Current
  - Manual Conversionfrom C to HDL Creates Errors
  - Disconnect Between System Model and HDL Model
  - Multiple System Tests
- SystemC (Executable-Specification)
  - Refinement Methodology
  - Written in a Single Language

Current Methodology



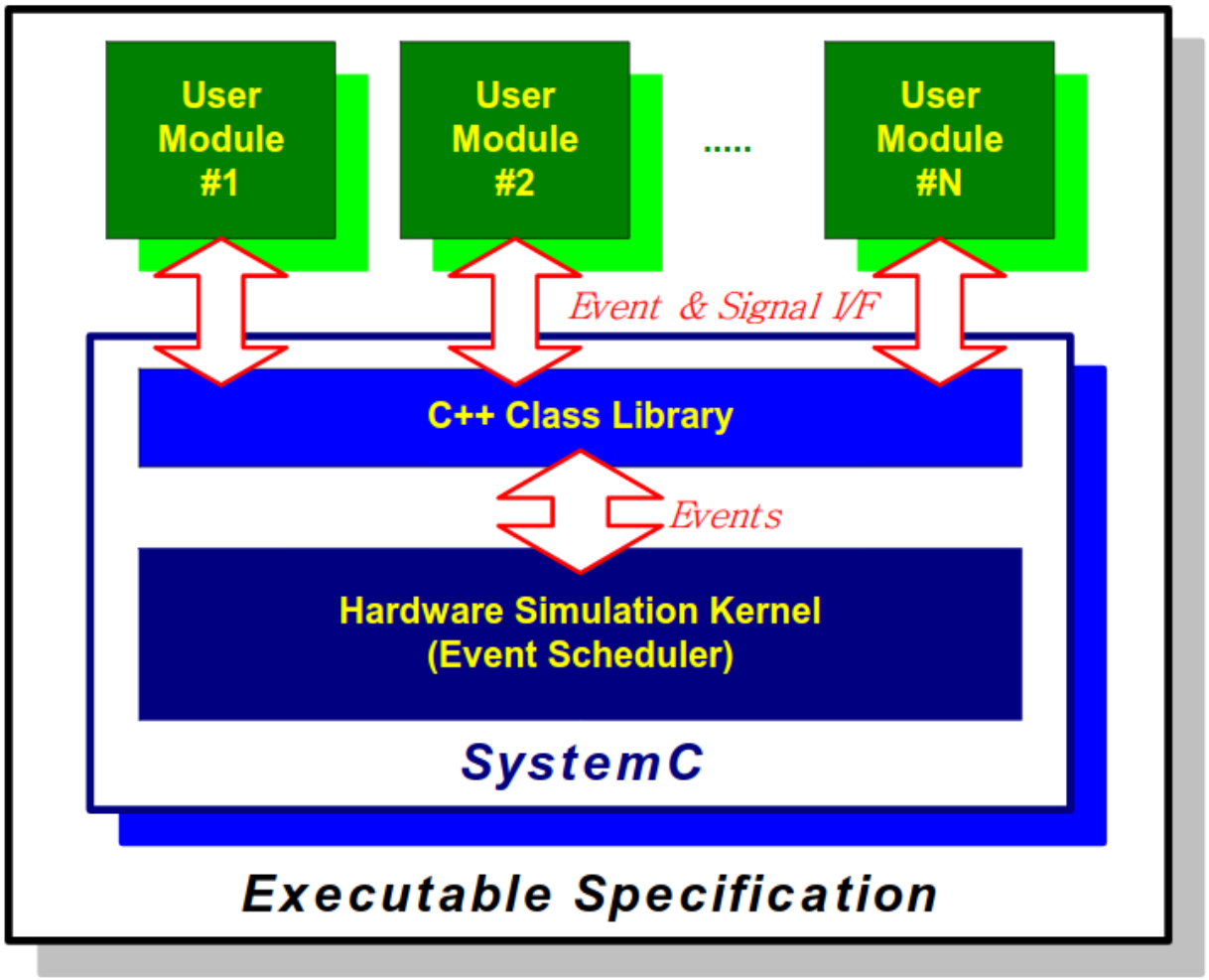
SystemC Methodology



Using Executable Specifications

- Ensure COMPLETENESS of Specification
  - “Create a program that Behave the same way as the system”
- UNAMBIGUOUS Interpretation of the Specification
- Validate system functionality before implementation
- Create early model and Validate system performance
- Refine and Test the implementation of the Specification

SystemC and User Module



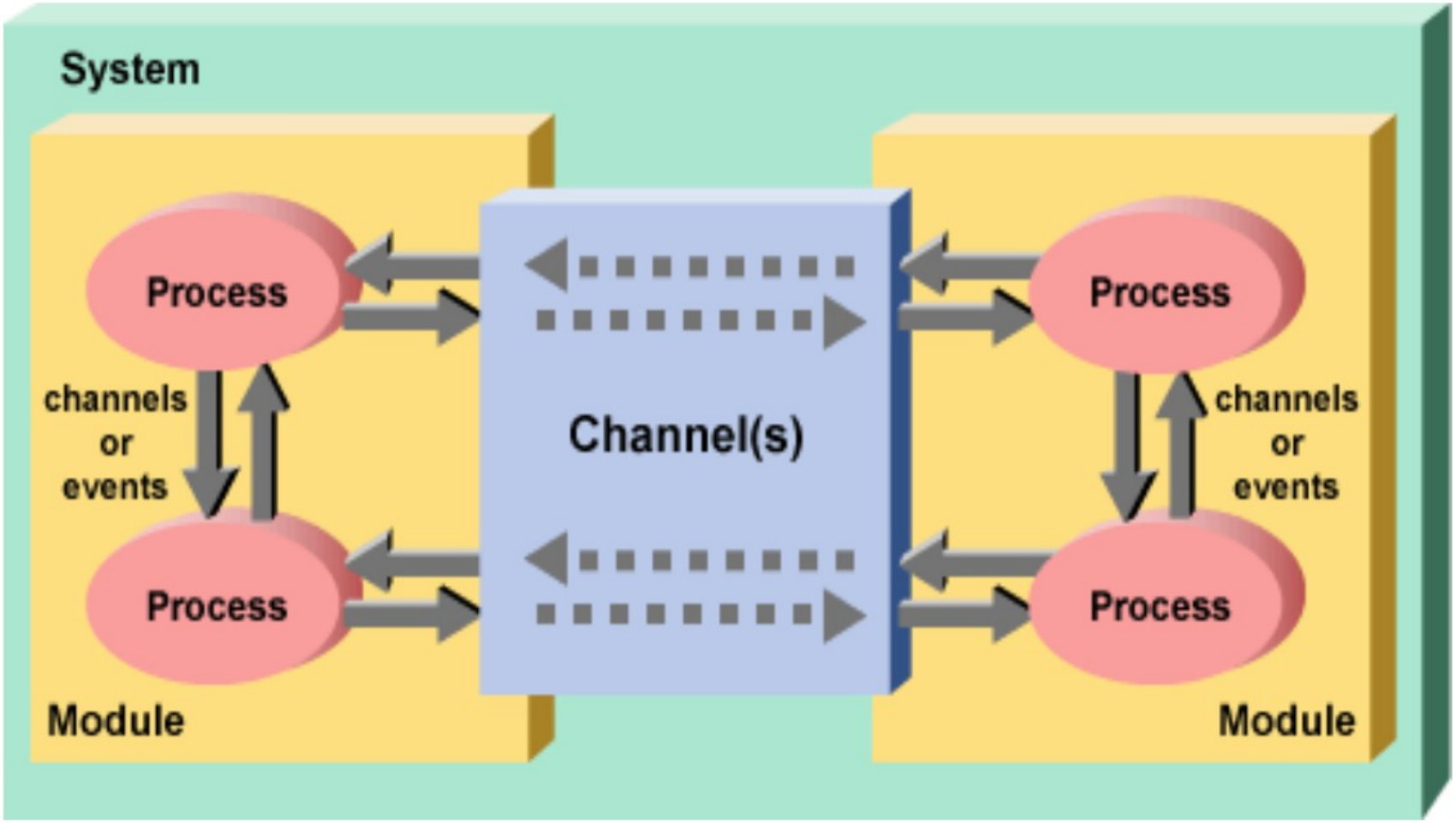
SystemC Highlights (1)

- SystemC2.0 introduces general-purpose
  - Events
    - Flexible, low-level synchronization primitive
    - Used to construct other forms of synchronization
  - Channels
    - A container class for communication and synchronization
    - They implement one or more interfaces
  - Interfaces
    - Specify a set of access methods to the channel
- Other comm& sync models can be built based on the above primitives
  - Examples
    - HW-signals, queues (FIFO, LIFO, message queues, etc) semaphores, memories and busses (both at RTL and transaction-based models)

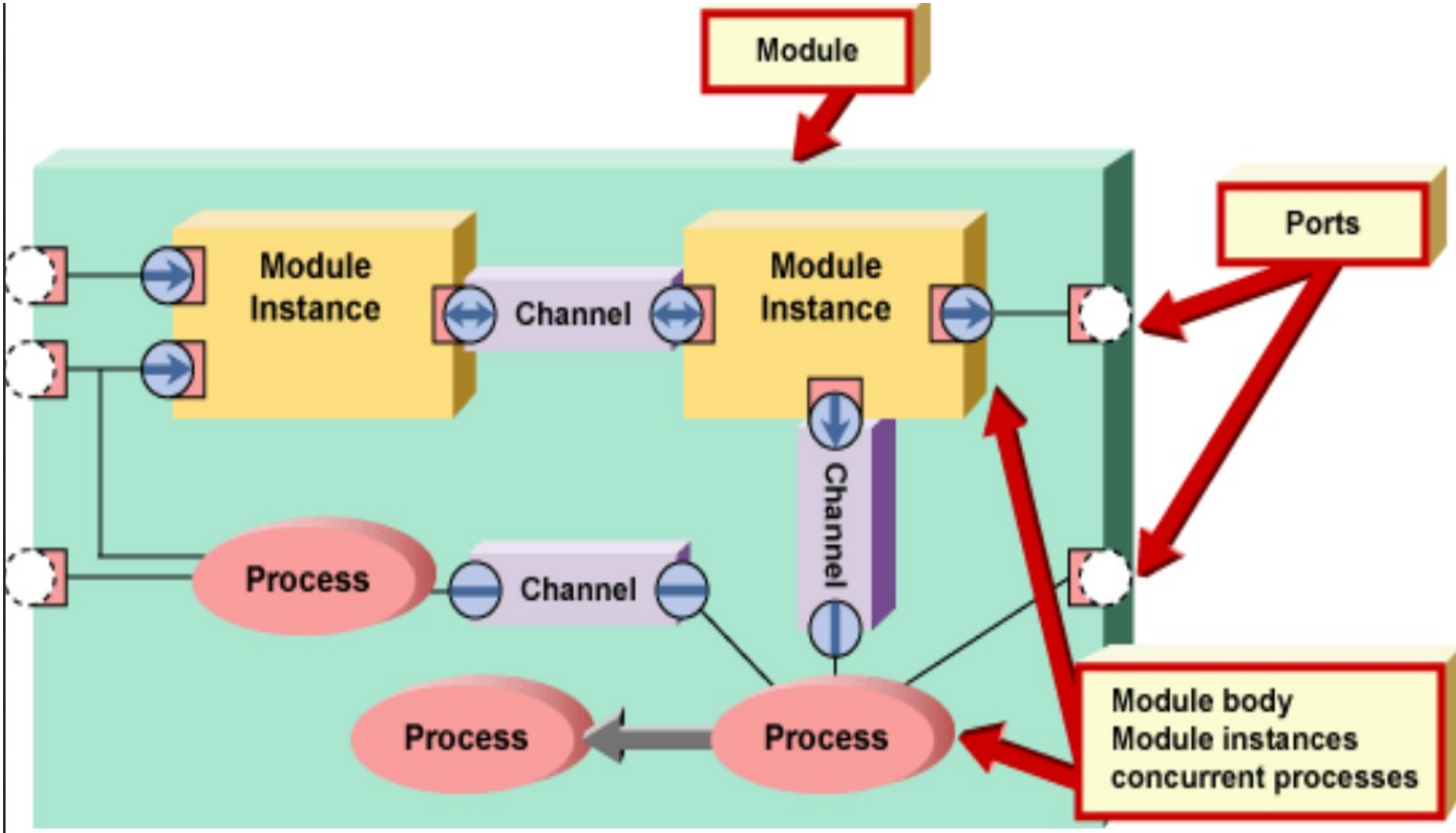
SystemC Highlights (2)

- Support Hardware-Software Co-Design
- All constructs are in a C++ environment
  - Modules
    - Container class includes hierarchical Modules and Processes
  - Processes
    - Describe functionality
    - Almost all SLDL have been developed based on some underlying model of network of processes
  - Ports
    - Single-directional(in, out), Bi-directional mode

A system in SystemC



A system in SystemC



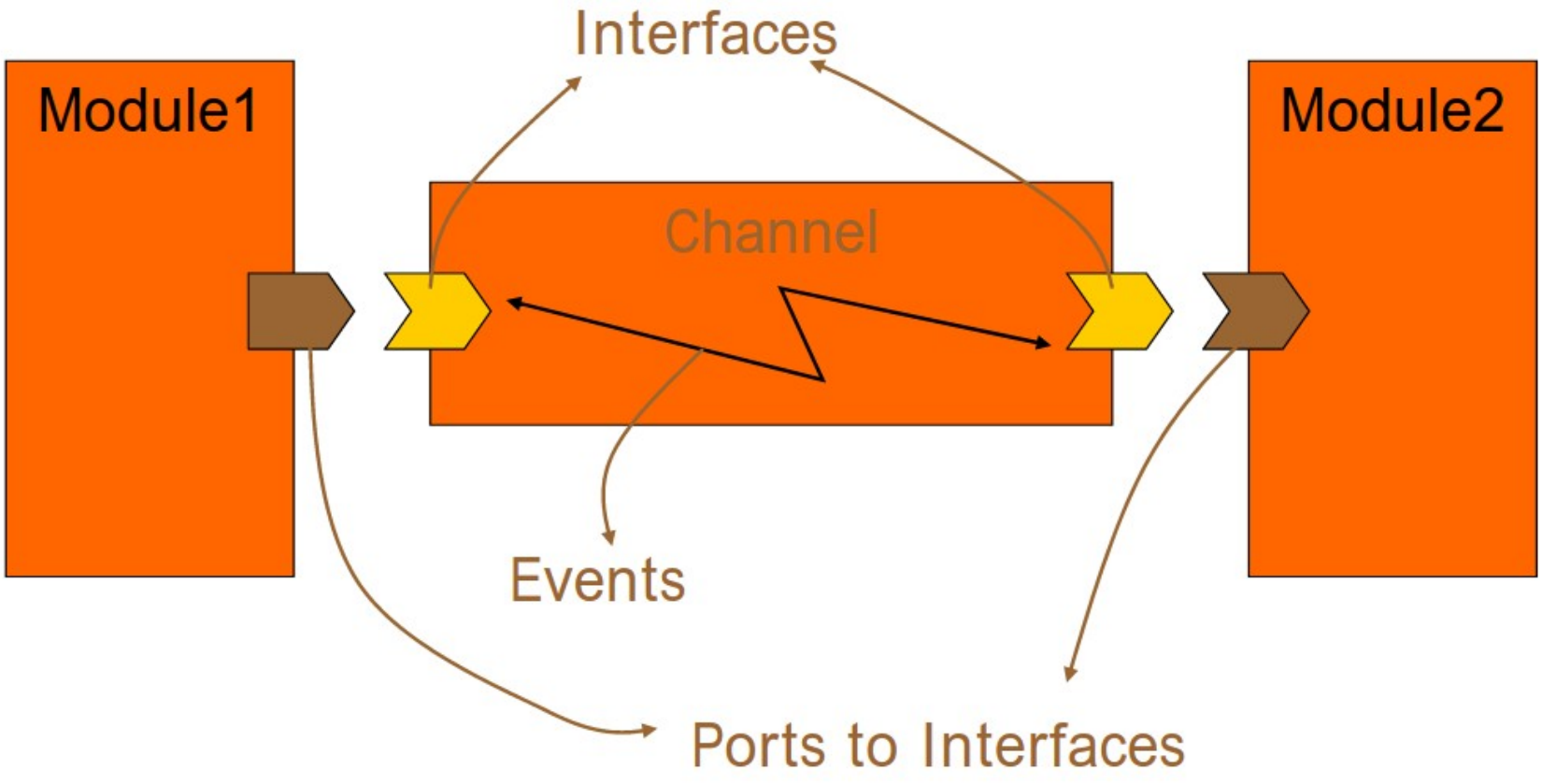
SystemC Highlights (3)

- Constructs in a C++ environment (continued)
  - Clocks
    - Special signal, Timekeeper of simulation and Multiple clocks, with arbitrary phase relationship
  - Event Driven simulation
    - High-SpeedEventDriven simulation kernel
  - Multiple abstraction levels
    - Untimedfrom high-level functional model to detailed clock cycle accuracy RTL model
  - Communication Protocols
  - Debugging Supports
    - Run-Time error check
  - Waveform Tracing
    - Supports VCD, WIF, ISBD

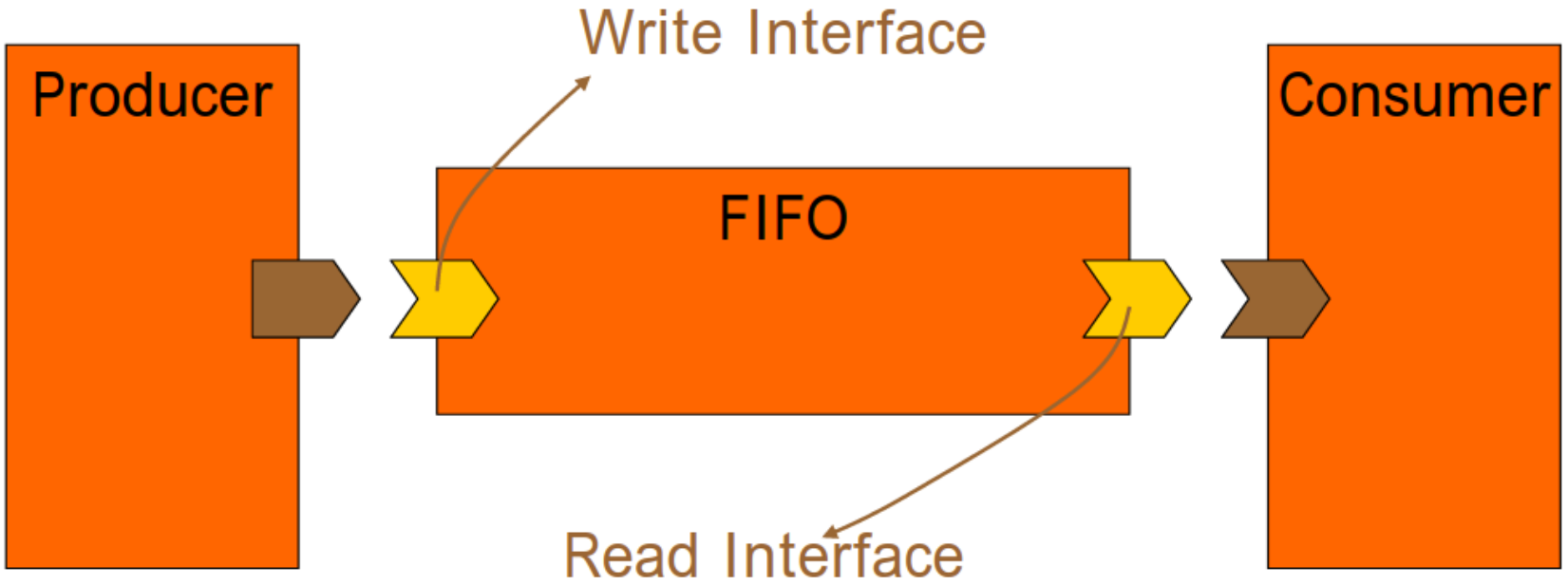
Data Types

- SystemCsupports
  - Native C/C++ Types
  - SystemCTypes
- SystemCTypes
  - Data type for system modeling
  - 2 value (‘0’,’1’)logic/logic vector
  - 4 value (‘0’,’1’,’Z’,’X’)logic/logic vector
  - Arbitrary sized integer (Signed/Unsigned)
  - Fixed Point types (Templated/Untemplated)

Communication and Synchronization (cont'd)



A Communication Modeling Example: FIFO



FIFO Example:Declaration of Interfaces

```
class write_if : public sc_interface
{
   public:
        virtual void write(char) = 0;
        virtual void reset() = 0;
};

class read_if : public sc_interface
{
   public:
        virtual void read(char&) = 0;
        virtual int num_available() = 0;
};
```

Declaration of FIFOchannel

```
class fifo: public sc_channel,
   public write_if,
   public read_if
{
   private:
        enum e {max_elements=10};
        char data[max_elements];
        int num_elements, first;
        sc_event write_event,
                 read_event;
        bool fifo_empty() {…};
        bool fifo_full() {…};

   public:
        fifo() : num_elements(0),
                 first(0);
```

```
void write(char c) {
   if (fifo_full())
        wait(read_event);
   data[ <you calculate> ] = c;
   ++num_elements;
   write_event.notify();
}

void read(char &c) {
   if (fifo_empty())
        wait(write_event);
   c = data[first];
   --num_elements;
   first =  …;
   read_event.notify();
}
```

Declaration of FIFO channel(cont'd)

```
    void reset() {
        num_elements = first = 0;
    }

    int num_available() {
        return num_elements;
    }
}; // end of class declarations
```

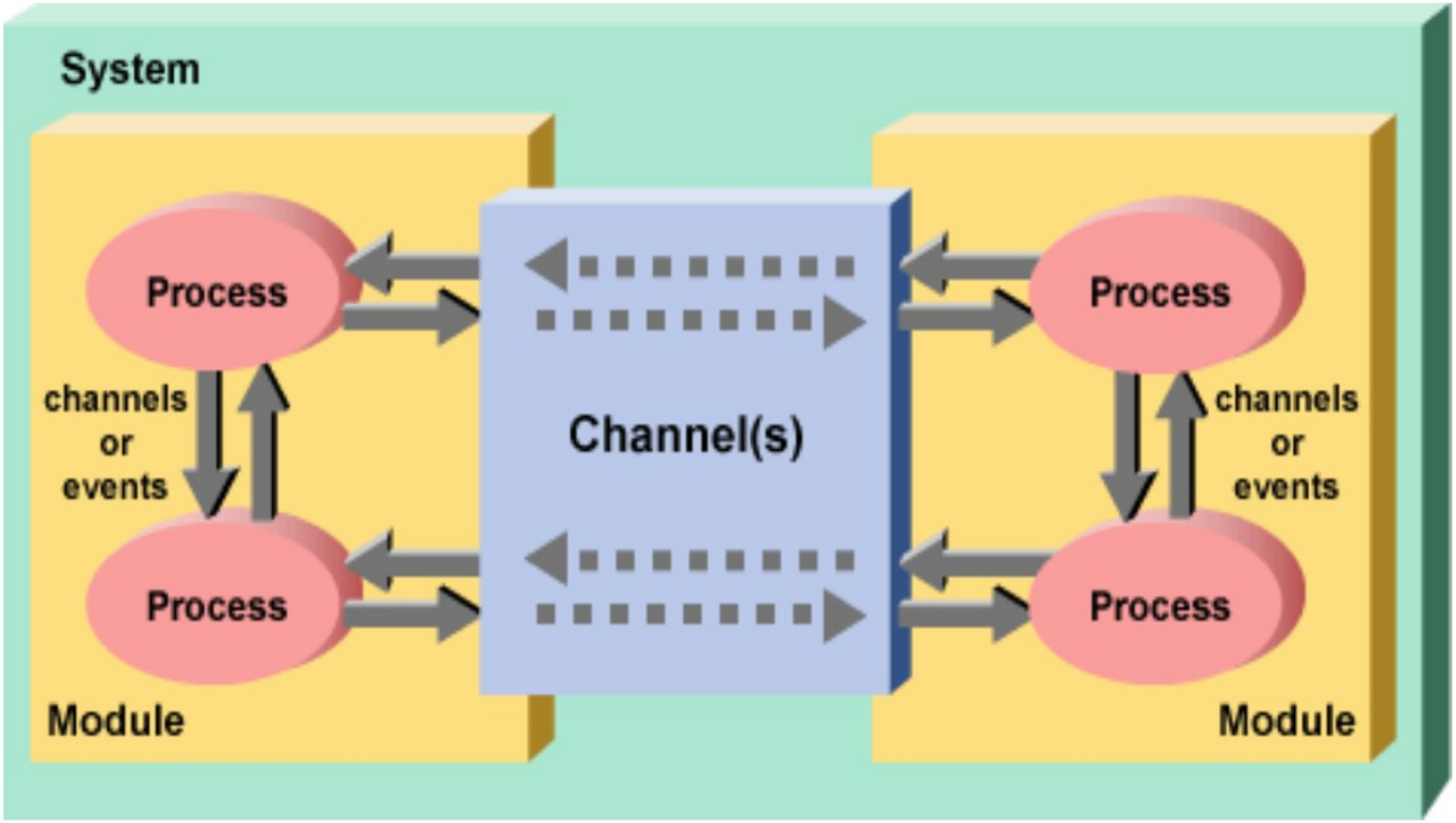
FIFO Example (cont'd)

- Any channel must
  - be derived from sc_channelclass
  - be derived from one (or more) classes derived from sc_interface
  - provide implementations for all pure virtual functions defined in its parent interfaces
- Note the following wait() callv
  - wait(sc_event) => dynamic sensitivity
  - wait(time)
  - wait(time_out, sc_event)
- Events
  - are the fundamental synchronization primitive
  - have no type, no value
  - always cause sensitive processes to be resumed
  - can be specified to occur:
    - immediately/ one delta-step later/ some specific time later

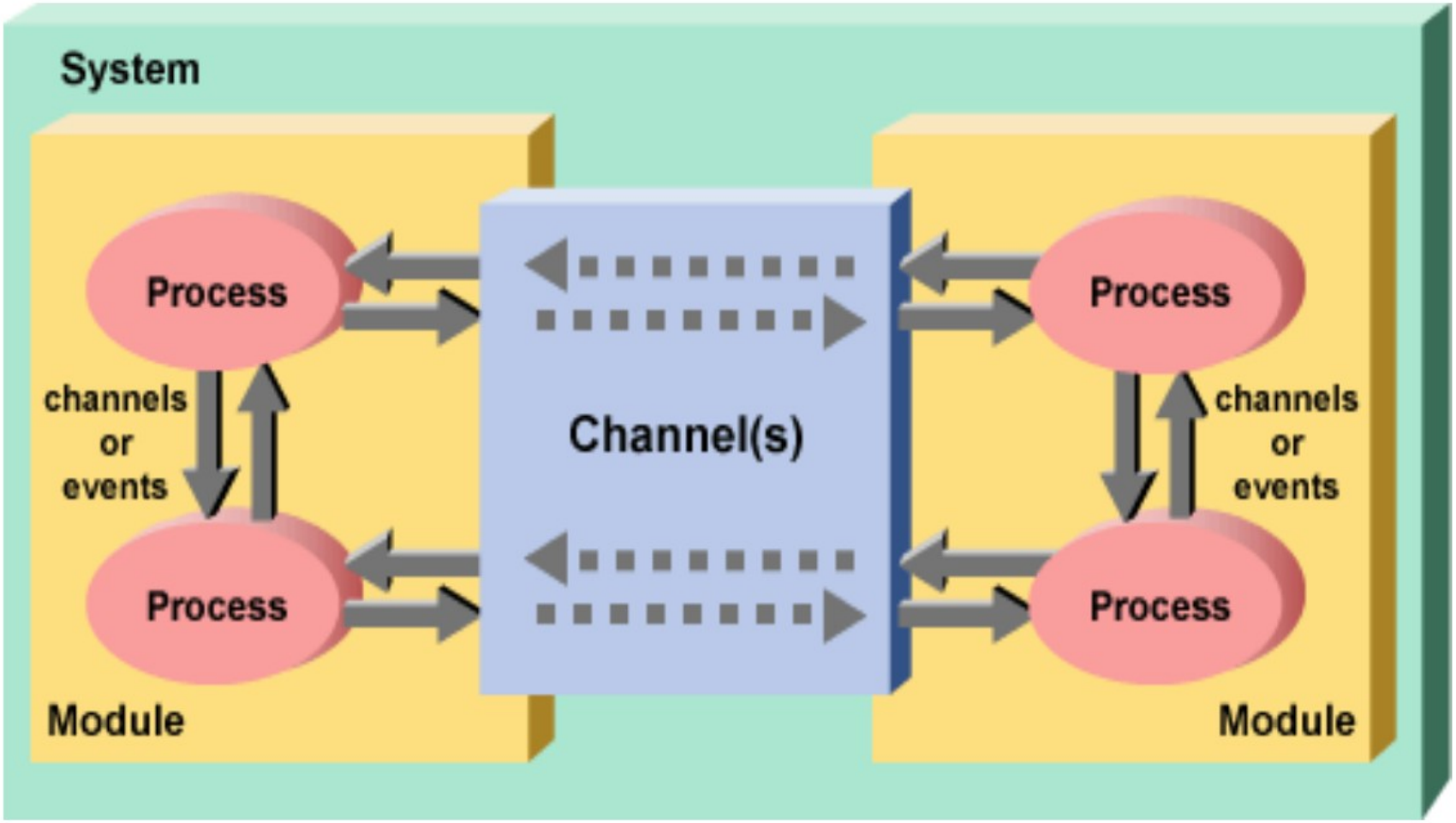
SystemC Highlights (2)

- Support Hardware-Software Co-Design
- All constructs are in a C++ environment
  - Modules
    - Container class includes hierarchical Modules and Processes
  - Processes
    - Describe functionality
    - Almost all SLDL have been developed based on some underlying model of network of processes
  - Ports
    - Single-directional(in, out), Bi-directional mode

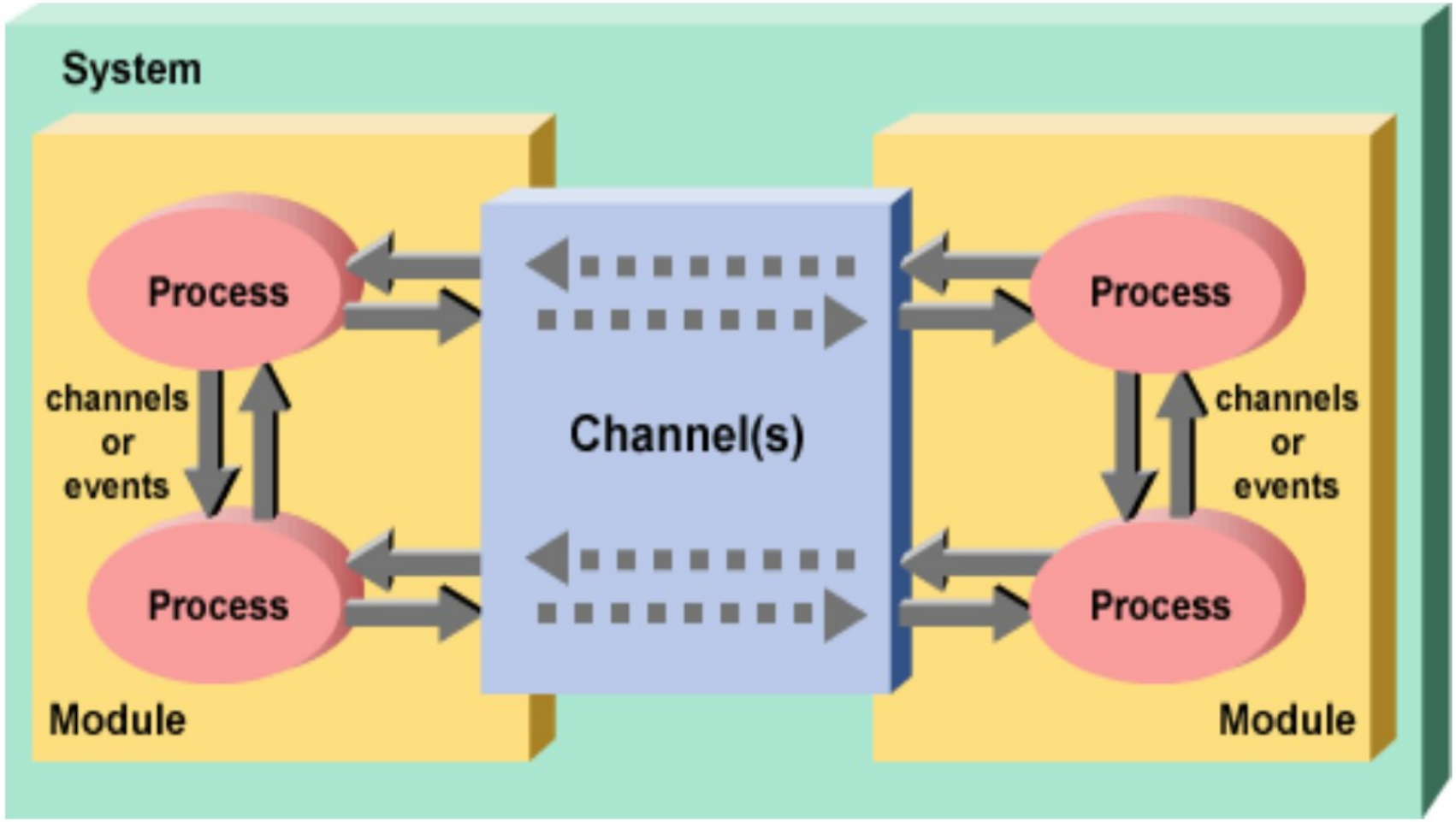
A system in SystemC



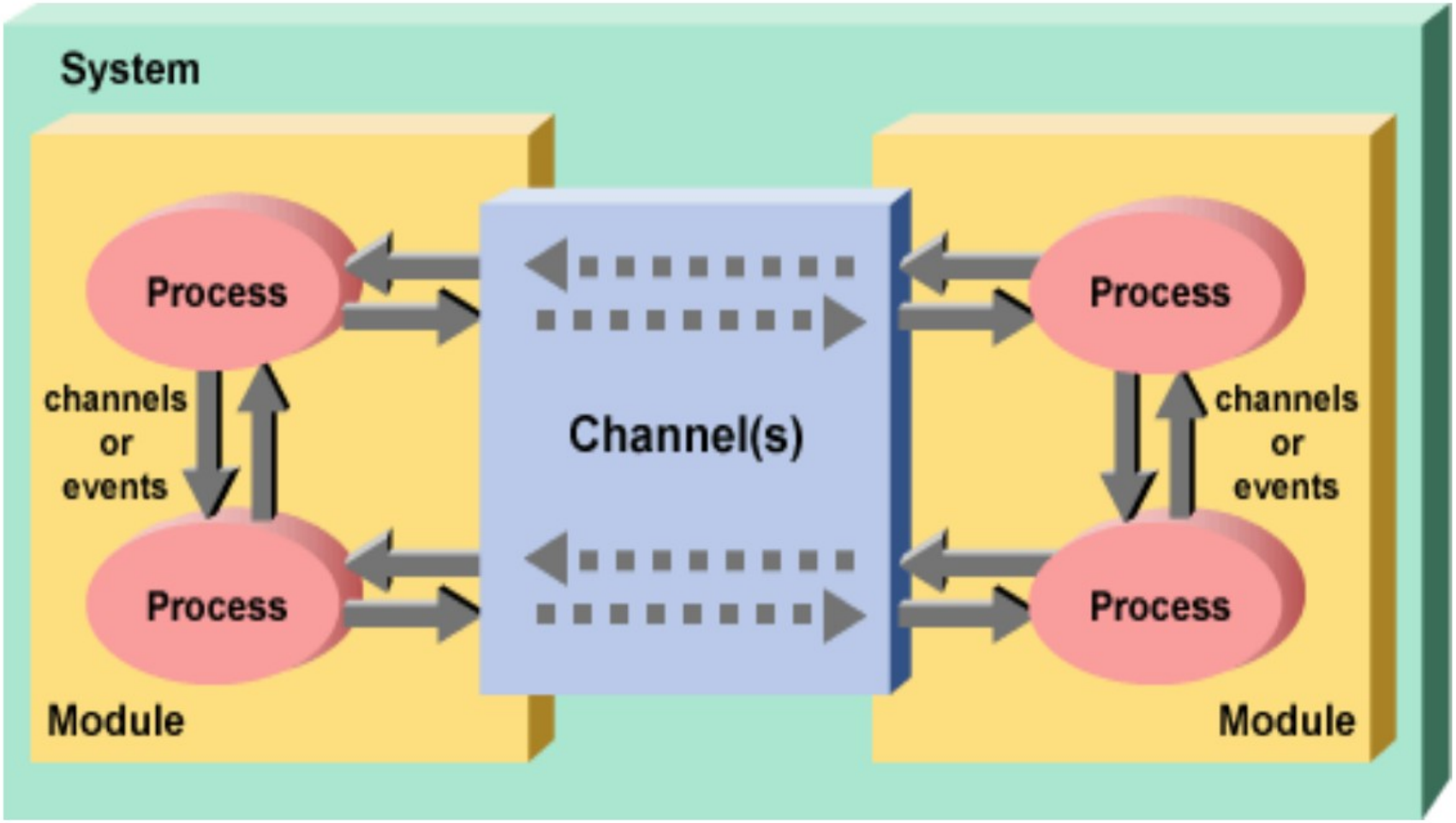
A system in SystemC



A system in SystemC



A system in SystemC



Final issues

- Come by my office hours (right after class)
- Any questions or concerns?