The Definitive Guide to SystemC

David C Black, Doulos

Presented to UT Austin class: EE382N.23
Embedded System Design and Modeling
UT Austin

UTexas Austin EE382N.23 Fall 2015

The SystemC Language

- Introduction to SystemC
  - Overview and background
  - Central concepts
  - The SystemC World
  - Use cases and benefits
- Core Concepts and Syntax
- Bus Modeling

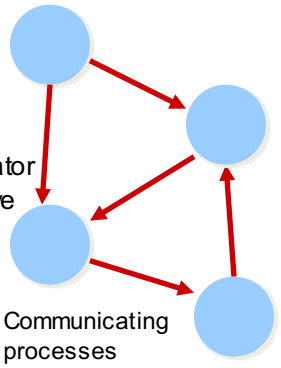
2

What is SystemC?

DOULOS

System-level modeling language
- Network of communicating processes (c.f. HDL)
- Supports heterogeneous models-of-computation
- Models hardware and software

C++ class library
- Open source proof-of-concept simulator
- Owned by Accellera Systems Initiative

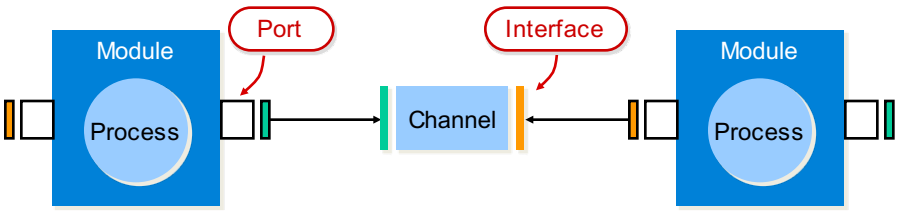


3

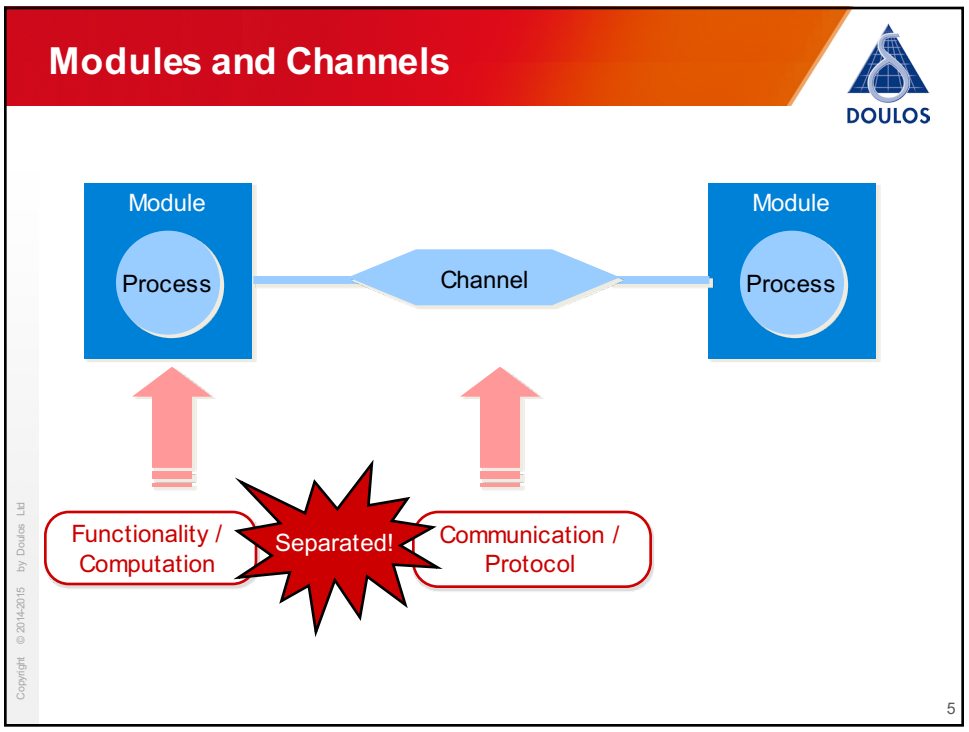
Features of SystemC

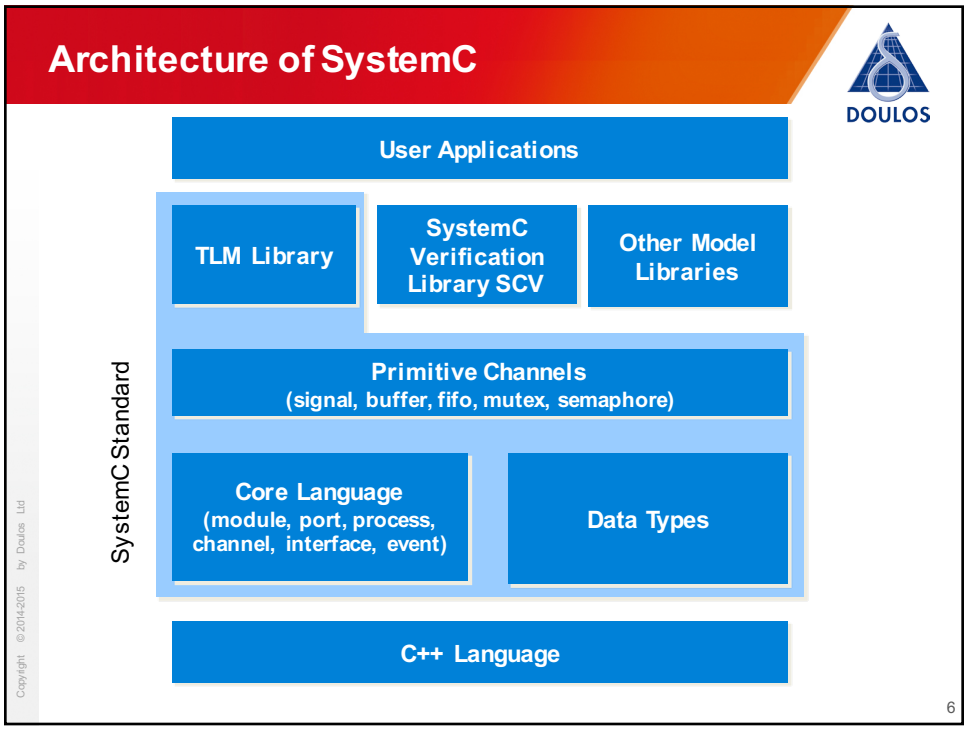
DOULOS

- Modules (structure)
- Ports (structure)
- Processes (computation, concurrency)
- Channels (communication)
- Interfaces (communication refinement)
- Events (time, scheduling, synchronization)
- Data types (hardware, fixed point)



4





Accellera Systems Initiative

Formed from Open-SystemC Initiative (OSCI) and Accellera
Website http://www.accellera.org/community/systemc

National Users Groups
- http://www-ti.informatik.uni-tuebingen.de/~systemc
- http://www.nascug.org
- http://www.iscug.in
- Others...

Language Working Group LWG
- IEEE 1666™ -2005
  - SystemC 2.2 released Mar 2006 (IEEE 1666 compliant)
- IEEE 1666™ -2011
  - SystemC 2.3.1 released Apr 2014 (IEEE 1666 compliant)
  - Includes TLM-2

7

Other Working Groups

Verification Working Group VWG
- SystemC Verification (SCV) library released 2003

Synthesis Working Group SWG
- Synthesisable subset of SystemC

Transaction-Level Modeling Working Group TLMWG
- TLM-1.0 released May 2005
- TLM-2.0 released June 2008
- TLM-2.0 part of IEEE 1666-2011

Analog and Mixed Signal Working Group AMSWG

Control, Configuration & Inspection Working Group CCIWG

8

What can you do with SystemC?

DOULOS

Discrete Event Simulation (events, time)
- Register Transfer Level (delta delays, bus resolution)
- Transaction Level (communication using function calls)
- Kahn Process Networks (infinite fifos, reads block when empty)
- Dataflow (input-execution-output stages)
- CSP (rendezvous, blocking reads & writes)

Continuous Time or Frequency Domain (Analog)
SystemC AMS

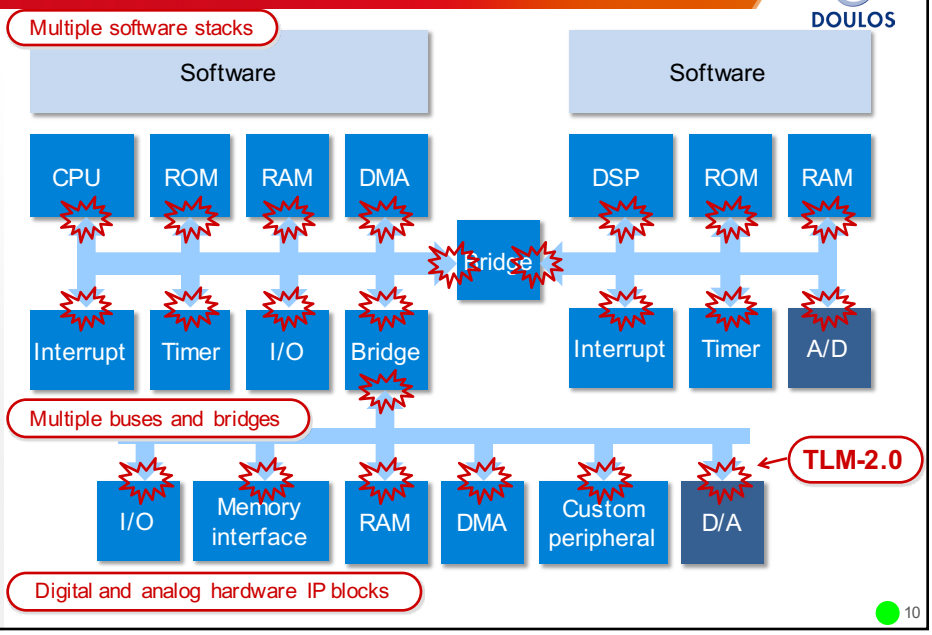
NOT gate level

NOT abstract RTOS modeling (process scheduling, priorities, pre-emption)
(originally planned as version 3)

9

Typical Use Case: Virtual Platform

DOULOS



10

What is SystemC Used For?

DOULOS

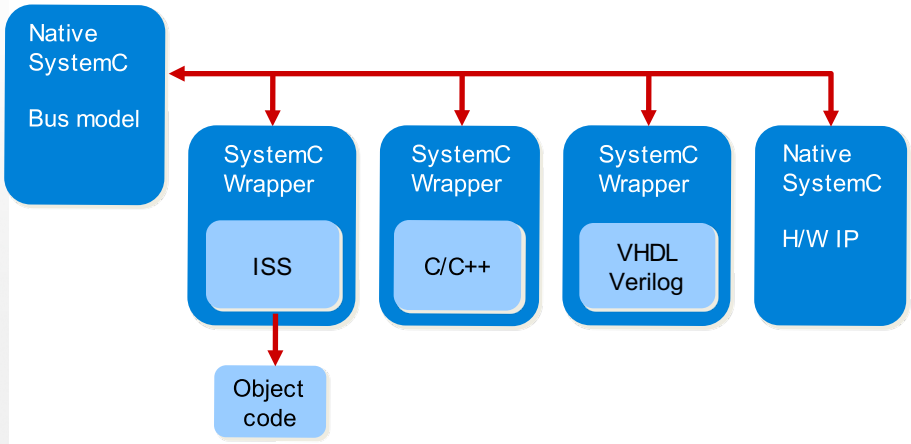
- Virtual Platform
  - Architectural exploration, performance modeling
  - Software development
  - Reference model for functional verification
  - Available before RTL - **early!**
  - Simulates much faster than RTL - **fast!**
- High-level synthesis
- Used by
  - Architects, software, hardware, and verification engineers

11

SystemC is Glue!

DOULOS

- Transaction-level modeling is communication-centric



12

Why SystemC in Particular?

- Industry standard IEEE 1666™
- Open-source C++ simulator
  - Available across multiple platforms (Linux, Windows, Mac)
  - No vendor lock-in, no licensing issues
- Easy integration with the C/C++ world
- Easy to put SystemC wrappers around existing HDL or C models
- Wide availability of tools, models and know-how
- Mature tool vendor support for co-simulation and debug

13

SystemC Data Types

In namespace sc_dt::

Template	Base class	Description
sc_int<W>	sc_int_base	Signed integer, W < 65
sc_uint<W>	sc_uint_base	Unsigned integer, W < 65
sc_bigint<W>	sc_signed	Arbitrary precision signed integer
sc_biguint<W>	sc_unsigned	Arbitrary precision unsigned integer
		(intermediate results unbounded)
sc_logic		4-valued logic: '0' '1' 'X' 'Z'
sc_bv<W>	sc_bv_base	Bool vector
sc_lv<W>	sc_lv_base	Logic vector
sc_fixed<>	sc_fix	Signed fixed point number
sc_ufixed<>	sc_ufix	Unsigned fixed point number

14

Data Type Characteristics

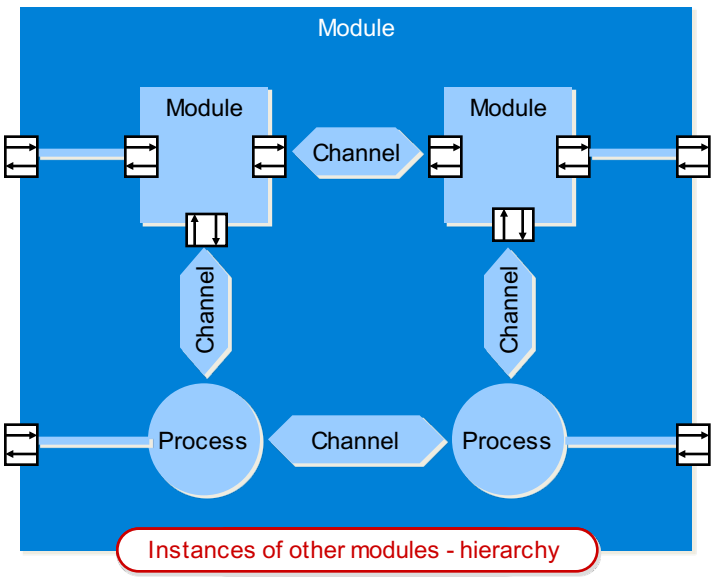
DOULOS

	C++	SystemC	SystemC	SystemC
Unsigned	**unsigned int**	**sc_bv, sc_lv**	**sc_uint**	**sc_biguint**
Signed	**int**		**sc_int**	**sc_bigint**
Precision	Host-dependent	Limited precision	Limited precision	Unlimited precision
Operators	C++ operators	No arithmetic operators	Full set of operators	Full set of operators
Speed	**Fastest**	Faster	Slower	**Slowest**

15

Modules

DOULOS



16

Example: Simple Multiplier

DOULOS

Class
Ports
Constructor
Process

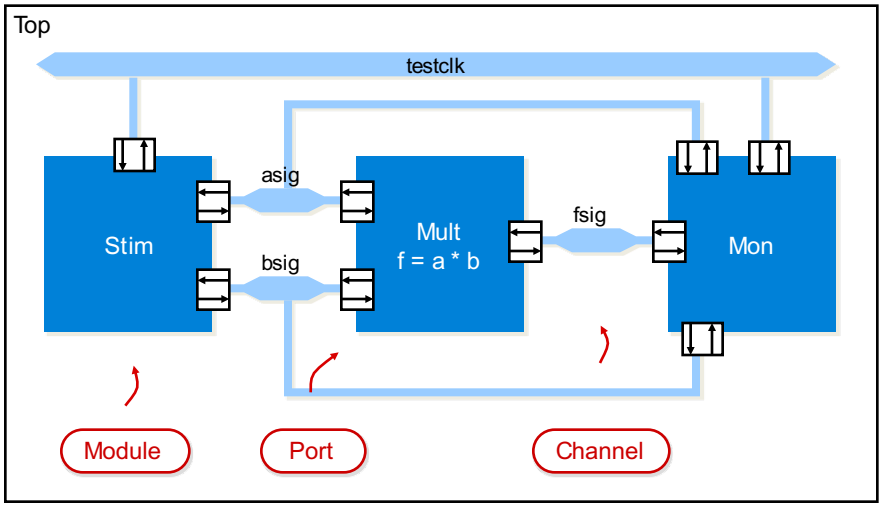
```
// mult.h

#include <systemc>

SC_MODULE(Mult)
{
  sc_in<int>  a;
  sc_in<int>  b;
  sc_out<int> f;

  SC_CTOR(Mult);

  void action();

};
```

```
// mult.cpp

#include "mult.h"
using namespace sc_core;

Mult::Mult
( sc_module_name nm )
: sc_module( nm )
{
  SC_HAS_PROCESS(Mult);
  SC_METHOD(action);
  sensitive << a << b;
}

void Mult::action(void)
{
  f = a * b;
}
```


17

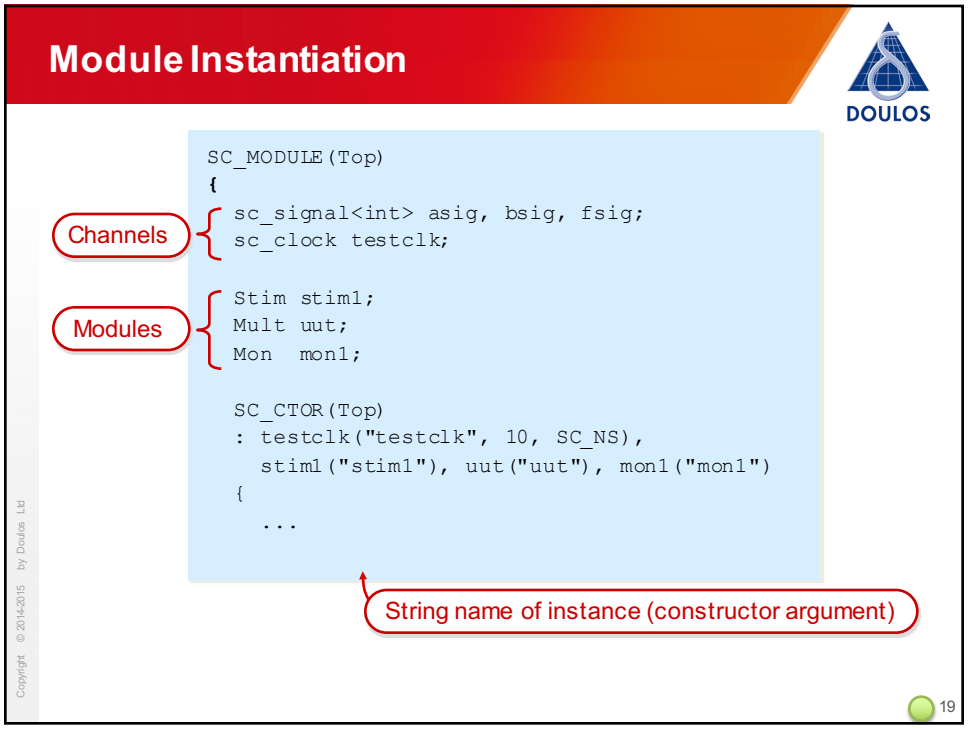
The Test Bench

DOULOS

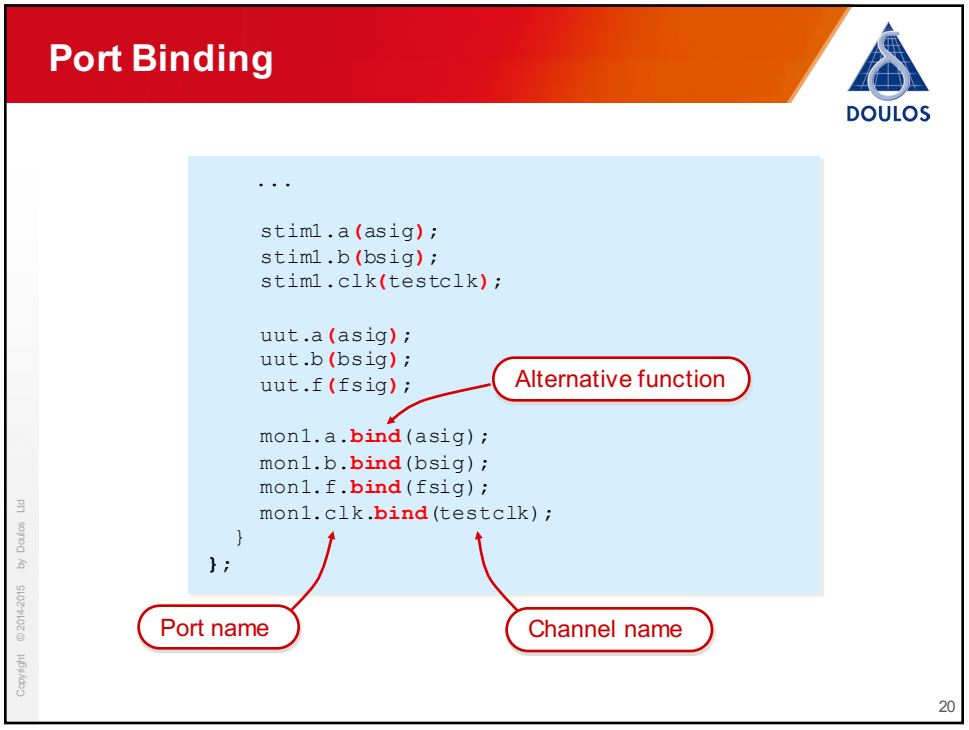


18

Module Instantiation



Port Binding



sc_main

DOULOS

- sc_main is the entry point to a SystemC application

```
#include "systemc.h"
#include "top.h"

int sc_main(int argc, char* argv[])
{

  Top top("top");

  sc_start();

  return 0;
}
```

Called from main()

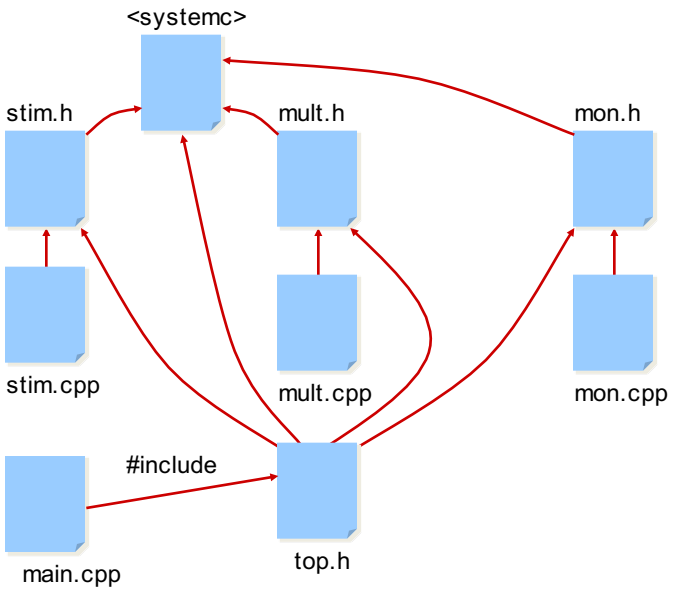
Instantiate one top-level module

End elaboration, run simulation

21

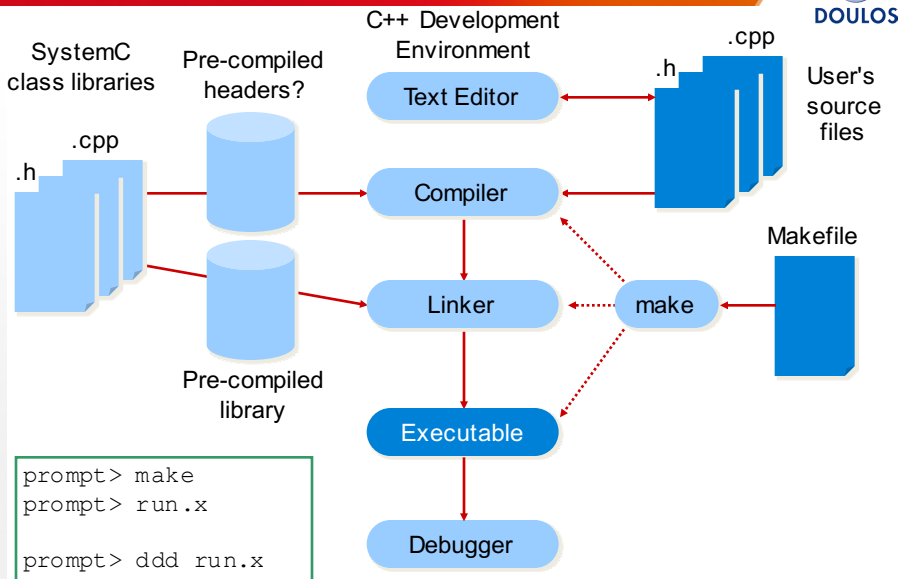
Summary of Files

DOULOS



22

Compilation and Simulation



23

Kinds of Process

- Processes
  - Must be within a module (not in a function)
  - A module may contain many processes
- Three different kinds of process
  - Methods SC_METHOD
  - Threads SC_THREAD
  - Clocked threads SC_CTHREAD (for synthesis)
- Processes can be *static* or *dynamic*

24

Dynamic Sensitivity

DOULOS

```
SC_CTOR(Module)
{
  SC_THREAD(thread);
    sensitive << a << b;
}

void thread()
{
  for (;;)
  {
    wait();
    ...
    wait(10, SC_NS);
    ...
    wait(e);
    ...
  }
}
```

Static sensitivity list

Wait for event on *a* or *b*

Wait for 10ns

Wait for event *e*

ignore *a* or *b*

25

sc_event and Synchronization

DOULOS

```
SC_MODULE(Test)
{
  int data;
  sc_event e;
  SC_CTOR(Test)
  {
    SC_THREAD(producer);
    SC_THREAD(consumer);
  }
  void producer()
  {
    wait(1, SC_NS);
    for (data = 0; data < 10; data++) {
      e.notify();
      wait(1, SC_NS);
    }
  }
  void consumer()
  {
    for (;;) {
      wait(e);
      cout << "Received " << data << endl;
    }
  }
};
```

Shared variable

Primitive synchronization object

Schedule event immediately

Resume when event occurs

26

sc_time

DOULOS

- Simulation time is a 64-bit unsigned integer
- Time resolution is programmable - must be power of 10 x fs
- Resolution can be set once only, before use and before simulation
- Default time resolution is 1 ps

```
enum sc_time_unit {SC_FS, SC_PS, SC_NS, SC_US, SC_MS, SC_SEC};
```

```
sc_time(double, sc_time_unit);
```

Constructor

```
void    sc_set_time_resolution(double, sc_time_unit);
sc_time sc_get_time_resolution();
```

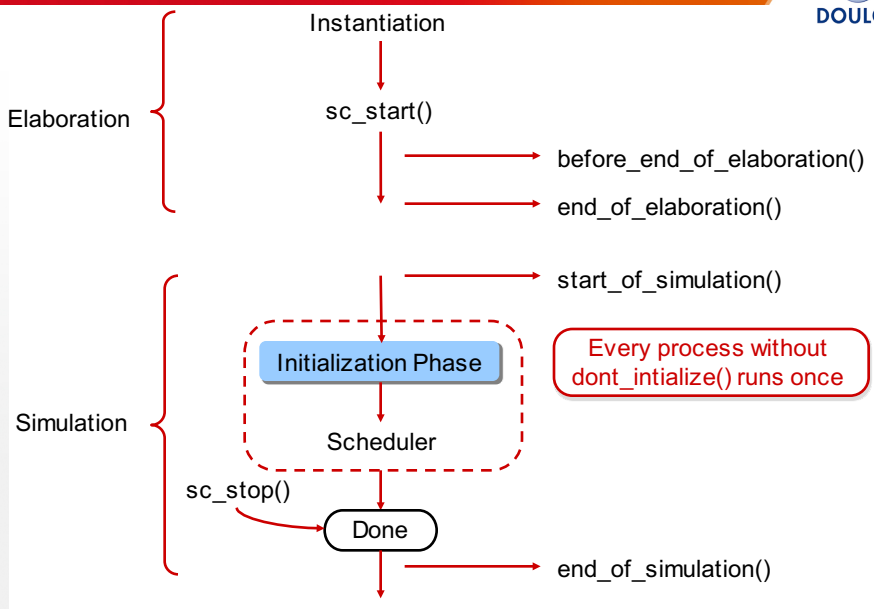
```
const sc_time& sc_time_stamp();
```

Get current simulation time

27

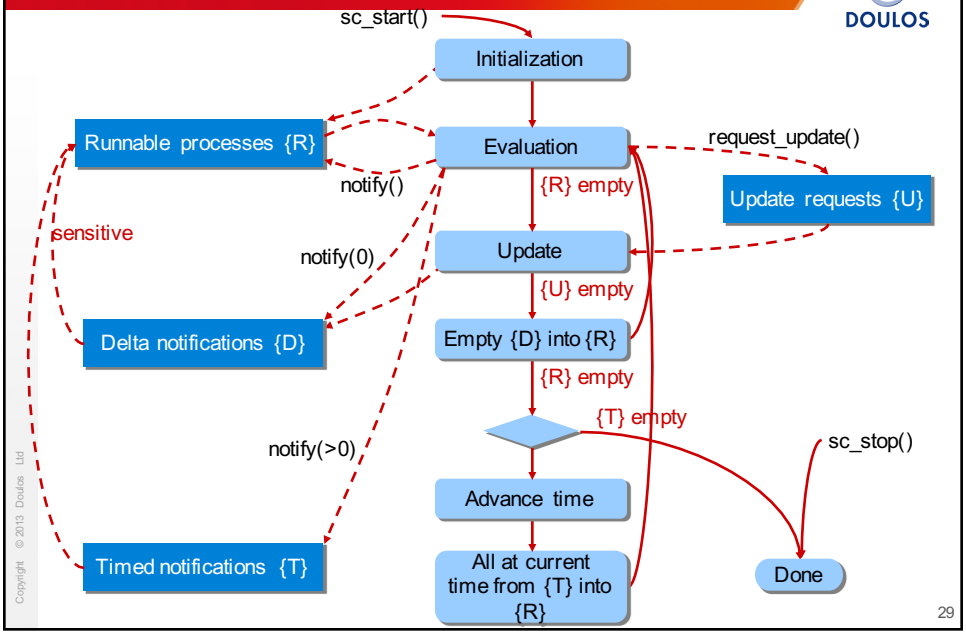
Elaboration / Simulation Callbacks

DOULOS



28

The Scheduler in Detail



29

Kinds of Channel

- Primitive channels
  - Implement one or more interfaces
  - Derived from sc_prim_channel
  - Have access to the update phase of the scheduler
  - Examples - sc_signal, sc_signal_resolved, sc_fifo
- Hierarchical channels
  - Implement one or more interfaces
  - Derived from sc_module
  - Can instantiate ports, processes and modules
- Minimal channels - implement one or more interfaces

30

Built-in Primitive Channels

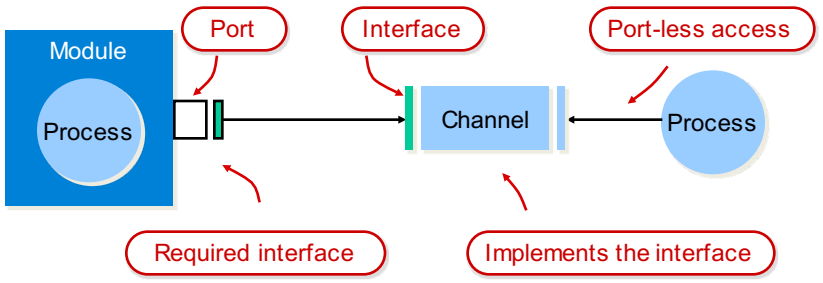
DOULOS

Channel	Interfaces	Events
sc_signal<T>	sc_signal_in_if<T> sc_signal_inout_if<T>	value_changed_event( )
sc_buffer<T>	Same as sc_signal	On every write()
sc_signal_resolved sc_signal_rv<W>	Same as sc_signal<sc_logic>	Same a sc_signal
sc_clock	Same as sc_signal<bool>	posedge & negedge
sc_fifo<T>	sc_fifo_in_if<T> sc_fifo_out_if<T>	data_written_event() data_read_event()
sc_mutex	sc_mutex_if	n/a
sc_semaphore	sc_semaphore_if	n/a
sc_event_queue	n/a	Every notify() invocation

31

Interface Method Call

DOULOS



An interface declares of a set of methods (pure virtual functions)

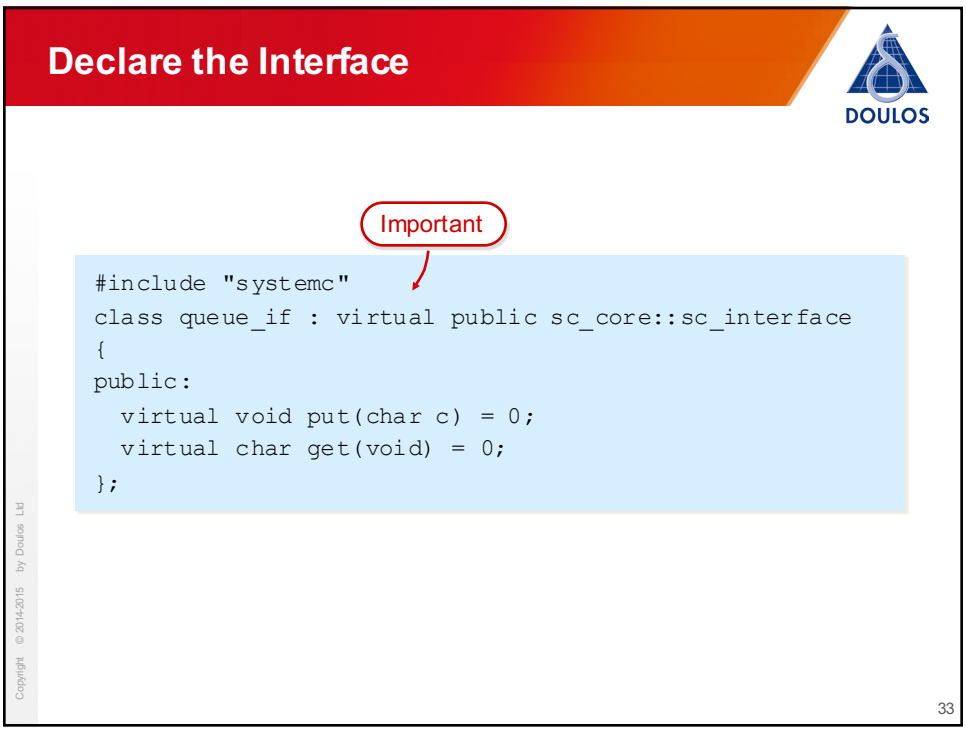
An interface is an abstract base class of the channel

A channel *implements* one or more interfaces (c.f. Java)

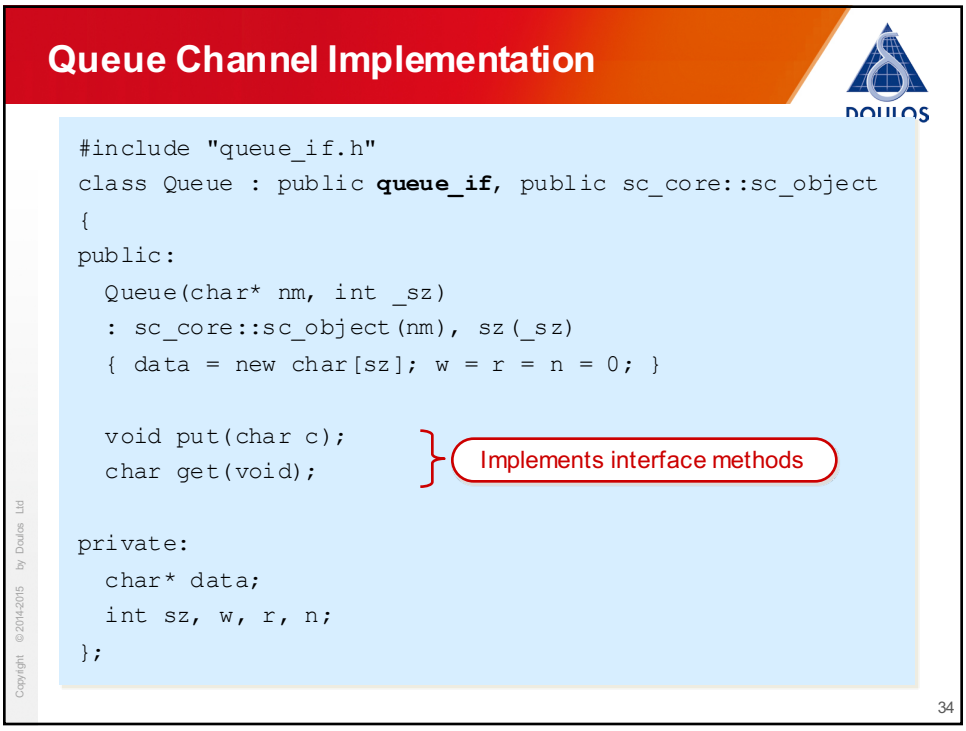
A module calls interface methods via a port

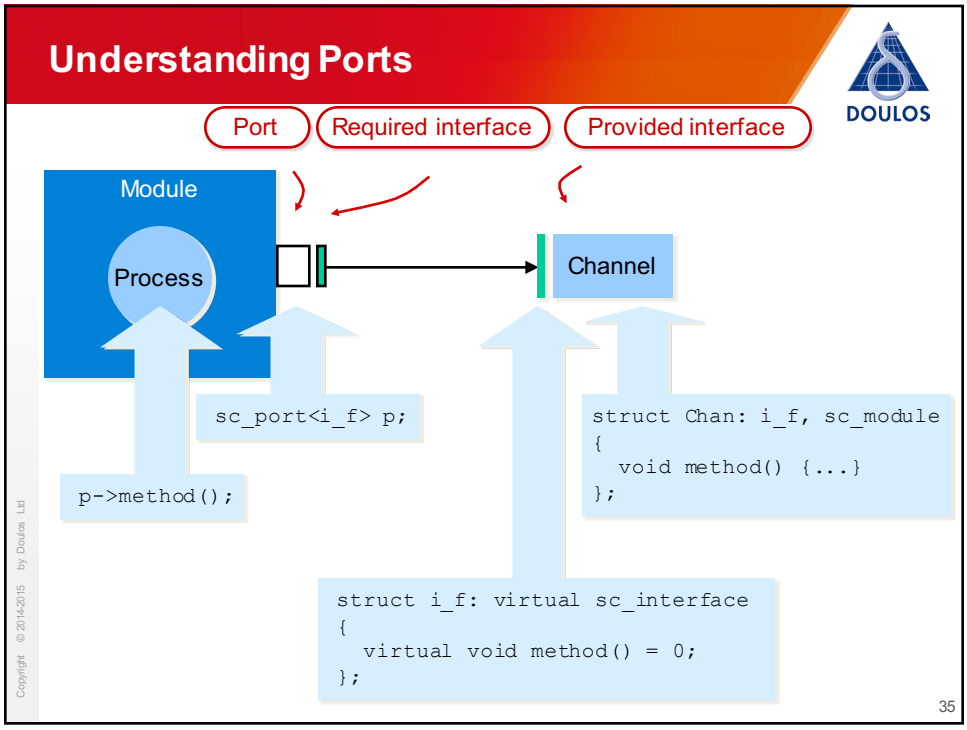
32

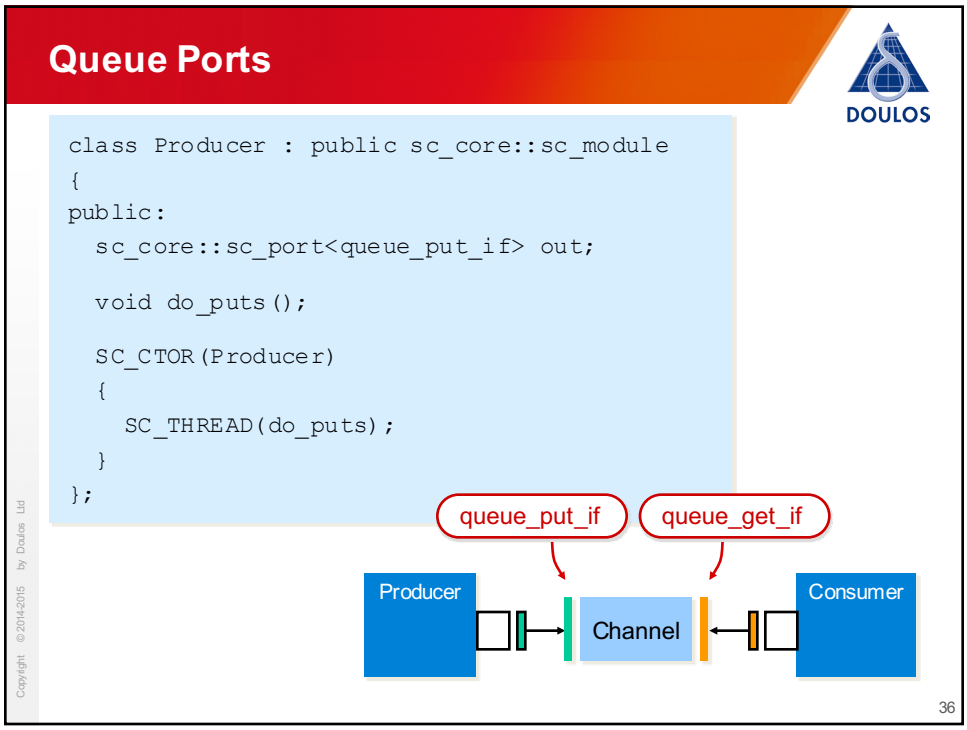
Declare the Interface



Queue Channel Implementation







Calling Methods via Ports

DOULOS

```
#include <systemc>
#include "producer.h"
using namespace sc_core;

void Producer::do_puts()
{
  std::string txt = "Hello World.";
  for (int i = 0; i < txt.size(); i++)
  {
    wait(SC_ZERO_TIME);

    out->put(txt[i]);
  }
}
```

Note: -> overloaded

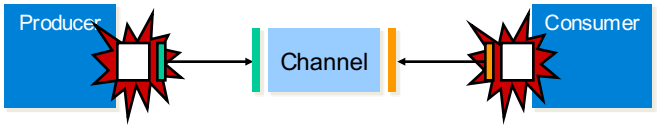
Interface Method Call

37

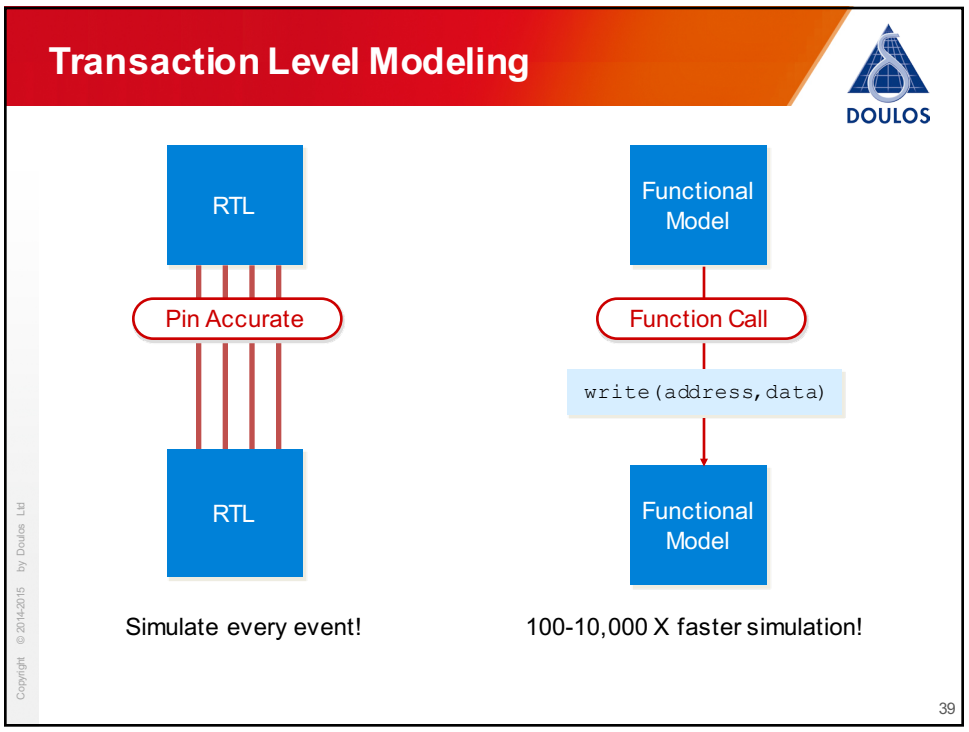
Why Ports?

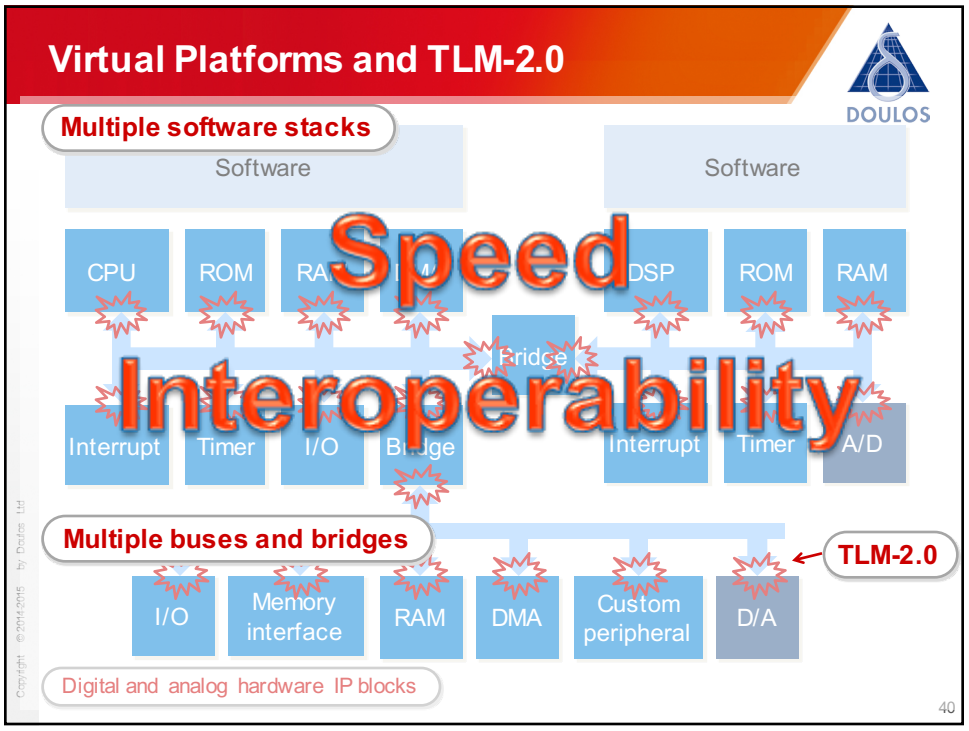
DOULOS

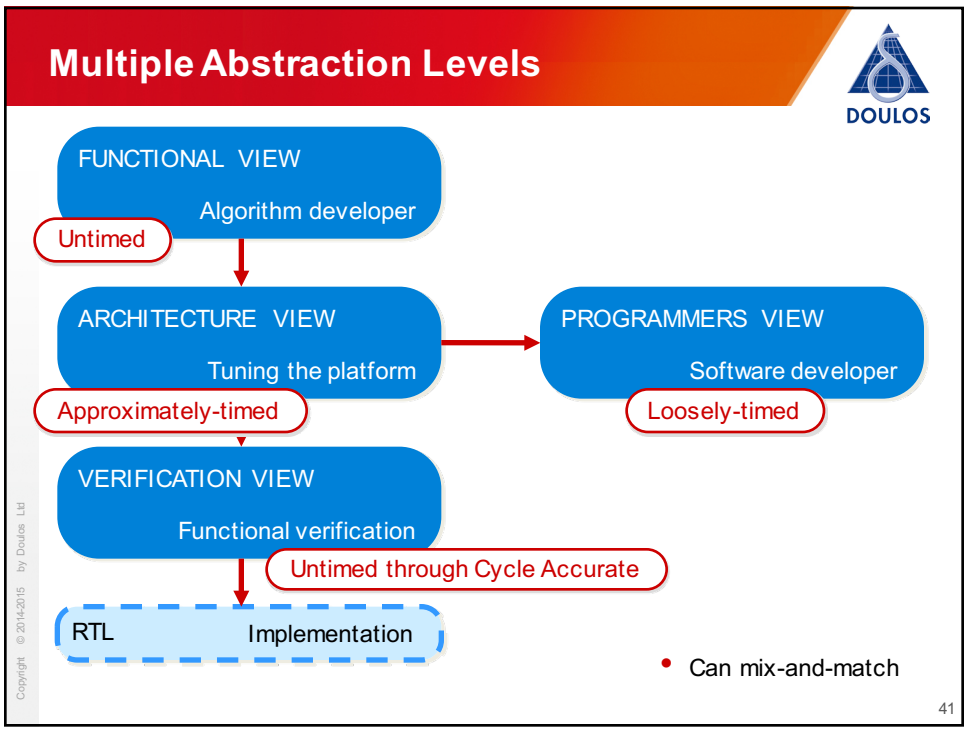
- Ports allow modules to be independent of their environment
- Ports support elaboration-time checks (register_port, end_of_elaboration)
- Ports can have data members and member functions

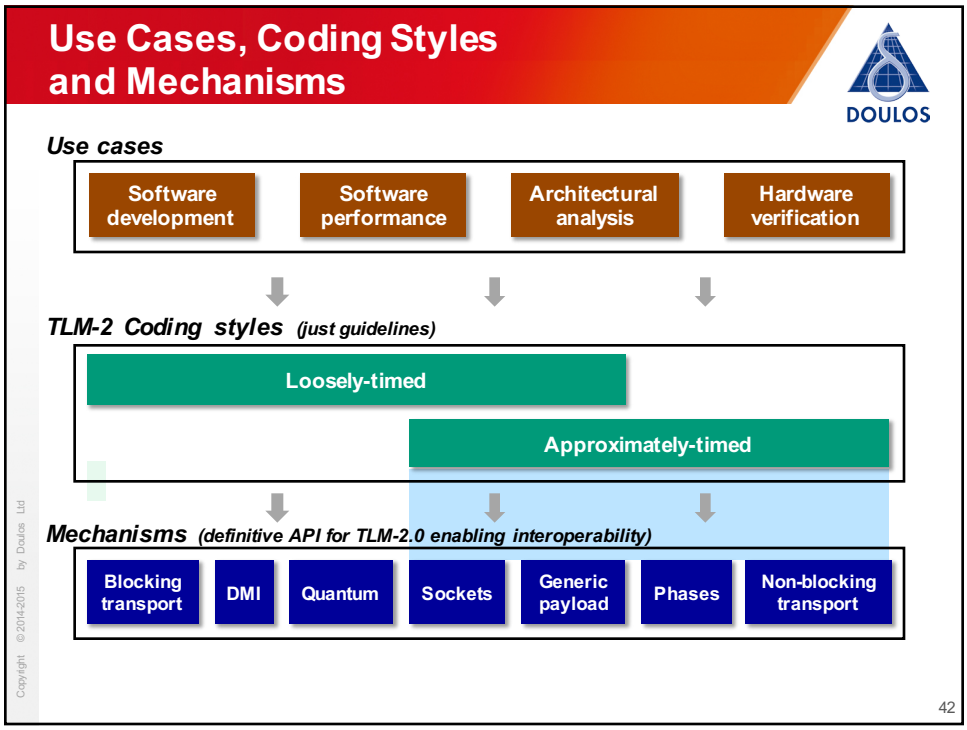


38


Transaction Level Modeling
DOULOS
RTL
Pin Accurate
RTL
Functional Model
Function Call
write(address,data)
Functional Model
Simulate every event!
100-10,000 X faster simulation!
39


Virtual Platforms and TLM-2.0
DOULOS
Multiple software stacks
Software
Software
CPU
ROM
DSP
ROM
RAM
Speed
Interoperability
Bridge
Interrupt
Timer
I/O
Interrupt
Timer
A/D
Multiple buses and bridges
TLM-2.0
I/O
Memory interface
RAM
DMA
Custom peripheral
D/A
Digital and analog hardware IP blocks
40





Coding Styles

DOULOS

Loosely-timed = as fast as possible

- Register-accurate
- Only sufficient timing detail to boot O/S and run multi-core systems
- b_transport – each transaction completes in one function call
- Temporal decoupling
- Direct memory interface (DMI)

Approximately-timed = just accurate enough for performance modeling

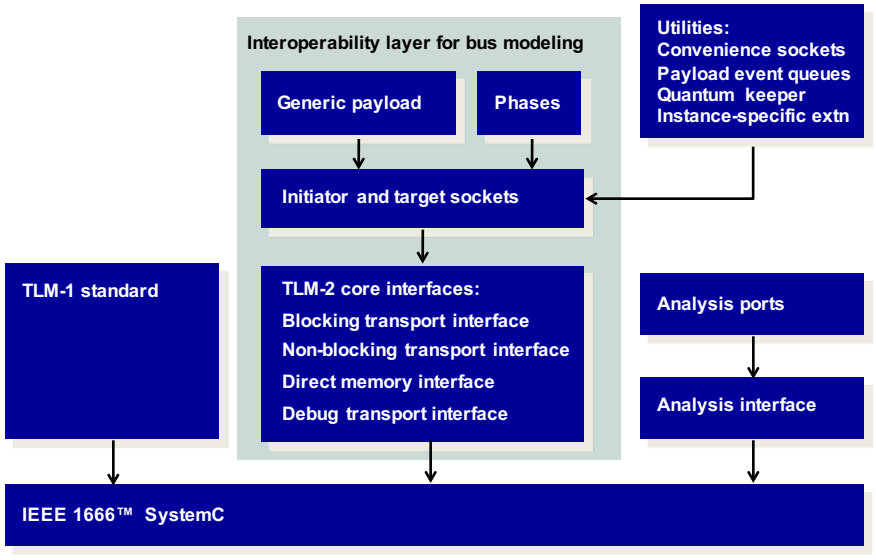
- *aka* cycle-approximate or cycle-count-accurate
- Sufficient for architectural exploration
- nb_transport – each transaction has 4 timing points (extensible)

Guidelines only – not definitive

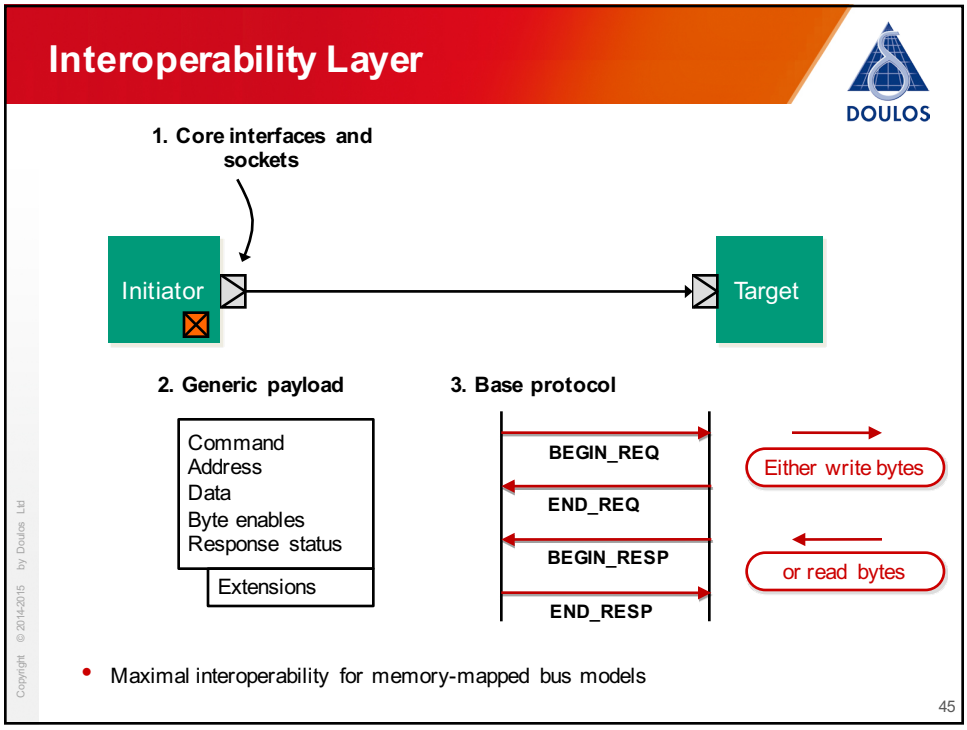
43

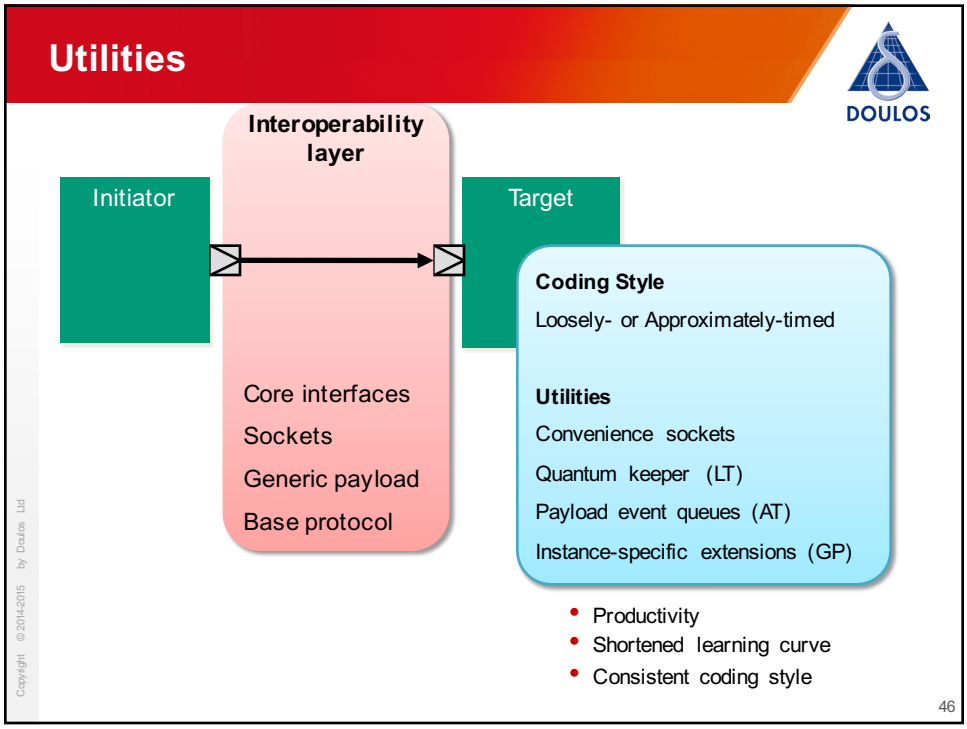
The TLM 2.0 Classes

DOULOS

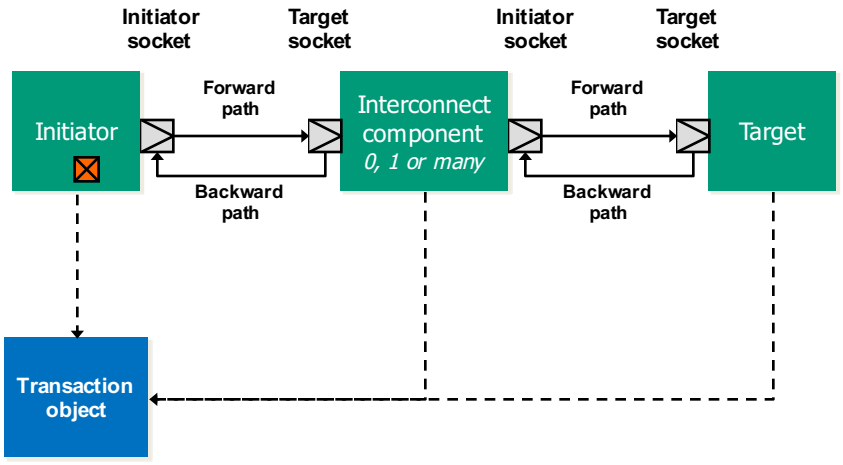


44





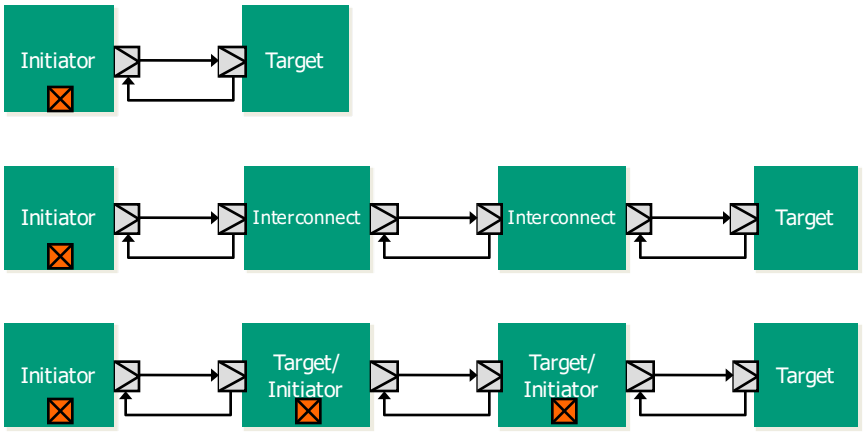
Initiators, Targets and Interconnect



- Single transaction object for request and response
- References to the object are passed along the forward and backward paths

47

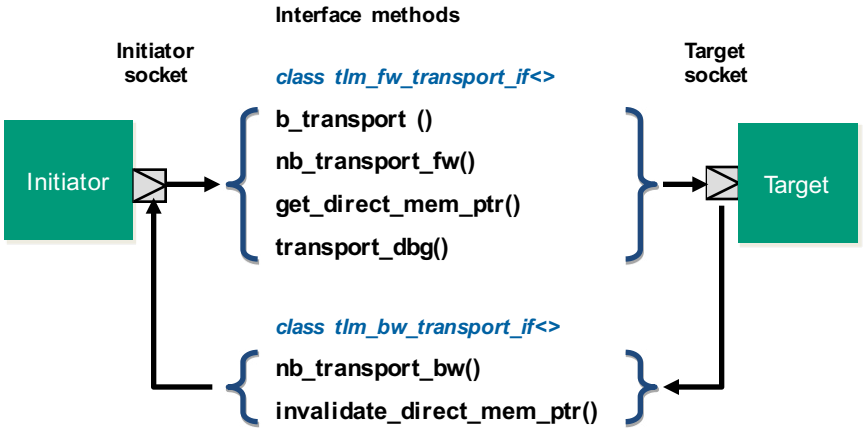
TLM-2 Connectivity



- Roles are dynamic; a component can choose whether to act as interconnect or target
- Transaction memory management needed

48

Initiator and Target Sockets



- Sockets provide fw and bw paths, and group interfaces

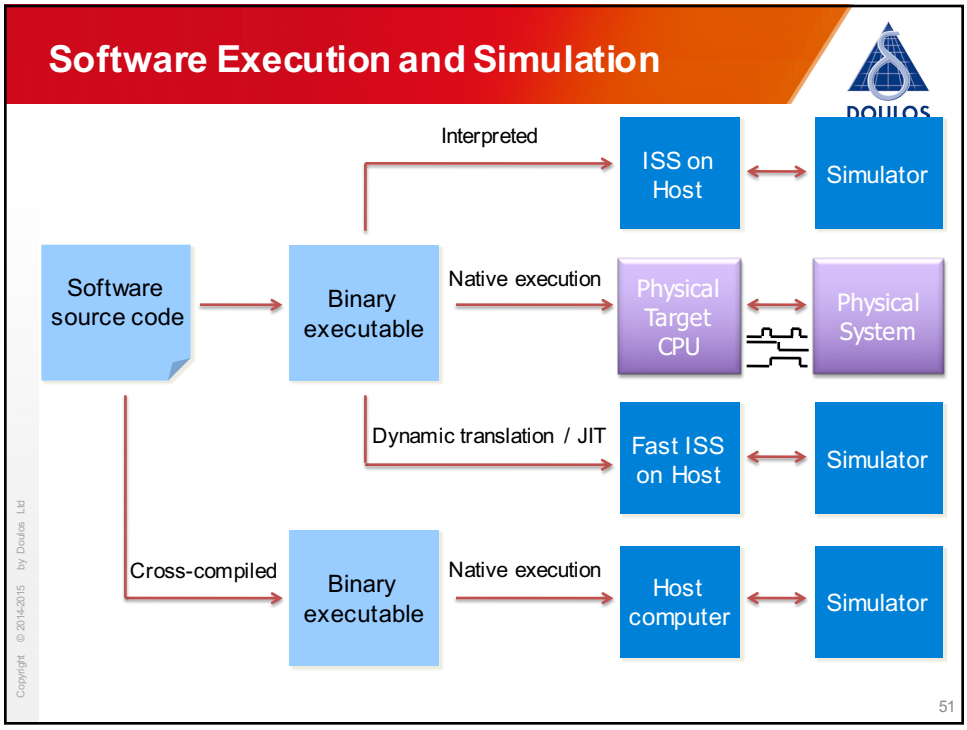
49

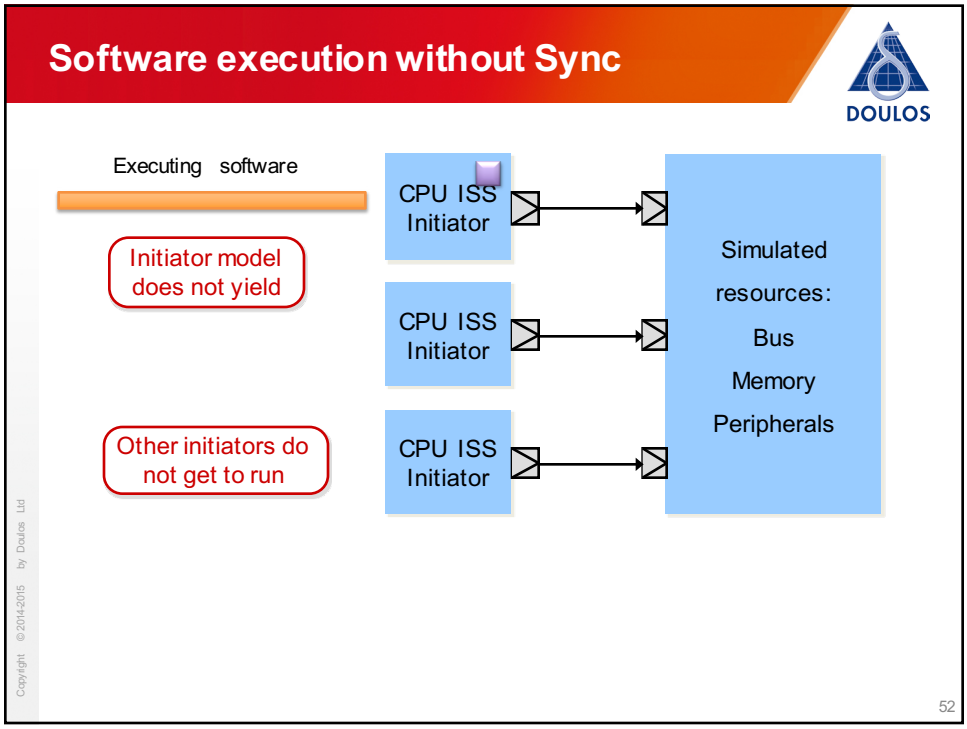
Socket Binding

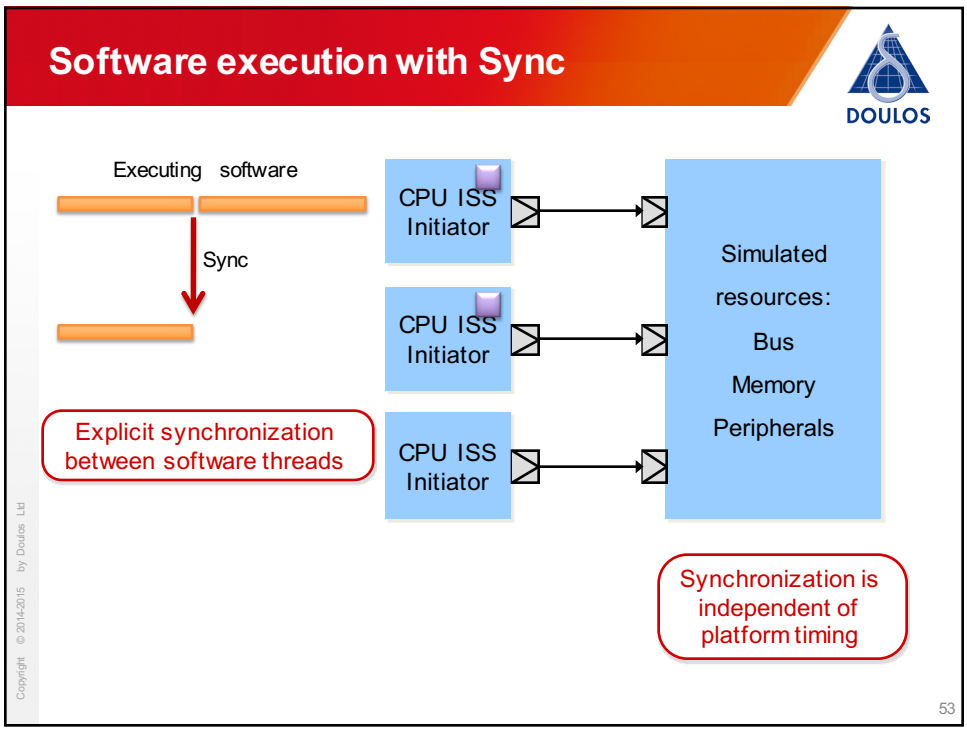
```
SC_MODULE(Top) {
  Initiator *init;
  Target    *targ;

  SC_CTOR(Top) {
    init = new Initiator("init");
    targ = new Target("targ");
    init->init_socket.bind( targ->targ_socket );
  }
  ...
```

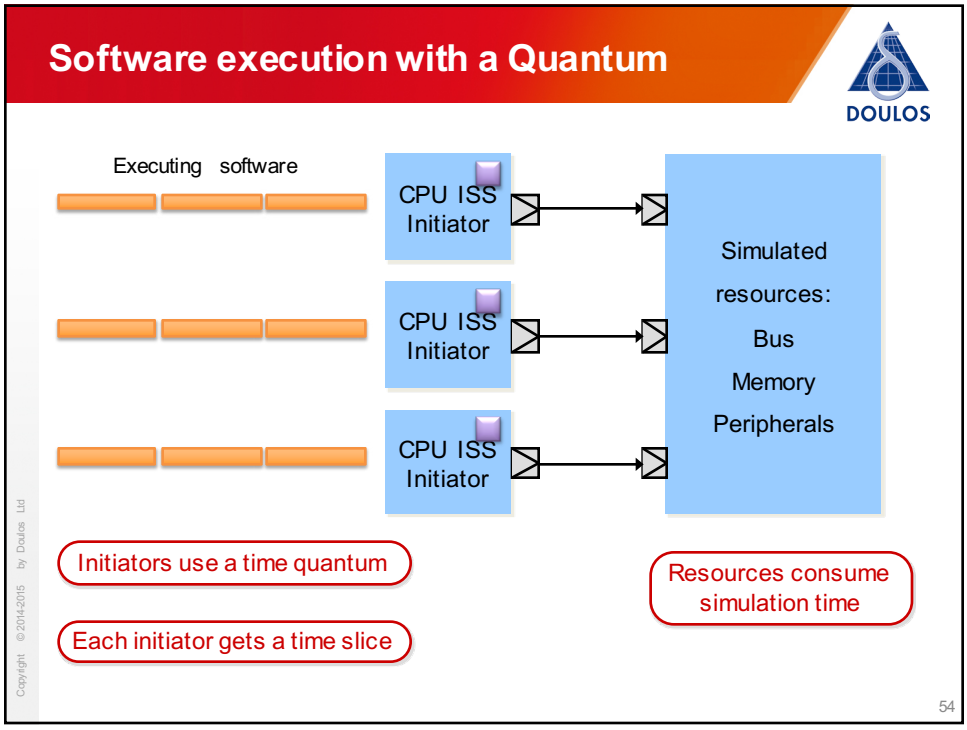
Bind initiator socket to target socket

50





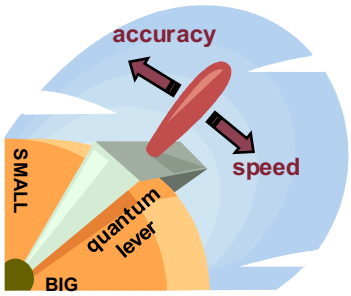




The Quantum

DOULOS

Quantum can be set by user

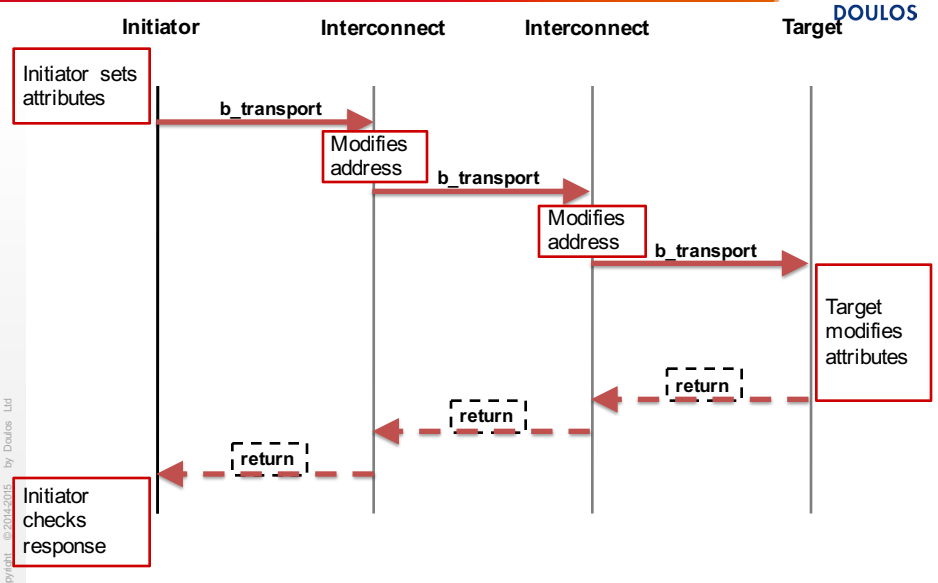


- Typically, all initiators use the same global quantum
- Individual initiators can be permitted to sync more frequently
- Not obliged to use the quantum at all if there are explicit synchronization points
- Quantum could be changed dynamically to "zoom in" (but no explicit support)

55

Causality with b_transport

DOULOS



56

Timing Annotation and b_transport

DOULOS

```
virtual void b_transport ( TRANS& trans , sc_core::sc_time& delay )
{
  Behave as if method were called at sc_time_stamp() + delay
  …
  delay = delay + latency;
}
```

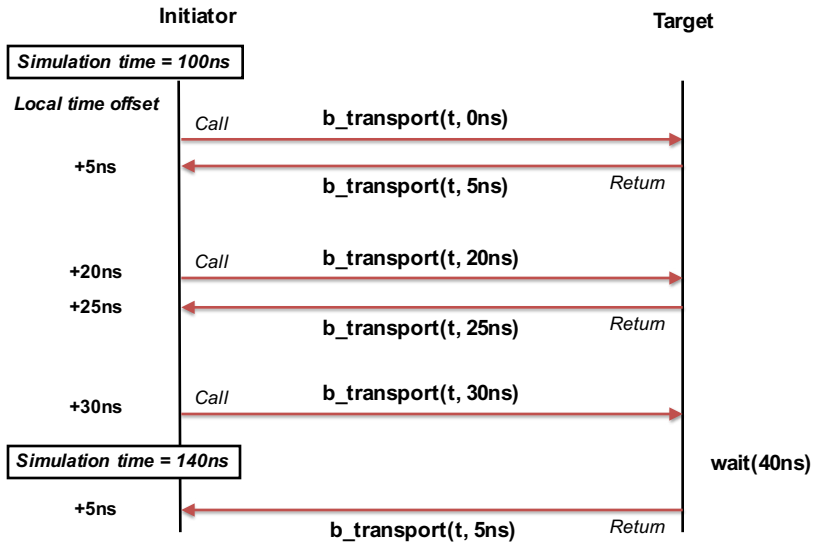
```
socket->b_transport( transaction, delay );

Behave as if method returned at sc_time_stamp() + delay
```

- Recipient may
  - Execute transactions immediately, out-of-order – Loosely-timed
  - Schedule transactions to execute at proper time – Approx-timed
  - Pass on the transaction with the timing annotation

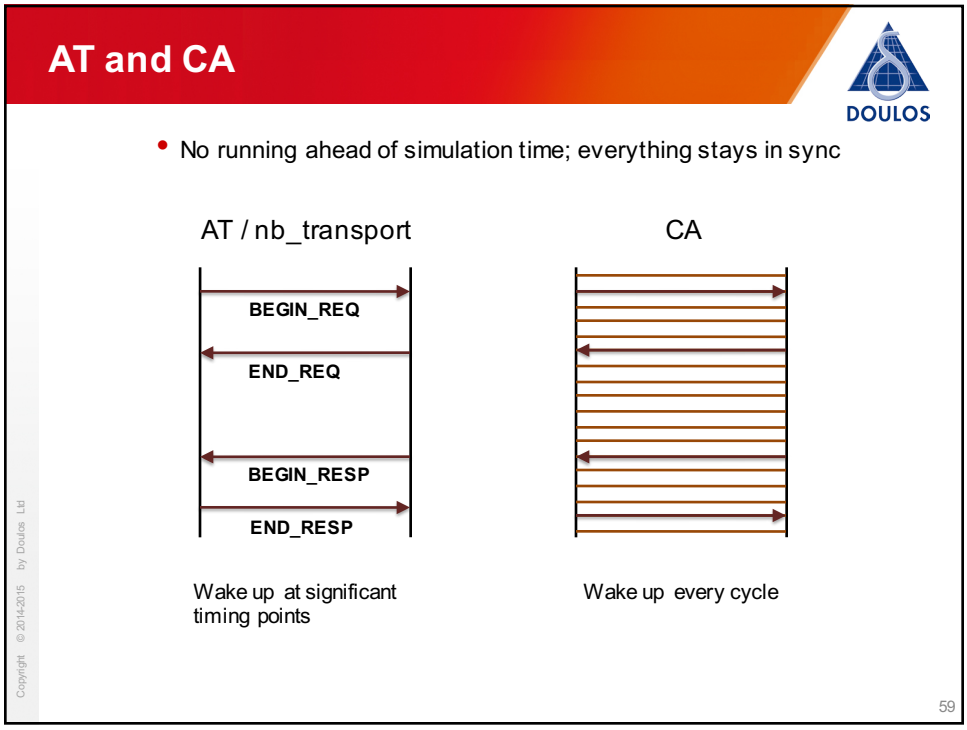
57

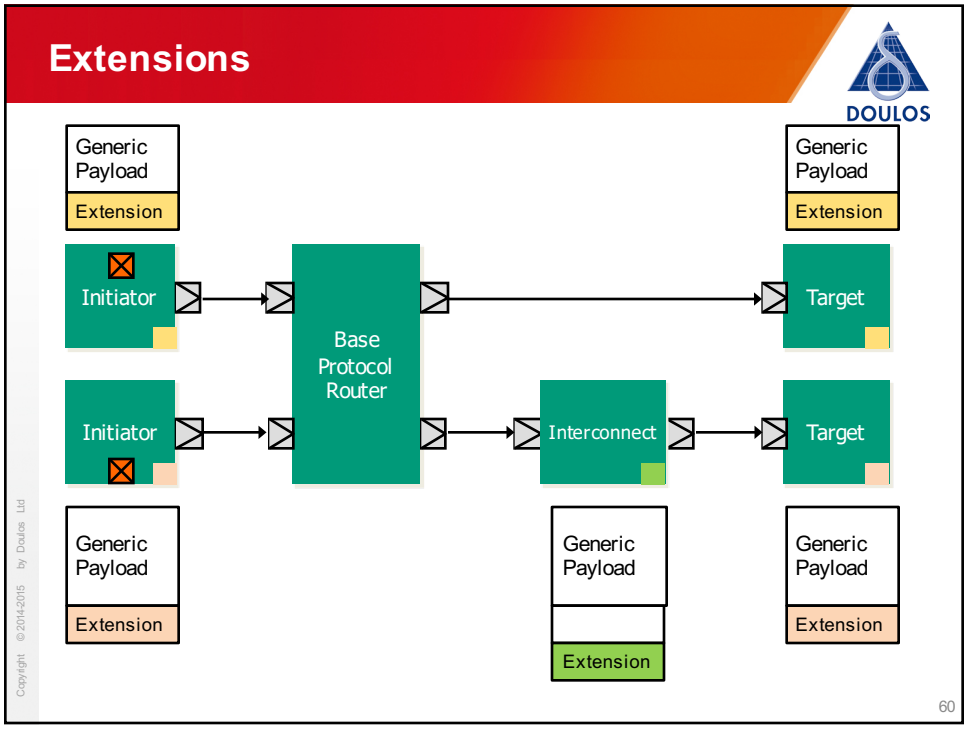
Temporal Decoupling

DOULOS



58





Accellera Systems Initiative

- Current Accellera Board Members
  - AMD
  - ARM
  - Cadence Design Systems Inc
  - Freescale Semiconductor
  - Intel Corporation
  - Mentor Graphics
  - NXP Semiconductors
  - Qualcomm Inc
  - Renesas Mobile
  - ST Microelectronics
  - Synopsys
  - Texas Instruments
- Website http://www.accellera.org
- European and North American SystemC Users Groups (and others)
  - http://www-ti.informatik.uni-tuebingen.de/~systemc
  - http://www.nascug.org

61

For More FREE Information

- IEEE 1666

standards.ieee.org/getieee/1666/download/1666-2011.pdf

- ASI SystemC 2.3.1

www.accellera.org

- On-line tutorials

www.doulos.com/knowhow/systemc

62