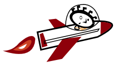
Plan

- Ch1 – Overview of System Design Using SystemC



- Ch2 – Overview of SystemC
- Ch3 – Data Types
- Ch4 – Modules
- Ch5 – Notion of Time
- Ch6 – Concurrency
- Ch7 – Predefined Channels
- Ch8 – Structure
- Ch9 – Communication
- Ch10 – Custom Channels and Data
- Ch11 – Transaction Level Modeling

Overview of SystemC

- Introduction
- SystemC Language Architecture
- Models of Computation
- TLM Based Methodology

History of SystemC

- SystemC is the confluence of four streams of ideas
  - Work at Synopsys with University of California, Irvive
  - Infineon (formerly Siemens HL), Frontier Design (IMEC)
  - Work within Open SystemC Initiative (OSCI)
  - Accellera Systems Initiative Language Working Group (LWG) since 2012 and SystemC 2.3
- Version 1.0 : Hardware design flow
  - RTL and behavioral level modeling
- Version 1.1
  - Timed functional modeling (e.g. for busses)
- Version 2.0.1 : System Design Flow
- Version 2.1 (October 2004) : Improve Software part
  - TLM modeling
  - Dynamic Threads (Software) …
  - New released (October 2005)
- Version 2.2 (March 2007)
- Version 2.3 (March 2012), including TLM

What is SystemC ?

- Add-on to C++ in order to express Hardware device
  - Concurrency
  - Communication mechanisms
  - Reactivity
  - Concept of time
- SystemC is not a language but rather a class library
- SystemC is not a panacea that will solve every design productivity issue
- SystemC is coupled with the SystemC Verification Library (SCV) and TLM for communication
- SystemC provides a common language for Sw and Hw

C++ language for Hardware (1/2)

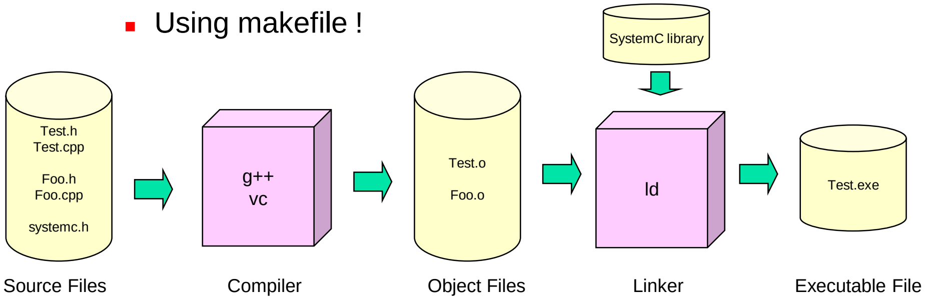
- Time Model
  - Time resolution with 64 bits (sc_time class)
  - Using enumerated type (SC_SEC to SC_FSEC)
  - Managed by kernel (like VHDL or Verilog language)
- Hardware Data Types
  - Wide variety of data types
  - Using explicit bit widths data types (sc_int<>, sc_fixed<>)
  - using four-state logic types (0,1,Z,X) named sc_logic
- Hierarchy and Structure
  - Using the principle of VHDL/Verilog language
  - Provide several constructs for implementing hardware hierarchy

C++ language for Hardware (2/2)

- Communication Management
  - Powerful mechanism for modeling communications
    - Basic Channel (FIFO, …)
    - Signals
    - Custom Channel
    - Hierarchical Channel
  - Providing several built-in channels common to Sw & Hw design (sc_mutex, sc_fifo, sc_signal, …)
  - Built-in interface classes to connect modules (entities for VHDL). For example : sc_mutex_if, sc_fifo_if …
- Concurrency
  - Simulation give an illusion of concurrency (like HDL languages)
  - Simulating each concurrent unit (SC_METHOD, SC_THREAD or SC_CTHREAD)
  - Modelsim tool (MENTOR) mixes HDL and SystemC languages

C++ Mechanics for SystemC

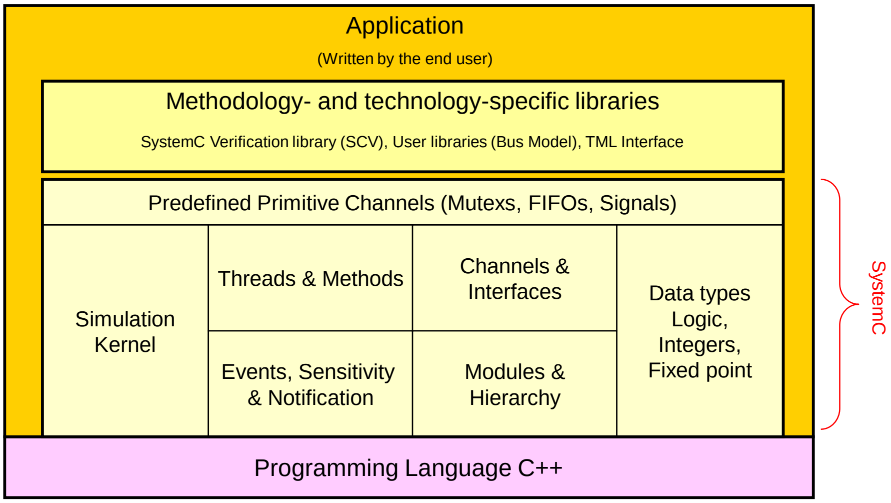
- C++ class library
- SystemC environment
  - SystemC-supported platform (Window, Linux …)
  - SystemC-supported C++ compiler (GNU, Visual . NET)
  - SystemC library (Compiled)
- Flow for compiling a SystemC Program
  - Using makefile !



Overview of SystemC

- Introduction
- SystemC Language Architecture
- Models of Computation
- TLM Based Methodology

SystemC Language Architecture



Application

(Written by the end user)

Methodology- and technology-specific libraries

SystemC Verification library (SCV), User libraries (Bus Model), TML Interface

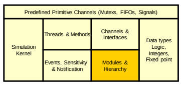
Predefined Primitive Channels (Mutexs, FIFOs, Signals)

Simulation Kernel	Threads & Methods	Channels & Interfaces	Data types Logic, Integers, Fixed point
	Events, Sensitivity & Notification	Modules & Hierarchy	

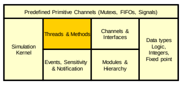
Programming Language C++

Components

- Modules and Hierarchy


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

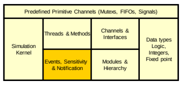
  - Correspond to a class (SC_MODULE)
  - Simulation processes are member functions of SC_MODULE class
  - Module is like an VHDL entity and architecture
- Threads and Methods


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

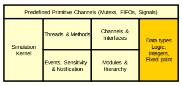
  - A methods or threads are a member function of a module (SC_MODULE)
  - Methods (SC_METHOD)
    - No argument, no return value, just a function …
    - Methods are invoked multiple times (Simulator kernel repeatedly calls the method)
  - Thread (SC_THREAD)
    - Simulator kernel invokes once
    - Can be suspended or resumed
    - Using Dynamic thread (v2.1)

Components – cont

- Events, Sensitivity, and Notification


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

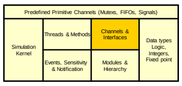
  - The methods and the threads are sensitive to
    - An event (sc_event)
    - Events (sc_event_queue) v2.1
  - An event triggers SC_METHOD or SC_THREAD
  - Events are fired through the “notify” function
  - The sensitivity list may be static or dynamic
  - Dynamic cases
    - For Method : next_trigger(arg) function
    - For Thread : wait(arg) function
- Data Types


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

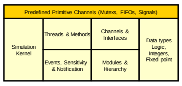
  - Mathematical calculations using sc_fixed<> and sc_int<> (DSP functions)
  - Familiar data type like sc_logic and sc_lv<> (std_logic and std_logic_vector in VHDL)

Components – cont

- Channels and Interfaces


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

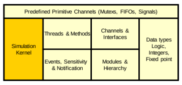
  - Couple entity/architecture in SystemC like VHDL (not Verilog !)
  - VHDL communications are signals/wire
  - SystemC uses either primitive channels or hierarchical channels
  - Connection of modules via ports
  - The implementation of a channel (sc_fifo) is an interface (sc_fifo_if)
- Predefined Primitive Channels


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

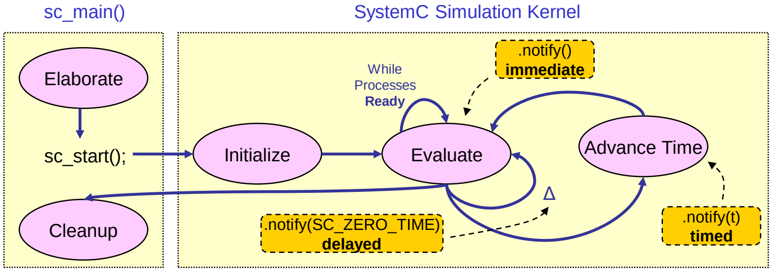
  - Mutex (sc_mutex), semaphore (sc_semaphore)
  - FIFO (sc_fifo)

Simulation

- SystemC Simulation Kernel


Predefined Primitive Channels (Mutexs, FIFOs, Signals)
Simulation Kernel
Threads & Methods
Channels & Interfaces
Data types Logic, Integers, Fixed point
Events, Sensitivity & Notification
Modules & Hierarchy

  - Elaboration
    - Execution of statements prior to sc_start() function
    - Initialization of data structure
    - Establishment of connectivity
    - Preparation of the next phase
  - Execution
    - Handing control to the SystemC simulation kernel



Phase Summary for SystemC

- Compilation
  - C++ compiler transforms C++ test into object code
- Linking
  - C++ linker builds executable out of objects and libraries
- Execution
  - Executable is started allocates system ressources
- Elaboration
  - SystemC kernel connects and initializes design portions
- Simulation
  - SystemC kernel works on event queue until no more events

Ecole polytechnique de l'université de Nice-Sophia Antipolis

Département Electronique

Overview of SystemC

- Introduction
- SystemC Language Architecture
- Models of Computation
- TLM Based Methodology

Introduction

- Fundamental to System Level Design
- Model of computation (MOC)
  - Model of time employed
    - real or integer values
    - untimed
  - Event ordering constraints within the system
    - Globally ordered
    - Partially ordered
  - Supported method(s) of communication between concurrent processes
  - The rule for process activation
- Example
  - VHDL : single fixed model of computation
  - No way to customize the given model

And SystemC ?

- Also single fixed MOC
  - Extremely general
  - Customized MOC
- Example of customization
  - Event (sc_event)
    - Notify(), Wait() : time is integer value (sc_event class)
    - Overloading these functions : time can be real value : sc_my_event class
  - Module (sc_module)
    - Correspond to a class sc_module (an entity in VHDL)
    - Overloading this class : RTL level module : sc_rtl_module
  - Channels, interfaces, ports

And SystemC ?

- Well know MOC
  - Static multirate dataflow
  - Dynamic multirate dataflow
  - Khan process networks
  - Discrete event as used for
    - RTL Hardware modeling
    - Network modeling (e.g. stochastic or “wait room” models)
    - Transaction-based SoC platform modeling
- SystemC can mixes Models of Computation

The RTL MOC

- Correspond to digital hardware synchronized by clock signals
  - Used by VHDL/VERILOG language
  - Supported by commercial hardware synthesis tools
- All communication between processes occurs through signals (sc_signal, sc_signal_rv, …)
- RTL Modules are Pin-Accurate and Cycle-Accurate
  - A port of an RTL module directly correspond to wires (real world)
  - SystemC signals closely mirror the behavior of VHDL signals
    - S <= A after 10 ns; -- VHDL
    - S = A; // SystemC
  - SystemC signals do not allow time delays to be specified when signal assignments are performed
    - But you can customized a signal …

Khan Process Networks (KPN)

- Effective MOC for building algorithmic models of signal-processing applications
  - Multimedia applications
  - Communications product domains
- Computing blocks (processes)
  - Concurrently execution
  - Connected by channels that carry sequences of data tokens
  - These channels are infinite length FIFO channels
    - Blocking read operations
    - Nonblocking write operations
- KPN systems are deterministic
- KPN have no concept of time (UnTimed Functional Model, UTFM)

- Practically …
  - addition time delays (Timed Functional Model, TFM)
  - FIFO are not infinite (blocking write operations are possible)

Static DataFlow (SDF)

- Special case of Khan process networks
  - Functionality within each process includes 3 stages
    - Reading of all input tokens
    - Execution of the computation within the process
    - Writing of all output tokens
  - The number of read/write tokens is executed, is fixed and knows at compile-time
- Tools can analyze the network and build static execution schedules for processes and compute bounds for all FIFOs at compile-time rather than execution-time
- SystemC : using DataFlow Modeling Style (Functional Modeling)

Transaction-Level Models

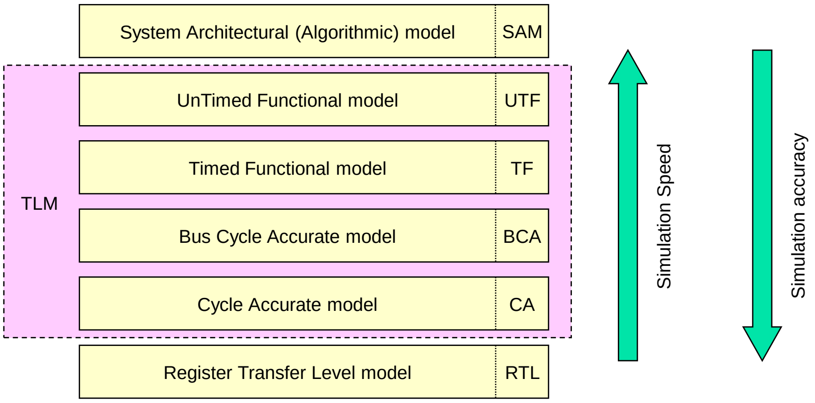
- Represents one specific type of the discrete-event MOC
- Communication models between modules using functions calls
  - Functions represent the transaction
  - Transactions can be viewed as having
    - Specific start time
    - Specific end time
    - Payload data

Ecole polytechnique
de l'université de Nice-Sophia Antipolis
Département Electronique

Overview of SystemC

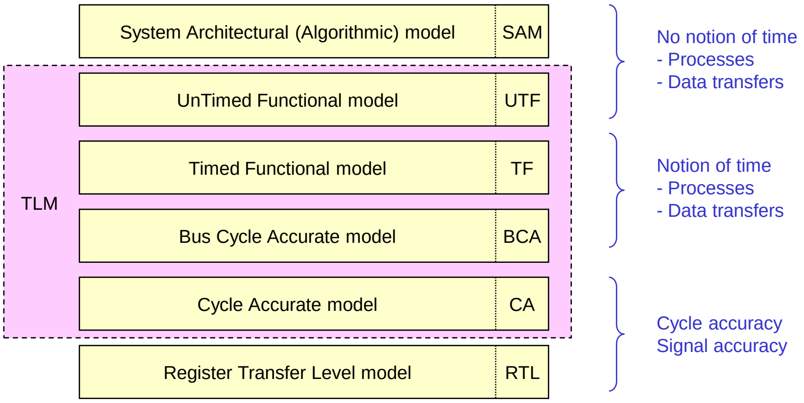
- Introduction
- SystemC Language Architecture
- Models of Computation
- TLM Based Methodology

Layer of Hardware Design



System Architectural (Algorithmic) model	SAM
UnTimed Functional model	UTF
Timed Functional model	TF
Bus Cycle Accurate model	BCA
Cycle Accurate model	CA
Register Transfer Level model	RTL

Scope of Layer



System Architectural (Algorithmic) model | SAM

UnTimed Functional model | UTF

Timed Functional model | TF

Bus Cycle Accurate model | BCA

Cycle Accurate model | CA

Register Transfer Level model | RTL

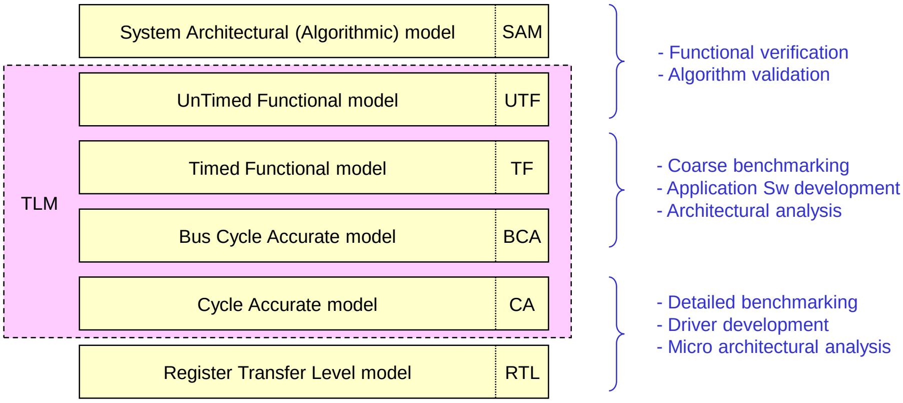
TLM

No notion of time
- Processes
- Data transfers

Notion of time
- Processes
- Data transfers

Cycle accuracy
Signal accuracy

Purpose of Layer



System Architectural (Algorithmic) model | SAM

UnTimed Functional model | UTF

Timed Functional model | TF

Bus Cycle Accurate model | BCA

Cycle Accurate model | CA

Register Transfer Level model | RTL

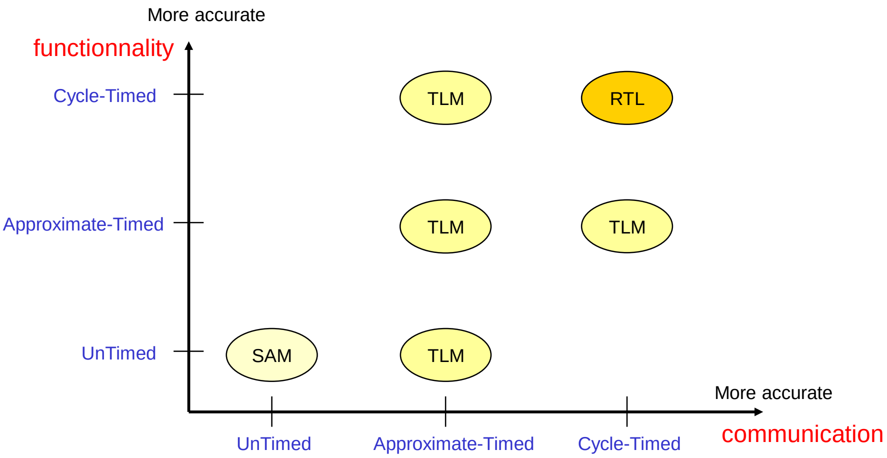
TLM

- Functional verification
- Algorithm validation

- Coarse benchmarking
- Application Sw development
- Architectural analysis

- Detailed benchmarking
- Driver development
- Micro architectural analysis

Abstraction Terminology



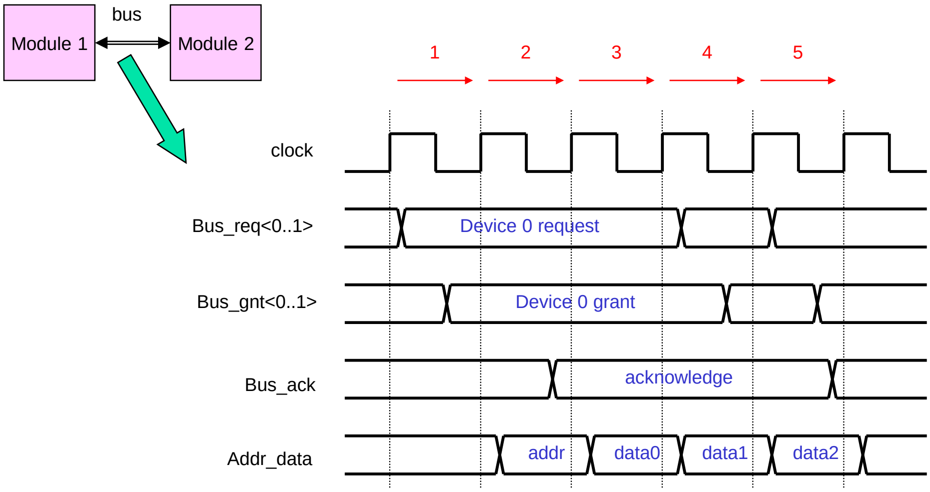
Timing of TLM

Model	Communication	Functionality
SAM	UT	UT
Component assembly	AT	UT
Bus arbitration	AT	AT
Bus functional	CT	AT
Cycle-accurate computation	AT	CT
RTL	CT	CT

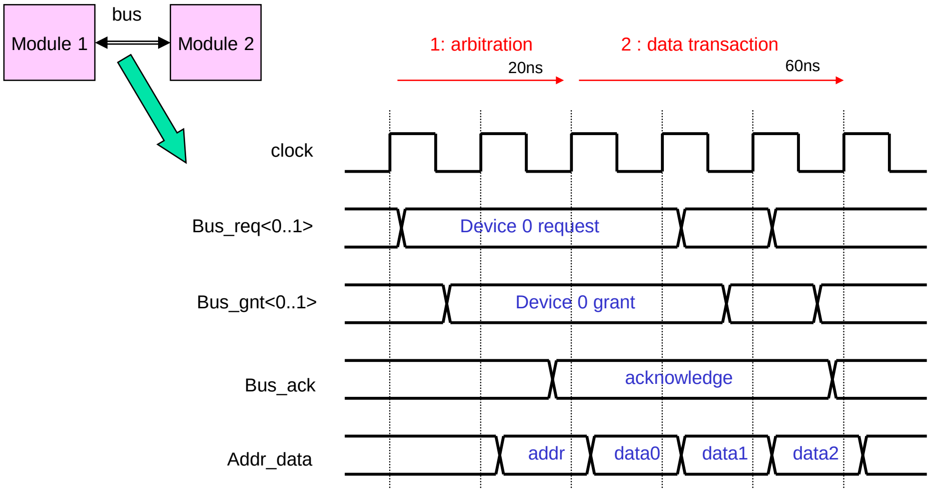
TLM

UT : UnTimed AT : Approximate-Time CT : Cycle-Time

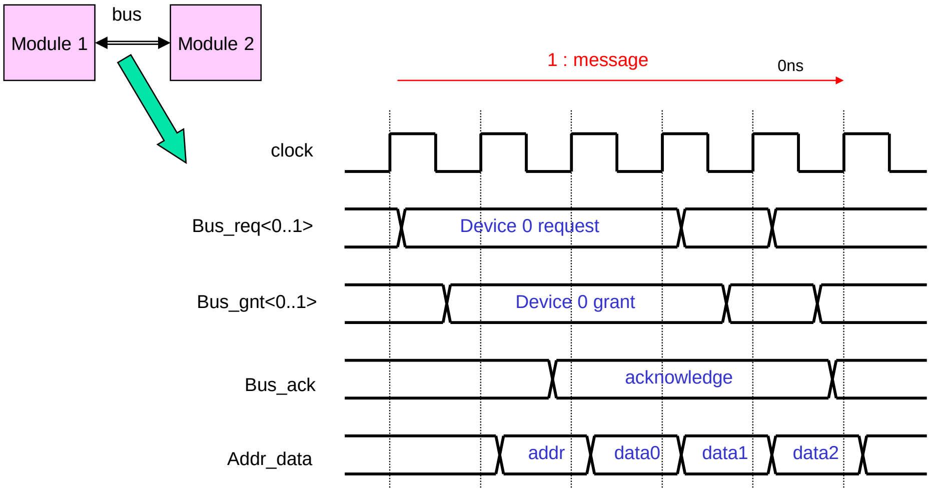
Example (1/3)
Cycle-Time



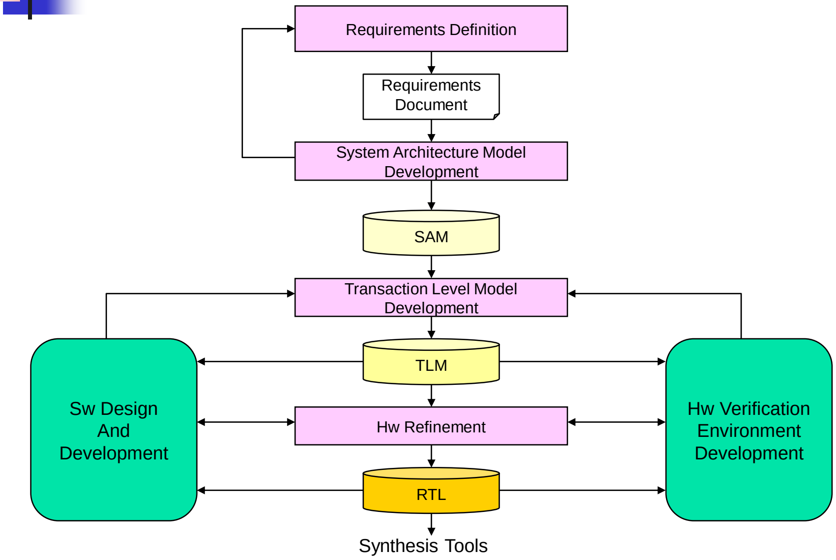
Example (2/3)
Approximate-Time



Example (3/3)
UnTimed



TLM Based Methodology
TLM Based Flow


Requirements Definition
Requirements Document
System Architecture Model Development
SAM
Transaction Level Model Development
TLM
Hw Refinement
RTL
Synthesis Tools
Sw Design And Development
Hw Verification Environment Development

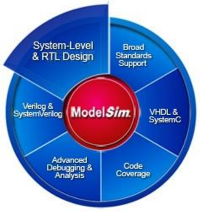
TLM Based Methodology Goals

- Goal 1
  - Refinement of implementation features such as Hw/Sw partitioning
  - Hw partitioning among ASICs, FPGAs and boards
  - Bus architecture exploration
  - Co-processor definition or selection
- Goal 2
  - Development platform for system software
- Goal 3
  - “Golden Model” for the hardware functional verification
- Goal 4
  - Hardware micro-architecture exploration
  - Basis for developing detailed hardware specification
- May be another goals ?

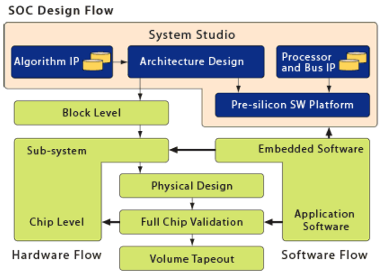
- SystemC is a good candidate for TLM Based Methodology !

EDA Tools

- MENTOR : ModelSim
  - Hardware/software verification
  - Performance Modeling
  - Design at Higher-Levels of Abstraction

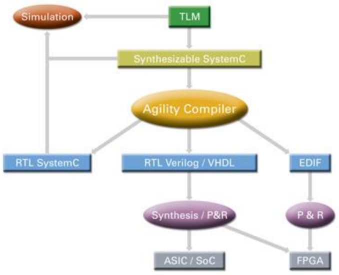


- SYNOPSYS : System Studio
  - Extensive support for SystemC™, the emerging architecture-level modeling language.
  - Enables complete end-to-end system simulation of the SoC in realistic virtual environments.



EDA Tools

- CELOXICA : Agility Compiler
  - Synthesize high-level models directly to FPGA for verification or RTL for ASIC flows using the industry's most mature C-synthesis technology
- Submit Design
- Cofluent Studio
- And so on …


Simulation
TLM
Synthesizable SystemC
Agility Compiler
RTL SystemC
RTL Verilog / VHDL
EDIF
Synthesis / P&R
P & R
ASIC / SoC
FPGA